

A privacy-preserving ticketing system

*Kristof Verslype Pieter Verhaeghe Jorn Lapon
Girma Nigusse Vincent Naessens
Bart De Decker*

Report CW 523, October 2008



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A privacy-preserving ticketing system

*Kristof Verslype Pieter Verhaeghe Jorn Lapon
Girma Nigusse Vincent Naessens
Bart De Decker*

Report CW523, October 2008

Department of Computer Science, K.U.Leuven

Abstract

Electronic identity (eID) cards are deployed in an increasing number of countries. These cards often provide digital authentication and digital signature capabilities, but have at the same time serious privacy shortcomings. We can expect that ordering and issuing tickets for events (e.g. soccer matches) will be increasingly done using eID cards, hence, severely threatening the user's privacy. This paper proposes two alternative ticketing systems that are using the eID card in a bootstrap procedure, but still are providing a high degree of privacy to the user.

Keywords : Ticket, Anonymity, Privacy, Security, Electronic Identity
MSC : Primary : D.4.6, Secondary : K.4.1.

Contents

1	Introduction	3
2	Requirements	4
3	Technologies	4
3.1	Pseudonym Certificates	4
3.2	Anonymous Credentials	5
3.3	Commitments	6
3.4	Provable One-Way Functions	6
4	Assumptions and Notation	6
4.1	Assumptions	6
4.2	Notation	7
5	Trivial eID-based Solution	7
5.1	Introduction	7
5.2	Roles	7
5.3	High Level Description	8
5.4	Protocols	8
5.5	Evaluation	8
6	Solution based on Pseudonym Certificates and Fixed Timeslots	9
6.1	Introduction	9
6.2	Roles	9
6.3	Assumptions	9
6.4	High Level Description and data structures.	10
6.5	Protocols	10
6.6	Evaluation	12
7	Solution based on Pseudonym Certificates and User defined timeslots	14
7.1	Roles	14
7.2	Assumptions	14
7.3	High Level Description	15
7.4	Scenarios and Data Structures	15
7.5	Protocols	18
7.6	Evaluation	20
8	Further improving the privacy	22
8.1	Minimizing personal data disclosure to T and reducing information	22
8.2	Minimizing personal data disclosure to T and PS	23
8.3	Pseudonym certificates with Commitments	24

9	Solution Based on Anonymous Credentials	24
9.1	Introduction	24
9.2	Roles	24
9.3	Assumptions	25
9.4	High Level Description	25
9.5	Protocols	25
9.6	Evaluation	26
10	Comparison	28
11	Related Work	29
12	Conclusions and Future Work	32

1 Introduction

Tickets are used for an innumerable number of events: soccer matches, music festivals, exhibitions, etc. These tickets are ever more bought electronically. An increasing number of countries issue electronic identity cards to their citizens. Examples are Belgium, Estonia and Austria. These eID cards usually allow the holder to authenticate and to digitally sign documents, but often, they are very privacy unfriendly. For example, authentication using the Belgian eID card will usually lead to the divulgement of important personal data such as your national registration number (NRN). Despite these privacy dangers, the use of the eID card is promoted by the governments. We can thus expect that in the near future, electronic ticketing systems will arise based on the eID card. A trivial solution is easy to devise. However, this solution is not acceptable because it further endangers the card holder's privacy as profiles can easily be compiled, linked to each other and to the identity of the card holder. An advantage of the use of eID cards is that it is straightforward to impose restrictions on the maximum number of tickets that can be bought by one user, hence, thwarting sales on black markets. Sometimes, special offers are available for buyers under or over a certain age or living in the region where the event is organized. Here too, eID cards can help in securely conveying (proving) that these conditions are satisfied for the buyer. However, the use of these cards will usually disclose more information than is required.

For big events with thousands of attendants, the police would be helped if tickets were not anonymous, but could be linked to the identity of the attendants, or at least to the identity of the buyers of these tickets. Especially, when rows or riots occur, it would make it easier to identify and prosecute the instigators. However, the use of tickets attributable to individuals poses severe privacy risks and brings us closer to a "Big Brother" state.

This report proposes three solutions where the eID card is needed to obtain an anonymized permit, allowing a user to obtain tickets in a privacy friendly way. The role of the eID card is thus reduced to a bootstrapping role. The first two solutions are based on pseudonym certificates, i.e. X.509 certificates containing a user's nym instead of a real identity. A third solution is based on the more enhanced anonymous credential systems, which allow to anonymously disclose only a subset of the personal attributes (or properties thereof) embedded in the credential. The three solutions are validated and compared with the trivial solution and with each other.

The main requirements are given in section 2. Section 3 introduces the required technologies. Section 4 explains notations and specifies the assumptions. Section 5 discusses the trivial protocol, and is followed by a discussion about the three privacy friendly solutions in sections 6, 7 and 9. In section 10, a comparison is given. Sections 11 and 12 examine the related work, draw the conclusions and describe future work.

2 Requirements

The requirements are now summed up. F4 and F5 are optional.

Functional/Security Requirements

- F1** Every event may have a policy that limits the number of tickets obtainable by one buyer. The policy may discriminate between different groups of buyers.
- F2** Event organizers may choose to offer a subscription for a series of events.
- F3** Every event can have a pricing policy that differentiates between different groups of buyers (e.g. youngsters or elderly people).
- (F4)** When abuse is detected or when serious incidents happen during the event, it should be possible to identify the buyer of the ticket(s) involved, but only with a court order.
- (F5)** Individuals who have been imposed a banning order for a particular event type, should not be able to buy tickets for this kind of events.

Privacy Requirements

- P1** Buyers of tickets should not directly be identifiable.
- P2** Except when subscriptions are used, it should not be possible to compile buyer's profiles.
- P3** It should not be possible to identify an individual on a blacklist.

3 Technologies

The main technologies required in this report are pseudonym certificates, anonymous credentials, commitment schemes and provable one-way functions.

3.1 Pseudonym Certificates

Pseudonym certificates [1] are traditional certificates where the identity information is replaced by a pseudonym. The certificate states that the identity of the user referred to by that pseudonym and the properties certified in the certificate have been verified by the issuer. Different shows of the same certificate are linkable, which can undermine anonymity.

The relevant functions for both classical and pseudonymous certificates are:

- $U \rightleftharpoons I$: $Cert \leftarrow \text{issueCertificate}(\text{attributes})$. I issues a certificate $Cert$ to U . I knows the certificate attributes , but not the private key corresponding to $Cert$. Pseudonyms, ids, expiry date, etc. are also considered attributes.

- $U \rightarrow V$: $\text{authenticate}(Cert)$. U proves possession of $Cert$ to verifier V . As a result, V gets to know all the attribute values embedded in $Cert$.

Enhanced Pseudonymous Certificates. We further extend the privacy without requiring considerable computational capabilities by replacing each certificate attribute att that contains personal properties (date of birth, social security number, etc.) by $H(att, \text{RANDOM})$. Showing such an enhanced pseudonym certificate thus only reveals personal data if the owner of the certificate also discloses the corresponding (att, RANDOM) tuple to the verifier. Evidently, the linkability issue persists.

3.2 Anonymous Credentials

Anonymous credential systems ([6], [5], [2]) allow for anonymous yet accountable transactions between users and organizations and allow for *selective disclosure* by showing properties of credential attributes (e.g. $age > 18$) while hiding all the other credential attribute information. In the Idemix system [5], different usages of the same credential are *unlinkable* (except when unique attribute values are revealed). Credentials can have features such as an expiry date, the allowed number of times it can be shown and the possibility to be revoked. A mix network ([15], [12]) is required to provide for anonymity at the network layer.

The (simplified) anonymous credential protocols relevant in this paper are:

- $U \rightleftharpoons O$: $(Nym, Sig) \leftarrow \text{generateSignedNym}(Cert)$. One can establish multiple non-transferable pseudonyms (i.e. nym) with the same organization. Here, the user signs the established Nym giving O a provable link between the nym and the identity certified in $Cert$.
- $U \leftarrow I$: $Cred \leftarrow \text{issueCredential}(Nym, attributes)$. A credential is issued by I on a pseudonym Nym . The credential is known only to the user and cannot be shared. Also, a number of attributes, not necessarily known by I , is embedded into the credential.
- $U \rightleftharpoons V$: $transcript \leftarrow \text{authenticate}(Cred, properties, [DeanCond], [Msg])$. A user U authenticates to verifier V by proving possession of a valid credential $Cred$. U can selectively reveal credential attributes or properties thereof. The resulting transcript for V may be deanonymizable upon fulfillment of condition $DeanCond$ (cfr. the $\text{deanonymize}()$). U may decide to sign a message Msg with his credential by a provable link between the transcript and the message. Different transcripts for the same credential are unlinkable (unless the value of a unique attribute is proved).
- $U \rightarrow V$: $\text{prove}(properties)$. Simplified notation of the above function. $Properties$ will refer to the credential used in the proof.
- D : $(Nym, DeanProof) \leftarrow \text{deanonymize}(transcript, condition)$. If a credential show is deanonymizable, the pseudonym Nym on which the credential

was issued can be revealed by a trusted deanonymizer D . $DeanProof$ proves the link between the transcript and the nym. D is only allowed to perform the deanonymization when $condition$ fulfills $DeanCond$ (included in the transcript).

3.3 Commitments

A commitment [14, 8] hides one (or more) values. Later the committer can open the commitment, or prove properties of the committed value(s). The following (simplified) commitment methods are relevant:

- $(Com, OpenInfo) \leftarrow \text{commit}(attribute)$. A new commitment containing a single attribute is generated as well as the opening info required to prove properties about the commitment (or to open it).
- $U \rightarrow V: \text{prove}(Com, properties, OpenInfo)$. Prove properties of commitments.

3.4 Provable One-Way Functions

We define a provable one-way function $out \leftarrow f(in)$ as a one-way function whereof the one knowing in can prove that he knows a in such that $out = f(in)$ in a zero-knowledge proof. Multiple arguments are possible as well.

As an example, according to the DL assumption, $out \leftarrow g^{in} \bmod p$ is such a function for p prime and g a generator of a multiplicative group with order q with $q|p-1$ and p and q sufficiently large.

4 Assumptions and Notation

The general assumptions and notation w.r.t. the protocols are now summed up.

4.1 Assumptions

- For every protocol, a server always first authenticates to U using a classical X.509 certificate. Also, an integrity and confidentiality preserving connection is established during a protocol. Anonymity at the network layer is added when necessary.
- A ticketing server can service multiple events. However, for each event, there is only one ticketing server.
- Tickets do only contain a ticket identifier (e.g. event name, date and seat number) and are unforgeable.

4.2 Notation

- Each protocol requires the following roles: user U (client), ticket server T (issues tickets to users), event organizer E and the court of justice J .
- $U \rightleftharpoons B \rightleftharpoons T$: $(\text{PayProof}_U, \text{PayProof}_T) \leftarrow \text{pay}(\text{price}, \text{Msg})$. U pays an amount of money, via an intermediary bank B , to T . A message can be linked to the payment. The bank can deliver proofs of the payment to both parties. The payment protocols can preserve U 's privacy.
- $U \rightleftharpoons T$: $(\text{desc}[], \text{price}, [\text{Proof}]) \leftarrow \text{negotiate}(\text{Cert} \vee \text{Cred}, \text{Nym} \vee \text{Id}, \text{event}, \text{eventPolicy}, \#\text{tickets}, \text{specification}, [\text{Restrictions}])$ allows U and T to agree on the exact seat numbers as well as on the total price. Therefore, U gives an identifier (Nym or Id), shows (properties of) credential/certificate attributes. The event policy can state e.g. that people younger than 18 get reductions. Evidently, the number and (general) specification of the tickets are given as well. The restrictions on the blacklists can further constrain the possibilities of the user. U can give T a proof of the agreement (signed by Cert or Cred).
- O : $\text{Nym} \leftarrow \text{retrieveOrGenerateNym}(\text{Id} \vee \text{Nym}^*)$ returns a newly generated nym if the user referred to by Id or Nym^* does not yet have a nym with O . However, if that user already has been given a nym in the past, it is simply retrieved from O 's local storage system.
- T : $\text{Restrictions} \leftarrow \text{retrieveRestrictions}(\text{Blacklist}, \text{Nym} \vee \text{Id})$. T looks up in a blacklist the restrictions of a person referred to by Nym or Id .
- G : $\text{Restriction}[] \leftarrow \text{getRestrictionBooleans}(\text{Id})$ implicitly uses all blacklists, and returns for each event type whether or not the user is blacklisted or not.
- Other, self explaining methods are: $\text{add}()$, $\text{lookup}()$, $\text{store}()$, $\text{update}()$ and $\text{generateTickets}()$.

5 Trivial eID-based Solution

5.1 Introduction

Without alternatives, this protocol will most likely be implemented in Belgium as the government is really promoting the use of the eID card in commercial applications. However, this protocol has serious privacy drawbacks.

5.2 Roles

In the trivial protocol, four roles are involved.

Abbreviation	Role
U	The user (client)
G	Government agency (maintain and distribute blacklists)
T	Ticket server (issues tickets to users)
E	Event organizer

5.3 High Level Description

U uses his eID card to authenticate to T , revealing a lot of personal data to T . A government agency G maintains a blacklist containing identifiable user ids. This blacklist is checked by T before issuing tickets.

5.4 Protocols

The user authenticates to T using his eID card. T first checks whether the user is blacklisted. Based on the user's id and personal attributes, the user can be given the possibility to buy a number of tickets as a result of the negotiation phase. After the payment and ticket issuance, T finally stores ticket selling info.

Identification in case of abuse is straight forward since T knows the *link* between the seat (or ticket) and the user's id.

(1.a) Getting a ticket	
(1) U \rightarrow T	: <code>authenticate(eID), EventType</code>
(2) T	: <code>Restrictions \leftarrow retrieveRestrictions(eID.NRN, Blacklists[EventType])</code>
(3) U \rightleftharpoons T	: <code>(SeatNb[], Price) \leftarrow negotiate(eID, event, eventPolicy, eventPolicy, #tickets, specification, [Restrictions])</code>
(4) U, T	: <code>if (SeatNb[] = \emptyset) abort</code>
(5) U \rightleftharpoons B \rightleftharpoons T	: <code>pay(price, H(SeatNb[], ...))</code>
(6) U \leftarrow T	: <code>tickets[] \leftarrow generateTickets(SeatNb[])</code>
(7) T	: <code>update [eID.NRN, event, tickets[]]</code>
(1.b) Maintaining the blacklists	
(1) J \rightarrow G	: <code>eID.NRN, Restrictions, eventType</code>
(2) G	: <code>Blacklists[EventType].add(eID.NRN, Restrictions)</code>
(3) G \rightarrow T	: <code>Blacklists</code>

Table 1: Protocols in trivial implementation

5.5 Evaluation

The functional/security requirements are trivially fulfilled. However for the privacy requirements, this protocol fails completely. T knows the user's id and all other attributes contained in the eID certificate (P1). User profiling is trivial for T as well as sharing and linking of profiles (P2). The users' NRNs are on the blacklist (P3). In addition, many countries simply forbid blacklists on which users are identifiable due to privacy legislation. Deployment will often thus result in omitting the F5 requirement.

6 Solution based on Pseudonym Certificates and Fixed Timeslots

6.1 Introduction

This approach improves the privacy of the user by introducing pseudonymous permits. First, each user is issued a unique pseudonymous root certificate by the government. This allows the user to obtain pseudonymous permit certificates from different permit servers. One permit server could for instance be responsible for one event type (e.g. soccer matches). With such a pseudonymous permit a user can buy tickets for events that happen in a small (permit specific) time period¹. The user will thus most likely need multiple permits. The blacklists no longer contain user identifiers, but pseudonyms.

6.2 Roles

Besides the already defined U , T and E , a government agency G is needed to issue root certificates and a permit server PS issues permit certificates.

Abbreviation	Role
U	The user (client)
G	Government agency (issues pseudonymous root certificates)
PS	Permit server (issues permit-certificates)
T	Ticket server (issues tickets to users)
E	Event organizer
J	Court of justice (investigates and deanonymizes, pronounces judgements resulting in rights restrictions of users)

6.3 Assumptions

- All certificates contain a unique serial number, a pseudonym or id, a public key and an expiry date.
- There can be many pseudonym servers (PS) and many ticket servers (T).
- For every event, the ticket server (T) accepts permits issued by a limited set of pseudonym servers. However, the user sets of different pseudonym servers do not overlap (necessary for requirement F1).
- Only one entity G can issue valid pseudonymous root certificates.
- Nyms that are no longer valid, are forgotten by the permit server.

¹The fixed time period is introduced to minimize linkabilities.

serial number	SN1	SN2	SN3	SN4
pseudonym	Nym1	Nym2	Nym3	Nym4
nym-validFrom	T1	T2	T3	T4
startdate	now	now	now	now
expirydate	T2	T3	T4	T5

Table 2: The issued permit certificates corresponding to figure 1.

6.4 High Level Description and data structures.

The user receives a pseudonymous root certificate (Cert^R), which contains a rootnym (Nym^R) and possibly other attributes (such as year of birth, citizenship, place of residency, ...). Cert^R is used to authenticate to the permit server PS. The user can apply to the PS for a pseudonym (Nym^P) that is valid during a predefined time period. Nym^P will be certified in a (pseudonymous) permit certificate (Cert^P). Each certificate also contains a public key used to verify authentications with Cert^P , and possibly (properties of) other attributes that were copied from the root certificate (Cert^R). Using permit certificates with non-overlapping time-slots, each user can have at most one valid Cert^P to order tickets for a particular event. The PS can refuse permits to users who have been sentenced to a banning order for events supported by the PS.

In figure 1, the concept of fixed timeslots is illustrated. The corresponding permit certificates issued to the user are shown in table 2. For a specific *PS*, all the possessors of a Cert^P enter a new timeslot at the same moment. Also, for all those people, the timeslots last equally long in order to minimize linkability. In the figure, the user requests four Cert^P s, in order to be able to buy tickets that are organised before $T5$. The first timeslot (for which the user is issued $\text{Nym}1$) is already ongoing. For the first event, the certificate containing $\text{Nym}2$ must be used. For Event2 and Event3, the certificate containing $\text{Nym}3$ is used. If the user wants to buy a ticket that is organized later than $T5$, he will need to request extra certificates Cert^P s, containing a $\text{Nym}5$, $\text{Nym}6$, etc. The *PS* stores the (Nym^R , Nym^P , timeslot) tuples. If the user loses a Cert^P , he can obtain new ones, but these will contain exactly the same nyms. Once a timeslot is over, all data can be removed by *PS*. E.g. once $T2$ has passed, (Cert^R , $\text{Nym}1$, $T1$ - $T2$) can be removed.

6.5 Protocols

Getting a root certificate. A governmental instance G assigns to each citizen one root pseudonym Nym^R . The first time U requests a root certificate Cert^R , a new Nym^R is generated and included in Cert^R . In case the user was already assigned a Nym^R in the past, that pseudonym is retrieved from G 's local storage instead. G finally stores the user's NRN and Cert^R s (which include Nym^R).

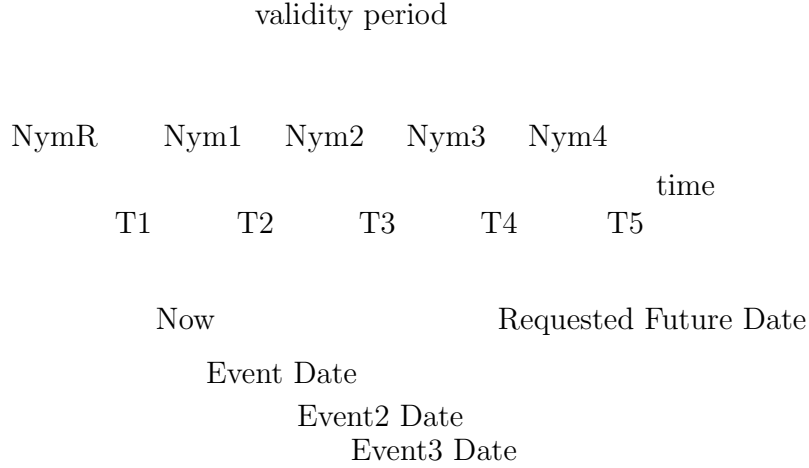


Figure 1: Validity periods of different nyms

Getting a permit certificate. U authenticates with a valid root certificate Cert^R to the PS . PS will issue a number of permit certificates Cert^P s which have to be used before a (user specified) date (validThru). For instance, the user can request permit certificates that allow him to buy soccer tickets for the upcoming year. PS generates a set of nyms (Nym^R) or retrieves them (if they were already assigned in the past): one nym per time period². Each nym Nym^P is also certified in a permit certificate Cert^P which also contains a validity period (for Nym^P), possibly a set of attributes, and an encryption of the user’s root pseudonym Nym^R . The validity periods of Nym^P s are non-overlapping. Hence, users cannot buy tickets for the same event using different nyms. Also, when a user requests a new permit for the same period (e.g. because the previous one was lost or the private key was stolen), PS will always use the same nym (Nym^P). Each Cert^P contains a probabilistic encryption of Nym^R with the public key of J . This allows law enforcement to eventually identify the user involved in case of abusive behavior (see further). PS finally updates the list of Cert^P s that are issued to Nym^R . PS can derive the time intervals for which a Nym^R has obtained a valid Cert^P from that list.

Buying tickets for an event. The user first authenticates to the ticket server T using the permit certificate Cert^P that is valid for that specific event and specifies the number of tickets he wants to order. T then obtains the restrictions associated with Nym^P on the blacklist. The user and the ticket server agree on the price of the tickets and the seats, based on the user’s nym, allowing to limit the number of tickets for that user. The limitations and price can depend on

²The length of the non-overlapping time periods is chosen by the PS in such a way that the number of events that fall in each period is limited.

certain attributes that are embedded in the permit certificate (such as the user's age) and on the restrictions on the blacklist. Finally, the ticket server updates the number of tickets that are sold to Nym^P for that event.

Updating anonymous blacklists. To fulfill requirement F4, anonymous blacklists are used. Four entities are involved in updating blacklists (see table 4).

A law enforcement entity J forwards the court orders (NRN , *Restrictions*) to G . G substitutes the NRNs with the corresponding Nym^R s and forwards the list to the permit server PS . PS can then add Nym^R to a blacklist for certain event types (i.e. PS will no longer issue Cert^P s to Nym^R for the event types that are specified in the blacklist).

Finally, PS retrieves the valid Nym^P s for each Nym^R with a banning order, substitutes every Nym^R -record in the blacklist with a number of Nym^P -records and forwards the new list to the ticket server T . T no longer issues tickets to pseudonyms in the blacklist. Note that the ticket service can even revoke tickets that were already issued to pseudonyms in the blacklist.

Identifying buyer of a ticket. To reveal the identity of a participant with a specified seat number, the ticket service T looks up the Nym^P of the user that ordered the ticket. The corresponding permit certificate Cert^P is kept by the ticket server and is passed to J . The latter can link Cert^P to Nym^R (as Nym^R is encrypted with the public key of J in Cert^P). G can reveal the user behind Nym^R (as G knows the mapping between NRN and Nym^R).

6.6 Evaluation

- F1 This requirement is easily fulfilled as each user has only one Nym^P to buy tickets for a particular event.
- F2 If Nym^P can be used to order tickets for multiple events (e.g. multiple soccer games during a World Cup contest), T can even restrict the total number of tickets that can be bought for the whole contest (i.e. a set of events).
- F3 A user can get a lower price for some tickets based on the attribute values of Cert^P . However, tickets can be passed on. Hence, T should be careful with price reductions.
- F4 Fulfilled (cfr. "*Identifying buyer of a ticket*" protocol).
- F5 Three entities are needed to ban a user from event types for which a user already has a permit certificate, namely G , PS and T . Two entities are needed to ban a user from event types for which a user does not yet have a permit certificate, namely G and PS .

(3.a) Getting a pseudonymous root certificate Cert^R		
(1)	$U \rightarrow G$: <code>authenticate(eID)</code>
(2)	G	: <code>$\text{Nym}^R \leftarrow \text{retrieveOrGenerateNym}(eID.NRN)$</code>
(3)	$U \leftarrow G$: <code>$\text{Cert}^R \leftarrow \text{issueCertificate}(\{\text{Nym}^R, \text{attributes} \dots\})$</code>
(4)	G	: <code>store [eID.NRN, Cert^R]</code>
(3.b) Getting a permit certificate Cert^P		
(1)	$U \rightarrow PS$: <code>authenticate(Cert^R)</code>
(2)	$U \rightarrow PS$: <code>validThru, attributes to include</code>
(3)	PS	: <code>\forall [from,till], from \leq validThru:</code>
(4)	PS	: <code>$\text{Nym}^P \leftarrow \text{retrieveOrGenerateNym}(\text{Cert}^R.\text{Nym}^R, [\text{from},\text{till}])$</code>
(5)	$U \leftarrow PS$: <code>$\text{Cert}^P \leftarrow \text{issueCertificate}(\{\text{Nym}^P, [\text{from},\text{till}], \text{attributes},$</code> <code>$\text{enc}_{pk_J}(\text{RANDOM} \parallel \text{Cert}^R.\text{Nym}^R)\})$</code>
(6)	PS	: <code>store [$\text{Cert}^R.\text{Nym}^R$. [from,till], Cert^P]</code>
(3.c) Buying tickets		
(1)	$U \rightarrow T$: <code>authenticate(Cert^P)</code>
(2)	$U \rightarrow T$: <code>event, #tickets, specification</code>
(3)	T	: <code>$\text{Restrictions} \leftarrow \text{retrieveRestrictions}(\text{Cert}^P.\text{Nym}^P, \text{EventType})$</code>
(4)	$U \rightleftharpoons T$: <code>(SeatNb[], price) \leftarrow negotiate($\text{Cert}^P.\text{attr}$, $\text{Cert}^P.\text{Nym}^P$, event,</code> <code>eventPolicy, #tickets, specification, [Restrictions])</code>
(5)	U, T	: <code>if (SeatNb[] = \emptyset) abort</code>
(6)	$U \rightleftharpoons B \rightleftharpoons T$: <code>pay(price, Hash(SeatNb[], ...))</code>
(7)	$U \leftarrow T$: <code>tickets[] \leftarrow generateTickets(SeatNb[])</code>
(8)	T	: <code>update [Cert^P, event, tickets[]]</code>

Table 3: Protocols with pseudonym certificates

P1 As discussed in "Identifying buyer of a ticket", four entities are needed to reveal the user's identity. Moreover, G (and maybe PS) are governmental instances. Hence, users can trust that players in the commercial sector (such as E and T) cannot identify users without help of governmental instances.

P2 Each Nym^P only has a limited validity period. The number of tickets that is issued to the same Nym^P is restricted. Hence, T and E can only compile limited profiles. PS can link all Nym^P s to the same Nym^R . However, multiple pseudonym servers PS can be used. If each PS can only issue permit certificates for specific types of events, the one PS cannot link multiple interests of the same Nym^R .

P3 Only Nym^R s and Nym^P s are kept in blacklists.

(4.a) anonymizing the blacklists		
(1)	$J \rightarrow G$: [NRN, <i>Restrictions</i> , eventType]
(2)	G	: $Nym^R \leftarrow \text{lookupNym}(\text{NRN})$
(3)	$G \rightarrow PS$: [Nym^R , <i>Restrictions</i> , eventType]
(4)	PS	: $Nym^P \leftarrow \text{lookupNym}(Nym^R, \text{eventType})$
(5)	$PS \rightarrow T$: [Nym^P , <i>Restrictions</i> , eventType]
(4.b) Identifying buyer of a ticket		
(1)	$J \leftarrow E$: complaint, seatNb
(2)	$J \rightarrow T$: event, seatNb
(3)	$J \leftarrow T$: [$Cert^P$, event, ticket] $\leftarrow \text{lookup}(\text{event}, \text{seatNb})$
(4)	J	: ($\text{RANDOM} \parallel Nym^R$) $\leftarrow \text{dec}_{pk_J}(Cert^P.\text{enc})$
(5)	$J \rightarrow G$: Nym^R
(6)	$J \leftarrow G$: $\text{NRN} \leftarrow \text{lookup}(Nym^R)$

Table 4: Protocols with pseudonym certificates (bis)

7 Solution based on Pseudonym Certificates and User defined timeslots

7.1 Roles

The same roles as in the previous section are required. Besides the already defined U , T and E , a government agency G is needed to issue root certificates and a permit server PS issues permit certificates.

Abbreviation	Role
U	The user (client)
G	Government agency (issues pseudonymous root certificates)
PS	Permit server(issues permit-certificates)
T	Ticket server (issues tickets to users)
E	Event organizer
J	Court of justice (investigates and deanonymizes, pronounces judgements resulting in rights restrictions of users)

7.2 Assumptions

- All certificates contain a unique serial number, a pseudonym or id, a public key and an expiry date.
- There are many permit servers (PS) and many ticket servers (T).
- For every event, the ticket server (T) accepts permits. issued by a limited set of pseudonym servers. However, the user sets of different pseudonym servers do not overlap (necessary for requirement F1).
- Only one entity G can issue valid pseudonymous root certificates.

serial number	SN1	SN2	SN3
pseudonym	Nym3	Nym3	Nym4
previous pseudonym	[Nym1,T3] [Nym2,T3] [Nym3,T4]	[Nym3,T4]	\emptyset
startdate	now	T3	T4
expirydate	T3	T4	T4+ δ

Table 5: The issued permit certificates corresponding to figure 2

- Nyms that are no longer valid, are forgotten by the permit server.

7.3 High Level Description

U receives a pseudonymous root certificate (Cert^R), which contains a rootnym (Nym^R) and possibly other attributes (such as year of birth, citizenship, place of residence, ...). The Cert^R will be used to authenticate to the permit server PS .

U can request a new nym (Nym^P) which will be certified in a permit certificate (Cert^P). Each certificate also contains a public key (used to verify authentications with Cert^P), and possibly other attributes that were copied from the root certificate (Cert^R). The certificate also lists previous nyms of U that could have been used instead of the current one (cfr. further).

The PS may refuse permits to U s who have been convicted with a banning order for events supported by the PS . Likewise, T may refuse to sell tickets to a U with a banning order for that particular event.

7.4 Scenarios and Data Structures

Figure 2 illustrates the use of nyms. Nyms typically have a **validity period** and a **lifetime**. The former one determines how long a nym can be used to buy tickets; the latter one is longer and determines how long the nym must be retained by the PS . Note that the validity period ends for a nym as soon as another nym has been requested. The lifetime ends when the last event for which the nym could have been used to buy tickets for that event has passed.

In the previous scenario, U requests at time (Now) a new nym (Nym4). The PS will revoke all certificates of previous nyms and will return (in this case) 3 certificates (cfr. table 2): the first one (SN1) will be valid immediately; the second one (SN2) is valid from time (T3) on; the last one is valid at time (T4). Note that the first two certificates also contain previous nyms that could have been used in previous transactions for events that are known to the PS . Hence, it is up to U to determine when to buy tickets: if he wants to buy them before T4, other nyms will be revealed to the ticket server.

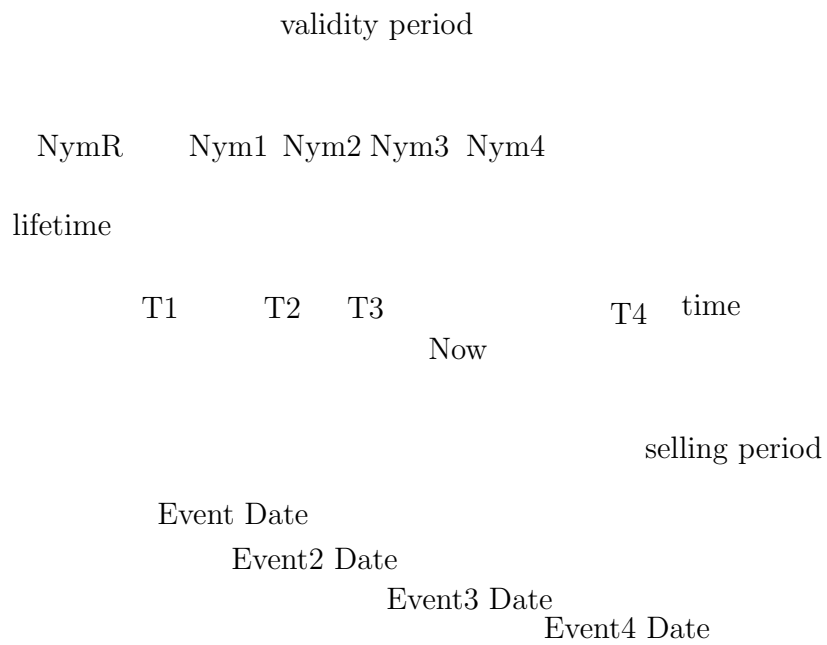


Figure 2: Validity and Lifetime periods of nyms

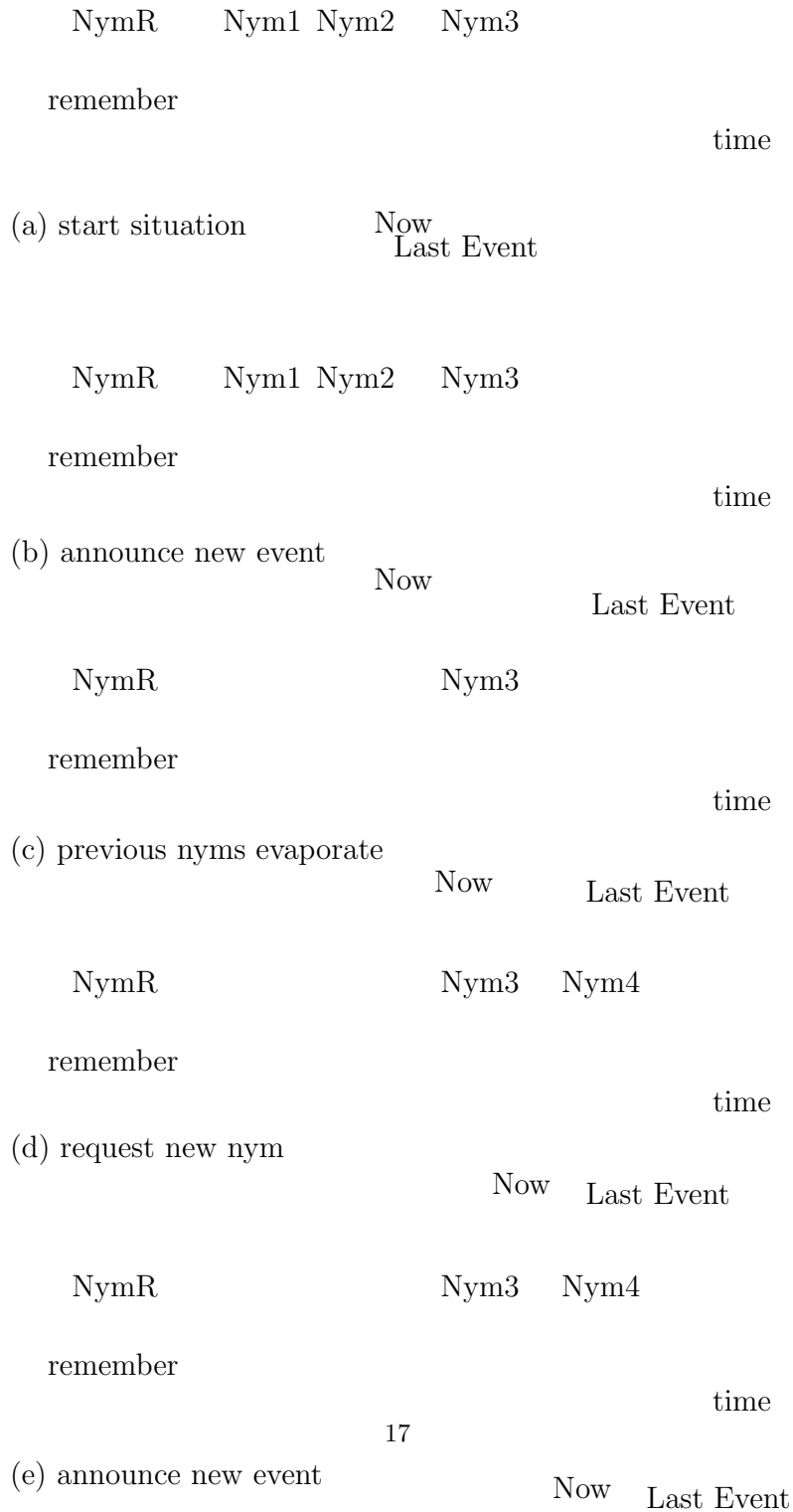


Figure 3: Scenario for nym's lifetime

7.5 Protocols

Table 6 shows the basic protocols.

Getting a pseudonymous root certificate. This protocol (see table 6.a) provides U with a certificate that is more privacy friendly than the certificates of his eID card. The national registration number is replaced by a pseudonym. Also, the name attribute will be replaced by less identifying attributes (such as year of birth, place of residence, ...).

In this protocol, U authenticates with his eID card to G . G will generate a root nym (Nym^R) or retrieve it if it was generated previously, and will issue a root certificate (Cert^R) to U . (Note that the certificate also contains a public key of U , which has been generated by U prior to the execution of this protocol; U will also prove that he knows the corresponding private key).

Getting a permit certificate. In this protocol (see table 6.b), U requests a new nym and corresponding certificates which are necessary for buying tickets. PS will first look up the current nym of U and sets its lifetime until the last event that has been announced to the PS . (For this event and previous ones, the nym could already have been used to buy tickets). All certificates for this nym are revoked (they cannot be used anymore for buying tickets). In the next step, PS will generate a new nym and issue certificates for that nym. The number of certificates depends on the lifetimes of the previous nyms. The first certificate, which will be valid instantaneously, also includes all the previous nyms that have not yet expired lifetimes. The second certificate will be valid when the oldest nym's lifetime expires, and includes the previous nyms that have not yet expired, etc.

Buying tickets. A buyer first authenticates with his permit certificate, then sends the specification for the tickets to be bought. T may refuse to sell tickets to U when U 's nym (Nym^P) is on the blacklist for this event. Otherwise, T will check whether the limit has not been reached yet (by checking the number of tickets bought with nym Nym^P and all other nyms that are included in (Cert^P)). If the limit has not been exceeded, payment is accepted, and the tickets are sent to U . (See also table 6.c)

Announcing a new event. As soon as tickets for a new event will be sold, the T must inform PS of that fact (see table 6.d). If the date of the event is later than any other event already known by PS , this date will extend the lifetimes of all currently valid nyms. Therefore, that date will be recorded in a local variable "validThru".

(6.a) Getting a pseudonymous root certificate Cert^R	
(1)	$U \rightarrow G$: <code>authenticate(eID)</code>
(2)	G : <code>Nym^R ← retrieveOrGenerateNym(eID.NRN)</code>
(3)	$U \leftarrow G$: <code>Cert^R ← issueCertificate(Nym^R, attributes ...)</code>
(4)	G : <code>store [eID.NRN, Cert^R]</code>
(6.b) Getting a permit certificate Cert^P	
(1)	$U \rightarrow \text{PS}$: <code>authenticate(Cert^R)</code>
(2)	$U \rightarrow \text{PS}$: <code>attributes to include</code>
(3)	PS : <code>(Nym^R, Nym^P, PrevNyms[]) ← lookup(Cert^R.Nym^R)</code>
(4)	PS : <code>revoke(certificates for previous nym Nym^P)</code>
(5)	PS : <code>Nym^{P'} ← generateNym()</code>
(6)	PS : <code>∀ nym validity expiry timestamps</code>
(7)	$U \leftarrow \text{PS}$: <code>Cert^P ← issueCertificate({Nym^{P'}, PrevNyms, attributes, enc_{pk,j}(RANDOM Cert^R.Nym^R)})</code>
(8)	PS : <code>update (Nym^R, Nym^{P'}, Cert^P[], append(PrevNyms, [Nym^P, validThru])</code>
(6.c) Buying tickets	
(1)	$U \rightarrow T$: <code>authenticate(Cert^P)</code>
(2)	$U \rightarrow T$: <code>event, #tickets, specification</code>
(3)	T : <code>if (blacklisted(Cert^P.Nym^P, event)) abort</code>
(4)	$U \rightleftharpoons T$: <code>(SeatNb[], price) ← negotiate(Cert^P.attr, Cert^P.Nym^P, event, #tickets, eventPolicy, specification)</code>
(5)	U, T : <code>if (SeatNb[] = ∅) abort</code>
(6)	$U \rightleftharpoons B \rightleftharpoons T$: <code>pay(price, Hash(SeatNb[], ...))</code>
(7)	$U \leftarrow T$: <code>tickets[] ← generateTickets(SeatNb[])</code>
(8)	T : <code>update [Cert^P, event, tickets[], append(previousNym, Cert^P.previousNym)]</code>
(6.d) Announcing new event	
(1)	$T \rightarrow \text{PS}$: <code>new event (date)</code>
(2)	PS : <code>if (date > validThru) validThru ← date</code>

Table 6: Protocols with pseudonym certificates and user chosen timeslot

Optional functional requirements. When blacklisting is implemented, the blacklists are forwarded from G to PS and from PS to T . To protect the privacy of U , the blacklists are anonymized (see also 7.a).

Note that PS must forward a new $[\text{Nym}^P, \text{banning order}]$ tuple, each time a U (with banning order) requests a new nym.

Identifying the buyer of a ticket is described in protocol 7.b. First the seat numbers are mapped onto tickets; then, T will recover the permit certificate with which the tickets were bought. This certificate contains an encrypted field $\text{Cert}^P.\text{ENC}$, which can be decrypted by PS . It contains the root nym Nym^R of the buyer. That root nym can be mapped by G onto the real identity of U .

(7.a) anonymizing the blacklists		
(1)	$J \rightarrow G$: $[\text{NRN}, \text{Restrictions}, \text{eventType}]$
(2)	G	: $\text{Nym}^R \leftarrow \text{lookupNym}(\text{NRN})$
(3)	$G \rightarrow PS$: $[\text{Nym}^R, \text{Restrictions}, \text{eventType}]$
(4)	PS	: $\text{Nym}^P \leftarrow \text{lookupNym}(\text{Nym}^R, \text{eventType})$
(5)	$PS \rightarrow T$: $[\text{Nym}^P, \text{Restrictions}, \text{eventType}]$
(7.b) Identifying buyer of a ticket		
(1)	$J \leftarrow E$: complaint, seatNb
(2)	$J \rightarrow T$: event, seatNb
(3)	$J \leftarrow T$: $[\text{Cert}^P, \text{event}, \text{ticket}] \leftarrow \text{lookup}(\text{event}, \text{seatNb})$
(4)	J	: $(\text{RANDOM} \parallel \text{Nym}^R) \leftarrow \text{dec}_{pk_J}(\text{Cert}^P.\text{enc})$
(5)	$J \rightarrow G$: Nym^R
(7)	$J \leftarrow G$: $\text{NRN} \leftarrow \text{lookup}(\text{Nym}^R)$

Table 7: Protocols with pseudonym certificates and user chosen timeslots (bis)

7.6 Evaluation

Functional/Security Requirements

F1 This requirement is fulfilled if –for every U – there is only one T and one PS per event. PS s can request many different nym; however, all the nym that could have been used to buy tickets for that event will still be retained in PS , and hence, they will all appear in the valid permit certificate. The T will have to sum up the number of tickets bought by all these nym and decide whether or not U is allowed to buy (additional) tickets.

Note that at any one time, there will always be at most one permit certificate per U and per PS valid. When a new nym is requested, all certificates for previous nym are revoked. The certificates for the new nym will have no overlapping validity periods.

If the permit certificate contains attributes that assert properties of the holder, these attributes can be taken into account. The PS

is trusted to either copy correctly these attributes or properties of these attributes from the root certificate (e.g. copy the year-of-birth attribute from the root certificate, or verify that "this year ≥ 18 " before including a "being of age" attribute in the permit certificate).

- F2** E may choose to offer a subscription for a series of events.
- F3** Every event can have a pricing policy that differentiates between different groups of Us (e.g. youngsters, residents of the city where the event is organized, ...).

Optional Functional/Security Requirements

- F4** Protocol 7.b describes which data must be exchanged in order to be able to deanonymize the buyer of a ticket.
We assume that the ticket contains the seat-number.
Note that all parties must collaborate (G , PS and T).
- F5** If blacklists are implemented, PS can refuse to issue a permit certificate for such a U . (This implies that U cannot buy any tickets at all)
Another possibility is that PS forwards the nym of U together with the banning order to T . Note that, in this case, the PS will have to contact T each time a U with a banning order requests a new nym.

Privacy Requirements

- P1** At least 3 different parties must collaborate in order to identify a user.
The following table describes what information can be derived by the individual servers and when two of them collude.

Colluding Parties	Additional Information obtained
G	ID, Nym^R , banning order(s)
SP	Nym^R , [Nym^P , $Nym^{P'}$, ...], banning order(s), $Cert^R$ note that Nym^P evaporates when its retention time as expired
T	Nym^P , [$Nym^{P'}$, ...], banning order(s) tickets, $Cert^P$
G + SP	ID, [Nym^P , $Nym^{P'}$, ...]
G + T	There is no way to link ID and Nym^P
SP + T	Nym^R , [Nym^P , $Nym^{P'}$, ...], tickets This does not identify the buyer

- P2** The T can compile a limited buyer's profile, if the buyer uses permit certificates that include previous nym that were used to buy tickets for other events.
It heavily depends on

1. the number of events that are organized in a time interval,
2. how long before the event date tickets can be bought (the longer this period, the higher the probability that linking of difference transactions will be possible)
3. the number of events that U wants to attend

As a rule of thumb, U should request a new nym after every transaction. The previous nym will still be retained by PS , and for at least the date of the event for which tickets have been bought (but it could be far longer, if for another event (far in the future) already tickets can be ordered).

P3 The blacklists are anonymized. The PS only receives [Nym^R, banning order] tuples. The T only receives [Nym^P, banning order] tuples.

8 Further improving the privacy

By using enhanced pseudonym certificates, the privacy can be further increased by reducing the amount of personal data that is revealed by U to other entities. First, we show how the amount of data released to T can be reduced, and thereafter we show how the amount of personal data shown to the PS can be minimized.

8.1 Minimizing personal data disclosure to T and reducing information ..

Instead of including a personal attribute att in the Cert^R, G can include $H(att, rand_G)$, where $rand_G$ is a random value generated by G . For each personal attribute, another $rand_G$ value can be generated. As part of the Cert^R protocol, G sends all the resulting $rand_G$ values to U . If U now authenticates to a PS , he send only those $rand_G$ values to PS , of the attribute values he wants to disclose to PS . The PS can now, in a similar way as the G , include the disclosed attribute values in the newly issued Cred^P. Again, for each personal attribute included in that certificate, PS will generate a $rand_{PS}$ value, include $H(att, rand_{PS})$ in the certificate and sent the corresponding $rand_{PS}$ values to U . Now, U is able to selectively disclose a subset of the attributes contained in Cert^P to the T s. Evidently, the T s can still share known the $rand_{PS}$ values and get hold of more information.

Of course, by splitting up one attribute in more attributes, the amount of data that is released can be further reduced. For example, if the T only wants to know whether you are older than 18, you can reveal your date of birth (DoB) attribute. However, if the DoB attribute is split into a year of birth, a month of birth and a day of birth attribute, usually, less information will need to be disclosed. In most cases, the year of birth attribute will satisfy. We can even go further by adding an age category attribute. However, this way of splitting up attributes becomes soon very impractical.

8.2 Minimizing personal data disclosure to T and PS

Still, PS knows a lot personal data of U . Can we reduce this? and at what price?

For each personal attribute att in $Cert^R$, G can include the hash value $H(att, rand_G)$ in $Cert^R$ instead, where $rand_G$ is a random value. Each $rand_G$ is sent to U as part of the issuance of $Cert^R$. After authentication by U to PS with $Cert^R$, a subset of the hashes thereof is selected and hashed again by PS , in a similar way, to $H(H(att, rand_G), rand_{PS})$ and included in $Cert^P$ which is sent to U , together with the corresponding $rand_{PS}$ values. This allows U to hide attributes for PS and to selectively disclose personal attributes to a T by authenticating using $Cert^P$ combined with sending a subset of the $(rand_G, rand_{PS})$ tuples to T .

Thus, no PS obtains any personal attributes contained in $Cert^R$ (instead, the attribute hash values are hashed again). Only a subset of the attributes are revealed to T by U when the latter wants to buy a ticket. Evidently, different T 's can collaborate in order to get hold of more personal attribute.

Now, different certPs are unlinkable: Each $Cert^P$ contains a different encryption of Nym^R , a different Nym^P and different values of hashed attributes values. However, in order to show an attribute to a T , the user needs to reveal not only $rand_{PS}$, but also $rand_G$, the latter is the same for each $Cred^P$ of the user containing that attribute. Thus if the same attribute is shown to two different T s, they can link both certificates to the same user.

Therefore we define new hash functions H' , H'' and H''' , each with two parameters. such that $H''(H'(msg, r_1), r_2) = H'''(msg, f(r_1, r_2))$ where the result of f cannot be used to derive either r_1 or r_2 , which are two random numbers (f could for instance simple addition), and H''' can be easily derived from H' and H'' .

H' is used for the first hashing by G , H'' by PS for the second hashing and H''' is used for the verification by T . In that case, the user sends the attribute value and an $r \leftarrow f(rand_G, rand_{PS})$ to T in order to reveal an attribute, while another $rand_U \leftarrow f(rand_G, rand_{PS}')$ is sent to T' to reveal the same attribute. r and r' are unlinkable.

We now illustrate how such a set of hash functions can be constructed. Therefore we need the strong RSA assumption: The RSA problem (finding the roots modulo a composite number with unknown factorisation) is intractable even when the solver is allowed to choose the public exponent e ($e > 2$). More specifically, given a modulus N of unknown factorization, and a ciphertext C , it is infeasible to find any pair (M, e) such that $C = msg^e \text{ mod } N$.

We now show how such functions H' , H'' , H''' and f can be defined.

- $msg' \leftarrow H'(msg, r) : m' \leftarrow m^e \cdot r^e \text{ mod } N$
- $msg' \leftarrow H''(msg, r) : m' \leftarrow m^e \cdot r \text{ mod } N$
- $msg' \leftarrow H'''(msg, r) : m' \leftarrow m^{2e} \text{ mod } N$
- $r \leftarrow f(r_1, r_2) : r \leftarrow r_1 \cdot r_2^e \text{ mod } N$

U needs to store either all $rand_G$ values in order to be able to disclose attributes to PS s. To disclose attributes to T s, U needs to store either all $rand_U$ values or all $rand_{PS}$ values.

8.3 Pseudonym certificates with Commitments

Instead of adding hashed values in $Cert^R$ and $Cert^P$ commitments can be included. This allows U to prove properties about attributes. However, this goes at a heavy computational cost. Moreover, it comes close to anonymous credentials, which can offer more enhanced privacy capabilities.

When a $Cert^R$ is issued, U first generates a commitment, sends it to G and proves properties thereof. Alternatively, the commitments can be generated by G (e.g., based on the attribute information associated with U 's eID card) and the opening info can be sent to U . To get a $Cert^P$ issued, U generates commitments based on the attribute values in $Cert^R$ sends these to PS , proves that they correspond to certain values in $Cert_G$ and also proves properties about the commitments. Based on the information PS now knows, he can issue a $Cert^P$. In a similar way, U will reveal properties of $Cert^P$ to T .

The advantage is that PS and T can only get hold of the minimum amount of personal data about U . The disadvantages are the computational cost while linkability issues persist.

9 Solution Based on Anonymous Credentials

9.1 Introduction

We further increase the user's privacy. The user needs a single permit - issued by a government agency - which allows the user to buy tickets for every event. In case of abuse, the transcript resulting from the permit show can be deanonymized. For each event type, there is a privacy-preserving blacklist, summing up the user's rights restrictions.

9.2 Roles

Besides U , E , T , and J , we define G as a government agency that issues permits and manages blacklists.

Abbreviation	Role
U	The user (client)
G	Government agency (issues permits and manages blacklists)
T	Ticket server (issues tickets to users)
E	Event organizer
J	Court of justice (investigates and deanonymizes, pronounces judgements resulting in rights restrictions of users)

9.3 Assumptions

In the ticketing system based on anonymous credentials, we assume the following:

- The anonymous credential system provides the unlinkability property to permits. The user does not reveal identifiable permit attribute properties.
- All E s and all T s and G have a unique, publicly available provable one-way function; $f^E()$ for E , $f^T()$ for T and $f^G(. , .)$ for G . Note that the latter requires two arguments. These functions could for instance be included in their X.509 certificate.
- The opening info generated by a commit method does not reveal any information about the content contained in the commitment. This is easily achieved using a symmetric key K :
 $Com^{new} \leftarrow (Com, enc_K(OpenInfo))$ and $OpenInfo^{new} \leftarrow K$ combined with integrity preserving measures (e.g. MACs).

9.4 High Level Description

The permit is an anonymous credential containing a set of personal attributes, a boolean value for each event type indicating whether or not the user is blacklisted, and two nyms. One nym (Nym^R) is known to G and used to blacklist persons. The other nym (Nym^P), is not known to G , but is used to generate an event specific nym, allowing T to keep track of the number of tickets sold to that person for that specific event.

Per event type, a blacklist is maintained by G . This blacklist contains user pseudonyms (Nym^R s). These nyms are converted to event specific nyms (Nym^E s) before the blacklist is sent to a specific T in order to avoid linkabilities.

9.5 Protocols

Getting an anonymous Permit Certificate. The actual issue of the permit (8.a.5) includes a subset of the user's personal attributes (*attributes*) contained in the user's eID. These can be selectively disclosed during a credential show protocol.

The permit contains for each event type a boolean $Restrictions[EventType]$ stating whether or not the user is blacklisted. G can easily extract this information out of the blacklists it manages (cfr. below).

Each permit contains two user unique pseudonyms Nym^R and Nym^P . Nym^R is known to both U and G and is the nym under which the permit is issued by G . G possesses a provable link Sig^R between the U 's id and his Nym^R . This can be used in case of disputes.

The second pseudonym in the permit, Nym^P , is known to the user U only and is included in the permit as an attribute that is not known to G . This is done using a commitment, whereof U proves that he knows the corresponding

$UserSecret$ and Nym^P (underlined in table 8) such that $Nym^P \leftarrow f^G(Nym^R, UserSecret)$.

To obtain a new permit, after the previous one was lost, step 6 changes. After recalculating $Nym^P \leftarrow f^G(Nym^R, UserSecret)$ and generating a new commitment $Com2 \leftarrow \text{commit}(Nym^P)$ (Step 4 and 5), U decrypts c , resulting in the opening info of the previous commitment. This allows U to prove that $Com.Nym^P = Com2.Nym^P$ (corresponds to step 6), convincing G that the same Nym^P was used.

Buying a Ticket. For each ticket order, U sends $Nym^E \leftarrow f^E(Nym^P)$ to T and proves possession of the corresponding Nym^P (8.b.1,2). The use of one-way functions gives the user for each event a different, but event-unique nym. This gives T the possibility to limit the number of tickets per user while at the same time, this function avoids linking of T 's customers to the customers of other T s. Collusion with G does not help, because G does not even know Nym^P .

When ordering a ticket, the user proves that he is not blacklisted by showing $Restrictions[EventType]$. If U is blacklisted, he sends $Nym^T \leftarrow f^T(Nym^R)$ to T and proves that Nym^T is correctly formed with $Cred^P.Nym^R$. T now looks up the exact restrictions associated with Nym^T on the blacklist (8.b.3). This limits linking possibilities and possible collusion with G . The latter can only be done for blacklisted Us .

The negotiation phase (8.b.4) requires the user's permit as input, such that RequestProof can be generated. RequestProof is a proof for G that U did request the negotiated tickets at the negotiated price. This proof is also deanonymizable by J which provably reveals Nym^R .

Blacklist Maintenance and Retrieval. A law enforcement entity J forwards the court orders ($NRN, Restrictions$) to G . G substitutes the NRNs with the corresponding Nym^R s. Each Nym^R is further converted to $Nym^T \leftarrow f^T(Nym^R)$ before the blacklist is sent to a specific T to avoid linkabilities and profiling by T (9.b).

Misbehaviour and Deanonymization. Protocol 9.c illustrates how the collaboration of E , T and G is required in order to obtain a (provable) link between the ticket and the user's id. The proof is (RequestProof, deanProof, Sig^R). If someone is put on a blacklist for EventType, his permit $Cred^P$ is revoked. U can obtain a new $Cred^P$, with the updated restrictions booleans Restriction[EventType], immediately.

9.6 Evaluation

We now evaluate by checking the requirements

Functional and Security Evaluation

(8.a) Getting the first anonymous permit certificate Cred^P		
(1)	$U \rightarrow G$: <code>authenticate(eID)</code>
(2)	$G \Leftarrow U$: <code>(Nym^R, Sig^R) ← generateSignedNym(eID.NRN)</code>
(3)	G	: <code>Restriction[] ← getRestrictionBooleans(eID.NRN)</code>
(4)	$U \Leftarrow G$: <code>Nym^P ← f^G(Nym^R, UserSecret)</code>
(5)	$U \rightarrow G$: <code>(Com, OpenInfo) ← commit(Nym^P)</code>
(6)	$U \rightarrow G$: <code>Com, prove(Com.Nym^P = f^G(Nym^R, UserSecret)), c ← enc_H(UserSecret)(OpenInfo)</code>
(7)	$U \Leftarrow G$: <code>Cred^P ← issueCredential(Nym^R, {Com.Nym^P, Restriction[], attributes})</code>
(8)	G	: <code>store [eID.NRN, Nym^R, Sig^R, Com, c]</code>
(8.b) Buying tickets		
(1)	$U \rightarrow T$: <code>Nym^E ← f^E(Cred^P.Nym^P), event</code>
(2)	$U \rightarrow T$: <code>authenticate(Cred^P, {Cred^P.Nym^P ≈ Nym^E, Cred^P.Restriction[EventType]})</code>
(3)	T	: <code>if(Cred^P.Restriction[EventType] == true) do</code>
(3.a)	$U \rightarrow T$: <code>Nym^T ← f^T(Cred^P.Nym^R)</code>
(3.b)	$U \rightarrow T$: <code>prove(Nym^T ≈ Cred^P.Nym^R)</code>
(3.c)	T	: <code>Restrictions ← retrieveRestrictions(Blacklist_T, Nym^T)</code>
(3.d)	T	: <code>end if</code>
(4)	$U \Leftarrow T$: <code>(SeatNb[], price, RequestProof) ← negotiate(Cred^P, event, Nym^E, #tickets, eventPolicy, [Restrictions])</code>
(5)	$U \Leftarrow B \Leftarrow T$: <code>(PayProof_U, PayProof_T) ← pay(price, Hash(SeatNb[], ...))</code>
(6)	$U \leftarrow T$: <code>tickets[] ← generateTickets(SeatNb[])</code>
(7)	T	: <code>update [event, Nym^E, RequestProof, tickets[]]</code>

Table 8: Protocols with anonymous credentials

F1 $Nym^E \leftarrow f^E(Nym^P)$ enables T to link ticket orders of the same U for the same event.

F2 A subscription can be issued by T or a coordinating organization. It can be an anonymous credential that contains Nym^P , Nym^R , the `Restriction[EventType]` booleans and information about the subscription. It can be pseudonymously shown to a ticketing service in order to obtain tickets without a payment phase. Alternatively, a multiple-use ticket with an expiry date can be issued.

F3 The user can selectively disclose properties in the permit.

F4 is explained in section 9.5.

F5 is done using the anonymized blacklists. Revocation of tickets issued to persons that were blacklisted after the ticket order is possible if Nym^R is systematically shown to T . However, the price is an increase in linkabilities.

Privacy Evaluation

(9.a) Maintaining the blacklists	
(1)	$J \rightarrow G$: $\text{Nym}^R, \text{Restrictions}, \text{EventType}$
(2)	G : $\text{Blacklists}[\text{EventType}].\text{add}(\text{Nym}^R, \text{Restrictions})$
(3)	$J \rightarrow G$: $\text{revokeCert}(\text{Nym}^R)$
(9.b) Obtaining a blacklist	
(1)	G : for each $(\text{Nym}^R, \text{Restrictions})$ in $\text{Blacklists}[\text{EventType}]$: $\text{Blacklist}_T.\text{add}(f^T(\text{Nym}^R), \text{Restrictions})$
(2)	$T \leftarrow G$: Blacklist_T
(9.c) Identifying buyer of a ticket	
(1)	$J \leftarrow E$: $\text{complaint}, \text{seatNb}$
(2)	$J \rightarrow T$: $\text{event}, \text{seatNb}$
(3)	$J \leftarrow T$: $\text{RequestProof} \leftarrow \text{lookup}(\text{event}, \text{seatNb})$
(4)	J : $\text{Nym}^R, \text{deanProof} \leftarrow \text{deanonymize}(\text{RequestProof}, \text{complaint})$
(5)	$J \rightarrow G$: $(\text{NRN}, \text{Sig}^R) \leftarrow \text{lookup}(\text{Nym}^R)$

Table 9: Protocols with anonymous credentials (bis)

P1 Deanonymization requires the collaboration of T , G and J as we argued in *Misbehaviour and Deanonymization*.

P2 We argued that a user has for each E a different $\text{Nym}^E \leftarrow f^E(\text{Nym}^P)$. Different E s thus should know the user's Nym^P – which remains hidden – to do linking. For blacklisted users, G can link Nym^R and Nym^T . Collusion of T and G is then possible.

P3 G knows the links between nym on a blacklist and the user's id. However, such convictions are publicly available. Collusion of T and G can reveal the identity associated with Nym^T .

10 Comparison

Table 10 compares the three approaches; the main functional/security requirements can be fulfilled while boosting privacy. To maintain user-friendliness, the interactions with e.g. PS can be done transparently to the user. The proposed solutions disallow a banned person to buy tickets for someone else (e.g. father for his children) and it is still possible that a person buys tickets and gives them to a banned person.

Estimates of the feasibility on the server side where done on an Intel 1.83GHz CPU. In the case of pseudonym certificates, steps 2.a.3 and 2.b.5, i.e. key generation, will be dominant if RSA is used; on average 377ms for 1024 bits and 4110 ms for 2048 bits. For the anonymous credential based protocols, issueCred and $\text{showCred}/\text{prove}$ are dominant (steps 4.a.6, 4.a.7, 4.b.2 and optionally 4.b.3.b). showCred will require less than 400ms and less than 1,500ms for 1024 and 2048 bits respectively, while issuing lasts less than 600 ms and 2000 ms. Happily, obtaining (one or more) permit certificates will

	Trivial	Fixed nym intervals	User defined nyms	Anonymous credentials
$F1$ - # Tickets	✓	✓	✓	✓
$F2$ - Subscription	✓	✓	✓	✓
$F3$ - Pricing	✓	✓	✓	✓
$F4$ - Deanonimization	✓	✓ (3)		✓(5)
$F5$ - Ban	—	✓	✓+ ticket revocability	✓(2)
$P1$ - Ticket anonymity	T knows user id	If no collusion of E, T, PS, G . T knows permit atts.		✓
$P2$ - No user profiles	T can link everything.	Linkability during limited, fixed period.	Limited linkability (4).	✓(1).
$P3$ - Blacklist unidentifiability	—	If no collusion PS, G .		only G can identify U .

- (1): If the user is blacklisted, G can collude with one or more T s.
(2): Ticket revocability is possible at the cost of increased linkabilites.
(3): Interaction of J with E, T, PS and G required.
(5): Interaction of J with E, T and G required.
(4): Depends on number of events during a time interval, the time period tickets can be ordered and the number of events the user wants to attend.

Table 10: Comparison of the three approaches

usually be spread in time.

11 Related Work

This section explores other related research works in the area of ticketing system. Each of the following researches propose distinct solutions for their ticketing system, although, their emphasis varied from proposing a framework into specifically addressing selected ticketing application issues.

Fujimura and Nakajima[9] propose a generalised framework for a general-purpose ticketing system. They argue that, implementing a ticket processing infrastructure, such as ticket issuing, ticket wallets and ticket validation systems, for individual ticketing applications is very expensive. Thus they propose a *ticket description method* and *processing architecture* framework for a general-purpose ticketing system using a set of common ticket processing components. In their model, the authors assume *similarity* between ticketing applications and digital cash [17] systems. This assumption influences their model requirement definition. As a result 77% of their framework requirement inherited from digital cash requirements (like *security, anonymity, transferability, divisibility, etc.*).

In their model, a user can obtain a ticket with *promises* from a ticket issuer. A promise can be a set of rights for the bearer of the ticket, like ‘let the bearer to enter a football match’ or ‘a flight between Brussels and Tokyo’. Though, the

paper falls short in describing the anonymity measures that needs to be taken to anonymize the user from other organization. In all expression of [9] model, the identity of the user is revealed to the issuer. Although, it is not clear from the paper how the identity of the user is expressed in the ticket itself and what identifying information the verifying organization uses for ticket validation.

In contrast with our protocol, [9] conception of anonymity seems reluctant and sometimes misleading. That can be clearly shown in their classification of ‘ticketing applications’ like plane tickets, software license tickets, gate card and driver license as ticketing applications that does not need anonymity at all. On the other hand [9] classifies ‘Transportation pass’ as a ticketing application that needs anonymity. However, in reality (i.e. at list from privacy perspective) there is no distinction between ticketing systems of ‘transportation pass’ and ‘plane ticket’. Therefore, passenger anonymity must be preserved in both ticketing applications. In comparison with [9], our protocol maintains anonymity of the user in a novel way, using pseudonym certificate and anonymous credential constructions. In all ticketing applications user anonymity is preserved as long as a certain form of dispute does not occur. As a result, user identity is free from privacy unfriendly user data collection, linking, analyzing and profiling.

Patrick et al.[16] propose a concept called *anonymous blacklisting* where a service provider (SP) is capable of revoking the right of misbehaving users without the knowledge of user identity. Anonymous credential protocols like the one in [5] [4] commonly used a Trusted Third Party (TTP) to selectively deanonymize (or link) misbehaving users at will. However, Patrick et al. strongly argued that deanonymizing a user with the help of a TTP at any time is a strong measure for misbehaving user for a privacy-preserving system. The reason is; strictly specifying the scope of deanonymization-reasons into a well-defined misbehaviour is not possible in some applications, as it can be clearly defined in some application like e-cash[7] as double-spending detection. On top of that, some applications might not necessarily need deanonymization to discourage misbehaving users, they can simply blacklist the user’s pseudonym, to block a user without revealing that user’s identity. Thus, the authors propose a scheme where user-misbehaviour is judged *subjectively* and blacklisted by each individual SP without the need for a TTP and its deanonymization process. Even if [16] subjective blacklisting by each SP reduces the size of a black list in comparison with the usual centralised blacklisting approach, it can empower a SP to arbitrarily discriminate (or freely blacklist) among its ticket users. Let say club X and Y plays on club-X stadium, X will be a SP who sells tickets for both X and Y supporters. Even though X sells anonymous tickets X still can collectively identify Y supporters through observation at the entrance, seat preference or other leading clues and collect their pseudonyms. As totally authoritative SP X can put Y supporters on its blacklist, to prevent them from attending the match at a later time. In comparison, our protocol does not allow SPs to blacklist users or to maintain their own blacklist. As discussed previously, in our protocol the blacklist is centrally managed by a *trusted* government instance and passed to SPs. Moreover, arbitrary user blacklisting is forbidden, since judicial confirmation is a must before adding a user into a blacklist. Therefore,

with our ticketing system SPs cannot identify and discriminate among ticket users as it is possible in [16] construction.

Heydt-Benjamin et al.[11] propose a hybrid electronic ticketing system which uses passive RFID transponders and higher powered computing devices such as smart phones or PDAs. Their hybrid ticketing system framework takes the advantage of e-cash, anonymous credentials and proxy re-encryption[10] to alleviate the concern of privacy in public transportation ticketing systems. The authors motivated that the ticketing system's migration from contact based (magnet-stripe) into contactless technologies in public transportation as a source of great risk for privacy and security. They also discourage maintaining passenger databases and linking it with payment information by transit authority. Given that same database which is considered as an *asset* by the transit authority can grow into a *liability*, since a data-breach can create a loose of money and reputation. As a result the authors propose a hybrid ticketing protocol framework where the movement of users can not be tracked by the transit authority or by an adversary. The authors also address various related concepts like ticket token resource constraint, ticket cloning detection and ticket revocation.

The use of electronic ticket is also considered in mobile user and mobile communication researches with the intent to buying services from service providers instead of products. In this respect Buttyan and Hubaux [3] propose a model for ticket based service access and also classify ticket types and ticket acquisition models. The author argued that the use of tickets for accessing services is advantageous since it provides flexibility, scalability and privacy. With such argument [3] assumes a ticket is inherently privacy friendly. In contrast, our protocol construction does not consider the ticket by itself as a privacy-preserving means to access services in mobile communication environment or any other application.

A related research by Patel and Crowcroft [13] on mobile users, also propose tickets as a means to access remote resources. The authors argued that the uses of tickets are promising for future 'homeless-services' which is attached to neither a particular location nor identities. Since these applications only need to know whether the user requesting the service has been authorized to do so or not. In other words, user identity is not mandatory. As [3], [13] also in general believed that tickets are privacy friendly by their nature. As we try to illustrate in our trivial ticketing protocol, such assumption can greatly lead to user privacy degradation and make it easy for others to gather, link and profile users.

In summery [9] ticketing framework, [16] anonymous blacklisting, [11] hybrid electronic ticketing and [3] [13] tickets for mobile users and communication are valuable contributions to built-up the future ticketing system. However, except [11] all the other works in the area fall short in properly addressing user privacy. A misconstruction and false generalisation of privacy capabilities advocated in [9], [3] and [13]. In comparison, our privacy-preserving ticketing protocol contains a novel construction to maintain user anonymity in all instances and avoids uncontrolled arbitrary deanonymization.

12 Conclusions and Future Work

Two privacy preserving ticketing systems were proposed; one based on pseudonym certificates and one on anonymous credentials. We showed that it is possible to offer the user a high degree of privacy, while the other requirements remain fulfilled. Still the privacy unfriendly eID card is used as bootstrap.

A prototype implementation will be made, using an applet for registration and ticket ordering. Entering the event can be done using a bar code reader. The influence of mix networks on the overall performance must be examined.

References

- [1] N. Asokan, Els Van Herreweghen, and Michael Steiner. Towards a framework for handling disputes in payment systems. Technical Report RZ 2996, 1998.
- [2] S. Brands. A technical overview of digital credentials, 1999.
- [3] L. Buttyn and J. P. Hubaux. Accountable anonymous access to services in mobile communication systems. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 1999.
- [4] J. Camenisch and E.V. Herreweghen. Design and implementation of the idemix anonymous credential system. In *ACM Computer and Communication Security*. 2002.
- [5] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 93–118, London, UK, 2001. Springer-Verlag.
- [6] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
- [7] A. Fiat D. Chaum and M. Naor. Untraceable electronic cash. In *LNCS: In CRYPTO 88*, volume 403. Springer Verlag, 1990.
- [8] I. Damgard, T. Pedersen, and B. Pfitzmann. Statistical secrecy and multi-bit commitments, 1996.
- [9] K. Fujimura and Y. Nakajima. General-purpose digital ticket framework. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 177–186, 1998.
- [10] M. Green G. Ateniese, K. Fu and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *In: Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS)*, 2005.

- [11] Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for public transportation. In *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*. Springer, 2006.
- [12] Matt Hooks and Jadrian Miles. Onion routing and online anonymity. *CS182S*, 2006.
- [13] B. Patel and J. Crowcroft. Ticket based service access for the mobile user. In *In Proceedings of Mobicom'97*, 1997.
- [14] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.
- [15] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 1997. IEEE Computer Society.
- [16] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81. ACM, 2007.
- [17] P. Wayner. *Digital Cash: Commerce on the Net*. Academic Press, 1996.