A-PoA: Anonymous Proof of Authorization for Decentralized Identity Management

Jan Lauinger, Jens Ernstberger, Emanuel Regnath, Mohammad Hamad, Sebastian Steinhorst
Technical University of Munich, TUM
Munich, Germany
firstname.lastname@tum.de

***Abstract*—Self-sovereign Identity Management (SSIM) promotes self-control of credentials without relying on external administration. However, the state-of-the-art SSIM based on Decentralized Identifiers and Verifiable Credentials (VCs) defined by the World Wide Web Consortium does not enable credential holders to verify whether a Credential Issuing Authority (CIA) legitimately issued a credential.**

As a remedy, our work constructs a secure authentication protocol, called A-PoA, to provide decentralized and anonymous authorization of CIAs. We leverage a cryptographic accumulator to enable the Root Authority (registering a Credential Schema) with the ability to authorize a CIA (registering a Credential Definition) to issue a credential. The proof of accumulator membership relies on a non-interactive zero-knowledge proof. This allows a credential holder or validator node to verify the validity of a CIA, while the CIA remains anonymous. Our security analysis shows the integrity and confidentiality of our protocol against hostile network participants and our experimental evaluation shows constant verification times independent of the number of authenticated CIAs. Hence, A-PoA introduces the missing building block to develop SSIM-capable and VC-compatible ecosystems acting as a drop-in replacement for traditional Public Key Infrastructure schemes.

***Index Terms*—Authentication, Authorization, Identity & Trust Management, Anonymous Credentials, Verifiable Credentials, RSA-Accumulators, Non-interactive Zero-Knowledge Proof**

I. Introduction

The term Identity Management (IdM) refers to data management around identification, authorization, and authentication of identifiers in any form. In recent years, different forms of IdM have emerged. Central IdM system design maintains the identity of users in a single system and continues to remain vulnerable to the single point of failure pattern [1]. Federated IdM systems distribute data of identities across trusted platforms and provide benefits of exchange and linking of identities. However, such systems enforce trade-offs between transparency, usability, and negatively affect user privacy [2].

More recent solutions target user-centric IdM to keep users or devices in full control of their identity data, removing

This work has received funding by the European Union's Horizon 2020 Research and Innovation Programme through the nIoVe project (https://www.niove.eu/) under grant agreement no. 833742.

With the support of the Technische Universität München - Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement no. 291763.

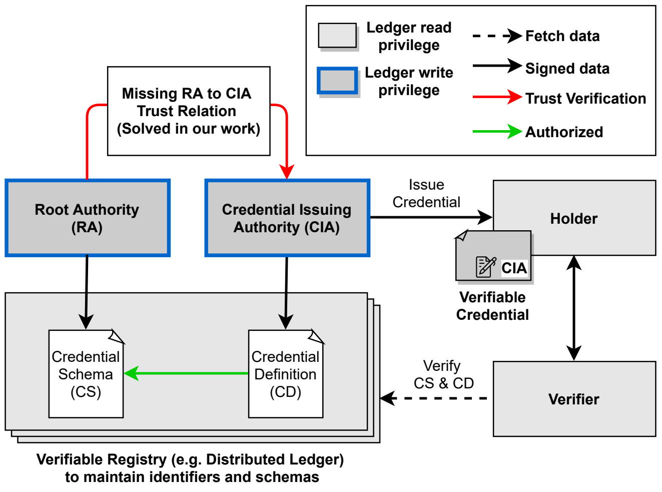

Fig. 1. Presentation of the ecosystem around Verifiable Credentials (VCs), highlighting the missing relation between the Root Authority (RA) (Schema Creator) and the Credential Issuing Authority (CIA) (Definition Creator & Credential Issuer). Our work specifies a protocol for the establishment of a trust relation between RAs and CIAs.

the dependence on third parties. Self-Sovereign Identity Management (SSIM), as a promising approach to user-centric IdM, enforces user control of attributes to bring a new stage of independence of administration services. In combination with a Verifiable Credential (VC), the SSIM scheme enables protection of private information as well as enhanced trust [3].

In the context of the VC ecosystem, a VC relies on a:

- **Credential Schema (CS)**, which specifies the name, version, and attributes that will appear in the credential.
- **Credential Definition (CD)**, which references a CS and specifies cryptographic metadata containing the signatures and data of the Credential Issuing Authority (CIA).

The cryptographic data in a CD will be used for verifying attributes of an issued credential. Hence, a CD enables a third party to cryptographically verify the validity of claims made in a credential by the respective CIA. Therefore, to issue a credential, a CIA must release a CD beforehand (see Fig. 1).

However, as Root Authorities (RAs) and Credential Issuing Authorities (CIAs) have equal privileges to write to the verifiable registry, the credential holder remains unable to verify whether a CIA is authorized to issue a credential. This issue remains an unsolved problem in the VC standardization by the World Wide Web Consortium (W3C) and motivates the remainder of our work.

To clarify the problem with a use case in the automotive domain and referring to Fig. 1, assume a vehicle Original Equipment Manufacturer OEM_i as a Root Authority (RA) (CS issuer). The CS, published by OEM_i, would specify attributes of a credential that is autonomously verified during a software update by On-Board Equipment (OBE). Without an authorization scheme for CIAs, every Software Provider SP_j would be able to register CDs that reference the CS of OEM_i. Thus, every SP_j would be able to issue credentials to Software Distribution Services SDS_k^j (acting as cred. holders). A vehicle (acting as verifier), manufactured by OEM_i, would autonomously verify certificates of every SDS_k^j as valid. This valid verification will cause safety issues for the vehicle if the software update is malicious.

To solve this issue, we propose an authorization scheme, called Anonymous Proof of Authorization (A-PoA), which enables RAs (such as OEM_i) to authorize CIAs (such as contracted SP_j) to issue credentials based on CSs that have been created by RAs. The main feature of our protocol is that A-PoA keeps the relation between RAs and CIAs anonymous to entities that verify authorization authenticity of CIAs. To securely construct this protocol, our A-PoA protocol makes use of the membership verification feature of the cryptographic RSA-accumulator, introduced by Camenisch et al. [4]. The accumulator allows to aggregate elements without revealing individual membership. Additionally, we apply the Non-interactive Zero Knowledge Proof of Knowledge of Exponent (NI-ZKPoKE) construction, introduced by Boneh et al. [5], to hide the accumulator elements in the membership verification phase of our protocol. By disguising the membership authentication with the NI-ZKPoKE, A-PoA does not reveal any structure of accumulator elements, hence, keeping anonymity of authenticated CIAs.

The evaluation of our work considers the security of the protocol and the performance. Concerning security, the requirement of integrity applies throughout all phases of the protocol whereas confidentiality partly applies. The performance evaluation considers dominating computations during the registration, authentication, and revocation phases. With the requirement of fast and scalable verification, A-PoA achieves constant verification times which we assume to happen more frequently compared to registration or revocation operations.

Summed up, the contribution of our work is as follows:

- In Sec. III, we construct a protocol that enables RAs (CS creators) to authorize and revoke certain CIAs (CD creators) to issue credentials.
- Our A-PoA protocol provides efficient verification of CIA authorization authenticity by leveraging succinct proofs (see Sec. III-D).
- Our security analysis proves protocol integrity, confidentiality of specific accumulator parameters, and anonymity of authorized CIAs (see Sec. IV).

Tab. I. Glossary of Notations: (1) Roles, (2) Accumulator Parameters, and (3) Arithmetic Modulo Primes & Composites.

Symbol	Definition
RA_i	Witness issuing (i) Authorities (CS creator)
CIA_h	Witness holder (h) Authorities (CD creator)
H_i	Credential Holders
VN_i	Validator Nodes / Verifiers
CS_i	Credential Schemas
CD_i	Credential Definitions
t	Discrete time / operation counter
X_t	Tails file at time t
X_0	Initial tails file at $t = 0$
x_i	i-th element of the tails file
w_i	Witness value associated with i-th element
a_t	Accumulator value at time t
p, q, p', q'	Large λ-bit prime numbers
$\mathbb{Z}_n$	$n \in \mathbb{N}$, $\mathbb{Z}_n = \{1, 2, \ldots, n\}$ = ring of integers mod n
$\mathbb{Z}_n^*$	Set of invertible elements in $\mathbb{Z}_n$
$\mathbb{G}_?$	Generic group of unknown order $\{(\mathbb{Z}_n)^*/\{\pm 1\}\}$
[-B, B]	Range of integers such that $\lvert\mathbb{G}\rvert/B$ is negligible
$\mathbb{QR}_n$	Subgroup of quadratic residues of $\mathbb{G}_?$, contains $x \in \mathbb{Z}_n^*$, if $\exists\ y \in \mathbb{Z}_n^*$, with $y^2 \equiv x \pmod{n}$
$\phi(n)$	Number of elements in $\mathbb{Z}_n^*$, if $p \cdot q = n$ then $\phi(n) = (p-1) \cdot (q-1)$
g, h	Generator of a Group $\mathbb{G}_?$

- We enable *verifiable* and *anonymous* trust hierarchies of authorities in SSIM-based VC ecosystems.

II. Preliminaries

In this section, we briefly introduce the background of our work. Sec. II-A shows the construction of cryptographic accumulators and their properties. The succeeding Sec. II-B introduces Zero-Knowledge (ZK)-protocols, which make up the basis of the accumulator membership proof of this work. Last, Sec. II-C presents the main concepts of the VC ecosystem, where we apply A-PoA. Tab. I introduces roles, accumulator parameters, and cryptographic notations used throughout our work.

A. Cryptographic Accumulator

One-way cryptographic accumulators, as initially introduced in [6], provide the ability to verify set membership of an element $x_i \in X$, with $i = \{1, 2, \ldots, N\}$, where N is the number of elements within a set X, without revealing individual elements of the set. Throughout this work, we require our accumulator to be dynamic and positive. Dynamic accumulators support additive and subtractive operations which increase and decrease the number of elements of the accumulator respectively [7]. Positive accumulators support proofs of membership.

Among multiple options of accumulator types, we chose the Rivest–Shamir–Adleman (RSA)-accumulator of the work in [4] due to the following reasons:

- An accumulator based on modular exponentiation has minimal storage requirements and *O(1)* verification complexity compared to Merkle-tree accumulators [8, 9].
- The RSA-accumulator can be used in combination with the group $\mathbb{Z}_n^*/\{\pm 1\}$, with the RSA modulus n. More

specifically, it relies on the strong RSA assumption [10], and the hardness of the discrete log problem [11] which makes it applicable to succinct Proof of Knowledge (PoK) of a discrete-log schemes, as introduced in the work [5].

The work in [4] defines the accumulator value as a quadratic residue a_t modulo n at time t, with $n = p * q$ as the RSA modulus. The value a is initialized through the generator $g_{\text{acc}} \in \mathbb{QR}_n$, where $\mathbb{QR}_n$ is the subgroup of quadratic residues of the generic group of unknown order $\mathbb{G}_?$. The security of the RSA-accumulator follows the strong RSA assumption with primes p, q, p', and q', with $p = 2p' + 1$ and $q = 2q' + 1$. Accumulator elements are odd positive prime integers because otherwise, an element could be proven a member of the set of elements even though it is not (exclusion requirement of element divisors). Adding an element $x_i \in X_t$, with X_t = $\{x_1, x_2, ..., x_i\}$, to the accumulator works by calculating $a_{t+1} = a_t^{x_i} \mod n$. The extraction the respective witness w_t for x_i calculates as $w_t = a_t^{X_t \setminus \{x_i\}} \mod n$. A successful verification of the element x_i with its witness w_t as $a = w_t^{x_i}$ mod n equals the latest accumulator value a_{t+1}. Deleting an element x_i from the set of elements in the accumulator requires the knowledge of the factorization of n. The deletion works by calculating $a_{t+1} = a_t^{x_i^{-1} \mod \phi(n)} \mod n$ and updating another witness w_t paired with x calculates as $w_{t+1} = w_t^c a_{t+1}^b$ mod n and relies on the Bezout coefficients b and c, with $bx + cx_i = 1$.

B. Zero Knowledge Proofs

A PoK allows a *prover* P to convince a *verifier* V that P knows the solution of a hard problem. We say that the conversation between P and V is a PoK of relation $\mathcal{R}$ if the properties *Completeness* and *Soundness* hold. If the property *ZK* holds, P can convince V without revealing anything about the solution [12]. Hence, a Zero-Knowledge Proof (ZKP) must fulfill the following properties:

- **Completeness** means that P, knowing the solution, can successfully convince V.
- **Soundness** means that P, not knowing a solution, will fail to convince V.
- **Zero Knowledge** in a PoK scheme requires V to learn nothing but the validity of a convincing assertion of P.

Schnorr et al. [13] introduced one of the initial 3-round/Σ interactive ZK-protocols which proves knowledge of the relation $\mathcal{R} = \{(\alpha, u) \in \mathbb{Z}_q \times \mathbb{G}) : g^\alpha = u\}$. The commitment scheme used in this protocol is hiding only and there exist a *knowledge extractor* as well as *simulator* model to prove *Soundness* and *ZK* respectively. Important to mention is that this protocol assumes Honest Verifier Zero-Knowledge (HVZK) due to the ability of V guessing the challenge during the protocol.

After generalizing the exponentiation relation of the Schnorr protocol to a homomorphism in finite abelian groups in [14], Bangerter et al. apply Schnorr in a group of unknown order [15]. On top of finding more general structures compatible with such Σ-protocols, Boneh et al. constructed the secure and succinct Zero Knowledge Proof of Knowledge of Exponent (ZKPoKE) protocol for the relation $\mathcal{R} = \{(w \in \mathbb{G}_?, \hat{x} \in [n]; x \in \mathbb{Z}) : w = g^x, x \mod n = \hat{x}\}$ that leverages the Pedersen commitment scheme [16] and verifiable delay functions, introduced in [17], which are secure under the adaptive root assumption in groups of unknown order. Their work in [5] provides the respective *knowledge extractor* and *simulator* model to prove *Soundness* and *HVZK*. The Pedersen commitment scheme hides the knowledge statement (Pedersen hiding property) and provides robustness against computationally unbounded provers (Pedersen binding property). With the Schnorr and ZKPoKE protocol being HVZK, it is possible to apply the Fiat-Shamir heuristic to transform the protocols into one-round/non-interactive protocols that provide ZK in the random oracle model [18]. As a result, our work takes the NI-ZKPoKE protocol of the work in [5] to prove membership of an accumulator element.

C. Ecosystem of Verifiable Credentials

Currently, *IRMA* and *Sovrin* count as the two main solutions that satisfy SSIM principles [19]. SSIM is an IdM solution that provides users with full control over their own identities. Considering that an identity consists of multiple identifiers with respective collections of attributes [20], SSIM enables a user to collect and manage identities across different domains while respecting privacy, unlinkability, and selective disclosure.

In this work, we consider the decentralized SSIM approach of *Sovrin*. The activities of *Sovrin* have led to the creation of the open source framework for SSIM, called Hyperledger *Indy* [21]. *Indy* builds upon the W3C standards of Decentralized Identifier (DID)s [22] and VCs [3] to implement the VC ecosystem, as shown in Fig. 1. *Indy* uses permissioned Distributed Ledger Technology (DLT) to implement the verifiable registry. However, it is possible to build a DLT-based SSIM ecosystem by using public blockchains. This would require a redesign to determine RA and CIA ledger write privilege (e.g. voting-based smart contracts). *Sovrin* relies on consortium agreements to contract organizations that run registry nodes (*Stewards*) and authorities with ledger write privilege (*Endorsers*).

The VC ecosystem utilizes DIDs to connect credentials to an entity (VC holder, VC verifier, CIA). DIDs are universally unique identifiers with a mandatory prefix, method, and method specific identifier, each separated by colons (e.g. *did:example:123456789abcdefghi*). Each DID resolves to a DID document which specifies security credentials and other mechanics (associated VCs, claims on the subject, etc.). Authorities with the ability to write to a verifiable registry require to register their DID at the verifiable registry publicly. VC holders do not register their DID at the verifiable registry publicly. To protect their privacy, credential holders can switch between/bind different DIDs to different credentials to obfuscate tracking of their own identity.

To issue a VC, registered authorities (e.g. RA and CIA) need to register a CS and CD at the verifiable registry. Upon reception of a VC which links to a CD and CS, a VC holder is able to prove claims found in the VC in a privacy-preserving manner [23]. VC verifiers leverage the public registry to validate references and signatures.

III. Approach

This section describes the A-PoA protocol which solves the missing/undefined trust relation between privileged/trusted authorities in the VC ecosystem. Sec. III-A provides an overview of A-PoA and introduces different phases around accumulator management. Subsequently, Sec. III-B to III-E go into the details of each protocol phase.

A. A-PoA Protocol Overview

In A-PoA, we associate an element x with a CIA and the accumulator value a_t with a specific CS. Since RAs create CSs, RAs count as accumulator managers. By adding an element x to the accumulator value a_t at time t, RA authorizes a CIA to issue a credential based on the specific CS. During the CIA registration phase, the CIA, as the creator of the element x, receives a witness w. A witness w can be extracted from a_t for every element x in a_t. With that, it is possible to verify the existence of x in a_t by using the respective witness w_t for x. The element-witness pair (x, w) enables the CIA to prove membership of x in a_t, thus, authenticating itself as an authorized authority to issue a credential based on a specific CS, which has been created by the accumulator manager RA. The RA maintains a so-called tails file to monitor authenticated CIAs in form of elements x. The fact that it is possible to remove an element x from the accumulator a_t provides RAs with the ability to revoke authenticated CIAs at any point in time.

The RSA-accumulator we use has been introduced in Sec. II-A. Necessary functions to manage the accumulator can be seen with the first five functions of Fig. 2. The last two functions of Fig. 2 generate and verify the NI-ZKPoKE proof of an element-witness pair (x, w). All functions apply at different stages throughout A-PoA and will be explained in the next sections. To enable anonymous access control of CSs in the VC ecosystem, we divide A-PoA into four main phases:

1) **Schema Registration** (Sec. III-B) requires a RA to publish a CS as well as an associated accumulator to the verifiable registry.
2) **CIA Registration and Authorization** (Sec. III-C) requires the accumulator manager RA to add an element to the accumulator and return a witness.
3) **CIA Authentication** (Sec. III-D) requires the CIA (witness holder) to generate an authentication proof for the CD transaction.
4) **Maintenance** (Sec. III-E) describes incoming addition and revocation updates of the accumulator which affects the authorization status of CIAs.

B. Schema Registration

Before we introduce our approach to authorize CIAs, it is necessary to set up the initial state of the verifiable registry. As we assume our verifiable registry to be permissioned, each node of the verifiable registry may have a number of pre-defined transactions defining the initial pool of network nodes. It is assumed that authorities $\{\mathrm{RA}_1, \mathrm{RA}_2, ..., \mathrm{RA}_i\}$ are initialized on the verifiable registry with writing permission through a set of genesis transactions. Each public entity is initialized through a special transaction on the verifiable registry disclosing their public DID. It is assumed that with each initialized CS_i, created by RA_i, an accumulator a_t is

```
GenAcc( $\lambda, X_0$ ):
    $p' \leftarrow Gen_{\text{prime}}(\lambda)$;        $q' \leftarrow Gen_{\text{prime}}(\lambda)$
    $p \leftarrow 2 \cdot p' + 1$;                $q \leftarrow 2 \cdot q' + 1$
    $g'_{\text{acc}} \overset{R}{\leftarrow} \mathbb{G}_?$;                  $n \leftarrow p \cdot q$
    $g_{\text{acc}} = (g'_{\text{acc}})^2 \mod n$;   $a_t = g_{\text{acc}}^{\prod_{i=0;x \in X_0}^{N} x_i} \mod n$
    return: $a_t$
GenAccElement( $\lambda$ ):
    $x \overset{R}{\leftarrow} \mathbb{Z}$
    return: $H_{\text{prime}}(x, \lambda)$
Add( $a_t, X_t, x_i$ ):
    if $x_i \in X_t$: return: $(X_t, a_t)$
    else:
        $X_{t+1} \leftarrow X_t \cup \{x_i\}$
        $a_{t+1} = a_t^{x_i} \mod n$
        return: $(X_{t+1}, a_{t+1})$
Revoke( $a_t, X_t, x_i$ ):
    if $x_i \notin X_t$: return: $(X_t, a_t)$
    else:
        $a_{t+1} = a_t^{x_i^{-1} \bmod \phi(n)} \mod n$
        $X_{t+1} \leftarrow X_t \setminus \{x_i\}$
        return: $(X_{t+1}, a_{t+1})$
GenWit( $x_i, X_t, g_{\text{acc}}$ ):
    $w_t = g_{\text{acc}}^{\prod(X_t \setminus \{x_i\})} \mod n$
    return: $w_t$
UpdateWit( $w_t, x_i, rev, a_t, x_k^{\text{deleted}}$ ):
    if $rev == true$:
        $\alpha \cdot x_i + \beta \cdot x_k^{\text{deleted}} = 1$;  ($\alpha$, $\beta$ Bezout)
        $w_{t+1} = w_t^{\beta} \cdot a_t^{\alpha}$
        return: $w_{t+1}$
    else:
        $w_{t+1} = w_t^{x_i} \mod n$
        return: $w_{t+1}$
GenProof( $w_x, x, a_t$ ):
    $k, \rho_x, \rho_k \overset{R}{\leftarrow} [-B, B]$;        $z = g^x h^{\rho_x}$
    $A_g = g^k h^{\rho_k}$;                      $A_{w_x} = w_x^k$
    $l \leftarrow H_{\text{prime}}(w_x, a_t, z, A_g, A_{w_x})$;    $c \leftarrow H(l)$
    $q_x \leftarrow \lfloor (k + c \cdot x)/l \rfloor$;        $q_\rho \leftarrow \lfloor (\rho_k + c \cdot \rho_x)/l \rfloor$
    $r_x \leftarrow (k + c \cdot x) \mod l$;  $r_\rho \leftarrow (\rho_k + c \cdot \rho_x) \mod l$
    $\pi \leftarrow \{l, z, g^{q_x} h^{q_\rho}, w_x^{q_x}, r_x, r_\rho\}$
    return: $\pi$
VerifyProof( $w_x, a_t, \pi$ ):
    $\{l, z, Q_g, Q_{w_x}, r_x, r_\rho\} \leftarrow \pi$;          $c = H(l)$
    $A_g \leftarrow Q_g^l g^{r_x} h^{r_\rho} z^{-c}$;        $A_w \leftarrow Q_{w_x}^l w_x^{r_x} a_t^{-c}$
    Verify $r_x, r_\rho \in [l]$;       $l = H_{\text{prime}}(w_x, a_t, z, A_g, A_w)$
    return: $\{0, 1\}$
```

Fig. 2. Pseudocode of the RSA-accumulator generation (*GenAcc*), element generation (*GenAccElement*), addition (*Add*), subtraction (*Revoke*) [7], and witness functions (*GenWit* & *UpdateWit*) [4] together with the verification based on the NI-ZKPoKE protocol (*GenProof* & *VerifyProof*) [5].

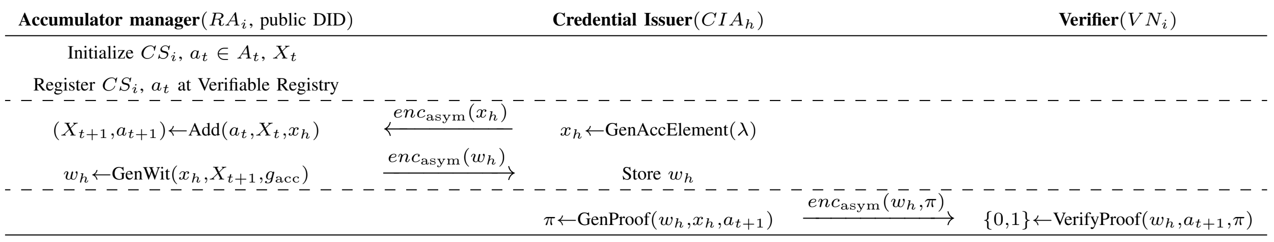
Accumulator manager(RA_i, public DID)
Credential Issuer(CIA_h)
Verifier(VN_i)
Initialize CS_i, $a_t \in A_t$, X_t
Register CS_i, a_t at Verifiable Registry
$(X_{t+1}, a_{t+1}) \leftarrow$Add$(a_t, X_t, x_h)$
$enc_{\text{asym}}(x_h)$
$x_h \leftarrow$GenAccElement(λ)
$w_h \leftarrow$GenWit$(x_h, X_{t+1}, g_{\text{acc}})$
$enc_{\text{asym}}(w_h)$
Store w_h
$\pi \leftarrow$GenProof(w_h, x_h, a_{t+1})
$enc_{\text{asym}}(w_h, \pi)$
$\{0,1\} \leftarrow$VerifyProof(w_h, a_{t+1}, π)

Fig. 3. A-PoA protocol of membership authorization and verification.

generated and registered at an accumulator registry A_t on the verifiable registry (see *GenAcc()* in Fig. 2). Likewise, RA_i creates an empty or initialized tails file X_0 at time $t = 0$. The schema registration protocol can be seen in the first phase of Fig. 3 which stops at the first dotted horizontal line.

C. CIA Registration and Authorization

As the first core part of our work, this section describes the secure generation of an accumulator element x_h and its corresponding witness w_h. The interacting authorities are the witness issuer RA_i (CS_i creator & accumulator manager) and the witness holder CIA_h (CD_i creator & credential issuer). Both witness issuer and holder have the privilege to write to the verifiable registry. We assume that the state of the verifiable registry is initialized as described in Sec. III-B. To achieve collision-resistance among accumulator elements, we utilize a hash function H_{prime} with prime domain. This hash function iteratively hashes an input and a counter to generate λ-bit accumulator elements. Increasing the counter finally creates an output of the hash function that is prime and co-prime to $\phi(n)$ [5]. Note that this function can be the target of timing and side-channel attacks. Thus H_{prime} leverages using dummy computations to prevent such attacks.

The protocol for authorizing a CIA_h to issue credentials based on CS_i (see intermediate phase in Fig. 3) is as follows:

- **GenAccElement:** CIA_h chooses an element $x'_h \in \mathbb{Z}$ at random and calculates a λ-bit $x_h \leftarrow H_{\text{prime}}(x'_h, \lambda)$ (see Fig. 2, line 9). CIA_h sends x_h to RA_i.
- **Add:** RA_i adds x_h to the local tails file X_t and updates the accumulator $(X_{t+1}, a_{t+1}) \leftarrow \mathit{Add}(a_t, X_t, x_h)$. RA_i updates the corresponding accumulator value a_t and writes the updated accumulator value a_{t+1} to the verifiable registry.
- **GenWit:** RA_i calculates a new witness $w_h \leftarrow \mathit{GenWit}(x_h, X_{t+1}, g_{\text{acc}})$ and sends w_h to CIA_h.

As the element x_h will be written into the tails file X_t (granting CIA_h access to reference CS_i) which is saved in the wallet of RA_i, x_h values need to be a pairwise distinct prime values to ensure collision-resistance as described by Barić et al. in [10]. The tails file X_t of RA_i contains a mapping of the DID of CIA_h to the value aggregated in the accumulator a_t.

As RA_i saves X_t in its own private wallet, witnesses will need to be recomputed every time a value is added or deleted from the accumulator. This is why accumulator updates introduce communication overhead as RA_i needs to multicast updated witnesses back to each CIA_h. Sec. III-E provides the details of additive and subtractive accumulator updates. As our focus lies on efficient verification where authorities are assumed to have sufficient computational power in the network, computational overhead is negligible.

D. CIA Authentication

As the second core part of our work, this section introduces authentication of CIA_h. A successful authentication results in authorization of CIA_h to perform CD_i transactions, enabling credential issuing. The third phase of Fig. 3 illustrates this phase of our protocol with the interaction between CIA_h and VN_i, acting as prover and verifier respectively.

When intending to issue credentials, CIA_h creates a CD_i transaction with a valid NI-ZKPoKE proof of knowledge of x_h. The function *GenProof()* of Fig. 2 shows the proof construction which has been developed in [5]. Including the NI-ZKPoKE proof which proves membership of x_h in a_t, a CD_i transaction is sent to the verifiable registry. Here, a validator node VN_i verifies if CIA_h is entitled to issue credentials referencing a CS_i of RA_i. This verification is achieved by verifying the NI-ZKPoKE proof. The NI-ZKPoKE proof does not disclose the actual value x_h and preserves confidentiality of the parameter x_h. Hence, there is no loss of privacy for CIA_h. Line 34 of Fig. 2 indicates how the Pedersen commitment hides x_h. Again, non-interactivity is achieved through leveraging the Fiat-Shamir heuristic [24], which models the unpredictability of the random choices of VN_i through the output of a hash-function. Thus, the protocol for proving membership of x_h in a_t is the following:

- **GenProof:** CIA_h generates the proof $\pi \leftarrow \mathit{GenProof}(w_h, x_h, a_t)$, where $x_h \in X_t$, and sends (w_h, π) to the verifier VN_i.
- **VerifyProof:** VN_i verifies the membership by invoking $\{0, 1\} \leftarrow \mathit{VerifyProof}(w_h, a_t, \pi)$.

Once VN_i verifies the membership of CIA_h, CD_i is written to the verifiable registry, effectively enabling CIA_h to issue credentials based on a CS_i issued by RA_i. An important remark is that the authentication of CIA_h should be based on the NI-ZKPoKE proof only. Likewise, the privilege of a CD_i write transaction should rely on a valid NI-ZKPoKE proof. This means that CIA_h must take a random DID for the communication with VN_i instead of using the publicly known and trusted DID.

E. Maintenance

Updating the witness of an authority has to be done each time a member is added or revoked from the accumulator tails file X_t. The witnesses can only be updated by RA_i. Therefore, the accumulator manager/RA_i sends an update message to all the members that are a part of the updated accumulator a_{t+1}. We distinguish two cases for the witness update protocol:

Addition: *UpdateWit(rev=false)* updates a witness w_t when a member is added to X_t. Therefore, *UpdateWit* adds the new element to the witness w_t, which itself is an accumulator value with an element less compared to the actual accumulator a_t. Addition of elements $\{x_{i+1}, x_{i+2}, ..., x_j\}$ (members of the accumulator) to the tails file $X_t = \{x_1, x_2, ..., x_i\}$ requires execution of $X_{t+1} = X_t \cup \{x_{i+1}, x_{i+2}, ..., x_j\}$. This can be seen in line 13 of Fig. 2. With X_t, the witness owner can calculate each witness by calculating $w_t = g_{\text{acc}}^{\prod(X_{t+1} \setminus \{x_i\})}$ individually. After the updates, each CIA_h receives their updated witness value.

Revocation: With the accumulator scheme, authority RA_i is able to revoke the trust from CIA_h by invoking $(X_{t+1}, a_{t+1}) \leftarrow$ *Revoke*(a_t, X_t, x_h). This removes the element x_h associated to CIA_h from the tails file X_t. Next, a new accumulator value is computed and written to the verifiable registry, effectively revoking the ability for CIA_h to prove its accumulator membership to the verifier VN_i or the credential holder. Additionally, CIA_h looses its ability to further issue credentials. Note that all credentials, which have already been issued during the time where CIA_h was authorized, are only invalidated if the credential contains the proof that allows credential holders to verify the validity (CS authorization) of RA_i.

Efficiently updating the membership witness upon deletion of x_k^{deleted} can be achieved by calling *UpdateWit(rev=true)*. This function call removes x_k^{deleted} from the accumulator by calculating the Bezout coefficients between x_i and x_k^{deleted} as described in [25]. The Bezout coefficients always exist since the domain $\mathbb{Z}_N^*$ of the RSA-accumulator contains odd prime integers only. Thus, the updated membership witness w_{t+1} can be computed such that the Bezout coefficients α and β solve the linear equation $\alpha \cdot x_i + \beta \cdot x_k^{\text{deleted}} = 1$ for $gcd(x_i, x_k^{\text{deleted}}) = 1$. Lines 27 and 28 of Fig. 2 show the calculation of the Bezout coefficients and the witness update. A complete description and correctness proof of the preceding relation is provided in [25].

IV. Security Analysis

The security analysis focuses on protocol integrity and accumulator element confidentiality. It first introduces the adversary model together with accessible system parameters of (1) the accumulator and (2) the communication protocols. The analysis of all parameters references the definitions under use and proves the respective security assumption that apply to the parameters.

We assume a safe accumulator manager that does not share the tails file over the network. Additionally, no hostile network participant is able to forge the private keys associated to DIDs in use. Network communication is secured using authenticated encryption and sessions keys are random to prevent replay attacks.

A. Adversary Model

Throughout the security analysis, we assume to have the following models of adversaries:

- $\mathcal{A}_1$ *(Network Eavesdropper)*: Suppose a hostile network participant, acting as $\mathcal{A}_1$, intends to eavesdrop and modify or decrypt all messages m exchanged throughout the introduced protocols.
- $\mathcal{A}_2$ *(Unforgeability)*: Suppose $\mathcal{A}_2$ is a malicious adversary, trying to forge a valid proof of an invalid identity. $\mathcal{A}_2$'s efforts can be based on previously seen witness pairs (x, w) (only w is known by $\mathcal{A}_2$) and accumulator values a.
- $\mathcal{A}_3$ *(Cheating Verifier)*: Suppose $\mathcal{A}_3$ is a malicious Verifier V that verifies the authentication proofs of a prover P. Then, $\mathcal{A}_3$ does not learn anything else than the validity of the statement proven by P.

Theorem 1. *Suppose authenticated encryption of messages throughout the communication protocols holds, then the authentication scheme introduced above is secure against $\mathcal{A}_1$ intending to eavesdrop and modify or decrypt messages m.*

Proof Supposing an adversary $\mathcal{B}$ that is able to provide $\mathcal{A}_1$ with the private keys (through guessing or collision) associated with the DIDs used during message exchange contradicts the security assumptions of authenticated encryption. Keys used in authenticated encryption are based on asymmetric cryptography which relies on e.g. the strong RSA assumption (introduced below).

Definition 1. *(Strong RSA assumption in generic groups of unknown order [5]) There is no probabilistic polynomial-time algorithm $\mathcal{P}$ that outputs w and an odd prime x such that $g^x \equiv u \pmod{n}$, except with negligible probability:*

$$PR\left[w^x \equiv g : \begin{array}{c} \mathbb{G}_? \xleftarrow{R} Gen(\lambda),\ g \xleftarrow{R} \mathbb{G}_? \\ w, x \in \mathbb{G}_? \times \mathbb{Z} \leftarrow \mathcal{A}_2(\mathbb{G}_?, g) \end{array}\right] < negl(\lambda)$$

Theorem 2. *Under the strong RSA assumption, the above mentioned RSA accumulator used in our scheme is secure against $\mathcal{A}_2$ and forging of membership.*

Proof In order to prove security against adversary $\mathcal{A}_2$, suppose there exists an adversary $\mathcal{B}$ that is able to find a collision, hence obtaining $\{x_1, x_2, ..., x_n, x', a'\}$ such that $(a')^{x'} = g^{x_1, ..., x_n} \pmod{n}$. $\mathcal{A}_2$ is given access to all public values initiated in the generation phase. Leveraging $\mathcal{B}$, $\mathcal{A}_2$ can break the RSA assumption as described in [10]. Let $x = x'$ and $r = x_1, ..., x_n$. Compute $\alpha, \beta \in \mathbb{Z}$ as Bezout coefficients for $\alpha \cdot r + \beta \cdot x' = 1$. Set $w = (a')^{\alpha} g^{\beta}$ satisfying $w^x \equiv g$. This indicates that if a value x_i is once revoked from the accumulator, the entity authorized before is unable to efficiently obtain membership without authentication.

Tab. II. Mean execution times (ms) of A-PoA with a 2048-Bit RSA-accumulator ($\lambda = 128$), k=50 elements, and 128-Bit hashes.

Protocol	Function	Time (ms)	COM	Big $\mathcal{O}$
Authorization	Prime Gen.	309.81	k	$\mathcal{O}(n)$
	Acc. Gen.	88.80	1	$\mathcal{O}(1)$
	Wit. Gen.	4383.60	k	$\mathcal{O}(n)$
Authentication	GenProof	40.06	1	$\mathcal{O}(1)$
	VerifyProof	23.42	0	
Revocation	Acc. Revoke	2244.85	(0-k)	$\mathcal{O}(n)$
$\sum$**Authorization**	N/A	4782.21	$2k+1$	$\mathcal{O}(n)$
$\sum$**Authentication**	N/A	63.48	1	$\mathcal{O}(1)$
$\sum$**Revocation**	N/A	2244.85	(0-(k-1))	$\mathcal{O}(n)$

Theorem 3. *Our scheme is secure against $\mathcal{A}_3$, if the one-time interaction between P and V can be simulated through an efficient simulator $\mathcal{S}$ and the Fiat-Shamir heuristic applies.*

Proof (Sketch). In order to prove security against $\mathcal{A}_3$, we rely on the simulator introduced by Boneh et al. in [5]. It is argued that the *GenProof* function can be efficiently simulated by $\mathcal{S}$, outputting $\{z', Q'_g, Q'_{w_x}, r'_x, r'_\rho\}$ which is statistically indistinguishable from the real protocol/transcript execution $\{z, Q_g, Q_{w_x}, r_x, r_\rho\}$. Hence, transcript T_{sim} generated through $\mathcal{S}$ is indistinguishable from T_{real}, satisfying the HVZK property for a proof of knowledge as described in Sec. II-B. Note that the NI-ZKPoKE is general zero-knowledge in the random oracle through the Fiat-Shamir heuristic. The reason for this is the replacement of the challenge with a hash which prevents $\mathcal{A}_3$ from guessing the challenge in advance.

V. Evaluation

This evaluation considers three aspects of our A-PoA construction: (1) aggregated protocol times, (2) scalability, and (3) privacy. After introducing hardware specifications and selected Bit-sizes, we consider protocol performance and scalability in Sec. V-A and privacy in Sec. V-B.

A. Timing Behavior of the RSA-Accumulator and Protocols

The performance evaluation has been conducted using a Lenovo Thinkpad T480s with 16 GB of RAM and a 1.90 GHz Intel(R) Core(TM) i7-8650U CPU and we used Python to implement A-PoA. All values collected during the evaluation of the accumulator data structure and protocols faced 100 repetitions and subsequent averaging to reduce deviations of the results.

Concerning the selection of parameters, our work takes accumulators with an RSA modulus n with 1024-Bit and 2048-Bit sizes, requiring $\lambda = 80$-Bit and $\lambda = 128$-Bit primes respectively. The sizes of the parameters of the modulus n and primes do not only affect the accumulator functions, but the NI-ZKPoKE protocol and its proof size, calculating as $\pi_{\text{size}} = 3 \cdot n_{\text{size-bit}} + 3 \cdot \text{Hash}_{\text{size-bit}}$. This means that taking the generic group of unknown order $\mathbb{G}_? = \mathbb{Z}_n^* \setminus \{\pm 1\}$ with a 2048-Bit modulus n and a 128-Bit hash-function (H_{prime}, H), the NI-ZKPoKE proof size π_{size} calculates to 861 Bytes for the

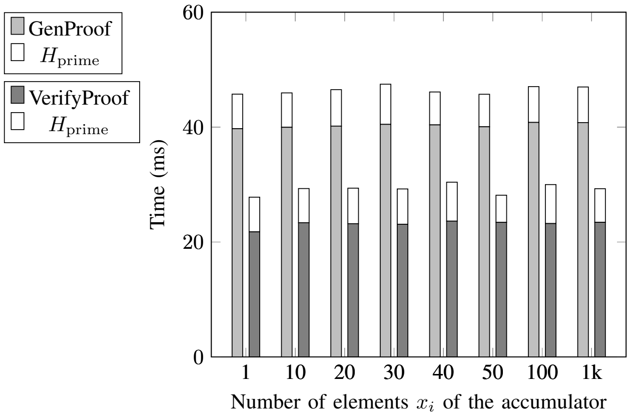

Fig. 4. GenProof (left, lightgray) and VerifyProof (right, gray) execution times (ms) of the NI-ZKPoKE protocol with 128-Bit polynomial time H_{prime} hash function and the RSA-accumulator (2048-Bit).

2048-Bit RSA-accumulator. The formula of π_{size} derives from the ZK proof with parameters $\{l, z, Q_g, Q_{w_x}, r_x, r_\rho\}$, where z, Q_g, and Q_{w_x} depend on the modulo n calculation (2048-Bit n), l depends on the output size of H_{prime} which causes r_x and r_ρ (remainders) to remain below l. Switching the modulo size of the RSA-accumulator and the hash output size affects π_{size} accordingly.

Protocol Times - Tab. II shows execution times of functions of the authorization, authentication, and revocation protocols for managing access to a CS. Additionally, the last column provides the communication overhead presented by each function. The evaluation of protocol times is based on an accumulator with $k = 50$ elements x_i. This decision enables comparison to the mean computation times of the protocols (revocation, verification) introduced by Hölzl et al. in [26] which we further discuss in Sec. VI. Regarding the communication overhead of 50 CIAs participating in the protocol requires 50 messages of each CIA_h to the accumulator manager RA_i, 1 message of RA_i to the verifiable registry, and again, 50 messages to communicate each witness w_h back to every CIA_h. The communication complexity of the revocation protocol depends on the application and the decision whether to notify CIAs. Due to the non-interactivity feature of the NI-ZKPoKE protocol, the authenticity verification is efficient with a single message.

Proof Verification - An evaluation of different numbers of elements aggregated in the accumulator with respect to the corresponding verification and proof creation time can be found in Fig. 4. On average and without considering H_{prime} (volatility caused by non-deterministic behavior), the proof verification is 42% faster than the proof creation for the 2048-Bit RSA modulus n with a hash size of 128-Bit and a security value $\lambda = 128$. We assume that the accumulator value has already been fetched from the verifiable registry and the verification is restricted to evaluating the *VerifyProof()* algorithm introduced in Fig. 2. The values depicted in Fig. 4 are averaged over 100 executions. Notably, the verification speed is constant ($O(1)$) for constant system parameters and

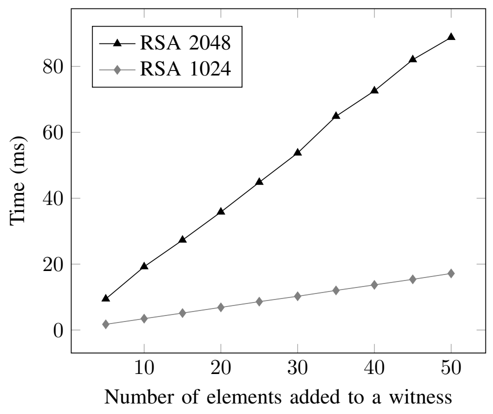

Fig. 5. Duration (ms) of adding elements x_i to an already existing witness w_t for a single holder witness update (numbers averaged by 100 repetitions).

does not depend on the number of elements aggregated in the accumulator. The times of executing H_{prime} vary and the optimization of this function goes beyond the scope of this work.

Witness Update - In our construction, the accumulator manager RA authorizes CIAs to access CSs through broadcasting witness updates back to authorized CIAs. The efficiency of the witness update operation depends on the number of elements that are added to or deleted from the accumulator at a given point in time. Addition of 10 elements to the witness of an authorized entity takes around $20\,\text{ms}$ with a 2048-Bit RSA modulus n and around $3.5\,\text{ms}$ by using a 1024-Bit RSA modulus n (see Fig. 5). Addition and deletion of CIAs does not occur very often as CIAs itself maintain their issued certificates at least the expiry time (90 days validity of Let's Encrypt certificates) [27]. Nevertheless, the communication overhead in A-PoA is a bottleneck that could be solved by having CIAs update their witnesses on their own or by having them rely on third parties. With our anonymity requirement, outsourcing of protocol computations is not possible and the bottleneck remains. However, leveraging more complex accumulators such as the *Braavos* accumulator of the work in [7] do not require updates upon addition of elements (but has weaker security guarantees) and we consider the usage of such accumulators as future work.

B. Privacy

Resulting from specifications of A-PoA, no entity except of the accumulator manager/RA and the authenticated CIA can reveal or track information of the CIA witness pair (x, w). The reason for this is the confidential communication between the RA and CIA authorities as well as the accumulator data structure itself. The aggregated elements in the RSA-accumulator are secure under the discrete logarithm problem and, by respecting the strong RSA assumption, face negligible collision chances. When proving membership of a CIA to an accumulator, the NI-ZKPoKE proof hides the accumulator element without revealing any structure of it. It is important to notice that the CIA must constantly switch its DIDs, even when communicating to the same entity periodically to prevent information tracking. This way, the authorized/privileged CIA remains anonymous.

VI. Related Work

The work of Hölzl et al. [28] leverages a so-called disposable dynamic accumulator in the context of a pseudonym-based signature scheme where pseudonyms are represented by tokens. The accumulator data structure of their work proves the validity of these tokens which make up the accumulator elements. In a similar way to our validity management of authorities per CS, the Electronic Identity (*eID*) issuer in [28] creates a single accumulator per secure element which handles credential storage. One-time verification tokens as pseudonyms and accumulator elements establish the privacy-preserving functionality, whereas in our work, accumulator computations and ZK-proofs provide anonymity.

The evaluation of the work of Hölzl et al. considers generation, binding, verification, and update times of a RSA-accumulator in the context of mobile *eID* management. Similar to their work, our authorization and revocation methods increase/decrease linearly depending on the number of elements in use. Fig. 5 shows this behavior for a changing number of elements x_i. By contrast, our execution times of the NI-ZKPoKE proof generation and verification remain constant (see Fig. 4). Our NI-ZKPoKE verification times are up to 25% faster compared to the verification times in the work [28].

In [29], Reyzin et al. utilize asynchronous accumulators with backwards compatibility to build a distributed Public Key Infrastructure (PKI). In their work, the accumulator is used to reference and store public keys of users. The witness value and the public key allow validity checking of security key pairs. However, compared to our work and with regard to a PKI, our concept is able to leverage hierarchies and anonymity proofs to manage user credentials with absolute privacy and user-centric control of credentials.

VII. Conclusion

Our secure A-PoA protocol enables RA authorities (CS issuers) with the ability to authorize or revoke CIAs (CD issuers) to reference CSs of the RAs. Hence, A-PoA enables verifiable hierarchies between trusted authorities in a SSIM-based VC-compatible ecosystem. Our scheme is based on a RSA-accumulator, providing authorization based on the accumulator membership, and a privacy-preserving NI-ZKPoKE protocol, which proves that an entity is member of a certain accumulator without revealing any information about the membership relation. The NI-ZKPoKE protocol enables resource-efficient and succinct verification without additional communication overhead. The verification time is constant and does not depend on the number of elements aggregated in the accumulator.

References

[1] M. Isaac and S. Frenkel, "Facebook security breach exposes accounts of 50 million users," *The New York Times*, vol. 28, 2018.

[2] P. Dunphy and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20–29, 2018.

[3] M. Sporny, D. Longley, and D. Chadwick, "Verifiable credentials data model 1.0," *W3C, W3C Candidate Recommendation, March*, 2019.

[4] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Annual International Cryptology Conference*. Springer, 2002, pp. 61–76.

[5] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to iops and stateless blockchains," in *Annual International Cryptology Conference*. Springer, 2019, pp. 561–586.

[6] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1993, pp. 274–285.

[7] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 301–315.

[8] M. P. Jhanwar and P. R. Tiwari, "Trading accumulation size for witness size: A Merkle tree based universal accumulator via subset differences." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1186, 2019.

[9] J. Lapon, M. Kohlweiss, B. De Decker, and V. Naessens, "Performance analysis of accumulator-based revocation mechanisms," in *IFIP International Information Security Conference*. Springer, 2010, pp. 289–301.

[10] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1997, pp. 480–494.

[11] K. S. McCurley, "The discrete logarithm problem," in *Proc. of Symp. in Applied Math*, vol. 42. USA, 1990, pp. 49–74.

[12] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

[13] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Conference on the Theory and Application of Cryptology*. Springer, 1989, pp. 239–252.

[14] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[15] E. Bangerter, J. Camenisch, and S. Krenn, "Efficiency limitations for σ-protocols for group homomorphisms," in *Theory of Cryptography Conference*. Springer, 2010.

[16] L. Chen and T. P. Pedersen, "New group signature schemes," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1994, pp. 171–181.

[17] B. Wesolowski, "Efficient verifiable delay functions," *Journal of Cryptology*, pp. 1–35, 2020.

[18] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *Journal of the ACM (JACM)*, vol. 51, no. 4, pp. 557–594, 2004.

[19] J. C. Nauta and R. Joosten, "Self-sovereign identity: A comparison of irma and sovrin," Technical Report TNO2019R11011, Tech. Rep., 2019.

[20] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, 2016.

[21] T. L. Foundation, "Hyperledger Indy Project," https://www.hyperledger.org/projects/hyperledger-indy, [Online; accessed 12-December-2020].

[22] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized identifiers (dids) v1. 0," *Draft Community Group Report*, 2020.

[23] M. Chase, C. Ganesh, and P. Mohassel, "Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials," in *Annual International Cryptology Conference*. Springer, 2016, pp. 499–530.

[24] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.

[25] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *International Conference on Applied Cryptography and Network Security*. Springer, 2007, pp. 253–269.

[26] M. Hölzl, M. Roland, O. Mir, and R. Mayrhofer, "Bridging the gap in privacy-preserving revocation: practical and scalable revocation of mobile eIDs," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1601–1609.

[27] C. M. Bruhner and O. Linnarsson, "Relay racing with x. 509 mayflies: An analysis of certificate replacements and validity periods in https certificate logs," 2020.

[28] M. Hölzl, M. Roland, O. Mir, and R. Mayrhofer, "Disposable dynamic accumulators: toward practical privacy-preserving mobile eids with scalable revocation," *International Journal of Information Security*, pp. 1–17, 2019.

[29] L. Reyzin and S. Yakoubov, "Efficient asynchronous accumulators for distributed PKI," in *International Conference on Security and Cryptography for Networks*. Springer, 2016, pp. 292–309.