An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials

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

[1] IBM Research Zurich jca@zurich.ibm.com
[2] Katholieke Universiteit Leuven / IBBT markulf.kohlweiss@esat.kuleuven.be
[3] University of California, Irvine csorient@ics.uci.edu

Abstract. The success of electronic authentication systems, be it e-ID card systems or Internet authentication systems such as CardSpace, highly depends on the provided level of user-privacy. Thereby, an important requirement is an efficient means for revocation of the authentication credentials. In this paper we consider the problem of revocation for certificate-based privacy-protecting authentication systems. To date, the most efficient solutions for revocation for such systems are based on cryptographic accumulators. Here, an accumulate of all currently valid certificates is published regularly and each user holds a *witness* enabling her to prove the validity of her (anonymous) credential while retaining anonymity. Unfortunately, the users' witnesses must be updated at least each time a credential is revoked. For the know solutions, these updates are computationally very expensive for users and/or certificate issuers which is very problematic as revocation is a frequent event as practice shows.

In this paper, we propose a new dynamic accumulator scheme based on bilinear maps and show how to apply it to the problem of revocation of anonymous credentials. In the resulting scheme, proving a credential's validity and updating witnesses both come at (virtually) no cost for credential owners and verifiers. In particular, updating a witness requires the issuer to do only one multiplication per addition or revocation of a credential and can also be delegated to untrusted entities from which a user could just retrieve the updated witness. We believe that thereby we provide the first authentication system offering privacy protection suitable for implementation with electronic tokens such as eID cards or drivers' licenses.

Keywords: dynamic accumulators, anonymous credentials, revocation.

1 Introduction

The desire for strong electronic authentication is growing not only for the Internet but also in the physical world where authentication tokens such as electronic identity cards, driving licenses, and e-tickets are being widely deployed and are set to become a pervasive means of authentication. It has been realized that thereby the protection of the citizens' privacy is of paramount importance and hence that the principle of *data minimization* needs to be applied: any individual

should only disclose the minimal amount of personal information necessary for the transaction at hand. While privacy is of course not a major concern for the primary use of these tokens, e.g., for e-Government, it becomes vital for their so-called secondary use. For instance, when accessing a teenage chat room with an e-ID card, users should only have to reveal that they are indeed between, say, 10 and 16 years old but should not reveal any other information stored on the card such as birth date, name or address.

In the literature, there exist a fair number of privacy-preserving technologies that allow one to meet these requirements. These technologies include anonymous credential systems [1–3], pseudonym systems [4–7], anonymous e-cash [8–10], or direct anonymous attestation [11]. Almost all of these schemes exhibit a common architecture with certificate issuers, users (certificate recipients) and certificate verifiers: Users obtain a signature from an issuing authority on a number of attributes and, at later time, can convince verifiers that they indeed possess a signature on those attributes [12]. Individual transactions are anonymous and unlikable by default and users can select which portions of a certificate to reveal, which portions to keep hidden, and what relations between certified items to expose.

A crucial requirement for all authorization and authentication systems is that certificates issued can be later revoked, in case of unexpected events or malicious use of the certificate. For traditional certificates, this is typically achieved either by publishing a certificate revocation list or by enforcing a short certificate lifetime via expiration date. For anonymous certificates, the former approach violates privacy while the latter is typically rather inefficient as it would require the users to frequently engage in the usually quite involved issuing protocol.

In principle, the approach of certificate revocation list can be made to work also for anonymous credentials by having the user to prove in zero-knowledge that her certificate is not contained on the (black) list. Such a proof, however, would not be efficient as the computational and communication cost of the user and the verifier become preventive as they grow at least logarithmic with number of entries in the list. The literature provides two kinds solutions that overcome this.

The first kind is called verifier local revocation [13, 14, 11, 15]. In the best solution here, the cost for the user is independent of the number of entries in the revocation list, but the computational cost of the verifier is linear in this number (at least a modular exponentiation or, worse, a pairing operation per entry). Thus, these solutions are not at all suited for large scale deployments.

The second kind [16, 17] employs cryptographic accumulators [18]. Such accumulators allow one to hash a large set of inputs in a single short value, the *accumulator,* and then provide evidence by an *accumulator witness* that a given value is indeed contained in the accumulator. Thus, the serial numbers of all currently valid credentials are accumulated and the resulting value is published. Users can then show to verifiers that their credential is still valid, by using their witness to prove (in zero-knowledge) that their credential's serial number is contained in the published accumulator. Such proofs can be realized with practical

efficiency [16, 17] and incur only cost to the user and the verifier that are independent of the number of revoked or currently valid credentials. The drawback of these solutions, however, is that the users need to update their accumulator witnesses and an update requires at least one modular exponentiation for each newly revoked credential. Assuming a driving license application and based on the, e.g., 0.07% rate of driver's license revocation in West Virginia USA [19], the number of credentials revoked will quickly become a couple of thousands per day. Thus, these solutions incur a computational (and communication) cost far greater that what an electronic token such as a smart card can possibly handle.

Our contribution. In this paper we are therefore considering revocation solutions that incur (virtually) no cost to the verifier and the users, and only limited costs to the issuer (or the revocation authority). More precisely, for each revocation epoch (e.g., every day), verifiers and users need to retrieve the issuer's current public key (i.e., the new accumulator value) while users further need to retrieve their witnesses (a single group element). Upon revocation of a credential, the revocation authority only needs to perform one multiplication per remaining user to update (and provide) the users' witnesses, a cost which can easily be handled by today's standards. We note that this update operation requires no secret keys and does not need to be performed by the issuer, i.e., it could be performed by other untrusted entities.

As building block for this solution, we introduce a novel dynamic accumulator based on bilinear maps and show how to employ it for revocation at the example of the Bangerter, Camenisch and Lysyanskaya private certificate framework [12], which is essentially a generalization of e-cash, anonymous credentials, and group signatures. Thus we provide for the first time a practical solution for anonymous authentication with e-ID cards.

Related Work. Camenisch and Lysyanskaya [17] introduce a dynamic accumulator and show its applicability to revocation in Anonymous Credential Systems as well as Identity Escrow and Group Signatures. Update of the proposed accumulator, as well as user witnesses, require a number of exponentiations that is linear in the number of users added to or revoked from the system. In [20], the authors extend the above accumulator, introducing witnesses and proofs that a value was *not* accumulated.

Nguyen [21] constructs a dynamic accumulator from bilinear pairings. Its application to an anonymous credential system require users to store large system parameters, in order to prove validity of their credential. Moreover, updating a witness takes one exponentiation per event and therefore is not efficient enough for what we are after (in the paper the authors write multiplication and use addition as base operation for the algebraic group as is done sometimes in connection with bi-linear maps and elliptic curve groups).

In [22], the authors propose a dynamic accumulator for batch update. Users who missed many witness updates, can request update information to the issuer and update their witness with one multiplication. In our scheme, we can provide the same feature, relaxing the requirement that the issuer takes part to the

witness update. We note, however, that the authors do not show how to achieve an efficient proof of knowledge of an element contained in the accumulator as is needed for the use of the accumulator for revocation of credentials.

Outline. The rest of the paper is organized as follow. In Section 2 we discuss assumptions and recall existing building blocks. In Section 3 we introduce our novel dynamic accumulator. In Section 4 we show how to extend the Bangerter et al. private certificate framework with an efficient revocation mechanism. Conclusion and further discussion are given in Section 5

2 Preliminaries

In this section we list assumptions and cryptographic tools used as building blocks of the introduced accumulator as well as our anonymous credential revocation system.

A function ν is *negligible* if, for every integer c, there exists an integer K such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *intractable)* if there exists no probabilistic polynomial time (p.p.t.) algorithm on the size of the input to solve it.

Bilinear Pairings Let G and G_T be groups of prime order q. A map $e : G \times G \rightarrow G_T$ must satisfy the following properties:

(a) *Bilinearity*: a map $e : G \times G \rightarrow G_T$ is bilinear if $e(a^x, b^y)t = e(a, b)^{xy}$;
(b) *Non-degeneracy*: for all generators $g, h \in G$, $e(g, h)$ generates G_T;
(c) *Efficiency*: There exists an efficient algorithm $\mathsf{BMGen}(1^k)$ that outputs (q, G, G_T, e, g) to generate the bilinear map and an efficient algorithm to compute $e(a, b)$ for any $a, b \in G$.

The security of our scheme is based on the following number-theoretic assumptions. Our accumulator construction is based on the Diffie-Hellman Exponent assumption. The unforgeability of credentials is based on the Strong Diffie-Hellman assumption. For credential revocation we need to prove possession of an accumulator witness for a credential. This proof is based on our new Hidden Strong Diffie-Hellman Exponent (SDHE) assumption.

Definition 1 (n-DHE). *Diffie-Hellman Exponent (DHE) assumption: The n-DHE problem in a group G of prime order q is defined as follows: Let $g_i = g^{\gamma^i}$, $\gamma \leftarrow_R \mathbb{Z}_q$. On input $\{g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}\} \in G^{2n}$, output g_{n+1}.*

The n-DHE assumption states that this problem is hard to solve.

Boneh, Boyen, and Goh [23] introduced the Bilinear Diffie-Hellman Exponent (BDHE) assumption that is defined over a bilinear map. Here the adversary has to compute $e(g, h)^{\gamma^{n+1}} \in G_T$.

Lemma 1. *The n-DHE assumption for a group G with a bilinear pairing $e : G \times G \rightarrow G_T$ is implied by the n-BDHE assumption for the same groups.*

Boneh and Boyen introduced the Strong Diffie-Hellman assumption in [24].

Definition 2 (n-SDH [24]). *On input $g, g^x, g^{x^2}, \ldots, g^{x^n} \leftarrow G$, it is computationally infeasible to output $(g^{1/(x+c)}, c)$.*

Boyen and Waters [25] introduced the Hidden Strong Diffie-Hellman assumption under which BB signatures [24] are secure for any message space. We require a variant of the Hidden Strong Diffie-Hellman assumption that we call the Hidden Strong Diffie-Hellman Exponent (n-HSDHE) assumption. The two assumptions are hitherto incomparable.

Definition 3 (n-HSDHE). *Given $g, g^x, u \in G$, $\{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$, and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$, it is infeasible to compute a new tuple $(g^{1/(x+c)}, g^c, u^c)$.*

2.1 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

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 [26], (2) proof of knowledge of equality of some elements in different representation [27], (3) proof that a commitment opens to the product of two other committed values [28–30], and also (4) proof of the disjunction or conjunction of any two of the previous [31].

When referring to the above proofs, we will follow the notation introduced by Camenisch and Stadler [32] 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 values (α, β, δ) denote quantities of which knowledge is being proven (and are kept secret), 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 can extract these quantities from a successful prover.

2.2 Signature Scheme with Efficient Protocols

For our credential system we use a signature scheme that is loosely based on weak Boneh and Boyen signatures [24, 16]. It is described in [33] and has been proven secure under the n-SDH assumption [34, 35]. It assumes a non-degenerate bilinear map $e : G \times G \rightarrow G_T$ of prime order q with generators $h, h_0, h_1, \ldots, h_\ell, h_{\ell+1}$. The signer's secret key is $x \in \mathbb{Z}_q$ while the public key is $y = h^x$.

A signature on a message $m \in \mathbb{Z}_q^*$ is computed by picking $c, s \leftarrow \mathbb{Z}_q^*$ and computing $\sigma = (h_0 h_1^m h_2^s)^{\frac{1}{x+c}}$. The signature is (σ, c, s). It is verified by checking whether $e(\sigma, yh^c) = e(h_0 h_1^m h_2^s, h)$. Multiple messages $m_1, \ldots, m_\ell \in \mathbb{Z}_q^*$ can

be signed as $\sigma = (h_0 h_1^{m_1} \cdots h_\ell^{m_\ell} h_{\ell+1}^{s})^{\frac{1}{x+c}}$ and verification is done by checking whether $e(\sigma, yh^c) = e(h_0 h_1^{m} \cdots h_\ell^{m_\ell} h_{\ell+1}^{s}, h)$.

Proving Knowledge of a Signature. Now assume that we are given a signature (σ, c, s) on messages $m_1 \ldots, m_\ell \in \mathbb{Z}_q$ and want to prove that we indeed possess such a signature. To this end, we need to augment the public key with a value $\tilde{h} \in G$ such that $\log_h \tilde{h}$ are not known.

Knowledge of a signature is proven as follows:

1. Choose random values $r \leftarrow \mathbb{Z}_q$ and $open \leftarrow \mathbb{Z}_q$ and compute a commitment $B = h^r \tilde{h}^{open}$ and a blinded signature $A = \sigma \tilde{h}^r$.
2. Compute the following proof

$$PK\{(c, s, r, open, mult, tmp, m_1, \ldots, m_\ell) : \\ B = h^r \tilde{h}^{open} \ \wedge \ 1 = B^c h^{-mult} \tilde{h}^{-tmp} \ \wedge \\ \frac{e(h_0, h)}{e(A, y)} = e(A, h)^c \cdot e(\tilde{h}, y)^{-r} \cdot e(\tilde{h}, h)^{-mult} \cdot \prod_{i=1}^{\ell} e(h_i, h)^{-m_i} \cdot e(h_{\ell+1}, h)^{-s}\} \ .$$

Why this proof works is explained in Appendix B.

3 A Pairing Based Dynamic Accumulator with Efficient Updates

We define and build a dynamic accumulator with efficient updates and assess its security. With efficient updates we mean that witnesses can be updated by any party without knowledge of any secret key and require only multiplications (no exponentiations) linear in the number of changes to the accumulator. Our construction is based on the broadcast encryption scheme by Boneh, Gentry and Waters [36].

3.1 Definition of Dynamic Accumulators

A secure accumulator consists of the five algorithms AccGen, AccAdd, AccUpdate, AccWitUpdate, and AccVerify.

These algorithms are used by the accumulator authority (short authority), an untrusted update entity, a user and a verifier. The authority creates an accumulator key pair (sk_A, pk_A), the accumulator $acc_\emptyset$ and a public state $state_\emptyset$ using the AccGen algorithm; it can add a new value i to the accumulator acc_V using the AccAdd algorithm to obtain a new accumulator $acc_{V \cup \{i\}}$ and state $state_{U \cup \{i\}}$, together with a witness wit_i. The accumulator for a given set of values V, can be computed using the AccUpdate algorithm.

Throughout these operations, acc_V and wit_i are of constant size (independent of the number of accumulated values). The authority does some bookkeeping about the values contained in the accumulator and the status of the

accumulator when a witness wit_i was created. These sets are denoted as V and V_w respectively. The bookkeeping information is made public and is only needed for updating witnesses, it is not needed for verifying that a value is contained in an accumulator.

Each time an accumulator changes, the old witnesses become invalid. It is however possible to update all witnesses for values $i \in V$ contained in the accumulator from the bookkeeping information V_w. This updating is the most performance intensive operation in existing accumulator systems. We show how it can be efficiently offloaded to an untrusted update entity that runs $\mathsf{AccWitUpdate}$ and is only given the accumulator state $state_U$ and the bookkeeping information V and V_w. The accumulator state $state_U$ also contains book keeping information U, the set of elements ever added to the accumulator (but not necessarily contained in the current accumulator). This is a superset of V and V_w.[4]

After users obtained an updated witness wit'_i for a value i for the current accumulator, they can prove to any verifier that i is in the accumulator, using the $\mathsf{AccVerify}$ algorithm.

$\mathsf{AccGen}(1^k, n)$ creates an accumulator key pair (sk_A, pk_A), an empty accumulator $acc_\emptyset$ (for accumulating up to n values) and an initial state $state_\emptyset$.

$\mathsf{AccAdd}(sk_A, i, acc_V, state_U)$ allows the authority to add i to the accumulator. It outputs a new accumulator $acc_{V\cup\{i\}}$ and state $state_{U\cup\{i\}}$, together with a witness wit_i for i.

$\mathsf{AccUpdate}(pk_A, V, state_U)$ outputs an accumulator acc_V for values $V \subset U$.

$\mathsf{AccWitUpdate}(pk_A, wit_i, V_w, acc_V, V, state_U)$ outputs a witness wit'_i for acc_V if wit_i was a witness for acc_{V_w} and $i \in V$.

$\mathsf{AccVerify}(pk_A, i, wit_i, acc_V)$ verifies that $v \in V$ using an up-to-date witness wit_i and the accumulator acc_V. In that case the algorithm accepts, otherwise it rejects.

Note that the purpose of an accumulator is to have accumulator and witnesses of size independent of the number of accumulated elements.

Correctness. Correctly accumulated values have verifying witnesses.

Security. For all probabilistic polynomial time adversaries $\mathcal{A}$,

$$Pr[(sk_A, pk_A, acc_\mathcal{O}, state_\mathcal{O}) \leftarrow \mathsf{AccGen}(1^k);$$
$$(i, wit_i) \leftarrow \mathcal{A}(pk_A, acc_\mathcal{O}, state_\mathcal{O})^{\mathcal{O}_{\mathsf{AccAdd}}(.),\mathcal{O}_{\mathsf{AccUpdate}}(.)} :$$
$$\mathsf{AccVerify}(pk_A, i, wit_i, acc_\mathcal{O}) = \mathsf{accept} \wedge i \notin V_\mathcal{O}] \;=\; neg(k)\;,$$

where the oracles $\mathcal{O}_{\mathsf{AccAdd}}(.)$ and $\mathcal{O}_{\mathsf{AccUpdate}}(.)$ keep track of shared variables $acc_\mathcal{O}$, $state_\mathcal{O}$ and a set $V_\mathcal{O}$ that is initialized to $\emptyset$. The oracle $\mathcal{O}_{\mathsf{AccAdd}}(i)$ computes and outputs $(acc_\mathcal{O}, state_\mathcal{O}, wit_i) \leftarrow \mathsf{AccAdd}(sk_A, i, acc_\mathcal{O}, state_\mathcal{O})$ and adds i to $V_\mathcal{O}$ while $\mathcal{O}_{\mathsf{AccUpdate}}(V)$ computes and outputs $acc_\mathcal{O} \leftarrow \mathsf{AccUpdate}(pk_A, V, state_\mathcal{O})$ and sets $V_\mathcal{O}$ to V.

[4] Allowing accumulators to change their state over time can allow for better performance tradeoffs. While our accumulator construction does not use this possibility in order to keep things simple, we outline such an optimization in Appendix D.

3.2 Construction

We now construct the algorithms AccGen, AccAdd, AccUpdate, AccWitUpdate, and AccVerify.

AccGen$(1^k, n)$. Run BMGen(1^k) to obtain the setup $params_{BM} = (q, G, G_T, e, g)$ of a bilinear map $e : G \times G \to G_T$.
Pick a random value $\gamma \in \mathbb{Z}_q$. Generate a key pair sk and pk for a secure signature scheme, for instance the BB signature scheme that is secure under the SDH assumption. Let $pk_A = (params_{BM}, pk, z = e(g,g)^{\gamma^{n+1}})$, $sk_A = (params_{BM}, \gamma, sk)$, $acc_\emptyset = 1$ and $state_\emptyset = (\emptyset, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$.[5]

AccAdd$(sk_A, i, acc_V, state_U)$. Compute $w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$ and a signature σ_i on $g_i \| i$ under signing key sk. The algorithm outputs $wit_i = (w, \sigma_i, g_i)$, an updated accumulator value $acc_{V \cup \{i\}} = acc_V \cdot g_{n+1-i}$, and $state_{U \cup \{i\}} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$.

AccUpdate$(pk_A, V, state_U)$. Check whether $V \subset U$ and outputs $\perp$ otherwise. The algorithm outputs $acc_V = \prod_{v \in V} g_{n+1-v}$ for values $i \in V$.

AccWitUpdate$(pk_A, wit_i, V_w, acc_V, V, state_U)$. Parse wit_i as (w, σ_i, g_i). If $i \in V$ and $V \cup V_w \subset U$, compute

$$w' = w \cdot \frac{\prod_{j \in V \setminus V_w} g_{n+1-j+i}}{\prod_{j \in V_w \setminus V} g_{n+1-j+i}} .$$

Output the updated witness $wit'_i = (w', \sigma_i, g_i)$. Otherwise output $\perp$.

AccVerify(pk_A, i, wit_i, acc_V). Parse $wit_i = (w, \sigma_i, g_i)$. Output accept, if σ_i is a valid signature on $g_i \| i$ under verification key pk and $\frac{e(g_i, acc_V)}{e(g,w)} = z$. Otherwise output reject.

In the construction above, we accumulate the group elements $g_1, \ldots, g_n$ instead of, e.g., the integers $1, \ldots, n$. Depending on the application, one would want to accumulate the latter, or more generally an arbitrary set of size n. In this case, the issuer of the accumulator would need to publish a mapping from this set to the g_i values that get actually accumulated. In order to avoid large public parameters during verification the issuer of the accumulator uses a signature scheme to sign the g_i together with the value to which they map. Thus, the verifier can check whether a given g_i is a (potentially) valid input to the accumulator (cf. discussion in Section 3.3).

We also note that the algorithm to update the witness does not require any secret information. Thus, the witnesses could be kept up-to-date for the users either by the users themselves, the issuer, or by some third party. In the latter

[5] We define $state_U = (U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ where U is book keeping information that keeps track of all elements that were ever added to the accumulator (but might have been subsequently removed). The rest of the state is static. See Appendix D for a modification that reduces the size of $state_U$.

two cases, the users can just retrieve the current valid witness whenever needed. In applications, one would typically define epochs such that the accumulator value and witnesses are only updated at the beginning of each epoch and remain valid throughout the epoch. Finally note that maintaining the witnesses for all users is well within reach of current technologies — indeed, all witnesses can be kept in main memory and the update performed rather quickly.

Correctness. Let acc_V be an accumulator for $sk_A = (params_{BM}, \gamma, sk)$, $pk_A = (params_{BM}, pk, z = e(g,g)^{\gamma^{n+1}})$, and $state_U = (U, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$. Then a correct accumulator always has a value $acc_V = \prod_{j \in V} g_{n+1-j}$. Moreover, for each $i \in V$ with up-to-date witness $wit_i = (w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}, \sigma_i, g_i)$ the following equation holds:

$$\frac{e(g_i, acc_V)}{e(g,w)} = \frac{e(g,g)^{\sum_{j \in V} \gamma^{n+1-j+i}}}{e(g,g)^{\sum_{j \in V}^{j \neq i} \gamma^{n+1-j+i}}} = e(g,g)^{\gamma^{n+1}} = z \ .$$

Security. Suppose there exists an adversary $\mathcal{A}$ that breaks the security of our accumulator. We show how to construct an algorithm $\mathcal{B}$ that either forges the signature scheme used to sign accumulated elements or breaks the n-DHE assumption.

Algorithm $\mathcal{B}$ has access to a signing oracle $\mathcal{O}_\sigma$ and obtains as input the corresponding signature verification key pk, the parameters of a bilinear map $params_{BM} = (q, G, G_T, e, g)$, and an instance of the n-DHE assumption $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in G^{2n-1}$. $\mathcal{B}$ provides $\mathcal{A}$ with $pk_A = (params_{BM}, pk, z = e(g_1, g_n))$, $acc_\emptyset = 1$ and $state_\emptyset = (\emptyset, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. The oracle queries of the adversary are answered as defined in the game except that $\mathcal{O}_\sigma$ is called for creating the signatures.

Given an adversary that can compute (i, wit_i) such that the verification succeeds even though $i \notin V_\mathcal{O}$. We parse wit_i as $(w, \hat{\sigma}_i, \hat{g}_i)$. If $\hat{g}_i$ does not correspond to g_i the adversary attacked the signature and $\hat{\sigma}_i$ is a signature forgery. Otherwise we learn from the verification equation that

$$e(g_i, acc_\mathcal{O}) = e(g,w)z$$

and

$$e(g, \prod_{j \in V} g_{n+1-j+i}) = e(g, w g_{n+1}) \ .$$

This means that

$$g_{n+1} = \frac{\prod_{j \in V} g_{n+1-j+i}}{w} \ .$$

For $i \in \{1, \ldots, n\} \setminus V$, all $g_{n+1-j+i}$ are contained in $state_\emptyset$ and it is possible to compute this value. This breaks the n-DHE assumption.

3.3 Efficient Proof That a Hidden Value was Accumulated

It is often only required for a user to prove that she possesses a value that is indeed contained in the current accumulator, or in other words, to prove membership of the current accumulator without revealing which value she possesses (or which index i is assigned to her). In this section, we give an efficient protocol that achieves this for our accumulator construction.

For the accumulator to be secure, the verifier needs to check that the value the user claims to own, is one of $g_1, \ldots, g_n$. In the previous construction, $g_1, \ldots, g_n$ are authenticated either by making them public as a whole or by having each one signed (in which case the user would provide the g_i and the signature to the verifier). However, using a public list would require the prover to either reveal g_i (which would violate privacy) or to prove that the g_i which she claims possession of, is a valid one. The latter, however, would require an involving proof that would make the use of the accumulator inefficient. We therefore resort to sign g_i values and then require the prover to prove that she knows a signature by the accumulator issuer on "her" g_i without revealing neither the signature nor the g_i value. As such a proof needs to be efficient, this requires a special signature scheme. Since user never reveal the accumulated valued they are proving possession of, it is possible to avoid signing $g_i||i$ as it is done in Section 3.2. This allows for a more efficient signature scheme and proof system.

Prerequisites. We instantiate the signature scheme used for signing the g_i with a variant of the weakly secure Boneh-Boyen scheme [24]. Instead of a g_i value we sign γ^i. The authentic g_i is a by-product of the signing process. For simplicity we reduce the security of the accumulator proof directly to the n-HSDHE assumption.[6] The n-HSDHE assumption is the weakest assumption under which we can prove our scheme. The n-HSDHE assumption is implied by the iHSDH assumption of [37].

The signer (the accumulator issuer) picks a fresh $u \leftarrow G$, secret key $sk \leftarrow \mathbb{Z}_q$ and public key $pk = g^{sk}$. A signature consists of the two elements $\sigma_i = g^{1/(sk+\gamma^i)}$ and $u_i = u^{\gamma^i}$ and is verified by checking that $e(pk \cdot g_i, \sigma_i) = e(g, g)$.

Let $pk_A = (params_{BM}, pk, z = e(g,g)^{\gamma^{n+1}})$, $sk_A = (params_{BM}, \gamma, sk)$ and $state_U = (\emptyset, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$ be as generated by the accumulator operations in the previous section. We also pick an additional $\tilde{h} \leftarrow G$ for commitments. The discrete logarithm of h and u with respect to g must be unknown to the prover.

Proof of Knowledge. For arbitrary $V \subset \{1, \ldots, n\}$ and $i \in V$, on input $acc_V = \prod_{i \in V} g_{n+1-i}$ and the corresponding witness $wit_i = (w, \sigma_i, u_i, g_i)$, where $w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$, for value i, the prover performs the following randomization:

[6] We do not prove the signature scheme itself secure, but we refer to [37] for a similar scheme.

Pick at random $r, r', r'', r''', open \in \mathbb{Z}_q$ and computing $\mathcal{G} = g_i\tilde{h}^r$, $\mathcal{W} = w\tilde{h}^{r'}$, $D = g^r\tilde{h}^{open}$, $\mathcal{S} = \sigma_i\tilde{h}^{r''}$, and $\mathcal{U} = u_i\tilde{h}^{r'''}$ respectively. Then the prover, proves

$$PK\{(r, r', r'', r''', open, mult, tmp) : D = g^r\tilde{h}^{open} \wedge 1 = D^{r''}g^{-mult}\tilde{h}^{-tmp} \wedge$$
$$\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g,g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''}e(\tilde{h}, \tilde{h})^{-mult}e(\tilde{h}, \mathcal{S})^r \wedge$$
$$\frac{e(\mathcal{G}, acc_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, acc_V)^r e(1/g, \tilde{h})^{r'} \wedge \frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^r e(1/g, \tilde{h})^{r'''}\} .$$

Theorem 1. *Under the n-DHE and the n-HSDHE assumptions the protocol above is a proof of knowledge of a randomization value r that allows to derandomize $\mathcal{G}$ to a value g_i, where i is accumulated in acc_V, i.e., $i \in V$.* The proof of this theorem can be found in Section C.1.

4 Efficient Revocation of Private Certificates

In this section we will show how to employ our accumulator to achieve efficient revocation for schemes where users get some form of certificate and then later can use these certificates in an anonymity protecting way. Such schemes include group signatures, anonymous credential systems, pseudonym systems, anonymous e-cash, and many others. Most of these schemes work as follows. In a first phase an issuer provides the user with a signature on a number of messages. Then, in a second phase the user convinces the verifier that 1) she owns a signatures by the issuer on a number of messages and 2) that these messages satisfy some further properties that are typically dependent on the particular purpose of the scheme. Based on this observation, Bangerter et al. [12] give a cryptographic framework for the controlled release of certified information. They also show how different applications (such as the ones mentioned above) can be realized. Thus, they basically generalize the concepts of anonymous credentials, anonymous e-cash, and group signatures into a single framework. We therefore just show how their framework can be extended with revocation to provide this features for all these applications. From this it will become clear how to extend particular schemes (e.g., the anonymous credentials and group signatures [16, 33]) with our revocation mechanisms.

More precisely, Bangerter et al. employ special signature protocols, called CL signatures [**?**], for issuing *private certificates* to users. A private certificate (1) consists of attributes and a signature over the attributes much alike a traditional certificate, only that a more powerful signature scheme is used.

$$cert = (\sigma, m_1, \ldots, m_l) \text{ with } \sigma = \mathsf{Sign}(m_1, \ldots, m_l; sk_I) \tag{1}$$

Let $(sk_I, pk_I) \leftarrow \mathsf{IssuerKeygen}(1^k)$ be the certificate issuer's keypair. The framework supports two types of protocols: 1) an interactive certificate issuing protocol $\mathsf{ObtainCert}$ that allows to obtain a signature on committed values without revealing these values and 2) efficient zero-knowledge proofs of knowledge of signature possession.

Let $(m_1, \ldots, m_\ell)$ denote a list of data items and $H \subset L = \{1, \ldots, \ell\}$ a subset of data items. Using the first protocol, a user can obtain a certificate on $(m_1, \ldots, m_\ell)$ such that the issuer does not learn any information on the data items in H, while it learns the other data items, i.e., $L \setminus H$.

The private certificates of a user remain private to the user, that is, they are never released (as a whole) to any other party: when using (showing) certificates for asserting attribute information, the user proves that she knows (has) certificates with certain properties. The user may release certain attributes, while only proving the knowledge of the rest of the certificate:

$$\mathsf{PK}\{(\sigma, m_1, \ldots, m_{\ell'}) : 1 = \mathsf{VerifySign}(\sigma, m_1, \ldots m_{\ell'}, m_{\ell'+1}, \ldots, m_\ell; pk_I) \wedge \ldots\}$$

In the above proof only the attribute values of $m_{\ell'+1}$ to m_ℓ are revealed.

Certificate revocation. We now extend the above framework with certificate revocation as follows. Let V be the set of valid certificates for an epoch with epoch information $epoch_V$. A certificate is assigned a unique identifier i (which will be embedded into it as one of the attributes) and a witness wit_i. We require that the user can prove to a verifier that she possesses a non-revoked certificate only if $i \in V$. This is achieved by having the user prove that the identifier embedded into her credential is a valid one for the current epoch. Thus, before engaging in a proof, the user needs to update her witness and both parties (the user and the verifier) need to obtain the most up-to-date epoch information $epoch_V$ for V. The user can either update the witness herself, or just retrieve the currently valid witness from a witness update entity. Indeed, a witness update computation does not require knowledge of any secret and can be performed by untrusted entities (e.g., by a third party or a high availability server cluster at the issuer). In particular, those entities are only responsible for computing user witnesses according to the current epoch information. Misbehavior by such entities would lead in a denial of service (the verification algorithm would *reject*, but would not break the security of the system). Also note that a witness update requires a number of multiplications that is linear in the number of elements added to or removed from the accumulator, hence providing such an update service to users is feasible (one could even hold all users' witnesses in main memory).

More formally, a certificate revocation system for the certification framework consists of updated IssuerKeygen and ObtainCert protocols, new algorithms UpdateEpoch and UpdateWitness for managing revocation, and a zero-knowledge proof system for a new predicate VerifyEpoch that allows to prove possession of a witness wit_i:

$\mathsf{IssuerKeygen}(1^k, n)$ creates the issuer key pair (sk_I, pk_I), the epoch information $epoch_\emptyset$, and $state_\emptyset$ for issuing up to n certificates.

$\mathsf{ObtainCert}(\mathcal{U}(pk_I, H, \{m_j\}_{j \in H}), \mathcal{I}(sk_I, H, \{m_j\}_{j \in L \setminus H}, epoch_V, state_U))$ allows a user to obtain a private certificate $cert_i$ from the issuer. The issuer computes and publishes the user's witness wit_i, and updated epoch information $epoch_{V \cup \{i\}}$ and $state_{U \cup \{i\}}$.

$\mathsf{UpdateEpoch}(V, state_U)$ outputs epoch information $epoch_V$, if $V \subset U$. Otherwise it outputs $\perp$.

$\mathsf{UpdateWitness}(wit_i, epoch_V, state_U)$ outputs an updated witness wit_i' if $V \subset U$. Otherwise it outputs $\perp$.

A user who knows a certificate $cert_i$ and a corresponding up-to-date witness wit_i can prove, to a verifier, possession of the certificate and its validity for the current epoch using the new predicate $\mathsf{VerifyEpoch}$ as follows. The user's secret input is $cert_i$. The common input of the protocol is the issuer's public key pk_I, the epoch information $epoch_V$, and a specification of the proof statement (this includes the information revealed about the certificate). In the example below the user chooses to keep the first ℓ' messages secret while he reveals the rest of the messages.

$$\begin{aligned}\mathsf{PK}\{(\sigma, m_1, \ldots, m_{\ell'}, i, wit_i) : 1 = &\mathsf{VerifySign}(\sigma, m_1, \ldots m_{\ell'}, m_{\ell'+1}, \ldots, m_\ell, i; pk_I) \wedge \\ 1 = &\mathsf{VerifyEpoch}(i, wit_i; epoch_V, pk_I)\}\ .\end{aligned}$$

Using the Bangerter et al. framework [12], it is not hard to extend this proof or combine it with other proof protocols given therein.

4.1 Adapted Signature Scheme for Accumulated Values

As described above, a user would have to prove that the value i encoded into her credential is also contained in the current accumulator. However, the accumulator construction as given in the previous section does not allow one to accumulate i directly but only $g_i = \tilde{g}^{\gamma^i}$. Now, instead of introducing a mapping of i to g_i (and including this in our proofs which would make them inefficient), we are going to make the mapping implicit by including g_i into the credential. Thus, the g_i values will be used both in the private certificate and the accumulator to represent the certificate id i. This requires that we extend the signature scheme in Section 2.2 to allow verification without knowing the secret exponent γ^i:

1. The signer creates $g, h, h_0, h_1, \ldots, h_\ell, h_{\ell+1} \leftarrow G$ and creates keys $x \in \mathbb{Z}_q$ and $y = h^x$.
2. Next, the signer publishes a list $(g_1 = g^\gamma, \ldots, g_n = g^{\gamma^n})$ that he allows in signatures.
3. The signer picks random $c, s \leftarrow \mathbb{Z}_q^*$ and then computes the signature as $(\sigma = (h_0 h_1^{m_1} \cdots h_\ell^{m_\ell} g_i h_{\ell+1}^s)^{\frac{1}{x+c}}, c)$.
4. A signature (σ, c, s) on messages $m_1, \ldots, m_\ell, \hat{g}_i$ is verified by checking that $\hat{g}_i$ is in the list of g_i values and that $e(\sigma, yh^c) = e(h_0(\prod_{j=1}^{\ell} h_j^{m_j})\hat{g}_i h_{\ell+1}^s, h)$ holds.

We note that the check that $\hat{g}_i$ is in the list of g_i values as prescribed in the last step will later on be replaced by a signature/authenticator on g_i as done for the accumulator in Section 3.3.

It is straightforward to reduce the security of this modified signature scheme to the original one with $\ell + 1$ messages as the signer knows the "messages" γ^i encoded by the g_i. We omit the details here.

4.2 Construction

$\mathsf{IssuerKeygen}(1^k, n)$**.** Run $\mathsf{BMGen}(1^k)$ to generate the parameters $params_{BM} = (q, G, G_T, e, g)$ of a (symmetric) bilinear map $e : G \times G \to G_T$. Pick additional bases $h, h_0, \ldots, h_{\ell+1}, \tilde{h}, u \leftarrow G$ and $x, sk, \gamma \leftarrow \mathbb{Z}_q$ and compute $y = h^x$ and $pk = g^{sk}$.[7] Compute $g_1, ..., g_n, g_{n+2}, ..., g_{2n}$, where $g_i = g^{\gamma^i}$, and $z = e(g, g)^{\gamma^{n+1}}$.
Output $(sk_I, pk_I) = ((params_{BM}, x, sk, \gamma), (params_{BM}, y, h, h_0, \ldots, h_{\ell+1}, \tilde{h}, u, pk, z))$, $epoch_\emptyset = (acc_\emptyset = 1, \emptyset)$, and $state_\emptyset = (\emptyset, g_1, ..., g_n, g_{n+2}, ..., g_{2n})$.

$\mathsf{ObtainCert}(\mathcal{U}(pk_I, H, \{m_j\}_{j\in H}), \mathcal{I}(sk_I, H, \{m_j\}_{j\in L\setminus H}, epoch_V, state_U)$. The user runs the following protocol to obtain a certificate $cert_i$ from the issuer:

1. The user chooses a random $s' \in \mathbb{Z}_q^*$, computes $X = \prod_{j\in H} h_j^{m_j} h_{\ell+1}^{s'}$, and sends X to the issuer.
2. The user (as prover) engages the issuer (as verifier) in the following proof
$$PK\{\{m_j\}_{j\in H}, s') : X = \prod_{j\in H} h_j^{m_j} h_{\ell+1}^{s'}\} ,$$
which will convince the issuer that X is correctly formed.
3. The issuer parses $epoch_V$ as (acc_V, V) and $state_U$ as $(U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. He then computes $epoch_{V\cup\{i\}} = (acc_V \cdot g_{n+1-i}, V \cup \{i\})$ and $state_{U\cup\{i\}} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ [8] .
4. The issuer chooses random $c, s'' \in \mathbb{Z}_q^*$ and then computes the signature $\sigma = ((\prod_{j\in L\setminus H} h_j^{m_j}) X g_i h_{\ell+1}^{s''})^{1/(x+c)}$.
5. The issuer computes $w = \prod_{j\in V}^{j\neq i} g_{n+1-j+i}$, $\sigma_i = g^{1/(sk+\gamma^i)}$, and $u_i = u^{\gamma^i}$ and sets $wit_i = (\sigma_i, u_i, g_i, w, V_\cup\{i\})$.
6. The issuer sends $(\sigma, c, s'', \{m_j\}_{j\in L\setminus H}, g_i, i)$ to the user and outputs wit_i, $epoch_{V\cup\{i\}}$, and $state_{U\cup\{i\}}$.
7. The user verifies the certificate gotten and outputs $cert_i = (\sigma, c, m_1, \ldots, m_\ell, g_i, s = s' + s'', i)$.

$\mathsf{UpdateEpoch}(V, state_U)$ checks whether $V \subset U$ and outputs $\perp$ otherwise. The algorithm creates $epoch_V$ for proving possessions of $cert_i$, $i \in V$. Let $acc_V = \prod_{i\in V} g_{n+1-i}$, output $epoch_V = (acc_V, V)$.

$\mathsf{UpdateWitness}(wit_i, epoch_V, state_U)$ aborts with $\perp$, if $V \not\subset U$. Otherwise it parses wit_i as $(\sigma_i, u_i, g_i, w, V_w)$. Let $w' = w^{\frac{\prod_{j\in V\setminus V_w} g_{n+1-j+i}}{\prod_{j\in V_w\setminus V} g_{n+1-j+i}}}$. The algorithm outputs $wit_i' = (\sigma_i, u_i, g_i, w', V)$.

Proof protocol. We now show a protocol that allows a user to prove possession of an unrevoked (and updated) credential $cred_i = (\sigma, c, m_1, \ldots, m_\ell, g_i, s, i)$ using $wit_i = (\sigma_i, g_i, u_i, w, V_w)$. The common input of the protocol is the issuer's public

[7] Note that the discrete logarithms of $g, h, \tilde{h}$ and u with respect to each other are mutually unknown.

[8] Both $epoch_{V\cup\{i\}}$ and $state_{U\cup\{i\}}$ could be signed by the issuer to prevent proliferation of fake accumulators.

key pk_I, the epoch information $epoch_V$, and a specification of the proof statement (this includes the information revealed about the certificate). In the example below the user chooses to keep the first ℓ' messages secret while he reveals the rest of the messages.

The user (as prover) picks $\rho, \rho', r, r', r'', r''' \leftarrow \mathbb{Z}_q$, and picks opening $open$, $open' \leftarrow \mathbb{Z}_q$ to commit to ρ and r respectively. He computes commitments $C = h^{\rho}\tilde{h}^{open}$, $D = g^{r}\tilde{h}^{open'}$ and blinded values $A = \sigma\tilde{h}^{\rho}$, $\mathcal{G} = g_i\tilde{h}^{r}$, $\mathcal{W} = w\tilde{h}^{r'}$, $\mathcal{S} = \sigma_i\tilde{h}^{r''}$, and $\mathcal{U} = u_i\tilde{h}^{r'''}$. The user sends $C, D, A, \mathcal{G}, \mathcal{W}, \mathcal{S}$ and $\mathcal{U}$ to the verifier and engages the verifier in the following proof:

$$
\begin{aligned}
PK\{(c, \rho, open, mult, tmp, m_1, \ldots, m_{\ell'}, s, r, open', mult', tmp', r', r'', r''') :& \\
C = h^{\rho}\tilde{h}^{open} \ \wedge \ 1 = C^{c}h^{-mult}\tilde{h}^{-tmp} \ \wedge & \quad (1)\\
\frac{e(h_0 \cdot \prod_{j=\ell'+1}^{\ell} h_j^{m_j} \cdot \mathcal{G}, h)}{e(A, y)} = e(A, h)^{c} \cdot e(\tilde{h}, h)^{r} & \quad (2)\\
\cdot\, e(\tilde{h}, y)^{-\rho} \cdot e(\tilde{h}, h)^{-mult} \cdot \prod_{j=1}^{\ell'} e(h_j, h)^{-m_j} \cdot e(h_{\ell+1}, h)^{-s} \wedge & \\
\frac{e(\mathcal{G}, acc_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, acc_V)^{r} e(1/g, \tilde{h})^{r'} \wedge & \quad (3)\\
D = g^{r}\tilde{h}^{open'} \ \wedge \ 1 = D^{c}g^{-mult'}\tilde{h}^{-tmp'} \ \wedge & \quad (4)\\
\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g, g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''} e(\tilde{h}, \tilde{h})^{-mult'} e(\tilde{h}, \mathcal{S})^{r} \wedge & \quad (5)\\
\frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^{r} e(1/g, \tilde{h})^{r'''}\} \ . & \quad (6)
\end{aligned}
$$

This proof merges the proof of knowledge of Section 3.3 with a proof of knowledge of an adapted signature as the ones described in Section 4.1. The latter is similar to the proof of knowledge of a signature in Section 2.2. Special care needs to be taken to bind the g_i in the accumulator to the g_i value in the adapted signature.

Theorem 2. *Under the n-HSDHE and the n-DHE assumptions, the protocol above is a proof of knowledge of an adapted signature on* $(m_1, \ldots, m_\ell, g_i)$ *such that* $i \in V$. The proof can be found in Section C.1.

5 Conclusion and Discussion

In this paper we have introduced a novel dynamic accumulator based on bilinear maps and have shown how it can be used to achieve efficient revocation in privacy-preserving systems such as group signatures or anonymous credential systems.

Previous proposals require expensive computations for updating witnesses and are not suitable for electronic token based systems with a large number of users, as the ones that will soon appear with the introduction of e-ID's, e-tickets

and alike. Our accumulator overcomes the aforementioned drawback introducing efficient witness updates. In the envisioned system, at the beginning of each epoch, the users retrieve their currently valid witness from an updating authority (as the number of revocation per epoch is likely to be very large, the users will typically not be able to handle them). As updating a witness in our scheme requires only a number of multiplication linear in the number of changes to the accumulator (in particular, linear in $|(V \setminus V_w) \cup (V_w \setminus V)|$) a single authority (which not necessarily needs to be the issuer) can keep the witness values for all users easily up-to-date (and in main memory). This is a key feature that enables the adoption of dynamic accumulators for revocation in privacy-preserving systems with large number of users as, e.g., in the case of electronic driving license systems. Although not necessary, there could even be several witness update entities, responsible for upgrading witnesses for groups of users. For example, in a national e-ID's systems, witness updates could be performed by per-county or per-city witness update entity. The latter requires only public parameters and are only responsible for correct computation of the witness updates for the users in their group. Malicious behavior by one of the witness update entities, does not break system security (recall that they only require public parameters) but can only lead to denial of service. That is, if a witness is not correctly computed (not reflecting the latest changes in the accumulator) it would prevent a user to prove validity of her credential. In this case, users can report to the issuing authority to obtain a valid witness update and signal the misbehaving of the witness update entity.

6 Acknowledgements

During this work, we enjoyed many discussion with Thomas Gross and Tom Heydt-Benjamin on various kinds of revocation of anonymous credentials. Thank you! The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483.

References

1. Camenisch, J., Van Herreweghen, E.: Design and implementation of the *idemix* anonymous credential system. Technical Report Research Report RZ 3419, IBM Research Division (May 2002)
2. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. Technical Report Research Report RZ 3295, IBM Research Division (November 2000)
3. Persiano, G., Visconti, I.: An efficient and usable multi-show non-transferable anonymous credential system. In Juels, A., ed.: Financial Cryptography. Volume 3110 of Lecture Notes in Computer Science., Springer (2004) 196–211
4. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Commun. ACM **28**(10) (1985) 1030–1044

5. Chaum, D., Evertse, J.H.: A secure and privacy-protecting protocol for transmitting personal information between organizations. In Odlyzko, A.M., ed.: CRYPTO. Volume 263 of Lecture Notes in Computer Science., Springer (1986) 118–167
6. Chen, L.: Access with pseudonyms. In Dawson, E., Golic, J.D., eds.: Cryptography: Policy and Algorithms. Volume 1029 of Lecture Notes in Computer Science., Springer (1995) 232–243
7. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In Heys, H.M., Adams, C.M., eds.: Selected Areas in Cryptography. Volume 1758 of Lecture Notes in Computer Science., Springer (1999) 184–199
8. Okamoto, T.: An efficient divisible electronic cash scheme. In Coppersmith, D., ed.: CRYPTO. Volume 963 of Lecture Notes in Computer Science., Springer (1995) 438–451
9. Chan, A.H., Frankel, Y., Tsiounis, Y.: Easy come - easy go divisible cash. In: EUROCRYPT. (1998) 561–575
10. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. [38] 302–321
11. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. [39] 132–145
12. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A cryptographic framework for the controlled release of certified data. In Christianson, B., Crispo, B., Malcolm, J.A., Roe, M., eds.: Security Protocols Workshop. Volume 3957 of Lecture Notes in Computer Science., Springer (2004) 20–42
13. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In Blaze, M., ed.: Financial Cryptography. Volume 2357 of Lecture Notes in Computer Science., Springer (2002) 183–197
14. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. [39] 168–177
15. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. IEICE Transactions **90-A**(1) (2007) 65–74
16. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. [40] 41–55
17. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In Yung, M., ed.: CRYPTO. Volume 2442 of Lecture Notes in Computer Science., Springer (2002) 61–76
18. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: EUROCRYPT. (1993) 274–285
19. of Motor Vehicles, W.V.D.: Wvdmv fy 2005 annual report. `http://www.wvdot.com/6_motorists/dmv/downloads/DMV-AnnualReport2005.pdf` (2005)
20. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In Katz, J., Yung, M., eds.: ACNS. Volume 4521 of Lecture Notes in Computer Science., Springer (2007) 253–269
21. Nguyen, L.: Accumulators from bilinear pairings and applications. In Menezes, A., ed.: CT-RSA. Volume 3376 of Lecture Notes in Computer Science., Springer (2005) 275–292
22. Wang, P., Wang, H., Pieprzyk, J.: A new dynamic accumulator for batch updates. In Qing, S., Imai, H., Wang, G., eds.: ICICS. Volume 4861 of Lecture Notes in Computer Science., Springer (2007) 98–112
23. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. [38] 440–456
24. Boneh, D., Boyen, X.: Short signatures without random oracles. In Cachin, C., Camenisch, J., eds.: EUROCRYPT. Volume 3027 of Lecture Notes in Computer Science., Springer (2004) 56–73

25. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In Okamoto, T., Wang, X., eds.: Public Key Cryptography. Volume 4450 of Lecture Notes in Computer Science., Springer (2007) 1–15
26. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology **4**(3) (1991) 161–174
27. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In Brickell, E.F., ed.: CRYPTO. Volume 740 of Lecture Notes in Computer Science., Springer (1992) 89–105
28. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: EUROCRYPT. (1999) 107–122
29. Camenisch, J.L.: 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.
30. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: EUROCRYPT. (1997) 318–333
31. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In Desmedt, Y., ed.: CRYPTO. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 174–187
32. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)
33. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. [40] 56–72
34. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In Halevi, S., Rabin, T., eds.: TCC. Volume 3876 of Lecture Notes in Computer Science., Springer (2006) 80–99
35. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic -taa. In Prisco, R.D., Yung, M., eds.: SCN. Volume 4116 of Lecture Notes in Computer Science., Springer (2006) 111–125
36. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In Shoup, V., ed.: CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 258–275
37. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In Canetti, R., ed.: TCC. Volume 4948 of Lecture Notes in Computer Science., Springer (2008) 356–374
38. Cramer, R., ed.: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005)
39. Atluri, V., Pfitzmann, B., McDaniel, P.D., eds.: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washingtion, DC, USA, October 25-29, 2004. In Atluri, V., Pfitzmann, B., McDaniel, P.D., eds.: ACM Conference on Computer and Communications Security, ACM (2004)
40. Franklin, M.K., ed.: Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. In Franklin, M.K., ed.: CRYPTO. Volume 3152 of Lecture Notes in Computer Science., Springer (2004)

A Generic Group Security of n-HSDHE

We provide more confidence in the n-HSDHE assumption by proving lower bounds on the complexity of a harder problem in the generic group model.

Definition 4 (n-HSDHE). *Given $g, g^x, u \in G$, $\{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$, and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$, it is infeasible to compute a new tuple $(g^{1/(x+c)}, g^c, u^c)$.*

To simplify our analysis we prove the generic group security of the weaker assumption in which the adversary is given the value γ instead of the hiding values $\{g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$ and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$.

Definition 5 (weakened n-HSDHE). *Given $g, g^x, u \in G$, $\{g^{1/(x+\gamma^i)}\}_{i=1\ldots n}$ and γ it is infeasible to compute a new tuple $(g^{1/(x+c)}, g^c, u^c)$.*

Any attack algorithm that can break the n-HSDHE assumption can be used to break the weakened n-HSDHE. All the reduction does is compute the values $\{g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$ and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$ from γ and hand the now completed n-HSDHE problem instance to the n-HSDHE attack algorithm. Consequently it is sufficient to prove that the weakened n-HSDHE assumption holds in the generic group model.

Let $e : G \times G \to G_T$ be a bilinear map over groups G and G_T of prime order q. In the generic group model, we encode group elements of G and G_T as unique random strings. Group operations are performed by an oracle, that operates on these strings and keeps an internal representation of the group. We set $\alpha : \mathbb{Z}_q \to \{0,1\}^*$ to be the opaque encoding of elements of G. Let g be a generator of G; $\alpha(a)$ maps $a \in \mathbb{Z}_q$ to the string representation of $g^a \in G$. The function $\tau : \mathbb{Z}_q \to \{0,1\}^*$ maps $a \in \mathbb{Z}_q$ to $e(g,g)^a \in G_T$.

We can represent all operations in terms of the random maps α, τ. Note that the maps do not need to be explicitly given. It is sufficient if the oracles create them using lazy evaluation. Let $a, b \in \mathbb{Z}_q$. We define oracle queries for the following operations:

Group Operation. $\alpha(a) \cdot \alpha(b) = \alpha(a+b)$. This is because $\alpha(a) \cdot \alpha(b) = g^a \cdot g^b = g^{a+b} = \alpha(a+b)$. The same holds for the group operation in τ.

Exponentiation by a Constant. $\alpha(b)^a = \alpha(ab)$. This is because $\alpha(b)^a = (g^b)^a = g^{ab} = \alpha(ab)$. The same holds for multiplication by a constant in τ.

Pairing. $e(\alpha(a), \alpha(b)) = \tau(ab)$. This is because $e(\alpha(a), \alpha(b)) = e(g^a, g^b) = e(g,g)^{ab} = \tau(ab)$.

When an adversary tries to break the n-HSDHE assumption to the generic group model, the adversary does not get $g, g^x, u \in G$, $\{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$, and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$ Instead, we encode these values using the random maps α and τ. The generators g and $e(g,g)$ become $\alpha(1)$ and $\tau(1)$ respectively.

We encode g^x as $\alpha(x)$, $g^{1/x+\gamma^i}$ as $\alpha(1/(x+\gamma^i))$ and g^x as $\alpha(x)$. Since g is a generator of G_1, there exists a $y \in \mathbb{Z}_q$ such that $g^y = u$. So we choose y at random and set $u = \alpha(y)$.

To break the n-HSDHE assumption, the adversary needs to output a triple (A, B, C) of the form $(g^{1/x+c}, g^c, u^c)$ for some $c \in \mathbb{Z}_q$. Normally, we can test that the triple is well-formed using the bilinear map: $e(A, g^x B) = e(g, g) \wedge e(C, g) = e(u, B)$.

In the generic group model we require that he outputs random representations $(\alpha_A, \alpha_B, \alpha_C)$. The adversary can either compute these values using the group oracles, or pick them at random, which he could also do by doing a generic exponentiation with a random constant. Thus it is meaningful to say that the adversary succeeds if $\alpha_A = \alpha(1/(x + c)) \wedge \alpha_B = \alpha(c) \wedge \alpha_C = \alpha(yc)$ for some c.

Theorem 3 (Weakened n-HSDHE is Hard in the Generic Group Model). *Let G and G_T be groups of prime order q, (where q is a k-bit prime) with bilinear map $e : G \times G \to G_T$. We choose maps α and τ at random. There exists a negligible function $\nu : \mathbb{N} \to [0, 1]$ such that for every PPTM $\mathcal{A}$:*

$$\Pr[x, y, \gamma \leftarrow \mathbb{Z}_q; (\alpha_A, \alpha_B, \alpha_C) \leftarrow \mathcal{A}(\alpha(1), \alpha(x), \alpha(y), \{\alpha(1/(x+\gamma^i))\}_{i=1\ldots n}, \gamma) : \exists c : \alpha_A = \alpha(1/x + c) \wedge \alpha_B = \alpha(c) \wedge \alpha_C = \alpha(yc)] \leq \nu(k)$$

Proof. Let $\mathcal{A}$ be a PPTM that can break the n-HSDHE assumption. We create an environment $\mathcal{R}$ that interacts with $\mathcal{A}$ as follows:

$\mathcal{R}$ maintains two lists: $L_\alpha = \{(F_{\alpha,s}, \alpha_s) : s = 0, \ldots, S_\alpha - 1\}$ and $L_\tau = \{(F_{\tau,s}, \tau_s) : s = 0, \ldots, S_\tau - 1\}$. The $F_{\alpha,s}$ and $F_{\tau,s}$ contain *rational functions*; their numerators and denominators are polynomials in $\mathbb{Z}_q[x, y]$. $\mathcal{R}$ uses $F_{\alpha,s}$ and $F_{\tau,s}$ to store the group action queries that $\mathcal{A}$ makes and α_s, τ_s to store the results. Thus $\alpha_s = \alpha(F_{\alpha,s})$ and $\tau_s = \tau(F_{\tau,s})$.

$\mathcal{R}$ chooses random strings $\alpha_0, \alpha_1, \alpha_2, \{\alpha_{2+i}\}_{i=1\ldots 4n-1}, \tau_0 \in \{0, 1\}^*$, and sets the corresponding polynomials as:

$$F_{\alpha,0} = 1 \qquad F_{\alpha,1} = x \qquad F_{\alpha,2} = y$$
$$F_{\alpha,2+i} = 1/(x + \gamma^i) \quad (1 \leq i \leq n) \qquad F_{\tau,0} = 1$$

$\mathcal{R}$ sets $S_\alpha = n + 3$ and $S_\tau = 1$. Then $\mathcal{R}$ sends the strings to $\mathcal{A}$. Whenever $\mathcal{A}$ calls the group action oracle, $\mathcal{R}$ updates its lists.

Multiplication. $\mathcal{A}$ inputs α_s and α_t. $\mathcal{R}$ checks that α_s and α_t are in its list L_α, and returns $\perp$ if they are not. Then $\mathcal{R}$ computes $F = F_{\alpha,s} + F_{\alpha,t}$. If F is already in the list L_α, then $\mathcal{R}$ returns the appropriate α_v. Otherwise, $\mathcal{R}$ chooses a random α_{S_α}, sets $F_{\alpha,S_\alpha} = F$ and adds this new tuple to the list. $\mathcal{R}$ increments the counter S_α by 1. $\mathcal{R}$ performs a similar operation if the inputs are in G_T.

Division. $\mathcal{A}$ inputs α_s and α_t. $\mathcal{R}$ checks that α_s and α_t are in its list L_α, and returns $\perp$ if they are not. Then $\mathcal{R}$ computes $F = F_{\alpha,s} - F_{\alpha,t}$. If F is already in the list L_α, then $\mathcal{R}$ returns the appropriate α_v. Otherwise, $\mathcal{R}$ chooses a random α_{S_α}, sets $F_{\alpha,S_\alpha} = F$ and adds this new tuple to the list. $\mathcal{R}$ increments the counter S_α by 1. $\mathcal{R}$ performs a similar operation if the inputs are in G_T.

Exponentiation by a Constant $\mathcal{A}$ inputs α_s and a constant $a \in \mathbb{Z}_q$. $\mathcal{R}$ checks that α_s is in its list L_α, and returns $\perp$ if it is not. Then $\mathcal{R}$ computes $F = F_{\alpha,s} \cdot a$. If F is already in the list L_α, then $\mathcal{R}$ returns the appropriate α_v. Otherwise, $\mathcal{R}$ chooses a random α_{S_α}, sets $F_{\alpha,S_\alpha} = F$ and adds this new tuple to the list. $\mathcal{R}$ increments the counter S_α by 1. $\mathcal{R}$ performs a similar operation if the inputs are in G_2 or G_T.

Pairing. $\mathcal{A}$ inputs α_s and α_t. $\mathcal{R}$ checks that α_s and α_t are in its list L_α and returns $\perp$ if they are not. Then $\mathcal{R}$ computes $F = F_{\alpha,s} \cdot F_{\alpha,t}$. If F is already in the list L_τ, then $\mathcal{R}$ returns the appropriate τ_v. Otherwise, $\mathcal{R}$ chooses a random τ_{S_τ}, sets $F_{\tau,S_\tau} = F$ and adds this new tuple to the list. $\mathcal{R}$ increments the counter S_τ by 1.

At the end of the game, $\mathcal{A}$ outputs $(\alpha_A, \alpha_B, \alpha_C)$. These values must correspond to bivariate polynomials $F_{\alpha,A}$, $F_{\alpha,B}$ and $F_{\alpha,C}$ in our lists. (If one of these values is not in our lists, then $\mathcal{A}$ must have guessed a random group element; he might as well have asked the oracle to perform exponentiation on a random constant and added a random value to the list. Thus we ignore this case.)

Since $\mathcal{A}$ must have computed these polynomials as a result of oracle queries, they must be of the form $a_0 + a_1 x + a_2 y + \sum_{i=1}^{n} a_{3,i}/(x + \gamma^i)$. If $\mathcal{A}$ is to be successful,

$$F_{\alpha,A}(x + F_{\alpha,B}) = 1 \text{ and} \tag{7}$$

$$F_{\alpha,C} = y F_{\alpha,B}. \tag{8}$$

For Equation (8) to hold identically in $\mathbb{Z}_q[x, y]$, $F_{\alpha,C}$ and $F_{\alpha,B}$ either need to be 0 or $F_{\beta,B}$ needs to be constant, because the only possible term for y in the polynomials is $y(a_2 + \sum_{i=1}^{n} a_{5,i}\gamma^i)$. In both cases, the term $(x + F_{\beta,B})$ in Equation (8) has maximum degree 1, and Equation (7) can only be satisfied identically in $\mathbb{Z}_q[x, y]$ if $F_{\alpha,A}$ has at least degree $q - 1$ in variable x. We know that the degree of $F_{\alpha,A}$ is at most n and conclude that there exists an assignment in $\mathbb{Z}_q$ to the variables x and y for which the Equations (7) and (8) do not hold. Since Equation (7) is a non-trivial polynomial equation of degree $\leq 2n$, it admits at most $2n$ roots in $\mathbb{Z}_q$.

Analysis of $\mathcal{R}$'s Simulation. At this point $\mathcal{R}$ chooses a random $x^*, y^* \in \mathbb{Z}_q^*$, and now sets $x = x^*$ and $y = y^*$. Using Equations (9), (10), and (11) $\mathcal{R}$ now tests if its simulation was perfect; that is, if the instantiation of x by x^* or y by y^* does *not* create any equality relation among the polynomials that was not revealed by the random strings provided to $\mathcal{A}$. Thus, $\mathcal{A}$'s overall success is bounded by the probability that any of the following holds:

$$F_{\alpha,i}(x^*, y^*) - F_{\alpha,j}(x^*, y^*) = 0 \text{ in } \mathbb{Z}_q, \text{for some } i, j \text{ and } F_{\alpha,i} \neq F_{\alpha,j}, \tag{9}$$

$$F_{\tau,i}(x^*, y^*) - F_{\tau,j}(x^*, y^*) = 0 \text{ in } \mathbb{Z}_q, \text{for some } i, j \text{ and } F_{\tau,i} \neq F_{\tau,j}, \tag{10}$$

$$F_{\alpha,A}(x^*, y^*)(x^* + F_{\beta,B}(x^*, y^*)) = 1 \wedge F_{\alpha,C}(x^*, y^*) = y^* F_{\beta,B}(x^*, y^*) \text{ in } \mathbb{Z}_q. \tag{11}$$

Each polynomial $F_{\alpha,i}$ and $F_{\tau,i}$ has degree at most n or $2n$, respectively. For fixed i and j, we satisfy Equations (9) with probability $\leq n/(q-1)$ and Equations (10) with probability $\leq 2n/(q-1)$. We can bound the probability that Equation (11) holds by $\leq 2n/(q-1)$.

Now summing over all (i,j) pairs in each case, we bound $\mathcal{A}$'s overall success probability

$$\epsilon \leq 2\binom{S_\alpha}{2}\frac{n}{q-1} + \binom{S_\tau}{2}\frac{2n}{q-1} + \frac{2n}{q-1} \leq \frac{2n}{q-1}(\binom{S_\alpha}{2} + \binom{S_\tau}{2} + 1).$$

Let n_G be the total number of group oracle queries made, then we know that $S_\alpha + S_\tau = n_G + n + 6$. We obtain that $\epsilon \leq (n_G+n+6)^2\frac{2n}{q-1} = O(n_G^2 n/q + n^3/q)$. □

The following corollary restates the above result:

Theorem 4. *Any adversary that breaks the weakened n-HSDHE assumption with constant probability $\epsilon > 0$ in generic bilinear groups of order q such that $n < O(\sqrt[3]{q})$ requires $\Omega(\sqrt[3]{\epsilon q/n})$ generic group operations.*

B Protocol To Prove Knowledge of a Signature

Here we explain the protocol

$$PK\{(c, s, r, open, mult, tmp, m_1, \ldots, m_\ell) :$$
$$B = h^r \tilde{h}^{open} \ \wedge \ 1 = B^c h^{-mult} \tilde{h}^{-tmp} \ \wedge$$
$$\frac{e(h_0,h)}{e(A,y)} = e(A,h)^c \cdot e(\tilde{h},y)^{-r} \cdot e(\tilde{h},h)^{-mult} \cdot (\prod_{i=1}^{\ell} e(h_i,h)^{-m_i}) \cdot e(h_{\ell+1},h)^{-s}\} \ .$$

as given in Section 2.2.

The first statement proves the prover's knowledge of values r, *open* for opening the Pedersen commitment $B = h^r \tilde{h}^{open}$. Assuming that computing $\log_h \tilde{h}$ is hard (this is implied by n-SDH), the next statements assert the prover's knowledge of values c, *mult*, and *tmp* such that $mult = rc$ and $tmp = open \cdot c$.

Let us consider the last line of the proof. It asserts the prover's knowledge of values $c, s, m_1, \ldots, m_\ell$ such that

$$e(A,y) \cdot e(A,h)^c \cdot e(u,y)^{-r} \cdot e(\tilde{h},h)^{-cr} = e(h_0,h) \prod_{i=1}^{\ell} e(h_i,h)^{m_i} \cdot e(h_{\ell+1},h)^s$$

holds. Here we have made use of the relation $mult = rc$. By simplifying this equation we obtain

$$e(A\tilde{h}^{-r}, yh^c) = e(h_0 \prod_{i=1}^{\ell} h_i^{m_i} \cdot h_{\ell+1}^s, h).$$

Clearly $(A\tilde{h}^{-r}, c, s)$ fulfills the verification equation of the signature scheme for messages $m_1, \ldots, m_\ell$.

Also note that the values A and B are random group elements.

C Proofs

C.1 Proof of Theorem 1

It is standard to show that from a convincing prover of the protocol

$$PK\{(r, r', r'', r''', open, mult, tmp) :$$

$$D = g^r \tilde{h}^{open} \wedge 1 = D^{r''} g^{-mult} \tilde{h}^{-tmp} \wedge \quad (12)$$

$$\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g,g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''} e(\tilde{h}, \tilde{h})^{-mult} e(\tilde{h}, \mathcal{S})^r \wedge \quad (13)$$

$$\frac{e(\mathcal{G}, acc_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, acc_V)^r e(1/g, \tilde{h})^{r'} \wedge \quad (14)$$

$$\frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^r e(1/g, \tilde{h})^{r'''} \} . \quad (15)$$

one can with overwhelming probability extract values r, r', r'', r''', and *mult* such that the Equations (14), (13), (15) hold. From Equation (14) we learn through simple transformation that $\frac{e(\mathcal{G}\tilde{h}^{-r}, acc_V)}{e(g, \mathcal{W}\tilde{h}^{-r'})} = z$. We distinguish three cases: In the first case $\mathcal{G}\tilde{h}^{-r}$ corresponds to a g_i in $state_U$ and $i \in V$. In this case the extraction was successful.

In the second case $\mathcal{G}\tilde{h}^{-r}$ corresponds to a g_i in $state_U$ but $i \notin V$. In this case we can use a successful prover to break the n-DHE assumption. The reduction obtains as input the parameters of a bilinear map $params_{BM} = (q, G, G_T, e, g)$, and an instance of the n-DHE assumption $(g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in G^{2n-1}$. It provides the prover with $pk_A = (params_{BM}, pk, z = e(g_1, g_n))$, acc_V and $state_U = (U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. The reduction computes the additional setup for the proof using a fresh sk. Given a successful prover it can extract r, r', r'', r''', and *mult* such that

$$e(\mathcal{G}\tilde{h}^{-r}, acc_V) = e(g, \mathcal{W}\tilde{h}^{-r'})z$$

and

$$e(g, \prod_{j \in V} g_{n+1-j+i}) = e(g, \mathcal{W}\tilde{h}^{-r'} g_{n+1}) .$$

This means that

$$g_{n+1} = \frac{\prod_{j \in V} g_{n+1-j+i}}{\mathcal{W}\tilde{h}^{-r'}} .$$

For $i \in \{1, \ldots, n\} \setminus V$, all $g_{n+1-j+i}$ are contained in $state_U$ and it is possible to compute this value. This breaks the n-DHE assumption (Consult also the proof in Section 3.2).

In the third case $\mathcal{G}\tilde{h}^{-r}$ does not correspond to a g_i in $state_U$. We will show that we can use such a prover to break the dedicated signature scheme (more concretely the n-HSDHE assumption) using the remaining Equations (13) and (6). The reduction works as follows. On input a HSDHE instance $(g, g^x, u, \{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}, \{g^{\gamma^i}\}_{i=n+2\ldots 2n})$, the reduction uses the g^{γ^i}

to build $state_U$ and the remaining values to construct the additional setup for the proof by setting $pk = g^x$ (and implicitly $sk = x$).

After extracting r, r', r'', r''', *open*, *mult* and *tmp* note that (based on Equation (12)) $mult = rr''$ and $tmp = openr''$ (or one can compute $\log_g h$ which would in turn allow us to break n-HSDHE). After obtaining a $\mathcal{G}\tilde{h}^{-r}$ that does not correspond to a value in $\{g^{\gamma^i}\}_{i=1\ldots n}$ it is easy to see from Equation (13) that $e(pk\mathcal{G}\tilde{h}^{-r}, \mathcal{S}\tilde{h}^{-r''}) = 1$. Let $c = \log_g \mathcal{G}\tilde{h}^{-r}$. then $\mathcal{S}\tilde{h}^{-r''} = g^{1/(x+c)}$. Similarly from Equation (15) we learn that $\frac{e(\mathcal{G}\tilde{h}^{-r}, u)}{e(g, \mathcal{U}\tilde{h}^{-r'''})} = 1$. If $\mathcal{G}\tilde{h}^{-r} = g^c$, then $\mathcal{U}\tilde{h}^{-r'''} = u^c$. This contradicts the n-HSDHE assumption.

As the malicious prover has no way to distinguish between the first or the second reduction (as well as the real setup), we can randomly pick one of the two reductions to break either the n-DHE or the n-HSDHE assumption (we only loose a factor of 1/2 in the tightness of the reduction). □

C.2 Proof of Theorem 2

We extract the value from the above proof. From Equation (1) we know that if $mult \neq \rho c$ or $mult' \neq rr''$, we can compute the discrete logarithm $\log_g h$. This contradicts the DL assumption.

From Equations (3,4,5,6) and the security of the accumulator proof protocol in Section 3.3 we know that $\mathcal{G}\tilde{h}^{-r}$ equals a g_i, $i \in V$, such that $\mathcal{W}\tilde{h}^{-r'}$ is a verifying accumulator witness for this value. Otherwise we break the n-DHE or the n-HSDHE assumption. (The reductions would be set up in the same way as in Appendix C.1.)

Now we consider Equation (2) of the proof. It asserts the prover's knowledge of values $m_1, \ldots, m'_\ell$ such that

$$e(h_0 \cdot \prod_{j=q}^{\ell'} h_j^{m_j} \cdot \prod_{j=\ell'+1}^{\ell} h_j^{m_j} \cdot \mathcal{G}, g) e(\tilde{h}, y)^{\rho} \cdot e(\tilde{h}, g)^{\rho c} \cdot e(h_{\ell+1}, g)^{s} = e(A, y) e(A, g)^{c} \cdot e(\tilde{h}, g)^{r} \ .$$

Here we have made use of the relation $mult = \rho c$. By simplifying this equation further we obtain

$$e(h_0 \cdot \prod_{j=q}^{\ell'} h_j^{m_j} \cdot \prod_{j=\ell'+1}^{\ell} h_j^{m_j} \cdot {h_{\ell+1}}^{s} \cdot \mathcal{G}/\tilde{h}^{r}, g) = e(A/\tilde{h}^{\rho}, y g^{c}) \ .$$

This shows that $(A/\tilde{h}^{\rho}, c, s)$ is a valid adapted signature for $(m_1, \ldots, m_\ell, \tilde{g}^{\gamma^i})$. □

D Computing Accumulator Parameters on the Fly

The accumulator presented in Section 3 allows for efficient witness update but it has public parameters of considerable size. The maximum number of values that

might be accumulated must be estimated when the accumulator is instantiated and envisioned usage scenarios would require a number of elements in the order of 2^30. We believe that modern computers can easily handle such a large number of elements

However, in this section we show how in the special case where at a given point in time we never accumulate values greater than $\tilde{n} \leq n$ we can bring the cost of storing public parameters down to $O(\tilde{n})$. We achieve this through lazy evaluation of the accumulator parameters.

Let $\tilde{n}$ be the greatest accumulated value. For a value $i \in V$ we need (1) the value g_{n+1-i} to add i to the accumulator, (2) the value g_i to verify membership in the accumulator, and (3) values $g_{n+1-j+i}$ for $j \in (V \setminus V_w) \cup (V_w \setminus V)$ to compute and update witnesses. Note that for all $j \in V \cup V_w$, j is smaller than $\tilde{n}$. We claim that the values $\{g_i, g_{n+1-i}, g_{n+i}\}_{i=1}^{\tilde{n}} \setminus \{g_{n+1}\}$ are sufficient for all of these operations. Obviously the first two requirements are met. Moreover, if $j > i$, then $g_{n+1-j+i} \in \{g_{n+1-i}\}_{i=1}^{\tilde{n}}$. Otherwise, $1 \leq j < i$ and $g_{n+1-j+i} \in \{g_{n+i}\}_{i=1}^{\tilde{n}}$. This establishes the third requirement.

In the ObtainCert protocol our revocation scheme $\tilde{n}$ is increased by one and the missing accumulator parameters are added to $state_U$. Note that $U = \{1, \ldots, \tilde{n}\}$. The updated state is then used to update the epoch information $epoch_V$ and the witnesses wit_i.