Revocable and Offline-Verifiable Self-Sovereign Identities

1st Andreas Abraham, 2nd Stefan More, 3rd Christof Rabensteiner, 4th Felix Hörandner
Institute of Applied Information Processing and Communications (IAIK)
Graz University of Technology, Graz, Austria
{aabraham, smore, crabensteiner, fhoerandner}@iaik.tugraz.at

***Abstract*—Identity management systems enable users (i.e., provers) to authenticate and provide attributes to verifiers by using certified credentials obtained from an authority. To accept such a credential, verifiers require information on whether the presented credentials are still valid or if they have been revoked. Up-to-date revocation information can be obtained from a revocation database; however, this requires that the verifier or prover is online. The problem becomes more interesting in the offline case when the prover (e.g., citizen) and verifier (e.g., police officer) do not have an Internet connection to query the revocation status of the presented credential (e.g., digital driver's license).**

In this paper, we extend the Self-Sovereign Identity (SSI) model to support both revocation as well as offline-verification. Our concept introduces attestations of validity for a point in time, which are issued by the SSI network for credentials that have not been revoked, i.e., added by authorized entities to a revocation list. The concept aims to be generic so that it can be used for various use cases, e.g., by giving users the control over the frequency of re-attestation. To show our concept's feasibility and practicality, we developed and evaluated an implementation that includes an efficient and privacy-preserving showing of credentials using non-interactive zero-knowledge proofs, all while being offline.

***Index Terms*—identity management, self-sovereign identity, offline authentication, revocation, distributed ledger**

I. Introduction

Previously paper-based identification (e.g., governmental id cards) has been migrated towards digital identity management solutions, while other domains, such as IoT, inherently require digital approaches [1], [3]. Besides identifying and authenticating the subject as well as providing certified attributes, an essential requirement towards an identity management system is to answer the questions: Is the provided information still valid? Is the certification mechanism still trustworthy (or has the signing key been stolen)? Such validity information is necessary to enable the receiver to operate or decide based on this information. Current digital solutions are able to address this issue only if their participants are online and, therefore, able to query the system for revocation status information (e.g., CRL [8], OCSP [20]). However, if the participants are offline (e.g., a police officer in a rural area), the question about the revocation status of data becomes much more challenging.

With paper-based identity documents, like passports, people are able to identify themselves and present attributes, such as name, date of birth, or nationality, without the need for an Internet connection. It is also possible to revoke such documents with some effort and time by physically destroying the document. With smartphones on the rise, governments worldwide develop systems that bring these credentials onto mobile devices in the form of electronic identities (eIDs), relieve users from the burden of carrying identity documents, and improve convenience in a digital age.

Self-Sovereign Identity (SSI) is a manifestation of a user-centric digital identity management model [2]. SSI puts the control over the identity data in the hand of the data subject by issuing and handing over a credential that contains certified attributes. This credential needs to be digitally signed by a qualified authority in order to protect the integrity and authenticity of the contained attributes and of the origin. The credential is stored by the user and, upon authentication, presented to the verifier. The verifier authenticates the credential by verifying its signature and by checking if the data has been revoked. In order to perform the verification process, the verifier relies on the distributed ledger (DL) of the SSI system for the key material and the revocation status.

Problem: Supporting both revocation as well as verification in an offline setting remains a challenge. In contrast to the paper-based world, it is not feasible to take back and destroy digital information. Instead, current digital systems follow two approaches: First, these systems may rely on queries against a revocation database, which are not possible without an Internet connection by either the sender or the receiver. Second, the system might issue credentials with a limited lifespan, which requires users to frequently re-obtain them. This process places a burden on the credential issuer.

Contribution: In this paper, we present an extension to the SSI concept that supports revocation while additionally enabling authentication and revocation-verification in settings where the participants are fully or partially offline.

In our concept (c.f. Section IV), the nodes of the SSI network issue attestations about the revocation status of a credential to its subject, which they sign using multi-signatures. Such an attestation is only issued if the credential has not been revoked by the credential's issuer or subject, who would instruct the SSI network to mark the revoked credential in a revocation list within the distributed ledger. Furthermore, these attestation is timestamped to signify that the credential was still valid, similar to OCSP-stapling [12].

Our concept builds a bridge between systems that rely on queries to a revocation database, and systems that require short-lived credentials while leveraging the best of both worlds: First,

978-1-6654-0392-4/20/$31.00 ©2020 IEEE
DOI 10.1109/TrustCom50675.2020.00136

it offers the possibility to obtain attestations of validity at a point in time for credentials that would rely on online revocation checks. Second, our concept provides a straight-forward process to obtain attestations at a frequency that can be adjusted to the user's needs and her use case, rather than burdening the user and issuer with re-issuing short-lived credentials.

Additionally, we present an implementation (c.f. Section V) as well as an evaluation (c.f. Section VI) to show that our concept is both feasible and practical. Selective disclosure is a desired property in identity management to preserve the users' privacy by revealing only the necessary parts of a credential. Our implementation employs *bulletproofs* [7] as zero-knowledge proofs to demonstrate that this privacy-enhanced showing of attributes is also compatible with our concept for revocable yet offline-verifiable SSI credentials.

II. Related Work

Common authentication protocols rely on an available Internet connection in order to successfully perform revocation checks. In a typical authentication process, a user authenticates at a service provider (SP) using the related digital identity e.g., on the mobile phone. The participants of the process follow an identity protocol like SAML 2 [16], OpenID Connect [17] or a proprietary protocol. All of these protocols have in common that, at some point, one or even both of the parties have to be online. In case of SAML or OpenID Connect, when authenticating towards an SP, an IdP is contacted to issue either an identity assertion or an authentication token. This assertion or token is used by the SP to get access to the related identity data. In contrast, our work proposes a concept that enables the fully offline authentication of user attributes. Additionally, our concept enables an SP to verify credentials by providing revocation information.

In SSI systems, communication is commonly based on peer-to-peer interaction between so-called agents [14]. This system does per se not require an Internet connection for authentication like performing the DIDComm [23], [24] protocol. This is due to the fact that, within SSI systems, users can create and verify their self-issued digital identities on their own. Nevertheless, when trying to use sensitive services like government services, a certain degree of assurance - the so-called level of assurance (LoA) - is required. In contrast to the state-of-the-art, the proposed concept allows provers to use trusted identity data that is issued for authentication in an offline setting.

III. Preliminaries

Self-Sovereign Identity System: In traditional IdM models, identity data reside in data silos located at identity providers (IdPs, such as government agencies) or service providers (SPs, such as social networks). A user who wants to provide certain identity attributes needs to authenticate towards the SP. In these models, users only partially control their identity data and cannot easily delete or transfer data to another system.

SSI can be seen as the next step of the user-centric IdM model [25] with the advantage of not having to trust a central authority. In contrast to traditional IdM models, the SSI model

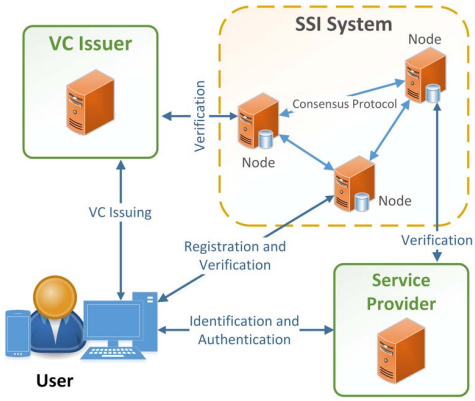

Fig. 1. Generic high-level architectural overview of an SSI system

grants the owner of a digital identity complete control over their identity data. Figure 1 depicts the main actors of an SSI system and their interactions: A core building blocks of an SSI system is the *distributed ledger* (DL). The DL stores the public information of users, including public keys. Thereby, the DL serves as a decentralized public key infrastructure (DPKI) and provides a transparent means to distribute keys and build trust. Another actor in the SSI system is the credential issuer: This issuer attests identity attributes of a user and issues *Verifiable Credentials* (VCs). Users receive their VCs and store them along with their private keys on their devices, which grants them full control over their identity. The last actor from Figure 1 is the SP. During authentication, the SP receives a VC from the user and verifies the VC with the help of the DL.

Consensus Protocol: In contrast to systems that are based on central trusted parties, Blockchains and DL based systems consist of nodes that are hierarchical equal without having a privileged central party. Even though this approach addresses trust and other issues, new challenges emerge. One challenge is that the nodes have to reach consensus about what data are written to the ledger. This challenge has been addressed through a variety of consensus protocols [19], related to different blockchain types such as public or private Blockchains, achieving different use cases.

SSI systems often utilize permissioned or private ledger; thus, they utilize either Byzantine fault tolerance (BFT) protocols or proof-of-authority (PoA) in which the authorized nodes also perform a BFT protocol. In the SSI system, the consensus protocol is responsible for determining if a node of the network is behaving faulty, by e.g. crashing, or malicious, by trying to intrude the network. This issue is addressed by performing certain steps redundant so that the network is resilient against those events.

Decentralized Identifiers and Documents: SSI systems rely on decentralized identifiers (DIDs) [22] to refer to users unambiguously. Users issue and register DIDs themselves without depending on a central issuing party. An example DID is `did:dl42:123456789`. DIDs start with the literal `did`, followed by a method which points to the DL (in the example

dl42), and the actual ID of the subject (123456789). DIDs can be resolved to DID documents, which contain information like public keys and service endpoints. DID and a key pair may be linked by using the public key as the ID or by using a part e.g. 16 bytes of the key.

IV. Concept

This section details the concept of offline verifiable SSIs. First, we introduce the actors, and next, we describe the individual phases, which happen before and during the offline authentication. Figure 2 depicts the architectural overview, including actors, interactions, and phases.

A. Actors

Our system consists of the following actors:

Prover: A prover (also referred to as user) wants to authenticate towards another party, which appears in a scenario where both participants are offline. For example, this process would replace an offline authentication of a citizen with a paper-based document.

Verifier: The verifier is a party to which the prover performs identification and authentication. Considering a traditional identity management model, the verifier would assume the role of a service provider (SP).

Devices: Devices represent the technical infrastructure used by the prover and the verifier in their interaction.
Such devices could typically be mobile general-purpose computing devices, like smartphones or tablets. Ideally, these devices support hardware-based protection of the cryptographic key material as well as credentials, which may rely on biometric authentication means that ensure the binding of devices to their users and prevent identity theft in case of device loss. Generally, being the prover requires more computational power, especially for performing cryptographic operations. The verifier's device can also be a constrained device responsible for giving access to e.g., a parking lot, since the verifier only performs the less computational expensive tasks. For simplicity's sake, in the remainder of this document, we will often refer only to prover and verifier instead of their devices.

Wallet: The wallet is an application that is installed at least on the prover's device. The wallet maintains the key material and the users' identity or other sensitive data mainly in the form of VCs. These data are cryptographically linked to the key material via DIDs, which enables the owner of the private keys to prove ownership over the DID and thereby of the linked VCs.

VC Issuer: A VC issuer (also referred to simply as issuer) creates verifiable credentials that include attributes for users. It is associated with the SSI network, having the ability to request writes. The VC issuer has to be recognized by the verifier in order to be able to make decisions based on the presented data

SSI System: SSI systems utilize distributed ledger technologies (DLT) to maintain digital identities without the need for a central trusted authority. The SSI system in this work consists of a network of semi-trusted nodes, which host a copy of the DL in a permissioned ledger system. The nodes build an trusted network based on the web-of-trust (WoT) concept. In the DL, the nodes also maintain a distributed revocation list.

B. Phases

Within our concept, we have identified five main phases: (1) The user registers at the system, and (2) obtains a verifiable credential about herself. Once necessary, (3) the user or the issuer may revoke the verifiable credential. Until the credential is revoked, (4) the system may generate attestations on such credentials' current validity. (5) Finally, parts of a credential are presented to the verifier. If the verifier is not online, it cannot query the SSI network about the credential's revocation status. Instead, we additionally forward the validity attestation (stating when the credential was valid) to enable the verifier to get an understanding of the credential's revocation status even if the verifier is offline. During verification in phase (5), the prover, the verifier, or both parties may be offline. Since the prover does not need to be online for revocation checks, we focus on the two latter cases (only the verifier is offline, both parties offline). Verifying the prover's identity by a police officer could be a use case when both parties are offline, whereas authenticating towards an electronic gate represents a use case in which only the verifier is offline.

In the following paragraphs, we describe our phases on a high level, while Table I elaborates on the process steps more formally.

(1) Registration: The user creates a key-pair and an identifier (DID). The DID is directly linked to the key-pair by utilizing a part of the public key (e.g., the first 16 bytes) or the whole public key as unique identifier in the DID. Consequently, the DID and public key are directly related. The key-pair for the DID is stored on the device protected via cryptography, e.g., encrypted by a key that is stored in the device trusted platform module and can only be unlocked by biometric authentication, such as fingerprint verification or facial recognition.

After creating the keys and the DID, the prover registers her DID document at the SSI system, which requires to prove the ownership of the DID. Finally, the nodes write the DID document, containing the user's public key, to the DL.

(2) Obtain Verifiable Credential: Next, the user wants to obtain a verifiable credential for a number of certified attributes from a VC issuer. The user proves ownership of her DID and may provide further evidence, which enables the issuer to determine her attributes. Finally, the issuer places the user's attributes, as well as the DID of user and issuer in a verifiable credential and signs this credential.

(3) Revocation: Verifiable credentials have to be revoked if the attributes within the credentials no longer apply to the user (e.g. the user is no longer an employee in a certain company), or for security reasons (e.g. if the key material of issuer or user is no longer sufficiently secure). Our system enables users as well as VC issuers to revoke previously issued credentials. In that case, they convince the SSI network nodes to add the

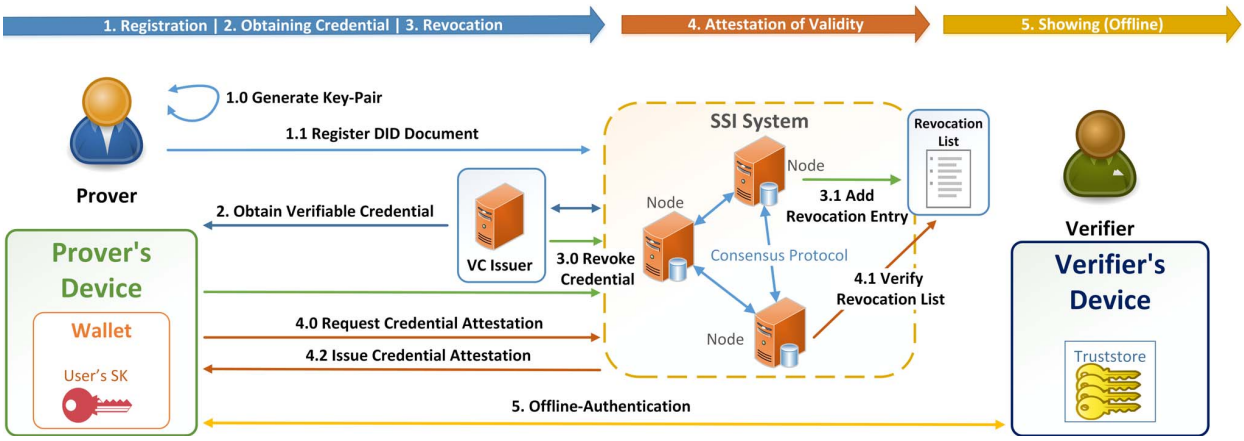

Fig. 2. Architectural Overview of the Proposed Architecture

credential to the revocation list. This approach clearly reflects the SSI spirit, in which the users are in full control over their own identity data.

(4) Attestation of Validity: The nodes of the SSI network generate assertions that – at the current moment – specified credentials are not revoked, i.e., that are not recorded in the revocation list. Such attestations help verifiers in assessing if a presented credential is (sufficiently recently) valid. Technically, if the credential is not in the revocation list, the nodes generate a multi-signature over a statement that identifies the credential as well as a current timestamp.

Different entities can trigger this attestation process: Users may request validity attestation for their credentials periodically, which enables them to control the re-attestation frequency according to the needs of their use cases. Alternatively, issuers may provide additional information on the recommended validity-time of attestations for their credentials, based on their knowledge of the type of data contained within the credentials. This validity-time information can then be used by the nodes of the SSI network to re-attest credentials in a batch process and make the fresh attestations available to the users.

The re-attestation period of the credential can be defined already when issuing the credential by the issuer. This period depends on the credential type like for instance, the minimal data set (MDS) consists of first and last name, birthdate, and a unique identifier. Most likely, these data do not change regularly; therefore, a validity time of one or even more days would be sufficient. A driver license credential might require a higher re-attestation interval in order to be accepted by a police officer. Nevertheless, the verifier decides during the offline authentication if the showed credential is sufficiently fresh.

The revocation and attestation process has to deal with the challenge that the nodes should not learn any possibly sensitive information contained within the user's credentials. However, we still have to ensure that only the subject or issuer are able to revoke credentials, while a link between the validity attestation and the presentation of a partial credential needs to remain. Possible approaches would be to employ revocation tokens (as we detail in Table I) or to record the hashes of issued credentials as well as their subject and issuer in an additional list at the ledger, which is checked in the revocation process.

(5-A) Fully Offline Showing: In this phase, users authenticate towards a verifier and present attributes, while both are offline, i.e., the prover as well as the verifier are without an Internet connection.

The prover sends a presentation of the credential containing the required attributes to the verifier. Furthermore, the prover also provides a previously retrieved attestation of validity for that credential.

The verifier initially verifies the prover's ownership over the presented DID as well as the received presentation of the credential, e.g. that it was issued by a trustworthy source, and that the presented attributes suffice for the verifier's use case. Additionally, the verifier inspects the attestation of validity, i.e. that this attestation is about the presented credential as well as that the timestamp of when validity was attested is sufficiently recent for the verifier's use case [13].

If mutual authentication is required, it is possible to perform our abstract model two times, once with reversed roles. First, the verifier takes the role of the prover to prove its identity and attributes (e.g. that it is a police officer), before the actual prover is sufficiently convinced to reveal its sensitive data (e.g. driver's license). On a technical level, this approach aligns with the DIDComm protocol [23], [24], where the verifier generates a QR code containing information to establish a DIDComm connection over Bluetooth or WiFi direct (or utilizing NFC). After the prover scans the QR code, a connection is established.

In an offline environment, the public keys required to verify the ownership of the DID cannot be gathered from the ledger. Instead, the prover supplies the public keys of itself and the issuer, which have to match the DIDs, as the DID consists of the first bytes of the key, or even contains the whole key. We rely on a trust store at the verifier, which contains the public keys of the SSI nodes that are used to verify the network's multi-signatures.

(5-B) Partially Offline Showing: If the prover is online,

TABLE I
PROTOCOL STEPS

General Parameters: Our protocol relies on a digital signature scheme SIG, a multi-signature scheme MSIG, as well as a non-interactive proof system Π.

(1) Registration:

on Prover, Issuer

1) generate key pair $(\mathsf{pk_P}, \mathsf{sk_P}) \leftarrow \mathsf{KeyGen}()$
2) create the identifier $\mathsf{DID_P}$ [22] from $\mathsf{pk_P}$ (whole key or first 16 bytes)
3) if necessary: provide evidence to SSI node (e.g. authenticate), which admit the user to be registered in the SSI network

on Node of the SSI system

4) verify prover's ownership of $\mathsf{DID_P}$
5) if necessary: verify evidence of prover
6) interact with other nodes to write $\mathsf{DID_P}$ to the distributed ledger

(2) Obtain Credential:

on Prover

1) request verifiable credential, by sending $\mathsf{DID_P}$ as well as any further evidence needed by the issuer (out-of-band)

on Issuer

2) verify prover's ownership of $\mathsf{DID_P}$
3) verify further evidence of prover
4) generate the verifiable credential cred', which includes the user's attributes $A = a_1, ..., a_n$, the prover's $\mathsf{DID_P}$, and the issuer's $\mathsf{DID_{Iss}}$
5) add the hash as identifier $\mathsf{id_{cred}} \leftarrow H(\mathsf{cred}')$ as additional attribute a_{n+1} to the credential, yielding cred
6) sign the credential $\sigma_{\mathsf{cred}} \leftarrow \mathsf{SIG.Sign}(\mathsf{cred}, \mathsf{sk_{Iss}})$
7) build a revocation token $\mathsf{rt} \leftarrow (\mathsf{id_{cred}}, \mathsf{DID_P}, \mathsf{DID_{Iss}})$
8) sign token $\sigma_{\mathsf{rt}} \leftarrow \mathsf{SIG.Sign}(s, \mathsf{sk_{Iss}})$
9) issue $(\mathsf{cred}, \sigma_{\mathsf{cred}})$ and $(\mathsf{rt}, \sigma_{\mathsf{rt}})$ to the user

(3) Revocation:

on Prover or Credential Issuer

1) authenticate to SSI system
2) send a request for revocation to the SSI system with signed revocation token $(\mathsf{rt}, \sigma_{\mathsf{rt}})$

on Network of Nodes in the SSI system

3) parse rt as $(\mathsf{id_{cred}}, \mathsf{DID_P}, \mathsf{DID_{Iss}})$
4) obtain issuer's public key $\mathsf{pk_{Iss}}$ based on $\mathsf{DID_{Iss}}$
5) verify the signature on the revocation token: $\mathsf{SIG.Verify}(\mathsf{rt}, \sigma_{\mathsf{rt}}, \mathsf{pk_{Iss}}) = 1$
6) verify that requester (i.e. her DID) is either the subject or issuer of the credential, i.e. equals the $\mathsf{DID_P}$ or $\mathsf{DID_{Iss}}$
7) interact with other nodes to add rt to the REV list

(4) Attestation of Validity:

(either) **on Prover**

1a) send a request for attestation to a node of the SSI system: $\mathsf{req} \leftarrow (\mathsf{id_{cred}}, \mathsf{DID_P}, \mathsf{DID_{Iss}})$

(or) **on Node of the SSI system**

1b) alternatively, the nodes may periodically trigger re-attestation, based on information by the issuers about issued credentials and their recommended validity-time

on SSI system

2) if the credential appears in the credential revocation list, i.e., $\mathsf{req} \in \mathsf{REV}$: abort
3) otherwise, build a validity attestation statement $\mathsf{stmt} \leftarrow (\mathsf{id_{cred}}, \mathsf{DID_P}, \mathsf{DID_{Iss}}, t)$, where t is the current timestamp
4) interact with n other nodes to create a multi-signature:
   - $\forall i \in [n] : \sigma_{\mathsf{stmt},i} \leftarrow \mathsf{MSIG.Sign}(\mathsf{stmt}, \mathsf{sk}_i)$ and
   - $\sigma_{\mathsf{stmt}} \leftarrow \mathsf{MSIG.ASigs}(\{(\mathsf{pk}_i, \sigma_{\mathsf{stmt},i})\}_{\forall i \in [n]})$
5) issue attestation $\mathsf{attest_{cred}} \leftarrow (\mathsf{stmt}, \sigma_{\mathsf{stmt}})$ to the prover

(5) Showing (Offline):

In the DIDComm protocol [23], [24], the verifier generates a QR code that contains an invitation. Our protocol continues, after the DIDComm connection has been established.

on Prover

1) receives the verifier's request for required and optional attributes $A_{\mathsf{req}} \leftarrow \{1, ..., a_n, a_{n+1}\}$, where $a_{n+1} = \mathsf{id_{cred}}$
2) generates a proof $\pi \leftarrow \Pi.\mathsf{Proof}(x, w)$ over the statement x:
   - PoK over $\{ \mathsf{SIG.Verify}(\mathsf{cred}, \sigma_{\mathsf{cred}}, \mathsf{pk_{Iss}}) = 1 \wedge$
   - $\forall a_i \in A_{\mathsf{req}} : a_i \in A$ where A is contained in $\mathsf{cred} \wedge$
   - presented $\mathsf{DID_P}$ is the subject of the credential $\mathsf{cred} \wedge$
   - presented $\mathsf{DID_{Iss}}$ is the issuer of the credential $\mathsf{cred}\}$

   and with witnesses w: cred, $A \setminus A_{req}$ and σ_{cred}
3) send to verifier: the requested attributes A_{req} including $\mathsf{id_{cred}}$, $\mathsf{DID_P}$, $\mathsf{DID_{Iss}}$, a proof π, the validity attestation $\mathsf{attest_{cred}}$, and $\mathsf{pk_P}, \mathsf{pk_{Iss}}$

on Verifier

4) has verified prover's ownership of $\mathsf{DID_P}$
5) verify that the received public keys $\mathsf{pk_P}, \mathsf{pk_{Iss}}$ match $\mathsf{DID_P}, \mathsf{DID_{Iss}}$
6) verify the proof π over statement x, i.e. $\Pi.\mathsf{Verify}(x, \pi) = 1$, which states that: the original credential contained the received attributes A_{req}, $\mathsf{id_{cred}}$, $\mathsf{DID_P}$, and $\mathsf{DID_{Iss}}$
7) verify the attestation $\mathsf{attest_{cred}} = (\mathsf{stmt}, \sigma_{\mathsf{stmt}})$:
   - $\mathsf{MSIG.Verify}(\mathsf{stmt}, \sigma_{\mathsf{stmt}}, \mathsf{pk}_{nodes}) = 1$, where pk_{nodes} is the list of public keys of the nodes, which are stored in the verifier's trust store
   - $\mathsf{id_{cred}}, \mathsf{DID_P}$ and $\mathsf{DID_{Iss}}$ from the attestation match the received values of the credential
   - the timestamp t in $\mathsf{attest_{cred}}$ is sufficiently recent

while the verifier is offline, fresh information can be obtained by the prover and forwarded to the verifier. As part of the showing phase (5), the prover could obtain a fresh attestation of validity for the credential that will be presented. A fresh attestation reduces the risk that the verifier accepts obsolete information.

Additionally, this concept may also be applied to the verifier's trust store. The makeup of the SSI network may change over time as nodes leave or enter the system, which has to be reflected in the verifier's trust store. The online prover could obtain updates to the verifier's trust store, which are signed by the network and forward them to the verifier. Such an update mechanism would be useful for verifiers that are never or only rarely able to come online.

V. IMPLEMENTATION

The implementation section focuses on a concrete instantiation of our generic architecture. The following subsections detail the different components of our implementation, showing the feasibility of our concept and further also details the used cryptography.

A. Implemented Components

This section describes the proof-of-concept implementation of various aspects to underline the feasibility of our concept.

Communication Channel: As our implementation uses mobile phones as devices of prover and verifier, we had to choose a communication technology for the showing/authentication process, considering support throughout various device

manufacturers and operating systems. Finally, we selected Bluetooth over Wifi direct and NFC because those technologies were not available on all devices and operating systems relevant to our communication. On this communication channel, our implementation uses the DIDComm protocol specification [24]. This protocol starts with displaying a QR code generated by the verifier to the prover. The prover receives an invitation message [23] to establish a DID connection.

Mobile-Phone Based Identity Wallet: An identity wallet, running on the mobile phone, is responsible for creating and managing cryptographic keys, identifiers, and credentials. Our implementation follows the common SSI approach used for communicating with a verifying party. We developed our own agent implementation to interact with the verifier's agent. Additionally, our wallet provides hardware-protected encrypted storage, which has to be unlocked through biometric authentication. In particular, access to key material, which is stored in the secure element of the mobile phone, is protected via fingerprint authentication. Verifiable credentials [21] are stored within the wallet in encrypted form.

Trust Store: The trust store holds a verifiable claim issued and signed by the SSI network. This claim contains a list of all nodes that are currently in the SSI network, including their DIDs and public keys. This list is signed by the SSI network. The trust store enables the verification of multi-signatures issued by the SSI network in an offline setting. In case new nodes become part of the network and other nodes have to leave the network, the trust store requires updates. The verifier might still accept multi-signatures, even if it has to include some additional keys provided by the prover within the verification process. However, a sufficiently large number of keys for that multi-signature has to be trusted via entries in the trust store.

SSI Network: The SSI network used in this work considers semi-trusted organizations as node operators, following a common approach of implementing SSI networks, e.g. Sovrin [11]. Such a network is considered a permissioned ledger system. Our system is built upon DIDs and verifiable credentials (VCs). The user who performs an offline authentication is following the DIDComm protocol [23], [24]. The data exchanged during the authentication was extended according to our needs.

Revocation: Our proposed system maintains a revocation list in the distributed ledger to track which credentials have been revoked, and therefore may no longer be attested for validity. A verifiable credential may be revoked by either the VC issuer or the user itself since only these two parties have the knowledge and authority about the validity of the data. When users obtain a VC, they also receive a revocation token rt, which establishes a link between the credentials identity $\mathsf{id}_{\mathsf{cred}}$ (i.e. a hash over the credential's content) as well as the credential's subject $\mathsf{DID}_{\mathsf{P}}$ and the issuer $\mathsf{DID}_{\mathsf{Iss}}$, without revealing any private information in the revocation process. The issuer also signs this token rt. Users and issuers keep this signed revocation token private until they want to revoke the associated credential. Given a revocation token with a valid signature by the specified issuer, the SSI nodes add it to the revocation list REV at the ledger. Before issuing a validity attestation, the nodes check the credential has not been revoked. That is, this the credential's $(\mathsf{id}_{\mathsf{cred}}, \mathsf{DID}_{\mathsf{P}}, \mathsf{DID}_{\mathsf{Iss}})$ is not part of the revocation list REV.

B. Instantiation of Cryptographic Schemes

This section details the concrete instantiation. For definitions of these cryptographic schemes, we refer to Appendix A.

Multi-Signatures: Our system relies on multi signatures, which represents an extension of traditional digital signature schemes. In this case, signatures on the same message with respect to some public keys, can be aggregated into one compact signature which is valid with respect to an aggregated public key. More formally, we define such signatures following the definition of Drijvers et al. [9]. The BLS signature scheme [6] is a prominent example of a signature scheme that can be extended to a multi-signature [4], [5], [18]. This signature scheme is special in the sense that aggregated signatures and public keys live in the same signature and key space. The verification algorithm for aggregated signatures is also the same as that of non-aggregated signatures.

Proof System: We further used Bulletproofs [7] to implement our proof system. Bulletproof offer properties like it does not rely on a trusted setup in comparison to SNARKs, and still offers efficient proofs with respect to the proof size as well as the proof creation and verification time. Additionally, Bulletproofs offer range proofs and proof of signature's ownership in an efficient way. In summary, we have instantiated SIG as well as MSIG with extended BLS signatures [4], [5], [18], and Π with Bulletproofs [7].

VI. Evaluation

This section demonstrates our approach's practicability by answering the question: *"Can two mid-range devices execute the Showing Phase of the protocol in Table I within 1 second[1]?"*. We conclude this section by discussing security-related aspects.

A. Practicability

In the practicability evaluation, we perform two empirical experiments and discuss their results. First, we analyze the time to compute the proof generation $\Pi.\mathsf{Proof}$ and the proof verification $\Pi.\mathsf{Verify}$ on a mid-range smartphone. Both steps belong to the Showing Phase and, because they run a zero-knowledge protocol, we expect those steps to consume the lions share of the computational effort. Then, we investigate how much time the parameter exchange from the Showing Phase takes. We give an upper bound for the parameters' size and send that amount over a Bluetooth channel between two mobile devices. We implement $\Pi.\mathsf{Proof}$ and $\Pi.\mathsf{Verify}$ using the Hyperledger Ursa Library [10], a Bulletproof implementation written in Rust[2]. We execute $\Pi.\mathsf{Proof}$ and $\Pi.\mathsf{Verify}$ on a mid-range smartphone from 2016 (Google Pixel 1) with a quad-core CPU (2x2.15 GHz / 2x1.6 GHz) which runs on Android 9. We chose this device since it has a sufficiently low computing power to represent a large user base. Table II shows the time

[1]According to [15], 1 second is the limit of the user's flow of thought to stay uninterrupted.

[2]https://www.rust-lang.org/

needed for running Π.Proof and Π.Verify with differently sized credentials. In the first row, the credential contains one attribute that is shown by the prover. In the other rows, the credential contains four attributes (name, nationality, date of birth, and sex), and the prover shows a subset of these attributes.

TABLE II
AVERAGE AND STANDARD DEVIATION OF TIME FOR CREATING AND VERIFYING A PROOF PERFORMED ON A PIXEL 1, AND SIZE OF THE PROOF ENCODED IN JSON.

Attributes Shown / Available	Π.Proof (ms)	Π.Verify (ms)	Proof Size (bytes)
1 / 1	41 (±1)	41 (±1)	3305
1 / 4	53 (±1)	53 (±2)	3883
2 / 4	49 (±1)	54 (±2)	3801
3 / 4	45 (±1)	49 (±2)	3678
4 / 4	41 (±1)	46 (±2)	3599

From Table II we can see that, in all cases, both Π.Proof and Π.Verify take less than 56 ms and that the resulting proof is less than 4 kB. We also see that the time to create proofs, the time to verify proofs, and the proof size decrease if the prover shows more attributes (i.e., hides less attributes).

To conduct the second experiment, we estimate the size of the exchanged data in the Showing Phase. For a typical credential (e.g., the four-attributes credential in Table II), we argue that the size of the exchanged data (A_{req}, proof π, the validity attestation $\mathsf{attest}_{\mathsf{cred}}$, and pk_{P}, $\mathsf{pk}_{\mathsf{Iss}}$) averages around 6 kB. The attestation is a signed JSON containing the $\mathsf{DID}_{\mathsf{Iss}}$, $\mathsf{DID}_{\mathsf{P}}$, the $\mathsf{id}_{\mathsf{cred}}$ and a timestamp, which is small in size. We round up this number to 10 kB to be on the safe side. We also run the experiment with larger credentials, where the exchanged data amounts to 100 kB (e.g., a credential with many attributes) or even 1000 kB (e.g., an image as an attribute).

In the Bluetooth experiment, the prover device (Samsung Galaxy XCover Pro) sends the data to the verifier device (Google Pixel 1) using Bluetooth 4.2 while being placed 1.5 meters apart. We repeat this experiment a 100 times. Figure 3 shows the distribution of the time between sending the data and receiving an acknowledgment on the prover device.

In 90% of the cases, exchanging 10 kB of data takes less than 125 ms, exchanging 100 kB takes less than 884 ms, and 1000 kB takes less than 5.5 seconds. We conclude that two mid-range devices exchange the data for both typical and large credentials within a second in most cases, but we note that occasional outliers push this limit up to two seconds. For the image credential, the time for exchanging 1000 kB becomes more stable and ranges between 4 and 6 seconds. We can deduce from both experiments that, in 90% of cases, the most time-consuming steps of the Showing Phase can be executed below 237 ms (125 + 112 ms) with a typical credential and below 996 ms (884 + 112 ms) with a large credential.

B. Security Analysis

This section provides an informal security analysis due to space limitations. We consider the following properties of our concept: The prover should not be able to present a VC nor a related validity attestation if they do not belong to the prover or have been revoked.

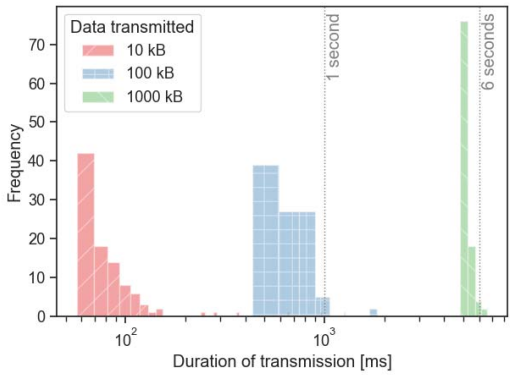

Fig. 3. Timing distribution of the Parameter Exchange from the Showing Phase between two Mobile Devices over Bluetooth

Credential Verification: VCs are linked to a person through the cryptographic binding of a DID to a public-private key pair. By proving the ownership of the DID, the verifier is convinced that the prover is in possession of the related private key. The verifier further validates the proof of the attributes ensuring that the attributes belong to the owner of the DID and were issued from the respective VC issuer.

Validity Attestation Verification: Furthermore, the verifier assesses the validity of the credential through the attached assertion. The attestation proves the time when the VC was still valid. The verifier, of course, has to decide if the presented point in time is sufficiently recent for its use case. The attestation is signed by the network of SSI nodes via a multi-signature. Such signatures are verified utilizing the public keys stored in the trust store, signed by the SSI network to ensure its integrity. An attacker that wants to add a public key to this trust store would have to have direct access to the verifier's device, which is a very strong assumption, as this attacker would have full control over the victim's device. Additionally, the trust store's integrity and authenticity is protected through the multi-signature.

Decentralized Revocation Ledger: By utilizing a DL for storing revocation information, issues such as the single point of failure and the central trusted party are directly addressed. Furthermore, by utilizing a trusted SSI network for the revocation check, each node can trigger the revocation check as well as the trigger the attestation process. Trust and power is distributed across the SSI network resulting in a DPKI serving as root of trust.

VII. CONCLUSION

This work proposes a general concept enabling offline authentication of revocable SSIs serving as a basis for future real-world implementations, e.g., in eGovernment. Especially because fully offline authentication is required to achieve the digital counterpart of a physical identity card and was not achieved yet reflects the importance of this work. Offline

verification is achieved by utilizing an attestation based system together with a trust store implementation. As part of our concept, we also provided a detailed protocol.

To show the feasibility and practicability of our system, we implemented a proof-of-concept. We provided an evaluation of this implementation, including benchmarks. The results show that the presented concept is feasible even if both parties are offline.

REFERENCES

[1] A. Abraham, F. Hörandner, O. Omolola, and S. Ramacher, "Privacy-preserving eid derivation for self-sovereign identity systems," in *Information and Communications Security ICICS 2019, Beijing, China*, ser. LNCS, vol. 11999. Springer, 2019, pp. 307–323.

[2] C. Allen, "The Path to Self-Sovereign-Identity," April 2016, [Online; accessed 15-02-2019]. [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[3] J. B. Bernabé, A. F. Skarmeta, N. Notario, J. Bringer, and M. David, "Towards a privacy-preserving reliable european identity ecosystem," in *Privacy Technologies and Policy - 5th Annual Privacy Forum, APF 2017, Vienna, Austria, June 7-8, 2017, Revised Selected Papers*, ser. LNCS, vol. 10518. Springer, 2017, pp. 19–33.

[4] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography*, ser. LNCS, vol. 2567. Springer, 2003, pp. 31–46.

[5] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *ASIACRYPT (2)*, ser. LNCS, vol. 11273. Springer, 2018, pp. 435–464.

[6] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *ASIACRYPT*, ser. LNCS, vol. 2248. Springer, 2001, pp. 514–532.

[7] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 315–334.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. T. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," *RFC*, vol. 5280, pp. 1–151, 2008.

[9] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 514, 2019.

[10] L. Foundation. (2020) Hyperledger ursa. [Online] Accessed 06 March 2020. [Online]. Available: https://www.hyperledger.org/use/ursa

[11] S. Foundation. (2020) Sovrin stewards. [Online] Accessed 01 July 2020. [Online]. Available: https://sovrin.org/stewards/

[12] D. E. E. III, "Transport layer security (TLS) extensions: Extension definitions," *RFC*, vol. 6066, pp. 1–25, 2011.

[13] S. Mödersheim, A. Schlichtkrull, G. Wagner, S. More, and L. Alber, "TPL: A Trust Policy Language," ser. IFIP Advances in Information and Communication Technology, vol. 563. Springer, 2019, pp. 209–223.

[14] A. Mhle, A. Grner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80 – 86, 2018.

[15] J. Nielsen, "Usability engineering," in *The Computer Science and Engineering Handbook*. CRC Press, 1997, pp. 1440–1460.

[16] OASIS. (2008) Security Assertion Markup Language (SAML) V2.0 Technical Overview. [Online] Accessed 16 July 2020. [Online]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[17] OpenID. (2020) OpenID Connect. [Online] Accessed 16 July 2020. [Online]. Available: https://openid.net/connect/

[18] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *EUROCRYPT*, ser. LNCS, vol. 4515. Springer, 2007, pp. 228–245.

[19] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, pp. 1–5.

[20] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - OCSP," *RFC*, vol. 6960, pp. 1–41, 2013.

[21] M. Sporny, D. Longley, and D. Chadwick. (2019) Verifiable Credentials Data Model 1.0 - Expressing verifiable information on the Web. [Online] Accessed 08 July 2020. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[22] W3C Working Draft, "Decentralized Identifiers (DIDs) v1.0 - Core Data Model and Syntaxes," 2019, online, Accessed: 2020-01-27. [Online]. Available: https://www.w3.org/TR/did-core/

[23] R. West, D. Bluhm, M. Hailstone, S. Curran, and S. Curren. (2020) Aries RFC 0023: DID Exchange Protocol 1.0. [Online] Accessed 30 June 2020. [Online]. Available: https://github.com/hyperledger/aries-rfcs/blob/master/features/0023-did-exchange/README.md

[24] R. West, D. Bluhm, M. Hailstone, S. Curran, S. Curren, and G. Aristy. (2020) Aries rfc 0434: Out-of-band protocols. [Online] Accessed 02 July 2020. [Online]. Available: https://github.com/hyperledger/aries-rfcs/blob/master/features/0434-outofband/README.md

[25] B. Zwattendorfer, T. Zefferer, and K. Stranacher, "An overview of cloud identity management-models," in *WEBIST (1)*. SciTePress, 2014, pp. 82–92.

APPENDIX A
CRYPTORAPHIC BUILDING BLOCKS

We are interested in an extension of signature schemes to multi-signature schemes. 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. We define such signatures following the definition of Drijvers et. al [9]:

Definition 1 (Multi-Signature Scheme): A multi-signature scheme MSIG is a signature scheme SIG with an additional triple $(\mathsf{APKs}, \mathsf{ASigs}, \mathsf{AVerify})$ of PPT algorithms, which are defined as follows:

$\mathsf{APKs}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$: This algorithm takes n public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ as input and outputs an aggregated public key pk_M.

$\mathsf{ASigs}((\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_n, \sigma_n), m)$: This algorithm takes n signatures $\sigma_1, \ldots, \sigma_n$ on the same message m and the corresponding public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, and outputs an aggregated signature σ_M on the message m or $\perp$ on error.

$\mathsf{AVerify}(\mathsf{pk}_M, m, \sigma_M)$: This algorithm takes an aggregated public key pk_M, a message $m \in \mathcal{M}$ and an aggregated signature σ_M as input and outputs a bit $b \in \{0, 1\}$ indicating the validity of the signature.

Also, we recall a standard definition of NIZK proof systems. Let $L \subseteq \mathsf{X}$ be an $\mathbf{NP}$-language with associated witness relation R so that $L = \{x \mid \exists w \colon R(x, w) = 1\}$.

Definition 2 (NIZK): A non-interactive proof system Π is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Proof}, \mathsf{Verify})$, defined as follows:

$\mathsf{Setup}(1^\kappa)$: This algorithm takes a security parameter κ as input, and outputs a common reference string crs.

$\mathsf{Proof}(\mathsf{crs}, x, w)$: This algorithm takes a common reference string crs, a statement x, and a witness w as input, and outputs a proof π.

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$: This algorithm takes a common reference string crs, a statement x, and a proof π as input, and outputs a bit $b \in \{0, 1\}$.

Our proof system requires 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 (proof reveals no information on the witness).