SSI Strong Authentication using a Mobile-phone based Identity Wallet Reaching a High Level of Assurance

Andreas Abraham[a], Christopher Schinnerl and Stefan More[b]
Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, Graz, Austria

Keywords: Self-Sovereign Identity, Identity Management, Distributed Ledger, Strong Authentication, Identity Wallet.

Abstract: Assurance in digital authentication means represents a fundamental requirement in the authentication process of digital identities. Different level-of-assurance (LoA) describe the trustworthiness of the authentication specified by various standards. Some traditional governmental identity systems achieve a high LoA. Nevertheless, the recent self-sovereign identity (SSI) model, which utilizes identity wallets to ensure that the identity data control remains with the related user, still lacks a high LoA, detaining the full potential of SSI such as using it for sensitive use-cases like for eGovernment or public administration services. This work tackles this problem by starting with assessing related LoA standards. Based on this assessment are requirements defined to achieve an LoA high. These requirements are utilized in the process of defining and evaluating our proposed concept. Our generic serves as the foundation for other developers, aiming to elevate the LoA in their SSI systems. The implementation of a proof-of-concept showcases the feasibility and practicability of our concept. In the evaluation, we identify measures provided by our concept, used to meet the defined requirements, and discuss the design decisions.

1 INTRODUCTION

Digitalization stimulates innovation in the field of digital identities and identity management (IdM). IdM models started with the isolated identity model (Zwattendorfer et al., 2014), in which the service provider (SP) and identity provider (IdP) are represented by the same party and evolved over time to the federated and user-centric identity models. In the user-centric model, the user's identity data are stored within the user domain, while a central trusted party still issues the identity data.

With the emergence of blockchain technology, new opportunities arose in various fields, including IdM. The concept of Self-Sovereign Identity (SSI) represents an evolvement of the user-centric identity model (Abraham, 2017). SSI makes users of identity data the sovereign owners of their data without the need for a central trusted party to manage it. The relevance and potential of SSI is shown by the research community (Houtan et al., 2020; Manski, 2020; Liu et al., 2020; Dong et al., 2020). Additionally, the European Commission (EC) recognized the potential of SSI early on by launching the European self-sovereign identity framework[1] (ESSIF) use case group within the European blockchain services infrastructure[2] (EBSI).

One essential component of an SSI system is the so-called identity wallet: A piece of software often supported by special hardware, which is responsible for storing and managing cryptographic key material, identifiers as wells as identity data (Kondova and Erbguth, 2020). Wallets are often implemented as mobile phone or browser applications. Since our daily life is becoming increasingly mobile phone-centered, the demand on mobile identity wallets is growing. This is reflected by the various implementations of identity wallets available, which we discuss in Section 3. Each of these wallet projects focuses on different aspects such as usability or security and privacy of the stored data.

An identity system should be applicable in various fields and also support usage of sensitive services such as eGovernment or online banking. In order for

[a] https://orcid.org/0000-0002-4163-9113
[b] https://orcid.org/0000-0001-7076-7563

[1] https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=262505734, Accessed: 2021-02-10.

[2] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EBSI, Accessed: 2021-02-10.

Abraham, A., Schinnerl, C. and More, S.
SSI Strong Authentication using a Mobile-phone based Identity Wallet Reaching a High Level of Assurance.
DOI: 10.5220/0010542801370148
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 137-148
ISBN: 978-989-758-524-1

the SP to have enough trust and guarantees in the authentication data to accept an identity, a high level of assurance (LoA) is required. The European eIDAS framework (Commission, 2015) defines LoA in three levels: low, substantial, and high. Each of these LoA specifies different requirements and for LoA high, the requirements are the strictest and most challenging to fulfill but also provide the most assurance in the data. Having a digital identity that achieves high LoA benefits both users and Service Providers (SP): Users can use their digital identity for a wider range of services, while SPs have a high certainty that the identity data are correct and correspond to the related person. This enables the use of an identity system for use-cases and sensitive services such as an ID card on the mobile phone, a passport or driving license. While traditional identity systems already achieve LoA high, at the moment none of the available SSI wallet solutions is capable of reaching the LoA *high*.

Contribution. In our paper, we tackle this issue by being the first work that proposes a generic concept enabling a mobile phone based identity wallet to achieve the LoA high. The proposed architecture can be used by implementers of existing wallets to achieve this highest LoA. A SSI wallet supporting the LoA high provides benefits such as increased trust in the identity data. This furthermore enables use of a SSI for sensitive services such as eGovernment or other services which require high quality of identity data. Our contribution consists of four main parts:

(I) **Evaluation of LoA Requirements.** To define the requirements for a mobile wallet implementation in order to achieve LoA high, we evaluate the relevant standards related to LoA, namely ISO 29115 (International Organization for Standardization (ISO), 2013) and the eIDAS implementation act (Commission, 2015). Based on this evaluation, we define seven system requirements, which we use as a basis for designing our architecture. These requirements can also be used to evaluate other wallet implementations.

(II) **Wallet Architecture.** Based on these requirements, we design a generic architecture of a wallet that reaches the LoA high. Components of this architecture were selected through an assessment of software and hardware solutions that meet those requirements. The resulting architecture uses a trust (TSP) as IdP to register the (de-central/sovereign) SSI, ensuring no in-person registration is required while still complying with LoA high requirements. Our wallet architecture utilizes the secure element of the mobile phone as well as a second key on a FIDO2 hardware token, which introduces two strong factors in addition to the temper-resistant key protection required to fulfill the LoA requirements. By building on existing SSI standards, we ensure that our architecture is compatible and thus useful for existing wallet implementations as well.

(III) **Demonstrator Implementation.** To show the feasibility of our architecture and its practicability, we implement a proof of concept (PoC) wallet for iOS. To protect the key material and ensure user presence, we use the iPhone's secure enclave and a YubiKey as FIDO2 token. By using cryptographic accumulators, our proof of concept also supports privacy-preserving revocation of identity data.

(IV) **Evaluation.** To demonstrate that our architecture achieves an LoA high, we evaluate our architecture with regard to our requirements. Additionally, we discuss the security, design decisions and other aspects of our work.

Outline. The rest of this paper is structured as follows: Section 2 briefly introduces the building blocks of our system. In Section 3 we discuss the related work in this field and give a high-level comparison with our work. Section 4 details the architecture of our system together with the actors, the relevant LoA requirements, and a formal protocol description. The implementation details of our PoC are shown in Section 5. We conclude the paper with an evaluation and discussion in Section 6.

2 PRELIMINARIES

2.1 Self-Sovereign Identity System

Identity management (IdM) describes the tools and processes required to manage digital identities throughout the whole identity lifecycle. IdM also evolved over time (Zwattendorfer et al., 2014) according to the needs of the ecosystem starting with the isolated IdM model in which the identity provider (IdP) and the service provider (SP) are located at the same party. A recent IdM model describes the user-centric model, which focuses on the user.

Self-sovereign identity (SSI) is an IdM model, which can be seen as the further evolution of the user-centric model. The two main benefits of SSI is that the user is in full control over their own identity data as well as not to have to rely on a central trusted authority or party.

Figure 1 depicts the main actors of an SSI system including their interactions. The architecture of an SSI system uses a distributed ledger (DL) to eliminate a central trusted authority and to distribute trust

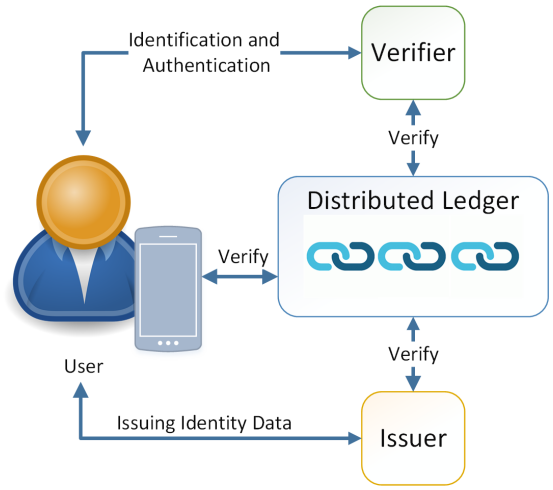

Figure 1: High-Level Architecture of an SSI System.

among the network of nodes. The SSI network, consisting of semi-trusted nodes, serves as a decentralized public key infrastructure (DPKI). Semi-trusted nodes are represented by organizations or companies such as banks or universities. Thus, the network might not be open for everyone to host a node, instead, a consortium is hosting those nodes.

The actors of such an SSI system are the users who want to use their digital identity in order to perform identification and authentication, e.g., towards online services. The issuer represents a party that issues credentials like an IdP. In contrast, the verifier is the party verifying the identity data provided by the user, which can be an SP. Notably, private information such as private key material or personal identifiable information (PII) are not stored on the ledger but stored off-ledger. Only public information such as public keys are stored on the ledger.

2.2 Decentralized Identifiers (DIDs)

Decentralized identifiers (DIDs) (W3C Working Draft, 2019) were designed with the main purpose of enabling SSIs. The main advantage is that for the creation of such a DID a central trusted party is not required. DIDs are URLs that resolve to an entry on the DL, the so-called DID document.

DID documents are stored on the DL and are used to verify the related DID. DID documents contain three main sections: the proof purposes, verification methods as well as service endpoints. DID documents also contain the public keys related to the DID. By signing a challenge with its private key, a verifier can resolve a DID to its DID document and verify the signature with the public key of the DID document. If the ledger in which the DID document is stored is a consortium ledger, there might be policies on how to register the DID document on the DL. Depending on those policies and on the governance of the DL, the existence of the DID document on the DL represents a root of trust.

2.3 Verifiable Credentials (VCs)

The W3C verifiable credential (VC) data model (Sporny et al., 2019) defines the data format for verifiable information. Credentials are identity-related information asserted for a specific identity. The VC specification states the mechanisms to express those credentials online to ensure that they are cryptographically secure, respecting privacy as well as being machine-readable. VCs are often used in SSI systems because of their flexible and lightweight format.

2.4 Level of Assurance (LoA)

The level of assurance (LoA) defines the level of confidence in digital identities when performing identification and authentication. The confidence lies in aspects such as the protection of assets like key material, the protection against unauthorized access or usage and maintenance of audits and logs, besides other aspects (International Organization for Standardization (ISO), 2013). This work focuses on the LoA specifications ISO 29115 (Entity authentication assurance framework (International Organization for Standardization (ISO), 2013)) and the European implementing regulation (EU) 2015/1502 (Commission, 2015). These two standards were selected since our work focuses on being compliant with European standards, with ISO 29115 representing the foundation of the European implementation regulation for LoA.

In these standards, the LoA framework consists of a technical and an organizational and management part. The technical part is split into three phases: (I) the enrollment phase, in which the identity proofing of the user and the registration is performed, (II) the credential management phase, concerning the credential creation, issuance, storage, revocation, etc., and finally (III) the entity authentication phase focusing on the authentication. The ISO standards defines four different LoA whereas the European regulation only defines three: low, substantial, and high, where ISO LoA 4 is equivalent to EU LoA high.

2.5 Fast Identification Online (FIDO)

Fast identification online (FIDO) (The FIDO Alliance, 2020) was founded by the FIDO alliance, an

open industry association with the main goal to develop an authentication standard aiming to get rid of the usage of passwords. Instead, FIDO supports many other authentication technologies such as biometry factor like fingerprint or face recognition, trusted platform modules (TPMs), security tokens on USB sticks, and more. When mentioning FIDO in this work, we refer to the current version of the FIDO specifications, called FIDO 2. FIDO 2 consists of two parts: The first part, Webauthn, is a standardized API used by websites to register and authenticate with hardware authenticators. A FIDO authenticator can be part of external hard or software or actually being a part of the user's device and responsible for generating public private key-pairs. The second part is the client-to-authenticator protocol (CTAP), which handles the communication between a client, such as a web browser, and the hardware authenticator. We chose the FIDO protocol as a foundation for our implementation as it allowed us to reuse existing libraries and SDKs. It also allowed us to explore adjusting and reusing an existing standard to fit our requirements.

3 RELATED WORK

This section details the work related to our project. In particular, we list an excerpt of SSI identity wallet implementations, focusing on wallets that are in an advanced development state. Nevertheless, none of the listed identity wallet projects below achieves the LoA high with respect to (Commission, 2015).

Our work focuses on achieving an LoA high, which differs from the related work, since achieving a certain LoA is not the focus of most of the projects.

Wallet Implementations. Jolocom SmartWallet[3] is an identity wallet by Jolocom (Jolocom, 2021). This wallet app can be used for managing identity data where the users are in control over their own data. Nonetheless, the focus in this wallet is not to achieve a certain LoA.

DIZME (this is me) is a project by the trust over IP foundation [4] aiming to fill the gap between SSI and eIDAS compliance (Foundation, 2021). This project details the governance and technology utilized to achieve certain LoA levels. Nevertheless, DIZME achieves LoA levels up to substantial.

Connect.Me[5] is a commercial wallet implementation by Evernym[6]. Connect.Me is a wallet focusing on holding and sharing credentials, which utilizes secure 1-to-1 communication channels for exchanging data as well as zero-knowledge proofs to achieve selective disclosure. Nevertheless, the main objective is not to achieve a certain LoA, instead providing a wallet implementation that works well with the Sovrin network and other Evernym components.

Alastria Wallet[7] is a wallet implementation of Alastria(Alastria, 2021). Alastria in a non-profit organization building SSI including its own distributed Ledger network. This wallet implementation aims to be use the Alastria network.

Research on Wallets. The work of (Dai et al., 2021) focuses on the Trustzone[8] on mobile devices to create and store cryptographic key material as well as to perform critical operations to ensure security and reduce attack risks. Providing tamper-resistant storage for cryptographic key material reflects one requirement of LoA high. Nevertheless, this work does not address other requirements to achieve LoA nor is the focus on achieving a certain LoA. Besides, a mobile phone's secure element might not be sufficiently secure enough to fulfill the tamper-resistant hardware requirement.

In the work of (Iqbal et al., 2020) lies the focus on mobile phone-based wallets, which are applying fingerprint as authentication factor especially considering the usability for the elderly. The usability of software components especially for the elderly for which it can be harder to understand the necessary steps when performing authentication describes an important aspect when designing and implementing digital wallets. The cryptographic key material in this work is protected through fingerprint verification as well as bind to the actual user. Nevertheless, the focus of this work is mainly on usability considering specifically the elderly and not achieving a certain LoA.

A specification to evaluate aspects of SSI systems also considering a digital wallet as storage is presented in (Naik and Jenkins, 2020). Additionally a comparison of Sovrin[9] and uPort[10] is shown consid-

[3]https://github.com/jolocom/smartwallet-app, Accessed at: 2021-02-09.

[4]https://trustoverip.org/, Accessed at: 2021-02-09.

[5]https://www.evernym.com/products/#ConnectMe, Accessed at: 2021-02-09.

[6]https://www.evernym.com, Accessed at: 2021-02-09

[7]https://github.com/alastria/alastria-wallet, Accessed at: 2021-02-09.

[8]https://developer.arm.com/ip-products/security-ip/trustzone, Accessed at: 2021-02-16.

[9]https://sovrin.org/, Accessed at: 2021-02-16.

[10]uPort is not Serto https://www.serto.id/, Accessed at: 2021-02-16.

ering various aspects such as sovereignty, storage-control, security and privacy besides others. This work also evaluates the storage of sensitive data which leads to the digital wallet.

In contrast, this work has the main focus on achieving the LoA high and considering more than only the tamper resistant storage requirements to achieve the LoA high.

4 CONCEPT

This section details our proposed concept, including involved actors, requirements, and a protocol description. Figure 2 illustrates the high-level architecture including actors, phases and main process flows.

4.1 Actors

Prover. A prover represents a user that wants to authenticate towards a verifier.

Identity Wallet. The identity wallet is software, which can also utilize hardware like a secure element to store key material, used on a mobile phone to manage key material as well as the verifiable credentials (VCs) of the prover.

Hardware Key. This hardware key represents a tamper-resistant hardware device in possession of the prover where the secret key of the prover is securely created, stored and used.

Registration Authority. The registration authority is responsible for adding the prover to the SSI system including registering the public keys. To authenticate a prover, it consults a trusted IdP. The role of registration authority can be performed by any node in the distributed ledger network.

Identity Provider (IdP). The IdP is responsible for authenticating the prover as well as to issue identity data for the prover. The registration authority utilizes the IdP for authenticating the prover.

Verifier. In our concept, the verifier represents a service provider (SP), like an online eGovernment service, or a party where the prover wants to perform authentication, for instance a police officer.

SSI Network. The SSI network serves as decentralized public key infrastructure (DPKI) in which semi-trusted nodes host a copy of the permissioned ledger. A consensus mechanism is used to agree on what is added to the DL.

4.2 Requirements

This subsection states the compiled system requirements a system needs to fulfill in order to achieve the LoA high. The requirements are the result of our evaluation of the two relevant LoA standards: the eIDAS implementation act (Commission, 2015) as well as the ISO 29115 standard (International Organization for Standardization (ISO), 2013). These standards cover the four main phases (I) enrollment, (II) electronic identification mean management, (III) authentication and (IV) management and organization.

We chose the two standards for evaluation over NIST (National Institute of Standards and Technology (NIST), 2020), since the eIDAS implementation act plays a vital role within Europe and provides a cross-border recognizable LoA. We evaluated ISO 29115 in addition since it provided the foundation of the eIDAS implementation act.

After assessing the two standards, we compile the following requirements **R** which have to be met in order to achieve the *LoA high*.

R1 Binding of Identity Data to the Prover. The identity proving used for this binding must fulfill *LoA high*. The binding is based on a unique identifier representing the prover. The binding has to follow nationally recognized procedures.

R2 Tamper-resistant Storage of the Key Material. The key material used for prover authentication must be stored in a tamper-resistant way to prevent it from misuse or theft.

R3 Ensure the Validity of the Identity Data. The verifier must be able to verify the status of the identity data, such as the revocation status.

R4 Authentication Mechanism. The authentication mechanism must utilize multi-factor authentication to strengthen the security and to prevent attacks such as guessing, replay attacks, communication manipulation and offline attacks.

R5 Identity Proving. The identity proving must be based on already existing electronic identification means that fulfill the same LoA and must not be repeated in person.

R6 Issuance of Identity Data. It must be ensured that the issued identity data are delivered only to the corresponding prover.

R7 Revocation of Identity Data. The prover must be able to revoke the related identity data used for authentication.

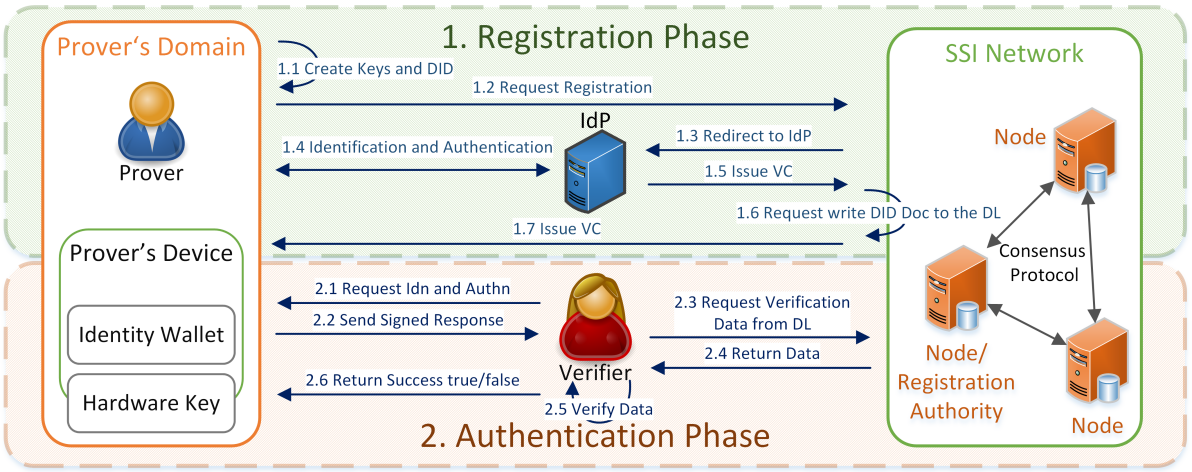

Figure 2: High-level architectural overview of our solutions consisting of its two main phases: 1. the registration phase and 2. the authentication phase.

4.3 Architecture

This subsection details our architecture, including a description of the registration and authentication phase as well as a formal definition of the protocols. The overall architecture including its actors, two phases and main process flows is depicted in Figure 2.

Phases. This paragraph introduces the generic protocol for the two main phases, first the registration and second the authentication phase.

In the **registration phase**, the DID document of the prover is registered on the DL. Protocol 1 shows this phase in a formal way. To start the process, the prover creates the necessary key material, DID, and DID document using their identity wallet on a smartphone and a secure hardware token.

The prover uses their phone wallet to create key material based on parameters provided by the registry. To achieve LoA high, the wallet creates two key pairs: The first pair sk_{uw}, pk_{uw} is created on the smartphone, while the second key pair sk_{uhw}, pk_{uhw} is created using a special tamper-resistant hardware, which secures the secret key and only exports the corresponding public key pk_{uhw}. To prove that the second key pair was actually generated on the secure hardware, this hardware device also generates a key attestation att_{uhw} for the generated key. After generating those two key pairs, the prover's wallet uses the two public keys pk_{uw}, pk_{uhw} to generate a DID DID_u (as described above) and corresponding DID document Doc_u. This DID document contains both public keys pk_{uw}, pk_{uhw}, the device attestation att_{uhw}, and is signed using both secret keys.

The registration phase continues on the registry, which receives the signed DID document from the prover and verifies the signatures using the encapsulated public keys. The registry also verifies the key attestation to ensure a proper hardware was used, and that the key was really generated on this hardware. The steps laid out by the WebAuthn specification for *Registering a New Credential* (W3C, 2020d) are being followed. Additionally, the certificate chain of the certificate, which was used for signing the key attestation, is build to ensure that it was issued by a trusted certificate authority (CA).

To authenticate the prover and to obtain related attributes, the registry forwards the prover to a trusted IdP. At the IdP, the prover performs identification and authentication, using an existing identity system with a sufficiently high LoA (for example a government ID). After doing so, the IdP uses the prover's identity to obtain the identity attributes $\{a_1, ..., a_n\}$. Those attributes are serialized into a verifiable credential VC_u and signed by the IdP using its DID DID_{IdP}'s secret key sk_{IdP}. The VC also contains the DID of the prover DID_u to link the VC to their identity. Afterwards, the IdP encrypts the VC with the public key of the prover, signs it again and sends it back to the registry, which verifies the signature on the encrypted VC. This way, the registry cannot learn the sensitive identity data of the prover. The registration phase concludes after the registry requested to write the DID document to the DL, and send the identity VC to the prover, who decrypts and stores it in their identity wallet.

In the **authentication phase**, the prover shows the identity VC to a verifier and proves ownership of the corresponding DID. Protocol 2 shows the steps of this process in a formal way..

To start this phase, the verifier creates a random

Registration:

on prover's wallet

1. send registration request to registry (node)
2. receive keys creation parameters $params_{key}$
3. generate wallet keys $sk_{uw}, pk_{uw} \leftarrow keygen()$
4. request hardware key generation including $params_{key} = \{param_1, ..., param_n\}$

on Hardware Key

5. request prover interaction
6. generate hardware key $sk_{uhw}, pk_{uhw} \leftarrow keygen()$
7. generate key attestation $att_{uhw} \leftarrow attest(pk_{uhw})$
8. returns pk_{uhw}, att_{uhw}

on prover's wallet

9. create identifier $DID_u \leftarrow genDID(pk_{uw}, pk_{uhw})$
10. create DID document $Doc_u \leftarrow genDoc(DID_u, pk_{uw}, pk_{uhw}, att_{uhw})$
11. sign DID Doc $\sigma_{Doc_u} \leftarrow sign(Doc_u, sk_{uw})$, $\sigma'_{Doc_u} \leftarrow sign(Doc_u, sk_{uhw})$
12. send key creation response to registry including Doc_u, σ_{Doc_u} and σ'_{Doc_u}

on Registry

13. check signatures on response $verify(\sigma_{Doc_u}, Doc_u, pk_{uw}) \wedge verify(\sigma'_{Doc_u}, Doc_u, pk_{uhw}) = true$
14. check key attestation of hardware key $verify(att_{uhw}, pk_{uhw}, pk_{device}) = true$ and authenticate pk_{device} using trusted pk_m of the hardware manufacturer.
15. forwad prover to IdP for authentication

on IdP

16. processes prover authentication using existing ID, retrieve person attributes $\{a_1, ..., a_n\}$
17. create $VC_u \leftarrow genVC(A_u, DID_u, DID_{IdP})$ where attributes $A_u = \{a_1, ..., a_n\}$
18. generate signature $\sigma_{VC} \leftarrow sign(VC_u, sk_{IdP})$
19. issue VC_u, σ_{VC} to registry

on Registry

20. verify signature $verify(\sigma_{VC}, VC_u, pk_{IdP}) = true$
21. request write of Doc_u to the DL
22. issue signed VC_u to prover

Protocol 1: Registration Protocol.

challenge c, signs it using its secret key sk_v, and sends c, σ_c together with DID_v the to the prover.

The prover retrieves the verifier's public key pk_v from the DL and uses it to verify the signature on the challenge. To prove ownership of their own DID, the prover signs the challenge using both prover keys sk_{uw}, sk_{uhw}. Since this is only possible if the prover is in possession of the phone as well as the secure token, and those factors comply with the stated requirements.

The prover sends both signatures as well as the signed VC to the verifier, which uses the DID from the VC to retrieve the prover's DID document. To ensure the prover is in possession of the secret keys corresponding to the DID, the verifier then verifies the signatures on the challenge.

Additionally, the verifier also checks if the hardware device used by the prover is trusted, and verifies if the hardware key was actually created by this device. This is checked the same way as during the registration phase by using the attestation data from the DL and its own PKI of trusted manufacturers. To ensure that the VC was issued by a trusted IdP, the verifier further checks the issuer. The authentication phase concludes with a revocation check of the DID and VC using the DL.

If all checks passed, the verifier retrieves the identity attributes from the now trusted VC for further processing.

Authentication:

on verifier

1. create challenge c and sign $\sigma_c \leftarrow sign(c, sk_v)$
2. send authentication request to the prover including c, σ_c and DID_v

on Prover's wallet

3. request $Doc_v \leftarrow resolve(DID_v)$
4. validate signature $verify(\sigma_c, c, pk_v) = true$
5. sign challenge $\sigma'_c \leftarrow sign(c, sk_{uw})$ and $\sigma''_c \leftarrow sign(c, sk_{uhw})$
6. send response including $DID_u, \sigma'_c, \sigma''_c, VC_u, \sigma_{VC}$ and σ'_{VC}

on Verifier

7. request $Doc_u \leftarrow resolve(DID_u)$
8. check DID ownership $verfiy(\sigma'_c, c, pk_{uw}) \wedge verify(\sigma''_c, c, pk_{uhw}) = true$
9. check key attestation of pk_{uhw} (same as in Protocol 1, Step 14)
10. request $Doc_{IdP} \leftarrow resolve(DID_{IdP})$
11. check identity data $verify(\sigma_{VC_u}, VC_u, pk_{IdP}) = true \wedge DID_u \in VC_u$
12. check validity status of VC $checkRevocation(VC_u) = true$

Protocol 2: Authentication Protocol.

DID Creation Definition. In our proposed architecture, we define a DID creation method, based on the DID specification (W3C Working Draft, 2019), to ensure that both public keys are directly linked to each other and also to ensure uniqueness. Both of the keys are stored on different devices, and one of the device is a tamper-resistant hardware. Since verifiers use the user's DID to retrieve the required key material needed to verify ownership proofs and credentials, we utilize our DID definition to support two keys. We achieve this by concatenating the public part of the mobile wallet key with the hardware public key:

$$genDID(pk1, pk2) = did : SSI : pk1 : pk2 \qquad (1)$$

The resulting identifier are separated in different parts by utilizing the colon " : ". The first part is the static string *did*, which represents the type of identifier, decentralized identifier (DID) in this case. *SSI* represents the DID method, which refers to the used ledger. $pk1$ represents the public key of the mobile phone and $pk2$ the public key of the hardware key.

5 IMPLEMENTATION

This section details the concrete instantiation of our generic architecture starting with an overview of the implemented components in Section 5.1. To preserve user's privacy, our implementation utilizes accumulator-based revocation, which is detailed in Section 5.3. Additionally, an overview of the interaction between the components of our demonstrator in Section 5.2. This implementation shows the feasibility of our concept.

5.1 Components

In this subsection, we detail the components of our implementation. For the proof-of-concept (PoC), we implement the mobile phone wallet including the usage of the hardware token. Additionally, the PoC further consists of a webserver used as IdP, registry and service provider (SP). Since our architecture and PoC are focused on the key handling and the authentication protocol, we implement this functionality and mocked parts which are not directly in focus such as the integration in a DL and IdP. All components communicate with each other using HTTPs redirects and callbacks in the prover's browser.

Wallet Application. For the mobile wallet, we implement a mobile application for iOS 14. The wallet leverages the iPhone's internal secure enclave[11] for generating and storing the wallet keys sk_{uw} and pk_{uw}. It communicates with an external hardware key over the phone's Lightning plug to generate and store the second key pair and corresponding key attestation. The public keys pk_{uw} and pk_{uhw} are utilized to generate DID_u and it's corresponding DID document, which is later transmitted to the registry.

Hardware Key. To achieve LoA high we extend the user's wallet with a hardware token. This hardware token is used to generate the second key pair sk_{uhw} and pk_{uhw} together with a key attestation att_{uhw} using the device's attestation certificate.

For our demonstrator, we chose the Yubikey 5Ci token[12]. This token was selected because of its iPhone compatibility, it is FIDO2-licensed, and the manufacturer offers extensive documentation and SDK support. By using an open authentication standard instead of implementing a custom protocol we greatly improve adaptability thanks to a variety of available FIDO2 devices. Even if it turned out that the Yubikey 5Ci does not fulfill a specific authority's security requirements, a different FIDO2 key could easily be utilized. Using key attestations, every registry and SP can decide what manufacturer and device they deem secure for their use case offering additional flexibility.

Registry, IdP and Service Provider. To simulate the interaction of our wallet implementation with registry, IdP, and SP, we develop a server implementation of these components mocking some of the functionality. These server components are developed as lightweight go applications and perform the operations described in Section 4.3 as well as the simulation of a DL.

5.2 Phases

Our concept consists of two phases. First, the registration phase in which the user creates keys and has to proof his identity towards an IdP and finally gets the VC issued. The second phase represents the authentication process towards a verifier.

[11] https://support.apple.com/guide/security/secure-enclave-overview-sec59b0b31ff/web, Accessed: 2021-02-02

[12] https://www.yubico.com/at/product/yubikey-5ci/, Accessed: 2021-02-02

5.2.1 Registration Phase

After initializing the registration, the wallet receives a key creation request from the registry. This request specifies what type of FIDO2 hardware key may be used for the SSI creation. First, the wallet forwards the specification to the hardware token, which creates a second key pair (sk_{uhw}, pk_{uhw}). The resulting credential creation response from the token contains the attestation response with the attestation object (W3C, 2020c) and the collected client data (W3C, 2020b). The attestation object contains the attestation att_{uhw} and the newly created pk_{uhw}, among other information. The collected client is a JSON object containing a random challenge generated by the server, the origin of the key creation request as well as the type of the request.

After the creation of hardware key material, the wallet uses the secure enclave to create the wallet key pair (sk_{uw}, pk_{uw}), protected by facial recognition.

The resulting public keys pk_{uhw} and pk_{uw} are then combined into a DID and DID document in the following way: Since the two authentication methods need to be uniquely identifiable within the DID document, the ids are extended with *#wallet-key* and *#H5LVnEVQG...* respectively. The former is simply a hardcoded identifier while the latter is the credential id returned by the authenticator which is required for the authenticator to be able to look up the key in its internal storage.

Before sending the DID document back to the registry, it needs to be signed with both keys. Thus, the wallet creates a FIDO2 credential assertion request, which contains the credential id. Instead of the hash of collected client data (as specified by FIDO2), the wallet passes a hash of the DID document for signing to the token, resulting in sig_{uhw}. The second signature sig_{uw} is created by signing the DID document hash using the wallet key sk_{uw}. Finally, the full credential creation response and the signed DID document are sent to the registry.

After successful verification of the credential creation response and signed DID document, the registry forwards the user for authentication to the IdP.

In our implementation, the authentication with the IdP is simulated, but in a real-world system, the IdP would authenticate the user with means to guarantee an LoA high. The IdP then prepares VCs using the JSON Web Token proof format as specified in the VC Data Model (W3C, 2020a) and signs them using its private key sk_{IdP}. Next, the IdP uses the pk_{uw} to encrypt the VC for the user and signs the hash of the encrypted VC again in order for the registry to very the signature.

Finally, the registry stores the signed DID document and the authenticator attestation att_{uhw} on the ledger and sends the VCs to the user. The user's wallet receives the VCs and stores them on the mobile device's local filesystem.

5.2.2 Authentication Phase

Similar to the registration, the authentication is initialized by the wallet with a request to the SP. The SP then resolves the user's DID contained in the request into a signed DID document by contacting the DL. After confirming that signatures on the DID document are correct, att_{uhw} is retrieved from the DID document and verified. If it is valid and was created by a trusted manufacturer, the SP extracts the credential ID from the DID document and responds with a challenge in form of a credential assertion response to the wallet.

The wallet needs to sign the challenge and the VCs used for login with both the wallet key sk_{uw} and the hardware token key sk_{uhw}. Therefore, it forwards the assertion response and collected client data to the hardware key. This time, the collected client data additionally contains a field for the VCs to also sign them as part of the assertion. To ensure user consent, the user needs to approve this signature via a user presence check by pressing a button on the token. After signing with the hardware key, the collected client data is also signed with the wallet key. To authorize the secure enclave to sign the request, the user needs to unlock the device using facial recognition.

The wallet concludes the authentication by creating a non-revocation proof for the used VCs (cf. Section 5.3) and submits it alongside the response to the SP.

5.3 Revocation

To ensure that the VCs used for authentication are still valid, it is important that a verifier retrieves revocation information from the DL. Our revocation implementation is based on the work of (Boneh et al., 2019; Li et al., 2007).

Accumulator-based revocation is a type of revocation that makes use of cryptographic accumulators. A cryptographic accumulator is a commitment to a set of values of a constant size. This is useful to store it on a DL where space is limited. The accumulator can be queried for a contained element without revealing any of the other elements. To use it for revocation, we make every element in the accumulator correspond to a non-revoked VC. As the issuer revokes VCs, the accumulator is updated. If a user wants to revoke a VC,

they need to contact the issuer to update the accumulator.

Our used accumulator is a RSA-based accumulator. For a given RSA group with modulus N and generator g, let C be a set of primes. Each prime corresponds to a VC. Further, let p_c be the product of all primes in C. Then the value of the accumulator A is $A = g^{p_c} mod N$. The membership proof w_x of prime x, which proofs that A contains x is given by $w_x = g^{\frac{p_c}{x}} mod N$. The proof can be verified against A as follows: $w_x^x mod N = g^{\frac{p_c}{x} x} mod N = g^{p_c} mod N = A$.

We refer to these prime numbers in S as validity tails (VT). Since the VTs need to be known to be able to compute w_x, but the issuer of a VC shouldn't be contacted every time the VC is used, the set of VTs for the accumulator never changes. The issuer publishes the accumulators' VTs in the form of a tails file to be retrieved for offline proof creation. Each index in the tails file corresponds to exactly one VT.

This also means that whenever the accumulator is updated on the DL, the revoked tail indices need to be shared on the DL. It indicates what VTs were removed from the accumulator.

During the authentication phase (cf. Protocol 2), the wallet prepares non-revocation proofs for the VCs it needs to transmit to the SP. To achieve this, it fetches the validity tails for the corresponding cryptographic accumulators from the DL using the VC's accumulator id. The accumulator id is contained within the signed VCs together with the VC's corresponding validity tail. The wallet uses the validity tails currently contained within the accumulator as indicated by the response to compute the witness.

The resulting non-revocation witnesses and the credential assertion response containing the hardware key signature are sent to the SP alongside the wallet signature and the collected client data. Next, the SP verifies the signatures on the collected client data and extracts the VCs. For each VC, it fetches the latest accumulator value from the ledger and verifies that the transmitted witness' and validity tail's RSA product match its value. If that is the case for both VCs, they are still valid (not revoked) and the authentication is successful.

DL nodes allow fetching the validity tails of a cryptgraphic accumulator specified within a VC to compute the witness for a non-revocation proof. Each of these tails contain a flag indicating whether a specific validity tail is currently contained within the accumulator. In a production use-case, the tails would not be stored directly on the ledger, but be provided by the VC issuer. Only a bitfield indicating whether a validity tail's inclusion state changed needs to be written to the ledger.

6 EVALUATION AND DISCUSSION

6.1 Requirements to Concept Mapping

This subsection details how our concept addresses the requirements in order to achieve a LoA high. First, we list and detail the measures **M** from our concept used to meet our defined requirements. Second, we map those measures to the related requirements **R**, depicted in Table 1, and show the fulfillment of all requirements.

M1 Identity Proving. The identity proving describes processes how the user proves his identity towards an authority during the registration process. In our concept, we are re-using previously performed identity proving by redirecting the prover during the registration to an IdP for authentication.

M2 Usage of Authorative IdP. During the registration process, the prover is being redirected to a trusted IdP for performing identification and authentication. After successfully authenticating the prover, the authoritative IdP issues the VC containing the prover's as well as the issuers DID with a LoA high.

M3 Authentication at IdP. The prover performs strong authentication towards the IdP to receive a VC.

M4 Hardware Key. Our concept is based on a hardware key stored in a tamper resistant way and further serves as additional authentication factor (possession).

M5 Hardware Key Attestation. The hardware key utilized in this work supports key attestation. This way, the hardware device can create a key attestation, which can further be used to verify that the hardware key was really created on certain hardware.

M6 Revocation Mechanism. Our concept provides a privacy-preserving revocation concept where the prover or issuer can revoke their credentials and verifier can check the validity of the data used during authentication.

M7 Mobile Phone with Biometry Support. In our proposed architecture, we utilize a mobile phone based wallet. The wallet key is protected by Biometric authentication like fingerprint or face recognition; this process increases the binding to the prover. Additionally, the mobile phone itself is another authentication factor (factor possession).

M8 Issue Encrypted VC. In our registration protocol, we propose that the issued VC is encrypted for the related prover. Even though the registration authority receives the encrypted and signed VC as proof of successfully identification and authentication, it cannot learn sensitive identity data of the prover.

Table 1: This table represents the mapping of requirements **R** to measures **M**.

	R1	R2	R3	R4	R5	R6	R7
M1	✓				✓		
M2	✓				✓		
M3	✓			✓			
M4		✓		✓			
M5		✓					
M6			✓				✓
M7	✓			✓			
M8						✓	

6.2 Security Evaluation

This subsections briefly summarizes the security evaluation of our proposed architecture. The main aspect covered by our evaluation is that an attacker should not be able authenticate at a verifier by using a VC belonging to someone else. Also, the verifier must be able to verify the validity of the identity data.

Identity Data Verification. The identity data in form of a VC are directly linked to the related prover by including its DID into the VC. Further, the prover's DID is linked to the key pairs in the prover's possession. Only after the user has performed strong authentication at an IdP (trust service provider according to (Commission, 2015)) and additionally proved the ownership of the DID (involving both key pairs), the DID of the prover is added to the VC and issued in encrypted form to the prover. Also, the verifier checks the validity status of the identity data by performing a revocation check of the VC. Additionally, since the VC is encrypted for the prover, an attacker could not decrypt it without having access to the private key of the prover.

Authentication Means. The trust in the authentication means is based on multi-factor authentication: The wallet key is protected by a biometric factor and thus linked to the related person. Additionally, in order to perform authentication, the prover has to be in possession of the smartphone and also of the hardware key. The provided VC is directly linked to the DID and thereby to both key pairs.

6.3 Discussion

Hardware Token. One one hand, the additional hardware token offers stronger trust and security in the authentication method. On the other hand, an additional device represents a hustle and usability drawback. After our analysis, we are convinced that without this additional tokens a LoA high cannot be achieved – even when creating and storing the key on secure elements on the mobile phones. The keys could be misused when e.g. the device is rooted.

Missing Key Backup. In many cases, a drawback of hardware tokens/keys is that they cannot be backed up. This is also the case for other authentication means: when loosing a physical ID card or a passport a user has to request a new one.

Usage of Standard IdP. In our concept, the IdP issues a VC for the user. Nevertheless, maybe not all IdPs want to or are capable of issuing VCs. In this case, it is possible to apply the work of (Abraham et al., 2019) and simply re-use an identity assertion and prove attributes from it. This way, our system can be applied to almost any existing IdM system.

7 CONCLUSIONS

In SSI, identity wallets are used to store and manage digital identity data of corresponding users as well as related cryptographic key material. Those wallets lack an LoA high, which is necessary to enable SSI systems for advanced use-cases such as accessing public administration services, or to establish a digital driver's license.

In our work, we tackle this problem by defining system requirements to achieve an LoA high. These requirements are based on an assessment of common LoA standards. We further defined a generic concept that can be used to achieve LoA high utilizing a mobile phone-based wallet. Additionally, we implemented a demonstrator showing the feasibility of our system utilizing a privacy-preserving revocation approach. In the evaluation, we map the measures, provided by our concept, to the requirements to highlight their fulfillment. We further performed a security evaluation of our system as well as discussed the design decisions. Thus, our generic concept can be utilized by e.g. other wallet projects in order to achieve high or to use the requirements and the measures to evaluate their system.

ACKNOWLEDGEMENTS

This work was supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement No. 871473 (KRAKEN).

REFERENCES

Abraham, A. (2017). Self-Sovereign Identity - Whitepaper about the Concept of Self-Sovereign Identity including its Potential. https://technology.a-sit.at/en/whitepaper-self-sovereign-identity/, Online, Accessed: 2021-02-10.

Abraham, A., Hörandner, F., Omolola, O., and Ramacher, S. (2019). Privacy-preserving eid derivation for self-sovereign identity systems. In *Information and Communications Security ICICS 2019, Beijing, China*, volume 11999 of *LNCS*, pages 307–323. Springer.

Alastria (2021). Alastria Builds Future. https://www.alastria.io/en/, Online, Accessed: 2021-02-09.

Boneh, D., Bünz, B., and Fisch, B. (2019). Batching techniques for accumulators with applications to iops and stateless blockchains. In Boldyreva, A. and Micciancio, D., editors, *Advances in Cryptology – CRYPTO 2019*, pages 561–586, Cham. Springer International Publishing.

Commission, T. E. (2015). Commission implementing regulation (eu) 2015/1502. Online, Accessed: 2021-01-20.

Dai, W., Wang, Q., Wang, Z., Lin, X., Zou, D., and Jin, H. (2021). Trustzone-based secure lightweight wallet for hyperledger fabric. *Journal of Parallel and Distributed Computing*, 149:66–75.

Dong, C., Wang, Z., Chen, S., and Xiang, Y. (2020). BBM: A blockchain-based model for open banking via self-sovereign identity. In Chen, Z., Cui, L., Palanisamy, B., and Zhang, L., editors, *Blockchain - ICBC 2020 - Third International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18-20, 2020, Proceedings*, volume 12404 of *Lecture Notes in Computer Science*, pages 61–75. Springer.

Foundation, T. L. (2021). dizme. https://www.dizme.io, Online, Accessed: 2021-02-09.

Houtan, B., Hafid, A. S., and Makrakis, D. (2020). A survey on blockchain-based self-sovereign patient identity in healthcare. *IEEE Access*, 8:90478–90494.

International Organization for Standardization (ISO) (2013). ISO/IEC 29115:2013(en) Information technology — Security techniques — Entity authentication assurance framework. Online, Accessed: 2021-01-20.

Iqbal, S., Irfan, M., Ahsan, K., Hussain, M. A., Awais, M., Shiraz, M., Hamdi, M., and Alghamdi, A. (2020). A novel mobile wallet model for elderly using fingerprint as authentication factor. *IEEE Access*, 8:177405–177423.

Jolocom (2021). Jolocom Decentralized identity & access management. https://jolocom.io/, Online, Accessed: 2021-02-09.

Kondova, G. and Erbguth, J. (2020). Self-sovereign identity on public blockchains and the gdpr. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 342–345, New York, NY, USA. Association for Computing Machinery.

Li, J., Li, N., and Xue, R. (2007). Universal accumulators with efficient nonmembership proofs. In Katz, J. and Yung, M., editors, *Applied Cryptography and Network Security*, pages 253–269, Berlin, Heidelberg. Springer Berlin Heidelberg.

Liu, Y., Lu, Q., Paik, H., Xu, X., Chen, S., and Zhu, L. (2020). Design pattern as a service for blockchain-based self-sovereign identity. *IEEE Softw.*, 37(4):30–36.

Manski, S. (2020). Distributed ledger technologies, value accounting, and the self sovereign identity. *Frontiers Blockchain*, 3:29.

Naik, N. and Jenkins, P. (2020). Self-sovereign identity specifications: Govern your identity through your digital wallet using blockchain technology. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 90–95.

National Institute of Standards and Technology (NIST) (2020). NIST Special Publication 800-63 - Digital Identity Guidelines. Online, Accessed: 2021-02-05.

Sporny, M., Longley, D., and Chadwick, D. (2019). W3C Verifiable Credentials Data Model 1.0. Online, Accessed: 2021-01-20.

The FIDO Alliance (2020). Fast Identification Online (FIDO) - Specifications . Online, Accessed: 2021-01-20.

W3C (2020a). Verifiable credentials data model 1.0. https://www.w3.org/TR/vc-data-model/#proof-formats, Online, Accessed: 2021-01-27.

W3C (2020b). Webauthn2: Collectedclientdata. https://www.w3.org/TR/webauthn/#dictdef-collectedclientdata, Online, Accessed: 2021-01-27.

W3C (2020c). Webauthn2: Packed attestation statement format. https://www.w3.org/TR/webauthn/#sctn-packed-attestation, Online, Accessed: 2021-01-27.

W3C (2020d). Webauthn2: Registering a new credential.

W3C Working Draft (2019). Decentralized Identifiers (DIDs) v1.0 - Core Data Model and Syntaxes. Online, Accessed: 2021-01-12.

Zwattendorfer, B., Zefferer, T., and Stranacher, K. (2014). An overview of cloud identity management-models. In *WEBIST (1)*, pages 82–92. SciTePress.