Comparison of Credential Status Mechanisms for the Digital Wallet Ecosystem

Riccardo Germenia[1,2][a], Salvatore Manfredi[2][b], Giada Sciarretta[2][c], Mario Scuro[2][d] and Alessandro Tomasi[2][e]

[1]*Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 9, Trento, Italy*

[2]*Center for Cybersecurity, Fondazione Bruno Kessler, Via Sommarive 18, Trento, Italy*

Keywords: Digital Credentials, Wallet, Credential Status, Revocation Mechanisms.

Abstract: Digital identity wallets enable citizens to verify their identity and manage digital credentials. A system that offers the possibility of using and presenting credentials, requires the ability to check for their validity, avoiding the use of revoked or suspended credentials. This paper compares traditional and emerging credential status mechanisms to identify the most suitable solutions for the wallet ecosystem, taking in consideration privacy aspects and the set of available features.

1 INTRODUCTION

A digital identity wallet is software able to store, manage and present digital documents called Credentials. It enables citizens to autonomously handle government-issued ID, driver's license, health records, or even academic certificates with their devices. These documents are cryptographically signed to guarantee that the information contained within the Credentials is authentic and has not been tampered with. This ecosystem considers the existence of three main roles: *Credential Issuer*, *Holder* (sometimes called Wallet User), and *Verifier* (sometimes called Relying Party). A *Credential Issuer* creates and issues Credentials to a *Holder*. A *Verifier* receives and verifies Credentials presented by a *Holder*. The *Verifier* checks that a presented Credential is issued by a trusted *Credential Issuer*, is not expired and has a valid status – i.e., is not revoked or suspended.

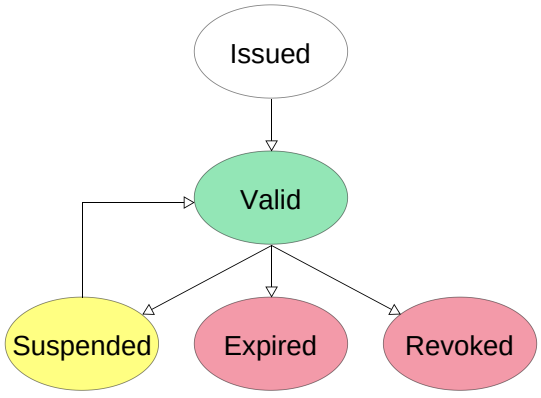

Figure 1: Credential Lifecycle.

Managing the lifecycle of a Credential, and in particular its status, is not a novel problem. If we consider a Web PKI scenario and X.509 certificates, well-known and established methods to implement revocation such Certificate Revocation List (CRL) (Russ and Paul, 1999) and Online Certificate Status Protocol (OCSP) (Galperin et al., 1999) existed for a long time. However, in the context of digital wallet Credentials, new standards are under discussion, such as Token Status List (Looker et al., 2025) and Status Assertions (De Marco et al., 2024), yet they currently lack either implementation or security considerations able to lead an informed choice on which mechanism to favor. In this paper, we focus on the analysis of Credential status mechanisms to understand which is the more suitable for the wallet ecosystem. We first describe the most well-established revocation mechanisms in the Web PKI scenario and the new draft specifications (Section 2). Then, we compare them in terms of privacy and supported features (Section 3) highlighting peculiar pros and cons which can guide

[a] https://orcid.org/0009-0009-9283-8030
[b] https://orcid.org/0000-0001-9645-6034
[c] https://orcid.org/0000-0001-7567-4526
[d] https://orcid.org/0000-0003-2410-3760
[e] https://orcid.org/0000-0002-3518-9400

Germenia, R., Manfredi, S., Sciarretta, G., Scuro, M. and Tomasi, A.
Comparison of Credential Status Mechanisms for the Digital Wallet Ecosystem.
DOI: 10.5220/0013635500003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 741-749
ISBN: 978-989-758-760-3; ISSN: 2184-7711

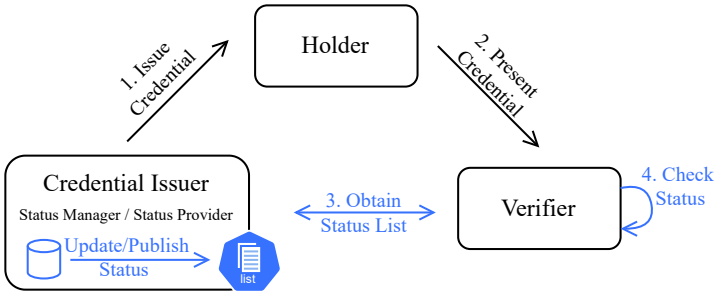

Figure 2: List-based Status Mechanism - High Level Flow.

an implementor's choice (Section 3.3). Finally, we conclude with a set of considerations, open challenges and a summary of the contributions provided by this comparison (Section 4).

2 STATUS MECHANISMS OVERVIEW

Unlike short-lived Credentials, which are subject to a reduced validity interval that reduces the window of opportunity for unauthorized use, long-lived Credentials are subject to a secure and privacy-preserving management of their lifecycle.

During its lifecycle (see Figure 1), a Credential can usually change its status from *issued* to *valid*, *suspended*, *revoked* or *expired*.[1] While the change from *issued* to *valid* or *expired* can be directly checked by the *Verifier* by looking at the "issued at", "not before" and "expiration" time claims inside the Credential, other status, such as *suspended* or *revoked*, need a more complex handling. To manage and distribute the status of a given Credential to *Verifier*s, two additional roles are usually involved: *Status Manager* and *Status Provider*. While the *Credential Issuer*, *Status Manager* and *Status Provider* roles are typically performed by the same entity, in some cases the *Status Provider* role can be delegated to a third-party trusted entity in order to distribute the workload.

A preliminary phase is usually performed by the *Status Manager* to generate the cryptographic parameters, create the status lists or setup the database to be used by the status mechanism. After this setup phase, depending on the status mechanism, the management and provision of a status involve the following phases:

***Credential Issuance*:** phase performed by the *Holder* to request a Credential to the *Credential Issuer*. The *Credential Issuer* generates the Credential and sends it back to the *Holder* with information required to fetch the status.[2]

***Status Update*:** phase performed by the *Status Manager* to update the status of an existing Credential. Depending on the mechanism, the updated status may not be directly accessible by a *Verifier*.

***Status List Publish*:** phase performed by the *Status Manager* to share a snapshot of the current statuses to the *Status Provider*.

***Refresh Status*:** phase performed by the *Holder* to obtain fresh status information of the Credential.

***Credential Presentation*:** phase performed by the *Holder* to present a Credential (and optionally its status assertion/proof). This phase involves the check of the status of the Credential by the *Verifier* (optionally requesting this information to the *Status Provider*).

The following sections provide a high level description of the different status mechanisms from the literature, grouped by type and with a description of the involved phases. To make the description easier to follow, we always use the term Credential even if some specifications refer to the status of Referenced Tokens or X.509 certificates; and we set our terminology for the entities involved.

2.1 List-Based Status Mechanisms

These mechanisms store the information of all managed statuses within *Status List*s. As shown in Figure 2, the flow consists of four main phases: *(1)* the *Credential Issuer* issues a Credential to a *Holder* with inside a reference to the corresponding *Status List*; *(2)* the *Holder* presents the Credential to a *Verifier*; *(3)* the *Verifier* obtains the *Status List* from the *Status Provider*; and *(4)* the *Verifier* checks the status of the Credential presented by the *Holder*.

[1] This list contains the most common values for the Credential status. Each use case scenario may define its own.

[2] To simplify the text description, we are considering the request, issuing and presentation of a single Credential. There are protocols that allow for batch Credential issuance and presentation of multiple Credentials.

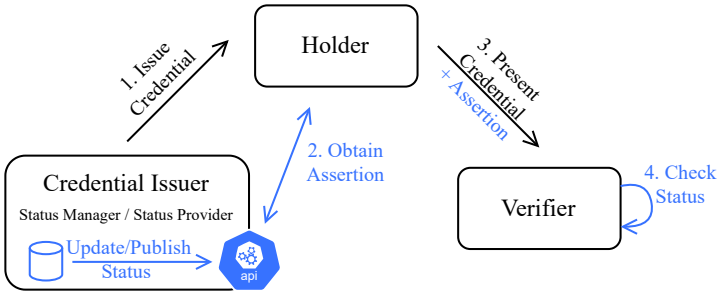

Figure 3: Assertion-based Status Mechanism - High Level Flow.

In this approach, the *Refresh Status* phase is not performed as the status is directly retrieved by the *Verifier* without *Holder* involvement, and the role of the *Status Provider* can be delegated to a third party as it only stores the *Status List* to be fetched.

2.1.1 CRL

Certificate Revocation Lists (CRLs) are a revocation mechanism defined in 1999, first designed to handle X.509 certificates within PKI repositories (Russ and Paul, 1999) and then repeatedly updated to better describe its structure and functioning (e.g., with (Russ Housley and Solo, 2002)). CRLs are lists designed to contain the unique identifier of Credentials that have been revoked before their scheduled expiration date and should no longer be trusted. Therefore the lifetime of a Credential does not match its presence in the list, and it only impacts the CRL management upon revocation.

2.1.2 Token Status List

Token Status List (Looker et al., 2025) (hereafter, SL) is an individual Internet-Draft (first published in June 2023) that defines how to manage the status of Credentials in JSON and CBOR formats (called Referenced Tokens[3]) conveyed via a bit array in a *Status List*. Since its issuance, the status of a Credential (either valid or not) is always contained in the *Status List*.

2.2 Assertion-Based Status Mechanisms

As shown in Figure 3, these mechanisms work by sending, together with the Credential, an assertion specifying the status information (*Status Assertion*) of that Credential to the *Verifier*. The flow consists of four steps: *(1)* the *Credential Issuer* issues a Credential to a *Holder*, *(2)* the *Holder* obtains the *Status Assertion* of its Credential from the *Status Provider*, *(3)* the *Holder* presents its Credential along with the *Status Assertion*, and *(4)* the *Verifier* checks the status of the presented Credential by validating the attached *Status Assertion*.

On one hand, this approach allows *Verifiers* to check the status of a Credential directly, without having to request the status from the *Status Provider*. On the other, a *Holder* has to store and periodically refresh the *Status Assertion* for its Credential.

2.2.1 OCSP with Stapling

Online Certificate Status Protocol (OCSP) is a revocation mechanism defined in 1999 (Galperin et al., 1999) to offer a real-time status fetching as an alternative to the "periodic updates" offered by CRLs (see Section 2.1.1). The *Verifier* queries the *Status Provider* each time it needs to verify the status of a Credential, rather than downloading the list at each presentation. In response to each request, the *Status Provider* consults its internal database (i.e., a CRL updated periodically) and returns the requested status to the *Verifier*. To simplify the process, in 2011 RFC 6066 (Donald E. Eastlake 3rd, 2011) defined a TLS extension to "staple" the status to the Credential itself (OCSP with stapling, OCSP w/s for short). This mechanism allows the *Holder* to obtain a signed proof of validity from the *Status Provider*, the proof will then be "stapled" (attached) to the Credential during the *Credential Presentation*.

The *Status Provider* role can be delegated to third-party trusted entity that will be able to autonomously sign the OCSP Responses.

2.2.2 OAuth Status Assertions

OAuth Status Assertions (De Marco et al., 2024) (hereafter, SA) is an individual Internet-Draft (first release in January 2024) that introduces an assertion-based approach in the wallet context. The main reason behind this proposal is to enhance privacy by reducing

[3](Looker et al., 2025) uses the term Referenced Token as it is a token containing a reference to a *Status List*, which gives to the *Verifier* additional information about the current status of the *referenced* token.

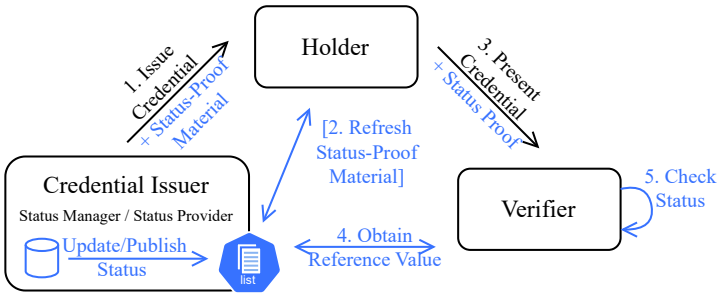

Figure 4: Hybrid Status Mechanism - High Level Flow.

the excessive monitoring of the *Holder*'s activities by *Verifiers*. Indeed there are use cases where the *Verifier* only needs to check the status of a Credential at the time of presentation. In these cases, it should not be allowed to check the status of a Credential over time.

In this specification, the *Status List Publish* phase is not performed as the Credential status is directly managed by the *Credential Issuer* who plays also the role of the *Status Manager* and *Status Provider*. Unlike OCSP with Stapling, the SA draft does not foresee the delegation of the *Status Provider* role.

2.3 Hybrid

As shown in Figure 4, hybrid mechanisms work by publishing a common reference value (it may be a *Status List* or be completely opaque as to the content of the set, such as a completely random number or a signed Merkle tree root), as well as providing *Holder*s with an individual value, e.g., a seed, or a witness. The *Holder* value may be used with the Credential and the common reference value to prove (non-)membership of a Credential in a set (*Status Proof*), which may be an allow – or deny – list. The flow consists of five steps: *(1)* the *Credential Issuer* issues a Credential and the information needed to generate a *Status Proof* to a *Holder*, *(2)* the *Holder* refreshes the *Status Proof* information of its Credential from the *Status Provider*, *(3)* the *Holder* presents its Credential along with the *Status Proof*, *(4)* the *Verifier* obtains the common reference value representing the statuses from the *Status Provider*, and *(5)* the *Verifier* checks the status of the presented Credential by verifying the membership of the Credential in the common reference value using the provided membership proof (*Status Proof*).

As this mechanism requires both the fetching of the common reference value from the *Status Provider* (as for list-based mechanisms) and the exchange of status info from the *Holder* (as for assertion-based mechanisms), we call it hybrid.

2.3.1 Accumulators

Cryptographic accumulators (hereafter, ACC) are schemes that represent a finite set (S) of elements as a single value (accumulator value α) and allow for the generation of the membership witness for each element x of the set, which may be used to verify that $x \in S$. After updating S, each witness must be updated as well. These schemes have been adopted in the context of anonymous credentials (Khovratovich et al., 2022; Hyperledger, 2024) and have recently been discussed for their application to revocation in the wallet scenario (Flamini et al., 2025).

There are different types of accumulators based on whether elements can be added and/or removed from the accumulated set, and whether the scheme supports membership and/or non-membership proofs. We refer to (Barić and Pfitzmann, 1997; Fazio and Nicolosi, 2002; Derler et al., 2015; Barthoulot et al., 2024) for a complete characterization. In this paper, we consider positive optimal accumulators as allow-lists – i.e., *Holder*s present proofs of membership and *Verifiers* reject presentations without such proof – supporting Zero Knowledge Proofs (ZKP), such as the ones described in (Camenisch and Lysyanskaya, 2002; Li et al., 2007; Baldimtsi et al., 2017) based on RSA and in (Nguyen, 2005; Vitto and Biryukov, 2022; Karantaidou and Baldimtsi, 2021; Jaques et al., 2022) based on Elliptic Curves. ZKPs allow *Holder*s to prove their Credentials are members of an accumulator without revealing the actual value of the element to the *Verifier*.

2.3.2 Dynamic Status List

Dynamic Status List (DSL), first presented as Dynamic Token Status List[4], then adopted as DSL by EBSI[5], is based on generating an allow-list of valid credentials using the time-based one-time password (TOTP) (M'Raihi et al., 2011) algorithm. The in-

[4]https://github.com/cre8/dynamic-token-stauts-list
[5]https://hub.ebsi.eu/vc-framework/credential-status-framework/vcs#dynamic-status-list

Table 1: Privacy Comparison.

	CRL	SL	OCSP w/s	SA	ACC	DSL
P1 - *Status Manager*-*Verifier* collusion protection	✗	✗	✗	✗	✗	✗
P2 - *Status Provider* tracking *Holder* protection	✓	✓*	✓	✓	✓	✓
P3 - *Verifier* unauthorized status check protection	✗	✗	✗	✓	✓	✓
P4 - *Verifiers* collusion protection	✗	✗	✗	✗	✗	✗
P5 - *Status Provider* tracking *Verifier* protection	✗*	✗*	✓	✓	✗*	✗*
P6 - Third Parties passive analysis protection	✗	✗	✓	✓	✗*	✗*

✓* and ✗* mean that the related protection is dependent on specific conditions.

tuition is that only the *Holder* and *Credential Issuer* know a shared secret seed and can generate the current *Status Proof* as a pre-image of a hash digest published in the list. This requires a cryptographic hash function H and an epoch duration T, depending on the use case – e.g., 24 hours – from which at any given time t the current epoch counter will be computed as $t' = \lfloor t/T \rfloor$. Since the resulting allow list may be orders of magnitude larger than a CRL, additional efficiency may be gained by applying a Bloom filter – plain (Bloom, 1970) or cascading, as in CRLite (Larisch et al., 2017) – to the resulting list.

Differently from accumulators, there is no need for the *Holder* to refresh their local value as the seed never changes and new values are computed using TOTP. The *Refresh Status* phase does not take place.

3 STATUS MECHANISMS COMPARISON

To better comprehend the differences between the grouping described in Section 2 and highlight some peculiarities, we compare and discuss them in terms of privacy and supported features. Please note that Tables 1 and 2 only take into account the status mechanisms as they are described in their related documents, without considering any optimization, optional mitigation or – for CRL and OCSP w/s – any wallet-specific customization. Furthermore, we consider that each entity behaves according to the specifications and without performing operations outside its designed role.

3.1 Privacy Considerations

What does privacy mean for Credential status mechanisms? How much do the solutions described in Section 2 preserve privacy? To answer these research questions, we have first identified six different privacy threats that may occur by managing statuses. The comparison between the status mechanisms is shown in Table 1.

Holder Privacy

P1 - *Status Manager*-*Verifier* collusion: Indicates if the *Status Manager* is able to collude with a *Verifier* to link revocation data.

P2 - *Status Provider* tracking *Holder*: Indicates if the *Status Provider* is able to track the *Holder*'s activities on *Verifier*(s).

P3 - *Verifier* unauthorized status check: Denotes if it possible to autonomously check the status of a Credential over time without the consent of the *Holder* (after a first *Credential Presentation*).

P4 - *Verifiers* collusion: Indicates if different *Verifiers* can collude to track the *Holder*'s activity.

Information Disclosure

P5 - *Status Provider* tracking *Verifier*: Signals if the *Status Provider* is able to track the requests of a *Verifier*.

P6 - Third Parties passive analysis: Indicates if third parties (i.e. entities that do not play any of the mentioned roles) are able to extract statistics about the revoked credentials.

Note that both P1 and P4 (collusion attacks) affect all mechanisms. Focusing on P4, if the same Credential is presented to different *Verifiers*, the correlation is possible even when selectively disclosing different claims due to linkable status information being provided together with the Credential.[6] A possible solution for all mechanisms is the use of batch issuance of Credentials. In this way, the *Holder* will present a different Credential to each *Verifier*. For some of these mechanisms (e.g., CRL and SL) this solution requires the management of an entry for each batch item in the *Status List*, one for each issued Credential, in order to prevent linking. Instead, ACC is able to manage a

[6]CRL Credential contains a unique serial number, SL Credential contains the *Status List* url and the related index, the SA *Status Assertion* contains the hash of the Credential, for ACC this is induced by the commitment in each Credential, for DSL both by the idx value in the Credential and the same value of the Status Proof within one epoch.

Table 2: Features Comparison

	CRL	SL	OCSP w/s	SA	ACC	DSL[a]
F1 - Implementation gap	●○○○	●●○○	●○○○	●●●○	●●●●	●●●●
F2 - *Holder* load	○○○	○○○	●○○	●○○	●●●	●●○
F3 - *Verifier* load	●●○	●●○	●○○	●○○	●●○	●●○
F4 - *Status Provider* load	●○○	●○○	●●●	●●●	●●●	●○○
F5 - *Holder* offline	✓	✓	✓*	✓*	✓	✓
F6 - *Verifier* offline	✓*	✓*	✓	✓	✗*	✓*
F7 - Verification data size	●●●○	●●○○	●○○○	●○○○	●○○○	●●●●
F8 - Covered statuses[b]	R, S	Any	R	Any	R	R
F9 - Status Format	ASN.1	JWT/CWT[c]	ASN.1	JWT/CWT	Not set[d]	Not set[d]

✓* and ✗* mean partially yes or partially no, respectively.

[a] We consider DSL without Bloom Filters.

[b] Revocation (R), Suspension (S) or any possible values (Any).

[c] *Status List* are structured in JSON and CBOR formats, then compressed and signed into JWT/CWT tokens.

[d] No common format exists. There does not appear to be any incompatibility with JWT or CBOR in principle.

unique status by providing a different hiding commitment in each issued Credential.

Assuming that the *Status Lists*/accumulator values do not contain status information of only one credential, and that the key used to sign *Status Assertions* is the same for all requests (honest *Status Manager* assumption), P2 is mitigated by all solutions. Regarding SL, while the current version of the draft discusses the use of herd privacy to mitigate the monitoring of *Holder*'s activities, it does not describe how to achieve it. In list-based mechanisms, herd anonymity is usually achieved by setting a minimum number of entries to be instantiated.

Regarding P3, it is interesting to note that even if OCSP w/s and SA are based on the same logic, the latter added means to prevent a malicious *Verifier* from fetching the status of a Credential without *Holder*'s consent. This is possible because in SA only the *Holder* is allowed to contact the *Status Provider* and fetch the status of its Credential. ACC and DSL also provide protection, because the common reference value changes at every publication and the verification algorithm cannot be run without a fresh proof, provided by the *Holder*.

OCSP w/s and SA provide protection against P5 by design, as the *Status Provider* interacts with the *Holder* and there is no contact with *Verifiers*. While CRL, SL, ACC and DSL are potentially exposed to this threat, as *Verifiers* only need to download the *Status List* or accumulated value once per publication, regardless of how many presentations happen in that period, the tracking is mitigated.

Finally regarding P6, by monitoring a CRL one knows exactly which credentials have been revoked, with an SL one knows the number of revoked credentials. For both CRL and SL a mitigation is to use decoys. OCSP w/s[7] and SA do not provide global information. An accumulator update message, if used by the scheme, may reveal the number of revoked Credentials since the last publication; and a DSL without Bloom Filters reveals the number of currently issued (valid and revoked) Credentials, while with Bloom Filter the number obtained is an estimate.

3.2 Features Considerations

Table 2 compares the following key features for each considered mechanism, more black circles (●) correspond to a lower rating.

F1 - Implementation Gap: Indicates the gap between reading the reference document (RFC/draft) to a working implementation. The defining elements are the existence of structured libraries and the level of detail in the mechanisms described.

F2 - *Holder* Load: Indicates the load that a *Holder* has to handle in order to use a given revocation mechanism. Since in a digital wallet ecosystem the *Holder* is usually a smartphone, the presence of limited resources plays a key factor in evaluating the mechanisms.

F3 - *Verifier* Load: Indicates the load that a *Verifier* has to handle. In a digital wallet ecosystem, *Verifier* role can be played by a server (with cheap processing power and storage) or by a smartphone (with limited power and capabilities, impacting a device's battery).

F4 - *Status Provider* Load: Indicates the load that a *Status Provider* has to handle. While the role

[7] Considering the scenario in which the *Credential Issuer* does not publish the underlying CRL.

of the *Status Provider* is usually played by the same entity that acts as a *Status Manager* (and thus with cheap processing power and storage), a *Status Provider* has both many resource-intensive tasks and, unless scaling strategies are employed, is contacted repeatedly. In some cases, this may result in performance degradation and excessive costs (Aas, 2024).

F5 - *Holder* Offline: Indicates if a *Holder* can perform a *Credential Presentation* without having internet access.

F6 - *Verifier* Offline: Indicates if a *Verifier* is able to validate the Credential provided by a *Holder* without having access to the internet.

F7 - Verification Data Size: Denotes the size of the exchange data to perform status verification. It goes from a list containing large data to simple variables transmitted within signed tokens.

F8 - Covered Statuses: Indicates the range of Credential statuses that a given status mechanism covers.

F9 - Status Format: Indicates the format used to elaborate and store the Credential status.

The reasoning behind each of the scores in Table 2 is as follows.

(F1). While CRL and OCSP w/s have reached a high level of maturity and are deployed worldwide, the other mechanisms are still drafts (SL, SA) or based only on a "core concept" (ACC, DSL). Focusing on SL and SA, while both are based on known formats (e.g., JWTs (Jones et al., 2015)), the former only requires the implementation of a list to store the index-status pair, the latter needs a larger integration of roles' behaviors, message transmission and data structure creation.

(F2). By design, *Holder*s are not involved in list-based mechanisms. Whereas they require to retrieve a status information for assertion-based and hybrid mechanism. In addition, hybrid mechanisms involve the *Holder* in generating fresh proofs from a locally held secret, which requires additional management. This may be based on symmetric (a seed for TOTP in DSL) or asymmetric cryptography (a witness in ACC).

(F3). Assertion-based mechanisms have less load on the *Verifier* as the only required action is verifying the signature and the status value of the received *Status Assertion*. List-based mechanisms require some extra load: while in CRL/DSL the most expensive operation is the list search, for SL the search is fast, what is expensive is the decompression. Finally, ACC is the most expensive as it requires the validation of a ZKP proof.

(F4). CRL, SL and DSL have a lower load on the *Status Provider* as *Verifier*s do not need to retrieve a list at each presentation, they can cache it and use the same list to check several Credentials. For ACC, together with the list fetching performed by the *Verifier*s, the *Holder*s need to contact the *Status Provider* to obtain updated witnesses (the frequency of this call depends on the frequency of status changes). Finally, for SA and OCSP w/s a *Status Provider*, unless scaling strategies are employed, it is contacted repeatedly by *Holder*s. In some cases, this has resulted in performance degradation and excessive costs for OCSP w/s (Aas, 2024). Due to the inability to let the *Status Provider* to be played by an entity different from the *Status Manager*, SA may suffer from an ever larger performance load.

(F5). By design, *Holder*s do not require status-related online communications in CRL, SL and DSL (no refresh phase). Regarding assertion-based/ACC, the *Holder* must possess a valid (and not expired) assertion/witness, requested before going offline.

(F6). ACC does not work if the *Verifier* is offline, unless the *Holder* downloads the current, or last known, accumulator value, and presents it to the *Verifier* during Credential presentation. CRL, SL and DSL work offline only if the *Verifier* possesses a valid (and not expired) *Status List*, fetched before going offline.

(F7). The content of a CRL changes dynamically: once a Credential is revoked, its entry is added to the list; and once it expires, the entry is removed. For each revoked Credential, CRLs store its unique serial (up to 159 bits long (Boeyen et al., 2008)) and a timestamp. SLs have a fixed size as they must preemptively instantiate a number of indexes equivalent to the amount of Credentials they plan to manage. When compared to CRLs, SLs are lighter as for each issued/revoked Credential they only update the status bits[8] associated with the Credential, do not store the entire serial number, and the list is compressed. OCSP w/s, SA and ACC common reference value are comparable in size, they have an almost-constant size. In ACC, depending on the scheme, *Holder*s may need to download a somewhat larger amount of data from which to generate a fresh proof, which may depend on how many publications

[8]Up to 8 bits for each Credential (Looker et al., 2025).

they have missed while being offline. If DSL are used without Bloom filter or compression, *Verifier*s need to download a huge list of the size of every currently valid Credential at every publication, whereas *Holder*s download nothing (the same seed is used all the time). The problem of *Verifier* download burden can be mitigated by cascading Bloom filters, but is not currently in the DSL specification or implementation.

(F8). CRL supports two status values: revoked and on hold. OCSP response is one of: good, revoked, unknown. Similarly, for DSL a Verifier may find the valid hash, the void hash (revoked), or not find either. SL defines three types – valid, invalid, suspended – and leaves room for up to 2^8 possible values. SA inherits status types from SL referring to the related IANA "Status Types" registry section.

3.3 Discussion

The selection of a status mechanism to implement is dependent on a variety of factors, including the privacy requirements and the features that each mechanism provides. Moreover, each mechanism implements in a different way the phases outlined in Section 2, resulting in varying computational costs that will influence each entity in a distinctive way.

Considering the scores assigned in Table 1 and Table 2, and taking into account that *Status Manager* and *Status Provider* can easily scale in terms of resources while *Holder* and a mobile *Verifier* cannot, we can outline the following use cases:

Resource-Constrained *Holder*. Given the limited computational power and memory, the list-based mechanisms are strongly advised in this use case as their usage load is shared between the *Status Manager* - which has to generate the list - and the *Verifier* - which will fetch the list and search for the presented Credential's status.

Resource-Constrained (Mobile App) *Verifier*. In the case of a *Verifier* with limited storage, assertion-based mechanisms are suggested instead as they require to sign the proof-of-validity for each Credential owned by a *Holder*. This shifts the load from the *Verifier* - that are only required to validate the received assertions instead of fetching and managing an entire list - to the *Status Provider* that has to sign the assertions and make them continuously available to the periodic status refresh performed by the *Holder*. Moreover, these mechanisms are a good fit for the "*Holder* offline" scenario as it would not need to perform the *Refresh Status* phase until its assertion expires.

Hybrid Mechanisms Costs Tradeoff. Hybrid mechanisms offer a different trade-off between communication and computation costs for all participants. DSL requires no interaction between Issuer and Holder after issuance and near-zero computational effort on the Holder, but has a huge communication cost from Provider to Verifier, which requires reliable connectivity and bandwidth. This makes it suitable for resource-constrained Holders, assuming capable Verifiers. ACC has very low communication costs for all parties, especially Verifiers when compared to list-based alternatives, but does require reliable connectivity for regular updates; it also has higher computation costs for all parties to generate and verify witnesses and ZKP.

4 CONCLUSIONS AND FUTURE WORK

It may come as a surprise to some that the community is working on new status mechanism drafts for the wallet ecosystem even though solutions such as CRL and OCSP w/s are widely deployed. From our analysis we notice that the main motivation is data protection, since OCSP and CRL were not designed with privacy in mind for websites. It is also an opportunity to address the shortcomings identified by CA and browsers over time, in particular the large size of CRL and volume of OCSP requests.

Comparing the new draft based on *Status List* (SL) with CRL, a better expressiveness is achieved, which allows to cover new status values specific for the wallet ecosystem, and the *Status List* is less heavy in terms of size. Regarding the *Status Assertion*-based, SA provides better privacy compared to OCSP w/s as it prevents a malicious *Verifier* to fetch the status of a Credential without the consent of a *Holder*. The choice between SL and SA depends on the use case. Each scenario needs to evaluate the level of privacy protection required (SA additionally provides protection against P3, P5, and P6) and how to distribute the load (for SL the load is mainly on the *Verifier*, while for SA the load is between the *Status Provider* and the *Holder*). Finally, we believe that hybrid mechanisms (ACC and DSL) are not yet ready for adoption, as there is a lack of a dedicated specification describing their use for credential status management and many aspects are not yet covered.

As future work, we plan to extend the scope of the comparison by analyzing the performances of each

mechanism (for each role) and providing insights on how an implementor might balance their costs to achieve an optimal revocation service.

ACKNOWLEDGEMENTS

This work has been partially supported by a joint laboratory between FBK and the Italian Government Printing Office and Mint and by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. The authors would like to personally thank Paul Bastian, Giuseppe De Marco, Francesco Antonio Marino, and Mirko Mollik for their valuable discussion and feedback.

REFERENCES

Aas, J. (2024). Intent to end OCSP service. https://letsencrypt.org/2024/07/23/replacing-ocsp-with-crls.html.

Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., and Yakoubov, S. (2017). Accumulators with applications to anonymity-preserving revocation. http://ia.cr/2017/043.

Barić, N. and Pfitzmann, B. (1997). Collision-free accumulators and fail-stop signature schemes without trees. https://link.springer.com/chapter/10.1007/3-540-69053-0_33.

Barthoulot, A., Blazy, O., and Canard, S. (2024). Cryptographic accumulators: New definitions, enhanced security, and delegatable proofs. https://ia.cr/2024/657.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. https://dl.acm.org/doi/10.1145/362686.362692.

Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., and Cooper, D. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://www.rfc-editor.org/rfc/rfc5280.html.

Camenisch, J. and Lysyanskaya, A. (2002). Dynamic accumulators and application to efficient revocation of anonymous credentials. https://iacr.org/archive/crypto2002/24420061/24420061.pdf.

De Marco, G., Steele, O., Marino, F. A., and Adomeit, M. (2024). OAuth Status Assertions. https://datatracker.ietf.org/doc/draft-demarco-oauth-status-assertions/03/. Work in Progress.

Derler, D., Hanser, C., and Slamanig, D. (2015). Revisiting cryptographic accumulators, additional properties and relations to other primitives. https://link.springer.com/chapter/10.1007/978-3-319-16715-2_7.

Donald E. Eastlake 3rd (2011). Transport Layer Security (TLS) Extensions: Extension Definitions. https://www.rfc-editor.org/rfc/rfc6066.

Fazio, N. and Nicolosi, A. (2002). 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. Paper written for course at New York University.

Flamini, A., Ranise, S., Sciarretta, G., Scuro, M., Smaniotto, N., and Tomasi, A. (2025). Public key accumulators for revocation of non-anonymous credentials. Cryptology ePrint Archive, Paper 2025/549.

Galperin, S., Adams, C., Myers, M., Ankney, R., and Malpani, A. N. (1999). X.509 internet public key infrastructure online certificate status protocol - ocsp. https://www.rfc-editor.org/rfc/rfc2560.

Hyperledger (2024). Anoncreds v2. https://github.com/hyperledger/anoncreds-v2-rs.

Jaques, S., Lodder, M., and Montgomery, H. (2022). ALLOSAUR: accumulator with low-latency oblivious sublinear anonymous credential updates with revocations. https://ia.cr/2022/1362.

Jones, M., Bradley, J., and Sakimura, N. (2015). RFC 7519 - JSON Web Token (JWT). https://www.rfc-editor.org/rfc/rfc7519.

Karantaidou, I. and Baldimtsi, F. (2021). Efficient constructions of pairing based accumulators. https://ia.cr/2021/638.

Khovratovich, D., Lodder, M., and Parra, C. (2022). Anonymous credentials with type-3 revocation, version 0.6. https://github.com/hyperledger/ursa-docs/tree/main/specs/anoncreds1.

Larisch, J., Choffnes, D., Levin, D., Maggs, B. M., and Mislove, Alan a nd Wilson, C. (2017). CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. https://ieeexplore.ieee.org/document/7958597.

Li, J., Li, N., and Xue, R. (2007). Universal accumulators with efficient nonmembership proofs. https://link.springer.com/chapter/10.1007/978-3-540-72738-5_17.

Looker, T., Bastian, P., and Bormann, C. (2025). Token Status List. https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/10/. Work in Progress.

M'Raihi, D., Rydell, J., Pei, M., and Machani, S. (2011). TOTP: Time-Based One-Time Password Algorithm. https://www.rfc-editor.org/info/rfc6238.

Nguyen, L. (2005). Accumulators from bilinear pairings and applications. https://link.springer.com/chapter/10.1007/978-3-540-30574-3_19.

Russ, H. and Paul, H. (1999). Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. https://www.rfc-editor.org/rfc/rfc2585.

Russ Housley, Tim Polk, W. S. F. and Solo, D. (2002). Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. https://www.rfc-editor.org/rfc/rfc3280.html.

Vitto, G. and Biryukov, A. (2022). Dynamic universal accumulator with batch update over bilinear groups. https://ia.cr/2020/777.