Analysis of Revocation Strategies for Anonymous Idemix Credentials

Jorn Lapon[1], Markulf Kohlweiss[3], Bart De Decker[2], and Vincent Naessens[1]

[1] Katholieke Hogeschool Sint-Lieven, Industrial Engineering
[2] Katholieke Universiteit Leuven, CS-DISTRINET
[3] Microsoft Research, Cambridge

Abstract. In an increasing information-driven society, preserving privacy is essential. Anonymous credentials promise a solution to protect the user's privacy. However, to ensure accountability, efficient revocation mechanisms are essential. Having classified existing revocation strategies, we implemented one variant for each. In this paper we describe our classification and compare our implementations. Finally, we present a detailed analysis and pragmatic evaluation of the strategies.

Keywords: Anonymous Credentials, Revocation, Privacy, Performance.

1 Introduction

Individuals release a lot of personal information to all kinds of service providers. Users have no impact on what is done with these data once they are released. Service providers can process them to build detailed profiles. The latter can be privacy sensitive (depending on the content) and possibly lead to discrimination, extortion, blackmail, or other illegal activities. For instance, a bookshop can detect a user's interest in books related to certain diseases and sell that information to insurance companies that can fix higher life insurance contributions/premiums. Similarly, many governments collect information about people who submitted controversial political statements on fora and discriminate these citizens.

During the last decades, many privacy enhancing technologies have been proposed and developed. They aim at offering a higher level of privacy (or anonymity) in the digital world. Examples are anonymous communication channels, anonymous e-mail and publication systems, privacy policy evaluation tools and anonymous credential systems. Anonymous credentials allow for anonymous authentication. Only the attributes – or properties thereof – that are required to access a service are proved to a service provider. For instance, users can prove to belong to a certain age category to get discounts on public transport tickets. Similarly, they only need to prove to live in the city in order to get access to the waste recycling center. Although those technologies are more privacy-friendly compared to certificate technology, revocation becomes more complex. Multiple revocation strategies have already been proposed in the literature, often with a theoretical security and performance analysis. However, a pragmatic assessment of revocation schemes for anonymous credentials is still lacking. Hence, it is very difficult to compare results due to varying security parameters and alternative software implementations. However, a critical and pragmatic comparison is crucial to bring those technologies to practice.

Contribution. The contribution of this paper is threefold. First, existing revocation schemes are classified in six categories or strategies. Second, one variant of each strategy has been implemented with comparable security parameters and added to an existing library, namely the Identity Mixer Library [1]. Third, the paper gives a detailed analysis and pragmatic evaluation of the implemented strategies. Amongst others, the security and anonymity properties, the connectivity and performance of the schemes are compared. Usable performance results are presented in the sense that all schemes were implemented within the same library and run on the same platform. From this evaluation, guidelines are derived that can be applied by software developers that have to select a revocation strategy – or a combination of them – in a particular setting.

The rest of this paper is structured as follows. Section 2 introduces basic building blocks that are used throughout the rest of this paper. Thereafter, revocation schemes are classified according to six categories. Section 4 gives a pragmatic complexity analysis and compares multiple functional properties of the schemes. Section 5.2 focuses on implementation details and results, extracted from a practical realization after which guidelines are formulated. The guidelines assist the developer in selecting a strategy – or a combination of them – in a particular setting. Finally, Section 7 discusses further work and general conclusions.

2 Preliminaries

We shortly discuss some key elements, important for a good understanding of the rest of the paper.

Anonymous Credentials. Anonymous credential systems [2–5] allow for anonymous yet accountable transactions between users and organizations. Moreover, selective disclosure allows the user to reveal only a subset of possible properties of the attributes embedded in the credential: e.g. a credential with the user's date of birth as an attribute can be used to prove that the owner is over 18 without disclosing the exact date of birth or other attributes. We focus on anonymous credentials in which multiple shows are unlinkable [2,3,6]. All these schemes originate from existing group signatures and identity escrow schemes.

Proofs of Knowledge. Anonymous credential systems heavily rely on proofs of knowledge. These allow a prover to convince a verifier about its knowledge of certain values, without leaking any useful information. For instance, when showing a credential, the prover only wants to prove knowledge of a valid credential, and possibly disclose properties of attributes in the credential. To address revocation, these proofs are further extended to prove that the shown credential is indeed not revoked.

Cryptographic Accumulators. An important, but more complicated revocation mechanism that will be discussed are based on so-called cryptographic accumulators. A *cryptographic accumulator*, first introduced by Benaloh and de Mare [7], is a construction which allows the accumulation of a number of elements into one value. The size of this value is independent of the number of elements incorporated. For each accumulated element, there is a witness that allows to prove that the element is contained in the accumulator. It must be infeasible, for the adversary, to find a witness for an element that is not included in the accumulator. Camenisch and Lysyanskaya [8] further extended this notion into *dynamic accumulators*. In dynamic accumulators adding and removing values and updating individual witnesses can be done dynamically [4].

3 Revocation Strategies

In traditional credential systems, verifying the revocation status of a credential is straightforward and involves a simple lookup of a revealed credential specific identifier in a list. Well-known examples are *OCSP* [9] and *CRL* [10] based schemes. This strategy can be used for both *local* and *global* revocation. A revocation authority controls the validity of the credential globally, while services can use the identifier for local access control. We can distinguish two types of lists: *blacklists*, in which only revoked credentials are listed; and *whitelists*, in which only valid credentials are listed. Moreover, the time can be limited between credential revocations and a service still accepting the credential show as valid (latency).

In anonymous credential systems, on the other hand, the credential specific identifier is no longer revealed. In fact, this is one of the key requirements of anonymous credentials. In literature several revocation strategies for anonymous credentials have been developed, with each their advantages and drawbacks. Although some revocation mechanisms perform well for small groups, we focus on revocation schemes suitable for large scale settings such as electronic identity cards and e-passports. Efficiency in processing and communication is crucial.

We distinguish three parties: the issuer I handles the issuance of credentials; the user U obtains credentials; and the service provider SP to whom users authenticate. Although the ultimate goal is to make the overhead caused by the revocation strategy as little as possible, most strategies assign substantial workload to one of these parties. Nevertheless, for some strategies there may be an additional overhead for other parties as well. Based on this, we identify four classes. In the first class, Limited Overhead, none of the parties gets a big payload to handle revocation. In fact we will see that none of those solutions are satisfactory for anonymous credential revocation. For the other three classes, handling revocation assigns a substantial payload to at least one of the parties (i.e. Issuer, User or Service Provider) in the scheme.

3.1 Limited Overhead

Pseudonymous Access [Nym]. Though, more related to service usage [11], a simple and efficient solution requires the owner to provably disclose a domain specific pseudonym [12, 13]. The service provider or a trusted party of that domain is in charge of creating and modifying the list of accepted or revoked pseudonyms. Although the domain specific pseudonym can be used for local access control, it cannot be used for a global revocation of the credential. Moreover, the user is no longer anonymous as all his transactions in the same domain are linkable.

Verifiable Encryption [`VE`*].* Although verifiable encryption is often cited in anonymous credential schemes related to anonymity revocation [14, 15], it could be used for revocation as well. Hence, the user verifiably encrypts the credential's identifier with the public key of the issuer. To verify the revocation status, the service provider sends the ciphertext to the issuer, that decrypts the ciphertext. The issuer can now use the obtained identifier to do a simple lookup of the revocation status of the corresponding credential and report it to the service provider. This solution is closely related to the OCSP protocol in traditional credential schemes, with only little overhead. However, the user requires a lot of trust in the issuer, since it is able to monitor the usage of the credential (i.e. to which service providers the credential is shown). A possible solution is to require the service provider to make this request over an anonymous channel. Furthermore, replacing the public key of the issuer with the public key of another trusted third party, allows to have a separate authority in charge of the revocation tasks. Moreover, if the encrypted identifier is replaced with a domain specific pseudonym, a domain specific revocation authority may take care of access control in a certain domain.

In spite of the `Nym` and `VE` strategies, a practical and privacy friendly revocation strategy with limited (constant) overhead is not yet available.

3.2 Issuer

In the most naive solution, both the group public key and the credentials of each user are reissued whenever a party is revoked or added to the group. This solution, though its security is high for its zero latency, results in an unacceptable overhead for both users and issuers in large scale settings, hence, it is impractical. The Limited Lifetime and Signature Lists, discussed below, are two schemes requiring the issuer to frequently generate updates for users.

Limited Lifetime [`LL`*].* In this scheme, an attribute expressing the lifetime of the credential, is enclosed. During each authentication, the user proves that the credential has not expired. The lifetime of a credential highly determines the usability of the revocation scheme. A short lifetime requires the user to frequently re-validate the credential, while a long lifetime makes the scheme insecure. Instead of reissuing new credentials, Camenisch et al. [16] pointed out that non-interactive credential updates are a useful replacement. The issuer generates credential update info for all valid credentials before the end of the credential's lifetime is reached. Before the user can authenticate, the user has to download this information and update his credential.

Signature Lists [`RL`*].* Similar to CRLs in traditional schemes, it is possible to maintain revocation lists in anonymous credential schemes. However, the verification is more complicated. Instead of the service provider performing the verification, the user has to prove that the credential is not revoked. In the case of whitelists, the list consists of signatures on the identifiers of each valid credential and a list identifier. The user selects the signature in the whitelist containing the identifier of his credential and then proves knowledge of the identifier together with the proof that the credential identifier in the signature is the same as the one contained in the credential being validated. Additionally, the list identifier is revealed, such that the service provider can verify that the latest list was used.

For blacklists, proving non-membership is more complex. T. Nakanishi et al. [17] propose an elegant solution by ordering the list of revoked identifiers. For each consecutive pair of identifiers, the issuer publishes a signature on the pair, together with an identifier of the list. During a credential show, the user then proves knowledge of his credential and a signature from the blacklist, such that the identifier in the credential lies between two revoked identifiers in the ordered blacklist. Similar as in the case of whitelists, the disclosed list identifier shows that the latest revocation list was used. If this proof verifies successfully, the service provider is ensured that the credential is valid with respect to the latest blacklist.

In the latter two schemes, the effort of the issuer is significant. For every change that requires the removal of a signature from a whitelist or addition to the blacklist, the issuer has to rebuild the entire revocation list with a new list of identifiers. In case of a join in the whitelist, it is sufficient to add only one signature to the latest whitelist. Likewise, re-approving a previously revoked credential can be done by replacing two consecutive signatures by one new signature. Nevertheless, authentication in both schemes proving (non-)membership results in a non-negligible, but constant overhead.

3.3 User

Accumulators [`Acc`*].* A more complex, but possibly more efficient solution for credential revocation is based on so-called *dynamic accumulators* [18–20]. The user needs to prove membership or non-membership in the accumulator, during authentication for whitelist, resp. blacklist revocation. The service provider therefore fetches the latest accumulator value from the revocation authority and if the proof of the credential show verifies correctly w.r.t. that accumulator value, the service provider is ensured that the credential has not been revoked. Except for the verification of a more elaborate proof, the service provider has no additional overhead. On the other hand, although building this proof can be done quite efficiently, it requires the user to first update its witness, which is time-consuming. The latter enables proving (non-)membership in the accumulator. Moreover, since revoking and possibly also adding credentials to the group change the value of the accumulator, a witness update is required. These updates require resources (e.g. exponentiations [18, 19], storage [20]) linear to the number of added or revoked credentials from the accumulator.

3.4 Service Provider

Verifier Local Revocation[`VLR`*].* For many applications, the resources available to users to perform these witness updates, are very limited. In this case verifier local revocation[21, 22], first introduced by E. Brickell et al. [33], may come to the rescue.

Service providers download a list of items each linked to a revoked credential. During authentication, the user provably reveals a token allowing the verifier to check that the token is not related to any of the items in the list. Therefore, as the service provider has to check each item in the list, verification takes a (maximum) number of resources linear with the number of revoked credentials. Batch verification techniques try to tackle this [23]. Note that in some VLR schemes [33,21], all signatures made with the same credential become linkable after its revocation. Therefore, more recent schemes ensure *backward unlinkability* [22] such that former credential shows remain unlinkable.

This strategy has been adapted by the Trusted Computing Group for the use in trusted platform modules (TPM) [24]. Note that in this case, revocation is only possible if the private key is revealed to the public. As long as the corrupted private key is kept secret by the adversary, revocation of the corrupted TPM is not possible.

4 Discussion

As we focus on strategies rather than on specific revocation schemes, the analysis of the strategies makes abstraction of scheme specific details. Nevertheless, we do not hesitate to pinpoint the advantages of some specific schemes.

Complexity. All strategies try to tackle the same problem in a different way. For some strategies, the complexity analysis is obvious, in others it is rather subtle. Table 1 shows the complexity of the most expensive computations for each scheme. We assume that the average number of valid users ($\#\tilde{U}$), is constant. The last column illustrates the frequency of occurrence of these complex computations.

Table 1. Total complexity of the most computationally intensive processing during an interval Δ

		Complexity	Description	Frequency
Nym		$Ord(1)$		–
VE		$Ord(1)$		–
LL	I:	$Ord(\#\tilde{U})$	creation of credential updates for each valid credential	$\frac{1}{\Delta_t}$
RL$_w$	I:	$Ord(\#\tilde{U})$	creation of signatures for each valid credential	$\frac{1}{max(\Delta_t,\Delta_c)}$
RL$_b$	I:	$Ord(\#R)$	creation of signatures for each ordered pair of revoked credentials	$\frac{1}{max(\Delta_t,\Delta_c)}$
Acc	U:	$Ord(\#R^{\Delta}[+\#J^{\Delta}])$	update of the user's witness	$\frac{1}{\Delta_c}$
VLR	V:	$Ord(\#R)$	verifying the list of revoked credentials	every verification

$\#\tilde{U}$:	average number of members	Δ_t:	time between list updates.
$\#R^{(\Delta)}$:	revoked members (since last update)	Δ_c:	time between revocations/joins.
$\#J^{(\Delta)}$:	joined members (since last update)		

The table confirms the classification in section 3. For both Nym and VE the workload is constant for every party. Further, the LL and RL strategies require the issuer to frequently compute updates, resp. signatures for valid or revoked credentials. As mentioned before, updating the list in the RL strategies is not required as long as no identifiers are removed from the list. As opposed to LL, in which after each time-interval, the issuer computes for every valid credential a new credential update.

Accumulator based strategies (Acc), on the other hand, alleviate the work of the issuer by moving a part of the computation to the users. In fact accumulator updates can be done quite efficiently and in batch by the issuer (e.g. 1 multibase exponentiation in the case of [8]). However, now the user has to perform a number of complex computations (i.e. exponentiations in [8,25]) linear in the number of added and removed credentials. The accumulator scheme by Camenisch et al. [26] is in this sense quite efficient. Using the so called state information, users can efficiently update their witness by a number of multiplications. However, in large scale settings, the information required to perform the update is considerably large. Hence, special update servers are required to make the updates efficiently, since they may keep the state information in memory. To keep the number of changes of the accumulator in whitelist based accumulators to a minimum, during setup the issuer can accumulate a large set of unused identifiers. Once the issuer issues a credential it fetches a free identifier from the set and includes it in the credential. As such, the accumulator does not change whenever new users join the group. Instead of updating the accumulator after each addition or removal, it is possible to update the accumulator value only after a certain time, similar to the case of RL schemes. However, to increase flexibility and decrease latency, a list of the latest accumulators can be published, and allow the service provider to decide which accumulator values are still acceptable. Hence, the service provider may decide to accept proofs with older accumulators. Finally, often the issuer can perform the witness updates more efficiently [8]. However, in this case, the user is subject to timing attacks in cases the issuer and service provider collude.

Finally, in the VLR strategy, the verifier carries the burden. In case of a valid credential, the verifier has to perform a computation for every item in the revocation list. There exist VLR schemes [27] that improve efficiency of the verification, however, for large scale settings the complexity of the credential show and the memory load become significant. Although verification of the validity of signatures (i.e. equality) can be done in batch [28], batch verification of revocation lists (i.e. inequality) has not been described in detail in literature.

Functional Properties. Table 2 gives an overview of some functional properties of the different strategies with respect to the basic scheme without revocation. It illustrates that there is no straightforward winner. Schemes that score clearly better with certain properties, perform worse on others, and vice versa. It is for instance clear that the Nym and VE strategies are less privacy friendly. In fact all other strategies allow full anonymity. However, to obtain full anonymity in LL, RL and Acc, the user should download the entire set of update information, since otherwise timing attacks are possible. Alternatively, a private information retrieval scheme may allow the user to download the required data more efficiently, while maintaining anonymity. Of course, in large scale settings, with many service providers and users, and since the download may be done well before the actual credential show, the dangers of timing attacks may be negligible.

The security of the LL and RL strategies, defined by the latency, perform worse than the other strategies, that allow for zero latency. Note that to decrease communication overhead, Acc and VLR can accept a non-zero latency, by accepting older accumulators resp. revocation lists.

Table 2. Functional properties ($\downarrow$ and $\uparrow$ shows the relation w.r.t. the basic credential scheme without revocation)

	Nym	VE	LL	RL	Acc	VLR
Anonymity	$\downarrow$	$\downarrow$				
Latency			$\uparrow$	$\uparrow$		
Netw. Conn.			U	U (SP)	U (SP)	SP
Download (U/SP)	-/-	-/-	$\uparrow$ / -	$\uparrow$ / -	$\uparrow$ / -	-/$\uparrow$
Global/Local	L	$G[L]$	G	G	G	G

To decrease latency in the case of LL and RL, the frequency of issuing update information resp. revocation lists should be higher than the frequency of revoking credentials. This is computationally expensive especially in the large scale settings that we envision. Nevertheless, both LL and RL can be useful in environments with lower security requirements.

VLR schemes use blacklisting. RL and Acc, on the other hand, allow for both black- and whitelisting. In the case of RL schemes, while a proof of membership may be more efficient in the case of whitelists, some settings advocate for blacklist based schemes with possibly more efficient updates. Especially, the "valid versus revoked credentials" ratio determines which strategy is the better choice. In the case of accumulator based revocation, the difference between white- and blacklists is rather subtle.

The table further shows that the user is required to be online for LL, RL and Acc. The service provider may have to download information for RL, Acc and VLR. However, for both RL and Acc it is possible to avoid downloads. In the case of RL, the service provider can simply verify the revealed validity time of the shown signature. If it lies in an acceptable (small) time interval, it accepts the credential show. Otherwise, it requires the use of a newer revocation list. In case of Acc, the user could provide the signed accumulator to the service provider. Note that the amount of data to be downloaded in case of the Acc by the user and VLR by the service provider may be substantial. For some VLR schemes, such as the one of Ateniese et al. [22], to obtain high security the revocation list requires frequent updates, resulting in even more data traffic.

Combining strategies. As already discussed, the different schemes expose different properties. To maximize the advantage of those properties, multiple strategies can be combined in the same credential scheme. For instance, an updatable lifetime may be used in parallel with accumulators. The lifetime may be sufficient in low-security environments, while a service requiring high-security may require the same user to prove membership in the latest accumulator. In another example, Nym could be used for local access control, while another strategy is used for verifying the global revocation status. In fact all strategies discussed are compatible and only require the issuer to include the appropriate attributes in the credential.

5 Implementation

5.1 Implementation Notes

One of the most versatile anonymous credential systems available to date is the Identity Mixer system [1], short Idemix. Some of the schemes (i.e. `LL`, `Nym` and `VE`) are readily available in this library. We extended the library with the other revocation strategies mentioned. For `RL` and `Acc` both a white- and a blacklist scheme is implemented as well as a `VLR` scheme. More details are given below. Note that our choice of schemes was restricted by the cryptographic schemes used in the Identity Mixer. For instance, the library does not implement pairings, heavily limiting the number of possible schemes. Note that the implementation was done respecting the architecture and design of the library as much as possible. In fact, all extensions can be optionally activated depending on the proofspecification. Most of the implementation effort went to the extended proofs of the credential shows. Except for the declaration and parsing of the appropriate attributes in the credentialspecifications, there are no major additions to the issuance of the credentials. An optional <*Revocation*> element has been added to the proofspecification, in which <*VLR*>, <*Accu*> and <*RevocationList*> elements allow to declare the revocation scheme applied during the credential show.

Like the calls in the library to the extensions for proving, for instance, inequality and commitments, we added calls to the appropriate extensions (i.e. `VLR`-*Prover* and *Verifier*, `Acc`-*Prover* and *Verifier* and `RL`-*Prover* and *Verifier*), in the *Prover* and *Verifier* class. These handle the revocation scheme specific proofs. The credential shows in Idemix are implemented as common three-move zero-knowledge protocols (called sigma protocol), made non-interactive using the Fiat-Shamir heuristic [29]. The provers, in a first round compute the scheme specific t-values (i.e. the values in the first flow of the sigma protocol) and add those values to the computation of the challenge c. Then, the prover computes the s-values (i.e. the responses of the third flow of the sigma protocol) and adds them to the message sent to the verifier. Finally, at the verifier, the $\hat{t}$-values are computed based on the s-values and verified with the received common values. The extensions use the security parameters used in the original Idemix library for the construction of the proofs.

Signature Lists. The signature lists for both white- and blacklists are instantiated by CL signatures, which are also used in the library. They allow to prove knowledge of the signature and its attributes, without revealing them. Moreover, it allows to prove relations such as equality of the identifier in the signature and the identifier in the credential in the case of whitelists.

For blacklist revocation, an implementation was made based on the scheme of Nakanishi et al [17]. As mentioned before, the revocation list consists of an ordered list of revoked identifiers, which are pair-wise signed by the revocation authority together with a list identifier. Additionally, an unused minimum and maximum identifier is included in the list. To prove that the credential is not revoked, the user proves knowledge of a signature in the revocation list with a certain identifier such that the identifier in the credentials lies in the interval formed by the identifiers in the signature. Therefore, for the implementation we recover the inequality provers available in Idemix.

CL-Accumulator scheme. Several accumulator based revocation schemes exist. An implementation in C++, comparing three of them is available in [30]. The schemes implemented there, are all whitelist revocation schemes. One of the schemes compatible with the Idemix library is the construction by Camenisch et al [18]. Building on this construction J. Li et al [31] extended the scheme with a non-membership proof, allowing the same accumulator construction to be used for blacklisting as well. The schemes have been implemented based on the membership proof in Section 3.3 *"Efficient Proof That a Committed Value Was Accumulated"* presented in [18] and the non-membership proof defined in Protocol 1 in Section 5 *"Efficient Proof That a Committed Value Was Not Accumulated"* in [31]

DAA-VLR scheme. Finally, the VLR scheme adopted by the TCG group [24] has been implemented. In contrast to what is implemented in TPMs, in which the private key is required for revocation, a separate random identity attribute id is enclosed in the credential. The latter is then used to perform the verification. This allows the issuer to revoke the credential based on this identity, and does not require the private key of the credential to be compromised. The protocol presented in [24] defines the issuance and proof of an entire DAA anonymous credential. Our implementation extends the credential show of Idemix with the proof of knowledge $\mathrm{PK}\{(id) : N_v = \zeta^{id} \wedge \zeta \overset{?}{\in}_R \langle\gamma\rangle\}$ with id the identity of the user, and ζ a randomly chosen base. and the verification of the list of revoked values by verifying that $\zeta^{id_i} \neq N_v$ for each id_i in the revocation list.

5.2 Results

This section reports the results of two experiments. The first experiment deals with the issuance and showing of a single credential. The second experiment analyses the time required for the complex computations as in the complexity analysis. The experiments use the default security parameters (i.e. $k = 160$ bit) proposed in Appendix A, Table 2 of the Idemix library specification [1], and are executed on a DELL Latitude P9600 @ 2.53GHz, 4GB RAM. Note that since most algorithms are probabilistic, large variations in timings are possible. To make the measurements as realistic as possible and minimize overhead caused for instance by class loading, the given numbers are averages over a large number of runs. Moreover, the communication overhead is not included.

Table 3 presents for each implemented scheme, the total time required to issue and show a credential. The credential-show includes the verification of the revocation status. Since for all schemes issuing a credential does not require complex calculations w.r.t. the `Basic` scheme, issuing a credential is about the same for most schemes. A small time difference may be noticed for all but the `Nym` scheme, caused by an additional attribute required by the revocation strategy. However, as could be expected, there is more variance in showing a credential. Only the time for a credential show in the `Nym` and `VLR` scheme lies close to the `Basic` scheme. For these schemes, the small overhead is caused by the computation and disclosure of a pseudonym. Note that in `VLR` this pseudonym is randomized. For the whitelist based $\mathtt{RL}_w$ scheme, the time is doubled w.r.t. the `Basic` scheme. Here, showing a credential implies two proofs, namely one proof for proving the knowledge of a credential, and an additional proof for proving the knowledge of a signature from the revocation list, with the same identifier as in the credential. The

Table 3. Timing analysis for issuing and showing a single credential (average over 200 rounds)

(in sec.)	Issue □	Show ■	
Basic	2.5	0.8	
Nym	2.5	0.9	
VLR	2.6	1.0	
Acc$_b$	2.6	3.5	
Acc$_w$	2.6	3.8	
LL	2.6	4.2	
RL$_w$	2.6	1.8	
RL$_b$	2.6	8.6	
VE	2.6	16.0	

overhead for the credential show in the white- and blacklist accumulator based schemes, is induced by the complex membership resp. non-membership proof. A more detailed analysis may be found in [30]. It is a bit surprising that showing a credential in the LL scheme takes even more time. The reason for this is that the scheme (as implemented in Idemix) requires an expensive range proof to show that the credential's expiration time, is larger than or equal to the current time. This way the epoch strategy is very flexible, as not all users have to update as frequently as others. However, if the lifetime attribute in credentials is synchronized and the same for all credentials, it is possible to simply disclose the lifetime value. As such, the credential show takes about as much time as in the case of the Basic scheme. Similarly, showing a credential in the RL$_b$ scheme requires an additional signature proof and two range proofs. The signature proof, proves knowledge of a signature in the revocation list and the range proofs prove that the identifier in the credential lies between the revoked identifiers in the proved signature. The worst scheme is the one based on verifiable encryption. This scheme may not be practical for revocation. Moreover, this result shows that using verifiable encryption for *anonymity revocation* implies a very large overhead as well.

In the second experiment, summarized in table 4, the most complex computations as discussed in section 4 have been verified in practice. Since the total amount of valid users, in our setting will be much larger than the number of revoked users, it is clear that LL and RL$_w$ require a lot of computations by the issuer. Hence, RL$_b$ might be more interesting. However, as noted in the previous experiment, showing a credential in the RL$_b$ scheme is expensive, and may seem impractical. The accumulator based schemes have practically no overhead at the issuer's side. However, before showing his credential, a user has to update his witness. The witness update takes approximately 20ms per revoked credential, since the previous update. As stated before, it is possible to avoid witness updates as a result of joining new credentials. If it is possible to let the user have frequent witness updates, then this overhead is spread over time and may be acceptable for applications. Finally, the VLR solution shows that it only takes approximately 3ms per revoked credential, to verify the validity of a credential.

Table 4. Time analysis of the most complex computations

(sec.)	Issuer	User	Verifier
LL	$1.4 * \#\tilde{U}$		
$\mathtt{RL}_w$	$1.31 * \#\tilde{U}$		
$\mathtt{RL}_b$	$1.50 * \#R$		
$\mathtt{Acc}_b$	0.16	$0.02 * \#R^{\Delta} + 0 * \#J$	
$\mathtt{Acc}_w$	0.16	$0.02 * \#R^{\Delta} + 0 * \#J$	
VLR			$0.003 * \#R$

For the Belgian eID card[1], there are about ten million users, and about 375.000 revocations a year. We have to note though that the certificates of youngsters and kids in Belgium are automatically revoked, giving an incorrect image of the number of actual revocations resulting from lost or stolen credentials. Moreover, Belgian citizens may opt to revoke their digital certificates themselves.

Applying the schemes to this large scale setting, we have the following results. Generating update information in the LL scheme would take about 160 days. For the $\mathtt{RL}_b$ scheme with 375.000 revocations, it takes about 6.5 days. Similarly, for the VLR scheme, verifying a credential show takes 18 minutes. For the latter, however, batch verification may increase the speed. Note that in literature, there is no batch verification scheme available that is tuned for the verification that a credential is not in the list of the VLR. For VLR, the batch verification should allow to verify that *none* of the tokens in the list match with the one being verified, while in literature the authors often refer to batch verification of signatures. In this case, batch verification allows to verify that *all* signatures are valid.

Although great improvements can be reached by faster implementations and processors (e.g. an implementation in C++ of the accumulator takes only 1.5ms instead of 20ms in Java in which Idemix is implemented), these numbers show that for large scale settings, the RL, LL and VLR schemes are impractical.

6 Applying Revocation Schemes - Guidelines

It is clear that there is not one strategy superior to all the others. Therefore, we end the analysis with an overview of which strategies are useful in which settings. Nevertheless, a combination of multiple strategies may sometimes offer the best trade-off. The guidelines are summarized in table 5 and discussed below.

Security. For high security environments (i.e. requiring low latency) accumulator based revocation is the most secure and privacy-friendly strategy, closely followed by some verifier local revocation schemes. For the latter, one has to select a VLR scheme carefully that provides adequate anonymity. On the other hand, for lower security environments, LL provides a reasonable trade-off. RL offers a similar solution but is not restricted to the issuer to act as revocation manager.

[1] Results obtained from `http://godot.be/eidgraphs`

Table 5. Feasibility of the schemes w.r.t. connectivity and resources. (●: positive; ○: neutral; else: negative)

	Security	Offline		Low Resources	
		U	SP	U	SP
Nym		●	●	●	●
VE		●	●	○	○
LL	○		●	○	●
RL	○		●	○	●
Acc	●		●		●
VLR	(●)	●		●	

Processing Environments. Often a user's credential is kept in resource constrained environments (e.g. a smartcard). In this case, `VLR` schemes require the least computational overhead for the user. Also `LL` is a possible alternative. `RL` and `Acc`, however, require complex computations, making these scenarios less effective in resource constrained environments. In some settings, the verifier has limited resources (e.g. a door lock). In this case `VLR` is not an option.

Connectivity. In case of `RL` and `Acc` the user requires frequent communication with the issuer. On the other hand, for the service provider in the case of Łand `RL`, it is sufficient to keep track of time to be able to verify the revocation status. This is especially important for *offline* service providers. In this case, when computing power is not an issue, the more secure accumulators may provide an alternative. Then the user should provide the latest accumulator, signed by the revocation authority. The verifier then simply checks the validity time of the revocation list.

Online environments offer more freedom. In some schemes, computation may be outsourced to other possibly trusted environments. For instance, verification in the `VLR` setting may be done by an external more powerful party. When the verifier outsources this verification to a more powerful trusted party, it actually implements a kind of OCSP scenario. Related to accumulator schemes, some schemes [20] also take advantage of remote witness updates.

7 Conclusion

This paper classifies existing revocation strategies for anonymous credential systems into six categories. The analysis shows that there is no straightforward winner, and the effectiveness and efficiency of a specific strategy heavily relies on the setting in which the mechanism is used. To maximise the applicability of anonymous credentials, only a combination of multiple strategies may provide some relieve. Therefore, guidelines are proposed that show which strategies should be applied in which settings.

The practicality and applicability of anonymous credential schemes in real-life settings is an on-going discussion and remain important aspects to analyse. In the implementation and comparison presented in this paper, we focussed on schemes that are suitable within the Identity Mixer Library. Nevertheless, revocatoin schemes, for instance, based on pairing based cryptography or suitable within the U-Prove credential scheme [32], may show to be better alternatives. For future work, we envision to compare the results of this paper, with implementations of more revocation schemes for anonymous credentials.

Acknowledgements. This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven, and the IWT-SBO projects DiCoMas and MobCom.

References

1. Specification of the Identity Mixer Cryptographic Library – Version 2.3.2. Technical report, IBM Research – Zurich (2010)
2. Chaum, D.: Security Without Identification: Transaction Systems to Make Big Brother Obsolete. Commun. ACM 28(10), 1030–1044 (1985)
3. Camenisch, J.L., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
4. Camenisch, J., Herreweghen, E.V.: Design and implementation of the idemix anonymous credential system. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 21–30. ACM, New York (2002)
5. Brands, S.: A Technical Overview of Digital Credentials (2002)
6. Bangerter, E., Camenisch, J.L., Lysyanskaya, A.: A Cryptographic Framework for the Controlled Release of Certified Data. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2004. LNCS, vol. 3957, pp. 20–42. Springer, Heidelberg (2006)
7. Benaloh, J.C., de Mare, M.: One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
8. Camenisch, J.L., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
9. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 internet public key infrastructure online certificate status protocol - ocsp (1999)
10. Housley, R., Polk, W., Ford, W., Solo, D.: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile (2002)
11. Brands, S., Demuynck, L., De Decker, B.: A practical system for globally revoking the unlinkable pseudonyms of unknown users. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 400–415. Springer, Heidelberg (2007)
12. Camenisch, J., Mödersheim, S., Sommer, D.: A formal model of identity mixer. Formal Methods for Industrial Critical Systems, 198–214 (2010)
13. Bichsel, P., Camenisch, J.: Mixing identities with ease. In: de Leeuw, E., Fischer-Hübner, S., Fritsch, L. (eds.) IDMAN 2010. IFIP AICT, vol. 343, pp. 1–17. Springer, Heidelberg (to apppear, 2010)
14. Camenisch, J.L., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
15. Backes, M., Camenisch, J., Sommer, D.: Anonymous yet accountable access control. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 40–46. ACM, New York (2005)

16. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving revocation with efficient update of anonymous credentials. In: Security and Cryptography for Networks, pp. 454–471 (2011)
17. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable group signature schemes with constant costs for signing and verifying. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009)
18. Camenisch, J.L., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
19. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
20. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
21. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 168–177. ACM, New York (2004)
22. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient revocation of group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
23. Zaverucha, G.M., Stinson, D.R.: Group testing and batch verification. In: Kurosawa, K. (ed.) Information Theoretic Security. LNCS, vol. 5973, pp. 140–157. Springer, Heidelberg (2010)
24. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM, New York (2004)
25. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
26. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
27. Demuynck, L., De Decker, B.: How to prove list membership in logarithmic time. CW Reports, KU Leuven, Department of Computer Science, vol. CW470 (2006)
28. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
29. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
30. Lapon, J., Kohlweiss, M., De Decker, B., Naessens, V.: Performance analysis of accumulator-based revocation mechanisms. In: Rannenberg, K., Varadharajan, V., Weber, C. (eds.) Security and Privacy - Silver Linings in the Cloud. IFIP AICT, vol. 330, pp. 289–301. Springer, Boston (2010)
31. Li, J., Li, N., Xue, R.: Universal Accumulators with Efficient Nonmembership Proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
32. Stefan Brands, C.P.: U-Prove Cryptographic Specification V1.0. Technical report, Microsoft Corporation (2010)
33. Brickell, E., Camenisch, J., Chen, L.: The DAA scheme in context. Trusted Computing, 143–174