Public Key Accumulators for Revocation of Non-Anonymous Credentials

Andrea Flamini$^{2[0000-0002-3872-7251]}$, Silvio Ranise$^{1,2[0000-0001-7269-9285]}$, Giada Sciarretta$^{1[0000-0001-7567-4526]}$, Mario Scuro$^{1[0000-0003-2410-3760]}$, Nicola Smaniotto$^{2[0009-0003-0523-9926]}$, and Alessandro Tomasi$^{1[0000-0002-3518-9400]}$

[1] Center for Cybersecurity, Fondazione Bruno Kessler, Trento, Italy
`{ranise,g.sciarretta,mscuro,altomasi}@fbk.eu`

[2] Department of Mathematics, University of Trento, Trento, Italy `andrea.flamini@unitn.it`, `nicola.smaniotto@studenti.unitn.it`

Abstract. Digital identity wallets allow citizens to prove who they are and manage digital documents, called credentials, such as mobile driving licenses or passports. As with physical documents, secure and privacy-preserving management of the credential lifecycle is crucial: a credential can change its status from issued to valid, revoked or expired. In this paper, we focus on the analysis of cryptographic accumulators as a revocation scheme for digital identity wallet credentials. We describe the most well-established public key accumulators, and how zero-knowledge proofs can be used with accumulators for revocation of non-anonymous credentials. In addition, we assess the computational and communication costs analytically and experimentally. Our results show that they are comparable with existing schemes used in the context of certificate revocation.

Keywords: Accumulators · Revocation · Digital identity.

1 Introduction

Revocation is one of the most difficult aspects of credentials to manage. In a Web PKI scenario, credentials are X.509 certificates and the most well-known and tested methods to implement revocation for X.509 certificates are Certificate Revocation List (CRL) [11] and Online Certificate Status Protocol (OCSP) [58]. Both are disliked for several reasons, to the extent that both are now served through alternative protocols (CRLLite [45], CRLSets [19], OCSP with Stapling [35,61]) and OCSP is being made entirely optional [34] and discontinued by some major providers [1].

Cryptographic accumulators – binding commitments to a set of elements – have been proposed as an alternative, e.g., as an allow-list of unrevoked credentials. The key advantages claimed by accumulators over CRL and OCSP are that only a short public value needs to be updated and distributed to verifiers by a revocation authority, and membership proofs can be efficiently done in Zero-Knowledge with a computational cost independent of the list size.

Accumulator schemes and their properties have been defined and surveyed in [5,30,25,6], to which we refer for a complete characterization. They have been previously adopted in the context of anonymous credentials [41,36]. Here we focus on their application to revocation of non-anonymous credentials in digital identity wallets.

There are three key roles in a digital identity wallet ecosystem: Issuer, Holder, and Verifier. An Issuer creates and issues Credentials to a Holder. A Verifier receives and verifies Credentials presented by a Holder. As part of the verification process, the Verifier should check that a presented Credential is not revoked. The main question we aim to answer is whether accumulators are a viable alternative to more established methods for revocation of Credentials in digital identity wallets.

Outline. In Section 2 we recall well-known revocation mechanisms for X.509 certificates on the web, CRL and OCSP, and highlight some of the issues with using them in the wallet scenario. In Section 3 we characterize the revocation process in the digital identity wallet context, highlighting new privacy challenges. In Section 4 we describe the most well-established public-key accumulators, showing their relationship with digital signatures. In Section 5 we recall the Zero-Knowledge Proofs (ZKP) for the presentation and proof of non-revocation of anonymous credentials, and consider how they can be used in non-anonymous credentials such as the ones being considered for the European Digital Identity (EUDI) Wallet. In Section 6 we assess the computational complexity and experimental performance of accumulators and ZKP in management, issuing, and presentation of credentials. Based on these observations, in Section 7 we compare accumulators with existing revocation schemes. Finally, Section 8 draws the conclusions and future work.

2 Revocation in Web PKI

Two well-known revocation mechanisms for X.509 certificates are Certificate Revocation List (CRL) [11] and Online Certificate Status Protocol (OCSP) [58]. In the web PKI scenario, by using the digital identity wallet terminology we can see a X.509 certificate as a Credential, Issuers are Certificate Authorities (CAs), Holders are websites with X.509 certificates, Verifiers are user agents connecting to these sites via TLS. Certificate status information is provided either as a list (CRL) of serial numbers of all currently unexpired and revoked certificates, which Verifiers download and check against the presented certificate serial number; or as a response (OCSP) for an individual certificate, the serial number of which must be disclosed by the Verifier to the Issuer at the time of presentation.

Note that both CRL and OCSP responses are signed by the Status Manager[3], respectively called CRL issuer and OCSP responder; the means by which a Verifier establishes trust in their public keys are beyond the scope of this analysis. For a recent survey of these mechanisms, see [42].

CRL and OCSP have several issues and proposed mitigations, summarized below. In a clear sign that OCSP issues are considered more significant overall, the CA/B forum has made OCSP optional and require CRL [34], and Let's Encrypt have announced their intent to end their OCSP service entirely [1].

2.i. *Availability.* Both approaches are threatened by lack of availability. If Verifiers are unable to retrieve either list or response in a reasonable time, browsers can either hard fail – deny access to the site; or soft fail – allow access without even warning the user, as if the revocation check had never even taken place.

2.ii. *Size and computation.* CRLs encode the serial number, revocation date, and revocation reason code of each revoked certificate, and can grow to be relatively large. While the size of an OCSP response is potentially much smaller than a CRL and thus seems advantageous for Verifiers, providing a high availability service responding to status queries for frequently visited sites is costly for Status Managers and requires a digital signature for each response, as opposed to a single signature per CRL.

2.iii. *Privacy and Information Disclosure.* When the Status Manager coincides with the Issuer, an additional concern for OCSP is that it enables tracking of Holder and Verifier activity by the Status Manager, because the Verifier must request the status of a specific certificate. This constitutes an information disclosure for the Verifier, which reveals to the Status Manager when and

[3] In some cases, the Status Manager and the Status Provider, i.e. the entity responsible to issue the status of a Credential, are played by two different entities. In this paper, for simplicity, we consider a unique entity and we use always the term Status Manager.

by whom its service is requested. A CRL by contrast provides so-called *herd privacy* because the Status Manager cannot associate a given status request with any one credential.

2.iv. *Dishonest Issuer-Status Manager.* The privacy issues with OCSP are structural, because the mechanism itself inherently reveals detailed information. Both OCSP and CRL are also subject to manipulation by dishonest Status Managers and Issuers that collude together. It is trivial for a Status Manager to create ad-hoc lists or OCSP responder endpoints specifically for individual certificates, and the only way to know these exist is to examine each certificate.

The following mitigations have been implemented for CRL and OCSP.

CRLSets/CRLite. Browsers preload CRL information as CRLSets [44,19] (Chrome) or CRLite [49,45] (Mozilla), but address the CRL size issue differently. CRLSets are a small subset of critical CRLs and primarily a means by which browsers can quickly block critical certificates in emergency situations. CRLite is a compressed data structure based on cascading probabilistic filters, such as Bloom [10]. While the information in CRLite is more complete, waiting for it to be updated is an additional delay over individual CRL, and the probabilistic nature of filters introduces the risk of false positives, potentially requiring a backup mechanism.

OCSP Stapling [28]. Websites periodically request an OCSP response, refreshing their locally held status proof, then attaching ("*stapling*") the latest copy of the response to the certificates they present. As an additional security measure, the Issuer (CA) may specify in the certificate itself that a stapled response must be included in the Holder's presentation [35]. As of May 2024, 50% of sites surveyed by ssllabs support OCSP stapling [55].

Both solutions avoid latency or availability issues for Verifiers contacting Status Managers (Item 2.i.), minimize requests and response signatures for the Status Manager (Item 2.ii.), and mitigate against privacy issues (Item 2.iii.).

3 Revocation in Digital Identity Wallets

The wallet scenario adds a number of significant privacy concerns to the Web PKI issues, which should be addressed to give users the same guarantees they would expect from using a physical scheme; we note these issues here, then make our assessment of how traditional revocation mechanisms compare with accumulators in addressing them in Section 7.

3.i. *Status Manager monitoring Holder.* Is the Status Manager able to track presentations the Holder makes to Verifiers?

3.ii. *Verifier monitoring Credential status.* Is it possible to keep checking the status of a Credential over time without the Holder's consent after a presentation?

3.iii. *Verifier linking Holder presentations.* Can one or more Verifiers track the Holder's activity? This may be between different presentations to the same Verifier, or to multiple colluding Verifiers. This may be due to any unique identifiers disclosed to the Verifier, such as Holder public keys or credential serial numbers.

3.iv. *Status Manager monitoring Verifier.* Is the Status Manager able to track the status requests made by a Verifier? This could leak commercially sensitive information.

3.v. *Third Parties monitoring Status Manager.* Can anyone analyze the information made publically available by the Status Manager to gain information about the Status Manager's handling of the scenario? This could lead to collection of revocation statistics.

Privacy risks 3.i., 3.iii. and 3.iv. are also discussed in the Architecture and Reference Framework (ARF) [26], defining the technical specifications for the European Digital Identity (EUDI) Wallet to be compliant with the revised eIDAS 2 regulation [29]. ARF is currently in development, some topics need to be addressed and discussed up to end of 2025. The first topic is related to "privacy risks and mitigations" and discusses linkability threats related to revocation checking.

ARF currently suggests the use of short-lived credentials that have a validity period of 24 hours or less, such that revocation will never be necessary,[4] or to support a list-based revocation mechanism: a Revocation List of identifiers of revoked or suspended attestations, or a Status List publishing status information for all valid attestations. While the ARF does not specify technologies to realize these revocation requirements, prominent list candidates include CRL and Token Status List [47]. An alternative proposal is OAuth Status Assertions [24].

In addition, ARF discusses the possibility of using Zero-Knowledge Proofs (ZKP) referring to the Cryptographers' feedback on the ARF [7], which focuses on anonymous Credentials [5]) through selective disclosure signatures – see [33,9] for reviews.

The feedback suggests forgoing revocation by issuing Credentials with short-lived validity: "the anonymous credential would contain an expiration date as dedicated attribute. When showing the credential, the owner then either discloses the attribute or proves in zero-knowledge that his credential hasn't expired yet." The scenario of long-lived Credential revocation remains open.

Accumulators are currently not part of ARF as they have been mainly adopted in the context of anonymous Credentials and an analysis of their efficacy and applicability in the context of non-anonymous Credentials is currently missing. To fill this gap, in this paper we aim to answer whether accumulators are a viable alternative to the current proposals for revocation of Credential in the wallet scenario by assessing the computational complexity and experimental performance of accumulators and ZKP in management, issuing, and presentation of Credentials.

4 Accumulators

Cryptographic accumulators are schemes for the concise representation as a single value α of a finite set $\mathcal{S}$ of elements, and the generation of a so-called membership witness w_x for each x, which may be used to verify that $x \in \mathcal{S}$. Depending on the scheme, it may also be possible to generate non-membership witnesses $\overline{w}$ for elements not in the set; and the proofs may be in zero knowledge.

Accumulator schemes and their properties have been defined and surveyed in [5,30,25,6], to which we refer for a complete characterization. Commonly used terms to categorize accumulators are:

Additive / subtractive / dynamic: elements can be efficiently [added to / removed from / both] the accumulated set.

Positive / negative / universal: supports [membership / non-membership / both] proofs.

When elements are added to or removed from an accumulated $\mathcal{S}$, the value α may change; we denote its value after t updates as α_t.

In general, accumulators *may* define the following functions.

[4] The 24-hour period originates from the relevant ETSI specifications. These require that the process of revocation must take at most 24 hours. Consequently, revocation makes no sense if the Credential is valid for less than 24 hours, because it will reach the end of its validity period before it is revoked.

[5] Anonymous Credentials are currently not in the ARF v1.5 (Annex 2, Topic 12) requires the support of: mDOC as specified in ISO/IEC 18013-5 [38], SD-JWT VC as specified in SD-JWT-based Verifiable Credentials [31] and W3C VCDM v1.1 as defined in W3C Verifiable Credentials Data Model [60].

`Gen`: Given a security parameter λ and a set of elements $\mathcal{S}$ in a domain D, output a Status Manager private key `sk`, public parameters `params`, an initial accumulated value α_0.

`GenWit`, $\texttt{Gen}\overline{\texttt{Wit}}$: For an element x, generate a witness w_x or non-witness $\overline{w}_x$ to prove membership, resp. non-membership, of x in $\mathcal{S}$.

`Add`, `Del`: Update the accumulator value α, either to include a new (not previously accumulated) element x, or to exclude a previously accumulated one.

`WitUp`, $\overline{\texttt{WitUp}}$: In a dynamic accumulator, membership and non-membership witnesses may need to be updated; different update functions may be defined depending on whether the update was an addition – $\texttt{WitUp}_+$, $\overline{\texttt{WitUp}}_+$– or a deletion – $\texttt{WitUp}_-$, $\overline{\texttt{WitUp}}_-$.

`VerMem`, $\texttt{Ver}\overline{\texttt{Mem}}$: Anyone with knowledge of an element x, a (non-)witness $(\overline{w}_x), w_x$, and an accumulator value α may verify (non-)membership of $(x \notin \alpha), x \in \alpha$, respectively. A trusted Status Manager may have its own, more efficient version `VerMemMan`.

`GenZKP`, `VerZKP`: Zero knowledge proofs can be defined for the Holder to prove knowledge of membership or non-membership of an element – $\texttt{GenZKP}_\in$, $\texttt{GenZKP}_\notin$ – without the Verifier learning the value of that element – $\texttt{VerZKP}_\in$, $\texttt{VerZKP}_\notin$.

Accumulators can be used as allow-lists or deny-lists, in principle. As allow-list, newly issued credentials would be added, revoked credentials would be removed, Holders would present proofs of membership – i.e., the accumulator would be positive – and Verifiers would reject presentations without such proof. As deny-list, revoked credentials would be added, Holders would present proofs of non-membership – i.e., the accumulator would be negative – and Verifiers would reject presentations without such proof.

4.1 Choosing an accumulator scheme

The main limitation of accumulators in real-world applications lies in the management of witness updates. Given a the number of additions and d the number of deletions occurred since the last time a witness was updated, Camacho and Hevia [15] showed a theoretical lower bound for the number of information/operations needed for updates. Universal accumulators require at least $O(a + d)$ operations; while, for accumulator schemes adapted to have *optimal update complexity* (α does not need to be updated either on the deletion or on the addition of new members to $\mathcal{S}$), the bound is reduced to $O(a), O(d)$ respectively for negative, positive accumulators.

In the wallet context, the best-performing choice is therefore to update only when a Credential is revoked as it is less frequent than Credential issuance. For example, public data on the eIDAS-notified Italian eID (CIE) – which may be considered analogous to Person Identification Data (PID) – the ratio between revoked and issued certificates is typically low; in 2022, 7 million new ID cards were issued [27] and 200 000 were revoked [48], resulting in a revocation rate of about 3%. This means that the best options are either positive accumulators used as allow-lists, or negative accumulators used as deny-lists. From now on we will consider only positive accumulators used as allow-lists.

The optimal update complexity property, not only improves update performance but also enables setting the time between accumulator updates based only on revocation, not on issuance – e.g., twice per day or once per week, as with certificates. With universal accumulators, newly issued credentials are not presentable until the next published update; with an optimal accumulator, membership in the accumulator is immediately provable by Holders who have been issued a valid witness.

Examples of positive optimal accumulators described in Sections 4.2 and 4.3 are based on RSA [16,46,4] and on Elliptic Curves [50,63,40,39].

Mode of operation. As noted in [40], a Status Manager may choose to publish update messages $\texttt{upmsg}$ for Holders to update their witness at each accumulator update, or re-initialize the accumulator value and re-compute and re-distribute each witness to each Holder. We refer to the former as *update-message* mode, and to the latter as *reset* mode. Reset mode is more computationally expensive for the Issuer but less so for Holders. Additionally, update-message mode reveals how many statuses were changed at each update and requires public availability of these lists for each Holder to download, while reset mode does not.

4.2 RSA accumulator

RSA accumulators were first proposed in [8] and later in [5]. Authors in [16] formalize the concept of accumulator and propose a dynamic and positive accumulator that supports zero knowledge proofs, making it applicable to the revocation of anonymous credentials. A universal accumulator based on RSA is described in [46], and in [4, Section 4.1] the authors propose an accumulator with optimal update cost which does not require the accumulator value to be updated on add operations. We focus on the latter, referred to as RSA-B, and defined by the following algorithms:

Status Manager algorithms. The Status Manager samples $p = 2p' + 1, q = 2q' + 1$ two safe primes of $\lambda/2$ bits, $n = pq$, g a generator for the group of quadratic residues $QR(n)$, which is also the initial value of the accumulator α_0. The set D of elements that can be accumulated consists of all odd primes less than $p'q'$.

$$\begin{aligned}
\texttt{Gen}(1^\lambda): &\quad p, q \xleftarrow{\$} \text{safe primes}; \quad n \leftarrow pq; \quad \texttt{sk} \leftarrow p'q'; \quad \texttt{params} \leftarrow [n, g]; && (1)\\
&\quad \alpha_0 \leftarrow g; \quad D = \text{primes} &&\\
\texttt{GenWit}(\texttt{sk}, \alpha_t, x): &\quad w_{x|t} \leftarrow \alpha_t^{(x)^{-1} \mod \texttt{sk}} && (2)\\
\texttt{Del}(\texttt{sk}, \alpha_t, x): &\quad \alpha_{t+1} \leftarrow \alpha_t^{(x)^{-1} \mod \texttt{sk}} \mod n; \quad \delta \leftarrow x && (3)
\end{aligned}$$

Adding a value requires only to compute the associated witness executing $\texttt{GenWit}$. When an element is deleted, the algorithm outputs the update material $\texttt{upmsg}$ composed by the new accumulator α_{t+1} and the value δ that a Holder will use to update its witness.

Holder algorithms. The Holder who is given a witness $w_{x|t}$ for the value x accumulated in α_t, can verify its membership by executing $\texttt{VerMem}$.

$$\begin{aligned}
\texttt{VerMem}(\alpha_t, x, w_{x|t}): &\quad \alpha_t \stackrel{?}{=} w_{x|t}^x \mod n && (4)\\
\texttt{WitUp\_}(\alpha_{t+1}, x, w_{x|t}, \delta): &\quad w_{x|t+1} \leftarrow w_{x|t}^c \cdot \alpha_{t+1}^b \mod n, \quad \text{where } \{b, c \mid bx + c\delta = 1\} && (5)
\end{aligned}$$

When some element is deleted from $\mathcal{S}$, the accumulator is updated and a Holder must update its witness executing $\texttt{WitUp\_}$. The algorithms $\texttt{GenZKP}_\in$ and $\texttt{VerZKP}_\in$, which are also supported by the scheme, are recalled in Section 5.3.

Batching Updates. As noted in [16], it is possible to batch multiple updates publishing the tuple $\texttt{upmsg} = (\alpha^{-\Delta}, \Delta)$; where $\Delta = \prod_{i=1}^k \delta_i$ is the product of all the removed elements since the previous update. This benefits both the Status Manager, which does not need to compute all the intermediate values of the accumulator, and the Holder, who does not need to download them.

4.3 Elliptic curve accumulator

Originally proposed in [50] as a dynamic accumulator, in [23] is presented as a positive accumulator with zero knowledge proofs. This accumulator was subsequently extended with proofs of non-membership in zero knowledge in [3]. Efficient batch operations and the use of Type-III pairings have been introduced in [63], and the algorithms to accumulate values with optimal update cost are proposed in [40]. A combination of Type-III with batch operations and optimal update cost appears in [39, Section 3]. The latter is defined by the following algorithms:

Status Manager algorithms. The public parameters of the scheme are given by a bilinear pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, and g_1, g_2 are respectively generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.

$$\begin{aligned}
\texttt{Gen}(1^\lambda):\quad & \texttt{sk} \xleftarrow{\$} \mathbb{Z}_p^*; && (6)\\
& \texttt{params} \leftarrow \left[(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathbf{e}), j \leftarrow g_2^{\texttt{sk}}\right] \quad p \text{ prime}, |p| = \lambda; &&\\
& u_0 \xleftarrow{\$} \mathbb{Z}_p^*; \quad \alpha_0 \leftarrow g_1^{u_0}; \quad D \leftarrow \mathbb{Z}_p^* \setminus \{-\texttt{sk}\} &&\\
\texttt{GenWit}(\texttt{sk}, \alpha_t, x):\quad & w_{x|t} \leftarrow \alpha_t^{(x+\texttt{sk})^{-1}} && (7)\\
\texttt{Del}(\texttt{sk}, \alpha_t, x):\quad & \alpha_{t+1} \leftarrow \alpha_t^{(x+\texttt{sk})^{-1}}; \quad \delta \leftarrow x && (8)\\
\texttt{VerMemMan}(\texttt{sk}, \alpha_t, x, w_{x|t}):\quad & \alpha_t \overset{?}{=} w_{x|t}^{x+\texttt{sk}} && (9)
\end{aligned}$$

Notably this scheme allows a Status Manager to run the VerMemMan algorithm which allows him to verify more efficiently (compared to VerMem) the membership of a value in the accumulated set.

Holder algorithms. The algorithms to update the witness upon deletion of an accumulated value, and to verify the membership of an element in the accumulator are the following.

$$\begin{aligned}
\texttt{VerMem}(\alpha_t, x, w_{x|t}):\quad & \mathbf{e}(\alpha_t, g_2) \overset{?}{=} \mathbf{e}(w_{x|t}, g_2^x j) && (10)\\
\texttt{WitUp\_}(\alpha_{t+1}, x, w_{x|t}, \delta):\quad & w_{x|t+1} \leftarrow (w_{x|t} \cdot \alpha_{t+1}^{-1})^{(\delta - x)^{-1}} && (11)
\end{aligned}$$

The algorithms GenZKP$_\in$ and VerZKP$_\in$, which are also supported by this scheme, are recalled in Section 5.3.

Batching Updates. In this case multiplying all the elements does not work. Given the update material at time i to be $\texttt{upmsg}_i = (\alpha_i, \delta_i)$, without loss of generality we assume $t = 1$ to be the last value of the accumulator to which Holder has updated; applying WitUp_ sequentially m-times we obtain the witness at time $m + 1$:

$$w_{x|m+1} = ((w_{x|1} \cdot \alpha_2^{-1})^{(\delta_1 - x)^{-1}} \cdots \alpha_{m+1}^{-1})^{(\delta_m - x)^{-1}} = w_{x|1}^{\prod_{j=1}^m (\delta_j - x)^{-1}} \prod_{i=1}^m \alpha_{i+1}^{-\prod_{j=i}^m (\delta_j - x)^{-1}} \qquad (12)$$

The problem above takes the name of Multi-Scalar Exponentiation, which is efficiently solvable adopting Pippenger's algorithm [52], or some of its more recent iterations. The key point is that the larger the upmsg, the more efficiently we can solve that product.

4.4 Security notions and an advanced construction

Security notions for accumulators. There are two main security notions associated to positive dynamic accumulators, namely the *adaptive soundness* and *non-adaptive soundness.* Informally, an accumulator scheme is *adaptively sound* if no polynomial time adversary $\mathcal{A}$ can win the following experiment. $\mathcal{A}$ takes in input an accumulator value α, and can perform a training where it sends a polynomial number of queries to the challenger to add or delete elements of its choice, choosing them adaptively. Let α be the value of the accumulator after the training: we say that $\mathcal{A}$ wins the experiment if can generate a pair (x, w) such that $\texttt{VerMem}(\alpha, x, w) = 1$ and x is not accumulated. An accumulator is *non-adaptively sound* if the adversary chooses in advance the set of elements $\mathcal{S}$ to add and remove in its queries, before the publication of the accumulator α. Then the addition and deletions performed during the training by the adversary are chosen adaptively but must come from $\mathcal{S}$. These security notions can be adapted to apply to additive accumulators, giving the adversary only the ability to perform queries to add elements. For a formal security definition of adaptive and non-adaptive soundness see [4, Section 2.2].

Notably, both the RSA and the EC accumulator presented in Sections 4.2 and 4.3 only satisfy the security notion of non-adaptive soundness. We now recall a construction [4] that allows one to achieve adaptive security; we will also take advantage of this in Section 5.2.

Combining accumulators. As formalized in [4, Section 3.2], it is possible to combine positive dynamic non-adaptively sound accumulators $\mathsf{ACC_{NA}}$, as the ones we described in Sections 4.2 and 4.3 with positive additive and adaptively sound accumulators $\mathsf{ACC_A}$, to obtain an adaptively sound dynamic accumulator $\mathsf{ACC_H}$. This construction is referred to as Construction H in [4], so we continue to refer to it with the same name.

We will consider a simplified version of Construction H where the role of additive adaptively sound accumulator $\mathsf{ACC_A}$ is covered by a digital signature scheme DS[6].

Definition 1 (Construction H of [4]). *Given* DS *a digital signature scheme and* $\mathsf{ACC_{NA}}$ *a non-adaptively sound dynamic accumulator for elements in D, we define the following dynamic and adaptively sound accumulator for elements in a set $\mathfrak{D}$: to add an element $X \in \mathfrak{D}$ in* $\mathsf{ACC_H}$*, the manager samples uniformly at random an element $x \xleftarrow{\$} D$, adds x to* $\mathsf{ACC_{NA}}$ *and signs (X, x). To remove X from* $\mathsf{ACC_H}$*, the manager removes x from* $\mathsf{ACC_{NA}}$*. A witness for X in* $\mathsf{ACC_H}$ *is given by the witness w_x for x accumulated in* $\mathsf{ACC_{NA}}$ *and the signature σ of (X, x). To prove membership of an element X in* $\mathsf{ACC_H}$*, the prover can reveal a pair (X, x), the witnesses for x in* $\mathsf{ACC_{NA}}$ *and the signature σ.*

For a more formal definition of the algorithms of the accumulator derived from Construction H see [4, Figures 5,6 and 7].

Note that the set $\mathfrak{D}$ can be defined as the set of all messages that can be signed using DS, or the set of messages with a specific structure. In the following sections $\mathfrak{D}$ will be the set of credentials issued to users.

We start by describing a popular application of Construction H. Non adaptively sound accumulators with optimal update, and supporting ZKP, have become the primitives to use to manage

[6] Note that an UF-CMA secure digital signature scheme can be seen as a positive, *additive* and adaptively sound accumulator for the set of messages. When a message x is accumulated, the manager signs the x and the signature σ is the witness for it. To verify that x is accumulated, the prover shows the signature σ that the verifier can check under the public key of the manager.

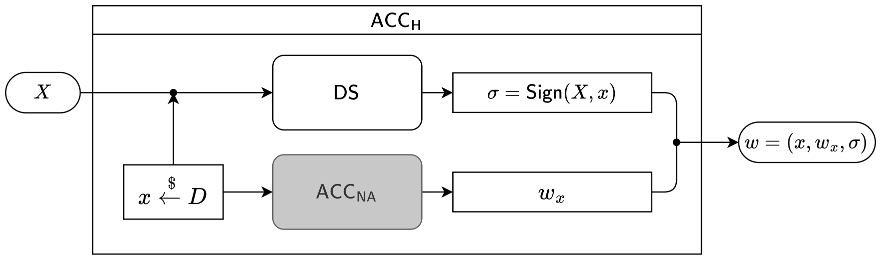


Fig. 1. An accumulator generated according to Construction H (Definition 1). Given $\mathsf{ACC_{NA}}$ a non-adaptively sound dynamic accumulator, and DS an additive and adaptively sound accumulator instantiated using a digital signature scheme, $\mathsf{ACC_H}$ is an adaptively sound, dynamic accumulator.

the revocation of anonymous credentials. This is thanks to their efficiency, the ability to be turned in adaptively sound accumulator for generic elements via Construction H, and the ability prove knowledge of an accumulated element x and a witness w_x in zero knowledge.

Revocation of Anonymous Credentials. If we consider Construction H of Definition 1, it is possible to define an accumulator for anonymous credentials starting from the RSA or EC accumulators of Section 4.2 and 4.3 as the dynamic non-adaptively sound accumulator $\mathsf{ACC_{NA}}$, and the selective disclosure signatures [33] like BBS [62], CL [17], or PS [53], as digital signatures DS. This combination yields an adaptively sound accumulator $\mathsf{ACC_H}$ for anonymous credentials, which supports ZKP of an accumulated element and a witness for it. In fact, we observe that, as we have recalled in Sections 4.2 and 4.3, (1) the EC and RSA non-adaptively sound accumulators support ZKP to prove knowledge of an accumulated element and a witness for it, and (2) the selective disclosure signatures [62,17,53] used to generate anonymous credentials, support ZKP to prove knowledge of a signature and of the messages signed in it (X, x). Therefore, if the value x accumulated in $\mathsf{ACC_{NA}}$ is included as an attribute of the anonymous credential, it is possible to combine the two ZKPs. More specifically, a prover can prove *in zero knowledge* that it knows an anonymous credential (a signature of the issuer and the signed attributes using DS), and that one of the signed attributes (the value x that the prover never discloses) is accumulated in $\mathsf{ACC_{NA}}$. The proofs can be combined using techniques to prove equalities of discrete logarithms [14,59,37].

Note that construction H is an abstraction, which we leverage to discuss the soundness of the construction resulting from the combination of digital signature DS and non-adaptively secure accumulator $\mathsf{ACC_{NA}}$. In practice, DS is managed by the Credential Issuer and $\mathsf{ACC_{NA}}$ may be managed by a separate Status Manager; there is no need to introduce a single entity concretely managing both, but this role is frequently played by the Issuer.

5 Accumulators for Revocation of Non-Anonymous Credentials

What has never been explicitly discussed in the literature, to the best of our knowledge, is what are the implications of using accumulator schemes to manage the revocation of non-anonymous Credentials like SD-JWT and mDOC. This analysis is particularly relevant to scenarios in which verifiable Credential systems admit Issuers of both anonymous and non-anonymous Credentials. In this case, it might be desirable to have a single mechanism to manage the revocation of Credentials, and an accumulator scheme could play that role.

5.1 Naive revocation of non-anonymous credentials via accumulators

Non-anonymous Credentials like SD-JWT and mDOC can be seen, in their essence, as a set of attributes that are committed to using hash based commitments, and then signed by the Issuer[7]. If we want to equip such Credentials with a revocation mechanism based on the use of accumulators, which would allow the revocation of anonymous and non-anonymous Credentials, we can instantiate an accumulator with the following structure, still based on Construction H, Definition 1.

A naive approach. To manage the revocation of non-anonymous credentials the Status Manager generates a non-adaptively sound and dynamic accumulator scheme $\mathsf{ACC_{NA}}$ as the one used for anonymous Credentials, and a standard digital signature scheme DS, which is the signature scheme used by the Issuer of the non-anonymous Credentials. The Status Manager samples a random x for each Credential and adds it to $\mathsf{ACC_{NA}}$, then the Issuer signs $X||x$ generating σ, so that when the Holder presents its Credential, it can show to the Verifier the signature σ of $X||x$, and the witness used to prove that x is accumulated in $\mathsf{ACC_{NA}}$. To revoke a credential X, the Status Manager removes x from $\mathsf{ACC_{NA}}$ and publishes the update message δ. Since the digital signature DS does not support ZKP to prove knowledge of a signature, the values X and x must be revealed to let the verifier verify σ. This approach is a straightforward application of Construction H, as done for anonymous credentials, but it has drawbacks in terms of privacy that must be addressed.

Privacy risks. The value x, as well as the signature $\sigma_{(X,x)}$ over (X, x), are unique identifiers of the Credential, and thus a source of correlation between presentations. This problem is intrinsic to non-anonymous Credentials, which for this reason must be single-use Credentials and are issued in batches. Therefore, each Credential in the batch must contain different commitments to the same attributes, and a different element x accumulated in $\mathsf{ACC_{NA}}$ to prevent Verifiers from linking presentations of the same Holder (privacy risk 3.iii., Section 3).

Since each Credential is meant to be presented only once, it might seem safe to reveal all the unique identifiers $X, x, \sigma_{(X,x)}, w_x$ during presentations. However, even if a Holder reveals x to a Verifier only once, upon revocation of the same Credential, $\delta \leftarrow x$ will be published by a Status Manager in an `upmsg` for $\mathsf{ACC_{NA}}$ (see the deletion algorithms 8 and 3). This message would allow the Verifier to know that the Credential it was presented has been revoked, allowing it to monitor the Credential status (privacy risk 3.ii., Section 3).

Apart from the privacy risks, note that every time the accumulator is updated the Holder must update all the witnesses associated to its credentials, and to revoke a credential the Status Manager must remove all the elements x associated with each copy.

Although for non-anonymous Credential it is not possible to keep hidden the signature of the Issuer on (X, x) by means of ZKP, it is still possible to hide the signed value x using commitment schemes that support ZKP in combination with the ZKP to prove knowledge of an accumulated element and the associated witness. We describe this approach in the next section.

5.2 Privacy preserving revocation of non-anonymous credentials

Commitment schemes are a building block for the design of signature schemes for anonymous Credentials. They are essential for the design of the ZKP to prove knowledge of a Credential, while disclosing only a subset of the attributes included in it. We take advantage of the properties of

[7] For now we can omit the public key of the Credential used to have the device binding.

commitments and the protocols used to prove knowledge of the committed value in ZK, to design a mechanism that allows a privacy preserving revocation of non-anonymous Credentials.

By privacy preserving revocation we mean that the Verifiers do not learn the value accumulated in $\mathsf{ACC}_{\mathsf{NA}}$, providing a solution to the problem of Verifiers monitoring the status of Credentials they are presented (privacy risk 3.ii.). The other advantage of the construction we present, compared to the naive construction described above, is that the Credentials issued in batches can be associated to *the same* x accumulated in $\mathsf{ACC}_{\mathsf{NA}}$, simplifying the revocation of all the copies of the same Credential for the Status Manager, and the update of the witness for the Holder, while keeping the presentations unlinkable (privacy risk 3.iii.).

The mechanism for privacy preserving revocation of non-anonymous Credentials is based on the use of a non-adaptively sound accumulator $\mathsf{ACC}_{\mathsf{NA}}$ to accumulate the randomness x, and, instead of signing x, together with the Credential X (as Construction H instructs), the Status Manager computes a commitment $\mathfrak{C}_x$ to x using a commitment scheme Comm *that supports ZKP*, and signs $X||\mathfrak{C}_x$ with the signature scheme DS. When the Holder presents its Credential, it proves that the value committed in $\mathfrak{C}_x$ is accumulated in $\mathsf{ACC}_{\mathsf{NA}}$ combining the ZKP for Comm with the ZKP for $\mathsf{ACC}_{\mathsf{NA}}$. The ZKPs will be presented in Section 5.3, now we focus on the Status Manager algorithms and the Holder algorithms to verify the membership of a credential and to update the witness.

Let $\mathsf{ACC}_{\mathsf{NA}} = (\mathtt{Gen}_{\mathsf{NA}}, \mathtt{GenWit}_{\mathsf{NA}}, \mathtt{Del}_{\mathsf{NA}}, \mathtt{VerMem}_{\mathsf{NA}}, \mathtt{WitUp}_{\_,\mathsf{NA}})$ be a non-adaptively sound accumulator for elements in D, Comm a commitment scheme supporting ZKP[8], $\mathsf{DS} = (\mathtt{Gen}_{\mathsf{DS}}, \mathtt{Sign}, \mathtt{Verify})$ a digital signature, and λ the security parameter, we define the accumulator accumulator $\mathsf{ACC}_{\mathsf{H}} = (\mathtt{Gen}_{\mathsf{H}}, \mathtt{GenWit}_{\mathsf{H}}, \mathtt{Del}_{\mathsf{H}}, \mathtt{VerMem}_{\mathsf{H}}, \mathtt{WitUp}_{\_,\mathsf{H}})$ for elements in the domain $\mathfrak{D}$ as follows.

Status Manager algorithms.

5.i. $\mathtt{Gen}_{\mathsf{H}}(1^\lambda)$:
  (a) $\mathtt{Gen}_{\mathsf{NA}}(1^\lambda)$: initialization of a non-adaptively sound accumulator, Equations (1) and (6);
  (b) $\mathtt{Gen}_{\mathsf{Comm}}(1^\lambda)$: initialization of a commitment scheme;
  (c) $\mathtt{Gen}_{\mathsf{DS}}(1^\lambda)$: initialization of a digital signature scheme which outputs the signature public parameters and the key pair $(\mathsf{sk}_{\mathsf{DS}}, \mathsf{pk}_{\mathsf{DS}})$.

  The initial value of the accumulator $\mathsf{ACC}_{\mathsf{H}}$ is α, and the public parameters are the output of the three algorithms above. The signing key $\mathsf{sk}_{\mathsf{DS}}$, and the manager key $\mathsf{sk}_{\mathsf{NA}}$ of $\mathsf{ACC}_{\mathsf{NA}}$ are part of the manager key sk_{H} of $\mathsf{ACC}_{\mathsf{H}}$.

5.ii. $\mathtt{GenWit}(\mathsf{sk}_{\mathsf{H}}, \alpha_t, X)$:
  (a) $x \overset{\$}{\leftarrow} D$: samples uniformly one element x;
  (b) $(\mathfrak{C}_x, \mathsf{Open}) \leftarrow \mathsf{Comm}(x)$: computes a commitment with Open being the opening information;
  (c) $w_{x|t} \leftarrow \mathtt{GenWit}_{\mathsf{NA}}(\mathsf{sk}_{\mathsf{NA}}, \alpha_t, x)$: includes x in $\mathsf{ACC}_{\mathsf{NA}}$ generating a witness $w_{x|t}$, Equations (2) and (7);
  (d) $\sigma \leftarrow \mathtt{Sign}(\mathsf{sk}_{\mathsf{DS}}, X||\mathfrak{C}_x)$: signs $X||\mathfrak{C}_x$;
  (e) $\mathtt{return}$ the witness $w_{X|t} = (x, (\mathfrak{C}_x, \mathsf{Open}), w_{x|t}, \sigma)$ to Holder.

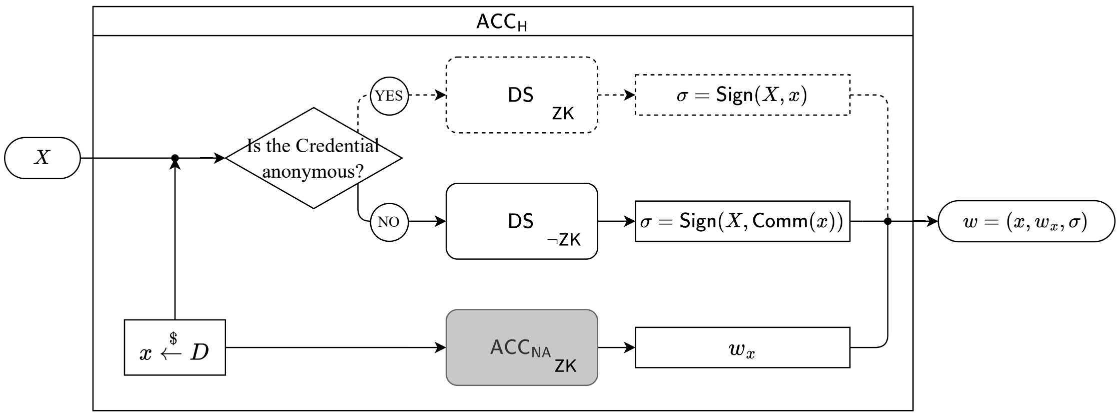
5.iii. $\mathtt{Del}(\mathsf{sk}_{\mathsf{H}}, \alpha_t, X)$:
  (a) $\mathtt{Del}_{\mathsf{NA}}(\mathsf{sk}_{\mathsf{NA}}, \alpha_t, x)$: removes x from $\mathsf{ACC}_{\mathsf{NA}}$, Equations (3) and (8);
  (b) $\mathtt{return}$ the update message δ.

[8] The commitment is never opened, so we do not define the opening algorithm.

Holder algorithms. Given the accumulator value α_t, the Credential X, the witness $w_{X|t}$ and the update message δ as described above, the Holder can run the following algorithms:

5.iv. $\texttt{VerMem}(\alpha_t, X, w_{X|t})$:
   (a) $\texttt{Verify}(\sigma, (X||\mathfrak{C}_x), \mathsf{pk}_{\mathsf{DS}})$: the Holder verifies the signature on $(X||\mathfrak{C}_x)$[9];
   (b) $\texttt{VerMem}_{\mathsf{NA}}(\alpha_t, x, w_{x|t})$: checks that the value x has not been removed from $\mathsf{ACC}_{\mathsf{NA}}$, Equations (4) and (10);
   (c) $\texttt{return}$ the status of X.

5.v. $\texttt{WitUp\_}(\alpha_{t+1}, X, w_{X|t}, \delta)$:
   (a) $\texttt{WitUp}_{\texttt{\_},\mathsf{NA}}(\alpha_{t+1}, x, w_{x|t}, \delta)$: updates the witness $w_{x|t}$, Equations (5) and (11);
   (b) $\texttt{return}$ the updated witness $w_{X|t+1} = (x, (\mathfrak{C}_x, \mathsf{Open}), w_{x|t+1}, \sigma)$ where Open is defined as in Item 5.ii.b.



$\mathsf{ACC_H}$
X
Is the Credential anonymous?
YES
NO
DS
ZK
¬ZK
$\sigma = \mathsf{Sign}(X, x)$
$\sigma = \mathsf{Sign}(X, \mathsf{Comm}(x))$
$w = (x, w_x, \sigma)$
$x \overset{\$}{\leftarrow} D$
$\mathsf{ACC_{NA}}$
w_x

Fig. 2. This figure represents an accumulator generated according to Construction H (Definition 1) that supports revocation of anonymous and non-anonymous Credentials. In the figure are depicted the operations to carry out in the witness generation for a newly added element X. We write $\mathsf{ACC_H}$ to refer to an adaptively secure and dynamic accumulator, $\mathsf{ACC_{NA}}$ to refer to a non-adaptively secure and dynamic accumulator, and DS to refer to an adaptively secure and additive accumulator instantiated with a digital signature scheme. We write ZK if the accumulator (the signature) supports ZKP to prove knowledge of an accumulated element (signed element) and of the associated witness (the signature). When the Credential is not anonymous, for simplicity we have omitted that the witness w actually is $(x, (\mathfrak{C}_x, \mathsf{Open}), w_{x|t}, \sigma)$, where $(\mathfrak{C}_x, \mathsf{Open}) \leftarrow \mathsf{Comm}(x)$.

Viable options for the commitment scheme are Pedersen [51] for elliptic curves or Damgård-Fujisaki [22] for RSA groups; hash-based commitments are not an option for this application.

Features of our construction. Our construction allows a Holder to keep the accumulated value x hidden, and therefore the presentations unlinkable with respect to the revocation update messages (privacy risk 3.ii.). It also allows the Status Manager of a system where anonymous and non-anonymous Credentials coexist to use the same non-adaptively sound accumulator $\mathsf{ACC_{NA}}$, as shown in Figure 2.

[9] This operation can be performed only once, since the revocation of a credential is performed by removing x from $\mathsf{ACC_{NA}}$.

Also, since non-anonymous Credentials must be issued in batches, given their single-use nature, the Status Manager must generate different commitments to the attributes included in the Credentials, and therefore different signatures, to make the presentation of different copies of Credential unlinkable (addressing privacy risk 3.iii.). Since the value x is hidden by the commitment $\mathfrak{C}_x$, *all the copies of Credential in a batch can refer to the same accumulated value* x as long as x is hidden in each copy using different commitments. This approach simplifies the revocation of all the Credentials in a batch, because removing x from the non-adaptively sound accumulator simultaneously revokes all copies of a Credential.

5.3 Zero Knowledge Proofs of Membership for Our Construction

In this section we describe the ZKP protocols that allow a Holder to prove knowledge of an element-witness pair (x, w) such that x is accumulated in $\mathsf{ACC_{NA}}$ (and w is its witness), and being $\mathfrak{C}_x$ the commitment in the Credential X, $\mathfrak{C}_x$ is a commitment to x (i.e. the Credential X has not been revoked). We do that for both the RSA and EC accumulators described in Sections 4.2 and 4.3. Building upon the standard proof of knowledge of an accumulated element and its witness – first introduced in [16] and revisited in [56] [10] for RSA, and in [12] for EC – we add in fraktur font the relations that must be added in order to bind the ZKP to X, proving that the accumulated element x is the element committed in $\mathfrak{C}_x$. For convenience, we adopt the Camenisch-Stadler notation [18][11].

RSA accumulator. The Status Manager runs $\texttt{SetupZKP}_\in$ generating some additional elements $g, h \in QR(n)$ and $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$[12], where $\log_g(h)$ and $\log_\mathfrak{g}(\mathfrak{h})$ must not be known to the Holder to ensure binding. Let $\mathfrak{C}_x = \mathfrak{g}^x \mathfrak{h}^l$ be the Damgård-Fujisaki commitment to x included in the Credential X. The Holder can generate a proof attesting the status of its Credential X running the following algorithm:

5.vi. $\texttt{GenZKP}_\in$: given as input the public parameters, the pair (x, w) and opening material l;
   (a) compute three random values $r_x, r_w, r_1 \xleftarrow{\$} \mathbb{Z}_{\lfloor n/4 \rfloor}$ for the auxiliary Damgård-Fujisaki commitments [22] $C_x = g^x h^{r_x}, C_w = w h^{r_w}, C_r = h^{r_w} g^{r_1}$;
   (b) prove knowledge of the pair (x, w) computing a proof:

$$\pi' \in PK\Bigg\{(x, l, r_x, r_w, r_1, xr_w, xr_1) : \mathfrak{C}_x = \mathfrak{g}^x \mathfrak{h}^l \wedge C_x = g^x h^{r_x} \wedge \\ \wedge C_r = h^{r_w} g^{r_1} \wedge \alpha = C_w^x \left(\frac{1}{h}\right)^{xr_w} \wedge 1 = C_r^x \left(\frac{1}{h}\right)^{xr_w} \left(\frac{1}{g}\right)^{xr_1}\Bigg\} \tag{13}$$

   (c) output the proof $\pi = (C_x, C_w, C_r, \pi')$, which – combined with the Credential X, which is signed together with the commitment $\mathfrak{C}_x$ – allows to check its status.

Finally, a Verifier can check the status of the Credential X as follows:

5.vii. $\texttt{VerZKP}_\in$: given the proof π, the pair $(X, \mathfrak{C}_x)$ and its signature σ, the public parameters and value of the accumulator $\mathsf{ACC_H}$, a Verifier checks that:
   (a) the signature σ is a valid signature of $(X, \mathfrak{C}_x)$;
   (b) π is a valid ZKP that the value committed in $\mathfrak{C}_x$ is still accumulated in $\mathsf{ACC_{NA}}$.

[10] We add few additional corrections to the generation of C_r.

[11] To clarify the protocol, we abuse the usual notation to include the values that are used by a honest Holder to generate the proof.

[12] A viable choice of $\mathfrak{G}$ can be $\mathfrak{G} = QR(n)$, and it is possible to set $\mathfrak{g} \leftarrow g, \mathfrak{h} \leftarrow h$ and then $C_x \leftarrow \mathfrak{C}_x$ included in the Credential.

EC accumulator. The Status Manager runs $\texttt{SetupZKP}_{\in}$ generating some additional elements $g, h, k, z \in \mathbb{G}_1$ and $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$. Let $\mathfrak{C}_x = \mathfrak{g}^x \mathfrak{h}^t$ be the Pedersen commitment included in the Credential X, and $E_{z,j} = \mathbf{e}(z, j), E_{z,g_2} = \mathbf{e}(z, g_2)$, public values in $\mathbb{G}_T$. The Holder can generate a proof attesting the status of X running the following algorithm:

5.viii. $\texttt{GenZKP}_{\in}$: given as input the public parameters, the pair (x, w) and opening material l;

(a) compute two random values $\sigma, \rho \xleftarrow{\$} \mathbb{Z}_p$ for auxiliary commitments $C_w = wz^{\sigma+\rho}, C_\sigma = h^\sigma, C_\rho = k^\rho$;

(b) prove knowledge of the pair (x, w) computing a proof:

$$\pi' \in PK\Big\{(x, l, \sigma, \rho, x\sigma, x\rho) : \mathfrak{C}_x = \mathfrak{g}^x \mathfrak{h}^l \wedge C_\sigma = h^\sigma \wedge C_\rho = k^\rho \wedge \\ \wedge 1 = C_\sigma^x \left(\frac{1}{h}\right)^{x\sigma} \wedge 1 = C_\rho^x \left(\frac{1}{k}\right)^{x\rho} \wedge \frac{\mathbf{e}(\alpha, g_2)}{\mathbf{e}(C_w, j)} = \mathbf{e}(C_w, g_2)^x \left(\frac{1}{E_{z,g_2}}\right)^{x\sigma + x\rho} \left(\frac{1}{E_{z,j}}\right)^{\sigma+\rho}\Big\} \tag{14}$$

(c) output the proof $\pi = (C_w, C_\sigma, C_\rho, \pi')$, which – combined with the Credential X, which is signed together with the commitment $\mathfrak{C}_x$ – allows to check its status.

$\texttt{VerZKP}_{\in}$ is done as in Item 5.vii., adopting the corresponding ZKP for the relations in Equation 14 and the commitment scheme Comm.

Note that the pairings in Equation 14 are values of $\mathbb{G}_T$ known by Holder and Verifier; the values $g_2, z, j = g_2^{\mathsf{sk}_{\mathsf{NA}}}$ are public parameters, hence $E_{z,g_2}, E_{z,j} \in \mathbb{G}_T$ can be precomputed.

6 Experimental Evaluation

The performance of some accumulator schemes – RSA, prime- and binary-field elliptic curves, and Bloom filter – has been measured in [43]. However, the elliptic curve schemes considered did not include pairing-based schemes, and the work could not have taken into account any of the optimizations published in [4,63,40,39]. In the following section we compare the two Accumulator schemes introduced in Section 4 and CRL based mechanisms in terms of Runtime and Size. About the evaluation of the accumulators, we evaluate both the operations for the use of accumulators with the naive approach described in Section 5.1, which does not involve the use of ZKP, and the operations to be executed using the construction described in Section 5.2, which involves the creation of ZKP.

6.1 Experimental Set-Up

Security level. The EC accumulator is based on the widely used BLS12-381 curve, which achieves a security level of 128 bits with an embedding degree of 12 over a 381-bit prime field [57]. To ensure a fair comparison, we instantiate the RSA accumulator with a 3072-bit modulus granting the same level of security.

Libraries. We use `allosaurust`[13], which implements both the universal [63] and positive [39] EC accumulators. For RSA, we provide our own implementation based on `unknown_order`[14]. It supports some of the most popular pure rust multiple-precision libraries, as well as the more mature `GMP`[15]

[13] https://github.com/sam-jaques/allosaurust/tree/master
[14] https://crates.io/crates/unknown_order
[15] https://gmplib.org/

and `OpenSSL`[16]. We prioritize performance adopting the latter as back-end. For CRLs we adopt `rustls-webpki`[17]. To allow a repeatable experiment we make available the repository containing our RSA accumulator implementation and benchmarks `accumulators-bench`[18]; the code is mainly written in rust with the exception of a few visualization options in python. In particular, our code does not include the CRL tests; we made use of the ones already available in the library.

Processor. All data were gathered using a Dell Latitude 7450, Intel Core Ultra 7 165H, 32GB RAM.

6.2 Runtime

Before delving into the runtime comparison, we first introduce the CRL operations and explain how to compare them with the operations described in Section 4.

6.i. `GenCRL`: Generate a new CRL from a list of certificates. Since this must be run each time we want to update the status of the credentials, we can compare it with `Del`.

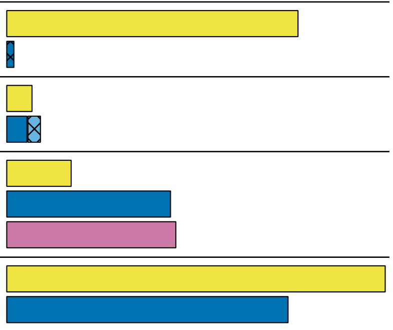
6.ii. `Parsing`: Given a serialized CRL, this process loads the list into memory, either by directly de-serializing the structure or by organizing it into a Binary Search Tree (BST). The difference can be summarized as dynamic fetching against caching. We model a Holder, who will likely search a CRL only occasionally, to run the first type of parsing; and a Verifier, who requires more efficient lookups, for the second.

6.iii. `Search`: Based on how a CRL has been parsed it is possible to run two types of searches; note that we specifically measure the lookup time for credentials that are not revoked, i.e., elements not in the list:
   (a) `Linear`: This operation search for a specific serial number and measures the time required to scan the entire list. Since the goal of this operation is for an Holder to asses the status of its credential, we compare it to `VerMem`.

[16] https://www.openssl.org/docs/man1.0.2/man3/bn.html
[17] https://docs.rs/rustls-webpki/latest/webpki/
[18] https://anonymous.4open.science/r/accumulators-bench-2C08/README.md

Table 1. Comparison restricted to the operations run by a Status Manager. Additional Cost represents the impact of a single revocation on the operation at hand, or in the case of `GenWit` a single addition; Cost represents the expected execution time on a daily basis ($d = 548, a = 19178$) and a CRL of size 30MB; `Gen` and `SetupZKP` are invoked once in the lifetime of the accumulator. We use a lighter color to show the overhead introduced by the construction presented in Section 5.3. The bar plots are presented in a logarithmical scale.

Manager	Additional Cost [ms]	Cost [ms] 🟨 RSA 🟦 EC 🟪 CRL
`Gen`	0	6.25×10^3
	0	2.39×10^{-1}
`SetupZKP`	0	1.135
	0	0.833
`Del`	1.08×10^{-2}	5.911
	2.46×10^{-1}	1.35×10^2
`GenCRL`	1.99×10^{-4}	1.59×10^2
`GenWit`	4.3	8.24×10^4
	2.42×10^{-1}	4.64×10^3



(b) `BST`: When the data is stored as a BST, it is possible to do serial number lookups without scanning the whole list. Since this is the operation a Verifier is most likely to use for status checks, we compare it to $\texttt{VerZKP}_{\in}$.

We recognize that organizing the data in a BST is beyond the scope of the standard specification. However, since it is common practice to use BSTs or other advanced data structures and the implementation we are testing supports it, we aim to provide a more practical perspective showing the performance enabled by this option.

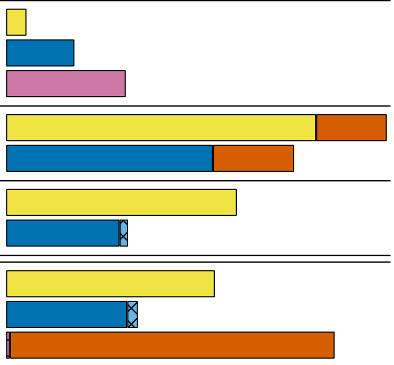
Table 1 displays the expected time for the operations performed by a Status Manager. In particular, RSA offers better performance for the `Del` operation, since batching is possible; batch `Del` for EC is excluded due to the need for `upmsg` generation. In this specific daily scenario, CRL appears to be the most time-consuming method to update the status of the credentials, this does not hold for a bigger number of revocations, around $d > 7 \times 10^2$ and $d > 15 \times 10^3$ recomputing a CRL would be less expensive than, respectively, updating an EC or an RSA Accumulator.

`Gen` and `SetupZKP` are only run during the initialization phase of an accumulator. `GenWit` must be performed for each element added to $\mathcal{S}$ making it the primary computational cost for a Status Manager. `GenWit` cost and scaling might seem concerning, but observe that the RSA and EC accumulators can be viewed as a special use of RSA and BLS [13] digital signature algorithms, respectively. In short, generating a witness w_x for an element x can be compared to generating a new signature of the same message α under the public key x, making `GenWit` comparable to signing an OCSP response, both in purpose and computational cost.

Table 2 presents the run-time for the operations performed by Holder and Verifier. We can see how the construction proposed in Section 5.2 introduces only a small overhead, which does not even affect the RSA version since we are choosing $C_x = \mathfrak{C}_x$. In general, the EC Accumulator offers better performance for ZKP-related computations, and the cost of performing a `Linear Search` in a CRL is equivalent to running the EC operation $\texttt{VerZKP}_{\in}$.

Table 2. Comparison restricted to the operations run by Holder and Verifier, where the latter only computes the last operation. Additional Cost represents the impact of a single revocation on the operation at hand; Cost represents the expected execution time for a single call with the exception of $\texttt{WitUp}_{-}$ which is reported as a daily operation ($d = 548$); Overhead indicates, in the CRL case, the amount of time it takes to parse a 30MB into a Binary Search Tree (BST), meanwhile for accumulators is the overhead introduced by not batching the operations. We use a lighter color to show the overhead introduced by the construction presented in Section 5.3. The bar plots are presented in a logarithmical scale.

Holder Verifier	Additional Cost [ms]	Cost [ms] 🟨 RSA 🟦 EC 🟪 CRL 🟧 Overhead
`VerMem`	0	3.93 $\times 10^{-1}$
	0	2.18
`Linear Search`	1.20×10^{-3}	6.68
$\texttt{WitUp}_{-}$	3.69×10^{-1}	2.03 $\times 10^{2}$
	5.95×10^{-2}	3.34 $\times 10^{1}$
$\texttt{GenZKP}_{\in}$	0	5.09 $\times 10^{1}$
	0	5.9
$\texttt{VerZKP}_{\in}$	0	3.46 $\times 10^{1}$
	0	6.9
`BST Search`	3.27×10^{-11}	8.2 $\times 10^{-5}$



The heaviest computational burden falls on the Holder, particularly for the `WitUp`$_{-}$ operation; which is assumed to be performed in epochs, as updating a witness for every individual revocation would result in significant overhead, included only in the bar plots. We also adapted the `allosaurust` implementation to use Pippenger's algorithm, which significantly optimizes batch updates, otherwise it would take the same time as the single updates.

If executed daily, `WitUp`$_{-}$ can be compared to parsing a CRL into a BST. From this perspective, it shifts the computational cost from the Verifier to the Holder. However, the key difference is that while a CRL contains all necessary revocation information, a witness depends on its most recent update. This means that a Holder must request a new witness if their current one is too outdated. A practical threshold can be set to determine when an update would take too long to compute; approximately $d = 10^4$ for RSA and $d = 10^5$ for EC, beyond which updates would take several seconds to complete. In our scenario, this threshold corresponds to a Holder being offline for approximately 1 month for RSA or 6 months for EC.

6.3 Size

Table 3 summarizes the sizes of the private `sk` and public parameters `params` for the Status Manager, along with the accumulator value α, the accumulated value x, the witness w, the message update after d deletions, and the zero-knowledge proof `GenZKP`$_{\in}$. When computing a Non-Interactive ZKP, the Holder can either send the commitments, or their digest. We assume the latter, but include the space complexity of the commitments in brackets. This first method is often preferred in implementations, since it favors better error handling.

For the RSA implementation, we use 3072-bit modulus n. The hash function used to create the accumulated element produces a 32-byte digest. It's important to note that the responses in the RSA ZKP are integers, so their size can vary based on the challenge and the blinded elements. To keep things simple, we refer to multiples of the size of the digest.

The EC implementation uses the pairing-friendly curve BLS12-381. The scalars in $\mathbb{Z}_p$ are 32 bytes, while the points in $\mathbb{G}_1$ are 48 bytes. The sizes for $\mathbb{G}_2$ and $\mathbb{G}_T$ are respectively 2 and 12 times larger.

None of the elements listed scale with the size of the set $\mathcal{S}$. Additionally, both the computational time and the size of a zero-knowledge proof remain constant.

However, note that the size of `upmsg` increases with the number of deletions. A 3% annual revocation rate for CIE with a uniform distribution would result in about 16 000 revocations each month. In the RSA case, this amounts to roughly 512 kB of update data, which every Holder would need to download and process. For an EC accumulator with the same assumptions, the monthly `upmsg` would be approximately 1.3 MB – more than double the size of the RSA version.

Table 3. Size comparison between RSA and EC accumulators.

	RSA			EC				
	$\mathbb{Z}_n^*$	*digest*	bytes	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	bytes
`sk`	1		384	1				32
`params`	1		384			1		96
α	1		384		1			48
x		1	32	1				32
w	1		384		1			48
`upmsg`	1	d	$32d + 384$	d	d			$80d$
`GenZKP`$_{\in}$	4(+4)	13(−1)	1952	6(−1)	3(+4)		(+1)	336

For an `upmsg` to reach the typical size of a CRL (e.g. 30 MB) it would require an Holder to be offline for more than two years. As discussed in Section 6.2 we will reach an usability threshold much earlier, in a concrete scenario for EC a Status Manager will keep published at most the `upmsg` for the last 6 months, leading to a constant size of around 7.8 MB of public material to handle.

6.4 Experimental Assessment Summary

The RSA accumulator is computationally more expensive than the EC accumulator in the setup phase, in the Holder witness update, and ZKP, both generation and verification, while the EC accumulator is more expensive in the Holder membership verification. In terms of size, the EC accumulator is smaller (or equal) for all parameters with the important exception of `upmsg`.

Concerning the performance comparison between Accumulators and CRL based revocation methods, the key distinction stands in the fact that the former vary their performance based on the number of additions or deletions to $\mathcal{S}$ since the last update, whereas the latter depend on the total amount of revocations. However, it must be noted that Accumulators have a significantly higher overhead as the number of revocations increases, as shown under the Additional Cost column in Tables 1 and 2.

Mode of operation. If the size of `upmsg` to be published by the Status Manager and downloaded by every Holder is not a concern, the EC accumulator would be computationally advantageous in `upmsg` mode for all roles.

Batch issuance. Using a single element x per batch of credentials would be possible. Using a different value per credential in a batch would be highly inefficient; if the accumulator is used in `upmsg` mode, it would force every Holder to update all witnesses for unused credentials – as well as requiring publication of revoked elements, in the same way that a list of revoked serials would have to be published; in reset mode, it would force the Issuer to re-issue a witness for every credential in the batch at every update, which would be more computationally expensive than re-issuing the entire batch with fresh signatures and a short duration.

7 Comparison of accumulators with other revocation mechanisms

We compare accumulators with the two most well-known revocation mechanisms – CRL and OCSP, with and without stapling ($\measuredangle$). We consider accumulators without ZKP (Section 5.1) in `upmsg` and in reset mode (Section 4), in which the Status Manager generates new witnesses at each update; and with zero-knowledge proofs (A+ZKP, Section 5.2), in either of the previous modes. We do not here consider Status List and Status Assertions as they are still in draft stage.

Accumulators *may* address the availability issue (Item 2.i.) in the same manner as OCSP stapling, i.e., by allowing the Holder to provide the Verifier an Issuer-signed copy of the public accumulator value. None of the academic literature address this option explicitly.

Our findings from the point of view of scaling and privacy (Items 2.ii. and 2.iii.) are summarized in Table 4. For scalability we compare Verifier request and Holder update size scaling – how much data do the Verifier and Holder have to download to enable the revocation mechanism – and we distinguish between none (0), constant ($O(1)$), scaling with the number of revocations ($O(d)$).

Accumulators vs. CRL. Accumulators are better than, or equal to, CRL in every privacy respect as well as Verifier request size. CRL have the main advantage that the Holder need not ever update their local information. This is a very significant advantage, considering that Holder devices may be frequently offline, and wallet providers may not wish to deal with frequent complaints. Additionally, considering the EUDI ARF requirements (Section 3), support for list-based mechanisms is mandatory for qualified attestations, while other mechanisms appear to be excluded. Even changing the ARF would require supporting an additional mechanism, rather than an alternative one. This also weighs in favor of CRL or Status List [47].

A_{reset} vs. OCSP with stapling. Accumulators in reset mode amount to the Issuer providing the Holder with a new witness at every update period, and revoked elements are never published as `upmsg`. They are almost identical to OCSP, except that they do not allow Credential status monitoring. In this mode, without ZKP (Section 5.1), Accumulators have a slight privacy advantage.

$A_{\texttt{upmsg}}$ vs. OCSP with stapling. Status Manager publishing `upmsg` allows monitoring of their activity as well as the status of previously verified Credential. `upmsg` are also likely larger than OCSP responses. In this mode, without ZKP (Section 5.1), Accumulators are equal to or worse than OCSP w/ stapling in every respect.

$A_{(mode)}$ + ZKP vs. OCSP with stapling. With ZKP, Accumulators mitigate against every identified privacy threat, in particular presentation linking and Credential status monitoring, and have equal Holder update size. They are better than, or equal to, OCSP w/ stapling in every privacy and size Criterion, possibly except Verifier request size.

$A_{(mode)}$ + ZKP vs. $A_{(mode)}$. The accumulators using the ZKP described in Section 5.3, compared with those that do not use ZKP according to the naive construction described in Section 5.1, allow the manager to issue batches of copies of a Credential associated with the same element x accumulated in $\mathsf{ACC}_{\mathsf{NA}}$.

Lastly, note that none of the revocation mechanisms considered here address the dishonest Issuer weakness (Item 2.iv.).

8 Conclusion

Wallet revocation mechanisms are the topic of active discussion and research. As underlined by the difficulties encountered in revocation mechanisms for web PKI, revocation is a difficult goal to achieve - and privacy-preserving revocation doubly so.

We have shown how the latest cryptographic accumulators can be integrated with (non-anonymous) digital identity credentials to provide greater privacy guarantees than existing standards for revocation, when combined with appropriate zero-knowledge proofs. We also present a mechanism that uses existing techniques to manage the revocation of both anonymous and non-anonymous credentials by adopting a single dynamic non-adaptively sound accumulator. Lastly, we show that, even if accumulators require greater computational effort from all parties involved, particularly for the

Table 4. Comparison of revocation schemes, assuming batch issuance: exposure to privacy risks and communication cost scaling with number of revocations d.

Criterion	$\mathrm{A}_{\texttt{upmsg}}$	$\mathrm{A}_{\mathrm{reset}}$	$\mathrm{A}_{(mode)}$ + ZKP	CRL	OCSP	OCSP + $\measuredangle$
Item 3.i. Status Manager monitoring Holder	N	N	N	N	Y	N
Item 3.ii. Verifier monitoring Credential status	Y	N	N	Y	Y	Y
Item 3.iii. Verifier linking Holder presentations	N	N	N	N	N	N
Item 3.iv. Status Manager monitoring Verifier	N	N	N	N	Y	N
Item 3.v. Third Parties monitoring Status Manager	Y	N	(as *mode*)	Y	N	N
Verifier request size	O(1)	O(1)	O(1)	O(d)	O(1)	0
Holder update size	O(d)	O(1)	(as *mode*)	0	0	O(1)

creation and verification of ZKP, and for the management of the witnesses, the overhead introduced may be, in some contexts, absolutely practical.

In future works we plan to extend experimental evaluation to other platforms to provide a better comparison with mobile devices. We plan to extend the comparison of accumulators with Status List and Status Assertion, once out of draft stage.

Acknowledgments. This work has been partially supported by the joint laboratory between the Bruno Kessler Foundation (FBK) and the Italian Government Printing Office and Mint (IPZS), and by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. The first author acknowledges support from Eustema S.p.A. through the PhD scholarship.

Disclosure of Interests. The authors have no competing interests.

References

1. Aas, J.: Intent to end OCSP service (07 2024), https://letsencrypt.org/2024/07/23/replacing-ocsp-with-crls.html
2. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002. pp. 418–433. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
3. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA. pp. 295–308. Springer (2009). https://doi.org/10.1007/978-3-642-00862-7_20
4. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. In: EuroS&P. pp. 301–315. IEEE (2017). https://doi.org/10.1109/EUROSP.2017.13, http://ia.cr/2017/043
5. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT '97. pp. 480–494. Springer (1997). https://doi.org/10.1007/3-540-69053-0_33
6. Barthoulot, A., Blazy, O., Canard, S.: Cryptographic accumulators: New definitions, enhanced security, and delegatable proofs. In: AFRICACRYPT 2024. pp. 94–119 (2024). https://doi.org/10.1007/978-3-031-64381-1_5, https://ia.cr/2024/657
7. Baum, C., Blazy, O., Camenisch, J., Hoepman, J.H., Lee, E., Lehmann, A., Lysyanskaya, A., Mayrhofer, R., Montgomery, H., Nguyen, N.K., Preneel, B., Shelat, A., Slamanig, D., Tessaro, S., Thomsen, S.E., Troncoso, C.: Cryptographers' feedback on the eu digital identity's arf (2024), https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/issues/200
8. Benaloh, J., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT '93. pp. 274–285. Springer (1994). https://doi.org/10.1007/3-540-48285-7_24
9. Bećirović Ramić, v., Cogo, E., Prazina, I., Cogo, E., Turkanović, M., Turčinhodžić Mulahasanović, R., Mrdović, S.: Selective disclosure in digital credentials: A review. ICT Express (2024). https://doi.org/10.1016/j.icte.2024.05.011
10. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (07 1970). https://doi.org/10.1145/362686.362692
11. Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., Cooper, D.: Internet X.509 public key infrastructure certificate and certificate revocation list (crl) profile (05 2008), https://datatracker.ietf.org/doc/rfc5280
12. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. Cryptology ePrint Archive, Paper 2004/174 (2004), https://eprint.iacr.org/2004/174

13. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**, 297–319 (09 2004). https://doi.org/10.1007/s00145-004-0314-9
14. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023), https://toc.cryptobook.us/
15. Camacho, P., Hevia, A.: On the impossibility of batch update for cryptographic accumulators. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. pp. 178–188. Springer (2010). https://doi.org/10.1007/978-3-642-14712-8_11, https://ia.cr/2009/612
16. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 61–76. Springer (2002). https://doi.org/10.1007/3-540-45708-9_5
17. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3. pp. 268–289. Springer (2003)
18. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B.S. (ed.) CRYPTO '97. pp. 410–424. Springer, Berlin, Heidelberg (1997). https://doi.org/10.1007/BFb0052252
19. Chromium: CRLSets, https://www.chromium.org/Home/chromium-security/crlsets/
20. Colombo, A.: Scalable and Privacy-Preserving Revocation of Verifiable Credentials. Master's thesis, EPFL, School of Computer and Communication Sciences (09 2024)
21. Cramer, R.J.F.: Modular Design of Secure yet Practical Cryptographic Protocols. Ph.D. thesis, CWI and University of Amsterdam (1996)
22. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. pp. 125–142. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_8
23. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. IACR Cryptol. ePrint Arch. (2008), http://ia.cr/2008/538
24. De Marco, G., Steele, O., Marino, F.A., Adomeit, M.: OAuth Status Assertions. Internet-Draft draft-demarco-oauth-status-assertions-03, Internet Engineering Task Force (Dec 2024), https://datatracker.ietf.org/doc/draft-demarco-oauth-status-assertions/03/, work in Progress
25. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: Nyberg, K. (ed.) CT-RSA 2015. pp. 127–144. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_7
26. The European Digital Identity Wallet Architecture and Reference Framework, version 1.6.0 (03 2025), https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/
27. Dipartimento per la Trasformazione Digitale: CIEusers, https://innovazione.gov.it/notizie/comunicati-stampa/digitale-nuovi-massimi-storici-per-i-numeri-del-2022/
28. Eastlake 3rd, D.: Transport layer security (TLS) extensions: Extension definitions (01 2011), https://datatracker.ietf.org/doc/rfc6066
29. Proposal for a Regulation of the European Parliament and of the Council amending Regulation (EU) no 910/2014 as regards establishing a framework for a European Digital Identity (2021), https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2021:281:FIN
30. Fazio, N., Nicolosi, A.: Cryptographic accumulators: Definitions, constructions and applications. Paper written for course at New York University. Available at http://www-cs.ccny.cuny.edu/~fazio/research.html (2002), paper written for course at New York University.
31. Fett, D., Yasuda, K., Campbell, B.: Selective Disclosure for JWTs (SD-JWT). Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-12, Internet Engineering Task Force (Sep 2024), https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/12/, work in Progress
32. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
33. Flamini, A., Sciarretta, G., Scuro, M., Sharif, A., Tomasi, A., Ranise, S.: On cryptographic mechanisms for the selective disclosure of verifiable credentials. Journal of Information Security and Applications **83**, 103789 (2024). https://doi.org/10.1016/j.jisa.2024.103789

34. Forum, C.: Ballot SC-063 v4: Make OCSP optional, require CRLs, and incenctivize automation (07 2023), https://cabforum.org/2023/07/14/ballot-sc-063-v4-make-ocsp-optional-require-crls-and-incentivize-automation/
35. Hallam-Baker, P.: X.509v3 transport layer security (TLS) feature extension (10 2015), https://datatracker.ietf.org/doc/rfc7633
36. Anoncreds v2 (10 2024), https://github.com/hyperledger/anoncreds-v2-rs
37. Specification of the identity mixer cryptographic library version 2.3.0 (04 2010), https://dominoweb.draco.res.ibm.com/reports/rz3730_revised.pdf, Security Team, Computer Science Dept., IBM Research Zurich
38. ISO/IEC 18013-5 personal identification - ISO-compliant driving licence - part 5: Mobile driving licence (mDL) application (09 2021), https://www.iso.org/standard/69084.html
39. Jaques, S., Lodder, M., Montgomery, H.: ALLOSAUR: accumulator with low-latency oblivious sublinear anonymous credential updates with revocations. IACR Cryptol. ePrint Arch. p. 1362 (2022), https://ia.cr/2022/1362
40. Karantaidou, I., Baldimtsi, F.: Efficient constructions of pairing based accumulators. In: CSF. pp. 1–16. IEEE (2021). https://doi.org/10.1109/CSF51468.2021.00033, https://ia.cr/2021/638
41. Khovratovich, D., Lodder, M., Parra, C.: Anonymous credentials with type-3 revocation, version 0.6 (04 2022), https://github.com/hyperledger/ursa-docs/tree/main/specs/anoncreds1
42. Korzhitskii, N., Carlsson, N.: Revocation statuses on the internet. In: Hohlfeld, O., Lutu, A., Levin, D. (eds.) Passive and Active Measurement. pp. 175–191. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72582-2_11, https://arxiv.org/abs/2102.04288
43. Kumar, A., Lafourcade, P., Lauradoux, C.: Performances of cryptographic accumulators. In: 39th Annual IEEE Conference on Local Computer Networks. pp. 366–369 (2014). https://doi.org/10.1109/LCN.2014.6925793
44. Langley, A.: Revocation checking and Chrome's CRL (02 2012), https://www.imperialviolet.org/2012/02/05/crlsets.html
45. Larisch, J., Choffnes, D., Levin, D., Maggs, B.M., Mislove, Alan a nd Wilson, C.: CRLite: A scalable system for pushing all TLS revocations to all browsers. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 539–556 (2017). https://doi.org/10.1109/SP.2017.17
46. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 253–269. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_17
47. Looker, T., Bastian, P., Bormann, C.: Token Status List. Internet-Draft draft-ietf-oauth-status-list-08, Internet Engineering Task Force (Feb 2025), https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/08/, work in Progress
48. Ministero dell'Interno: CIE Certification Authority, https://www.cartaidentita.interno.gov.it/en/public-and-business-administration/certification-autority/
49. Mozilla: CA/revocation checking in Firefox (02 2024), https://wiki.mozilla.org/CA/Revocation_Checking_in_Firefox
50. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. pp. 275–292. Springer (2005). https://doi.org/10.1007/978-3-540-30574-3_19
51. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO '91. pp. 129–140. Springer (1992). https://doi.org/10.1007/3-540-46766-1_9
52. Pippenger, N.: On the evaluation of powers and monomials. SIAM Journal on Computing **9**(2), 230–250 (1980). https://doi.org/10.1137/0209022, https://doi.org/10.1137/0209022
53. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings. pp. 111–126. Springer (2016)
54. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: International conference on the theory and applications of cryptographic techniques. pp. 387–398. Springer (1996)
55. Qualys: SSL pulse (05 2024), https://www.ssllabs.com/ssl-pulse/
56. Ringers, S.: Using the RSA or RSA-B accumulator in anonymous credential schemes. IACR Cryptol. ePrint Arch. p. 11 (2023), https://ia.cr/2023/011

57. Sakemi, Y., Kobayashi, T., Saito, T., Wahby, R.S.: Pairing-Friendly Curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-11, Internet Engineering Task Force (Nov 2022), https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves/11/, work in Progress
58. Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 internet public key infrastructure online certificate status protocol – OCSP (06 2013), https://datatracker.ietf.org/doc/rfc6960
59. Schoenmakers, B.: Lecture notes cryptographic protocols (2025), https://berry.win.tue.nl/2DMI00/
60. Sporny, M., Longley, D., Chadwick, D.: Verifiable credentials data model (03 2022), https://www.w3.org/TR/vc-data-model/
61. Sullivan, N.: High-reliability OCSP stapling and why it matters (07 2017-07-10), https://blog.cloudflare.com/high-reliability-ocsp-stapling
62. Tessaro, S., Zhu, C.: Revisiting BBS signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 691–721. Springer (2023)
63. Vitto, G., Biryukov, A.: Dynamic universal accumulator with batch update over bilinear groups. In: Galbraith, S.D. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 13161, pp. 395–426. Springer (2022). https://doi.org/10.1007/978-3-030-95312-6_17, https://ia.cr/2020/777

A ZKP and Instantiations

We explicitly describe the ZKP algorithms introduced in Section 5.3, and how they can be instantiated as revocation mechanism for non-anonymous verifiable credentials (like mDOC/SD-JWT). We don't describe the combination with anonymous credentials as it is widely discussed [16,4,40,56].

A.1 ZKP for RSA accumulator

To prove knowledge of an accumulated element, the Holder must execute the algorithm $\texttt{GenZKP}_{\in}$ described in Section 5.3. In the following we describe how to compute the core proof π:

$$\pi \in PK\Big\{(x, r_x, r_w, r_1, xr_w, xr_1): \\ C_x = g^x h^{r_x} \wedge C_r = h^{r_w} g^{r_1} \wedge \alpha = C_w^x \left(\frac{1}{h}\right)^{xr_w} \wedge 1 = C_r^x \left(\frac{1}{h}\right)^{xr_w} \left(\frac{1}{g}\right)^{xr_1}\Big\} \tag{15}$$

This means that a Holder is generating a Proof of Knowledge (PoK) that keeps the values $x, r_x, r_w, r_1, xr_w, xr_1$ secret but assures that the given relations are satisfied. All the commitments C_x, C_w, C_r are sent by the Holder to the Verifier as part of the proof, so they are shared knowledge.

The first two equations are used to prove that the Holder knows how to open the commitments C_x and C_r and a proof of discrete logarithm equality, which aims to show that the value x and the randomness r_w appearing in the following equations are the same as used in the generation of commitments. The next equation:

$$\alpha = C_w^x \left(\frac{1}{h}\right)^{xr_w} \tag{16}$$

is the core of the proof and shows that C_w is a commitment to a witness for x. The last equation:

$$1 = C_r^x \left(\frac{1}{h}\right)^{xr_w} \left(\frac{1}{g}\right)^{xr_1} \tag{17}$$

is to show that the exponent of $\frac{1}{h}$ in the previous equation is indeed of the form xr_w.

We describe the ZKP as a sigma-protocol [21], which is possible to turn into a non-interactive protocol using the Fiat-Shamir transform [54,32,2]:

1. Holder initializes a proof generating the commitments C_w, C_x and C_r and generates a random element for each secret value:

$$v_x \sim 2^{k'+k''+\emptyset} \tag{18}$$

$$v_{r_x}, v_{r_w}, v_{r_1} \sim \lfloor n/4 \rfloor 2^{k'+\emptyset} \tag{19}$$

$$v_{xr_w}, v_{xr_1} \sim \lfloor n/4 \rfloor 2^{k'+k''+\emptyset} \tag{20}$$

   where k' is the dimension of the challenge c, k'' the accumulated element x and $\emptyset$ is an additional security parameter [37]. Then the Holder computes the commitments

$$\begin{aligned} T_x &= g^{v_x} h^{v_{r_x}} \\ T_r &= h^{v_{r_w}} g^{v_{r_1}} \\ T_w &= C_w^{v_x} \left(\frac{1}{h}\right)^{v_{xr_w}} \\ T_u &= C_r^{v_x} \left(\frac{1}{h}\right)^{v_{xr_w}} \left(\frac{1}{g}\right)^{v_{xr_1}} \end{aligned}$$

   and sends $(C_w, C_x, C_r, T_x, T_r, T_w, T_u)$ to Verifier
2. the Verifier generates random challenge $c \xleftarrow{\$} \{0,1\}^{k'}$;
3. the Holder computes the responses

$$\begin{aligned} s_x &= v_x - c \cdot x \\ s_{r_x} &= v_{r_x} - c \cdot r_x \\ s_{r_w} &= v_{r_w} - c \cdot r_w \\ s_{r_1} &= v_{r_1} - c \cdot r_1 \\ s_{xr_w} &= v_{xr_w} - c \cdot xr_w \\ s_{xr_1} &= v_{xr_1} - c \cdot xr_1 \end{aligned}$$

4. Finally, the Verifier can check

$$\begin{aligned} T_x &= C_x^c g^{s_x} h^{s_{r_x}} \\ T_r &= C_r^c h^{s_{r_w}} g^{s_{r_1}} \\ T_w &= \alpha^c C_w^{s_x} \left(\frac{1}{h}\right)^{s_{xr_w}} \\ T_u &= C_r^{s_x} \left(\frac{1}{h}\right)^{s_{xr_w}} \left(\frac{1}{g}\right)^{s_{xr_1}} . \end{aligned}$$

When the ZKP is made non-interactive, it is given by the values

$$(C_x, C_w, C_r, c, s_x, s_{r_x}, s_{r_w}, s_{r_1}, s_{xr_w}, s_{xr_1})$$

where $c = H(C_x, C_w, C_r, T_x, T_r, T_w, T_u)$.

When we instantiate the RSA accumulator as a revocation mechanism for verifiable credentials mDOC/SD-JWT, the value C_x is not computed by the Holder during the presentation of the credential, but it is precomputed when the credential is issued, and it is included in the credential as an attribute. All the other operations described above remain unchanged.

A.2 ZKP for EC accumulator

To prove knowledge of an accumulated element, the Holder must execute the algorithm $\texttt{GenZKP}_{\in}$ described in Section 5.3. In the following we describe how to compute the core proof π:

$$\pi \in PK\Bigg\{(x,\sigma,\rho,x\sigma,x\rho) : C_\sigma = h^\sigma \wedge C_\rho = k^\rho \\ \wedge 1 = C_\sigma^x\left(\frac{1}{h}\right)^{x\sigma} \wedge 1 = C_\rho^x\left(\frac{1}{k}\right)^{x\rho} \wedge \frac{\mathbf{e}(\alpha,g_2)}{\mathbf{e}(C_w,j)} = \mathbf{e}(C_w,g_2)^x\left(\frac{1}{E_{z,g_2}}\right)^{x\sigma+x\rho}\left(\frac{1}{E_{z,j}}\right)^{\sigma+\rho}\Bigg\} \tag{21}$$

where we recall that $z,h,k,g_2,j = g_2^{\mathsf{sk}}$ are public values, therefore $E_{z,g_2} = \mathbf{e}(z,g_2)$ and $E_{z,j} = \mathbf{e}(z,j)$ can be precomputed.

As for the RSA case, this means that a Holder is generating a Proof of Knowledge (PoK) that keeps the values $x,\sigma,\rho,x\sigma,x\rho$ secret but assures that the given relations are satisfied. All the commitments C_σ,C_ρ,C_w are sent by the Holder to the Verifier as part of the proof, so they are shared knowledge.

The first two equations are used to prove that the Holder knows how to open the commitments C_σ and C_ρ and a proof of discrete logarithm equality, which aims to show that the value x and the randomnesses σ,ρ appearing in the following equations are the same as used in the generation of commitments.

The following two equations are to show that the exponent of $\dfrac{1}{E_{z,g_2}}$ is in the form $x(\rho+\sigma)$.

The last equation is the core of the proof, indeed applying pairing properties:

$$\mathbf{e}(w,g_2^x j) = \frac{\mathbf{e}(C_w,g_2^x j)}{\mathbf{e}(z^{\sigma+\rho},g_2^x j)} = \frac{\mathbf{e}(C_w,g_2)^x\mathbf{e}(C_w,j)}{\mathbf{e}(z,g_2)^{x\sigma+x\rho}\mathbf{e}(z,j)^{\sigma+\rho}}$$

Therefore, proving that the last equation holds and the commitment are correctly generated, as granted by the previous relations, implies that (10) must hold.

We describe the ZKP as a sigma-protocol [21], which is possible to turn into a non-interactive protocol using the Fiat-Shamir transform [54,32,2]:

1. Holder initializes a proof generating the commitments C_w,C_σ,C_ρ, then it samples uniformly a random element for each secret value $v_x,v_\sigma,v_\rho,v_{x\sigma},v_{x\rho} \xleftarrow{\$} \mathbb{Z}_p$ and computes
$$\begin{aligned} T_\sigma &= h^{v_\sigma} \\ T_\rho &= k^{v_\rho} \\ T_{\bar\sigma} &= C_\sigma^{v_x}\left(\frac{1}{h}\right)^{v_{x\sigma}} \\ T_{\bar\rho} &= C_\rho^{v_x}\left(\frac{1}{k}\right)^{v_{x\rho}} \\ T_{\mathbf{e}} &= \mathbf{e}(C_w,g_2)^{v_x}\left(\frac{1}{E_{z,g_2}}\right)^{v_{x\sigma}+v_{x\rho}}\left(\frac{1}{E_{z,j}}\right)^{v_\sigma+v_\rho} \end{aligned}$$
and sends $(C_w,C_\sigma,C_\rho,T_\sigma,T_\rho,T_{\bar\sigma},T_{\bar\rho},T_{\mathbf{e}})$ to the Verifier
2. the Verifier samples a random challenge $c \xleftarrow{\$} \mathbb{Z}_p$ and sends it back to the Holder

3. the Holder computes the responses:

$$\begin{aligned}
s_x &= v_x - c \cdot x \\
s_\sigma &= v_\sigma - c \cdot \sigma \\
s_\rho &= v_\rho - c \cdot \rho \\
s_{x\sigma} &= v_{x\sigma} - c \cdot x\sigma \\
s_{x\rho} &= v_{x\rho} - c \cdot x\rho
\end{aligned}$$

   and sends it to the Verifier
4. the Verifier checks that

$$\begin{aligned}
T_\sigma &= C_\sigma^c h^{s_\sigma} \\
T_\rho &= C_\rho^c k^{s_\rho} \\
T_{\bar{\sigma}} &= C_\sigma^{s_x} \left(\frac{1}{h}\right)^{s_{x\sigma}} \\
T_{\bar{\rho}} &= C_\rho^{s_x} \left(\frac{1}{k}\right)^{s_{x\rho}} \\
T_{\mathbf{e}} &= \left(\frac{\mathbf{e}(\alpha, g_2)}{\mathbf{e}(C_w, j)}\right)^c \mathbf{e}(C_w, g_2)^{s_x} \left(\frac{1}{E_{z,g_2}}\right)^{s_{x\sigma}+s_{x\rho}} \left(\frac{1}{E_{z,j}}\right)^{s_\sigma+s_\rho}
\end{aligned}$$

When the ZKP is made non-interactive, it is given by the values

$$(C_\sigma, C_w, C_\rho, c, s_x, s_\rho, s_\sigma, s_{x\rho}, s_{x\sigma})$$

where $c = H(C_w, C_\sigma, C_\rho, T_\sigma, T_\rho, T_{\bar{\sigma}}, T_{\bar{\rho}}, T_{\mathbf{e}})$.

When we instantiate the EC accumulator as a revocation mechanism for verifiable credentials mDOC/SD-JWT, we must add a Pedersen commitment to x: $C_x = g_1^x z^r \in \mathbb{G}_1$ in the Credential which is signed by the Issuer, where $g_1, z \in \mathbb{G}_1$ are the public values previously defined, for which $\log_{g_1} h$ is unknown. Then the Holder will execute the ZKP described above with the following modifications in the sigma protocol execution:

1. The Holder executes the operations described above in Item 1 of the the sigma protocol above, and additionally samples at random $v_r \xleftarrow{\$} \mathbb{Z}_p$, computes $T_x = g_1^{v_x} z^{v_r}$, and send to the Verifier also the value T_x;
2. the Holder receives the challenge c;
3. the Holder in addition to computing the responses described in Item 3 of the sigma protocol above, computes $s_r = v_r - c \cdot r$, and sends it to the Verifier
4. the Verifier performs the checks described in Item 4 and also checks that $T_x = C_x^c g_1^{s_x} z^{s_r}$.

B Witnesses as digital signatures

We observe that the RSA and Elliptic Curve accumulators can be viewed as a special use of RSA and BLS [13] digital signature algorithms, respectively. In short, generating a witness w_x for an element x can be compared to generating a new signature of the same message α under the public key x.

As well as being of academic interest, this view can be of assistance to readers familiar with those algorithms, and it is informative insofar as their performance and comparison with other revocation mechanisms are concerned. Moreover, if more efficient ZKP are defined for a signature scheme than those known in the literature for accumulators, they can be applied – as done in [20, Section 4.3].

A textbook RSA digital signature and verification for message m, with private key d and public key $e, N \leftarrow pq$ is:

$$\texttt{Sign}(m, d, N): \ \sigma \leftarrow m^d \mod N \tag{22}$$

$$\texttt{Verify}(m, e, N): \ \mathit{bool} \leftarrow m \overset{?}{=} \sigma^e \mod N \tag{23}$$

The public exponent e is fixed and known to signer and verifier, typically $e = 65537$; the private exponent is $d \leftarrow e^{-1} \mod \phi(N)$, the totient function $\phi(N) = (p-1)(q-1) = 4p'q'$. One may consider the RSA accumulator as akin to signing the same public message with different private and public exponents for each element, in the following sense:

– from the Status Manager key $\texttt{sk} = p'q'$, the Status Manager generates a new private key $d_x \leftarrow x^{-1} \mod \texttt{sk}$ for every element x to be accumulated;
– generating a signature for the public message $m = \alpha_t$ is equivalent to generating a new witness $w_{x|t} \leftarrow \alpha^{d_x} \mod N$
– verifying that the element x is accumulated is equivalent to verifying that the signature of the message α_t under the private key x^{-1} is valid under the public key x, N.

In a similar fashion, one can describe an analogy between a BLS signature and the EC accumulator. Given a bilinear pairing $\mathbf{e} : G_1 \times G_2 \longrightarrow G_T$, with g_2 generator of G_2, a BLS digital signature and verification for a message m, with private key d and public key $j \leftarrow g_2^d$ is:

$$\texttt{Sign}(d, m): \ \sigma \leftarrow m^d \tag{24}$$

$$\texttt{Verify}(m, j): \ \mathit{bool} \leftarrow \mathbf{e}(\sigma, g) \overset{?}{=} \mathbf{e}(m, j) \tag{25}$$

One may consider the EC accumulator as again signing the same public message with different private and public exponents:

– from the Status Manager key $\texttt{sk}$, the Status Manager generates a new private key $d_x \leftarrow (\texttt{sk}+x)^{-1}$ for every element x to be accumulated;
– generating a signature for the public message $m = \alpha_t$ with the private key $d_x = (\texttt{sk} + x)^{-1}$ is equivalent to generating a new witness $w_{x|t} \leftarrow \alpha_t^{d_x}$;
– verifying that the element x is accumulated follows the same steps as BLS under the public key $g_2^{\texttt{sk}}$.

The similarity in principle between signatures and witnesses was previously argued in [4,40], in which digital signatures serve as positive additive accumulators, and proving membership of an element x, is achieved by proving knowledge of a signature over x. Here we argue that also for some positive dynamic accumulator schemes, witness generation and verification algorithms are equivalent to signing and verification algorithms of well-known digital signatures on a common message α.