

Anonymous Updating of Credentials

Liesje Demuyne

Bart De Decker

Report CW 430, December 2005



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Anonymous Updating of Credentials

*Liesje Demuynck**

Bart De Decker

Report CW430, December 2005

Department of Computer Science, K.U.Leuven

Abstract

In this paper, a high-level overview of credential systems is presented. Particular attention is paid to the two most elaborated systems to date: the credential system by Brands on the one hand and the CL system of Camenisch and Lysyanskaya on the other hand. Both systems are described in detail and a comparison is made based on their cryptographic building blocks, linkability properties and performance measures. It is then shown how the protocols in both systems can be extended to enable the anonymous updating of credentials. The resulting systems are evaluated using a case study implementing a privacy-friendly loyalty card.

Keywords : Privacy, Anonymous credential systems.

CR Subject Classification : E.3 [Data]: Data Encryption – Public key cryptosystems.

*Research Assistant of the Research Foundation - Flanders (FWO - Vlaanderen)

1 Introduction

As new communication technologies arise, large amounts of data are made globally accessible without additional costs. The gathering of personal information has become extremely easy. Hence, adequate measures should be taken to protect an individual's privacy. One such measure is to allow an individual to control the dissemination of his personal information. For this purpose, digital credentials can be used. As with their real-life counterparts (e.g. passports and driver's licenses), digital credentials contain a number of attributes, validated by an authorized entity during an issue protocol. An individual can show his credential to any verifier. However, different credentials of the same user cannot be linked to each other, nor can they be linked to their owner. Moreover, the attributes encoded into the credential can be selectively disclosed. A passport credential owner can, for example, prove that the nationality attribute encoded into his passport belongs to a European country, without revealing anything else about any other attribute encoded into this credential.

Although different uses of the same credential are unlinkable, the use of unique or rare attributes can introduce linkabilities between credentials. This is particularly true when a credential must be updated. An update of a credential is actually a special type of issue protocol. In this protocol, a new credential is created of which the attributes are based on the attributes of an old credential. In order to limit linkabilities, the new credential should be unlinkable towards the old one. Next to this, the actual values of some attributes in both credentials should not be revealed to the issuer. Rather, it is sufficient for the issuer to know the relation between these attributes in the old and the new credential.

In this paper we review the two most elaborated credential systems to date. These are the system by Brands [2] on the one hand, and the system of Camenisch and Lysyanskaya [3] (CL system) on the other hand. We point out their major differences and propose suitable update protocols for both systems. We then define a case study to compare both systems with respect to their updating protocols.

The remainder of this paper is structured as follows. In section 2 we give an overview of anonymous credential systems. Particular attention is paid to the two most elaborated systems to date: Brands' system and the CL system. In section 3, we extend both systems with suitable anonymous update protocols. Using these protocols, an old attribute value can be replaced with a new one and a given value can be added to an attribute. Next, we develop a loyalty card application based on anonymous credentials in section 4. Based on this application, both systems are compared with respect to their updating protocols. Finally, we end with the major conclusions of our work in section 5.

2 Anonymous credential systems: an overview

In this section we give an overview of existing credential systems and their most important properties. We start by giving a historical overview of more primitive

credential systems, this then leads to the detailed description of both Brands' and the CL system.

2.1 Pseudonym systems

The concept of anonymous credentials was put forward by David Chaum [7] in 1985. Chaum described a scenario in which multiple users have pseudonyms with organizations. The users transfer credentials from and to organizations based on these pseudonyms. This scenario will be the general scenario in all credential systems.

Chaum and Evertse [8] proposed a credential system based on RSA. In their implementation, a credential does not contain any attributes. It is retrieved by a user after interaction with an issuer, and can then be shown to different verifiers without creating a link between these shows. However, showing the same credential multiple times to the same verifier will introduce a link between these shows. Also, a semi-trusted third party is needed to sign all credentials.

Damgård [11] developed a credential system which has essentially the same linkability properties as the system of Chaum et al. The credentials are of a primitive type; they do not contain any attributes and revocation is not considered. Also, a trusted third party is needed. This party's role is limited to validating the user's identity. The system is unforgeable under well-accepted intractability assumptions (i.e. the existence of pairs of claw-free functions and trapdoor one-way permutations). However, its importance lays mainly in its theoretical value, as the scheme itself is not a practical one.

Chen [10] presented a discrete-log based credential system. Her credentials do not contain any attributes. Furthermore, a credential can be used only once. When it is shown more than once using different pseudonyms, these pseudonyms can be linked. Also, a trusted authority is needed to validate the correct form of a user's pseudonym. As with Damgård's system, this authority is not needed during the actual credential transfer.

Lysyanskaya et al.[13] presented a theoretical credential system based on any one-way function. This construction makes use of general zero-knowledge proofs and therefore is not usable in practice. They also propose a practical construction for a credential system secure against the sharing of credentials. The system is based on a non-standard discrete-log assumption and does not support credential attributes. Also, a trusted authority is needed to correctly identify the users. As with the system of Chen, different shows of the same credential can be linked. An adaptation was proposed to obtain credentials which can be shown multiple times without introducing linkabilities. However, this adaptation requires the cooperation of the issuer whenever a credential is shown.

Acquisti [1] created a credential system based on Acid mixing. Credentials are issued by a MIX and are retrieved in batches. This retrieval either completes correctly or is aborted. Hence, by invoking an abortion of the retrieval protocol, denial-of-service attacks can be performed. A credential owner is only

anonymous within a certain batch. Also, a credential can only be used once without introducing linkabilities.

Finally, Verheul [14] proposed a credential system based on so-called self blindable credential certificates. An attractive property of this system is the possibility to combine different credentials into a new credential which is unlinkable towards its composing parts, giving the user a high degree of control on the dissemination of his personal information. Next, we briefly present a high level description of the ideas put forward by Verheul, together with a discussion of his work.

Setting.

A credential is of the form $(s, p, cred_{x_1}, \dots, cred_{x_n})$. Here, p is the users pseudonym and s is the corresponding secret key only known to the owner and used to prove possession of p . The $cred_{x_i}$ are then subcredentials containing statements certified by possibly different certification authorities x_i . Each subcredential $cred_{x_i}$ is a tuple of the form $(\sigma_{x_i-p}, C_{pk_{x_i}})$. Here, $\sigma_{x_i-p} = sign_{sk_{x_i}}(p)$ is a signature generated with the authorities secret key sk_{x_i} on the users pseudonym p , and $C_{pk_{x_i}} = Cert(pk_{x_i}, "x_i \text{ statement}")$ is a traditional certificate (e.g. an X.509 certificate issued by an external CA) binding the authority's public key pk_{x_i} to a specific property (e.g. a property stating that all messages signed with the corresponding secret key are pseudonyms of European citizens).

For the signing of the pseudonyms, a proofless variant of the Chaum-Pedersen signature scheme [9] is used. This special variant is constructed in a group where the decisional Diffie-Hellman problem is simple, but the Discrete Log and the computational Diffie Hellman problem are still hard. It enables anyone to transform a (message,signature)-pair (m, σ) into a new pair (m', σ') , using a publicly known blinding function and a secret blinding factor r . In particular, this allows to blind a signed pseudonym (s, p, σ_{x-p}) into a new signed pseudonym $(s', p', \sigma_{x-p'})$. If the blinding factor r is chosen randomly, the resulting pseudonym p' will also be a random value. Furthermore, given r , the blinding is invertible in the sense that any signature on p' can be transformed into a new signature on the original pseudonym p .

In the remainder, we will denote the blinding of a signed pseudonym with a random r as $(s', p', \sigma_{x-p'}) = \text{Blind}((s, p, \sigma_{x-p}), r)$ and its inverse as $(s, p, \sigma_{y-p}) = \text{Blind}^{-1}((s', p', \sigma_{y-p'}), r)$. Note that, although the pseudonyms, their secret keys and the blinding factor are the same, the signatures and their creators may be different. In the latter case, signature $\sigma_{y-p'}$ can be obtained from another signature $\sigma_{y-p''}$ by means of a blinding function or directly from y during a signing protocol.

System registration.

For the initial system registration, a user registers with a pseudonym provider pp . This registration is generally not anonymous and results in a first pseudonym p , the corresponding secret key s , and an initial subcredential $cred_{pp} = (\sigma_{pp-p}, C_{pk_{pp}})$. Here, $C_{pk_{pp}}$ is the same for each issued credential and states that pp is authorized to issue valid initial subcredentials.

Issuing a credential.

Before retrieving a subcredential from a certification authority cp , a user must prove that he is registered to the system. To achieve this, he creates and stores a random value r_{cp} and computes the tuple $(s_{cp}, p_{cp}, \sigma_{pp-p_{cp}}) = \text{Blind}((s, p, \sigma_{pp-p}), r_{cp})$. The tuple $(p_{cp}, (\sigma_{pp-p_{cp}}, C_{pk_{pp}}))$ is then shown to the certification authority cp , together with a proof that the user knows the secret key corresponding to p_{cp} . This proof can, for example, be a signature with the corresponding secret key on a nonce specified by cp , or a zero-knowledge proof of knowledge. If this succeeds, the user retrieves a new subcredential $cred_{cp} = (\sigma_{cp-p_{cp}}, C_{pk_{cp}})$ on p_{cp} . Again, $C_{pk_{cp}}$ is a non-unique certificate stating the applicability of the subcredential.

Showing a credential.

Suppose the user wants to show a credential composed of subcredentials $cred_{pp}, cred_{cp_1}, \dots, cred_{cp_n}$ to a verifying entity v . In a first step, the user creates a new signed pseudonym $(s_v, p_v, \sigma_{pp-p_v})$ by blinding his first signed pseudonym (s, p, σ_{pp-p}) with a random blinding factor r_v . Afterwards, all other subcredentials $cred_{cp_k} (1 \leq k \leq n)$ are transformed to conform with p_v by performing two consecutive blinding operations as follows: $(s_v, p_v, \sigma_{cp_k-p_v}) = \text{Blind}(\text{Blind}^{-1}((s_{cp_k}, p_{cp_k}, \sigma_{cp_k-p_{cp_k}}), r_{cp_k}), r_v)$. Here, the inner unblinding operation transforms cp_k 's signature $\sigma_{cp_k-p_{cp_k}}$ on p_{cp_k} in a signature σ_{cp_k-p} on p , and the outer operation transforms σ_{cp_k-p} in a signature $\sigma_{cp_k-p_v}$ on p_v . Note that efficiency can be improved by performing the inner unblinding operation beforehand (e.g. after issuance) and by storing the result. The resulting tuple $(p_v, (\sigma_{pp-p_v}, C_{pk_{pp}}), (\sigma_{cp_1-p_v}, C_{pk_{cp_1}}), \dots, (\sigma_{cp_n-p_v}, C_{pk_{cp_n}}))$ is then sent to the verifier together with a proof that the user knows the secret s_v corresponding to p_v .

Credential revocation.

To allow the revocation of credentials, certification authorities use signing keys with short validity periods. When the expiration time elapses, the authority publishes the new signatures only on the pseudonyms of which the credential has not been revoked during this time period.

Discussion.

The system proposed by Verheul has a lot of attractive properties. Due to the blinding facilities of the pseudonyms, multiple shows of the same credential will not introduce any linkabilities, even when they are performed with respect to the same verifier. Also, although not discussed in our global overview, techniques can be put in place to discourage the sharing of credentials. The most important feature of the system is however the ability to selectively disclose credential attributes. Indeed, each subcredential can be seen as a certified attribute, for which the user can decide himself whether or not to disclose it to a given verifier. This selective disclosure, however, is limited to a single decision of disclosure. In many situations this will be sufficient, but we might think

of special circumstances in which a user wants to disclose only a property of a given attribute, while hiding the attribute itself. As an example, given an attribute stating his date of birth, a user might just want to reveal that he is older than 18. Ideally, this should not require the retrieval of an extra credential certifying this adulthood. Finally, the revocation of credentials is rather limited, and requires the use of short-lived keys. In particular, it only allows to revoke a credential based on its issuance. When we want to revoke a credential as a consequence of a show protocol, we need other techniques which require the use of trusted third parties.

2.2 Elaborated systems

We first describe the general structure of an elaborated credential systems. Afterwards, a high level overview of both Brands' system and the CL system is given. This overview highlights the main differences between both systems.

2.2.1 General description

A credential system consists of users \mathcal{U} and organizations \mathcal{O} . Users retrieve and show credentials, while organizations issue credentials (issuers \mathcal{I}) or verify them (verifiers \mathcal{V}). A credential itself is a tuple $Cred = (sk, pk, auth_{\mathcal{I}})$ composed of a secret key, a public key and an authorization of the issuer. The secret key is only known to the credential owner, while its public key and its authorization may also be known to other entities. Next to this, a credential may have additional properties, for example, it may be one-show or multiple-show.

Given a parameter $ST \in \{Brands, CL\}$ representing the system. In order to retrieve a credential, user \mathcal{U} performs an issue protocol with \mathcal{I} . This protocol will be denoted as $\mathcal{U} \leftrightarrow \mathcal{I} : getCred(ST, pk_{\mathcal{I}}, attrs, properties) \rightarrow Cred$. Here, $pk_{\mathcal{I}}$ is the issuer's public key, $attrs$ is a list of variables referring to the attributes to be incorporated into the credential and $properties$ specifies the credential properties which are known to \mathcal{I} when the credential is issued. These properties include, for example, values of attributes, relations between attributes, or whether or not the credential is one-show.

Once in possession of a credential, \mathcal{U} can show it to a verifier \mathcal{V} . During this protocol, properties of attributes encoded into the credential can be selectively disclosed. The show protocol is denoted as $\mathcal{U} \leftrightarrow \mathcal{V} : showCred(ST, Cred, shown properties)$. Parameter $Cred$ denotes the credential to be shown, while the *shown properties* are the credential properties explicitly revealed during the protocol.

Finally, a credential can be revoked by an organization. In the remainder of the paper, this will be denoted as $\mathcal{O} : revokeCred(ST, Cred)$.

2.2.2 Cryptographic building blocks

Both the CL system and Brands' system make use of the same cryptographic primitives; commitments, zero-knowledge proofs of knowledge and signature schemes for signing blinded messages.

Commitments can be seen as the digital analogue of “non-transparent sealed envelopes”. They allow to hide the committed values (non-transparency property), while ensuring that these values cannot be modified after commitment (sealed property). Statements about the committed attributes can be proven using zero-knowledge proofs of knowledge. Such a proof is an interactive two-party protocol between a prover and a verifier. After successful execution, the verifier is convinced that the prover knows the values enclosed into the commitment and that these values satisfy certain predefined properties. However, apart from the proven statements, the verifier does not gain any additional information about the committed values. In the remainder, a variation of the notation introduced by Camenisch and Stadler [6] will be used for representing proofs of knowledge. For instance, $\mathcal{U} \leftrightarrow \mathcal{V} : PK\{(x_1, \dots, x_l) : y = \text{commit}(x_1, \dots, x_l)\}$ denotes a zero-knowledge proof of knowledge, between prover \mathcal{U} and verifier \mathcal{V} , of the values x_1, \dots, x_l enclosed into commitment y . Here, the symbols introduced in brackets denote the values of which knowledge is to be proven, while all other values are known to both the prover and the verifier.

Signature schemes are used to create the issuer’s authorization of a credential. An additional property of the schemes is that the signer can sign messages without having to know their values. Also, adequate control measures are taken to ensure the correctness of the message to be signed.

2.2.3 Brands’ system

Brands proposes a myriad of techniques. These techniques can be divided into two settings: the DL (discrete logarithm) setting and the RSA setting. Both settings introduce their own properties regarding security and functionality. The focus in this paper, however, lies on the RSA-setting. Although somewhat less efficient than the DL setting, it is the only setting in which anonymous updating of credentials can easily be achieved¹.

Setting.

A credential is a tuple $(sk, pk, \sigma = \text{sign}_{sk_{\mathcal{I}}}(pk))$. The secret key sk consists of some randomness and the attributes encoded into the credential. The corresponding public key pk is a commitment to these attributes and σ is a signature of an authorized issuer on this public key. The used commitment scheme is a one-way and collision-intractable function whose security is based on the one-wayness of RSA. Its structure is as follows.

system parameters: The system parameters consist of a tuple (n, v, g_1, \dots, g_l) with n an RSA modulus, $v < n$ prime with $\text{gcd}(v, \phi(n)) = 1$, and $g_k \in_{\mathcal{R}} \mathbb{Z}_n^*$ for each $k \in \{1, \dots, l\}$

¹Note that an anonymous updating strategy can be applied to the Chaum-Pedersen based DL-issue protocol ([2], section 4.5.2). Using this strategy, the user creates the new credential himself and convinces the issuer of its correct form w.r.t. the old credential by using zero-knowledge proofs. This technique is however not very efficient.

commitment: A commitment $c = \text{commit}(x_1, \dots, x_l, r)$ for inputs $x_k \in \mathbb{Z}_v$ ($k \in \{1, \dots, l\}$) and randomness $r \in_{\mathcal{R}} \mathbb{Z}_n^*$ is defined as follows. $c = (\prod_{k=1}^l g_k^{x_k}) r^v$.

A myriad of zero-knowledge proofs concerning the secrets underlying a commitment can be constructed. Next to proofs of representation, additional proofs can be generated for so-called atomic formulae F . These formulae consist of linear relations between the secrets, connected by AND-connectives and maximal 1 NOT connective. Also, conjunctions of disjunctions of atomic formulae and efficient proofs that a committed secret lies within a certain interval, can be constructed. This last technique, together with the division of the attribute space \mathbb{Z}_v into a positive and negative range, allows to prove formulae $(x > y)$ or $(x < y)$ for possibly secret positive values x and y , by proving that $(x - y)$ (respectively $(y - x)$) still lies within the positive range.

Issuing a credential.

During the issue protocol, issuer \mathcal{I} creates a signature σ on the public credential key pk . This signature is created in such a way that neither pk nor σ are known to \mathcal{I} . Hence, further use of the credential cannot be linked to its issue protocol.

Suppose the system parameters of the signature scheme are a tuple (n, v) , with $n = pq$ the product of two primes and $v < n$ a prime such that $\text{gcd}(v, \phi(n)) = 1$. The secret signing key of the issuer is then defined as $sk_{\mathcal{I}} = (p, q)$, while his public key $pk_{\mathcal{I}}$ is a tuple $(f, h_0, g_1, \dots, g_l) \in_{\mathcal{R}} (\mathbb{Z}_n^*)^{l+2}$. Next to this, a value s superpolynomial in the binary size of n , and a one-way hash-function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_s$ are defined and made public.

The output of the issue protocol is a credential $Cred = (sk, pk, \sigma)$. Secret key sk is a tuple $(x_1, \dots, x_l, \alpha)$. Here, the x_k ($k \in \{1, \dots, l\}$) are the attributes encoded into the credential and α is a randomly chosen value in \mathbb{Z}_n^* . Public key pk is a commitment $(\prod_{k=1}^l g_k^{x_k}) h_0 \alpha^v$ and signature σ is a tuple $(c_0, r_0, r_1) \in \mathbb{Z}_s \times \mathbb{Z}_n^* \times \mathbb{Z}_v$, such that $c_0 = \mathcal{H}(pk, (r_0)^v f^{c_0} (pk)^{-r_1})$.

Signature σ will be computed using a restrictive blind signature scheme. This is a blind signature scheme in the general sense (i.e. the issuer will not be able to link the resulting public key pk , nor its signature σ , towards its issue protocol), but with the additional property that \mathcal{I} can put restrictions on the attributes encoded into the final key pk . The issue protocol itself is an interactive protocol between \mathcal{U} and \mathcal{I} , which takes as public input a commitment $h = (\prod_{k=1}^l g_k^{x_k}) h_0 \beta^v$ and returns as private output for the user a public key $pk = h \gamma^v$ for a random $\gamma \in_{\mathcal{R}} \mathbb{Z}_n^*$ chosen by \mathcal{U} , and a signature σ on pk .

The attributes encoded in the initial commitment h may not be known to \mathcal{I} , apart from some properties explicitly proven by \mathcal{U} through the use of zero-knowledge proofs. As an extension of the original issue protocol proposed by Brands, these attributes can be created in such a way that they correspond to one of the sets $X_{\mathcal{U}}, X_{\mathcal{I}}$ or $X_{\mathcal{J}}$. Here, $X_{\mathcal{U}} = \{k | x_k \text{ chosen by } \mathcal{U}\}$ are the indices of attributes chosen by \mathcal{U} and not known to the \mathcal{I} , $X_{\mathcal{I}} = \{k | x_k \text{ chosen by } \mathcal{I}\}$ corresponds to attributes chosen by \mathcal{I} and known to both \mathcal{U} and \mathcal{I} , and $X_{\mathcal{J}} = \{k | x_k = x_{ku} + x_{ki}, x_{ku} \text{ chosen by } \mathcal{U}, x_{ki} \text{ chosen by } \mathcal{I}\}$ are indices of attributes chosen by \mathcal{U} and \mathcal{I} jointly, such that only \mathcal{U} knows the final attribute

and neither \mathcal{U} nor \mathcal{I} can control its value. The latter can be used to create random attributes for which both user and issuer provide input.

The extended issue protocol for the given system parameters and public signing key, is described in figure 1.

Figure 1: extended issue protocol for Brands' system

| step 1: creation of initial commitment h | |
|---|--|
| \mathcal{U} | : choose $x_{ku}(k \in X_{\mathcal{J}}), x_k(k \in X_{\mathcal{U}}), \alpha \in_{\mathcal{R}} \mathbb{Z}_n^*$ |
| $\mathcal{U} \rightarrow \mathcal{I}$ | : send($C = (\prod_{k \in X_{\mathcal{J}}} g_k^{x_{ku}})(\prod_{k \in X_{\mathcal{U}}} g_k^{x_k})\alpha^v$) |
| $\mathcal{U} \leftrightarrow \mathcal{I}$ | : optional: zero-knowledge proof of attribute properties |
| \mathcal{I} | : choose $x_{ki}(k \in X_{\mathcal{J}}), x_k(k \in X_{\mathcal{I}})$ |
| $\mathcal{U} \leftarrow \mathcal{I}$ | : send($(x_{ki})_{k \in X_{\mathcal{J}}}, (x_k)_{k \in X_{\mathcal{I}}}$) |
| \mathcal{U} | : compute $x_k = x_{ki} + x_{ku} (k \in X_{\mathcal{J}})$ |
| \mathcal{U}, \mathcal{I} | : compute $h = Ch_0(\prod_{k \in X_{\mathcal{J}}} g_k^{x_{ki}})(\prod_{k \in X_{\mathcal{I}}} g_k^{x_k})$ |
| step 2: credential issue protocol with common input h | |
| \mathcal{I} | : choose $a_0 \in_{\mathcal{R}} \mathbb{Z}_n^*$ |
| $\mathcal{U} \leftarrow \mathcal{I}$ | : send(a_0) |
| \mathcal{U} | : choose $\beta, \alpha_1 \in_{\mathcal{R}} \mathbb{Z}_n^*, \alpha_2, \alpha_3 \in_{\mathcal{R}} \mathbb{Z}_v$ compute $pk = h\beta^v$ compute $c_0 = \mathcal{H}(pk, \alpha_1^v f^{-\alpha_2} (pk)^{-\alpha_3} a_0)$ compute $c'_0 = c_0 + \alpha_2 \pmod v$ |
| $\mathcal{U} \rightarrow \mathcal{I}$ | : send(c'_0) |
| \mathcal{I} | : choose $r'_1 \in_{\mathcal{R}} \mathbb{Z}_v$ compute $r'_0 = (h^{r'_1} a_0 / f^{c'_0})^{1/v}$ |
| $\mathcal{U} \leftarrow \mathcal{I}$ | : send(r'_0, r'_1) |
| \mathcal{U} | : check $f^{c'_0} (r'_0)^v h^{-r'_1} = a_0$ compute $r_0 = r'_0 \beta^{r'_1} \alpha_1 f^{-((c_0 + \alpha_2) \text{ div } v)} (pk)^{-((r'_1 + \alpha_3) \text{ div } v)}$ compute $r_1 = r'_1 + \alpha_3 \pmod v$ compute $\sigma = (c_0, r_0, r_1)$ |

Showing a credential.

A user \mathcal{U} wanting to show his credential to a verifier \mathcal{V} sends both the public key pk and the signature to \mathcal{V} . \mathcal{U} then performs a proof of knowledge of the secrets underlying pk and possibly some additional relations among the attributes. Verifier \mathcal{V} accepts the credential if and only if he accepts both the proof of knowledge as well as the signature.

Note that, due to the fact that pk and σ are revealed to \mathcal{V} , different shows of the same credential can be linked to each other.

Revocation of credentials.

Credentials can be revoked by revoking their public key in the same way as certificates. However, as a credential's public key is not known to its issuer, it is

not possible to revoke a credential based on its issue protocol. In other words, a credential can only be revoked after its public key is revealed during a show protocol. This problem can be circumvented by the use of whitelisting (resp. blacklisting) techniques. A unique value, chosen by the issuer, is encoded into the credential during its issue protocol. A user then has to prove in each show protocol that this value is on the whitelist (resp. not on the blacklist). Note that this is essentially the same technique as used in the CL system.

Anonymous updating of a credential.

Based on an old credential $Cred$, a new credential $Cred'$ can be created in which attributes x_k are augmented with values δ_k . Consider for example a credential public key $pk = (\prod_{k=1}^l g_k^{x_k}) h_0 \alpha^v$. Being convinced of the validity of $Cred$, issuer \mathcal{I} can create a new credential identical to $Cred$, but with a value δ_k added to attribute x_k . For this, a new issue protocol between \mathcal{U} and \mathcal{I} is executed with public input $h = (pk)g_k^{\delta_k}$.

Note that \mathcal{I} does not have to know the original value of x_k , nor will he be able to link the old credential with the new one due to the blinding property of the issue protocol.

Limited-show credentials.

Limited-show credentials can only be shown a predetermined number of times. Detection of the overspending of a credential can be either on-line or off-line. While on-line detection only reveals the fact of overspending, off-line detection compromises the user's anonymity.

In Brands' system, on-line detection can be trivially achieved by using the property that a credential's public key is shown during each show protocol. Next, it is shown how one-show credentials with off-line detection are realized, the construction of t -show credentials is analogous.

Off-line detection is realized by fixing an initial parameter, to be used in the zero-knowledge proofs for showing a credential, beforehand. When the credential is shown twice using this same parameter, all the secrets underlying its public key are revealed. The parameter itself is fixed during the issue protocol and incorporated into the hash function \mathcal{H} by the user. Note that \mathcal{I} does not know this value. Hence, it cannot be used to link a credential to its issue protocol.

Even if a credential was issued as a limited-show credential, it can always be shown as a regular credential during its show protocol. This is true for both the on-line and the off-line detection case. Hence, a limited-show credential can be transformed into an unlimited-show credential and vice versa.

2.2.4 The CL system

The CL system is a full-fledged credential system. It contains techniques for issuing and showing credentials using the notion of pseudonyms. A pseudonym can be established between a user and an organization, and credentials can be issued and optionally verified with respect to this pseudonym. They are a use-

ful tool for building up pseudonymous reputations with organizations, but more importantly their use enables the introduction of various security properties into elaborated applications. In order not to overcomplicate things, pseudonyms are omitted in this paper. Rather, a simplified version of Idemix without pseudonyms is presented. Note, however, that an extension towards a system with pseudonyms is straightforward.

Recently a new signature scheme based on bilinear maps, with security based on a discrete-log based assumption, has been presented [5]. This scheme can be plugged into the current CL system, allowing for simpler and more efficient protocols. In this paper, however, the “old” credential system based on the old signature scheme is discussed. The anonymity properties of this system are the same as for the new system. Also, note that the revocation of credentials in the old system needs some adaptation to be workable in the new system.

Setting.

A credential in the CL system is a tuple (sk, pk, σ) . As with Brands, public key pk is a commitment based on the secrets in sk . However, signature σ is now a signature on the attributes (x_1, \dots, x_l) encoded into the credential. Contrary to Brands’ system, knowledge of a signature on a committed message can be proven in zero-knowledge. Also, signatures can be generated on committed messages.

The used commitment scheme ([12]) is statistically hiding and computational binding if factoring is hard. Next to this, it allows to commit to integer values instead of elements within a fixed group. Its structure is as follows.

System parameters: The public key of the system consists of a special RSA modulus $n = pq$ with $p = 2p' + 1$, $q = 2q' + 1$ and p, p', q, q' prime, and values $g_1, \dots, g_l, b \in_{\mathcal{R}} QR_n$ (the group of quadratic residues modulo n). Remark that with high probability $\langle g_1 \rangle = \dots = \langle g_l \rangle = \langle b \rangle = QR_n$.

Commitment: A commitment $C = \text{commit}(x_1, \dots, x_l, r)$ to elements x_k ($k \in \{1, \dots, l\}$) with randomness $r \in \mathbb{Z}_{\lfloor n/4 \rfloor}$ is defined as $C = g_1^{x_1} \dots g_l^{x_l} b^r \pmod n$. Note that $\lfloor n/4 \rfloor$ is a lower bound on $|QR_n| = p'q'$.

Next to commitments, zero-knowledge proofs are an important aspect of Idemix. These proofs include, among others, proofs of knowledge of a representation of an element y with respect to a base-tuple (g_1, \dots, g_l) and proofs of knowledge and equality of discrete logarithms $x = \log_{g_1} y_1 = \log_{g_2} y_2$. Next, there are proofs of knowledge of a discrete logarithm $x = \log_g y$, where x lies in an interval $[a, b]$, and proofs of knowledge and equality in different groups. It is also possible to demonstrate a proof of formulae $(x > y)$ or $(x < y)$ for values x and y . Consider for example the case of $(x > y)$, a demonstration of this property is achieved by demonstrating that $x - y > 0$ through a proof of knowledge of 4 integers t_1, \dots, t_4 , not all equal to 0, such that $x - y = \sum_{i=1}^4 t_i^2$. This must be done in zero-knowledge with respect to $x - y$ and the t_i ’s.

Issuing a credential.

An important aspect of the issue protocol is the creation of a signature on

attributes (x_1, \dots, x_l) . Suppose the public signing key of the issuer is composed of a special RSA modulus n and random values $g_k (k \in \{1, \dots, l\}), b, d \in_{\mathcal{R}} QR_n$, while its secret key is the factorization of n . The issue protocol of a credential consists of two steps.

During a first step, secret key $sk = (x_1, \dots, x_l, s)$ and corresponding commitment $pk = (\prod_{k=1}^l g_k^{x_k})b^s$ are created. Here, value s is some randomness only known to \mathcal{U} but chosen jointly by \mathcal{U} and \mathcal{I} , while $x_i (1 \leq i \leq l)$ are attributes corresponding to one of the sets $X_{\mathcal{U}}, X_{\mathcal{I}}$ or $X_{\mathcal{J}}$. Additionally, by using zero-knowledge proofs, the user can demonstrate relational properties between the attributes. In the second step of the issue protocol, \mathcal{I} chooses a random prime e and computes the value $c = ((pk)d)^{1/e} \bmod n$. The resulting signature σ on attribute values (x_1, \dots, x_l) is the tuple (c, e, s) , and can be verified by checking the relation $c^e = g_1^{x_1} \dots g_l^{x_l} b^s d \bmod n$. This signature is unforgeable under the strong RSA assumption.

The complete issue protocol is described in figure 2. Note the use of two extra system parameters; value $h \in_{\mathcal{R}} QR_n$ belongs to the issuer’s public key, while value g is a generator of a group G_q of prime order q . Also, note the use of rangechecks in the protocol. These checks are needed to guarantee that certain parameters lie within a correct “security interval”. For ease of representation and understanding, however, abstraction is made of the exact nature of these rangechecks.

Showing a credential.

In order to show his credential to a verifier, \mathcal{U} cannot simply show its public key and signature to \mathcal{V} . This is because both pk and values (c, e) of signature (c, e, s) are known to \mathcal{I} . As a consequence, showing these values would reveal a link between the issuing and the showing of a credential. The solution for \mathcal{U} is to provide a zero-knowledge proof of both the attributes and a valid signature. Due to the zero-knowledge property of this proof, different shows of the same credential cannot be linked. Also, additional properties of attributes can be demonstrated.

In the remainder, a zero-knowledge proof of a signature (c, e, s) on attributes (x_1, \dots, x_l) will be referred to as $PK\{(x_1, \dots, x_l, c, e, s) : verify_{pk_{\mathcal{I}}}((x_1, \dots, x_l), (c, e, s))\}$, with $pk_{\mathcal{I}}$ the public signing key of \mathcal{I} .

Revocation of credentials.

Credential revocation is performed by maintaining a public whitelist of valid credentials. This list is implemented using a dynamic accumulator [4] containing all values e of signatures (c, e, s) belonging to valid credentials. Whenever a user shows a credential, he proves in zero-knowledge that its corresponding value e is incorporated into the accumulator. When it is revoked, the value e is removed from the accumulator.

Note that this revocation is based on the issue protocol. An advantage of this approach is the ability to revoke credentials before they are shown. When revocation must be based on a show protocol, however, a link towards the cre-

Figure 2: issue protocol for the CL system

| step 1: creation of public key pk | |
|---|---|
| \mathcal{U} | : choose $x_{ku} (k \in X_{\mathcal{J}})$ and random s_u, r |
| $\mathcal{U} \rightarrow \mathcal{I}$ | : send($C = (\prod_{k \in X_{\mathcal{J}}} g_k^{x_{ku}}) b^{s_u} h^r$) |
| $\mathcal{U} \leftrightarrow \mathcal{I}$ | : $PK\{((x_{ku})_{k \in X_{\mathcal{J}}}, s_u, r) :$ $C = (\prod_{k \in X_{\mathcal{J}}} g_k^{x_{ku}}) b^{s_u} h^r \wedge \text{rangechecks}\}$ |
| \mathcal{I} | : choose $x_{ki} (k \in X_{\mathcal{J}}), x_k (k \in X_{\mathcal{I}})$, random value s_i |
| $\mathcal{U} \leftarrow \mathcal{I}$ | : send($(x_{ki})_{k \in X_{\mathcal{J}}}, (x_k)_{k \in X_{\mathcal{I}}}, s_i$) |
| \mathcal{U} | : compute the following values: $x_k = x_{ku} + x_{ki} (k \in X_{\mathcal{J}})$ $x_k = \text{as chosen by } \mathcal{U} \text{ or } \mathcal{I} (k \in X_{\mathcal{U}} \cup X_{\mathcal{I}})$ $s = s_u + s_i$ |
| $\mathcal{U} \rightarrow \mathcal{I}$ | : send($pk = (\prod_{k=1}^l g_k^{x_k}) b^s$) |
| $\mathcal{U} \leftrightarrow \mathcal{I}$ | : $PK\{((x_k)_{k \in (X_{\mathcal{U}} \cup X_{\mathcal{J}})}, (x_{ku})_{k \in X_{\mathcal{J}}}, s, s_u, r) :$ $\frac{pk}{\prod_{k \in X_{\mathcal{I}}} g_k^{x_k}} = (\prod_{k \in (X_{\mathcal{U}} \cup X_{\mathcal{J}})} g_k^{x_k}) b^s \wedge$ $C b^{s_i} = (\prod_{k \in X_{\mathcal{J}}} g_k^{x_{ku}}) b^s h^r \wedge$ $g^{x_{ki}} = (1/g)^{x_{ku}} g^{x_k} (\forall k \in X_{\mathcal{J}}) \wedge \text{rangechecks}\}$ |
| step 2: signature creation | |
| \mathcal{I} | : choose random prime e , compute $c = ((pk)d)^{1/e}$ |
| $\mathcal{U} \leftarrow \mathcal{I}$ | : send((c, e)) |
| \mathcal{U} | : check $c^e = (pk)d$ set $\sigma = (c, e, s)$ |

dential's issue protocol must be revealed².

Limited-show credentials.

The CL system allows for one-show credentials with both on-line and off-line verification. Here, off-line detection reveals a link with the credential's issue protocol.

On-line verification: The main idea behind on-line verification is to incorporate a unique value into the credential, and to show this value during every show protocol. For this an extra attribute $r \in X_{\mathcal{J}}$ is chosen jointly during the issue protocol. Whenever the credential is shown, value $v = g^r$ is provided to the verifier together with a zero-knowledge proof that $\log_g v$ is encoded into the credential. Verifier \mathcal{V} accepts this show protocol only if v has not been shown previously.

Off-line verification: Off-line verification is enabled by creating a hidden link between a credential's issue and show protocol. This link is revealed only when the credential is doublespent. Its creation is as follows.

²The conditional revealing of this link is realized by decrypting a verifiable encryption of the linking information. This encryption is provided by \mathcal{U} during the show protocol.

During the issue protocol, three jointly chosen attributes r_1, r_2 and $r_3 \in X_{\mathcal{I}}$ are incorporated into the credential. Next to this, \mathcal{U} provides \mathcal{I} with the value $w = g^{r_2}$ and proves in zero-knowledge that w is formed correctly. When the credential is shown, \mathcal{U} receives a challenge $c \in_{\mathcal{R}} \mathbb{Z}_q$ from verifier \mathcal{V} and replies with values $m = g^{r_1}$ and $t = cr_2 + r_3 \pmod q$. Additionally, \mathcal{U} proves in zero-knowledge that m and t are formed correctly. After doublespending, with high probability two tuples (m, c_1, t_1) and (m, c_2, t_2) with $c_1 \neq c_2$ can be found. The value $g^{(t_1 - t_2)/(c_1 - c_2)}$ can then be computed and used to find a link towards the issue protocol.

In the remainder of this paper, an issue and a show-protocol with on-line (off-line) verification will be represented by adding the tag `dblsp-onl`(r_1) (`dblsp-offl`(r_1, r_2, r_3)) to the attribute properties or shown formulae. Note again that a limited-show credential (either with on-line or off-line verification) can always be shown as a regular credential during the show protocol.

2.2.5 Comparison

The techniques used in Brands' system and the CL system follow the same basic principles regarding the encoding and selective disclosure of attributes. In both systems, the credential public key is a commitment based on the attribute values and the randomness incorporated into the secret key. Also, attributes are selectively disclosed by means of zero-knowledge proofs. The main difference between both systems is the signature scheme used by \mathcal{I} and its implications.

- In the CL system, the issuer signs a list of messages. The resulting signature is (partly) known and therefore recognizable by the issuer. However, as a signature can be generated on committed messages, the signed messages may not be known to \mathcal{I} . The signature scheme also enables zero-knowledge proofs of knowledge of a signature on committed messages. Making use of this property, different shows of the same credential cannot be linked to each other nor to their issue protocol.
- In Brands' system, the public key is signed using a blind signature scheme. The attributes encoded into the credential may not be known to \mathcal{I} , nor does he know the signed public key. As a natural consequence of the latter property, a credential's issue protocol is unlinkable towards its show protocols. However, as a signature on a public key cannot be shown in zero-knowledge, different shows of the same credential are still linkable.

Next to this, revocation of credentials is performed differently in both systems. The CL system only allows for revocation of credentials based on their issue protocol. In Brands' system, however, credentials can either be revoked based on their show protocol (and hence without a link to their issue protocol), or based on their issue protocol. Moreover, both techniques can be combined.

Also note that doublespending detection in Brands' system allows to retrieve all the secrets underlying a credential. This is a stronger anonymity breach than in the CL system, where only either the fact of doublespending or a link towards

Table 1: Efficiency Measures for the CL system and Brands system

| issue protocol: | | | | |
|-----------------|----------------------|----------------------|---------------|--------|
| | expon. \mathcal{U} | expon. \mathcal{I} | transf. data | moves |
| Brands standard | $l + 3j + 2u + 13$ | $l + j + 5$ | $l + j + 8$ | 8 |
| + rev/onl/offl | +0/0/0 | +0/0/0 | +0/0/0 | +0/0/0 |
| CL standard | $l + 6j + u + 9$ | $l + 6j + 10$ | $l + 3j + 16$ | 10 |
| + revocation | +1 | +1 | +1 | +0 |
| + onl dblsp | +7 | +5 | +4 | +0 |
| + offl dblsp | +22 | +23 | +14 | +0 |
| show protocol: | | | | |
| | expon. \mathcal{U} | expon. \mathcal{V} | transf. data | moves |
| Brands standard | $2l + 2$ | $l + 5$ | $l + 7$ | 4 |
| + rev/onl/offl | +0/0/0 | 0/0/0 | 0/0/0 | 0/0/0 |
| CL standard | $l + 11$ | $l + 11$ | $l + 12$ | 4 |
| + revocation | +10 | +10 | +9 | +0 |
| + onl dblsp | +3 | +3 | +3 | +0 |
| + offl dblsp | +7 | +8 | +8 | +2 |

the issue protocol is revealed. However, it is the user itself who has the ability to prohibit this linking.

Table 1 gives an overview of the communication and computation complexity for the issue and show protocols of both systems. The computation complexity is measured by the number of exponentiations performed by the parties involved in each protocol, while the communication complexity is measured by the number of individual elements to be transferred and the number of interactions needed to complete each protocol. When retrieving a credential, we consider the standard protocol in which a credential is issued containing l attributes, each of which belongs either to the sets $X_{\mathcal{I}}, X_{\mathcal{U}}$ or $X_{\mathcal{J}}$. We denote the size of each of these sets $|X_{\mathcal{I}}|, |X_{\mathcal{U}}|$ and $|X_{\mathcal{J}}|$ to be i, u and j respectively. Hence, $l = i + u + j$. The show protocol considers the same credential, shown without revealing any of its attributes. Additionally, we measure the extra costs needed to enable the additional functionalities of credential revocation and one-show credentials with on-line or off-line detection.

An important remark when considering both protocols is the difference in the zero-knowledge proofs used. Both are standard three move protocols. However, unlike in the proofs used in the CL system, Brands' RSA-based proofs require an additional exponentiation for each response that is computed based on the challenge. On the other hand, revocation and double spending detection come for free in Brands' system. Also note that for the issue protocol of both systems, the performance measures form an upper bound based on the sizes of the sets $X_{\mathcal{I}}, X_{\mathcal{U}}$ and $X_{\mathcal{J}}$. More specifically, if one of these sets is empty, some steps in the issue protocol may be skipped, which leads to better efficiency.

3 Anonymous updating of credentials

3.1 Introduction and notation

During a credential update, a new credential is issued based on an older credential. Both credentials are signed using the same keypair and are unlinkable towards each other. The new credential's attributes are based on the attributes of the older credential. Also, its issue protocol is conducted in such a way that the issuer is only aware of the relation between the attributes in the old and in the new credential, without knowing their actual values.

The update protocol itself is a combination of both a show protocol and an issue protocol. During the show protocol, the user proves possession of the old credential, while in the issue protocol the new updated credential is issued. In this section, updating techniques are presented for both the CL system and Brands' system. In both systems, the resulting credential is unlinkable towards the old credential. The following update actions are allowed.

- Add a value δ_k to an attribute x_k in the credential. The original attribute x_k does not have to be known to \mathcal{I} .
- Replace an attribute x_k in the credential by a newly chosen attribute x'_k . The new attribute can be chosen by the user, the issuer, or jointly. In particular, replacing an attribute's value by 0 may in some cases correspond to the removal of this attribute from the credential.

Consider a credential $Cred = (sk, pk, \sigma)$ containing attributes (x_1, \dots, x_l) . The result of an update is a new credential $Cred' = (sk', pk', \sigma')$ with attributes (x'_1, \dots, x'_l) . The new attribute values x'_k are either the old values ($x'_k = x_k$), the old values augmented with a value δ_k ($x'_k = x_k + \delta_k$), or completely new values. We define the following sets.

- Denote $L = \{1, \dots, l\}$ the set of attribute indices, then $L = C \cup S$, where S contains the indices of attributes which remain unchanged and $C = L \setminus S$ are the indices of attributes to be updated.
- $C = R \cup D$, where $R = R_{\mathcal{U}} \cup R_{\mathcal{I}} \cup R_{\mathcal{J}}$ are the indices of attributes which are replaced by a new value chosen by the user, issuer or jointly, and D represents the indices of attributes updated by a publicly known value δ chosen by \mathcal{I} . Note that $R_{\mathcal{U}}, R_{\mathcal{I}}$ and $R_{\mathcal{J}}$ form a partition of R , and R and D form a partition of C . Also note that δ is always chosen by the issuer. This is justified, as a δ chosen jointly or by the user is essentially the same as a new value chosen jointly or by the user.

In the sequel, the update protocol between a user \mathcal{U} and issuer \mathcal{I} will be denoted by $\mathcal{U} \leftrightarrow \mathcal{I} : \text{updateCred}(ST, Cred, \text{shown props}, \text{update props}) \rightarrow Cred'$. Parameters $ST \in \{\text{Brands}, \text{CL}\}$ and *shown props* respectively represent the system used and the properties of the old credential explicitly demonstrated during the show protocol. Finally, *update props* is a description of the changes made to the attributes in $Cred$ and known to \mathcal{I} .

3.2 Anonymous updating of credentials in Brands' system

Brands [2] proposed updating techniques for adding a publicly known value δ to a credential attribute. In this section, these techniques are extended towards an update protocol where old attributes can be replaced by new ones. The new values can be chosen by the user, the issuer or jointly.

Let $Cred = (sk, pk, \sigma)$ be the credential to be updated. $Cred$ contains a secret key $sk = (x_1, \dots, x_l, \alpha)$, a public key $pk = (\prod_{k=1}^l g_k^{x_k})h_0\alpha^v$, and signature $\sigma = (c_0, r_0, r_1)$. Figure 3 describes the updating protocol. This protocol is again divided into two steps. During step 1, an initial commitment h on the new attributes is created. Afterwards, a blind signature σ' on the new public key $pk' = (\prod_{k=1}^l g_k^{x'_k})h_0(\alpha')^v$ is created.

Figure 3: anonymous updating by Brands

| step 1: creation of initial commitment h | |
|---|--|
| 1.1 | \mathcal{U} : choose $\alpha_1 \in_{\mathcal{R}} \mathbb{Z}_n^*$, create $C = (\prod_{k \in R} g_k^{x_k})\alpha_1^v$ |
| 1.2 | $\mathcal{U} \rightarrow \mathcal{I}$: send(C, pk, σ) |
| 1.3 | \mathcal{I} : verify $_{pk_{\mathcal{I}}}(pk, \sigma)$ |
| 1.4 | \mathcal{U}, \mathcal{I} : compute $h_1 = (pk)/C$ |
| 1.5 | $\mathcal{U} \leftrightarrow \mathcal{I}$: $PK\{((x_k)_{k \in R}, \alpha_1) : C = (\prod_{k \in R} g_k^{x_k})\alpha_1^v\}$ |
| 1.6 | $\mathcal{U} \leftrightarrow \mathcal{I}$: $PK\{((x_k)_{k \in (L \setminus R)}, z) : (h_1 = \prod_{k \in (L \setminus R)} g_k^{x_k})h_0z^v\}$ |
| 1.7 | \mathcal{U} : choose $x'_{ku}(k \in R_{\mathcal{U}}), x'_{ki}(k \in R_{\mathcal{I}}), \alpha_2 \in_{\mathcal{R}} \mathbb{Z}_n^*$ |
| 1.8 | \mathcal{U} : compute $a = (\prod_{k \in R_{\mathcal{U}}} g_k^{x'_{ku}})(\prod_{k \in R_{\mathcal{I}}} g_k^{x'_{ki}})\alpha_2^v$ |
| 1.9 | $\mathcal{U} \rightarrow \mathcal{I}$: send(a) |
| 1.10 | $\mathcal{U} \leftrightarrow \mathcal{I}$: $PK\{((x'_{ku})_{k \in R_{\mathcal{U}}}, (x'_{ki})_{k \in R_{\mathcal{I}}}, \alpha_2) :$ $a = (\prod_{k \in R_{\mathcal{U}}} g_k^{x'_{ku}})(\prod_{k \in R_{\mathcal{I}}} g_k^{x'_{ki}})\alpha_2^v\}$ |
| 1.11 | \mathcal{I} : choose $\delta_k(k \in D), x'_k(k \in R_{\mathcal{I}}), x'_{ki}(k \in R_{\mathcal{I}})$ |
| 1.12 | $\mathcal{U} \leftarrow \mathcal{I}$: send($(\delta_k)_{k \in D}, (x'_k)_{k \in R_{\mathcal{I}}}, (x'_{ki})_{k \in R_{\mathcal{I}}}$) |
| 1.13 | \mathcal{U} : compute the following values: $x'_k = x'_{ku} + x'_{ki} (k \in R_{\mathcal{I}})$ $x'_k = x_k + \delta_k (k \in D)$ |
| 1.14 | \mathcal{U}, \mathcal{I} : compute $h_2 = a(\prod_{k \in D} g_k^{\delta_k})(\prod_{k \in R_{\mathcal{I}}} g_k^{x'_k})(\prod_{k \in R_{\mathcal{I}}} g_k^{x'_{ki}})$ |
| step 2: credential issue protocol with common input $h = h_1 h_2$ | |
| \mathcal{U} retrieves signature σ' on $pk' = (h_1 h_2)(\alpha')^v$ with $\alpha' \in_{\mathcal{R}} \mathbb{Z}_n^*$ | |

During steps 1.1-1.6, user \mathcal{U} proves possession of the credential to be recertified. This is done by sending \mathcal{I} his credential public key $pk = Ch_1$, together with a valid signature σ on pk . Afterwards, \mathcal{I} checks the signature and \mathcal{U} proves in steps 1.5 and 1.6 that he knows a representation of pk . Note that these steps can be extended for \mathcal{U} to prove additional properties of the credential's attributes. Additionally, steps 1.1-1.6 ensure that the attributes to be replaced are removed from the original public key pk , resulting in a smaller commitment h_1 .

In this regard, step 1.5 ensures that only elements in R are removed from pk , while step 1.6 ensures that these elements are removed entirely. Finally, in steps 1.7-1.14, h_1 and the newly chosen attributes are combined into a new common input h for the issue protocol.

Note that, due to the unlinkability properties of the issue protocol, the new credential will not be linkable towards the old one.

3.3 Anonymous updating of credentials in the CL system

Let $Cred = (sk, pk, \sigma)$ be the credential to be updated. $Cred$ contains a secret key $sk = (x_1, \dots, x_l, s)$, a public key $pk = (\prod_{k=1}^l g_k^{x_k})b^s$, and a signature $\sigma = (c, e, s)$ such that $c^e = (pk)d$. Its update protocol consists of two steps; in the first step a new public key $pk' = (\prod_{k=1}^l g_k^{x'_k})b^{s'}$ is agreed on. Afterwards, the issuer chooses value e' and computes $c' = ((pk')d)^{\frac{1}{e'}}$ in step 2. The complete update protocol is described in figure 4. Again, for ease of representation and understanding, abstraction is made of security rangechecks.

During step 1, \mathcal{U} creates a new public key pk' based on both the attributes of the old credential as well as the input values, chosen by \mathcal{U} and/or \mathcal{I} , for the new attributes. In substep 1.8, \mathcal{U} must prove to \mathcal{I} that this new public key is correctly formed with respect to these inputs. More specific, \mathcal{U} proves that all values x'_k with $k \in R_{\mathcal{I}}$ are included in pk' , and that x'_k for each $k \in D$ is included and correctly formed as $x_k + \delta_k$ (1). Next to this, \mathcal{U} proves that randomness s' is correctly formed as $s'_u + s'_i$ (2) and that $x'_k = x'_{ku} + x'_{ki}$ for each $k \in R_{\mathcal{J}}$ (3). Proofs (2) and (3) assure that the input provided by \mathcal{I} for jointly chosen values is incorporated into the resulting public key. Finally, in a last proof \mathcal{U} demonstrates possession of a valid credential containing the old attributes x_k on which the new attributes are based, this additionally provides a proof that $x'_k = x_k$ for indices $k \in S$ (4).

Note that other properties of the old or new credential can easily be incorporated into this proof. Also note that the new credential is unlinkable towards the old one. This is due to the zero-knowledge property of the provided proofs. Finally, as the update protocol is a special version of an issue protocol, the revocation of a credential created during an update protocol will be based on this update protocol. In particular, when a credential is revoked based on a show protocol, a link with its update protocol must be revealed.

4 Case study: a loyalty card

In this section, we study the use of credentials in a specific type of application. Namely, in applications where different uses of the same real-life credential are not linkable and where the credential can be updated anonymously. An example of such an application is an anonymous electronic purse. Different uses of the same purse should not be linkable to each other. Moreover, when money is transferred to and from the purse, observers should only be aware of the amount of money that is transferred.

Figure 4: anonymous updating in Idemix

| step 1: creation of public key pk' | |
|--|---|
| 1.1 | \mathcal{U} : choose values $x'_{ku} (k \in R_{\mathcal{J}})$ and random values s'_u, t |
| 1.2 | $\mathcal{U} \rightarrow \mathcal{I}$: send($C = (\prod_{k \in R_{\mathcal{J}}} g_k^{x'_{ku}}) b^{s'_u} h^t$) |
| 1.3 | $\mathcal{U} \leftrightarrow \mathcal{I}$: $PK\{((x'_{ku})_{k \in R_{\mathcal{J}}}, s'_u, t) :$ $C = (\prod_{k \in R_{\mathcal{J}}} g_k^{x'_{ku}}) b^{s'_u} h^t \wedge \text{rangechecks}\}$ |
| 1.4 | \mathcal{I} : choose $x'_{ki} (k \in R_{\mathcal{J}}), x'_k (k \in R_{\mathcal{I}}), \delta_k (k \in D)$, random s'_i |
| 1.5 | $\mathcal{U} \leftarrow \mathcal{I}$: send($(x'_{ki})_{k \in R_{\mathcal{J}}}, (x'_k)_{k \in R_{\mathcal{I}}}, (\delta_k)_{k \in D}, s'_i$) |
| 1.6 | \mathcal{U} : compute the following values: $x'_k = x_k + \delta_k (k \in D)$ $s' = s'_u + s'_i$ $x'_k = x'_{ku} + x'_{ki} (k \in R_{\mathcal{J}})$ $x'_k = \text{as chosen by } \mathcal{U} \text{ or } \mathcal{I} (k \in R_{\mathcal{U}} \cup R_{\mathcal{I}})$ $x'_k = x_k (k \in S)$ |
| 1.7 | $\mathcal{U} \rightarrow \mathcal{I}$: send($pk' = (\prod_{k \in L} g_k^{x'_k}) b^{s'}$) |
| 1.8 | $\mathcal{U} \leftrightarrow \mathcal{I}$: $PK\{(x_k)_{k \in C}, (x'_k)_{k \in L \setminus (R_{\mathcal{I}} \cup D)}, s', (x'_{ku})_{k \in R_{\mathcal{J}}}, t, c, e, s) :$ $\frac{pk'}{(\prod_{k \in R_{\mathcal{I}}} g_k^{x_k})(\prod_{k \in D} g_k^{\delta_k})} = (\prod_{k \in (S \cup R_{\mathcal{U}} \cup R_{\mathcal{J}})} g_k^{x'_k})(\prod_{k \in D} g_k^{x_k}) b^{s'}$ $\wedge C b^{s'_i} = (\prod_{k \in R_{\mathcal{J}}} g_k^{x'_{ku}}) b^{s'_u} h^t$ $\wedge g^{x'_{ki}} = (1/g)^{x'_{ku}} g^{x'_k} (\forall k \in R_{\mathcal{J}})$ $\wedge \text{verify}_{pk_{\mathcal{I}}}(x_1^*, \dots, x_l^*). (c, e, s)$ $\wedge \text{rangechecks}\}$ |
| | (1) |
| | (2) |
| | (3) |
| | (4) |
| | with $x_k^* = \begin{cases} x'_k & \text{if } k \in S; \\ x_k & \text{if } k \in C. \end{cases}$ |
| step 2: signature creation | |
| 2.1 | \mathcal{I} : choose random prime e' , compute $c' = ((pk')d)^{1/e'}$ |
| 2.2 | $\mathcal{U} \leftarrow \mathcal{I}$: send(c', e') |
| 2.3 | \mathcal{U} : check $(c')^{e'} = (pk')$ |

Other examples are driver's licences with points and loyalty cards. These systems can be implemented using anonymous credentials, but not without a twist. Next, a loyalty card system based on anonymous credentials is developed and discussed. The implementation of the other applications is similar.

4.1 Description of the application

Retailers frequently offer loyalty cards to their customers. A loyalty card contains a record of points, initially set to zero and increased whenever the customer buys goods at the store. The points represent benefits for the user. When the record reaches a certain threshold, the points can be traded for cash or discounts.

Our loyalty card is anonymous; it cannot be used for tracking the activities of its owner or for creating a shopping pattern. Moreover the card can be shared by different entities, for example the members of the same household.

Use of a loyalty card is divided into the following three situations.

- **retrieving the card.** In order to obtain a loyalty card, the customer performs a payment. Optionally, he provides contact information for retrieving advertisements regarding the store's newest services. Finally, the point record of the newly issued card is set to zero.
- **showing the card.** A loyalty card can be shown without introducing changes to its stored data. A customer, for example, may need to show his loyalty card in order to enjoy a special self scanning service. The use of this service, however, does not increase his point record. Furthermore, showing the same loyalty card different times does not provide any linking information with either the retrieval phase of the card, nor with other show or update actions performed with the same card.
- **updating the card.** A card must be updated whenever its point record changes. This is the case whenever a customer buys items or retrieves a discount. Again, this updating may not provide any linking with other actions performed using the same card.

Next, we discuss the implementation of a loyalty card based on anonymous credentials. We first present a solution in an ideal credential system with optimal unlinkability. This construction is then tested against both Brands' system and the CL system.

4.2 Implementation in an ideal credential system

Let P be a credential system with ideal unlinkability features. More specific, different shows of the same credential in P cannot be linked to each other nor to their issue protocol. Also, a credential's revocation is based on its show protocol (which may be part of an update protocol) and does not introduce linkabilities with its other show protocols nor with its issue protocol.

Figure 5: loyalty card implementation in an ideal credential system

| parameters: | |
|--------------------|--|
| \mathcal{C} | customer |
| \mathcal{S} | shop owning keypair $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ for credential issuance |
| P | ideal credential system P |
| z | points to be added to the card |
| t | threshold balance: when the card balance exceeds t , the customer retrieves a discount and t points are deducted from the card |

protocol 1 - retrieving the card:

$\mathcal{C} \leftrightarrow \mathcal{S}$: payment, optional provision of contact information
 $\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{getCred}(P, pk_{\mathcal{S}}, x_1, (X_{\mathcal{I}} = \{1\} \wedge x_1 = 0)) \rightarrow Cred$

protocol 2 - showing the card (without change of balance):

$\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{showCred}(P, Cred, null)$

protocol 3 - updating the card (discount retrieval):

$\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{updateCred}(P, Cred, (x_1 > t), D = \{1\} \wedge \delta_1 = -t) \rightarrow Cred'$
 $\mathcal{C} \leftrightarrow \mathcal{S}$: discount retrieval
 $\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{revokeCred}(P, Cred)$

protocol 4 - updating the card (buy items):

$\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{updateCred}(P, Cred, null, D = \{1\} \wedge \delta_1 = z) \rightarrow Cred'$
 $\mathcal{C} \leftrightarrow \mathcal{S}$: $\text{revokeCred}(P, Cred)$

An implementation of a loyalty card satisfying the anonymity requirements is described in figure 5. The card itself is a credential $Cred$ issued by the shop and containing only one attribute x_1 . This attribute represents the card's balance and is initialized to zero during the registration protocol. Showing the card is now the same as showing the credential $Cred$. To retrieve a discount, \mathcal{C} performs a credential update protocol with the shop. He proves that its balance x_1 exceeds a threshold value t and then retrieves a new credential with new balance $x'_1 = x_1 - t$. Also, to prevent misuse, the old credential $Cred$ is revoked. Finally, when items are bought, a credential update protocol is executed in which the newly acquired points are added to x_1 . The old credential is again revoked.

In general, a visit to the shop is a combination of buying items and retrieving discounts. To achieve this, the credential updates in the buy or discount protocols can be combined into one credential update and one revocation. Also,

as only a card owner knows his card's balance, it is he who must initiate the discount retrieval protocol.

4.3 Implementation using Brands credentials

Using the techniques proposed by Brands, the solution described in section 4.2 introduces unwanted linkabilities. More specific, if the user updates his card and then shows it a number of times before updating the card a second time, these shows will be linkable to each other and to the second of the two update protocols. This is due to the use of credential show and revocation protocols, which both require the customer to provide his unique public credential key to the shop.

Luckily, this problem can be fixed easily by replacing the credential show in protocol 2 with a credential update and a subsequent revocation of the old credential. The resulting protocol is described below.

protocol 2:

| |
|---|
| $C \leftrightarrow S$: updateCred(<i>Brands</i> , <i>Cred</i> , <i>null</i> , $S = \{1\}$) \rightarrow <i>Cred'</i> $C \leftrightarrow S$: revokeCred(<i>Brands</i> , <i>Cred</i>) |
|---|

4.3.1 Evaluation

Different uses of the same loyalty card are not linkable to each other. When retrieving a loyalty card, the issued credential cannot be linked towards its issue protocol. Moreover, each credential is shown only once (namely during the credential's update protocol) and then immediately revoked and replaced by a new credential. This new credential is based on the old credential but not linkable towards it. Also, the attributes encoded into the credential are never revealed and hence cannot be used for linking purposes.

An important consideration in the development of Brands' system was to achieve extreme efficiency in the show and the issue-protocols. As a consequence, the main performance bottleneck are the credential revocation checks performed online. Whenever a credential is updated, its old public key is added to a revocation list. After a while, this list will become very long and tedious to handle. A solution to this problem is to give each credential only a limited lifetime, encoded as an expiration date attribute. The revocation list then only consists of credentials which have not yet been expired. A drawback of this approach is that customers are forced to use their loyalty card on a regular basis in order not to let it expire. Also, it must be investigated how this can be implemented without introducing unwanted linkabilities (e.g due to revealing expiration dates). Another solution is to discard any revocation and instead use a one-show credential with a unique identifying attribute. During a credential update, the one-show property of this credential is renewed but the unique attribute remains the same. This allows for off-line double-spending detection, resulting in the retrieval of the identifier. This identifier can then be blacklisted, prohibiting its

owner to use any of its (wrongfully) obtained credentials. Furthermore, disclosure of the identifier also provides a link towards the card retrieval protocol, and hence towards its owner's identity. Therefore, legal actions against the attacker can be taken.

4.4 Implementation in the CL system

An implementation of the application described in section 4.2 again introduces unwanted linkabilities. Different consecutive updates of the same loyalty card can be linked, as well as the card retrieval protocol and the first card update ever executed. The source of these problems is the fact that a credential's revocation introduces a link with its issue protocol (which may be incorporated into a credential update).

A solution for this problem is described in figure 6. The solution makes no use

Figure 6: loyalty card implementation in the CL system

| | |
|---|--|
| parameters: | |
| \mathcal{C} | customer |
| \mathcal{S} | shop owning keypair $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ for credential issuance |
| BL | credential blacklist |
| z | points to be added to the card |
| t | threshold balance |
| protocol 1 - retrieving the card: | |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : payment and identification |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : $\text{getCred}(CL, pk_{\mathcal{S}}, (x_1, x_2, x_3, x_4), (X_{\mathcal{I}} = \{1\} \wedge X_{\mathcal{J}} = \{2, 3, 4\} \wedge \text{dblsp-offl}(x_2, x_3, x_4) \wedge x_1 = 0)) \rightarrow Cred$ |
| protocol 2 - showing the card (without change of balance): | |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : $\text{updateCred}(CL, Cred, x_3 \notin BL \wedge \text{dblsp-offl}(x_2, x_3, x_4), R_{\mathcal{J}} = \{2, 4\}) \rightarrow Cred'$ |
| protocol 3 - updating the card (discount retrieval): | |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : $\text{updateCred}(CL, Cred, (x_1 > t \wedge \text{dblsp-offl}(x_2, x_3, x_4) \wedge x_3 \notin BL), (D = \{1\} \wedge R_{\mathcal{J}} = \{2, 4\} \wedge \delta_1 = -t)) \rightarrow Cred'$ |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : discount retrieval |
| protocol 4 - updating the card (buy items): | |
| $\mathcal{C} \leftrightarrow \mathcal{S}$ | : $\text{updateCred}(CL, Cred, (\text{dblsp-offl}(x_2, x_3, x_4) \wedge x_3 \notin BL), (D = \{1\} \wedge R_{\mathcal{J}} = \{2, 4\} \wedge \delta_1 = z)) \rightarrow Cred'$ |

of credential revocations. Instead, one-show credentials are used in combination with off-line double spending detection. A credential now has four attributes. Attribute x_1 represents the balance of the card and is initially set to zero. Next to this, attributes x_2, x_3 and x_4 are used to implement the credential's one-show property. Whenever a credential is updated, the one-show property of this credential is renewed by resetting its attributes x_2 and x_4 to new random values. However, the value x_3 will remain the same in all credentials belonging to the same loyalty card. When double-spending is detected, x_3 can be reconstructed. Remember that a commitment g^{x_3} was given to the shop during the retrieval of the loyalty card. Therefore, this value can be used to identify the double spender and to take appropriate actions. Moreover, x_3 is put on a blacklist BL , prohibiting the card owner from using his (wrongfully) obtained credentials ever again. Note that honest customers can easily prove in zero knowledge that their attribute x_3 is not on a blacklist. This can for example be done by creating a commitment $g^{x_3}h^r$ and by proving that the committed value is the same attribute as encoded into the shown credential and that it is different from any of the elements on the blacklist.

4.4.1 Evaluation

As long as the user does not double-spend any of his credentials, different uses of his loyalty card cannot be linked. This is a result of the unlinkability property of a credential update protocol. Moreover, all unique credential attributes that must be revealed (e.g. for double spending detection purposes), will only be revealed once and hence do not introduce any additional linkabilities. As a credential is one-show, it must not be revoked. However, customers must now provide double spending information and prove that their credential attribute x_3 is not on the blacklist. It is assumed that the blacklist will not be very large and that the protocols can still be executed efficiently.

5 Conclusions

In this paper, a high-level overview of anonymous credential systems was presented. Particular attention was paid to the two most elaborated systems to date; the system proposed by Brands on the one hand and the system developed by Camenisch et al on the other hand. Both systems were compared based on the used cryptographic techniques and their linkability properties. Afterwards, an extension was proposed for both systems to handle anonymous updating of credentials. This extension was then evaluated based on a case study implementing a loyalty card.

References

- [1] Alessandro Acquisti. Anonymous credentials through acid mixing. draft, 2003.

- [2] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.
- [3] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.
- [4] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.
- [5] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, pages 56–72, 2004.
- [6] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, pages 410–424, 1997.
- [7] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
- [8] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO*, pages 118–167, 1986.
- [9] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.
- [10] Lidong Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, pages 232–243, 1995.
- [11] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *CRYPTO*, pages 328–335, 1988.
- [12] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, pages 125–142, 2002.
- [13] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199, 1999.
- [14] Eric R. Verheul. Self-blindable credential certificates from the weil pairing. In *ASIACRYPT*, pages 533–551, 2001.