



Privacy-Preserving eID Derivation to Self-Sovereign Identity Systems with Offline Revocation

Andreas Abraham  Karl Koch, Stefan More 
Graz University of Technology
Graz, Austria
{andreas.abraham,karl.koch,stefan.more}@iaik.tugraz.at

Sebastian Ramacher 
AIT Austrian Institute of Technology
Vienna, Austria
sebastian.ramacher@ait.ac.at

Miha Stopar 
XLAB d.o.o.
Ljubljana, Slovenia
miha.stopar@xlab.si

Abstract—Digital identities play a vital role in an increasingly digital world. These identities often rely on central authorities to issue and manage them. Central authorities have the drawback of being a central trusted party, representing a bottleneck and single point of failure with exclusive control of identity-related data. Self-sovereign identity (SSI) tackles those problems by utilizing distributed ledger technology and making users the sovereign owners of their identity data. Nevertheless, SSI, as recent technology, still lacks qualified identity data. This is especially a problem since sensitive services like eGovernment or banking services require identity data issued by a qualified identity provider; thus, SSI-based identities cannot be used for these services.

In this paper, we propose a concept for deriving identity data from an existing identity system into an SSI in a fully privacy-preserving way by additionally supporting offline verification. This way, we enable a chain of trust from the existing identity system to the SSI system by introducing a novel trust model. Our concept utilizes novel cryptographic primitives to support efficient and privacy-preserving identity showing as well as revocation. To underline the feasibility of our concept, we implement a proof system and benchmark the related use cases.

Index Terms—self-sovereign identity, eID derivation, offline revocation, zero-knowledge proofs

I. INTRODUCTION

When users want to access a resource at a service provider (SP), they first need to prove their identity. Those SPs commonly use digital identities issued by government identity providers (IdPs) or commercial solutions provided by online services to identify and authenticate their users. A drawback of such centralized systems is the increased risk of data exposure and leakage [9], [35], [10]. Additionally, since IdPs hold a large set of sensitive personal data, their systems are an attractive target for attacks and cyberwar [19]. In recent years, decentralized identity models gained traction, resulting in models such as self-sovereign identities (SSIs) [2], [30]. Systems following the SSI model do not store sensitive identity data in centralized data silos but instead put the identity data in the hands of the users. To authenticate credentials and distribute identities, SSI uses distributed ledgers (DLs).

When SPs offer access to sensitive data or services, they require a strong proof of the user’s real identity and therefore

have additional requirements for the digital identities they accept. Providing such identities is one of the main ideas of qualified IdPs, which issue identities that provide strong assurance about the identity attributes. In contrast, in the SSI model anyone can issue statements in the form of credentials about everyone. While this is a strength of SSI and increases its flexibility, it means that SSIs are not qualified identities, which represents a disadvantage for the use cases requiring strong identity proofs.

A solution to enrich SSIs with qualified attributes is to derive SSIs from existing qualified identities – an obvious candidate is the eID issued by governments. This derivation process involves the transformation of an eID data schema into an SSI data schema while maintaining the trustworthiness of the attributes. Existing solutions [5] propose a decentralized derivation system which uses the DL’s nodes to ensure that the transformation was performed correctly. While in some solutions the nodes of the DL need to access the plaintext of the sensitive identity attributes, more recent concepts [3] use privacy-preserving mechanisms to not reveal any sensitive attributes to the derivation system and enable selectively sharing of some attributes without revealing the others. At the same time, revocation of credentials needs to be supported which is often an issue when building privacy-preserving systems. While previous work [4] introduces concepts to revoke SSIs and present them to verifiers in an offline environment, the revocation is not fully privacy-preserving.

In this paper, we introduce a novel approach to derive an SSI credential from an existing identity in a privacy-preserving way. Using our approach, users can selectively disclose attributes and prove the validity status of their credentials to an offline verifier. Additionally, users can revoke credentials without revealing their identity.

Contribution 1: Privacy-Preserving Derivation and Credential Showing. Our system enables users to derive an SSI credential from an existing eID, and DL nodes to attest that this SSI credential was indeed derived from that eID. In contrast to the existing techniques, the attestation step does not reveal any of the attributes to the DL nodes. We achieve this by employing non-interactive-zero-knowledge (NIZK) proof systems, and in particular succinct non-interactive arguments of knowledge (SNARKs), to prove the authenticity of the original credential and that both credentials contain the same

attributes. This proof is verified by the DL nodes, which together sign an attestation statement and send it back to the user. Since the proof is constructed using the IdP’s identity assertion, the attestation process can be used with existing IdPs and does not require *any modifications* to the IdP, which increases the adaptability of our approach. We also enable users to reveal only parts of their SSI credential to a verifier. This is again achieved by constructing another proof, this time of the derived SSI credential and the attestation statement. When combining these two steps, our system facilitates the use of attributes originating from a qualified source in the SSI world while retaining the users’ privacy.

Contribution 2: Privacy-Preserving Revocation of Credentials. Additionally, we enable revocation of credentials in a privacy-preserving way with a revocation registry stored on the DL. By using cryptographic accumulators users can revoke their credentials in this registry without revealing any personal information. During the showing of a credential, the user provides a (non-)revocation witness to the verifier. By doing so we additionally enable verifiers to verify the validity status of a credential while being offline. To enable privacy-preserving revocation, we do not send the revocation accumulator witness directly but instead extend the statements in the proofs for the credentials with the revocation status statement.

Contribution 3: Implementation and Evaluation. To show the feasibility of our approach, we provide a proof-of-concept implementation based on Groth’s SNARK [22] and demonstrate its practicability by providing a performance evaluation. We make no assumption on the signature schemes employed by the IdPs. We have the flexibility of choosing schemes for showings, which results in significantly faster circuits for showings. While these circuits become larger and slower to evaluate as the number of credentials in the system increases, we apply techniques from privacy-preserving retrieval of witnesses in Certificate Transparency [25] to keep the overheads in check.

II. PRELIMINARIES

This sections recalls background information regarding the used building blocks.

Self-Sovereign Identity. Self-sovereign identity (SSI) is a recent identity model, which arose with emergence of the blockchain and distributed ledger technology (DLT). SSI was described as further evolvement of the user-centric identity model [37] identified in [2]. In contrast to the user-centric model, SSI aims to address the central trusted party by utilizing a distributed ledger (DL) and to give the users full control over their identity data. Figure 1 illustrates a high-level overview of the basic components of an SSI system including main flows between the actors. The user in the SSI system wants to use their digital identity, stored in the user’s domain e.g., the mobile phone, to authenticate themselves towards the verifier. The user receives beforehand their identity data from a so-called issuer. The DL consists of nodes hosting a copy of the ledger and utilizes a certain consensus mechanism to determine what is appended to the ledger. Commonly, SSI

systems utilize consortium or permissioned DL in which semi-trusted organizations like universities or companies participate as nodes. Additionally, the DL serves as decentralized public key infrastructure (DPKI) addressing the central trusted party. Importantly, no sensitive user information are stored on the DL but rather off-ledger.

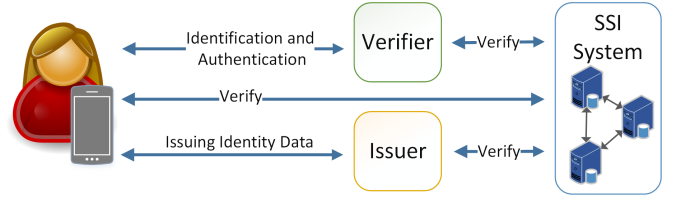


Figure 1. High-level architecture of an SSI system including basic components and process flows

Decentralized Identifier. Decentralized Identifier (DID) [36] defines one of the main building blocks of SSI. DIDs were specified with the main objective to enable and support SSIs. The main advantage of these type of identifier is that they can be created without relying on a central trusted party. DIDs, e.g., `did:method:1234567890ABCDEFGH`, consists of three parts: `did` describes the type of identifier, which is DID in this case. `method` describes the ledger on which the DID is registered and to which it resolves to. The last part, `1234567890ABCDEFGH` in our case, defines a unique identifier within `method`. This unique identifier can be a random number or a public key or parts of it. This DID resolves to the so-called DID document on the related DL. The DID document contains public information like endpoints or public keys.

Verifiable Credential. Besides DIDs, the verifiable credential (VC) [34] forms a data format in the SSI processes. VCs preferably utilize JSON, which represents a light-weight and machine-readable data format to represent and transport data. VCs can include various attributes related to a user starting with basic information like name and date of birth up to complex use cases such as a passport or driver license.

Byzantine Fault Tolerance Consensus. DL-based systems utilize consensus protocols to agree upon what data are written to the ledger. These consensus mechanisms depend on the use-case and on the used DL (public or consortium/permissioned). In the SSI case, mainly consortium or permissioned ledgers are used, thus, no expensive proof-of-work, or similar is necessary. Instead, non-public DLs take advantage of protocols like the Byzantine-fault-tolerance (BFT) protocol [15] based on the eponymous problem [29]. Such a BFT protocol is used to detect faulty or malicious nodes in a network with nodes of the same permissions. Generally, BFT protocols can handle $f = \lfloor (N - 1)/3 \rfloor$ misbehaving nodes out of N nodes.

Accumulator-based Revocation. Accumulators were first introduced by Benaloh and de Mare [8] and allow one to accumulate a finite set \mathcal{X} into a succinct value. For every element in this set, membership witnesses can be computed efficiently yet it should be computationally infeasible to find a membership

witness for non-accumulated values. Accumulators facilitate privacy-preserving revocation mechanisms, which is especially relevant for privacy-friendly authentication mechanisms like group signatures [13] and credential systems [14] such as Sovrin [28] and Hyperledger Indy.¹ Following ideas of [20], the credentials contain a unique revocation attribute, i_R . Each user obtains a membership witness proving that their i_R is contained in an accumulator. On revocation, the corresponding i_R gets removed from the accumulator and hence verification of the revocation attribute no longer succeeds.

Proof Systems. Non-interactive zero-knowledge proof systems (NIZKs) allow a prover to convince the verifier of truthfulness of some statement without revealing secret witnesses. Since the seminal work of Groth [22], efficient and succinct NIZKs in the form of SNARKs and STARKs gained popularity both in academia and practice. Frameworks such as Bellman² provide convenient methods to transform arbitrary circuits into rank-1 constraint systems and then proofs of the underlying system. We refer to Appendix A for formal definitions.

III. RELATED WORK

This section introduces the related work in the field of identity data derivation.

The National Institute of Standards and Technology (NIST) published already in 2014 a guideline for derived personal identity verification (PIV) credentials [32]. This publication specifies the derivation process in which the user, which is in possession of a PIV card, can derive a PIV credential onto her mobile phone. A similar way to derive identity data was introduced by the H2020 project Aries³. In contrast, Aries focuses more on using biometric authentication means to strengthen the binding of user to the related identity data. The LIGHTest project⁴ has its main focus on a cross-domain trust infrastructure that combines different trust domains and provides verification of electronic transactions across national borders. This project also proposes an eID-derivation schema [27], [26] where digital identities are being created based on national IDs like passports. Nevertheless, these concepts introduce a central trusted party that is responsible for the derivation process.

This work tackles this problem by proposing an identity-derivation concept without relying on a central trusted party.

Addressing the central trusted party was already the focus of [2]. In this work, Abraham et al. introduced a way how to derive identity data into a different identity system and without relying on a central trusted party by utilizing a distributed ledger (DL). The nodes in this network are semi-trusted and performing the derivation process. They also counter check their derivation result to ensure correctness. The trust is distributed to the nodes in the network. Even though this work solved fundamental trust issues, privacy concerns arose since all nodes had access to the identity data in plain.

¹<https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0011-cred-revocation/README.html>, accessed on 2021-04-15

²<https://github.com/zkcrypto/bellman>, accessed on 2021-04-15

³<https://www.aries-project.eu/>, accessed on 2021-04-15.

⁴<https://www.lightest.eu>, accessed on 2021-04-15

This privacy problem was addressed by [3]. The proposed concept in this paper tackled the privacy concerns of the previous work of the identity derivation process also without relying on a central trusted party. The privacy concerns were addressed by utilizing a non-interactive-zero-knowledge (NIZK) proof system. This work solved the trust issues raised by the previous work but other problems occurred. Revocation is a critical part in a privacy-preserving setting and was fully addressed. Furthermore, the utilized proof system was not efficient enough w.r.t computation time.

In contrast, this work tackles these issues by utilizing a novel proof system, based on succinct non-interactive arguments of knowledge (SNARKs), that offers efficient proving and verification algorithms as well as constant-size proofs. Additionally, we also add a privacy-preserving revocation mechanism based on accumulators.

The offline revocation was mentioned in [4] and achieved by introducing an attestation that proves when the last revocation check was performed. Based on this attestation, a verifier could decide if the identity data with the related attestation is fresh enough to accept it.

In contrast, this work improves on the privacy aspect of the offline revocation since the approach in [4] could leak metadata.

IV. CONCEPT

In this section, we discuss the general idea of our concept. We start by introducing the involved actors who interact with each other. Next, we give an overview of the stages of importing identity attributes into the SSI system, requesting an attestation at the ledger, and how a prover uses their imported data to prove some attributes about their identity. We also illustrate the process in which a prover can revoke their identity credential and attestation in a privacy-preserving way.

A. Actors

The actors involved in our proposed concept are introduced below and illustrated in Figure 2 considering both the source IdM system as well as the target SSI system.

Prover. A prover represents a user that wants to use their existing eID in an SSI system to authenticate towards service providers (SPs). Additionally, the prover is able to revoke their identity data if the device where the data are stored is lost or stolen or when the data are invalid or not used anymore.

Identity Wallet. The identity wallet describes hardware and software in the prover's domain used to store and manage identity data and key material of the prover. Additionally, the wallet is also responsible for creating the proofs.

Identity Provider. At the existing IdM system, an IdP provides the identity data of a related prover used to be imported into the SSI system by utilizing our concept.

SSI System. The SSI system consists of semi-trusted nodes hosting a consortium ledger and using a consensus protocol to decide what is written to the ledger. The SSI system serves as decentralized public key infrastructure (DPKI) as well as

attesting network. Additionally, the SSI system also stores the revocation list.

Verifier. A verifier represents either an SP where the prover requests access to a resource or service, or a person like a police officer during a police check where the prover proves their identity. The verifier can verify the provided identity data even while being offline (without internet connection) by verifying an attestation provided by the SSI network. The offline verification is achieved by utilizing a so-called trust store that contains all public keys of the SSI network and is stored on the verifier side. This trust store is being updated whenever a new node is joining the network.

B. Stages

This subsection details the four main stages of our concept such as (1) the import of the identity data, (2) obtaining the attestation, (3) performing revocation, and finally (4) the showing of selected identity attributes.

Import Identity Data. In the first stage of our concept, import identity data, the prover, which is in possession of an eID, imports their identity data into their identity wallet. To start this process, the prover actively selects import identity data from existing eID within the identity wallet. An authentication request is sent to the IdP. Next, the prover has to perform identification and authentication towards the IdP. After success, the IdP issues an identity assertion to the prover which contains related identity attributes. This assertion was imported into the identity wallet.

Obtain Attestation. In order to use the imported identity data in other IdM systems, like an SSI system, the prover obtains an attestation from the SSI network. This process starts automatically after the import of identity data was successfully finished. First, a proof is created stating the knowledge of secret attributes signed by the IdP and that there is a commitment to these attributes and also that there is a valid signature on the attributes. Next, the attestation is requested at the SSI network by providing the proof as well as the commitment. The SSI network, which runs an adapted version of the BFT protocol, receives the request including the proof and distributes it to the network. Furthermore, the nodes in the SSI network verify the proof and sign the commitment. If the verification was successful, the nodes add the revocation ID to the revocation accumulator. In the next step, the nodes exchange their signed result, verify it again and create a multi-signature by aggregating all signatures. The resulting attestation is issued to the prover and stored in the wallet.

Revocation. In case that the imported identity data are not any longer valid due to a change of name etc., or in case that the device used for storing the identity data is broken, lost, or was stolen, the prover can revoke their identity data. Since only the prover has knowledge about the revocation ID from the revocation accumulator, no adversary would be able to maliciously revoke the prover's identity data. The revocation process itself is performed by removing the revocation ID from the revocation accumulator.

Showing. In the showing stage, the prover uses their identity data towards a prover in order to perform identification and authentication. First, the prover selects the identity attributes that should be revealed. Next, a proof is created stating that the revealed attributes are part of the secret attributes that were imported from the IdP. The prover authenticates towards the verifier by performing a challenge-response protocol and by providing the revealed attributes, together with the showing proof and the signed commitment. The signatures are verified by either gathering the public keys from the SSI network (online case) or by utilizing the public keys from the key-store (offline case). Additionally, the verifier checks the proofs as well as the validity of the identity data by verifying the revocation witness.

C. Construction

For the following discussions, we consider credentials as an abstract set of attributes, i.e, a credential $cred$ consists of set of attributes $A = \{a_1, \dots, a_n\}$. In practice, they will be encoded in a certain way, which however is not of interest for the description of our concept. Similar to the approach taken by Abraham et al. [4], we denote by a_{n+1} a special attribute for revocation which is appended to A and signed by the IdP together with all other attributes. Thereby, we link the randomly sampled a_{n+1} to the credential. The idea for privacy-preserving revocation is to follow the accumulator-based approach discussed in [28]: the SSI network keeps a compact accumulator with all valid a_{n+1} and a list of all revoked a_{n+1} .

Now, for attestation, credential owners interact with the SSI nodes in the following way: they convince the nodes that their credential is valid by producing a proof that attests that they know the attributes to a credential and that the signature obtained from the IdP is valid for that credential. They also present the revocation attribute a_{n+1} in plain so that nodes can check their list of revoked credentials. Additionally, to avoid the need to always link back the attributes to the original signatures, the prover sends a Pedersen commitment $C = Com(A)$ and proves the relation between the signature and the commitment. Once the SSI nodes are satisfied with the proofs, they update the accumulator Λ with a_{n+1} and produce a membership witness wit . This witness is signed using a multi-signature scheme and the aggregated signature together with the witness is sent to the credential owner.

The purpose of the proofs is to prove knowledge of the attributes signed by the IdP and that the commitment matches these attributes. On a high level, we want to ensure that for the IdP's signature σ and the prover's commitment C

$$\Sigma.Verify(pk_{Iss}, A, \sigma) = 1 \wedge Com(A) = C$$

with $A = \{a_1, \dots, a_{n+1}\}$ holds true whereas the attributes $\{a_1, \dots, a_n\}$ are not revealed to the SSI nodes.

Showings follow a similar strategy, but works relative to the commitment only. Additionally, the service provider needs to ensure that the credential was not revoked, and hence the prover presents the membership witness. In this case, we allow

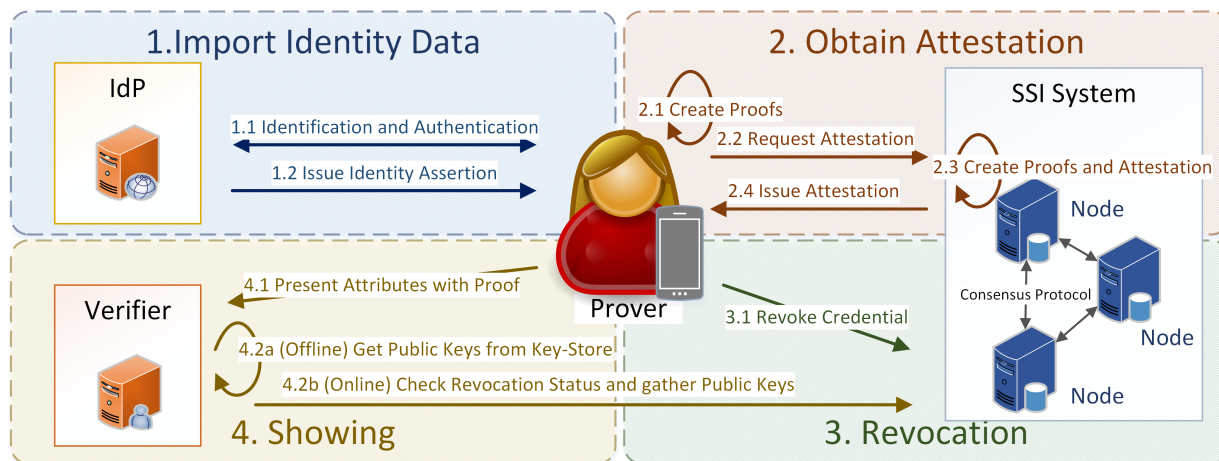


Figure 2. Architectural overview of the proposed concept including the actors and main process flows within the four defined stages

that a subset of the attributes is revealed, but want to keep a_{n+1} hidden. Hence, the verifier needs to be convinced that the following statement holds true:

$$\text{Com}(A) = C \wedge \Lambda. \text{Verify}(\text{pk}_{\text{Iss}}, \Lambda, \text{wit}_{a_{n+1}}, a_{n+1}) = 1.$$

The prover reveals some attributes A_r as part of this process, but keeps $A_p = A \setminus A_r$ secret. In addition, the holder sends the commitment C and the aggregated signature σ_A in plain to the verifier.

V. IMPLEMENTATION

While our concept is described on a generic level, the concrete choice of cryptographic primitives is important for both practical purposes and the security properties that can be obtained. On the side of the IdP, we have to be conservative with our choice to support existing IdPs. Hence, we may only assume that Σ is instantiated with an RSA-based signature scheme or ECDSA (cf. also the available signature schemes in the current SAML specification [33]). This is however problematic as we would be required to prove statements with respect to a random oracle for attestation (cf. for a discussion on that topic in a different context [1]). Fortunately for us, XML signatures actually sign the hash value of a canonicalized version of the XML document, i.e., they are signatures on a binding commitment. Therefore, it is enough for us to prove that the credential holder knows A such that $H(A) = h$ and h as well as the signature on h can be shared with the SSI nodes for verification.

For all other actors of the system, we are not required to be as conservative but use well-aligned primitives instead. BLS [12] is the natural choice for the multi-signature scheme Σ_M (cf. [3]). It features both small public keys and signatures that are very simple to aggregate.

The choice of the commitment scheme and the accumulator need to be aligned with the proof system to be efficient. When using a SNARK with BLS-381 [7] as pairing-friendly curve, schemes that work on elliptic curves over the induced prime order field perform particularly efficient. Hence, proofs with

respect to a Pedersen commitment defined over the Jubjub curve⁵ work well. The accumulator is more tricky to instantiate efficiently with respect to the NIZK.

First of, one could use Merkle tree as an accumulator. In this case, a SNARK-optimized hash function such as Poseidon [21] helps to reduce the number of constraints. This approach however requires the users to repeat the attestation process to obtain updated witnesses whenever the accumulator changes. With a Merkle-tree approach as proposed by Kales et al. in the context of certificate transparency [25], this problem can be alleviated to some extent. There the Merkle tree is split into individual smaller trees. Their root is then accumulated into the bigger tree but the small trees stay constant once accumulated, i.e., a fixed amount of revocation IDs is accumulated into a sub-accumulator Λ_i which are then accumulated into Λ . Once Λ_i is full, witnesses relative to Λ_i stay constant as long as only revocation IDs are added. Only if a revocation ID is removed and hence revoked, witnesses of the users within the same sub-accumulator need to be updated. For the proofs, only a proof with respect to a sub-accumulator would need to be provided. The witness for the inclusion of Λ_i in Λ could be transported in plain without the need to hide it.⁶

Alternatively, the pairing-based accumulator [31] offers constant-size witnesses and proofs as well as efficient accumulator updates. More importantly, with this type of accumulator it is possible to publish witness-independent witness updates, meaning that on a change users do not need to perform a new attestation. With the recent work on threshold accumulators by Helminger et al. [23], the trust put into the accumulators can be further reduced by distributing trust to the SSI nodes. Nevertheless, as these accumulators require pairing-friendly curves, they are hard to instantiate with a curve that works efficiently with the prime fields induced by the order of BLS-381.

⁵<https://z.cash/technology/jubjub/>, accessed on 2021-04-15

⁶The only information that would leak here is the time of the attestation but this bit of information could also be derived from observing the SSI blockchain.

Alternatively, BLS-381 could be replaced by another pairing-friendly curve that is suitable for proof compositions [24].

A. Evaluation

Finally, we present the evaluation of our proof-of-concept implementation based on Bellman written in Rust.⁷ We used Bellman’s community edition⁸ and configured it to use BLS-381 as pairing-friendly curve. We also used Bellman-BigNat⁹ for a unified interface to hash functions and bellman-playground¹⁰ for a Merkle-tree implementation. The primitives we used are SHA-256 and Pedersen commitments for the attestation and the same commitment scheme, and a Merkle tree implemented using Poseidon [21] for the showing. To show the difference to a hash function that is not optimized for SNARKs, we also illustrate the performance of a showing using a SHA-256-based Merkle tree. The benchmarks were performed on an Intel(R) Core(TM) i7-8650U with 16 GB RAM running on Ubuntu 20.04.

In terms of user attributes, we performed the benchmarking using the minimal data set of eID assertions [18] consisting of the family (3-16 bytes) and given name (3-16 bytes), the date of birth (10 bytes), and a personal identifier (31 bytes). In addition to these four attributes, each user also has a revocation ID (31 bytes). Thus, all in all, each user has 5 attributes in their credential. Note though, that the length of these attributes was to fit into one field element. With our choice of parameters, a field element can represent up to 31 bytes. When requiring attributes that are larger than 31 bytes, additional field elements are required to represent them, and hence additional constraints are required as well. While this involves some cost, it will only make a larger difference when passing the block size of the hash function. This will result in hundreds to thousands of constraints that need to be considered in the circuit.

For attestation, benchmarks show that proof creation takes 15.88 ± 0.4 seconds which is caused by the large size of the hash function input (> 6 KB) as well as the hash function itself. The verification is fast and only takes 20.25 ± 1.6 ms.

The results for the proof generation in the showing phase are depicted in Figure 3 for both SHA-256 and Poseidon as hash functions for the Merkle tree. While we performed the benchmarks with showings of 0 to 4 attributes, this choice does not influence the number of constraints significantly, and hence the timings are (up to some noise) the same. When using Poseidon, we observe that each level in the Merkle tree adds approximately 20 ms for the prover, which is consistent with the constant increase in the number of constraints. Even with Merkle trees containing over a million revocation IDs, proving only takes 582 ms. For SHA-256 on the other hand, the picture is different. We also see a constant increase in the number of constraints, but the observed runtime quickly

increases, starting at around 2.29 seconds for a Merkle tree with only 32 elements, and increases to over 8.1 seconds for 2048 elements. Hence, using SHA-256 as hash function for the Merkle tree is impractical even for small sizes. Combining these results with the sub-accumulator approach shows that we can easily lift good performance from millions to billions of users in the system.

The time for proof verification, on the other side, stays constant and averages at ≈ 3.7 ms regardless of the concrete parameters as expected from a SNARK. Note also, that the verification in this case is faster than for attestation, since the public inputs for attestation, e.g. the overhead due to the XML encoding, are significantly larger.

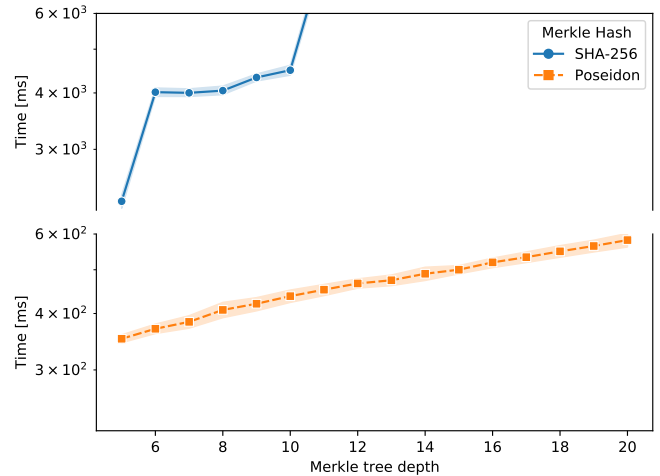


Figure 3. Benchmarks for proof generation during showings for various depths of the Merkle tree. The time is displayed as mean and standard deviation over 100 runs.

In all cases, the proofs are the size of three group elements. With our choice of curve the proof takes 192 bytes if the elements are stored in compressed form.

B. Security Evaluation

For the security of the system, we discuss the three main security properties informally. The first two are the same as in [3]: First, users must not be able to present an assertion to the validator nodes that was not signed by the IdP or where they do not know any of the attributes. Assuming that an adversary would be able to convince the validator nodes otherwise, the adversary is either able to produce a forgery of the signature scheme or produce a proof without knowing any witness. Similarly, when users authenticate to an SP, a user must not be able to authenticate itself if the attributes are not known or the assertion was not checked by the validator network. In both cases, an adversary would break soundness of the proof system or the unforgeability of the signature schemes.

Second, neither SPs nor the validator nodes should be able to learn any of the hidden attributes. Assume that an adversary

⁷The code is available on Github: <https://github.com/sebastinas/ppoid-bench>.

⁸<https://github.com/matter-labs/bellman>, accessed on 2021-04-15

⁹<https://github.com/alex-ozdemir/bellman-bignat>, accessed on 2021-04-15

¹⁰<https://github.com/kilic/bellman-playground>, accessed on 2021-04-15

would be able to reveal one of the attributes that were not revealed by the users itself. Given that an adversary only gets to see a hiding commitment and the corresponding proof, an adversary would break the commitment scheme or the zero-knowledge property of the proof system. We want to note, that with respect to the validators we cannot achieve this notion due to use of XML signatures, though. Even if the commitment on the XML document was made hiding by including additional randomness, security would only hold in the random oracle model. Then we would again be unable to provide proofs.

Third, a user must not be able to convince SPs that their credential was not revoked if it was. Assuming an adversary would be able to do so, again, the soundness of the proof system would be broken, or it would be possible to produce membership witnesses for non-members of the accumulator. The latter would break the security guarantees of the accumulator.

C. Discussion

Limitations. In our proposed concept, only the prover can revoke their derived identity data. In order that the IdP would also be able to revoke, it would have to add a random identifier to the identity assertion. Even though this is just a small change at the IdP, we still consider it as limitation. Furthermore, the maximal length of the attributes needs to be fixed a priori as it influences the circuits. Note also, that varying the circuits to accommodate various lengths may leak bits of information that could lead to unintended disclosure of otherwise hidden attributes.

The use of Merkle trees in this scenario comes with some drawbacks. First, it requires the use of techniques such as sub-accumulators to reduce the costs when updating membership witnesses. Second, the size of accumulators influences the overall performance of showings. Large trees require a SNARK-optimized hash function such as Poseidon or Vision [6] to be practical.

To obtain anonymity against validator nodes, a hiding commitment scheme or a signature scheme whose security can be proven without random oracles is required. However, for XML signatures, we are currently limited by the allowed schemes in the standard.

Future Work. Replacing the Merkle tree with a dynamic public-key accumulator offering witness-independent updates will help to reduce the work involved in updating user's membership witnesses. Therefore, we believe this road is worth investigating in future work. More importantly, this change will also make the circuits used for showings independent of the size of the accumulated sets.

VI. CONCLUSION

Bringing traditional IdM systems together with the recent SSI systems is still an open issue. This work tackles this issue by introducing a novel trust model in which identity data are being derived into an SSI system in a fully-privacy-preserving way, but by still maintaining trust in the derived data. Moreover, our concept also enables offline revocation in a

privacy-preserving manner by utilizing a ZK proof system. To show the feasibility, we implemented a prototype, evaluated, and benchmarked it.

REFERENCES

- [1] B. Abdolmaleki, S. Ramacher, and D. Slamanig, "Lift-and-shift: Obtaining simulation extractable subversion and updatable snarks generically," in *CCS*. ACM, 2020, pp. 1987–2005.
- [2] A. Abraham, "Whitepaper – Self-Sovereign Identity – A-SIT Technologie," 2017. [Online]. Available: <https://technology.a-sit.at/en/whitepaper-self-sovereign-identity/>
- [3] A. Abraham, F. Hörandner, O. Omolola, and S. Ramacher, "Privacy-preserving eid derivation for self-sovereign identity systems," in *ICICS*, ser. LNCS, vol. 11999. Springer, 2019, pp. 307–323.
- [4] A. Abraham, S. More, C. Rabensteiner, and F. Hörandner, "Revocable and offline-verifiable self-sovereign identities," in *TrustCom*. IEEE, 2020, pp. 1020–1027.
- [5] A. Abraham, K. Theuermann, and E. Kirchengast, "Qualified eid derivation into a distributed ledger based idm system," in *TrustCom/BigDataSE*. IEEE, 2018, pp. 1406–1412.
- [6] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooche, and A. Szepieniec, "Design of symmetric-key primitives for advanced cryptographic protocols," *IACR Trans. Symmetric Cryptol.*, vol. 2020, no. 3, pp. 1–45, 2020.
- [7] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *SCN*, ser. LNCS, vol. 2576. Springer, 2002, pp. 257–267.
- [8] J. C. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures (extended abstract)," in *EUROCRYPT*, ser. LNCS, vol. 765. Springer, 1993, pp. 274–285.
- [9] H. Berghel, "Equifax and the latest round of identity theft roulette," *Computer*, vol. 50, no. 12, pp. 72–76, 2017.
- [10] —, "The equifax hack revisited and repurposed," *Computer*, vol. 53, no. 5, pp. 85–90, 2020.
- [11] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *PKC*, ser. LNCS, vol. 2567. Springer, 2003, pp. 31–46.
- [12] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *ASIACRYPT*, ser. LNCS, vol. 2248. Springer, 2001, pp. 514–532.
- [13] E. Bresson and J. Stern, "Efficient revocation in group signatures," in *PKC*, ser. LNCS, vol. 1992. Springer, 2001, pp. 190–206.
- [14] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO*, ser. LNCS, vol. 2442. Springer, 2002, pp. 61–76.
- [15] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
- [16] D. Derler, C. Hanser, and D. Slamanig, "Revisiting cryptographic accumulators, additional properties and relations to other primitives," in *CT-RSA*, ser. LNCS, vol. 9048. Springer, 2015, pp. 127–144.
- [17] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," in *USENIX*. USENIX Association, 2020, pp. 2093–2110.
- [18] eIDAS eID Technical Subgroup, "eIDAS SAML attribute profile," 2019. [Online]. Available: <https://ec.europa.eu/cedigital/wiki/download/attachments/82773108/eIDAS%20SAML%20Attribute%20Profile%20v1.2%20Final.pdf>
- [19] L. Fritsch, "Identity management as a target in cyberwar," in *Open Identity Summit*, ser. LNI, vol. P-305. Gesellschaft für Informatik e.V., 2020, pp. 61–70.
- [20] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *NDSS*. The Internet Society, 2014.
- [21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *USENIX Security*, 2021.
- [22] J. Groth, "On the size of pairing-based non-interactive arguments," in *EUROCRYPT (2)*, ser. LNCS, vol. 9666. Springer, 2016, pp. 305–326.
- [23] L. Helming, D. Kales, S. Ramacher, and R. Walch, "Multi-party revocation in sovrin: Performance through distributed trust," in *CT-RSA*, ser. LNCS, vol. 12704. Springer, 2021, pp. 527–551.
- [24] Y. E. Housni and A. Guillevis, "Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition," in *CANS*, ser. LNCS, vol. 12579. Springer, 2020, pp. 259–279.

- [25] D. Kales, O. Omolola, and S. Ramacher, “Revisiting user privacy for certificate transparency,” in *EuroS&P*. IEEE, 2019, pp. 432–447.
- [26] F. M. Kamm, “Definition of device system architecture and derivation scheme of mobile ids,” 2017. [Online]. Available: <https://www.lightest.eu/static/deliverables/D7.2.pdf>
- [27] —, “Definition of requirements for derivation and attestation of mobile ids,” 2017. [Online]. Available: <https://www.lightest.eu/static/deliverables/D7.1.pdf>
- [28] D. Khovratovich and J. Law, “Sovrin: digital signatures in the blockchain area,” 2016. [Online]. Available: <https://sovrin.org/wp-content/uploads/AnonCred-RWC.pdf>
- [29] L. Lamport, R. E. Shostak, and M. C. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
- [30] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, “A survey on essential components of a self-sovereign identity,” *Comput. Sci. Rev.*, vol. 30, pp. 80–86, 2018.
- [31] L. Nguyen, “Accumulators from bilinear pairings and applications,” in *CT-RSA*, ser. LNCS, vol. 3376. Springer, 2005, pp. 275–292.
- [32] NIST, “Guidelines for Derived Personal Identity Verification (PIV) Credentials,” *NIST Special Publication 800-157*, pp. 1–82, 2014.
- [33] OASIS. Saml (security assertion markup language) specifications. [Online]. Available: <http://saml.xml.org/saml-specifications>
- [34] M. Sporny, D. Longley, and D. Chadwick, “Verifiable Credentials Data Model 1.0,” 2019. [Online]. Available: <https://www.w3.org/TR/vc-data-model/>
- [35] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein, “Data breaches, phishing, or malware?: Understanding the risks of stolen credentials,” in *CCS*. ACM, 2017, pp. 1421–1434.
- [36] W3C, “Decentralized Identifiers (DIDs) v1.0 First Public Working Draft:” [Online]. Available: <https://www.w3.org/TR/did-core/>
- [37] B. Zwattendorfer, T. Zefferer, and K. Stranacher, “An overview of cloud identity management-models,” in *WEBIST (1)*. SciTePress, 2014, pp. 82–92.

APPENDIX

NIzkS. Let $L \subseteq X$ be an NP-language with associated witness relation R so that $L = \{x \mid \exists w: R(x, w) = 1\}$. A non-interactive proof system Π consists of algorithms $\text{Setup}(1^\kappa)$ producing a common reference string crs , $\text{Proof}(\text{crs}, x, w)$ taking the crs , a statement x and a witness w , and outputting a proof π , and $\text{Verify}(\text{crs}, x, \pi)$ taking the crs , a statement x , and a proof π , and outputting the verification status.

We require such a proof system to be complete (all proofs for statements in the language verify), sound (a proof for a statement outside the language verifies only with negligible probability) and zero-knowledge (a proof reveals no information on the witness). In addition, when we talk about SNARKs, we also require the proofs to be succinct (independent of the witness size) and knowledge sound (there exists an extractor that using some trapdoor is able to extract the witness from the prover).

Commitments. A commitment scheme consists of algorithms $\text{Gen}(1^\kappa)$ outputting public parameters, $\text{Com}(m)$ taking a message m and producing a commitment C and an opening D , and $\text{Open}(C, D)$ taking a commitment C and opening D and outputting the verification status. Secure commitments must satisfy correctness, hiding (commitments are indistinguishable), and binding (a commitment cannot be opened to a different message).

Accumulators. We recall the formalization of accumulators from [16].

Definition 1. A static accumulator is a tuple of efficient algorithms $(\text{Gen}, \text{Eval}, \text{WitCreate}, \text{Verify})$, which are defined as follows:

$\text{Gen}(1^\kappa, t)$: This algorithm takes a security parameter κ and an upper bound t on the number of elements to be accumulated. It returns a key pair $(\text{sk}_\Lambda, \text{pk}_\Lambda)$, where $\text{sk}_\Lambda = \emptyset$ if no trapdoor exists.

$\text{Eval}((\text{sk}_\Lambda, \text{pk}_\Lambda), \mathcal{X})$: This algorithm takes a key pair $(\text{sk}_\Lambda, \text{pk}_\Lambda)$ and a set \mathcal{X} to be accumulated and returns an accumulator $\Lambda_{\mathcal{X}}$ together with auxiliary information aux .

$\text{WitCreate}((\text{sk}_\Lambda, \text{pk}_\Lambda), \Lambda_{\mathcal{X}}, \text{aux}, x_i)$: This algorithm takes a key pair $(\text{sk}_\Lambda, \text{pk}_\Lambda)$, an accumulator $\Lambda_{\mathcal{X}}$, auxiliary information aux and a value x_i . It returns \perp , if $x_i \notin \mathcal{X}$, and a witness wit_{x_i} for x_i otherwise.

$\text{Verify}(\text{pk}_\Lambda, \Lambda_{\mathcal{X}}, \text{wit}_{x_i}, x_i)$: This algorithm takes a public key pk_Λ , an accumulator $\Lambda_{\mathcal{X}}$, a witness wit_{x_i} and a value x_i . It returns a bit indicating whether wit_{x_i} is a witness for $x_i \in \mathcal{X}$.

A dynamic accumulator is a static accumulator with an additional tuple of efficient algorithms $(\text{Add}, \text{Remove}, \text{WitUpdate})$ which makes it possible to add and remove elements from the accumulator, as well as update existing membership witnesses accordingly, respectively.

For accumulators we require it to be intractable to present witnesses for non-members. Examples of dynamic accumulators include Merkle trees and pairing-based accumulators [31]. For the latter, witness updates can be performed using witness-independent update tokens, whereas for Merkle trees witness updates require a recomputation in the worst case.

Signatures. We recall the standard notion of digital signatures.

Definition 2. A signature scheme Σ is a triple $(\text{KeyGen}, \text{Sign}, \text{Verify})$ of PPT algorithms, which are defined as follows:

$\text{KeyGen}(1^\kappa)$: This algorithm takes a security parameter κ as input and outputs a secret key sk and a public key pk .

$\text{Sign}(\text{sk}, m)$: This algorithm takes a secret key sk and a message m as input and outputs a signature σ .

$\text{Verify}(\text{pk}, m, \sigma)$: This algorithm takes a public key pk , a message m , and a signature σ as input and outputs a bit $b \in \{0, 1\}$.

We require a signature scheme to be correct and to provide existential unforgeability under adaptively chosen message attacks (EUF-CMA). Furthermore, we are interested in an extension of signature schemes to multi-signature schemes [17]. In this case, signatures on the same message w.r.t. some public keys, can be aggregated into one compact signature which is valid w.r.t. an aggregated public key. It extends a signature scheme with algorithms $(\text{APKs}, \text{ASigs}, \text{AVerify})$ which aggregate public keys as well as signatures and verify aggregated signatures, respectively. The BLS signature scheme [12] is a prominent example of a signature scheme that can be extended to a multi-signature [11].