DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

BLOCKCHAIN-BASED SHARING SERVICES FOR SMART CITIES

by

Aslı BAHCE

June, 2023

İZMİR

BLOCKCHAIN-BASED SHARING SERVICES FOR SMART CITIES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering

by
Aslı BAHCE

June, 2023
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"BLOCKCHAIN-BASED SHARING SERVICES FOR SMART CITIES"** completed by **ASLI BAHCE** under supervision of **ASSOC. PROF. DR. SEMİH UTKU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Semih UTKU

Supervisor

Thesis Committee Member

Thesis Committee Member

Examining Committee Member

Examining Committee Member

Prof. Dr. Okan FISTIKOĞLU
Director

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Dr. Semih UTKU, for his invaluable guidance, encouragement, and support throughout my research. His expertise and insight have been instrumental in shaping my thesis, and I am grateful for the time and effort he has dedicated to helping me achieve my goals.

I would also like to thank to Prof. Dr. Alp KUT and Assoc. Prof. Dr. Yusuf Yeşilce for their unwavering support and encouragement throughout my academic journey. Their expertise, advice, and feedback have been invaluable in helping me develop my research skills and achieve my academic goals.

I would also like to extend my thanks to my colleague and friend, Recep Yıldız, for his support throughout the thesis writing process. His constructive feedback, suggestions, and willingness to help have been invaluable in shaping my work, and I am grateful for his presence as a friend and collaborator.

I would like to express my deep appreciation to my friends and family for their unwavering support and encouragement throughout my studies. Their love and encouragement have been a source of strength and inspiration, and I am grateful for their belief in me.

Finally, I would like to thank all the participants who generously gave their time and shared their experiences with me, making this research possible. Their contributions have been invaluable, and I am grateful for their willingness to participate in my study.

Aslı BAHCE

BLOCKCHAIN-BASED SHARING SERVICES FOR SMART CITIES

ABSTRACT

The concept of Self-Sovereign Identity (SSI), where we can store our personal information and documents securely, as proven by the relevant authorities, and share them with the people we want, has become widespread as the personal secure data storage method that we need most in recent years. We have to physically carry our evidence-based identities obtained from events or institutions such as transportation, event entrances, museum visits, and gymnasiums. In the process of verifying this information, we may need to prove our identity physically or show documents. Keeping our identity, age, health status, educational status, and many other personal data inaccessible to others and without questioning the accuracy when we share them also restricts their use in different areas. Sharing the data depending on the consent of the people and presenting the data in an accurate and reliable way has become one of our most important needs today. Many manually managed processes have now become automated. Technological solutions have made it possible for people to access different identities from a single environment (transport cards, museum entrance cards, etc.) with this automation created. With the SSI concept, it has been possible to transform different identities into a structure that we keep in our own wallet (mobile device). With the concept of SSI, it has become possible for us to save our data as distributed, inaccessible, unalterable, and incorruptible. In addition, with the possibility of public access to the information with the consent of the people, it is possible to instantly share the information with the people/institutions we want. In this study, mobile application processes created on SSI infrastructure will be explained. It will be possible to add different institutions to the developed system and to identify the added institutions to the users. The identities given by the institutions will be able to be proven by different institutions. Thus, the operating structure of this system will be transferred with a verifiable and reliable infrastructure. Our study, it is aimed to contribute to the general operating processes of those who will develop the SSI infrastructure. In order to develop such a system, it has been determined that ACA-Py is the most suitable infrastructure framework. After comparing the efficacy of Askar

and Indy for the selection of the wallet to be used, it was determined that the Askar wallet would be used in the system.

Keywords: Self-sovereign identity, blockchain, reliability, consistency, mobile app.

AKILLI ŞEHİRLER İÇİN BLOCKCHAIN TABANLI PAYLAŞIM HİZMETLERİ

ÖZ

Kişisel bilgilerimizi ve belgelerimizi güvenli bir şekilde, ilgili otoritelerce kanıtlanmış olarak, saklayabileceğimiz ve istediğimiz kişilerle paylaşabileceğimiz Kişi Hakimiyetinde Kimlik (SSI- Self Sovereign Identity) kavramı, son yıllarda en çok ihtiyacımız olan kişisel güvenli veri saklama yöntemi olarak yaygınlaşmaktadır. Ulaşım, etkinlik girişleri, müze ziyaretleri, spor salonları gibi etkinlikler veya kurumlardan elde ettiğimiz kanıta dayalı kimliklerimizi fiziksel olarak taşımak zorunda kalıyoruz. Bu bilgilerin doğrulanması sürecinde de fiziksel olarak kimliğimizi kanıtlamak veya belgeler göstermemiz gerekebilmektedir. Kimliğimizi, yaşımızı, sağlık durumumuzu, eğitim durumumuzu ve daha birçok farklı kişisel veriyi başkaları tarafından ulaşılamaz ve paylaştığımızda doğruluğu sorgulanamaz şekilde barındırmak farklı alanlarda kullanımını da kısıtlamaktadır. Kişilerin rızasına bağlı olarak verinin paylaşılması verinin doğru ve güvenilir bir şekilde sunulması günümüzde en önemli ihtiyaçlarımızdan biri haline gelmiştir. Manuel yönetilen birçok süreç artık otomasyon haline gelmiştir. Teknolojik çözümler; oluşturulmuş bu otomasyonlar ile kişilerin farklı kimliklere tek bir ortamdan (ulaşım kartları, müze giriş kartları, vb.) erişebilir olmasını mümkün hale getirmiştir. SSI kavramı ile farklı kimlikleri kendi cüzdanımızda (mobile cihaz) sakladığımız bir yapıya dönüştürmek mümkün olmuştur. SSI kavramı ile verilerimizi dağıtık, erişilemez, değiştirilemez ve bozulamaz olarak kaydetmemiz mümkün hale gelmiştir. Ayrıca bilgilerin kişilerin rızası ile public olarak erişilmesi olanağı ile istediğimiz kişi/kurumlarla bilgilerin anlık olarak paylaşılabilmesi sağlanabilmektedir. Bu çalışmada, SSI altyapısını geliştirecek olanların genel işleyiş süreçlerine katkıda bulunmak amaçlanmaktadır. Böyle bir sistem geliştirmek için ACA-Py'nin en uygun altyapı çerçevesi olduğu belirlenmiştir. Cüzdanın seçimi için Askar ve Indy'nin etkililiği karşılaştırıldıktan sonra, sistemin içinde Askar cüzdanının kullanılmasına karar verilmiştir.

Anahtar kelimeler: Kişi hakimiyetinde kimlik, blok zinciri, güvenilirlik, tutarlilik, mobil uygulama

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Depending on the developing and changing living conditions, we use various materials, mobile applications, and identities for many different purposes today. Although this situation causes us to carry a lot of things physically; it has made it difficult for us to govern and to prove our identity to the authorities. Although this problem is seen in many areas (Waraporn et al., 2009), if we reduce the sample, it is encountered in city life and different areas. We have to have a different card or mobile application in many areas such as transportation, event entrances, museum visits, and gyms (Pesonen & Horster, 2012). In addition, we may need to show documents to prove our identity.

Self-Sovereign Identity (SSI) as an idea, where we can store our personal information and documents only in a non-public way, as proven by the relevant authorities, and share them with anyone we want, has become widespread as the secure data storage method we need most in recent years (Shuaib et al., 2022). Keeping our identity, age, health status, educational status, and many other personal data inaccessible and unquestionable when we share it has become one of our most significant needs in today's world where data storage and presentation become difficult.

Today, when city governments make serious investments and breakthroughs in technology (Joghee et al., 2020), smart cities have begun to form (Law & Lynch, 2019). Many manually managed processes have now become automated. These processes required people to be included in different identities (transport cards, museum entrance cards, etc.) with these automations. With the SSI concept, it has been possible to transform these different identities from many physical instruments into a structure where we store data in our own wallet (mobile application). Thus, individuals will be able to present their data, which they have stored as proven by the relevant authorities, to any person/institution at any time. People will be hosting their data in a way that cannot be shared without their permission.

If we explain the operation of the system with an example; at an event entrance in smart cities using the proposed system; an event is planned for those over the age of 18 and where “Exclusive Card Owner” users can log in. Smart city residents, who have this data under their own control in the mobile application, will be capable of entering the event by proving both their membership type and age at once, with a simple application at the event entrance. Whereas, in the classical system, he had to show the event entrance card, prove that this card belongs to him, and present the document showing that he is over 18.

So how can this system be made more secure and unbreakable? The basis of these questions has been resolved with the blockchain infrastructure. With the concept of blockchain, it has become possible for us to save our data as distributed, inaccessible, unalterable, and incorruptible.

In the proposed system that we developed with SSI on the blockchain, we can solve this problem much more securely and effectively. In this case, our data cannot be accessed publicly, stored as evidenced by the authorized authorities, cannot be changed, or corrupted, and can be shared instantly with the people/institutions we want. With this system, which will greatly facilitate the lives of both smart city residents and institutions managed by smart cities in different areas, many problems that make our lives easier, such as saving time, preventing erroneous discharges, transporting/managing a large number of physical materials, will be solved.

With this study, an end-to-end application has been developed with the concept of self-sovereign identity. Users can carry their identities, which they hold for different purposes, on their mobile devices as safe, provable, and incorruptible instead of physically carrying them in their wallets, and institutions can see and accept these identities as proof.

This thesis is organized as follows: it comprises six chapters followed by subsequent sections.

Chapter two presents self-sovereign identity and blockchain in detail, including relevant studies, literature reviews, and field research.

Chapter three offers detailed explanations and definitions of self-sovereign identity, blockchain technology, and their intersection, providing comprehensive insights into the subject matter.

Chapter four presents a description of a scenario of the system, the system architecture, workflow description based on the Indy framework and the application of this workflow to the ACA-Py framework, technologies that have been used in the system, and the application explanation with application screens.

In Chapter five, the study's findings and decisions made during the initialization of the system are presented.

Chapter six concludes the thesis by providing a summary of the proposed tool, key findings, conclusions, and future directions for research.

CHAPTER TWO
LITERATURE REVIEW

Despite the fact that Self-Sovereign-Identity is a very old notion, there have been substantial studies on the topic since 2016 according to the literature (Soltani et al., 2021). It has been determined through research that the majority of published works are concerned with blockchain technology. Particularly, anonymous and personal encryption technologies as well as the anonymity capabilities provided by blockchain technology are advantageous for the practical application of this notion (Li et al., 2020). When the studies on the subject in the field of blockchain are examined, it is seen that since 2016, extensive studies have been put forward by the Sovrin community on this subject. In these studies, different technical solutions have been presented under the Hyperledger community in order to meet the Self-Sovereign-Identity and Zero-Knowledge-Proof techniques (Kulabukhova, 2019). Hyperledger-Fabric and Hyperledger-Indy are the most up-to-date technologies used in blockchain, mainly in the areas of Self-Sovereign-Identity and Zero-Knowledge-Proof.

In addition, the Verus blockchain solution also produces solutions on Self-Sovereign-Identity, but it is at a very beginner level compared to the Hyperledger infrastructure in terms of supporting community and development level.

When the solution technologies related to Self-Sovereign-Identity used in the study are examined, there are commercial Evernym products and non-commercial open-source Aries solutions. While Evernym commercially develops products on the Hyperledger-Indy library, the Aries community develops open-source solutions using Hyperledger-Indy libraries on different infrastructures. The Canadian British Columbia government is actively offering Self-Sovereign-Identity using these solutions developed by the Aries community.

When the studies in this field are examined; first of all whether blockchain infrastructure is required for SSI is investigated. In their study, Bokkem et al. (2019) examined the evolving landscape of Self-Sovereign Identity solutions and evaluated

their implementations using various criteria to assess the necessity of blockchain technology in this domain. While they concluded that blockchain technology is not an absolute prerequisite for a Self-Sovereign Identity solution, their research revealed that blockchain-based solutions outperformed other solutions in terms of meeting multiple criteria. Therefore, the study suggests that blockchain-based solutions may be better equipped to meet the requirements of a Self-Sovereign Identity solution on average, compared to other alternatives.

Due to the sensitive nature of the data, the majority of Self-sovereign identity use cases target healthcare systems. Numerous studies examine whether or not SSI applies to the actual world. Here are some studies demonstrating that blockchain-based SSI is the most appropriate and trustworthy solution for identity management and user control of data. The research of Houtan et al. (2020) found that blockchain technology, with its distributed and decentralized structure, can offer significant benefits in healthcare. Blockchain has the capacity to transform the industry by giving patients authority over their own data and self-sovereign identity. However, the researchers acknowledge that the development of purely decentralized Healthcare Information Systems (HIS) presents significant technical and architectural challenges. To address these challenges, the study examines the design tradeoffs and evaluates the current state of the art in HIS design.

The results indicate that BC-powered EHR and PHR systems are crucial in realizing a decentralized and self-governing healthcare ecosystem. While prior research, as noted by Akande (2018), has primarily focused on exploring the technological capabilities of such solutions, there remains a gap in understanding their broader effects on the healthcare ecosystem and the complete range of related activities.

The push for distributed trust through the elimination of intermediaries and decentralization is the core principle driving the development of pure blockchain solutions. However, the most reliable existing solutions, which are hybrid blockchain architectures, have not fully met the requirement for patient protection of privacy and

the forgotten right. Although this issue has been addressed by current solutions, none have offered a conclusive solution or proof of concept.

Shuaib et al. (2021) argue that blockchain adoption over the traditional client-server paradigm is motivated by the desire to achieve greater productivity, stability, reliability, and timeliness within statuary systems. This is because blockchain technology is dispersed and decentralized, which renders middlemen unnecessary. In healthcare, the confidentiality of patients' personal information is paramount. By granting patients self-sovereign identity, they are empowered with complete control over their identity and information.

This study proposes that by enabling patients to control their personal information, self-sovereign identification can improve compliance with legislation and change data processing procedures in healthcare. By providing patients with control over their personal data, healthcare providers can become more service-oriented and develop new products and applications. The cost-effectiveness of self-sovereign identities can be seen in their ability to eliminate inefficient onboarding processes and provide fast data access and authentication. Overall, self-sovereign identity appears to have great potential for improving the efficiency and effectiveness of healthcare services.

Moreover, the reliability and accuracy of data in food supply chains are critical. In their research, Cocco et al. (2021), present a practical use case that demonstrates the need for a system that ensures full visibility of process and food certifications. Currently, accredited certification bodies (issuers) issue these certifications, which are maintained in paper format by various supply chain participants (holders). The researchers propose a Self-Sovereign Identity-based model that aims to address the storage and accessibility of these certifications, as well as appropriate eligibility verification. The certification system operates on the assumption that the certifying body will issue a certificate and store it on IPFS. Only when a product or process user (holder) seeks quality certification, does the system store key information on the blockchain. The blockchain is used to store certification verification details, and

certificate returns and validation are automated, providing a more efficient and reliable system for managing food supply chains.

Stockburger et al. (2021) suggest a second use case for the SSI application: public transportation. The research investigates the application of SSI management in the public transportation sector, which encompasses multiple operators and countries. In particular, the study looks into the use of a decentralized identity management system built on blockchain technology and using the SSI framework to offer high levels of security. The authors created a low-fidelity prototype utilizing the Hyperledger Indy blockchain to show people how to manage their identification credentials more efficiently. The prototype presents the EMC, a verifiable credential designed according to W3C global technical standards, which can be used as an identity card on European public transportation. Standardized issuing and validation of the EMC would lower identity management costs and eliminate third-party vendor lock-in. The decentralized method proposed would also improve the market by allowing users to port their data to the best option without changing system architecture, utilizing self-sovereign identity-embedded principles.

In their paper, Liu et al. (2020) explore the emerging field of self-sovereign identity (SSI) systems and the ways in which blockchain technology can be leveraged to support these systems. One key contribution of their work is the development of a set of 12 design patterns for blockchain-based SSI systems. These patterns are meant to aid architects in comprehending and applying SSI principles to system design.

The authors categorize these patterns into three groups based on the lifecycles of the three primary objects involved in self-sovereign identity: key management patterns, decentralized identifier management patterns, and credential design patterns. Key management patterns focus on the generation, storage, and use of cryptographic keys, which are central to ensuring the security and privacy of SSI systems. Decentralized identifier management patterns address the management of unique identifiers that are not tied to any particular central authority or organization. Finally, credential design patterns focus on the design and issuance of verifiable credentials,

which allow individuals to prove their identity or certain attributes of their identity to others.

The use of an identifier registry as a smart contract to preserve ties between an identifier and the location of associated off-chain identity data elements is a crucial design pattern. This smart contract maps each identifier to a storage place (such as IPFS or Dropbox) for identity attributes, which only the identifier owner can change. This approach helps to ensure that individuals have control over their own identity data and can selectively share it with others as needed.

Additionally, the authors include identifier documents that provide instructions for using each identifier, such as public keys for digital signatures or service endpoints for engagement. These documents help to ensure that individuals can effectively use their identifiers to prove their identity or access services.

Overall, the work of Liu et al. (2020) provides valuable insights into the design and implementation of blockchain-based SSI systems. By identifying key design patterns and providing detailed guidance on their use, the authors offer a roadmap for architects and developers seeking to build secure, decentralized, and user-centric identity systems.

There are studies concerning claim-based models. Mühle et al. (2018) distinguish two main approaches regarding this topic: the Identifier Registry Model and its extension, the Claim Registry Model. They claim they will provide a more unified view of verifiable claims regarding blockchain-based SSI and elucidate terminology distinctions. On-chain and off-chain storage solutions for verifiable claims are presented along with their advantages and disadvantages. The Identity Registry Model and the Claim Registry Model have both extended the concept of verifiable claims. These decentralized registries were made possible by blockchain technology, and while decentralized storage is optional, it can also be decentralized. This leaves as the only centralized entities in the system the claim-issuers and their position of trust. On the other hand, Baars (2016) has investigated the possibility of self-sovereign identity,

which allows users to control their digital identity. They concluded that in most systems, users cannot store their data, with only one decentralized system currently in existence, but lacking widespread adoption. Currently, only one decentralized system exists, but it has not yet garnered widespread adoption. In their case study, they explored a solution for attribute exchange, that allows KYC attribute exchange after completing Customer Due Diligence, such as when opening a bank account. This simplifies client onboarding by eliminating duplicate documentation and can be used by insurance companies and mortgage lenders. The key research issue was: How to design decentralized identity management architecture so entities can exchange attributes and verify claims without a central authority? A blockchain-based Decentralized Identity Management System (DIMS) proof-of-concept results from the sub-research topic. A storage tier ensures that claims are not reputable and by whom. Blockchain technology replaces the need for a central authority with cryptographically secured consensus methods.

In another study on claim-based models, Stokkink & Pouwelse (2018) propose a solution that obtains legally valid identity at the passport level, which requires third-party truth attestations. The authors demonstrate that the claimed model is independent of the blockchain structure and proof methods used. The suggested remedy is the first decentralized digital passport in history as well as a genuine permissionless peer-to-peer identity commons. The authors anticipate sub-second performance for both the creation and verification of provable claims for real-world deployment. The study found that claim verification is faster than claim creation, which makes the technology user-friendly and encourages its usage.

Kondova & Erbguth (2020) provide an overview of the Sovrin self-sovereign identity platform, which runs on the public permissioned Hyperledger Indy blockchain, as well as uPort and Jolocom, which operate on the public permissionless Ethereum blockchain. The authors also examine the compliance of blockchain-based identity systems with the European Union's General Data Protection Regulation (GDPR). They discuss which components of self-sovereign identity data qualify as personal data, how the GDPR applies to them, and who acts as a controller under the

regulation. By removing central authorities that control issued credentials, SSI offers greater privacy protection for individuals. The authors suggest that SSI can meet GDPR requirements by safeguarding data subjects' privacy while ensuring compliance with the regulation.

Naik & Jenkins (2020) made a comparison of two SSI solutions: uPort and Sovrin. Using the proposed specifications, they evaluated uPort and Sovrin SSI and highlighted their strengths and weaknesses. In addition, they adhere to recuperation, cost-free, security, privacy, protection, and accessibility standards. Nevertheless, the degree of support for each specification varies between the two systems. While Sovrin currently provides more security and privacy features, uPort's design architecture is straightforward and user-friendly.

In their study, Liu, Lu, Paik, Xu, et al. (2020) proposed a comprehensive SSI platform architecture that supports the design pattern like a service (DPaaS) concept, which they consider essential for SSI application development. The authors identified the lifecycles of three main SSI objects (keys, identifiers, and credentials) and presented a range of SSI services, including management of key and credential verification, supported by two classifications of operations: design pattern services and regular services. The researchers deployed a sample of their platform on Alibaba Cloud, using 16 GB of random-access memory, and a 40-GB disk, four virtual CPUs. API requests were made in batches of 20 calls each, and the Java Meter ran for an hour. According to the results, the number of transactions for registration was approximately 70, while the number of transactions for time-limited access remained around 200. Additionally, they used 1,000 credentials to test the anchoring to the blockchain 1,000 times, and the response time was approximately 0.284s. Overall, the authors found that their platform performed well enough to be used at the company level. The study's findings contribute to the development of SSI platforms that can support a wide range of SSI services and provide a DPaaS approach, which can lead to better SSI application development.

CHAPTER THREE
DEFINITIONS

3.1 Self-Sovereign Identity

3.1.1 What is Self-Sovereign Identity

Self-sovereign identity (SSI) is a word used to define the digital movement that acknowledges a person's right to own and govern their identity without administrative interference. SSI enables people to interact with the same level of freedom and trust in the digital world as they do in the offline one. SSIs are a type of digital identity that is administered in a decentralized way. Users can self-manage their digital identities by utilizing this technology, which eliminates the need for them to rely on third-party suppliers to store and centrally manage their data (Sovrin Foundation, 2018).

3.1.2 How Does SSI Work

In the analog world, proving or confirming one's identity typically involves presenting an ID card to the relevant authorities, such as when visiting a bank, or government agency or checking into a hotel. This allows us to retain control over our data as the ID card goes back into our wallet after use.

However, when it comes to digital identification, users have little control over their data and how it is used. Whether shopping online, communicating with banks or government agencies, or interacting with apps and websites, users are forced to reveal personal information that is often stored in third-party databases. Even logging into a website or app with social media accounts exposes users to data collection, storage, and tracking.

The need for a trustworthy and reliable system that enables individuals, companies, and machines to interact and network securely and digitally has led to the emergence of self-sovereign identities (SSIs). With the SSI system, users can utilize their digital wallets and authenticate their own identity using the credentials they have been issued,

thus eliminating the need to surrender control of personal information to third-party databases.

SSI is called self-sovereign because it puts each person in control of their own identity, making them their own sovereign nation. Users can now manage their information and relationships independently, and their digital existence is no longer tied to any organization. This gives them the power to control their data, keep it secure, and prevent identity theft by hackers or unauthorized third parties. With SSI, users can access new goods and services with ease, without the fear of losing control over their data.

3.2 Blockchain

3.2.1 What is Blockchain

Blockchain is a database. Traditional databases can be accessed by users with permission, and they can filter and manipulate data. Blockchain differs from these at this point. Depending on the blockchain types, the user group accessing the data may vary, but the data cannot be manipulated. Blockchain can be defined as an immutable list of data. It consists of blocks. When data is added to a block, it cannot be changed later. Each block is linked to the one before it using a cryptographic key. Thanks to this, the blockchain can be traced, but it cannot be broken (Akash Takyar, 2023).

3.2.2 Properties of Blockchain

Blockchain technology provides a highly secure and transparent method of storing and transmitting information. It is distinguished by several key features that ensure data security, transparency, immutability, decentralization, and consensus. To maintain security, the blockchain uses a combination of cryptography and network structure. Cryptography is frequently used to sign data and demonstrate that the owner of the funds approves of a transaction. The non-centralized network structure removes central points of failure, making it impractical and expensive for hackers to compromise the network.

Transparency is another key feature of blockchain technology. All computers on the network can view transaction records, and transactions are available to anybody with an internet connection.

Immutability is a critical feature of blockchain technology. The information stored on the blockchain cannot be changed or deleted, giving a high level of confidence in the data.

Decentralization is another key feature of blockchain technology. Information is transferred and validated by nodes located throughout the network instead of a central server, making the system more egalitarian and less susceptible to single points of failure.

Consensus is the final key feature of blockchain technology. Every node on the blockchain network participates in consensus, which ensures the accuracy and reliability of the data that is recorded.

3.2.3 Structure of Blockchain

Blockchain consists of blocks that are sequentially connected to each other. Transactions are stored in these blocks. The blocks that make up the blockchain consist of 2 parts, the header, and the body. Figure 3.1 shows the block's organizational structure.

The block header consists of 6 parameters. These are version, previous hash, difficulty, timestamp, nonce, and Merkle root hash. Version specifies the software and protocol version on the blockchain. In this way, updates in the blockchain can be tracked. The previous hash is the 256-bit Merkle root hash of the previous block. Difficulty also called nBits is a unit of measure indicates that how difficult it is to find a hash under a given target. Timestamp indicates the creation time of the block. It is an approximate value and is stored as seconds from Unix Epoch. Nonce (number only used once), is the random value used when calculating the hash value of the block.

Miners must find this value to solve the block. The Merkle root hash is the hash value generated from all transactions in the block.

The block body, which is the core component of a blockchain block, comprises a transaction counter and a collection of individual transactions. Based on the size of each transaction and the overall block size restriction imposed by the network's protocols, the maximum number of transactions that may be supported in a single block varies.

As each transaction carries a unique set of data, including sender and recipient addresses, transfer amounts, and transaction fees, the total size of the block is directly proportional to the number of transactions it contains. Therefore, network participants often seek to optimize the size of transactions and the overall block size limit to ensure smooth and efficient transaction processing while minimizing the resource requirements of the network (Zheng et al., 2017). The size of data to be stored in blocks can be customized. For example, the number of items in the block can be limited or the block size limited. It depends on the design of the blockchain.

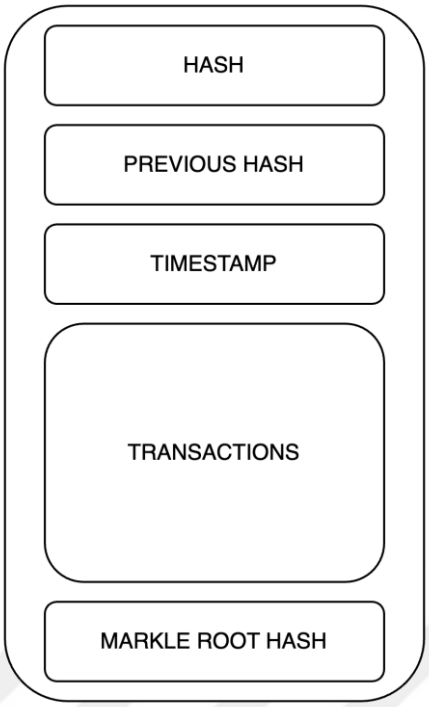


Figure 3.1 Structure of a block

3.2.4 Types of Blockchain

The blockchain is a technology that enables secure transactions and information exchange between nodes in a network while ensuring that the records remain unchangeable. Different blockchains use different structures and configurations to achieve this objective. Blockchain technology can be categorized into two types: public and private.

Public blockchains are open to anyone with an internet connection and allow all stakeholders to view and verify transactions on the network (Guegan, 2017). These networks are completely transparent and distributed, with each authorized node having a copy of the blockchain data. However, due to the large number of nodes, transaction processing can be relatively slow, limiting scalability (Chauhan et al., 2018). Proof of work consensus is used to ensure trust, but this requires significant time and energy.

Private blockchains, on the other hand, are restricted to verified participants only, ensuring that the network is secure and accessible only to authorized stakeholders.

Since all participants are already verified, there is no need for proof of work, resulting in faster transaction processing and higher scalability (Pongnumkul et al., 2017). Permissions are designated to participants according to their roles and responsibilities, which may vary within the network. Identity management can be centralized, but this may negatively impact security.

In summary, both public and private blockchains serve the same purpose of secure transactions and information exchange. However, they differ in terms of their accessibility, transparency, scalability, consensus mechanisms, and identity management. Each type of blockchain has its advantages and disadvantages, and the choice of which one to use depends on the specific needs of the users.

Public and private blockchains were compared in advantages and disadvantages. A comparison of blockchain types is given in Table 3.1.

Table 3.1 Advantages and disadvantages of public and private blockchain

	Public Blockchain	**Private Blockchain**
Advantages	Open to all, Fosters trust among users, Provides robust security	Provides rapid transaction processing, Easily scales to handle high volumes, Achieves energy efficiency
Disadvantages	Low transaction throughput, Limited scalability, High energy consumption	Centralized to some extent, The lower level of transparency, Limited immutability

3.2.5 Consensus Mechanisms

Blockchains rely on consensus mechanisms to reach an agreement on the validity of transactions and maintain the state of the system. The consensus mechanism is an essential feature of any blockchain, as it ensures that all nodes in the network have a consistent view of the ledger. There are several consensus mechanisms in use, but the three most common are proof of work, proof of stake, and practical Byzantine fault tolerance.

Proof of Work (PoW) is a consensus mechanism that relies on miners to validate new transactions. Miners compete to solve a complex mathematical problem, and the first to solve it is rewarded with newly minted cryptocurrency. This mechanism is used by Bitcoin and Ethereum, among other blockchains. However, PoW is known for being slow and energy-intensive, as the mining process requires a significant amount of computational power (Baliga, 2017).

Proof of Stake (PoS), on the other hand, is a consensus mechanism that allows users to validate transactions by staking their cryptocurrency holdings. Validators are chosen randomly based on the amount of cryptocurrency they have staked, and their chances of being selected increase with the amount of cryptocurrency they stake. This mechanism is considered more energy-efficient than PoW, as it does not require extensive computational power (Panda et al., 2019).

Practical Byzantine Fault Tolerance (pBFT) is a consensus mechanism that is commonly used in permissioned blockchains. In a distributed network, some nodes may fail or be malicious, which can lead to inconsistencies in the ledger. pBFT ensures consensus by requiring at least two-thirds of the network's validators to agree on the validity of a transaction before it is added to the ledger. This mechanism is known for being fast and efficient, but it is less decentralized and less transparent than other consensus mechanisms (Baliga, 2017).

In summary, the consensus mechanism is a critical component of any blockchain, as it ensures that all nodes have a consistent view of the system. PoW, PoS, and pBFT are the most commonly used consensus mechanisms, each with its own strengths and weaknesses. While PoW is known for being slow and energy-intensive, PoS is more energy-efficient, and pBFT is fast and efficient but less transparent and decentralized.

3.3 Self-Sovereign Identity Using Blockchain

3.3.1 Application of SSI on Blockchain

Self-Sovereign Identity (SSI) is a revolutionary concept that puts the control of personal identity back in the hands of individuals. Instead of relying on third-party servers to store and verify identity information, SSI leverages the power of blockchain technology to provide a decentralized, tamper-resistant, and secure system.

With SSI, users can store their public identifiers on a blockchain, which is maintained by many independent servers, making it highly resistant to tampering and hacking attempts. This eliminates the need for centralized servers controlled by third-party entities, as is the case with traditional login systems and passport proofs using video identification.

One of the key benefits of SSI is that users can create multiple identities that are encrypted and pseudonymized. This allows them to interact with different entities without revealing their true identities, thus enhancing privacy and security. Users can also selectively disclose their identity attributes, choosing which data they want to share under specific circumstances.

Blockchain technology has emerged as a key underlying infrastructure for Self-Sovereign Identity (SSI) solutions, with Ethereum blockchain being one of the popular choices. SSI solutions enable individuals to have full control over their personal data, establishing a secure and decentralized identity that is not reliant on traditional centralized authorities. A crucial component of SSI is the use of Decentralized Identifiers (DID), which allow individuals to create and manage their unique digital identity in a censorship-resistant environment.

In the DID ecosystem, a DID represents a unique personal identity that is created by a document issuer and stored on a blockchain. The DID contains an encrypted link to a signed document by an authority, which includes information related to service endpoints and authorized public keys. By using cryptographic techniques such as

digital signatures, individuals can securely and privately manage their identities, while providing verifiable proof of their identity when necessary.

There are several registered DID methods, including Hyperledger Indy, W3C, Ethereum, and Bitcoin, each with its unique features and benefits. Hyperledger Indy, for example, is designed to support complex identity use cases, while W3C focuses on interoperability and standardization. Ethereum, on the other hand, is known for its robust smart contract capabilities, making it suitable for building complex decentralized applications. Finally, Bitcoin, the first and most well-known blockchain, has also introduced its own DID method, which leverages the security and immutability of the Bitcoin blockchain to create decentralized identities.

Overall, the use of blockchain technology and DIDs in SSI solutions has the potential to revolutionize the way individuals manage and protect their personal data. With the ability to create and control their unique digital identities, individuals can better protect their privacy and security while also unlocking new opportunities for secure and decentralized digital interactions.

Verifiable credentials are another essential component of any SSI ecosystem. They attest to specific information related to a DID. For example, a digital driver's license is one type of verifiable credential. Government agencies can create digitally verifiable identity profiles or DIDs for beneficiaries, which contain important information such as name, social security number, date of birth, and age. These DIDs can be stored in a blockchain-based network, enabling service providers to authenticate beneficiaries quickly and securely.

During credential verification, the user may be required to authenticate the process using a one-time password. Service providers, such as banks, only see encrypted codes rather than personally identifiable information, thus enhancing privacy and security.

Overall, SSI offers a promising solution to the challenges of digital identity verification, providing a decentralized, secure, and flexible system that puts individuals in control of their personal data.

3.3.2 Hyperledger

Hyperledger is a collaborative open-source initiative hosted by The Linux Foundation. The initiative is focused on advancing cross-industry blockchain technologies and is supported by leaders from finance, banking, the Internet of Things (IoT), supply chains, manufacturing, and technology. Blockchain technology has the potential to revolutionize the way businesses and industries operate, and Hyperledger aims to support this transformation by creating distributed ledger frameworks, code bases, and open-source that support business transaction at the enterprise level.

A blockchain is a peer-to-peer distributed ledger that is supported by "smart contracts," among other helper technologies, and founded on consensus. These components can be used to create a new wave of transactional software that is based on accountability, transparency, and trust. They streamline business processes and legal constraints and can be thought of as an operating system for data-sharing networks, marketplaces, decentralized digital communities, and micro-currencies. This technology has the potential to make tasks less expensive and complicated in the real world.

To advance blockchain technology acceptance for use in commerce, an open-source collaborative software development approach is required. Hyperledger is a community of software developers who build blockchain frameworks and platforms with the aim of achieving transparency, longevity, interoperability, and support.

Hyperledger has several goals. First, it aims to create open-source, distributed ledger frameworks and code bases that support business transactions at the enterprise level. Second, it provides community-driven, open, and neutral infrastructure supported by technical and business governance that encourages innovation and fosters collaboration among developers. Third, it builds technical communities that develop

blockchain and shared ledger proofs of concept (POCs), use cases, field trials, and deployments. Fourth, it informs the public about the blockchain technology business opportunity and its potential advantages for many companies and sectors. Finally, Hyperledger encourages the creation of a community of communities using a toolkit approach with several platforms and frameworks. This allows for a broad range of use cases, creating a more comprehensive and flexible ecosystem for developers to build upon.

Hyperledger believes that blockchain technology has the potential to revolutionize the way businesses and industries operate. By fostering collaboration, innovation, and education, it aims to help bring this potential to fruition. The initiative is committed to developing and promoting cross-industry blockchain technologies to create a better future for businesses and society as a whole (The Linux Foundation, 2022).

3.3.3 Hyperledger Indy

Hyperledger Indy is a cutting-edge distributed ledger software that empowers digital identities, giving users the ability to control their own data across different administrative domains, applications, and other silos. It offers a flexible, modular ecosystem for creating secure, private, and powerful identities that can be used on their own or in conjunction with other blockchain networks.

Indy provides an innovative solution to the problem of centralization of identity by putting individuals in charge of their own privacy and disclosure. With Indy, people can create their own digital identities that are portable, interoperable, and secure. Indy's underlying technology enables a wide range of use cases and features, including connection contracts, revocation, unique payment workflows, asset and document management, innovative forms of escrow, curated reputation systems, and seamless integrations with other cutting-edge technologies.

Indy is built on open-source, distributed ledger technology, which is a decentralized form of database maintained by a pool of participants rather than a single, central authority. Data is stored redundantly across many nodes, and

transactions are validated by multiple machines using industry-standard cryptography. The system is designed with best practices in key management and cybersecurity, ensuring that the data is protected and secure. With Indy, users can rely on a trustworthy, public source of truth that is robust to system failure, resilient to hacking, and highly resistant to subversion by hostile entities.

In summary, Hyperledger Indy provides a powerful, modular ecosystem for creating secure and private digital identities. It enables users to take control of their own data, offering an innovative solution to the problem of centralization of identity. Built on open-source, distributed ledger technology, Indy offers a trustworthy, public source of truth that is resistant to subversion and hacking, making it a reliable platform for a range of use cases and applications.

3.3.4 Sovrin

Sovrin is a next-generation digital identity network that leverages the power of distributed ledger technology to provide a secure, private, and decentralized platform for identity management. It is an open-source project built on top of a permissioned public ledger, meaning that while anyone can use the network, the nodes that govern it are controlled by the non-profit Sovrin Foundation.

The Sovrin Identity Network (SIDN) is a global network of distributed nodes that maintain a shared ledger of identity-related data. Each node has a copy of the ledger and is responsible for validating identity transactions to ensure that the ledger remains consistent and accurate. The nodes are hosted and administered by trusted organizations known as "stewards".

Stewards are responsible for maintaining the integrity of the SIDN by validating transactions and ensuring that the network operates according to a set of established rules and protocols. They do this by using a combination of cryptography and an advanced Byzantine fault tolerance algorithm that ensures that even if some nodes on the network fail or are compromised, the overall network remains secure and functional.

The SIDN is designed to give individuals control over their own digital identities. Instead of relying on centralized authorities such as governments or corporations to manage identity, individuals can use Sovrin to create, own, and control their own digital identities. This provides a secure, private, and decentralized platform for identity management that is resistant to fraud and hacking.

One of the key features of Sovrin is its use of "self-sovereign identity", which allows individuals to create and manage their own digital identities without relying on third-party identity providers. This is achieved through the use of public key cryptography and verifiable credentials, which allow individuals to prove their identity without revealing unnecessary personal information.

Sovrin is built on top of the Hyperledger Indy framework, which provides a set of tools, libraries, and reusable components for creating decentralized identity solutions. It is highly modular and customizable, allowing organizations to build their own identity solutions on top of the Sovrin platform.

In summary, Sovrin is an innovative and highly secure platform for digital identity management that uses distributed ledger technology to provide individuals with control over their own identities. It is an open-source project built on top of a permissioned public ledger and is designed to be highly scalable, secure, and customizable (Windley, 2021).

3.3.5 Tools

Hyperledger Indy SDK is a software development kit that provides a foundation for self-sovereign identity based on distributed ledger technology. The SDK is designed to enable developers to build private, secure, and robust identity solutions. The primary artifact of the Indy SDK is a c-callable library, which can be accessed through various programming languages and the Indy CLI tool (Hyperledger Foundation, 2018).

Hyperledger Aries Cloud Agent Python (ACA-Py) is another essential tool for building non-centralized identity applications and services that operate on non-mobile environments. As a foundation for constructing Verifiable Credential (VC) ecosystems, ACA-Py offers a wide range of features, including the ability to issue, verify, and hold verifiable credentials using both the Hyperledger Indy AnonCreds verifiable credential format and the W3C Standard Verifiable Credential format with JSON-LD and LD-Signatures and BBS+ Signatures.

ACA-Py is designed to work in the second and third layers of the Trust Over IP framework, taking advantage of DIDComm messaging and Hyperledger Aries protocols. With this tool, developers can build solutions that offer trusted, secure, and decentralized digital identities, enabling secure transactions across multiple domains.

Built on the Aries Interop Profile (AIP) 1.0 and most of the AIP 2.0 features, ACA-Py is tailored to run on servers, and the "cloud" in the name highlights its cloud-native design. By using ACA-Py, developers can focus on building VC solutions using familiar web development tools, eliminating the need to master low-level cryptography and Trust over IP-type Aries protocols. ACA-Py provides a flexible and scalable way to create decentralized identity applications that can be deployed to a wide range of environments, from cloud-based servers to embedded devices.

Overall, the combination of the Hyperledger Indy SDK and Hyperledger Aries Cloud Agent Python (ACA-Py) provides developers with powerful tools to create decentralized identity solutions. These tools enable developers to create private, secure, and decentralized identity solutions, which can be used across different applications and platforms. By leveraging distributed ledger technology and advanced cryptographic techniques, these tools help establish trust, accountability, and transparency at the core of identity management (Province of British Columbia, 2021).

CHAPTER FOUR
SYSTEM

4.1 Scenario

The system is based on storing different types of identities in the user's wallet. For this approach, first of all, the user will get a type of identity from an institution. Let's say the user will get an exclusive card from the municipal theater center. With this card, users can watch all theaters for free. Here is the end-to-end flow for this sample:

1. The user will go to the municipal theater center to get an exclusive card.
2. Municipal theater center client opens the "Institution Web Application" and it selects the identity type "exclusive card owner" from the predefined identity types and enters the information for the user.
    a. Exclusive card owner credential definition is predefined over the "Back Office Web Application" when the institution is onboarded to the system.
3. "Institution Web Application" generates a QR code that contains the institution wallet information and the "exclusive card owner" credential attributes.
4. User scans the QR code using the "User Mobile Application".
    a. In this step, a connection is created between the institution wallet and the user wallet.
    b. Credential is issued by the institution to the user.
5. User's identity screen is refreshed and the user can see the new identity ("exclusive card owner") in his/her wallet.

Until this step, the user gets the identity to the wallet and now the user can share this information with any business, the business knows the identity is reliable. When the user goes to the theater:

1. The user opens the "User Mobile Application". The identities of the user are listed on the page. The user selects the "exclusive card owner" identity and the application generates a QR code and shows it on the screen.

a. The QR code contains the user's wallet information and credential information.

2. The theater employee opens the "Business Mobile Application", selects from options "is the user exclusive" and scans the QR code.
    a. Verification request options are created over the "Back Office Web Application" when the business is onboarded to the system.
    b. When the business application scans the QR code, first a connection between the user and the business is created.
    c. Business requests a proof and the user generates a proof according to the credential information (is the user exclusive).
3. Because the user is exclusive, on the "Business Mobile Application", "user is exclusive" information is displayed.

As defined end to end, a user is issued a verifiable credential by an institution, and a business can verify the user has this credential. The high-level system flow is defined in Figure 4.1.

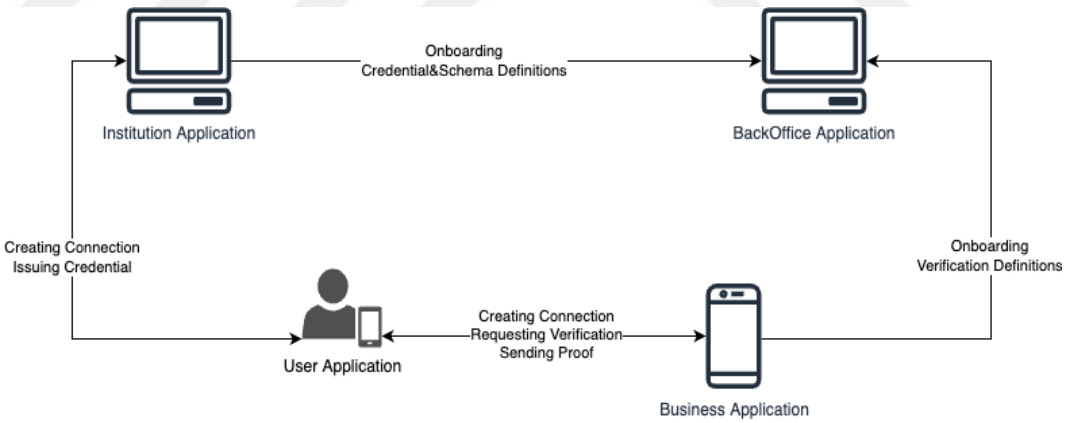


Figure 4.1 System high-level flow

4.2 System Architecture

In this study, an SSI (Self-sovereign identity) solution is produced. The high-level system architecture is shown in Figure 4.2.

SSI, self-managing identity is the person's possession and control of his identity without administrative authorities. SSI allows people to interact with the same confidence and freedom capacity in the digital world as in the offline world.

Hyperledger-Indy was used to provide the SSI solution. Hyperledger is an open-source blockchain project that provides bridge services for developing open industrial or corporate blockchain projects. Hyperledger-Indy is a distributed ledger for identity management.

In order to perform SSI operations on Hyperledger-Indy, we need a network that verifies our identity provider and software solutions with which we can perform identification. Sovrin was used as the network solution. Sovrin is a non-profit closed Hyperledger-Indy network to provide a controlled consensus where identity checks can be carried out. Sovrin network access is open to all, and the network nodes that provide the consensus of the transactions on the ledger are managed by the non-profit Sovrin Foundation in this case. Each identity information written on the network is realized as a result of the approval and control of the Sovrin Foundation. For this reason, it can be verified which authority each digital identity information on the network belongs to. Aries Agent and Indy-SDK were used as software solutions during the interaction with the network and the creation user applications. Aries Agent is a library that provides RFC (Request for Comment) related to Indy. RFCs identify key topics that help standardize the Aries ecosystem. Indy SDK is an open-source library provided by Indy and provides us with infrastructure on different platforms for us to perform identity transactions.

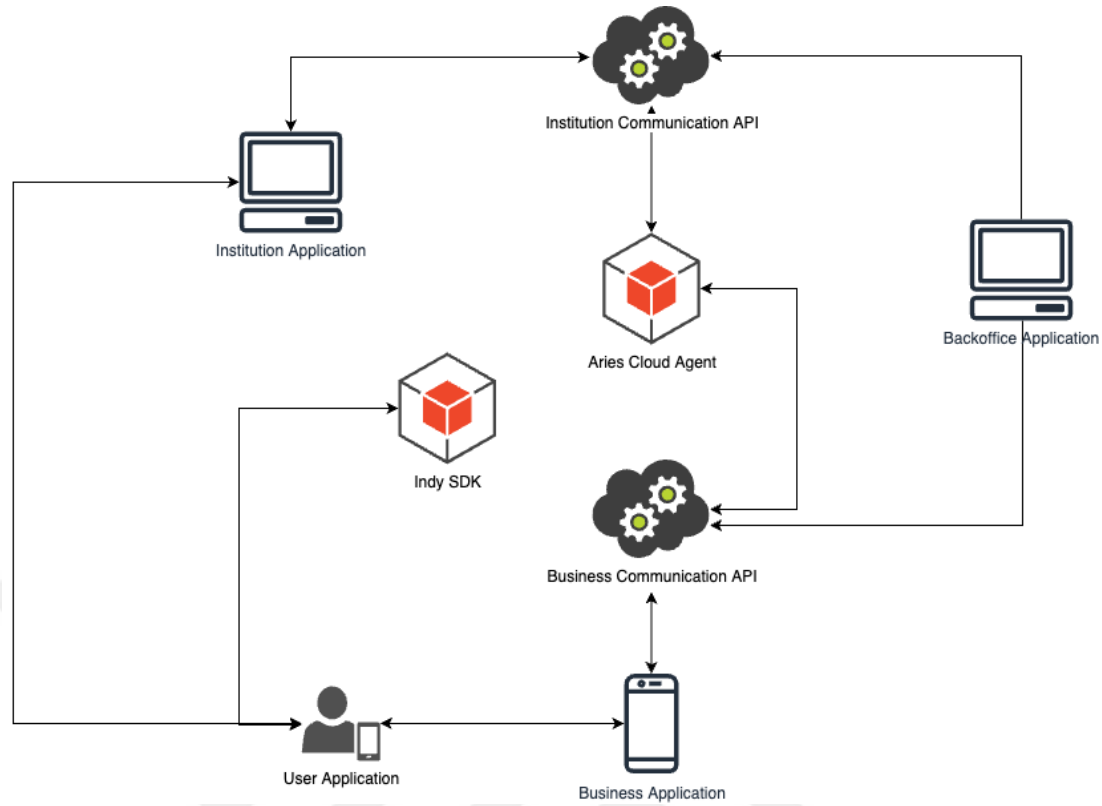


Figure 4.2 System high-level architecture

Within the scope of this study, an architecture (Figure 4.2) was created using these technologies. Within this architecture;

- A mobile application has been developed for users to keep their own identities.
- A service infrastructure has been developed for users who will use the mobile application to provide their identity information according to these principles. This infrastructure is built on the Aries Cloud Agent library.
- A business layer has been created that will create the identities that will be given to individuals according to certain rules and provide the processes to communicate this according to the Indy infrastructure.
- Streams are created by creating an intermediate layer that will provide a communication layer between the SSI infrastructure and the communication of other units and that can translate different message structures.
- A mobile application has been developed for businesses to verify users' identities.

- A web application has been developed for institutions to issue users' identities.

4.2.1 Sovrin Components

Holder: Holder is the role that owns one or more verifiable credentials and generates verifiable presentations from them. In this example, the holder is an individual, but it might also be an organization/company. The entity that the information attests to is the holder.

In this study, users issue their credentials to the related institution.

Issuer: Issuer is the role that asserts claims about one or more subjects, generates a verifiable credential based on these claims, and sends the verifiable credential to a holder. Corporations, non-profit organizations, trade associations, governments, and people are examples of issuers.

By utilizing the Sovrin Network, businesses can issue credentials that can be stored in the digital wallets preferred by identity holders. These credentials contain attributes, such as a home address or social security number, which can be used to generate Zero-Knowledge Proofs (ZKPs). With the Sovrin Network, individuals can generate ZKPs to prove their identities without having to disclose sensitive personal information, thus providing a higher level of privacy and security.

In this study, institutions are issuers. User credentials are issued by the institution to which the identity is related.

Verifier: A role that processes one or more verified credentials, optionally contained within a verifiable presentation. The verifier is the people, organization, company, or government to whom the holder must demonstrate the authenticity and reliability of the information.

A public register for Credential Definitions, Schema definitions, and Issuer public key is provided by the Sovrin network. Issuer public keys are utilized in the process of verifying proofs derived from credentials that have been issued.

In this study, businesses are verifiers. Businesses verify proofs that have been offered by users.

We can summarize the functions of the components in the system, which are shown in Figure 4.3, as follows:

- An issuer's public identification is recorded in a Verifiable Data Registry, where it can be accessed by anybody.
- The Credential, which is signed by the Holder's public identity, is issued to the Holder by the Issuer.
- A digital wallet is something that the Holder has possession of and may use to maintain their credentials.
- The Credential Holder will present their document to the Verifier.
- By accessing and resolving the public identifier on the Verifiable Data Registry, the Verifier is able to establish that this data is reliable.

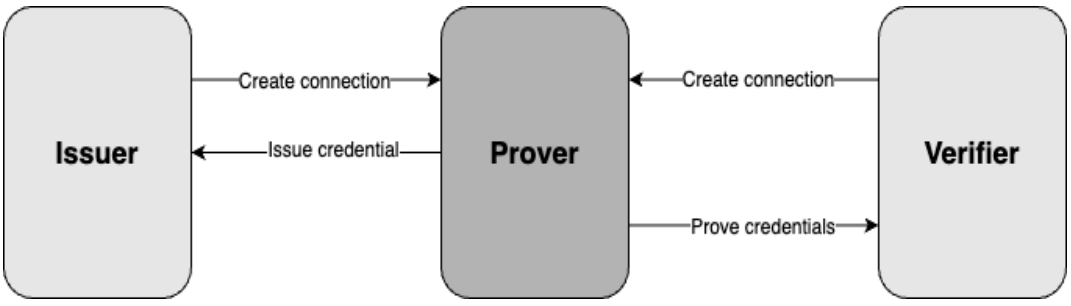


Figure 4.3 Sovrin components

4.2.2 Workflow Description Based On Indy

4.2.2.1 Infrastructure

The ledger is specifically designed to store Identity Records that describe Ledger Entities, including Service Endpoints, Public Keys, Credential Definitions, and Credential Schemas. Each Identity Record is associated with a single, globally unique Decentralized Identifier (DID) that can be resolved through the distributed ledger without relying on a centralized resolution authority. This approach enhances the security and privacy of identity management, as it eliminates the need for a single point of failure and ensures that identity information is under the control of the individual. Multiple DIDs can be owned by each Identity Owner in order to safeguard their privacy.

In this work, we shall make use of two distinct kinds of DIDs. The first one is Verinym. The Legal Identity of the Person Using the Verinym is Linked to the Verinym's Ownership. All parties should be able to verify, for instance, that a particular DID is utilized by a government for the publication of schemas for a particular document type. The second one is Pseudonym. In the context of ongoing digital interaction, a Pseudonym is a Blinded Identifier used to protect privacy. If a Pseudonym is utilized to maintain a single digital relationship, it is referred to as a Pairwise-Unique Identifier. Pairwise-Unique Identifiers will be used to maintain secure connections between participants in this work.

A new Identity Record, known as an NYM transaction, is generated when a DID is created and registered on the Ledger. The NYM transaction serves several purposes, including the creation of new DIDs that are registered on the Ledger, the establishment and rotation of verification keys, and the setting and modification of roles associated with the DID. Essentially, the NYM transaction is a critical element of the DID registration process, enabling identity owners to securely manage their digital identities on the Ledger.

Publishing with a DID verification key verifies that someone owns this DID because only that person knows the signature key and any DID-related procedures that require it.

As the ledger is publicly accessible, individuals or entities who wish to release DIDs need to obtain the role of Trust Anchor on the ledger. A Trust Anchor is a recognized organization or person that is already a part of the ledger and has the capability to assist others in joining the system.

Obtaining the Trust Anchor role is a crucial first step toward recording transactions on the ledger. Institutions will need to acquire the Trust Anchor role on the distributed ledger to create Verinyms and Pairwise-Unique Identifiers for users.

To become a Trust Anchor, it is necessary to contact an organization or person that already holds the Trust Anchor role on the blockchain. In the test ledger, only NYMs with the Steward role will be employed, and all Stewards are automatically granted the Trust Anchor role.

A connection to the Indy nodes pool is required to view and write the ledger's transactions after getting the required permissions.

The ledger stores the list of nodes in the pool as NODE transactions. Libindy can restore NODE transactions from Genesis transactions. Each Pool Configuration is specified by its respective pool configuration name and JSON representation. The path to the file containing the list of Genesis transactions is the most significant field in pool configuration json.

4.2.2.2 Initialize Back Office Wallet

Initialize Back office flow is shown in Figure 4.4.

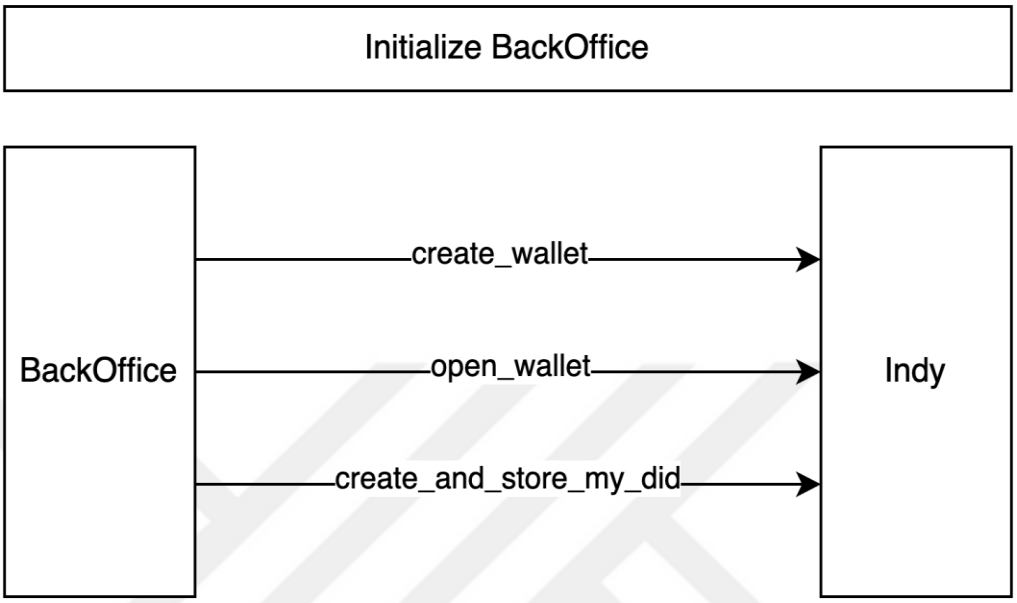


Figure 4.4 Initialize back office flow

The Back office Agent should acquire ownership of the DIDs that correspond to NYM transactions on the ledger with the Steward role.

The wallet is a concept possessed by Libindy. The wallet is a secure repository for cryptographic information such as keys, DIDs, etc. The agent must first construct a named wallet by executing "wallet.create wallet" in order to safely store and handle the Steward's DID and signing key. Once the wallet is created, it can be opened using "wallet.open_wallet", which will return a handle that can be used in subsequent calls to interact with the wallet. With the wallet handle, The agent is capable of a wide range of tasks, including adding and removing data from the wallet, as well as signing and verifying transactions.

After opening the wallet, a new DID record can be added to it by calling the "did.create_and_store_my_did" function. This function generates a new DID and its corresponding verification key, which are both stored in the wallet. Although the signing key for this DID is also stored in the wallet, it cannot be directly accessed.

4.2.2.3 Onboarding of Institutions and Businesses

Each connection consists of two Pairwise-Unique Identifiers (DIDs). The first DID is owned by one party while the second is owned by another. Both parties are aware of both DIDs and comprehend the connection between them. The relationship that exists between them cannot be shared with other people; it is unique to those two parties due to the fact that different DIDs are used for every paired interaction.

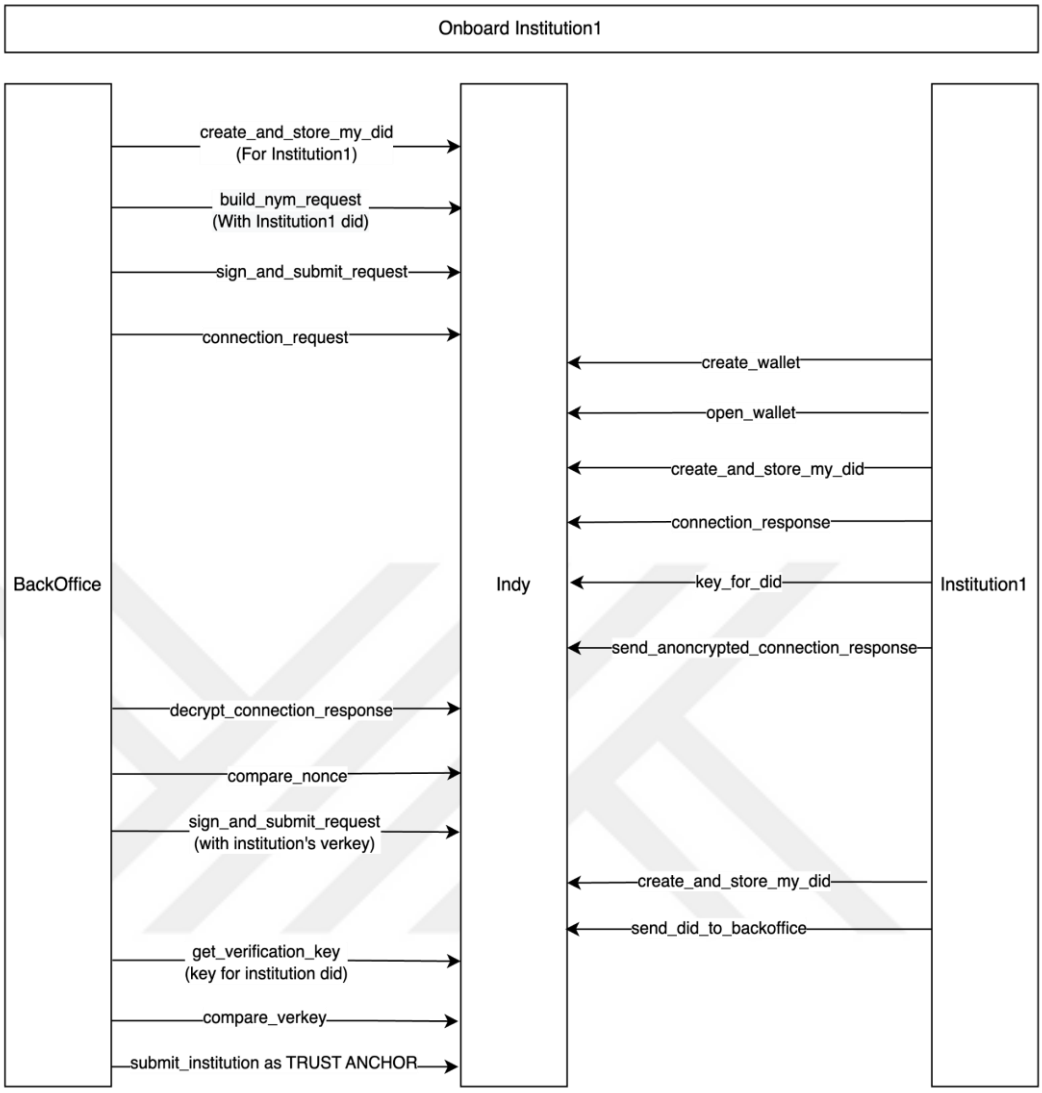


Figure 4.5 Institution onboarding flow

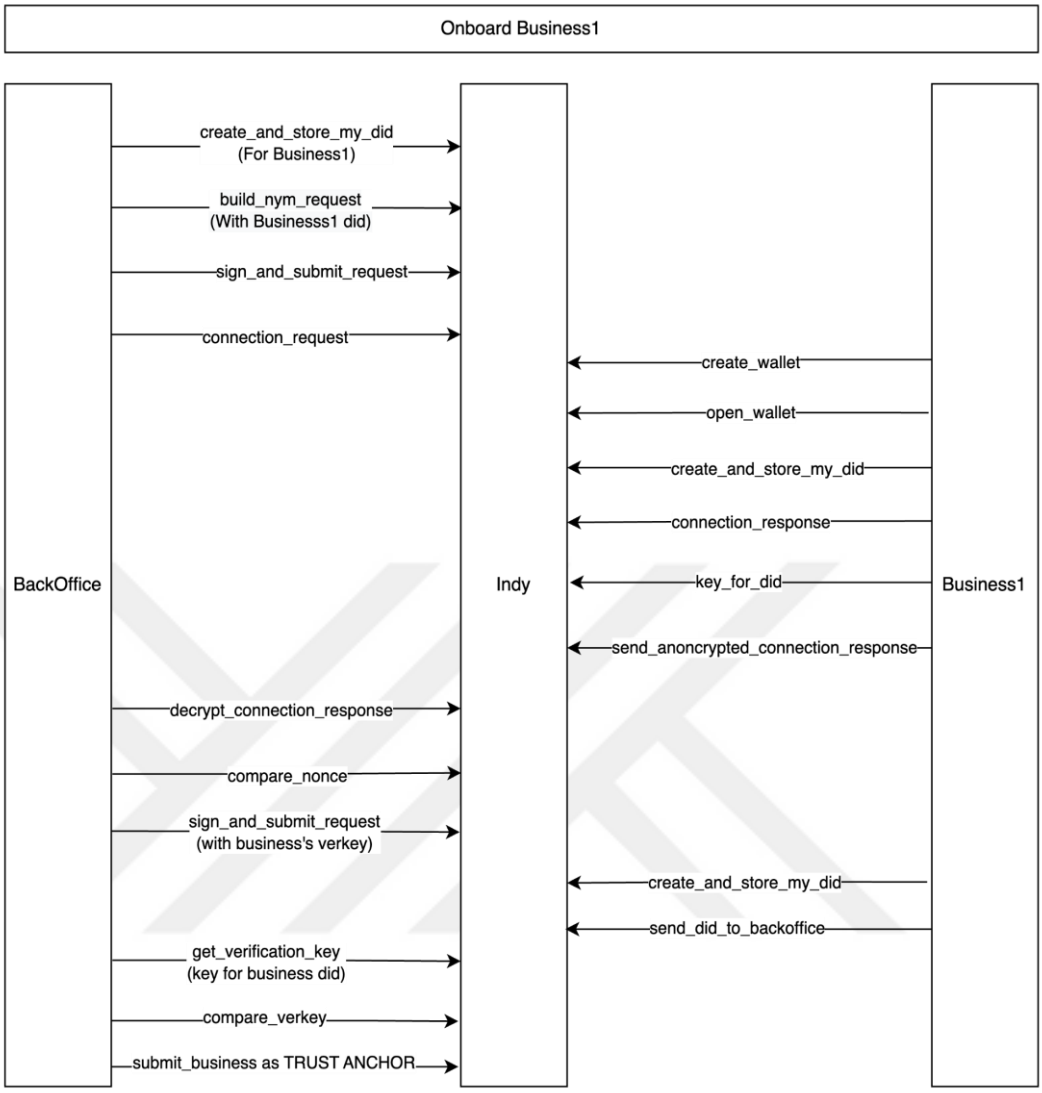


Figure 4.6 Business onboarding flow

Onboarding processes for institutions and businesses are the same. Here is the flow for onboarding that is shown in Figure 4.5 and Figure 4.6; same for the both institutions and businesses:

- Backoffice creates a new DID record in the wallet by invoking "did.create_and_store_my_did", which he will use exclusively for secure interactions with the Institution.
- Backoffice transmits the matching NYM transaction to the Ledger in order to construct the NYM request, then invokes "ledger.sign_and_submit_request" to send the request.

- Back office generates the connection request with the DID and Nonce.
- Back office sends the Institution a connection request.
- Institution accepts Back office's connection request.
- The institution creates a wallet.
- Institution1 populates its wallet with a new DID record. Only secure communications with the back office will be supported.
- Institution1 generates a connection response containing the DID, Verkey, and Nonce produced in response to a connection request.
- Institution1 requests the Verification key of the Back office's DID from the ledger.
- Institution1 encrypts the connection response anonymously using the Back office verkey. Using its private key, only the recipient is capable of decrypting these messages. The Recipient can check that the message is correct, but it can't check the identity of the Sender.
- Institution1 sends the Back office response to the anonymously encrypted connection.
- Back office decrypts the connection response anonymously.
- Back office verifies the identity of Institution1 by comparing Nonce.
- Ledger receives the NYM transaction for Institution1's DID from the Back office.
- After establishing the connection, Institution1 must produce a new DID record for use as the Verinym in the Ledger. Institution1 generates a new DID and stores it in its wallet.
- Institution1 composes a message that includes the newly generated DID and verkey.
- Institution1 verifies the message's authenticity and encrypts it.
- Back office receives an encrypted message from Institution1.
- Back office decrypts the communication that was received.
- Back office requests the Verification key of Institution1's DID from the ledger by invoking "did.key_for_did".
- Back office verifies the authenticity of Institution1 by comparing the Message Sender Verkey and the Institution1 Verkey received from the Ledger.

- Back office transmits the corresponding NYM transaction to the TRUST ANCHOR ledger.
- Institution1 now has a DID associated with his identification in the Ledger.

Back office is the sender of this transaction, but Institution1 will be the owner of DID because it uses Institution1's Verkey.

4.2.2.4 Creating Schema and Credential Definitions

A Credential Schema serves as the fundamental semantic framework that defines the possible attributes of a Credential. Modifying an existing Schema is not possible. In case of any updates required, a new Schema with a different version or name must be generated. Any Trust Anchor can create and store a Credential Schema in the Ledger. For instance, the Back office generates and releases the Theater Card Credential Schema to the Ledger.

Credential Definition is similar to Credential Schema in that the keys used by the Issuer to sign Credentials also conform to a certain Credential Schema. Again, updating data in an existing Credential Definition is not possible. Consequently, if a Credential Definition must be modified, a new Credential Definition must be generated by a new Issuer DID. Any Trust Anchor is able to create and save Credential Definitions in the Ledger. Institution1, for example, generates and publishes to the Ledger a Credential Definition for the known Theater Card Credential Schema.

The Trust Anchor retrieves the particular Credential Schema from the Ledger and generates the Credential Definition that corresponds to the received Credential Schema. The secret Credential Definition section of this Credential Schema is similarly stored in the wallet, however, it cannot be accessed directly.

The Trust Anchor transmits to the Ledger the relevant Credential Definition transaction. This is how the Credential Definition for the Theater Card Credential Schema is developed.

Creating a schema and creating a credential definition using the defined schema flow are shown in Figure 4.7.

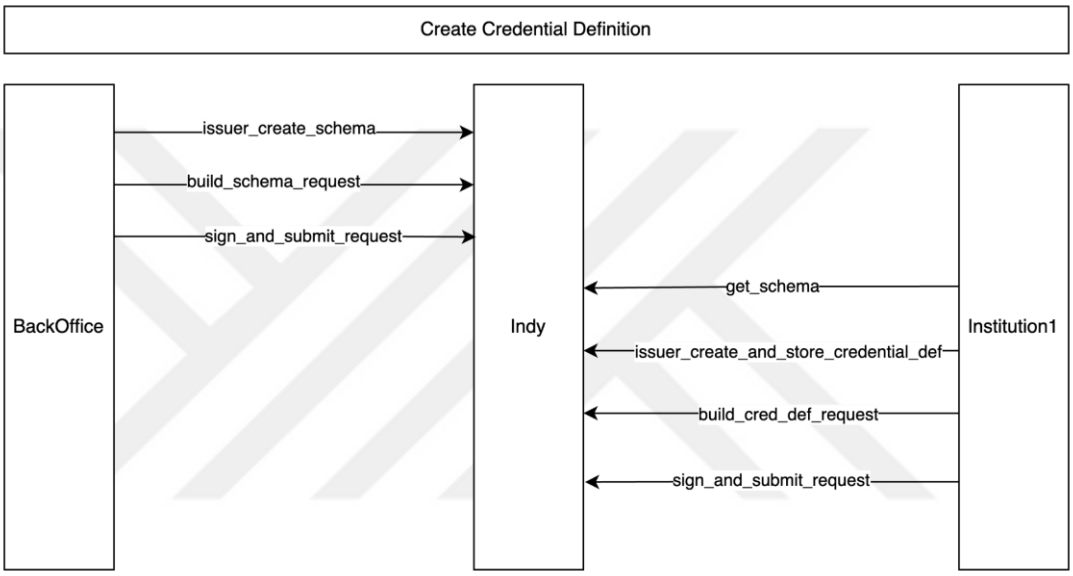


Figure 4.7 Create credential definition flow

4.2.2.5 Issuing Credential

A credential is an identity-related piece of information. This is assertedly accurate information. A credential is issued by a credential issuer. Any entity that has been identified by the ledger as an identity owner, also known as a DID, has the ability to act as an issuer of credentials. This means that they are able to issue verifiable credentials for any other DID that they can authenticate on the ledger.

The utility and dependability of a credential are dependent on the issuer's standing with regard to that credential.

First, all communications between actors are encrypted. The credential's value is issued by Institution1. A user desires to view the attributes included within the "Theater Card" Credential. These attributes are known since a Theater Card Credential Schema has been recorded in the Ledger. Nonetheless, the Theater Card Credential has not yet been sent to the user in a usable format. The User wishes to utilize this Credential. The user must first submit a request for it, but first, a Master Secret must be generated. A Master Secret is a piece of secret information leveraged by a Prover to ensure that a credential applies exclusively to them. The Master Secret integrates information from numerous Credentials to demonstrate that they all pertain to the same subject (the Prover). Only the Prover should know the Master Secret.

The user generates a Master Secret within their own wallet. Additionally, the user must obtain the Credential Definition that corresponds to the credential definition id in the Theater Card Credential Offer. At this point, the user possesses everything necessary to submit a Credential Request for the Institution1 Theater Card Credential.

Institution1 generates both unencoded and encoded values for every attribute in the Theater Card Credential Schema. Institution1 generates the user's Theater Card Credential. The Theater Card Credential has been issued at this time. It is stored in the user's wallet.

The user is in possession of it, much in the same way that he or she would be holding a physical theater card that had been provided to them.

Issuing a credential to a user by an institution flow is shown in Figure 4.8.

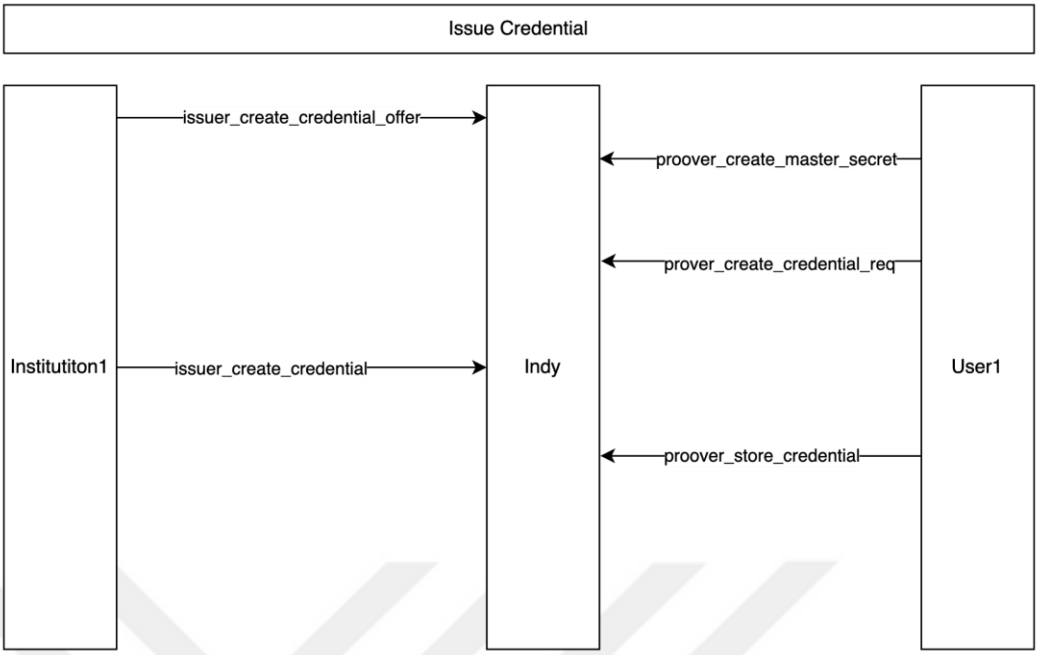


Figure 4.8 Issue credential flow

4.2.2.6 Proof

When a business wants to make a proof, it should send a connection invitation to the user. The process is different from back office-business/institution connections because the user wallet is using Indy-SDK, but the flow is the same.

After the user accepts the connection invitation, the business sends a proof request that contains the schemas' and attributes' predicates to be verified.

According to the proof request, the required credential attributes should be satisfied by the user. To accomplish this using Indy-SDK, the user must obtain the Credential Definition and Credential Schema for each credential used in the same manner as when creating a Credential Request.

At that point, the user creates the Proof for the business and when the business examines the received Proof, they will observe the following format:

- The business obtained all requested attributes.
- The business desires to examine the Validity Proof.
- To accomplish this, the business must obtain each Credential Definition and Credential Schema for each identifier given in the Proof in the same manner as the user.
- Now the business has everything necessary to verify the user's Proof.

The proof flow is shown in Figure 4.9.

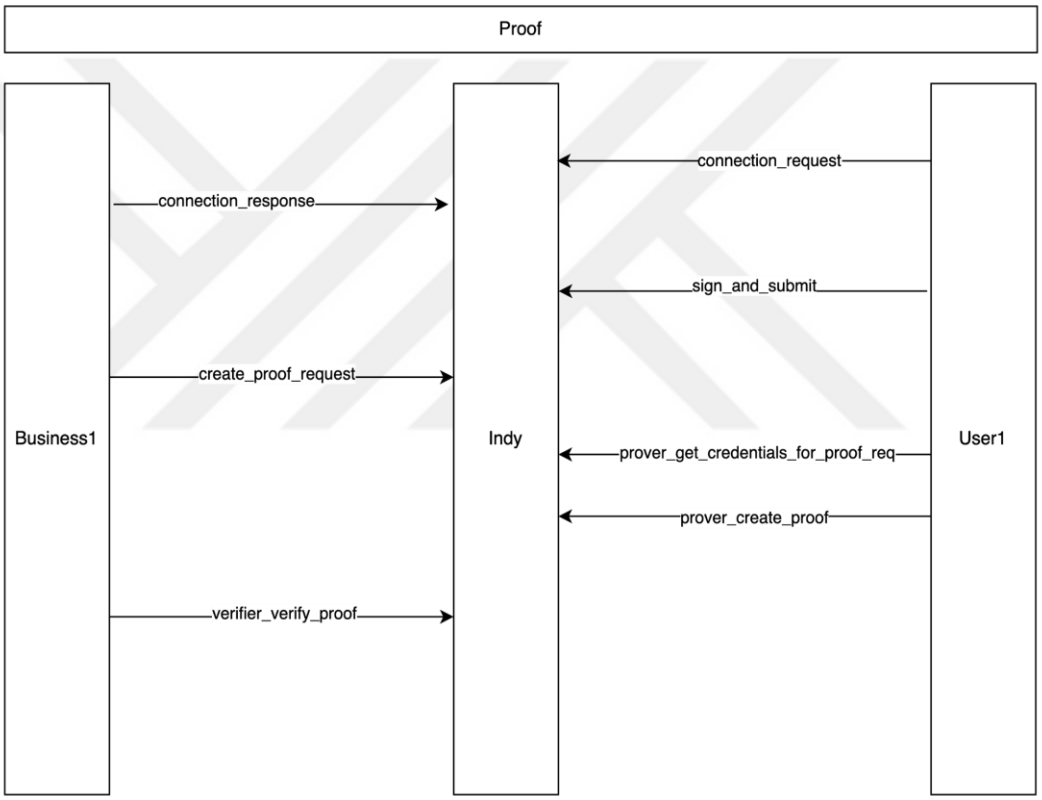


Figure 4.9 Proof flow

4.2.3 Application of ACA-Py

A single instance of an Aries agent consists of two components: the agent itself and a controller (Figure 4.10).

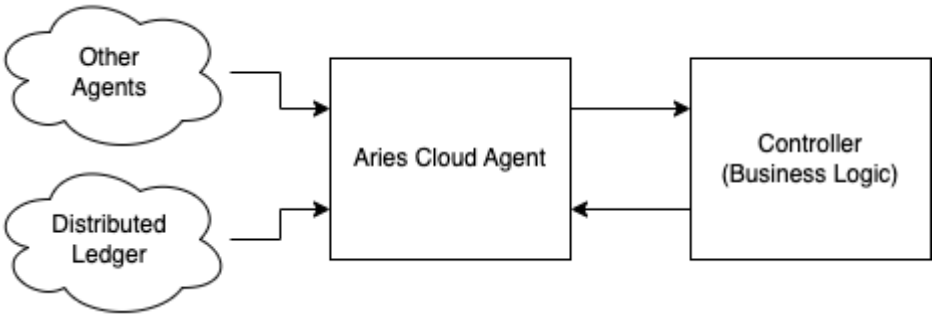


Figure 4.10 Aries agent structure

Aries Cloud Agent is responsible for all of the core Aries functionality, including working with other agents, handling secure storage, sending event notifications to the controller, and getting instructions from the controller.

The controller defines the behavior of a particular agent instance, including how to respond to agent events and when to trigger the agent to initiate events.

The agent communicates events to the controller, and the controller responds with instructions for the agent's administrator. The agent provides a REST API to the controller for all of the administrative messages it is set to handle, and the controller registers a webhook with the agent in order to receive HTTP callbacks for event notifications.

Aries Agents interact with one another through a messaging system known as DIDComm (DID Communication). DIDComm allows secure, asynchronous, end-to-end encrypted messaging between agents, with messages routed through an intermediary agent configuration.

ACA-Py utilizes the Ursa library and the indy sdk for cryptographic requirements and interaction with the indy blockchain, respectively.

4.2.3.1 Infrastructure

Initially, a Postgresql database is started up as the wallet storage. Then an Indy network was established. A Von network, which is a development-level, portable Indy Node network with a Ledger Browser, is utilized to establish an Indy network. The Ledger Browser enables a user to view the status of a network's nodes and to filter Ledger Transactions. Also, public DIDs are published via the Von network.

After the Von network has been activated, a DID is created as the TRUST_ANCHOR role with a seed and made public. This seed is stored in the configuration file and utilized by the Back office instance.

4.2.3.2 Initialize Back Office Wallet

To be able to create sub-wallets for institutions and businesses, multi-tenancy is enabled in ACA-Py. In multi-tenancy, a single agent is run and resources are shared between the tenants of the agent. Institutions and businesses as different tenants have their own wallets and DIDs.

In ACA-Py, multi-tenancy distinguishes between a base wallet and sub-wallets that are different tenants. When multi-tenancy is turned off, a sub-wallet is able to perform all operations with a single-tenant ACA-Py instance. But the base wallet has a different role and functionalities. Base wallet manages the sub-wallets and uses different endpoints on multi-tenant Admin API. In that way, the base wallet stores all configurations and information of sub-wallets and routes the messages between sub-wallets. The base wallet can not do the actions like issuing credentials, presenting proof, etc.

In the design of this work, the back office is the base wallet and manages the institutions and businesses as sub-wallets.

The base wallet creates the sub-wallets in managed mode. This makes the ACA-Py fully control the wallet. Thus, the messages that come from other agents can be unlocked and processed. To determine which tenant to route the message to, the mediator routing type is used.

The default mediator set is stored in the base wallet, and this set is used by sub-wallets. When a connection or key is generated by a sub-wallet, it will be registered at the mediator through the base wallet relationship. Even now, the base wallet continues to perform the function of a relay by directing messages to the appropriate sub-wallets.

4.2.3.3 Onboarding of Institutions and Businesses

For both onboarding processes, a tenant registration operation is started to create wallets. Using the [post] /multitenancy/wallet endpoint, the back office wallet sends the tenant information which includes wallet name, wallet key, webhook urls for the wallet, and other configuration parameters.

Webhook URLs are for calling back to the controller. ACA-Py uses webhook events to establish the connection. It is possible to designate multiple webhook targets but unique webhook targets for each wallet are created in this work. Wallet id is included in the events in the “x-wallet-id” header to recognize the separation of the wallet.

When a wallet is created for an institution or a business, a wallet id is created and this wallet id is stored as a record related to the institution or the business.

When an operation will be called on behalf of the business or the institution, a token is created with [post] /multitenancy/wallet/{wallet_id}/token endpoint. To create a token, the admin API key, wallet key, and wallet id of the tenant are needed. Additionally, once a token is created, the previous one becomes invalid.

JSON Web Tokens (JWTs) are being utilized as an additional authentication technique for sub-wallets. Every single admin API request for the wallet that uses the Bearer authorization scheme requires the provision of a JWT token as a parameter.

4.2.3.4 Creating Schema and Credential Definitions

As defined before, institutions are issuers of the system. Through the back office, on behalf of institutions, schema definitions are created according to the identity type of institutions. Schema definitions cover the fields of the data. Then credential definitions, that include these schema definitions, are created. Using credential definitions, institutions issue credentials to the users. These credentials are the identities of users and will be proofed by businesses (Khalifa Toumi, 2020).

Schema Definition:

Endpoint: [post] /schemas

Request: Schema name, version, and attributes of the schema.

Credential Definition:

Endpoint: [post] /credential-definitions

Request: Credential tag and created schema definition id.

4.2.3.5 Issuing Credential

In issuing a credential process, there are two stakeholders: institutions as issuers that send offers to issue a credential, and users as holders that send credential requests.

The first step is creating a connection between the institution and the user. If the connection exists, the existing one is used. Using the connection and the credential definition id, the institution sends an offer that includes the identity attributes to the user with [post] /issue-credential/send-offer. The user gets that offer and accepts with [post] /issue-credential/records/{cred_ex_id}/send-request. A credential exchange id is generated when the institution sends the offer.

4.2.3.6 Proof

In the proof process, there are two stakeholders: businesses as verifiers that request proofs, and users as holders that present proofs.

As with the beginning of all transactions, a connection is needed between the verifier and the holder. If the connection exists, the existing connection is used; if not, a new connection is created.

As defined in the application part, the content of the proof requests is created as a record for businesses through the back office application. When a business requests proof, the requested attributes, and restrictions come from this record. When the business scans a QR code, the connection is found or created between this business and the user, and this connection id is dynamically changed for each proof request. The business sends this combination of the request with [post] /present-proof/send-request to the user.

When the request comes to the user, the user creates a proof according to the restrictions of the proof request with owned credentials. When this proof is presented with [post] / present-proof /records /{pres_ex_id} / send-presentation from the user to the business, the business verifies the identity of the user and uses that identity according to the need.

4.2.4 Technologies

Python is used for ACA-Py implementations. Python is an object-oriented high-level programming language that can be interpreted. Python also has dynamic semantics. Because of its high-level in-built dynamic typing, data structures, and dynamic binding, the language is well-suited for Rapid Application Development and may also be used as a scripting or glue language to connect pre-existing components. The Python programming language and its comprehensive standard library are open-source and freely distributable in source or binary form for all major platforms.

React.js is used for Frontend implementations. For the purpose of developing user interfaces with JavaScript, the React library is a declarative, powerful, and adaptable tool. It is possible to assemble complicated user interfaces out of small, independent pieces of code known as components.

ASP.NET is used for Backend implementations. The ASP.NET Web API is a framework for developing HTTP services that can be accessed by any client, including mobile devices and browsers. It is an ideal platform for RESTful application development.

Flutter is used for the implementation of mobile applications. Google's Flutter is an open-source framework for developing natively built, cross-platform applications with a single codebase.

The development of our application was made using PostgreSQL as the database management system. PostgreSQL, which has become widespread as an open-source database, is also used by many businesses. PostgreSQL structurally supports JSON and SQL queries. This allows it to support relational and un-relational structures. Postgre SQL, which is used by many businesses, has become widespread and has gained a consistent structure for a very long time.

4.3 Application

4.3.1 BackOffice Application

Institution onboarding: For each onboarding operation, a new wallet is created for institutions (Figure 4.11). A connection is created between the back office wallet and the institution wallet.

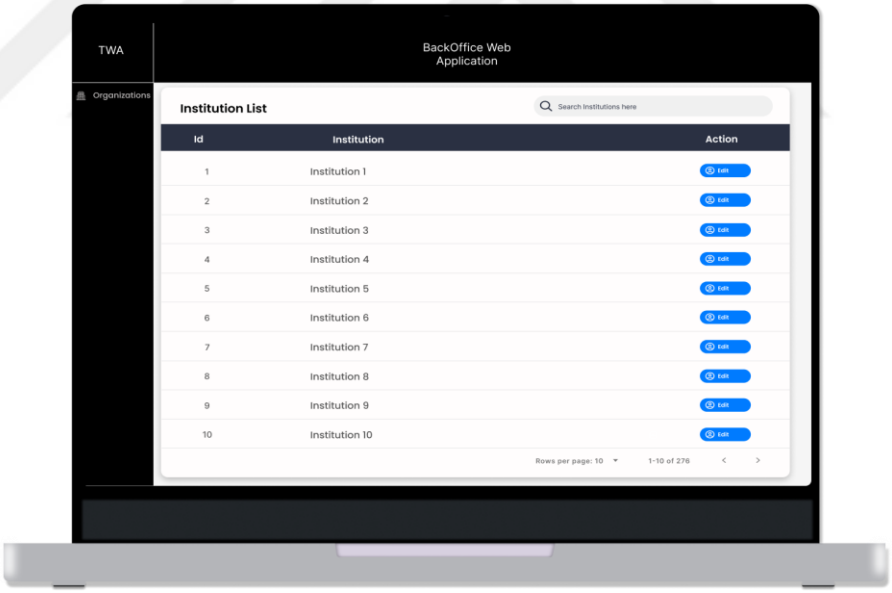
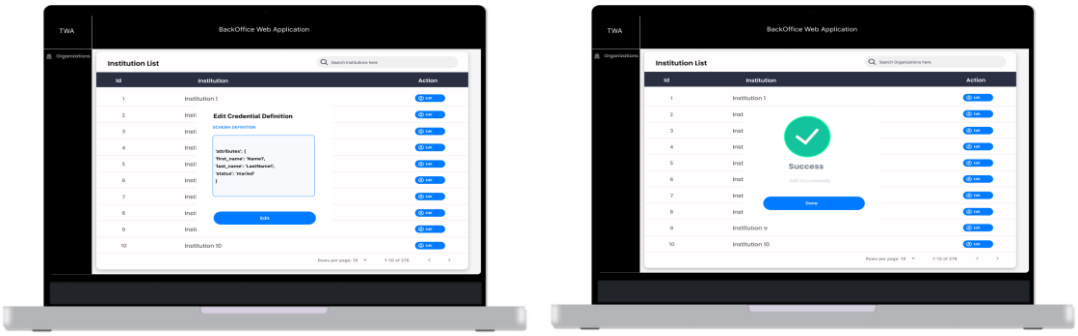


Figure 4.11 Institution list

Create credential definitions for institutions: A schema and credential definition is created for each institution (Figure 4.12). Definitions are submitted by the back office wallet, the created definition IDs are mapped by the institution on the database.



(a) (b)

Figure 4.12 Creating credential definition; Creation (a) and Success (b)

Business onboarding: For each onboarding operation, a new wallet is created for businesses (Figure 4.13). A connection is created between the back office wallet and the business wallet.

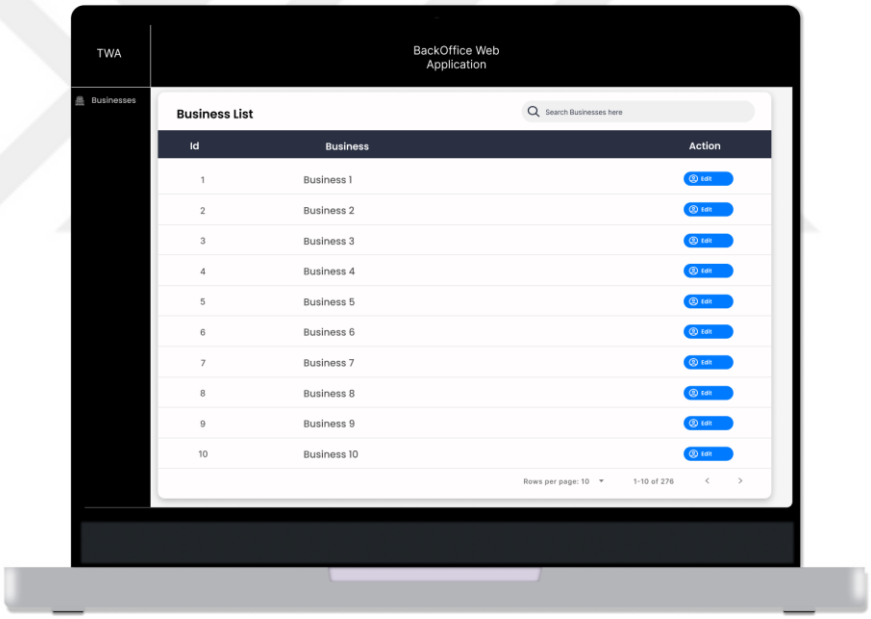
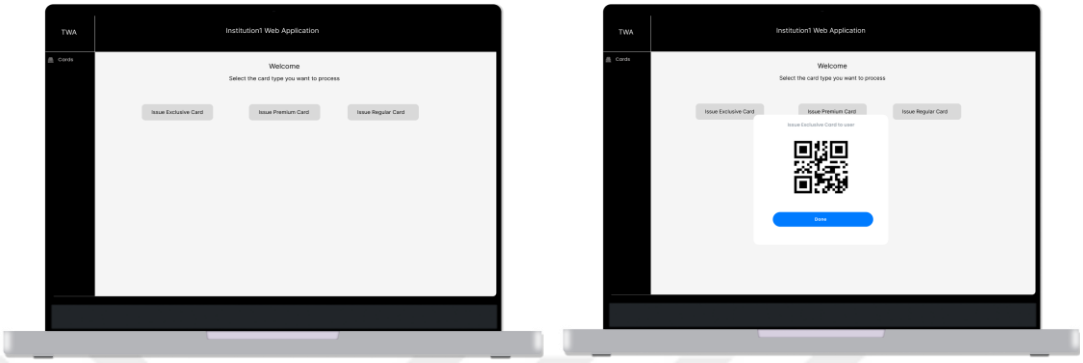


Figure 4.13 Business list

4.3.2 Institution Application

Predefined credential attributes are known by the institution (Figure 4.14 (a)). Every institution creates a credential definition using its mapped credential and schema definition attributes (mapped in the database at the onboarding operation). A QR code that contains the institution and credential information is generated by the institution

application (Figure 4.14 (b)). This QR code will be scanned by the user application and the credential will be issued to the user by the institution.



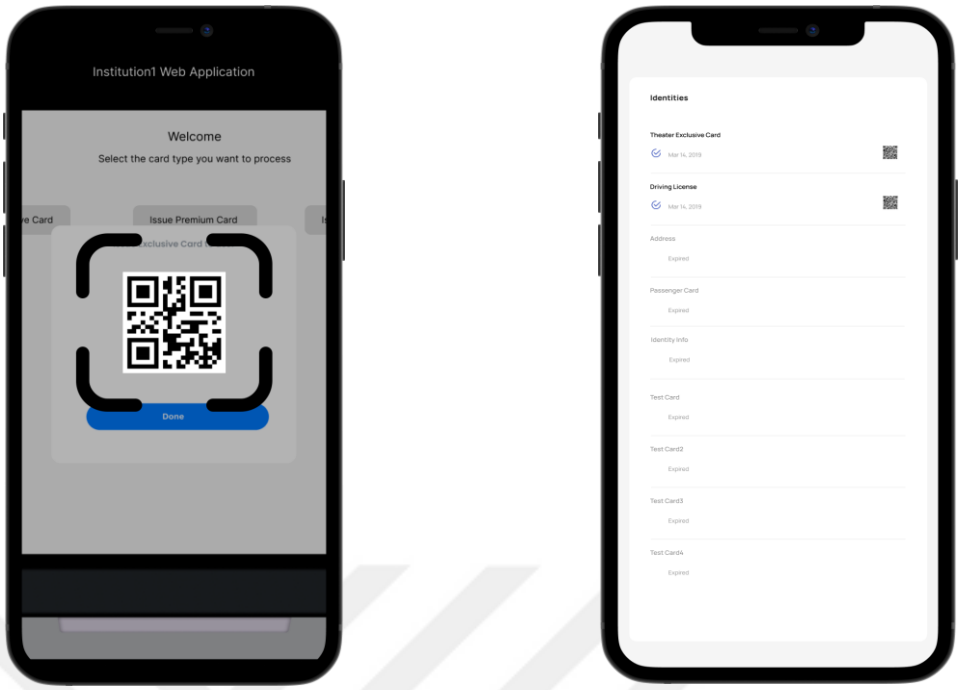
(a) (b)

Figure 4.14 Credential issuing; Credential selection (a) and Issue (b)

4.3.3 User Application

Users collect credentials (identities) by scanning QR codes (Figure 4.15 (a)) that have been generated by institutions. Many credentials can be issued for the user from different institutions. When the QR code is scanned, a connection between the institution and the user is created and a credential is issued to the user by the institution.

Users can list all the credentials they have (Figure 4.15 (b)). Credentials are in the users' wallets.



(a) (b)

Figure 4.15 Credential list; Scanning to issue (a) and Listing (b)

Users can generate QR codes that contain the user's wallet information and credential information (Figure 4.16). When a business, that has been onboarded to the system before, scans this QR code, the credential information of the user can be verified by the business.

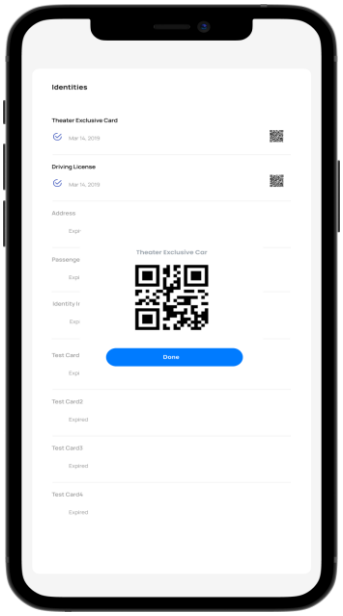
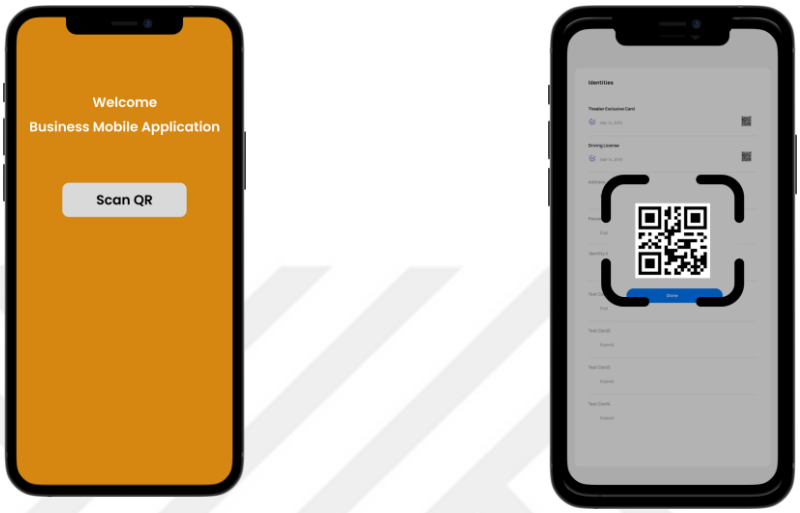


Figure 4.16 Generate credential QR code

4.3.4 Business Application

When a business scans (Figure 4.17 (a)) the QR code that has been generated by a user (Figure 4.17 (b)), a connection is created between the user and the business. Then the user can prove his/her credential to the business.



(a) (b)

Figure 4.17 Business application; Home screen (a) and Scaning a credential (b)

Businesses can see the details of verified credential data on the business mobile application (Figure 4.18). The credential's owner, the user, shared this information with a business. Businesses can also see the authority that has issued this credential.

Figure 4.18 Show the proof result

CHAPTER FIVE
RESULTS AND DISCUSSION

As a result of this study, the real-life use of Blockchain-based SSI technology has been implemented.

Although there are many options to develop this structure, the advantages and disadvantages of each option were examined and the most suitable one for the study was decided.

Considering the concept of a smart city, since one of the stakeholders of the developed system is directly human, it is considered that the system should be safe and consistent. In this direction, it was decided that it would be appropriate for the infrastructure to be blockchain-based. It is the most important rationale that a blockchain-based system is secure, consistent, and immutable.

During the process of the study, the changing and developing technology was followed and put into practice. The SSI technology, which has existed for a long time, but whose applications on the blockchain are increasing, forms the infrastructure of the study for the same reason. Since SSI is a structure that provides identity management under the dominance of the person as a concept, the SSI structure overlapped and was used in this study by implementing it without making any adverse changes to the structure. At this stage, it was decided that building a system on Hyperledger Indy would give the most appropriate result.

It was decided to use the ACA-Py framework, which uses the Hyperledger Indy infrastructure and provides a more advanced infrastructure developed on Indy protocols. The reason for choosing this framework is detailed in the following sections.

5.1 ACA-Py Properties

Table 5.1 defines the following ACA-Py properties:

- Supported types and signatures for properties of verifiable credentials.
- Supported protocols for the usage of decentralized identifiers (DIDs), communication between DIDs, exchanging verifiable credentials, authorizing transactions, and data communication.
- Multi-tenancy support that allows the creation of multiple agent instances.
- Mediator support that allows secure messaging between agent and client over DIDComm connection.
- Multi Indy ledger support that allows the ability to use multiple Indy ledgers for resolving a DID by the ACA-Py agent.
- Scaled Deployment Support that allows establishing and change of the running ACA-Py and storage components capacity.
- Transaction Endorser Support allows a non-privileged agent to ask another agent to sign their transactions so that they can be written to the ledger.
- Secure Storage Types describes the support for a range of functionalities, including storage, cryptography, Indy credentials, and Indy ledger operations.
- Number of contributors is the count of people that have made commits to the open-source code.

Table 5.1 ACA-Py properties

Supported Signatures	CL signature, ED signature, BBSBLS signature
Supported Credential Types	Hyperledger Indy AnonCreds, W3C Standard Verifiable Credentials using JSON-LD
Supported Protocols	Aries Interoperability Protocol 1.0 (AIP 1.0), Aries Interoperability Protocol 2.0 (AIP 2.0)
Multi-Tenancy Support	Multi-tenancy in ACA-Py enables numerous tenants to use the same instance of ACA-Py with a distinct context. Each tenant receives a private, encrypted wallet containing only their own info.

Table 5.1 continues

Mediator Support	A mediator is simply a specialized agent. Over a DIDComm connection, this agent transmits information to a client.
Multi Indy ledger support	ACA-Py supports multi Indy ledger with automatic detection.
Scaled Deployment Support	Enables deployment in highly scalable environments, including Kubernetes systems, where ACA-Py and its storage components can be horizontally scaled to handle increased traffic and demand.
Transaction Endorser Support	ACA-Py has a protocol called "Endorser Protocol" that lets a non-privileged agent called an "Author" ask another agent called an "Endorser" to sign their transactions so that they can be written to the ledger. This is needed for Indy ledgers, where new agents usually only get "Author" rights.
Supported Secure Storage Types	Aries Askar, Indy SDK "indy-wallet"
Number of Contributors	100
License	Apache 2.0

5.2 ACA-Py vs Indy

In Table 5.2, comparisons are explored between ACA-Py and Indy.

Both frameworks' Multi-Tenancy support, which permits the creation of numerous instances, is compared. Businesses and institutions should be utilized as multiple agents in this work. Without support for multiple tenants, we would have to build separate environments for each agent.

Authentication support, which is required to guarantee security, is compared to gaining access to the functionalities when the agents are up and running.

The W3C specification specifies a mechanism for expressing these types of credentials on the Web in a manner that is cryptographically secure, respects privacy, and is machine-verifiable. Since generic credentials were created in this study, supporting credentials to be created in W3C standards in the future will ensure that credentials can be used by different stakeholders.

Performance and consistency are the most essential characteristics of a product. The performance status of the two frameworks is included in the comparison.

Table 5.2 ACA-Py and Indy comparison

	ACA-Py	**Indy**
Multi-Tenancy	Supported	Not supported
Authentication	Supported	Not supported, Needs custom implementation
W3C Credentials	Supported	Not supported
Performance	With ACA-Py Askar, the DB connection implementation has been completely changed, so the performance and consistency problems have been resolved.	In the previous versions, sqlite db was used as a wallet and could be run as a single instance. Postgresql support came as a plugin to solve the single instance problem. However, performance and inconsistency problems remain.

5.3 ACA-Py Askar vs Indy Performance

Table 5.3 shows the results of Askar and Indy's performance of interacting agents. It is created from the load test results made and announced by the community (Paul Wenzel, 2022).

Table 5.3 Askar and Indy performance results

	Askar	**Indy**
Number of Iterations	Started: 68251 In Progress: 24 Finished: 68227	Started: 18369 In Progress: 2534 Finished: 15835
Number of Connection Invitations	Total: 68251 Failed: 1	Total: 18361 Failed: 119
Number of Credential Offers	Total: 68250 Failed: 8	Total: 18083 Failed: 1957
Number of Proof Requests	Total: 68242 Failed: 2 Incorrect: 13	Total: 16123 Failed: 116 Incorrect: 0
Number of Revocation Verifications	Total: 136 Failed: 0	Total: 0 Failed: 0

Table 5.4 shows the performance comparison of Askar and Indy wallets. In Indy's previous versions, sqlite db was used as a wallet and could be run as a single instance. Postgresql support came as a plugin to solve the single instance problem. However, performance and inconsistency problems remain in the Indy wallet. With Askar, the DB connection implementation has been completely changed, so the performance and consistency problems have been resolved.

These results are created by up-and-running two agents locally. Locally, a local Postgresql database and local instance of von-network are initiated. A did for the trust anchor is registered on the von-network. The subsequent outcomes are extracted:

- Start-up Duration: Start-up time for an agent.
- Connection Duration: Duration of creating a connection between two agents.
- Credential Definition Publish Duration: Publication of a schema definition and a credential definition utilizing the published schema duration over an agent.
- Average Time Per Credential: Average credential exchange times between two agents when issuing credentials.

Table 5.4 Askar and Indy performance comparison

	Askar	**Indy**
Start-up Duration	34.62s	67.58s
Connection Duration	1.32s	2.03s
Credential Definition Publish Duration	6.19s	9.88s
Average Time Per Credential	2.46s	2.54s

5.4 ACA-Py vs Aries Go

Hyperledger Aries Framework Go enables interoperable communication and data exchange using distributed ledger technologies and peer-to-peer interactions. Although Aries Go has support for many important issues such as W3C credentials, universal wallet, mediator, and BBSBLS Signature, ACA-Py was preferred because Aries Go's development is still ongoing and it is not a product ready.

A comparison of the ACA-Py and Aries Go frameworks is presented in Table 5.5.

Table 5.5 ACA-Py and Aries Go comparison

	ACA-Py	**Aries Go**
Multi-Tenancy	Supported	Not supported, Needs custom implementation
Authentication	Supported	Under development
W3C Credentials	Supported	Supported
Supported Signatures	CL signature, ED signature, BBSBLS signature	ED signature, BBSBLS signature
Universal Wallet	Not supported	Supported
Mediator Service	Supported	Supported
Supported Protocols	Aries Interoperability Protocol 1.0 (AIP 1.0), Aries Interoperability Protocol 2.0 (AIP 2.0)	Aries Interoperability Protocol 2.0 (AIP 2.0)

CHAPTER SIX
CONCLUSION AND FUTURE WORK

6.1 Conclusion

As a result of this study, the real-life use of Blockchain-based SSI technology has been realized. The concept of SSI has started to be applied in many fields today. The studies carried out in this field, combined with the advantages of blockchain technology, have enabled the creation of safe and reliable data. With this technology, it is ensured that personal data is stored in accordance with the laws of protection of personal data all over the world.

Considering the concept of a smart city, it is thought that the system should be safe and consistent since one of the stakeholders of the developed system is directly human. In this direction, the fact that the infrastructure is blockchain-based has been evaluated as the best option for this system in the most up-to-date technology. The most important reason is that a blockchain-based system is secure, consistent, and immutable.

With the application developed on the infrastructure established with the concept of identity under personal dominance and blockchain security, users are able to manage their information and decide with whom to share and hide it. Since the information of the users is stored as signed by the authorities, it is ensured that all information is presented as proven correct. With the established system, the obligation to use different documents, identity cards, and/or applications for different institutions has been eliminated. It is possible to store all identities by combining them in a single application. Institutions and organizations have become able to approve the reliability of the shared document without the need for an institution-specific proof flow. The guarantee that the stored documents are unalterable and incorruptible demonstrates the reliability of the system and the necessity of its use.

In order to create such a system, it has been decided that the most appropriate framework for the infrastructure to be used is ACA-Py. The most important reasons for the selection ACA-Py are; multi-tenancy support that allows the creation of multiple agent instances, mediator support that allows secure messaging between agent and client over DIDComm connection, multi-Indy ledger support that allows the ability to use multiple Indy ledgers for resolving a DID by the ACA-Py agent, scaled deployment support that allows to establish and change the running ACA-Py and storage components capacity, and authentication support which is required to guarantee security.

Afterward, performance measurement was made between Askar and Indy for the selection of the wallet to be used. According to these measurements, performance and inconsistency problems exist for the Indy wallet. With Askar, the DB connection implementation has been completely changed, so the performance and consistency problems have been resolved. Askar wallet is more consistent and faster. Compared to Askar, start-up time for an agent, duration of creating a connection between two agents, publication of a schema definition and a credential definition utilizing the published schema duration over an agent, and average credential exchange times between two agents when issuing credentials are all less than an Indy wallet. Because of these results, Askar Wallet is used in the system.

When ACA-Py and Aries Go are compared to select a framework; ACA-Py is chosen over Aries Go because Aries Go's development is still ongoing and it is not yet product-ready. Aries Go supports a number of critical issues, such as W3C credentials, universal wallet, mediator, and BBSBLS Signature, but its development is not yet complete and it is not yet product-ready.

6.2 Future Work

Since the infrastructure utilized in this study is a new technology, it is quite open to development and change. In the blockchain infrastructure, the SSI system is currently being improved.

First of all, in this study, we used the standard Indy credentials (AnonCreds). As a consequence of the enhancements, ACA-Py now supports W3C standard verifiable jsonld credentials. Unlike Indy credentials, JSON-LD credentials can be issued without a schema or credential definition. Using Linked Data Contexts, everything required to issue the credential is embedded within the credential itself. As a development of the current system, the W3C standard verifiable credentials will be used to assign schema-recognizable identities to users.

While ACA-Py is selected as the framework, the primary reason is that the Aries go framework is not yet product-ready. Although the development is not completed, multitenancy and authentication properties can be implemented and the infrastructure can be changed with the Aries Go framework. The most important reasons for replacing the infrastructure with Aries Go are that it supports a universal wallet, different did methods can be used and it is ledger agnostic. It is possible to use the "ion" method that operates on the Bitcoin network by making use of side tree protocols. Despite being the technique with the most legitimacy, ion transactions are painfully slow because they take place on the Bitcoin network. Because of this, a more efficient method is able to be developed across various blockchain networks (such as Avalanche, Near, and Solana).

Lastly, the implementation of support for the bounded BBSBLS signature that enables verifying the prover is currently under development for the ACA-Py framework. After the development of this feature has been finished, it will be incorporated into the system, and a feature will be introduced that will allow verifying the users within the proof process.

REFERENCES

Akande, A. (2018). Disruptive power of blockchain on the insurance industry. *Master's Thesis, Universiti of Tartu, Institute of Computer Science, Software Engineering Curriculum*.

Akash Takyar. (2023). *A COMPARISON OF VARIOUS BLOCKCHAIN PROTOCOLS*. Https://Www.Leewayhertz.Com/Comparison-of-Blockchain-Protocols/.

Baars, D. S. (2016). *Towards self-sovereign identity using blockchain technology*. http://essay.utwente.nl/71274/

Baliga, A. (2017). Understanding Blockchain Consensus Models [White Paper]. *Semanticscholar.Org*, *4*.

Chauhan, A., Malviya, O. P., Verma, M., & Mor, T. S. (2018). Blockchain and Scalability. *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 122–128. https://doi.org/10.1109/QRS-C.2018.00034

Cocco, L., Tonelli, R., & Marchesi, M. (2021). Blockchain and self sovereign identity to support quality in the food supply chain. *Future Internet*, *13*(12). https://doi.org/10.3390/fi13120301

Guegan, D. (2017). *Public Blockchain versus Private blockchain*. https://EconPapers.repec.org/RePEc:mse:cesdoc:17020

Houtan, B., Hafid, A. S., & Makrakis, D. (2020). A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare. *IEEE Access*, *8*, 90478–90494. https://doi.org/10.1109/ACCESS.2020.2994090

Hyperledger Foundation. (2018). *Indy SDK*. Https://Hyperledger-Indy.Readthedocs.Io/Projects/Sdk/En/Latest/Docs/Index.Html.

Joghee, S., Alzoubi, H. M., & Dubey, A. R. (2020). Decisions effectiveness of FDI investment biases at real estate industry: Empirical evidence from Dubai smart city projects. *International Journal of Scientific and Technology Research*, *9*(3).

Khalifa Toumi. (2020, October 21). *How to use Hyperledger Aries cloud Agent for a classical workflow issuer-holder-verifier*. Https://Medium.Com/@khalifa.Toumi/How-to-Use-Hyperledger-Aries-Cloud-Agent-for-a-Classical-Workflow-Issuer-Holder-Verifier-9dd595f2f847.

Kondova, G., & Erbguth, J. (2020). Self-sovereign identity on public blockchains and the GDPR. *Proceedings of the ACM Symposium on Applied Computing*, 342–345. https://doi.org/10.1145/3341105.3374066

Kulabukhova, N. V. (2019). *ZERO-KNOWLEDGE PROOF IN SELF-SOVEREIGN IDENTITY*.

Law, K. H., & Lynch, J. P. (2019). Smart City: Technologies and Challenges. *IT Professional*, *21*(6). https://doi.org/10.1109/MITP.2019.2935405

Liu, Y., Lu, Q., Paik, H. Y., & Xu, X. (2020, July 1). Design Patterns for Blockchain-based Self-Sovereign Identity. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3424771.3424802

Liu, Y., Lu, Q., Paik, H. Y., Xu, X., Chen, S., & Zhu, L. (2020). Design Pattern as a Service for Blockchain-Based Self-Sovereign Identity. *IEEE Software*, *37*(5), 30–36. https://doi.org/10.1109/MS.2020.2992783

Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems*, *107*, 841–853. https://doi.org/10.1016/j.future.2017.08.020

Mühle, A., Grüner, A., Gayvoronskaya, T., & Meinel, C. (2018). A survey on essential components of a self-sovereign identity. In *Computer Science Review* (Vol. 30, pp. 80–86). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2018.10.002

Naik, N., & Jenkins, P. (2020). Self-Sovereign Identity Specifications: Govern Your Identity through Your Digital Wallet using Blockchain Technology. *Proceedings - 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2020*. https://doi.org/10.1109/MobileCloud48802.2020.00021

Panda, S. S., Mohanta, B. K., Satapathy, U., Jena, D., Gountia, D., & Patra, T. K. (2019). Study of Blockchain Based Decentralized Consensus Algorithms.

TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 908–913. https://doi.org/10.1109/TENCON.2019.8929439

Paul Wenzel. (2022, November 18). *ACA-Py Load Test Results*. Https://Github.Com/Lissi-Id/Acapy-Load-Test-Results.

Pesonen, J., & Horster, E. (2012). Near field communication technology in tourism. *Tourism Management Perspectives*, *4*. https://doi.org/10.1016/j.tmp.2012.04.001

Pongnumkul, S., Siripanpornchana, C., & Thajchayapong, S. (2017). Performance Analysis of Private Blockchain Platforms in Varying Workloads. *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 1–6. https://doi.org/10.1109/ICCCN.2017.8038517

Province of British Columbia. (2021). *Aries Cloud Agent Python Code Documentation*. Https://Aries-Cloud-Agent-Python.Readthedocs.Io/En/Latest/.

Shuaib, M., Alam, S., Shabbir Alam, M., & Shahnawaz Nasir, M. (2021). Self-sovereign identity for healthcare using blockchain. *Materials Today: Proceedings*. https://doi.org/10.1016/j.matpr.2021.03.083

Shuaib, M., Hassan, N. H., Usman, S., Alam, S., Bhatia, S., Mashat, A., Kumar, A., & Kumar, M. (2022). Self-Sovereign Identity Solution for Blockchain-Based Land Registry System: A Comparison. In *Mobile Information Systems* (Vol. 2022). https://doi.org/10.1155/2022/8930472

Soltani, R., Nguyen, U. T., & An, A. (2021). A Survey of Self-Sovereign Identity Ecosystem. *Security and Communication Networks*, *2021*, 1–26. https://doi.org/10.1155/2021/8873429

Sovrin Foundation. (2018, December 6). *Sovrin*. https://sovrin.org/faq/what-is-self-sovereign-identity/

Stockburger, L., Kokosioulis, G., Mukkamala, A., Mukkamala, R. R., & Avital, M. (2021). Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation. *Blockchain: Research and Applications*, *2*(2). https://doi.org/10.1016/j.bcra.2021.100014

Stokkink, Q., & Pouwelse, J. (2018). Deployment of a Blockchain-Based Self-Sovereign Identity. *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)*, 1336–1342. https://doi.org/10.1109/Cybermatics_2018.2018.00230

The Linux Foundation. (2022). *About Hyperledger Foundation*. Https://Www.Hyperledger.Org/About.

van Bokkem, D., Hageman, R., Koning, G., Nguyen, L., & Zarin, N. (2019). *Self-Sovereign Identity Solutions: The Necessity of Blockchain Technology*.

Waraporn, N., Sithiyavanich, M., Jiarawattanasawat, H., & Pakchai, N. (2009). Virtual credit cards on mobile for M-commerce payment. *Proceedings - IEEE International Conference on e-Business Engineering, ICEBE 2009; IEEE Int. Workshops - AiR 2009; SOAIC 2009; SOKMBI 2009; ASOC 2009*. https://doi.org/10.1109/ICEBE.2009.40

Windley, P. J. (2021). Sovrin: An Identity Metasystem for Self-Sovereign Identity. *Frontiers in Blockchain*, *4*. https://doi.org/10.3389/fbloc.2021.626726

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*. https://doi.org/10.1109/BigDataCongress.2017.85