Bridging the Gap in Privacy-Preserving Revocation: Practical and Scalable Revocation of Mobile eIDs

Michael Hölzl
Institute of Networks and Security, JKU Linz
Altenbergerstraße 69, 4040 Linz, Austria
hoelzl@ins.jku.at

Michael Roland
University of Applied Sciences Upper Austria
Softwarepark 11, 4232 Hagenberg, Austria
michael.roland@fh-hagenberg.at

Omid Mir
Institute of Networks and Security, JKU Linz
Altenbergerstraße 69, 4040 Linz, Austria
mir@ins.jku.at

René Mayrhofer
Institute of Networks and Security, JKU Linz
Altenbergerstraße 69, 4040 Linz, Austria
mayrhofer@ins.jku.at

ABSTRACT

Providing methods to anonymously validate the user's identity is essential in many applications of electronic identity (eID) systems. A feasible approach to realize such a privacy-preserving eID is the usage of group signature protocols or pseudonym-based signatures. However, providing a revocation mechanism that preserves privacy is often the bottleneck for the scalability of such schemes. In order to bridge this gap between practicability and privacy, we propose a scalable and efficient revocation scheme suitable for smart cards in a mobile eID architecture. By using a pseudo-random function, we derive one-time revocation tokens for the revocation check and generate proofs of validity using a new method referred to as disposable dynamic accumulators. Our scheme thereby preserves unlinkability and anonymity of the eID holder even beyond revocation and does not require online connectivity to a trusted party for the verification and revocation check.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; *Privacy-preserving protocols*; • **Applied computing** → E-government;

KEYWORDS

Electronic identities, privacy-preserving revocation, scalability, dynamic accumulators, smart cards

ACM Reference format:
Michael Hölzl, Michael Roland, Omid Mir, and René Mayrhofer. 2018. Bridging the Gap in Privacy-Preserving Revocation: Practical and Scalable Revocation of Mobile eIDs. In *Proceedings of SAC 2018: Symposium on Applied Computing, Pau, France, April 9–13, 2018 (SAC 2018),* 9 pages.
https://doi.org/10.1145/3167132.3167303

SAC 2018, April 9–13, 2018, Pau, France

ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/10.1145/3167132.3167303

1 INTRODUCTION

State-of-the-art technology that provides a privacy-preserving eID can broadly be categorized into two main approaches: *pseudonym-based signatures* and *group signatures*. *Pseudonym-based signatures* [8, 9, 28] use public-key cryptography (e.g. RSA, ECC) and provide each prover with a list of pseudonyms. These pseudonyms usually consist of a private key, a public key, and a signed certificate from the issuer. For every signature creation within an identity validation process, the prover may use different pseudonyms. *Group signatures*, first introduced by Chaum and van Heyst [14], provide methods that allow each member within a group to sign messages on behalf of the whole group [5, 6, 13, 25].

A major issue in such signature schemes is the missing capability of revoking individual members without undermining their privacy. Although expensive for a large population, the easiest approach to achieve revocation is to re-enroll the whole group of valid members whenever an identity is revoked. A less complex mechanism is to let provers attest that no entry on a revocation list connects to their own identity [7, 26]. While this can be done in a zero-knowledge fashion and therefore provides good privacy, it also becomes very slow in large groups. To improve the performance, this check can also be done by the verifier (referred to as verifier-local revocation [5]) but has the downside that additional, potentially privacy degrading information, is sent to the verifier. A property called backward unlinkability [21, 25] ensures that users' privacy is not compromised when revoked. That is, even if an identity appears on the revocation list, adversaries cannot link any previous verification to this specific identity. While a few existing schemes already preserve privacy beyond revocation, they lack efficiency, scalability, or offline capability, which are required for a mobile eID. This is particularly problematic for governmental eIDs with a large population.

To address this issue in practicability of privacy-preserving revocation, we propose a new scalable and efficient scheme that preserves privacy of the user beyond revocation. We make use of a simple, but effective, generation of revocation tokens which provides anonymity in the population as well as unlinkability and backward unlinkability. To prove the validity of these tokens, we introduce a new method which we refer to as *disposable dynamic accumulators*, a variant of the dynamic accumulator [12]. Applying a protocol that splits the computation of this accumulator between two entities allows its execution on computationally restricted

prover devices, such as smart cards. By using Bloom filters [4] we keep the revocation list small and the revocation check efficient (can be performed in constant time). Finally, we evaluate our scheme for populations with multiple hundred thousands of revoked eIDs and show that it stays efficient for mobile devices and smart cards.

To summarize, our main contribution is a new practical revocation scheme, that is:

- *efficient for the prover*: Can be efficiently executed on devices with limited resources, such as smart cards.
- *efficient for the verifier*: Revocation checks are fast and performed in constant time on mobile devices.
- *scalable*: Even with large populations, the execution time of the revocation check stays constant and the required data remains small.
- *privacy-preserving*: Revoked as well as non-revoked eIDs can not be linked or deanonymized.
- *offline capable*: Revocation checks can be done with offline provers as well as offline verifiers.

2 PRIVACY-PRESERVING eID

We define a privacy-preserving eID based on the requirements of group signature protocols in [13]: *anonymity*, *unforgeability*, and *unlinkability*. Additionally, we consider *backward unlinkability* [21, 25], *revocability*, and *scalability*:

- *Anonymity*: The identity of users shall not be determinable within the whole population (k-anonymity with k being the population size).
- *Unforgeability*: Only members of the group can create valid eID signatures.
- *Unlinkability*: Verification processes and revocation information of the same user shall not be linkable.
- *Backward unlinkability*: It shall not be possible to link verification processes of a revoked eID.
- *Revocability*: It shall be possible for issuer, verifier (i.e. service provider), and eID holder to revoke eIDs in a privacy-preserving manner.
- *Scalability*: Verification and revocation checks shall be efficient for large populations.

Adversary Model. We consider two general types of adversaries: (i) malicious provers trying to forge a valid proof of an invalid identity (*forgery* or *identity theft*) and (ii) malicious verifiers trying to compromise the privacy of an eID holder. Malicious verifiers are further divided into:

$\mathcal{A}_1$ Single malicious verifier: Verifiers trying to deanonymize the verification process of a prover (e.g. trace activities).

$\mathcal{A}_2$ Colluding verifier: Two or multiple verifiers that cooperate and exchange verification data.

$\mathcal{A}_3$ Global adversary: An adversary that can passively eavesdrop all eID verification processes.

$\mathcal{A}_4$ Global adversary colluding with all verifiers: All verifiers cooperate with the global adversary.

Our proposed architecture and revocation scheme is able to protect against these adversaries $\mathcal{A}_1$ - $\mathcal{A}_4$. Similar to related schemes (e.g. [9, 10, 19, 22]), we thereby assume the malicious verifiers not to be in collusion with the issuer.

3 RELATED WORK

Lapon et al. [20] provide a good survey of privacy-preserving revocation strategies. The simplest, but impractical solution for large groups, is to generate new keys and let valid members re-enroll when an eID revocation occurs.

A less complex mechanism is to only send revocation information, such as a list of revocation tokens (revocation list), to verifiers and let them perform the revocation check. Boneh et al. [5] referred to this as verifier-local revocation (VLR) and it was further enhanced by Nakanishi and Funabiki [25, 26] and others [21, 29] with a property referred to as backward unlinkability. However, due to the need to perform complex operations on every item on the list, the scheme does not scale well for large populations.

Dynamic accumulators [3, 12] build the basis for a more efficient revocation mechanism [11, 28]. Identifiers of all group members are accumulated into one single value, which itself does not grow in size. Each prover has a so-called witness that enables them to prove that their identifier is in the accumulated value, and therefore, have a valid anonymous eID. Similarly, the scheme by Baldimitsi et al. [1] describe a variant that only has to be updated when an entry is deleted from the accumulator. The downside of these accumulator schemes is the requirement of witness updates when an eID has been added or revoked. Hence, it requires continuous connection to receive these updates.

The scheme by Lueks et al. [22] constrains the usage of revocation tokens to a specific time epoch and verifier. Revocation checks can be done with an epoch specific revocation list received from a semi-trusted party. Their concept therefore requires a trustworthy time source and constrained devices, such as smart cards, are usually not equipped with their own clock. Another downside is the requirement of short epochs to remain unlinkable and the high communication cost.

In the probabilistic revocation scheme by Kumar et al. [19], each signer gets a list of *alias codes* from a semi-trusted authority and includes one in each verification process. Instead of a list, the manager distributes a revocation code, a sample-by-sample addition of all revoked alias codes. During verification, the verifier then uses cross-correlation in order to probabilistically check if the alias code has been revoked.

Camenisch et al. in [10] propose a revocation scheme especially targeting eID systems using attribute-based credentials. The scheme can be executed on smart cards and enables the prover to generate up to n-different pseudonyms within a certain time epoch. The pseudonyms are derived from so-called revocation handles, received and maintained by a semi-trusted revocation authority. While this approach has good privacy-properties, it comes with the drawback of high communication cost due to often changing revocation lists.

4 CONCEPT AND ARCHITECTURE

The general architecture of our proposed system is shown in Figure 1 and consists of an eID issuer, a revocation manager, a prover, and a verifier. Prover and verifier both use mobile devices for the verification of the prover's identity. The data exchange between these two devices is done using near field communication (NFC) or a similar wireless connection. A special focus thereby lies on

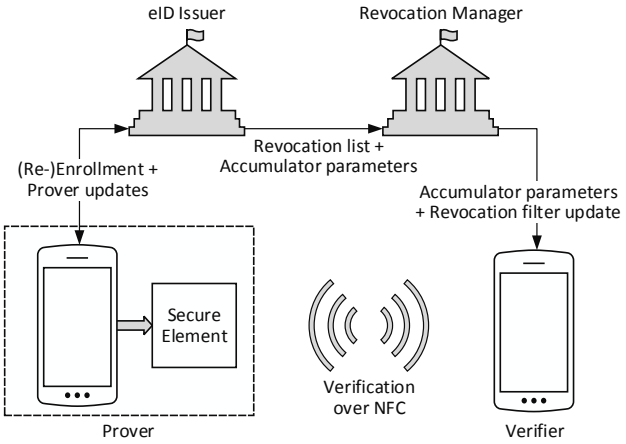

Figure 1: General architecture of our mobile eID system.

enabling mobility of the users. This results in the additional requirement that verification should be possible when verifier and prover devices are offline (e.g. no network connectivity or roaming).

In order to protect sensitive data of the eID, the prover uses a secure element (SE) as credential storage. An SE is a smart card variant which is usually shipped as an embedded integrated circuit in mobile devices together with NFC [23] and provides certain features: data protection against unauthorized access and tampering, code execution of small applications (applets) directly on the card (using the embedded microprocessor) and hardware supported execution of cryptographic operations (e.g. RSA, AES, SHA, etc.) for encryption, decryption, and hashing of data. However, the constrained execution performance and memory on the SE have strong implications on the protocol and architecture design of the eID (see performance evaluation in [18]).

An application running on the mobile device of the prover acts as a proxy between the verifier and the SE as well as between eID issuer and SE. We will refer to this as the eID management application (*eID-MA*). This application runs on the application processor of the mobile device in a potentially insecure environment and, therefore, cannot be trusted to store eID credentials.

In our scenario, the credentials on the SE consist of a secret revocation handle w_{se} (known by issuer and SE) and a counter value. These credentials must never leave the SE. In other words, all computations that require these credentials need to be performed in the environment of the SE. Our proposed architecture acknowledges this requirement and we present a scheme that can be executed within this secure but constrained environment in reasonable time.

eID Operations. We consider three main operations in the life cycle of an eID: enrollment, verification and revocation. During enrollment, the SE of the user u (i.e. the prover) generates the SE revocation handle w_{se} as well as a random start value for the counter value c_{se} and sends them to the issuer. This communication is done in a secure channel tunneled by the eID-MA on the prover's mobile device (e.g. using the GlobalPlatform secure channel protocol) and involves further identity validation in an out-of-band channel.

Verification is done using an NFC link between the verifier mobile device, the SE and the prover management application as proxy. The verification also involves a revocation check by the verifier. Hence, each verifier receives information from the revocation manager prior to that operation.

Revocation can be initiated by the eID issuer, service providers, or an eID holder, and is primarily performed by the issuer. The revocation manager receives a revocation list of all revoked eIDs and is then responsible for the distribution to all verifiers. The SE is not aware of its revoked status and still generates tokens and eID proofs when requested.

5 PRIVACY-PRESERVING AND PRACTICAL eID REVOCATION

5.1 Building Blocks and Preliminaries

We use QR_N to denote the set of quadratic residues modulo N. Furthermore, we assume a k-bit hash function $\mathrm{H}\colon \{0,1\}^* \to \{0,1\}^k$ and a method to establish a password-authenticated secure channel between the management application (eID-MA) on the prover's mobile device and the eID application on the SE (eID-SE). We refer to this method as *EstPA-SC* and ensure with this function that verifications are performed by the legitimate holder of the eID (e.g. usage of EC-SRP with PIN/password entry as in [17]). Further notations of our scheme are listed in Table 1.

5.1.1 Probabilistic Data Structures. Our proposed revocation mechanism makes use of *Bloom filters* [4] as one implementation of a probabilistic data structure. Such a probabilistic data structure differs from a deterministic data structures in the way that data is stored. In particular the characteristic of a Bloom filter has the advantage that significantly less memory is required and searching as well as storing in a growing data set has constant cost.

A Bloom filter consists of a bit array with m bits initialized to 0 and the usage of j different hash functions. A newly added element is hashed with these j hash functions, where each result defines

Table 1: Notation used in this paper.

w_{se}	Secret revocation handle. Known by SE and issuer.
c_{se}	Current counter value of the SE.
c_{max}	Maximum revocation tokens a prover can generate within an offline phase.
g	Generator for the accumulators.
G	Elliptic curve generator for revocation tokens.
rt_i	Pseudo random secret revocation token. Can be computed by an SE and issuer.
Rt_i	SE specific public one-time revocation token.
da_{se}	Disposable dynamic accumulator of all current revocation tokens for secure element *se*.
$\mathcal{L}, \mathcal{F}$	Revocation list $\mathcal{L}$ and revocation filter $\mathcal{F}$.
bloom ($\mathcal{L}$)	Bloom filter creation over all entries of list $\mathcal{L}$
BChk ($\mathcal{F}, i$)	Check if Bloom filter $\mathcal{F}$ contains i; yields 0|1.
Sign (sk, m)	Signature creation (e.g. ECDSA) over message m with private key sk; yields signature σ.
Ver (pk, m, σ)	Verification of signature σ with public key pk over message m; yields 0|1.

one array position i ($= \mathrm{H}^j(x) \mod m, 1 \leq i \leq j$) that is set to 1. To test if an element is a member in the data set, we hash the element with the same j hash functions and check if all resulting array positions are set to 1. If any of the positions is set to 0, the element is definitely not in the filter (i.e. false negative probability $p = 0$). If all bits are set to 1, the element is *probably* in the filter.

The advantages of using Bloom filters for the revocation list are: they require less space and the the computational effort for searching and storing is always constant. The downside of probabilistic data structures is the chance of false positives when querying an item in the data set. This is a result of potential collisions within other entries in the set. Fortunately, false positives can be well-controlled. The false positive probability p of a Bloom filter is computed with [24]

$$p = (1 - e^{-j \cdot n/m})^j \,, \tag{1}$$

where j is the number of hash functions used, n is the number of entries in the filter and m the length of the bit-array.

We can also compute the required bit array size m of a Bloom filter with a given false positive probability p and a given maximum number of stored elements n as

$$m = -\frac{n \cdot \ln p}{(\ln 2)^2} \,. \tag{2}$$

5.1.2 Dynamic Accumulators. Cryptographic accumulator were first introduced by Benaloh and de Mare [3] and are schemes where a set of k elements is combined into a single short value. This accumulated value and an additional fixed size witness are used to verify if an element is a member of that set.

An extension to that scheme is the RSA-based dynamic accumulator by Camenisch et al. [12], which allows to dynamically add and delete elements in the accumulator and to efficiently update the witnesses (i.e. adding or deleting is independent of the number of elements). To accumulate a set $\mathcal{X} = \{x_1, x_2, \cdots, x_k\}$, the elements of this set have to be relatively prime to $\phi(N) = (p-1)(q-1)$, where $\phi(\cdot)$ denotes the Euler totient function and $N = p \cdot q$ the RSA modulus. The accumulator for this set $\mathcal{X}$ is computed with

$$\mathrm{acc}(\mathcal{X}) = g^{x_1 \cdot x_2 \cdot x_3 \cdots x_k} \bmod N \,, \tag{3}$$

where $g \in_R QR_N$ is the generator and initial value. Efficiently deleting an element x_i from the accumulator $a = \mathrm{acc}(\mathcal{X})$ to get the updated value a', can be performed with the knowledge of the factorization of $N = p \cdot q$ with

$$a' = a^{({x_i}^{-1} \bmod \phi(N))} \bmod N \,. \tag{4}$$

A downside of this approach is that each holder of a valid witness has to perform an update operation when the accumulator changes (i.e. continuous connectivity required). In this paper, we introduce an adaption to this scheme that is not affected by this limitation.

5.2 Revocation Scheme

In general, our proposed scalable revocation scheme is based on using one-time revocation tokens which can be generated by the SE as well as the issuer. When the prover is asked to provide identity verification, the SE generates an elliptic curve public key as an one-time revocation token and a proof of validity. This validity proof is done using a newly introduced variant of the dynamic accumulator, the disposable dynamic accumulator (DDA). The DDA is created by the eID issuer for each SE and is used by the verifier to check the validity of the received token (i.e. only the eID issuer can create valid DDA entries). The invalidity check (i.e. revocation check) is performed using Bloom filters previously retrieved from the revocation manager.

Without the knowledge of revocation handle w_{se} and the counter value c_{se}, the one-time revocation tokens are provably unlinkable. Only an entity that knows them can link these revocation tokens. Similar to recent related work [9, 10, 19, 22], we assume the existence of a semi-trusted entity for the management of these tokens.

In the upcoming section, we describe the four methods for the management of these revocation tokens (*TokenGen*, *DDAGen*, *Update* and *Bind*), where in general, we build upon these assumptions:

(1) The verifier cannot be trusted from the perspective of issuer/revocation manager and prover.
(2) The eID management application of the prover cannot be trusted to keep credentials safe. Hence, no operation requiring secret keys shall be performed with it.
(3) The SE and issuer are trustworthy and can keep credentials safe. If an SE is still compromised and the credentials leak, their usage should be detectable.

5.2.1 Revocation Token Generation (TokenGen). Within an offline phase, the prover's SE is able to generate and prove up to c_{max} different *one-time revocation tokens* Rt_i using the previously stored revocation handle w_{se} as well as the counter value c_{se}. This is done by computing a one-time secret token rt_i and then multiplying it with the elliptic curve generator point G:

$$rt_i = \mathrm{H}(w_{se} \,||\, g \,||\, c_{se}) \tag{5}$$

$$Rt_i = rt_i \cdot G \tag{6}$$

The counter is initialized to a random value during the enrollment and is incremented as soon as one token Rt_i was generated.

It is important to note that each single revocation token should only be used once and therefore provides anonymity in the population and unlinkability of verification processes.

5.2.2 Disposable Dynamic Accumulators (DDAGen). While this generation of revocation tokens is computationally very simple, it introduces a significant weakness: A verifier cannot verify the valid generation of the token. Hence, the verifier would need to trust the entity which generates the tokens. Therefore, we need a method that allows to prove the validity of these tokens while at the same time protecting the privacy of the prover.

A simple solution would be to let the issuer create signatures over each single token and store them along with the tokens on the smart card. The verifier could then check the validity of the token using this signature and the issuer's public key. However, this is neither space nor communication efficient and therefore not reasonable for smart cards with strict memory constraints.

We address this issue by proposing the *disposable dynamic accumulators*, a variant of the dynamic accumulators which can only be created by a public issuer and modified by a dedicated owner.

Construction. Let $N = p \cdot q$ be an RSA modulus, where p, q are strong primes, and $g \in_R QR_N$ be a public generator of the accumulator. The disposable dynamic accumulator function computes an

accumulated value of all modular inverses of elements of the set $\mathcal{X} = \{x_1, x_2, \cdots, x_k\}$ with

$$\text{dacc}(\mathcal{X}) = g^{(x_1 \cdot x_2 \cdots x_k)^{-1} \bmod \phi(N)} \bmod N. \tag{7}$$

For efficiency, the modular inverse has to be computed only once over the product of all elements. It is important to note that the elements of the set $\mathcal{X}$ need to be pairwise distinct prime values in order to be able to ensure collision-resistance as proven by Baric and Pfitzmann in [2]. Benaloh and de Mare in [3] also suggest that values to be accumulated should either be hashed or encrypted before adding them to the accumulator. Hence, we apply the function $\mathbf{r}(x)$ on every element in the set $\mathcal{X}$, which hashes the input and computes a prime representative (e.g. using a method described in [16, 27]) that is also co-prime to $\phi(N)$. We assume that the used hash function is collision-resistant and the probability of generating the same prime twice is negligible. Also note that the probability of a prime not being co-prime to $\phi(N)$ is negligible [15].

Consequently, the definition of the disposable dynamic accumulator function has to be extended to

$$\text{dacc}(\mathcal{X}) = g^{(\mathbf{r}(x_1) \cdots \mathbf{r}(x_k))^{-1} \bmod \phi(N)} \bmod N. \tag{8}$$

Witness construction. The main difference of this scheme compared to a standard dynamic accumulator is the possibility to dispose an element x_i from the accumulated value $da_{\mathcal{X}} = \text{dacc}(\mathcal{X})$ by computing ${da_{\mathcal{X}}}^{x_i}$. This property proves to be very useful for the protection of the privacy of the user while still giving the verifier the certainty that an element has been accumulated by the issuer of the accumulator. For that purpose, the prover can compute a witness for each element $x_i \in \mathcal{X}$ in $da_{\mathcal{X}}$ with the function

$$\text{wit}(da_{\mathcal{X}}, \mathcal{X}, x_i) = {da_{\mathcal{X}}}^{\prod_{x \in \mathcal{X} \setminus \{x_i\}} \mathbf{r}(x)} \bmod N, \tag{9}$$

where the result is equal to a disposable dynamic accumulator with only one element x_i

$$da_{x_i} = \text{wit}(da_{\mathcal{X}}, \mathcal{X}, x_i) = g^{\mathbf{r}(x_i)^{-1} \bmod \phi(N)} \bmod N. \tag{10}$$

Verify element. The prover sends this witness da_{x_i} together with the value x_i to the verifier, who can verify the validity of the element x_i by checking

$$g \stackrel{?}{=} {da_{x_i}}^{\mathbf{r}(x_i)} \bmod N. \tag{11}$$

Note that the verifier only requires the public global generator g and modulus N and not the accumulated value $da_{\mathcal{X}}$ itself in order to validate the element. The verifier can assert that an element has been accumulated into the disposable accumulator of the prover by the trusted eID issuer. However, the verifier cannot associate the element with one specific disposable accumulator.

5.2.3 Prover Online Phase (Update). During an online phase, the SE communicates with the issuer in a secure channel, tunneled by the eID-MA on the prover's mobile device. The SE sends the number of used tokens Δc since the last update. The previously stored counter value associated with the prover w_{se} is then increased by this Δc. The eID issuer also computes all possible future revocation tokens of that SE and accumulates them into one disposable dynamic accumulator da_{se}, where the size k of this accumulator is a chosen number of offline revocation tokens c_{max}. Figure 2 depicts all these actions that are performed when a prover gets online and connects to the eID issuer.

eID Issuer		**eID-SE**
Look up previous c_{se}	$\xleftarrow{(\Delta c)}$	$\Delta c = c_{se} - c_{lb}$
$c_{se} = c_{se} + \Delta c$		
$\mathcal{X} = \{c_{se}, \ldots, c_{se} + c_{max} - 1\}$		
$\mathcal{T} = \{rt_i : rt_i = \text{H}(w_{se} \,\|\| g \,\|\| x_i) \; \forall\, x_i \in \mathcal{X}\}$		
$\mathcal{R}_{se} = \{Rt_i : Rt_i = rt_i \cdot G, \; \forall\, rt_i \in \mathcal{T}\}$		
$da_{se} = \text{dacc}(\mathcal{R}_{se})$		
$\text{store}(c_{se})$	$\xrightarrow{(c_{max}, da_{se})}$	$c_{lb} = c_{se}$
		$c_{ub} = c_{se} + c_{max}$
		$\text{store}(c_{lb}, c_{ub}, da_{se})$

Figure 2: Prover online phase.

eID-SE		**eID-MA**
	$\xleftrightarrow{\text{EstPA-SC}}$	User inputs PIN/password
$\mathcal{X} = \{c_{lb}, \ldots, c_{ub}\}$	$\xleftarrow{\quad\quad}$	
$\mathcal{T} = \{rt_i : rt_i = \text{H}(w_{se} \,\|\| g \,\|\| x_i), \forall\, x_i \in \mathcal{X}\}$		
$\mathcal{R}_{se} = \{Rt_i : Rt_i = rt_i \cdot G, \; \forall\, rt_i \in \mathcal{T}\}$	$\xrightarrow{(\mathcal{R}_{se}, da_{se})}$	$\text{store}(\mathcal{R}_{se}, da_{se})$

Figure 3: Binding protocol of the management application (eID-MA) and the secure element (eID-SE). Dashed lines indicates communication within the password-authenticated secure channel.

In the return message, the SE receives this disposable dynamic accumulator da_{se} as well as the new maximum revocation token count c_{max}. Then it computes the new lower ($c_{lb} = c_{se}$) as well as upper bound counter value ($c_{ub} = c_{se} + c_{max}$) and stores them. In order to prevent the usage of previous accumulators in case the SE has been compromised, the SE replaces the old value with the new one. During eID verification, the SE is able to increase its local counter value c_{se} up to the upper bound c_{ub} and prove its validity using da_{se}. When this value is reached, the SE will not be able to create proofs for further revocation tokens and the management application on the mobile device will inform the user that online connectivity is required (verification is not possible in the meantime). Note that the maximum number of revocation tokens c_{max} could be adapted to the usage behavior of each individual user.

5.2.4 Binding Prover Devices (Bind). We assume that the prover consists of a mobile device application and an SE. As previously stated, the SE is affected by computational constraints. To address these limitations, we define an additional protocol step that allows to split the computation between these two devices and reduce the verification time (i.e. time where the user is involved).

The steps of this protocol are shown in Figure 3 and need to be performed after a prover online phase:

(1) The user enters a PIN/password in the eID-MA (start of the application).

(2) The eID-MA and the SE establish a secure channel using the PIN/password of the user (EstPA-SC).
(3) If the channel was successfully established (i.e. legitimate holder of the eID operates the application), the SE computes the list $\mathcal{R}_{se}$ of all future public revocation tokens Rt_i and returns it together with the current da_{se} to the eID-MA.
(4) The management application stores the list $\mathcal{R}_{se}$ and da_{se} in the local application storage.

By entering the PIN/password on the device, the user establishes a trust relationship between the management application and the SE. The consecutive operations can be performed in the background after the credentials have been entered. The information that is thereby exposed by the SE can potentially be misused by an adversary (e.g. phone stolen, malicious software) to link single verifications of the user. However, the security of the scheme is not compromised (secret credentials never leave the SE).

The binding can be done on any mobile device trusted by the user. Therefore, it is possible to use the same eID on multiple different devices (e.g. SIM card which is transferred to another device).

5.3 Verification Protocol

The verification happens after the prover has been enrolled to the system and performed the *prover online phase* of Section 5.2.1 at least once. Also the binding protocol of the prover should have been performed once.

5.3.1 Validity and Invalidity Checks. The sequence of actions for the eID verification is shown in Figure 4 and consist of a proof generation and a validity/invalidity check. A verifier v initiates the process by sending a random challenge ch to the prover.

Token and proof generation. When requested, the management application on the mobile device (MD) forwards the challenge to the SE to generate the current secret revocation token rt_i and public revocation token Rt_i. In the return message, the mobile application receives the public token and reads the list of all tokens $\mathcal{R}_{se}$ and the current disposable dynamic accumulator da_{se} from its memory. The application can now create the witness da_{Rt_i} for that token using the witness function of the disposable dynamic accumulator and sends it to the SE. Finally, the SE verifies the validity of this witness and creates a signature over the tuple (ch, da_{Rt_i}). This signature is returned to the mobile application and forwarded to the verifier together with the public token Rt_i and the witness da_{Rt_i}. In the different identity validation use cases, the return message and the signature would also include relevant data attributes. A detailed discussion on these attribute queries are out of scope of this paper.

Validity check. The check that is performed by the verifier takes the revocation token Rt_i, the challenge, the disposable dynamic accumulator da_{Rt_i}, and the signature σ of the prover as inputs. The verifier first validates the signature over the sent challenge and the received disposable dynamic accumulator da_{Rt_i} using the revocation token Rt_i as public key. If this check succeeds, the verifier tests the validity of the received public revocation token with

$$g \stackrel{?}{=} da_{Rt_i}{}^{\mathbf{r}(Rt_i)} \bmod N, \tag{12}$$

where the tuple (g, N) are global parameters retrieved from the revocation manager. If any of the checks fail, the verifier aborts.

Invalidity check. If the validity check was successful, the verifier tests if the revocation token Rt_i has been revoked. For that purpose, the verifier downloads the *revocation filter* $\mathcal{F}$, a Bloom filter containing the tokens of all revoked eIDs. This filter is managed and generated by the revocation manager (see revocation management in Section 5.4)

5.3.2 False Positive Mitigation. Due to the usage of Bloom filters, the revocation scheme is affected by the possibility of false positives during revocation checks. To mitigate the likelihood, the scheme provides the flexibility for a verifier to request up to f_{max} one-time revocation tokens. Requests for more than f_{max} revocation tokens within the same verification attempt will be denied by the SE to protect against denial-of-service and brute-force attacks. When the maximum number of revocation tokens has been reached, the SE will only generate new tokens if the user triggers a new verification within the mobile device application. The equation for computing the false positive probability is, therefore, extended to:

$$p' = p^{f_{max}} = (1 - e^{-jn/m})^{j \cdot f_{max}}. \tag{13}$$

In case the eID has been revoked, all retrieved tokens will indicate a positive result in the revocation check. That is, there are no false negatives with Bloom filters. This has the implication that a revoked eID will more quickly run out of available revocation tokens and will therefore invalidate sooner (i.e. the counter reaches the maximum and the SE cannot generate new proofs).

In the unlikely scenario of false positives with all generated revocation tokens (i.e. prover is sure that the eID has not been revoked), the verifier is forced to perform an online check with the revocation manager. However, an evaluation of this possibility is beyond the scope of this paper.

5.4 Revocation Management

The issuer is responsible for managing a global revocation list $\mathcal{L}$. Only the revocation manager can query that list and generate the revocation filter $\mathcal{F}$ or a differential filter update $\mathcal{F}'$ for verifiers.

Revocation. This process can be initiated by the user or the issuer. When an SE of a user is revoked, the issuer adds all possible revocation tokens Rt_i of that SE to the global revocation list $\mathcal{L}$. The tokens are computed as:

$$\mathcal{X} = \{c_{se} \ldots c_{se} + c_{max} - 1\} \tag{14}$$

$$\mathcal{T} = \{rt_i : rt_i = \mathrm{H}(w_{se} \,||\, g \,||\, x_i), \forall\, x_i \in \mathcal{X}\} \tag{15}$$

$$\mathcal{R}_{se} = \{Rt_i : Rt_i = rt_i \cdot G, \forall\, rt_i \in \mathcal{T}\} \tag{16}$$

$$\mathcal{L} = \mathcal{L} \cup \mathcal{R}_{se} \tag{17}$$

where $\mathcal{X}$ is a list of all valid integer values for the revocation token generation. The value c_{se} is the last counter value that was sent by the SE to the issuer during an online phase. The resulting revocation list $\mathcal{L}$ is then sorted and sent to the revocation manager.

Filter Retrieval. When the verifier requests the current revocation filter $\mathcal{F}$, the revocation manager computes it using the current revocation list $\mathcal{L}$ from the issuer and sends the result to the verifier ($\mathcal{F} = \mathrm{bloom}(\mathcal{L})$). The revocation manager acts as proxy between issuer and verifier but does not have the capability to link revocation tokens of the same identity.

eID-SE		eID-MA		Verifier
	$\xleftrightarrow{\text{EstPA-SC}}$	User inputs PIN/password		
$rt_i = \mathrm{H}(id_{se} \,\|\|\, g \,\|\| c_{se})$	$\xleftarrow{(ch)}$		$\xleftarrow{(ch)}$	choose random ch
$Rt_i = rt_i \cdot G$				
$c_{se} = c_{se} + 1$	$\xrightarrow{(Rt_i)}$	Readout $\mathcal{R}_{se}, da_{se}$		
verify $g \stackrel{?}{=} da_{Rt_i}{}^{\mathbf{r}(Rt_i)}$	$\xleftarrow{(da_{Rt_i})}$	$da_{Rt_i} = \mathrm{wit}(da_{se}, \mathcal{R}_{se}, Rt_i)$		
$\sigma = \mathrm{Sign}(rt_i, ch \,\|\|\, da_{Rt_i})$	$\xrightarrow{(\sigma)}$		$\xrightarrow{(\sigma, Rt_i, da_{Rt_i})}$	$\mathrm{Ver}(Rt_i, ch \,\|\|\, da_{Rt_i}, \sigma)$
				$g \stackrel{?}{=} da_{Rt_i}{}^{\mathbf{r}(Rt_i)} \bmod N$
				$\mathrm{BChk}(\mathcal{F}, Rt_i)$

Figure 4: Identity verification scheme between verifier and the secure element (eID-SE) as well as the management application on the mobile device (eID-MA) of the prover. Dashed lines indicate communication done within the password-authenticated secure channel. The tuple (g, N) are globally known parameters and only change during an accumulator re-enrollment phase.

Differential Filter Updates. A major benefit of our scheme is the possibility to retrieve differential revocation filter updates. This stands in contrast to approaches where the complete list changes after a certain time epoch [10, 22]. Instead, the revocation manager can remember the last access time of the verifier and generate a differential filter $\mathcal{F}'$ of revocation tokens added since then. Due to the structure of Bloom filters (i.e. many consecutive zeros), this differential filter can be compressed efficiently before sending.

5.5 Security Analysis

As defined in the adversary model of the proposed scheme in Section 2, we consider attackers that attempt to forge a valid proof of an invalid identity (unforgability) and adversaries $\mathcal{A}_1 - \mathcal{A}_4$ that attempt to compromise the privacy of the eID holder. In this subsection, we discuss the security of the proposed revocation scheme against forgery attacks. Note that the proof presented here is only a sketch. A full proof is beyond the scope of this paper.

To protect against forgeries, our proposed scheme relies on the security of the disposable dynamic accumulator as well as the SE. Attacks on the security of the used signature scheme (e.g. ECDSA) or the server backend are out of the scope of this paper. For any PPT adversary $\mathcal{A}$, we say that proposed scheme is unforgeable if advantage $\mathbf{Adv}_A$ is negligible:

$$\mathbf{Adv}_A := \Pr[(g, N) \leftarrow \mathrm{Gen}(1^\kappa), (rt', da') \leftarrow \mathcal{A}(g, N, G) : g = da'^{\mathbf{r}(Rt')}, Rt' = rt' \cdot G]$$

Furthermore, we build upon the security assumption:

Definition 5.1 (Strong RSA assumption [2]). Let κ be the security parameter. Given a κ-bit RSA modulus N and the value $z \in_R \mathrm{QR}_N$, there is no probabilistic polynomial-time algorithm $\mathcal{P}$ that outputs y and a prime x such that $x > 1$ and $y^x = z \bmod N$, except with negligible probability.

Theorem 5.2. *In the random oracle model, suppose an adversary $\mathcal{A}$, who can ask at most q_H hash queries and break the unforgability of the proposed scheme in polynomial time with advantage ε, then there exists an adversary $\mathcal{B}$ that breaks the strong RSA assumption with advantage ε/q_H.*

Proof. Suppose there exists an adversary $\mathcal{A}$, who can break the unforgability of the proposed scheme, then we can construct another adversary $\mathcal{B}$ who uses $\mathcal{A}$ as a black-box to break the strong RSA assumption with non-negligible probability.
Setup: $\mathcal{B}$ is given a hash function modeled as a random oracle, the modulus N as the product of two large safe primes, and a value $z \in_R \mathrm{QR}_N$. Furthermore, the adversary $\mathcal{B}$ defines an elliptic curve generator G and invokes $\mathcal{A}$ with (z, N, G).
Queries: $\mathcal{A}$ can query $\mathcal{B}$ about the following:

- *r-Hash query*: For each query Rt_i, $\mathcal{B}$ responds with a randomly chosen prime number $h_i \in \mathrm{QR}_N$ with consistency.
- *Accumulator query*: $\mathcal{A}$ requests an accumulator witness da_i of the revocation token Rt_i. In response to this query, $\mathcal{B}$ first picks a random value $j \in \{1, \cdots, q_H\}$ as a guess which query will correspond to the eventual forgery. We assume that $\mathcal{A}$ always queries the ***r*-Hash oracle** for token i before this query and the corresponding h_i is already stored in an internal table. If $i \neq j$, $\mathcal{B}$ computes $da_{Rt_i} = z^{\mathbf{r}(Rt_i)^{-1}} \bmod N$ and returns da_{Rt_i}. As we consider the result of $\mathbf{r}(\cdot)$ to be prime, it is easy to see that each distinct query has a unique solution. If $i = j$, $\mathcal{B}$ declares failure and aborts.

Output: Finally, $\mathcal{A}$ outputs a forgery (rt', da') and wins if the conditions $z = da'^{\mathbf{r}(Rt')} \bmod N$ and $Rt' = Rt_j$, where $Rt' = rt' \cdot G$ hold.

Adversary $\mathcal{B}$ can now use this adversary $\mathcal{A}$ to efficiently break the strong RSA assumption. That is, the output $(h' := \mathbf{r}(Rt'), da')$ can be transformed into a solution $(x := h', y = da')$ of the instance (z, N) of the strong RSA problem.

$\mathcal{B}$ wins the game if $\mathcal{A}$ successfully forges a witness da_j and queried the r-hash oracle for Rt_j ($Rt' \in Rt_i$), but never requests the accumulator oracle for Rt_j. The value j is independent of the views of $\mathcal{A}$ and hence $\mathcal{B}$ obtains a forgery with at least ε/q_H. □

Corollary 5.3. *The proposed scheme is secure against identity theft by a malicious prover.*

Besides forging new identities, an adversary can also attempt to take over existing identities. The security of an identity of an eID holder is based on the difficulty to find d for given elliptic

curve points D and G of order N such that $D = d \cdot G \bmod N$. This is referred to as the elliptic curve discrete logarithm problem (ECDLP). In order to steal the identity of an eID holder, an adversary $\mathcal{A}$ needs to eavesdrop a message $(\sigma, Rt_i, da_{Rt_i})$ and break the ECDLP problem in order to get rt_i and create a valid signature over a random challenge ch.

Corollary 5.4. *The proposed revocation scheme is secure against forgery attacks even when the integrity of the SE is compromised.*

The security of our scheme relies on the usage of an SE as a tamper-resistant storage for the credentials. This is a state-of-the-art technology for protecting sensitive information (e.g. SIM/bank cards) and protects the integrity of the eID in cases where the mobile device gets stolen or malicious software is able to exploit the operating system and read the memory of the prover device. If the integrity of an eID is still broken (e.g. attack on the integrity of the SE), an adversary acquires the credentials w_{se} and c_{se} as well as the current disposable dynamic accumulator da_{se}. Based on Theorem 5.2, the adversary cannot use random tokens which are not accumulated in da_{se}. As soon as a compromised SE is detected (e.g. stolen/lost phone), the issuer will add all its tokens to the revocation list and reject update requests by this SE. Note that misuse of tokens could also be determined by a collaboration between issuer and verifier and requires some computational effort by the issuer (i.e. compute the revocation tokens of all users).

6 SYSTEM EVALUATION

In the evaluation we assume a 256-bit hash function, 128 byte revocation handle w_{se} and 4 byte counters. To get prime representatives for $\mathbf{r}(\cdot)$, we use the function `nextProbablePrime()` of the Java BigInteger class. To keep this also efficient on the SE, we include a byte array in the update message of the issuer, indicating the difference between hash and prime representative of each revocation token. For simplicity, we will omit this data in the evaluation.

6.1 Performance Analysis

As we depend on the usage of the computationally restricted SE, we especially focus on the protocols involving this device:

Token Generation (TokenGen). For the token generation, the SE has to increment an integer, compute one hash and perform an elliptic curve multiplication. We executed these operations on a Yubikey NEO with JavaCard 3.0.1. For the generator g, we used a 256-byte value and the 32-byte hash of this value in the secret token generation. Including transfer time, the average token generation time over 100 measurements was 161.4 ms ± 1.3 ms (standard deviation). The transfer time of the interface used for sending and receiving data was 10.0 ms ± 0.1 ms.

Binding Prover Devices (Bind). Binding the eID-SE and eID-MA together is a step where all provable revocation tokens are generated on the SE and sent to the eID-MA. Hence, the execution time of this step equals the time for the generation of one token (as previously elaborated), multiplied with the maximum number of allowed offline revocation tokens c_{max}. For example, with $c_{max} = 100$, the execution time on the smart card is $100 \cdot 161.4\,\text{ms} \approx 16\,\text{s}$.

Table 2: Mean computation times of the protocols that depend on the number of offline revocation tokens c_{max}. That is, the verification protocol and the prover online phase (Update).

c_{max}	Verification protocols			Prover
	eID-SE	eID-MA	Verifier	Update
10	526 ± 4 ms	13 ± 3 ms	77 ± 19 ms	81 ± 26 ms
50	526 ± 4 ms	66 ± 6 ms	77 ± 19 ms	291 ± 18 ms
100	526 ± 4 ms	135 ± 12 ms	77 ± 19 ms	557 ± 28 ms
200	526 ± 4 ms	274 ± 12 ms	77 ± 19 ms	1081 ± 37 ms
300	526 ± 4 ms	424 ± 13 ms	77 ± 19 ms	1643 ± 100 ms

Prover Online Phase (Update) and DDAGen. The online phase of the prover involves the eID-SE of the prover and the issuer. The eID-SE only needs to subtract and add one value. More computational effort is required by the issuer, where all future revocation tokens of an eID are accumulated into one disposable dynamic accumulator.

We evaluated these steps performed by the issuer on a Thinkpad T440s with an Intel i7-4600U@2.1GHz dual-core CPU and depict the results in Table 2. The computation time depends on the choice of c_{max}, the maximum number of revocation tokens within an offline phase. Generating the update for $c_{max} = 10$ takes on average 81 ms and linearly increases with c_{max}.

Verification Protocol. The steps for the verification protocol are split over eID-SE, eID-MA of the prover and verifier mobile device (see Figure 4). The SE, for example, has to increment an integer, create a new revocation token, verify the disposable dynamic accumulator (one modular exponentiation) and create an elliptic curve signature. For the validity check (signature and disposable dynamic accumulator proof) we used 256-bit ECDSA and 2048-bit RSA. For the invalidity check (revocation check), we used an existing Bloom filter implementation and performed $f_{max} = 5$ queries. We ran 100 measurements on the same smart card as previously elaborated and used a OnePlus One with Android version 5.1.1 as eID-MA and verifier device. We assume that eID-MA has pre-computed the product of all prime representatives of $\mathcal{R}_{se}$ during the binding protocol. Hence, computing the witness for a token requires dividing this product by the token and one modular exponentiation.

Table 2 lists the results of the evaluation of the verification protocol steps for different maximum revocation tokens c_{max} (excl. transfer between the devices). While the computation times of the verifier and the eID-SE stays constant, the execution time on the eID-MA linearly increases with c_{max}. For a chosen $c_{max} = 100$, the total time of all three devices is around 740 ms.

6.2 Quantifying Scalability

The scalability of a revocation scheme usually depends on the complexity of the revocation check and the size of the revocation list. Our scheme has the benefit that the revocation check does not depend on the number of eID nor on the population size and can be performed in constant time.

Only the size of the revocation list plays an important role in the scalability of our scheme, which itself can adjusted to a required false positive probability. In our terms, a reasonable target probability p' for an eID architecture should be below 10^{-9}. That is, one

verification process within one billion identity validations in the entire system requires the user to authorize a second query. The chance that the second query fails is then 10^{-18}. Due to our false positive mitigation technique, there are f_{max} possible revocation tokens within one such query. The false positive probability p of a single revocation token, given the target probability p', is computed as $p = \sqrt[f_{max}]{p'}$ (see Equations 1 and 13). Consequently, with $f_{max} = 5$ we get a single false positive probability $p = 1.58 \cdot 10^{-2}$.

Furthermore, for the total filter size, we have to decide on a number of maximum revocation tokens c_{max} for each user. This also defines the number of filter entries for a revoked eID ($n = c_{max} \cdot z$). In the evaluation, we set this maximum token count to $c_{max} = 100$.

With Equation 2 we can now calculate the required filter size m with a target probability of $p = 1.58 \cdot 10^{-2}$ and z revoked eIDs. The results are a required size of 5.1 MB for 50,000 revocations or 10.3 MB for 100,000 revocations. Half a million revoked eIDs require 51 MB of uncompressed storage by the verifier, and so on. Note that the verifier can thereby download differential updates of this filter. Due to the characteristics of Bloom filters (many consecutive zeros), these updates can also be efficiently compressed.

7 CONCLUSION

In this paper, we proposed a novel revocation scheme for a privacy-preserving and scalable mobile eID revocation. Our scheme is based on the usage of a new variant of the dynamic accumulators, the disposable dynamic accumulator, and a pseudo-random function for the creation of revocation tokens. In the evaluation we have shown that this simple, but effective, function has low computational costs for prover as well as verifier. Furthermore, our scheme benefits from its good scalability and offline capabilities and has the potential to be deployed in large populations. It only requires irregular updates of the prover, small revocation list updates due to the use of probabilistic data structures and has constant verification time, independent of revocation list or population size.

8 ACKNOWLEDGMENTS

This work has been carried out within the scope of *u'smile*, the Josef Ressel Center for User-Friendly Secure Mobile Environments, funded by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

REFERENCES

[1] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. 2017. Accumulators with Applications to Anonymity-Preserving Revocation.. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. http://eprint.iacr.org/2017/043.pdf

[2] Niko Baric and Birgit Pfitzmann. 1997. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *Proc. EUROCRYPT '97*. LNCS, Vol. 1233. Springer Berlin Heidelberg, 480–494.

[3] Josh Benaloh and Michael de Mare. 1993. One-way accumulators: A decentralized alternative to digital signatures. In *Proc. EUROCRYPT '93*. Springer-Verlag New York, Inc., 274–285.

[4] Burton H. Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commununications of the ACM* 13, 7 (July 1970), 422–426. https://doi.org/10.1145/362686.362692

[5] Dan Boneh and Hovav Shacham. 2004. Group Signatures with Verifier-local Revocation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*. ACM, 168–177.

[6] Ernie Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*. ACM, 132–145.

[7] Ernie Brickell and Jiangtao Li. 2007. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES '07)*. ACM, 21–30.

[8] Julien Bringer, Hervé Chabanne, Roch Lescuyer, and Alain Patey. 2014. Efficient and Strongly Secure Dynamic Domain-Specific Pseudonymous Signatures for ID Documents. In *Financial Cryptography and Data Security*. LNCS, Vol. 8437. Springer, 255–272.

[9] Julien Bringer, Hervé Chabanne, Roch Lescuyer, and Alain Patey. 2016. *Hierarchical Identities from Group Signatures and Pseudonymous Signatures*. Springer Berlin Heidelberg, 457–469.

[10] Jan Camenisch, Manu Drijvers, and Jan Hajny. 2016. Scalable Revocation Scheme for Anonymous Credentials Based on N-times Unlinkable Proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society (WPES '16)*. ACM, New York, NY, USA, 123–133. https://doi.org/10.1145/2994620.2994625

[11] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. 2009. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography – PKC 2009*. LNCS, Vol. 5443. Springer, 481–500.

[12] Jan Camenisch and Anna Lysyanskaya. 2002. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology – CRYPTO 2002*. LNCS, Vol. 2442. Springer, 61–76.

[13] Jan Camenisch and Markus Stadler. 1997. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO '97*. LNCS, Vol. 1294. Springer, 410–424.

[14] David Chaum and Eugène Van Heyst. 1991. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*. LNCS, Vol. 547. Springer, 257–265.

[15] Hermann de Meer, Manuel Liedel, Henrich C. Pöhls, Joachim Posegga, and Kai Samelin. 2012. *Indistinguishability of one-way accumulators*. Technical Report. MIP-1210, Faculty of Computer Science and Mathematics, University of Passau.

[16] Michael T. Goodrich, Roberto Tamassia, and Jasminka Hasić. 2002. An Efficient Dynamic and Distributed Cryptographic Accumulator. In *Information Security*. LNCS, Vol. 2433. Springer Berlin Heidelberg, 372–388.

[17] Michael Hölzl, Endalkachew Asnake, René Mayrhofer, and Michael Roland. 2015. A Password-authenticated Secure Channel for App to Java Card Applet Communication. *International Journal of Pervasive Computing and Communications* 11 (Oct. 2015), 374–397. https://doi.org/10.1108/IJPCC-09-2015-0032

[18] Michael Hölzl, René Mayrhofer, and Michael Roland. 2013. Requirements for an Open Ecosystem for Embedded Tamper Resistant Hardware on Mobile Devices. In *Proc. MoMM 2013: International Conference on Advances in Mobile Computing & Multimedia*. ACM, 249–252. https://doi.org/10.1145/2536853.2536947

[19] Vireshwar Kumar, He Li, Jung-Min (Jerry) Park, Kaigui Bian, and Yaling Yang. 2015. Group Signatures with Probabilistic Revocation: A Computationally-Scalable Approach for Providing Privacy-Preserving Authentication. In *Proc. of the 22nd Conference on Computer and Communications Security*. ACM.

[20] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. 2011. Analysis of Revocation Strategies for Anonymous Idemix Credentials. In *Communications and Multimedia Security*. LNCS, Vol. 7025. Springer, 3–17.

[21] Benoît Libert and Damien Vergnaud. 2009. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *Cryptology and Network Security*. LNCS, Vol. 5888. Springer, 498–517.

[22] Wouter Lueks, Gergely Alpár, Jaap-Henk Hoepman, and Pim Vullers. 2015. Fast revocation of attribute-based credentials for both users and verifiers. In *ICT Systems Security and Privacy Protection*. LNCS, Vol. 455. Springer, 463–478.

[23] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. 2008. NFC Devices: Security and Privacy. In *Third International Conference on Availability, Reliability and Security (ARES'08)*. 642–647. https://doi.org/10.1109/ARES.2008.105

[24] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA.

[25] Toru Nakanishi and Nobuo Funabiki. 2006. A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability. In *Advances in Information and Computer Security*. LNCS, Vol. 4266. Springer, 17–32.

[26] Toru Nakanishi and Nobuo Funabiki. 2008. A Short Anonymously Revocable Group Signature Scheme from Decision Linear Assumption. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS '08)*. ACM, 337–340. https://doi.org/10.1145/1368310.1368359

[27] Tomas Sander, Amnon Ta-Shma, and Moti Yung. 2000. Blind, Auditable Membership Proofs. In *Financial Cryptography*. LNCS, Vol. 1962. Springer Berlin Heidelberg, 53–71.

[28] Jinyuan Sun, Chi Zhang, Yanchao Zhang, and Yuguang Fang. 2010. An Identity-Based Security System for User Privacy in Vehicular Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems* 21, 9 (Sept. 2010), 1227–1239.

[29] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. 2007. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proc. of the 14th ACM Conference on Computer and Communications Security*. ACM, 72–81.