An Untraceable Credential Revocation Approach Based on a Novel Merkle Tree Accumulator

Author x
- -
————@—. -

Author x
- -
————@—. -

Author x
- -
————@—. -

Author x
- -
————@—. -

Author x
- -
————@—. -

***Abstract*—As digital identity gains increasing importance, current centralized digital identity systems face significant limitations. Recent years have seen a shift towards decentralized and self-sovereign identity models, underpinned by verifiable credentials and cryptographic signatures. A critical challenge in these systems is credential revocation and updating, presenting unique issues absent in centralized systems. Traditional approaches, like Certificate Revocation Lists, risk reintroducing centralization into revocation mechanisms. In this context, blockchain systems offer a viable support for implementing revocation mechanisms without compromising the advantages of decentralized identity. However, the use of public, permissionless blockchains introduces challenges, particularly concerning data publication costs and privacy. This work proposes a credential revocation approach utilizing a Merkle tree-based accumulator. Our accumulator enables a trade-off between anonymity and proof complexity, offers resistance to credential tracking, and relies solely on hash function collision resistance, making it quantum-safe. Additionally, we have integrated the accumulator within a broader identity management library and report its performance on standard hardware.**

***Index Terms*—Revocation, Identity Management Systems, Self-Sovereign Identity, Privacy and Scalability.**

I. Introduction

In today's digital age, identity has become a fundamental aspect of daily experience. Currently, the most widespread identity models are based on protocols like OAuth [Har12], where one or more identity providers vouch for an individual's identity at a service provider. In this model, if Bob wishes to use Alice's application, he is redirected to Carol's service, which identifies Bob and then vouches for his identity to Alice. This architecture offers a smoother user experience compared to models that require each application to independently identify Bob, but it comes with several limitations:

- The identity provider is a single point of failure and is privy to all services where a user identifies themselves.
- The range of information for which an identity provider can vouch is limited (for example, Google can vouch for the control of an email inbox, but not for the possession of a driving license).
- Identity providers can arbitrarily limit their services to certain applications.

An alternative and emerging model is the so-called "Self-Sovereign Identity" (SSI) [All16], based on identifiers, digital signatures, and credentials. In this model, users are represented by identifiers often linked to cryptographic keys. Anyone can act as an issuer, issuing a signed document (a credential) attesting to something (e.g., that Bob has earned a degree). The credential is then handed over to a holder, who can subsequently present it to a verifier who verifies its origin through the signature.

The SSI model, and particularly some perspectives representing its usage, envision a world where credentials are widely used, and anyone should be able to act as an issuer. However, a key requirement in any identity model is the ability to revoke or amend assertions [BT10]. While this is not a significant issue in centralized or federated systems, where the identity or attribute provider is always queried during identification, it becomes more complex in credential-based systems. Here, the verifier usually checks the credential in the possession of the holder, which, of course, is unaffected by any subsequent revocation.

The situation is analogous to X509 certificates [Ger+97] in Public Key Infrastructure (PKI), where a certificate linked to a trust chain up to a recognized authority is correctly verified. In X509, revocation is addressed through Certificate Revocation Lists (CRL) [Ger+97], Online Certificate Status Protocol (OCSP) [Mye+99], or Certificate Transparency (CT) [Sch+18].

The burden on the issuer in these approaches includes maintaining high uptime and managing query loads effectively. This is problematic in a context where anyone might wish to act as an issuer of information or credentials. For instance, if a university decides to issue digital degree credentials, must it maintain a CRL service with high uptime and infrastructure capable of handling extensive queries?

Ideally, issuers would require infrastructures similar to those of identity providers. A potential solution could involve major players offering centralized services for CRL publication, but this introduces obvious centralization issues. Blockchain offers a potential solution: anyone can publish authenticated (digitally signed) information on a blockchain, which then makes it available in a distributed, uncensorable, tamper-proof, and automatically versioned manner.

However, directly publishing a CRL on a decentralized blockchain like Ethereum can be problematic, as such blockchains tend to discourage large data uploads by making them expensive. A viable solution might involve publishing only a commitment of the current revocation list on the blockchain, and then requiring the issuer to present a validity proof to the verifier relative to the current commitment.

Cryptographic accumulators are an effective tool to implement this type of architecture. Generally, accumulators are cryptographic protocols that allow for the representation of set elements in a compact structure (the accumulator) and enable selected actors (e.g., holders of a certain credential) to generate easily verifiable validity proofs.

A potential issue with this revocation scheme is traceability: if the set of valid proofs is public, a verifier, after assessing the validity of a certain credential c, could "track" the fate of the credential and gain information about its validity (and thus about its holder).

In this context, our contribution is a state-of-the-art trace-resistant accumulator that enables a trade-off between the effort required from the issuer to generate a proof and the bits of anonymity that protect it. Requiring only hash functions that are collision and preimage resistant, this accumulator also features quantum resistance, positioning it as a tool for credential revocation within the Self-Sovereign Identity model. In this paper, we propose a credential validity representation based on Merkle trees (MT) that offers an efficient revocation scheme, users' privacy preservation, data storage and computation minimization for credential issuers or validating authorities. Also, the scheme provides protection against certificate's status tracking, and minimal data exchange to generate and verify a non-revocation proof. This data structure can be deployed for centralized traditional IDMS, distributed systems and BC-based SSI solutions; we consider a SSI use case in this paper as a proof of concept, however the deployment options are not limited to BC-based identity. The major contributions of this work are the following:

1) We provide a background on the state of the art of IDM systems, BC-based IDMS, Self-sovereign identity and how revocation has been addressed in these environments, as well as discussing the limitations and challenges of currently deployed approaches.
2) We design a novel efficient accumulator, dedicated to deal with revocation problems using a Merkle tree architecture. The approach minimizes the necessary public storage, eliminates extra communication exchanges to update proofs and provides privacy against traceability of certificates status. We also provide a security analysis of our revocation protocol for conformity with standard properties and protection against known attacks.
3) We further implement a working experimental prototype as a proof of concept, for a SSI IDMS use case, describing and simulating the workflow sequences between different actors. A performance evaluation of the obtained results is discussed, proving the feasibility and applicability of our proposal.

The remainder of this output is organized as follows. Section II provides backgrounds on IDMS, blockchain, self-sovereign identity and revocation schemes. In section III, a state of the art on revocation methods is presented, focusing on recent revocation applications in different fields. The detailed process of our proposal is described in section IV followed by a security analysis of the protocol in section V. We discuss a use case for our protocol and conclude with observations and future work discussions in section VII and section VIII respectively.

II. Background

A. Identity management systems

An efficient digital IDentity Management System (IDMS) provides Individuals with a mechanism that allows them to identify and authenticate themselves in order to communicate with each other, in a fast, secure and privacy preserving manner. IDMS can be classified into four categories [All16]: 1) Centralized Identity: The Control of digital identities is completely reserved to centralized authorities. 2) Federated Identity: Users can access multiple sites and services provided by different authorities using only a single digital ID, and identification documents are administrated and federated by multiple authorities. 3) User Centric Identity: An individual can use personal authentication devices (PADs), typically smartphones and tablets without the need for federal authority, making the user able to control and be at the center of all his identity operations; subsequently, adding a layer of user consent. 4) Self sovereignty Identity: A model where users have control over the management of their IDs, we discuss this model in more details in the following sub sections.

B. Credential revocation

In a system based on credentials, it may be necessary to revoke or invalidate a credential for various reasons: compromise of part of the cryptographic material, termination or modification of the conditions that justified the issuance of the credential in the first place (for example, think of the withdrawal of a driving licence), or even simply to correct compilation errors. Both users and verifiers are considered untrusted entities and shouldn't be able to change a revocation status of a certificate, as that should be a prerogative of the issuer. A proof of validity (non revocation) must be provided to or be accessible by the user (if he doesn't have the ability to generate it himself). The cost of a scheme must be defined in the size of certificate validity proof and its verification time, which must be small and efficiently verifiable [ALO98]. As mentioned in the Introduction (I), the two protocols typically used on the Internet are CRLs and OCSP, but because of their limitations many vendors chose to push limited sets of revocations to clients through software updates, creating different new constraints and multiple delays as a result of updates' frequency and shortcomings in the ability to cover all revocation instances [KC21].

C. Blockchains as Distributed Processing Systems

A blockchain [Hin17] can abstractly be viewed as a distributed processing system characterized by the following attributes:

- Maintains a specific state at all times.
- Possesses a defined and immutable transition function.
- Permits anyone to propose transactions at any moment.

- Employs a distributed protocol to determine the order of transactions, without the dominance of any single entity.
- Ensures that the established order is stable and immutable within certain extensive limits.
- In permissionless blockchains, the transaction order selection (consensus protocol) is open to all, contingent on the availability of generally accessible resources (tokens in proof of stake and energy and hardware in proof of work), rather than identity.
- Allows direct access to the current system state, achieved through consensus protocol transactions, without reliance on intermediaries.
- Adaptable to various applications, from establishing credible monetary policy assets like Bitcoin to facilitating credibly neutral coordination mechanisms such as decentralized autonomous organizations (DAOs).

In the realm of identity systems, blockchains can enable diverse functionalities and solve specific challenges. Particularly in credential revocation, they offer a mechanism for issuers to verifiably and accessibly communicate the current validity of the credentials issued. However, many existing blockchains discourage the publication of large data quantities in the state through memory-proportional fees, for reasons tied to decentralization [Bur+19]. The mechanism proposed in this paper aims to harness the accessibility and verifiability of data on a blockchain system, while minimizing the amount of data exposed in its state.

D. Self-Sovereign Identity

According to the classification in [All16], Self-Sovereign Identity (SSI) represents the fourth evolutionary stage in Identity Management Systems (IDMS). In SSI, individuals possess full sovereignty over their digital identity documents. They maintain complete control over who can access their personal information and who cannot. This model empowers individuals to store and manage their data autonomously, free from external authorities' interruptions and interventions.

1) SSI Components:

The main actors in a SSI system are:

- Issuers: They represent authorities and trusted organizations that can create and assign verifiable credentials (VC), update them or revoke their credibility when necessary.
- Holders: They are the users or clients that own a decentralized identity, they can request credentials from issuers and use them to authenticate their rights to access certain services.
- Verifiers: They are generally service providers (SP), they verify the validity of verifiable credentials presented by holders to access services. This verification process is supposed to require few interactions between the verifiers and the identity holders through the BC network.

The relevant terms in a Blockchain-based SSI IDMS are [SC22]:

- Claim: Any assertion representing an information about the ID holder.
- Credential: Collection of claims made about an ID holder by an Issuer (A Certified trusted entity)
- Verifiable Credential (VC): A data structure or a document signed by the issuer (or cryptographically bounded as proof of integrity) proving that the credentials presented by the holder to the verifier are authentic and correct. Can include revocation information (list or method).
- Verifiable Presentation (VP): Usually a brief pointer redirecting towards the physical storage address of the VC, or a signed set of VCs required by a verifier.
- Decentralized Identifiers (DIDs): Public elements that serve to identify and represent real individuals. They are associated with methods that define specific details like the underlying encryption protocols; Furthermore, each DID must be unique and associated with key pairs.

2) Revocation of Verifiable Credentials:

Traditional methods used in centralized systems can not be adopted by DLT-based IDMS due to communication and computation thresholds; Thus, the following approaches have been explored:

- Forward Revocation lists: An issuer generates validity proofs for each VC, and stores them on a public list; a verifier checks if an entry on the list matches a credential in order to consider it valid.
- Status lists: An issuer creates a data structure on a smart contract that has a status variable set to false by default, when a VC is revoked, he must send uploads to set the revocation status to true.
- Signature lists: Issuer proactively uploads a list of signatures of non revoked users.
- Verification data registries: An issuer generates an empty list of revoked VC, when he revokes a credential, he adds its identifier (e.g. an hash of the credential) value to the list. Verifier must check if the credential is present in the registry (similar approach to signature lists).
- Cryptographic Accumulators: Valid VCs are added to a constant-size accumulator, a user must prove that his VC is still contained in the accumulators. Conversely, with revocation accumulators, verifiers can check non-inclusion proofs.

List based approaches are troublesome even in traditional centralized systems; in a decentralized BC setup, the computation and communication loads can easily become prohibitively costly.

III. Related work

Many researchers have demonstrated interest in acknowledging the issues with the current IDMS systems and revocation methods. Authors of [Che+23] propose a DID protocol to solve centralized identity management down time problems in online social networks; the protocol changes the W3C identity model and achieves revocation through dynamic accumulators with range proof and anonymous credentials to prove membership. Schumm et al. [SMP23] proposed a revocation framework using cryptographic accumulators, that does not require witness updates, by way of epochs where the accumulator changes state; this solution combines static accumulators and probabilistic sub-accumulators such as Bloom filter. In [Yan+23], they deal with the scalability problem of handling large-scale vehicle-generated pseudonyms, by deploying a whitelisting method for pseudonym revocation through a dynamic accumulator tree structure based on bilinear mapping. In [Yu+20], authors proposed a selective revocation approach using dynamic accumulators with digital signatures. [Xu+23] introduces a dual-policy revocable attribute-based encryption for secure data sharing with dynamic user groups; revocation is managed with a tree-based data structure to reduce the revocation overhead from linear to logarithmic, where the set of nodes are non-revoked data users that will be used to derive key-updating material.

The authors in [Tes+23] present a DLT-based revocation scheme for vehicular ad-hoc networks; when an authority revokes misbehaving vehicles, revocation information is stored on the ledger through a zero-value transaction (costing only transfer fee). Moreover, a dedicated unit verifies a certificate's status through the transaction's history sent by the issuing authority, and considers it valid if no matching transaction is found. Mundhe et al. [Mun+23] discuss the burden of CRLs for Vehicular ad-hoc networks authentication, and propose a hybrid BC-based conditional privacy preserving authentication scheme where transactions are only issuance or revocation of an ID; a revoked ID can be verified by computing the root hash of a Merkle Patricia tree (MPT) that contains valid IDs in its leaves and comparing it with the identity root stored in the recent block. [Gar+23] and [DGG23] provide revocation for IoT applications using blockchain's smart contracts and wallets, by updating a list of records indexed with the wallets' addresses on the contract. In [Tak+23], authors implement secure revocation by revoking public keys of digital identities; an authority generates non-revocation proofs and associate them with timestamps and zero knowledge range proofs. Li et al. [LH23] achieve rapid ID verification by constructing certificate revocation ledgers rather than using CRLs or centralized VAs; a trusted authority can revoke a malicious entity through a smart contract that sets its certificate's status on the ledger to revoked.

The solution proposed in [Shi+23] supports threshold ID tracing and threshold credential revocation; A number of tracers must collaborate to store registration and revocation information on a BC ledger, securely using threshold signatures. Authors of [Den+23] exploited a sorted user binary tree where each node has a serial number to achieve attribute-based and user-based revocation; instead of sharing a CRL, the protocol extracts the nodes that only have non-revoked credentials on their children nodes or a leaf pointing to one. This improves the computation overhead of verifying a revocation list, since an abstracted smaller serial-based list is investigated. [Yue+18] uses complete sub-tree method in order to create a revocation token that the authority can regularly update for each group member; the verifier only needs to verify the signer's revocation token without obtaining the latest revocation list. A similar work [Yue+22] presents a scheme that uses a PS (Pointcheval Sanders [PS16]) signature method for key management and combines it with sub-trees for revocation; authors also made use of a broadcast encryption framework named NNL [NNL01], which serves to send messages from a broadcast center to a group of receivers and to restrict the decryption ability of some receivers, ; this combination allowed them to use NNL cipher-text as a revocation list where each valid owner must apply for a revocation to declare his validity in an epoch.

This section is centered around novel state of the art proposals, and did not include applications that are widely in use despite their limitations and costs; the solution that we propose focuses on both efficiency and privacy for traditional systems and decentralized settings. Our approach eliminates traceability of revocation status, in fact even after sharing a a credential with a verifier, this latter can not use a previous proof to have knowledge if the status has changed or not, unless he obtains the consent of the user. Based on our investigations, we have not found any similar work with the same revocation objectives, namely protecting customer privacy while ensuring efficiency.

IV. Proposed Revocation solution

In the design of our accumulator, we partition credentials into 2^k anonymity sets, where k is a parameter that can be adjusted based on the application's requirements. Each credential within these sets is identified by a unique ID. Alongside this, each credential is associated with a specific *salt*, known exclusively to the issuer and the holder of that particular credential.

For each anonymity set, the issuer generates a general nonce. Utilizing this nonce, the credential IDs, and their corresponding salts, the issuer then computes *representatives* for each valid credential within the set. These representatives are crucial in maintaining the integrity and confidentiality of the credentials. A holder, possessing their credential's salt and ID, can independently calculate the representative for their credential. If this representative is present in the corresponding set, the holder can then create an inclusion proof to present to a verifier.

In the remainder of this section we provide details on the structure, operations and properties of the structure.

A. Roles and functionalities

In accordance with the actions compose the issuance and use of credentials in a self-sovereign identity context defined in the II-D1 section, the functions offered by the accumulator according to the different roles are:

- *Issuer*: Has the authority to modify the status of a credential by initially adding it and subsequently removing it from the set of valid credentials.
- *Holder*: Possesses the capability to generate proof of the validity or invalidity of a cred issued to them.

Algorithm 1: Initialize Accumulator

```
Result: Establishes an empty accumulator
1 $\Gamma \leftarrow$ empty state
2 $\kappa \leftarrow$ initialize the anonymity sets parameter
3 for $j < 2^\kappa$ do
4     $\gamma_j \leftarrow$ random nonce
5     Calculate $\Sigma_j$ and $\rho_j$ accordingly
6 end
```

- *Verifier*: Can authenticate a proof produced by a holder, relying solely on a compact representation (accumulator) of the valid credentials set.
- *Privacy* (Non-Traceability of Status): A proof which confirms the validity of a cred with a specific identifier at time t does not allow for subsequent tracking of the credential's status, i.e., it does not facilitate checking its validity at a later time when the accumulator is updated.

B. Data structure and algorithms

The accumulator is depicted through various data structures and parameters, each with distinct visibility levels:

- Γ: A set containing tuples $< id, seed_{id}, valid_{id} >$ for every credential issued. Each credential is denoted by an id (either a hash of the credential or a unique identifier generated and signed by the issuer). The issuer pairs every credential */id* with a $seed_{id}$, utilized to generate pseudo-random nonces linked with credentials. Γ is confidential and maintained privately by the issuer. $seed_{id}$ is also known to the holder of the credential identified by id. $valid_{id} \in \{true, false\}$ indicating the current status of the credential.
- κ: A number used to identify the number of bits from h'_{id} that are used to identify a credential's anonymity set, with $h'_{id} = H(id||seed_{id})$, and H a collision resistant hashing function.
- Σ_j: A set of *representatives* c_i corresponding to the valid credentials. Each representative is calculated by the issuer as $c_{id} = H(id||w_{id})$ for every id with $valid_{id} = true$, where $w_{id} = H(seed_{id}||\gamma_j)$ and j the corresponding anonymity set.
- γ_j: A public nonce, established by the issuer and periodically modified for each anonymity set j. Both Σ_j and γ_j are public and stored on cost-effective, available, decentralized and uncensorable platforms (e.g., BitTorrent, InterPlanetary File System (IPFS)).
- ρ_j: The Merkle root of the set Σ_j. It is derived from Σ_j, considering the elements c_i as leaves, belonging to anonymity set j.

C. Operations on the accumulator

1) Accumulator Initialization: As detailed in Algorithm 1, the initialization of the accumulator sets up its state and parameters.

Algorithm 2: Add Credential

```
Input: Credential ID id
Output: Updated accumulator state with the added
        credential
1 $seed_{id} \leftarrow$ RandomValue()
2 $valid_{id} \leftarrow$ true
3 $\Gamma \leftarrow \Gamma \cup \{\langle id, seed_{id}, valid_{id} \rangle\}$
4 $k \leftarrow$ IdentifySet($\kappa, h'_{id}$)
5 rehash($k$)
6 Recalculate $\Sigma_k$ and $\rho_k$
```

Algorithm 3: Revoke Credential

```
Input: Credential ID id
Output: Updated accumulator state with the revoked
        credential
1 $valid_{id} \leftarrow$ false
2 $k \leftarrow$ IdentifySet($\kappa, h'_{id}$)
3 rehash($k$)
4 Recalculate $\Sigma_k$ and $\rho_k$
```

2) Adding a Credential: Algorithm 2 details the procedure for incorporating a new credential into the accumulator. This process adds a credential *id* to the set of valid credentials, updating the accumulator's state.

3) Revoking a Credential: Algorithm 3 describes the process to revoke the validity of a credential identified by *id* in the accumulator.

4) Rehashing a Set: The 'rehash' operation, as used in the revocation process, is detailed in Algorithm 4. It involves modifying the nonce of the corresponding anonymity set to reflect the change in its state.

5) Generating Proof of Validity: Algorithm 5 outlines the process for a holder, with knowledge of $seed_{id}$, to create a proof of validity for a credential.

Algorithm 4: Rehash Anonymity Set

```
Input: Anonymity set index $k$
Output: Updated anonymity set with modified nonce
1 Modify $\gamma_k$ for anonymity set $k$
```

Algorithm 5: Generate Proof of Validity

```
Input: Credential ID id
Output: Proof of validity for the credential
1 $k \leftarrow$ IdentifySet($\kappa, h'_{id}$)
2 $w_{id} \leftarrow H(seed_{id} \,||\, \gamma_k)$
3 $c_{id} \leftarrow H(id \,||\, w_{id})$
4 if $c_{id} \in \Sigma_k$ then
5     $\pi_{id} \leftarrow$ GenerateMerkleProof($c_{id}, \rho_k$)
6     return $\pi_{id}$, $w_{id}$
7 end
```

Algorithm 6: Verify Proof of Validity

```
Input: Credential ID id, Anonymity set index $k$,
       Witness $w_{id}$, Proof $\pi_{id}$
Output: Validation result of the proof
1 $c_{id} \leftarrow H(id \,||\, w_{id})$
2 return ValidateMerkleProof($c_{id}, \pi_{id}, \rho_k$)
```

6) Verifying Proof of Validity: Algorithm 6 describes the verification process for a proof of validity for a credential, ensuring its authenticity.

D. Properties

Completeness: If $valid_{id} = true$, a holder can invariably generate a validity proof π_{id} for id. Note that the holder must access Σ_k to do this; if the issuer withholds this data, the holder cannot create π_{id}. This, however, does not compromise completeness, as the holder always has the option to revoke the validity of id. Nonetheless, the availability of $\Sigma_{i,(i\in[1..2^\kappa])}$ depends on the infrastructure where it's hosted (e.g., a cloud-based HTTP server), potentially introducing a single point of failure. This risk can be effectively mitigated using distributed storage solutions like IPFS, where any stakeholder interested in the system's functionality can store and make the data available (in IPFS terms, they can *pin* the data). Additionally, the issuer might be obligated to keep Σ_i accessible for a certain period, using services like Ethereum blobs, which offer guaranteed data availability at low costs for limited time spans [But+22].

Soundness: If $valid_{id} = false$, then $c_{id} \notin \Sigma_k$ with k the corresponding anonymity set. Therefore, assuming the hash function used to compute ρ_k is collision-resistant, the holder cannot create a valid inclusion proof.

Non-Traceability: If a verifier is aware of a witness w_{id} computed for a specific γ_k, they cannot use it to compute a new witness w'_{id} for the same id with a different γ'_k. This is because the verifier cannot deduce $seed_{id}$ from $w_{id} = H(seed_{id}||\gamma_k)$. Consequently, they cannot calculate the new commitment c'_{id}, nor check whether $c'_{id} \in \Sigma'_k$.

E. Credential Issuance

Initially, an issuer $\mathcal{I}$ generates a Merkle Tree (MT) accumulator structure by invoking the `Init()` function. To issue a credential for a user $\mathcal{U}$, $\mathcal{I}$ uses `add(id)` to incorporate $\mathcal{U}$'s credential into the accumulator. $\mathcal{I}$ then securely sends the credential document and $seed_{id}$ back to $\mathcal{U}$. $\mathcal{I}$ publishes $\{\kappa, \gamma_j, \Sigma_j, \rho_j \quad (\text{with } j \in [1..2^\kappa])\}$ on a selected efficient public storage. ρ_i can be stored in a smart contract for SSI systems, ensuring revocation information integrity and ease of updating, referred to as Off-chain storage with blockchain-based persistent access.

F. Credential Verification

To generate the proof of a credential *id* owned by $\mathcal{U}$, $\mathcal{U}$ first acquires the relevant γ_k from $\mathcal{I}$'s public storage, then invokes `generateProof(id)`. If *id* is not revoked, $\mathcal{U}$ successfully generates $\{\pi_{id}, w_{id}\}$ and sends an authentication request with $\{$*id*, Credential, w_{id}, $\pi_{id}\}$ to a verifier $\mathcal{V}$. $\mathcal{V}$ computes the relevant anonymity set number k using $h'_{id} = H(id \,||\, seed_{id})$, and then executes `verifyProof(id, k, w_id, π_id)`.

G. Credential Revocation

The Issuer $\mathcal{I}$ initiates credential revocation by executing `revoke(id)`, which effectively removes *id* from the corresponding Merkle Tree (MT) and updates the relevant Σ_k and γ_k in its public storage. Subsequently, $\mathcal{I}$ uploads the updated ρ_k to a smart contract, if necessary, to ensure the persistence and integrity of the revocation information.

H. Complexity Analysis of the Accumulator

This section delves into the complexity analysis of the key operations in the proposed accumulator, to understand the performance and scalability of the system.

1) Adding a Credential to an Anonymity Set: Consider an anonymity set of size $n = \frac{N}{2^k}$, in the average case in which the N credentials are uniformly distributed over the anonimity sets

- Worst-case scenario (e.g., when using a linked list for the set): $O(n)$, which is the number of credentials in the set
- Recalculation of the Merkle root: $O(n\log(n))$.

2) Generating a Proof: The process involves:

- Locating the *representative* in the anonymity set, which is $O(n)$ for an unordered set and $O(\log(n))$ for an ordered set enabling binary search.
- Computing the proof: $O(n\log(n))$.

3) Verifying a Proof: The verification of a proof is computationally efficient:

- Verification complexity: $O(\log(n))$.

This complexity analysis provides an objective assessment of the accumulator's performance in terms of scalability, pertinent to credential management systems.

V. Security Analysis

The security of this solution relies on the hardness of the cryptographic primitives that we keep open for developers, such as the discrete logarithm assumption. In this analysis, we consider an adversary model where an attacker can intercept communications between different actors or compromise encryption keys.

A. Non-forgeability

An attacker must modify the MT accumulator to unrevoke a cred *id*, which is only possible when possessing every random $seed_i$, and a nonce γ_k of the correct anonymity set k. Because editing a leaf causes an update of a list of credentials and the MT root hash. Moreover, if the accumulator is published on a blockchain, this property is true by default, since only the issuer will be able to modify the accumulator by signing a transaction that interacts with a smart contract.

B. Non-repudiation

A user holding *id* accesses a service maliciously and immediately sends a revocation request to the issuer, then he denies committing any malevolent action. This claim can only be valid if the user loses the cred *id*, its proof π_{id}, its commitment c_{id} and his encryption key pairs, which in turn requires the exposure of the witness w_{id}. Thus, as long as one of the latter is secured, the non-repudiation property holds. Moreover, a successful attack doesn't require the permanent revocation of a cred, a change of the anonymity set nonce γ_k will invalid any compromised proof π_{id}, since a new proof requires the knowledge of $seed_{id}$.

C. Resistance to common Identity attacks

This approach also provides security against the following well-known attacks:

- Impersonation: The objective is to steal a cred *id* of a user and access services; possessing *id* doesn't allow an attacker to generate pi_{id} without $h'_{id} = H(id||seed_{id})$ and the witness $w_{id} = H(seed_{id}||\gamma_k)$ to identify the corresponding anonymity set k. Namely, a stolen private key and cred *id* does are not enough to authenticate. Also, the possession of the list of w_m and a list of archived proofs π_m used during the previous *m* updates allows an issuer to recognize the legitimate owner and relink his new key pairs with *id*.
- Man-In-The-Middle: This attack requires the interception and modification of exchanges between the system's actors, however transactions can only be signed with the use of private keys. Assuming on of these keys got compromised, can a MIM attack bypass our revocation scheme? Since a commitment c_i is exchanged when issuing *id*, an attacker must target the interactions between *id*'s holder and the issuer during the issuance process in order to generate a correct proof. However, the attacker can not succeed because he can not provide a valid proof that satisfies the legitimate user. Hence, a MIM attack would not be completed without detection, since the user will not have the ability to prove a new credential; besides, an additional hand shake between issuers and users would be enough to protect against it.
- Denial of access: An adversary would send a false revocation request to deny access to the legitimate user, this requires him to eavesdrop the communication and capture a previous revocation request message. Our revocation system can demand proving the knowledge of $seed_{id}$ for a cred *id* by providing a proof π_{id}. Therefore, the system is protected if the encryption key of the user was not compromised; if so, the adversary can hijack or generate a legitimate revocation requests. An update of the nonce γ_k prior the validation of a request protects against this attack, since a new π_{id} will be needed and can only be generated with w_{id} and γ_k.

VI. Use Case

We consider an SSI framework as a proof of concept that supports the creation, resolution and update of DIDs; it allows the creation, revocation and verification of VCs. Reducing computation and communication costs is more imperative for a BC network, our MT accumulator provides improvements of greater importance in a decentralized configuration.

The issuing of a VC takes place exactly how we described in section IV, however in the SSI setup there is no CA and every participating member must have an Ethereum blockchain account, ie: a public address and key pairs. All the exchanged signatures and communications are encrypted using the Ethereum keys. The issuer must send the MT root hash to a smart contract every time the tree is updated.

The verification process described in Fig.1 shows a scenario where the verifier initializes an authentication process after a user requests a service, this latter must provide the corresponding anonymity set and queries the issuer's public storage for the leaves and the MT root hash. He then computes the proof and sends it to the verifier along with the updated commitment. Verifier can now compare the proof with the value stored on the smart contract and authorize the user. The public storage can be an IPFS managed by the issuer, or any other decentralized storage characterized by high availability.

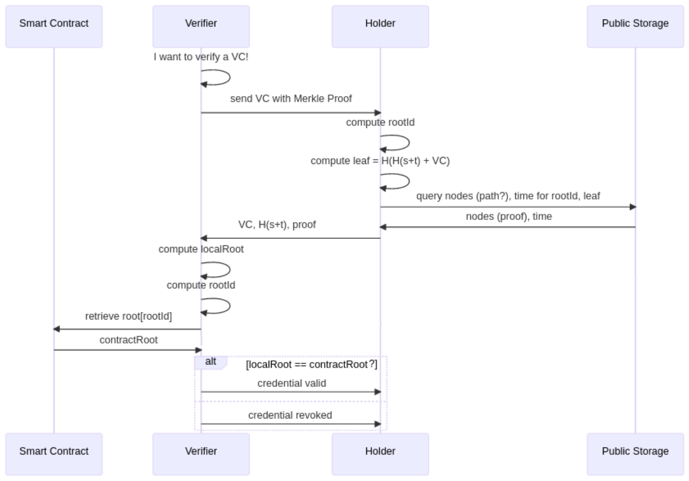

Fig. 1. Merkle Tree Accumulator Verification sequence

As per the revocation process in Fig.2, an issuer is required to remove the credential's commitment from the relevant MT; he generates a new nonce $t2$ for the anonymity set and updates its leaves, recalculates the root hash and uploads it to the smart contract, while applying changes on his public storage.

VII. Performance Evaluation

In this section, we present experiment results from the experiments we conducted to evaluate the feasibility and effectiveness of our approach through a working proof of concept. A library[1]coded in Typescript was developed to model a SSI solution, using a locally-hosted, Ethereum-compatible BC deployed with Hardhat[2]. Our library uses Polygon wallet[3] and binds it with DIDs implemented using the *'did-jwt'* open

[1]Link not added to be conform with the double-blinded peer-review.
[2]N. Foundation. Hardhat, . URL https://hardhat.org/.
[3]P. labs. Polygon. URL https://polygon.technology/.

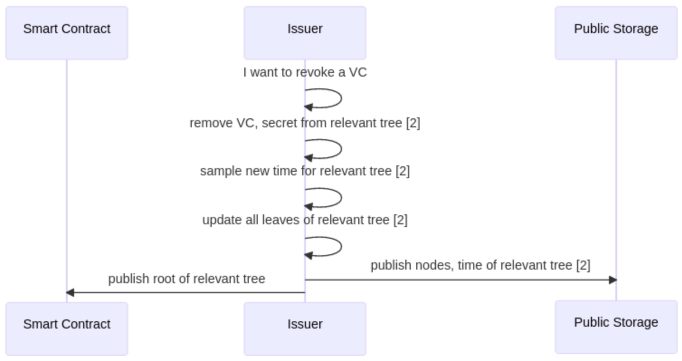
Smart Contract
Issuer
Public Storage
I want to revoke a VC
remove VC, secret from relevant tree [2]
sample new time for relevant tree [2]
update all leaves of relevant tree [2]
publish nodes, time of relevant tree [2]
publish root of relevant tree
Smart Contract
Issuer
Public Storage

Fig. 2. Merkle tree Accumulator Revocation sequence

source library[4], we modified polygon's DID verifiable data registry smart contract and extended it to support our scheme. A large number of credentials were simulated by filling the Merkle Trees with generated data, simulating credentials creation, adding a new leaf to the corresponding MT, updating all the other leaves and publishing them on a public storage, then publishing the MT root on the blockchain. Verification requires a holder to retrieve the relevant Merkle Tree leaves from the public storage, computing the commitment c_i of the cred *id* and the MT proof, and finally sending *id*, MT proof π_{id} and w_{id} to the Verifier who, in turn, recomputes the commitment c_i and verifies the validity of the proof against the root retrieved from the blockchain.

Additionally, a revocation flow includes removing a leaf from its tree, updating the set nonce and all the other leaves, then publishing them on a public storage; and computing and updating the root hash on the smart contract.

Tests were executed on a laptop with the following specification: CPU: i7-8565U, Frequency: 1.8GHz base/ 4.6GHz boost, Cache: 8M, RAM 13 GB available for VM, OS: Ubuntu 22.04.3 LTS Virtual Machine Via VMware 17 player on a Windows 10, Execution environment: NodeJS v20.10.0.

Tab.I showcases results from multiple executions of the protocol with different values for κ and different numbers of of VC per tree. Efficiency of the proof generation is evaluated through the execution time and the size of the data acquired from the public storage; whereas, verification of a VC is evaluated using the execution time and the size of the proof. As expected, the time execution of issuance and revocation operations are independent of κ, sitting at around 300ms for issuance and revocation when leaves of the MTs contain 5k credentials and κ set to 8. Moreover, using the same parameters the proof generation process costs around 40ms and the download of 2.4mb and 50ms were enough to verify it by sending only 3kb of data to the verifier.

We set κ to a smaller value: '2' and increased the number of VC per tree to study the scalability of our solution in case of very large large of credentials; Tab.II shows that issuance and revocation have identical execution times and that the size of the downloaded data to generate the proofs grows linearly with the number of credentials per tree; also, proof generation time increases but with a smaller rate that demonstrates the scalability of the protocol if the exchanged data is managed efficiently.

[4]https://github.com/decentralized-identity/did-jwt.

K	VC per tree	total VC num	Issuance Time	revoke Time	Proof generation (time- size)	verification (time- size)
2	100	410	170ms	180ms	(1ms-0.04mb)	(42ms-1.5kb)
4	100	1.6k	165ms	160ms	(1ms-0.04mb)	(32ms-1.5kb)
8	100	25k	185ms	200ms	(1ms-0.04mb)	(53ms-1.5kb)
2	1k	4k	195ms	175ms	(7ms-0.5mb)	(50ms-3.5kb)
4	1k	16k	235ms	200ms	(7ms-0.5mb)	(50ms-3.5kb)
8	1k	256k	200ms	160ms	(7ms-0.5mb)	(50ms-3kb)
2	5k	20k	300ms	320ms	(35ms-2.4mb)	(38ms-3.5kb)
4	5k	80k	275ms	320ms	(35ms-2.4mb)	(44ms-3kb)
8	5k	1.28m	350ms	320ms	(44ms-2.4mb)	(52ms-3kb)

TABLE I
EXECUTION TIME OF THE SYSTEM'S FUNCTIONS BASED ON VALUE K(ms)

VC per tree	total VC num	Issuance Time	revoke Time	Proof generation (time- size)	verification (time- size)
5k	20k	432ms	444ms	(37ms-2.4mb)	(45ms-3kb)
25k	100k	1800ms	1720ms	(304ms-12mb)	(80ms-3.5kb)
75k	300k	1900ms	1830ms	(400ms-36mb)	(85ms-5kb)
125k	500k	3200ms	3350ms	(730ms-61mb)	(45ms-4kb)
250k	1m	7000ms	7000ms	(1647ms-122mb)	(112ms-3.5kb)

TABLE II
EXECUTION TIME OF THE SYSTEM'S FUNCTIONS BASED ON NUMBER OF CREDENTIALS PER TREE(ms) , K=2

Discussion: As for future challenges and improvements for this protocol, the definition of anonymity sets can be improved using efficient classifiers that can generate balanced sets, since the the size of the MT affects some operations' complexity. Additionally, the time to generate a MT can be improved by defining a function, that can update a tree in a polynomial time, while considering that few features must be guaranteed in order to deploy such approach, like adding serialization to the nodes, or making use of sort options without disclosing the privacy properties preserved by the protocol.

VIII. CONCLUSION

In this paper we presented a novel accumulator data structure based on hierarchical Merkle tree; the protocol solves the state of art limitations and challenges in revocation management for identity management systems. We also discussed how cumbersome traditional strategies are in terms of computational costs and time consumption. Our approach combines Merkle tree features with cryptographic hashing to generate an accumulator data structure that is efficient for updates, it does not require interactions to generate proof of membership and provides privacy protection against traceability, owing the fact that the public information does not include details on the status of a credential. The research on accumulator-based revocation systems shows that this method, even in its early state, is still suitable for SSI and general BC-based IDMS. Scalability concerns can be addressed by introducing efficient methods to create anonymity sets, and by personalizing Merkle tree libraries, which will in turn improve verification of proofs and revocation's overall performance.

References

[Ger+97] Edgardo Gerck et al. "Overview of Certification Systems: x. 509, CA, PGP and SKIP". In: *The Black Hat Briefings* 99 (1997).

[ALO98] William Aiello, Sachin Lodha, and Rafail Ostrovsky. "Fast digital identity revocation". In: *Advances in Cryptology — CRYPTO '98*. Ed. by Hugo Krawczyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–152. ISBN: 978-3-540-68462-6.

[Mye+99] Michael Myers et al. *X. 509 Internet public key infrastructure online certificate status protocol-OCSP*. Tech. rep. 1999.

[NNL01] Dalit Naor, Moni Naor, and Jeff Lotspiech. "Revocation and Tracing Schemes for Stateless Receivers". In: *Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 41–62. ISBN: 978-3-540-44647-7.

[BT10] Elisa Bertino and Kenji Takahashi. *Identity management: Concepts, technologies, and systems*. Artech House, 2010.

[Har12] Dick Hardt. *The OAuth 2.0 authorization framework*. Tech. rep. 2012.

[All16] Christopher Allen. 2016. URL: http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html.

[PS16] David Pointcheval and Olivier Sanders. "Short Randomizable Signatures". In: *Topics in Cryptology - CT-RSA 2016*. Ed. by Kazue Sako. Cham: Springer International Publishing, 2016, pp. 111–126. ISBN: 978-3-319-29485-8.

[Hin17] Michael Nofer; Peter Gomber; Oliver Hinz. "Blockchain". In: 2 (2017), pp. 183–187. URL: https://link.springer.com/article/10.1007/s12599-017-0467-3#citeas.

[Sch+18] Quirin Scheitle et al. "The rise of certificate transparency and its implications on the internet ecosystem". In: *Proceedings of the Internet Measurement Conference 2018*. 2018, pp. 343–349.

[Yue+18] Xiaohan Yue et al. "An Efficient and Secure Anonymous Authentication Scheme for VANETs Based on the Framework of Group Signatures". In: *IEEE Access* 6 (2018), pp. 62584–62600. DOI: 10.1109/ACCESS.2018.2876126.

[Bur+19] Scott P Burger et al. "Why distributed?: A critical review of the tradeoffs between centralized and decentralized resources". In: *IEEE Power and Energy Magazine* 17.2 (2019), pp. 16–24.

[Yu+20] Yong Yu et al. "Blockchain-Based Anonymous Authentication With Selective Revocation for Smart Industrial Applications". In: *IEEE Transactions on Industrial Informatics* 16.5 (2020), pp. 3290–3300. DOI: 10.1109/TII.2019.2944678.

[KC21] Nikita Korzhitskii and Niklas Carlsson. "Revocation Statuses on the Internet". In: *Passive and Active Measurement*. Ed. by Oliver Hohlfeld, Andra Lutu, and Dave Levin. Cham: Springer International Publishing, 2021, pp. 175–191. ISBN: 978-3-030-72582-2.

[But+22] Vitalik Buterin et al. *EIP-4844: Shard Blob Transactions*. https://eips.ethereum.org/EIPS/eip-4844. [Online; accessed 19-December-2023]. Feb. 2022.

[SC22] Frederico Schardong and Ricardo Custódio. "Self-Sovereign Identity: A Systematic Review, Mapping and Taxonomy". In: *Sensors* 22.15 (2022). ISSN: 1424-8220. DOI: 10.3390/s22155641. URL: https://www.mdpi.com/1424-8220/22/15/5641.

[Yue+22] Xiaohan Yue et al. "A practical privacy-preserving communication scheme for CAMs in C-ITS". In: *Journal of Information Security and Applications* 65 (2022), p. 103103. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2021.103103. URL: https://www.sciencedirect.com/science/article/pii/S2214212621002799.

[Che+23] Zhiwei Chen et al. "An efficient revocable identity-based encryption with ciphertext evolution in the cloud-assisted system". In: *CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE* 35.22 (Oct. 2023). ISSN: 1532-0626. DOI: 10.1002/cpe.7735.

[Den+23] Shijie Deng et al. "Flexible revocation in ciphertext-policy attribute-based encryption with verifiable ciphertext delegation". In: *MULTIMEDIA TOOLS AND APPLICATIONS* 82.14 (June 2023), pp. 22251–22274. ISSN: 1380-7501. DOI: 10.1007/s11042-022-13537-0.

[DGG23] Xinrui Duan, Yajun Guo, and Yimin Guo. "Design of anonymous authentication scheme for vehicle fog services using blockchain". In: *WIRELESS NETWORKS* (Aug. 2023). ISSN: 1022-0038. DOI: 10.1007/s11276-023-03471-w.

[Gar+23] Abba Garba et al. "LightCert4IoTs: Blockchain-Based Lightweight Certificates Authentication for IoT Applications". In: *IEEE ACCESS* 11 (2023), pp. 28370–28383. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3259068.

[LH23] Hongzhi Li and Dezhi Han. "Blockchain-assisted secure message authentication with reputation management for VANETs". In: *JOURNAL OF SUPERCOMPUTING* 79.17 (Nov. 2023), pp. 19903–19933. ISSN: 0920-8542. DOI: 10.1007/s11227-023-05394-x.

[Mun+23] Pravin Mundhe et al. "Blockchain-based conditional privacy-preserving authentication scheme in VANETs". In: *MULTIMEDIA TOOLS AND APPLICATIONS* 82.16 (July 2023), pp. 24155–

24179. ISSN: 1380-7501. DOI: 10.1007/s11042-022-14288-8.

[SMP23] Daria Schumm, Rahma Mukta, and Hye-young Paik. “Efficient Credential Revocation Using Cryptographic Accumulators”. In: *2023 IEEE INTERNATIONAL CONFERENCE ON DECENTRALIZED APPLICATIONS AND INFRASTRUCTURES, DAPPS*. IEEE International Conference on Decentralized Applications and Infrastructures. 5th IEEE International Conference on Decentralized Applications and Infrastructures (IEEE DAPPS), Athens, GREECE, JUL 17-20, 2023. IEEE; IEEE Comp Soc. 2023, pp. 127–134. ISBN: 979-8-3503-3535-4. DOI: 10.1109/DAPPS57946.2023.00025.

[Shi+23] Rui Shi et al. “Threshold Attribute-Based Credentials With Redactable Signature”. In: *IEEE TRANSACTIONS ON SERVICES COMPUTING* 16.5 (Sept. 2023), pp. 3751–3765. ISSN: 1939-1374. DOI: 10.1109/TSC.2023.3280914.

[Tak+23] Kazuo Takaragi et al. “Secure Revocation Features in eKYC- Privacy Protection in Central Bank Digital Currency”. In: *IEICE TRANSACTIONS ON FUNDAMENTALS OF ELECTRONICS COMMUNICATIONS AND COMPUTER SCIENCES* E106A.3 (Mar. 2023), pp. 325–332. ISSN: 0916-8508. DOI: 10.1587/transfun.2022CIP0008.

[Tes+23] Andrea Tesei et al. “A transparent distributed ledger-based certificate revocation scheme for VANETs”. In: *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS* 212 (Mar. 2023). ISSN: 1084-8045. DOI: 10.1016/j.jnca.2022.103569.

[Xu+23] Shengmin Xu et al. “A Secure EMR Sharing System With Tamper Resistance and Expressive Access Control”. In: *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING* 20.1 (Jan. 2023), pp. 53–67. ISSN: 1545-5971. DOI: 10.1109/TDSC.2021.3126532.

[Yan+23] Ming Yang et al. “Scalable and auditable self-agent pseudonym management scheme for intelligent transportation systems”. In: *COMPUTERS & ELECTRICAL ENGINEERING* 108 (May 2023). ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2023.108735.