International Journal of Information Security (2020) 19:401–417
https://doi.org/10.1007/s10207-019-00458-7

REGULAR CONTRIBUTION

Disposable dynamic accumulators: toward practical privacy-preserving mobile eIDs with scalable revocation

Michael Hölzl[1] · Michael Roland[1] · Omid Mir[1] · René Mayrhofer[1]

Published online: 31 July 2019

Abstract
Providing methods to anonymously validate user 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 a system. In order to bridge this gap between practicability and privacy, we propose a new pseudonym-based mobile eID signature scheme suitable for smart cards and secure elements that also enables efficient and scalable revocation checks. By using a pseudorandom function, we derive one-time verification tokens used for identity verification as well as revocation checks and generate proofs of validity using a new method referred to as *disposable dynamic accumulators*. Our scheme preserves unlinkability and anonymity of the eID holder even beyond revocation and does not require online connectivity to a trusted party for verification and revocation checks.

Keywords Electronic identities · Privacy-preserving revocation · Scalability · Dynamic accumulators

1 Introduction

State-of-the-art technology that provides a privacy-preserving eID can be broadly categorized into two main approaches: *pseudonym-based signatures* and *group signatures*. *Pseudonym-based signatures* [8,9,32] 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 [5,6,13,27].

A major issue in such signature schemes is the missing capability of revoking individual members without undermining their privacy. This is due to signatures that become linkable after revocation. As a result, each usage of a single eID could be linked together in order to create a full activity profile of its holder. This is particularly problematic when such a system is used in a variety of scenarios in form of a mobile eID (e.g., to identify with the police, age check by disco bouncer, etc.) That is, a core privacy guarantee of this eID would relinquish as soon as, for example, the mobile device of an eID holder is stolen or lost.

Although expensive for a large population, the easiest approach to achieve privacy-preserving 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 [7,28]. 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 [23,27] 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

This paper is the extended full version of [20].

✉ Michael Hölzl
hoelzl@ins.jku.at

[1] Institute of Networks and Security, JKU Linz, Altenbergerstraße 69, 4040 Linz, Austria

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 eID systems, we propose a new pseudonym-based signature scheme that enables scalable identity revocation and preserves the privacy of users beyond it. We make use of a simple, but effective, generation of pseudonyms (i.e., we refer to them as one-time verification 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, in this paper we introduce a new pseudonym-based signature scheme that is:

- *Efficient for the prover* Can be efficiently executed on devices with limited resources (e.g., smart cards).
- *Efficient for the verifier* Identity verification as well as revocation checks are fast and performed in constant time on mobile devices.
- *Scalable* Even with large populations, the execution time of the verification and revocation checks stay constant and the required data remains small.
- *Privacy-preserving* Revoked as well as non-revoked eIDs can not be linked or deanonymized.
- *Offline capable* Identity verification and 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* [23,27], *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 three types of adversaries:

1. Malicious provers trying to forge a valid proof of an invalid identity. Such adversaries have access to all public parameters of the system and try to mislead a genuine verifier into believing that they are a valid eID holder (i.e., *forgery* or *identity theft*).
2. Malicious provers that try to circumvent the revocation check although their eIDs have been revoked. Such attackers have access to all public parameters and can use their eIDs to attempt to make genuine verifiers believe that they still own a valid eID.
3. Malicious verifiers trying to compromise the privacy of an eID holder. That is, a verifier that attempts to attack any of the previously elaborated privacy properties of an eID holder: *anonymity*, *unlinkability* and *backward unlinkability*. We assume that such an attacker can either actively participate in a verification process, passively listen to a communication, or perform both. Hence, we divide this adversary into different strength levels:

   $\mathcal{A}_1$ Single malicious verifier: Verifiers trying to deanonymize the verification process of a prover.
   $\mathcal{A}_2$ 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 eID architecture and revocation scheme is able to provable secure against these cases (see the proof in Sects. 5.6.1 and 5.7).

3 Related work

In order to provide a privacy-preserving eID, numerous related publications propose the use of group signature protocols [5,13,14,27]. A famous example for such a protocol is Direct Anonymous Attestation (DAA) by Brickell et al. [6], deployed in so-called Trusted Platform Modules (TPM) for a privacy-preserving attestation of software, but lacks in privacy-preserving revocation.

For eID revocation, Lapon et al. [22] 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 [27,28], and others [23,33] 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. To improve this, Raya et al. [29] as well as Haas et al. [17] proposed the usage of Bloom filters for a scalable search and storage in the certificate revocation list in vehicular ad hoc networks (VANET).

Dynamic accumulators [3,12] build the basis for a more efficient revocation mechanism [11,32]. 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, that it has a valid anonymous eID. Similarly, the scheme by Baldimitsi et al. [1] describes 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, continuous connectivity is required to receive these updates.

The scheme by Lueks et al. [24] constrains the usage of revocation tokens to a specific time epoch and verifier. Revocation checks are performed with epoch specific revocation lists downloaded from a semi-trusted party. Consequently, they require a trustworthy time source, which is problematic for constrained devices (e.g., smart cards). Another downside is the requirement of short epochs to remain unlinkable and the resulting high communication cost.

In the probabilistic revocation scheme by Kumar et al. [21], 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. [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.

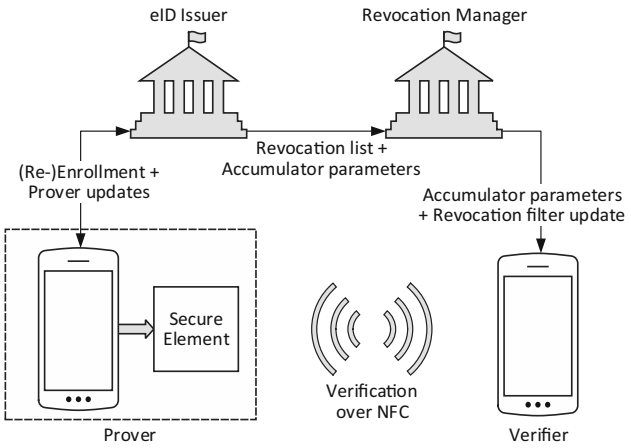

Fig. 1 General architecture of our mobile eID system

4 Concept and architecture

The general architecture of our proposed system is shown in Fig. 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 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 [30] 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 evaluation in [19]).

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 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 identity 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.

5 Privacy-preserving and practical eIDs

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 $\mathrm{H}\colon \{0,1\}^* \rightarrow \{0,1\}^k$ 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 [18]). 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 deterministic data structures in the way how 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 one array position i ($= \mathrm{H}^j(x) \bmod m, 1 \leq i \leq j$) 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 that they require less space and that 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 [25]

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

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

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

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

5.1.2 Dynamic accumulators

Cryptographic accumulators 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 $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$, 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

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

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

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

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 eID operations

We consider four operations in the life cycle of an eID:

– During *enrollment*, the SE of the user u (i.e., the prover) generates the SE identity 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

Table 1 Notation used in this paper

w_{se}	Secret identity handle; known by SE and issuer
c_{se}	Current counter value of the SE
c_{max}	Maximum verification tokens a prover can generate within an offline phase
g	Generator for the accumulators
G	Elliptic curve generator for verification tokens
rt_i	Pseudorandom secret verification token; can be computed by an SE and issuer
Rt_i	SE specific public one-time verification token
da_{se}	Disposable dynamic accumulator of all current verification 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

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. This validation as well as a discussion on the secure channel is out of scope.
- *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.
- *Re-enrollment* involves the redistribution of new public parameters (i.e., public keys, revocation lists) and happens on an irregular basis (e.g., once or twice a year). While this phase is not mandatory for the functionality of our scheme, the re-enrollment helps to reduce the size of revocation lists and strengthen the security.
- *Revocation* can be initiated by the eID issuer, service providers, or an eID holder, and is primarily performed by the issuer by marking the identity handle w_{se} of an eID holder as revoked (note that the issuer would probably request auxiliary verification information in order to validate a legitimate revocation request by an eID holder). Afterward, 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. However, it will not get updates by the eID issuer anymore or be able to participate in the re-enrollment phase.

5.3 Proposed scheme

In general, our proposed privacy-preserving mobile eID scheme with scalable revocation is based on using one-time verification 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 a one-time verification 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 the identity handle w_{se} and the counter value c_{se}, the one-time verification tokens are provably unlinkable (see Sect. 5.7). Only an entity that knows them can link these verification tokens.

In the upcoming section, we describe the four methods for the management of these verification tokens (*TokenGen*, *DDAGen*, *Update*, and *Bind*).

Security Assumptions Our proposed concept and scheme builds upon the following security assumptions:

1. The verifier cannot be trusted from the perspective of issuer/revocation manager and prover.
2. The eID-MA 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.

5.3.1 Identification token generation (TokenGen)

Within an offline phase, the prover's SE is able to generate and prove up to c_{max} different one-time verification tokens

Rt_i using the previously stored identity 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:

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

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

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 verification token should only be used once and therefore provides anonymity in the population and unlinkability of verification processes.

5.3.2 Disposable dynamic accumulators (DDAGen)

While this generation of verification 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 the 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 $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ with

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

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 [16,31]) 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

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

Witness construction The main difference between this scheme and a standard dynamic accumulator is the possibility to dispose an element x_i from the accumulated value $\text{da}_{\mathcal{X}} = \text{dacc}(\mathcal{X})$ by computing $\text{da}_{\mathcal{X}}^{x_i}$. 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 $\text{da}_{\mathcal{X}}$ with the function

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

The witness is equal to a disposable dynamic accumulator with only one element x_i:

$$\text{da}_{x_i} = \text{wit}(\text{da}_{\mathcal{X}}, \mathcal{X}, x_i) \tag{10}$$

$$= g^{\mathbf{r}(x_i)^{-1} \bmod \phi(N)} \bmod N. \tag{11}$$

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

$$g \stackrel{?}{=} \text{da}_{x_i}^{\mathbf{r}(x_i)} \bmod N. \tag{12}$$

Note that the verifier only requires the public global generator g and modulus N and not the fully accumulated value $\text{da}_{\mathcal{X}}$ 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 x_i and the witness da_{x_i} with their corresponding disposable accumulator $\text{da}_{\mathcal{X}}$.

5.3.3 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 verification 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 verification tokens $c_{\max}$.

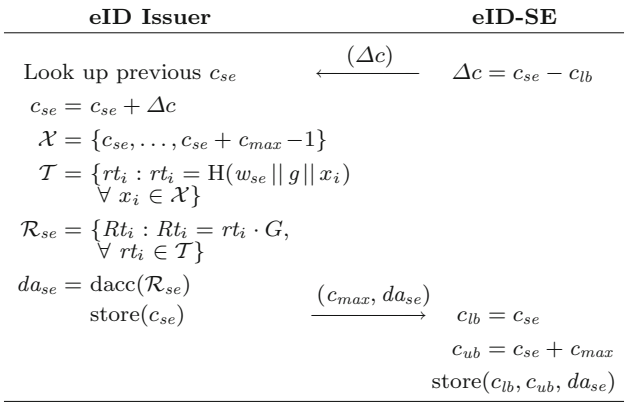

Fig. 2 Prover online phase

Figure 2 depicts all these actions that are performed when a prover gets online and connects to the eID issuer.

In the return message, the SE receives this disposable dynamic accumulator $\mathrm{da}_{\mathrm{se}}$ as well as the new maximum verification token count $c_{\max}$. Then it computes the new lower bound ($c_{\mathrm{lb}} = c_{\mathrm{se}}$) as well as upper bound ($c_{\mathrm{ub}} = c_{\mathrm{se}} + c_{\max}$) counter values 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 $\mathrm{da}_{\mathrm{se}}$. When this upper bound is reached, the SE is no longer able to create proofs for further unlinkable verification tokens and the management application on the mobile device will inform the user that online connectivity is required before verification is possible again. Note that the maximum number of verification tokens $c_{\max}$ could be varied between users and can be adapted to the usage behavior of each individual user.

5.3.4 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 and the construction of a witness in each eID verification process would therefore be too time-consuming. To address these limitations, we define an additional protocol step that allows to split computation between the two devices to reduce verification time (i.e., the time where the user is involved).

The steps of this protocol are shown in Fig. 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).

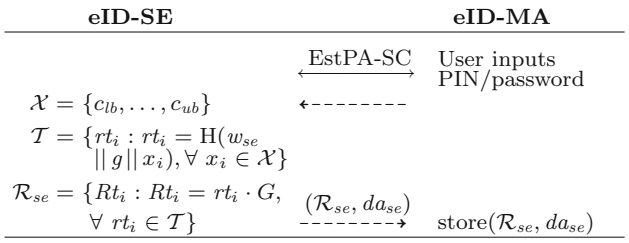

Fig. 3 Binding protocol of the management application (eID-MA) and the secure element (eID-SE). Dashed lines indicate communication within the password-authenticated secure channel

3. If the channel was successfully established (i.e., legitimate holder of the eID operates the application), the SE computes the list $\mathcal{R}_{\mathrm{se}}$ of all future public verification tokens Rt_i and returns it together with the current $\mathrm{da}_{\mathrm{se}}$ to the eID-MA.
4. The management application stores the list $\mathcal{R}_{\mathrm{se}}$ and $\mathrm{da}_{\mathrm{se}}$ 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. Subsequent 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 multiple 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-based SE which is transferred to another device).

5.4 Verification protocol

The verification happens after the prover has been enrolled to the system and has performed the *prover online phase* (Sect. 5.3.1) at least once. Also the binding protocol of the prover should have been performed once.

5.4.1 Validity and invalidity checks

The sequence of actions for the eID verification is shown in Fig. 4 and consists 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 (eID-MA) forward the challenge to the SE to generate the current secret verification token rt_i and public verification token Rt_i. In the return message, the mobile application receives the public

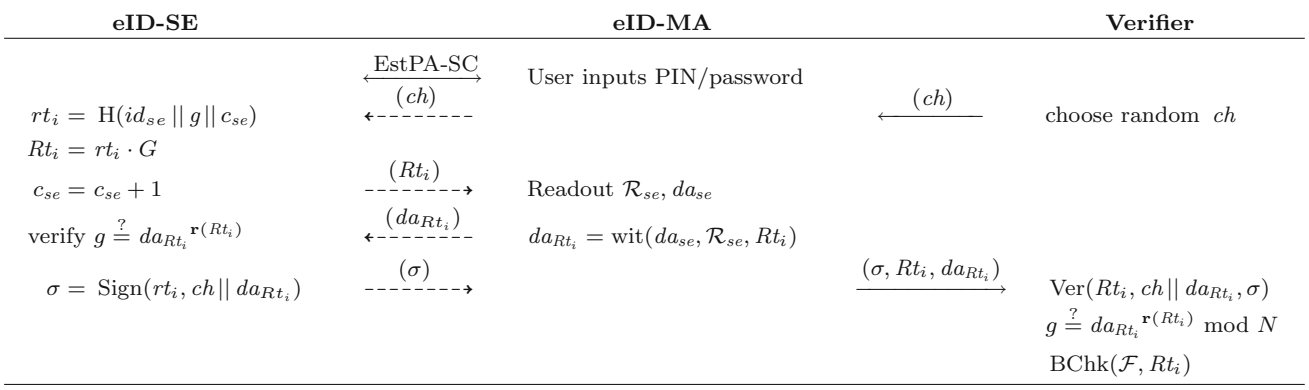

Fig. 4 Identity verification scheme between verifier and the secure element (eID-SE) as well as the management application on the mobile device (eID-MA) of the prover. Dashed lines indicate communication done within the password-authenticated secure channel. The tuple (g, N) are globally known parameters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

token and reads the list of all tokens $\mathcal{R}_{\text{se}}$ 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 (e.g., ECDSA) over the tuple (ch, da_{Rt_i}) using secret verification token 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 is beyond the scope of this paper.

Validity check The check that is performed by the verifier takes the verification 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 verification token Rt_i as public key. If this check succeeds, the verifier tests the validity of the received public verification token with

$$g \stackrel{?}{=} \text{da}_{Rt_i}{}^{\mathbf{r}(Rt_i)} \bmod N, \tag{13}$$

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 verification 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 Sect. 5.5)

5.4.2 False positive mitigation

Due to the usage of Bloom filters, our 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 verification tokens. Requests for more than $f_{\max}$ verification 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 verification tokens has been reached, the SE will only generate new tokens if the user triggers a new verification within the mobile device application. Therefore, the equation for computing the false positive probability is extended to:

$$p' = p^{f_{\max}} = \left(1 - e^{-\frac{j \cdot n}{m}}\right)^{j \cdot f_{\max}}. \tag{14}$$

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 verification 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 verification 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. Note that this is assumed to happen very rarely (see evaluation in Sect. 6.4).

5.5 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 either be initiated by the user or the eID issuer. When an SE of a user is revoked, the issuer adds all possible verification tokens Rt_i of that SE to the global revocation list $\mathcal{L}$. The tokens can be computed with:

$$\mathcal{X} = \{c_{\text{se}} \dots c_{\text{se}} + c_{\max} - 1\} \tag{15}$$

$$\mathcal{T} = \{rt_i : rt_i = \text{H}(w_{\text{se}}||g||x_i), \forall\, x_i \in \mathcal{X}\} \tag{16}$$

$$\mathcal{R}_{\text{se}} = \{Rt_i : Rt_i = rt_i \cdot G, \forall\, rt_i \in \mathcal{T}\} \tag{17}$$

$$\mathcal{L} = \mathcal{L} \cup \mathcal{R}_{\text{se}} \tag{18}$$

where $\mathcal{X}$ is a list of all valid integer values for the verification 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 ($\mathcal{F} = \text{Bloom}(\mathcal{L})$). The revocation manager acts as a proxy between issuer and verifier but does not have the capability to link verification tokens of the same identity.

Differential filter updates A major benefit of our proposed 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 [10,24]. Instead, the revocation manager can remember the last access time of the verifier and generate a differential filter $\mathcal{F}'$ of verification 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.

5.6 Security analysis

5.6.1 Formal proof

As defined in the adversary model of our scheme in Sect. 2, we consider attackers that attempt to forge a valid proof of an invalid identity (unforgeability), attackers that attempt to circumvent the revocation check, and adversaries $\mathcal{A}_1 - \mathcal{A}_4$ that attempt to compromise the privacy of the eID holder. In this subsection, we provide a proof that our proposed scheme protect against forgery attacks and attacks on the revocation check. 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 the proposed scheme is unforgeable if advantage $\mathbf{Adv}_A$ is negligible:

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

Furthermore, we build upon the security assumption:

Definition 1 (*Strong RSA assumption* [2]) Let κ be the security parameter. Given a κ-bit RSA modulus N and the value $z \in_R \text{QR}_N$, 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 1 *In the random oracle model, suppose an adversary $\mathcal{A}$, who can ask at most q_H hash queries and break the unforgeability 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 unforgeability 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 $z \in_R \text{QR}_N$. 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 $h_i \in \text{QR}_N$ with consistency.
- *Accumulator query* $\mathcal{A}$ requests an accumulator witness da_i of the verification token Rt_i. In response to this query, $\mathcal{B}$ picks a random value $j \in \{1, \dots, q_H\}$ as a guess which query will correspond to the eventual forgery. We assume that $\mathcal{A}$ always queries the $\mathbf{r}$-*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 $\text{da}_{Rt_i} = z^{\mathbf{r}(Rt_i)^{-1}} \bmod N$ 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 $z = \text{da}'^{\mathbf{r}(Rt')} \bmod N$ 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 $(h' := \mathbf{r}(Rt'), \text{da}')$ can be transformed into a solution $(x := h', y = \text{da}')$ 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 1 *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 $(\sigma, Rt_i, \mathrm{da}_{Rt_i})$ and break the ECDLP problem in order to get rt_i and create a valid signature over a random challenge ch. We also evaluate the possibility of breaking the ECDLP problem in "Appendix A".

Corollary 2 *The proposed scheme is secure against malicious provers attempting to circumvent the revocation check.*

An adversary with a previously valid (and now revoked) eID, might attempt to circumvent the revocation check. As this check is performed by the verifier using a revocation filter on its device, the attacker needs to send a valid public identity token Rt_i that is not on this revocation filter. For that purpose, the adversary could either try to forge a new token or try to reuse a previously created one. From Theorem 1, we know that the former is not possible without breaking the strong RSA assumption. The latter would only be possible if the adversary has access to the identity handle w_{se} or the private part of the previously created identity token rt_i. As we assume that the SE is trustworthy and can keep credentials safe, we can also assume that the adversary has no access to these credentials and can therefore not compute previous token proofs to circumvent the revocation check.

Corollary 3 *The proposed identity scheme can protect 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 $\mathrm{da}_{\mathrm{se}}$. Based on Theorem 1, the adversary cannot use random tokens which are not accumulated in $\mathrm{da}_{\mathrm{se}}$. However, attackers could also be able to use previous identity tokens (if they were able to observe the corresponding disposable accumulators or witnesses). In such a scenario, the issuer would still be able to revoke all tokens of this SE (since the last re-enrollment) and reject further update requests. As we assume that these scenarios only rarely happen, the effect of adding all identity tokens of an SE to the revocation list size would still be negligible. Note that misuse of tokens of a compromised SE could also be determined by a collaboration between issuer and verifier and requires some computational effort by the issuer (i.e., compute the verification tokens of all users).

5.6.2 Malicious eID issuer

Our adversary model assumes that the eID issuer is trustworthy and can keep credentials safe. In fact, a malicious issuer could generate an arbitrary number of valid verification tokens in our scheme. They could also create disposable dynamic accumulators as well as validity proofs for a specific identity handle w_{se} by generating the same one-time verification tokens rt_i, Rt_i. As a result, they would be able to act exactly as any SE within the eID system.

One possible approach to mitigate the risk of malicious issuers is to slightly adapt the protocol and store the identity handle w_{se} only on the SE. During a prover online phase, the SE would then send the product of hashed public verification tokens $\mathbf{r}(Rt_i)$ instead of the number of used tokens Δc. Rather than generating SE specific public verification tokens, the eID issuer computes the modular inverse of this received product and returns the disposable accumulator to the prover. Consequently, a malicious eID issuer will only see the public one-time verification token of an SE and cannot generate the corresponding secret parts anymore.

The downside of this mitigation is the additional computational effort that has to be done by the SE. Hence, this is another trade-off between security (as well as privacy) and performance of the scheme. A more detailed evaluation of this adapted version of our scheme is beyond the scope of this paper.

5.7 Privacy analysis

In this section, we consider the adversaries $\mathcal{A}_1 - \mathcal{A}_4$ from Sect. 2 that attempt to compromise users' privacy. The strongest adversary is $\mathcal{A}_4$ (i.e., global adversary colluding with all verifiers) and should thereby not be able to compromise any of the previously established privacy properties of any user (i.e., anonymity, linkability, backward unlinkability). The game is to link the verification messages of a single user (i.e., multiple Rt_i and da_{Rt_i} values) and, hence, being able to identify or trace the activities of an eID holder.

Anonymity and Unlinkability In the formal privacy analysis in "Appendix A", we show that our proposed scheme

protects against this strongest adversary under the random oracle assumption. We further prove that it is infeasible to determine the identity of the user without breaking the one-way property of the hash function and the ECDLP.

Revocation list and differential filter updates The list update that is sent to the revocation manager as well as the differential filter update for each verifier in our scheme comes with a drawback to the unlinkability property. If the subset of revoked eIDs in these updates is small, there is a chance for an adversary to link verification tokens of the same revoked prover. In the worst case, an update only consists of one revoked eID. An adversary could use this differential to test multiple verification tokens and, upon a positive result, they can be certain that these tokens come from the same prover. Hence, the issuer has to protect against these attacks by avoiding small updates. For example, only provide updates with a minimum number of revoked eIDs.

Note that this only affects linkability of future verifications after revocation occurred. That is, after a prover update, previous verification tokens of a revoked eID are not in the revocation list and therefore not linkable (i.e., backward unlinkability is not affected).

6 System evaluation

We consider the following aspect of an eID scheme: (1) administration effort, (2) performance in terms of computation and space requirements, and (3) scalability. We thereby assume a 256-bit hash function, 128-byte identity handle w_{se} and 4-byte counters. Additionally, we also outline the used techniques to implement the prototype on an SE.

6.1 Secure element implementation details

A major limiting factor to the performance of our scheme is the usage of SEs. The specific implementation of the software applet that performs the protocol steps therefore plays a significant role in the overall computation and space requirements. Hence, we briefly describe some details of the implemented prototype:

For computing modular operations, we make use of the techniques presented in [18]. That is, we exploit the RSA encryption function to perform modular exponentiation and operate on big numbers using byte arrays.

To get prime representatives for $\mathbf{r}(\cdot)$, we use the function `nextProbablePrime()` of the Java BigInteger class in our eID issuer side implementation. To also keep this efficient on the SE, we include a byte array in the update message of the issuer, indicating the offset between hash and prime representative as a single byte for each verification token.

6.2 Administration effort

The proposed scheme shifts computational load to the server backend of the issuer with the benefit of a more efficient verification and revocation check on the prover and verifier devices. We argue that it is more reasonable to increase the performance of the server than the performance of all SEs and verifier devices.

Since we assume regular re-enrollments of new disposable dynamic accumulators to reduce the revocation filter size and to ensure forward privacy, a transition phase of this re-enrollment is an essential part of the implementation of the scheme. One assumption is that the prover does not have continuous online connectivity, hence, does not immediately get the new accumulator. Therefore, the verifier needs to possess both, the previous and the current revocation filter as well as accept tokens with previous and current accumulator parameters in this phase. Note that the effective size of the new revocation filter is zero at the beginning of an accumulator phase. Previously revoked eIDs will not receive new disposable dynamic accumulators and can no longer create valid proofs.

6.3 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 time over 100 measurements was $161.4\,\text{ms} \pm 1.3\,\text{ms}$ (standard deviation). The transfer time of the interface used for sending and receiving data was $10.0\,\text{ms} \pm 0.1\,\text{ms}$. The performance of the verifier-side is elaborated in the scalability analysis in Sect. 6.4.

Binding prover devices (bind) Binding the eID-SE and eID-MA together is a step where all provable verification 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 verification tokens $c_{\max}$. For example, with $c_{\max} = 100$, the execution time on the smart card is $100 \cdot 161.4\,\text{ms} \approx 16\,\text{s}$.

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

verification 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 verification 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 Fig. 4). The SE, for example, has to increment an integer, create a new verification 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 precomputed the product of all prime representatives of $\mathcal{R}_{\text{se}}$ 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 verification 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.

Storage In terms of data storage required by the SE, our scheme only needs to store the accumulator g, the identity handle w_{se}, the disposable dynamic accumulator da_{se} including the modulus N, a byte array indicating the hash value/prime representative difference for each verification token, as well as three integers c_{se}, $c_{\text{lb}}, c_{\text{ub}}$. That is, the scheme only requires 1008 bytes of additional space (assuming a 128 byte identity handle, 2048 bit RSA and 4 byte integers).

The storage required by eID-MA depends on the number of maximum verification tokens and grows linearly. For example, with $c_{\max} = 100$, a 256 bit elliptic curve and a 2048 bit accumulator, we require 6656 bytes of storage. The required storage on the verifier's device depends on the revocation filter size and is evaluated in Sect. 6.4.

Data transfer The data transfer between prover and verifier depends on the size of the used signature protocol and dynamic accumulator. If we assume 256 bit elliptic curves (EC) and 2048 bit accumulators: we need to transmit a 33 byte verification token (compressed EC point), a 256 byte accumulator and a 64 byte ECDSA signature from the prover to the verifier. The verifier has to send a 32 byte challenge to the prover.

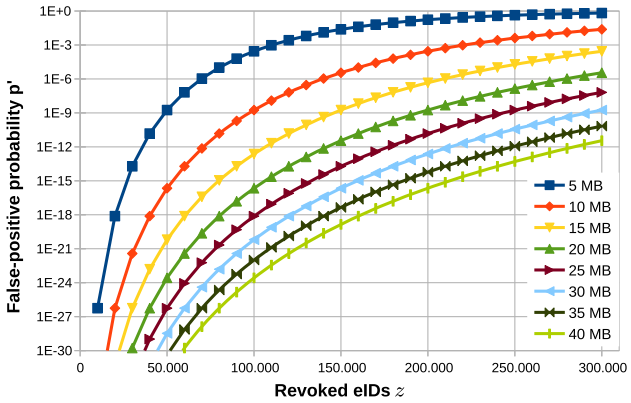

Fig. 5 Cumulative false positive probability p' of our scheme with $f_{\max} = 5$ and different filter sizes

Data transfer between prover and issuer in the online phase shall be done within a secure channel and consists of: the counter value (4 byte integer), a dynamic accumulator (256 bytes), and a byte array for the prime representative differences ($c_{\max} = 100$ bytes).

The amount of data transferred between verifier and revocation manager depends on the size of the revocation filter or the size of the differential update.

6.4 Revocation scalability

The scalability of a revocation scheme usually depends on the complexity of revocation checks and the size of the revocation list. Our scheme has the benefit that the revocation check neither depends on the number of issued eIDs nor on the population size and can be performed in constant time. This works due to the usage of Bloom filters where every check only requires the computation of j hash and modulo operations. Only the size of the revocation list as well as the false positive probability play an important role in the scalability of our scheme:

False positive probability The false positive probability p' of the revocation filter is computed with Eq. (14) and depends on the size of the filter, the number z of revoked eIDs as well as the mitigation parameter $f_{\max}$. For this evaluation, we set the mitigation parameter to a constant value of 5.

Figure 5 depicts this probability with increasing number of revoked eIDs and different filter sizes. For a filter size of 5 MB, the probability is far below 10^{-3} if less than 100,000 eIDs are revoked and rapidly grows up to 1 afterward. The diagram also illustrates that the number of revoked eIDs fit-

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

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

ting into the filter (i.e., probability below a certain threshold) linearly increases with the filter size.

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 a second query fails is then 10^{-18}. Due to our false positive mitigation technique, there are $f_{\max}$ possible verification tokens within one such query. The false positive probability p of a single verification token, given the target probability p', is computed as $p = \sqrt[f_{\max}]{p'}$ (see Eqs. 1 and 14). Consequently, with $f_{\max} = 5$ we get a single false positive probability $p = 1.58 \cdot 10^{-2}$.

Filter size For the total filter size, we have to decide on a number of maximum verification 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 Eq. (2) we can now calculate the required filter size m with a target probability of $p = 1.58 \cdot 10^{-2}$ ($p' = 1 \cdot 10^{-9}$) 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 download differential updates of this filter. Due to the characteristics of Bloom filters (many consecutive zeros), these updates can also be efficiently compressed.

6.5 Comparison with related work

In order to make a comparable evaluation of our scheme, we only consider related work in the field of privacy-preserving eIDs that also propose a revocation mechanism with the same privacy properties as our scheme (anonymity, unlinkability, backward unlinkability). As these related papers specialized on revocation, we particularly investigated and evaluated properties related to the revocation capability of an eID system. That is, we compared the following properties with related work in this field: scalability, efficiency, verifier-local capability, as well as offline capability of the revocation scheme. Furthermore, the scalability and efficiency properties are divided into the complexity to perform the revocation check, the required sizes for tokens as well as the revocation list, and the requirement of provers to receive updates when a revocation occurs. We analyze these properties as time and space complexity in the big O notation with respect to a growing revocation list. To reduce the size of this comparison, we only investigated related work that already provides the privacy properties (anonymity, unlinkability, backward unlinkability).

The comparison result with related work by Boneh et al. [5], Nakanishi et al. [26], Camenisch et al. [11], Kumar et al. [21], and Lueks et al. [24] is shown in Table 3. In comparison with other work, our scheme provides full offline capabilities and has constant cost for the revocation check and the token size. Only the revocation list grows linearly with the number of revoked eIDs z. For the revocation list size, there are other approaches which provide better scalability through constant sizes (see accumulators by Camenisch et al. [11]). However, our scheme outperforms in other scalability properties: accumulators require witness updates for all provers when eIDs are revoked, thus, keeping all provers updated in a large population becomes significantly more difficult. Due to this, it also lacks the offline capability.

Another comparable approach is the probabilistic revocation by Kumar et al. [21]. Similar to our scheme, it has an increasing revocation list size and a constant time for the revocation check. However, their scheme does not only depend on the size of the revocation list, but also on the size of the population. Hence, with an increasing population size (and therefore an increasing number of revoked IDs), the size of the alias codes needs to be expanded in order to keep the same false positive probability and a sufficient number of their so-called *piecewise-orthogonal-codes*.

Also comparable is the approach proposed by Lueks et al. [24]. Their scheme provides almost the same scalability properties as our design. However, their usage of a time-bound generator for the revocation token would require a smart card to connect to a trustworthy time source. Hence, the offline capability can not be fulfilled for a smart card scenario. Furthermore, the whole revocation list has to be downloaded in each time epoch, which need to be very short in order to provide unlinkability (i.e., large and frequent data transfer between verifier and revocation manager required).

Table 3 Comparison of privacy and scalability features of revocation schemes in different privacy-preserving eID systems

	Boneh et al. [5]	Nakanishi et al. [26]	Camenisch et al. [11]	Kumar et al. [21]	Lueks et al. [24]	Our scheme
Complexity of revocation check	$O(z)$	$O(z)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Revocation token size	$O(1)$	$O(1)$	$O(1)$	$O(z)$	$O(1)$	$O(1)$
Revocation list size	$O(z)$	$O(z)$	$O(1)$	$O(z)$	$O(z)$	$O(z)$
Prover updates on revocation	–	–	$O(z)$	–	–	–
Verifier-local	✓	✓	✗	✓	✓	✓
Diff. revocation list updates	✓	✓	–	✗	✗	✓
Offline capable	✓	✗	✗	✓	✗	✓

$O(1)$ stands for constant performance and $O(z)$ for linear growth with the number of revoked eIDs. '–' indicates not required properties of a scheme

7 Conclusion

In this paper, we proposed a privacy-preserving mobile eID system that provides scalable identity revocation. Our scheme is based on the usage of a new variant of dynamic accumulators, the disposable dynamic accumulator, and a pseudorandom function for the creation of identity tokens. By using Bloom filters, we keep the revocation list small and checks efficient. In the evaluation, we have shown that the introduced methods for verification and revocation checks have low computational costs for prover as well as verifier. Our scheme outperforms related work in terms of scalability, efficiency and/or privacy in offline scenarios. Furthermore, the beneficial scalability and offline capabilities allow for deployment to large populations. Our scheme only requires irregular updates of the prover and small revocation list updates due to the use of probabilistic data structures. Moreover, it has constant verification time, independent of revocation list and population size.

Acknowledgements Open access funding provided by Johannes Kepler University Linz. This work is supported by Johannes Kepler Open Access Publishing Fund. Moreover, it has partly been carried out within the scope of the Josef Ressel Center for User-Friendly Secure Mobile Environments (*u'smile*), funded by Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

A Formal Proof

In this section, we formally analyze the security and privacy of the proposed scheme, to demonstrate that the possibility of breaking the user's anonymity by an attacker is negligible. The hash function H, used by the participants and the attacker, is thereby modeled as a random oracle. Let *Hash* depict the random oracle and be simulated with a tuple (m, k) table of binary strings. When the adversary executes a query $H(m)$, the random oracle returns k if m exists; otherwise, it generates a uniformly random string k and keeps the pair (m, k) in the table.

We have four types of participants in our scheme: the prover P_i, the verifier V_j, the revocation manager and the issuer. Note that $\prod_{p_i}^{p}$ and $\prod_{v_i}^{v}$ are the instances of p and v of P_i and V_j, respectively. The formal security of the model is based on a game involving a challenger $\mathcal{C}$ and a polynomial-time adversary $\mathcal{A}$, which will be described below. During the game, the attacker is allowed to make the following queries to oracles that are responded to by a challenger $\mathcal{C}$.

Execute $\left(\prod_{p_i}^{p}, \prod_{v_i}^{v}\right)$ Is performed by $\mathcal{A}$ in order to get the messages transmitted among two truthful parties. This is modeled as an eavesdropping attack.

Send $\left(\prod_{p_i}^{p}, \prod_{v_i}^{v}, M\right)$ This query is appropriate for modeling an active attack. $\mathcal{A}$ has the ability to modify the transmitted message and send it to an instance of $\prod_{p_i}^{p}$ or $\prod_{v_i}^{v}$ and wait to receive a response message.

TestAnon (M_i, w_0, w_1) This oracle query is used to simulate the semantic security of the identity. After querying the oracle, the transcript of $\prod_{p_i}^{p}$ with identity w_0 or w_1 will be returned according to a random coin bit c. If $c = 1$, the adversary returns the transcript of $\prod_{p_i}^{p}$ with identity w_1; otherwise the adversary generates the transcript of $\prod_{p_i}^{p}$ with identity w_0.

In the random oracle model, the adversary $\mathcal{A}$ is challenged in an experiment to distinguish between an instance's real identity and a random number. After carrying out the above

queries, $\mathcal{A}$ produces its guess c' of c generated in the test-query. We say $\mathcal{A}$ breaks the security of the scheme, if $\mathcal{A}$ can correctly guess c' and wins the game when $c' = c$.

Let $Succ^{anon}$ denote the event in which the adversary can successfully guess c' and win the game. The advantage of $\mathcal{A}$ in breaking the anonymity of the protocol is defined to be

$$Adv_P^{anon} = \left[2 \cdot Pr\left[Succ^{anon}\right] - 1\right]. \tag{19}$$

We say that the proposed protocol provides anonymity if the advantage $Adv_P^{anon} \leq \epsilon$ is negligible.

Theorem 2 *Under the assumption of a secure one-way hash function, our scheme is provably secure against an adversary $\mathcal{A}$ for deriving the identity handle w_{se} of a prover. The probability of this by Adversary $\mathcal{A}$ is*

$$Adv_P^{anon} \leq \frac{q_h^2}{|Hash|} + \frac{2 \cdot q_s}{|D|} + 2 \cdot Adv_P^{ECDLP}(t), \tag{20}$$

where, q_h, q_s and $|Hash|$, define the number of Hash *queries, the number of send queries, the size of the hash function space, and let $|D|$ be the size of the uniformly distributed dictionary, respectively.*

Proof Assume, there is a probabilistic polynomial-time algorithm U that can break a one-way hash function by cooperating with adversary $\mathcal{A}$. We denote a sequence of games $GAME_i$, where $i = [0, 4]$. Suppose $Succ_i$ defines the event, where the adversary succeeds in guessing the bit c and wins the game. The game will start from $GAME_0$ as a real attack against the proposed scheme P and ends with the game $GAME_4$, which results in a negligible advantage of breaking the anonymity and unlinkability properties of the proposed scheme.

$GAME_0$ This game represents a real attack by adversary $\mathcal{A}$ against protocol P in the random oracle model. At the beginning of this game, the bit c is chosen at random. By definition, we have

$$Adv_P^{anon} = \left[2 \cdot Pr\left[Succ_0^{anon}\right] - 1\right]. \tag{21}$$

$GAME_1$ In this game, we simulate all oracles (*Execute*, *Send* and *TestAnon*) for each query and keep three lists to store the answers. The adversary has to make a decision whether the output of *TestAnon* is the real identity or a random number. From the simulation, we can see that the transcript distribution of the game $GAME_0$ and $GAME_1$ are indistinguishable from the real experiment. Therefore, message eavesdropping cannot help to rise the winning possibility of $\mathcal{A}$'s game. We have,

$$Pr[Succ_1] = Pr[Succ_0]. \tag{22}$$

$GAME_2$ This game simulates all oracles from $GAME_1$, except that we halt all executions in which a collision occurs in the transcript. The adversary transforms $GAME_1$ into $GAME_2$ by adding the simulation of both: the *Send* and *Hash* oracles. $GAME_2$ creates an active attack where the adversary tries to convince a participant to accept a forged message. The adversary calls several *Hash* queries to find a hash collision. Note that the exchanged messages $Rt_i = rt_i \cdot G$ and $\mathrm{da}_{Rt_i} = g^{\mathbf{r}(Rt_i)}$ are associated with w_{se} and counter value c_{se}, which are secrets. Hence, even if the *Send* and *Hash* oracles are queried by an adversary there will be no collision. Thus, the games $GAME_1$ and $GAME_2$ are indistinguishable unless collisions of group points and hash value happen. According to the birthday paradox, this gives the following probability:

$$Pr[Succ_1] = Pr[Succ_2] \leq \frac{q_h^2}{2|Hash|}. \tag{23}$$

$GAME_3$ In this game, we abort the scheme if $\mathcal{A}$ has been lucky in guessing the values w_{se} and rt_i without oracle queries. The experiments $GAME_3$ and $GAME_2$ are indistinguishable unless the participants reject a valid authentication value. Furthermore, for every transcript, there is only one w_{se} which can be tested by the adversary. Hence,

$$Pr[Succ_2] - Pr[Succ_3] \leq \frac{q_s}{|D|}. \tag{24}$$

$GAME_4$ The last game is transformed from $GAME_3$ and modeled as an attack wherein $\mathcal{A}$ tries to obtain w_{se} from a verification token $Rt_i = rt_i \cdot G$, where $rt_i = \mathrm{H}(w_{\mathrm{se}}||g||c_{\mathrm{se}})$. The adversary queries the *Execute* oracle and receives messages $M_1 = (ch)$, $M_2 = (\sigma, Rt_i, \mathrm{da}_{Rt_i})$. However, the identity Rt_i is computed as $Rt_i = rt_i \cdot G$ and computing rt_i using one of these available transcripts is computationally infeasible to the adversary. The experiments $GAME_4$ and $GAME_3$ are indistinguishable unless the following event occurs:

– The adversary solves the ECDLP problem to compute rt_i for a given $Rt_i = rt_i \cdot G$.

$$Pr[Succ_3] - Pr[Succ_4] \leq Adv_P^{\mathrm{ECDLP}}(t). \tag{25}$$

– In addition, whatever value the bit c involved in the *TestAnon* queries, the answer is random, and independent for all the sessions. Therefore,

$$Pr[Succ_4] = \frac{1}{2}. \tag{26}$$

According to the equation in $GAME_0$, we have

$$\frac{1}{2} Adv_P^{anon} = \left| Pr[Succ_0] - \frac{1}{2} \right|. \tag{27}$$

From all the games, we also have

$$\left| Pr[Succ_0] - \frac{1}{2} \right| \leq \frac{q_h^2}{2\,|Hash|} + \frac{q_s}{|D|} + Adv_P^{\mathrm{ECDLP}}. \qquad (28)$$

Thus, using the two equations above, we get

$$Adv_P^{anon} \leq \frac{q_h^2}{|Hash|} + \frac{2 \cdot q_s}{|D|} + 2 \cdot Adv_P^{\mathrm{ECDLP}}(t). \qquad (29)$$

Therefore, the proposed protocol is secure when the range of the hash function and the size of the dictionary are large. □

Corollary 4 *The revocation scheme provides backward unlinkability against the strongest adversary.*

According to Theorem 2, no polynomial adversary can acquire the secret identity handle w_{se} if the underlying secure hash function problem is hard. Therefore, we showed that in the proposed scheme it is not feasible for any adversary $(\mathcal{A}_1 - \mathcal{A}_4)$ to link previous and future verification tokens with revoked tokens on the revocation list.

References

1. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. In: IEEE European Symposium on Security and Privacy, EuroS&P (2017)
2. Baric, N., Pfitzmann, B: Collision-free accumulators and fail-stop signature schemes without trees. In: Proc. EUROCRYPT '97, LNCS, vol. 1233, pp 480–494. Springer, Berlin (1997)
3. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Proc. EUROCRYPT '93, pp. 274–285. Springer, New York (1993)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
5. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS '04, pp. 168–177. ACM (2004)
6. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS '04, pp. 132–145. ACM (2004)
7. Brickell, E., Li, J.: 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, pp. 21–30. ACM (2007)
8. Bringer, J., Chabanne, H., Lescuyer, R., Patey, A.: Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents. In: Financial Cryptography and Data Security, LNCS, vol. 8437, pp. 255–272. Springer (2014)
9. Bringer, J., Chabanne, H., Lescuyer, R., Patey, A.: Hierarchical identities from group signatures and pseudonymous signatures, pp. 457–469. Springer, Berlin (2016)
10. Camenisch, J., Drijvers, M., Hajny, J.: 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, pp 123–133. ACM (2016)
11. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Public Key Cryptography—PKC 2009, LNCS, vol. 5443, pp. 481–500. Springer (2009)
12. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Advances in Cryptology—CRYPTO 2002, LNCS, vol. 2442, pp. 61–76. Springer (2002)
13. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Advances in Cryptology—CRYPTO '97, LNCS, vol. 1294, pp. 410–424. Springer (1997)
14. Chaum, D., Van Heyst, E.: Group signatures. In: Advances in Cryptology—EUROCRYPT'91, LNCS, vol. 547, pp. 257–265. Springer (1991)
15. de Meer, H., Liedel, M., Pöhls, H.C., Posegga, J., Samelin, K.: Indistinguishability of one-way accumulators. Technical report, MIP-1210, Faculty of Computer Science and Mathematics, University of Passau (2012)
16. Goodrich, M.T., Tamassia, R., Hasić, J.: An efficient dynamic and distributed cryptographic accumulator. Information Security. LNCS, vol. 2433, pp. 372–388. Springer, Berlin (2002)
17. Haas, J., Hu, Y.-C., Laberteaux, K.: Efficient certificate revocation list organization and distribution. IEEE J. Sel. Areas Commun. **29**(3), 595–604 (2011)
18. Hölzl, M., Asnake, E., Mayrhofer, R., Roland, M.: A password-authenticated secure channel for app to java card applet communication. Int. J. Pervasive Comput. Commun. **11**, 374–397 (2015)
19. Hölzl, M., Mayrhofer, R., Roland, M.: 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, pp. 249–252. ACM (2013)
20. Hölzl, M., Roland, M., Mir, O., Mayrhofer, R.: Bridging the gap in privacy-preserving revocation: practical and scalable revocation for a privacy-aware mobile eID. In: Proceedings of SAC 2018: Symposium on Applied Computing, Pau, France, 04. ACM (2018)
21. Kumar, V., Li, H., Park, J.-M.J., Bian, K., Yang, Y.: 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 (2015)
22. Lapon, J., Kohlweiss, M., Decker, B.D., Naessens, V.: Analysis of revocation strategies for anonymous idemix credentials. In: Communications and Multimedia Security, LNCS, vol. 7025, pp. 3–17. Springer (2011)
23. Libert, B., Vergnaud, D.: Group signatures with verifier-local revocation and backward unlinkability in the standard model. In: Cryptology and Network Security, LNCS, vol. 5888, pp. 498–517. Springer (2009)
24. Lueks, W., Alpár, G., Hoepman, J.-H., Vullers, P.: Fast revocation of attribute-based credentials for both users and verifiers. In: ICT Systems Security and Privacy Protection, LNCS, vol. 455, pp. 463–478. Springer (2015)
25. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York (2005)
26. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In: Advances in Cryptology—ASIACRYPT 2005, LNCS, vol. 4266, pp. 533–548. Springer (2005)
27. Nakanishi, T., Funabiki, N.: A short verifier-local revocation group signature scheme with backward unlinkability. In: Advances in Information and Computer Security, LNCS, vol. 4266, pp. 17–32. Springer (2006)
28. Nakanishi, T., Funabiki, N.: 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, pp. 337–340. ACM (2008)
29. Raya, M., Papadimitratos, P., Aad, I., Jungels, D., Hubaux, J.P.: Eviction of misbehaving and faulty nodes in vehicular networks. IEEE J. Sel. Areas Commun. **25**(8), 1557–1568 (2007)
30. Roland, M., Langer, J.: Comparison of the usability and security of NFC's different operating modes in mobile devices. e & i Elektrotechnik und Informationstechnik **130**(7), 201–206 (2013)
31. Sander, T., Ta-Shma, A., Yung, M.: Blind, auditable membership proofs. In: Financial Cryptography, LNCS, vol. 1962, pp. 53–71. Springer, Berlin (2000)
32. Sun, J., Zhang, C., Zhang, Y., Fang, Y.: An identity-based security system for user privacy in vehicular ad hoc networks. IEEE Trans. Parallel Distrib. Syst. **21**(9), 1227–1239 (2010)
33. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Blacklistable anonymous credentials: blocking misbehaving users without ttps. In: Proc. of the 14th ACM Conference on Computer and Communications Security, pp. 72–81. ACM (2007)