SISSI: An Architecture for Semantic Interoperable Self-Sovereign Identity-based Access Control on the Web

Christoph H.-J. Braun
braun@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Vasil Papanchev
vasilpapanchev@gmail.com
Karlsruhe Institute of Technology
Karlsruhe, Germany

Tobias Käfer
tobias.kaefer@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

We present an architecture for authentication and authorization on the Web that is based on the Self-Sovereign Identity paradigm. Using our architecture, we aim to achieve semantic interoperability across different approaches to SSI. We build on the underlying RDF data model of the W3C's recommendation for Verifiable Credentials and specify semantic access control rules using SHACL. Our communication protocol for an authorization process is based on Decentralised Identifiers and extends the Hyperledger Aries Present Proof protocol. We propose a modular architecture that allows for flexible extension, e. g., for supporting more signature schemes or Decentralised Identifier Methods. For evaluation, we implemented a Proof-of-Concept: We show that a Web-based approach to SSI outperfoms a blockchain-based approach to SSI in terms of End-to-End execution time.

CCS CONCEPTS

• **Information systems** → **World Wide Web**; • **Security and privacy** → **Access control**; *Privacy protections*; • **Computer systems organization** → *Distributed architectures.*

KEYWORDS

Web, Solid, Access Control, Linked Data, Self-Sovereign Identity

ACM Reference Format:
Christoph H.-J. Braun, Vasil Papanchev, and Tobias Käfer. 2023. SISSI: An Architecture for Semantic Interoperable Self-Sovereign Identity-based Access Control on the Web. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583409

1 INTRODUCTION

The recent *Self-Sovereign Identity* (SSI) paradigm, described in the seminal [1], aims to put users in center and control of their digital identity. SSI sees increasing attention [16] and adoption across the globe [8, 19, 33, 38] and across domains, e. g., the financial sector [20], public sector [24, 38] or even healthcare [29]. A diverse ecosystem of technologies and SSI providers[1] has been established where now interoperability [52] and seamless support across technologies and providers becomes a pressing question.

To address this problem, Web services may be tempted to integrate the various SSI providers in a wrapper-based approach, as showcased in [27]. In such scenarios the overall authentication and authorization flow remains unchanged to today's typical e. g. OIDC-based process [43]. Only the Identity Provider (IDP) is retrofitted to authenticate SSIs and then to provide verified attributes to the authorization server (AS), as illustrated in Figure 1a.

With such retrofitting approach however, adapting to changes from SSI providers remains responsibility and overhead of the Web service. Moreover, during interaction between client and AS, the IDP still acts as a trusted middleman that is not needed with SSI.

To help overcome such issues, and to showcase a truly SSI-based access control system, we propose an approach as illustrated by Figure 1b: Instead of retrofitting an IDP, we omit the IDP completely. As identities are truly self-sovereign, clients themselves should authenticate and prove claims to the AS. No IDP needs to broker between client and AS. Moreover, clients should have free choice of SSI, i. e., not being locked in to a specific SSI technology or SSI provider. To achieve these design goals, we build on a stack of Web technologies: We rely (a) on a uniform data model, the Resource Description Framework (RDF) [18], (b) on semantic interoperability by means of URIs and ontologies, and (c) communication protocols from the SSI community.

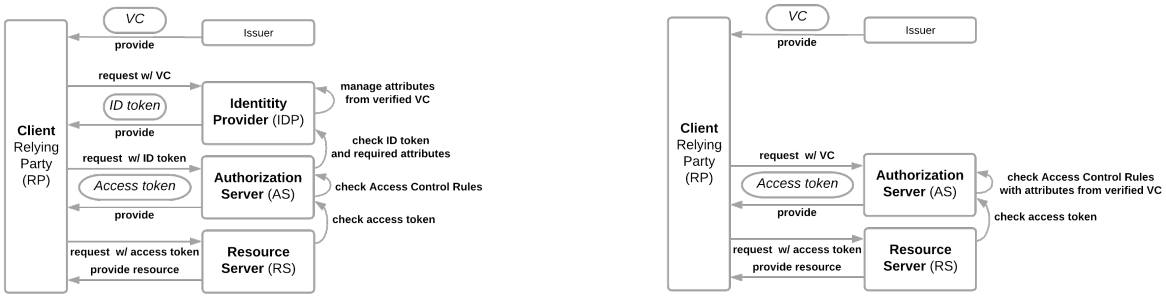
We thus present an architecture for *Semantic Interoperable Self-Sovereign Identity-based (SISSI)* access control, i. e., authentication and authorization, on the Web. The architecture is based on the Self-Sovereign Identity (SSI) paradigm [1] and achieves semantic interoperability of corresponding concepts using existing Web Standards: We build on the semantic data model underlying the W3C's recommendations for Decentralised Identifiers (DIDs) [47] and Verifiable Credential (VC) data model [48]. As their data model adheres to the Resource Description Framework (RDF) [18], we seamlessly model Access Control Rules (ACRs) using the Web Access Control (WAC) ontology [14] which is well-established e. g. in the Solid community [15]. Moreover, we extend WAC to express requirements of particular VCs using the W3C recommendation of the Shapes Constraint Language (SHACL) [32]. Using SHACL additionally allows us to re-use the ACRs for formally validating that a presented VC satisfies the requirement without leaving our semantic model. The SISSI architecture also builds on well-established protocols within the SSI community, specifically for communication on the HL Aries Present-Proof Protocol [31] on top of the DIDComm Messaging Protocol [17] in combination with PeerDIDs [28].

To prove the viability of the SISSI architecture, we implement a simple Proof-of-Concept Access Control System (ACS) which is

[1] A service or organisation providing governance of identifiers and associated claims, e. g., Sovrin (https://sovrin.org) or Jolocom (https://jolocom.io/).

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583409



(a) IDP-based approach where the IDP is retrofitted to support SSI.

(b) Truly SSI-based approach where no IDP is required.

Figure 1: High-level overview of differences in SSI-supporting approaches to authentication and authorization.

comprised by an AS implementing the SISSI architecture. From Figure 1b, our work focuses on the AS and its interactions with a client. Therefore, additionally a user agent acts as a client executing an authorization process at the AS. We evaluate the End-to-End execution time of an authorization process to identify bottlenecks and optimisation potential. Moreover, as SSI solutions are typically blockchain-based, we compare the impact of different DID Methods to illustrate the difference between Web-based and blockchain-based DIDs on system performance, as SISSI is infrastructure-agnostic.

Our paper is structured as follows: We introduce technical foundations (Section 3) and survey related work (Section 4). The main contribution of our work is an interoperable and truly SSI-based approach to access control on the Web, see Figure 1b, based on:

- a formalisation of the SISSI architecture (Section 6) with
- a uniform semantic data model (Section 5) and
- a performance evaluation of a Proof-of-Concept implementation (Section 8).

In Section 7, we cover the corresponding communication protocol. In Section 9, we summarise the design choices in our architecture.

2 ILLUSTRATING EXAMPLE

Consider a Web server for cross-university course material sharing where users must be authenticated as students of officially recognised universities to gain access to the course material. The universities follow an SSI-based approach to manage rights and thus issue digital credentials to their students, rendering a central or federated IDP unnecessary, see Figure 1b. In this SSI approach, students would be able to authenticate directly to the Web server.

Yet, universities may rely on different SSI providers for managing student credentials. These SSI providers choose how their credentials are designed, i. e., the employed DID Method (Sec. 3.2) and VC flavour (Sec. 4.4). Alice's university relies on SSI provider M: Her credential is based on Web-based DIDs and the VC flavour of JSON-LD with BBS+ signatures. Bob's university uses provider S: His credential is based on Indy-based DIDs and JSON Web Tokens and Signatures (JWT/JWS). Charlie's university uses provider U: Her credential is based on Web-based DIDs and JWT/JWS. This heterogeneity introduces interoperability challenges.

In a wrapper-based approach to address the interoperability challenge, one wrapper is required for each SSI provider: Each wrapper comes with its own pair of drivers, one for the DID and one for the VC flavour; in our example, three wrappers and thus at least six drivers are deployed in total.

With SISSI, the components to implement are pushed down to the basis, i. e. only four drivers, two for the DIDs (Web and Indy) and two for the VC flavours (JSON-LD with BBS+ and JWT/JWS). Moreover, when yet another SSI provider is used and the required drivers are already deployed in the SISSI-based system (e. g. Indy-based DID and JSON-LD with BBS+), no additional overhead occurs as this provider is already implicitly supported.

On top, SISSI capitalises on the RDF data model that underlies VCs, as a basis for both credentials and access control. We thus can employ the existing SHACL W3C Recommendation for RDF data shapes to define access control rules, instead of a custom combination of the unfinished JSON Schema and JSONPath, as [10].

3 PRELIMINARIES

3.1 Linked Data

3.1.1 URIs, HTTP and RDF. The Linked Data principles [4] are a set of practices for data publishing on the Web: Uniform Resource Identifiers (URIs) [5] act as names for things. The Hypertext Transfer Protocol (HTTP) [22] is used to dereference URIs. When dereferencing a URI, the retrieved content is expressed as a graph using the Resource Description Framework (RDF) [18].

An RDF graph $G \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of all RDF graphs, is defined as a set of triples. RDF graphs can be serialised, e. g. in Turtle (our examples) or JSON-LD. With $\mathcal{U}$ as the set of all URIs, $\mathcal{B}$ as the set of all blank nodes, and $\mathcal{L}$ as the set of all literals, a triple t can be defined as $t \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. An RDF dataset $D \in \mathcal{D}$, where $\mathcal{D}$ denotes the set of all RDF datasets, is a set of named graphs, i. e., couples (n, G_n) where $n \in (\mathcal{U} \cup \mathcal{B})$ and $G_n \in \mathcal{G}$, and an unnamed *default graph* $(\_, G)$ with $G \in \mathcal{G}$.

3.1.2 Shapes Constraint Language (SHACL) [32]. allows to describe conditions in an RDF graph (*shape graph*), using which other RDF graphs (*data graphs*) can be validated. With SHACL, we can express constraints, e. g. that a VC's issuer must be from a given set, the VC must not be expired, or the VC must assert a specific attribute.

3.2 Self-Sovereign Identity (SSI)

The SSI vision [1] postulates that agents should in control of their digital identity. In SSI, agents can assume three roles: Issuer, who

(cryptographically) attests some claims, e. g., about a holder. Holder, who manages those attested claims. Verifier, who (cryptographically) verifies presented claims[2], e. g., for authentication and authorization of the holder. Decentralised Identifiers (DIDs) [47] and Verifiable Credentials (VCs) [48] are technical foundations for SSI.

3.2.1 Decentralised Identifiers (DIDs) [47]. are a W3C Recommendation to allow an agent to prove control over an identifier (DID) without involving a third party, and thus to enable decentralized verifiable digital identity and interactions. A DID is a URI that links a *DID subject*, e. g., an agent, with a *DID Document*. The DID Document provides information on cryptographic keys and verification methods using which the control over the DID can be proven.

More formally, let $\mathcal{U}_{DID} \subset \mathcal{U}$ denote the set of all DIDs. A DID Document D_{DID} may be serialised in JSON-LD, thus $D_{DID} \in \mathcal{D}$. A *DID Method* defines how to retrieve a DID Document from a DID[3]. Let $\mathcal{M}$ denote the set of all DID Methods. We define resolving a DID as $\delta_m : U_{DID} \mapsto \mathcal{D}$ with $m \in \mathcal{M}$.

PeerDIDs [28] are one such DID Method; *did:peer* $\in \mathcal{M}$. PeerDIDs encode the entire DID Document in the DID itself and are therefore static. As PeerDIDs are thus independent from any registry (e. g. a blockchain), PeerDIDs are fast, cheap and scalable DIDs, well-suited for private peer-to-peer interactions.

DIDComm Messaging [17] is a message-based, connectionless and asynchronous protocol used for encrypting and signing messages. Using the cryptographic keys from the DID Documents of the communicating parties, *DIDComm messages* can be exchanged securely between DID subjects, e. g., identified by PeerDIDs.

3.2.2 The Verifiable Credentials (VCs) [48] Data Model. is a W3C Recommendation for modelling verifiable claims. The recommendation builds on JSON-LD. As per the VC JSON-LD context, a VC c is an RDF dataset, thus $c \in \mathcal{D}$. VCs can also be serialised as JSON Web Token (JWT) [30]; conversely, JWT credentials can be converted to JSON-LD [48]. Let $\mathcal{D}_{VC} \subset \mathcal{D}$ denote the set of all VCs.

A VC is composed of three components: the claim, the credential (metadata) and the proof. Claims and credential form the *credential graph* $G_c \in \mathcal{G}$. The credential graph links to a *proof graph* $G_s \in \mathcal{G}$ with the credential's digital signature. We thus define a VC $c = \{(\_, G_c), (n_s, G_s)\}$, with the credential graph as the default graph.

The claims part of the credential graph contains the triples for which the VC has been created. The credential metadata contains information on the credential itself, e. g. the credential's issuing date. The proof graph consists of all information required to establish a credential's validity such as signature scheme and cryptographic key. Various signature schemes exist, see Section 4.4. Let $\mathcal{S}$ denote the set of all such signature schemes.

To present a VC to another agent, the VC Recommendation introduces the *Verifiable Presentation (VP)* [48]. On top of the information from a VC, a VP may include information such as a license, or a cryptographic challenge to prevent replay attacks. Such information is described in a *presentation graph* $G_p \in \mathcal{G}$ that links to credential graphs G_c and to a *presentation proof graph* $G_{ps} \in \mathcal{G}$ with its digital signature. We define a VP $c_p = \{(\_, G_p), (n_c, G_c), (n_s, G_s), (n_{ps}, G_{ps})\}$. Let $\mathcal{D}_{VP} \subset \mathcal{D}$ denote the set of all VPs.

3.2.3 The Hyperledger (HL) Aries Present-Proof Protocol [31]. is a protocol for VP exchange on top of the DIDComm Messaging protocol [17]. DIDComm defines two core message types: A *request-presentation* message that contains a Verifiable Presentation Request (VPR) and is sent to request VCs, and a *presentation* message that contains a VP and is sent in response. The protocol is agnostic to the specific formats of VPRs, VPs, and VCs.

4 RELATED RESEARCH

SISSI enables semantically described SSI-based access control on the Web. We thus survey related work in the intersections of access control, SSI and Semantic Web, next to challenges in SSI in general.

4.1 Access Control on the Semantic Web

Access control on the Semantic Web is mainly developed within the Solid project [44]. The Solid Protocol [15] (currently) uses Access Control Lists (ACLs) [14] in RDF to specify access control to Web resources, e. g. to grant access to specific agents or agent groups. Instead of DIDs, Solid uses WebIDs [45], i. e. HTTP URIs that can be dereferenced to yield profile documents, to identify agents.

Recently, the Solid community started to specify Access Control Policies (ACPs) [7], as an alternative to ACLs. The current specification draft mentions VCs in the context of access requests. The draft also specifies a matcher pattern based on a newly developed ontology, instead of using a standard such as SHACL, as we do.

Werbrouck et al. proposed a pattern-based access control framework for Solid [51], also based on SHACL. However, they rely on nanopublications [34] (1) as data model for modelling verifiable claims, which, in contrast to VCs, e. g. does not define how to identify the issuing agent, and (2) as verification protocol, which builds on necessarily public information and is thus not self-sovereign.

4.2 SSI and the Semantic Web

The few works in Semantic Web that address the ideas behind SSI often do not use the term SSI, as SSI is typically associated with blockchain technology [39]: Solid [15, 44] implements the ideas of data sovereignty and privacy using Web technologies: Instead of a blockchain that stores a DID document about an agent identified by that DID, a Web server under the agent's control provides a profile document about the agent identified by a WebID [45].

Works that combine Solid and VCs include: Ezike's approach to issue, handle and revoke VCs [21], such that access to Web resources can be granted using VCs, and Braun and Käfer's approach for selective disclosure of attributes in VCs [9], such that access to Web resources can be granted in a privacy-preserving manner. In contrast, SISSI considers the modelling of access control rules, and the interoperation of different DID Methods and VC flavours.

4.3 SSI-based Access Control

While there is an abundance of blockchain-based access control systems for various domains and use-cases [25, 41, 42], and while there is a thriving ecosystem around SSI technologies [46, 54], only few works exist in the intersection of SSI and access control:

[2]While trusting the issuer about the truthfulness of the attested claims.
[3]DID Methods can be defined and registered by anyone, see [50].

SSIBAC [3] is an Access Control System based on SSI, implemented using the SSI framework Hyperledger Indy[4]. SSIBAC does not specify how to define access control policies. In contrast, SISSI offers a driver-based architecture to interoperate across different SSI solutions, and defines access control rules using Web standards.

Works also exist about retrofitting SSI into existing solutions and approaches: Kuperberg et al. survey solutions that integrate SSI into Identity and Access Management [35] and find that solutions are very heterogeneous and the field lacks standardisation. Grüner et al. present a solution that retrofits SSI into an IDP, specifically they integrate the SSI providers uPort and Jolocom into an identity management system that is based on OpenID Connect (OIDC) [43] and the Security Assertion Markup Language (SAML) [40]. To achieve interoperability, they use one wrapper per API of the corresponding SSI solutions' SDKs, see Figure 1a. In contrast, SISSI achieves interoperability by semantic modelling of the VCs, the use of correspondingly modelled authorization rules, and the direct use of the SSI protocols that underly the SSI solutions.

Regarding the interoperability of SSI systems, Grüner et al. conceptually analyse different approaches [26], and show that there is no interoperability concepts superior to all others. In their taxonomy of SSI interoperability concepts, the server-side *identity broker* concept is the closest match to SISSI, but with the important difference that in SISSI, the hosting entity does not need to register an identity at each SSI provider. In their interoperability level scheme, SISSI falls into the highest level by supporting authorisation.

4.4 General Challenges in SSI

In expert interviews, Laatikainen *et al.* [36] identify technical challenges for the adoption of SSI. We explain those for the Web context:

Vertical Interoperability: To prevent tight vertical coupling and dependencies between different technical components within an SSI-based information system, components should strive to be agnostic from the details of other technologies, e. g. a messaging protocol for VC exchange should be independent of VC formats.

Horizontal Interoperability: The W3C standards for DIDs [47] and VCs [48] contain many degrees of freedom to allow for use-case specific optimisations. To date, there are 136 officially registered DID Methods, and we distinguish 4 flavours of VCs, cf. [53]: JSON-LD VCs with LD-Signatures [49], JSON-LD VCs with BBS+ Signatures [37], credentials expressed as JWTs [30], and Anonymous Credentials [11] with Camenisch-Lysyanskaya [12, 13] or BBS+ signatures [2, 6]. Moreover, different VC implementations differ in how DIDs are linked or how status verification works. Laatikainen et al. note that the lack of interoperability between different SSI solutions significantly impedes the adoption of SSI.

5 DATA MODELING

To form the foundation for tackeling the outlined challenges, we rely on RDF as a uniform data model and achieve semantic interoperability by re-using URIs and ontologies across VCs and ACRs. In this section, we briefly cover a small addition to the HL Aries Present-Proof [31] message types, the proposed semantic VPR and ACR format modeling the required credentials using SHACL [32].

[4]https://www.hyperledger.org/use/hyperledger-indy

5.1 HL Aries Present-Proof DIDComm Messages

We re-use the DIDComm message types for requesting and presenting VCs as defined by the HL Aries Present-Proof Protocol [31]. Additionally, we define a message type, *access-request*, to indicate the desire to access a particular resource and, if applicable, to receive a corresponding VPR, i. e., *request-presentation* message. Moreover, we "semantify" all message types by providing JSON-LD contexts[5] to lift the messages' content to RDF, i. e., any message $d_{msg} \in \mathcal{D}$.

We define a semantic VPR format, the *SHACL-based Presentation Request Format* based on SHACL shape graphs describing the requested VCs. More formally, we define a VPR c_r as an RDF dataset; $c_r \in \mathcal{D}$. Let $\mathcal{D}_{VPR} \subset \mathcal{D}$ denote the set of all such VPRs. The SHACL shape graphs are specified in the ACRs that control access to the desired resource. Upon *access-request*, the graphs are used in the VPR *request-presentation* message, see Appendix A for an example.

5.2 Access Control Rules (ACRs)

We model ACRs based on the Solid's Web Access Control (WAC) [14]. We extend the WAC ontology to link a standard WAC ACR to a SHACL shape graph describing required VCs. The example ACR, illustrated by Listing 1, specifies that a student VC issued by one of the specified agents, e. g., universities, needs to be presented. Combinations of multiple VCs can either be specified by an RDF object list or by using the native logic operators of SHACL.

```
@prefix acl: <http://www.w3.org/ns/auth/acl#> .
@prefix cred: <https://www.w3.org/2018/credentials#>.
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix sissi: <https://purl.org/sissi/messages/ns#>.

_:ACR_student
  a                 acl:Authorization ;
  acl:accessTo      <http://example.org/r1> ;
  acl:agent         acl:AuthenticatedAgent ;
  acl:mode          acl:Read ;
  sissi:requiredCredential _:studentIssuerShape .

_:studentIssuerShape
  a                 sh:NodeShape ;
  sh:targetClass cred:VerifiableCredential ;
  sh:class          <http://example.org/edu#Student> ;
  sh:property
    [  sh:path    cred:issuer ;
       sh:in    ( <did:ethr:0x3:0x032..5d7> <did:indy:local:YY8..6tZ> ) ] .
```

Listing 1: An example ACR using WAC+SHACL

6 SYSTEM ARCHITECTURE

Our proposed SISSI architecture, depicted in Figure 2, is comprised of the following components:

For **Communication**,

- the *Message Security (MS)* protects external communication according to the DIDComm Messaging Protocol [17].
- the *Message Model (MM)* generates messages according to our extended data model from Section 5.1.

[5]https://purl.org/sissi/messages/v1

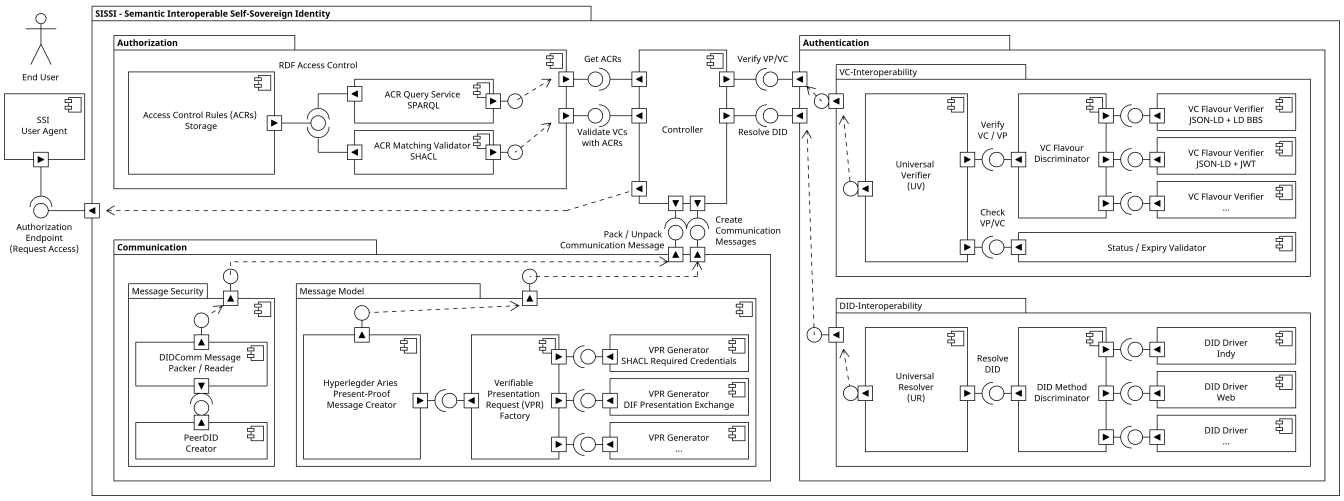


Figure 2: The SISSI architecture of an authorization server (AS) from Figure 1, plus a user and their SSI agent as a client.

For **Authorization**,

the *RDF Access Control (RAC)* manages the ACRs as modelled in Section 5.2. Moreover, it validates VCs against those ACRs to derive an authorization decision.

For **Authentication**,

the *Universal DID Resolver (UR)* enables DID interoperability, i. e., resolving DIDs of different DID Methods to their DID Documents.

the *Universal VC Verifier (UV)* enables VC Interoperability, i. e., verifying different VC flavours.

The *Controller (C)* that orchestrates the internal services and exposes an external API to execute an authorization process.

More formally, we define the SISSI architecture as a tuple of its components, which we introduce in detail in the upcoming sections:

$$SISSI = (MS, MM, UR, UV, RAC)$$

As the *Controller (C)* only acts as architectural glue and proxy to expose the internal service orchestration for executing an authorization process (Section 7), it may be omitted. For completeness and with regards to our evaluation, we additionally introduce an *SSI User Agent (A)* along-side our SISSI architecture in Section 6.6. The agent A interacts as the client from Figure 1 with our SISSI AS as the AS. Thus, the agent A takes also active part in our evaluation as the subject that is exposed to the evaluated End-to-End latency of our system (Section 8.2).

6.1 Message Security (MS)

The *Message Security (MS)* component is responsible for securing the communication between the SISSI AS and the user agent. To this end, we propose relying on the DIDComm Messaging Protocol [17] in combination with static PeerDIDs [28] for identification. We specify our *MS* component as a set of functions:

$$MS = (\{pack, unpack, send, gen_{peerDID}\})$$

Upon receiving a DIDComm Message, $unpack : \mathbb{R} \mapsto \mathcal{D}$ is applied to decrypt and authenticate the message, and to retrieve its content, i. e., an RDF dataset. Such message content is transmitted to the *Controller* for further processing. Once the *Controller* supplies the *MS* with a response, i. e. an RDF dataset, $pack : \mathcal{D} \mapsto \mathbb{R}$ is applied to sign and encrypt a DIDComm Message for the supplied RDF dataset. The DIDComm message is then *send* to the user agent as an HTTP response. To identify the server during communication, the *MS* generates a single PeerDID, henceforth called *ServerDID*, using $gen_{peerDID} : \mathbb{R} \mapsto \mathcal{U}_{PeerDID}$.

The ServerDID contains the DIDComm Messaging service endpoint, i. e. the AS's inbox, and the public key to be used by all user agents to encrypt their DIDComm Messages. How exactly the ServerDID is disseminated on the Web depends on the use-cases of the server and is considered out-of-scope for our work. For example, a resource server might provide a QR code containing the ServerDID or an HTTP URI to an RDF dataset describing the AS.

6.2 Message Model (MM)

The *Message Model (MM)* component provides the messages for requesting and receiving VCs. We propose relying on our extended protocol model from Section 5.1. We thus formally specify our *MM* component as a set of functions:

$$MM = (\{xtrct_{ar}, gen_{vpr}, xtrct_{vp}, gen_{resp}\})$$

with $xtrct_{ar} : \mathcal{D} \mapsto \mathcal{U}$, to extract a requested URI from an access request message, and $gen_{vpr} : R^n \mapsto \mathcal{D}$, to generate a message containing a VPR based on a set of ACRs, and $xtrct_{vp} : \mathcal{D} \mapsto \mathcal{D}_{VP}$, to extract a VP from a message containing a VP, and $gen_{resp} : \{0, 1\} \mapsto \mathcal{D}$ to generate a response message based on an access control decision.

6.3 Universal DID Resolver (UR)

DID interoperability refers to seamless support for DIDs across different DID Methods. An open-source project by the Decentralized Identity Foundation (DIF) already implements the desired functionality: the DIF Universal Resolver[6]. It implements a driver-based

[6]https://github.com/decentralized-identity/universal-resolver

architecture for resolving DIDs of different DID Methods. We thus formally specify a *Universal DID Resolver (UR)* in our architecture:

$$UR = (M, \mu, \Delta)$$

with the set of supported DID Methods $M \subseteq \mathcal{M}$, and a function $\mu : U_{DID} \mapsto M$ that determines a DID's DID Method $m \in M$, and the set of implemented driver functions $\Delta = \{\delta_m | m \in M\}$ whose elements are functions $\delta_m : U_{DID} \mapsto \mathcal{D}$ with $m \in M$ that resolve a DID to its DID Document. Slightly abusing notation, we denote the partial composition of μ and δ by $\delta|_{m=\mu} : U_{DID} \mapsto (D)$. A driver δ_m for a particular DID Method m is contributed to the DIF UR by the corresponding community. Currently, over 30 DID Methods are supported, e. g., $\{$*did:btcr*, *did:ethr*, *did:indy*, *did:peer*, *did:web*$\} \subset M$.

6.4 Universal VC Verifier (UV)

VC interoperability refers to seamless support for VCs across different VC flavours' signature schemes. To this end, we propose a driver-based architecture where each driver implements verification of VCs of a specific flavour. We thus formally specify a *Universal VC Verifier (UV)* in our architecture:

$$UV = (S, \sigma, V)$$

where $S \subseteq \mathcal{S}$ is the set of supported signature schemes, a function $\sigma : (\mathcal{D}_{VC} \cup \mathcal{D}_{VP}) \mapsto S$ that determines a VC's or VP's signature scheme, and $V = \{v_s | s \in \mathcal{S}\}$ is the set of implemented driver functions whose element functions $v_s : (\mathcal{D}_{VC} \cup \mathcal{D}_{VP}) \mapsto \{0, 1\}$ verify a VC or VP and return a boolean, i. e., true if and only if the VC or VP is verified. Slightly abusing notation, we denote the partial composition of σ and v by $v|_{s=\sigma} : (\mathcal{D}_{VC} \cup \mathcal{D}_{VP}) \mapsto \{0, 1\}$.

6.5 RDF Access Control (RAC)

The *RDF Access Control* component manages Access Control Rules (ACRs) for resources on some (external) resource server. ACRs are expressed using Solid's Web Access Control specification [14] which we extend with further semantics for defining required credentials using SHACL [32]. We thus formally specify an *RDF Access Control (RAC)* component:

$$RAC = (R, \{q, r\})$$

with a set of ACRs $R \subseteq \mathcal{D}$, and a set of functions, i. e., a query function $q : \mathcal{U} \mapsto R^n$ that determines any applicable ACR for an input URI, and a rule evaluation function $r : {\mathcal{D}_{VC}}^n \times R^n \mapsto \{0, 1\}$ that matches one or multiple VCs with one or multiple ACRs and returns a boolean, i. e., true if and only if the VCs satisfy at least one of the ACRs. That is, if all VC shape graphs of at least one ACR are matched by the VCs.

6.6 SSI User Agent

We formally define the *SSI User Agent (A)* and its functionality relevant to our archticture:

$$A = (W, \{qr, gen_{vp}\} \cup MS)$$

with a set of wallets $W = \{w | w = (U_{DID}, C)\}$ and a set of functions. Each wallet w is a couple of a DID and a set of associated VCs C with $C \subseteq \mathcal{D}_{VC}$. The set of functions is comprised by a function $qr : W \times \mathcal{D}_{VPR} \mapsto C^n$ to select VCs based on a VPR, and a function $gen_{vp} : C^n \times \mathcal{D}_{VPR} \mapsto \mathcal{D}$ to generate a message containing a VP from some VCs and a VPR, according to the credential exchange protocol of the *MM* component, e. g., HL Aries Present-Proof, and the functionality provided by an *MS* component, e. g., (un-) packing messages using the DIDComm Messaging Protocol [17].

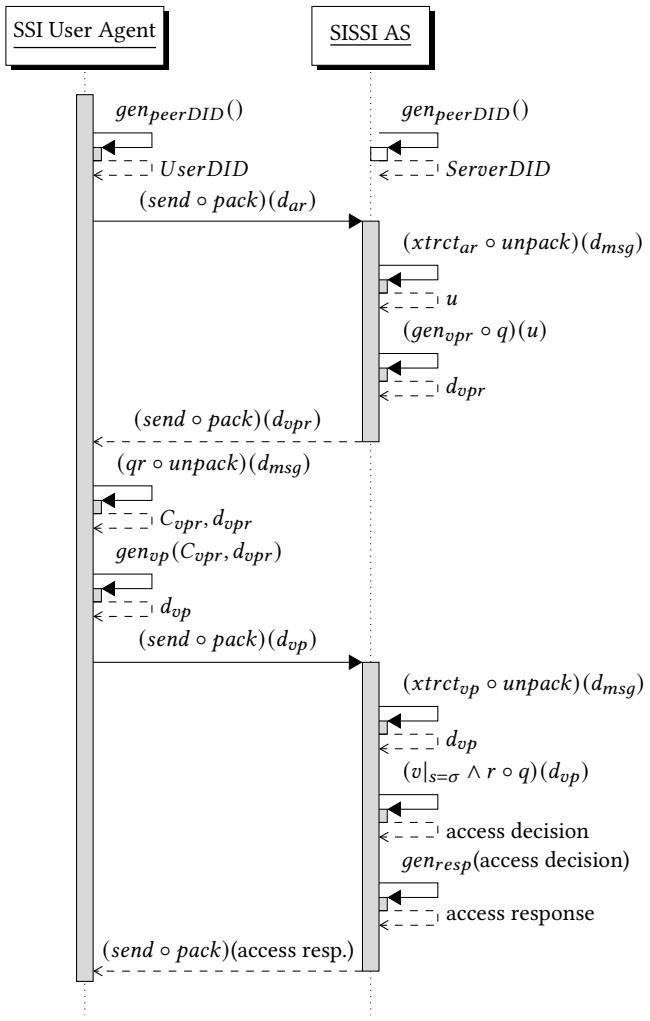


Figure 3: Authorization Process (see Section 6 for notation)

7 COMMUNICATION PROTOCOL

In this section, we briefly describe the communication protocol between the user's SSI agent and the SISSI AS. A compact illustration is provided by Figure 3.

Initialisation. To set up the AS, the set of ACRs R are provided, e. g., by connection to a triple store or from Web-accessible RDF documents. On the user-side, the SSI user agent A loads its sets of wallets W containing the user's DIDs and VCs. For communication, a ServerDID, chosen by the AS, and a UserDID, chosen by the user, are used by the corresponding *MS* in *pack* and *unpack* procedures. Either a long-lived DID or a new PeerDID, generated via $gen_{peerDID}$, is chosen. The ServerDID is made known to user agents, e. g., published on the Web or shared upon request.

Access Request. The user agent creates an *access-request* DIDComm message $d_{ar} \in \mathcal{D}$ specifying the requested resource by its URI $u \in \mathcal{U}$. It submits the message via HTTP POST to the AS's inbox: $(send \circ pack)(d_{ar})$. Upon receiving the request, the *MS* $unpack$s the d_{ar} from the message and *MM* extracts the URI u of the requested resource,$u = xtrct_{ar}(d_{ar})$. Via $(gen_{vpr} \circ q)(u)$, the appropriate ACRs $R_u = q(u)$ for the requested resource are determined and a corresponding *request-presentation* message $d_{vpr} = gen_{vpr}(R_u)$ is created. The message d_{vpr} is packed and sent to the user agent in a HTTP response: $(send \circ pack)(d_{vpr})$. A HTTP status code [23] of *401 Unauthorized* may emphasise the need for authentication.

Responding to the VPR. The user agent unpacks the received *request-presentation* message. By matching the VPR's SHACL shape graphs against the user's VCs, a subset of VCs to present is selected: $C_{vpr} = qr(d_{vpr})$ with $C_{vpr} \subseteq C$. The selected credentials C_{vpr} are then included in a new VP $d_{vp} = gen_{vp}(C_{vpr}, d_{vpr})$. Upon the user's approval, the user agent packs the VP *presentation* message and submits it via HTTP POST to the AS's inbox: $(send \circ pack)(d_{vp})$.

Handling the VP. Upon receiving the request, the *MS* $unpack$s the *presentation* message. The VP d_{vp} is extracted from the message d_{msg} via $d_{vp} = xtrct_{vp}(d_{msg})$. Then, $(v|_{s=\sigma} \wedge (r \circ q))(d_{vp})$ is executed: The VP d_{vp} is verified using the *UV*, $v|_{s=\sigma}(d_{vp})$. Any DID that is associated to any proof in the VP, is resolved using the UR's $\delta|_{m=\mu}$. By verifying the VP's proof, e. g., including a cryptopgraphic challenge, authenticity of the VP is ensured and thus agent A is authenticated. In addition, the VCs presented by d_{vp} are validated against the ACRs R_u, $(r \circ q)(d_{vp})$. Finally, an authorization decision message is generated, gen_{resp}, packed and send back to agent A as an HTTP response, $(send \circ pack)$.

The exact format of the authorization response depends on how the AS is integrated with a resource server (RS). For example, if the AS acts solely as an AS, some access token may be provided to the user agent. In case the AS is simultaneously an RS, the requested resource may be served.

8 EVALUATION

To prove the viability of the SISSI architecture, we implement a simple Proof-of-Concept SISSI access control system (ACS)[7]. We evaluate the performance of an authorization process, emphasising the impact of an implementor's choice of infrastructure, as SISSI is infrastructure-agnostic. The prototype implements all components presented in Section 6, i. e., including DIDComm and HL Aries Present-Proof Protocol. The user agent is initialized with a wallet storing multiple DIDs and VCs, and provides scripts for communication with the AS as described in Section 7. Overall, we show that the End-to-End (E2E) latency for an authorisation process only takes a couple seconds depending on network latency and DID Method. Web-based DIDs outperform blockchain-based approaches in terms of execution time.

8.1 Setup

The evaluation was executed on a Virtual Machine (VM) running SMP Debian 4.19.98-1+deb10u1 (2020-04-27) x86_64 with 4 CPU cores (2.4Ghz) and 8GB RAM. The server is located in Karlsruhe, Germany. We host ACS components on Docker version 20.10.18.

[7] https://github.com/vpapanchev/ssi-acs

As an overall evaluation of the SISSI ACS, we analyse the E2E latency of an authorization process, as depicted in Figure 3. We use an SSI user agent which applies caching and in-memory credentials to minimise impact of the user agent on the evaluation. We compare two versions of the ACS: One without parallel resolution of DIDs and one with parallel execution. Other optimizations have not been implemented, i. e., our prototype does not fetch JSON-LD contexts in parallel and does not use caching of remote resource.

We compare three evaluation cases for authorization requests: Indy-based, Ethr-based and Web-based requests. The evaluation case depends on the type of DID to be resolved when verifying a VP or VC. We thus configure the *UR* with three DID Methods: an EthrDID driver connected to the Ethereum test network Ropsten[8] and an IndyDID driver connected to a locally deployed VON network[9]. A third driver supports Web-based DIDs, which we serve on a Solid Pod hosted at https://solidweb.org. This way, we compare a typical Web-based scenario using WebDIDs with blockchain-based scenarios distinguishing between a local scenario (Indy) and a remote scenario (Ethr).

We initialize the SSI user agent with separate wallets, each for one type of DIDs. Each wallet stores JSON-LD credentials expressed as JWTs. Further, we define such ACRs that exacly one VC is required per access request. Therefore, the handling of each request requires exactly two DID resolutions: one for the DID associated with the VC proof and one for the DID associated with the VP proof.

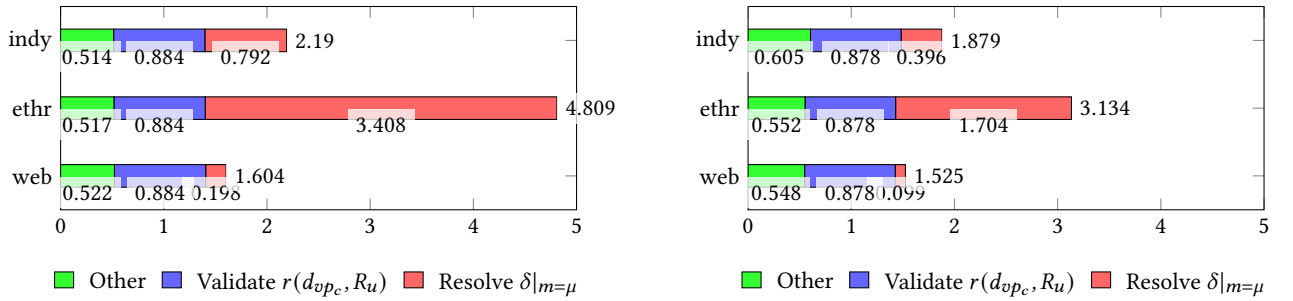
8.2 Results

We calculate the results for the three evaluation cases as the average of 500 executions per request type. Figure 4a shows the average E2E latency with sequential DID resolution for **Indy-based requests, 2.190 seconds**, for **Ethr-based requests, 4.809 seconds**, and for **Web-based requests, 1.604 seconds**. Figure 4 also decomposes the E2E latency into its components; mostly comprised of DID resolution time, $\delta|_{m=\mu}$, and the validation of the required VCs against ACRs, $r(d_{vp_c}, R_u)$. In r, a significant part of execution time is due to fetching remote JSON-LD context from the Web, according to our measurements around 0.46 seconds. Other overhead is caused by minor ACS routines with negligible performance impact on their own, e. g., cryptographically verifying VPs, which only takes on average around 0.050 seconds.

During the E2E evaluation, we measure the average resolution time for **IndyDIDs, 0.396 seconds** and for **EthrDIDs, 1.704 seconds**. Resolution times vary between different DID Methods mainly due to the difference in network latency between a remote call for EthrDIDs and a local one for IndyDIDs. Resolution of **Web-based DIDs, 0.099 seconds**, also requires remote calls to non-local Web resources, but those are answered almost instantaneously.

As network latency to a remote resource can hardly be improved here, the only possible optimization option regarding DID resolution is parallel request execution. Moreover, avoiding network latency by only deploying local blockchain nodes is hardly feasible due to the required resources to support any blockchain-based DID. Hence, remote calls pose a more realistic scenario. But even when served locally, blockchain-based DIDs induce significant overhead.

[8] https://ropsten.etherscan.io/
[9] https://github.com/bcgov/von-network



(a) Composition of E2E latency (in *s*) with sequential DID resolution. **(b) Composition of E2E latency (in *s*) with parallel DID resolution.**

Figure 4: Composition of E2E latency (in *s*): Resolve DIDs (red, right-most bar part), Validating VCs against ACRs (blue, center bar part), and the sum of other procedures (green, left-most bar part).

As DID resolution poses the majority of execution time, especially in the blockchain-based scenarios, we evaluate the same procedures in a system with parallel DID resolution. As expected, with some parallelisation overhead (noteable in slightly increased *other* time in Figure 4b), total time is reduced roughly by the time of one DID resolution. Other differences are within margin of error.

Our results show that a barely optimised system, even without caching remote resources, E2E execution time is considerably lower for Web-based scenarios. Introducing blockchains into the system comes at a significant overhead: When participating locally in a blockchain network, as in our Indy-based evaluation case, the blockchain mainly lies in execution overhead of fetching relevant transactions and re-creating the DID document. However, relying only on local blockchain nodes is not really feasible in the context of our system. Then, any supported DID Method should be locally executable and thus any blockchain should be served locally. Instead, and more realisticly, blockchain-based DIDs are resolved in remote calls as in our Ethr-based evaluation case. For those cases however, overhead in the magnitude of seconds is introduced by DID Method execution and network latency.

9 CONCLUSION

We presented the SISSI architecture for access control on the Web. The architecture is based on the Self-Sovereign Identity paradigm and achieves semantic interoperability of corresponding concepts using existing Web Standards. Instead of retrofitting an IDP, we omit the IDP completely as identities should remain truly self-sovereign.

Access control is defined using user- and machine-understandable ACRs. Moreover, we capitalise on the VC data model [48] as it provides an underlying RDF data model out-of-the-box. This allows for seamless integration with semantically modelled ACRs [14] . We rely on W3C standard of SHACL [32] for formal validation while providing explicit semantics at the same time.

The SISSI architecture builds on well-established protocols within the SSI ecosystem, e. g., relying on the HL Aries Present-Proof Protocol [31] on top of the DIDComm Messaging Protocol [17] for communication. Combining the DIDComm Messaging protocol with PeerDIDs [28] provides an additional layer of privacy. However, whether to use a PeerDID or a long-lived DID for the communication is ultimately still the user's choice.

Moreover, our architecture allows for a flexible and incremental extensibility for adding new services or substituting existing components. The independent evolvability is an important benefit, as, due to the decentralized nature and rapid development of the SSI paradigm, updates in the core SSI technologies are to be expected. For example, as there currently exist 136 registered DID Methods [50] allegedly adhering to the W3C DID standard [47] without a common interface, we argue that the standard falls behind creating interoperability of DID systems. Therefore, approaches as used in our architecture have become a necessity in the first place: Resolving DIDs of different DID Methods to their DID Documents is possible relying on the Universal DID Resolver. Moreover, the set of supported DID Methods M can dynamically be extended, e. g., to support new DID Methods, by deploying additional drivers. Similar to the W3C DID standard [47], the W3C VC data model standard [48] does not define a standard way of verifying a credential thereby creating the horizontal interoperability challenge as outlined in Section 4.4. Hence, approaches as proposed with our architecture are necessary to solve such issue: Relying on our Universal VC Verifier, support for verifying different VC flavours is provided and extensible by integration of new drivers. The SISSI architecture thus achieves the desired interoperability across SSI technologies and addresses the in Section 4.4 outlined challenges.

We found with our evaluation that even a simple Proof-of-Concept of the SISSI architecture performs reasonably well: Depending on the DID Methods, an authorization process takes between 1.5 and 3.1 seconds. A Web-based approach notably outperfoms blockchain-based approaches in terms of execution time.

With our architecture, we aim to provide the foundation for tackling practical issues induced by the controversial W3C recommendations for VCs [48] and DIDs[47]. We hope to showcase the viability of semantic self-sovereign identities for access control on the Web. With our work, we aim to showcase how to leverage the already existing explicit semantics in SSI technologies and ecosystems, as we believe that SSI on the Web has a lot of potential, e. g., within the Solid Protocol [15].

ACKNOWLEDGMENTS

This work is supported in part by the German federal ministry of education and research (BMBF) in MANDAT (FKZ 16DTM107B).

REFERENCES

[1] Christopher Allen. 2016. *The Path to Self-Sovereign Identity.* http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[2] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic *k*-TAA. In *Proc. of the 5th SCN.*

[3] Rafael Belchior, Benedikt Putz, Günther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. 2020. SSIBAC: Self-Sovereign Identity Based Access Control. In *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021*, Guojun Wang, Ryan K. L. Ko, Md. Zakirul Alam Bhuiyan, and Yi Pan (Eds.). IEEE, 1935–1943.

[4] Tim Berners-Lee. 2006. *Linked Data.* https://www.w3.org/DesignIssues/LinkedData

[5] Tim Berners-Lee, Roy Fielding, and Larry Masinter. 2005. *Uniform Resource Identifier (URI): Generic Syntax.* Internet Standards Track document. IETF. https://www.ietf.org/rfc/rfc3986.txt.

[6] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *Proc. of the 24th CRYPTO (LNCS, Vol. 3152).* Springer, 41–55.

[7] Matthieu Bosquet. 2022. *Access Control Policy (ACP).* Editor's Draft. W3C Solid Community Group. https://solid.github.io/authorization-panel/acp-specification/

[8] Andre Boysen. 2021. Decentralized, Self-Sovereign, Consortium: The Future of Digital Identity in Canada. *Frontiers Blockchain* 4 (2021), 624258. https://doi.org/10.3389/fbloc.2021.624258

[9] Christoph Braun and Tobias Käfer. 2022. Attribute-based Access Control on Solid Pods using Privacy-friendly Credentials. In *Proceedings of the Posters and Demo Track at the 18th International Conference on Semantic Systems (SEMANTiCS) (CEUR Workshop Proceedings, Vol. 2451).* CEUR-WS.org.

[10] Daniel Buchner, Brent Zundel, and Martin Riedel. 2021. *Presentation Exchange v1.0.0.* DIF Ratified Specification. DIF: Decentralized Identity Foundation. https://identity.foundation/presentation-exchange/spec/v1.0.0/#json-schema

[11] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Proc. of EUROCRYPT 2001 (LNCS, Vol. 2045).* Springer, 93–118.

[12] Jan Camenisch and Anna Lysyanskaya. 2002. A Signature Scheme with Efficient Protocols. In *Revised Papers of the 3rd SCN (LNCS, Vol. 2576).* Springer, 268–289.

[13] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Proc. of the 24th CRYPTO (LNCS, Vol. 3152).* Springer, 56–72.

[14] Sarven Capadisli. 2022. *Web Access Control.* Editor's Draft. W3C Solid Community Group. https://solid.github.io/web-access-control-spec/

[15] Sarven Capadisli, Tim Berners-Lee, Ruben Verborgh, and Kjetil Kjernsmo. 2021. *Solid Protocol.* Version 0.9.0. W3C Solid Community Group. https://solidproject.org/TR/protocol

[16] Spela Cucko and Muhamed Turkanovic. 2021. Decentralized and Self-Sovereign Identity: Systematic Mapping Study. *IEEE Access* 9 (2021), 139009–139027. https://doi.org/10.1109/ACCESS.2021.3117588

[17] Sam Curren, Tobias Looker, and Oliver Terbu. 2021. *DIDComm Messaging.* Editor's Draft. DIF: Decentralized Identity Foundation. https://identity.foundation/didcomm-messaging/spec/

[18] Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. *RDF 1.1 Concepts and Abstract Syntax.* W3C Recommendation. W3C. https://www.w3.org/TR/rdf11-concepts/.

[19] Shelby Solomon Darnell and Joseph Sevilla. 2021. 3 Stages of a Pan-African Identity Framework for Establishing Self-Sovereign Identity With Blockchain. *Frontiers Blockchain* 4 (2021), 631640. https://doi.org/10.3389/fbloc.2021.631640

[20] Flaviene Scheidt de Cristo, Wazen M. Shbair, Lucian Trestioreanu, Radu State, and Aanchal Malhotra. 2021. Self-Sovereign Identity for the Financial Sector: A Case Study of PayString Service. In *2021 IEEE International Conference on Blockchain, Blockchain 2021, Melbourne, Australia, December 6-8, 2021*, Yang Xiang, Ziyuan Wang, Honggang Wang, and Valtteri Niemi (Eds.). IEEE, 213–220. https://doi.org/10.1109/Blockchain53845.2021.00036

[21] Kayode Yadilichi Ezike. 2019. *SolidVC : a decentralized framework for Verifiable Credentials on the web.* Master's thesis. MIT EECS. https://hdl.handle.net/1721.1/121667

[22] Roy Fielding and Julian Reschke. 2014. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.* Internet Standards Track document. IETF. https://www.ietf.org/rfc/rfc7230.txt.

[23] Roy Fielding and Julian Reschke. 2014. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* Internet Standards Track document. IETF. https://www.ietf.org/rfc/rfc7231.txt.

[24] Maria Freytsis, Iain Barclay, Swapna Krishnakumar Radha, Adam Czajka, Geoffrey H. Siwo, Ian J. Taylor, and Sherri L. Bucher. 2021. Development of a Mobile, Self-Sovereign Identity Approach for Facility Birth Registration in Kenya. *Frontiers Blockchain* 4 (2021), 631341. https://doi.org/10.3389/fbloc.2021.631341

[25] Fariba Ghaffari, Emmanuel Bertin, Julien Hatin, and Noël Crespi. 2020. Authentication and Access Control based on Distributed Ledger Technology: A survey. In *2nd Conference on Blockchain Research & Applications for Innovative Networks and Services, BRAINS 2020, Paris, France, September 28-30, 2020.* IEEE, 79–86.

[26] Andreas Grüner, Alexander Mühle, and Christoph Meinel. 2021. Analyzing Interoperability and Portability Concepts for Self-Sovereign Identity. In *20th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2021, Shenyang, China, October 20-22, 2021.* IEEE, 587–597.

[27] Andreas Grüner, Alexander Mühle, and Christoph Meinel. 2021. ATIB: Design and Evaluation of an Architecture for Brokered Self-Sovereign Identity Integration and Trust-Enhancing Attribute Aggregation for Service Provider. *IEEE Access* 9 (2021), 138553–138570.

[28] Daniel Hardman. 2021. *Peer DID Method Specification.* Editor's Draft. https://identity.foundation/peer-did-method-spec/

[29] Bahar Houtan, Abdelhakim Senhaji Hafid, and Dimitrios Makrakis. 2020. A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare. *IEEE Access* 8 (2020), 90478–90494. https://doi.org/10.1109/ACCESS.2020.2994090

[30] M. Jones, J. Bradley, and N. Sakimura. 2015. *JSON Web Token (JWT).* Internet Standards Track document. IETF. https://www.ietf.org/rfc/rfc7519.txt.

[31] Nikita Khateev and Stephen Curran. 2021. *Aries RFC 0454: Present Proof Protocol 2.0.* Aries RFC. Hyperledger Indy Community. https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md

[32] Holger Knublauch and Dimitris Kontokostas. 2017. *Shapes Constraint Language (SHACL).* W3C Recommendation. W3C. https://www.w3.org/TR/shacl/

[33] Andre Kudra. 2022. Self-Sovereign Identity (SSI) in Deutschland. *Datenschutz und Datensicherheit* 46, 1 (2022), 22–26. https://doi.org/10.1007/s11623-022-1555-1

[34] Tobias Kuhn, Christine Chichester, Michael Krauthammer, Núria Queralt-Rosinach, Ruben Verborgh, George Giannakopoulos, Axel-Cyrille Ngonga Ngomo, Raffaele Viglianti, and Michel Dumontier. 2016. Decentralized provenance-aware publishing with nanopublications. *PeerJ Comput. Sci.* 2 (2016), e78.

[35] Michael Kuperberg and Robin Klemens. 2022. Integration of Self-Sovereign Identity into Conventional Software using Established IAM Protocols: A Survey. In *Open Identity Summit 2022, Copenhagen, Denmark, July 7-8, 2022 (LNI, Vol. P-325)*, Heiko Roßnagel, Christian H. Schunck, and Sebastian Mödersheim (Eds.). Gesellschaft für Informatik e.V., 51–62. https://doi.org/10.18420/OID2022_04

[36] Gabriella Laatikainen, Taija Kolehmainen, and Pekka Abrahamsson. 2021. Self-sovereign identity ecosystems: benefits and challenges. In *Scandinavian Conference on Information Systems.* Association for Information Systems.

[37] Tobias Looker and Orie Steele. 2022. *BBS+ Signatures 2020.* Draft CG Report. W3C Credentials CG. https://w3c-ccg.github.io/ldp-bbs2020/#the-bbs-signature-suite-2020

[38] Stanislav Mahula, Evrim Tan, and Joep Crompvoets. 2021. With blockchain or not? Opportunities and challenges of self-sovereign identity implementation in public administration: Lessons from the Belgian case. In *DG.O'21: The 22nd Annual International Conference on Digital Government Research, Omaha, NE, USA, June 9-11, 2021*, Jooho Lee, Gabriela Viale Pereira, and Sungsoo Hwang (Eds.). ACM, 495–504. https://doi.org/10.1145/3463677.3463705

[39] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. 2018. A survey on essential components of a self-sovereign identity. *Comput. Sci. Rev.* 30 (2018), 80–86.

[40] Nick Ragouzis, John Hughes, Rob Philpott, Eve Maler, Paul Madsen, and Tom Scavo. 2008. *Security Assertion Markup Language (SAML) V2.0 Technical Overview.* Committee Draft 02. OASIS. https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[41] Imen Riabi, Hella Kaffel Ben Ayed, and Leïla Azouz Saïdane. 2019. A survey on Blockchain based access control for Internet of Things. In *15th International Wireless Communications & Mobile Computing Conference, IWCMC 2019, Tangier, Morocco, June 24-28, 2019.* IEEE, 502–507.

[42] Sara Rouhani, Rafael Belchior, Rui Santos Cruz, and Ralph Deters. 2021. Distributed attribute-based access control system using permissioned blockchain. *World Wide Web* 24, 5 (2021), 1617–1644.

[43] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. 2014. *OpenID Connect Core 1.0.* Final Specification. https://openid.net/specs/openid-connect-core-1_0.html

[44] A. Sambra, E. Mansour, Sandro Hawke, Maged Zereba, N. Greco, Abdurrahman Ghanem, Dmitri Zagidulin, A. Aboulnaga, and Tim Berners-Lee. 2016. *Solid : A Platform for Decentralized Social Applications Based on Linked Data.* Technical Report. MIT CSAIL & Qatar Computing Research Institute.

[45] Andrei Sambra, Henry Story, and Tim Berners-Lee. 2014. *WebID 1.0 - Web Identity and Discovery.* W3C Editor's Draft. W3C. https://www.w3.org/2005/Incubator/webid/spec/identity/

[46] Reza Soltani, Uyen Trang Nguyen, and Aijun An. 2021. A Survey of Self-Sovereign Identity Ecosystem. *Secur. Commun. Networks* 2021 (2021), 8873429:1–8873429:26.
[47] Manu Sporny, Amy Guy, Markus Sabadello, and Drummond Reed. 2022. *Decentralized Identifiers (DIDs).* W3C Recommendation. W3C. https://www.w3.org/TR/did-core/
[48] Manu Sporny, Grant Noble, Dave Longley, Daniel C. Burnett, Brent Zundel, and Kyle Den Hartog. 2021. *Verifiable Credentials Data Model.* W3C Recommendation. W3C. https://www.w3.org/TR/vc-data-model/
[49] Manu Sporny, Drummond Reed, and Orie Steele. 2020. *Linked Data Cryptographic Suite Registry.* Draft Community Group Report. W3C Credentials Community Group. https://www.w3.org/TR/vc-data-model/
[50] Orie Steele and Manu Sporny. 2023. *DID Specification Registries.* W3C Group Note. W3C DID Working Group. https://www.w3.org/TR/did-spec-registries/#did-methods
[51] Werbrouck, Jeroen and Taelman, Ruben and Verborgh, Ruben and Pauwels, Pieter and Beetz, Jakob and Mannens, Erik. 2020. Pattern-based access control in a decentralised collaboration environment. In *Proceedings of the 8th Linked Data in Architecture and Construction Workshop, LDAC 2020* (Dublin), Vol. 2636. 118–131.
[52] Hakan Yildiz, Axel Küpper, Dirk Thatmann, Sebastian Göndör, and Patrick Herbke. 2022. A Tutorial on the Interoperability of Self-sovereign Identities. *CoRR* abs/2208.04692 (2022). https://doi.org/10.48550/arXiv.2208.04692 arXiv:2208.04692
[53] Kaliya Young. 2021. *Verifiable Credentials Flavors Explained.* Technical Report. COVID-19 Credentials Initiative. https://www.lfph.io/wp-content/uploads/2021/02/Verifiable-Credentials-Flavors-Explained.pdf.
[54] Razieh Nokhbeh Zaeem, Kai Chih Chang, Teng-Chieh Huang, David Liau, Wenting Song, Aditya Tyagi, Manah Khalil, Michael Lamison, Siddharth Pandey, and K. Suzanne Barber. 2021. Blockchain-Based Self-Sovereign Identity: Survey, Requirements, Use-Cases, and Comparative Study. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT).* ACM, 128–135.

A EXAMPLES

```
{
  "@context": "https://purl.org/sissi/messages/v1",
  "@type": "AccessRequest",
  "@id": "<uuid-access-request>",
  "target": "https://example.org/resources/r1",
  "mode": "http://www.w3.org/ns/auth/acl#read"
}
```

Listing 2: An example access-request message

```
{
  "@context": "https://purl.org/sissi/messages/v1",
  "@type": "https://didcomm.org/present-proof/2/
     request-presentation",
  "@id": "<uuid-request-presentation>",
  "will_confirm": false,
  "present_multiple": false,
  "formats": [
    {
      "attach_id": "attachment_0",
      "format": "https://purl.org/sissi/messages/ns#
     shaclVPR"
    },
    {
      "attach_id": "attachment_1",
      "format": "https://purl.org/sissi/messages/ns#
     shaclVPR"
    }
  ],
  "request_presentations~attach": [
    {
      "@id": "attachment_0",
      "@type": "https://purl.org/sissi/messages/ns#
     shaclVPR",
      "mime-type": "application/json+ld",
      "data": {
        "string": "<VPR as stringified jsonld   | see
     example (vpr_shacl.jsonld)>",
        "base64": "<VPR as base64 encoded   | as
     alternative>"
      }
    },
    {
      "@id": "attachment_1",
      "@type": "https://purl.org/sissi/messages/ns#
     shaclVPR",
      "mime-type": "text/turtle",
      "data": {
        "string": "<VPR as turtle plain string  | see
     example (vpr_shacl.ttl)>",
        "base64": "<VPR as base64 encoded   | as
     alternative>"
      }
    }
  ]
}
```

Listing 3: An example request-presentation message

```
{
    "@context": "https://purl.org/sissi/messages/vpr/
     v1",
    "@type": "https://purl.org/sissi/messages/vpr/ns#
     shaclVPR",
    "required_credentials": [
        [
            {
            "@type":["http://www.w3.org/ns/shacl#
     NodeShape"],
            "http://www.w3.org/ns/shacl#targetClass"
     :{"@id":"https://www.w3.org/2018/credentials#
     VerifiableCredential"},
            "http://www.w3.org/ns/shacl#class":[{"@id
     ":"<http://example.org/edu#Student>"}],
            "http://www.w3.org/ns/shacl#property":[{"
     @id":"_:b2"}]
            },
            {
            "@id":"_:b2",
            "http://www.w3.org/ns/shacl#path": {"@id":
     "https://www.w3.org/2018/credentials#issuer"},
            "http://www.w3.org/ns/shacl#in":[
                {"@id":"_:b3"}]},
```

```
            {"@id":"_:b3",
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#first":{"@id":"did:ethr:0x3:0x032..5d7"},
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#rest":{"@id":"_:b4"}},
            {"@id":"_:b4",
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#first":{"@id":"did:indy:local:YY8..6tZ"},
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#rest":{"@id":"http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"}}
        ]
    ],
    "options": {
        "nonce": "<challenge nonce>",
        "domain": "<challenge domain>"
    }
}
```

Listing 4: An example SHACL VPR as JsonLD

```
@prefix sh: <http://www.w3.org/ns/shacl#>.
@prefix cred: <https://www.w3.org/2018/credentials#>.
@prefix sissi: <https://purl.org/sissi/messages/vpr/ns#>.

_:vpr  a  <https://purl.org/sissi/messages/vpr/ns#shaclVPR>;
    sissi:options [
        sissi:nonce "<challenge nonce>" ;
        sissi:domain "<challenge domain>"
    ];
    sissi:required_credentials _:studentIssuerShape .

_:studentIssuerShape
    a              sh:NodeShape ;
    sh:targetClass cred:VerifiableCredential ;
    sh:class       <http://example.org/edu#Student> ;
    sh:property
      [   sh:path  cred:issuer ;
          sh:in    ( <did:ethr:0x3:0x032..5d7>
                     <did:indy:local:YY8..6tZ> ) ] .
```

Listing 5: An example SHACL VPR as Turtle

```
{
  "@context": "https://purl.org/sissi/messages/v1",
  "@type": "https://didcomm.org/present-proof/2/presentation",
  "@id": "<uuid-presentation>",
  "last_presentation": true,
  "formats": [
    {
      "attach_id": "attachment_0",
      "format": "https://purl.org/sissi/messages/ns#shaclVP"
    }
  ],
  "presentations~attach": [
    {
      "@id": "attachment_0",
      "@type": "https://purl.org/sissi/messages/ns#shaclVP",
      "mime-type": "application/json+ld",
      "data": {
        "string": "<VP as stringified jsonld | see example (vp.jsonld)>>",
        "jwt": "<VP as JWT | as alternative>",
        "base64": "<VP as base64 encoded | as alternative>"
      }
    }
  ]
}
```

Listing 6: An example presentation message

```
{
    "@context": "https://purl.org/sissi/messages/v1",
    "@type": "https://purl.org/sissi/messages/ns#AccessResponse",
    "@id": "<uuid-access-response>",
    "target": "https://example.org/resources/r1",
    "mode": "<http://www.w3.org/ns/auth/acl#read>",
    "ok": "<true|false>",
    "accessToken" : "..."
}
```

Listing 7: An example access-response message