Extended Self-Sovereign Identity Based Access Control

Guilherme De Seiça Ribeiro Do Quental De Menezes

Thesis to obtain the Master of Science Degree in

Information Systems and Software Engineering

Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia

Examination Committee

Chairperson: Professor José Carlos Martins Delgado
Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia
Member of the Committee: Professor Sérgio Luís Proença Duarte Guerreiro

September 2021

“The best way to predict the future is to invent it.” - Alan Kay

Dedicated to my family

Acknowledgments

Ao Professor Miguel Correia por ter aceite ser meu orientador e pela disponibilidade completa em me ajudar em todos os desafios ao longo da realização deste trabalho.

Aos vários Professors de Ciber-Segurança que me despertaram o interesse nesta área.

Aos meus pais, obrigado por todo o vosso amor, apoio e conselhos importantes que me deram ao longo do meu percurso académico.

À India, obrigado por todo o apoio, amor, carinho, pelos risos em momentos de stress, pelo incentivo e, acima de tudo, pela ajuda que me deste nesta fase final do meu percurso académico.

Aos meus amigos que ajudaram a tornar os meus dias, em tempos de pandemia, menos monótonos e a descomprimir do stress dos mesmos.

A todos os que, de uma forma ou outra, me influenciaram ao longo de todo o meu percurso como estudante.

Resumo

Com o avanço da tecnologia, os dados do utilizador estão a ser partilhados entre diferentes entidades e instituições. Grande parte destas instituições confiam em autoridades centralizadas quando os utilizadores fornecem provas das suas identidades. Estas situações trazem perigos já que estas identidades não são controladas pelo utilizador, podendo ser revogadas ou negadas sem a sua permissão. O controlo de recursos virtuais sofrem os mesmos perigos da centralização, tornado o estudo do controlo de acesso importante. Têm vindo a emergir soluções que tentam resolver os problemas mencionados, recorrendo à Decentralized Ledger Technology (DLT). Através desta nova tecnologia, as aplicações e os serviços virtuais tornam-se descentralizados, ou seja, não ficam sobre o controlo de nenhuma autoridade centralizada. Têm sido feitos desenvolvimentos de conceitos novos de identidades virtuais para fornecer identidades descentralizadas aos utilizadores, dando-lhes assim controlo completo sobre elas. Self-Sovereign Identity Based Access Control (SSIBAC) é um modelo de controlo de acesso que junta ABAC, um modelo de controlo de acesso convencional, com tecnologia blockchain e com identificadores descentralizados para fornecer descentralização aos utilizadores e serviços. Neste documento, propomos Extended Self-Sovereign Identity Based Access Control, versão do SSIBAC aumentada que junta os modelos RBAC, RBAC With Resource Roles e ACL para aumentar o tipo de organizações que podem integrar esta solução e fornecer descentralização e o controlo das indentidades virtuais aos seus utilizadores. O nosso objectivo é fornecer o controlo completo de recursos e de identidades aos utilizadores e contribuir para a descentralização de serviços virtuais e aplicações.

Palavras-chave: descentralização, identificadores descentralizados, controlo de acesso, autorização com base em funções, autorização com base em listas

Abstract

With the advancement of technology, data is being shared among different entities and institutions. Most of these institutions trust central authorities when users provide proof of their identity. This situation comes not without danger as these identities are not controlled by the users and can be revoked or denied without the users' permission. Control over online resources suffers similar risks of centralization, making access control also an important subject to study. Solutions have been emerging to tackle those problems involving Decentralized Ledger Technology (DLT). Through this new technology, online services and applications become decentralized, i.e., no central authority controls them. Efforts have also been made to implement new concepts of identity in order to provide decentralized identities and give their control back to the users. Self-Sovereign Identity Based Access Control (SSIBAC) is an access control model that joins ABAC, a conventional access control model, with blockchain technology and decentralized identifiers, providing decentralization to users and services. In this document, we propose Extended Self-Sovereign Identity Based Access Control, a SSIBAC extended version that adds the RBAC, RBAC With Resource Roles and ACL models, increasing the type of organizations that can integrate this solution and give decentralization and identity control to their users. Our goal is to provide full resource and identity control to users and contribute to the decentralization of online services and applications.

Keywords: decentralization, decentralized identifiers, self sovereign identity, access control, role-based authorization, list-based authorization

Contents

List of Figures

Chapter 1

Introduction

In a world where more and more people are using the Internet, data is being collected and shared between a large variety of entities, involving people, companies, medical institutions or even governments. In the real world, our identities are tied to central authorities and governments through citizen cards, social security numbers or driver's licenses. On the Internet, most official social and economic services and applications, like digital markets, business companies or social networks, require a *digital identity* issued or verified by those central authorities or governments to allow data to be shared among these institutions.

1.1 Motivation

Gathering and sharing this kind of information, poses a danger to our right to personal privacy as our identities and personal data can be controlled, denied or revoked [All16]. With the advancement of technology and its ubiquity, and with large amounts of data collected and analysed, threats like identity theft or data breaches are raising concerns and harming the trust placed in online services. Because of this, *access control* is becoming an important study matter in order to prevent illegal resource access.

1.2 Topic Overview

Bitcoin was the first decentralized software, i.e. not controlled or managed by a single centralized entity or authority, to emerge that managed transactions implementing a cryptocurrency and providing security. Using a *blockchain*, Bitcoin offers a ledger of records distributed across many computers around the world and security through the use of cryptography.

Since then, blockchain technology has been included in many projects that had in mind other uses for it. After Bitcoin made this technology's possibilities known to the world, ordering food, Uber-like transportation and other services and smart devices' applications could be made possible with Ethereum's *smart contracts*. These are applications distributedly executed on the Ethereum's decentralized network. Ethereum is just one platform among many other blockchain-based solutions that emerged in recent years and showed the potential of Decentralized Ledger Technology (DLT) [Pec17].

Work has been done to give control over an identity from central authorities and federated models back to the users in a decentralized manner through self-sovereign identities. This way the identity can not be denied or revoked without the user's consent [TR16].

There are solutions that combine blockchain technology and access control practices such as BlendCAC [XCBC18] as a way to provide decentralized controlled access to resources, but there isn't a definitive approach that provides the same benefits while using a layer of self-sovereign identity between the user and the blockchain.

1.3 Objectives

In order to achieve authentication based on Decentralized Identifiers (DIDs) and decentralized authorization over a resource, we initially aimed to build a solution centered on the Ethereum' *smart contracts*. Information about DIDs would be gathered with the help of a new technology called *universal resolver*.

The QualiChain project, financed by the European Commission, designed a solution for the verification of a person's academic qualifications in a job seeking context [SVSG20]. This solution uses blockchain technology to provide the authenticity and integrity of qualification certificates in a decentralized platform. As a use case, our original goal was to combine the work done by the authors with a layer of DIDs, furthering their research and providing true decentralization over identities. More specifically, we aimed to develop a service capable of providing authorization and integrity over a person's qualifications certificate by integrating a decentralized version of OAuth2 framework.

Due to increasing difficulties in more than one technological aspect of our intended solution, we were forced to change our approach and contribute in a different way to the development of a decentralized solution for the previously stated problems. We decided to extend the work done by the authors in [BPP$^+$20]. More specifically our objective was to build a more generic version of SSIBAC model that included more traditional access control models adapted to decentralization.

As such, we propose Extended Self-Sovereign Identity Based Access Control (extSSIBAC), a SSIBAC extended version that adds the RBAC, RBAC With Resource Roles and ACL models, increasing the type of organizations that can integrate this solution and give decentralization and identity control to their users. Our goal is to provide full resource and identity control to users and contribute to the decentralization of online services and applications.

1.4 Thesis Outline

The remainder of the document is structured as follows. Section 2 details related work, starting with information about blockchain technology, where work is presented about the most common types of identity management systems as well as some developments on the Hyperledger Project, important to understand our solution. Section 3 describes the proposed solution architecture as well as its components. Section 4 defines the methods of evaluation used to assess the capability of the proposed solution. Finally, Section 5 concludes this document.

Chapter 2

Background and Related Work

To understand the work completed in this project, this section presents our understanding of blockchain technology and its advantages towards building decentralized applications as well as an overview of Ethereum and Hyperledger Indy, two famous examples of *permissionless* and *permissioned* blockchain systems. Next, a framework to build applications based on decentralized identities will be explained. This section also talks about two new concepts of decentralized online identity. Lastly, access control and some of its mechanisms are mentioned to understand how they can be integrated with a decentralized way of identifying an entity.

2.1 Blockchain

In the context of trying to conceptualize a way to manage access control with decentralized identities, this section presents the current state-of-the-art about blockchain technology, mentioning two different kinds of blockchains as well as the value of smart contracts towards decentralized applications.

2.1.1 Blockchain and Bitcoin

Blockchain is a digitally distributed ledger technology that is having an increasing impact in the industry, commerce and global economy. Besides security properties, like integrity and availability, features such as cutting middleman organizations, namely banks and other third parties, make it appealing for developing new ways to make secure transactions cheaper [Und16]. A blockchain is an append-only data structure consisting of nodes, globally distributed computers on a network, that agree on a set of blocks. These blocks contain information regarding states and transactions and are consecutively connected to each other through cryptographic pointers forming a chain of blocks and maintaining their history protected. This means that when a new block is attached it becomes immutable [DLZ+18].

Since the nodes in the blockchain do not completely trust each other, a transaction operation is replicated between all nodes after going through a consensus process that validates it. The consensus process is done by miners, computers in this network that detect transactions requests and examine them to check the validity of the owner's transaction by calculating a mathematical relationship between the owner's cryptographic keys that are used to digitally sign each new transaction. A miner's main objective is to ensure that these transactions are irreversible. Each new block points to its predecessor, making them tamper-proof and final. These new blocks have to be accepted by all nodes hence a consensus protocol must be used, Bitcoin's proof-of-work (PoW) is an example of this situation where only the node that gets the right result when computing a mathematical problem is allowed to append the new block to the blockchain [Pec17].

A blockchain can be a public, fully decentralized system also designated as permissionless where the network is decentralized and one does not need to disclose their real-world identity making it necessary to use PoW like protocols to decide which computer has the right block to be added to the chain and protect the ledger against attacks. One can also play the role of the miner and use its protocols for verifying a transaction and no central authority can delete or modify the records in the blockchain. Since some financial institutions are required to have their well identified clients, there was need to find an alternative to permissionless blockchains and as such, a blockchain can also be a permissioned system where some access control and a small degree of trust exists between all nodes. Only selected participants can view the information in the system and add new blocks to the chain, and as they are well identified the use of proof-of-work protocols is unnecessary [Pec17].

2.1.2 Ethereum and Smart Contracts

Ethereum is one of the most popular permissionless blockchain systems. This section considers the original version of Ethereum [BE17], not the Ethereum 2.0 that is currently being launched and with slightly different characteristics. Ethereum is similar to Bitcoin apart from the fact that it allows for the creation of decentralized applications and smart contracts using a Turing-complete programming language. An account in Ethereum can be one of two types: an Externally Owned Account (EOA) that contains no code whose messages can be sent via transactions signed with its private key or a Contract Account (CA) that holds code and when a message is received, the code is executed enabling it to read and write to internal storage [BE17].

Decentralized Applications (DApps) are run by Ethereum's state machine and the result is stored in the blockchain. A DApp is composed by a smart contract and a web frontend user interface.

A smart contract is a program that is executed by the Ethereum Virtual Machine (EVM). Once deployed, it can not be modified. A smart contract is written in a high-level programming language like *Solidity* and then compiled into the low-level byte code language executed by the EVM. Its result is always the same depending on the context of the transaction that launched its execution and the blockchain's state at the time of execution [AW18].

Each smart contract is identified by its own unique address and the only way to execute it is via a transaction, by calling one of its functions with the contract address. It is possible to have a chain of contract calls, but the first has to have been called in a transaction by an Externally Owned Account, which is controlled by the users [AW18]. A Smart Contract can be implemented for several tasks including conditional effects, such as transferring timely payments of a security deposit or moving a certain amount of money between accounts when a certain event, namely the trigger, occurs [LCO$^+$16].

A transaction contains a message, the STARTGAS and GASPRICE values, some amount of *Ether*, Ethereum's cryptocurrency, and other data. STARTGAS represents the limit of computational steps that a code executed by the transaction is allowed to take, preventing infinite loops and GASPRICE is the fee that needs to be paid by the sender for each computational step. This serves as a deterrent to denial-of-service attacks, because the attacker would have to pay for the all resources expended. If the totality of the value of GASPRICE is consumed before the end of the execution, all state changes revert back, the execution is stopped and the fee for the work done is still paid. All transactions are signed by the sender's private key [BE17].

2.2 Identity Management

Nowadays most systems manage the identities of their users through centralized methods usually based on trusted entities. Even though this way of identifying an individual is useful in some cases, in other situations it is thought to be a threat to human society's autonomy as habits, transactions and internet activity can be surveilled by governments interfering with personal privacy and having some power over a person's behaviour. Some activities and actions such as using public transportation and other public services do not require a centralised way of managing identities [GA19]. Furthermore, centralized Identity Management (IdM) systems have been suffering from regular data breaches, facing cases of identity fraud and other cyber attacks, making it less secure for people's privacy [DP18].

Active Directory from Microsoft is an example of a centralized IdM system. It allows companies' network administrators to maintain all kinds of data in centralized repositories and as soon as information gets into the system, it can be accessed and use throughout the organizations. This data includes personal information about companies' workers, such as names and email addresses. Active Directory uses Access Control Lists (ACLs) to manage permissions on who can access each object and the worker's identities are kept in plain text as one of the object's attributes [Ise99].

Decentralised IdM systems that may solve some of these problems, have been emerging recently via the use of blockchain technology where the system control is shared between members and new forms of managing identities are created [GA19].

Hyperledger Indy is an example of decentralized IdM system. Indy gives its users some identity control by enabling them to manage to whom they want to disclose their identities without the influence of centralized entities [SS18]. Indy focuses on the privacy of the users' identities and being part of the Hyperledger Project, Indy accomplishes this by using decentralized ledger technology. Since data is distributed across thousands of computers, it cannot be denied [uub18].

2.2.1 Decentralized Identity

An Identity that is not regulated by any central authority and is controlled only by its owner is designated self-sovereign identity [All16]. Its ownership can not be denied and a decentralized system can facilitate the implementation of this concept of identity. Before self-sovereign identity, central authorities had full control of a user's digital identity, it was in their power to deny its existence and users had to manage multiple identities which they had no control over [All16].

Decentralized Identity can be achieved by implementing the concept of self-sovereign identity using decentralized ledger technology. This way, the history of activities and data changes cannot be tampered with, users keep control over their identifiers and since their identity information is kept in a decentralized ledger, no central authority can deny it or change it. Namecoin is a cryptocurrency with a blockchain [KCE$^+$15], similar to Bitcoin that was able to achieve decentralization and security while keeping names readable [DP18]. It stores name/value pairs on the blockchain and users can trade these values between them for a cost. This makes Namecoin a namespace and users have full control over the names on the blockchain. Namecoin has also some limitations, for example, the amount of readable and meaningful names to humans are limited as opposed to the normal computer generated identifiers on other blockchain systems [KCE$^+$15].

Effort was made to create an alternative to the conventional way of creating and using virtual identifiers in order to offer decentralization in a secure way. Decentralized Identifiers (DIDs) were created and work is being done towards building a tool capable of resolving them by the Decentralized Identity Foundation (DIF), called Universal Resolver.

A DID is composed of three parts: a URL scheme identifier, an identifier for the DID method and a DID method-specific identifier. A DID method defines how the syntax of a DID can be implemented in different blockchains. A DID is associated with a DID Document, which contains information that identifies the user that it belongs to, such as public keys and other data that can be used to authenticate the user and prove his or her identity. This document is retrieved by the Universal Resolver [W3C20].

The Universal Resolver can be included in architectures or protocols in order to determine information behind each DID, like cryptographic keys and service endpoints. Most DID registration methods are supported by this tool regardless of the blockchain system used. Even though the Universal Resolver is still being worked on, it is a good tool to be used when implementing the use of DIDs with decentralized ledger technology [Sab17].

Decentralized Identifiers are a specification of the World Wide Web Consortium (W3C), whose main goal is to produce guidelines that ensure the Web's development. W3C plans for DIDs to become a W3C Recommendation, and as such, we believe that DIDs are a good option when developing solutions that aim to provide full identity control to users. [W3C20].

2.2.2 Hyperledger Indy

Hyperledger Indy enables the storing of digital identities on distributed ledgers by providing the necessary tools and components. Decentralized Identifiers are the core of these digital identities. Indy is a public permissioned blockchain, meaning everyone can access it, but to be able to place transactions on the ledger an entity needs to have the role of Trust Anchor. A Trust Anchor is an entity that is recognized by the ledger. It is within its abilities to publish new DIDs on the ledger, which is done through a transaction. One of its most important fields is the Verkey (target verification key), that allows an entity to verify that someone who knows the corresponding signing key is the unique owner of that DID. Furthermore, a Trust Anchor can create Credential Schemas. These schemas delineate the attributes of a specific credential that can be used to define a specific document. Indy's ledger stores public data such as public keys, credential definitions, credential schemas and service endpoints[uub18].

2.2.3 Hyperledger Aries

Hyperledger Aries is application framework that provides tools that allow interactions, based on a blockchain, between entities. It offers a blockchain interface to create, read and sign transactions. Aries has a cryptographic storage to keep secrets and other sensitive information safe, to build clients for issuing and proving Verifiable Credentials(VC). Using Aries' secure messaging system, depended on DIDs, entities are able to interact outside the blockchain environment through those clients [H$^+$20].

A Verifiable Credential is the digital equivalent of a physical credential and it represents the same important information. The owner of a VC can generate a Verifiable Presentation (VP). This VP can then be shared with a verifying entity, allowing it to validate the claim that the owner of that VP possesses a VC with specific characteristics [W3C].

Zero-knowledge protocol can be used to support the exchanging of VPs [H$^+$20]. Zero-knowledge proof is a cryptographic technique where, without revealing any sensitive information or further details, an entity can prove to another that a specific statement is true [YL20]. Finally, Aries provides a few framework implementations that allow for a variety of different use cases.

Developers can add application-specific code to the Aries agent framework to create particular applications. One of its agent frameworks is the Hyperledger Aries Cloud Agent Python (ACA-Py). This framework provides tools for environments based on VCs [acp]. With ACA-Py, for example, one entity is able to prove it owns a certain credential in order to be provided with a service from another entity. In this case, both entities are represented by two different Aries' agents identified by each owner's Decentralized Identities that are based on a distributed ledger [ari].

2.3 Access Control Based on Blockchain

With the growth of objects and new technologies that are being connected to the Internet, data is being increasingly collected and shared between entities, sometimes without the user's consent [ZKS$^+$18]. Privacy and authentication are some of the issues that come with the relationship between smart devices and the Internet with access control being one of the top concerns.

2.3.1 Access Control

Most software, like a computer's operating system, ensure the protection against malicious access attempts to resources. This is done with access control and its main focus is to manage what actions are allowed to be taken by users and prevent data breaches. This kind of control is enforced by a reference monitor, an abstract component that relies on an authorization database to allow or deny a certain action. This component is the intermediary between the users and the resources [SS94]. The reference monitor should be impossible to circumvent, impossible to be maliciously modified and it should be verifiable to ensure its correctness [CS17]. It is important to mention that authentication is different from access control and ideally these two phases of a system are separated as access control performs its function usually after the user identity has been verified [SS94].

The standard that presents the eXtensible Access Control Markup Language (XACML) defines an access control model [ANP$^+$03]. This models breaks down the access control process in a set of components. This division is useful to understand that process, so we briefly introduce them here. The Policy Administration Point (PAP) is the component that manages access authorization policies. The Policy Decision Point (PDP) is the component that effectively evaluates if access should be granted or denied. The Policy Enforcement Point (PEP) is the component that intercepts the subject's access resource and passes it to the PDP for a decision to be taken. Finally, the Policy Information Point (PIP) is the component that stores the attributes analysed in the process (e.g., resources and subjects).

Access Control List (ACL), Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC) are some of the approaches used in IT systems nowadays, as mechanisms that decide who gets to access what resource regarding some security policies.

The ACL mechanism associates to every resource a list of subjects that can access it and their respective access level (e.g., "only read" or "read and write"). RBAC defines each user's access right according to their roles and each role's privileges [DRFM18]. ABAC grants access to users by considering their attributes associated with the resource they are trying to access and its access policies [HFK$^+$13].

These mechanisms lack the ability to provide good tools to face the rapid growth of the number of smart devices or the challenge of adapting to a panoply of different technologies with different identity authentication specifications [XCBC18]. As progress is made with blockchain technology and smart contracts, some ideas have been appearing to deliver reliable decentralized access control mechanisms.

2.3.2 Access Control with Blockchain

Since the previous mechanisms for access control are normally based on centralized entities, [ZKS$^+$18] presents an access control framework to tackle access control problems in an IoT system composed of a server, a storage device, a user device, an IoT gateway and finally an IoT device. The Ethereum smart contract platform is used to create Access Control Contracts (ACCs), which are deployed by peers in order to control access requests to a resource from other peers, Judge Contracts (JC) which judge the behaviour of peers who want to access that resource and arrange a penalty in case of a misbehaviour report sent by an ACC and lastly, a Register Contract (RC) which register all the information from misbehaviour-judging methods and manage them. This work does not mention any way of identifying users nor does it show the possibility of integrating a decentralized identity management system with its architecture as this was not the authors' main focus.

[MMR17] is another example of a solution to improve access control mechanisms using blockchain technology. In this case, the access control policies are saved in a distributed ledger. The system's Policy Enforcement Point redirects a user's access request to an Authorization System. There, the Policy Administration Point interacts with the blockchain and reconstructs the security policy related to the action being requested with the information retrieved. After that, the access request is evaluated against the security policy and a decision is returned in the end. Once again, no specific decentralized means of identifying the users is mentioned.

We argue that by combining blockchain technology, decentralized identifiers and most access control models, we can have systems working in their intended way without the need to exchange personal information of their users and with better security by integrating blockchain functionalities in the access control process.

Chapter 3

extSSIBAC

This section starts by providing an overview of the Qualichain Certificate Verification system and the SSIBAC model, essential to fully understand the proposed solution. After that, our extSSIBAC solution and its architecture are described. Finally, this section ends by detailing our use case and implementation of our work.

3.1 QualiChain Diploma Verification and SSIBAC

3.1.1 QualiChain Certificate Verification Overview

As stated previously, Serranito et al.[SVSG20] presented the design and implementation of a system that provides a decentralized qualification certificate verification in a job seeking context. The authors identify three important external entities: Higher-Education Institutions (HEIs), recruiters and job seekers. The system is composed of a consortium smart contract, which contains data about a set of HEIs, and each HEI is represented by a HEI smart contract which contains information about the certificates.

The consortium smart contract keeps an identifier for every HEI registered in the system and membership of each new HEI is decided by a voting scheme between the current members. A HEI needs only to use the HEI Client to register, revoke or verify a certificate. This Client, in turn, calls that HEI smart contract corresponding methods. The system does not store the full virtual certificate information but keeps a hash of each certificate's PDF along with each certificate corresponding job seeker ID. Finally, recruiters use the Recruiting App to verify the job seekers' education certificate.

The authors allow different representation of each job seeker's ID, but there is not an actual implementation of the system using DIDs to identify a job seeker. An access control mechanism is also not present so in order to allow the use of decentralized identifiers and provide security when accessing the certificates' hashes through an access control mechanism, we propose added components to this system's architecture as well as a change in the system initial behaviour in order to provide resource authorization.

3.1.2 SSIBAC

As stated before, authentication methods and access control management using centralized models, based on personal sensitive user information, face a wide variety of risks and challenges. This risks include the gathering of unnecessary user data to perform access control, privacy violation through data breaches, ineffective security policies and the improper sharing of private user information. Belchior et al.[BPP^{+}20] argue that Self-Sovereign Identity (SSI) in conjunction with the use of blockchain technology, is a viable tool for an alternative user authentication method, allowing for the existence of a more secure access control management. These authors further explain that by combining blockchain technology, decentralized identifiers (DIDs) and verifiable credentials (VCs), they were able to propose a decentralized and safe access control management model called SSIBAC, Self-Sovereign Identity Based Access Control.

On the SSIBAC model, a user is identified by a DID. The permission to access a resource is granted after a decision is calculated considering a set of variables. After a request is put forward, one of these variables is the mapping of a verifiable credential to a permission validator. In this case, an instance of SSIBAC was initialized with the ABAC model, so the permission validator is a specific attribute policy that a user must fulfill in order to access that resource. Those VCs are issued by trusted issuers. DIDs, VCs and verifiable presentations (VPs) are supported by blockchain technology.

The SSIBAC model can be explained in terms of a few simple steps:

- A user starts by requesting a verifiable credential from a certain issuer
- After issuing that VC, the credential schema and the proof of the VC emission are stored on a decentralized ledger
- The user is then able to request access to a specific resource
- To gain access, the user needs to generate a VP and reply to a VP request from the verifier
- After the VC validation, the verifier redirects an access control request to an access control engine
- Following the access control decision computation, access is given if the user's credentials meet the verifier's system's security policies

In the SSIBAC use case, the decision computation is done based on a Zero-Knowledge proof related to a certain attribute. This allows maintaining privacy about the parameters used for evaluating access (e.g., what is the exact age of the user). However, it is also restrictive because the access control models we introduce require verifying if there is an exact match of certain parameters (e.g., is the user really John).

SSIBAC allows for access control management using decentralized technologies in order to provide user identity sovereignty and protect user privacy, where users disclose only the necessary information to gain access to a certain resource, preventing unnecessary risks and mitigating the consequences of data breaches [BPP+20].

3.2 extSSIBAC

[BPP+20] describes an instance of SSIBAC with the ABAC model. Our work is to extend this Self-Sovereign Identity Based Access Control prototype to include other relevant access control models. More specifically: RBAC, RBAC with resource roles and ACL.

3.2.1 Access Control Models Supported

These are the access control models supported by our prototype.

RBAC Model

This access control structure is defined by the relationship between a user and a set of roles. These roles are then associated with different permissions.

Users are validated to access a resource if they possess the proper role. This role can be assigned by the service the user is using or by a different separate entity. This model can be defined by a set U of users, a set R of roles and a set S of services. A user u with a role r is allowed to access a specific service s if there is a relationship between (u,r) $\in$ UA and (r,s) $\in$ SA where UA $\subset$ U $\times$ R and SA $\subset$ R $\times$ S [CKY18].

In a business context, roles can be associated with different job functions and can be appointed to users based on their qualifications and duties. These roles can be reassigned and permissions can be added or removed from a specific role. This model permits the access control management to be considerably more simplified [San98].

RBAC with Resource Roles Model

extSSIBAC allows for the mapping between users and roles to be done in a secure way by preserving user identity using decentralized identifiers. RBAC with resource roles is an access control model where both users and resources can be assigned roles, allowing for the creation of groups, making extSSIBAC able to meet the complexity of a more elaborate system. This way, with RBAC With Resource Roles, resource access group policies can be defined relating to multiple resources instead of one policy per resource [cas].

ACL Model

The Access Control Lists model focuses on building a relationship between objects and users. An action on an object can only be performed if the user or a group of users carrying it are listed on that object's ACL and if that action is associated with those users [Bar97]. Although extSSIBAC was implemented so that it could also be instantiated with the ACL model, in our solution, a user's privacy protection is limited due to the way our use case is configured.

3.2.2 Arquitecture

extSSIBAC's architecture is similar to the original SSIBAC model. It tries to be adaptable to the structure of all kinds of organizations. Figure 3.1 provides a simple schematic of our solution's general architecture. Our system is composed by three Hyperledger Aries framework agents, a decentralized ledger and an access control engine.

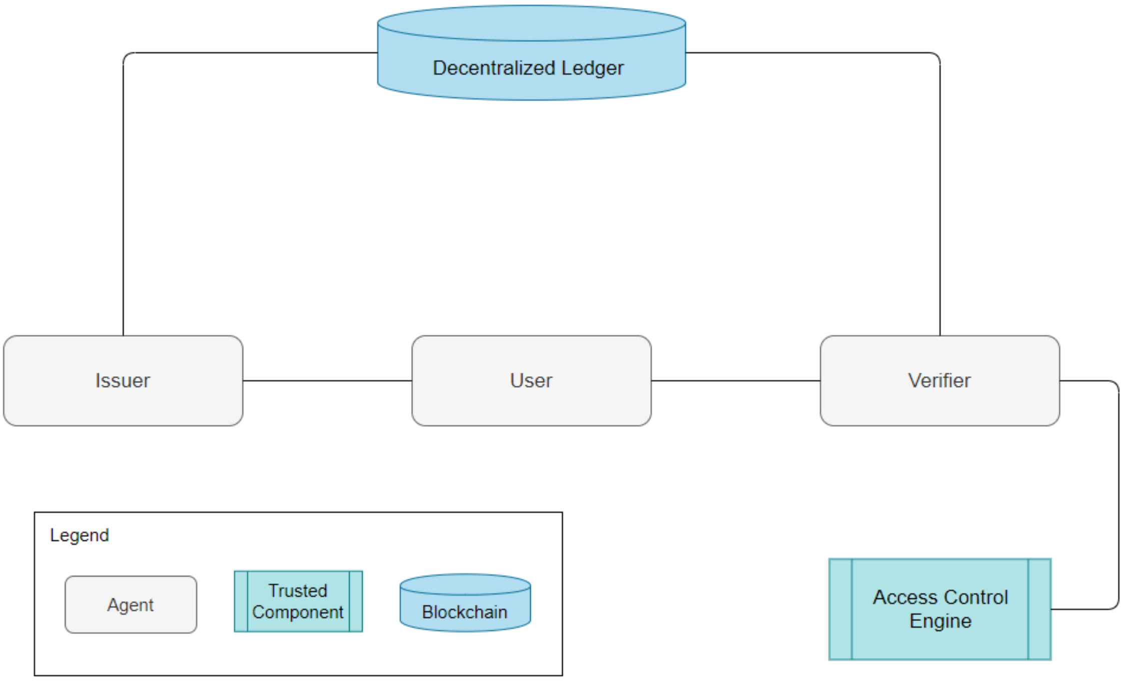

Figure 3.1: General architecture scheme

A user would get a verifiable credential from a trusted issuer. That VC's schema and proof of emission would then be stored on a decentralized ledger, this way, no official credential associated with a specific user could be denied or altered. The user would then be able to use that VC to access a specific resource or service. The verifier, which in this case can be a company, service or a specific system, would ask for a VP from the user. After that, the user would be able to prove to the verifier that they have the necessary credentials to use or access that particular resource or service. Next, after receiving the VP from the user, the verifier would then confirm on the blockchain that the credentials the user presented exist and are valid. Finally, the verifier would redirect the request to access to its access control model in order to calculate a decision for that request and return it to the user.

3.2.3 Integrating the Access Control Models in the Arquitecture

In our use case, the verifier entities choose what model they want to use to control access to a specific resource or object. This is done using a built-in access control library. In our system, the Policy Information Point (PIP) is represented by the users as they will be the source of the credentials necessary to access a resource. The verifier entities represent both the Policy Administration Point (PAP) and the Policy Enforcement Point (PEP) as they will manage the security access policies and redirect the access requests to the Policy Decition Point (PDP). Finally, the Access control engine, the PDP, will be provide a decision.

Even though in our use case every entity will have a personal DID, a unique pseudonym (unique DID) is created for each specific connection. On the one hand, this is a great way to provide extra security and privacy to all entities involved, as a profile about any entity is impossible if their identifying DID is always changing. On the other hand it brings certain challenges depending on the access control model being used. In the ABAC case, the service provider will request a VP to make sure the user possesses the necessary attribute values. The access control engine will infer the attribute value to allow access through ZKP and compare it to the security policies in the system. This way, the user does not fully disclose the values of the VC issued to them by the trusted issuer.

With RBAC and RBAC with resource roles, in a business context, there is no need to keep the role of a user secret. A service provider will request the role from the user in a VP. As such, a specific role value will be disclosed in a VP generated by the user. Only if the user was issued a credential with the proper role value from a trusted issuer, they will be given access. From the organizational point of view, it is also possible to assign roles to resources and access them without disclosing anymore information about the user than already necessary with the basic

RBAC model.

As a new DID is created each time a new connection is established, this poses a challenge when using the ACL model. As explained before, a user needs to be on an ACL of a resource in order to be able to perform a certain action. As such, the user needs to be properly identified. This can still be done with decentralized identifiers if the system in question openly doesn't protect against behaviour profiling, but in our use case this doesn't happen, so each user has a username registered in the system, that also needs to be part of a VC issued by a trusted issuer and is verified through the normal exchange of VPs between the user and the verifier.

3.2.4 extSSIBAC Use Case

As mentioned in the document written on the previous phase of this work, we initially aimed to proposed a solution similar to the SSIBAC model, but based on Ethereum and the smart contract technology. The idea was that users (job seekers) would apply for an opening at a company by sending their qualification certifications, emitted by their university, and other relevant information. Recruiting organizations would then be able to join a QualiChain Certification application that would validate the applicant's certification. This application would receive the applicant's DID and the DID of their Higher-Education Institution (University). After that, their DID would be verified by the Universal Resolver, a technology able to resolve DIDs and obtain important information about their owners. The next step was for the HEI smart contract to ask permission from the user (job seeker) to share the certification's hash with the recruiting organization. Confirmation that it was a valid certification from the user would come by comparing the hash received from the applicant and the one returned by the recruiting application. The scientific addition by the proposed work was to combine the Qualichain Certification Verification system with the new Universal Resolver technology, decentralized identifiers and a relevant access control model.

Unfortunately our investigation was challenging particularly due to the fact that there was a lack of detailed documentation about the new technologies proposed and communication with the developers was difficult and slow.

We decided to change perspectives and contribute with an extension of the work mentioned in [BPP$^+$20]. We combined a few more access control models with SSIBAC to prove that a system to manage access to a resource using ledger technology and decentralized identifiers was possible to be implemented with other relevant access control models.

extSSIBAC aims to aid in the resource organization and security policy building by providing the means to choose what better access control model fits a specific organization. Decentralization need not to be discarded in favor of a specific access model. Entities have their credentials securely stored on a decentralized ledger and without the right credentials access is denied.

Figure 3.2 shows our final solution arquitecture and access control flow. Alice starts by requesting a verifiable credential from her university of her qualifications (1). After the University (issuer) issues Alice's VC (2), it publishes the credential schema and proof of credential emission on a decentralized ledger(3). A request to use a service provided in the context of the QualiChain project is sent to the QualiChain Service Provider (QualiChain Certification Verification system)(4). A verifiable presentation request is generated (5) and sent back (6) to Alice. After generating a VP, Alice sends the VP back to the QualiChain Service Provider (7). Next, Alice's VC is validated (8) and the access control request is redirected to the Access Control Engine (9). Here, the AC model choosen by the service provided calculates a decision (10) that is returned to the QualiChain Certificate Verification System Agent (11). Finally, if Alice presented the proper credentials, the decision will be to allow access (12).

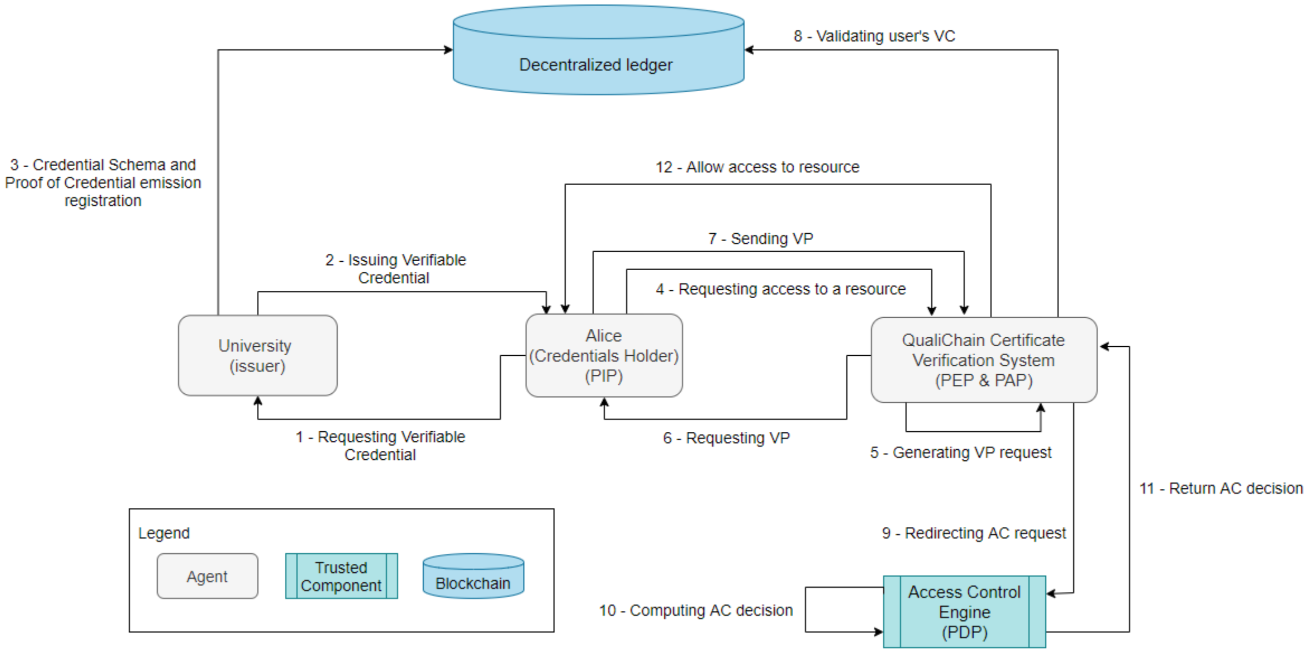
Decentralized ledger
8 - Validating user's VC
3 - Credential Schema and Proof of Credential emission registration
12 - Allow access to resource
7 - Sending VP
2 - Issuing Verifiable Credential
4 - Requesting access to a resource
University (issuer)
Alice (Credentials Holder) (PIP)
QualiChain Certificate Verification System (PEP & PAP)
1 - Requesting Verifiable Credential
6 - Requesting VP
5 - Generating VP request
11 - Return AC decision
9 - Redirecting AC request
Legend
Agent
Trusted Component
Blockchain
10 - Computing AC decision
Access Control Engine (PDP)

Figure 3.2: extSSIBAC applied to a QualiChain Project scenario

3.3 Implementation

The decentralized ledger from our solution was implemented by using Hyperledger Indy, a blockchain where the credential schemas, proof of verifiable credential emissions and the decentralized identifiers are stored. As such, for testing purposes, the GreenLight Dev Ledger, brought fourth by the Verifiable Organizations Networks (VON) blockchain test net for developers, based on Hyperledger Indy, was used.

Figure 3.3 shows extSSIBAC applied to our use case with the technical components in place. Alice, the university and the QualiChain Credential Verification system are represented by three Hyperledger Aries Cloudagent Python agents (ACA-Py) built to be able to establish connections and exchange credentials between them. Each agent will have a designated decentralized identifier and a unique DID will be created for each new connection. The university agent is capable of generating verifiable credentials and both the Alice agent and the QualiChain agent are capable of generating verifiable presentations. Every agent is able to communicate with the GreenLight Dev Ledger.

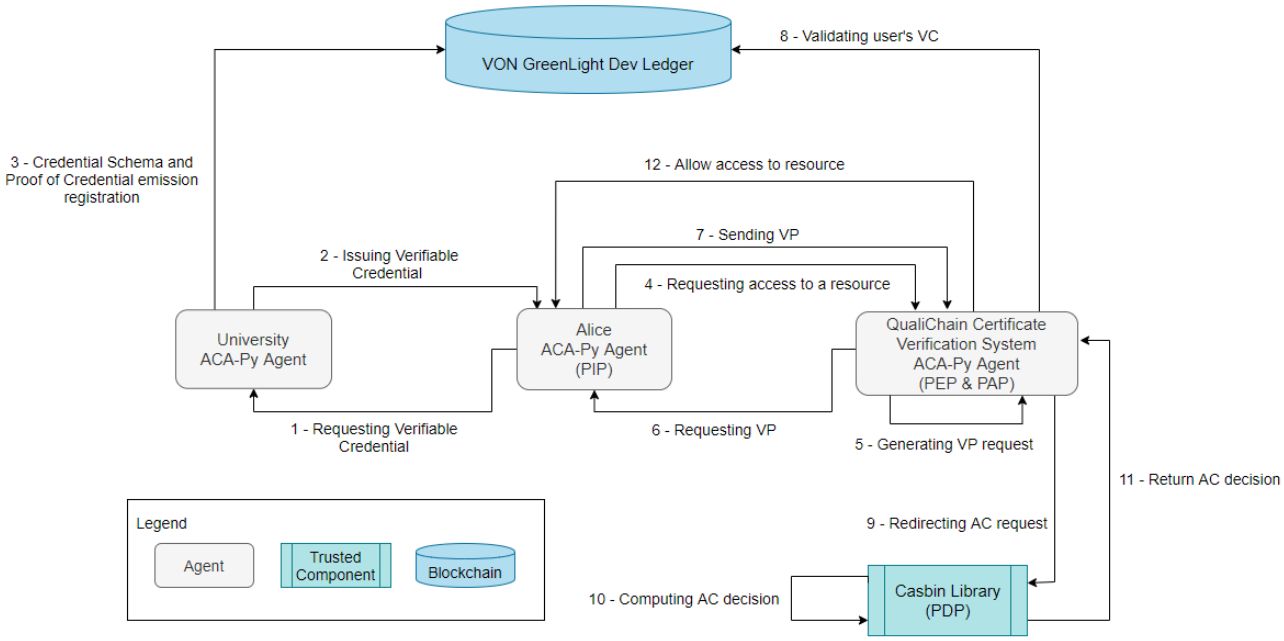
VON GreenLight Dev Ledger
8 - Validating user's VC
3 - Credential Schema and Proof of Credential emission registration
12 - Allow access to resource
7 - Sending VP
2 - Issuing Verifiable Credential
4 - Requesting access to a resource
University ACA-Py Agent
Alice ACA-Py Agent (PIP)
QualiChain Certificate Verification System ACA-Py Agent (PEP & PAP)
1 - Requesting Verifiable Credential
6 - Requesting VP
5 - Generating VP request
11 - Return AC decision
9 - Redirecting AC request
Legend
Agent
Trusted Component
Blockchain
10 - Computing AC decision
Casbin Library (PDP)

Figure 3.3: extSSIBAC with the technical system components

The access control models proposed by our solution are implemented using Casbin, an access control model open-source library and ABAC, RBAC, ACL are a few of the models supported. Our solution is implemented using the Python programming language, also supported by Casbin, as it is effortless to install and use. Casbin's workflow involves two phases, configuration and implementation. On the configuration phase, each model is defined based on the requirements of each system that is employing Casbin. The implementation phase is done in a policy file, where we define the subject that can or can not do a certain action over a resource [cas]. Each access control model has their own policy Configuration and policy Comma Separated Values files. After Alice's credentials are validated, a Casbin enforcer will be called to try and match Alice's access request with the QualiChain system's security policies. If everything is within the system's access policies, Alice will be allowed to perform the action requested.

In an instance of extSSIBAC with the RBAC model, the university will emit a credential with a role field where its value will be Alice's role in that institution (student or teacher). The QualiChain service provider will first validate if Alice has a valid credential with the role she claims to have and then it will check if that role is enough to perform the action Alice is requesting. With RBAC with Resource Roles, Alice will take the hypothetical role of someone with the function of QualiChain service provider administrator (data_group_admin), where she will be able to read two different objects belonging to the same group that are identified with the same role. Finally, in extSSIBAC instantiated with the ACL model, Alice will have a specific username that will be written in a specific resource's ACL in order to be allowed to perform some action over it.

Chapter 4

Results

This section describes our testing environment and methodology as well as our understanding of the results obtained. Our solution aims to be an extension of the SSIBAC model. As such, our objective was to make sure that our system behaved in an acceptable way with the additions brought by the work done in this dissertation.

4.1 Testing Environment and Methods

For testing purposes, a portable computer was used with the following specifications:

- Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 processor
- NVIDIA GeForce GTX 965M
- 16.0 GB RAM
- 64-bit Windows 10 operationg system
- Intel 802.11ac (2x2) Wi-Fi and Bluetooth 4.2 Combo

A Oracle VM Virtual Box Manager was used to create a virtual machine where our solution was implemented, with the following specifications:

- Ubuntu 20.04 64-bit operating system
- 10 GB base memory
- Nº of processors: 1 Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 processor
- 16.0 GB RAM

Hyperledger Aries software uses a group of specifications and protocols that were installed with the following versions:

- aiohttp (version 3.5.4)
- aiohttp-apispec (version 1.1.2)
- aiohttp-cors (version 0.7.0)
- apispec (version 2.0.2)
- async-timeout (version 3.0.1)
- base58 (version 1.0.3)
- Markdown (version 3.1.1)
- marshmallow (version 3.0.0)
- msgpack (version 0.6.1)
- prompt_toolkit (version 2.0.9)
- pynacl (version 1.3.0)
- requests (version 2.23.0)
- casbin (no version specified)

Hyperledger Indy version 1.11.1 was installed and in total, our solution was tested 30 times, 10 for each access control model and the complete process can be divided into three phases (startup, connect and access control) as explained in [BPP+20]. Since it was not possible to test our solution in the same testing environment as the one used by the authors and since the ABAC model was already implemented, we also decided to test the ABAC model 10 times in our own testing environment and use those as a baseline.

4.2 Evaluation and Results

Our startup phase began by initializing the three agents (University, Alice and QualiChain service provider). This was done using Docker containers, one container for each agent. A Docker container is a piece of software that encapsulates all application code and the related dependencies in order for it to be easily transported and used on any computing environment [doc]. By launching the agents using Docker containers, we save the effort to install all their necessary components. Also included in this initial phase was the time it took for the university agent to publish the schema for the credentials on the blockchain.

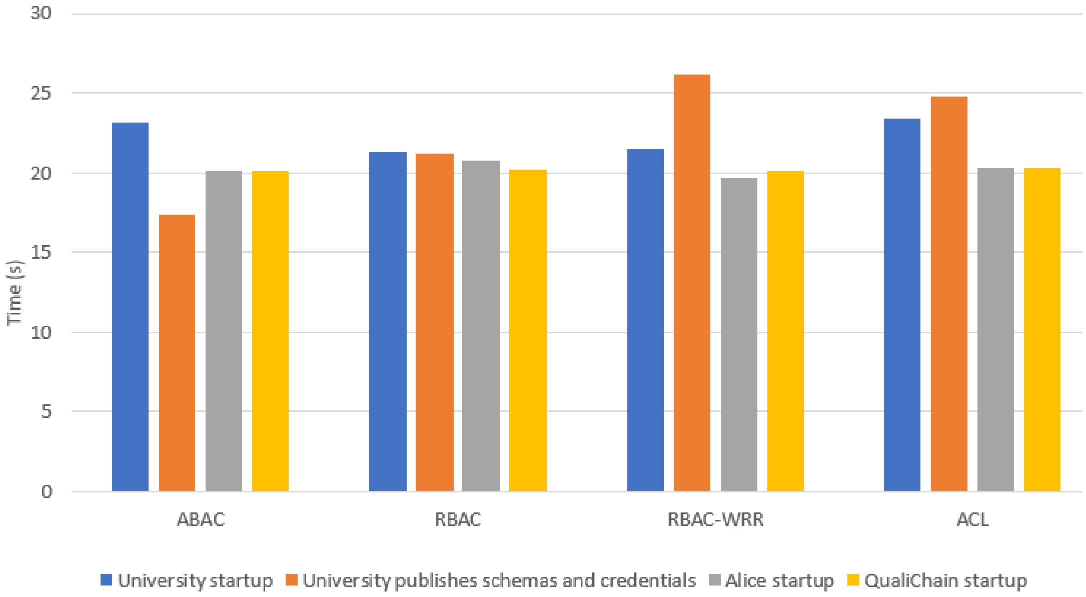

Figure 4.1: Startup Phase Duration.

Figure 4.1 shows the average time taken for the agents to initialize for each access control model. Both Alice and QualiChain agents took approximately 20 seconds to initialize with every access control model. The university agent had some time variations, mainly when publishing the credential schema on the blockhchain. Since the schemas had the same fields on every test, we can assume that the time discrepancy (17, 21, 26 and 24 seconds with ABAC, RBAC, RBAC With Resource Roles and ACL respectively) was due to the internet connectivity state and the communication between the university agent and the decentralized ledger.

Since the startup phase accounts for the majority of the duration of the total access control proccess, Figure 4.2 displays only the duration of the connect and access control phases. The connect phase is composed by three parts, Alice first connects to the University, then the University issues a verifiable credential to Alice and finally Alice connects to the QualiChain service provider. The access control phase encompasses the process of Alice requesting access to a resource, the verifiable presentation request sent to Alice in return, the VP sent from Alice to QualiChain and lastly the final evaluation of the access control request. The Connect phase took an average total of 2.1 seconds with every access control model. The full Access Control phase took 1.7 (ACL), 2 (RBAC-WRR), 2 (RBAC) and 1.6 (ABAC) seconds. Here, the process of Alice building the verifiable presentation as a reply to the VP request sent by the QualiChain occupied the greatest part of this phase duration.

When we take into account the total duration of the access control process of our baseline (ABAC) we can accept the small variation (max. one hundredth of a second) between each model's access control phase, since each of them utilizes different security policies and policy configurations. As a result, the evaluation of the access control requests had small duration variations.

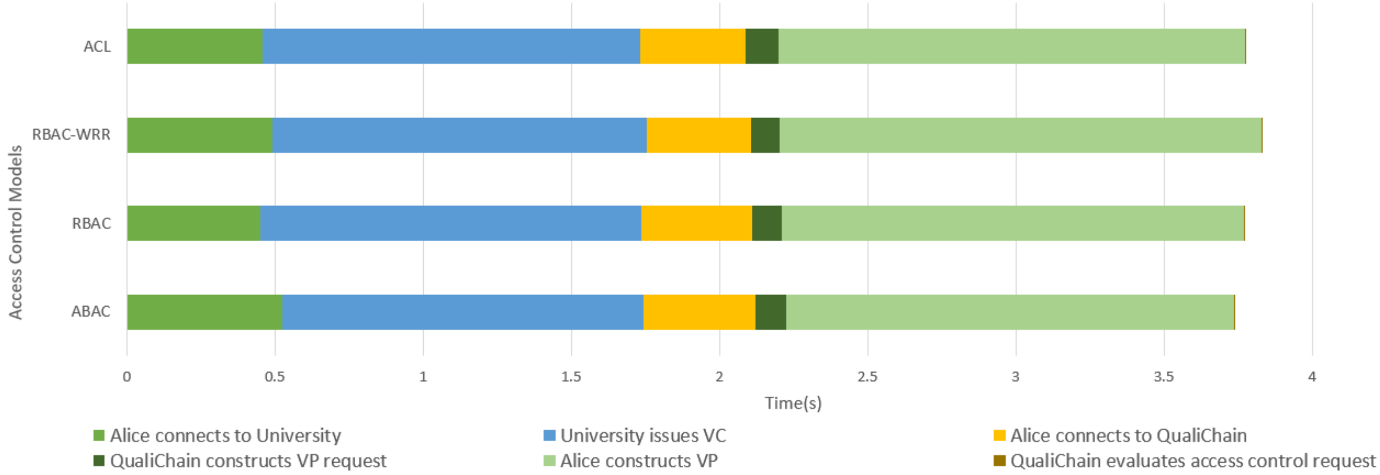

Figure 4.2: AC Process Duration.

Chapter 5

Conclusions

In this document, we propose Extended Self-Sovereign Identity Based Access Control model, an extended version of the SSIBAC model. Since the SSIBAC leverages blockchain technology and a self-sovereign identity solution but was developed only with the ABAC model instantiated, we developed a simple version with the RBAC, RBAC With Resource Roles and ACL models adapted to decentralization in addition to the ABAC model already implemented. This way, organizations can opt to use extSSIBAC when ABAC does not suit their needs.

5.1 Achievements

Our solution employs the Casbin library as it supports all the models mentioned in this project. Hyperledger Aries offers the infrastructure of our architecture's agents and Hyperledger Indy is used as our decentralized ledger. Overall, the main access control process takes on average 1.8 seconds with every access control model (1.6 with ABAC, 2 with RBAC, 2 with RBAC WRR and 1.7 seconds with ACL model). We consider this duration acceptable when dealing with solutions that must offer private data protection when controlling access to their resources.

extSSIBAC has some limitations when trying to protect systems against user behaviour profiling, since when using the RBAC and ACL models, our solution's access control process deals not only with DIDs but also with attribute roles and usernames.

5.2 Future Work

For future work, our solution can be implemented with concurrent programming in order to support multiple requests at the same time. Also, ideally, extSSIBAC could be adapted to use a decentralized solution for the access control models and security policies instead of relying on a single programming library to support their implementation.

Bibliography

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

[All16] Christopher Allen. The path to self-sovereign identity. *Life with Alacrity*, 2016.

[ANP⁺03] Anne Anderson, Anthony Nadalin, B Parducci, D Engovatov, H Lockhart, M Kudo, P Humenn, S Godik, S Anderson, S Crocker, et al. extensible access control markup language (xacml) version 1.0. *OASIS*, 2003.

[ari] aries big picture. `https://github.com/hyperledger/aries-cloudagent-python/blob/main/docs/GettingStartedAriesDev/AriesBigPicture.md`.

[AW18] Andreas M Antonopoulos and Gavin Wood. *Mastering Ethereum: building smart contracts and dapps*. O'Reilly Media, 2018.

[Bar97] John Barkley. Comparing simple role based access control models and access control lists. In *Proceedings of the second ACM workshop on Role-based access control*, pages 127–132, 1997.

[BE17] Vitalik Buterin and Ethereum team. Ethereum - a next-generation smart contract and decentralized application platform. White Paper, 2014-17.

[BPP⁺20] Rafael Belchior, Benedikt Putz, Guenther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. Ssibac: Self-sovereign identity based access control. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1935–1943. IEEE, 2020.

[cas] Casbin library. `https://casbin.org/docs/en/supported-models`.

[CKY18] Jason Paul Cruz, Yuichi Kaji, and Naoto Yanai. Rbac-sc: Role-based access control using smart contract. *Ieee Access*, 6:12240–12251, 2018.

[CS17] Miguel Pupo Correia and Paulo Jorge Sousa. *Segurança no Software.* FCA, 2nd edition, 2017.

[DLZ+18] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1366–1385, 2018.

[doc] Docker container. `https://www.docker.com/resources/what-container`.

[DP18] Paul Dunphy and Fabien AP Petitcolas. A first look at identity management schemes on the blockchain. *IEEE Security & Privacy*, 16(4):20–29, 2018.

[DRFM18] João Pedro Dias, Luís Reis, Hugo Sereno Ferreira, and Ângelo Martins. Blockchain for access control in e-health scenarios. *arXiv preprint arXiv:1805.12267*, 2018.

[GA19] Geoffrey Goodell and Tomaso Aste. A decentralised digital identity architecture. *Available at SSRN 3342238*, 2019.

[H+20] David Huseby et al. Hyperledger Aries. `https://wiki.hyperledger.org/display/ARIES/Hyperledger+Aries`, may 2020.

[HFK+13] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

[Ise99] David Iseminger. *Active directory services for Microsoft windows 2000.* Microsoft Press, 1999.

[KCE+15] Harry A Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*. Citeseer, 2015.

[LCO+16] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269, 2016.

[MMR17] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP international conference on distributed applications and interoperable systems*, pages 206–220. Springer, 2017.

[Pec17] Morgen E Peck. Blockchains: How they work and why they'll change the world. *IEEE spectrum*, 54(10):26–35, 2017.

[Sab17] Markus Sabadello. A universal resolver for self-sovereign identifiers, 2017.

[San98] Ravi S Sandhu. Role-based access control. In *Advances in computers*, volume 46, pages 237–286. Elsevier, 1998.

[SS94] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[SS18] Chinmay Saraf and Siddharth Sabadra. Blockchain platforms: A compendium. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–6. IEEE, 2018.

[SVSG20] Diogo Serranito, Andre Vasconcelos, and Miguel Correia Sergio Guerreiro. Blockchain ecosystem for verifiable qualifications. In *2nd Conference on Blockchain Research Applications for Innovative Networks and Services BRAINS 2020 September 28 – 30, 2020 Paris, France*, 2020.

[TR16] Andrew Tobin and Drummond Reed. The inevitable rise of self-sovereign identity. *The Sovrin Foundation*, 29(2016), 2016.

[Und16] Sarah Underwood. Blockchain beyond Bitcoin. *Communications of the ACM*, 59(11):15–17, 2016.

[uub18] Hyperledger Indy Revision 6b6c21b2. Indy Walkthrough. `https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/getting-started/indy-walkthrough.html`, 2018.

[W3C] W3c on vcs. `https://www.w3.org/TR/vc-data-model/#what-is-a-verifiable-credential`.

[W3C20] W3C. Decentralized Identifiers. `https://www.w3.org/TR/did-core/`, April 2020.

[XCBC18] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. BlendCAC: A blockchain-enabled decentralized capability-based access control for IoTs. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1027–1034. IEEE, 2018.

[YL20] Xiaohui Yang and Wenjie Li. A zero-knowledge-proof-based digital identity management scheme in blockchain. *Computers & Security*, 99:102050, 2020.

[ZKS$^+$18] Yuanyu Zhang, Shoji Kasahara, Yulong Shen, Xiaohong Jiang, and Jianxiong Wan. Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, 6(2):1594–1605, 2018.