Scalable Revocation Scheme for Anonymous Credentials Based on n-times Unlinkable Proofs

Jan Camenisch
IBM Research - Zurich
Zurich, Switzerland
jca@zurich.ibm.com

Manu Drijvers
IBM Research - Zurich and
Dept. of Computer Science,
ETH Zurich
Zurich, Switzerland
mdr@zurich.ibm.com

Jan Hajny
Brno University of Technology
Brno, Czech Republic
hajny@feec.vutbr.cz

ABSTRACT

We propose the first verifier-local revocation scheme for privacy-enhancing attribute-based credentials (PABCs) that is practically usable in large-scale applications, such as national eID cards, public transportation and physical access control systems. By using our revocation scheme together with existing PABCs, it is possible to prove attribute ownership in constant time and verify the proof and the revocation status in the time logarithmic in the number of revoked users, independently of the number of all valid users in the system. Proofs can be efficiently generated using only offline constrained devices, such as existing smart-cards. These features are achieved by using a new construction called n-times unlinkable proofs. We show the full cryptographic description of the scheme, prove its security, discuss parameters influencing scalability and provide details on implementation aspects. As a side result of independent interest, we design a more efficient proof of knowledge of weak Boneh-Boyen signatures, that does not require any pairing computation on the prover side.

Keywords

Revocation, attribute-based credentials, privacy, smart-cards, blacklisting, eID, e-ticketing.

1. INTRODUCTION

Privacy-enhancing attribute-based credentials (PABCs) allow users to anonymously prove their personal attributes, such as age, citizenship, birthplace, etc. In typical PABC applications, such as national eIDs and e-ticketing, programmable smart-cards are used to store user attributes and execute the proving protocol. Technologies like IBM Identity Mixer and Microsoft U-Prove [14, 39] based on privacy-enhanced group signatures [6, 7, 18] enhance the attribute proofs by providing additional privacy-protecting features, such as the selective disclosure of attributes, untraceability and unlinkability of verification sessions.

WPES'16, October 24 2016, Vienna, Austria

ISBN 978-1-4503-4569-9/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2994620.2994625

When using PABCs, the revocation of users' credentials is required on a daily basis. Both the revocation of malicious users and the revocation of honest users, who for example lose their smart-cards, needs to be done. However, the strong protection of users' privacy makes it difficult to do the revocation. It is a non-trivial problem to find and exclude a particular user if all users' transactions are anonymous, unlinkable and untraceable. In fact, the missing practical revocation of PABCs is currently one of the most significant technological barriers preventing the large-scale deployment of PABC technology [27, 29, 34, 44].

So far, many schemes for PABC revocation have been proposed. These proposals can be divided into the following categories:

- **Blacklisting of Credential Identifiers [39]**: a constant unique identifier is embedded to each attribute proof of a particular user. If the user needs to be revoked, the identifier is put on a blacklist. The main disadvantage of this approach is that all the user's transactions are linkable because they all share the same identifier. Thus, one of the crucial features of PABCs, the unlinkability of proofs, cannot be provided. Blacklisting of identifiers is used by U-Prove.

- **Blacklisting of Keys and Revocation Handles [3, 8, 41]**: similarly as the identifiers, also the users' keys and revocation handles can be used for revocation. In this case, a private attribute (user's key or dedicated handle) stored in a credential gets blacklisted. This approach has the advantage that the unlinkability of proofs can be still provided because the private attributes are never disclosed in an open form. The main disadvantage of most schemes using this technique is that the user key (or handle) must be known by the entity that initiates the revocation process. Thus, achieving verifier-driven and issuer-driven revocation is difficult. Furthermore, the computational complexity of the revocation status verification is linearly dependent on the number of revoked users. The BLAC scheme [41] provides the verifier-driven revocation at the cost of communication and computation complexity linear to the number of revoked users, for both users and verifiers in this case. Furthermore, BLAC needs costly bilinear pairing operations at verifiers' side. Blacklisting of keys is used in Direct Anonymous Attestation [8, 10, 11] and some implementations of Identity Mixer.

- **Epochs of Lifetime [15]**: a time interval (an epoch), in which the credential is valid, can be embedded to all attribute proofs. If this technique is used, the user must periodically update his credentials every time a new epoch starts. The disadvantage of this method is that users must renew their credentials and that the revocation is never immediate. The credential gets revoked only after its lifetime expires and is not renewed. Epochs of lifetime are one of the methods used by Identity Mixer.

- **Accumulator Proofs [17, 28, 31, 35]**: a user efficiently proves that he is on a whitelist of non-revoked users. In this case, if any user gets revoked, all other users must re-compute their whitelist membership witnesses. Therefore, the main disadvantage of this method is that the users must periodically update their credential information from a central authority, thus cannot stay completely offline.

- **Verifiable Encryption of Secrets [25, 37, 38]**: a user-specific revocation handle or a user key is embedded to each attribute proof in an encrypted form. The verifier is able to check that the identifier is present but is unable to decrypt. The decryption is done only in justified cases by a separate revocation authority. The verifier usually needs to interact with the revocation authority during each credential proof verification. Furthermore, the authority must be trusted not to decrypt in unjustified cases. This approach is therefore better suited for the de-anonymisation (inspection) than the revocation of users.

- **Hash Chains [26, 42]**: Nymble [26] was designed to provide fast revocation primarily in anonymous networks, like TOR. Although Nymble provides anonymity, unlinkability and backward anonymity, it lacks properties important for PABCs. In particular, Nymble lacks pseudonym-credential binding, is not primarily designed to support permanent revocation and inspection and is suitable rather for online applications than smart-card-based ones because users need to obtain and store unique pseudonyms for all their future transactions (which is unrealistic with current smart-card memory size). Nymble's proving protocol has computational complexity linear to the number of revoked users and users need to communicate with a revocation authority during all transactions (this was changed to distributed blacklist updates in [42]).

- **Combined Techniques [32, 43]**: the scheme of Lueks et al. [32] combines the epoch-based technique with the blacklisting of revocation handles. It provides verifier-local revocation (VLR) with constant verifiers' and users' proving times. Nevertheless, the unlinkability property is provided only if users prove their credentials to the same verifier no more than once in an epoch. At the same time, the revocation authority must re-compute the revocation list for all verifiers individually in each epoch. Therefore, it is very difficult to make epochs short enough so that the unlinkability is truly achieved and the scheme remains practical at the same time. An extension presented in the original paper allows full multi-show unlinkability but suffers from vulnerabilities to attacks on credential indistinguishability. The scheme by Verheul [43] further extends the idea of the global-mode revocation presented in [32] by adding the protection against the attacks on the unlinkability property. Unfortunately, the scheme retains the negative properties of the original schemes and has unrealistic requirements on existing computational devices, i.e., it requires bilinear pairing on smart-cards and number of pairings at verifier's side that is linear to the number of all revoked credentials.

Each of these revocation techniques has its pros and cons and is suitable for particular applications. However, none of the existing schemes is able to simultaneously preserve all the privacy-enhancing features of PABCs, be implemented on offline constrained devices and be scalable enough to be practically implemented in large-scale applications with millions of users and thousands of revoked users. This virtually prevents PABCs from being used in applications such as national eIDs, public transportation and large physical access control systems.

1.1 Our Contribution

In this paper, we propose the first revocation scheme compatible with existing PABC technologies that provides: 1) verifier-local revocation (VLR) [5]: revocation of invalid users that does not affect remaining users, 2) anonymity, selective attribute disclosure, untraceability and unlinkability: all the core privacy-enhancing features of PABCs, 3) user-, verifier- and issuer-driven revocation: revocation that can be initiated by all system entities, 4) scalability: the scheme can be parametrized to suit applications with few or millions of users, 5) speed: a verifier's computational complexity is independent of the number of valid users in the system and only logarithmic in the number of revoked users. Using fixed system parameters, a user's complexity is constant and good enough for the implementation on existing smart-cards.

We achieve this by letting the user construct n unlinkable pseudonyms, computed with a verifiable-random function on input that is proven to be a member of a certain set. The revocation authority can compute all n pseudonyms of a user, which lets a verifier efficiently perform the revocation check. This construction resembles the traceable signatures by Chow [21], where it is used to let the group manager efficiently find all signatures by one member.

We introduce a new efficient proof of knowledge of weak BB-signatures which is of independent interest. We estimate that computing the proof is twice as fast compared to the existing proof of knowledge, and the prover does not compute any pairings. These proofs-of-knowledge allow for more efficient set membership proofs and range proofs, and can be used to severely improve the efficiency of many existing works [9, 13, 21].

1.2 Paper Outline

In Sec. 2, we specify the notation and the main cryptographic building blocks we use. In Sec. 3, we show the general architecture of revocation based on n-times unlinkable proofs and provide the security model. In Sec. 4, we present our revocation scheme and prove its security. In Sec. 5, we provide more details on implementation aspects, including the choice of system parameters and performance analysis.

2. PRELIMINARIES

This section introduces the cryptographic preliminaries used in our scheme. In addition, we present an improved way of proving knowledge of a weak Boneh-Boyen signature, which is of independent interest.

2.1 Notation

We describe proof of knowledge protocols (PK) using the efficient notation introduced by Camenisch and Stadler [20]. The protocol for proving the knowledge of discrete logarithm of c with respect to g is denoted as $\text{PK}\{\alpha : c = g^{\alpha}\}$. The proof of discrete logarithm equivalence with respect to different generators g_1, g_2 is denoted as $\text{PK}\{\alpha : c_1 = g_1^{\alpha} \wedge c_2 = g_2^{\alpha}\}$. The symbol ":" means "such that", "|" means "divides", "$|x|$" is the bitlength of x and "$x \in_R \{0,1\}^l$" is a randomly chosen bitstring of maximum length l. We write $a \xleftarrow{\$} A$ when a is sampled uniformly at random from A.

We use signatures with efficient protocols [18, 2] as the main building block in our scheme. Using these signature schemes, one can prove knowledge of a signature and a message such that the signature signs the message rather than revealing the signature. This allows one to convince a verifier that a message is signed multiple times in an unlinkable manner. In the description of our revocation scheme, we use a generic signature scheme with efficient protocols denoted by Σ as a modular building block. We use Σ.`KeyGen`, Σ.`Sign` and Σ.`Verify` to refer the to `KeyGen`, `Sign` and `Verify` algorithms of Σ. Σ can be instantiated by, e.g., CL02 [18], CL04 [19], or weak Boneh-Boyen signatures [4], where the latter would yield the most efficient scheme.

2.2 Weak Boneh-Boyen Signature

Weak Boneh-Boyen signatures [4] are unforgeable against a weak chosen message attack under the qSDH assumption. A signature can only sign a single message, but are more efficient and smaller in size than CL02 signatures. The signature cannot be randomized, but it does allow for efficient proofs-of-knowledge.

Setup: input security parameter l_q, generate groups $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle, \mathbb{G}_T = \langle e(g_1, g_2) \rangle$ of prime order $q : |q| = l_q$ with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Take $x \xleftarrow{\$} \mathbb{Z}_q$, compute $w = g_2^x$, and output $sk = x$ as private key and $pk = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, w)$ as public key.

Sign: input message $m \in \mathbb{Z}_q$, pk, and sk, output $\sigma = g_1^{\frac{1}{x+m}}$.

Verify: input the signature σ, m and pk, output 1 iff $e(\sigma, w) \cdot e(\sigma^m, g_2) = e(g_1, g_2)$ holds.

2.3 Set Membership and Range Proofs

Camenisch et al. [9] introduce a zero-knowledge proof to prove m is part of a public set $S = \{m_1, \ldots, m_l\}$, denoted $SPK\{(m) : m \in S\}$. They introduce an entity that uses a signature scheme with efficient protocols to sign every element of S and publishes these signatures. A prover proves knowledge of a signature on m. By unforgeability of the signature scheme and soundness of the zero-knowledge proof, this guarantees that m is in the set S. Camenisch et al. instantiate the signature scheme with weak Boneh-Boyen signatures [4].

One application of set membership proofs are range proofs. By taking the set S as all numbers in a range, a prover can prove that his witness lies in the range.

2.4 Zero-Knowledge Proofs for Boneh-Boyen Signatures

Camenisch et al. [9] propose proving knowledge of a weak Boneh-Boyen signature σ on m under key $w = g_2^x$ by taking a random $r \xleftarrow{\$} \mathbb{Z}_q$, $\sigma' \leftarrow \sigma^r$ and proving $\pi \leftarrow SPK\{(m, r) : e(\sigma', w) = e(\sigma', g_2)^{-m} e(g_1, g_2)^r\}$. Computing proof (σ', π) takes one pairing operation and three exponentiation in $\mathbb{G}_1$, verification requires two pairing computations and two exponentiations in $\mathbb{G}_T$.

We now present a more efficient way of proving knowledge of a weak Boneh-Boyen signature: The prover takes random $r \xleftarrow{\$} \mathbb{Z}_q^*$, sets $\sigma' \leftarrow \sigma^r$ and $\bar{\sigma} \leftarrow \sigma'^{-m} g_1^r$. Prove $\pi \leftarrow SPK\{(m, r) : \bar{\sigma} = \sigma'^{-m} g_1^r\}$. A proof consists of $(\sigma', \bar{\sigma}, \pi)$. To verify, check $\sigma' \neq 1_{\mathbb{G}_1}$, verify $\pi \in SPK\{(m, r) : \bar{\sigma} = \sigma'^{-m} g_1^r\}$ and $e(\bar{\sigma}, g_2) = e(\sigma', w)$. Computing this proof takes 5 exponentiations in $\mathbb{G}_1$. Verification requires two pairings and three exponentiations in $\mathbb{G}_1$.

Note that our method does not require the prover to work in $\mathbb{G}_2$ or $\mathbb{G}_T$, nor does it compute pairings. This severely improves the efficiency and the ease of implementation. By timing the primitive operations[1], we expect that generating our proof requires less than half of the time compared to the original proof. In addition, it can be executed on smart cards that support ECC, whereas it is currently hard to find smart cards that support pairings. The verification cost increases with less than five percent.

We must show that our proof is in fact a valid zero-knowledge proof of knowledge of a weak Boneh-Boyen signature. We have two extra constraints: First, we can only prove knowledge of signatures $\sigma \neq 1_{\mathbb{G}_1}$. An honest signer only creates signatures equal to one with negligible probability. Second, a pair $(\bar{g}, \bar{g}^x) \in \mathbb{G}_1^2$ must be available to the simulator in order to prove zero-knowledge. This constraint is easy to fulfill, the signer can simply create such a pair and add it to his public key. Note that when reducing the unforgeability of the weak BB signature to the qSDH assumption, private key x is not known, but the qSDH instance contains (g_1, g_1^x), which allows for a pair $(\bar{g}, \bar{g}^x)$ to be simulated.

LEMMA 2.1. *Our construction forms a zero-knowledge proof of knowledge of a weak Boneh-Boyen signature $\sigma \neq 1_{\mathbb{G}_1}$.*

We prove this lemma in Appendix A.

3. REVOCATION USING N-TIMES UNLINKABLE PROOFS

We use the standard architecture of PABC schemes involving Users (U), Issuers (I), Verifiers (V) and the Revocation Authority (RA), as depicted in Fig. 1. The revocation scheme creates a revocation handle, that will be placed as an attribute in every credential of the PABC scheme. A user presenting a PABC credential will also prove that the revocation handle in his credential is not revoked. As demonstrated by Camenisch et al. [12], a revocation scheme can be seen as an extension to a PABC scheme. We analyze the security of a revocation scheme individually.

[1]Measured using the IAIK ECCelerate library on 256-bit Barreto-Naehrig Curves.

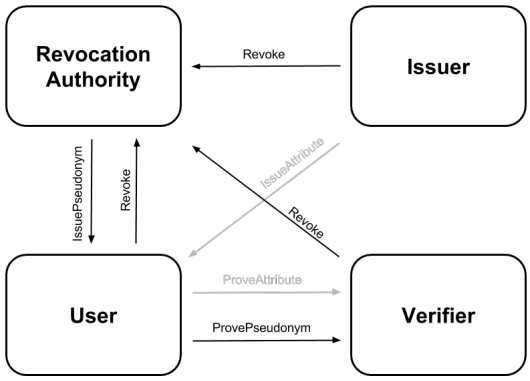

Figure 1: Architecture of standard PABC scheme with revocation.

Our approach to revocation lets a user first go to the RA to receive a revocation handle w, while the RA adds information to its list of revocation handles rh. A user can create n unlinkable pseudonyms in each $epoch$[2] using w and prove that these are correctly formed. The RA can revoke the user by computing all pseudonyms the user can make using the information from rh, and placing the pseudonyms on individual epoch's revocation lists RL_{epoch}. A verifier performing a revocation check will check that a presented pseudonym is not part of RL_{epoch}. We first present the algorithms in more detail, after which we present the security definitions of a revocation scheme using n-times unlinkable proofs.

3.1 Algorithms

A revocation scheme consists of algorithms `IssuePseudonym`, `ReceivePseudonym`, `ProvePseudonym`, `VerifyPseudonym` and `Revoke`. Other related algorithms, such as the `IssueAttribute` and `ProveAttribute`, are marked grey in Fig. 1 and are not covered further in the text as existing PABC schemes are expected to be used.

- $(spar, pk_{RA}, sk_{RA}) \leftarrow$ `Setup`$(1^{\mathcal{K}}, n)$: inputs a security parameter $1^{\mathcal{K}}$ and parameter n for the n-times unlinkable proofs, outputs public system parameters $spar$ and a public key pk_{RA}, and as RA's private output a private key sk_{RA}.
- $(rh') \leftarrow$ `IssuePseudonym`$(spar, sk_{RA}, rh) \leftrightarrow$ `ReceivePseudonym`$(spar, pk_{RA}) \rightarrow (w)$: RA's inputs are the system parameters $spar$, the RA's key sk_{RA} and the list of revocation handles rh, the User's inputs are just the public parameters. The user's output is the revocation handle w[3], that the user will use to create the n-times unlinkable proofs. RA's private output is the updated list of revocation handles rh'.
- $(C, \pi, c) \leftarrow$ `ProvePseudonym`$(spar, w, epoch, ctr)$: inputs system parameters $spar$, a revocation handle w, a current epoch identifier $epoch$, a counter ctr, and outputs a randomized pseudonym C, a commitment c, and proof π that proves that the pseudonym and commitment are correctly formed. Commitment c is a commitment to revocation handle w, that will be used to bind the revocation handle to the PABC scheme.
- $(0/1) \leftarrow$ `VerifyPseudonym` $(spar, pk_{RA}, C, \pi, c, epoch, RL_{epoch})$: inputs system parameters $spar$, RA's public key pk_{RA}, a pseudonym C with corresponding proof π and commitment c, an epoch with corresponding revocation list RL_{epoch} and outputs 1 iff the pseudonym and proof are valid, otherwise outputs 0.
- $(RL_{epoch}, rd', revoked') \leftarrow$ `Revoke`$(spar, rh, rd, \{C_R, epoch_R\}, epoch, revoked)$: inputs system parameters $spar$, a revocation handle list rh, a revocation database rd, pseudonym(s) for revocation C_R, their epoch identifiers $epoch_R$, next epoch identifier $epoch$ and the list of revoked handles $revoked$ and outputs the revocation list RL_{epoch} for the next epoch, the updated revocation database rd' and the updated list of revoked handles $revoked'$.

3.2 Security Model

As we take a new approach to revocation, the security requirements cannot be captured by existing definitions for revocation. For example, the definitions by Camenisch et al. [16] require revocation privacy when the RA is corrupt. Our revocation mechanism is based on the fact that the RA can compute all pseudonyms that a user can make, so we cannot guarantee privacy when the RA is corrupt. Furthermore, we use a revocation database rd that other revocation schemes do not use, again requiring us to create a different model.

Like Camenisch et al. [16], we have properties for revocation completeness, revocation soundness, and revocation privacy. Revocation completeness ensures that unrevoked users will always pass the revocation check. Revocation soundness states that whenever the RA is honest, users cannot claim to have an unrevoked revocation handle when they are in fact revoked. The adversary can play the role of corrupt users and request pseudonyms through the $\mathcal{O}^{\texttt{IssuePseudonym}}$ oracle, revoke users using $\mathcal{O}^{\texttt{Revoke}}$, and move to the next epoch with $\mathcal{O}^{\texttt{NextEpoch}}$. The adversary wins by creating a pseudonym and proof that pass verification in the current epoch, although the user is revoked. Revocation privacy ensures that honest users do not lose privacy by proving they posess an unrevoked revocation handle, when the RA is honest. Two honest users are created, and through oracle $\mathcal{O}^{\texttt{ProvePseudonym}}$ the adversary can request proofs of having an unrevoked revocation handle. Finally the challenger outputs a proof by one of the two honest users, and the adversary wins if he can decide which user created this proof. The full definition is presented in Fig. 2.

4. REVOCATION SCHEME

We now describe our instantiation of a revocation scheme based on n-times unlinkable proofs. On a high level, we limit the amount of pseudonyms a user can create by letting users create pseudonyms using a pseudo-random function on inputs from a limited set of 'randomizers'. Zero-knowledge proofs are used to prove that the pseudonyms are correctly formed and the randomizers are taken from a set. In this description, we let Σ denote a signature scheme with efficient protocols, i.e., there is an efficient zero-knowledge

[2]A time period in which the revocation list gets updated.
[3]A unique revocation handle embedded into a credential as an attribute. This same revocation handle would be placed in every credential of the PABC scheme. The handle creates the pseudonym - credential binding.

DEFINITION 3.1 (REVOCATION CORRECTNESS). *For every efficient adversary $\mathcal{A}$, there exists a negligible function ν such that the following holds:*

$$\Pr[\mathsf{VerifyPseudonym}(spar, pk_{RA}, C, \pi, c, epoch, RL) = \mathsf{reject} \ :$$
$$(spar, (sk_{RA}, pk_{RA})) \leftarrow \mathsf{Setup}(1^\kappa, n),$$
$$rh \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, rh) \leftrightarrow \mathsf{ReceivePseudonym}(spar, pk_{RA}) \rightarrow w,$$
$$ctr \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{IssuePseudonym}}, \mathcal{O}^{\mathtt{Revoke}}, \mathcal{O}^{\mathtt{NextEpoch}}}(spar, w), (C, \pi, c) \leftarrow \mathsf{ProvePseudonym}(spar, w, epoch, ctr)] \leq \nu(\kappa) \ ,$$

where the oracles $\mathcal{O}^{\mathtt{IssuePseudonym}}$, $\mathcal{O}^{\mathtt{Revoke}}$, and $\mathcal{O}^{\mathtt{NextEpoch}}$ are defined as follows.

$\mathcal{O}^{\mathtt{IssuePseudonym}}$**:** *On input* ($\mathtt{IssuePseudonym}$)*, the oracle runs algorithm $rh \cup \{w\} \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, rh)$ with $\mathcal{A}$ performing the user role. If the algorithm successfully terminates, the oracle updates $rh \leftarrow rh \cup \{w\}$.*

$\mathcal{O}^{\mathtt{Revoke}}$**:** *On input* ($\mathsf{Revoke}(w_i)$) *with $w_i \in rh$ and $w_i \neq w$, the oracle sets $revoked \leftarrow revoked \cup \{w_i\}$ and runs $(RL, rd, revoked) \leftarrow \mathsf{Revoke}(spar, rh, rd, \emptyset, epoch, revoked)$.*

$\mathcal{O}^{\mathtt{NextEpoch}}$**:** *On input* ($\mathtt{NextEpoch}$)*, the oracle increases epoch and runs $(RL, rd, revoked) \leftarrow \mathsf{Revoke}(spar, rh, rd, \emptyset, epoch, revoked)$.*

DEFINITION 3.2 (REVOCATION SOUNDNESS). *For every efficient adversary $\mathcal{A}$, there exists a negligible function ν such that the following holds:*

$$\Pr[\mathsf{VerifyPseudonym}(spar, pk_{RA}, C, \pi, c, epoch, RL) = \mathsf{accept} \ :$$
$$(spar, (sk_{RA}, pk_{RA})) \leftarrow \mathsf{Setup}(1^\kappa, n), \mathsf{ComOpenVf}(c, w, o)$$
$$(C, \pi, c) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{IssuePseudonym}}, \mathcal{O}^{\mathtt{Revoke}}, \mathcal{O}^{\mathtt{NextEpoch}}}(spar, pk_{RA}), w \in revoked] \leq \nu(\kappa) \ ,$$

where the oracles $\mathcal{O}^{\mathtt{IssuePseudonym}}$, $\mathcal{O}^{\mathtt{Revoke}}$, and $\mathcal{O}^{\mathtt{NextEpoch}}$ are defined as follows.

$\mathcal{O}^{\mathtt{IssuePseudonym}}$**:** *On input* ($\mathtt{IssuePseudonym}$)*, the oracle runs algorithm $rh \cup \{w\} \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, rh)$ with $\mathcal{A}$ performing the user role. If the algorithm successfully terminates, the oracle updates $rh \leftarrow rh \cup \{w\}$.*

$\mathcal{O}^{\mathtt{Revoke}}$**:** *On input* ($\mathsf{Revoke}(w_i)$) *with $w_i \in rh$, the oracle sets $revoked \leftarrow revoked \cup \{w_i\}$ and runs $(RL, rd, revoked) \leftarrow \mathsf{Revoke}(spar, rh, rd, \emptyset, epoch, revoked)$.*

$\mathcal{O}^{\mathtt{NextEpoch}}$**:** *On input* ($\mathtt{NextEpoch}$)*, the oracle increases epoch and runs $(RL, rd, revoked) \leftarrow \mathsf{Revoke}(spar, rh, rd, \emptyset, epoch, revoked)$.*

DEFINITION 3.3 (REVOCATION PRIVACY). *For every efficient adversary $\mathcal{A}$ there exists a negligible function ν such that for every epoch and $ctr \in \mathbb{Z}_n$, the following holds:*

$$\Pr[b' = b \ :$$
$$(spar, (sk_{RA}, pk_{RA})) \leftarrow \mathsf{Setup}(1^\kappa, n),$$
$$rh \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, \emptyset) \leftrightarrow \mathsf{ReceivePseudonym}(spar, pk_{RA}) \rightarrow w_0,$$
$$rh' \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, rh) \leftrightarrow \mathsf{ReceivePseudonym}(spar, pk_{RA}) \rightarrow w_1,$$
$$b \in_R \{0, 1\}, (C, \pi, c) \leftarrow \mathsf{ProvePseudonym}(spar, w_b, epoch, ctr)$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{ProvePseudonym}}}(spar, pk_{RA}, C, \pi, c)] \leq \nu(\kappa) \ ,$$

where oracle $\mathcal{O}^{\mathtt{ProvePseudonym}}$ is defined as follows.

$\mathcal{O}^{\mathtt{ProvePseudonym}}$**:** *On input* ($\mathtt{ProvePseudonym}, b'', epoch', ctr'$) *the oracle runs algorithm $\mathsf{ProvePseudonym}(spar, w_{b''}, epoch', ctr')$. If the algorithm successfully terminates with output (C, π, c), the oracle outputs (C, π, c). Queries with both $epoch = epoch'$ and $ctr = ctr'$ are ignored.*

Figure 2: Our security model for revocation schemes.

proof of knowledge of a valid signature on a message. We require Σ to be existentially unforgeable against known message attacks [24]. We use a commitment scheme where $\mathsf{ComOpenVf}(c, m, o)$ denotes checking that commitment c opens to message m with opening o. Let $\mathcal{H}$ be a collision resistant hash function.

4.1 Algorithm Instantiations

4.1.1 $\mathtt{Setup}(1^{\mathcal{K}}, n)$

Choose system parameters (j, k) such that $n = k^j$. Choose a group $\mathbb{G} = \langle g \rangle$ of prime order $q : |q| = \mathcal{K}$. Choose random elements $(\alpha_1, \ldots, \alpha_j) \xleftarrow{\$} \mathbb{Z}_q^j$ and compute $g_i = g^{\alpha_i}$ for all $i = 1, \ldots, j$. Create a key pair for the signature scheme $(sk_{RA}, pk_{RA}) \leftarrow \Sigma.\mathtt{KeyGen}(1^{\mathcal{K}})$ and choose a set of so-called randomizers $S = \{e_1, \ldots, e_k\}$ with every e-value $e_i \xleftarrow{\$} \mathbb{Z}_q$. Every e_i is signed with $\sigma_i \leftarrow \Sigma.\mathtt{Sign}(\mathtt{sk_{RA}}, \mathtt{e_i})$. Output the RA key pair (sk_{RA}, pk_{RA}) and the public system parameters $spar = (g, q, \mathbb{G}, k, j, (g_1, \ldots, g_j), (\alpha_1, \ldots, \alpha_j), \Sigma, \{(e_1, \sigma_1), \ldots, (e_k, \sigma_k)\})$.

4.1.2 $\mathtt{IssuePseudonym}(spar, sk_{RA}, rh) \leftrightarrow$ $\mathtt{ReceivePseudonym}(spar, pk_{RA})$

The RA chooses $w \xleftarrow{\$} \mathbb{Z}_q$ and adds w to the list of revocation handles $rh' \leftarrow rh \cup \{w\}$. The RA receives output (rh') and the user receives output w.

4.1.3 $\mathtt{ProvePseudonym}(spar, w, epoch, ctr)$

The user can only create k^j unlinkable pseudonyms per epoch, so if $ctr \geq k^j$, the user aborts. If $ctr < k^j$, we can view ctr as a k-ary number $(ctr_0, \ldots, ctr_{j-1})$ with $ctr = \sum_{i=0}^{j-1}(ctr_i \cdot k^i)$. Every ctr_i will indicate which e-value to use as the i-th randomizer. The pseudonym is computed as $C = g^{1/(w+\sum_{i=0}^{j-1} \alpha_i e_{ctr_i} + \mathcal{H}(epoch))}$. Then, the user constructs a zero-knowledge proof π proving that the pseudonym is formed correctly. This proof will be usually combined with the `ProveAttribute` protocol of the PABC scheme so that it is proven that also the credential contains the revocation handle w as an attribute, thus the credential and pseudonym are bound. We model this with commitment c, that commits to the revocation handle. In practise, c could be instantiated by the credential of the PABC scheme. Proof π also proves that c is correctly formed. User outputs (C, π, c). The `ProvePseudonym` algorithm is depicted in Fig. 3.

4.1.4 $\mathtt{VerifyPseudonym}(spar, pk_{RA}, C, \pi, c, epoch, RL_{epoch})$

Output 1 iff $\mathcal{H}(C) \notin RL_{epoch}$ and the zero knowledge proof is valid. By employing short hashes, the size of the revocation list updates and the revocation list itself remains acceptable, see Sec. 5 for details. Each time a verifier obtains an attribute proof, he can check the public revocation list and see if the pseudonym presented was revoked or not. This operation (simple sorted list look-ups) is logarithmic in the number of revoked users, thus computationally efficient even for millions of revoked users. Concrete numbers and implementation details are presented in Sec. 5.

4.1.5 $\mathtt{Revoke}(spar, rh, rd, \{C_R, epoch_R\}, epoch, revoked)$

This algorithm is run before each epoch is started.

For every revocation handle w on the list rh, compute hashes of all pseudonyms this user can create in this epoch, by computing $\mathcal{H}(C_i)$ for $i = 0, \ldots, k^j - 1$ with $C_{ctr} = = g^{1/(w+\sum_{l=1}^{j} \alpha_l e_{ctr_l} + \mathcal{H}(epoch))}$ and $(ctr_1, \ldots, ctr_j) \in \mathbb{Z}_k^j$ is the k-ary expension of ctr. Create tuples of the hashed pseudonym and its revocation handle w, and add all those tuples to a sorted list rd_{epoch}. Add rd_{epoch} to a revocation database $rd' \leftarrow rd \cup \{rd_{epoch}\}$. This will allow the RA later to find the revocation handle of a user given a pseudonym, with minimal computational effort.

Although this might seem to be a costly procedure, we show in the section devoted to implementation aspects (Sec. 5) that both computational and storage requirements can easily be met using standard hardware.

For users that are being revoked, look up $\mathcal{H}(C_R)$ in rd_{epoch_R} to find their revocation handle w_R and add them to the list of revoked handles $revoked' \leftarrow revoked \cup \{w_R\}$.

For users with revocation handle $w \in revoked'$, add all the computed hashed pseudonyms for the current $epoch$ to a sorted list RL_{epoch}.

Output $(RL_{epoch}, rd', revoked')$.

4.2 n-times Unlinkable Proofs

The revocation scheme presented in this paper is based on a property we call n-times unlinkability. Given a small set of signed public randomizers, a prover is able to construct a large number of proofs for statements about discrete logarithms. Without the knowledge of witnesses and randomizers, these proofs are provably unlinkable. However, the entity knowing the witnesses is able to efficiently link all the proofs. Similarly as we've employed the n-times unlinkability to enhance credential schemes, our construction can be further used in many other schemes based on the proof of knowledge protocols, such as the identification schemes, e-voting, e-cash, attestation schemes, etc.

4.3 Security Proof

We formally prove the security of our scheme in Appendix B.

5. IMPLEMENTATION ASPECTS

To prove that our scheme is practical even for large-scale applications, we provide a detailed analysis of the scheme's complexity and discuss its performance on real devices in this section. We let Σ be instantiated by weak Boneh-Boyen signatures and use the proof of knowledge protocol from Sec. 2.4.

5.1 Computational Requirements

The computational requirements of all algorithms are summarized in Table 1. We considered billinear pairing (P), scalar multiplication on elliptic curve (E) and table look-ups (L), and omitted operations with minor computational costs (such as hashes, random number generation, addition and subtraction).

The `IssuePseudonym` $\leftrightarrow$ `ReceivePseudonym` algorithm has only small constant complexity as it requires only a random number generation and its sharing.

The `ProvePseudonym` algorithm is the crucial part of the scheme as it runs on constrained devices, such as smartcards. It involves no bilinear pairing and only a small number[4] of EC multiplications. The computational complexity does not depend on the number of all users in the system nor the number of revoked users.

[4]The j parameter is usually between 1 and 4, see Sec. 5.3.3.

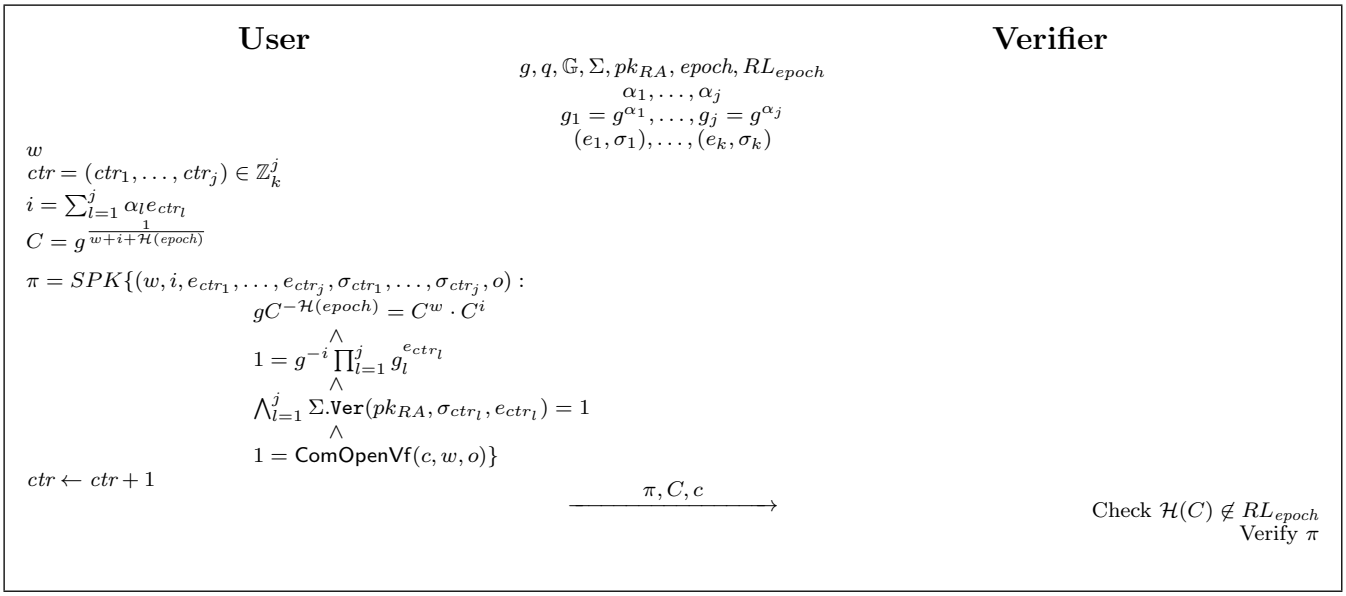

Figure 3: `ProvePseudonym` and `VerifyPseudonym` algorithms.

Table 1: Computational Requirements.

	U	RA	V
`IssuePseudonym`	0P, 0E, 0L	0P, 0E, 0L	-
`ProvePseudonym`	$(5j+3)$E	-	-
`VerifyPseudonym`	-	-	$2j$P $(4+3j)$E $\log(k^j\|U_R\|)$L
`Revoke`	-	$k^j\|U\|$E $\log(k^j\|U\|)$L	-

$|U_R|$: number of revoked users.
$|U|$: total number of users.
P: bilinear pairing operation e.
E: EC point scalar multiplication.
L: Look-ups: sorted table look-ups.

To verify a proof using the `VerifyPseudonym` algorithm, a verifier must compute $2j$ pairings and a small number of EC multiplications. To check user's revocation status, a verifier must do a number of revocation list look-ups that is logarithmic in the number of revoked users. The revocation check can be done also in constant time using associative arrays.

The `Revoke` algorithm is run by RA once in an epoch. To update the revocation database for a new epoch, a number of scalar point multiplications linear to the number of all system users must be computed. Although this number might get very high in large-scale applications, the computations can be easily finished in reasonable time using standard hardware and available cryptographic libraries. We present concrete numbers in Sec. 5.3.3. To revoke a user, a number of a revocation database look-ups that is logarithmic in the number of users must be computed.

Table 2: Storage Requirements.

	U	RA	V
`IssuePseudonym`	$\|spar\| + \|q\|$	$\|spar\| + \|q\|$	-
`ProvePseudonym`	$\|spar\|+$ $+j \log_2 k$	-	-
`VerifyPseudonym`	-	-	$\|spar\|+$ $+k^j\|U_R\|\|\mathcal{H}\|$
`Revoke`	-	$\|spar\|+$ $+k^j\|U\|\|\mathcal{H}\|T+$ $+\|q\|\|U_R\|$	-

$|U_R|$: number of revoked users.
$|U|$: total number of users.
$|\mathbb{G}_1|$: $\mathbb{G}_1$ element size.
$|q|$: $\mathbb{G}_1$ order size.
$|\mathcal{H}|$: size of the hash used.
T: maximum number of epochs (credential lifetime).
$|spar|$: constant size of pre-shared system parameters.

5.2 Storage Requirements

The storage requirements of all algorithms are summarized in Table 2.

The `IssuePseudonym` $\leftrightarrow$ `ReceivePseudonym` algorithm requires $|q|$ bits of storage at both user and RA.

The `ProvePseudonym` algorithm requires to store the counter and system parameters. The storage size is independent of the number of users and revoked users.

The `VerifyPseudonym` algorithm needs to use a revocation list. Its size is linearly dependent on the number of revoked users. The size of the list remains in units of megabytes even in large-scale applications, see Sec. 5.3.3 for concrete numbers.

The `Revoke` algorithm requires RA to store hashes of all pseudonyms in a revocation database rd. The size of rd is linear to the number of all users, but acceptable, see Sec. 5.3.3 for actual numbers. RA also needs to store the list of revoked handles of size linear to the number of revoked users.

5.3 Experiments

5.3.1 Parameters

The security parameter $1^{\mathcal{K}}$ specifies the order of the group. We set this parameter to $\mathcal{K} = 160$ and $\mathcal{K} = 224$ in our experiments, to match the ECRYPT security levels 4 and 6 [30].

The (j, k) parameters have a direct influence on how many times a user's pseudonym can be randomized during one epoch. In our analysis, we consider $(j = 2, k = 10)$. This choice allows users to generate $10^2 = 100$ pseudonyms per epoch.

Based on regulations and recommendations on revocation timing [40, 23], we chose an epoch to last one day.

To prove usability of our scheme in large-scale applications, we consider 8 million users and 10'000 revoked users. That models a national eID system of a mid-size country.

Choosing 60-bit hashes results in collision probability of around 10^{-9}. We note that a hash collision causes false rejection of a valid, non-revoked pseudonym, thus has effects only on usability, not security.

Finally, we consider that users' devices (smart-cards) are issued for 4-year periods in which they remain completely offline.

5.3.2 Hardware

We used an oldish mid-range server, namely the 2009 IBM x3550 M2 with two Intel Xeon 2.27 GHz processors with 8 cores each and 32 GB RAM, to represent verifiers' and RA's hardware. The smart-card selection was more difficult as only 3 out of our 19 programmable cards of all platforms (JavaCard, Multos, BasicCard) supported the plain EC scalar multiplication operation. We selected the NXP JavaCard J3D081 [36] as it was the fastest one.

5.3.3 Speed

The speed of bilinear pairing (P) and scalar point multiplication (E) virtually determine the performance of our algorithms as these operations have significantly higher (by orders) complexity than others (including table look-ups used only $\mathcal{O}(\log(|U|))$-times, resp. $\mathcal{O}(\log(|U_R|))$-times)[5] and because the scalar point multiplication is the most used operation in our scheme. We implemented the bilinear pairing operation using the PBC library [33] and used the 160-bit and 224-bit pairing-friendly Barreto-Naehrig (BN) curves [1]. The scalar point multiplication was implemented using a curve of the same size and type, using the PBC library on the server and ECDSA API on the smart-card. On the server, we used pre-processing and parallel processing on all cores.

We note that the pairing operation was not benchmarked on the card because currently it is not supported by any card available on the market, is difficult to implement without card manufacturer's full technical support and, most importantly, because our scheme does not require users to do any pairing operation.

The results of our benchmarks are presented in Table 3. We present the average of 10 runs for pairing and operations done on a smart-card. The time of multiplication on the server is the average of 16 mil. runs (1 million multiplications per CPU core).

[5] As we consider 8 million users and 10'000 revoked users, we have $\log(|U|) \approx 23$ and $\log(|U_R|) \approx 13$.

Table 3: Benchmarks of Primitive Operations.

	SC160	SC224	PC160	PC224
Pairing [ms]	-	-	1.94	3.25
Mult. [ms]	60	90	0.0102	0.0189

SC160: JavaCard with Fp160BN curve.
SC224: JavaCard with Fp224BN curve.
PC160: Server with Fp160BN curve.
PC224: Server with Fp224BN curve.

Table 4: Speed Estimates using Fp160BN curve.

	User	RA	V
`IssuePseudonym`	< 10 ms	< 10 ms	-
`ProvePseudonym`	780 ms	-	-
`VerifyPseudonym`	-	-	8 ms
`Revoke`	-	137 min	-

Using the results of benchmarks, we are able to estimate the running times of the algorithms proposed. These numbers are presented in Tables 4 and 5, respectively.

Based on our measurement, we expect the time of proving a revocation status to get under 1 second using standard smart-cards. The revocation database re-computation would take around 137 minutes. This operation is done only once in an epoch and we stress that this task is fully parallelizable. The revocation database update is done at the central RA, thus we expect much stronger server than the obsolete one used in our benchmark.

5.3.4 Storage

All storage requirements are negligible considering space available on current servers and smart-cards, except the revocation database stored at RA and the revocation list distributed to verifiers.

The `Revoke` algorithm needs to store hashes of all pseudonyms of all users. Using the formula from Table 2 and the parameters of our mid-size country eID scenario, we get the maximum of $8,76$ TB that needs to be stored at RA.

The `Revoke` algorithm also periodically produces the revocation list containing the hashes of all pseudonyms of revoked users, valid for the actual epoch. For our model scenario, the revocation list will not exceed 7.5 MB. This amount of data is shared by all verifiers and must be distributed by RA before each epoch starts.

6. CONCLUSION

In this paper, we addressed the problem of efficient revocation of privacy-enhancing attribute-based credentials. Until now, the revocation of attribute-based credentials has been difficult in scenarios that expect large number of users equipped with only offline, constrained devices, such as smart-cards. We proposed the first revocation scheme that runs in

Table 5: Speed Estimates using Fp224BN curve.

	User	RA	V
`IssuePseudonym`	< 10 ms	< 10 ms	-
`ProvePseudonym`	1170 ms	-	-
`VerifyPseudonym`	-	-	13 ms
`Revoke`	-	252 min	-

short constant time at user's side and in time logarithmic in the number of revoked users at verifier's side, using only offline devices for storing credentials. Our revocation scheme allows deploying attribute-based credentials in new applications in which the privacy protection has not been sufficiently addressed yet, such as the public transportation, e-ticketing and large-scale physical access control. As the next step, we'll focus on the optimization of the scheme based on evaluation of additional signature schemes and on the experimental implementation on tamper-proof user devices.

7. ACKNOWLEDGEMENT

Research was supported by the Czech Science Foundation project nr. 14-25298P "Research into cryptographic primitives for secure authentication and digital identity protection", the Technology Agency of the Czech Republic project TA04010476 "Secure Systems for Electronic Services User Verification" and the National Sustainability Program LO1401. This work has been supported by the ERC under Grant PERCY #321310. For the research, infrastructure of the SIX Center was used.

8. REFERENCES

[1] Paulo S. L. M. Barreto and Michael Naehrig. *Pairing-Friendly Elliptic Curves of Prime Order*, pages 319–331. Springer Berlin Heidelberg, 2006.

[2] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Proceedings of the 5th Conference on Theory of Cryptography*, TCC'08, pages 356–374. Springer-Verlag, 2008.

[3] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 600–610, New York, NY, USA, 2009. ACM.

[4] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2007.

[5] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 168–177. ACM, 2004.

[6] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 302–318, London, UK, 1994. Springer-Verlag.

[7] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[8] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 132–145, New York, NY, USA, 2004. ACM.

[9] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 234–252. Springer Berlin Heidelberg, 2008.

[10] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In Michael Franz and Panos Papadimitratos, editors, *TRUST 2016*, pages 1–20. Springer International Publishing, 2016.

[11] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography – PKC 2016*, pages 234–264, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[12] Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. *Policies and Research in Identity Management: Third IFIP WG 11.6 Working Conference, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings*, chapter Concepts and Languages for Privacy-Preserving Attribute-Based Authentication, pages 34–52. Springer Berlin Heidelberg, 2013.

[13] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Unlinkable priced oblivious transfer with rechargeable wallets. In Radu Sion, editor, *Financial Cryptography and Data Security 2010*, pages 66–81, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[14] Jan Camenisch and et Al. Specification of the identity mixer cryptographic library. Technical report, IBM Research - Zurich, 2010.

[15] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. Solving revocation with efficient update of anonymous credentials. In *Proceedings of the 7th international conference on Security and cryptography for networks*, SCN'10, pages 454–471. Springer-Verlag, 2010.

[16] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015*, pages 3–24, Cham, 2016. Springer International Publishing.

[17] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 61–76, London, UK, UK, 2002. Springer-Verlag.

[18] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd international conference on Security in communication networks*, SCN'02, pages 268–289. Springer-Verlag, 2003.

[19] Jan Camenisch and Anna Lysyanskaya. *Advances in Cryptology – CRYPTO 2004*, chapter Signature Schemes and Anonymous Credentials from Bilinear Maps, pages 56–72. Springer Berlin Heidelberg, 2004.

[20] Jan Camenisch and Markus Stadler. *Advances in Cryptology — CRYPTO '97*, chapter Efficient group signature schemes for large groups, pages 410–424. Springer Berlin Heidelberg, 1997.

[21] Sherman S. M. Chow. Real traceable signatures. In Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*

2009, pages 92–107, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[22] Yevgeniy Dodis and Aleksandr Yampolskiy. *Public Key Cryptography - PKC 2005*, chapter A Verifiable Random Function with Short Proofs and Keys, pages 416–431. Springer Berlin Heidelberg, 2005.

[23] CAB Forum. Baseline requirements for the issuance and management of publicly-trusted certificates version 1.1.6.

[24] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[25] Jan Hajny, Petr Dzurenda, and Lukas Malina. Privacy-pac: Privacy-enhanced physical access control. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, ACM CCS 2014, pages 93–96, New York, NY, USA, 2014. ACM.

[26] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous ip-address blocking. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, pages 113–133, 2007.

[27] Jorn Lapon. *Anonymous Credential Systems: From Theory Towards Practice (Anonieme credential systemen: van de theorie naar de praktijk)*. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science, July 2012. De Decker, Bart (supervisor), Naessens, Vincent (cosupervisor).

[28] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Performance analysis of accumulator-based revocation mechanisms. In Kai Rannenberg, Vijay Varadharajan, and Christian Weber, editors, *Security and Privacy – Silver Linings in the Cloud*, volume 330 of *IFIP Advances in Information and Communication Technology*, pages 289–301. Springer Berlin Heidelberg, 2010.

[29] Jorn Lapon, Markulf Kohlweiss, Bart Decker, and Vincent Naessens. *Communications and Multimedia Security: 12th IFIP TC 6 / TC 11 International Conference, CMS 2011, Ghent, Belgium, October 19-21,2011. Proceedings*, chapter Analysis of Revocation Strategies for Anonymous Idemix Credentials, pages 3–17. Springer Berlin Heidelberg, 2011.

[30] Katholieke Universiteit Leuven. Ecrypt ii yearly report on algorithms and keysizes. "http://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf".

[31] Zi Lin and Nicholas Hopper. Jack: scalable accumulator-based nymble system. In *Proceedings of the 2010 ACM Workshop on Privacy in the Electronic Society, WPES 2010, Chicago, Illinois, USA, October 4, 2010*, pages 53–62, 2010.

[32] Wouter Lueks, Gergely Alpár, Jaap-Henk Hoepman, and Pim Vullers. Fast revocation of attribute-based credentials for both users and verifiers. In *ICT Systems Security and Privacy Protection - 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings*, pages 463–478, 2015.

[33] Ben Lynn. The pairing-based cryptography library. https://crypto.stanford.edu/pbc/.

[34] Wojciech Mostowski and Pim Vullers. *Security and Privacy in Communication Networks: 7th International ICST Conference, SecureComm 2011, London, UK, September 7-9, 2011, Revised Selected Papers*, chapter Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards, pages 243–260. Springer Berlin Heidelberg, 2012.

[35] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer Berlin / Heidelberg, 2005.

[36] NXP. P5cd016/021/041 and p5cx081 family. http://cache.nxp.com/documents/data_sheet/P5CD016_021_041_Cx081_FAM_SDS.pdf.

[37] Tatsuaki Okamoto and Shigenori Uchiyama. *Advances in Cryptology — EUROCRYPT'98*, chapter A new public-key cryptosystem as secure as factoring, pages 308–318. Springer Berlin Heidelberg, 1998.

[38] Pascal Paillier. *Advances in Cryptology — EUROCRYPT '99*, chapter Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, pages 223–238. Springer Berlin Heidelberg, 1999.

[39] Christian Paquin. U-prove cryptographic specification v1.1. Technical report, Microsoft Corporation, 2011.

[40] European Parliament. Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec.

[41] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without ttps. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 72–81, New York, NY, USA, 2007. ACM.

[42] Patrick P. Tsang, Apu Kapadia, Cory Cornelius, and Sean W. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Trans. Dependable Sec. Comput.*, 8(2):256–269, 2011.

[43] Eric R. Verheul. Practical backward unlinkable revocation in fido, german e-id, idemix and u-prove. *IACR Cryptology ePrint Archive*, 2016:217, 2016.

[44] Pim Vullers and Gergely Alpár. *Policies and Research in Identity Management: Third IFIP WG 11.6 Working Conference, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings*, chapter Efficient Selective Disclosure on Smart Cards Using Idemix, pages 53–67. Springer Berlin Heidelberg, 2013.

APPENDIX

A. PROOF OF LEMMA 2.1

First, we now prove Lemma 2.1, i.e., we show that our proof of knowledge of weak Boneh-Boyen signatures is a valid proof of knowledge.

PROOF. **Completeness** If $\sigma \neq 1_{\mathbb{G}_1}$, we have $\sigma' \neq 1_{\mathbb{G}_1}$. The verification of π will pass by completeness of the underlying zero-knowledge proof. The final check $e(\bar{\sigma}, g_2) = e(\sigma', w)$ also passes:

$$\begin{aligned} e(\bar{\sigma}, g_2) &= e(\sigma'^{-m} g_1^r, g_2) \\ &= e(g_1^{r-\frac{rm}{x+m}}, g_2) \\ &= e(g_1^{\frac{rx}{x+m}}, g_2) \\ &= e(g_1^{\frac{r}{x+m}}, g_2^x) \\ &= e(\sigma', w) \end{aligned}$$

Soundness By soundness of proof π, we can extract (m, r) such that $\bar{\sigma} = \sigma'^{-m} g_1^r$. By $e(\bar{\sigma}, g_2) = e(\sigma', w)$, we have $\bar{\sigma} = \sigma'^x$, giving $\sigma'^x \cdot \sigma'^m = g_1^r$, so $\sigma' = g_1^{\frac{r}{x+m}}$. As we have $\sigma' \neq 1_{\mathbb{G}_1}$, we know that $r \neq 0$, so we can compute $\sigma \leftarrow \sigma'^{\frac{1}{r}}$ to extract a weak BB signature on m.

Zero-Knowledge To simulate a proof of knowledge of a signature on m, use the pair $(\bar{g}, \bar{g}^x) \in \mathbb{G}_1^2$. Take $r \xleftarrow{\$} \mathbb{Z}_q^*$ and set $\sigma' \leftarrow \bar{g}^r, \bar{\sigma} \leftarrow (\bar{g}^x)^r$, and simulate π. Note that $(\sigma', \bar{\sigma})$ are distributed as in a real proof: σ' is uniform in $\mathbb{G}_1^*$ and $\bar{\sigma} = \sigma'^x$. Use the simulator of π to complete the simulated proof.

□

B. SECURITY PROOF

We now prove that our revocation scheme is revocation correct, revocation sound, and revocation private.

THEOREM B.1. *Our revocation scheme is revocation correct, as defined in Def. 3.1, in the random oracle model.*

PROOF. We must prove that an honestly generated proof of non-revocation will pass verification, even when other potentially corrupt users are revoked, which we prove using a sequence of games.

Game 1: The challenger runs $(spar, (sk_{RA}, pk_{RA})) \leftarrow \mathsf{Setup}(1^\kappa)$, then runs $rh \leftarrow \mathsf{IssuePseudonym}(spar, sk_{RA}, rd) \leftrightarrow \mathsf{ReceivePseudonym}(spar, pk_{RA}) \rightarrow w$, and sets $(C, \pi, c) \leftarrow \mathsf{ProvePseudonym}(spar, w, epoch, ctr)$. It gives $spar$ and w to the adversary, simulates the random oracle honestly, and answers the queries honestly. This is the real experiment.

Game 2: The challenger now aborts when it detects a hash collision. In the random oracle model, this probability is negligible, so GAME 2 ≈ GAME 1.

Game 3: The challenger now aborts when $\mathcal{H}(C)$ is equal to one of the elements in $\mathcal{H}(C') \in RL$. By GAME 2, we now that this only occurs when $C = C'$ for some C'.

Upon an $\mathcal{O}^{\mathsf{Revoke}}$ query, one $w_i \in rh$ is revoked and all corresponding pseudonyms are added to RL: For every $(ctr_0, \ldots, ctr_{j-1}) \in \mathbb{Z}_k^j$, $C' = g_1^{\frac{1}{w_i + \mathcal{H}(epoch) + \sum_{i=0}^{j} \alpha_i e_{ctr_i}}}$. Since $w_i \in rh$ and every $w_i \in rh$ is taken uniformly at random from $\mathbb{Z}_q$, the probability that for some C' we have $C' = C$ is negligible.

Game 4: Verification of (C, π, c) consists of verifying π and checking $\mathcal{H}(C) \notin RL$. Proof π is valid by construction, and by GAME 3, we cannot have $\mathcal{H}(C) \in RL$, so completeness holds. □

THEOREM B.2. *Our revocation scheme is revocation sound, as defined in Def. 3.2, assuming Σ is existentially unforgeable against a known message attack, in the random oracle model.*

PROOF. Suppose an adversary $\mathcal{A}$ can win the revocation soundness game, then we can break the existential unforgeability of Σ with a known-message attack.

The challenger receives public key pk_{RA} from the unforgeability game. From the unforgeability game, the challenger receives a list $\{(e_1, \sigma_1, \ldots, (e_k, \sigma_k))\}$, which it uses as the signatures in *spar*. When the adversary outputs (C, π, c) with $\mathsf{VerifyPseudonym}(spar, pk_{RA}, C, \pi, c, epoch, RL) = \mathsf{accept}$, we know that π is valid and $\mathcal{H}(C) \notin RL$. From π we can extract $(w, e_1, \ldots, e_j, \sigma_1, \ldots, \sigma_j)$ by rewinding such that $C = g_1^{\frac{1}{w + \mathcal{H}(epoch) + \sum \alpha_i e_{ctr_i}}}$, $\Sigma.\mathtt{Ver}(pk_{RA}, \sigma_i, e_i) = 1$ for $i = 1, \ldots, j$, and $w \in revoked$. By $w \in revoked$ and $\mathcal{H}(C) \notin RL$, at least one of the e-values used must not be taken from *spar*, because when w was revoked, all possible pseudonyms with the e-values from *spar* were computed and added to RL. This e-value and corresponding signature σ lets the challenger win the existential unforgeability game. □

THEOREM B.3. *Our revocation scheme is revocation private, as defined in Def. 3.3, under the l-DBDHI assumption in the random oracle model.*

PROOF. To show that the adversary has a negligible advantage guessing b, we use a sequence of games in which the final game is independant of b.

Game 1: First, the challenger honestly performs the protocol and answers oracle queries correctly.

Game 2: Now, the challenger simulates the zero-knowledge proofs π part of the challenge (C, π, c).

Game 3: Next, pseudonym C from the challenge is taken as a random element in $\mathbb{G}_1$ in $\mathsf{ProvePseudonym}$. By the pseudorandomness of the Dodis-Yampolskiy VRF [22] (which holds under the l-DBDHI assumption), no adversary can notice this change. Note that as the credential has a limited lifetime, l will be bounded by a constant.

Game 4: Finally, we let c commit to a random value $w' \xleftarrow{\$} \mathbb{Z}_q$. By the hiding property of the commitment scheme, this change is not noticeable. Now, the game is independant of the choice of b, showing that no adversary can have a nonnegligible advantage of guessing b correctly. □