SSI-Federation: Facilitating Identity Federation using Self Sovereign Identity for Web-services

by

Sajid Imam Mahir
20101138
Navid Alvi Ahsan
20101377
Md. Mehrab Hasan Eshan
20101498

A Thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sajid Imam Mahir
20101138

Navid Alvi Ahsan
20101377

Md. Mehrab Hasan Eshan
20101498

Approval

The thesis titled "SSI-Federation: Facilitating Identity Federation using Self Sovereign Identity for Web-services" submitted by

1. Sajid Imam Mahir(20101138)
2. Navid Alvi Ahsan(20101377)
3. Md. Mehrab Hasan Eshan(20101498)

of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering in May, 2024.

Examining Committee:

Supervisor:
(Member)

Md Sadek Ferdous, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

SSI-Federation: Facilitating Identity Federation using Self Sovereign Identity for Web-services

Abstract

Identity federation means entrusting an entity's online identity verification to an external organization. Identity Federation's basic concept is that an IdP or Identity provider ensures an entity's identity to the SP or the Service Provider an entity that provides web service. This is an old concept having the issue of how securely the information will be gathered and stored. To provide security of personal information and to get an overall convenience efficiently Self-sovereign identity or SSI is used. SSI is different from any other verification system due to its peer-to-peer decentralized system with the help of blockchain. This process provides an entity full control of how much personal information they are sharing and who they are sharing it with, with the convenience of service access without login credentials. This reduces the dependency on a specific third party making the process more secure whilst ensuring proper privacy over their data. In SSI like the Identity Federation, there are also two entities other than the user which are Issuers and Verifiers where issuers are trusted credential providers, and the Verifiers are trusted to verify them when requested. Still, the issue here is that there is no connection between the Issuer and the Verifier which concerns the issue of trust among these two entities. We provide a solution to both of these problems by first using SSI as the base model and then enabling the Issuer and Verifier of it to establish trust among themselves before the user requests a service through SSI. For this to succeed the Verifier will also play the role of the SP and the Issuer can be thought of as the IdP. This hybrid system of ours contains an external trust layer over SSI which makes it function like Federated Identity by also keeping the characteristics of SSI with the help of hyperledger-based blockchain technologies.

Keywords: SSI; Identity Federation; Hyperledger; Blockchain; Fabric; Aries; Indy.

Acknowledgement

Firstly, all praise to the Great Allah, for whom our thesis has been completed without any major interruption.

Secondly, to our supervisor, Dr. Md. Sadek Ferdous, sir, for his kind support and advice in our work. He helped us with our shortcomings and whenever we needed help.

Thirdly, to Research Assistant Md. Yeasin Ali, who helped us in our thesis planning and work and also guided us as a mentor.

And finally, to our parents, without their constant support, it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The main driving force of SSI is thought to be the users [17]. As a result, till now it has not been able to provide incentives to service providers to adopt it. However, as long as service providers do not embrace an SSI solution, the overall adoption of SSI cannot be imagined. The internet is home to a vast number of individuals, the majority of whom are strangers to each other [21]. However, there is a pressing need for an identity layer to establish trust and security. Unfortunately, the existing identity management models have proven inadequate in addressing this challenge. Consequently, as the internet expands, a significant gap has emerged in the establishment of reliable and trustworthy identities, resulting in a notable increase in cybercrime. Furthermore, the existing models have also witnessed instances of privacy rights violations. To tackle these critical issues pertaining to identity management, Self-Sovereign Identity (SSI) can be a promising resolution. But suddenly removing old infrastructure is very difficult, although necessary in the long run. The system we proposed in this paper first establishes an identity federation between the Issuer and Verifier, which can also be interpreted as IdP and SP connection. This is possible through exchanging DIDs(Decentralized Identifier) of bothentities,s which at a later point would be necessary to establish trust among them before even the verifying starts of the usual SSI. We have used Hyperledger Fabric [30] and Hyperledger Aries [28] technologies both sequentially a ledger software and a library of the renowned Hyperledger Foundation which started alongside SSI itself for implementing our new steps connected with the old ones of SSI thus, resulting in a completion of our hybrid model.

1.2 Problem Statement

SSI is the latest identity management model where the user has the most power, but there is a concern about the Issuer and Verifiers if they are completely trustworthy or not, as there is no prior connection or contract among them. Assuming users want to take service from a trustworthy SP or Verifier the Issuer itself may be a fraud or leak data in that case the Verifier won't have any say.

- First problem statement is to provide an external trust layer over SSI to build a trust relation between the Issuer and Verifier which also does the job of Identity federation.

In recent years we saw how the world and its activities shifted online and brought in the necessity of registering everywhere. This just didn't increase the hassle of one's creating multiple profiles and remembering credentials but also providing personal information to all those platforms without control. The laws of identity [1] suggest that minimal identifying information should be taken from the user, as in all scenarios an e-Commerce platform does not need my birthdate, rather it can just get my age range to give me suggestions. So blindly giving all the personal information of an individual to a platform creates security concerns.

- Second problem statement is to provide users total control over their own user information and to specify the scope of using it by other entities.

- Third problem statement is to provide a secure identity model where Third parties don't have a user's raw personal data.

Single-sign-on, or SSO is the most popular federated identity service. This allows [8] a user to authenticate once to get both connectivity and access permission. Although this is a beneficial process to overcome registering on every platform, the question still remains whether a user has full access to their personal information or not. In this case, the personal information does not go directly to the Service provider but still goes to a third party [32]. Here, concerns remain if this third party maintains security standards and whether their system is vulnerable or not. Moreover, there is no solution for insiders in that organisation or even identity theft from hackers.

1.3 Research Objectives

We wish to facilitate an SSI-based model with an external Identity Federation layer with the help of blockchain and hyperledger technologies like Fabric and Aries in our paper. This should satisfy the shortcomings of the already existing Identity models and make them robust for a more user-centric distributed architecture. To make this possible, the following objectives have been developed:

- **RO1: To Construct an effective model of Identity based on pre-existing models.** The first objective of our research is to review the existing works of Federated Identity and SSI models to get a good base point for our research. Different services of them are being used with different working procedures. To ensure consistency, we will review a few of the topics to get a good idea of Identity Management.
- **RO2: Comparison Among Existing Identity Models.** The second objective of our research is to find out the issues of the existing models so that, we can identify and also mitigate those issues in our research. The analysis will give an idea of which aspects our research should focus on.

- **RO3: User-centric Identity System.** The third objective of our research is to propose a User-centric Federated Identity management system that allows users to have full control and disclosure over their personal information. This allows us to solve the problem that the existing models have.
- **RO4: Using an SSI Framework for Authentication.** Since there should not be any data siloes to ensure consistency and security the SSI framework is to be implemented with this proposal to make it decentralized. The issuer in this case doesn't store user credentials but rather only the list of trusted Verifers or service providers thus not becoming the single point of failure.
- **R05: An external layer over SSI to build more trust.** The current SSI structure doesn't connect the Issuer and Verifier beforehand. As a result, if the VC issuing or processing was done by some less trustworthy entity that could affect the integrity of SSI. For this reason, we provided an external trust layer among these entities so that this problem can also be mitigated.
- **RO6: Proposing a Proof of Concept(PoC) including Architecture, Protocol, and full implementation of a new Identity Management System.** The sixth and last objective is to finally propose a better structure for SSI and Federated Identity services so that it can mitigate any of the existing privacy issues of the usual structures.

1.4 Structure

In this research paper, we briefly discuss the issues with the present identity systems and our research objectives in Chapter 1. The background of identity management systems like Identity Federation, SSI, Blockchain, etc., along with some core concepts, have been discussed in Chapter 2. Some relevant scholarly articles and research papers have been reviewed in Chapter 3 with a comparison table. The requirements of our system and the threat model have been discussed in Chapter 4. In Chapter 5, we went into detail about the architecture of our presented system, including its full implementation, valid use-cases, and even the protocol flow for them. Chapter 6 contains our analysis based on the requirements presented and research objectives, covering how much we succeeded in fulfilling this paper. Moreover, it also includes advantages, limitations, implementation challenges, and future work based on our proposed system. Lastly, we focused on the significance of our paper to wrap up in Chapter 7.

Chapter 2

Background

Identity Management

Identity management is how user identities are managed for a specific service traditionally by the Service Providers [3]. Due to the rise in internet users new concepts of identity management have already emerged to fulfill the user's needs. Basically, identity management has two parts, firstly identifying a user uniquely by providing credentials and identifiers and, secondly authorizing them and controlling their scope of service when logged in. The identity of an entity is the representation of it on a service platform. These entities have characteristics or attributes that are considered identifiers that identify them which Figure 2.1 correctly depicts. Currently, there are three types of user-centric identity concepts and we are going to focus on two of them which are: Federated Identity and Self-Sovereign Identity (SSI).

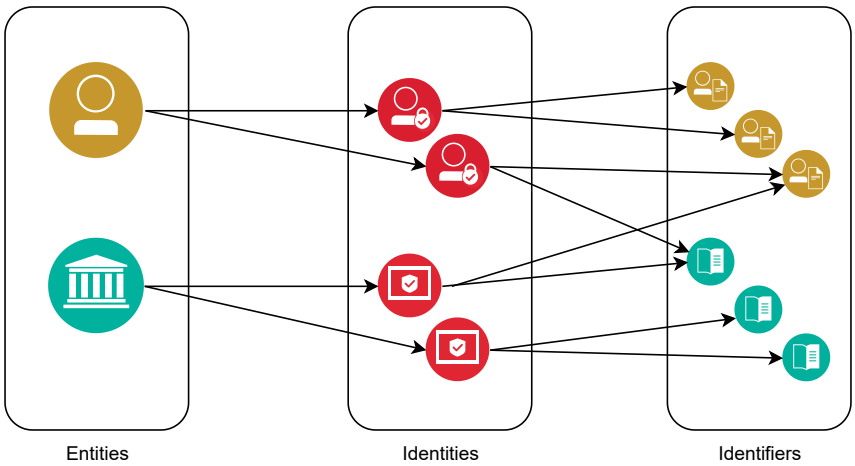

Figure 2.1: Relation between Entities, Identities, and, Identifiers. [3]

Federated Identity

Federated Identity Model was introduced in the industry to deal with some of the problems that can be found in the traditional old identity model where for each

service the user has to create an account [21]. It encompasses three main entities- IDP(Identity Provider), SP(Service Provider) or Orgs, and User. The basic idea is to create one account for an IDP and receive services from all the SPs related to the IDP the relationship between these entities can be seen in Figure 2.2. Orgs or sites with a common IDP comprise a federation and the Orgs in the federation are called relying parties. Security Assertion Mark Language (SAML) [8], (ID-FF) WS-Federation, Liberty Identity Federation, etc. are some of the popular FIM(Federated Identity Management) solutions.

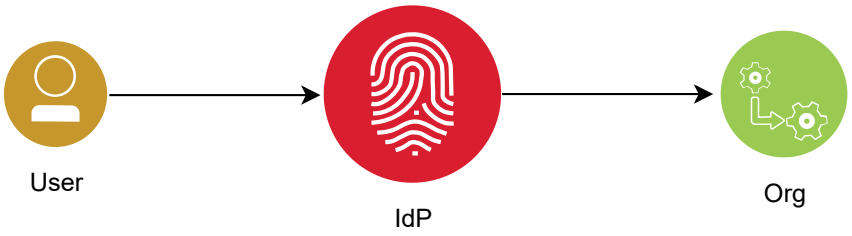

Figure 2.2: Relations involved in federated identity model. [21]

SAML

SAML [8] version 2.0 is considered an industry standard FIM protocol. SAML (Security Assertion Markup Language) version 2 is a norm of protocols mainly used in authentication and authorization information exchange between different systems. It enables SSO(Single Sign-On), which means you can access several Apps with one login which means without having to provide your credentials repeatedly. IDP and SP are the two main entities in the SAML framework. Users authenticate using IDP and a special token is generated called SAML assertion which is used on SP's side for verification. SAML assertion is like a digital identity that proves a user's identity. Each SP has an Assertion Consumer Service which receives and validates assertions from IDP. Once a user is validated, he is granted access based on the data in the assertion. Here in Figure 2.3 from [12] a basic SAML protocol flow is shown describing the communication among different entities.

OAuth

OAuth is an open standard that solely focuses on the authorization of an entity and provides federated identity to a user. OAuth introduced a layer [26] between the user and the service provider which focuses on that. This authorization layer acts as a middleman for access to service from an SP and also handles the authorization process on its own. Here, no user credentials are required rather the OAuth Server provides a token to the service provider including scope, lifetime, attributes, etc. after authorization. Although, this is widely used it does not support any encryption and signatures making it vulnerable to a lot of security threats.

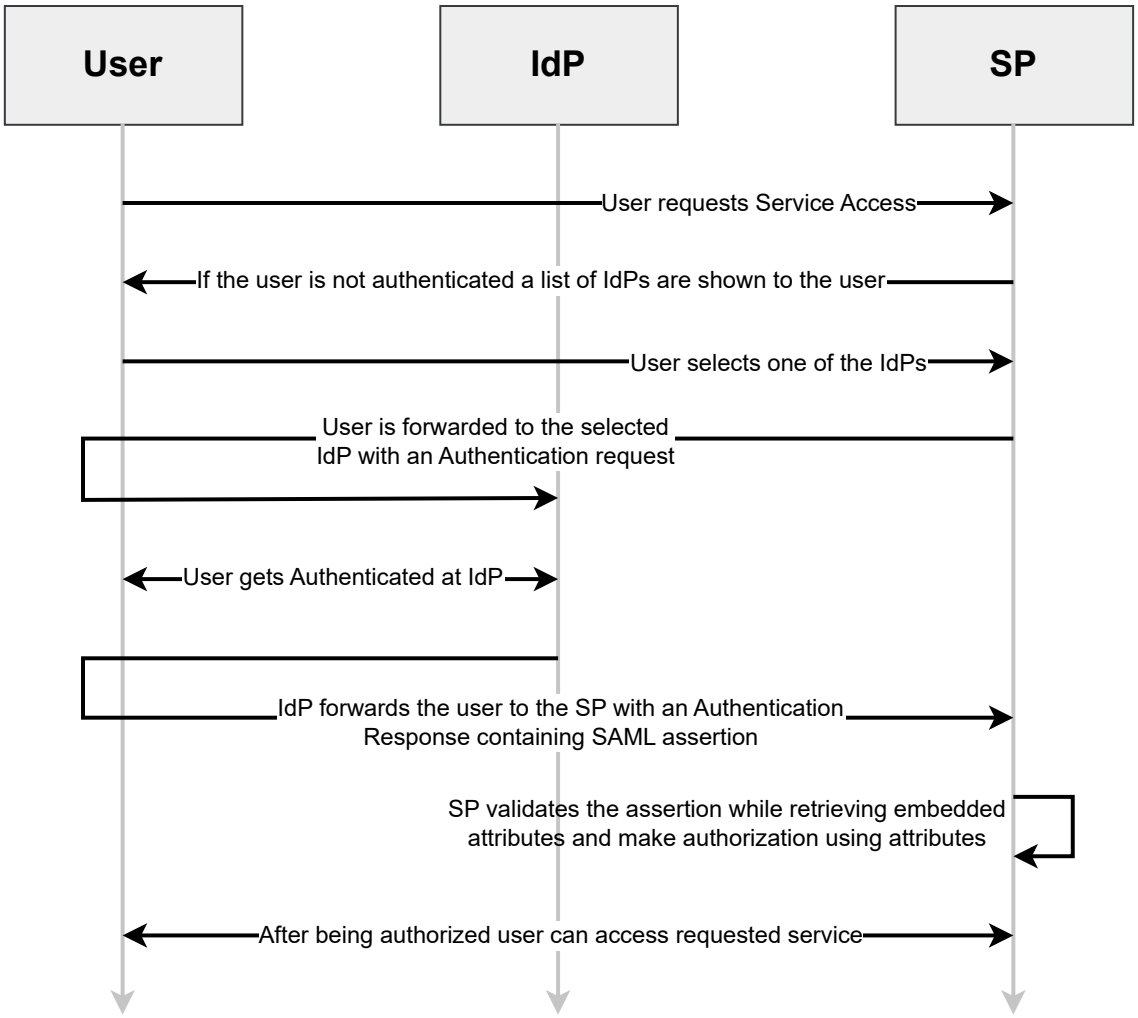

Figure 2.3: Protocol Flow in SAML. [12]

OpenID

OpenID is an open standard that acts as a trust anchor [25] to provide identity federation. Trust anchors are trusted third parties that issue statements about other entities. OpenID is similar to OAuth having the use of tokens but here, identity is given first priority rather than what the user is allowed to do. Here, different levels of entity statements chain together until a trust anchor is found. Then a verified trust chain federation policy is applied to generate metadata describing the leaf entity thus identity federation is done. Leaf entity here refers to Relying party or, Provider party defined by OpenID Connect.

SSI

Self-Sovereign-Identity (SSI) [21] represents a transformation in the control of identity or authority. Internet's rapid growth has made the need for an identity layer quite clear. Finding that identity layer has led us to SSI which revolutionized the traditional internet where control is at the center of a network to peer-to-peer network. The control or the crux of the federated identity model is with the issuers

and verifiers. On the contrary, the decentralized SSI identity model has shifted this control to the users who can exist on the network independently of others. They are not in the palm of any identity provider for their existence on the network. SSI puts the individual at the center of their digital identity. It leverages decentralized technologies to make sure that users can freely control their own data and eliminates the need for any third party.

Blockchain

Blockchain [21] is a highly secure distributed digital ledger that is not in control of any central authority. But unlike a traditional ledger, which is controlled by a single authority a blockchain is decentralized. Each transaction is considered a digital document that needs to be signed. When someone wants to add a new transaction to the blockchain, they sign it digitally using their private key. Transactions are grouped together into blocks. Once a block is created and linked to the block before it using the hash of that block, it becomes very difficult to modify it. Altering a transaction would require recalculation of the hash of the block. Since the hash of the block that is immediately before it is also part of the hash, tampering with a transaction is immediately noticeable as every new block is replicated among the peers in the network.

Verifiable Credentials

Verifiable Credentials or VCs are the very center of every SSI-based architecture [21]. A physical form of verifiable credentials can be seen in every part of our day to day life, from credit cards to licenses. But these physical credentials can be easily stolen, misplaced, or even misused by the wrong person. However, the new standard of Verifiable Credentials, or VCs, allows the physical credentials to produce digital VCs which are then used for verification anywhere using mobile phones or any online methods [21]. It has now become the full World Wide Web Consortium (W3C) open standard. Popular use cases of VCs include opening bank accounts, receiving local access passes, etc.

Decentralized Identifier

An equivalent to the Verifiable Credentials are the cryptographic Decentralized Identifiers or DIDs [21]. A DID is a unique identifier or a URI and can be either a URL or URN. It is a string of characters which identify a specific resource. By looking up or resolving a DID, the metadata which is a standardized set of information can be retrieved [21]. A DID has four core properties including the identifier being permanent (persistent), resolvable, cryptographically verifiable, and decentralized [21]. The DIDs can be useful by itself as being used as identifiers, but they can also be used by applications or used to construct advanced URLs based on a DID.

Trust in SSI

The trust triangle of Self-Sovereign Identity (SSI) [21] represents the three core entities that contribute to establishing trust within the SSI ecosystem. The entities are issuers, holders, and verifiers. Issuers issue credentials which can be government agencies, financial institutions, universities, corporations, and even individuals. Holders can request credentials from issuers and keep them in their digital wallets. When verifiers require proof from holders then the holder's agent presents the proof which contains a digital signature of the issuer. The solution for associating a public key with a user is solved by DID (Decentralized Identifiers). Trust among these three entities can be seen in Figure 2.4.

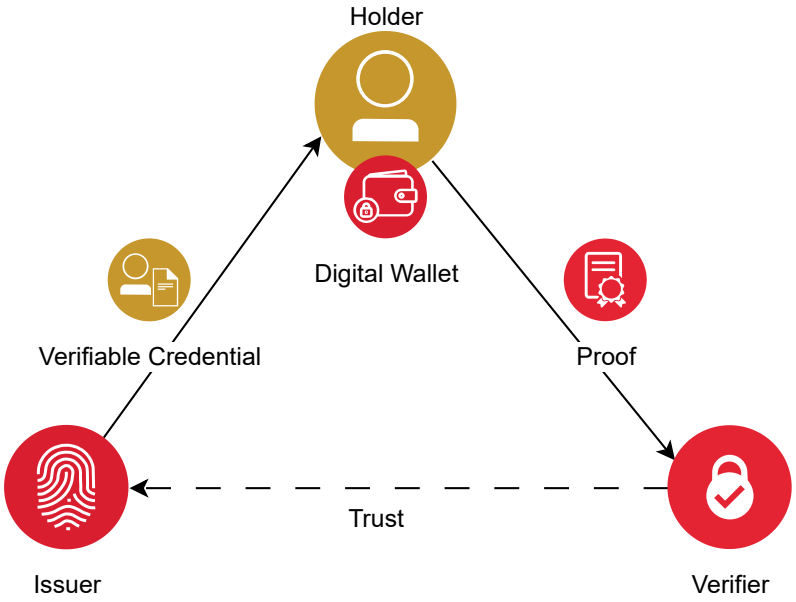

Figure 2.4: The trust-triangle in SSI ecosystem [21]

Chapter 3

Literature Review

Bangladesh has adopted web-based services like many developed countries to provide efficient services to its citizens. As these web services became really popular, issues like identity management became a headache. Moreover, the massive landscape of web-based services requires collaborations among business partners. Thus, there was a need for interoperability between systems and identity management systems and Federated Identities came to be. The main objective of the paper [9] seems to bring this FIM to the attention of different stakeholders that provide Web-enabled services in Bangladesh by discussing its advantages and use cases in the government and Higher Education Institutes, two of the most important web-enabled service sectors in Bangladesh. Finally, [9] discusses the security and privacy issues related to FIM.

The paper [18] proposes an access control model for identity management called SSIBAC. It implements decentralised authentication using blockchain and central authorization using traditional methods. To ensure the protection of user privacy which traditional centralized and federated models fail to achieve, SSI principles are taken into account. SSI is able to alleviate the impact of data breaches and provide users with full control over their data without storing it in any central database. Although the SSI authentication mechanism was properly developed, authorization and access control felt somehow lacking. This paper [18] aims to fill this gap using DID, VC, and blockchain. A prototype of the solution is tested with quite a large number of access control requests. Finally, the limitation of the SSIBAC is discussed.

The paper [13] proposes the foundations for a regulated self-sovereign-identity management system. The proposed systems encompass User, AP(Attribute Provider), SP(Service Provider), and RB(Regulatory Body), four roles. RB licenses the AP and monitors their activities. RB also maintains a blockchain where all certificate requests and certified attributes are stored. A user wallet is one of the fundamental pieces in this system which assures the secrecy and ownership of attributes by referencing the AP. When a user interacts with an SP and is asked to prove the ownership of an attribute he wants to disclose, he provides the reference to the attribute certificate which is linked with AP and utilizes related keys in his personal wallet. Trust between SP and AP is important here although the AP will never know which of the SPs the user is related to. APs do not need to trust RBs, hence, making it impossible to forge attribute certifications for RBs.

Requirements served by various IdM Systems are scattered and a clear picture of

which IdM Systems fulfill which requirements can not be found. This paper [10] presents this comparison among six different IdM Systems to remove that vagueness and find an ideal one. Firstly, all requirements were defined properly and brought under eight large requirement categories. Those were used as the metrics for comparison to find the ideal IdM system. However, it is stated that the systems analyzed did not meet the requirements to be ideal because either they failed to fulfill the requirements or were not usable in practical situations. Finally, the paper [10] ends with a concerned remarking that most IdM systems lack privacy requirements.

This paper [2] highlights the trust issues in existing identity management models. It [2] shows how complex and different trust relationships are in various identity management models and how confusing that can be for the stakeholders. The comparison between various models can serve as the basis for assessing the cost of satisfying trust requirements. Finally, personal identity management seems to be the most promising because of greater flexibility and low additional trust requirements, hence, it is suitable to combine with other models to improve user experience.

The paper [17] aims to provide a service provider-friendly SSI integration architecture. The existing SSI solutions are focused solely on user requirements. However, as long as service providers do not embrace an SSI solution the overall adoption of SSI can not be imagined. The paper [17] proposes an integration architecture that enables SPs to integrate SSI functionality into their system by defining protocols, standards, and interfaces so that they can use SSI systems to verify user identities. Their proposed gateway aims to provide seamless connectivity and interoperability between service providers and SSI platforms. The identity provider here is built over a blockchain system and the SP depends on the gateway for proper validation of claims.

SSI is currently using new blockchain technologies [19] to maintain the system of identification which is currently a really important matter. The authors of the paper [19] propose a new model where SSI and blockchain can work hand in hand with popular standards like OAuth 2.0. The whole system of authentication resides on the blockchain model and not any service provider. By using blockchain, the model shows how there is no single service provider for which the authentication depends rather it is decentralized. Moreover, as the model complies with OAuth 2.0 it is easy for previous users and is also really scalable.

Recently, there has been a rise in future possibilities of SSI due to the emergence of blockchain technology. The paper [16] shows this possibility by merging SSI with Blockchain to cover different aspects of identity management. FIM systems make the IdP a centralized entity and also pose the threat of a source error [24]. So, the paper [24] designs and implements a decentralized system to solve this problem. Blockchain technology is this decentralized system which is a ledger-based solution trained in a distributed network of peer nodes. Being added to a blockchain system the IdP can now be trusted and only attributes are stored rather than the whole data. This system has been analyzed under a sophisticated quality protocol named ProVerif.

Blockchain gives the system immutability, decentralization, and a verifiable ledger to validate online transactions. Still, problems of scalability, security, and privacy

issues remain which the paper [14] solves by providing a novel privacy-preserving solution giving users the opportunity to become anonymous and take control of their data while performing digital transactions through an SSI model shown in Figure 3.1. The reduction of intermediaries due to blockchain is creating a revolution in sectors like IoT, healthcare, and even in day-to-day life activities in a smart city. Many current sophisticated privacy technologies were used to examine this paper. The proposal here allowed us to strengthen the scenarios of blockchain privacy-preserving features.

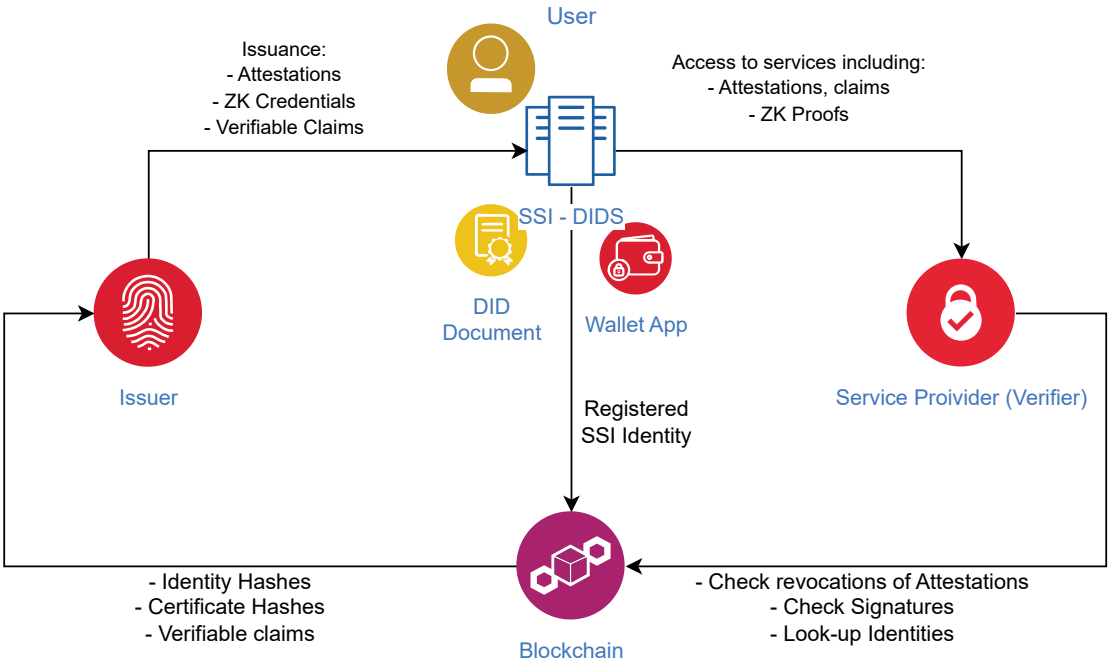

Figure 3.1: SSI Model in Blockchain [14]

Modbus an Industrial Internet of Things Protocol is one of the most widely used protocols but has the challenge of security. According to the [20] paper, a solution is possible based on SSI over hyper ledger fabric blockchain. The decentralized approach of blockchain is used here which gives a device self-custody of its identifiers and verification info. SSI has been implemented here over a Hyperledger Fabric Blockchain to maintain on-chain authentication, authorization, and other aspects of an access control system. This system also provides scalability where more than one organization is in use. To make this system fully machine compatible the authors imply to use OAuth machine-to-machine scheme.

As time goes on and with the advances of technology and digital information the importance of trustworthy identification has become necessary [15]. This paper [15] explores the realm of SSI solutions and finds out the necessities of blockchain technology in them. The internet itself doesn't have an identity protocol for which service providers come into play. There are certain problems with this structure and Federated Identities like Facebook and Google's trusted identity policy are being used. But even that is not the best solution and SSI comes here with the notion of combining DLT or, Digital Ledger Technology combined with it. Blockchain looks really promising here but there is still the possibility of private/public key pair

theft. Even keeping this in mind research shows how blockchain-based solutions provide more security than others. This paper [15] comes to the conclusion that blockchain-based technologies are a good foundation for SSI solutions.

SSI management is a revolutionary approach to the traditional login systems where the users can control their own data [22]. This system is implemented by replacing traditional usernames and passwords with credentials that are more secure. SAML is a standard for exchanging information between IdPs and SPs. The author [22] designed and developed a concept of implementing SSI on the existing infrastructure of federation identity by creating a hybrid authentication process where the verification is done via SAML. This allows the verification process to be limited to the IdP and can use SSI credentials to replace the conventional usernames and passwords.

Traditional ways of user authentication are still present in current-day web services. As a result, it still is one of the major reasons for security breaches. In response to this a new framework of SSI in web services was proposed called SSI4Web [23]. In that proposal, there is no need for passwords which gives flexibility. The framework includes a mobile wallet through which VC will be presented but, both the issuer and verifier entity is taken care of by an SSI Agent entity. The paper [23] takes advantage of a few important services or projects of Hyperledger Foundation which are Aries, Indy. It also includes a detailed protocol, architecture, and practical use cases of how and in which scenarios this SSI4Web framework can be used. Finally, it gives us an idea that this can also be out for a test to establish different models like FIM.

Federation Model allows users to share access to multiple domains and their resources without needing to log in every time [5]. SSO is one of the most common implementations of this model allowing users to access multiple websites after verifying their identity at a single website which can be achieved by using SAML. The author [5] studied the underlying trust mechanisms and put forward the concept of an extension to the standard SAML enabling the identity federation management to be more dynamic. The current frameworks do not allow the system to be dynamic and as such, SAML is used, allowing flexible and better trust decisions. SAML is a kind of framework based on XML which allows authentication between entities. SAML has an extension point known as metadata.

Recently, organizations are shifting to using a common authentication system to verify their identities for their internal services and applications [6]. This enables the users to verify their identities easily and securely without needing to authenticate for each service. With the increasing number of services, general users have to remember their passwords and usernames for each service. Additionally, each user has to sign up for each service, creating duplicate information for each user. In response to these problems, the author [6] implements a centralized SSO for different systems using SAML, allowing users to control their own data and providing flexible and secure data trade between IdPs and SPs. This also reduces the number of shared accounts, enhancing the user experience, by using the same credentials for every service.

Single Sign-On allows different companies to build a federation such that users can use a single username and password to sign into all the services of different com-

panies, eliminating the redundancy of creating multiple accounts [4]. The OASIS SAML 2.0 SSO is a new standard for SSO protocols. Companies like Google, implement their Single Sign-On based on SAML, such as the Sign-On services for Google Web Applications. In this paper, the author [4] proposes a new model correlating to the SAML-SSO model and analyzes the issues of the models with the help of SATMC, a great model that checks security protocols, identifying an unknown flaw within the Google-SSO protocol. The author also tried to recreate the identified attack to confirm the flaw.

SAML SSO is the standard in which users can log in to different services after authenticating through a single service built on SAML [7]. SAML provides a few security suggestions, but does not account for every instance. As such, a SAML SSO designer may miss a few flaws depending on the security suggestions, which can compromise the overall security of the solution itself. This was further evident by the security flaw identified in the previous paper [4]. This flaw was identified using a model checker for security flaws. In this paper, the author [7] will demonstrate that by using such model checkers, we can ensure the development of SSO solutions by ensuring the security of the solution. Model checking can identify security flaws in distributed systems and services, which in turn helps make the solution more secure.

Below in Table 3.1 we have tried to show how different research papers that we reviewed related to identity federation's features align with user entity in a tabular format. Where the properties apply we used "✓", where they do not apply we marked "X", and where we are unsure we used " $-$ ".

	Belchior et al. [18]	Ferdous et al. [9]	Coelho et al. [13]	Ferdous et al. [10]	Grüner et al. [17]	Hong et al. [19]	Yildiz et al. [22]	Ferdous et al. [23]	Our Paper
System Design	✓	X	✓	X	✓	✓	✓	✓	✓
Independent User	X	-	✓	-	✓	✓	-	✓	✓
User Control, Consent	✓	-	-	-	✓	✓	✓	✓	✓
Minimal Disclosure	✓	-	✓	-	✓	✓	-	✓	✓
Justifiable Parties	✓	✓	✓	✓	✓	✓	✓	✓	✓
Security Requirements	✓	✓	X	✓	X	✓	X	✓	✓
Protocol	✓	X	X	X	✓	X	X	✓	✓
Encrypted Channel	✓	✓	✓	✓	✓	✓	✓	✓	✓
Consortium	X	✓	✓	-	X	✓	✓	X	✓
DID	✓	X	X	X	✓	X	✓	✓	✓

Table 3.1: Comparison between various reviewed papers related to identity federation and SSI aligning with user entity.

Chapter 4

Threat Modeling & Requirements

4.1 Threat Modeling

Threat modeling is the use of the model to point out possible security issues in any system [11]. This step helps the system to be robust by finding out issues in a system even before coding it. Since, our work focuses on SSI, FIM, Hyperledger, and Blockchain where identity management and federation are fundamental things we must do threat modeling. The model we are choosing for our Threat Model is STRIDE [11], a well-renowned threat model that helps your system to maintain: legitimacy, non-repudiation, integrity, confidentiality, availability, and authorization.

- **T1-Spoofing:** An adversary can act to be an authorized user, issuer, or verifier even if it's not part of the system consortium.
- **T2-Tampering:** An adversary or issuer could change the TAL metadata of a trusted entity and allow entry into the consortium as an outsider.
- **T3-Repudiation:** An Issuer can refuse the release of VCs even though it was released for the requested Verifer or SP.
- **T4-Information Disclosure:** An unauthorized attacker might get hold of an authorized user's attributes and the Issuer might provide some attributes of an authorized user to the Verifier without the user knowing.
- **T5-Denial of Service(DoS):** Whole services can become inaccessible due to any DoS attack on the Issuer or Verifier.
- **T6-Elevation of Privilege:** If the authorization medium and process is faulty there might be unwanted authorization which may lead to unauthorized users getting the privilege of using the system.

We also considered the following threats after considering STRIDE threats:

- **T7-Replay Attack:** An attacker can take hold of a req/resp during communication and submit it to use it independently.
- **T8-Lack of consent:** The user attribute might be used without permission of the actual holder.

- **T9-Lack of control and transparency:** In the whole process of generating VC to verification of the user the user him/herself might have almost no control.

4.2 Requirement Analysis

Before implementation of the system, we needed to define a well-defined requirement analysis which is really necessary for a successful application or, system development. We have shown many functional and non-functional (security, privacy) requirements for our proposed system. Functional requirements represent what is necessary for the system to run and non-functional requirements represent the features necessary to ensure that the security threats have been taken care of.

4.2.1 Functional Requirements

- **F1:** A mechanism will be established for Issuers and Verifiers to establish trust relationships before user requests, ensuring secure interactions within the SSI framework.
- **F2:** Users will have control over their credentials and will share the necessary credentials with Issuers and Verifiers.
- **F3:** The new system will allow the Issuers to issue verifiable credentials (VCs) and the Verifiers to verify the credentials without compromising privacy.
- **F4:** The new system will be executed in a way so that the already existing Federation system and SSI-based structures don't need to change.
- **F5:** Blockchain technology should be established in such a way that it doesn't delay the process of authentication rather it makes the system robust.
- **F6:** A decentralized identity management system will be utilized based on blockchain technology.

4.2.2 Non-Functional (Security) Requirements

- **S1:** Since this system follows the SSI protocol, only trusted Issuers and Verifiers can participate in the federation which mitigates the risk of spoofing (T1). Moreover, only the trusted entities are storing the metadata, diminishing the chances of tampering (T2).
- **S2**: Establishing a mechanism to ensure that the verifiable credentials (VCs) are non-repudiable, thereby mitigating repudiation threats (T3).
- **S3:** Since SSI protocol is strictly maintained using VCs, the trusted Issuers cannot expose any attributes of the user without explicit permission from the user as only the user wallet app contains the VC. This mitigates the chances of information disclosure (T4).

- **S4:** The system must be maintained available even during potential denial of service (DoS) attacks, ensuring that trusted Verifiers have consistent access to necessary services. This mitigates Denial of Service (T5).
- **S5:** The system implements authorization mechanisms with VCs and blockchain technology to prevent unauthorized elevation of privileges, ensuring only authorized users can access the system removing the chances of elevation of privileges (T6).
- **S6:**The system provides users with complete transparency and control over their identity data throughout the process, from VC issuance to verification, mitigating issues of lack of control and transparency (T9).

4.2.3 Non-Functional (Privacy) Requirements

- **P1:** The system uses unique session tokens and timestamps to protect against replay attacks, ensuring that intercepted communications cannot be reused maliciously. This overcomes the chances of replay attacks (T7).
- **P2:** The system ensures strict SSI protocol and only allows the trusted entities to access the attributes after getting the permission of the user addressing concerns of lack of consent (T8).

Chapter 5

System Model & Architecture

5.1 Architecture & Implementation

In this section, we provide the complete architecture based on our system requirements to fulfill the features and mitigate security issues. Figure 5.1 represents our architecture in a flow where we start from the Verifier's connection with Issuer.

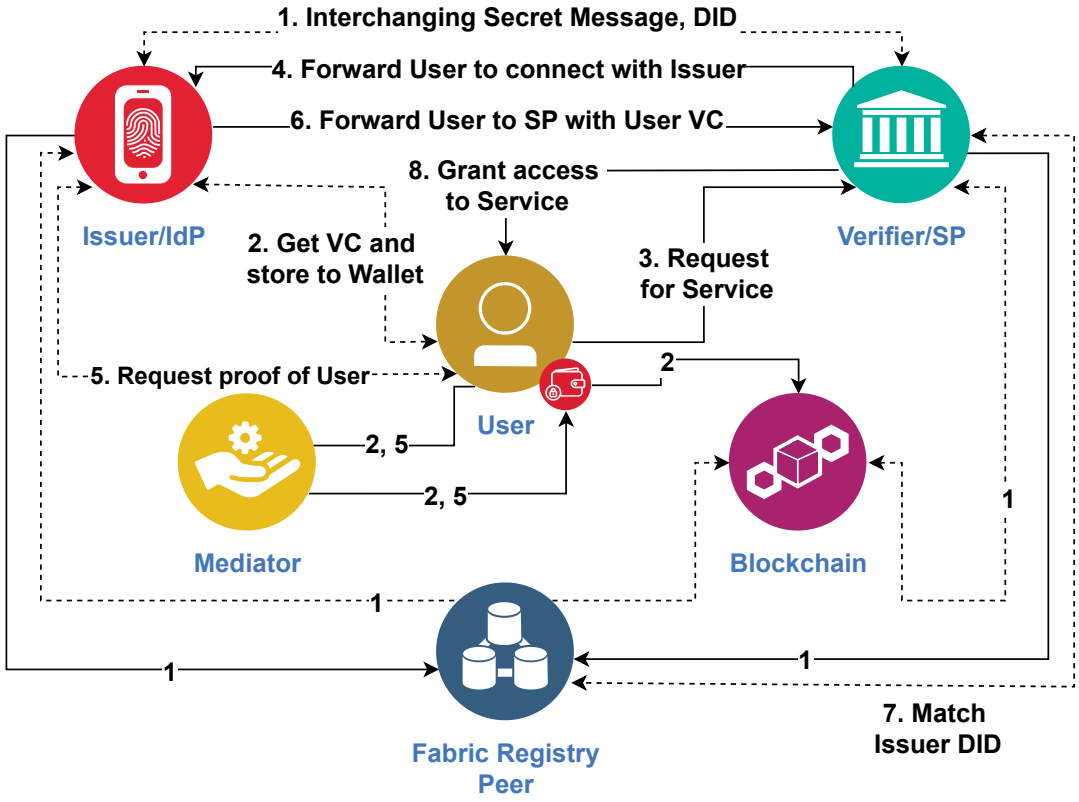

Figure 5.1: System Architecture

The main goal of our SSI-Federation system is to provide a more trusted interaction between SSI agent entities which are Issuer, and Verifier almost following the obligations of the Identity Federation to enable the users to trust both entities and access web services passwordless. As there are only a few SSI frameworks to follow we took the help of the SSI4Web framework [23] and modified it where necessary for

the base SSI part. Moreover, we even modified the Verification step of the Verifier to enable us to implement Federated Identity over SSI.

- In our SSI-Federation system, the main novelty starts at the beginning of the process where the Verifier or SP wants to connect with a certain Issuer or IdP. In normal SSI this connection is not there as this corresponds with Federated Identity but to make this connection possible Unique DID of both Issuer and Verifier needs to be shared. This DID sharing process depends on the Issuer Org and Verifier Org where they could physically agree to a contract or, online however they agree to share a `secret_ref` which only the other entity knows and also some necessary info they agreed to like Domain, Org name, etc. After knowing the `secret_ref` Verifier will send a request to the Issuer with it, its own DID, and what the VC format is through which the Users will be verified before accessing. The Issuer will first match the `secret_ref` and if it matches it will store the DID of the Verifier in its own Fabric registry. In response to this request, the Issuer will send back the `secret_ref` it was provided with to the Verifier and its own DID to successfully connect with each other agreeing on a consortium.

- The User before even accessing the Verifier will first access the trusted Issuer and according to the schema defined will fill out the necessary credentials and will get a VC for it which will be sent back to the user's Wallet App through a Mediator. Until now, the whole process has been the security extension of our system.

- Now the normal SSI architecture comes into play when the User requests a service to the SP or Verifier.

- The Verifier forwards the User to connect with the Issuer and the Issuer requests proof of the User whether it has already taken VC or not. The User in this case fills minimal verifiable credentials to be connected with the Issuer if VC was already taken, and no password is needed here.

- The Issuer upon verification of the User takes the VC of the User and forwards it to the Verifier, which also contains the DID of the Issuer.

- The Verifier resolving the VC matches the DID of the Issuer with the before stored one in its own Fabric registry to verify the User whether it took VC from a trusted Issuer or not. In normal SSI this step is not present; moreover, the Verifier would have checked the VC with the blockchain whether it matched or not. On the contrary, our SSI-Federation model skips this step which allows no User information to be read by the Verifier rather upon trusting the Issuer will complete the connection.

- Lastly, in our architecture after matching DID the Verifier will grant access to the requested resources of the User thus completing our system's architecture.

This Proof of concept was developed leveraging different frameworks and tools, including Hyperledger Aries [28], Hyperledger Indy [31] and, Hyperledger Fabric [30].

The idea of web-service access was made with the help from the SSI4Web framework [23]. Hyperledger Indy is a blockchain platform which provides tools, libraries, and components for creating decentralized identities to be used in SSI Applications. In our SSI-Federation system, we have used the Indy testbed called BCovrin Test http://test.bcovrin.vonx.io/. Hyperledger Aries is an interoperable toolkit used for developing digital credentials used in SSI applications. We have used the Hyperledger Aries Cloud Agent Python (ACA-Py) library [29] in our SSI-Federation system. In our SSI-Federation system, the issuer also acts as a SSI agent which has been developed using NodeJs [33]. This SSI agent utilizes the ACA-Py library for its SSI functionalities. Moreover, we have used the open source Aries-compliant mobile App of the Aries Mobile Agent React Native [27] project as our Mobile Wallet. Finally, we have used Hyperledger Fabric [30] to store our decentralized identifiers to our permissioned blockchain ledgers.

5.2 Use-case & Protocol Flow

Now this section shows the different use cases with their protocol flows and sequence diagrams to enact the architecture that we proposed to solve our stated issues. First, in Table 5.1 we can see the mathematical and cryptographic notation that we will use throughout this part.

5.2.1 Data Model

Table 5.1: Cryptographic Notations

Notations	Description
U	The identity holder or Service user
I	Issuer organization or Identity Provider
SP	Service Provider or Verifying organization
B	Blockchain
Cc	Consortium Chaincode
U_W	User wallet
M	Mediator
K_U^I	Public key of U for I
$K_U^{-1\|I}$	Private key of U to be used for I
K_I^U	Public key of I for U
$K_I^{-1\|U}$	Private key of I to be used for U

DID_I	Decentralised identifier of I
DID_{SP}	Decentralised identifier of SP
VC_I^U	Verifiable credential issued by I for U
N_i	A nonce
$\{\}_{\mathrm{K}}$	Encryption operation using a public key K
$\{\}_{K^{-1}}$	Signature using a private key K^1
$H(M)$	SHA-512 hashing operation of message M
$[...]_K$	Communication over an channel encrypted with key K
[...]	Communication over an unencrypted channel
[[...]]	Optional data

The raw data we have used in our proposed model has been illustrated in Table 5.2 with the appropriate description. The system we proposed works like a responsive system that gives some responses (indicated by *res* in table) upon giving some requests (indicated by *req* in table). Our system consists of seven types of *req* which are: *didReq*, *reqRegister*, *credReq*, *inviteReq*, *attrReq*, *proofReq*, *serviceReq*. The first two of these requests are made during the external identity federation layer that helps us establish the connection. To add to that the next three requests are that of the normal SSI flow but for only acquiring VC from the issuer and the last two requests are meant for the modified Web-SSI protocol we made for the flow to complete. There are also five types of responses that are executed in response to the mentioned requests.

The *didReq* contains the $secret\_ref_x$ of the Issuer or Verifier they got this while the organizations communicated with each other for the trust relationship to be established based on the DID they get. To add to that, the *didReq* also contains DID_x of the Issuer or Verifier so that while requesting for the other entity's DID they could also join the consortium on that platform. To enter the Chaincode consortium a registration request has to be executed with *reqRegister*. This request consists of the requesting entity's $domain_x, org_x, DID_x$ so that it can remain in the Chaincode registry and join the consortium. The *credReq* request is actually a simple request for establishing a connection with the Issuer which is why only minimal $email, alias$ are to be taken here. The *attrReqandproofReq* contain simple attributes of the user to create VC or just give proof of authenticity. For the user to get VC a SSI connection needs to be started as stated before and that starts with *inviteReq* which contains a URL of the Issuer and a fresh nonce. In response to this two responses are sent back which are $inviteResp1, inviteResp2$. The responses contain the DID pairs of the invite request so that they can clearly identify themselves while communicating. Lastly, the request that starts the SSI in a manner *serviceReq* contains the requested service URL and optionally *invitecookie* or *sessioncookie*.

The *didResp* in itself is actually a request although it is a response to a request

and it contains the same as *didReq* and also works the same. In Table 5.2 we also included some string value responses like *attrResp* which has the name-value pair of user attributes. The response of *credReq* is *credResp* which contains the VC generated by I for U. The VC structure and sign format have also been added to the table. Finally, *proofResp* actually contains the requested VC to prove a user while requesting a service.

Table 5.2: Data Model

$req \triangleq \langle didReq, reqRegister, credReq, inviteReq, attrReq, proofReq, serviceReq \rangle$
$resp \triangleq \langle didResp, inviteResp[1-2], attrResp, credResp, proofResp \rangle$
$didReq \triangleq \langle \texttt{secret\_ref}_x, DID_x \rangle$
$reqRegister \triangleq \langle domain_x, org_x, DID_x \rangle$
$credReq \triangleq \langle email, alias \rangle$
$attrReq \triangleq \langle a_1, a_2, ..., a_n \rangle$
$proofReq \triangleq \langle a_1, a_2, ..., a_n \rangle$
$inviteReq \triangleq \langle url_I, N_i \rangle$
$serviceReq \triangleq \langle url_{si}, [[inviteCookie, sessionCookie]] \rangle$
$didResp \triangleq \langle \texttt{secret\_ref}_x, DID_x \rangle$
$inviteResp1 \triangleq \langle N_i, DID_U^I \rangle$
$inviteResp2 \triangleq \langle N_i, DID_I^U \rangle$
$attrResp \triangleq \langle (att_1, attv_1)(att_2, attv_2)..., (att_n, attv_n) \rangle$
$credResp \triangleq \langle VC_I^U \rangle$
$proofResp \triangleq \langle VC_I^U \rangle$
$VC_I^U \triangleq \langle (att_1, attv_1)(att_2, attv_2)..., (att_n, attv_n)_{K_I^{-1\|U}} \rangle$

5.2.2 Algorithms

Now here is four important algorithms of our system where Algorithm 1 *addOrg* handles Organizations to be a part of the Consortium, Algorithm 2 is *proofStatus* which verifies VC and makes required attributes available, Algorithm 3 sends back attributes and DID_I securely to the SP, and lastly Algorithm 4 verifies the received-attributes, DID_I and gives User service.

Algorithm 1 Snippet for addOrg Function

```
function addOrg(request, response)
    reference ← SP's reference from req body
    sp_public_did ← SP's public DID
    doc ← search, retrieve reference doc
    if doc then
        blockchain_response ← Resolve sp_public_did
        data ← I's public DID and reference
        ack_resp ← sends data to "SP − URL/federation − entry − ack"
        if ack_resp then
            sp_data ← extract SP's name, domain
            doc ← execute chaincode with sp_public_did, sp_data
        end if
    end if
end function
```

The initial step involves the exchange of secret reference codes and documents between Issuers and Service Providers. Following this, the respective administrators of Issuers and Service Providers store each other's reference, domain, and organization name in their local databases. The Service Provider's administrator then presents the Service Provider's *sp_public_did* DID and $reference$ via a registration form on the Issuer's website triggering the execution of *addOrg* (Algorithm 1). It extracts the secret reference and DID from the request body, checks for a profile in the local DB(Database) with the reference, resolves the DID from the $INDY$ blockchain that is, retrieved from the $DIDDoc_{SP}$ and sends a request to the "$/federation-entry-acknowledgement$" endpoint of the Service Provider with the Issuer's DID and secret reference. This endpoint undertakes similar actions as Algorithm 1, that is it checks the reference, retrieves the $DIDDoc_I$ and then executes the Fabric chaincode function which creates a transaction with DID_I, $issuer\_data$. This function stores the data in the ledger by adding the transaction details to the state database. If there was a failure in any of the steps then further steps do not get executed. Finally, it sends a response indicating success or failure. If the response from the SP's endpoint indicates success then, the Issuer also executes the Fabric chaincode function which creates a transaction with DID_{SP}, *sp_data*, storing the data in the ledger by adding the transaction details to the state database.

Algorithm 2 Snippet for proofStatus function

```
function proofStatus(data)
    response ← present proof request to wallet with data and gets VC^I_{mobile−wallet}
    Retrieve: public DIDDoc^I_{mobile−wallet},corresponding public K^I_{mobile−wallet}
    verfied ← checkSignature(VC^I_{mobile−wallet},K^I_{mobile−wallet})
    if verified then
        global_attributes ← extract revealed attributes from VC^I_{mobile−wallet}
        return True
    end if
    return False
end function
```

The *proofStatus* function (Algorithm 2) creates a present-proof request with required claims and sends it to the user's wallet. If the user presents $VC^{I}_{mobile-wallet}$, this function will receive it. The Issuer **DID** of the VC is retreived from the presentation message. $DIDDoc^{I}_{mobile-wallet}$ will be retrieved from $INDY$ blockchain and then the public key $K^{I}_{mobile-wallet}$ will be retrieved from there. Using the public key, the digital signature of $VC^{I}_{mobile-wallet}$ will be checked. If nothing was tampered with, then the revealed attributes get stored in *global_attributes* variable, and Algorithm 2 returns true to *requestProof* (Algorithm 3).

Algorithm 3 Snippet for requestProof function

```
1:  function requestProof(req, res)
2:      required_sp_attributes ← from SP's request body
3:      data ← createPresentProofRequestTemplate(required_sp_attributes)
4:      proofstatus ← proofStatus(data)
5:      if proofstatus then
6:          revealed_attributes ← revealed attributes from global_attributes
7:          revealed_attributes.did ← Issuer's public DID
8:          queryString ← createQueryStringWithHmac(global revealed attributes)
9:          Redirect to "SP − URL/callback" with querystring
10:     end if
11:     if not proofstatus then
12:         Redirect to error page
13:     end if
14: end function
```

The *requestProof* function (Algorithm 3) first extracts the Service Provider's required attributes from the request body and stores it in the *required_sp _attributes* variable. Then it creates an Aries Present Proof Request template with *required_sp_attributes* and calls Algorithm 2. If Algorithm *proofStatus* function returns **True**, then it creates a payload object *revealed_attributes* with the stored revealed attributes from *global_attributes* variable and Issuer's public DID. A hash of the payload is created with Issuer's secret reference HMAC-SHA256($reference_I$, *revealed_attributes*). Then the user is redirected to "$SP-URL/callback$" endpoint and the hash, and the original payload is passed in a query-string.

Algorithm 4 Snippet for callback function

```
1: function CALLBACK(req, res)
2:     data ← extractData(queryString)
3:     received_hmac ←  Extract hmac from data
4:     Delete hmac from data
5:     issuer_did ← extractDID(data)
6:     match ← verifyHmac(data, receivedHmac))
7:     if match && matchDID(issuer_did) then
8:         session_cookie ← data
9:         redirect to service
10:    else
11:        redirect back to signup_with_issuer page
12:    end if
13: end function
```

The *callback* function (Algorithm 4) gets executed in Service Provider's back-end. It first extracts the original payload and stores it in a *data* variable. Next, it extracts the hash from the query-string and stores it in *received_hmac* and then deletes the hash from the *data* storage. It also extracts Issuer's DID and stores it in *issuer_did*. Then it regenerates a hash with Issuer's secret reference HMAC-SHA256($reference_I$, *data*) and compares this generated hash with the hash in *received_hmac*. If the hashes match, then it proceeds to check if the DID in *issuer_did* is present in the *Fabric* state database. If it is, then a session for the user is created, and the user is redirected to the service access page. Otherwise, the user is redirected to the sign-up page.

5.2.3 SP & I external connection form Identity Federation

Now we will discuss the use case of Identity Federation among the Issuer and Verifier organization before the SSI protocol starts and this is based on the Algorithm 1. Table 5.3 shows the protocol flow of this use case.

Table 5.3: External Identity Federation Protocol

M0	$SP \leftrightarrow I$	:	SP and I share and store each-others `secret_ref`, Domain, Organization name
M1	$SP \rightarrow I$	:	$[N_0, \text{didReq}(\texttt{secret\_ref}_{SP}, DID_{SP})]_{HTTPS}$
M2	$I \rightarrow B$	:	$[N_1, \text{didResolve}(DID_{SP})]_{HTTPS}$
M3	$I \rightarrow SP$	:	$[N_1, \text{didResp}(\texttt{secret\_ref}_{I}, DID_I)]_{HTTPS}$
M4	$SP \rightarrow B$	:	$[N_0, \text{didResolve}(DID_I)]_{HTTPS}$
M5	$SP \rightarrow Cc$	:	$[N_0, \text{reqRegister}(domain_{SP}, org_{SP}, DID_I)]_{HTTPS}$
M6	$SP \rightarrow I$	:	$[N_1, registeredACK]_{HTTPS}$
M7	$I \rightarrow Cc$	:	$[N_1, \text{reqRegister}(domain_I, org_I, DID_{SP})]_{HTTPS}$

From the presentation of Table 5.3 and the sequence diagram in Figure 5.3 interactions start:

I. The flow starts with SP and I first communication with each other and agreeing on terms to connect with each other with the help of some secret codes and identifiers so that even online they can verify and identify themselves. After this is completed the SP will first go to the I website and will first fill necessary values shown in Figure 5.2 which actually correspond to the request shown in M1. in Table 5.3.

Figure 5.2: Add reference to Chaincode and System storage.

II. The I will match the secret value provided to its own registry and acquire the SP's DID for which it can identify this SP at a later time in response to this the I will send back a similar request in the form of a response with the same format of values in step M2,3.

III. The resolving procedure is the same as M2 for the M4 step and the SP then stores the DID of I for later verification usage.

IV. Finally, both the SP and I consequently store the resolved DID with the help of the same Chaincode but in their private storage systems. In this case, SP actually responds with an Acknowledgment to I that it has added the I as a trusted organization, and then finally I does the same too which completes steps M5, M6, and M7.

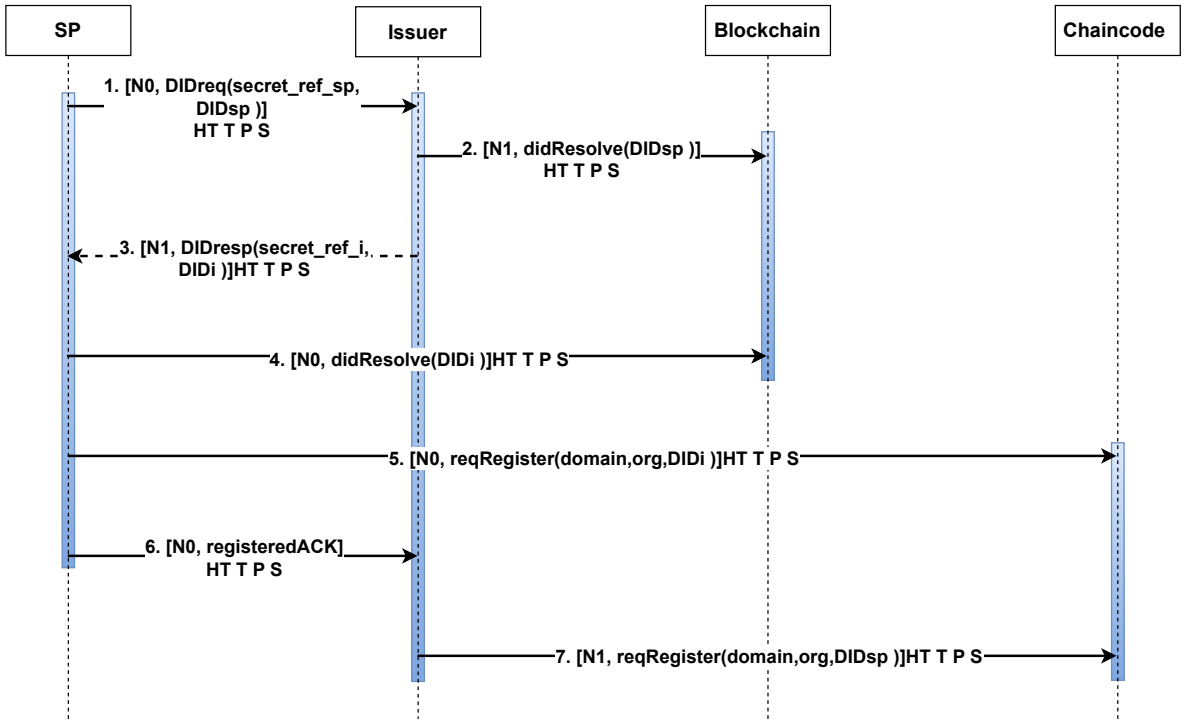

Figure 5.3: SP and I identity federation Sequence Diagram.

5.2.4 Modified-SSI Interactions among all the Entities

U to I Interactions for VC to start SSI:

This use case shows how before U even requests for a service it first goes to the I to get VC for later usage and how the first steps of our modified-SSI implementation occur. Table 5.4 demonstrates the use case's protocol flow.
From the presentation of Table 5.4 and the sequence diagram in Figure 5.9 interactions start from I domain:

I. User first on the I domain and selects the Generate Invitation service which requires the U to fill in minimal credentials like showed in M3 and this is how the first three steps work.

II. Now the I interface shows a QR code to the U like Figure 5.4 which U has to scan with their Mobile Wallet App and then an offer like Figure 5.6 will be shown in the app which the U will accept this invitation which is step M4.

Table 5.4: Web-SSI Protocol for VC

M1	$U \rightarrow I$	:	$[N_1, \text{Generate Invitation}]_{HTTPS}$
M2	$I \rightarrow U$	:	$[N_2, credReq]_{HTTPS}$
M3	$U \rightarrow I$	:	$[N_2, \text{Fill email, alias}]_{HTTPS}$
M4	$I \rightarrow U$	:	$[N_3, inviteReq]_{HTTPS}$
M5	$U_W \rightarrow M$	:	$[I, N_3, inviteResp1]$
M6	$M \rightarrow I$	:	$[N_3, inviteResp1]$
M7	$I \rightarrow M$	:	$[U, N_3, inviteResp2]$
M8	$M \rightarrow U_W$	:	$[N_3, inviteResp2]$
M9	$I \rightarrow U$	:	$[N_4, \text{Credenital Fillup Form}]_{HTTPS}$
M10	$U \rightarrow I$	:	$[N_4, \text{Submit Credential Request}]_{HTTPS}$
M11	$I \rightarrow U_W$	:	$[N_5, attrReq]_{HTTPS}$
M12	$U_W \rightarrow I$	:	$[N_5, attrResp]_{HTTPS}$
M13	$I \rightarrow M$	:	$[U, \{N_5, credResp\}_{K_U^I}]$
M14	$M \rightarrow U$	:	$[\{N_5, credResp\}_{K_U^I}]$

Figure 5.4: Invite QR Presentation

III. The wallet now generates key pair K_U^I & $K_U^{-1|I}$ and also the DID of the U with the help of key pairs. The DID Doc is pushed to the blockchain to get it registered for further use and trustworthiness. The wallet and this point generates $inviteResp1$ & $inviteResp2$ which contains specific DIDs so that a connection can be established with the wallet and all these steps could be done with the help of M. This is how steps M5, M6, M7, and M8 took place.

IV. At this point the I shows a Form like Figure 5.5 where U fills up valid cre-

dentials of themselves according to the SP's format so that later the U can access it. After filling up the U clicks the Submit button thus finishing the steps M9, M10, M11, and M12. Now like Figure 5.7 the mobile wallet would show this notification which the U needs to confirm.

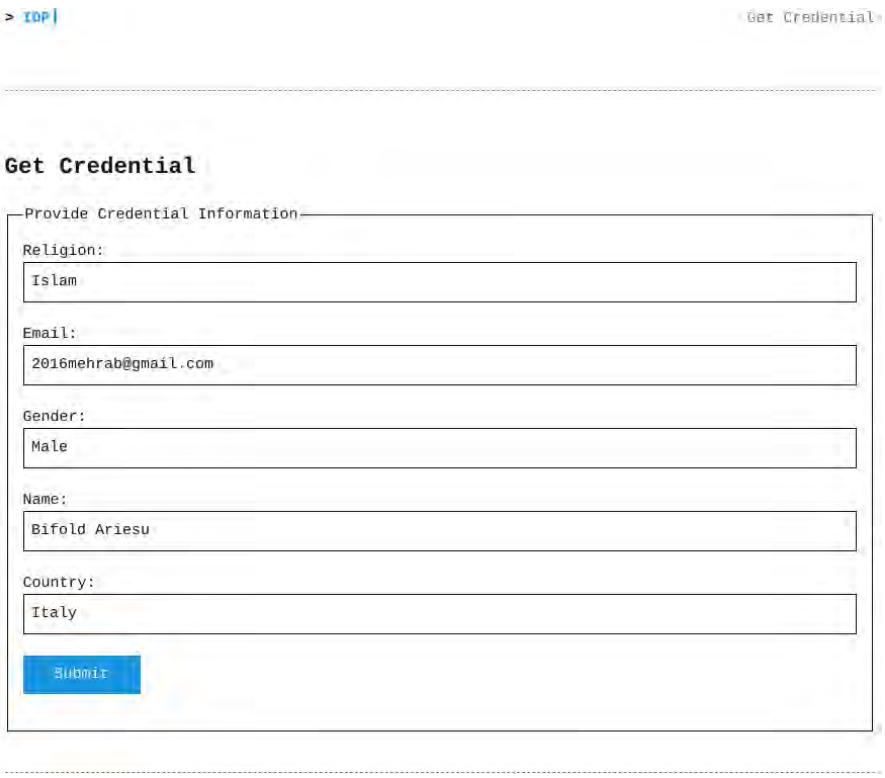

Figure 5.5: Credential Fillup Form

V. The I now with the credentials prepares the VC and a *credResp* with the VC which then is sent to the U via M in steps M13 and M14. The wallet also checks the signature of the VC using the Public key of the I for this connection which was previously generated and stored in the wallet app. If the whole process is successful then the VC will be stored and something like this will be shown. (Figure 5.8)

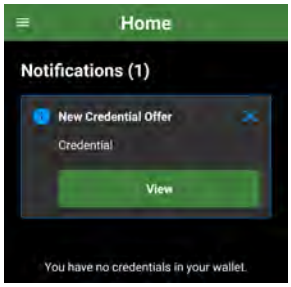

Figure 5.6: Invitation Proposal

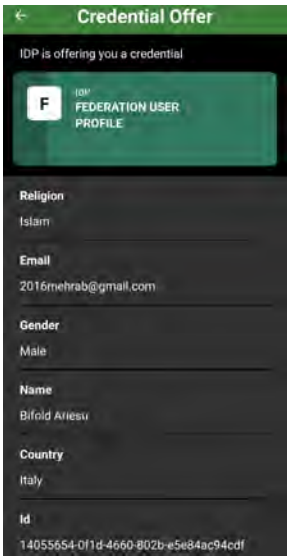

Figure 5.7: Credential Offered

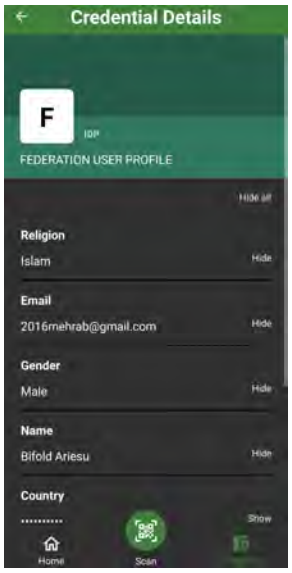

Figure 5.8: Credential Details

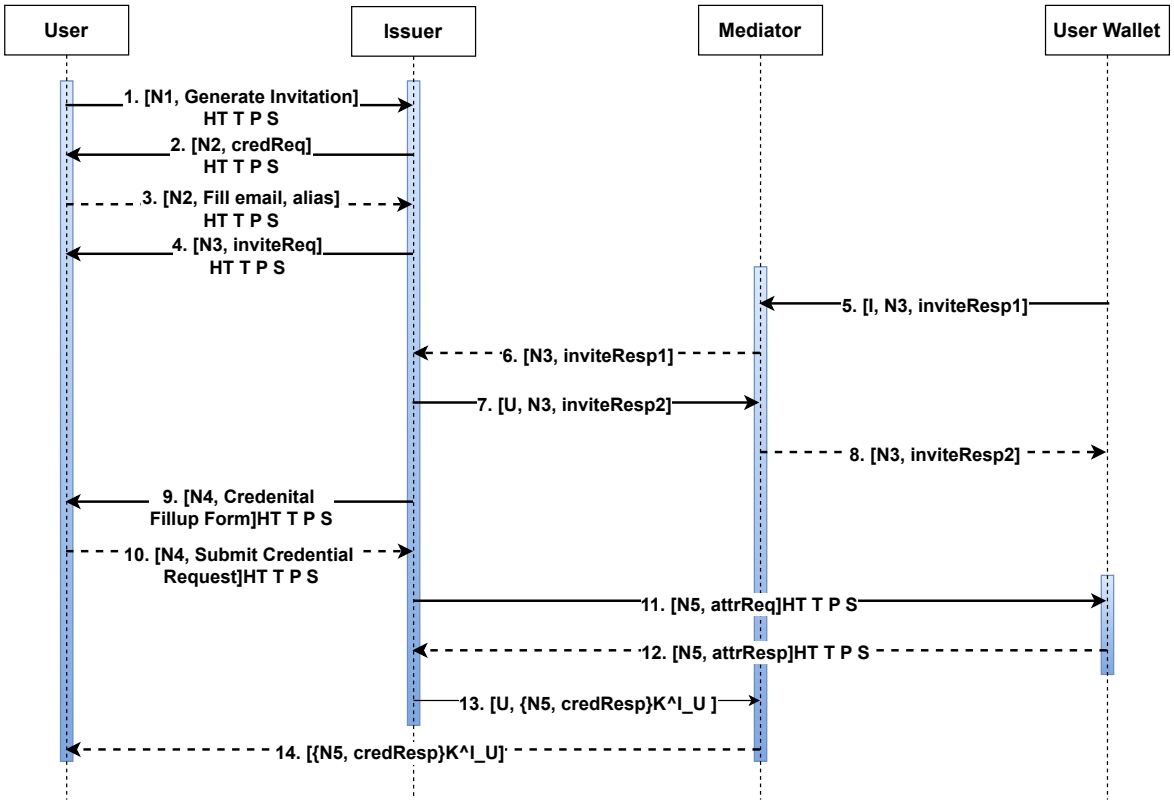

Figure 5.9: User and Issuer interactions to get VC Sequence Diagram.

U to SP Modified SSI interactions to access Service:

This use case shows how U interacts with SP to get Service access where the first steps of SSI or VC generation aren't mentioned as it is already done in Table 5.4 which is a modification. A few other interactions are also modified which will be covered soon to make our system complete which are a part of Algorithm 2, Algo-

rithm 3, and Algorithm 4. Table 5.5 shows the protocol flow for this use case.

Table 5.5: Modified Web-SSI Protocol

M1	$U \rightarrow SP$	:	$[N_1, \text{serviceReq}(url_{S1})]_{HTTPS}$
M2	$SP \rightarrow U$	:	$[N_2, \text{Issuer Options (SSI)}]_{HTTPS}$
M3	$U \rightarrow SP$	:	$[N_2, \text{Select Issuer } (I_X)]_{HTTPS}$
M4	$I \rightarrow U$	:	$[N_3, \text{Connection Page}]_{HTTPS}$
M5	$U \rightarrow I$	:	$[N_3, \text{Fill Email Identifier}]_{HTTPS}$
M6	$I \rightarrow U$	:	$[N_4, \text{Show Attribute Page}]_{HTTPS}$
M7	$U \rightarrow I$	:	$[N_4, \text{Approve Requested Attributes}]_{HTTPS}$
M8	$I \rightarrow M$	:	$[U, N_5, proofReq_{K_U^I}]$
M9	$M \rightarrow U_W$	:	$[N_5, proofReq_{K_U^I}]$
M10	$U_W \rightarrow M$	:	$[I, N_5, proofResp_{K_I^U}]$
M11	$M \rightarrow I$	:	$[N_5, proofResp_{K_I^U}]$
M12	$I \rightarrow U$	:	$[N_6, attrResp]_{HTTPS}$
M13	$I \rightarrow SP$	:	$[N_6, \text{U}, attrResp, DID_I]_{HTTPS}$
M14	$SP \rightarrow Cc$	:	$[N_7, \text{matchDID}(DID_I)]_{HTTPS}$
M15	$SP \rightarrow U$	:	$[N_1, url_{S1}, sessionCookie]_{HTTPS}$

From the presentation of Table 5.5 and the sequence diagram in Figure 5.15 interactions start from SP domain:

I. The U requests a service to an SP at the service url_{S_i}. SP then shows a dropdown list of trusted I like Figure 5.10 who have a connection with the SP. U will select the I that he/she already has VC from or will get a VC from. (M1, M2, M3)

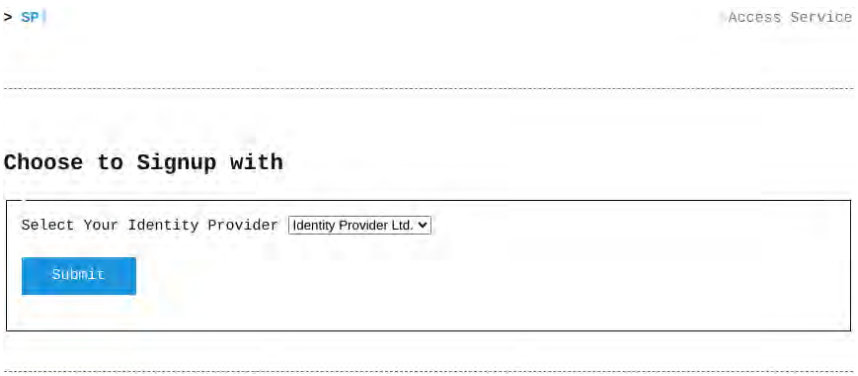

Figure 5.10: Issuer Selection

II. If the U has the VC following the previous protocol of Table 5.4 then an interface like Figure 5.11 will appear where U will just enter a minimal identifier which in this case is the unique Email and login with it. (M4, M5)

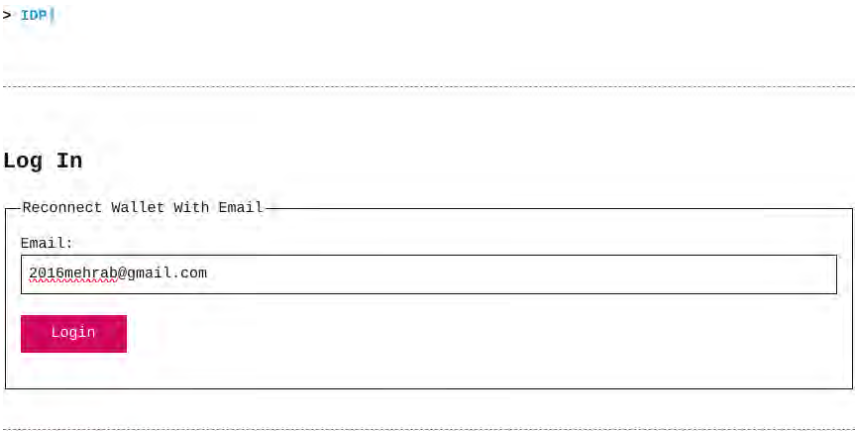

Figure 5.11: Reconnect with I.

III. Now the I will show an interface like Figure 5.12 where a list of attribute fields will be shown which the U will approve. It is completely up to the U if he/she wants to consent to these attributes for accessing the SP or just terminate the whole process. We are assuming the U approves the requested attributes which completes steps M6, and M7.

Figure 5.12: Approval of attributes.

IV. I now generates a $proofReq$ which is sent to the U wallet app through the M. On the mobile wallet a notification like Figure 5.13 will pop up and U will view the request. Since the parsing part was already done while generating the VC the wallet will show the VC to the U for approval like Figure 5.14. After approval, the wallet will send back a $proofResp$ using this VC to the I completing steps M8, M9, M10, and M11.

V. As in our modification the SP doesn't match the VC so only an encrypted version of it with DID_I and reference is sent to the SP following M12 and M13.

VI. SP will match only the DID_I with its chaincode if the I is trusted or not and if it is true the U is taken to the requested service url with a *sessioncookie*. This is how our whole system is implemented by finally completing steps M14 and M15.

Figure 5.13: Proof Request.

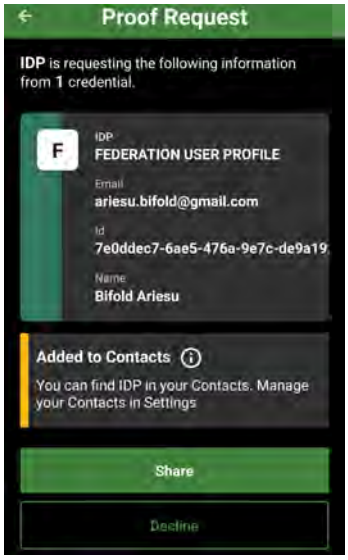

Figure 5.14: Share attributes for proof.

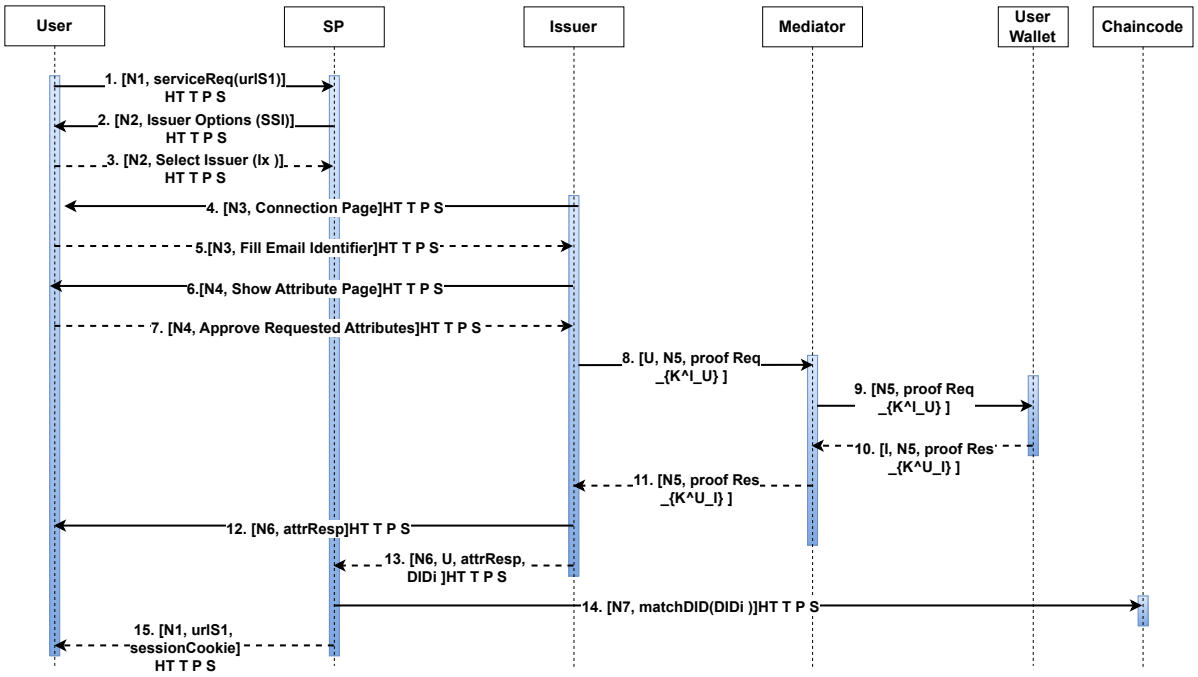

Figure 5.15: User and SP service access interactions Sequence Diagram.

Chapter 6

Discussion

6.1 Requirement Analysis

6.1.1 Functional Requirements:

Our system fulfills all of our mentioned functional requirements. The external trust layer we present mitigates the issue of the Issuer and Verifier trusting each other so, F-1 is fulfilled. The user interacts with the Issuer before even the SSI connection starts and provides credentials with consent and those credentials are stored not on the Issuer platform nor on the Verifier platform as a result we can say the user has complete control over their credentials. Moreover, as the Issuer and Verifier don't even have user credentials the privacy issue is also mitigated so through the use of Wallet App which contains the VC the F-2 and F3 requirements have been met. In our system, there is no separate entity only an inner layer of Chaincode is used, and entities like IdP, SP, Issuer, and Verifier stay as they are with slight modifications for the betterment of usage thus not changing the structures of both FIM and SSI. As a result, even with the modifications F-4 is fulfilled. Using blockchain technologies enabled the system to remain immutable where needed and keep track of all activities moreover, as there was the use of private blockchains there was hardly any delay so, it completed F-5. The identity management system on the base of our system is SSI which follows the laws of decentralized identities moreover we used no data silos letting us further enable that and fulfilling F-6 as well as all functional requirements.

6.1.2 Security Requirements:

In our system there is clear use of SSI and an external trust layer enables it to become more trustworthy which is one of our main concerns. Moreover, only trusted entities with valid DID are allowed in the consortium as a result there is no issue of tampering either thus fulfilling both S-1 and S-2 of the security requirements. The issue of the Issuer faking creating the VC for a user is a flaw in our protocol as the SP doesn't verify the VC. As a result, the requirements of S-2 are partially met. This system is configured in this manner the SP or Verifier doesn't read the VC rather it believes the Issuer to fulfill their prerequisite and provide a valid VC if that is fulfilled. To add to that, the Issuer doesn't store the VC so they are not necessarily keeping any attributes thus fulfilling S-3. As DoS attacks are prone to every web service our system is indifferent in this regard so, additional steps are needed to solve it. So,

this S-4 is out of our scope to fulfill. As the user both needs the Wallet App and Issuer web services to be verified so, verification of an unauthorized user is highly unlikely as records are there to keep track. On the other hand, Issuers and Verifiers are trusted as they first form a contract before any of the services are accessed so, the S-5 requirement is fulfilled. Lastly, for fulfilling S-6 we can only give the user control over getting the VC and consenting to use it other processes are taken care of by the Issuer and Verifier.

6.1.3 Privacy Requirements:

The user or identity holder in our system uses their own Wallet App while acquiring VC and even while sending it to the Issuer. Here, only the user can control their VC and mainly the deletion and creation of VC so, it fulfills the privacy requirements P-1, P-2.

6.2 Research Objective Analysis

The implemented system of ours completely fulfils the research objectives in the following ways:

- **RO1, RO2**: We have reviewed already working Identity Federation, SSI models, and even different interpretations of it. This allowed us to find out the ideal model for our prospect which is the combination of both Identity Federation and SSI. Moreover, we also found the lackings of other models and systems which helped us to mitigate them in our own model. We showed our findings through a comparison table between these papers to get a better understanding of this thus finalizing that we have achieved these two research objectives.

- **RO3, RO4**: We explicitly wanted to provide a user-centric federated identity federation system. The usual FIMs are still being used but, they hardly look at the user's control aspect of it. We provided this in our system by keeping the base with a user-centric identity management system which is obviously SSI. The SSI authentication process is also very robust using VCs and we have kept that in our system with the user at its core. The user gives the values for their VC but they are not stored anywhere since we didn't want any data siloes this also completing our these two research objectives.

- **RO5**: The system we built has an external trust layer which was a key motivational aspect of our research. This trust layer was built with the help of Fabric registry services and handled through Aries which are both well-known Hyperledger Foundation projects. The layer gives our system structure the touch of Identity Federation which SSI leaked so fulfilling a really important research objective.

- **RO6**: This research objective has been achieved through the detailed architecture, protocol, and use case implementations we have presented in this research paper.

6.3 Advantages & Limitations

The new system of Identity Management we proposed has several advantages which are discussed here:

1. The external layer we provided gives a sense of trustworthiness towards our system. Trust in this case does not only cover the Issuer and Verifier but also the User too. Because users do not have to think about their credentials being misused, as the Issuer and Verifier don't have it due to the trust model.

2. If multiple Verifiers or SPs require different credentials attributes, the user doesn't need to create separate VCs but rather use the same one. For this to function more effectively the Verifiers and Issuers should agree on the same VC format to make the whole system easier and more accessible.

3. Even if there is an attack on the system from any adversaries the chance of identity theft is close to none. To start, the system is based upon SSI making it decentralised, and to add to that the VC which has user identity is there with the user and not any other entity. Moreover, cryptographic techniques have been used for data transmission which is safe in itself.

4. We have used ACA-PY which can communicate with public blockchains and also Fabric, and Indy which are private blockchains. So, our system is capable of working with both Public and Private blockchains.

5. There is no single point of failure or central authority in our system, moreover, it could be said that the whole system is rather user-centric. The user can initiate the verification procedure and even the management of VCs thus entrusting more to the users than other systems.

Even with all these advantages, there are some limitations to our PoC which are discussed below:

1. Our system is somewhat complex as SSI itself is not integrated into most services that include identification. To add to that, our external trust layer piles up on that complexity. For a non-technical user, this may be hard to start.

2. The use of Wallet App in our system is necessary and if that fails there is no backup system. Moreover, users might not want the task of managing their credentials which is a disadvantage for some.

3. Our system relies on external services of the Hyperledger Foundation like Aries, Fabric, Indy, etc. which are still in development and may turn out to be points of failure if dependencies don't match.

6.4 Challenges

These are the challenges that we faced during the implementation:

1. The documentation for using the aries-cloud agent python is very poor and mostly non-existent. Moreover, it is very platform-dependent. Similarly, the instructions to build the Aries Bifold mobile wallet application are not very clear, and it is certainly not production-ready because of the slowness.

2. When building a multiple-organization fabric network, there are a lot of limitations regarding adding more organizations and the Certificate Authority (CA) permissions. Moreover, if we run the network on virtual machines, obtaining permission for external ports and sending requests from one node to another, including TLS certificates, proves to be a challenge.

6.5 Future Work

These are the future opportunities of this research which we would like to work on:

1. There is a constant need to use a mobile wallet app for our system and we need to work on a system that doesn't require this and works like a backup system of this wallet mechanism.

2. The use of zero-knowledge proofs can be implemented in our system to make it more secure and easily accessible.

3. Integrate this system in various domains like Univerisity, E-Governance, etc. to see its actual usability and the scalability factor so that if the number of users gets drastically increased, it will act so that we can make it better.

4. We could try to adopt our trust layer or system attributes over already widely used Identity Federation systems such as SAML, OAuth, and OpenID to make the transition to using SSI more convincing.

Chapter 7

Conclusion

Online services still mostly use centralized or, federated identity management systems where data silos are in use. In both of these systems, the third parties have user attributes and can exploit them if they want; moreover, these systems are prone to attack due to the data silos. Even in recent days a lot of people suffered due to major data breaches on these systems. Now if big organizations actually pay heed and the user base cares about their personal information from being jeopardized identity management systems should switch to a decentralized setup. Moreover, even where entities provide users digital credentials like the IdP in FIM and the Issuer in SSI there should be a pre-existing trust among the service provider and these VC issuing entities as organization-level fraud can also happen. In all of these cases the user suffers the most so, the web identity management systems should be user-centric rather than only thinking about the benefits of the organizations.

The system we propose mitigates these issues since the digital credential or VC issuing entity has a trust relationship beforehand with the SP and even the issuer itself does not contain these VCs. So, the underlying issues most identity management systems have are not in our system. Only the integration of systems like ours should bring significant downfall to data breaches, information theft, and even mass-level hacking. In addition, it will also introduce users to the use of digital identity wallet apps in their day-to-day activities. We have started to use e-wallets for buying things so, it shouldn't be too hard to presume that using web services is also similar to that and the use of digital identity wallets is necessary.

Like the real world, the online world is also a necessary part of our lives where trust is necessary. The main hurdle in this trust for online life is the ease of impersonating another person and the amount of online profile that goes on. If practices of SSI and Blockchain technologies could be widespread almost all of these issues could be mitigated and our model goes one step further in that regard which also thinks about the user's security and privacy. The model we present can make accessing web services much better and user-friendly with the notion of trust, security, and privacy. With our contributions, we believe this could bring in a new domain of research.

Bibliography

[1] K. Cameron, "The laws of identity," 2005.

[2] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope, "Trust requirements in identity management," ser. ACSW Frontiers '05, Newcastle, New South Wales, Australia: Australian Computer Society, Inc., 2005, pp. 99–108, ISBN: 1920682260.

[3] A. Jøsang and S. Pope, "User centric identity management," Jan. 2005.

[4] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, "Formal analysis of saml 2.0 web browser single sign-on," Oct. 2008, pp. 1–10. DOI: 10.1145/1456396.1456397.

[5] P. Arias Cabarcos, F. Almenárez Mendoza, A. Marín-López, and D. Díaz-Sánchez, "Enabling saml for dynamic identity federation management," in *Wireless and Mobile Networking*, J. Wozniak, J. Konorski, R. Katulski, and A. R. Pach, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 173–184, ISBN: 978-3-642-03841-9.

[6] J. Lewis, "Web single sign-on authentication using saml," *IJCSI International Journal of Computer Science Issues*, vol. 2, Sep. 2009.

[7] A. Armando, R. Carbone, L. Compagna, and G. Pellegrino, "Automatic security analysis of saml-based single sign-on protocols," 2011.

[8] J. Jiang, H. Duan, T. Lin, F. Qin, and H. Zhang, "A federated identity management system with centralized trust and unified single sign-on," in *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, 2011, pp. 785–789. DOI: 10.1109/ChinaCom.2011.6158260.

[9] M. S. Ferdous, M. Chowdhury, M. Moniruzzaman, and F. Chowdhury, "Identity federations: A new perspective for bangladesh," May 2012. DOI: 10.1109/ICIEV.2012.6317397.

[10] M. S. Ferdous and R. Poet, "A comparative analysis of identity management systems," in *2012 International Conference on High Performance Computing Simulation (HPCS)*, 2012, pp. 454–461. DOI: 10.1109/HPCSim.2012.6266958.

[11] A. Shostack, *Threat Modeling: Designing for Security*, 1st. Wiley Publishing, 2014, ISBN: 1118809998.

[12] M. S. Ferdous, "User-controlled identity management systems using mobile devices," Ph.D. dissertation, Jun. 2015. DOI: 10.13140/RG.2.1.3905.3287.

[13] P. Coelho, A. Zúquete, and H. Gomes, "Federation of attribute providers for user self-sovereign identity," *Journal of Information Systems Engineering Management*, vol. 3, Nov. 2018. DOI: 10.20897/jisem/3943.

[14] J. Bernal Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. Torres Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: Review and challenges," *IEEE Access*, vol. 7, pp. 164 908–164 940, 2019. DOI: 10.1109/ACCESS.2019.2950872.

[15] D. van Bokkem, R. Hageman, G. Koning, L. Nguyen, and N. Zarin, *Self-sovereign identity solutions: The necessity of blockchain technology*, 2019. arXiv: 1904.12816 [cs.CR].

[16] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In search of self-sovereign identity leveraging blockchain technology," *IEEE Access*, vol. 7, pp. 103 059–103 079, 2019. DOI: 10.1109/ACCESS.2019.2931173.

[17] A. Grüner, A. Mühle, and C. Meinel, "An integration architecture to enable service providers for self-sovereign identity," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019, pp. 1–5. DOI: 10.1109/NCA.2019.8935015.

[18] R. Belchior, B. Putz, G. Pernul, M. Correia, A. Vasconcelos, and S. Guerreiro, "Ssibac: Self-sovereign identity based access control," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 1935–1943. DOI: 10.1109/TrustCom50675.2020.00264.

[19] S. Hong and H. Kim, "Vaultpoint: A blockchain-based ssi model that complies with oauth 2.0," *Electronics*, vol. 9, p. 1231, Jul. 2020. DOI: 10.3390/electronics9081231.

[20] S. Figueroa-Lorenzo, J. Añorga Benito, and S. Arrizabalaga, "Modbus access control system based on ssi over hyperledger fabric blockchain," *Sensors*, vol. 21, no. 16, 2021, ISSN: 1424-8220. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5438.

[21] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. Manning, 2021, ISBN: 9781617296598. [Online]. Available: https://books.google.com.bd/books?id=Nh4uEAAAQBAJ.

[22] H. Yildiz, C. Ritter, L. T. Nguyen, B. Frech, M. M. Martinez, and A. Küpper, "Connecting self-sovereign identity with federated and user-centric identities via saml integration," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021, pp. 1–7. DOI: 10.1109/ISCC53001.2021.9631453.

[23] M. S. Ferdous, A. Ionita, and W. Prinz, "Ssi4web: A self-sovereign identity (ssi) framework for the web," in *Blockchain and Applications, 4th International Congress*, J. Prieto, F. L. Benítez Martínez, S. Ferretti, D. Arroyo Guardeño, and P. Tomás Nevado-Batalla, Eds., Cham: Springer International Publishing, 2023, pp. 366–379, ISBN: 978-3-031-21229-1.

[24] M. K. B. Shuhan, S. M. Hasnayeen, T. K. Das, M. N. Sakib, and M. S. Ferdous, *Decentralised identity federations using blockchain*, 2023. arXiv: 2305.00315 [cs.CR].

[25] V. Dzhuvinov, *OpenID Connect Federation 1.0 - draft 29 — openid.net*, https://openid.net/specs/openid-connect-federation-1_0.html, [Accessed 23-May-2023].

[26] D. Hardt, *RFC 6749: The OAuth 2.0 Authorization Framework — datatracker.ietf.org*, https://datatracker.ietf.org/doc/html/rfc6749#section-1, [Accessed 23-May-2023].

[27] Hyperledger, *Aries mobile agent react native*, https://github.com/hyperledger/aries-mobile-agent-react-native, Accessed: 2024-05-15.

[28] Hyperledger, *Hyperledger aries*, https://www.hyperledger.org/projects/aries, Accessed: 2024-05-15.

[29] Hyperledger, *Hyperledger aries cloud agent - python*, https://github.com/hyperledger/aries-cloudagent-python, Accessed: 2024-05-15.

[30] Hyperledger, *Hyperledger Fabric*, https://www.hyperledger.org/projects/fabric, Accessed: 2024-05-15.

[31] Hyperledger, *Hyperledger indy*, https://www.hyperledger.org/projects/indy, Accessed: 2024-05-15.

[32] IMI, *Federated Identity Management Challenges - Identity Management Institute® — identitymanagementinstitute.org*, https://identitymanagementinstitute.org/federated-identity-management-challenges/, [Accessed 23-May-2023].

[33] Node.js, *Node.js*, https://nodejs.org/en/, Accessed: 2024-05-15.