SERVICE AND TIMEFRAME DEPENDENT UNLINKABLE ONE-TIME PSEUDONYMS

Kristof Verslype, Bart De Decker  
Katholieke Universiteit Leuven,  
Department of Computer Science,  
Celestijnenlaan 200A, 3001 Heverlee, Belgium  
firstname.lastname@cs.kuleuven.be

Abstract: A solution is presented to allow a service provider to limit the number of times per timeframe that a user can access each single service, while maintaining complete unlinkability of different visits by that user. Since the solution is built upon existing building blocks such as anonymous credentials, it is extremely flexible.

1 Introduction

Privacy on the Internet is gaining importance since the user does not have control over the released personal data and since digital data can be processed and spread very easily. Moreover, the certificates that are used in practice contain several attributes, such as a user identifier. Combined with collected transactional data, this allows the service provider to compose an extended profile linked to the user’s identity.

Anonymous credentials try to overcome these shortcomings. They allow for the selective disclosure of properties of credential attributes. For instance “\(\text{age} > 20\)” could be proven, while the user’s zip code and date of birth remain hidden although they are included in the credential. Secondly, anonymous credential systems can offer unlinkability of different credential usages, making different visits of a user to a service provider unlinkable.

During issuance, a global show limit can be set on these credentials, which limits the number of times the user can use his credential and this limit is independent of the service that is used.

More versatile limits have been proposed, allowing a service provider to set an access limit towards a specific service or allowing for a global show limit per timeframe. This paper combines both approaches and guarantees unlinkability of accesses by the same user. The service provider can dynamically set the number of times the owner of a valid credential can access a service. This limit can optionally apply to a single timeframe. Both the access limit and the timeframe duration can be chosen and dynamically adjusted by the service provider. Additional restrictions might apply depending on the user’s properties, thus further limiting the user’s access rights.

For instance, users with a ‘golden membership’ credential can access a specific service an unlimited number of times, and ‘regular members’ only 50 times a month. People with a credential without membership privileges can access the service only a very limited number of times, depending on the personal properties they disclose. If they prove being younger than 18, they can access the service once a month, but only in weekends and if they are older than 18 and disclose their exact age, they can access it twice a week.

Section 2 gives an overview of the related work. Section 3 describes the requirements, used notations and assumptions. The required building blocks are presented in section 4. A new building block, the extended provable nym generator, is presented in section 5. Those blocks are combined in our solution in section 6. We conclude in section 7.

2 Related Work

Solutions to the problem of restricting the number of times a user can access a service provided by a service provider while the different accesses by the same user are unlinkable have been presented. Two of these
solutions (Teranishi et al., 2004), (Nguyen and Safavi-naini, 2005) are based on group signatures. As a consequence, the user can only prove being a member of the group. Another system (Camensich et al., 2006b) was specifically developed for e-Cash and therefore, did not need personal attribute disclosure properties. Those systems lack the flexibility provided by anonymous credential systems. Solutions to limit the number of times a token can be used in a single time interval were proposed (Damgard et al., 2006; Camenisch et al., 2006a).

This paper combines both approaches in a flexible way as explained in the introduction and builds upon anonymous credential systems. Most of the previous schemes allow for deanonymization in case of abuse. This is omitted in the presented solution since the underlying anonymous credential system already provides this functionality.

3 Requirements, Notation and Assumptions

The requirements are now summed up.

**Dynamic service access restriction.** The service provider can limit the number of times a user can access each service per timeframe. The limits and timeframe size can dynamically change and can depend upon the disclosed data during the credential show.

**User anonymity.** A service access by a user cannot be linked to (1) an access to another service, (2) an access to the same service in the same or (3) a different timeframe or (4) to the user’s identity.

**Flexibility.** The flexibility of anonymous credentials must be preserved.

Dishonest entities are represented with a tilde (e.g. $\tilde{P}$). All operations are either in a subgroup of $G_p$, where $G_p$ has order $p$, or in $\mathbb{Z}_q$. $p$ and $q$ are prime. The notation $P_1 \equiv P_2 : (y_1; y_2) \leftarrow$ protocol $(x_0; x_1; x_2)$ is used throughout the paper and represents a protocol run between $P_1$ and $P_2$. $x_1$ and $x_2$ are the inputs only known by $P_1$ and $P_2$ respectively. $x_0$ is the input known by both $P_1$ and $P_2$ and provided by $P_1$, $P_2$ or both. $y_1$ defines the output for $P_1$, $y_2$ for $P_2$.

Two number theoretic assumptions are relevant:

**Discrete log (DL) Assumption.** Let $g$ be a generator for a finite cyclic group $G$, let $x \in \mathbb{Z}_q$ and $y = g^x$. Finding $x$ when only $g$ and $y$ are known is intractable.

**Representation (R) Assumption.** Brands, 2000). Finding a representation w.r.t $(g_1, g_2, \ldots, g_m)$ in group $G$ for a given value $y$ is as difficult as solving the DL problem.

4 Building blocks

This section elaborates on existing cryptographic building blocks.

4.1 Proofs of Knowledge

A proof of knowledge (PK) (Bellare and Goldreich, 1992) is a protocol in which a prover $P$ convinces a verifier $V$ that it possesses a secret. More formally, the following relation is defined: $R = \{(y, x) : y \in L, x \in W(y)\}$, where $L$ is a language in NP, $x$ the secret, $W(y)$ the set of possible witnesses for a public value $y$ that should be accepted in the proof and a PK must have the following properties:

**Completeness.** If $(y, x) \in R$ then $Pr[P(y, x) \equiv V(y) \rightarrow accept] = 1$.

**Validity.** There exist a polynomial-time extractor $K$ having oracle access to a potentially cheating prover $P$. $K$’s success probability in extracting $x$ is at least as high as the probability that $P$ succeeds in convincing the verifier $V$. More formally; $Pr[K(y) \in R(y)] \geq Pr[P(y) \equiv V(y) \rightarrow accept] - \kappa(y)$, where $\kappa(y)$ denotes the probability that verifier $V$ might accept $y$, even though prover $P$ does not know a witness $x$.

A zero-knowledge proof of knowledge (ZKPK) additionally has the following property:

**Zero-knowledge.** No other data is revealed by $P$. Formally, there exists a simulator $\Sigma$ such that the following two probability ensembles are indistinguishable: $\{P(y, x) \equiv V(y) \rightarrow \}_y \in L$ and $\{S(y) \rightarrow \}_y \in L$.

The notation of a PK is:

$$P \rightarrow V : \{\emptyset, proof\} \leftarrow PK\{(x_1, \ldots, x_m) : properties(x_1, \ldots, x_m)\}.$$  

where $x_1, \ldots, x_m$ are the hidden values about which properties are proved. The following properties that can be proven are relevant for this paper: (1) knowledge of a discrete logarithm modulo a prime (Schnorr, 1991), (2) a committed value lies in an integer interval$^1$ (Boudot, 2000), (3) conjunctions and disjunction of the previous (Cramer et al., 1994). (4) More generally, proving knowledge of representation and equality of secrets is possible, i.e. some secrets used in different operands of a conjunction can be equal. This is denoted as $PK\{(x_1, \ldots, x_n) : \land_{i=1}^{n} \Pi_{j \in I} g_{ij}^{x_{ij}} = y_i\}$ where

$^1$The scheme was based on composite moduli, while this paper uses DL based commitments. However, the same is possible using the DL commitments. What is actually done to prove that $a < x < b$ is proving that $x - a$ and $b - x$ are positive, which is done by proving that $x - a$ and $b - x$ are both the sum of four squares.
x₁, ..., xₙ are the different secrets, where n is the number of (conjunctive) operands, Jᵢ ⊆ {1, ..., l} with l the number of bases and eᵢ is the index of the secret used in yᵢ w.r.t. base gᵢ. For such proof of representations and equality of secrets, the following properties hold: (a) They are sigma protocols; they have a commitment²-challenge-response structure. (b) For each operand, one multi-base exponentiation by P is required, and one by V. (c) For each operand, one commitment (to a random value) is sent by P. (d) Per proof, one challenge is sent by V. (e) For each secret, one response is sent by P. If a generator g of order q is used, which generates a subgroup in G_p, with p and q prime, a commitment has size |q|, a challenge and a response have size |p|.

### 4.2 Commitment Schemes

A commitment scheme (Pedersen, 1992; Damgard et al., 1996) allows an entity to commit to a set of values, while keeping these secret. The commitment hides the values towards the verifier, but allows the creator to prove properties of the committed values. The following protocols are relevant in this paper:

- **P : (C, O) ← commit(x₁, ..., xₘ).** A commitment to one or more attributes xᵢ is created. A secret random value t of sufficient length is chosen in order to make two commitments to the same values unlinkable. The opening info O = (x₁, ..., xₘ, t) is required in the protocol below.

- **P → V : (∅, proof) ← PK{(O) : properties(C)}.** P proves properties of values committed in commitment C to V in a ZKP.

A commitment has the following properties:

#### Hiding. The commitment hides all information about the committed values.

#### Binding. The values are fixed when the commitment is created and cannot be changed afterwards.

#### Semantically secure. Even if the committed values are known and equal, an attacker cannot distinguish between two commitments to the same values.

A commitment to m values x₁, ..., xₘ with randomizer t can be implemented as y ← g₁^x₁g₂^x₂ ... gₘ^xₘh^t with the group description and all bases gᵢ and h publicly known. Commitment schemes based on the DL assumption or the RSA assumption exist.

²In this subsection, the term 'commitment' is used in a broad sense. In the sequel of this paper, the term commitment refers to what is discussed in the next subsection.

### 4.3 Joint Secure Random Number Generation

The joint secure random number generation (Camenisch and Lysyanskaya, 2001) between U and O results in a value only known to U, but O is reassured that it is random and of sufficient length. The relevant method and protocol for this paper are:

- **CA : S ← genRandGenParams().** Generates the basic input parameters for the protocol below.

- **U ⇔ O : (r, t, C) ← agreeSecureRand(S; ∅; ∅; ∅).** Agree on a secure random value r, which will be committed in C. t is the randomizing factor for the commitment. (r, t) is U’s opening info.

### 4.4 Anonymous Credentials

Anonymous credential systems (Chaum, 1985; Camenisch and Lysyanskaya, 2001; Camenisch and Herrewegen, 2002; Brands, 1999) allow for anonymous yet accountable transactions between users and organizations. They allow for selective disclosure; the user is able to reveal a limited set of properties about the attributes embedded in the credential. In the sequel of this paper, Idemix credentials are assumed. Multiple Idemix credential shows are unlinkable if no uniquely identifying attribute data are revealed.

The relevant simplified protocols that apply to anonymous credentials are:

- **U ⇔ I : (cred; ∅) ← issueCred(certᵢ, comsᵢ;_attsᵢ; opensᵢ; SKᵢ).** I issues to U a credential with attributes attsᵢ. Additionally, values committed by U (comsᵢ) can be included as attributes in the credential. This requires the corresponding opening info (opensᵢ).

- **U ⇔ V : (∅; proof) ← showCred(comsᵢ, propsᵢ; credᵢ, opensᵢ; ∅; msgᵢ).** U proves to V the possession of a valid credential cred. U can selectively disclose credential attributes or properties thereof (described in propsᵢ). These properties can involve a set of committed values (comsᵢ) with corresponding opening info (opensᵢ). U may decide to sign a message msgᵢ with his credential, creating a provable link between the proof and the message.

Additionally, proofs resulting from a showCred() protocol can be deanonimizable by a predetermined trusted third party. Anonymous credentials can further be issued to be shown only once, a predetermined number of times or an unlimited number of times in total, independent of the services that are contacted. However, these features are not used in this paper.
5 An Extended Provable Nym Generator (EPNG)

5.1 Definition

An extended provable nym generator (EPNG) is a function \( nym \leftarrow f(id, i, t, f) \) fulfilling the five properties listed below. In the next section, each service has its own EPNG, \( id \in \mathbb{Z}_q \) is a secret user identifier, \( tf \in \mathbb{Z}_q \) the current timeframe and \( i \in \mathbb{Z}_q \) is a value, between one and the access limit, that has not yet been used during timeframe \( tf \) by that user. Output \( nym \) is a one-time pseudonym computed by the user and sent to the service provider.

One-wayness. It is infeasible to derive from a specific \( nym \) and EPNG \( f \) the corresponding \( id, i \) or \( tf \).

Deterministic. Applying the same EPNG multiple times on the same \( (id, i, t, f) \) tuple results in the same output \( nym \).

Semi-collision free. (1) For the same \( id \) and \( f \), it is infeasible to find an \( (i, tf) \) and \( (i', tf') \) with \( (i, tf) \neq (i', tf') \) such that \( f(id, i, tf) = f(id, i', tf') \). (2) For two randomly chosen \( id \) and \( id' \) and for \( k \) and \( l \) small compared to \( id \) and \( id' \), it is infeasible to find \( i, i' \in [1, k] \), \( t, t' \in [1, l] \) such that \( f(id, i, tf) = f(id', i', tf') \).

Unlinkability. (1) \( nym_1 = f(id, i, tf) \) and \( nym_2 = f(id, i', tf') \) with \( (i, tf) \neq (i', tf') \) are unlinkable. (2) \( nym_1 = f_1(id, i, tf) \) and \( nym_2 = f_2(id, i, tf) \) where \( f_1 \neq f_2 \) are unlinkable.

Provable. If \( nym = f(id, i, tf) \), \( id, i \) and \( tf \) are known, one can prove (1) that \( nym \) is well formed (i.e., the result of applying the EPNG on \( id, i \) and \( tf \)) and (2) properties about \( id, i \) and \( tf \).

Two functions/protocols are important.

- \( f \leftarrow \text{genEPNG}() \). The generation of an EPNG.
- \( P \vdash V : (\text{com}_{id, \text{com}_t}, \text{open}_{id, \text{open}_t}; \text{com}_{id, \text{com}_t, \text{proo} \text{f}}) \leftarrow \text{proveEPNG}(f, \text{nym}, \text{id}, \text{i}, t, f; \varnothing) \). \( P \) generates commitments to \( id \) (\( \text{com}_{id} \)) and \( i \) and \( tf \) (\( \text{com}_{id, tf} \)), and convinces \( V \) that these committed values were used as input for EPNG \( f \) to generate \( nym \). This enables \( P \) to prove properties over these input values using the corresponding opening info \( (\text{open}_{id, \text{open}_t}) \). \( \text{Proof} \) is a proof for \( V \) that the commitments contain the proper values w.r.t. \( nym \).

5.2 A Simple Provable Nym Generator

The Provable Nym Generator (PNG) is an EPNG without the timeframe parameter. A concrete PNG can be defined as

\[
\text{nym} \leftarrow g^{1/(id+i)} \tag{1}
\]

in a group \( G_p \) of prime order \( p \), where \( g \) generates a subgroup of prime order \( q \). This concrete PNG is shown in table 1. Its security is based on the DL assumption. The security of the first four properties can easily be derived from the proofs given in the next subsection, where a concrete EPNG is given. Since the provePNG() implementation approach is different (for efficiency reasons) from the proveEPNG(), the provability property is argued below.

After creating the two commitments (A), \( P \) proves that he knows an \( (id + i) \) such that \( \text{nym} \leftarrow g^{1/(id+i)} \) (B). The \( s \) value can only be calculated if \( (id + i) \) is known. The resulting \( c, s \) and \( t \) values are used to convince \( V \) that the committed values are indeed \( id \) and \( i \). \( h \) generates a subgroup of order \( q \) in \( G_p \) and \( \log_{g}(h) \) is unknown. It is indeed a ZKP.

Completeness. This can easily be seen by expanding the two comparisons verified by \( V \).

Validity. The probability that \( \hat{P} \) passes is \( 1/(q, k) \) since he has to guess the right values \( id \in \mathbb{Z}_q \) and \( i \in [1, k] \). If an extractor \( K \) runs subprotocols A and B with challenge \( c = c_i \in \mathbb{Z}_q \), resets the machine and runs A and B again with challenge \( c = c_i \in \mathbb{Z}_q \), he can extract \( id + i \), since the same \( t \) will be used. Guessing \( i \) then results in the correct \( id \). \( K \) can thus extract the \( id \) and \( i \) with a probability of at least \( 1/k \).

Zero-knowledge. A simulator \( S \) could have constructed the proof as follows: (1.1) choose \( s \in \mathbb{Z}_r \)

<table>
<thead>
<tr>
<th>P</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\text{com}_{id, \text{com}<em>t}, \text{open}</em>{id, \text{open}<em>t}; \text{com}</em>{id, \text{com}_t, \text{proof}}) \leftarrow \text{proveEPNG}(f, \text{nym}; \text{id, i, t, f}; \varnothing) )</td>
<td>( (G_p, g, h, p, q) \leftarrow f )</td>
</tr>
<tr>
<td>( c \in \mathbb{Z}_q )</td>
<td>( (G_p, g, h, p, q) \leftarrow f )</td>
</tr>
<tr>
<td>( \text{com}_{id} \leftarrow g^{id} h^c )</td>
<td>( \text{com}_{id, \text{com}_t} )</td>
</tr>
<tr>
<td>( t \in \mathbb{Z}_q )</td>
<td>( f )</td>
</tr>
<tr>
<td>( T \leftarrow \text{nym} )</td>
<td>( \text{nym} )</td>
</tr>
<tr>
<td>( s \leftarrow c.(id + i) + t )</td>
<td>( g^c.T \equiv \text{nym}^s )</td>
</tr>
<tr>
<td>( \text{com}_{id} \leftarrow (id, c) )</td>
<td>( \text{com}_{id, \text{com}_t} )</td>
</tr>
<tr>
<td>( \text{open}_{id} \leftarrow (id, c) )</td>
<td>( \text{proof} \leftarrow (T, c, s, w) )</td>
</tr>
<tr>
<td>( \text{open}_{id} \leftarrow (id, c) )</td>
<td>( \text{com}_{id, \text{com}_t} )</td>
</tr>
</tbody>
</table>

Table 1: The provePNG() protocol for \( nym \leftarrow g^{1/(id+i)} \).
The presented EPNG is defined as $\text{proveEPNG}(\cdot)$ and requires six modular exponentiations by $P$, for $V$ and the communication cost is $6.|p| + 5.|q|$. Computing $nym$ by $P$ requires one multi-base modular exponentiation by $P$, sending it to $V$ adds a communication cost of $|p|$. The efficiency of the solution presented in section 6 will also depend on other factors such as the efficiency of proving properties about the committed values. The same holds for the proveEPNG() function.

### 5.3 A Concrete Extended Provable Nym Generator

The presented EPNG is defined as

$$nym = g_1^{1/(id+i)} g_2^{1/(id+g)}$$

in a group $G_p$ with known prime order $p$. $g_1$ and $g_2$ each generate a subgroup of prime order $q$, $log_{g_1}(g_2)$ is unknown. This also implies that $log_{g_2}(g_1)$ is unknown, since $q$ is known. Indeed, finding the inverse of an element in $Z_q$ is easy using the extended Euclidean algorithm.

The four first properties are fulfilled if the group parameter $q$ and input $id$ are sufficiently large and if for each two EPNGs with respectively $(g_1, g_2)$ and $(g_1', g_2')$ as bases, $log_{g_2}(g_1')$ or $log_{g_2}(g_1)$ is unknown:

**One-wayness** follows from the R assumption if $(id, t, f)$ is chosen out of a sufficiently large domain to avoid that all possible inputs are tested for a specific $nym$. Since $i$ and $tf$ are relatively small (see later), the domain of $id$ needs to be sufficiently large.

**Deterministic.** Follows from the deterministic underlying algebra.

**Semi-collision free.** (1) Since $log_{g_1}(g_2)$ is unknown, for a specific $id$, finding $i, t, df, tf \in Z_q$ with $(i, tf) \neq (i', tf')$ such that $g_1^{1/(id+i)} g_2^{1/(id+g)} = g_1^{1/(id+i')} g_2^{1/(id+g)}$ can be considered as two subproblems $g_1^{1/(id+i)} = g_1^{1/(id+i')}$ and $g_2^{1/(id+g)} = g_2^{1/(id+g')}$. Since each element has exactly one inverse in $Z_q$, this is impossible.

(2) If $1/n$ is considered as biggest negligible value, $n \in \mathbb{N}_0$, $i \in [1,k]$, $tf \in [1,f]$ and $k, l \in \mathbb{Z}_q$, then the size of the EPNG’s output domain must be at least $k.l.n$. This is now proven. Say, $nymset$ and $nynset$ are the sets of all possible nymes that can be generated with $id \in Z_q$ and $id' \in Z_q$ respectively and say, the output domain of the EPNG $f$ has size $k.l.n$. Both sets thus have size $k.l$. Let $nym \in \mathbb{R}$ $nymset$. Then, $Pr[nym \in nymset'] = 1/(n.k.l)$ and the probability that there is an overlap between $nymset$ and $nymset'$ – i.e. a collision arises – is thus $(k.l)/(n.k.l) = 1/n$. The output size of $f$ is at least $q$ (if $log_{g_1}(g_2) \in Z_q$). Thus, when $q$ is chosen, it must be at least $n.k.l$, where $k.l$ is an estimated maximum value.

**Unlinkable.** Due to the one-wayness it is infeasible to derive $id$ from $nym$. Let $nym_1 = g_1^{1/(id+i)} g_2^{1/(id+g)}$ and $nym_2 = g_1^{1/(id+i')} g_2^{1/(id+g')}$ with $i' \in Z_q$ and $x = i' - i$. This means that $nym_2 = nym_1 . (g_1^{1/(id+i)} / g_1^{1/(id+i')})$. Since $id$ cannot be found, different nymes using the same $id$ and $tf$ are unlinkable. Similarly, different nymes using the same $i$ cannot be linked.

Since $log_{g_1}(g_2)$ is unknown, no relations between $1/(id+i)$ and $1/(id+g')$ and thus between $i$ and $tf$ can be found.

(2) The bases $(g_1, g_2)$ and $(g_1', g_2')$ of two EPNGs must not lead to linkabilities. This can only happen if $log_{g_1}(g_1') = log_{g_2}(g_2')$ and if this value is known (DL assumption). If $nym = g_1^{1/(id+i)} g_2^{1/(id+g)}$, $nym' = (g_1')^{1/(id+i)} (g_2')^{1/(id+g)}$ and $a = log_{g_1}(g_1') = log_{g_2}(g_2')$, then $nym' = (g_1')^{1/(id+i)} (g_2')^{1/(id+g)} = (g_1^{1/(id+i)} a) g_2^{1/(id+g)} = nym a$.

**Provability.** The implementation of $P \equiv V : (com_{id, id, tf, open_id, open_id, com_{id}, com_{id}, proof}) \leftarrow \text{proveEPNG}(f, nym; id, i, t, f, \emptyset)$ is shown in table 2 and returns commitments to $id$, and to $i$ and $tf$, such that $V$ is convinced that $nym = g_1^{1/(id+i)} g_2^{1/(id+g)}$. An additional base $h$ of order $q$ is required where $log_{g_1}(h)$ and $log_{g_2}(h)$ are unknown.

The commitments to $id$, $i$ and $tf$ are created by $P$ and sent to $V$. Additionally, $P$ sends to $V$ a "helper" commitment $C$ to the values $x_1 = \frac{1}{x_1}$ and $y_1 = \frac{1}{y_1}$. In the first proof of knowledge, $P$ proves to $V$ that $nym = g_1 x_1 g_2 y_1$. In the second proof of knowledge, $P$ proves that the commitment $com_{id}$ is indeed a commitment of a value $id$ towards basis $g_1, g_2$, or stated otherwise; $P$ proves that the committed value w.r.t. $g_1$ is the same as the committed value w.r.t. $g_2$. In the third proof of knowledge, $P$ proves to $V$ that the sum of the committed values $id$ and $i$ equals the inverse in $Z_q$ of $x_1$, which is committed in $C$. In the final proof of knowledge $P$ proves to $V$ that the sum of the committed values $id$ and $tf$ equals the inverse in $Z_q$ of $y_1$, which is committed in $C$. The combination of these four proofs of representation and equality of secrets thus guarantees that the committed values $id$, $i$ and $tf$ correspond to $nym = g_1^{1/(id+i)} g_2^{1/(id+g)}$. 


5.4 Efficiency

The calculation of \(nym\) requires one multi-base modular exponentiation by \(P\), and sending it to \(V\) results in a communication cost of \(|p|\) bits.

The four proofs of knowledge in the proveEPNG() can be merged into a single one by applying conjunctions. The resulting proof can be computed more efficiently since some operands appear in multiple proofs. Secondly, the second proof can be omitted if \(P\) will prove something about \(id\) after the proveEPNG() protocol, which will always be the case. Indeed, if \(com_id = g_1^{id} = g_2^{id}\) with \(id \neq id'\), \(P\) will be unable to prove anything about the value committed w.r.t. \(g_1, g_2\). This leads to a more efficient, slightly relaxed proof which replaces the four previous ones:

\[
(\emptyset; pr) \leftarrow PK \{ (x_1, x_2, y_1, y_2, l, c, t, t_0, r_0) : nym = g_1^{x_1} g_2^{y_1} \land C = g_1^{x_2} g_2^{y_2} \land com_id \land com_gf = g_1^{x_1} g_2^{y_1} \land g_1 = (\frac{1}{n}) \land C_1 \land g_2 = (\frac{1}{n}) \land C_2 \}
\]

Since there are now three commitments in proveEPNG(), 10 secrets, 1 challenge and 5 operands, 13 modular exponentiations are required, and the total communication cost is \(8 |p| + 11 |q|\). It was argued previously that \(q\) should be enlarged to \(n.k.l\), where \(1/n\) is negligible. The next section shows that \(k\) and \(l\) is used as a maximum for access limits and for timeframe numbers and will typically be small compared to \(q\) (E.g. \(|k| \leq 16\) and \(|l| \leq 24\) bits). A typical value for \(|n|\) is 160 bits (e.g. in hash functions). The \(|q|\) for the EPNG must thus be chosen larger than the usual size of 160 bits, but not by more than 25%. The protocol requires four interactions.

Note that the number of multi-base modular exponentiations remains constant, even if the number of input parameters of proveEPNG() is increased.

6 Integration in Anonymous Credential System

This section presents and evaluates a solution for the problem stated in section 3.

6.1 Solution

The main roles are the user \(U\), a service provider \(SP\) and the registration authority \(R\). Each user can receive from \(R\) a (service independent) anonymous credential \(cred_U\), which contains a user-specific secure random number \(id\). \(R\) also issues service certificates to \(SP\) s - one for each service. Each service certificate contains an EPNG \(fs\) description, which will allow to set access limits per timeframe while guaranteeing unlinkability of accesses.

Setup registration authority. A certificate is issued by a certificate authority \(CA\) to the registration authority \(R\). It contains a freshly generated parameter set \(S_{id}\), necessary for the joint secure random number generation (see section 4.3). It is possible that \(cert_R\)
is self-signed ($R = CA$).

**Registration of service.** $SP$ registers a new service. Therefore, $SP$ provides $R$ with the necessary service description $desc$, (i.e., service provider name, service name, functionality and potentially the access limit policy). $R$ issues to $SP$ a service certificate $cert$, containing $SP$’s public key, $desc$, and a new extended provable nym generator $f_x ← genEPNG()$.

**Issue user credential.** $U$ receives a credential that can be used for accessing services with a $k$-show per timeframe limit. Typically, this phase will be preceded by a mutual authentication phase. First, $R$ checks in a register whether $U$ has already been registered. If not, a new random value ($id$) to be included in the credential is agreed using the agreeSecureRand() protocol with $S_d$ as input. This results in a commitment $C$ to this random value, and opening info $O$. Finally, $R$ issues a new anonymous credential which contains the committed random number and updates the register.

\[
U ← R: (\text{cred}_{i}, \text{register}) ← \text{issueUserCredential}(\text{certs}, \text{atts}, \text{coms}, id_{i}, \text{open}; SK_{R}; \text{register})
\]

1. $U ← R: \text{if (hasValidCred(id_{i}, \text{register})) abort}$
2. $U ← R: (O, C, C) ← \text{agreeSecureRand}(\text{cert}_{R}, S_{d}; O; \emptyset)$
3. $U ← R: (\text{cred}_{i}; O) ← \text{issueCredential}\{\text{coms} \cup C\}, \text{atts}, \text{cert}_{R}; \{\text{open}; O\}; SK_{R}$
4. $U; R: \text{return (\text{cred}_{i}; \{\text{register} \cup (\text{id}_{i}, C)\})}$

**Accessing a service with $k$-limit in timeframe $tf$.** $U$ wants to access a service. First, $SP$ authenticates towards $U$ using the correct service certificate $cert$, and corresponding $SK$. Then, $U$ selects an $i \in [1, . . . , k]$ that has not yet been used during timeframe $tf$ for that specific service. The single-use nym $nym ← cert_{i}, f_{x}(\text{cred}_{i}, \text{id}_{i}, \text{tf}, \text{tf})$ is sent to $SP$. The latter checks in the service’s history whether that nym has already been used. If not, $U$ proves that the $nym$ has been correctly formed (i.e., is the result of applying the EPNG to $id$, $i$ and $tf$, $0 < i \leq k \land \text{id} \in \text{cred}_{i}$ and the current timeframe has been used, without revealing $id$ or $i$. In addition, personal properties $props$ (of attributes embedded in the credential) can be proven or a message can be signed during the credential show. Finally, $SP$ updates the service’s history $history$, with the new $nym$ and $U$ updates his list of used $i$ for that service during timeframe $tf$.

### 6.2 Evaluation

**Service access restriction.** The value $id$ is only known to $U$ and both $R$ and $SP$ (which has to trust $R$) are ensured that it is random, sufficiently long, and embedded in the user’s credential as a result of the properties of joint random number generation and the possibility to include committed values in anonymous credentials. If the $id$-domain is sufficiently large, the probability that two users have the same $id$ is negligible. The verification whether $id \in \text{register}$ during the user credential issuance guarantees that each user has no more than one $id$.

The EPNG’s semi-collision free property 1 guarantees that the user is able to generate $k$ different nyms for a single timeframe and the determinability guarantees that the user is unable to generate more than $k$ nyms. The probability that two users have the same single-use nym for the same service is negligible due to the EPNG’s semi-collision free property 2.

**User anonymity.** The EPNG’s unlinkable property guarantees that two nyms of the same user for the same or for other services during the same or during different timeframes cannot be linked. Evidently, the anonymity does not hold if identifying data were released as part of the anonymous credential show protocol. The $R$ is trusted not to introduce and disclose relationships between different EPNGs.

**Flexibility.** All anonymous credentials’ functionality is trivially preserved (e.g., conditional deanonymization). Based on the disclosed personal properties, $SP$ can define 1) whether and when users are given access 2) the timeframe duration and access limit for that user type. This combination allows for non-adjacent timeframes. For instance, a timeframe might last one week, but only in weekends non-members can access the service. Timeframe size and limit can change dynamically. The proofs can be made non-interactive (Blum et al., 1988), giving them proof value to parties other than $V$.

**Efficiency.** The efficiency of the access service protocol, which is executed most frequently, is now considered. The EPNG in section 5.3 requires 13 multi-base
modular exponentiations (which can be computed almost as efficiently as single-base ones). Proving that \( i \) lies in the proper interval, is done in 'less than 20 modular exponentiations' (Boudot, 2000) and proving equality of \( tf \) with a given timeframe is done in two modular exponentiations. For small \( k \), it might be more efficient to prove that \( i = 1 \lor \ldots \lor i = k \).

Instead of doing a range proof for \( i \), its exact value can be disclosed, since the user can randomly choose an unused \( i \). This is more efficient. However, the service provider then knows that two different nyms in the same timeframe, using the same \( i \) cannot originate from the same user. This allows the service provider to get a more exact (higher) lower threshold \( k \). If \( k = 1 \) or \( tf = \infty \), the EPNG is reduced to a PNG \( nym = g_2^{1/(id+1)} \) or \( nym = g_1^{1/(id+1)} \), requiring 10 modular exponentiations. If \( k = 1 \) and \( tf = \infty \), function \( nym \leftarrow g_1^{id} \) can be used, of which the correctness can be proven in two modular exponentiations. Finally, the performance depends on the anonymous credential system and the properties that are proved.

7 Conclusions

This paper presents a solution to dynamically restrict the number of times a user can access a service during a single timeframe. The solution is built on anonymous credential systems, making it extremely flexible. Both the size of the timeframes and the access limit may vary according to the service policy and the user’s properties that were disclosed.

Future work includes usage of an EPNG to set sticky policies by the credential issuer to credentials, while hiding the policies for the verifier.

Acknowledgements. This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven and the IWT-SBO project (ADAPID) "Advanced Applications for Electronic Identity Cards in Flanders".

REFERENCES


Camenisch, J. and Herreweghen, E. V. (2002). Design and implementation of the idemix anonymous credential system.


