End-to-End encrypted Peer to Peer Chat System with SSI

by

Razin Rayan Rahat
20101001
Shahriar Ahmed
22241145
Abrar Awsaf Talukder
23141077
Ilmy Islam
23141082
Mahpara Chowdhury
20101607

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

Department of Computer Science and Engineering
Brac University
January 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:

Rahat	Shahriar
Razin Rayan Rahat 20101001	Shahriar Ahmed 22241145
Abrar Awsaf	Ilmy Islam
Abrar Awsaf 23141077	Ilmy Islam 23141082

Mahpara Chowdhury

Mahpara Chowdhury
20101607

Approval

The thesis titled "End-to-End encrypted Peer to Peer Chat system with SSI" submitted by

1. Razin Rayan Rahat (20101001)
2. Shahriar Ahmed (22241145)
3. Abrar Awsaf Talukder (23141077)
4. Ilmy Islam (23141082)
5. Mahpara Chowdhury (20101607)

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

Examining Committee:

Supervisor:
(Member)

Sadek

Dr. Md Sadek Ferdous
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

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

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The paper is an honest work. The paper is a statement of our comprehensive investigation and implementation. As the members hereby declare that, all of the resources used to conduct the research are recorded and cited properly. Moreover, the paper has never been published or submitted for granting of a degree or any other reason to another university or institution.

Abstract

Chat applications are among the most popular Internet applications and a huge number of people use them on a regular basis. As their use has grown, different security and privacy concerns have received attention from the users and the professionals. Many users consider what they chat with their family and friends to be extremely private and they want a certain amount of assurance that their chats are securely exchanged and are not exposed to any unauthorized parties. Towards this aim, many chat applications employ an End-to-End (E2E) Encryption mechanism. This is to safeguard the encryption keys during key exchange as these keys are crucial to ensure the security of the chat histories. Unfortunately, the existing key exchange mechanisms for E2E encryption are prone to Man-in-the-Middle (MITM) attacks. In addition, such mechanisms sometimes use a central server for exchanging keys which raises privacy and security concerns as these central servers may not be trustworthy. In this research, we would like to address these issues, by introducing a novel SSI (Self-sovereign Identity) based End-to-End Chat System which supports a Peer-to-Peer (P2P) key exchange mechanism.

Keywords: Self-sovereign identity (SSI); Peer-to-peer; End-to-end Encryption; Verifiable credentials; Decentralized identity; DIDcomm; Trust Layer; Key Management.

Dedication

We dedicate this research to all developers, cyber security experts and software engineers who are working everyday to make the internet safe and secure for the users; so that they can participate in the internet and do their task without any fear of cyber attacks.

Acknowledgement

Firstly, all praise to the Almighty Allah for whom our thesis have 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 whenever we needed help. And finally to our parents without their support it may not be possible.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

DID	Decentralized Identity
$E2E$	End-to-End Encryption
$P2P$	Peer-to-peer
PKI	Public Key Infrastructure
SSI	Self-sovereign Identity
VC	Verifiable Credentials

Chapter 1

Introduction

1.1 Introduction

Self-Sovereign Identity (SSI) is a new identity model which focuses on decentralization of internet infrastructure which is facilitated by modern technologies like blockchain or distributed ledgers. Self-sovereign Identity (SSI) has arrived as a symbol of hope for them because traditional mechanisms have failed to gain the trust of its users. SSI is bringing a revolution on how identity should be managed by shifting the center of the power of identity from an institution to its owner or user. But its popularity and necessity emerged because of the failure of traditional centralized systems. The internet has been prone to many types of attacks because it doesn't have an identity layer, most of the services are built on centralized client-server models which have crucial problems like single point of failure and data breach. Also the users of the internet don't feel safe about different companies owning sensitive data about themselves. Identity theft is a very common problem users face when using web services using login credentials. In the last few years, many big service providers faced data breaches resulting in disclosure of sensitive user data. In a recent study, it was found that more than 80 percent of USA's IT companies have found that their systems have been compromised in order to steal, change or make important public information data [3]. That's why it has become necessary to encrypt the user data so that not only unauthorized access can be revoked but also to ensure that the company or government can't go through the sensitive information. WhatsApp is using a very well established protocol called Signal which introduced End to end encryption introduced by Signal. But still these protocols are used by centralized services which can harm the user's sovereignty. And in this era of this digitalized world, where there are constant data and security breaches, users' fear about their sovereignty has risen to its peak. As a result, more and more people are becoming interested in exploring new technologies like decentralized systems that give them more freedom in terms of maintaining and managing their identity while countering various threats that are very common in current systems and architectures. This is why, to eradicate the issue of user sovereignty, we are suggesting a SSI based E2EE peer to peer chat system which is built on one of the most famous tools of Hyperledger called Hyperledger Aries. We will be using DIDs and VCs to establish a secure channel between entities and work on the existing DIDcomm to develop an enhanced chatting system. Not only will we try to mitigate current problems faced by popular social media platforms by just building a SSI based E2EE peer to peer chat system but also we will work on other domains apart from sending peer to peer messages securely. Apart from building the peer to peer chat system, we will try to integrate two more features that are so important for modern chat systems. The first one is file transfer and sharing with other users and the second one is a group messaging chat system with multiple users. The file sharing and transferring system will enable the users to share various types of files in different formats with their fellow users. The group messaging chat system will enable users to make communities and chat with multiple users at the same time while creating discussions and threads they want to do. These two features will enhance the user experience greatly by adding convenience, efficiency, contextual communication and enhanced collaboration while it will give the builders the scalability and a scope to future expansions of the system.

1.2 Motivation

In the initial phase, the internet was built without an identity layer through which we can know who we can connect with or what we are doing or may do with it. Since we are unaware of these things, this little issue transformed into a greater threat, exposing us to a great range of threats, security attacks and breaches. To illustrate, popular social media platforms that adopt centralized servers have become honeypots for attackers to steal personal information of those users. For instance, in 2019 security breaches on Facebook and Instagram exposed confidential information associated with almost 49 million users. Similar breaches of security caused damage to Myspace, Linked in, TikTok, Twitter, and exposed hundreds of millions of their users' login credentials, including profile name, phone number, age, location, real name, activities, and so on [12]. Experts implemented End-to-End (E2E) encryption to solve this problem but the security and privacy concerns still remain as those systems use centralized client-server systems for Key exchange mechanism which is prone to attacks like Man-in-the-middle attack. It is proven that traditional centralized systems are unable to counter such attacks and this is where efforts should be made to look out for possible alternative solutions, and one of them is the concept of decentralized systems and a Nobel digital identity management called Self-sovereign Identity (SSI). Using blockchain technology, SSI, decentralized systems, and infrastructures we can reduce the risks associated with centralized systems. So far, there have been only a few attempts to create decentralized chat applications but their work on file transfer and group chat systems feature is very limited. So we tried to elevate those things by working on file sharing and group messaging on a decentralized system. It is highly possible that both existing traditional and decentralized applications will have positive impact and influence in terms of countering major security threats and working on file sharing, transfer and group messaging systems.

1.3 Problem Statement

The conventional way of communication involves centralization which involves servers and other intermediaries. This client-to-server structure is prone to many vulnerabilities and attacks. The centralization of the chatting system can cause problems like a single point of failure and can cause disclosure of data through multiple types of attack. The conventional username and password login system is prone to fail. Also these chatting systems often sell user data to different companies and also are bound to disclose them to the government which defies user's privacy. Also the encryption models used in the centralized systems are often criticized by consumers. The summary of the problems are given below:

- No privacy for user identity
- Centralized intermediaries
- Threats to user authentication and authorization
- No verification of received message or sender
- No user sovereignty
- Repetitive username-password login system

1.4 Research Objective

- RO1 - To understand modern centralized communication models, self-sovereign identity, blockchain and existing decentralized communication systems.
- RO2 - To understand how SSI solves identity management and authentication.
- RO3 - To understand how DIDcomm transfers messages using existing protocols.
- RO4 - Designing the architecture of the decentralized messaging system using numerous threat modeling and requirements analysis leveraging existing DIDcomm protocol.
- RO5 - Defining the protocol of decentralized messaging system to detail the interactions among various types of components, entities to ensure efficient and error-free functioning.
- RO6 - Evaluating the performance of the decentralized system.

1.5 Report Structure

In Chapter 1, we give an introduction, where the problem statement is discussed along with research objectives which focus on topics like SSI, Blockchain, P2P etc. We try to elaborate why existing technologies have failed to satisfy users' expectations and how using new technologies our system can achieve that as well as our motivation to do it. After that, Chapter 2 states the background of our research that tells us about the Seven Building Blocks of SSI as well as Hyperledger Aries which is a major tool for the development of our system. Then Chapter 3, presents the details about the previous works in this domain focusing on decentralized chat applications in our literature review. Moreover, some comparisons are shown among our system, traditional popular social media platforms and existing decentralized systems. In Chapter 4, we display the proposal where we discussed topics like methodology, threat modeling, and requirement analysis such as functional and security requirements. Following up, in Chapter 5, we have shown the implementation through some visualizations and some detailed protocol flow with use case description to explain the entire system and its process. Next in Chapter 6, we make a discussion about analyzing the requirements, the advantages and limitations of our system and some scopes for future development and work. Finally to conclude, Chapter 7 marks the conclusion of our report.

Chapter 2

Background

2.1 SSI

The history of human communication has been forever changed by the groundbreaking innovation known as the internet. But the developers were blind to a fundamental point. The identity layer was not present when the internet was created. To be emphasized, the TCP/IP protocol was utilized in the early stages of the internet and was believed to be a huge success. However, as the network has expanded to include billions of users, an identification layer is no longer there. Except for the IP address, people are unaware of the identity of their communication partners. This is where SSI emerged as a game-changing technology that aided programmers in creating the internet's missing link. Self-sovereign Identity, or SSI, is an identity that is independent of and not subject to any individual, authority, or government. It offers Decentralization, Interoperability, Portability, and Reusability and boosts confidence and security. SSI is a self-asserted and user-centric identity. These essential components of SSI serve to build the identity layer that the Internet lacks, enhancing the ecosystem's safety, security, and dependability. [13]

2.2 Seven building blocks of SSI

2.2.1 Verifiable Credentials

Credentials are the documents that people keep in their wallets to demonstrate their identification or position of authority. Credentials are tamper-resistant documents that contain information on the subject of the credentials that a certain authority claims to be authentic. These authenticated credentials are not just limited to people or documents. Verifiable credentials [13] are the documents which are cryptographically verified. These digitally signed verifiable credentials are issued by an issuer, who then provides them to a holder who then uses them to validate their identity to a verifier who checks the validity of the information and the issuer's signature. In contrast to physical credentials, VC can instantly validate a claim using the internet and cryptography.

2.2.2 The Trust Triangle

Issuers, Holders, and Verifiers, sometimes known as the Trust Triangle [13], are the three basic responsibilities that are involved in venture capital. First off, since they are the ones that give credentials to a holder or subject, Issuers are the primary source of credentials. Governmental organizations, academic institutions, and financial entities are all examples of issuers. Second, the Holders ask Issuers for VCs, which they then save in their digital wallets in case they need to authenticate their identity. Holders can be either a human, an animal, or a machine (IoT). The verifiers then take VCs from holders and determine whether the Issuers and the set of data or claims are true or not. Holder's representative is in charge of presenting claims to the verifiers for verification. Anyone can be verifier.; human, organization, agencies or universities.

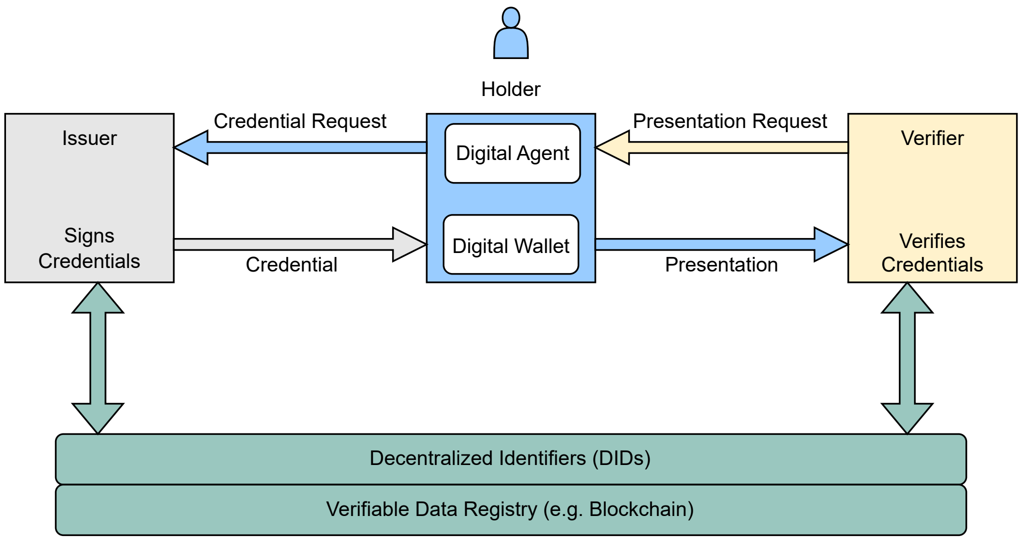


Figure 2.1: Trust Triangle

In the Fig. 2.1 it shows the trust triangle of holder, issuer and verifier. Verifier has to trust the issuer or the system will not work as in real life. DIDs and verifiable data registry(blockchain) will be used to verify the VC. Holder can request VC from the issuer and will hold that onto his wallet and can present it to the verifier if requested.

2.2.3 Digital wallets

Digital wallets [13] store VCs and digital credentials within them, just like how people use real wallets to protect and control their physical credentials and assets. The key motivations behind the development of digital wallets were to make VCs easily accessible, available, and portable across all of a person's devices, as well as to protect the credentials from malicious hackers or eavesdroppers. Digital wallets are used in real life in a variety of ways, including smartphone wallets and cryptocurrency wallets (edge wallet and cloud wallet). Open standards should be used by SSI digital wallets, and they should cooperate with the digital agents. Because they have open standards, digital wallets offer additional services, thus any VC that satisfies the requirements of the VC should be approved. The wallets can be installed on all the devices that a person uses frequently, and a backup can be made by transferring the contents of the wallets to other digital wallets.

2.2.4 Digital agents

In real life, people handle and control physical wallets, but since they do not understand binary language, they cannot manage digital wallets in the same way. The "Digital Agent" [13] module in SSI infrastructures allows for this. The digital agents hold the owner's digital wallets and credentials and present them. Digital agents communicate with one another to establish connections and exchange credentials utilizing decentralized, secure messaging networks in addition to giving verifiable credentials.n There are some existing digitals agents like Edge agents and Cloud agents. Some cloud agents store and synchronize the credentials for their owners.

2.2.5 Decentralized identifiers

Decentralized identifiers (DIDs) [13] are a sort of verified decentralized digital identification. This kind of globally unique identifier is unique. A DIDs can be used to refer to anything, including a person, group, object, data model, etc. DIDs support decentralized registries, identity providers, and certificate authorities, which is the primary distinction between DIDs and federated identifiers. It gives entities the ability to demonstrate their authority over them through the use of cryptographic proofs like digital signatures. Decentralised Identifiers (DIDs) provide certain advantages, including user control, improved privacy, and security. Contrary to other traditional identity suppliers, DIDs allow people to own and control their own identities. DIDs increase user autonomy by giving them more control. Additionally, users or people have more control over their data and credentials as a result of power shifts, which makes them feel safer and more secure. People have the option to revoke their identification, and their data will be protected from additional dangers such data breaches, unauthorized access, and exposure of sensitive information. DID-to-DID relationships are also permanent, secure, regarded as trustworthy, expandable, and end-to-end, while DIDs are permanent, manageable, cryptographically verifiable, and decentralized.

2.2.6 Blockchain

Any type of verifiable data registry or decentralized networks can be used to transport and store DIDs, but there is a reason that the creators chose Blockchain [13] for the SSI infrastructures. Blockchain is essentially an extremely impenetrable distributed ledger that is not under the control of a single person or an authority. A new block that has been added to the Blockchain cannot be altered once it has been added. Blockchain is a good choice for this since it offers a reliable source of data that any peer may trust without depending on a single centralized authority and DIDs are tamper-resistant. Blockchain may trade off a number of database features, such as performance and scalability, but also frees the hands of centralized authorities. Blockchain uses a module called "triple play of cryptography" to shift the power from centralized servers to peers. In Blockchain, every transaction is digitally signed, transactions are grouped into blocks and cryptographically hashed so that it can be linked with the previous block, and every block is replicated across all of the peer nodes. These kinds of factors are the reasons why Blockchain is used in SSI infrastructures.

2.2.7 Trust Framework

The framework [13] for building trust between partners is essentially a collection of rules, values, and procedures. The goal of SSI Infrastructure is to create a system where two parties can communicate on the internet with a level of trust that is mutually acceptable to both parties. However, establishing confidence in cryptography is more difficult than it is in everyday life. For instance, when performing Bitcoin financial transactions, we must be cautious of rules governing money transmission. However, if the issuer has enough confidence in the agencies, then verifiers can check the rules governing money transmission utilizing the VCs. Cryptographic trust can therefore play an anchor function for Issuers of VCs using DID. Another kind of trust triangles between parties are produced by this framework for trust. In this triangle, in the case of trustworthy issuers, verifiers rely on governing authorities. Trust frameworks, which are essentially the antithesis of VCs, establish the guidelines that must be followed when providing a VC to a holder. The verifiers can also determine the basis for believing a VC to be a valid document.

2.3 Hyperledger Aries

A common, reusable, interoperable toolkit is what Hyperledger Aries [13], an open source project run by the Linux Foundation, utilises to build, transmit, and store valid digital credentials. Aries offers shared, reusable components and standards that help parties create a secure communication connection. Key elements like the agent framework, verified credentials, DIDcomm, interoperability, decentralised key management, etc. are provided by Hyperledger. It is well-liked because it enables developers to construct a decentralised identity solution ecosystem in accordance with their specifications and preferences.

Chapter 3

Literature Review

In 2013, Ibrahem et al. [1] offered a web-based messaging system that uses a new Diffie-Hellman protocol iteration to support secure key exchange and mutual authentication between client and server. Zero Knowledge Proof (ZKP) and password authentication were used in the proposed system, together with Hmac for integrity and AES for confidentiality. The limits of current messaging systems are highlighted, along with the necessity for secure messaging services. Potential attacks on their suggested system are discussed in the study, including man-in-the-middle, replay, and brute force attacks, among others. There are no code implementations in the paper. The paper's benefits include increased security in online chat platforms. Due to the usage of encryption methods, their suggested system may, nevertheless, demand more processing resources than current messaging systems.

Unger et al. [4] published a paper that takes authentication, confidentiality, integrity, availability, usability, and adoption as some of the factors to evaluate and systematize the current secure messaging solutions. It also proposes an evaluation framework that evaluates the security, usability, and ease-of-adoption of secure messaging solutions. The approach also considers the trade-offs between usability and adoption and security. The trust establishment, conversation security, and transport privacy are the three main difficulties in secure texting, according to the paper. End-to-end encryption, forward secrecy, deniability, and perfect forward secrecy are common security features found in secure messaging tools.

In 2017, Ali et al. [5] proposed a secure chatting application with end to end encryption for android platform. The writer suggests an end-to-end encrypted secure messaging app for Android devices that enables users to connect via text, voice, and photo communications while maintaining the highest level of security. The paper suggests employing public key cryptography methods and the Elliptic Curve as solutions. Symmetric algorithm RC4 for voice and image security procedures, standard AES method with a 128-bit key, and Diffie Hellman Key Exchange algorithm to produce and exchange keys for symmetric encryption of data. Threat modeling or code implementations are not specifically mentioned in the study. The paper's advantages include a very secure chat program that enables communication between users, but one potential disadvantage is that the symmetric method RC4 is known to have significant weaknesses.

In 2017, Zupan et al. suggested a decentralized messaging system called HyperPubSub [6] that provides a secure messaging for a multi federated, permissioned environment.The paper also proposes about the challenges associated with the decentralized messaging system and how HyperPubSub addresses these challenges. The paper basically proposes HyperPubsub as a reusable blockchain middleware. Implementing decentralized messaging systems can bring challenges due to issues like scalability, security and privacy. HyperPubsub addresses their challenges and gives a safe and private messaging system that is examined using blockchains for validation and monetization purposes. It also provides verifiability of past publish subscribe operations through the use of smart contracts and blockchain records. It is designed with several key features like verification, secure messaging. Unlike other blockchain messaging systems it provides past pub/sub operations using Smart Contracts and blockchain records.The Demonstration has been made implementing Kafka and Hyperledger.

Another model was also proposed in 2018 which also used Ethereum blockchain for decentralized secure messaging in a trustless environment . Abdulaziz et al. [7] suggested Whisper, an Ethereum P2P communication protocol for decentralized applications which utilizes õξVp2p Wire Protocol. The messages are asymmetrically encrypted using the Elliptic Curve Integrated Encryption Scheme (ECIES) together with the SECP-256k1 public key or symmetrically encrypted using the Advanced Encryption Standard Galois/Counter Mode (AES-GCM). Whisper protocol can be interacted with using web3-shh package. The two major functions of the suggested model are account management and messaging. This model also solves the problem of encryption, DoS attack and network traffic analysis.

An approach to develop a secured peer-to-peer communication was written in 2019. Khacef et al. [9] designed a new model with Ethereum to perform encrypted messaging. In this proposed model, each user's identity needs to be validated using a smart contract which will ensure trust between them. Each interaction with the smart contract is recorded as a transaction on the blockchain which is immutable. It uses ECDSA algorithm to generate a public and a private key. The users keep their respective private key safely and register their identity in the proposed model using public key. After the smart contract verifies the registered identities a mutual connection is established. ECDH algorithm is used to generate a shared secret which is used to encrypt messages using a symmetric key algorithm. The advantage of this model is it removes central authorities and uses blockchain and public key for identity verification. Also the smart contract is Turing complete which helps it to perform as it is expected. It also ensures confidentiality, message integrity, authentication and reliability. The limitation of this model is that the smart contract performs sequentially, resulting in performance issues and also the immutability of smart contract after deployment.

In 2019, Schilinger et al. [10] discussed how important it is to have a secure online social network (OSN) that forbids server-accessing hackers from reading user-private messages. The lack of privacy and security in present OSNs, which can result in sensitive information being accessed by unauthorized persons, is one of the issues discussed in this study. A user-friendly approach is required since many users might not be technically sophisticated enough to use encryption tools themselves. A threat model is included in the study that lists prospective attackers (such OSN administrators, hackers, and government organizations) and their skills (like accessing server data or intercepting network traffic). End-to-end encryption using the RSA and AES algorithms is the suggested method. The study contains a JavaScript and Web Cryptography API-based prototype implementation of their suggested solution. Increased privacy and security for OSN users as well as user-friendliness for individuals who might not be familiar with encryption technologies are advantages of this method. Due to the usage of cryptography based on JavaScript, there may be speed issues as well as compatibility concerns with older web browsers.

In 2020, U. P. Ellewala et al. [11] suggested in a paper a blockchain based secure messaging system instead of traditional centralized system. The paper mentions different limitations of traditional system and how this lead to some big loss. The authors proposed a instant messaging application based on blockchain technology. The application contains different models for privacy like authentication model, smart contracts and data loss prevention model and encrypting model. It also uses cryptographic hash mode to verify integrity of messages and smart contracts. The result of the paper is a open specification for a secure and private chat application which uses blockchain. Because of blockchain it gets benefits like data tamper immunity which will be helpful for forensic purposes. Also it gets confidentiality, audibility, shared single source of truth etc. In conclusion the paper is a roadmap for blockchain based chat application.

In 2021, Shi et al. [14] suggested another protocol Bitmessage. It is a blockchain based decentralized encrypted peer to peer communication protocol to provide better anonymity and privacy by using a stealth address as a destination of a delivered message which is encrypting a message with a receiver's public key, which is almost impossible to identify except the intended receiver. This proposal also includes an anti-spasm mechanism with proof of space which prevents accidental leakage of data; this also reduces time and resource costs by eliminating computationally heavy hash operations. Thus, Bitmessage achieves high anonymity, practicality, anti-spasticity and efficiency. Bitmessage uses a Proof-Of-Work mechanism to reduce spam which requires a high hash collision to send a message also wastes energy (a huge energy consumption) and takes a huge time(an average of four minutes to complete).

In 2021, Singh et al. [15] proposed a blockchain-enabled end-to-end encryption for instant messaging service. The writers suggested using Matrix, an open-source decentralized protocol and network for real-time communication. It uses Ethereum blockchain's secret store feature. Its new end-to-end encryption system fully separates the encryption of a message from its transport. Among other reasons, The Secret Store was chosen because it offers an abstraction over the underlying cryptographic fundamentals. Additionally, it enables the development of a sophisticated control system to access the encryption keys, and hence the exchange. Like the previous papers, this also uses Solidity smart contracts. Tests such as varying length of messages, varying threshold and access time to smart contract were performed to evaluate the performance of the proposed system.

Again in 2022, Halder et al. [17] made another architecture fybrrChat, a trustless network architecture that introduces a decentralized messaging system that ensures impenetrable security for critical and sensitive use cases. The P2P architecture of fybrrChat is much more scalable than existing platforms due to its serverless architecture. The Distributed Hash Table (DHT) in fybrrChat is also decentralized and uses the storage of devices using fybrrChat for intermediate message storage. The paper then discusses the security features of fybrrChat, including end-to-end encryption, which ensures that only the intended recipient can read the message. The authors also discuss how fybrrChat uses cryptographic hash functions to ensure message integrity and prevent tampering. The authors suggest implementing a hybrid approach that combines P2P technology with cloud-based storage to improve scalability while maintaining security.

In 2023, Bigos et al. [18] published a security analysis of modern day chatting apps such as Signal, WhatsApp and Telegram . His analysis focuses on the encryption protocols used by each app and the security features they offer. The paper evaluates the strengths and weaknesses of each app and provides a summary of their overall security posture. Additionally, the paper discusses other considerations such as user base, data collection and usage policies, and other features that may impact the security of the apps. The paper highlights that signal is the most secure among those 3 while WhatsApp uses a modified version of the signal protocol and Telegram offers two types of encryption: server-client encryption, and End-to-end encryption. Also they found that all 3 of those apps collect user data and only signal doesn't store any data. In Experimental Results the authors describe how they decrypted all HTTPS traffic using a tool from Fiddler. That created a fake certificate that they allowed on their testing machine and were able to successfully intercept and decrypt Signal's HTTPS traffic. This allowed them to analyze the data being transmitted and identify potential vulnerabilities in the app's security posture. They also analyzed WhatsApp and Telegram encryption protocols using Wireshark to capture and analyze network traffic between two devices running those.

In 2019, M. Chase et al. [8] in their paper tried to improve the privacy and scalability of End-to-End Encrypted messaging by building SEAMless, a Privacy-Preserving Verifiable Key Directory (VKD) system. It provides users with the ability to verify their keys without any service provider interference with this system and key update techniques crucial for dynamic messaging security, empowers users to monitor and verify their key history, data integrity and security by enforcing Verifiable Random Functions (VRFs) and Merkle Trees. This approach has proven to have positive impact as the SEAMless provides efficient server updates and low computational times and it signifies a remarkable achievement in messaging applications.

In 2022 A. H. Enge et al. [16] in their paper tried to propose an architecture for secure decentralized P2P messaging. In their paper they first talk about problems with traditional systems where you always need the internet to do something, then they give the benefit of a messaging system which can operate without the internet. In their motivation they talk about how new technologies like ssi and didcomm with BLE(Bluetooth Low Energy) have opened a new door to enable communication or establish connection without the internetBluetooth Low Energy. Then they gave an architecture and explained how connections should be established and encrypted messages should be sent. Though this is a very fascinating idea there are still few limitations. First of all, a lot of functional and nonfunctional requirements are needed for any kind of connection establishment or message exchange. Then the system was not implemented and the authors themselves have doubts whether it is possible to implement it with current SSI available to the public.

3.1 Comparison

Social Media Platforms vs E2EE P2P Chat System with SSI: So far we have shown the extensive literature review of fourteen papers that emphasizes on the previous works which has been done in the domain and in future we will also show methodology, architecture and implementation of our system, but before that let's compare our system denoted as 'E2EE P2P Chat' with existing chat systems that are growing popular or already popular among the people in Table 3.1. In the table, Black dot represents, 'Yes' meaning that this system has satisfied certain criteria or do have this particular feature within their system. while the White dot represents 'No' meaning they do not fulfill the criteria; and lastly Half filled fot represents 'Partially Yes' meaning they or may not have this attribute in their system.

Based on statistics, Messenger, WhatsApp, Telegram and Signal are four of the most popular chat systems around the world right now.If we compare a few important features of our End to End Encrypted Peer to Peer Chat System with SSI with those existing chat systems, we can see few differences and advantages of our systems. First of all, the biggest advantage this system has is that it is a peer to peer decentralized system. Unlike those existing centralized systems, which can face many issues such as single point of failures, system crash and disruptions. Similar things can be seen in terms of storage, this system uses local storage while other popular systems use servers to store data which is prone to several attacks and breaches. Similar to other popular chat applications, it has chat features where users can build connections with each other and have chat with each other by exchanging messages. Those systems can share and transfer all types of files such as PDF, Text file, Images, Excel Sheet, Powerpoint Slides etc. Our system is also capable of transferring files but till now it only supports Images to share and transfer. In Chapter 2, we have already discussed Verifiable Credentials, what it does and why it is so important. Now, one of the biggest advantages this system possesses is it supports Verifiable Credentials which can be further used in future works while talking about traditional centralized chat applications, they don't support VCs.

Decentralized Chat applications vs E2EE P2P Chat System with SSI: As we have seen a comparison between our proposed system and popular social media platforms right now in the world, let's dive into our domain and then compare our proposed system denoted as 'E2EE P2P Chat' with existing decentralized systems that has already been built. In the Table 3.2 apart from our proposed system, we have chosen 4 different decentralized system, and those are A Decentralized system using Bluetooth LE and DIDcomm denoted as System-1, mentioned in [16], A Secure Messaging Platform based on Blockchain denoted as System-2, mentioned in

Table 3.1: Comparison of popular chat applications with the proposed system

	Traditional Chat Applications vs Proposed System				
Features	Messenger	Telegram	WhatsApp	Signal	E2EE P2P Chat
Centralization	Server	Server	Server	Server	Peer-to-Peer
Storage	Server	Server	Server	Server	Local
Chat System	●	●	●	●	●
File Transfer	●	●	●	●	◒
VC Support	○	○	○	○	●

$E2EEP2PChat$ ≡ Represents 'The proposed system in this paper'

● ≡ Represents 'Yes'

○ ≡ Represents 'No'

◒ ≡ Represents 'Partially Yes (Image Only)'

Table 3.2: Comparison of decentralized chat applications with the proposed system

Systems	Decentralized	Implemented	File Sharing	Group Chat	Architecture	Storage	Protocol
System-1[16]	●	○	○	○	●	$\mathcal{B}$	○
System-2[11]	●	○	○	○	$\mathcal{H}$	$\mathcal{B}$	○
Bitmessage Plus	●	○	○	○	○	$\cup$	○
FybrrChat	●	●	$\cup$	$\cup$	$\mathcal{H}$	$\mathcal{B}$	●
E2EE P2P Chat	●	●	●	◒	$\mathcal{H}$	$\oplus$	●

$E2EEP2PChat$ ≡ The proposed system in this paper

$System-1$ ≡ A Decentralized system using Bluetooth LE and DIDcomm [16]'

$System-2$ ≡ A Secure Messaging Platform based on Blockchain [11]

● ≡ Represents 'Yes'

○ ≡ Represents 'No'

◒ ≡ Represents 'Partially Yes'

$\mathcal{H}$ ≡ Represents 'High Level'

$\mathcal{B}$ ≡ Represents 'Blockchain'

$\oplus$ ≡ Represents 'Local Storage'

$\cup$ ≡ Represents 'Not Mentioned'

[11], Bitmessage Plus mentioned in [14] and finally, FybrrChat mentioned in [17].

In the table, Black dot represents, 'Yes' meaning that this system has satisfied certain criteria or do have this particular feature within their system. while the White dot represents 'No' meaning they do not fulfill the criteria; and lastly Half filled fot represents 'Partially Yes' meaning they or may not have this attribute in their system. $\mathcal{H}$ represents High level architecture design where the architecture has been explained using entities and components rather than in low level details, $\mathcal{B}$ represents the system has either used Blockchain, or Distributed System respectively; and $\bigoplus$ represents instead of using any leverage or server the system used Local Storage for data storage and queuing. Lastly, $\bigcup$ represents a particular concepts or matter not mentioned or discussed in that paper.

All of these 5 systems that are mentioned in the table are decentralized but only FybrrChat and our propose system are the two only systems that has been implemented so far. Our system End to End Encrypted Peer to Peer Chat System with SSI is the only system among these 5 that has File Sharing and Transfer, and still working on integrating the Group Messaging Chat feature. In case of architecture, three of them including our system has High level architecture mentioned in the paper, while System-1 has a detailed exploration about their architecture, but Bitmessage Plus has no architecture mentioned in their paper. Three of the systems have used Blockchain for storage function, while our system has used Local storage for storing and queuing messages and file, but Bitmessage Plus didn't mentioned about storage. In case of defining a protocol flow or using any existing protocol flow, FybrrChat used a protocol called webRTC which is a very famous protocol for peer to peer systems. Our propose system has introduced protocol flow for connection establishment, messaging exchanging, file transfer and group messaging. The rest of three systems has no protocol flow.

Chapter 4

Proposal

We want to develop a chat system where individuals can chat with each-other without any central institution, thus achieving decentralization of the system. Our goal is to introduce a SSI (Self-sovereign Identity) based End-to-End Chat System which supports a Peer-to-Peer (P2P) key exchange mechanism. Because of SSI (Self-sovereign Identity) our user identities will also be decentralized which will be managed by verifiable credentials(VC) giving users extra control over their identity and credentials. This will also enhance the systems security.

4.1 Methodology

This is a security-based proposal so it is important to follow a systematic methodology to ensure a rigorous and comprehensive study. In this paper we will follow a systematic approach.
At the beginning we select a topic in phase 1 (Fig. 4.1). Then in phase 2 (Fig. 4.1) we try to identify an existing problem in the current situation. For this we looked at the current situation of the existing chat system and did some literature search & review. We find that all current chat systems/applications are centralized for providing service and identifying a user. In phase 3 Fig.4.1 we decide our objective which is building a non-centralized system. The purpose of this study is to present an alternative system where a user will get identified in a decentralized way with the help of SSI and the system will be P2P for decentralized service and E2EE for security. In phase 4 (Fig.4.1) to review information our core motive was to look for the works of others which are peer reviewed and published in reputed venues.To gather the papers our process was to follow the reference in promising research papers and also aiming in the keywords of our research like : SSI, Blockchain, P2P, end to end encryption etc which helped us to find the relevant work.We followed a thorough search method in which we reviewed the information in the following manner:

1. Read each paper's abstract and literature review.
2. Made an effort to comprehend the problems the authors have highlighted.
3. Searched for the possible solutions that the authors have suggested.
4. Looked for threat modeling and code implementations in the paper.

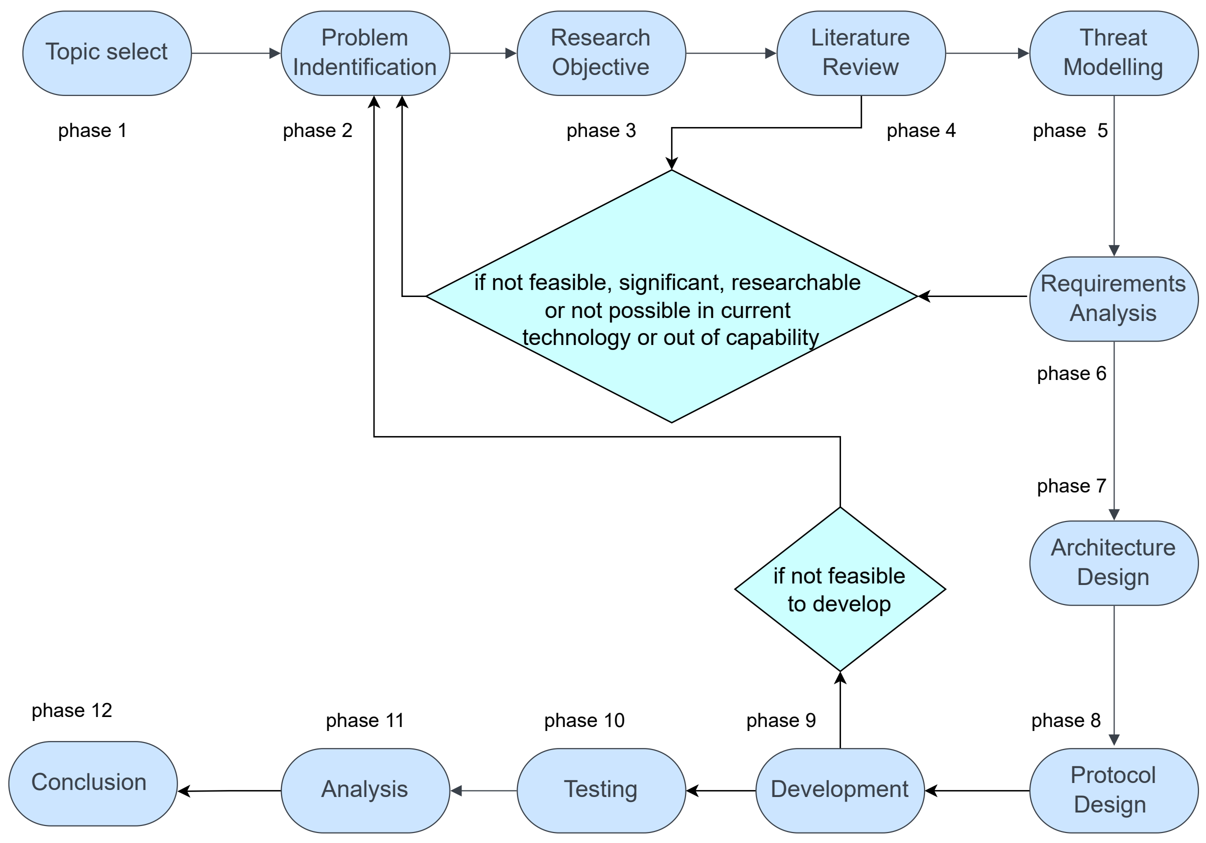


Figure 4.1: Core Methodology and Work Plan

5. Searched for the benefits and drawbacks that they have demonstrated.

These methods helped us to gain precise knowledge about earlier research and work in our research in a well prepared way. Now we enter into core methodology and to build the system we try to follow the following steps one after another. The steps are:

1. Threat Modeling
2. Requirement Analysis
3. Architecture Design
4. Protocol Flow
5. Development
6. Test

In threat modeling phase 5 (Fig.4.1) we identify the threats or attacks our system will be immune to. For that we used the STRIDE [2] model. But we also address some more vulnerabilities that will be tackled in our system which will be obtained by decentralization using DIDComm [13] protocol and VC technology.
After that, in phase 6 (Fig.4.1) we did the requirement analysis and focused on 2 things, Core functionality and security functionality. In core functionality we

address the functional requirements the system needs to have to function properly. In the non functional requirements we highlight security features which will mitigate threats. We also mention some internal functions to maintain privacy. In the next phase 7 (Fig. 4.1)we did the architecture design. Our architecture is based on SSI. In the architecture we showed how components will connect with each other and what will be in between the connections. The DIDComm protocol will be used by a mediator in the middle to establish connection.
In phase 8 (Fig. 4.1) we showed the protocol which will be maintained while sending messages from one wallet to another. Finally, in phase 9 we will develop the system using AFJ(Arice Javascript Framework), SSI, didcomm, mediator, mobile wallets etc. our system will support decentralized P2P, E2EE messaging. The system will identify users in a decentralized way. After developing In phase 10 (Fig. 4.1)we need to test the system to check its scalability, performance, speed. We also need to check its security under the mentioned attacks. In phase 11 (Fig. 4.1)we will analyze the requirements and evaluate requirements, performance, security Finally in phase 12 (Fig. 4.1) we will give our conclusion about the whole system. We have done up-to phase 9 which is some implementation. And in subsequent semesters we will focus on enhancing the system's features and will follow all the phases from phase 9 to upwards to complete our task properly.

4.2 Threat modeling

For the suggested system, it is very important to find out potential threats and analyze them before building and deploying it. For this purpose we have used STRIDE, a well-established methodology used for identifying threats and ensuring a system meets the security requirements. The security threats that we have considered for the system are given below:

- T1: Spoofing: An attacker can pretend to be a user of the system and can use it for malicious intentions.
- T2: Tampering: An attacker can modify important information such as chat data, receiver's DID, routing data, verifiable credential etc.
- T3: Repudiation: This threat can occur if a user of the system denies sending or receiving important data through the chat system.
- T4: Information Disclosure: The system can lose its anonymity which can result in disclosure of important information such as chat data, verifiable credentials etc.
- T5: Denial of Service: The system can be targeted for DoS attack, making it unavailable for the verified users
- T6: Elevation of Privilege: An attacker can use other attack vectors to elevate their privilege in the system. Apart from these threats we have also considered additional threats which are important for secured chat service. These are –
- T7: Replay attack: An attacker can eavesdrop a message transfer, capture it and resend or delay the transfer which can result in miscommunication between entities.

- T8: Lack of consent: A message can be sent without the consent of the sender which can contain sensitive information.
- T9: Lack of control: Sensitive data which is supposed to be accessed by only the sender and the receiver can be accessed by any third party.

4.3 Requirement analysis

In this section, we present a set of functional, security and privacy requirements. The functional requirements focus on the core functionalities of the system. The security and privacy requirements ensure they mitigate the threats we identified.

4.3.1 Functional Requirements

- F1: The system must integrate SSI mechanisms into its services so that the end users can communicate only after following SSI steps.
- F2: The system must include a mobile agent or wallet for the end users which will facilitate data transfer between entities.
- F3: The system must use verifiable credentials and DID for each entity to create a channel or route for communication.
- F4: The system should support transferring plain texts after encrypting it through mobile agents owned by end users.
- F5: The system must ensure that there is local storage to store sensitive data.
- F6: The users should be able to access the previous chats they had with the other entity.
- F7: The users should be able to upload files and send it to the other users within the system.
- F8: The users should be able to download files sent by another user.
- F9: The users should be able to create new group by selecting and adding new members.
- F10: The users should be able send messages all members of a group simultaneously.

4.3.2 Security Requirements

- S1: The system must ensure that the users are authenticated using verifiable credentials based on SSI and only after that they can access chat service. This will mitigate the T1 and T6 threat.
- S2: The system must ensure the verifiable credential or chat information is not modified by an unauthorized entity which will mitigate T2 threat.

- S3: The system must use digital signature from issuers in verifiable credential to make it tamper-evident. This will mitigate the T3 threat.
- S4: The system must ensure the chat data is encrypted from the end device and only can be decrypted from the receiver's end device so that the T4 threat can be mitigated.
- S5: The system should use already available measures to avoid DoS attack to mitigate T5 threat.
- S6: The system must take necessary measures against replay attacks to mitigate the T7 threat

Chapter 5

Architecture

The fundamental purpose of the system is to verify a user in a decentralized way. But a user cannot communicate with others directly. So, they need a wallet to get the functionalities and an agent for mobile end-point by which they participate in the whole process. In the Fig. 5.1 shows a high level architecture of SSI based chat system that has 2 users Alice and Bob and both of them hold a mobile wallet which contains an agent.

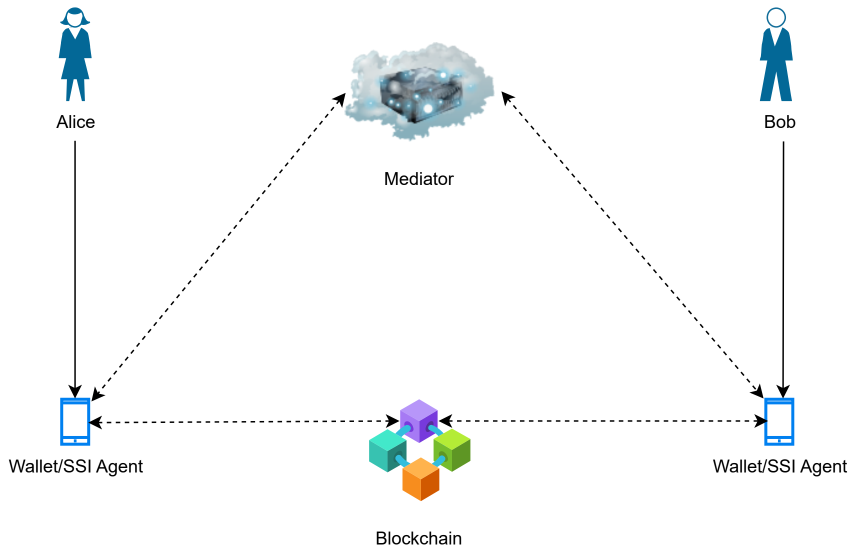


Figure 5.1: SSI based chat system

Two wallets cannot communicate with one another without a central server. So, the connection is established using a mediator. Mediators can send and route messages globally to other mediators and they can also communicate locally with the wallet agent. Apart from routing messages, Mediators also can create secure communication channel, verifying credentials, and provides privacy and anonymity. At first the communication between two wallets are established physically by scanning a qr code. In Fig. 5.1 the straight solid arrow indicates the user holds an agent or wallet while the dotted bidirectional arrow means there is a connection or communication and it points towards the network dependent entitites. A user first creates a link of his wallet agent and converts that into a qr code with some additional information.If Alice wants to send a message to Bob first she needs to establish a connection by mediator. DIDcomm protocol will be used here. Finally Alice can send a text to Bob. This architecture is high level design of a simple SSI based chat system using which 2 users can communicate through messages exchanging, file sharing and transferring. Blockchain in both of these architectures can be used to provide immutable records, trust and verification as well as decentralization. Although in both of these diagrams, Blockchain is used only for leveraging the public DIDs.

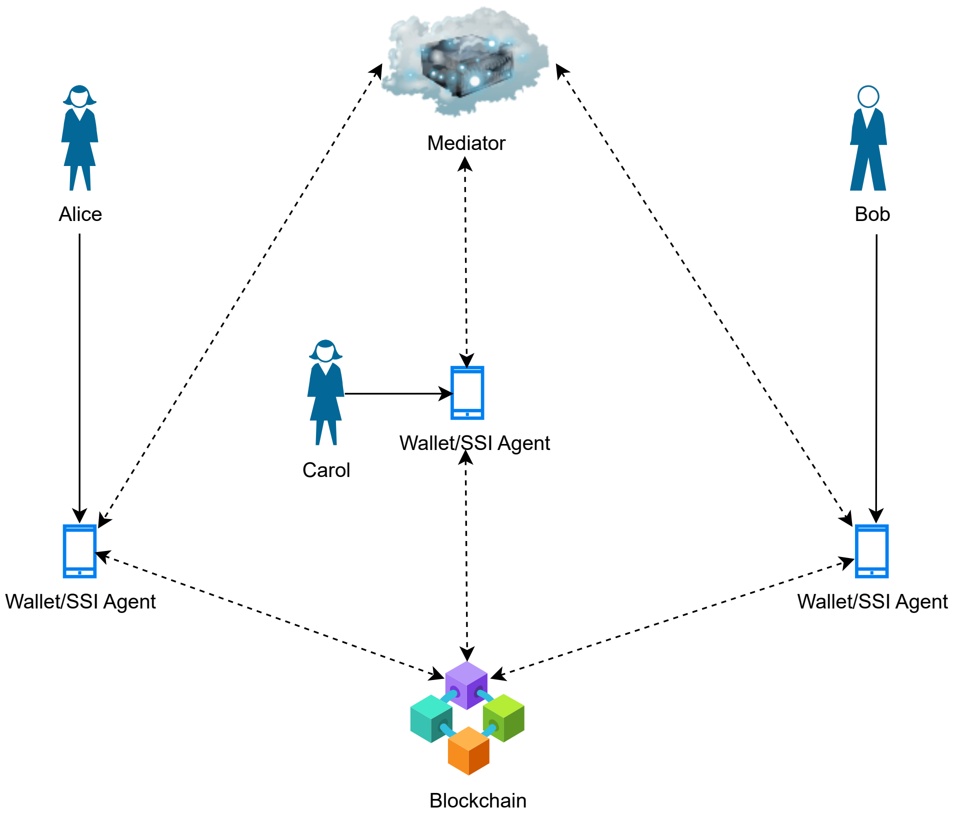


Figure 5.2: SSI based group chat system

In the Fig. 5.2 a high level architecture of a SSI based group chat system has been shown among three users, Alice, Bob and Carol. This architecture is pretty similar compared to Fig. 5.1 in terms of functionality and processes. In both of these architecture, Mediators, Blockchain together with Wallets or SSI agents provides the system with identity creation and management, secure communication and messaging, credentials issuing, verifying and sharing and privacy maintenance as well.

5.1 Use Case and Protocol Flow

Now in this section, we will present a use case scenario with protocol flows and visual representation of how two different users can establish connection between them and send messages or files to each other using this system. First in the Table 5.1, we will describe some of the mathematical notations which will be used to describe the Data Model, Protocol Flows of Connection Establishment, Messages Exchanging

Table 5.1: Cryptographic Notation - One-to-One Chat

Notations	Descriptions
U1	User-1
U2	User-2
M	Mediator
K_{cek}	Symmetric or Content Encryption Key
K_{U2}^{U1}	Public Key of U2 to be used with U1
$K_{U2}^{-1\|U1}$	Private Key of U2 to be used with U1
K_{U1}^{U2}	Public Key of U1 to be used with U2
$K_{U1}^{-1\|U2}$	Private Key of U2 to be used with U1
K_{M}^{U1}	Public Key of Mediator M to be used with U1
$K_{M}^{-1\|U1}$	Private Key of Mediator M to be used with U1
K_{U1}^{M}	Public Key of U1 to be used with Mediator M
$K_{U1}^{-1\|M}$	Private Key of U1 to be used with Mediator M
K_{M}^{U2}	Public Key of Mediator M to be used with U2
$K_{M}^{-1\|U2}$	Private Key of Mediator M to be used with U2
K_{U2}^{M}	Public Key of U2 to be used with Mediator M
$K_{U2}^{-1\|M}$	Public Key of U2 to be used with Mediator M
$\{\}_K$	Encryption using a public or symmetric key
$\{\}_{K^{-1}}$	Digital Signature using a private key
$\{K, K^{-1}\}$	Sender's Key Pairs
msg	Textual Messages send between Users
DID_{U2}^{U1}	Decentralized Identifiers of U2 for U1
DID_{U1}^{U2}	Decentralized Identifiers of U1 for U2
VC_{U1}^{U2}	Verifiable Credential issued by U1 to U2
$[.....]_K$	Communication over an encrypted channel using key K
$[.....]$	Communication over an unencrypted channel
$()_{base64enc}$	Base-64 encoding of image
$()_{base64dec}$	Base-64 decoding of image
$()_{resize}$	Resize of Image

and Image Transfer showed in Table 5.2, 5.3, 5.4 and 5.5 respectively.

Data Model: Then comes the Table 5.2, Data Model which gives a detailed description of the types of requests and responses this system makes and the types of data this system uses and what they contain. First inviteReq contains url, recKeys and routKeys which are Recipient Keys and Routing Keys (optional in most cases) of the request service respectively. For inviteReq there are two corresponding responses, inviteRes-1 and inviteRes-2. The first one contains the Decentralized Identifiers (DID) of U2 to be used with U1, DID_{U2}^{U1}; while the later one contains the Decentralized Identifiers (DID) of U1 for U2, DID_{U1}^{U2}. Next, we have packMsg which contains walletHandle, basically key pairs of sender, Message encrypted with content encryption key which is the text that is sent by the user, senVerKeys and recVerKeys which are verification keys of sender and recipients respetively, and lastly content encryption key encrypted with recVerKeys.

Table 5.2: Data Model

Request/Response	Contents of Request or Response
inviteReq	(url, recKeys, routKeys)
inviteRes-1	(DID_{U2}^{U1})
inviteRes-2	(DID_{U1}^{U2})
walletHandle	(Sender's Key Pairs)
senVerKeys	(Public key of Sender to be used with Recipients)
recVerKeys	(Public keys of Recipients to be used with Sender)
packMsg	(walletHandle, $\{\{msg\}_{K_{cek}}$, $\{K_{cek}\}_{recVerKeys}\}$, senVerKeys, recVerKeys)

Table 5.3: Protocol Flow - Connection Establish

Request or Response	Contents of Request or Response
1. U2 $\rightarrow$ U1	$[newInviteGen]_{http}$
2. U1 $\rightarrow$ U2	$[inviteReq]_{http}$
3. U2 $\rightarrow$ M	$[U1, \{inviteRes - 1\}_{K_{U1}^{U2}}]_{K_M^{U2}}$
4. M $\rightarrow$ U1	$[\{inviteRes - 1\}_{K_{U1}^{U2}}]_{K_{U1}^M}$
5. U1 $\rightarrow$ M	$[U2, \{inviteRes - 2\}_{K_{U2}^{U1}}]_{K_M^{U1}}$
6. M $\rightarrow$ U2	$[\{inviteRes - 2\}_{K_{U2}^{U1}}]_{K_{U2}^M}$

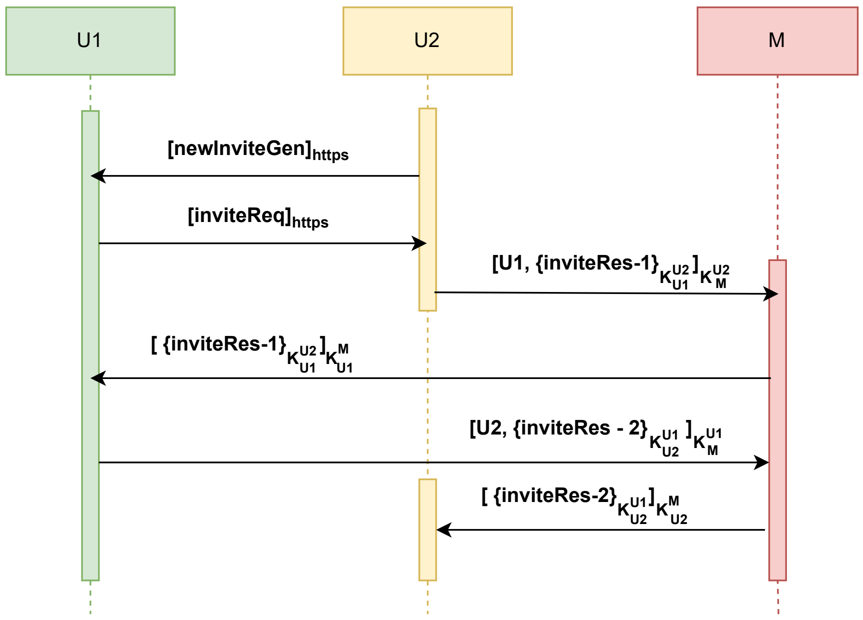

U1
U2
M
$[newInviteGen]_{https}$
$[inviteReq]_{https}$
$[U1, \{inviteRes-1\}_{K_{U1}^{U2}}]_{K_M^{U2}}$
$[\{inviteRes-1\}_{K_{U1}^{U2}}]_{K_{U1}^M}$
$[U2, \{inviteRes - 2\}_{K_{U2}^{U1}}]_{K_M^{U1}}$
$[\{inviteRes-2\}_{K_{U2}^{U1}}]_{K_{U2}^M}$

Figure 5.3: Sequence Diagram - Connection Establish

Protocol Flow - Connection Establish: Now we will demonstrate how this system helps two users to establish a fully secure and encrypted connection between them in Table 5.3.

Table 5.4: Protocol Flow - Messages Exchanging

Request or Response	Contents of Request or Response
1. U2 $\rightarrow$ M	$[U1, \{packMsg\}_{K_{U1}^{U2}}]_{K_M^{U2}}$
2. M $\rightarrow$ U1	$[\{packMsg\}_{K_{U1}^{U2}}]_{K_{U1}^M}$
3. U1 $\rightarrow$ M	$[U2, \{packMsg\}_{K_{U2}^{U1}}]_{K_M^{U1}}$
4. M $\rightarrow$ U2	$[\{packMsg\}_{K_{U2}^{U1}}]_{K_{U2}^M}$

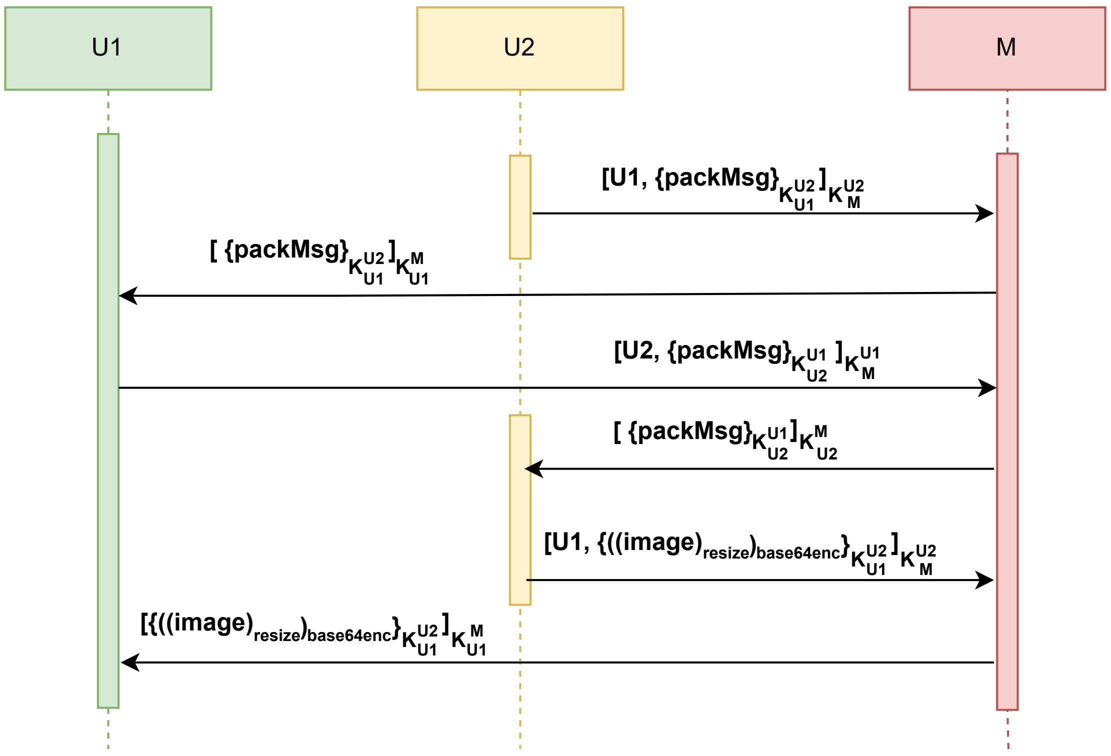

U1
U2
M
[U1, {packMsg}$_{K_{U1}^{U2}}$]$_{K_M^{U2}}$
[{packMsg}$_{K_{U1}^{U2}}$]$_{K_{U1}^M}$
[U2, {packMsg}$_{K_{U2}^{U1}}$]$_{K_M^{U1}}$
[{packMsg}$_{K_{U2}^{U1}}$]$_{K_{U2}^M}$
[U1, {((image)$_{resize}$)$_{base64enc}$}$_{K_{U1}^{U2}}$]$_{K_M^{U2}}$
[{((image)$_{resize}$)$_{base64enc}$}$_{K_{U1}^{U2}}$]$_{K_{U1}^M}$

Figure 5.4: Sequence Diagram - Message Exchanging and File Transfer

- In the step 1, U2 submits a request to U1 to generate invitation request via Bifold Wallet.
- Then in the step 2, U1 generates an Invitation which is encoded as QR code and returns the inviteReq to the U2.
- At step 3 and step 4, U2 scans the QR code and generates a key pair of public and private key and a DID of U2 for U1, DID_{U2}^{U1}. The U2 then uses the wallet to create inviteRes-1 using the DID and encrypts it with the public key of the U1 to be used with U2, K_{U1}^{U2}; and sends it to the mediator, M in order to forward it to the U1 over an encrypted channel that is encrypted with K_M^{U2}. The mediator then at first decrypts it with its private key, encrypts it with U1's public key to be used with M, K_{U1}^M and then send it to U1. The overall process will be carried on an encrypted channel. U1 will then receive the data from Mediator and in this overall process the data is always encrypted with K_{U1}^{U2}.

- Similarly at step 5 and step 6, U1 first uses the wallet to create another invitation, inviteRes-2 using the DID, encrypts it with the public key of the U2 to be used with U1 and sends it to the Mediator in order to forward it to the U2; the Mediator then decrypts it and sends it to U2 after encrypting with the public key of U2 to be used with M.
- In this way, both U1 and U2, using their user wallet retrieves and stores the DID, required connection data from both of the inviteRes-1 and inviteRes-2, integrate and stores them to save the connection in the wallet and establish a secure and encrypted connection between them.
- The Sequence Diagram at the Figure 5.3 shows the flow of the process of connection establishing.

Protocol Flow - Messages Exchanging: After successfully establishing the connection between U1 and U2; it's time for both of the user to send messages to each other in the Table 5.4.

- In the step 1, U2 first encrypts the packMsg with the public key of U1 to be used with U2, K_{U1}^{U2}; and instructs the Mediator to forward the encrypted message to U1. Then before sending it to the Mediator, U2 encrypts the entire channel that passes message with the public key of Mediator to be used with U1, K_M^{U1} which indicates that the channel from where the messages are passing through is secure and encrypted.
- In the step 2; Mediator at first decrypts the message and see the instruction that tells it to forward the message to U1; so, the Mediator then forwards the message to the U1.
- Similarly in the above manner in setp 3 and 4, U1 then send the packMsg to the Mediator by encrypting with the public key of U2 for U1, K_{U2}^{U1} and then the channel with public key of the Mediator for U1, K_M^{U1}; instructing it to forward to the U2. The Mediator then decrypts it with its private key, reads the instruction, send it to the U2. Thus both of the user completes the process of exchanging messages between them. Again, the channel is securely encrypted.
- The Sequence Diagram at the Figure 5.4 shows the flow of the process of messages exchanging.

Table 5.5: Protocol Flow - File Transfer

Request or Response	Contents of Request or Response
1. U2 → M	$[U1, \{((image)_{resize})_{base64enc}\}_{K_{U1}^{U2}}]_{K_M^{U2}}$
2. M → U1	$[\{((image)_{resize})_{base64enc}\}_{K_{U1}^{U2}}]_{K_{U1}^{M}}$

Protocol Flow - File Transfer: It's time for both users to send images to each other in the Table 5.5.

- First U1 will select the image from mobile storage and then press the send button. After pressing the send button in the backend of the app the image will first get resized and then the resized image will be encoded into a base64 string and the string will be sent through mediator to the another user. Into the backend of the U2's app the base64 encoded string will be decoded into a image which U2 can derictly see into his/her inbox. The Sequence Diagram at the Figure 5.4 shows the flow of the process of file transfer.

Table 5.6: Cryptographic Notations - Group Chat

Notations	Descriptions
A	User A (Alice - group member)
B	User B (Bob - group member)
C	User C (Carol - group member)
DID_X	Decentralized Identifier (DID) of User X
H(DID_X, DID_Y, DID_Z)	Hash of concatenation of DIDs of group members X, Y, Z
K_A^M	Public Key of A to be used with M
$K_A^{-1\|M}$	Private Key of A to be used with M
K_B^M	Public Key of B to be used with M
$K_B^{-1\|M}$	Private Key of B to be used with M
K_C^M	Public Key of C to be used with M
$K_C^{-1\|M}$	Private Key of C to be used with M
K_M^A	Public Key of M to be used with A
$K_M^{-1\|A}$	Private Key of M to be used with A
K_M^B	Public Key M to be used with B
$K_M^{-1\|B}$	Private Key of M to be used with B
K_M^C	Public Key of M to be used with C
$K_M^{-1\|C}$	Private Key of M to be used with C
K_A^B	Public Key of A to be used with B
$K_A^{-1\|B}$	Private Key of A to be used with B
K_A^C	Public Key of A to be used with C
$K_A^{-1\|C}$	Private Key of A to be used with C
K_B^A	Public Key of B to be used with A
$K_B^{-1\|A}$	Private Key of B to be used with A
K_B^C	Public Key of B to be used with C
$K_B^{-1\|C}$	Private Key of B to be used with C
K_C^A	Public Key of C to be used with A
$K_C^{-1\|A}$	Private Key of C to be used with A
K_C^B	Public Key of C to be used with B
$K_C^{-1\|B}$	Private Key of C to be used with B
msg	Messages exchanged between users
notify	A group chat joining notification
$[.....]_K$	Communication over an encrypted channel using K

Protocol Flow - Group Establish and Message Exchanging: Table 5.6 similarly discusses some cryptographic notations which includes cryptographic notations for users, their public keys, DIDs, messages and notifications.

- Table-5.6 is the protocol flow that describes the process of sending and receiving messages by users in group chat discussions. In the step-1, A (User-A), selects B (User-B) and C (User-C) as group members, creates a chat window with them and uses the hash of the catenation of all of the users' DIDs, H(DID_A, DID_B, DID_C) as the label to create a group chat. For now, this protocol will assume that in order to create a group all of the members of the group have to connected with each other. If all of them are connected with each other, then the users will be able to create a group with their selected members and start their discussions. In step-2 and step-3, B and C follow the same process by selecting the other two users, use the hash, H(DID_A, DID_B, DID_C) as labels and create a group chat from their end.

Table 5.7: Protocol Flow - Group Chat Creation

1. A selects B, C, creates chat window, use H(DID_A, DID_B, DID_C) as label.
2. B selects A, C, creates chat window, use H(DID_A, DID_B, DID_C) as label.
3. C selects A, B, creates chat window, use H(DID_A, DID_B, DID_C) as label.
4. A → M : $[\{B, \{notify\}_{K_B^A}\}]_{K_M^A}$
5. M → B : $[\{notify\}_{K_B^A}]_{K_B^M}$
6. A → M : $[\{C, \{notify\}_{K_C^A}\}]_{K_M^A}$
7. M → C : $[\{notify\}_{K_C^A}]_{K_C^M}$
8. B → M : $[\{A, \{notify\}_{K_A^B}\}]_{K_M^B}$
9. M → A : $[\{notify\}_{K_A^B}]_{K_A^M}$
10. B → M : $[\{C, \{notify\}_{K_C^B}\}]_{K_M^B}$
11. M → C : $[\{notify\}_{K_C^B}]_{K_C^M}$
12. C → M : $[\{A, \{notify\}_{K_A^C}\}]_{K_M^C}$
13. M → A : $[\{notify\}_{K_A^C}]_{K_A^M}$
14. C → M : $[\{B, \{notify\}_{K_B^C}\}]_{K_M^C}$
15. M → B : $[\{notify\}_{K_B^C}]_{K_B^M}$
16. A → M : $[\{B, \{msg\}_{K_B^A}\}]_{K_M^A}$
17. M → B : $[\{msg\}_{K_B^A}]_{K_B^M}$
18. A → M : $[\{C, \{msg\}_{K_C^A}\}]_{K_M^A}$
19. M → C : $[\{msg\}_{K_C^A}]_{K_C^M}$
20. B → M : $[\{A, \{msg\}_{K_A^B}\}]_{K_M^B}$
21. M → A : $[\{msg\}_{K_A^B}]_{K_A^M}$
22. C → M : $[\{A, \{msg\}_{K_A^C}\}]_{K_M^C}$
23. M → A : $[\{msg\}_{K_A^C}]_{K_C^M}$

- From step step-4 to step-7, A instructs the Mediator, M to send B and C a simple notifications that confirms that he (A) joined the group that has been created by them individually from their end, and instruct them to send it in an encrypted channel that is encrypted with K_M^A, with encrypting the data with public key of respective users, K_B^A and K_C^A. Mediator, M then sends those notifications to B and C as per the instructions. This channel is encrypted too making it a secured communication among the Mediator M, B and C.

- From step-8 to step-15, both B and C will follow similar processes where they will instruct the Mediator , M to pass their data to their group chat members; M will follow the same instructions for B and C and forward it to their fellow group users.

- From step-16 to step-19, A instructs the Mediator, M to send B and C some data that he sends to the group in an encrypted channel, encrypted with K_M^A. This time instead of sending a notification, A will send a textual message to start a communication or discussion between the users. Mediator M then sends those messages to B and C, K_B^A and K_C^A as per the instructions in step-14 and step-15, encrypting the data using the public key of respective users and sending it within an encrypted channel to make it more secure. From step-16 to step-21, both B and C will follow similar processes to continue their discussion and Mediator M will follow the same instructions for B and C. Thus, completing the entire process of group communication among multiple users.

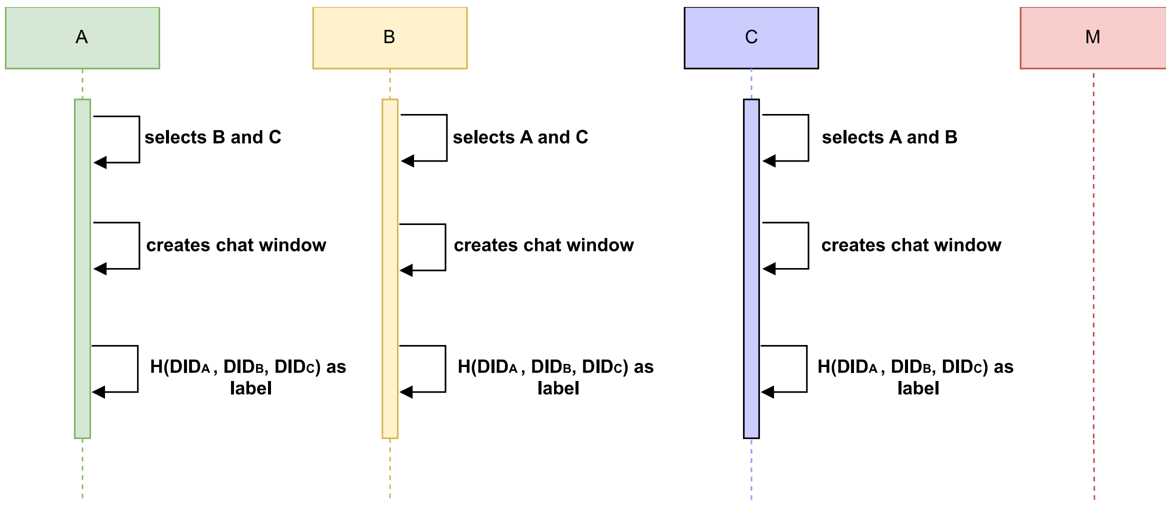


Figure 5.5: Sequence Diagram - Group Members Selection

The Sequence Diagram at the Figure 5.5, 5.6, 5.7 shows the flow of the process of group chat features.

Use case and Protocol through Visualization: Now that we have already understand the protocol flow and its use case descriptions, it's time to see the visual representation of the app, how it functions and how a user will operate through the application, make a connection and have a discussion with their other connected users. In the protocol flow, we have mentioned two users, U1 and U2; let's say, U1 wants to make a connection with U2 and have some discussion. Below, through visual images we will present how user U1 will communicate with user U2.

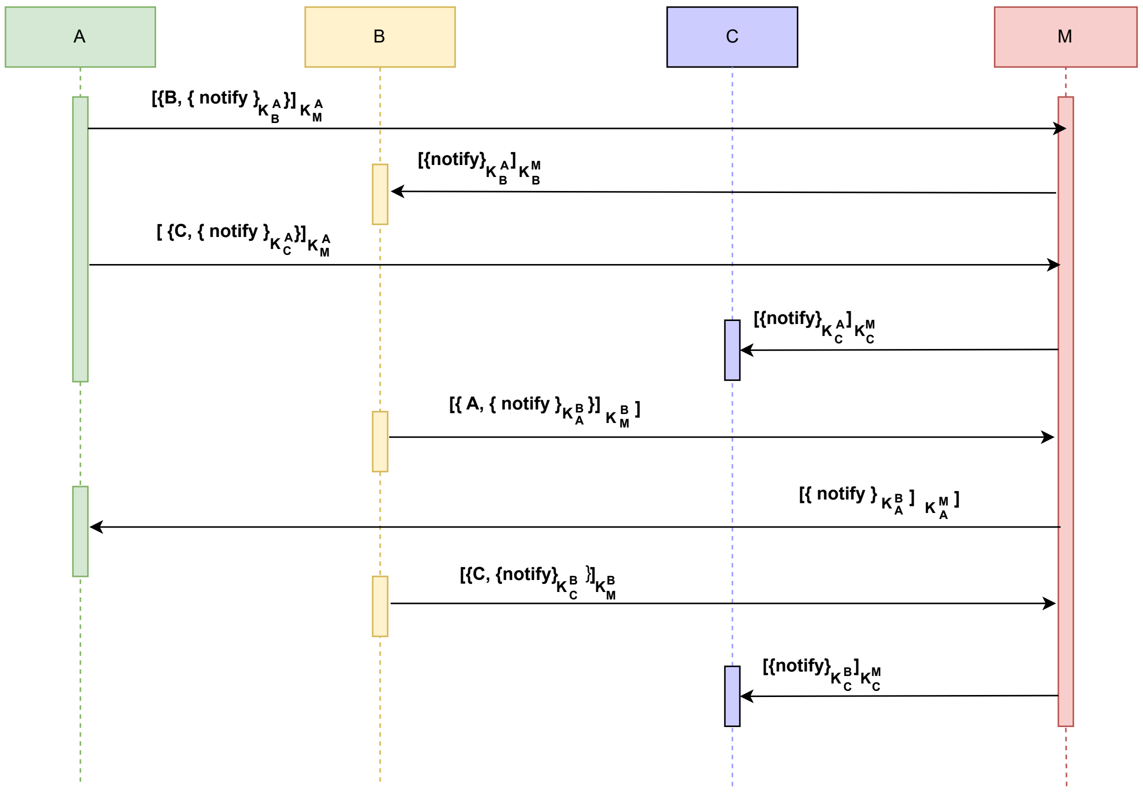

A
B
C
M
[{B, { notify }$_{K_B^A}$}]$_{K_M^A}$
[{notify}$_{K_B^A}$]$_{K_B^M}$
[{C, { notify }$_{K_C^A}$}]$_{K_M^A}$
[{notify}$_{K_C^A}$]$_{K_C^M}$
[{ A, { notify }$_{K_A^B}$}] $_{K_M^B}$]
[{ notify }$_{K_A^B}$] $_{K_A^M}$]
[{C, {notify}$_{K_C^B}$ }]$_{K_M^B}$
[{notify}$_{K_C^B}$]$_{K_C^M}$

Figure 5.6: Sequence Diagram - Group Chat Connection Establishment

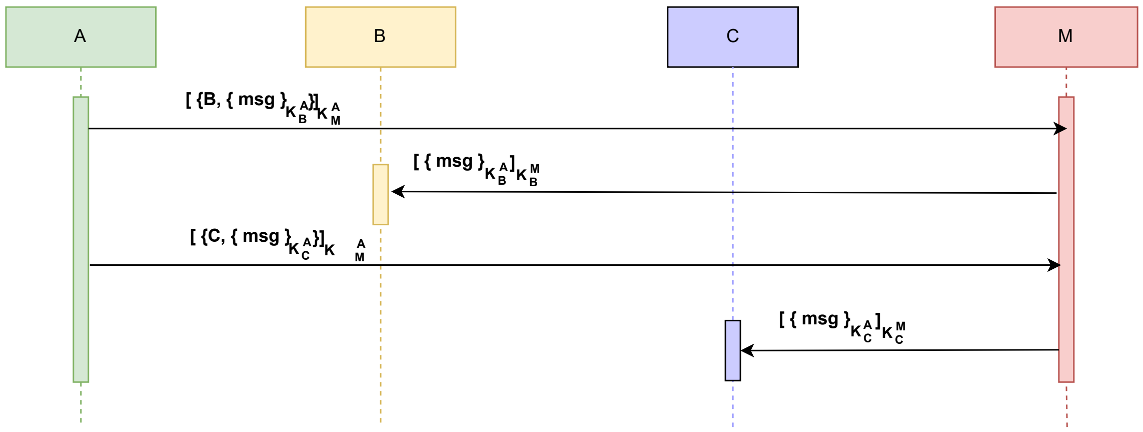
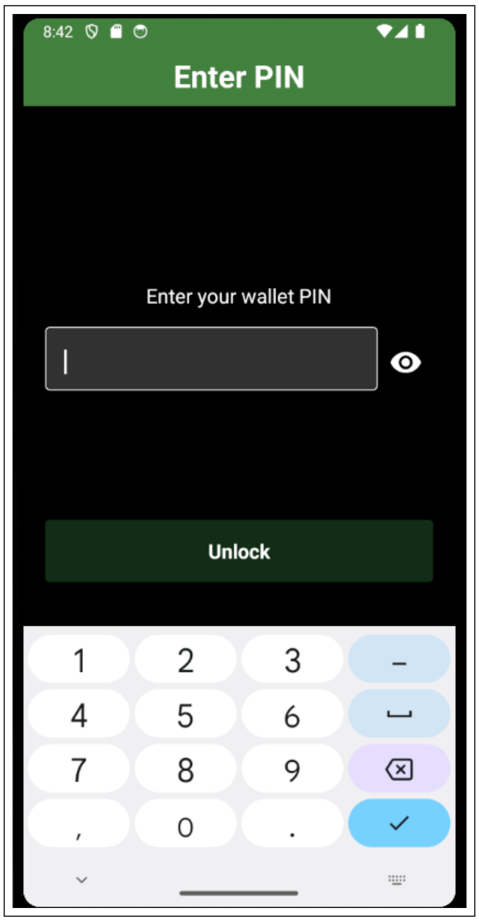

A
B
C
M
[{B, { msg }$_{K_B^A}$}]$_{K_M^A}$
[{ msg }$_{K_B^A}$]$_{K_B^M}$
[{C, { msg }$_{K_C^A}$}]$_{K_M^A}$
[{ msg }$_{K_C^A}$]$_{K_C^M}$

Figure 5.7: Sequence Diagram - Group Chat Messages Exchanging

1. Starting the Aries Bifold Wallet Application:
- Aries Bifold is a mobile application which is installed on U1's mobile device. To start the application first U1 taps the application to open it. A window appeared in front of U1 asking to enter a pin to get inside the app. This is for the security reason that even if someone gets physical access to U1's phone still they may not access the application. After giving the correct pin U1 press a button called 'Unlock' to get inside the application (In the Figure: 5.8a). It will the U1 to the home screen of the Bifold Wallet Application (In the Figure: 5.8b).



(a) Aries Bifold - Entering the PIN and unlocking the application

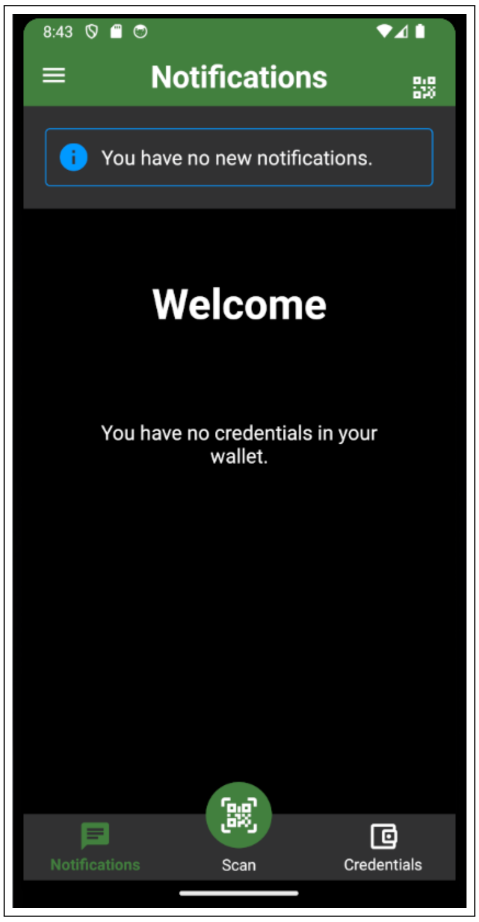
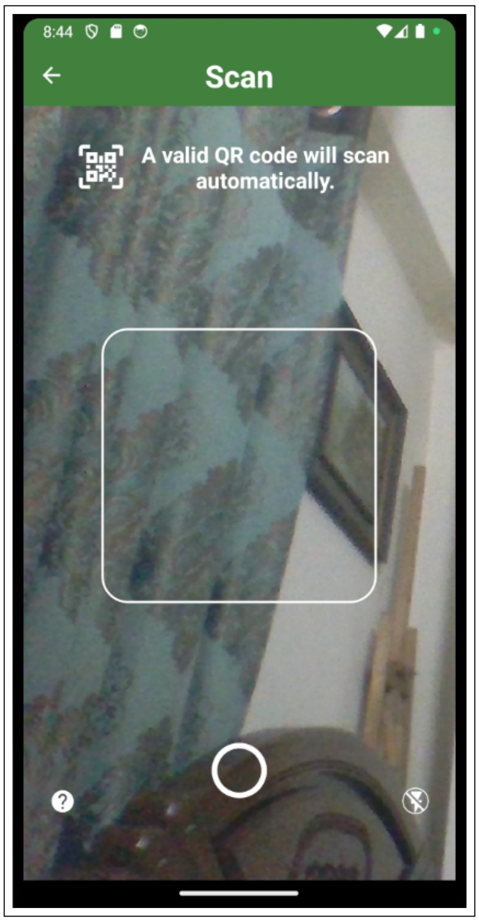

8:43
Notifications
You have no new notifications.
Welcome
You have no credentials in your wallet.
Notifications
Scan
Credentials
(b) Aries Bifold - Home Screen

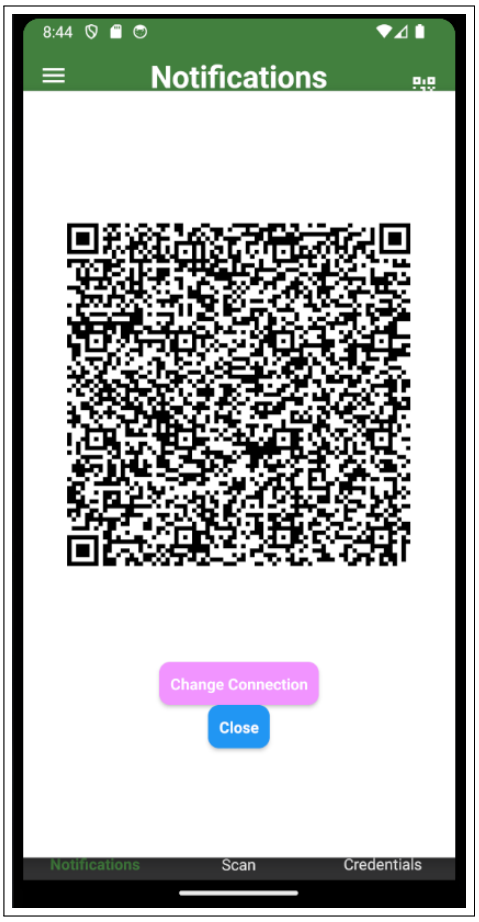
Figure 5.8: Starting the Application

2. U1 Building the connection with U2:
- Inside the application there is a QR code like button at the top right (In the Figure 5.8b). To generate a QR code U1 press on the button, a new window opens with a QR code on it (In the Figure 5.9b). This OR code is being used to make a connection. This OR code contains information like an invitation URL. To establish connection between U1 and U2, U2 needs to follow the same process to get inside the application on his/her device.

- Inside the application, U2 will find a button called 'Scan' at the bottom middle (In the Figure 5.8b). U2 presses the 'Scan' button and opens a scanner inside the app (In the Figure 5.9a). Now from his/her device with the help of that scanner U2 scans the QR code (In the Figure 5.9b) which is being generated in U1's device. After scanning, a connection was established between U1 and U2 with the help of a mediator.



(a) Aries Bifold - Scanning Screen

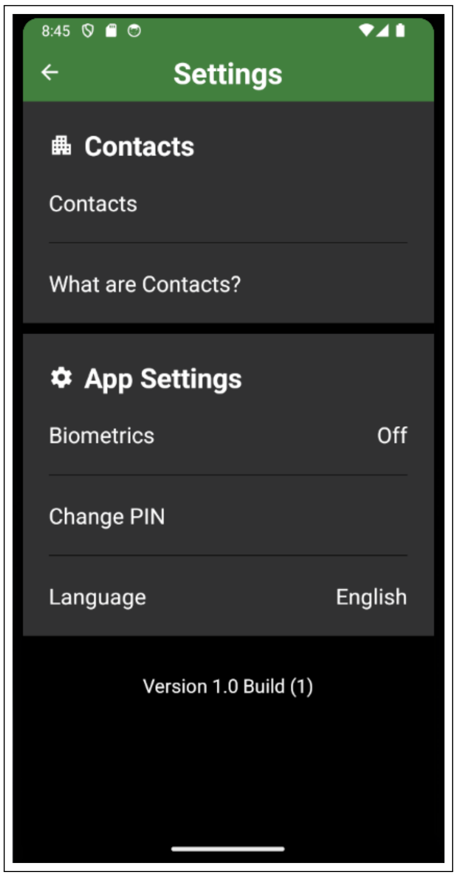


(b) Aries Bifold - Generated QR Code

Figure 5.9: Building the Connection between users

3. Accesing the contacts and sending messages:
- There is a button which looks like 3 horizontal lines symbol at the top left corner (In the Figure 5.8b). U1 presses the button and comes to a page named Settings. In Settings there is Contacts (In the Figure 5.10a). U1 taps into the Contacts. This will let U1 get inside Contacts where U1 will find all his/her connections (In the Figure 5.10a). From there U1 selects the connection which is with U2 and gets inside a chat window (In the Figure 5.12b and Figure 5.10b).

- There U1 writes his/her msg and presses the send button to send the message to U2. U2 can see the message U1 has sent him/her in a chat window inside his/her contacts. And can send messages to U1 from that chat window (In the Figure 5.12b and Figure 5.10b). React Native uses TCP Socket to transfer messages. TCP socket is mainly used to transfer small size data which is very useful here as there is no centralized database.
4. Sending Image files to other users:


8:45

Settings

Contacts

Contacts

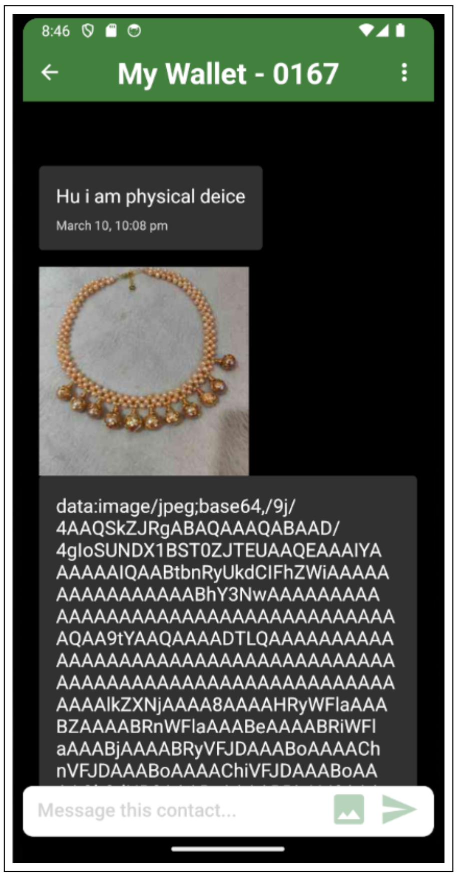
What are Contacts?

App Settings

Biometrics	Off
Change PIN	
Language	English

Version 1.0 Build (1)

(a) Aries Bifold - Accessing Chats

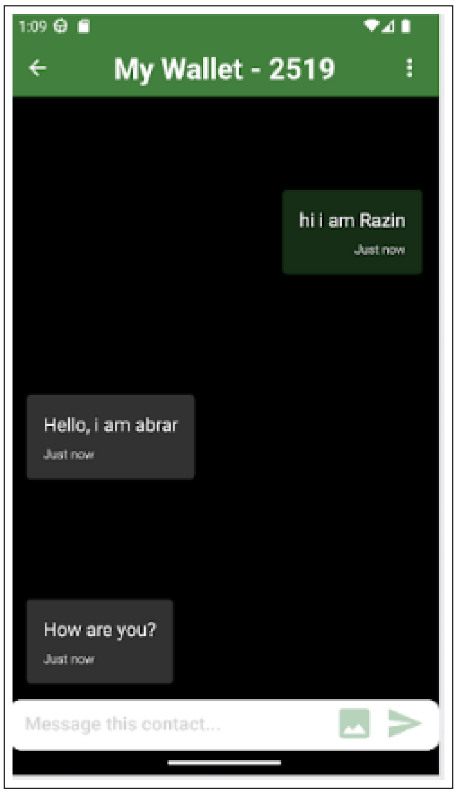

8:46

My Wallet - 0167

Hu i am physical deice
March 10, 10:08 pm

data:image/jpeg;base64,/9j/
4AAQSkZJRgABAQAAAQABAAD/
4gIoSUNDX1BST0ZJTEUAAQEAAAIYA
AAAAAIQAABtbnRyUkdCIFhZWiAAAAA
AAAAAAAAAABhY3NwAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAA9tYAAQAAAADTLQAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAlkZXNjAAAA8AAAAHRyWFlaAAA
BZAAAABRnWFlaAAABeAAAABRiWFl
aAAABjAAAABRyVFJDAAABoAAAACh
nVFJDAAABoAAAAChiVFJDAAABoAA

(b) Aries Bifold - Sending Messages-1


1:09

My Wallet - 2519

hi i am Razin
Just now

Hello, i am abrar
Just now

How are you?
Just now

(c) Aries Bifold - Sending Messages-2

Figure 5.10: Accessing chats and Messages Exchanging between users

- To send image U1 select the image icon which is at the bottom of the chat window (In the Figure 5.12b). After pressing the image icon a floating window pops up where U1 can select images from his/her gallery (In the Figure 5.11a). After selecting, the image is encoded to base64 and checked if the size is less than a specific threshold. If yes, the image is being passed with react conditional rendering. If not the image is being resized and encoded. Then U1 can find the image floating beside the text box inside the chat window (In the Figure 5.11b). The floating image has a copy button and a cross button (In the Figure 5.11b). Cross button will remove the image. U1 can copy the encoded base64 string using that copy button over the image and paste it in the text box (In the Figure 5.11c and Figure 5.12a). Then U1 can send that image by pressing the send button (In the Figure 5.12a). If a base64 string is passed through the react image tag it will render the image (In the Figure 5.12b and Figure 5.12c). Because U2's device will receive the same string it will render the same image. This is why messages and images are synchronized in the app.

5. Group messaging chat among multiple users:
- So far, the messaging, file sending and receiving between two users have been completed; the only part of the implementation that is yet to be done is group messaging feature where more than two users who are all connected with each other, send messages and have a discussion among them. Although, we have already designed a protocol for group messaging among multiple users by following good design principles, but we have fully completed the implementation yet.

- Till now, we have been able to design a UI screen where we will be able see the users from contact list and select them for the group creation (In the Figure 5.13).

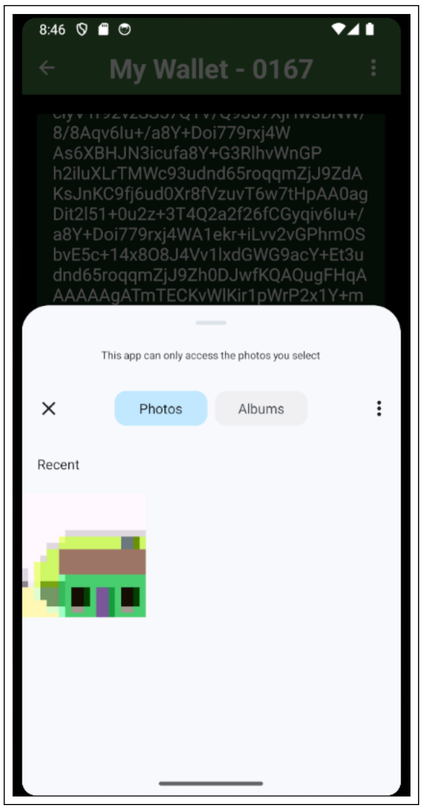
- But we are facing some issue from here. At first, we had to build a UI where a user can select other users with whom that user wants to create a group. For that we created a floating action button in the contacts screen UI (In the Figure 5.13).

- By clicking that action button, the application will show the user a contact list of the users that are connected with him/her. But we are facing some problems to build the logic of the coding implementation for that.

- First of all, our primary approach was to bring the entire contact list, store it into the application and bring it into the state of the application. But the problem in this approach is we were facing decision making difficulties in handling the data of multiple users, whether we should use react state to do that or shift to SQLite.

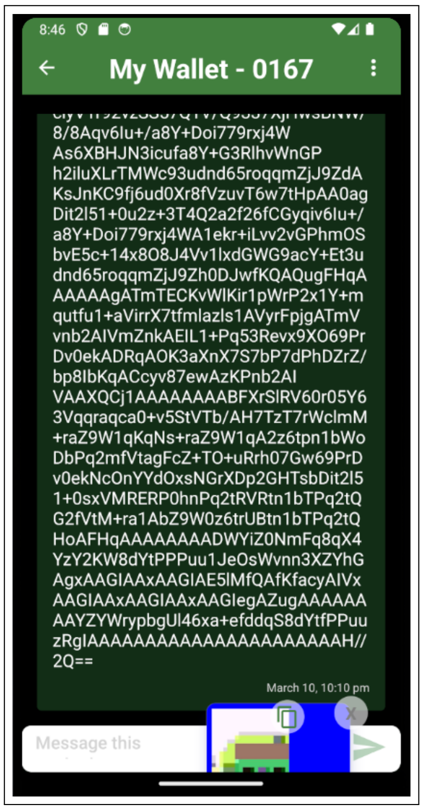
- To illustrate, if we use SQLite, we are here facing the problem of mapping the data using SQLite due to problems; the problems were basically regarding the permission to read SQLite existing database as it was under the Aries Aksar, so it was not permissible to read or edit it.

- On the other hand, as the application has already been implemented, accessing



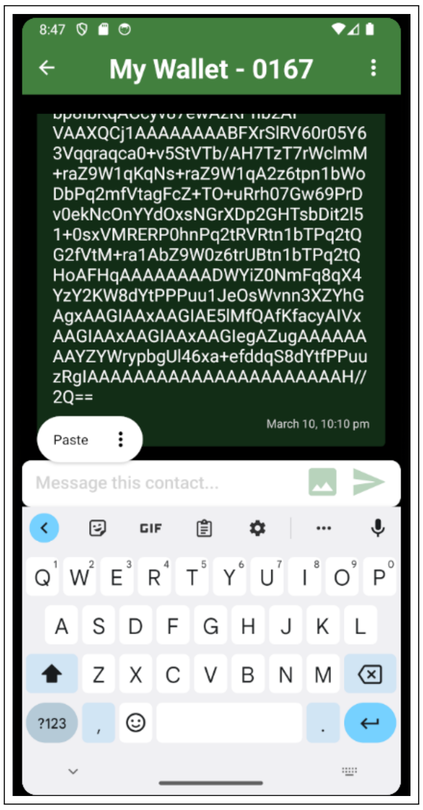
8/8Aqv6lu+/a8Y+Doi779rxj4W
As6XBHJN3icufa8Y+G3RlhvWnGP
h2iluXLrTMWc93udnd65roqqmZjJ9ZdA
KsJnKC9fj6ud0Xr8fVzuvT6w7tHpAA0ag
Dit2l51+0u2z+3T4Q2a2f26fCGyqiv6lu+/
a8Y+Doi779rxj4WA1ekr+iLvv2vGPhmOS
bvE5c+14x8O8J4Vv1lxdGWG9acY+Et3u
dnd65roqqmZjJ9Zh0DJwfKQAQugFHqA
AAAAAgATmTECKvWlKir1pWrP2x1Y+m

(a) Aries Bifold - Pressing Image Option



8/8Aqv6lu+/a8Y+Doi779rxj4W
As6XBHJN3icufa8Y+G3RlhvWnGP
h2iluXLrTMWc93udnd65roqqmZjJ9ZdA
KsJnKC9fj6ud0Xr8fVzuvT6w7tHpAA0ag
Dit2l51+0u2z+3T4Q2a2f26fCGyqiv6lu+/
a8Y+Doi779rxj4WA1ekr+iLvv2vGPhmOS
bvE5c+14x8O8J4Vv1lxdGWG9acY+Et3u
dnd65roqqmZjJ9Zh0DJwfKQAQugFHqA
AAAAAgATmTECKvWlKir1pWrP2x1Y+m
qutfu1+aVirrX7tfmlazls1AVyrFpjgATmV
vnb2AlVmZnkAEIL1+Pq53Revx9XO69Pr
Dv0ekADRqAOK3aXnX7S7bP7dPhDZrZ/
bp8lbKqACcyv87ewAzKPnb2Al
VAAXQCj1AAAAAAAABFXrSlRV60r05Y6
3Vqqraqca0+v5StVTb/AH7TzT7rWclmM
+raZ9W1qKqNs+raZ9W1qA2z6tpn1bWo
DbPq2mfVtagFcZ+TO+uRrh07Gw69PrD
v0ekNcOnYYdOxsNGrXDp2GHTsbDit2l5
1+0sxVMRERP0hnPq2tRVRtn1bTPq2tQ
G2fVtM+ra1AbZ9W0z6trUBtn1bTPq2tQ
HoAFHqAAAAAAAADWYiZ0NmFq8qX4
YzY2KW8dYtPPPuu1JeOsWvnn3XZYhG
AgxAAGIAAxAAGIAE5lMfQAfKfacyAlVx
AAGIAAxAAGIAAxAAGlegAZugAAAAAA
AAYZYWrypbgUl46xa+efddqS8dYtfPPuu
zRgIAAAAAAAAAAAAAAAAAAAAAH//
2Q==

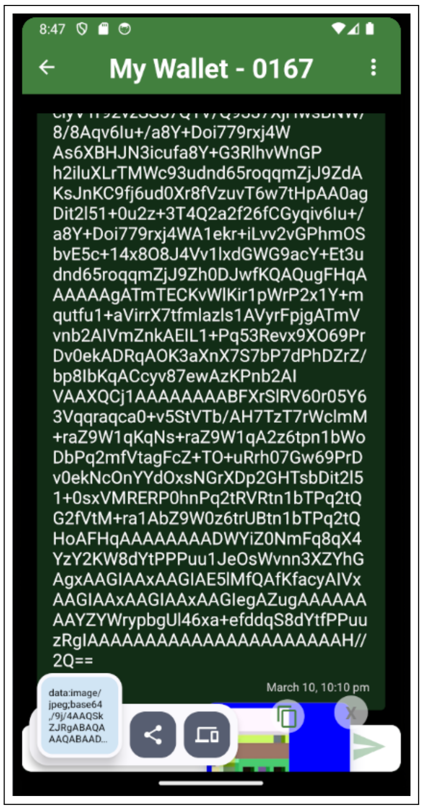
(b) Aries Bifold - Selecting the image



VAAXQCj1AAAAAAAABFXrSlRV60r05Y6
3Vqqraqca0+v5StVTb/AH7TzT7rWclmM
+raZ9W1qKqNs+raZ9W1qA2z6tpn1bWo
DbPq2mfVtagFcZ+TO+uRrh07Gw69PrD
v0ekNcOnYYdOxsNGrXDp2GHTsbDit2l5
1+0sxVMRERP0hnPq2tRVRtn1bTPq2tQ
G2fVtM+ra1AbZ9W0z6trUBtn1bTPq2tQ
HoAFHqAAAAAAAADWYiZ0NmFq8qX4
YzY2KW8dYtPPPuu1JeOsWvnn3XZYhG
AgxAAGIAAxAAGIAE5lMfQAfKfacyAlVx
AAGIAAxAAGIAAxAAGlegAZugAAAAAA
AAYZYWrypbgUl46xa+efddqS8dYtfPPuu
zRgIAAAAAAAAAAAAAAAAAAAAAH//
2Q==

(c) Aries Bifold - Copying it to the clipboard

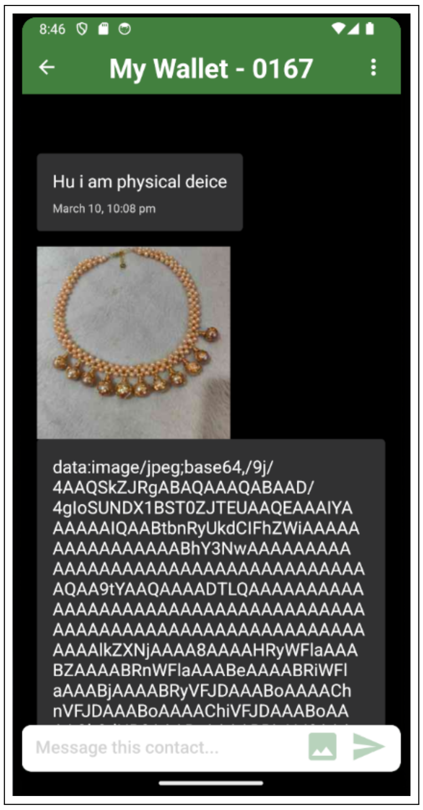
Figure 5.11: Images transfer between users - 1



8/8Aqv6lu+/a8Y+Doi779rxj4W
As6XBHJN3icufa8Y+G3RlhvWnGP
h2iluXLrTMWc93udnd65roqqmZjJ9ZdA
KsJnKC9fj6ud0Xr8fVzuvT6w7tHpAA0ag
Dit2l51+0u2z+3T4Q2a2f26fCGyqiv6lu+/
a8Y+Doi779rxj4WA1ekr+iLvv2vGPhmOS
bvE5c+14x8O8J4Vv1lxdGWG9acY+Et3u
dnd65roqqmZjJ9Zh0DJwfKQAQugFHqA
AAAAAgATmTECKvWlKir1pWrP2x1Y+m
qutfu1+aVirrX7tfmlazls1AVyrFpjgATmV
vnb2AIVmZnkAElL1+Pq53Revx9XO69Pr
Dv0ekADRqAOK3aXnX7S7bP7dPhDZrZ/
bp8lbKqACcyv87ewAzKPnb2AI
VAAXQCj1AAAAAAAABFXrSlRV60r05Y6
3Vqqraqca0+v5StVTb/AH7TzT7rWclmM
+raZ9W1qKqNs+raZ9W1qA2z6tpn1bWo
DbPq2mfVtagFcZ+TO+uRrh07Gw69PrD
v0ekNcOnYYdOxsNGrXDp2GHTsbDit2l5
1+0sxVMRERP0hnPq2tRVRtn1bTPq2tQ
G2fVtM+ra1AbZ9W0z6trUBtn1bTPq2tQ
HoAFHqAAAAAAAADWYiZ0NmFq8qX4
YzY2KW8dYtPPPuu1JeOsWvnn3XZYhG
AgxAAGIAAxAAGIAE5lMfQAfKfacyAlVx
AAGIAAxAAGIAAxAAGlegAZugAAAAAA
AAYZYWrypbgUl46xa+efddqS8dYtfPPuu
zRgIAAAAAAAAAAAAAAAAAAAAAH//
2Q==

March 10, 10:10 pm

(a) Aries Bifold - Pasting the image URL

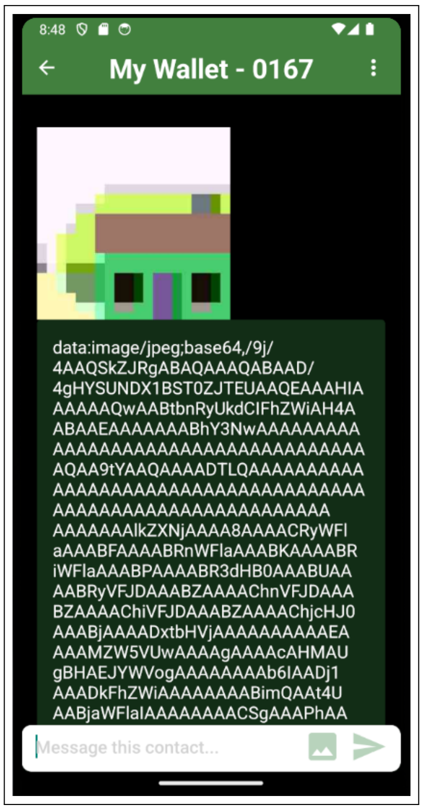


Hu i am physical deice

March 10, 10:08 pm

data:image/jpeg;base64,/9j/
4AAQSkZJRgABAQAAAQABAAD/
4gIoSUNDX1BST0ZJTEUAAQEAAAIYA
AAAAAIQAABtbnRyUkdCIFhZWiAAAAA
AAAAAAAAAAABhY3NwAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAA9tYAAQAAAADTLQAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAlkZXNjAAAA8AAAAHRyWFlaAAA
BZAAAABRnWFlaAAABeAAAABRiWFl
aAAABjAAAABRyVFJDAAABoAAAACh
nVFJDAAABoAAAAChiVFJDAAABoAA

(b) Aries Bifold - Image-1 Send



data:image/jpeg;base64,/9j/
4AAQSkZJRgABAQAAAQABAAD/
4gHYSUNDX1BST0ZJTEUAAQEAAAHIA
AAAAAQwAABtbnRyUkdCIFhZWiAH4A
ABAAEAAAAAAABhY3NwAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAA9tYAAQAAAADTLQAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAlkZXNjAAAA8AAAACRyWFl
aAAABFAAAABRnWFlaAAABKAAAABR
iWFlaAAABPAAAABR3dHB0AAABUAA
AABRyVFJDAAABZAAAAChnVFJDAAA
BZAAAAChiVFJDAAABZAAAAChjcHJ0
AAABjAAAADxtbHVjAAAAAAAAAAEA
AAAMZW5VUwAAAAgAAAAcAHMAU
gBHAEJYWVogAAAAAAAAb6IAADj1
AAADkFhZWiAAAAAAAABimQAAt4U
AABjaWFlaIAAAAAAAACSgAAAPhAA

(c) Aries Bifold - Image-2 Send

Figure 5.12: Images transfer between users - 2

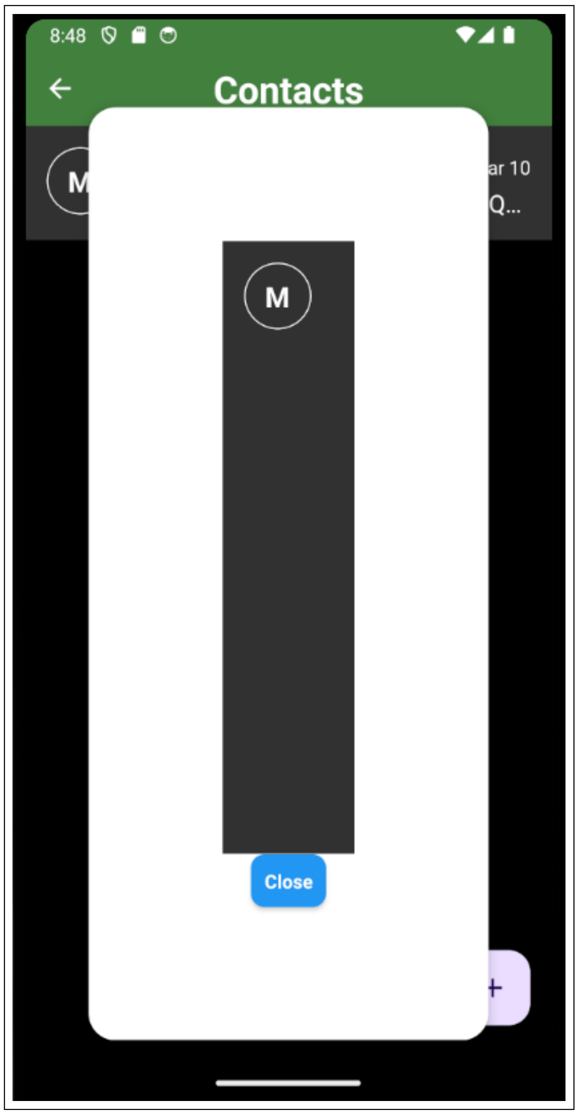


Figure 5.13: Selecting Users UI

the state using react state is causing several problems. The only possible solution remains to use a separate state to handle but there is another issue as it will make the application too heavy to handle and it may crash.
- For these kinds of diificulties, the implementation part of group messaging chat feature is not fully completed till now.

Chapter 6

Discussion

6.1 Analysing Requirements

The proposed system has several functional and security requirements to guide the development process while meeting the end product's expectations. Let's analyze those requirements:

Functional requirements:

- The system is built on the top of Hyperledger Aries, which is a robust framework providing tools and libraries designed to develop and support Self-sovereign Identity (SSI) systems. So, incorporating the Hyperledger Aries has integrated the SSI mechanisms into the chat's services fulfilling the F1.
- In order to manage and present credentials to verifiers, managing DIDs, establishing connections and secure communication, providing authentication and authorization, and providing user interface for the ease of users, a mobile agent or wallet is a must. We use an application called Aries Mobile Agent React Native aka Aries Bifold Wallet to provide all the facilities mentioned above, thus ensuring the F2.
- The system satisfies F3 by sending DIDs while establishing connection. DIDs are used within the VC framework to ensure that credentials are not dependent on any centralized registry or authority.
- The system allows users to send plaintext messages, send or download images to other users which are encrypted using a series of Symmetric Key Cryptography and Public-Key Cryptography methods by mobile agents of the end users before transferring the messages to targeted users to ensure the F4, F7 and F8.
- All the messages and images are saved in the local storage of the user and can be retrieved from there fulfilling F5 and F6.

Security requirements:

- To provide safe, passwordless authentication, our system authenticates and gives access to its users by verifying them using Verifiable Credentials based on SSI. It also provides authorization as it mitigates chat or information modification by an unauthorized person satisfying S1 and S2.

- Aries Bifold Wallet provided by Hyperledger Aries provides Verifiable Credentials with digital signature to make it tamper-evident, thus making it safe, secure and handling S3.
- Our system follows a protocol that is made by using the existing DIDcomm protocol, it ensures that data or files which are being sent by the users are encrypted from the sender's end and only being decrypted from the receiver's end. Thus providing integrity and confidentiality of the data and images ensuring S4.
- The SSI mechanisms provided by the Hyperledger Aries and Aries Bifold Wallet help the proposed system against attacks like DoS attack, Replay attack. As our system is built on the top of these frameworks and wallets, our system is also capable of dealing and countering these attacks satisfying S5 and S6.

6.2 Advantages

The advantages of the End-to-End Encrypted Peer-to-Peer Chat System with SSI are:

- The system is decentralized, so it is more secure than existing centralized systems.
- The system uses SSI, making the system more user-centric and giving the user the authority of managing identity.
- Unlike the existing decentralized systems, our proposed system is capable of sending and receiving files.
- The proposed system has well defined protocol for connection establishment, messaging, file sharing, and group messaging.
- The messages and credentials are saved in the local storage, making it harder to attack or steal.

6.3 Limitations

The limitations of the End-to-End Encrypted Peer-to-Peer Chat System with SSI are:

- Till now, the system can share image files, but it is still incapable of sending many other types of files like PDF, Word or Excel.
- The protocol and design of the group messaging feature is completed, but it is yet to be implemented and integrated in the system.
- In the protocol, we have assumed that group members are all connected with each other for group messaging.

- As the system is yet to be fully implemented, we haven't done any usability or performance testing even though it has followed good design principles.

6.4 Future Works

- To enhance the file sharing feature of the system in the future, the system should be capable of sending different types of files.
- To make the communications and discussion more enthusiastic, in future, a fully functional group messaging feature could be implemented.
- As we haven't tasted the image sharing, it can be tested in future for its speed, scalability, interoperability, robustness etc.

Chapter 7

Conclusion

In conclusion, we have presented a decentralized messaging system built upon Arise Bifold. The Use of HyperLedger Arise makes it a SSI using decentralized system which supports passwordless entry and VC framework. The system supports image sharing and also have a protocol for establishing group messaging feature. The motivation was to improve the existing decentralized systems so that they can compare with regular centralized messaging systems and give more security to its users. There is a high level architecture for how messages will be sent. We have given threat modeling, requirement analysis and security analysis. We hope this will help researchers and developers in future to create a better decentralized messaging system with all the features we can find in centralized messaging systems and even more.

Bibliography

[1] M. K. Ibrahem and T. A. M. Ali, "Secure messaging system using zkp," 2013.

[2] A. Shostack, *Threat Modeling, Designing for Security.* 2014.

[3] J. Graham, *New cfo survey: More than 80 percent of firms say they've been hacked*, Duke Today, Jun. 2015. [Online]. Available: https://today.duke.edu/2015/06/cfohacking#:~:text=These%20are%20some%20of%20the,or%20make%20public%20important%20data.

[4] N. Unger, S. Dechand, J. Bonneau, *et al.*, "Sok: Secure messaging," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 232–249. DOI: 10.1109/SP.2015.22.

[5] A. Ali and A. Sagheer, "Design of secure chatting application with end to end encryption for android platform," *Iraqi Journal for Computers and Informatics (IJCI)*, vol. 43, p. 6, Jun. 2017. DOI: 10.25195/2017/4315.

[6] N. Zupan, K. Zhang, and H.-A. Jacobsen, "Hyperpubsub: A decentralized, permissioned, publish/subscribe service using blockchains: Demo," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, ser. Middleware '17, Las Vegas, Nevada: Association for Computing Machinery, 2017, pp. 15–16, ISBN: 9781450352017. DOI: 10.1145/3155016.3155018. [Online]. Available: https://doi.org/10.1145/3155016.3155018.

[7] M. Abdulaziz, D. Çulha, and A. Yazici, "A decentralized application for secure messaging in a trustless environment," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 2018, pp. 1–5. DOI: 10.1109/IBIGDELFT.2018.8625362.

[8] M. Chase, A. Deshpande, E. Ghosh, and H. Malvai, "Seemless: Secure end-to-end encrypted messaging with less trust," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ACM, London, United Kingdom, Nov. 2019.

[9] K. Khacef and G. Pujolle, "Secure peer-to-peer communication based on blockchain," in *Web, Artificial Intelligence and Network Applications*, L. Barolli, M. Takizawa, F. Xhafa, and T. Enokido, Eds., Cham: Springer International Publishing, 2019, pp. 662–672, ISBN: 978-3-030-15035-8.

[10] F. Schillinger and C. Schindelhauer, "End-to-end encryption schemes for online social networks," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, G. Wang, J. Feng, M. Z. A. Bhuiyan, and R. Lu, Eds., Cham: Springer International Publishing, 2019, pp. 133–146.

[11] U. P. Ellewala, W. Amarasena, H. S. Lakmali, L. Senanayaka, and A. Senarathne, "Secure messaging platform based on blockchain," in *2020 2nd International Conference on Advancements in Computing (ICAC)*, vol. 1, 2020, pp. 317–322. DOI: 10.1109/ICAC51239.2020.9357306.

[12] L. Olivo, *7 social media sites and their data breaches, They say your data is protected, but how have they measured up?* Human ID, Oct. 2020. [Online]. Available: https://human-id.org/blog/biggest_social_media_breach_history/.

[13] D. Reed, *Self-Sovereign Identity, Decentralized digital identity and verifiable credentials*. Manning Publications, 2021.

[14] L. Shi, Z. Guo, and M. Xu, "Bitmessage plus: A blockchain-based communication protocol with high practicality," *IEEE Access*, vol. 9, pp. 21 618–21 626, 2021. DOI: 10.1109/ACCESS.2021.3056135.

[15] R. Singh, A. N. S. Chauhan, and H. Tewari, *Blockchain-enabled end-to-end encryption for instant messaging applications*, 2021. arXiv: 2104.08494 `[cs.CR]`.

[16] A. H. Enge, A. Satybaldy, and M. Nowostawski, *An architectural framework for enabling secure decentralized p2p messaging using didcomm and bluetooth low energy*, 2022.

[17] D. Halder, S. Bhushan, G. Shreya, and P. Kumar, *Fybrrchat: A distributed chat application for secure p2p messaging*, 2022. arXiv: 2207.02487 `[cs.SI]`.

[18] C.-E. Bogos, R. Mocanu, and E. Simion, *A security analysis comparison between signal, whatsapp and telegram*, Cryptology ePrint Archive, Paper 2023/071, https://eprint.iacr.org/2023/071, 2023. [Online]. Available: https://eprint.iacr.org/2023/071.