Solving Revocation with Efficient Update of Anonymous Credentials

Jan Camenisch[1], Markulf Kohlweiss[2], and Claudio Soriente[3]

[1] IBM Research – Zurich, Switzerland, jca@zurich.ibm.com
[2] KU Leuven, Belgium, markulf.kohlweiss@esat.kuleuven.be
[3] Universidad Politécnica de Madrid, Spain, csoriente@fi.upm.es

Abstract. Anonymous credential system promise efficient, ubiquitous access to digital services while preserving user privacy. However, their diffusion is impaired by the lack of efficient revocation techniques. Traditional credential revocation measures based on certificate revocation lists or online certification authorities do not provide privacy and can not be used in privacy-sensitive contexts. Revocation techniques specifically geared towards anonymous credential systems exhibit involved, interactive protocols – for the credential issuer as wells as users – to update information so that users can prove that their credential is still valid, e.g., not included in a revocation list. Some techniques put additional computational burden on the credential consumers. In this paper we introduce a novel, non-interactive technique to update valid credentials. With the proposed protocol, credential issuers can update valid credentials off-line and publish a small per-credential update value on a public bulletin-board. Users can later download their values and re-validate their credentials. Our solution outperforms all prior solutions in terms of communication and computational costs for the users and credentials consumers and the issuer's effort is comparable to the best prior proposals.

1 Introduction

The increasing number of ubiquitous digital services calls for efficient and pervasive means of authentication. User-centric identity management solutions like for instance Cardspace [1] do not only provide such an authentication mechanism, but also allow for the exchange of user attributes. To promote a global deployment of such systems and in order to maximize their benefit for democratic societies, authentication and authorization systems must offer a good balance between security, privacy, and performance. Anonymous credential systems as introduced by Chaum [24] offer strong authentication and the best possible privacy protection. The recent efficient realizations such as *idemix* [15] and *U-Prove* [10] are well suited to be used in practice even when using smart cards as authentication tokens [6].

In an anonymous credential system, the credential issuer provides a user with credentials that certify her attributes and permissions. The issued credentials allow the users in turn to perform transactions in which they disclose only the minimum amount of information required to obtain a service. Moreover, credential issuers do not learn which certified information are shown to which credential consumers, and issuers and consumers cannot link any transactions.

When using credentials to access a service it is of course crucial to ensure their validity and the information they carry. In particular, the support for revocation is essential for any credential or certification system, independent of what privacy protecting features it offers. There are many reasons why a credential needs to be revoked. The user might have lost her right to carry the credential, the secret key underlying the credential might have been compromised, or just because the attributes stated in the credential became outdated. Also, sometimes the application scenario might require a *rich revocation semantic* where a credential might only need to be "partially revoked": for instance, an expired European passport can still be used to travel within Europe but not to travel to the USA, or a driver's license revoked because of speeding could still be valid to prove the owner's age or address. Thus the validity checks that need to be done and therefore the means to use for revocation depend on the particular application scenario.

A possible solution to revocation in the case of non-anonymous credentials is to "blacklist" all serial numbers of revoked credentials in a so-called *certificate revocation list* [26] that can be queried on- or off-line. This solution does not work as such for anonymous credentials, as revealing a unique serial number of a credential would violate the unlinkability requirement. However, the general principle of publishing a list of all valid (or invalid) serial numbers can still work if, rather than revealing the serial number of their credential, users leverage the minimum disclosure feature of anonymous credentials to prove that it is among the list of valid serial numbers, i.e., that this number is not among the invalid ones. A number of protocols that work along these lines have been proposed [8, 12, 13, 30, 33] where the solution by Nakanishi, Fujii, Hira and Funabiki [30] seems to be the most elegant one.

A solution inspired by revocation lists is the use of so-called dynamic accumulators [18, 16]. Here, all valid serial numbers are accumulated (i.e., compressed) into a single value that is then published. In addition, dynamic accumulators provide a mechanism that allows the user to prove that the serial number of her credential is contained in the accumulated value. Whenever a credential is revoked, a new accumulator value is published that no longer contains the revoked serial number. Accumulator based schemes require, however, that users keep track of the changes to the accumulator to be able to execute their validity proofs. Camenisch, Kohlweiss and Soriente [16] proposed another accumulator where updates only require multiplications; moreover, computing the credential update information for the users can be performed by any party as it requires no secrets. They achieve this at the cost of a very large state, linear in the overall number of issued credentials. Moreover, accumulator-based solutions allow only to invalidate a credential as a whole and do not enable a rich revocation semantic for scenarios where partial revocation is required.

A common drawback of the solutions described so far is that they all make proving and verifying ownership of credentials less efficient (typically about a factor of 2 or worse), as not only possession of the credential has to be proven but also that it is still valid w.r.t. the revocation list/accumulator.

Another solution to revocation of credentials is to limit their lifetime by means of an expiration date and periodically re-issue non-revoked credentials. Here credentials are made valid only for a specific period of time (epoch), such as, only for a week, a couple of days, or hours, depending on the revocation requirements. This requires of course

that the credentials are re-issued periodically. As for anonymous credentials, issuing is an interactive protocol between the user and the issuer, this puts quite a burden on the infrastructure, not only in terms of bandwidth and computational power, but also in terms of availability. Indeed, an issuing of credentials such as electronic ID cards does typically not happen via the Internet but only in secured environments (as to protect the signing key) and often involves physical interactions with the user such as visiting a postal office.

In this paper we study to what extent existing credential systems allow for a non-interactive update of credentials. The issuing protocol of an anonymous credential systems typically consists of a protocol between the user and the issuer at the end of which the user gets a signature on a number of attributes, some of them chosen by and secret to the user and some of them chosen by the issuer. The idea we follow here is that the users and the issuer need to run an initialization protocol only once and thereafter the issuer can just update some values and publish them. Users can then retrieve these values and then recompute their credentials to make them valid again for the new time period. In fact, the period for which a credential is valid is only one of the attributes that a credential can hold; the issuer might want to update other attributes as well and enable richer revocation semantic. Our solution has the advantage that the verifier does not need to check any revocation lists and furthermore that the showing and verification of credentials are as efficient as possible, i.e., there are *no* extra work or space incurred by enabling revocation. Moreover the costs for updating credentials are minimal for users and are comparable to other solutions for the issuer. In fact, the issuer can (pre-) compute the update off-line and then periodically published the update values.

Performance and tradeoffs. Different applications have very diverse revocation requirements. The number of total users, the ratio of revoked users to unrevoked ones, the frequency of credential use, and the speed with which revocation has to take effect are just some of the parameters that influence the design of a revocation system for anonymous credentials. In order for the system to scale, the issuer must be able to handle a large number of users. At the same time, computational resources of user devices may be limited.

Our solution does not support immediate revocation or very short epochs (e.g., one hour) as it requires the issuer to provide credential updates for all non-revoked user. Accumulator-based revocation solutions are better suited for short revocation epochs as the issuer is not required to provide per-user updates, i.e., each non-revoked user can update its own witness. However, in application scenarios with infrequent credential usage such as the Belgian electronic identity card (eID) scenario with large number of issued (2,25 million per year) and revoked users (375.000 per year)[4] witness updates become exorbitantly expensive, e.g., 10 minutes for the CL accumulator [18], according to Lapon et al. [29].

While lacking the feature of immediate revocation, our solution only requires the user to download a short public credential update value and allows for rich revocation semantic at no additional cost for the show protocol. Hence, it is currently the most suitable system for the large scale deployment of anonymous credential systems.

[4] See `http://godot.be/eidgraphs`.

As validity period based revocation mechanisms cannot revoke credentials immediately, they can be combined with accumulator-based solutions for time-critical applications such as for instance passport control. In the example for the Belgian eID scenario, one could set the validity period to a day and use accumulators for immediate revocation. Thus, users would have to process about 1000 revocation updates per day while the computational load of the issuer to compute credential updates is still feasible. In addition, for some less critical uses of the eID, the verifier might not have to check for immediate revocation and hence relieve the user of the accumulator-proof in the show protocol.

Organization. Instead of considering a whole credential system, we first isolate the problem by looking at the core building block of many anonymous credential schemes, i.e., a signature scheme with efficient protocols [19]. We recall this and other cryptographic building blocks in Section 2. In particular we look at the issuing protocol of these signatures. In Section 3 we propose a new mechanism for the issuing of such signatures: it consists of an interactive part run once and a non-interactive part that can be repeated arbitrarily many times and that allows the issuer to change the messages (attributes) of the resulting signature to their current values. We give a definition of these protocols and procedures and their security requirements in Section 3.1. We then provide a sample construction of these protocols for a signature scheme based on bilinear maps in Section 3.2. In Section 4 we discuss how our new protocols can be used to construct an anonymous credential system with efficient revocation and attribute updates. Finally, in Section 5 we discuss for which other signature and credential schemes similar constructions can be developed. We conclude in Section 6.

2 Preliminaries

In this section we recall the cryptographic tools used by our scheme. After discussing efficient zero-knowledge proofs for prime order groups, we look at signature and commitment schemes that operate in the same setting. In particular, aforementioned zero-knowledge proofs will allow us to prove possession of a signature and to prove that a blindly issued signature signs a committed messages.

2.1 Discrete-Logarithm-Based Zero-Knowledge Proofs for Prime Order Groups

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [32], (2) proof of knowledge of equality of some element of representations different elements [25], (3) proof that a commitment opens to the product of two other committed values [21, 23, 9], and also (4) proof of the disjunction or conjunction of any two of the previous [28].

When referring to the proofs above, we will follow the notation introduced by Camenisch and Stadler [22] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^{\alpha} h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta}\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = g^{\alpha}h^{\beta}$ and $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta}$ holds*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that the some elements in the representation of y and $\tilde{y}$ are equal.) The convention is that variables in parenthesis, such as "(α, β, δ)", denote quantities of which knowledge is being proven, while all other values are known to the verifier. For prime-order groups which include all groups we consider in this paper, it is well known that there exists a knowledge extractor which acts as a verifier and can extract these quantities from a successful prover, if the latter can be rewinded. Also, these proofs can all be done efficiently (in four rounds and $O(k)$ communication, where k is a security parameter) by using the transformation by Cramer, Damgård and MacKenzie [27].

2.2 CL-Signature Schemes

A CL-signature scheme CLS [19] extends a conventional signature scheme and consists of five procedures $(\mathsf{KGen}, \mathsf{CLSig}, \mathsf{CLSVer}, \mathsf{CLSProof}, \mathsf{CLSPrVer})$. The procedure KGen generates the public and secret key of the signer, CLSig produces a signature σ on a block of messages $m_1, \ldots, m_n$ on input the secret key, and CLSVer outputs 1 iff σ is a valid signature on $m_1, \ldots, m_n$ w.r.t. the signer's public key. Finally, $(\mathsf{CLSProof} \leftrightarrow \mathsf{CLSPrVer})$ is an interactive protocol where a user can prove to a verifier knowledge of a valid signature on some message $m_1, \ldots, m_n$ such that the verifiers does not learn any information about the signatures and messages apart from the set $\{m_j\}_{j \in R}$, where $R \subset \{1 \ldots n\}$ is arbitrarily chosen by the user. The security requirements are that the signature scheme be unforgeable and that the $(\mathsf{CLSProof} \leftrightarrow \mathsf{CLSPrVer})$ be a zero-knowledge proof of knowledge.

Camenisch and Lysyanskaya have presented a scheme secure under the Strong RSA assumption [19], one under the LRSW assumption [20], and one that is based on the Boneh, Boyen and Shacham [7] group signature scheme under the Strong Diffie-Hellman assumption [20]. In the following we described a variant of the latter that was proposed and proved secure by Au, Susilo and Mu [2].

A CL-signature scheme based on the Au et al. signature scheme. The signature scheme assumes a non-degenerate bilinear map $\hat{e} : G \times G \rightarrow G_T$ of prime order q with generators $h, h_0, h_1, \ldots, h_n$, where n is a system parameter. The signer's secret key is $x \in \mathbb{Z}_q$ while the public key is $y = h^x$.

A signature on messages $m_1, \ldots, m_n \in \mathbb{Z}_q$ is a tuple $(A, r, \hat{r}, s)$ where $r, s \xleftarrow{\$} \mathbb{Z}_q$ are values chosen at random by the signer. The value $\hat{r} \in \mathbb{Z}_q$ is a value that can be chosen at random by the user in an interactive issuing protocol. For non-interactive signature generation, i.e., the CLSig procedure, we assume that $\hat{r} = 0$. The original signature scheme aggregates r and $\hat{r}$ into one value. We keep the two values separate to ease exposition in the following protocols. The value A is computed by the signer as $A = (hh_0^{r+\hat{r}} h_1^{m_1} \cdots h_n^{m_n})^{1/(x+s)}$. A signature is verified by checking if $\hat{e}(A, h^s y) = \hat{e}(hh_0^{r+\hat{r}} h_1^{m_1} \cdots h_n^{m_n}, h)$ holds.

We now show how to implement the $(\mathsf{CLSProof} \leftrightarrow \mathsf{CLSPrVer})$ protocol. In this protocol the user proves knowledge of a signature on messages $m_1 \ldots m_n$ but only reveals an arbitrary subset $\{m_j\}_{j \in R},\ R \subset \{1 \ldots n\}$ to the verifier. Given a signature

$(A, r, \hat{r}, s)$ on messages $m_1 \ldots, m_n \in \mathbb{Z}_q$, we want to prove that we indeed possess such a signature. To this end, we need to augment the public key of the signature with values $u, v \in G$ such that $\log_h u$ and $\log_h v$ are unknown. Proving knowledge of a signature can be done by choosing random values $w, w' \stackrel{\$}{\leftarrow} \mathbb{Z}_q$, computing $\tilde{A} = Au^w$, $B = v^w u^{w'}$ and executing the following proof of knowledge:

$$PK\{(\alpha, \beta, s, w, w', \{m_j\}_{j\in\{1\ldots n\}\setminus R}, r') : B = v^w u^{w'} \ \wedge \ 1 = B^{-s} v^{\alpha} u^{\beta} \ \wedge \\ \frac{\hat{e}(\tilde{A}, y)}{\hat{e}(h\prod_{j\in R} h_j^{m_j}, h)} = \hat{e}(\tilde{A}^{-s} u^{\alpha} h_0^{r'} \prod_{j\in\{1\ldots n\}\setminus R} h_j^{m_j}\ , h)\hat{e}(u, y)^w\} \ ,$$

where $r' = r + \hat{r}$, $\alpha = sw$, and $\beta = sw'$.

Let us explain this proof protocol. The first statement proves the prover's knowledge of values w and w' such that $B = v^w u^{w'}$. The next statement asserts the prover's knowledge of values α, β, and s such that $\alpha = sw$ and $\beta = sw'$. Let us consider the last line. It asserts the prover's knowledge of further values $\{m_j\}_{j\in\{1\ldots n\}\setminus R}$ such that

$$\begin{aligned}\hat{e}(\tilde{A}, y) =& \hat{e}(h \prod_{j\in R} h_j^{m_j}, h)\hat{e}(\tilde{A}^{-s} u^{\alpha} h_0^{r'} \prod_{j\in\{1\ldots n\}\setminus R} h_j^{m_j}\ , h)\hat{e}(u, y)^w \\ =& \hat{e}((\frac{u^w}{\tilde{A}})^s h h_0^{r'} \prod_{j=1}^{n} h_j^{m_j}\ , h)\hat{e}(u, y)^w\end{aligned}$$

holds, where we have made use of the relation $\alpha = sw$. We can further reformulate this equation into the following one

$$\hat{e}(\frac{\tilde{A}}{u^w}\ , y)\hat{e}((\frac{\tilde{A}}{u^w})^s\ , h) = \hat{e}((\frac{\tilde{A}}{u^w})^{s+x}\ , h) = \hat{e}(h h_0^{r'} \prod_{j=1}^{n} h_j^{m_j}\ , h) \ ,$$

where x is the secret key of the signer. Thus we must have

$$(\frac{\tilde{A}}{u^w})^{s+x} = h h_0^{r'} \prod_{j=1}^{n} h_j^{m_j} \ ,$$

i.e., that the prover knows a signature $(\tilde{A}u^{-w}, r', s)$ on the messages $m_1, \ldots, m_n$.

It was proved by Au et al. [2] that the above signature is unforgeable under adaptively chosen message attack if the Q-SDH assumption [7] holds, where Q is the number of signature queries. The authors also showed that the associated proof of knowledge is perfect honest-verifier zero-knowledge.

2.3 Commitment Scheme

A *commitment scheme* is a two-phase scheme that allows a user to *commit* to a hidden value, while preserving the ability of the user to *reveal* the committed value at a later stage. The standard definition of a non-interactive commitment scheme consists of a

setup algorithm ComSetup, and an algorithm Com that is used both in the commit and reveal stage. $\mathsf{ComSetup}(1^k)$ outputs public parameters $params_{\mathsf{Com}}$ for the commitment scheme. $\mathsf{Com}(params_{\mathsf{Com}}, x, open)$ is a deterministic algorithm that computes C, a commitment to x, using randomness $open$. One opens a commitment C by revealing x and $open$ and verifying that $\mathsf{Com}(params_{\mathsf{Com}}, x, open) = C$.

A secure commitment scheme is *hiding*: the value committed to must remain undisclosed until the reveal stage, and *binding*: the only value that may a commitment can be opened to is the one that was chosen in the commit stage. In our protocols we make use of a commitment scheme that is computationally binding and perfectly hiding:

Definition 1 (Computational Binding). *For all probabilistic polynomial time (p.p.t.) algorithms that on input* $params_{\mathsf{Com}} \leftarrow \mathsf{ComSetup}(1^k)$ *output* $x, x', open, open'$, $x \neq x'$*, the probability that* $\mathsf{Com}(params_{\mathsf{Com}}, x, open) = \mathsf{Com}(params_{\mathsf{Com}}, x', open')$ *is a negligible function* ν *in* k*.*

Definition 2 (Perfectly Hiding). *Let* U_k *be the uniform distribution over the opening values under public parameters* $params_{\mathsf{Com}} \leftarrow \mathsf{ComSetup}(1^k)$*. A commitment scheme is* perfectly *hiding if for all* $x \neq x'$ *the probability ensembles* $\{\mathsf{Com}(\mathsf{ComSetup}(1^k), x, U_k)\}_{k\in\mathbb{N}}$ *and* $\{\mathsf{Com}(\mathsf{ComSetup}(1^k), x', U_k)\}_{k\in\mathbb{N}}$ *are equal.*

Pedersen commitments. We use the perfectly hiding commitment scheme proposed by Pedersen [31], that is binding under the discrete logarithm (DL) assumption. For the parameters $params_{\mathsf{Com}}$ we will reuse generators u, v of a group G of prime order q from the CL-signature scheme's public key. These values fulfill the property that $\log_u(v)$ is unknown. A commitment C to $x \in \mathbb{Z}_q$ is generated by choosing at random $open \xleftarrow{\$} \mathbb{Z}_q$ and computing $C = \mathsf{Com}(params_{\mathsf{Com}}, x, open) = u^x v^{open}$. The commitment is opened by revealing x and $open$.

In the issuing protocol we also use a generalized of Pedersen commitments computed as $C = h_0^{open} h_1^{x_1} \cdots h_n^{x_n}$ that allows to commit to multiple values.

3 Issue Protocol for CL-Signatures with Updates

We formalize the security properties required from a CL-signature scheme with updates, and give an exemplary construction based on the Au et al. signature scheme.

3.1 Definitions

Let $\mathsf{CLS} = (\mathsf{KGen}, \mathsf{CLSig}, \mathsf{CLSVer}, \mathsf{CLSProof}, \mathsf{CLSPrVer})$ be a secure CL-signature scheme and let $\mathsf{C} = (\mathsf{ComSetup}, \mathsf{Com})$ be a secure commitment scheme. *A blind issuing and update scheme* for CLS and C consists of five additional procedures SKeygen, SObtSig, SIssSig, SIssUpd, and SObtUpd that are defined as follows.
Let ℓ be the number of blindly signed messages. We write $m_{1..n}$ as a shorthand for $m_1, \ldots, m_n$, similarly for $m_{1..\ell}$,$m_{\ell+1..n}$, $open_{1..\ell}$ and $C_{1..\ell}$.

$\mathsf{SKeygen}(1^k)$. This procedure combines the functions of KGen and $\mathsf{ComSetup}$. On input the security parameter k, the algorithm generates the secret and public keys for the signature scheme and the parameters for the commitment scheme. It then augments these keys with all the parameters needed for the issue, and update procedures. It outputs the augmented secret sk_I and public key pk_I of the issuer. The latter also includes the commitment parameters $params_{\mathsf{Com}}$.

$\mathsf{SObtSig}(pk_I, m_{1..n}, open_{1..\ell}) \leftrightarrow \mathsf{SIssSig}(sk_I, C_{1..\ell}, m_{\ell+1..n})$ is a protocol between the user and the issuer. Before running the protocol, the user commits to the messages $m_1, \ldots, m_\ell$ that are to be signed blindly. The opening information $open_{1..\ell}$ is part of the user's input, while the commitments $C_{1..\ell}$ are part of the issuer's input. The user's part $\mathsf{SObtSig}$ outputs the signature σ on messages $m_1, \ldots, m_n$, and the issuer's part $\mathsf{SIssSig}$ outputs the signature state $state_\sigma$ that will be later used to update signatures.

$\mathsf{SIssUpd}(sk_I, state_\sigma, m'_{\ell+1..n})$ on input the state value $state_\sigma$ for blinded messages $m_1, \ldots, m_\ell$, this procedure outputs a value $update_\sigma$ that allows to obtain an updated signature on messages $m_1, \ldots, m_\ell, m'_{\ell+1}, \ldots, m'_n$.

$\mathsf{SObtUpd}(pk_I, m_{1..n}, m'_{\ell+1..n}, \sigma, update_\sigma)$ combines the signature σ on messages m_1, $\ldots, m_n$ (those for which the user ran the issuing protocol initially) and the value $update_\sigma$ to obtain the signature σ' on messages $m_1, \ldots, m_\ell, m'_{\ell+1}, \ldots, m'_n$.

We require that the additional procedures do not damage the security of the original signatures scheme. We formulate this as the following two security requirements: *signer privacy* and *user privacy*. Informally, *signer privacy* requires that the user does not learn anything from interacting with the issuer via $\mathsf{SIssSig}$ and the updates from the issuer via $\mathsf{SIssUpd}$ other than signatures on the list of messages on which these protocols and procedures are run. In particular, this includes that the user shall not be able to forge signatures on other lists of messages.

The *user privacy* requirement states that the issuer does not learn anything about the messages $m_1, \ldots, m_\ell$ when interacting with via $\mathsf{SObtSig}$ with the user.

Signer privacy. The idea here is that no p.p.t. adversary $\mathcal{A}$ can tell if it is obtaining signatures from an honest issuer $\mathcal{I}$ running $\mathsf{SIssSig}$ and receiving signature updates via $\mathsf{SIssUpd}$ or whether it interacts with a simulator $\mathcal{S}$ with algorithms $\mathsf{SSimIssSig}$ and $\mathsf{SSimUpd}$ for issuing and updating signatures that does not know the issuer's secret key but only has access to a signing oracle. We formalize this using two experiments:

Experiment $\mathbf{Real}^{\mathsf{SP}}_{\mathcal{A}}(\mathbf{k})$ proceeds as follows:

1. Run $\mathsf{SKeygen}(1^k)$ and hand the secret and public keys to $\mathcal{A}$. Receive messages $m_1, \ldots, m_n$ and openings $open_1, \ldots, open_\ell$ from $\mathcal{A}$. Compute commitments $C_1 \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_1, open_1); \ldots; C_\ell \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_\ell, open_\ell)$. Run algorithm $\mathsf{SIssSig}(sk_I, C_{1..\ell}, m_{\ell+1..n})$ with $\mathcal{A}$. The experiment stores the value $state_\sigma$ output by $\mathsf{SIssSig}$.
2. Repeat until $\mathcal{A}$ stops with output b. Receive messages $m'_{\ell+1}, \ldots, m'_n$ from $\mathcal{A}$. Retrieve $state_\sigma$. Otherwise run $\mathsf{SIssUpd}(sk, state_\sigma, m'_{\ell+1\ldots n})$, hand $update_\sigma$ to $\mathcal{A}$.

Experiment $\mathbf{Simulated}^{\mathsf{SP}}_{\mathcal{A}}(\mathbf{k})$ proceeds as follows:

1. Run $\mathsf{SKeygen}(1^k)$ and hand the secret and public keys to $\mathcal{A}$. Receive messages $m_1, \ldots, m_n$ and openings $open_1, \ldots, open_\ell$ from $\mathcal{A}$. Compute commitments $C_1 \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_1, open_1); \ldots; C_\ell \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_\ell, open_\ell)$. Compute $\sigma \leftarrow \mathsf{CLSig}(sk_I, (m_{1..n}))$ and run $\mathsf{SSimIssSig}(\sigma, comm_{1..\ell}, m_{\ell+1..n})$ with $\mathcal{A}$. The experiment stores the output $state_{\mathcal{S}}$ of $\mathsf{SIssSig}$ and messages $m_1, \ldots, m_\ell$.
2. Repeat until $\mathcal{A}$ stops with output b. Receive messages $m'_{\ell+1}, \ldots, m'_n$ from $\mathcal{A}$. Compute $\sigma' \leftarrow \mathsf{CLSig}(sk_I, m_{1..\ell}, m'_{\ell+1..n})$ and run $\mathsf{SSimUpd}(\sigma', state_{\mathcal{S}}, m'_{\ell+1..n})$, hand $update_\sigma$ to $\mathcal{A}$.

The simulator is allowed to rewind the adversary. Let the adversary's advantage in distinguishing between the two experiments be $Adv^{\mathsf{SP}}_{\mathcal{A}}(k) = |Pr[\mathbf{Real}^{\mathsf{SP}}_{\mathcal{A},\mathcal{I}}(\mathbf{k}) = 1] - Pr[\mathbf{Simulated}^{\mathsf{SP}}_{\mathcal{A},\mathcal{S}}(\mathbf{k}) = 1]|$. Signer privacy requires that $Adv^{\mathsf{SP}}_{\mathcal{A}}(k)$ is a negligible function in k.

We have defined *signer privacy* in terms of the issue and update sequence of a single signature, but our definition is strengthened by the fact that the adversary is given the issuers secret key sk_I. A simple hybrid argument can be used to show that this definition implies privacy for many credentials as long as the signature issue protocols are executed sequentially.

User privacy. No p.p.t. adversary $\mathcal{A}$ can tell if it is issuing signatures to an honest user $\mathcal{U}$ running $\mathsf{SObtSig}$ or to a simulator $\mathcal{S}$ running $\mathsf{SSimObtSig}$ that does not know the users secret inputs. We formalize this using two experiments:

Experiment $\mathbf{Real}^{\mathsf{SP}}_{\mathcal{A}}(\mathbf{k})$ proceeds as follows:

1. Receive a signature public key pk_I, messages $m_1, \ldots, m_\ell$, and openings $open_1$, $\ldots$, $open_\ell$ from $\mathcal{A}$.
2. Run $\mathsf{SObtSig}(pk_I, m_{1..\ell}, open_{1..\ell})$ with $\mathcal{A}$. The experiment outputs the adversary's output b.

Experiment $\mathbf{Simulated}^{\mathsf{SP}}_{\mathcal{A}}(\mathbf{k})$ proceeds as follows:

1. Receive a signature public key pk_I, messages $m_1, \ldots, m_\ell$, and openings $open_1$, $\ldots$, $open_\ell$ from $\mathcal{A}$. Compute $C_1 \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_1, open_1); \ldots; C_\ell \leftarrow \mathsf{Com}(params_{\mathsf{Com}}, m_\ell, open_\ell)$.
2. Run $\mathsf{SSimObtSig}(pk_I, comm_{1..\ell})$ with $\mathcal{A}$. The experiment outputs the adversary's output b.

Again, the simulator is allowed to rewind the adversary. Let the adversary's advantage in distinguishing the two experiments be $Adv^{\mathsf{UP}}_{\mathcal{A}}(k) = |Pr[\mathbf{Real}^{\mathsf{UP}}_{\mathcal{A},\mathcal{U}}(\mathbf{k}) = 1] - Pr[\mathbf{Simulated}^{\mathsf{UP}}_{\mathcal{A},\mathcal{S}}(\mathbf{k}) = 1]|$. User privacy requires that $Adv^{\mathsf{UP}}_{\mathcal{A}}(k)$ is a negligible function in k.

Note that we require that only the user's input $m_1, \ldots, m_\ell$ be hidden from the issuer, but not necessarily the user's output σ. The reason that this is sufficient is that in actual applications (for example, in anonymous credentials), a user would never show σ in the clear; instead, she would just prove that she knows σ.

Definition 3. *We say that* $\mathsf{UCLS} = \mathsf{SKeygen}, \mathsf{CLSig}, \mathsf{CLSVer}, \mathsf{CLSProof}, \mathsf{CLSPrVer}, \mathsf{Com}, \mathsf{SKeygen}, \mathsf{SObtSig}, \mathsf{SIssSig}, \mathsf{SIssUpd}, \mathsf{SObtUpd}$) *is a secure CL-signature scheme with updates if the algorithms* ($\mathsf{SKeygen}, \mathsf{CLSig}, \mathsf{CLSVer}, \mathsf{CLSProof}, \mathsf{CLSPrVer}$) *constitute a secure CL-signature scheme, the algorithms* ($\mathsf{SKeygen}, \mathsf{Com}$) *constitute a secure commitment scheme, and* $\mathsf{SKeygen}, \mathsf{SObtSig}, \mathsf{SIssSig}, \mathsf{SIssUpd}, \mathsf{SObtUpd}$ *fulfill the* signer privacy *and* user privacy *properties.*

3.2 Construction

The main insight that leads to our construction is that issuing credential based on CL-signatures typically consists of two stages: 1) the user sends to the issuer some form of commitment to the messages that she wants to be included in the credential and 2) the issuer extends that commitment into one that covers all the message to be signed and then computes the signature of all these messages. As the second stage essentially consists only of computations by the issuer followed by sending the user the signature, this stage can be repeated any number of times with new messages chosen by the issuer and instead of sending the result on-line to the user, it can be provided as an update by any form of communication (e.g., provided for download at a website).

This approach of two stages is possible because for all the CL-signature schemes that we consider [17, 19, 20] and in particular for the signature scheme by Au, Susilo and Mu [2] on which we base our explicit construction, signing consists of computing a group element from a number of bases where the message to be signed are used as exponents. Hence, this group element can also be considered as a Pedersen commitment to all the message. This holds even for the group element computed by the user for the messages that are hidden from the issuer provided that the user proves to the issuer that she did do these computations correctly.

We describe a construction for CL-signatures with updates and prove the security of the issuing protocol and the update algorithms.

$\mathsf{SKeygen}(1^k)$. On input 1^k, pick a non-degenerate efficiently computable bilinear map $\hat{e} : G \times G \to G_T$ of prime order q with $G = \langle h \rangle$. Pick additional bases $h_0, h_1, \ldots, h_\ell, h_{\ell+1}, \ldots, h_n \xleftarrow{\$} G$. The signer's secret key is $x \xleftarrow{\$} \mathbb{Z}_q$ while the public key is $y = h^x$. Publish $pk_I = (q, G, G_T, e, h, h_0, h_1, \ldots, h_\ell, h_{\ell+1}, \ldots, h_n, y, u, v)$. The secret key sk_I includes the public key material and x. To speed up computation the issuer can choose values $x_1, \ldots, x_n \leftarrow \mathbb{Z}_q$ and compute $h_i = h^{x_i}$ for $i = 1..n$. This allows to compute a product $\prod_{i=1}^n h_i^{m_i}$ as $h^{\sum_{i=1}^n x_i m_i}$.

$\mathsf{SObtSig}(pk_I, m_{1..n}, open_{1..\ell}) \leftrightarrow \mathsf{SIssSig}(sk_I, comm_{1..\ell}, m_{\ell+1..n})$.

1. $\mathcal{U}$ picks $\hat{r} \xleftarrow{\$} \mathbb{Z}_q$, computes $P = h_0^{\hat{r}} \prod_{i=1..\ell} h_i^{m_i}$ and sends it to $\mathcal{I}$.
2. $\mathcal{U}$ engages with $\mathcal{I}$ in the following proof of knowledge to convince $\mathcal{I}$ that P is correctly formed.

$$PK\{(\hat{r}, m_{1..\ell}, open_{1..\ell}) \bigwedge_{i=1}^{\ell} C = \mathsf{Com}(params_{\mathsf{Com}}, m_i, open_i) \wedge P = h_0^{\hat{r}} \prod_{i=1..\ell} h_i^{m_i}\} .$$

3. $\mathcal{I}$ picks $s, r \xleftarrow{\$} \mathbb{Z}_q^*$, computes $A = (hPh_0^r \prod_{i=\ell+1}^n h_i^{m_i})^{1/(x+s)}$ and sends (A, r, s) to $\mathcal{U}$.

4. $\mathcal{I}$ outputs $state_\sigma = P$.
5. $\mathcal{U}$ outputs $\sigma = (A, r, \hat{r}, s)$.

$\mathsf{SIssUpd}(sk_I, state_\sigma, m'_{\ell+1..n})$. This algorithm is periodically run by $\mathcal{I}$ to update a signature with state $state_\sigma$. $\mathcal{I}$ proceeds with the following steps.

1. $\mathcal{I}$ picks $s', r' \xleftarrow{\$} \mathbb{Z}_q^*$, computes $A' = (hPh_0^{r_i} \prod_{i=\ell+1}^{n} h_i^{m'_i})^{1/(x+s)}$.
2. $\mathcal{I}$ outputs $update_\sigma = (A', r', s')$.

If the issuer chooses $h_i = h^{x_i}$ the computation of A only requires two exponentiation (or rather one two-base multi-exponentiation).

$\mathsf{SObtUpd}(pk_I, m_{1..n}, m'_{\ell+1..n}, \sigma, update_\sigma)$. Given a signature $\sigma = (A, \hat{r}, r, s)$ and $update_\sigma = (A', r', s')$ output $\sigma' = (A', r', \hat{r}, s')$, if $\mathsf{CLSVer}(pk_I, \sigma', m_{1..\ell}, m'_{\ell+1..n}) = 1$ and $\perp$ otherwise.

Theorem 1. *Under the Strong Diffie-Hellman assumption (that implies the Discrete Logarithm assumption), the algorithm above together with the Au et al. CL-signature scheme and the Pedersen constitute a secure CL-signature scheme with updates.*

Lemma 1. *The* $\mathsf{SIssSig}$ *and* $\mathsf{SIssUpd}$ *algorithms above together with the Au et al. CL signature scheme and the Pedersen commitment scheme fulfill the* signer privacy *property assuming the security of the zero-knowledge proof of knowledge and commitment scheme.*

Proof. Given a list of commitments $C_{1..\ell}$ messages $m_1, \ldots, m_n$ and a signature $\sigma = (A, \tilde{r}, 0, s)$ as input $\mathsf{SSimIssSig}$ simulates the adversaries view. Upon receiving the value P, it interacts with the adversary in a proof of knowledge. The adversary proves that she knows messages $m_1, \ldots, m_\ell$ corresponding to $C_{1..\ell}$ and the randomness $\hat{r}$ used to create P. The simulator uses the knowledge extractor of the proof of knowledge to obtain $\hat{r}$, and returns $(A, \tilde{r} - \hat{r}, s)$ to the adversary. The state $state_\mathcal{S}$ of the simulator corresponds to $\hat{r}$.

For each request to generate a signature update, $\mathsf{SSimUpd}$ receives messages $m'_{\ell+1}$, $\ldots, m'_n$ and a signature $\sigma' = (A', \tilde{r}', 0, s')$ as input. The simulator uses $\hat{r}$ to returns $update_\sigma = (A', \tilde{r}' - \hat{r}, s')$.

We proof using a sequence of games that $\mathbf{Real}_\mathcal{A}(\mathbf{k})$ and $\mathbf{Simulated}_\mathcal{A}(\mathbf{k})$ are indistinguishable.

Game 1 corresponds to the $\mathbf{Real}_{\mathcal{A},\mathcal{I}}(\mathbf{k})$ experiment.

Game 2 is the same as Game 1, but the knowledge extractor of the proof of knowledge is used to extract $\hat{r}, \tilde{m}_{1..\ell}, open_{1..\ell}$. If extraction succeeds proceed as in Game 1, otherwise abort. *The probability $\nu_1(k)$ to distinguish between Game 1 and Game 2 is bounded by the knowledge extraction error of the proof of knowledge protocol.*

Game 3 is the same as Game 3, except that the game aborts, if the values $\tilde{m}_{1..\ell}$ extracted from the proof of knowledge differ from the values $m_{1..\ell}$ output by the adversary. *The probability $\nu_2(k)$ to distinguish between Game 2 and Game 3 is bounded based on the security of the commitment scheme.*

Game 4 computes the response of the issuing protocol and all update protocols, by first computing a signature $\sigma = (A, \tilde{r}, 0, s)$ for messages $m_1, \ldots, m_n$ and $m_{1..\ell}$,$m'_{\ell+1,n}$ respectively, and then replying with $(A, \tilde{r} - \hat{r}, s)$. Game 4 is identically distributed to Game 3 and corresponds to experiment $\mathbf{Simulated}_{\mathcal{A},\mathcal{S}}(\mathbf{k})$.

The adversaries advantage in distinguishing the games is bounded by $Adv_{\mathcal{A}}(k) < \nu_1(k) + \nu_2(k)$.

Lemma 2. *The* SObtSig *algorithm above together with the Au et al. CL-signature scheme and the Pedersen commitment scheme fulfill the* user privacy *property.*

Proof. Given the issuers public key pk_I and commitments $comm_{1..\ell}$ as input, the simulator SSimObtSig picks a random value P. Then it uses the zero-knowledge simulator to interact with the adversary in the following proof protocol:

$$PK\{(\hat{r}, m_{1..\ell}, open_{1..\ell}) : \bigwedge_{i=1}^{\ell} C = Commit(params_{\mathsf{Com}}, m_i, open_i) \wedge P = h_0^{\hat{r}} \prod_{i=1..\ell} h_i^{m_i}\} .$$

As both the commitments $C_{1..\ell}$ and P are perfectly hiding Pedersen commitments, this is a proof of a true statement, and the simulation is perfect.

4 Anonymous Credential Systems with Efficient Revocation and Attribute Update

In this section we will show how to use CL-signatures with updates to design an anonymous credential system where credential revocation is accomplished through an efficient, non-interactive protocol for updating credentials. The considered scenario consists of three types of players:

A credential issuer ($\mathcal{I}$) that issues and manages anonymous credentials. *One or more credential verifiers* that provide services to users upon show of valid credentials. *A set of users* that anonymously obtain credentials from $\mathcal{I}$ and show them in a privacy preserving way to verifiers in order to access their services.

Several research papers describe how to construct anonymous credential schemes from CL-signatures. The efficient anonymous credential scheme of Camenisch and Lysyanskaya [17] made use of CL-signatures as an implicit building block that the same authors later formalized in [19]. Their basic system, however, does not support attributes. A non-interactive variant of such a credential system was also proposed by Belenkiy, Chase, Kohlweiss and Lysyanskaya in [5]. Bangerter, Camenisch and Lysyanskaya [3] describe a flexible anonymous certification framework that allows for blind issuing and selective show of credentials that certify multiple user and issuer chosen attributes. CL-signatures also form the basis for direct anonymous attestation (DAA) [14]. The DAA protocol makes use of a blindly certified user secret sk_U. This value never leaves the trusted platform module and protects the credential against theft and abuse.

We describe a credential system that combines the features in the above schemes. In addition, we allow for efficient revocation through the inclusion of time period information. The anonymous credential system uses the SObtSig $\leftrightarrow$ SIssSig protocol to issue a credential with the following information to the user: the users secret sk_U, the credential serial number id, the time period for which the credential is valid t, and d attributes $a_1, \ldots, a_d$ chosen by the issuer. The key sk_U is only known to the user and

is certified blindly. The update feature of our CL-signature scheme allows the issuer to publish update information for new time periods. As an added benefit, the issuer can update the users attribute. The latter allows rich revocation semantic as credential can be partially (i.e., only some of the credential attributes) revoked and/or updated.

More formally, our anonymous Credential System with efficient revocation and updates consists of the following algorithms:

$\mathsf{IssuerKeygen}(1^k)$ This algorithm is run once by $\mathcal{I}$ to setup system parameters. It runs $\mathsf{SKeygen}(1^k)$ to create sk_I and pk_I of a CL-signature with Updates scheme. It also outputs an empty set $state$ where to store issued credentials.

$\mathsf{UserKeygen}(1^k)$ This algorithm is run only once for each $\mathcal{U}$ before she interacts with the $\mathcal{I}$ to obtain any credential. $\mathcal{U}$ obtains her secret key sk_U and the corresponding public key pk_U that might be advertised as the user identity.

$\mathsf{ObtainCert}(\mathcal{U}(pk_U, sk_U), \mathcal{I}(pk_I, sk_I, a_{1..d}, state, t)$ In this protocol, $\mathcal{U}$ obtains a certified credential with unique serial number id. The latter is arbitrarily chosen by $\mathcal{I}$. For example, given $state$ as the set of all issued credentials, $\mathcal{I}$ might set id to the next available serial number.

1. The user commits to her secret key as $C = \mathsf{Com}(params_{\mathsf{Com}}, sk_U, open)$ and user sends her public key and the commitment to the issuer. The user does a proof of knowledge that the public key corresponds to the commitment. This provides the issuer with the guarantee that the credential will be issued to the correct user.
2. Now the issuer sends the attributes $id, t, a_1, \ldots, a_d$ to the user.
3. The user and the issuer run $\mathsf{SObtSig}(pk_I, sk_U, id, t, a_{1..d}, open) \leftrightarrow \mathsf{SIssSig}(sk_I, C, id, t, a_{1..d})$, respectively. Note that sk_U is the only blindly signed message, and $n = d + 3$. The user obtains the signature σ and the issuer obtains $state_\sigma$.
4. The issuer adds record $(1, id, state_\sigma, a_1, \ldots, a_d)$ to $state$; the first element of the record flags the credential as currently valid.
5. The users output is the certificate $cert = (\sigma, id, t, a_1, \ldots, a_d)$.

$\mathsf{InvalidateCerts}(state, id)$ This algorithm is periodically run by $\mathcal{I}$ to revise validity status of issued credentials. For each credential to be revoked, let id be its serial number, $\mathcal{I}$ replaces record $(1, id, state_\sigma, a_1, \ldots, a_d)$ in $state$ with record $(0, id, state_\sigma, a_1, \ldots, a_d)$; the first element of the record flags the credential as revoked.

$\mathsf{UpdateAttributes}(state, id, a'_{1..d})$ This algorithm is run by $\mathcal{I}$ before producing the updates of each valid credential. It is used to update credential attributes for valid credentials. The issuer replaces $(1, id, state_\sigma, a_1, \ldots, a_d)$ in $state$ with $(1, id, state_\sigma, a'_1, \ldots, a'_d)$ to reflect the changes to the credential attributes for the current time period.

$\mathsf{CertUpdate}(pk_I, sk_I, state, t)$ This algorithm is periodically run by $\mathcal{I}$ to update credentials that are still valid. For each record $(1, id, state_\sigma, a_1, \ldots, a_d)$ in $state$, the issuer runs $\mathsf{SIssUpd}(sk_I, state_\sigma, id, t, a_{1..d})$ and publishes the resulting update value together with the new attribute values as $update_{id,t} = (update_\sigma, a_1, ...a_d)$. Note that revoked credentials in $state$ do not get updated.

$\mathsf{ProveCert}(\mathcal{U}(sk_U, cert, update_{id,t}, R), \mathcal{V}(pk_I, t, R)$ This algorithm is run at time t by an user $\mathcal{U}$ and a verifier $\mathcal{V}$ before the latter grants any service to the former. At the

end of the protocol, $\mathcal{V}$ only learns that $\mathcal{U}$ has a credential issued by $\mathcal{I}$ with attributes $\{a_i\}_{i\in R}$ that is valid at time t. First, $\mathcal{U}$ parses $update_{id,t}$ as $(update_\sigma, a_1, ...a_d)$ and updates her credential running $\mathsf{SObtUpd}(pk_I, a_{1..d}, \sigma, update_\sigma)$. The updated certificate is $cert = (\sigma', id, t, a_1, \ldots, a_d)$. Where $t, a_1, \ldots, a_d$ correspond to the current time period and the updated attribute values. Later, she can show her credential an arbitrary number of time to any verifier. At each show, the user sends messages $\{a_i\}_{i\in R}$ to the issuer, and performs the following zero-knowledge proof of knowledge:

$$\mathsf{PK}\{(\sigma, sk_U, \{a_i\}_{i\in\{1,\ldots,d\}\setminus R}) : 1 = \mathsf{CLSVer}(pk_I, \sigma, sk_U, id, t, a_{1..d}) \wedge \ldots\} \ .$$

Note that in the above example, credentials are revoked as a whole. Partial revocation can be achieved using multiple flags per credential. To use the driving licence example of Section 1, each credential would have a flag to define credential validity for driving permissions and an additional flag to define its validity for owner identification purposes. Flags could be encoded in the credential and updated independently as required.

Security discussion. In [17] a secure anonymous credentials scheme is defined using an ideal functionality. The authors show that the extraction and zero-knowledge properties of the proof system and the unforgeability of the signature scheme guarantee that an adversary attacking the real world anonymous credential system cannot do more damage than an adversary interacting with the ideal functionality. The signer privacy and user privacy properties introduced by Belenkiy et al. [5] formalize the needed properties for the issuing protocol. In addition to what is shown in [5] we show that the non-interactive signature updates do not leak additional information about the issuers secret key. The credential show protocol is unchanged and relies on the security of the zero-knowledge proof of knowledge of signature possession.

5 Efficient Updates for Other Signatures and Anonymous Credential Schemes

The construction we give in this paper employs a signature scheme based on bilinear maps. There are however a number of other constructions for signature schemes with efficient protocols and constructions for group signatures and credential systems. In this section we discuss whether the approach of introducing validity time periods and publishing credential/signature update information also applies to other schemes.

CL-signatures. Camenisch and Lysyanskaya have proposed a number of different signature schemes that allows efficient proofs of knowledge of a signature. Apart from the one we have already used in our construction, they have proposed a scheme based on the strong RSA assumption [19] and one based on the LRSW assumption [20]. As all of these schemes follows the same principles, they can all be extended in the same way as what we did in our construction. We quickly sketch this for the widely used CL-signature scheme based on the strong RSA assumption:

The signature σ consists of a tuple (A, r, e) with $A = (hh_0^r h_1^{m_1} \dots h_n^{m_n})^{-e} mod\ \mathfrak{m}$, where $\mathfrak{m}$ is an RSA modulus. Similarly to our construction based on bilinear maps, the issue protocol starts with the user sending a value $P = h_0^{\hat{r}} h_1^{m_1} \dots h_\ell^{m_\ell}$ that is then extended by the issuer to compute a signature on blinded values $m_1, \dots, m_\ell$ and issuer chosen messages $m_{\ell+1}, \dots, m_n$. This last step can be repeated for different messages $m'_{\ell+1}, \dots, m'_n$ to implement the update.

Blind-signatures based schemes. The credential schemes by Brands [10, 11] employ a blind Schnorr signature scheme to achieve anonymity. This signature scheme uses hash functions in a crucial way to achieve unforgeability.

Conceptually, a Schnorr blind signature protocol consists of three steps. The *commitment* step, the *challenge* step, and the *response* step. The first and the last step are computed by the signer. To achieve blindness, the user (signature receiver) needs to compute the challenge as a hash value on the values of the commitment step and the user's public key $h' = g_0^{\alpha} \prod_{i=1}^{n} g_i^{m_i}$, that encodes his attributes (see Chapter 4 of [11]). The user then blinds (randomizes) the challenge before the signer can compute signature values in the final response step using its signing secret key. It thus seems inherent that the user needs to do this hash function computation for every signature and thus, signature updates cannot be done non-interactively. A solution that works partially is as follows. The user could prepare many blind signatures and then send them all at once to the signer. The signer could then finish the individual protocols as needed (e.g., one in each epoch). This, however, works only if all the messages (e.g., attributes of a credential) are fixed at the time the user prepares all these blind signatures. Thus, the signer would not be able to update any of the messages in the update phase which seems to be a severe limitation. Furthermore, the user would have to store all the blinding values of all the prepared blind signatures (or to regenerate them from a seed using a suitable pseudo-random function).

Other schemes. Belenkiy el al. [5, 4] have proposed so-called P-signatures that are based on bilinear maps and allow one to use Groth-Sahai non-interactive proofs for proving knowledge of a signature. However, issuing signatures in their schemes is highly interactive and it seems not possible to apply our approach to these schemes as they are now.

An approach that works for all interactive and non-interactive CL and P-signature schemes is to combine an interactively issued signature that contains the attribute values that should be blindly signed and the credential identifier id, with a plain signature that contains the same identifier, the time period t, and all issuer attributes. The disadvantage of this approach is that the prove protocol becomes twice as expensive, as the user now has to prove possession of two signatures.

When considering related schemes such as group signatures and identity escrow, we see that our approach can in general not be used as they do not have a means to include a validity time period identifier. However, many of them are constructed along the lines of using a CL-signature to sign a group member's secret key and then defining a group-signature to be a non-interactive proof of knowledge of a CL-signature by the group manager on a secret key. For these schemes, it is of course not hard to extend them such

that the group manager signs also a second message being an epoch identifier and hence our approach can be used.

6 Conclusion

Despite a growing concern for user privacy in a cyber-world, the diffusion of Anonymous Credential Systems is impaired by the lack of efficient protocols. Use of non-interactive protocols, rich revocation semantic and minimal overhead at show time are key features to enable the adoption of Anonymous Credential Systems in privacy-preserving scenario with large number of users.

In this paper we have introduced a signature scheme with updates that can be used in anonymous credential systems to enable efficient, semantically rich revocation. Our scheme allows for non-interactive credential update as well as partial revocation/update. Moreover, it enjoys no overhead in the show protocol to prove that a credential is non-revoked. Updates can be performed off-line and later published on a public bulletin board for users to download them. Also, users can miss an arbitrary number of updates, that is, the latest update to their original credential suffices to prove its possession. Compared to previous solutions for revocation, our approach is much more efficient for showing and verifying credentials (there is no additional cost), more flexible (it addresses even updates of attributes), and has a similar overhead for managing revocation status as previous solutions.

Acknowledgements. Markulf Kohlweiss was supported in part by IBBT, the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). This work was further supported in part by the European Commission through the ICT and IST programmes under the following contracts: ICT-216483 PRIMELIFE and ICT-216676 ECRYPT II. Claudio Soriente has been partially funded by the Spanish National Science Foundation (MICINN) under grant TIN2007-67353-C02 and by the Madrid Regional Research Council (CAM) under grant S2009/TIC-1692 and and by the European Commission under project NEXOF-RA (FP7-216446).

References

1. Windows cardspace. online, 2010. Available from `http://www.microsoft.com/windows/products/winfamily/cardspace/`, accessed in April 2010.
2. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.
3. Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In Scott C.-H. Huang, David McCallum, and Ding-Zhu Du, editors, *Security Protocols*, number 3957 in Lecture Notes in Computer Science, pages 20–42. Springer Verlag, 2006.
4. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2009.

5. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.
6. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In to appear, editor, *ACM Conference on Computer and Communications Security*, 2009.
7. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Verlag, 2004.
8. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 168–177. ACM, 2004.
9. Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 318–333. Springer Verlag, 1997.
10. Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates — Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.
11. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.
12. Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415. Springer, 2007.
13. Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *12th Australasian Conference on Information Security and Privacy, ACISP 2007*, pages 400–415, 2007.
14. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proc. 11th ACM Conference on Computer and Communications Security*, pages 225–234. acm press, 2004.
15. Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proc. 9th ACM Conference on Computer and Communications Security*. acm press, 2002.
16. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *12th International Conference on Practice and Theory in Public Key Cryptography, PKC 2009*, pages 481–500, 2009.
17. Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, 2001.
18. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer Verlag, 2002.
19. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2003.
20. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer Verlag, 2004.

21. Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer Verlag, 1999.
22. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.
23. Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.
24. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
25. David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.
26. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.
27. Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–372. Springer, 2000.
28. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.
29. Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Performance analysis of accumulator-based revocation mechanisms. In *Proceedings of the 25th International Conference on Information Security (SEC 2010)*, IFIP Conference Proceedings, page 12, Brisbane, AU, 2010. Springer-Verlag.
30. Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480. Springer, 2009.
31. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 1992.
32. Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
33. Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, 2007.