An Integration Architecture to Enable Service Providers for Self-sovereign Identity

Andreas Grüner, Alexander Mühle, Christoph Meinel
Hasso Plattner Institute (HPI)
University of Potsdam, 14482, Potsdam, Germany
Email: {andreas.gruener, alexander.muehle, christoph.meinel}@hpi.uni-potsdam.de

***Abstract*—The self-sovereign identity management model emerged with the rise of blockchain technology. This paradigm focuses on user-centricity and strives to place the user in full control of the digital identity. Numerous implementations embrace the self-sovereign identity concept, leading to a fragmented landscape of solutions. At the same time, traditional identity and access management protocols are largely disregarded and facilities to issue verifiable claims as attributes are not available. Therefore, service providers barely adopt these solutions. We propose a component-based architecture for integrating self-sovereign identity solutions into web applications to foster their adoption by service providers. Furthermore, we outline a sample implementation as a gateway that enables uPort and Jolocom for authentication, via the OpenID Connect protocol, as well as the retrieval of email address attestations for these solutions.**

***Index Terms*—Blockchain, distributed ledger technology, digital identity, self-sovereign identity, trust, identity management**

I. Introduction and Motivation

Blockchain technology advanced from an elementary digital cash system, that did not require an issuing bank, to a general decentralized execution platform [1]. Therefore, blockchain technology enables the implementation of a decentralized identity provider [2] that is not owned by a single entity. Thus, the identity provider does not represent a trusted third party and enables digital identities that are under full control of the associated subject. Christopher Allen [3] proposed principles of self-sovereignty highlighting user control, access, consent and protection. These characteristics are favourable for a user and constitute the basis for the self-sovereign identity (SSI) management model.

In the course of this development, a multitude of competing SSI solutions that pursue different concepts and utilize various blockchains have emerged [4]. Sovrin [5] applies several distributed ledgers that are dedicated to identity management. The core of uPort [2] is a set of smart contracts implemented on Ethereum [6]. Jolocom [7] also relies on smart contracts that are stored on Ethereum, but follows different implementation guidelines. Blockstack [8] implements a meta approach that is independent of a specific blockchain. For a service provider, the integration into all SSI solutions is neither efficient nor achievable in terms of cost and maintenance effort.

Standardization initiatives drive the definition of protocols. World Wide Web Consortium (W3C) working groups specify standards for Decentralized Identifiers (DID) [9] and verifiable claims. A DID uniquely references a digital identity and enables the derivation of the corresponding SSI platform. Besides that, the DID Auth [10] standard is being developed to provide a common authentication protocol for SSI solutions. However, these norms are partially not yet mature enough and have limited adoption. Apart from that, established identity and access management protocols, for instance, OpenID Connect (OIDC) [11] are widely unconsidered.

The SSI ecosystem significantly depends on the attributes of a digital identity. Trust in attributes cannot be derived from the reputation of the identity provider as a trusted third party anymore. Therefore, attributes are modelled as verifiable claims that are comprised of claims and attestations. A service provider requires attested claims, for instance, name, address or age, for service provisioning.

Overall, the SSI solutions are focused foremost on user requirements, but obviously, disregard the demands on the side of the service provider. However, the adoption of SSI implementations by service providers is a fundamental prerequisite to evolve the complete ecosystem. A user will employ an SSI solution if enough service providers accept this solution. Vice versa, the service provider will embrace an SSI solution if the user base is large enough.

Our main contribution is a service provider friendly integration architecture with the following characteristics:

1) Abstraction from a single SSI solution
2) Integration by an established protocol
3) Providing a verifiable claim issuance facility

Furthermore, we have implemented a prototype of the architecture as a gateway[1] and integrated it for authentication with the tele-TASK[2] portal.

The remainder of this paper is organized as follows. In Section II, we discuss related work and compare it to our approach. Subsequently, we present in detail our architecture in Section III. In Section IV, we outline a sample implementation as a gateway. Afterwards, we evaluate our solution in Section V and show its practical feasibility based on a generic authentication process. Finally, we discuss observations in Section VI and conclude our research in Section VII.

[1]The prototype of the gateway is available under https://ssixa.de
[2]tele-TASK is available under https://tele-task.de

978-1-7281-2522-0/19/$31.00

II. Related Work

The Universal Resolver [12] is an approach to integrate SSI solutions. The basic concept is to resolve a given DID to a corresponding DID Document [9]. The DID Document is comprised of information about the subject of the DID, public keys for authentication, authorization data, various timestamps and service endpoints. Service endpoints refer to further interaction possibilities of the corresponding DID. A driver-based implementation is the core part of the Universal Resolver. For each SSI solution, a driver is implemented for querying required information to construct the DID Document or to directly retrieve the document from the SSI platform.

In our opinion, there are some drawbacks. The Universal Resolver has a specified application programming interface but does not implement a standard identity and access management protocol. The Universal Resolver literally transforms a DID to a DID Document [12]. Subsequently, authentication and authorization need to be conducted based on public key information or service endpoints of the DID Document. In case the listed public keys are used for authentication, SSI specific additions, e.g. supplementary verifications, might need to be implemented on the service provider side. Overall, the Universal Resolver abstracts from a dedicated SSI solution by increasing the necessary steps and complexity for authentication.

III. Architecture

The architecture enables a service provider to offer SSI solutions for user authentication and attribute-based authorization. Fig. 1 presents an overview of the architecture and the surrounding actors. The service provider offers a web application for user interaction, for instance an online shop. Additionally, the user owns an SSI client to create and control its digital identity.

An implementation of our integration architecture connects the web application of the service provider, the SSI solution and the user. It essentially represents a gateway that mediates communication between these parties and retrieves, respectively provides, the required digital identity data. The architecture encompasses the components Verified Claim Issuer, Claim Name Translator, SSI Broker, Trust Engine and Protocol Handler.

A. Verified Claim Issuer

The Verified Claim Issuer component enables a user to retrieve attested claims for its self-sovereign identity. It provides a user interface to obtain input and conduct a verification process. The component reflects a framework for verification due to different validation processes for distinct types of claims. Additionally, the component may communicate to further systems to collect validation data.

- **Input:** Name and value of a claim for a digital identity
- **Output:** Verified Claim for a digital identity

B. Claim Name Translator

The Claim Name Translator module conveys claim names from various formats into a consistent internal representation and vice versa. Different authentication protocols and SSI solutions identify the same logical attribute by distinct names. To achieve interoperability, a translation between the different names is required.

- **Input:** Claim name in a domain
- **Output:** Claim name in another domain

C. Protocol Handler

The Protocol Handler controls input and output from established identity and access management protocols. The component abstracts from the peculiarities of a single protocol and enables a service provider application to chose between a variety of standards for integration. For each supported protocol a respective implementation is required that is depicted as e.g. *Protocol 1* in Fig. 1. The Protocol Handler component and the associated protocols are the connection point to the web application.

- **Input:** Credentials of a digital identity
- **Output:** Assertion of a digital identity

D. SSI Broker

The SSI Broker manages the interaction with different SSI solutions that are outlined as e.g. *SSI 1* in Fig. 1. In general, the SSI Broker conceptualizes the interaction with the SSI implementations to authenticate a user, to retrieve attributes from a user and to assign verifiable claims to a digital identity of a user. On the one hand, the SSI Broker can interact directly with the SSI client of the user by creating Quick Response (QR) codes that are used, e.g. for login. Additionally, direct communication to the SSI network is done for verification of the digital identity and the retrieved attributes.

- **Input:** Selected interface function and required data
- **Output:** Communication to SSI client and platform

E. Trust Engine

The Trust Engine is the central element that applies a trust model to the attributes of the digital identity. The attestation issuers of a verifiable claim are evaluated. In case, the attestation issuers or a combination of different issuers are considered as trustworthy by the assessed trust model, the claim is taken as a valid attribute of the digital identity and forwarded to the service provider upon request. The recorded trust model is required to be aligned to the opinion of the service provider about the trustworthiness of certain issuers. Either the service provider hosts an instance of the gateway or trusts the hosting party.

- **Input:** Claim and its attestations
- **Output:** Validity of the claim

IV. Implementation

In the following section, we outline specific details of a sample implementation of our proposed integration architecture. Thereby, we present the implemented modules according to the different components.

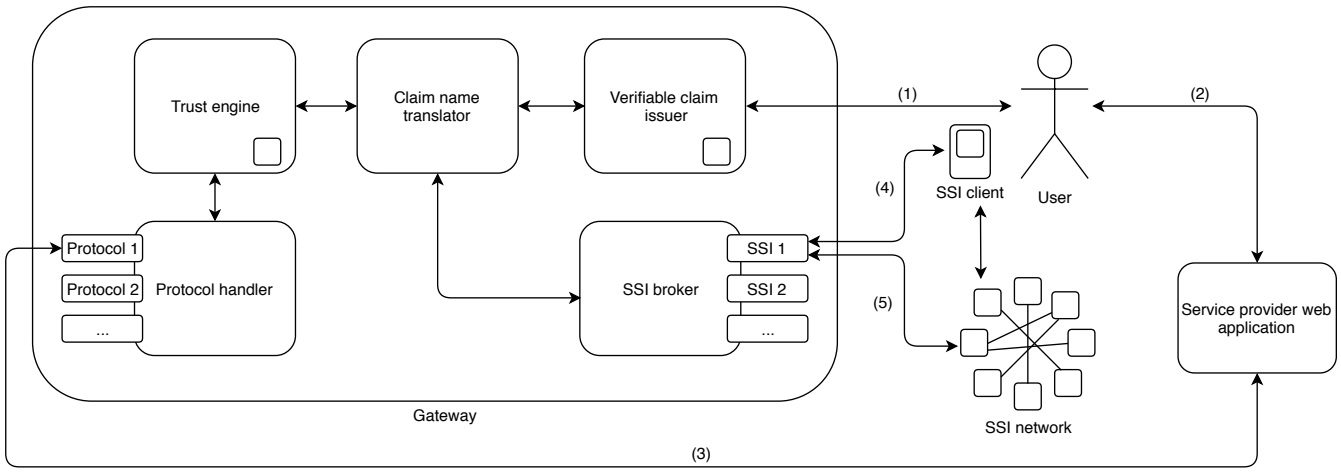

Fig. 1. Architecture and environment

A. Verified Claim Issuer

The implementation encompasses the attestation of email addresses. After authentication to the gateway, the user is able to start the verification process of an email address. In the first step, the user enters an email address and starts the validation. The gateway sends a verification email to the address. This email comprises a link to the gateway containing a randomly generated number. By opening the provided link, the user proves control over the mailbox. Subsequently to the successful completion of the verification, the user is able to retrieve a verifiable claim about the email address. The user obtains the claim with the support of the SSI client by scanning the provided QR code.

B. Claim Name Translator

The Claim Name Translator compensates for differences in the naming of claims within the supported protocols and SSI solutions. Claim name translation is used for issuing verifiable claims and the retrieval of user attributes during the authentication process. An overview of the different names is listed in Table I. The email address of a user is requested via the OIDC authentication flow as attribute *email*. In a comparable way, the SSI solution uPort references the email address by the term *email*. In contrast, Jolocom refers to this property with the notion *ProofOfEmailCredential*. The gateway utilizes the default OIDC claim name for an attribute in case the evaluation of the trust model succeeds. Otherwise, the attribute is made available through the claim name by appending *_unverified*. An application may decide to use unverified attributes for non-critical scenarios, e.g. displaying a welcome message for a user.

C. Protocol Handler

The protocol handler implements the authorization code flow of the OIDC protocol including the authorization, user info and token endpoints. A web application can request user authentication with blockchain-based self-sovereign identity methods by using the relative path */blockchain/challenge* as authorization endpoint. The attributes of the digital identity of the user are not stored in the gateway. In contrast, the properties are temporarily retrieved during the authentication process. The characteristics are encoded into the SSI specific authentication challenge as requested claims.

	uPort	**Jolocom**	**OIDC**
Email	email	ProofOfEmailCredential	email
Name	name	ProofOfNameCredential	name
Firstname	firstname	ProofOfFirstnameCredential	given_name
Lastname	lastname	ProofOfLastnameCredential	family_name

TABLE I
OVERVIEW ABOUT CLAIM NAMING AND TRANSLATION

D. SSI Broker

The SSI Broker controls the communication with the implemented SSI solutions through a standard interface. The integration interface offers the following functionality.

- **Create Identity** The command creates an application identity for the gateway on the respective SSI solution. This digital identity is used by the gateway to issue and request verifiable claims. Additionally, it serves as a trusted communication partner for the SSI client.
- **Create Challenge** The function creates an authentication challenge for the user. The challenge contains a request for required attributes and a random number including a callback address. The user processes the challenge with its SSI client and sends a response back.
- **Verify Challenge** The function verifies the user's response to a previously generated challenge. The user is successfully authenticated if the verification succeeds.
- **Create Verifiable Claim** The command generates a verifiable claim for a certain attribute of a user's digital identity. The verifiable claim is issued under the digital identity of the gateway.

The implemented functions to create an authentication challenge is required to produce interaction patterns. Differences in these patterns between the SSI solutions indicate a missing adoption of standards. An interaction pattern is a JSON Web Token (JWT) that is exchanged between the gateway and the SSI client. We concentrate on a presentation of the authentication challenges for uPort (see Fig. 2) and Jolocom (see Fig. 3).


```
{
  "iss": "2oyjAieDKiMvL...VRd3AbEZU3QZN",
  "iat": "1548559818",
  "requested": ["email"],
  "verified": ["email"],
  "permissions": ["notifications"],
  "callback": "https://.../We333MaPRGli1k",
  "net": "0x4",
  "exp": "1548588618",
  "type": "shareReq"
}
```


Fig. 2. uPort challenge

Common factors of both challenges are default JWT attributes. These properties indicate the issuer (*iss*), the issuing time (*iat*) and an expiry time (*exp*). The JWT is valid between the mentioned timestamps. Considering the issuer attribute, the first difference becomes obvious. In the uPort challenge, the issuer attribute references the multi network identifier of the digital identity of the gateway. In contrast, the Jolocom JWT contains as issuer the DID of the digital identity including the reference to a specific key. Furthermore, either JWT has a property that indicates the request type for the SSI solution. Jolocom named the attribute *typ* and uPort refers to it by *type*.


```
{
  "interactionToken": {
    "credentialRequirements": [{
        "type": ["Credential",
          "ProofOfNameCredential"
        ],"constraints": []}],
    "callbackURL": "https://...BKkWFFACSKqW"},
  "typ": "credentialRequest",
  "iat": "1548581905743",
  "exp": "1548585505743",
  "iss": "did:jolo:3c79034...c6ba9#keys-1",
  "jti": "b53f28695fe0a"
}
```


Fig. 3. Jolocom challenge

Further attributes of the JWT are listed on the top-level by uPort. Jolocom creates a nested hierarchy with the name *interactionToken*. Both challenges list a callback address as a target for the response. uPort lists the network in the JWT to differentiate between the testing and production networks of Ethereum. Although Jolocom is also based on Ethereum, a network identifier is omitted.

The attributes *requested* and *verified* are used to obtain verifiable claims during the authentication process with the SSI client. Any self-attested claims are provided by the *requested* property. The *verified* keyword retrieves only claims with attestations. In contrast, Jolocom specifies the *credentialRequirements* tag to nominate verifiable claims. For all requests, the keyword *Credential* is supplied besides the actually requested claims. Additionally, constraints can be determined.

E. Trust Engine

The trust engine controls the validation of the verifiable claims of a digital identity upon an authentication request. The engine applies a predetermined trust model for evaluation. The implemented trust model reflects the opinion of the gateway's hosting entity towards the trustworthiness of one or several attestation issuers. Referring to a general trust model [13], we implemented an entirely simplified version that accepts verifiable claims that are issued by the digital identity of the gateway itself. These verifiable claims are considered trustworthy and will be provided as verified attributes within the flow of the OIDC protocol.

V. Evaluation

For evaluation, we conduct an user authentication at the tele-TASK portal. During the authentication process, all components of the integration architecture interoperate with each other. Upon successful authentication, the user is logged in at tele-TASK and required attributes are available. The gateway is hosted in the same trust boundary as tele-TASK.

tele-TASK is a web application that enables users to watch recorded videos. The attribute-based access model differentiates two categories of users. Users in the first category have extended access rights and are allowed to view additional recorded sessions. Whereas persons that belong to the second cluster have limited access to certain video streams. These two categories are distinguished by the email address of the user.

Certain prerequisites are required for executing a successful authentication process. The user has created a digital identity with uPort and obtained a verifiable claim for an email address from the gateway. The gateway considers its attestations as trustworthy. Furthermore, the tele-TASK portal is known by the gateway with a client identifier, a corresponding secret and a set of redirect Uniform Resource Identifiers (URI) to fulfil requirements of the OIDC protocol.

In the first step, the user opens tele-TASK and selects the option to authenticate via SSI solutions by using the gateway. Based on this request, tele-TASK redirects the user to the gateway. Fig. 4 outlines the redirection call. The target of the call is the relative path of the gateway that is used for SSI based authentication. Information about the actual authentication workflow is transmitted as parameters of the request. The scope argument contains *openid* to indicate the usage of the OIDC protocol. Additionally, the attribute *email* is requested. The value *email* refers to the respective attribute of the user. Furthermore, a redirection URI (*redirect_uri*) and a client identifier (*client_id*) are transmitted.

After redirection, the gateway requests the authentication of the user. The user is able to select its preferred SSI solution

```
https://gateway.local/blockchain/challenge?
scope=openid+email&
redirect_uri=https%3A%2F%tele-task.local&
client_id=v5Zd7isg8932ghjk&
response_type=code
```

Fig. 4. Redirect URI

that is either uPort or Jolocom. uPort is preselected and the respective authentication challenge is presented as QR code. The QR code contains an encoded JWT token that is signed by the identity of the gateway. It comprises the attribute email as requested information of the user's digital identity. The user scans the QR code with the uPort mobile app. uPort decodes the QR code and verifies the obtained JWT with regard to the signature originating from the identity of the gateway. Furthermore, uPort extracts the requested attributes from the JWT and prompts the user for consent to transmit the corresponding verifiable claims to the gateway. In this case, the attested claim of the email address is relayed. When the user confirmed to convey the data, the uPort mobile app creates a JWT that comprises the requested email attribute and sends the token to the callback address listed in the authentication challenge.

The gateway parses the information of either JWT and verifies the signatures. Furthermore, the received verifiable claim about the email address of the user is validated against revocation and expiry. Subsequently, the implemented trust model is applied to the verifiable claim of the email address. As the attestation issuer is the gateway itself, the provided email address is accepted as a verified attribute of the user's digital identity. Based on this information, the gateway generates the OIDC ID token. The ID token contains the attributes *email*, *ssi* and *sub* (see Fig. 5).

```
{
  "email": "max.mustermann@test.com",
  "ssi": "uport",
  "sub": "001c67fc2e3f91b...77f95ff77546"
}
```

Fig. 5. ID token

Finally, the user is forwarded to the tele-TASK portal along conveying the ID token. tele-TASK analyzes the ID token and extracts the email address. Having the email address as a trustworthy attribute of the user, tele-TASK is enabled to make the access decision.

VI. Discussion

A significant characteristic that differentiates self-sovereign identity solutions from traditional identity management approaches is the nature of decentralization. The identity provider is implemented on a blockchain and therefore, it does not represent a trusted third party anymore. Our implemented gateway supports on the one hand service provider by offering self-sovereign identity solutions to their users. On the other hand, it can introduce centralization to a certain extent. The service provider does not verify the digital identity and associated verifiable claims on the blockchain. In contrast, the service provider trusts the gateway for correct execution.

VII. Conclusion

The SSI model focuses on the users by placing them in full control of their digital identity and associated data. However, service provider adoption has been neglected due to a fragmented SSI landscape, non-usage of established identity and access management protocols and missing verifiable claim issuance facilities. We have proposed an SSI integration architecture for service providers to address these shortcomings. Furthermore, we have implemented a gateway that integrates to web applications via OIDC for authentication. The gateway mediates the communication and allows the user to authenticate with uPort or Jolocom by providing verifiable claims as attributes of the digital identity.

References

[1] C. Meinel, T. Gayvoronskaya, and M. Schnjakin, "Blockchain: Hype oder innovation," Hasso-Plattner-Institute, Prof.-Dr.-Helmert-Strae 2-3, 14482 Potsdam, Germany, 2018.

[2] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena. (2016) uport: A platform for self-sovereign identity. [Online]. Available: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf [Accessed: 2018-07-19]

[3] C. Allen. (2016) The path to self-sovereign identity. [Online]. Available: http://www.lifewithalacrity.com/previous/2016/04/the-path-to-self-sovereign-identity.html [Accessed: 2019-08-22]

[4] Unknown. (2018) Awesome decentralized identity. self-sovereign, blockchain and decentralized identity resources. [Online]. Available: https://github.com/infominer33/awesome-decentralized-id [Accessed: 2019-08-22]

[5] D. Reed, J. Law, and D. Hardman. (2016) The technical foundations of sovrin. a white paper from the sovrin foundation. [Online]. Available: https://www.evernym.com/wp-content/uploads/2017/07/The-Technical-Foundations-of-Sovrin.pdf [Accessed: 2019-08-22]

[6] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. [Online]. Available: https://pdfs.semanticscholar.org/ac15/ea808ef3b17ad754f91d3a00fedc8f96b929.pdf [Accessed: 2019-08-22]

[7] Jolocom. (2018) Jolocom whitepaper. self-sovereign and decentralised identity by design. [Online]. Available: https://github.com/jolocom/jolocom-lib/wiki/Jolocom-Whitepaper [Accessed: 2019-08-22]

[8] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proceedings of the 2016 USENIX Annual Technical Conference*. Usenix. The Advanced Computing Systems Association, 2016.

[9] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, and M. Sabadello. (2018) Decentralized identifiers (dids). data model and syntaxes for decentralized identifiers (dids). [Online]. Available: https://w3c-ccg.github.io/did-spec/ [Accessed: 2019-08-22]

[10] RWoT. (2018) Did auth: Scope, formats, and protocols. [Online]. Available: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2018/blob/master/topics-and-advance-readings/DID%20Auth [Accessed: 2019-08-22]

[11] OpenID Foundation. Openid connect core 1.0. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html [Accessed: 2019-08-22]

[12] Universal resolver. [Online]. Available: https://github.com/decentralized-identity/universal-resolver [Accessed: 2019-08-22]

[13] A. Grüner, A. Mühle, T. Gayvoronskaya, and C. Meinel, "A quantifiable trust model for blockchain-based identity management," in *Proceedings of the 2018 IEEE International Conference on Blockchain*, 2018.