Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://dergipark.org.tr/pub/jaida

Vol. 3, No. 1, pp. 1-16, 2023
Research Article
Received: 24/04/2023 Accepted: 25/05/2023

A Case Study for Mobile Wallet Implementation in Self-Sovereign Identity Infrastructure

Aslı BAHCE, Semih UTKU *

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. 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 accurately and reliably has become one of our most important needs today. 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. This study will explain mobile application processes created on SSI infrastructure. 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. This study, it is aimed to contribute to the general operating processes of those who will develop the SSI infrastructure. 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: Blockchain; consistency; mobile app; reliability; self-sovereign identity.

1. 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 prove our identity to the authorities. Although this problem is seen in many areas [1], 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 [2]. In addition, we may need to show documents to prove our identity. The concept of SSI, 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 [3]. 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 important needs in today's world where data storage and presentation become difficult. Today, when city governments make serious investments and breakthroughs in technology [4], smart cities have begun to form [5]. 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 this automation. 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 the 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 control in the mobile application, will be able to enter the event by proving both their membership type and age at once, with a simple application

*Corresponding author

ASLI BAHCE; Dokuz Eylül University, Natural and Applied Sciences, Computer Engineering, Türkiye; e-mail: 2015901300@ogr.deu.edu.tr; 0000-0002-7669-2678

SEMİH UTKU; Dokuz Eylül University, Faculty of Engineering, Computer Engineering Department, Türkiye; e-mail: semih@cs.deu.edu.tr; 0000-0002-8786-560X

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. This study will explain end-to-end mobile application processes created on SSI infrastructure. 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. Within the scope of the study, ACA-Py - Indy and ACA-Py - Aries Go frameworks were compared, and a contribution was made to the literature on their use and features in a real system. In addition, Indy and Askar wallet performance comparisons are made and the advantages that come with Askar are stated.

The details of the proposed solution and definitions are given in Sections 3 and 4. The next Section presents the identification of the problem and the related work. Section 4 also presents an integration example. Section 5 presents the results and discussion and finally, the last section depicts the conclusion and future work.

2. Related Works

Even though SSI is a very old notion, there have been substantial studies on the topic since 2016 according to the literature [6]. 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 [7]. 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 to meet the SSI and Zero-Knowledge-Proof (ZKP) techniques [8]. Hyperledger-Fabric and Hyperledger-Indy are the most up-to-date technologies used in blockchain, mainly in the areas of SSI and ZKP.

In addition, the Verus blockchain solution also produces solutions on SSI, 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 SSI 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 SSI 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, [9] examined the evolving landscape of SSI 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 an SSI 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 an SSI solution on average, compared to other alternatives.

Due to the sensitive nature of the data, the majority of SSI 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 [10] found that blockchain technology, with its distributed and decentralized structure, can offer significant benefits in healthcare. By empowering patients with control over their data and an SSI, blockchain has the potential to revolutionize the field. 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 Electronic Health Record (EHR) and Patient Health Record (PHR) systems are crucial in realizing a decentralized and self-governing healthcare ecosystem. While prior research, as noted by [11], has primarily focused on exploring the technological capabilities of such solutions, there remains a gap in understanding their broader impact on the healthcare ecosystem and the entire value chain. 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 privacy protection and the right to be forgotten. Although this issue has been addressed by current solutions, none have offered a conclusive solution or proof of concept.

[12] argue that blockchain adoption over the traditional client-server paradigm is motivated by the desire to achieve greater reliability, stability, timeliness, and productivity within legal systems. This study suggests that SSI can enhance compliance with regulations and revolutionize data processing practices in healthcare by empowering patients to manage their identity information. By providing patients with control over their data, healthcare providers can become more service-oriented and develop new products and applications. The cost-effectiveness of SSIs can be seen in their ability to eliminate inefficient onboarding processes and provide fast data access and authentication. Moreover, the reliability and accuracy of data in food supply chains are critical. In the study conducted by [13], they present a practical use case that demonstrates the need for a system that ensures full visibility of process and food certifications. The researchers propose an SSI-based model that aims to address the storage and accessibility of these certifications, as well as appropriate eligibility verification. The system assumes that the certifying body issues a certificate by storing it in the InterPlanetary File System (IPFS) and only stores key information on the blockchain when a user (holder) seeks a quality certification for their product or process. 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. [14] 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 investigates the application of a blockchain-based decentralized identity management system leveraging the SSI framework to provide high-level security. The authors created a low-fidelity prototype utilizing the Hyperledger Indy blockchain to demonstrate how individuals can manage their identity credentials more effectively. 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. Another example study was carried out in the field of architecture [15], They compiled 12 design models for blockchain-based self-driving identity systems, aimed at helping architects understand and apply concepts in system design. The authors divided these patterns into three groups: key management patterns, decentralized identifier management patterns, and credential design patterns. On the other hand, there are studies concerning claim-based models. [16] 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 the other hand, Baars (2016) has investigated the possibility of SSI, which allows users to control their digital identity. In their case study, they explored a solution for attribute exchange, that allows Know Your Customer (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?

In another study on claim-based models, [18] proposes a solution that obtains legally valid identity at the passport level, which requires third-party truth attestations. The proposed solution is the world's first decentralized digital passport and authentic peer-to-peer identity commons that operates on a permissionless basis. [19] provide an overview of the Sovrin SSI 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). The authors suggest that SSI can meet GDPR requirements by safeguarding data subjects' privacy while ensuring compliance with the regulation. [20] 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. A comprehensive SSI platform architecture supporting the design-as-a-service (DPaaS) was proposed in the [21] study. The authors determined the lifecycles of the three main SSI objects (keys, identifiers, and credentials) and provided a range of SSI services, including key management and credential authentication, supported by two transaction classifications.

In Table 1, examinations were made based on the most recent and current studies. At this point, it is seen that our work is more valuable than other studies, with the examinations and measurements made in different frameworks and wallets. After that, a resource has been created to guide those who will work in this field.

Table 1. *Other studies comparison.*

	[12]	[13]	[14]	[18]	[20]	Proposed System
Application Area	Healthcare	Food supply chain	Public transportation	Citizen	N/A	Citizen
SSI	Yes	Yes	Yes	Yes	Yes	Yes
Blockchain	Yes	Yes	Yes	Yes	Yes	Yes
Network	N/A	Ethereum	Sovrin	N/A	Ethereum & Sovrin	Sovrin
Storage Provider	N/A	IPFS	Indy Wallet	N/A	Digital Wallet & Edge Wallet	Askar Wallet
Comparison	No	No	No	No	Yes	Yes
Performance Measurement	No	No	No	Yes	No	Yes

3. Methodology

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. Blockchain technology has emerged as a key underlying infrastructure for SSI solutions, with Ethereum blockchain being one of the popular choices. 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. The DID contains an encrypted link to a signed document by an authority, which includes information related to service endpoints and authorized public keys. The followings are the elements of SSI:

- A *decentralized identifier* (DID) is equivalent to a unique personal identity. Document issuer will create DIDs in a censorship-resistant blockchain environment. There will be an encrypted link of the DID to a signed document by an authority.
- *Verifiable credentials* attest information to a specific DID. These are the important standards of any SSI ecosystem. For example, a digital driving license that allows you to drive cars is one example of a verifiable credential.
- *Hyperledger Indy* is a cutting-edge distributed ledger software. 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.
- *Sovrin* is an open-source identity network built on distributed ledger technology.
- *Hyperledger Indy SDK*, which provides a distributed-ledger-based foundation for SSI.
- *Hyperledger Aries Cloud Agent Python* (ACA-Py) is a foundation for building decentralized identity applications and services running in non-mobile environments.

3.1. 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. The municipal theater center client opens the "Institution Web Application", 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. The 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. The user's identity screen is refreshed and the user can see a 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 1.

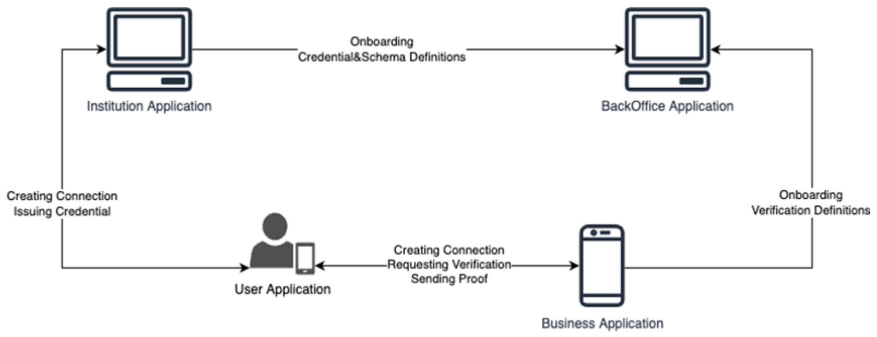

Figure 1. *System high-level flow.*

3.2. System architecture

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. 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 of 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.

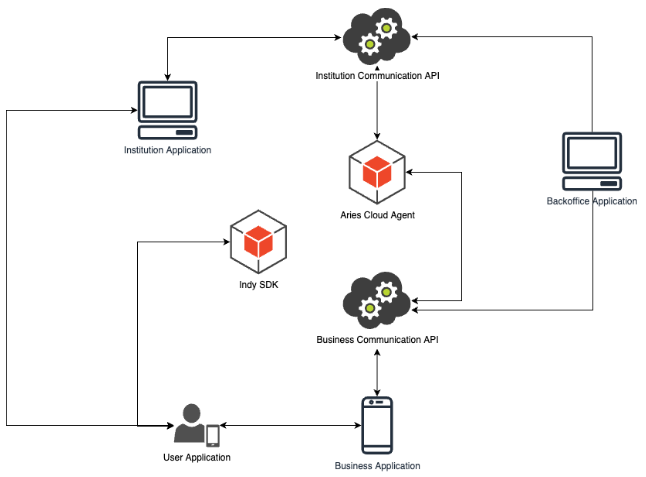

Figure 2. *System high-level architecture.*

Within the scope of this study, an architecture (Figure 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.

3.2.1. Sovrin components

A public register for Credential Definitions, Schema definitions, and Issuer public keys 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 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 can establish that this data is reliable.

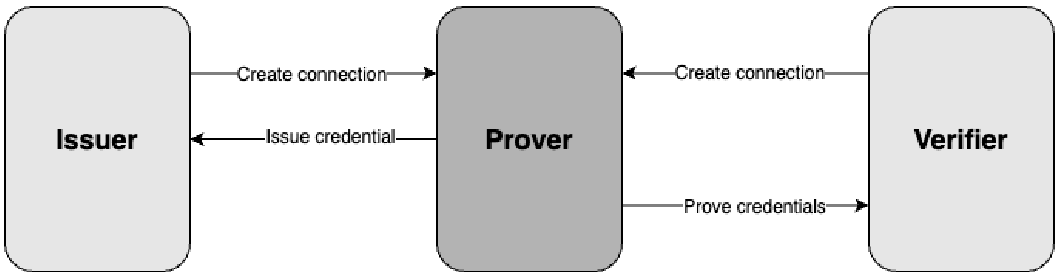

Figure 3. *Sovrin components.*

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. 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.

3.2.2. Workflow description based on Indy

The ledger is designed to store Identity Records describing Ledger Entities. Public Keys, Service Endpoints, Credential Schemas, and Credential Definitions are examples of Identity Records. Each Identity Record is connected with a single DID. Multiple DIDs can be owned by each Identity Owner 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 used by a government to publish 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.

The generation of a DID that is known to the Ledger itself constitutes an Identity Record (NYM transaction). The NYM transaction can be used for the generation of new DIDs known to the ledger, the establishment and rotation of a verification key, and the setting and modification of roles. 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. Because the ledger is accessible to the public, anyone who wishes to publish DIDs must first get the role of Trust Anchor on the ledger. A Trust Anchor is a person or organization that is already known to the ledger and can help others bootstrap themselves into the system.

Adding the Trust Anchor role to the ledger is the first step in being able to record transactions on the ledger. Institutions will require the Trust Anchor role on the distributed ledger to generate Verinyms and Pairwise-Unique Identifiers for Users. To become a Trust Anchor, you must contact a person or organization with the Trust Anchor role on the blockchain. In the empty test ledger, only NYMs with the Steward role will be utilized, but all Stewards are automatically Trust Anchors. A connection to the Indy nodes pool is required to write and view 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.

3.2.3. Initialize backoffice wallet

The Backoffice 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 DIDs, keys, etc. To store the Steward's DID and signkey, the agent must first construct a named wallet by invoking "wallet.create_wallet." The designated wallet can then be opened using "wallet.open_wallet." This call returns the wallet handle that can be used in further libindy calls to refer to the opened wallet. After opening the wallet, a DID record can be created in this wallet by executing "did.create_and_store_my_did", which returns the generated DID and the verkey portion of the generated key. The signkey component for this DID will also be saved in the wallet, but it cannot be read directly. Initialize Backoffice flow is shown in Figure 4.

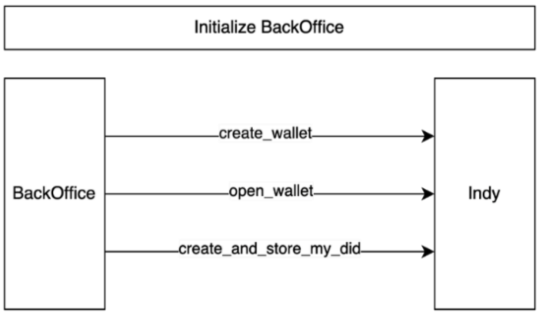

Figure 4. *Initialize backoffice flow.*

3.2.4 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.

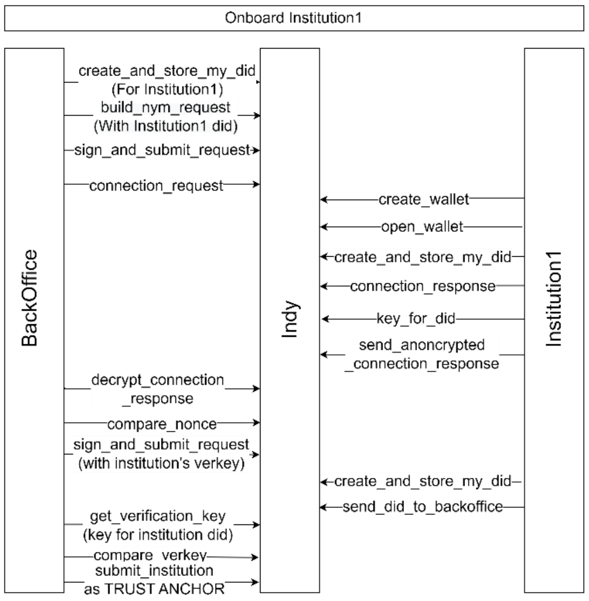
Onboard Institution1
BackOffice
Indy
Institution1
create_and_store_my_did (For Institution1)
build_nym_request (With Institution1 did)
sign_and_submit_request
connection_request
create_wallet
open_wallet
create_and_store_my_did
connection_response
key_for_did
send_anoncrypted _connection_response
decrypt_connection _response
compare_nonce
sign_and_submit_request (with institution's verkey)
create_and_store_my_did
send_did_to_backoffice
get_verification_key (key for institution did)
compare verkey
submit_institution as TRUST ANCHOR

Figure 5. *Institution onboarding flow.*

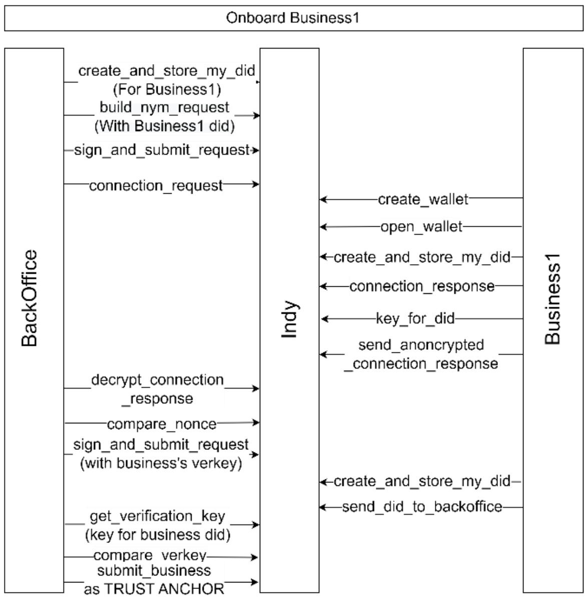
Onboard Business1
BackOffice
Indy
Business1
create_and_store_my_did (For Business1)
build_nym_request (With Business1 did)
sign_and_submit_request
connection_request
create_wallet
open_wallet
create_and_store_my_did
connection_response
key_for_did
send_anoncrypted _connection_response
decrypt_connection _response
compare_nonce
sign_and_submit_request (with business's verkey)
create_and_store_my_did
send_did_to_backoffice
get_verification_key (key for business did)
compare verkey
submit_business as TRUST ANCHOR

Figure 6. *Business onboarding flow.*

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. Onboarding processes for institutions and businesses are the same. The flow for onboarding that are shown in Figure 5 and Figure 6; same for the both institutions and businesses:

3.2.5. Creating schema and credential definitions

Credential Schema is the foundational semantic framework that describes the set of possible attributes for a Credential. There is no way to update an existing Schema. If the Schema is to be modified, a new Schema with a different version or name must be generated. Any Trust Anchor can generate and save a Credential Schema in the Ledger. Backoffice, for example, generates and publishes 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 is shown in Figure 7.

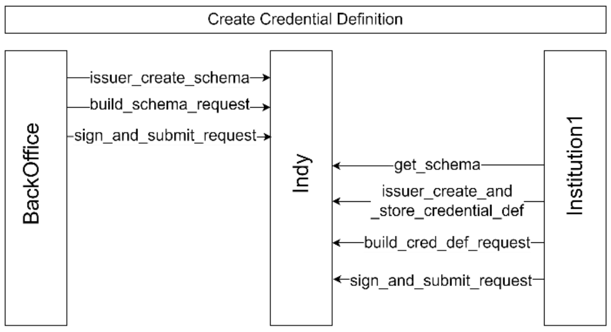

Figure 7. *Create credential definition flow.*

3.2.6. Issuing credential

A credential is an identity-related piece of information. This is assertedly accurate information. A credential is issued by a credential issuer. An issuer may be any identity owner identified by the Ledger, and any issuer may issue a credential for any identity owner it can identify. 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 used by a Prover to ensure that a credential applies exclusively to them. The Master Secret is an input that 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 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 8.

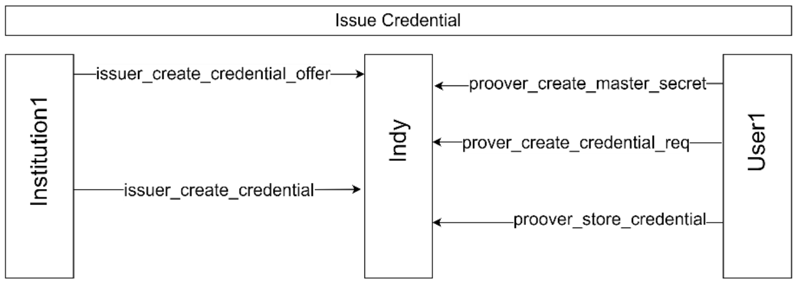
Issue Credential
Institution1
Indy
User1
issuer_create_credential_offer
issuer_create_credential
proover_create_master_secret
prover_create_credential_req
proover_store_credential

Figure 8. *Issue credential flow.*

3.2.7. Proof

When a business wants to make a proof, it should send a connection invitation to the user. The process is different than backoffice-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 Schema and Credential Definition 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 Schema and Credential Definition 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 9.

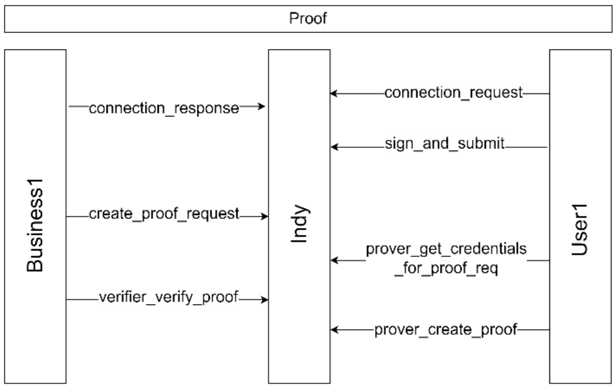

Figure 9. *Proof flow.*

3.3 Application of ACA-Py

A single instance of an Aries agent consists of two components: the agent itself and a controller. Aries agent structure is depicted in Figure 10.

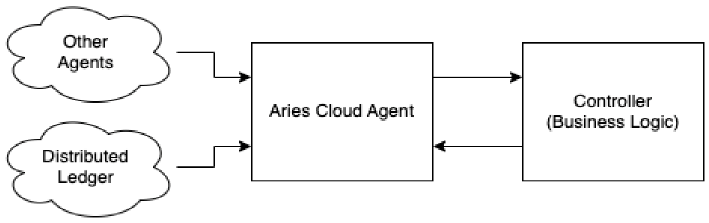

Figure 10. *Aries agent structure.*

Aries Cloud Agent is responsible for all of the core Aries functionality, including interacting 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 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.

3.3.1. 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 Backoffice instance.

3.3.2. Initialize backoffice 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 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.

3.3.3. Onboarding of institutions and businesses

For both onboarding process, a tenant registration operation is started to create wallets. Using the [post] /multitenancy/wallet endpoint, the backoffice 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 “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 in 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.

3.3.4. Creating schema and credential definitions

As defined before, institutions are issuers of the system. Through the backoffice, 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.

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.

3.3.5 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.

3.3.6. 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. 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, business verifies the identity of the user and uses that identity according to the need.

4. Results

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. Table 2 defines the ACA-Py properties and in Table 3 comparisons are given between ACA-Py and Indy.

Table 2. *ACA-Py properties.*

Property	Definition
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 info.
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	Allows deployments in scaled environments, such as Kubernetes systems, in which ACA-Py and its storage components can be horizontally scaled to accommodate the load.
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

Table 3. *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.

4.1. ACA-Py Askar vs Indy performance

Table 4 shows the results of Askar and Indy performance of interacting agents. It is created from the load test results made and announced by the community.

Table 4. *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 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.

Table 5. *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

4.2. ACA-Py vs Aries Go

Hyperledger Aries Framework Go allows communication and data exchange based on interoperable 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, the ACA-Py was preferred because Aries Go's development is still ongoing and it is not product ready. A comparison of ACA-Py and Aries Go frameworks is presented in Table 6.

Table 6. *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)

5. Conclusions

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 selecting ACA-Py is; 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.

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 sidetree 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 the various blockchain networks (such as Avalanche, Near, 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.

Declaration of Interest

The authors declare that there is no conflict of interest.

Author Contributions

Aslı Bahce: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft, Results and discussions.
Semih Utku: Methodology, Validation, Writing - review & editing, Supervision.

References

[1] N. Waraporn, M. Sithiyavanich, H. Jiarawattanasawat, and N. Pakchai, “Virtual credit cards on mobile for M-commerce payment,” in Proceedings - IEEE International Conference on e-Business Engineering, ICEBE 2009; IEEE Int. Workshops - AiR 2009; SOAIC 2009; SOKMBI 2009; ASOC 2009, 2009. doi: 10.1109/ICEBE.2009.40.

[2] J. Pesonen and E. Horster, “Near field communication technology in tourism,” Tour Manag Perspect, vol. 4, 2012, doi: 10.1016/j.tmp.2012.04.001.

[3] M. Shuaib et al., “Self-Sovereign Identity Solution for Blockchain-Based Land Registry System: A Comparison,” Mobile Information Systems, vol. 2022. 2022. doi: 10.1155/2022/8930472.

[4] S. Joghee, H. M. Alzoubi, and A. R. Dubey, “Decisions effectiveness of FDI investment biases at real estate industry: Empirical evidence from Dubai smart city projects,” International Journal of Scientific and Technology Research, vol. 9, no. 3, 2020.

[5] K. H. Law and J. P. Lynch, “Smart City: Technologies and Challenges,” IT Prof, vol. 21, no. 6, 2019, doi: 10.1109/MITP.2019.2935405.

[6] R. Soltani, U. T. Nguyen, and A. An, “A Survey of Self-Sovereign Identity Ecosystem,” Security and Communication Networks, vol. 2021, pp. 1–26, Jul. 2021, doi: 10.1155/2021/8873429.

[7] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” Future Generation Computer Systems, vol. 107, pp. 841–853, Jun. 2020, doi: 10.1016/j.future.2017.08.020.

[8] N. V Kulabukhova, “ZERO-KNOWLEDGE PROOF IN SELF-SOVEREIGN IDENTITY,” 2019.

[9] D. van Bokkem, R. Hageman, G. Koning, L. Nguyen, and N. Zarin, “Self-Sovereign Identity Solutions: The Necessity of Blockchain Technology,” Apr. 2019.

[10] B. Houtan, A. S. Hafid, and D. Makrakis, “A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare,” IEEE Access, vol. 8, pp. 90478–90494, 2020, doi: 10.1109/ACCESS.2020.2994090.

[11] A. Akande, “Disruptive power of blockchain on the insurance industry,” Master’s Thesis, Universiti of Tartu, Institute of Computer Science, Software Engineering Curriculum, 2018.

[12] M. Shuaib, S. Alam, M. Shabbir Alam, and M. Shahnawaz Nasir, “Self-sovereign identity for healthcare using blockchain,” Mater Today Proc, Mar. 2021, doi: 10.1016/j.matpr.2021.03.083.

[13] L. Cocco, R. Tonelli, and M. Marchesi, “Blockchain and self sovereign identity to support quality in the food supply chain,” Future Internet, vol. 13, no. 12, Dec. 2021, doi: 10.3390/fi13120301.

[14] L. Stockburger, G. Kokosioulis, A. Mukkamala, R. R. Mukkamala, and M. Avital, “Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation,” Blockchain: Research and Applications, vol. 2, no. 2, 2021, doi: 10.1016/j.bcra.2021.100014.

[15] Y. Liu, Q. Lu, H. Y. Paik, and X. Xu, “Design Patterns for Blockchain-based Self-Sovereign Identity,” in ACM International Conference Proceeding Series, Association for Computing Machinery, Jul. 2020. doi: 10.1145/3424771.3424802.

[16] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, “A survey on essential components of a self-sovereign identity,” Computer Science Review, vol. 30. Elsevier Ireland Ltd, pp. 80–86, 2018. doi: 10.1016/j.cosrev.2018.10.002.

[17] D. S. Baars, “Towards self-sovereign identity using blockchain technology.” Oct. 2016. [Online]. Available:

http://essay.utwente.nl/71274/

[18] Q. Stokkink and J. Pouwelse, “Deployment of a Blockchain-Based Self-Sovereign Identity,” in 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, Jul. 2018, pp. 1336–1342. doi: 10.1109/Cybermatics_2018.2018.00230.

[19] G. Kondova and J. Erbguth, “Self-sovereign identity on public blockchains and the GDPR,” in Proceedings of the ACM Symposium on Applied Computing, Association for Computing Machinery, Mar. 2020, pp. 342–345. doi: 10.1145/3341105.3374066.

[20] N. Naik and P. Jenkins, “Self-Sovereign Identity Specifications: Govern Your Identity through Your Digital Wallet using Blockchain Technology,” in Proceedings - 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2020, 2020. doi: 10.1109/MobileCloud48802.2020.00021.

[21] Y. Liu, Q. Lu, H. Y. Paik, X. Xu, S. Chen, and L. Zhu, “Design Pattern as a Service for Blockchain-Based Self-Sovereign Identity,” IEEE Softw, vol. 37, no. 5, pp. 30–36, Sep. 2020, doi: 10.1109/MS.2020.2992783.