SSIBAC: Self-Sovereign Identity Based Access Control

Rafael Belchior*, Benedikt Putz†, Guenther Pernul†, Miguel Correia*, André Vasconcelos* and Sérgio Guerreiro*
*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{rafael.belchior, miguel.p.correia, andre.vasconcelos, sergio.guerreiro}@tecnico.ulisboa.pt
†*Chair of Information Systems*, University of Regensburg, Germany
{benedikt.putz, guenther.pernul}@wiwi.uni-regensburg.de

***Abstract*—Ineffective data management practices pose serious issues to individuals and companies, e.g., risk of identity theft and online exposure. Self-sovereign identity (SSI) is a new identity management approach that ensures users have full control of their personal data. In this work, we alleviate data breach and user privacy problems by showing how SSI can fit within the context of established enterprise identity and access management technologies. In light of recent endeavors, we explore the use of decentralized identifiers, verifiable credentials, and blockchains that support SSI. We propose Self-Sovereign Identity Based Access Control (SSIBAC), an access control model for cross-organization identity management. SSIBAC leverages conventional access control models and blockchain technology to provide decentralized authentication, followed by centralized authorization. The access control process does not require storing user sensitive data. A prototype was implemented and evaluated, processing 55,000 access control requests per second with a latency of 3 seconds.**

***Index Terms*—self sovereign identity, decentralized identity, authorization, attribute-based authorization, access control**

I. Introduction

Centralized *access control* (AC) systems face several challenges and risks [1], [2]: cumbersome policy management, lack of flexibility of setup and configuration, ineffective policy enforcement, risk of privacy leakage, and availability (single point of failure). These translate into issues of *authentication*, *authorization*, and *accountability* (AAA). Some authors argue that these challenges, allied to dynamic threats such as the landscape complexity and the lack of collaborative tools, yield conventional *access control models* (ACM) insufficient to respond to today's enterprise needs [1].

Companies collect user data to perform access control and other processes (e.g., user profiling). Data collection exposes users to data breaches and to abusive data analysis, both cases of privacy violation. We can define *privacy* as the ability of users to selectively disclosure their information. Data breaches are common due to, for example, poor security practices and ineffective personal data management processes. Examples can be found in enterprises across all industries and have generated great financial damage [3].

To protect individuals, several regulations have been proposed, such as the *General Data Protection Regulation* (GDPR) [4] and the California Consumer Privacy Act (CCPA) [5]. These regulations protect the personal data of natural persons and pose obligations to the entities holding their data (controllers) and those processing it (processors). Companies that do not respect these regulations are subject to fines, in case there is mishandling of personal data.[1] Despite regulatory efforts, data breaches still occur.

To empower the user with control over of his data, while proving dynamic, trustable, and decentralized ACMs, we refer to the concept of *Self-Sovereign Identity* (SSI) [6]. SSI is a good match to the blockchain promise of decentralization [7], [8]. In SSI, the user stores identity data and decides which data to disclose. Unlike existing schemes such as OpenID Connect (OIDC) [9], Shibboleth [10], and Microsoft Passport [11], there is no need to entrust an intermediary identity provider with storing identity data [12]. SSI can alleviate the impact of data breaches and provide the user with flexibility managing the identity: instead of spreading data and information among different service providers, the user has full control of his personal data and discloses only required information. By using *zero-knowledge proofs* (ZKPs), SSI allows satisfying predicates based on user data without revealing that data [13]. This provides privacy for access control processes, where a user needs to satisfy a certain predicate to access resources. SSI also allows a single identity to be linked to sets of attributes emitted by different organizations. Therefore, it fosters interoperability across administrative domains and applications [14].

Blockchain is a suitable technology to support SSI, as it is decentralized and supports peer-to-peer interaction [7]. Furthermore, it can be used to obtain a reliable infrastructure for decentralized access control, mitigating some of its traditional problems, such as the lack of adaptability to dynamic environments [1]. Although the use of a replicated immutable appendable log could raise concerns regarding the GDPR, SSI allows technical privacy protection, achieving GDPR compliance [7]. In particular, SSI does not compromise GDPR's view on the right of users to rectify and remove data and promote the identification and regulation of data processors. Conversely, the application of SSI to the access control process can also protect users' privacy.

SSI provides a model for *authentication and issuing credentials*. However, a structured approach to use it for *authorization and access control* is still missing, arguably due to its novelty. To fill this gap, we leverage three emerging technologies:

[1] https://www.enforcementtracker.com/insights

blockchain, *decentralized identifiers* (DIDs) [15], and *verifiable credentials* (VCs) [16]. We define a novel ACM, referred to as *Self-Sovereign Identity Based Access Control* (SSIBAC), that can be more appropriate to today's enterprise needs in terms of data privacy and security. We show how DIDs and VCs can be integrated with attribute-based access control in a federated setting, minimizing data disclosure and data redundancy. For transparency and accountability regarding access requests, VCs can be used with blockchain-based ACMs.

We go beyond existing work on attribute-based access control by ensuring user privacy. While privacy has been a concern for ABAC models [17], existing ABAC models still store all identity data with a single identity provider. With SSIBAC, selective disclosure of attributes and range proofs for numerical values ensure that data is only disclosed on a need-to-know basis.

In summary, the major contributions of this paper are:

- a novel SSI-based ACM called SSIBAC, with a focus on data privacy and sovereignty;
- an implementation of SSIBAC, relying on an attribute-based model;
- an evaluation of the implementation, providing insights on the bottlenecks of the solution and future research directions; the prototype processed 55,000 access control requests per second with a latency of 3 seconds.

The paper is structured as follows. In Section II, we provide preliminaries. We formally define the SSIBAC model in Section III, followed by an instance of such model using ABAC in Section IV. We report and discuss evaluation results in Section V. Section VI highlights related work. Finally, in Section VII we conclude the paper.

II. Preliminaries

This section introduces concepts regarding self-sovereign identity and access control.

A. Centralized and Federated Identity

Enterprise identity systems typically focus on roles or attributes associated with each user, enabling the execution of their duties. In enterprise identity and access management (IAM) the friction caused by centralized systems is most apparent in federation scenarios, where external users have to be granted access to internal systems. An example is *Eduroam*, a federation of educational organizations that provide internet access to each others users [18]. Traditionally, this is achieved by requesting data through identity federation systems [17]. However, identity federation systems are not interoperable among different standards and bridges are required to interact across federations [19].

B. Decentralized Identifiers

The SSI concept allows a user – individual, organization, or "thing" (e.g., a device or a computer program representing a process) – to present its credentials to a third party without intermediaries. This process is enabled by DIDs, a concept defined by the W3C [15]. A DID represents an identity and allows trustable interactions, rooted on a *verifiable registry* (e.g., a blockchain), and public-key cryptography [20]. DIDs are controlled by *DID subjects*. A DID resolves to a DID document with metadata, which also provides the means for authenticating the DID subject. The DID subject can prove the ownership of a DID through a private key associated with a DID's public key. A DID can be defined as a three-part string representing the format `did:<method>:<identifier>`, where `<method>` represents the DID method (the specification for a specific type of DID) that the `<identifier>` uses [15]. To facilitate the management of DIDs, one can leverage *user agents* (or simply agents), i.e., software processes acting on behalf of a DID subject [13].

C. Verifiable Credentials

A *verifiable credential* (VC) provides a standard way to digitally express credentials in a way that is cryptographically secure, privacy-respecting, and machine-verifiable [13], [16]. An entity called *issuer* generates and signs such credentials with its private key: this enables a third-party to verify the issuer of a VC (the DID of the issuer is typically associated with the credential). A *verifier* can look up the public key of a given DID, associated with a given credential on a verifiable data registry (e.g., a public blockchain).

For example, the VON ledger[2] is a Hyperledger Indy-based blockchain storing public DID documents, *credential definitions* (representing the schema of a VC, i.e., the attributes the VC should hold) and revocation registries (repositories containing information about revoked credentials). *Credential schemas* allow a verifier to check the *claims* against a vocabulary of admissible claims. A claim is an assertion on a subject. For example, an issuer defines a set of possible attributes in a schema that may later be issued in VCs associated with that schema. A VC, therefore, consists of claims made about a subject by an issuer. The subject and issuer are represented by unique identifiers, which we assume to be DIDs for our purposes. A VC is trusted as long as its issuer is trusted.

At verification time, the holder of a verifiable credential (often the subject itself) creates a *verifiable presentation* (VP), which contains metadata and proofs for a subset of the contained claims. The VP creation process might be required by a verifier, through a *verifiable presentation request* (VPR). The VP is sent to a verifier, that confirms the VC held by the subject satisfies a specific predicate. The VP can be issued using ZKPs, "containing derived data instead of directly embedded verifiable credentials" [16]. For simplicity, we deem that the result of a generated VP can be true or false, if the predicate is satisfied or not, respectively.

D. Access Control

Access control systems provide selective access to a set of resources, under a specific set of conditions. Common ACMs include *Role-Based Access Control* (RBAC) [21] and *Attribute-Based Access Control* (ABAC) [22], besides many others, e.g.,

[2]https://vonx.io/

the classical Access Control Matrix, Access Control Lists, Capabilities, Mandatory Access Control, and Discretionary Access Control [23].

In ABAC, a commonly used ACM, access rights are granted based on attributes, i.e., the attributes the subject holds and the attributes expressing the environmental context. According to the XACML specification [24], which is suitable to implement an ABAC system, several components are cooperating in the access control process. The *subject* (that we also call user) is the entity that requires access to a resource. A *client* is a device that requests access to a resource on behalf of a subject. A *Policy Enforcement Point* (PEP) intercepts access requests from a user, redirecting them to the *Policy Decision Point* (PDP), and enforcing its AC decision. The PDP is the component that computes the result of an access control request (ALLOW or DENY), using an access control policy and information stored on the *Policy Information Point* (PIP). The PIP contains information about the subject's attributes. The *Policy Retrieval Point* (PRP) stores and retrieves access control policies, which are managed by the *Policy Administration Point* (PAP). Although the literature separates the attribute storage (PIP) from the access control policy storage (PRP), we refer to them as the same entity, for brevity. Moreover, *accountability* is achieved by tracking the access control requests issued by the subject, and the corresponding access control decision calculated by the PDP. This process allows the system to establish a history of access to resources.

According to Sandhu et al., an ACM should satisfy the following principles [25]:

Principle 1. *Least privilege: Only those permissions required for the tasks performed by the user in the role are assigned to the role.*

Principle 2. *Separation of duties: Invocation of mutually exclusive roles can be required to complete a sensitive task, such as requiring an accounting clerk and an account manager to participate in issuing a check.*

We define a principle related to the Principle of Least Privilege, that aims to alleviate the impact of data breaches:

Principle 3. *Context-Based Privilege: Only the strictly required information for computing an access control decision should be stored and processed.*

III. SSIBAC

This section presents the SSIBAC access control model. The SSIBAC model is an evolution of classical ACMs that integrates the concept of SSI and mechanisms that implement it with blockchain. The major idea of SSIBAC is to map VCs (encoded into VPRs, and their responses, VPs) to access control policies, stored at the PRP, that are parsed by an underlying ACM, in order to achieve context-based privilege, and thus data privacy and sovereignty.

SSIBAC abstracts previous models and can be instantiated using one of those models, e.g., RBAC or ABAC. This means that a particular instantiation of SSIBAC reuses concepts and mechanisms of the underlying model. SSIBAC regulates the access of subjects to resources by evaluating access control rules against *permission validators*. Permission validators allow mapping VPs to attributes, roles, or other abstractions of data. For instance, if SSIBAC is instantiated with RBAC, the permission validator is the role, whereas if SSIBAC is instantiated with ABAC, the permission validator is the set of subject and contextual attributes. A user is uniquely identified by a DID (although a user can hold multiple DIDs), and has a set of VCs, issued by *issuers*. An issuer is a trusted entity that issues VCs.

A permission validator, along with an access control request, allows the PDP to calculate an access control decision. We consider a function ψ that maps VCs to permission validators, depending on the input. For example, $\psi_{(i,ATTRIBUTE)}$ maps all the user verifiable credentials from user$_i$ to attributes that can be used by an ABAC system. Conversely, $\psi_{(i,ROLE)}$ maps the VCs from $user_i$ to roles, which can be evaluated by an RBAC system. In practise, the initialization of ψ depends on the underlying access control system to be used, and its mapping is trivial. We, therefore, establish the bridge between DIDs, VCs, and the permission validators of ACMs, by saying "this VC corresponds to an attribute/role defined in a specific AC policy". This function can be considered the component that facilitates interoperability among ACMs, similarly to meta-access control models [26].

We define a function χ that maps access control policies to VPs. This function bridges the access control policies used by conventional ACMs with peer-to-peer interactions supported by a trusted data verifier. Function χ can be defined in 1) an ad-hoc way, 2) automated by parsing the schema fields and creating a VP containing the same fields, plus a condition defined on the access control policy. Verifiable presentations are then tied to an access control request, requested by verifiers. Verifiers can also be providers of resources (i.e., the entity processing the access control request is the same that holds and delivers the resource).

The infrastructure supporting the issuing of DIDs, VCs, and VPs is a verifiable data registry (in our specific case, a verifiable credential registry). This data registry can be decentralized, e.g., a blockchain. For instance, a blockchain can record the schema of a verifiable credential, along with its issuer: this allows peer-to-peer validation of VCs without resorting to the issuer.

We define our model rigorously as follows:

SSIBAC components

- a set of users $\mathcal{U} = \{u_1, u_2, ...\}$. Each $user_i$ is identified by a DID and holds a public/private key pair (K_p^i, K_s^i) associated to that DID and a set of VCs $\mathcal{L}_i = \{l_1^i, l_2^i, ...\}$;
- a set of resources $\mathcal{R} = \{r_1, r_2, ...\}$;
- a set of issuers $\mathcal{I} = \{i_1, i_2, ...\}$ that issue VCs for users;
- a set of verifiers $\mathcal{V} = \{v_1, v_2, ...\}$ who request VPs and mediate the access control flow. Typically, they are also resource providers;
- a set of permission validators $\mathcal{P} = \{p_1, p_2, ...\}$;

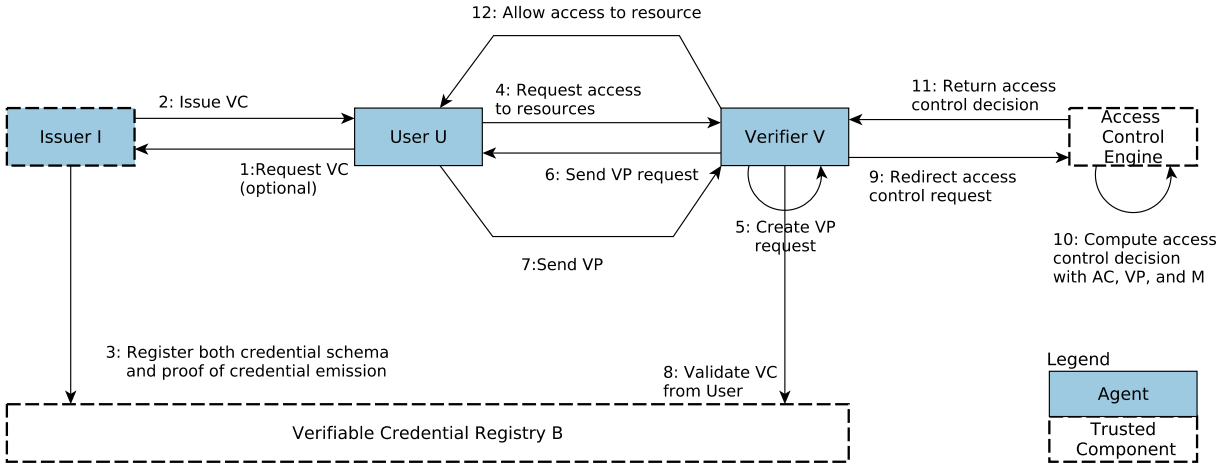

Fig. 1: Access control flow enforced by the SSIBAC model

- a set of injective functions $\psi = \{\psi_1, \psi_2, ...\}$, such that $\psi_i : \mathcal{L}_i \rightarrow \mathcal{P}_k$, i.e., function ψ_i maps the VCs from $user_i$ to permission validator P_k;
- an injective function $\chi : \mathcal{AC} \rightarrow \mathcal{VP}_R$, mapping access control policies to VPRs.

Besides the core components, SSIBAC has several input parameters, which can be set before instantiation or computed at run-time.

SSIBAC parameters

- a set of supporting ACMs $\mathcal{M}$;
- a set of access control policies $\mathcal{AC}$, representing the rules of a particular business context;
- a set of VPs $\mathcal{VP}$, translated from access control policies;
- a Verifiable Data Registry $\mathcal{B}$, the trust anchor for the peer-to-peer interactions (allows checking the validity of DIDs, VCs, and VPs).

Figure 1 illustrates the access control flow enforced by our model. A user is issued verifiable credentials (steps 1 and 2), which are rooted in a verifiable credential registry (3). The user then requests access to a set of resources (4). The verifier creates a VPR from the access control policy underlying the requested resource. This access control policy may be collected from a trusted PRP (5) and sends it to the user (6). The verifier assumes that the user owns the necessary attributes on the verifiable credentials to be able to respond to the challenge. After the challenge is sent in the form of a VP (7) and validated (8), the verifier gives as input the result of the validation process to an access control engine (or PDP) (9, 10). The result of the decision may be influenced by extra factors, e..g. the context of the request. If the decision from the access control engine is ALLOW, access to the resource is provided (11, 12).

IV. Authentication and Access Control with SSIBAC and ABAC

In this section, we describe *SSIBAC instantiated with the ABAC model*. We chose ABAC because it is much adopted and provides fine-grained and flexible access control [27].

A. System Description and Assumptions

To integrate decentralized identity with attribute-based access control, attributes need to be issued to a specific DID. This can be accomplished using VCs [13].

We instantiate the SSIBAC model with $\mathcal{M} = \{ABAC\}$, a user u_1, an issuer i_1, a verifier v_1, one resource r_1, a public blockchain $\mathcal{B}$. The permission validator p_1 is the attribute from ABAC ($\psi_1 : \mathcal{L}_1 \rightarrow \{p_1\}$) . In other words, our access control engine will calculate an access control decision based on ABAC/XACML access control policies, so we need VPs to encode user attributes. Let $\mathcal{L}_1$ represent the subset of verifiable credentials held by user u_1, and let $\Lambda_1 = \{\lambda_1, ..., \lambda_i\}$ be a subset of attributes derived from $\mathcal{L}_1$.

Function χ maps a verifier's access control policy AC_1, containing the rules to access $r1$, to a VPR, by parsing the schema fields from the VC(s), as well as the access control policy, and the necessary conditions for an ALLOW decision. The VPR is issued by v_1, while the corresponding VP, VP_1 generated by u_1. Access to a certain resource is granted given that v_1 returns an ALLOW decision, under the condition that the result of the VPR, VP_1, is true. In other words, the AC_1 encoded by VPR, and evaluated by VP_1, is satisfied.

The access control decision could be comprised of a more complex policy, e.g., the result of VP_1 and a set of contextual conditions, such as the day of the week. For this, the PIP could be hybrid: sensitive data is owned by the subject, whereas general information used to identify him is also saved on a local database. User attributes are mapped to the verifiable credentials emitted to a specific DID, i.e., the subject holding ownership of the DID, with a specific schema. The access control policies are mapped to VPRs made on-chain. By doing so, we allow users to keep their information private, as verifiable presentations can handle selective disclosure based on zero-knowledge proofs [13].

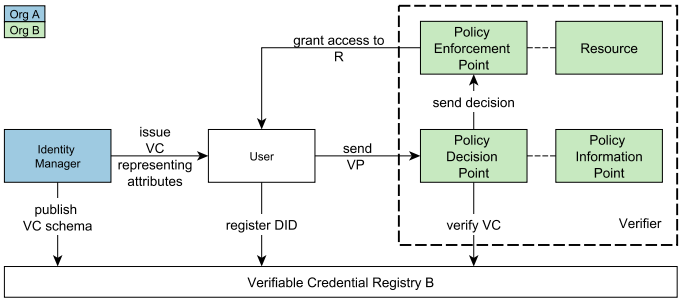

Fig. 2: SSIBAC in a multi-organizational setting, in light of the XACML standard perspective

Figure 2 shows how ABAC components are integrated within SSIBAC. One can observe that the verifier acts as both PEP and PAP. It allows or denies access to resources through access control policies, and administrates such policies. The access control engine contains a PDP that can be embedded in the verifier, or be an external component. We opted for a centralized PDP, although decentralized ones are possible and have been implemented [27].

We remark that a decentralized PIP may be used. Sensitive information for enabling access control decisions can be held by the subject. A combination of a subject-owned PIP and a traditional, local PIP can be useful: sensitive attributes can be kept private by the subject, while other attributes such as the user ID are stored by the organization.

B. Threat Model and Security Requirements

Regarding SSIBAC's threat model, we assume an honest-but-curious verifier. This means that the verifier performs the access control decision honestly, but may try to learn about the users' attributes. Since verifiable presentations are supported by ZKPs, the verifier will, very likely, obtain incomplete information – selective disclosure is achieved. However, selective disclosure is usually not enough, as organizations can collude to cooperatively infer information about the user. Thus, *unlinkability* [28] is also desired. Our model achieves unlinkability given that a person utilizes a DID for each specific purpose.

The security requirements are threefold:

1) *Selective choice of participants*: only users holding the VCs which map to the permission validators required in an access control policy can access the resources specified on the same access control policy.
2) *Data confidentiality*: recalling Principle 3, the access control engine should perform decisions based on the least information possible. The ZKPs allows a user to disclose as least information as possible.
3) *Accountability and non-repudiation*: issuers are held accountable for the VCs they issue. User credentials are auditable, as the blockchain provides the trust anchor for checking its validity. In other words, a verifier can verify that the presented credentials are valid and come from a trusted party, at its description. We provide a trade-off between privacy and accountability as the interactions between DIDs are peer-to-peer and thus not necessarily recorded; unlinkability is established if a DID interacts does not interact with several parties, disclosing (part) of their VCs.

V. Evaluation

In this section, we evaluate an instance of SSIBAC based on the real-world use case scenario from the European Commission (EC) project QualiChain.[3]

A. Use Case: Decentralised Qualifications

The QualiChain project aims to propose a blockchain-based approach for disrupting the archiving, management, and verification of educational and employment qualifications. In particular, QualiChain will support the storage, sharing, and verification of academic and other qualifications along with several additional services, provided by the platform. To comply with GDPR legislation, and protect its users' privacy and data, a non-intrusive access control mechanism has to be deployed. In particular, QualiChain aims to follow the principle of *context-based privilege*, in which only the strictly necessary data to provide a service is requested from the diploma holder.

This project has several stakeholders:

- *certification seekers*, e.g., graduated students. They are referred to as diploma holders upon receiving a verifiable credential for their diploma;
- *certification providers*, e.g., higher education institutes;
- *certification validators*, e.g., potential employers.

Universities issue verifiable credentials for students, which can be used to authenticate on the QualiChain platform. It is desirable to use SSI-based access control in this scenario so that QualiChain does not need to store any personal data: access to services is provided on-demand, based on the verifiable proofs that the student provides.

SSIBAC can be useful for access control in QualiChain. We focus on granting a diploma holder access to a service provided by QualiChain. Figure 3 illustrates this process.

We instantiate our model defined in Section IV with $u_1 = Alice$, $i_1 = IST$, $v_1 = QualiChain$, and $r_1 = JobOffers$, a service provided by the QualiChain Platform. In this use case, a recent graduate, Alice, requires a university diploma in the form of a VC from a higher education institution, IST (step 1). The university issues a VC to Alice, and publishes the corresponding proof on a decentralized ledger, in our case the Sovrin blockchain based on Hyperledger Indy (steps 2 and 3). Listing 1 depicts the *credentialSubject* schema issued for the student. Alice now becomes a diploma holder.

Upon accessing the platform, which may require Alice a VP certifying she owns a non-revoked VC issued by IST, Alice can have access to several services. We consider the job offer service, enabling PhD diploma holders to find research

[3]https://qualichain-project.eu/

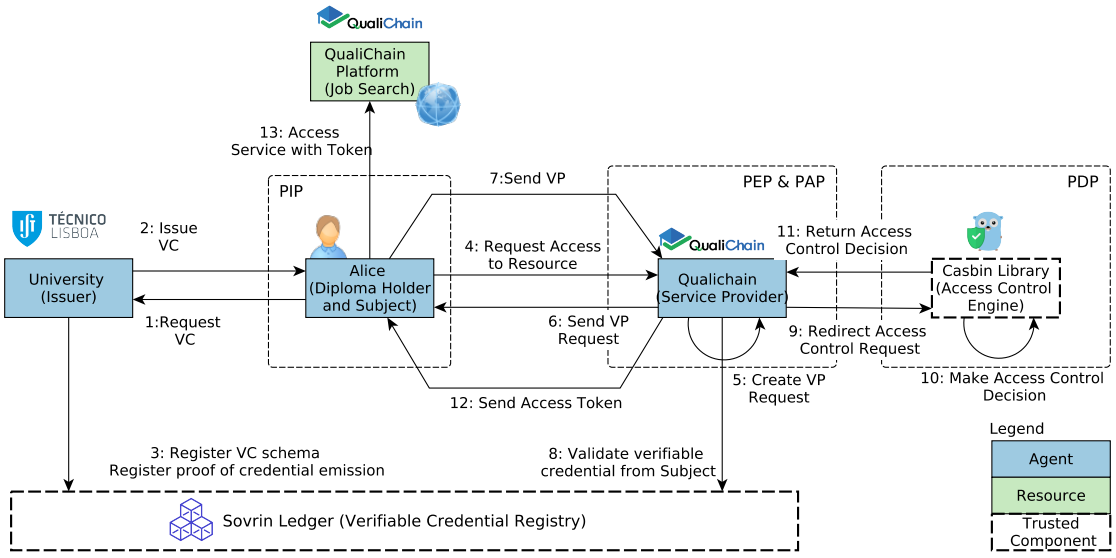

Fig. 3: SSI-based ACM applied to the QualiChain scenario

```
"firstName": "Alice",
"lastName": "Anderson",
"age": 25,
"id": "1234",
"timestamp": 1590092610,
"degree": {
    "university": "IST",
    "type": "BachelorDegree",
    "name": "Bachelor of Science", "EQF": "6",
    "course": "Computer Science",
    "grade": "4",
    "gradeScale": "0-4",
    "skills": "[]",
    "degreeId": "80970"
},
"metadata": [...],
"proof": [...]
```

Listing 1: High-level example of a verifiable credential, issued for Alice, $\mathcal{L}_{Alice}$. Λ_{Alice} = *{firstName, LastName, ... , degreeId}*, $p1$ = attribute (VC mapped to attributes by χ), χ parses the VC fields such that it outputs a VP containing a challenge invoking a subset of Λ_{Alice}

positions, available for people with a European Qualification Framework[4] level higher than 6 (MSc and PhD graduates).

Alice would like to access the service that allows searching for job offers on research positions (step 4). In order to provide her access to that service (a resource), QualiChain creates a VPR and sends it to Alice (steps 5 and 6). The VP encodes the access control policy for accessing the researcher position service: if the "EQF" field of a diploma holder is higher than 6, access to the service is provided. Alice constructs and provides the corresponding ZKP (step 7), and the QualiChain agent verifies that such VP is true (step 8). After validating that the proof comes from Alice, it redirects the result to the access control engine (step 9), which later returns the processing outcome (steps 10 and 11). The access control policy states that if the result from the VP is true, then access is granted. However, additional checks could be performed (e.g., the access control engine could verify if Alice had already accepted another job offer). If the verifiable presentation is valid, it means that it satisfies the encoded access control policy sent on step 6. As Alice's EQF is not higher than 6, Alice cannot access the desired service (steps 12 and 13 do not take place).

B. Implementation

We now describe the implementation of the SSIBAC prototype and the experimental setting.

As our Verifiable Credential Registry $\mathcal{B}$, we chose Hyperledger Indy [29]. Hyperledger Indy is a state-of-the-art public blockchain that provides "tools, libraries, and reusable components for providing digital identities". The Hyperledger Aries project [30] was leveraged to create agents (representations of users) that manage their wallets and perform operations on the distributed ledger. Aries serves as the infrastructure for "blockchain-rooted, peer-to-peer interactions". In other words, we mediate communication between agents and the supporting blockchain through Aries. We based our implementation on the demo provided by Hyperledger Aries.[5]

We ran our experiments on the GreenLight Dev Ledger[6] provided by the VON blockchain test net. We leveraged Google Cloud Platform as our infrastructure. A c2-standard-8

[4]https://www.cedefop.europa.eu/en/events-and-projects/projects/european-qualifications-framework-eqf

[5]https://github.com/hyperledger/aries-cloudagent-python

[6]http://dev.greenlight.bcovrin.vonx.io/

(8 vCPUs, 32 GB memory) virtual machine running Ubuntu 20.04 was used. After the appropriate setup, we deployed three Docker containers, each representing an agent: Alice, IST, and QualiChain. A fourth agent was deployed to aid the evaluation process, by collecting information on the performance of the process. We executed each experiment 100 times and discarded the first and the last 10 to avoid outliers. In total, we executed 400 experiments.

C. End-to-End Latency

First, we measure the end-to-end latency of the process. For that, we divide our process into three phases: startup, connect, and access control.

The *Startup* phase comprises the time to set up the necessary infrastructure for conducting access control based on decentralized identity. In particular, this phase includes the time the system needs to register the agents' DIDs into the blockchain, the time to initialize the four agents (Alice, IST, QualiChain, and Performance), and the time for the IST agent to publish the schema of a university degree.

The *Connect* phase connects the agents, exchanges verifiable credentials, and prepares the environment for the access control phase. Alice connects to IST, IST issues a variable number of verifiable credentials to Alice, and after that posts a corresponding proof on the blockchain. Next, Alice connects to the QualiChain agent.

In the *Access Control* phase, Alice requests a resource from QualiChain. QualiChain requests a verifiable presentation (output of χ) that contains the necessary permission validators (attributes Λ_1, according to ψ) in order to conduct the access control process (conducted by ABAC). Alice constructs the proof and sends it to QualiChain. QualiChain then handles the proof, confirms its validity, and conducts the access control process.

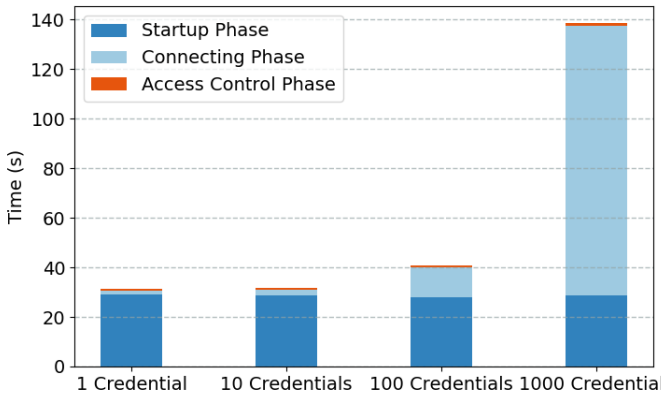

Fig. 4: Latency depending on the number of emitted credentials

Figure 4 depicts the cumulative time necessary to conduct each phase, in seconds, as a function of the number of credentials emitted to Alice. The *Startup* phase takes 29.1, 28.5, 28.0, and 28.7 seconds if the number of verifiable credentials issued was 1, 10, 100, or 1000, respectively. Conversely, the

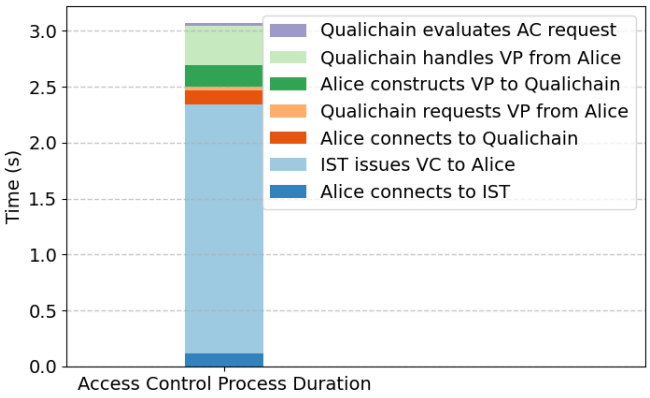

Fig. 5: Duration of the various steps in the *Connecting* and *access control* phases (startup phase omitted), with 10 issued credentials

Connect phase took 1.3, 2.5, 12.0, and 109.0 seconds. A linear regression for these results yields the function $t(n) = 1.283 + 0.108n$, where $t(n)$ is the *Connect* time and n the number of credentials. The access control phase took 0.9 seconds regardless of the number of credentials. Table I depicts the latency associated with each phase of the access control process, as a function of the number of credentials issued. One can observe that the average duration of the *Startup* and *Access Control* phases is practically constant. The constant startup phase duration is expected, as no step of this phase depends on the number of issued credentials.

The *Startup* phase is a major bottleneck, accounting for most of the duration of the overall process (92.9%, 89.3%, 68.4%, and 21.9% for 1, 10, 100 and 1000 issued credentials respectively). For a system issuing a large number of VCs, the *Connect* phase is the bottleneck. As shown before, the system scales linearly with the number of credentials emitted. As credentials will not be emitted regularly, the *Connect* phase duration can be significantly reduced. Figure 5 represents the sub-phase duration for the *Connect* and *Access Control* phases. The overall process takes 3.39 seconds (considering 10 issued credentials), being the credential issuing step responsible for 66% of the connecting phase and access control phases together (2.23 out of 3.39 seconds).

For a single credential, the first access control request can take up to around 31.4 seconds (*Startup*, *Connecting*, and *Access Control* phases) or 2.23 seconds (*Connecting* and *Access Control* phases) to be served. In particular, the *Startup* and *Connecting* phases require setting up infrastructure and peer-to-peer connections specific to our experimental setting, which explains high latencies. In practice, the full process of credential issuance and a subsequent access control request should normally take around 2.23 seconds. In the QualiChain scenario, the attributes do not vary frequently, so we deem this latency suitable.

TABLE I: Evaluation of latency as a function of the number of credentials

# Credentials \ Process Phase	Startup		Connecting		Access Control		Total Time
	μ	σ^2	μ	σ^2	μ	σ^2	
1 Credential	29.14	5.5	1.30	0.03	0.92	1.05	31.36
10 Credentials	28.53	5.24	2.47	0.09	0.92	1.05	31.92
100 Credentials	28.00	4.85	11.97	0.33	0.92	1.05	40.89
1000 Credentials	28.68	4.84	108.93	0.92	0.92	1.05	130.53

D. Throughput

The throughput in terms of access control requests per second is associated with the latency of the previous phases. Credentials are issued during the *Connecting* phase, with about 9.3 credentials issued per second for 1000 credentials (see Table I). This limit is due to sequential processing in our demo application, since all credentials are sequentially created, signed and submitted. The throughput performance is also tied to the hardware in which the experiment is running and the throughput of the Hyperledger Indy consensus algorithm.

The time for evaluating the access control policy is negligible, as we achieve around 55,000 access control evaluations per second considering only the *Access Control Phase*. Considering that the necessary credential for the verifiable presentation has been emitted and belongs to the subject, the time needed for processing each access control decision is 0.9 seconds.

E. Revocation

Credential revocation is an important concern for access control systems. If a credential is revoked (e.g., the university revokes Alice's diploma), the verification of the presentation by the verifier will fail. Thus, when Alice attempts to access a resource using her revoked VC, access will be denied. This does not incur a performance degradation of our system, as the process is the same with a valid credential, and revoking a credential only costs a transaction.

VI. Related Work

Ferdous et al. formalize the concept of SSI and show how it can be integrated with the lifecycle of an IDMS [31]. They envision blockchain-based smart contracts to provide a decentralized environment where SSI can be successfully applied. However, no solution is implemented, and it is not clear how SSI can be applied to access control. Coelho et al. detail a self-sovereign identity system [32] using roles, attributes, policies, and user wallets. The proposed system does not support multiple ACMs, and lacks an implementation.

Several authors provide an overview of SSI, but do not elaborate on any implementation or real-world applications [33], [34]. Others present SSI solutions for several domains such as healthcare [35], [36], or general identity management systems [8], [37]. However, either access control is not the focus of the paper, authors do not provide an implementation of such a system, or validation through a real-world scenario is missing.

In recent years several authors proposed a variety of blockchain-based access control schemes. Maesa et al. proposed a general blockchain-based access control system using ABAC, XACML-coded policies, and the Ethereum blockchain [38]. There are also attempts to decentralize the access control decision engine, therefore providing support for dynamic access control policies, as well as distributed policy decision engines [27]. Despite authorization being decentralized, data sovereignty is not achieved (authentication is partially centralized).

On the contrary of most of these works, our approach provides access control based on decentralized authentication, leaving the user in control of identity data. Moreover, our work goes beyond existing work by providing a formalization of an ACM following the concepts of SSI, while also implementing it as part of an education ecosystem use case. Finally, we provide not only flexibility at the ACM level but also portability regarding the user identity.

VII. Conclusion

In this paper, we propose the Self-Sovereign Identity Based Access Control (SSIBAC), the first approach to access control based on decentralized identity. We explore this topic by instantiating our SSIBAC model with attribute-based access control, which is applied to a real-world case, the EU QualiChain project. Our implementation assures that the context-based privilege is achieved, by promoting peer-to-peer interactions, providing the basis for the access control process. Our experimental evaluation shows that each access control request can be served in around 0.9 seconds. Although more time-consuming than traditional centralized access control systems, access control based on self-sovereign identity can alleviate the data privacy problem, which we consider an acceptable trade-off for applications not requiring high throughput.

Our work has some limitations. The trust of the system is strongly tied to a single access control engine, so verifiers are single points of failure. We plan to address these limitations in future work, e.g., by decentralizing the access control engine and conducting the authorization process in a distributed and decentralized way (for example using a smart contract). This provides accountability and non-repudiation with regard to verifiers. An additional future work direction is to explore cross-chain authorization, leveraging recent blockchain interoperability techniques. This would allow us to provide decentralized authorization by binding verifiable presentations to jointly agreed access control policies.

Acknowledgment

We thank the reviewers that provided suggestions to improve this paper. This work was supported by the European Commission program H2020 under the grant agreement 822404 (project QualiChain) and by national funds through Fundação para a Ciência e a Tecnologia

(FCT) with reference UIDB/50021/2020 (INESC-ID). The authors thank Google for the Google Research Program grant that supported this research.

REFERENCES

[1] H. Martins and S. Guerreiro, "Access Control Challenges in Enterprise Ecosystems: Blockchain-Based Technologies as an Opportunity for Enhanced Access Control," in *Global Cyber Security Labor Shortage and International Business Risk*. IGI Global, 2018, vol. i.

[2] D. Maesa, P. Mori, and L. Ricci, "A blockchain based approach for the definition of auditable access control systems," *Computers & Security*, vol. 84, pp. 93–119, 2019.

[3] Verizon, "2020 Data Breach Investigations Report," Verizon, Tech. Rep., 2020. [Online]. Available: https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf

[4] European Parliament and European Council, "Regulation 2016/679 of the European Parlament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing - Directive 95/46/EC," 2016.

[5] C. Barrett, "Are the EU GDPR and the California CCPA becoming the de facto global standards for data privacy and protection?" *Scitech Lawyer*, vol. 15, no. 3, pp. 24–29, 2019.

[6] C. Allen, "The path to self-sovereign identity," *Life with Alacrity*, 2016.

[7] G. Kondova and J. Erbguth, "Self-sovereign identity on public blockchains and the GDPR," *Proceedings of the ACM Symposium on Applied Computing*, pp. 342–345, 2020.

[8] F. Wang and P. De Filippi, "Self-Sovereign Identity in a Globalized World: Credentials-Based Identity Systems as a Driver for Economic Inclusion," *Frontiers in Blockchain*, vol. 2, p. 28, 1 2020.

[9] N. Sakimura, NRI, J. Bradley, Ping Identity, M. Jones, Microsoft, B. Medeiros, Google, C. Mortimore, and Salesforce, "OpenID Connect Core 1.0 incorporating errata set 1," 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html

[10] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, "Federated security: The Shibboleth approach," *Educause Quarterly*, vol. 27, no. 4, pp. 12–17, 2004.

[11] R. Oppliger, "Microsoft .NET Passport and identity management," *Information Security Technical Report*, vol. 9, no. 1, pp. 26–34, 2004.

[12] C. Schläger, M. Sojer, B. Muschall, and G. Pernul, "Attribute-based authentication and authorisation infrastructures for e-commerce providers," in *Lecture Notes in Computer Science*, 2006.

[13] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model 1.0: Expressing verifiable information on the Web (W3C Recommendation)," 2020. [Online]. Available: https://w3c.github.io/vc-data-model/

[14] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," *arXiv*, 2020. [Online]. Available: http://arxiv.org/abs/2005.14282

[15] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized Identifiers (DIDs) v1.0: Core architecture, data model, and representations - W3C Working Draft 23 July 2020," 2020. [Online]. Available: https://w3c.github.io/did-core/

[16] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model 1.0," 2020. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[17] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control ABAC definition and considerations," *NIST Special Publication*, 2014.

[18] L. Florio and K. Wierenga, "Eduroam, providing mobility for roaming users," in *Proceedings of the EUNIS 2005 Conference, Manchester*, 2005.

[19] H. L'Amrani, B. E. Berroukech, Y. El Bouzekri El Idrissi, and R. Ajhoun, "Toward interoperability approach between federated systems," in *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, 2017.

[20] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[21] D. Ferraiolo and R. Kuhn, "Role-Based Access Control," in *In 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.

[22] V. Hu, D. Ferraiolo, R. Chandramouli, and R. Kuhn, *Attribute-Based Access Control*. Artech House, 2017.

[23] R. S. Sandhu and P. Samarati, "Access Control: Principles and Practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[24] R. Erik, "OASIS eXtensible Access Control Markup Language (XACML) Version 3.0," Oasis, Tech. Rep., 2013. [Online]. Available: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[25] S. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38 – 47, 1996.

[26] S. Guerreiro, *Challenges of Meta Access Control Model Enforcement to an Increased Interoperability*. IGI Global, 2018, vol. 43, no. 01.

[27] S. Rouhani, R. Belchior, R. S. Cruz, and R. Deters, "Distributed Attribute-Based Access Control System Using a Permissioned Blockchain," *arXiv pre-prints*, 2020. [Online]. Available: http://arxiv.org/abs/2006.04384

[28] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity: a proposal for terminology," in *Designing privacy enhancing technologies*. Springer, 2001, pp. 1–9.

[29] Hyperledger Contributors, "Hyperledger Indy," 2020. [Online]. Available: https://www.hyperledger.org/use/hyperledger-indy

[30] Hyperledger, "Hyperledger Aries," 2020. [Online]. Available: https://www.hyperledger.org/use/aries

[31] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In Search of Self-Sovereign Identity Leveraging Blockchain Technology," *IEEE Access*, vol. 7, pp. 103 059–103 079, 2019.

[32] P. Coelho, A. Zúquete, and H. Gomes, "Federation of attribute providers for user self-sovereign identity," *Journal of Information Systems Engineering & Management*, vol. 3, no. 4, 2018.

[33] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 11 2018.

[34] N. Naik and P. Jenkins, "Self-Sovereign Identity Specifications: Govern Your Identity Through Your Digital Wallet using Blockchain Technology," in *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2020, pp. 90–95.

[35] B. Houtan, A. S. Hafid, and D. Makrakis, "A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare," *IEEE Access*, vol. 8, pp. 90 478–90 494, 2020.

[36] A. Othman and J. Callahan, "The Horcrux protocol: A method for decentralized biometric-based self-sovereign identity," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 10 2018.

[37] T. Zhou, X. Li, and H. Zhao, "EverSSDI: Blockchain-based framework for verification, authorisation and recovery of self-sovereign identity using smart contracts," *International Journal of Computer Applications in Technology*, vol. 60, no. 3, pp. 281–295, 2019.

[38] D. Maesa, P. Mori, and L. Ricci, "A blockchain based approach for the definition of auditable Access Control systems," *Computers and Security*, vol. 84, pp. 93–119, 2019.