


People's Democratic Republic of Algeria

Ministry of High Education and Scientific Research

University of Akli Mohand Oulhadj Bouira

Faculty of Science and Applied Science

Computer Science department

Master 2 Thesis

In Computer science

Speciality: GSI

Theme

Secured Messenger Application

Supervisor:

- Lyes Badis
- Dr.Noureddine LASLAA (UBKU Qatar)

Realised by:

- Ladjal Khalil
- Hanniche Yazid

2020/2021

Acknowledgments

First and foremost, We would like to praise **Allah** the Almighty, the Most Gracious, and the Most Merciful for His blessing given to us during our study and in completing this project. May Allah's blessing goes to His **final Prophet Muhammad (peace be upon him)**, his family and his companions.

Our gratitude and appreciation also goes to everyone who supported us, **family** and **Friends**. Thank you for your enthusiasm and encouragements. It is a humbling experience to acknowledge those who have, mostly out of kindness, helped along our journey, even if it was with simple kind words or a small advice.

We owe our deepest gratitude to our supervisor **Mr.Lyes Badis** for his frequent help and support, for his time, patient and guidance. We are also thankful to **Dr.Noureddine LASLAA of (UBKU Qatar)** and his recommendations, it would not have been possible to complete this project without their help, guiding us in an area of study and domain new to us. We definitely gained and learned a lot from this journey thanks to them.

Dedication

I dedicate this humble work to my family and many friends. A special feeling of gratitude to my loving parents, Especially my mother whose prayers and words of encouragement and push for tenacity ring in my ears. To my beloved sisters and brother "Ishak".
Also a special dedication to my grandparents especially my grandfather "Belkhir" may Allah give him an easy and pleasant journey and shower blessings on his grave. I will be ever grateful for that i had him, and am sorry that he has not lived to see me graduate.

I also devote this work to my friends who have supported and was there to listen to me, they know who they are. Especially my best friend Brahim.

To my friend and partner in this project ***Yazid****.*

Khalil.

Dedication

I am honored to present my dedication to the dearest people who supported me during my journey, especially my brother and sister who have been with me. Also with the encouragement of my uncle "Laazize" through his valuable advice. As well devote my work to my deceased parents and uncle "Ghani" peace be upon them whom I hope would be proud of me and what I achieved, without forgetting my loyal friends who stood there by my side. I give my gratitude to my partner and friend Khalil for accomplishing this work together.

Yazid.

Contents

Table of contents **i**

Table of figures **iv**

List of abbreviations **vii**

General introduction **1**

1 State of the art **4**

1.1 Introduction: 4

1.2 Definition: 4

1.3 A Brief History: 5

1.3.1 A brief history about SMS and instant messaging 5

1.3.2 A brief history about email service: 7

1.4 Categories of messaging apps and services: 7

1.5 Security in messaging apps and services: 8

1.5.1 Instant Messaging Encryption Tools & Methods: 8

1.5.2 End to end encryption: 10

1.5.3 Open Source Protocols XMPP and SIP: 11

1.6 Some inconveniences and security problems in IM Apps: 17

1.6.1 End-to-End Encryption in Messaging Services and National Security—Case of WhatsApp Messenger[14]: 17

1.6.2 Link preview in IM Apps and it's risks[18]: 18

1.6.3 Few privacy conflicts in some famous IM Apps: 20

1.7 Conclusion: 21

2 Decentralized applications to secure users data 22

Decentralized applications to secure users data 22

2.1 Introduction . . . 22
2.2 Decentralized Systems . . . 22
2.3 Data security and decentralized systems . . . 23
2.4 BlockChain technology . . . 23
2.4.1 Definition of blockchain . . . 24
2.4.2 The structure of blockchain . . . 25
2.4.3 Features of blockchain: . . . 26
2.4.4 How blockchain works: . . . 27
2.4.5 Types of consensus algorithms . . . 28
2.4.6 Cryptocurrencies and Cryptography . . . 29
2.5 Peer to Peer (P2P) architecture . . . 33
2.6 Applications based on blockchain and p2p . . . 34
2.7 Conclusion . . . 41

3 Conception 42

3.1 Introduction . . . 42
3.2 The application's global architecture: . . . 42
3.2.1 Contact adding . . . 43
3.2.2 Messages exchanging . . . 43
3.2.3 Mutual Key exchanging . . . 45
3.3 Application features : . . . 47
3.4 Data structures and access to messages: . . . 47
3.5 Modeling: . . . 47
3.5.1 Functional view : . . . 48
3.5.2 Dynamic view : . . . 49
3.6 Conclusion . . . 52

4 Implementation and realization 53

4.1 Introduction . . . 53
4.2 Work environments . . . 53
4.3 Libraries and frameworks: . . . 56

4.3.1 React.js 56
4.3.2 Node.js: 56
4.3.3 Jquery.js 57
4.3.4 Stacks.js: 57
4.3.5 Different Scenarios and graphic presentation of the stacks App . . . 59
4.4 Scenarios and presentation of our simulation 65
4.4.1 First step: Sign in 65
4.4.2 Second step: Login 66
4.4.3 Third step: contact request 66
4.4.4 Forth step: messages exchange 68
4.5 Conclusion 69

General conclusion **70**

List of Figures

1.1 Proposed instant messaging secure model by (Yusof and Abidin, 2011) . . 9
1.2 XMPP archy & Communication Model 13
1.3 Basic call flow of SIP 16
1.4 Example of link preview on discord (an IM app) 19
1.5 Security and link preview comparison in different IM apps 20

2.1 Centralized and decentralized systems 23
2.2 The structure of bitcoin blockchain network:dailyblockchain.github.io . . . 25
2.3 Blockchaine structure 25
2.4 How blockchain work [22] 27
2.5 How blockchain work[20] 29
2.6 Hash function work 31
2.7 Digital signature 32
2.8 Architecture of DOSN 35
2.9 How friends request work in Tawki [32] 36
2.10 Stacks architecture 39
2.11 Interaction between Gaia hub and client 40

3.1 How users add each other 44
3.2 Figure of message exchange between the sender and receiver 45
3.3 Figure on how users exchange their mutual key 46
3.4 Data structures and access to messages 47
3.5 Sequence diagram of the registration process 49
3.6 Sequence diagram of the first contact between two users 50

3.7 Sequence diagram of mutual key exchange between two contacts 50
3.8 Sequence diagram of message exchange between two users 51

4.1 SignIn/Login function 57
4.2 SignOut function 58
4.3 Writing data to the cloud storage as a Json file 58
4.4 Reading data from cloud storage as a Json file 59
4.5 Front Page/Login Page 59
4.6 Sign-In/sign-Up options 60
4.7 User log-in with an existing ID 60
4.8 Sign-Up step 1 (Create a Password) 61
4.9 Sign-Up step 2 (ADD email) 61
4.10 Sign-Up step 3 (Save recovery key) 62
4.11 Editing/completing a stacks profile 63
4.12 App interface as a Logged In user 64
4.13 SignIn page 65
4.14 login page 66
4.15 Search for other contact 67
4.16 Search result and send request button 67
4.17 Messages exchange 68

List of abbreviations

API	Application Programming Interface.
IM	Instant Messaging
SSL	Secure Sockets Layer
TLS	Transport Layer Security
OTR	Off-the-Record
E2EE	End to end encryption
CA	certificate authority
XMPP	Extensible Messaging and Presence Protocol
SIP	Session Initiation Protocol
SDP	Session Description Protocol
UA	User Agent
UAC	User A gent Client
P2P	Peer to Peer
DHT	Distributed hashed Table
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
BNS	Blockchain Naming System
UML	Unified Modeling Language
POW	Proof of work.
POS	Proof of stake.
DS	digital signature.
DOSN	decentralized online social network.
BNS	Blockchain Naming System.
dApp	decentralized application.

General introduction

Work context:

In this humble work, we discus and study a secured Instant Messaging Application, because we know that users always value their privacy and security. We will make a research on how to improve the security flaws of existing IM Apps. Thus, we will propose a new secured Messaging Application.

We are aware that there are all kinds of IM Apps spread across all platforms and a lot of these apps claim that they value the security/privacy of their users. So, why another Instant messaging App ?

Messaging Apps security varies from one to another, Some are highly secured as they state while the others don't provide as much security. The users of course prefer to keep safe, secure their data, their private information and conversations. However, even the Messaging apps with the high security measures and standards had their downs and have failed their users in a way or another so even when the conversations are well encrypted there is some other kind of threat to the users such as public information provided to the app, list of contacts, locations, Phone numbers ...etc and by putting this data into a centralized entity. Lots of user data will be exposed to third parties and other threats, which drives users into losing trust, knowing that their data could be traded behind their backs or left unprotected. That's where our research comes in.

Expected objectives:

We will try to make a Secured Messaging App with the following properties:
-No personal data should be provided by the user to any party.
-Every thing will be encrypted including conversations and user information.
-A decentralized app that uses peer to peer technology.
In the end if we establish all these properties we should be able maximize the security and make anyone feel very safe starting an online conversation.

Organization of the report:

Our report is organized in 4 chapters:

First chapter(State of the art):

In this chapter we'll bring a general idea about this subject:
-Some definitions.
-A brief history of IM Apps.
-Categories of IM Apps.
-Security in IM Apps.
-Security problems and inconveniences.

Chapter 2: Alternative to centralized system:

In this chapter we will discuss a solution to the security problems mentioned in chapter1, And represent the technologies behind this alternative and this discussion will include:
-Decentralized systems.
-Data security and decentralized systems.
-Blockchain technology
-Peer to peer system
-Some applications based on blockchain and peer to peer

Chapter 3: Conception

In this chapter, we propose our project analysis and conception with several diagrams and illustrations. We'll talk about:

-The application's global architecture.

-Messages exchanging.

-Application features.

-Data structures and access to messages.

-Modeling.

-Dynamic view.

Chapter 4: Implementation and realization

In this chapter, we will present the different tools and the different languages used for our implementation choices.

-Work environments.

-Libraries and frameworks.

-Scenarios and presentation of the Stacks progress.

-Scenarios and presentation of our simulation.

Chapter 1

State of the art

1.1 Introduction:

We live in a world where we rely pretty much on technology in almost everything we do in our lives. We are spoiled with lots of luxuries that make life much easier for us, and one of the luxuries technology is providing us with is the variant messaging/ Instant messaging applications and services that exist today. These messaging apps can be seamlessly blended into our everyday workflow and we are using them on daily basis to stay connected simply because we as humans need to socialize, communicate and interact. You can communicate in real-time(instantly) and exchange messages that can be text, voice, or even attachments such as photos and videos. You can reach out or answer an inquiry from any device, close the app, and return to whatever you were doing. Then, whenever the other party is available to respond, they will, and then you can take your turn. So, what is a messaging application?

1.2 Definition:

A messaging app is any type of software that serves under the purpose of allowing users to send and receive information using internet connection, the exchange can be instant. People tend to use messaging apps that rely on internet especially mobile apps which are the most installed apps, as a replacement for paid chatting services such as Short Message Service (SMS) or Multimedia Messaging Service (MMS).
Messaging apps can transmit or receive a much wider range of data types than (SMS) or

(MMS). In addition to voice calls, video calls and text, messaging-app users can send and receive files, images, audio, location data, emojis and (in some cases) documents.

Messaging apps were primarily designed for private communication between individuals or small groups, but are increasingly being used in new ways, including:

* **Broadcast or bulk messaging**: The capacity to send messages or other content to a large number of people.
* **Encryption** End-to-end encryption means that the content of a message can be viewed only by the people sending and receiving messages. It cannot be decrypted and read by the company itself.
* **Bots or "chatbots"** A piece of code that performs automated functions within an app (often in natural language like a human hence "chatbot"), such as replying to users' questions with short pieces of information. [28]

Some of the common features of the apps include:

* Ability to make voice and video calls.
* They are free.
* Instant messaging.
* Photo sharing.
* There is no restriction to the number or the length of messages you can send or receive.

1.3 A Brief History:

1.3.1 A brief history about SMS and instant messaging

Modern instant messaging and SMS both began their march to prominence in the early and mid-1990s. The difference between the two is subtle: SMS (the acronym for "short service message") allows mobile phone users to send each other text messages without an internet connection, whereas instant messaging enables similar functionality via the web.

In 1985, Commodore launched Quantum Link (or "Q-Link"), an online service for Commodore 64 and 128 that enabled multi-person chat, file sharing, electronic email, games, and news via modem connection. Quantum Link changed its name to America Online (AOL) in 1991, and by the mid-90s was the leading US internet service provider and portal to the web.

The company launched AOL Instant Messenger (AIM) in 1997 and purchased competitor ICQ in 1998 to consolidate its primacy over instant messaging. Along with a few competitors, it also pioneered chat robots like StudyBuddy and SmarterChild that providedà information and played games with users.

In 2006, AIM controlled 52 percent of the instant messaging market, but it struggled to monetize and went into rapid decline in the face of competition from services like Google Talk, Yahoo! Chat, MSN Messenger, and Skype. The growing popularity of BlackBerry Messenger in the late 2000s also pointed to a bright future for mobile messaging.

By the time mobile chat apps like WhatsApp and Kik arrived in 2009, SMS was king. Mobile texting became a key mode of global, personal communication, earning billions of dollars for telecommunications companies.

But time and technology did not prove kind to telephone-service companies. As smartphones began to proliferate, messaging apps were an increasingly accessible solution to a simple problem: SMS is expensive in most countries, so why not text or call much more cheaply, or for free, via the mobile web?

By September 2015, WhatsApp and Messenger had 1.6 billion active, monthly accounts combined—outpacing Facebook's 1.49 billion active, monthly accounts. The company also launched an open API for Messenger, encouraging developers and publishers to build custom apps for the ecosystem. It simultaneously began beta testing Businesses on Messenger, a tool facilitating e-commerce and customer support. In August 2015, Instagram, another of Facebook's acquisitions, launched enhanced one-to-one messaging—encouraging users to share photos and videos from the news feed within private

chats.[9]

1.3.2 A brief history about email service:

1971: The creation of electronic mail

In 1971, Ray Tomlinson invented and developed the first electronic messaging system. This began with a file transfer program called CPYNET. Tomlinson then used code from CPYNET when adapting a time-share message program. (Aptly called SNDMSG.)

Tomlinson's new message program enabled ARPANET users to send messages to different computers on the network.

Tomlinson has since forgotten the content of the first electronic message ever sent.

Along with the invention of electronic mail is the invention of email addresses. (And the use of the @ symbol.) So, right from the beginning, we've been addressing emails as 'username' @ 'name of computer'.

May of 1978 saw the first mass email ever sent. Gary Thuerk, a marketer, sent an email to 400 addresses on ARPANET, of the 2600 that existed.It was an advertisement for the Digital Equipment Corporation. This marks the first spam message in the history of email. And, like most spam, it was negatively received.

1989: World wide web and lotus notes

The creation of the internet as we know it, naturally, features in the history of email. In 1989, Sir Tim Berners Lee invented the world wide web. And email would greatly benefit from it.

At the end of 1989, Lotus Development Corporation released Lotus Notes email software. 35,000 copies of the software sold in the first year of its release. IBM later bought the company in 1995, and Lotus Notes became IBM Notes. [2]

1.4 Categories of messaging apps and services:

1. **Email:** An e-mail (electronic mail) is a method of exchanging messages ("mail") between people using electronic devices.The exchanged messages may contain text, files, images, or other attachments sent through a network to a specified individual

or group of individuals, each individual must have a valid email address in order for them to communicate properly via emails.

2. **Traditional Cellular based services (SMS & MMS)**

3. **Modern Instant messaging (chatting):** instant messaging is a technology that allows users to interact and trade short messages in real time using internet. instant messaging is mostly refereed to as chatting.
   and here are some of the most used apps in this category: **Facebook's messenger and whatsapp, Snapchat, WeChat** (a chinese app), **Line, Discord, Signal Private Messenger**...etc.

4. **Audio/video Chat**: We can find video/audio chatting options in almost every app now days but some applications are specialised more than the others in this type of communication, these kind of apps usually target a specific audience like: business or education, and we can take as an example: **Google Meet, Zoom,** which are both widely used especially lately during the pandemic of **Covid**-19. Some other apps in this category in the other hand they have the interest of a massive audience of online gamers simply because they make communication much easier and online gaming much fun. **ex: Discord, Team-Speak,**...

1.5 Security in messaging apps and services:

Instant messaging applications has taken the place as the most installed mobile apps, the desktop and web versions are also widely used. Thus, it makes them vulnerable to many security related dangers.

Over the time some methods or techniques were developed and used to protect people's privacy and their data. So let's highlight some of these methods:

1.5.1 Instant Messaging Encryption Tools & Methods:

The increase of security breaches of some messaging apps such as Facebook's messenger and Whats-app makes encryption algorithms and methods a must. Encryption ensures every user can communicate securely, it helps keeps the content of messages safe and

prevents any unwanted third parties from sniffing what may be sensitive and private. This helps companies and businesses from losing their money and keeps individuals safe.

According to the latest research by PricewaterhouseCoopers ("Global Information Security Survey: 2015 Results by Industry," 2015), average of global cyber attacks is increased to 48% with 42.8 million detected incidents. According to a recent report by security company Postini("Instant messaging targeted for malicious worm attack," 2006), instant messaging threats are increased by 1700% in 2005 and 90% of these threats are highly destructive worms. According to Sanchez J. (Sanchez, 2014), most of these highly-popular instant messaging applications including WhatsApp have some known vulnerabilities.[4]

In 2011 a secure model for instant messaging was proposed by Yusof and Abidin by adding additional hash algorithm to encrypt the path between transceiver and routing modules as shown in the following Fig:

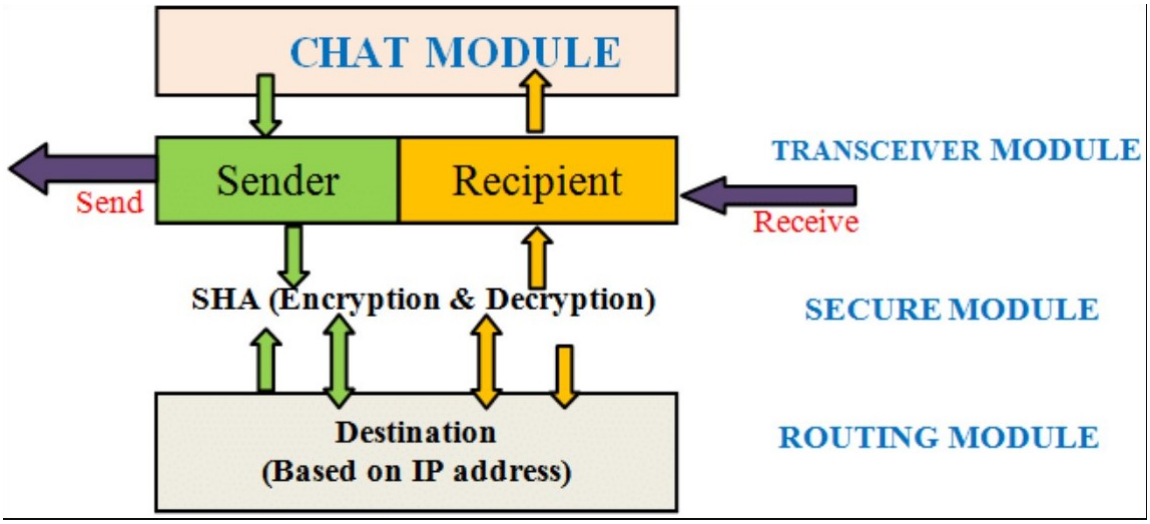

Figure 1.1: Proposed instant messaging secure model by (Yusof and Abidin, 2011)

Bodriagov and Buchegger (Bodriagov and Buchegger, 2011) propose a new approach that combines asymmetric and symmetric cryptography methods to encrypt the communication channel. The proposed method does not have sufficient efficiency and functionality for peer-to-peer social networks[4].

Here some commonly used instant messaging encryption tools and methods:

1) Secure Sockets Layer (SSL):

Secure Sockets Layer (SSL) is the most widely deployed and used security protocol that provides privacy, authentication, and integrity to internet communications. SSL eventually evolved into Transport Layer Security (TLS).

A website that implements SSL/TLS has "HTTPS" in its URL instead of "HTTP."

How does SSL/TLS work? it has two main components:

1- Handshake protocol: Handshake is an authentication process between the SSL client and the server to confirm that both are really who they claim to be.

2- Record layer formats application protocol:

It provides messages with a header for each message and a hash which signs the data to provide integrity, confirming that the data is not tampered with before reaching its target.

2) Off-the-Record (OTR) Messaging:

OTR is a cryptography protocol that provides privacy for instant messaging conversations. this protocol uses a combination of AES symmetric-key algorithm with 128 bits key length, the Diffie–Hellman key exchange with 1536 bits group size, and the SHA-1 hash function.

3) Private Browsing:

Private browsing is a feature in a browser when used a temporary session will be created, no data or information will be saved such as browsing history or cookies private browsing also will block all installed third-party extensions by default.
Authors recommend to use web-based instant messaging applications in private browsing mode.

1.5.2 End to end encryption:

End-to-end encryption (E2EE) is a method of secure communication that prevents third-parties from accessing data while it's transferred from one end system or device to another.

In E2EE, the data is encrypted on the sender's system or device and only the recipient is able to decrypt it. Nobody in between, be they an Internet service provider, application

service provider or hacker, can read it or tamper with it.

The cryptographic keys used to encrypt and decrypt the messages are stored exclusively on the endpoints, a trick made possible through the use of public key encryption. Although the key exchange in this scenario is considered unbreakable using known algorithms and currently obtainable computing power, there are at least two potential weaknesses that exist outside of the mathematics. First, each endpoint must obtain the public key of the other endpoint, but a would-be attacker who could provide one or both endpoints with the attacker's public key could execute a man-in-the-middle attack. Additionally, all bets are off if either endpoint has been compromised such that the attacker can see messages before and after they have been encrypted or decrypted.

The generally employed method for ensuring that a public key is in fact the legitimate key created by the intended recipient is to embed the public key in a certificate that has been digitally signed by a well-recognized certificate authority (CA). Because the CA's public key is widely distributed and generally known, its veracity can be counted on, and a certificate signed by that public key can be presumed authentic. Since the certificate associates the recipient's name and public key, the CA would presumably not sign a certificate that associated a different public key with the same name.

The first widely used E2EE messaging software was Pretty Good Privacy, which secured email and stored files, as well as securing digital signatures. Text messaging applications frequently utilize end-to-end encryption, including Jabber, TextSecure and Apple's iMessage[8].

1.5.3 Open Source Protocols XMPP and SIP:

XMPP and SIP are a widely used open source protocols for real time communication. XMPP and SIP both fulfill the key functionalities of instant messaging which are: presence, session management and transfer of data.

1) XMPP (Extensible Messaging and Presence Protocol):

Characteristics:

* **XMPP uses XML** to exchage data between the Client and server.

* **Decentralization:**There is no main server as other messaging apps such as Whatsapp or Messenger.

* **XMPP via HTTP:** Originally XMPP used TCP(Transmission Control Protocol) as it's transport protocol. However, the XMPP community developed an HTTP transport protocol for clients, it can go through firewalls easily and can work when TCP ports are blocked.

* **Extensibility:** this protocol can and has been extended with new features.

* **Presence:** Helps clients see their contacts online status.

* **Channel Encryption:** XMPP servers can be isolated (company intranet as example), and secure authentication and point-to-point encryption have been built into the core XMPP specifications, as well as $Off-the-RecordMessaging$ (OTR) is an extension of XMPP enabling encryption of messages and data. It has since been replaced by a better extension, multi-end-to-multi-end encryption end-to-end encryption between users. This gives a higher level of security, by encrypting all data from the source client and decrypting again at the target client; the server operator cannot decrypt the data they are forwarding.

* **Architecture:** XMPP architecture is similar to the email architecture.

XMPP core Methods list: (1) Setup of XML stream (2) Channel Encryption (3) Authentication (4) Error Handling (5) Communication (6) Presence (7) Request-Response interaction (8) Security [1].

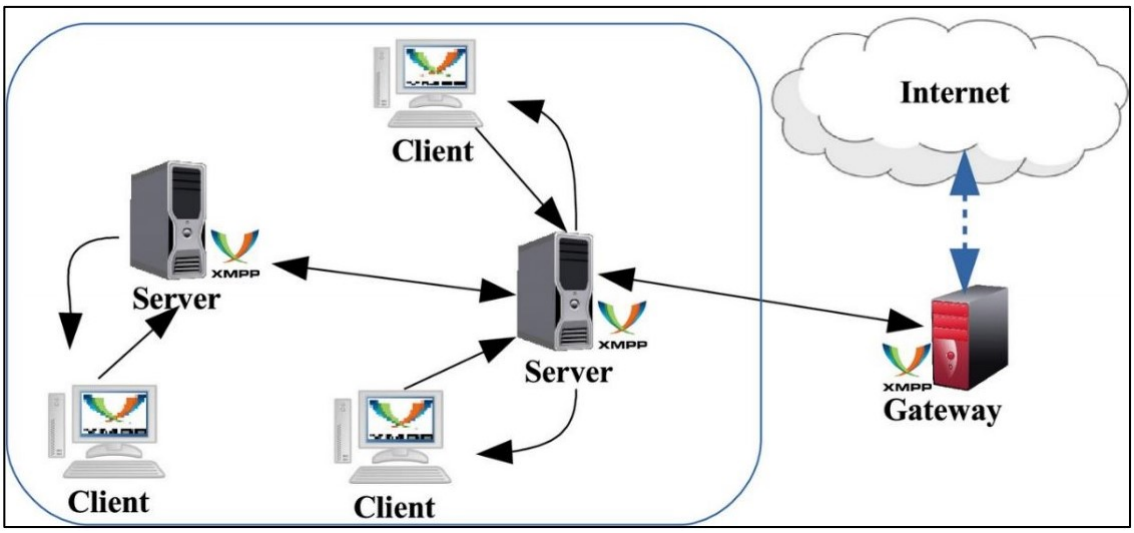

Figure 1.2: XMPP archy & Communication Model

Security and encryption in XMPP:

XMPP's security level is pretty high, also it's developer and community have always been working on making it more secure with reliable encryptions and security. And i will quote from an Encryption Manifesto regarding the XMPP protocol:

"We, as operators of federated services and developers of software programs that use the XMPP standard for instant messaging and real-time communication, commit to establishing ubiquitous encryption over our network on May 19, 2014.
...XMPP technologies were first released on January 4, 1999, byJeremie Miller. Since then, channel encryption using Secure Sockets Layer (SSL) and Transport Layer Security (TLS) has been optional on the Jabber/XMPP network. Out of respect for the users of our software and services, we believe it is time to make such encryption mandatory."

"This commitment to encrypted connections is only the first step toward more secure communication using XMPP, and does not obviate the need for technologies supporting end-to-end encryption (such as Off-the-Record Messaging or OTR), strong authentication, channel binding, secure DNS, server identity checking, and secure service delegation. Although we have worked to implement and deploy such technologies and will continue to do so, we believe that encrypting the traffic on the XMPP network is a necessary precondition to offering further security improvements. "[29]

2) Session Initiation Protocol (SIP):

The Session Initiation Protocol (SIP) is used for signaling and controlling multimedia communication sessions and it is a communication protocol. The most common use of SIP in instant Messaging over Internet Protocol(IP),as well as in Internet telephony for audio and video calls. SIP works with other application layer protocols that identify and carry the session media. SIP uses a Session Description Protocol (SDP) for Media description and identification. For the transmission of audio and video streams SIP commonly used in the Real-time Transport Protocol (RTP) / Secure Real-time Transport Protocol (SRTP). For secure transmissions SIP uses Transport layer Security (TLS). SIP applications can run on TCP, UDP, or other networking protocols.
SIP takes the help of SDP which describes a session and RTP used for delivering voice and video over IP network.SIP can be used for unicast or multicast sessions. Other applications are instant messaging, file transfer, video conferencing, streaming multimedia distribution and online games.[1]

There are two types of User Agents (UA), it's operated [1]

User Agent Client (UAC): Generates requests and send to servers.
User Agent Server (UAS): Gets requests, processes requests and generate responses.

Server:

There are several types of server–[1]

Proxy Server: Proxy server acts as both server and client. The main role of proxy server id routing means that the request is sent to another entity which is nearest to the targeted user. Proxy interprets and write a specific part of the request before it is forwarding. There are two types of proxy server 1.Stateless proxy server 2.stateful proxy server.
Registrar Server: registrar server accepts a REGISTER request and places the information in location server .Location service links with more than one IP address to SIP URI.one or more user agent can register at the same URI, with all registered users receive calls to the URI. It is logical element and end point of the REGISTER request.

It's located with SIP Proxies.

Redirect Server: Redirect server generate a response in 3xx (redirection) form. It directs the client to contact a set of URIs. It's allows proxy the direct SIP session invitation to external domains. **Location Server:** Store the addresses of registered user agent in Registrar.

SIP functionalities:

Function	Description
User location and registration	End points (telephones) notify SIP proxies of their location; SIP determines which end points will participate in a call.
User availability	SIP is used by end points to determine whether they will "answer" a call.
User capabilities	SIP is used by end points to negotiate media capabilities, such as agreeing on a mutually supported voice codec.
Session setup	SIP tells the end point that its phone should be "ringing;" SIP is used to agree on session attributes used by the calling and called party.
Session management	SIP is used to transfer calls, terminate calls, and change call parameters in mid-session (such as adding a 3-way conference).

Request method of SIP [1]:

* INVITE: Used to establish a media session between user agents.
* ACK: It used to acknowledge the reliable message exchange for INVITEs.
* BYE:Terminates a connection between two users
* CANCEL: Terminates a pending request.
* OPTIONS: Requests information about a server's capabilities.
* REGISTER: Registers current location of Users.

∗ INFO:Used for mid-session signaling.

Basic call flow of SIP

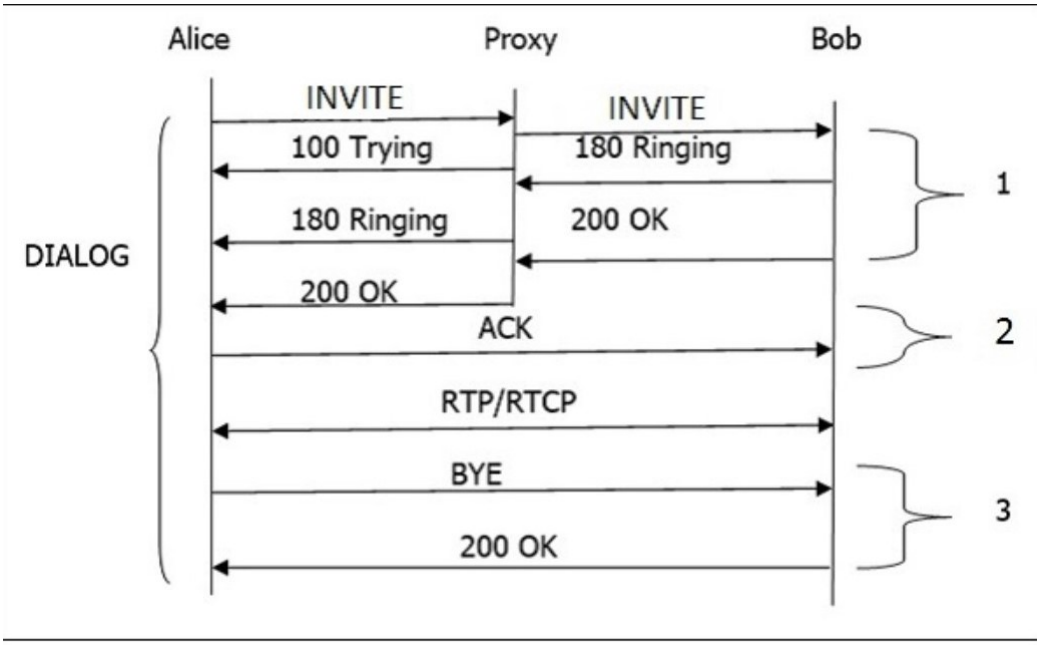

Figure 1.3: Basic call flow of SIP

SIP security and encryption:

‘

SIP would be vulnerable to attack if it is not protected, that's why SIP protocol uses "secure SIP" which is a security mechanism was used for HTTP sessions and it was defined by SIP RFC 3261 for sending SIP messages over a Transport Layer Security-encrypted channel.

In a Secure SIP session, the SIP user agent client contacts the SIP proxy server requesting a TLS session. This SIP proxy server responds with a public certificate and the SIP user agent then validates the certificate. Next, the SIP user agent and the SIP proxy server exchange session keys to encrypt or decrypt data for a given session. From this point, the SIP proxy server contacts the next hop and similarly negotiates a TLS session, ensuring that SIP over TLS is used end-to-end[35].

1.6 Some inconveniences and security problems in IM Apps:

As we saw in the last section, developers, companies and the community try to add encryption and security measures to IM apps so that they are secured. Never the less, there are still some inconveniences and problems related to the security in IM Apps such as:

1.6.1 End-to-End Encryption in Messaging Services and National Security—Case of WhatsApp Messenger[14]:

One of the methods used in the security field of IM Apps is End to end encryption (E2EE). E2EE is a necessity if one wants a secured Messaging app and it has been very useful, Never the less there are some inconveniences and conflicts that are brought to the table because of it, such as the Case of WhatsApp Messenger with the government of the USA and it's national security.

According to a Javelin Strategy and Research Report in 2012, one in every four people who have a breach in their online data becomes a victim of identity theft as a result of that. End-to-End Encryption provides an effective way to prevent these attacks, and if it had been implemented properly by Yahoo Inc., it could have prevented large-scale attacks like the one Yahoo suffered in 2016 and 2013, where almost 500 million, and more than 1 billion accounts were respectively compromised[14].

Governments, and secret services on the other hand are asking encrypted messaging services like WhatsApp to allow them access to their users' data. There is growing risk to public safety as organized crime, terrorists, and child pornographers are drawn to the use of E2EE apps like WhatsApp that are technically impossible to access. According to, a defendant in a serious felony case told another individual on a recorded jailhouse call that "end-to-end encryption is another gift from God". Criminal defendants across the United States are benefiting from E2EE while the safety of all other American communities is in peril. However, providing a backdoor would not only be a breach of privacy to WhatsApp users, but creating a way for the authorities to read encrypted messages would also make the system vulnerable to cyber-attacks from criminals and other hackers[14].

By implementing backdoors, it also means that the service is not truly end-to-end

encrypted in the first place[14].

A report shows a design feature in WhatsApp messaging service that could potentially allow some encrypted messages to be read by unintended recipients. WhatsApp allows undelivered messages to be stored in their servers for up to 30 days before they are deleted. The report also noted that the WhatsApp implementation of the security protocol used in its E2EE allows for the generation of secret keys between communicating parties in a WhatsApp conversation.However, new keys get generated when a user gets a new phone or reinstalls WhatsApp. Messages for the user which may have been waiting to be delivered while the user was offline are then re-encrypted and resent automatically by the sender, without the sender having had an opportunity to verify whether the recipient is the person intended to receive the message. A sender is notified after the event if the sender has opted to turn on a notification in settings, but not otherwise. "This re-encryption and resending of previously undelivered messages could potentially allow a third party to intercept and read a user's undelivered messages in a situation where, for example, they had stolen a user's sim card. When the third party put the stolen sim card in another phone, they could then theoretically collect any messages that had not yet been delivered to the user in question. WhatsApp Inc. has since responded to this claim, saying that the feature in question is a design trade off, meant to prevent users from losing their messages if they switch phones or reinstall the app.

1.6.2 Link preview in IM Apps and it's risks[18]:

Link previews are a ubiquitous feature found in just about every chat and messaging app, and with good reason. They make online conversations easier by providing images and text associated with the file that's being linked.

Unfortunately, they can also leak our sensitive data, consume our limited bandwidth, drain our batteries, and, in one case, expose links in chats that are supposed to be end-to-end encrypted. Among the worst offenders, according to research published on Monday, were messengers from Facebook, Instagram, LinkedIn, and Line. More about that shortly. First a brief discussion of previews.

When a sender includes a link in a message, the app will display the conversation along with text (usually a headline) and images that accompany the link. It usually looks

something like this:

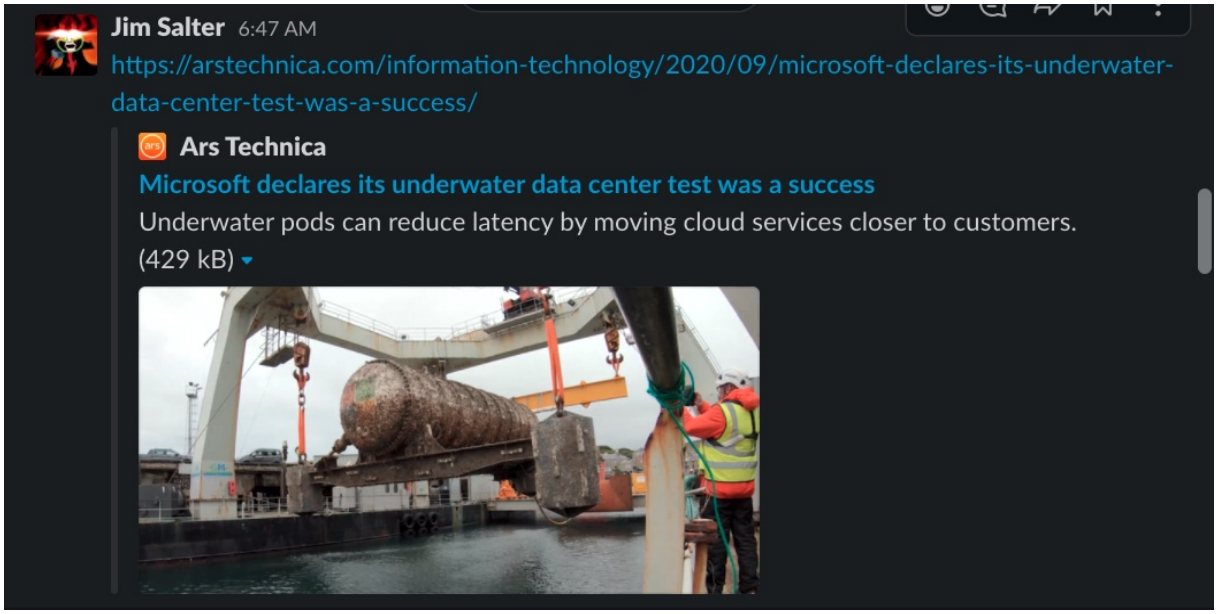

Figure 1.4: Example of link preview on discord (an IM app)

For this to happen, the app itself or a proxy designated by the app has to visit the link, open the file there, and survey what's in it. This can open users to attacks. The most severe are those that can download malware. Other forms of malice might be forcing an app to download files so big they cause the app to crash, drain batteries, or consume limited amounts of bandwidth. And in the event the link leads to private materials—say, a tax return posted to a private OneDrive or DropBox account—the app server has an opportunity to view and store it indefinitely[18].

Link previews in chat apps can cause serious privacy problems if not done properly.There are several cases of apps with vulnerabilities such as: leaking IP addresses, exposing links sent in end-to-end encrypted chats, and unnecessarily downloading gigabytes of data quietly in the background.

	End-to-End Encrypted	Unauthorized Copies of Private Information	Getting Servers to Download Large Amounts of Data	Crashing Apps and Draining the Battery	IP Exposure	Leaking Encrypted Links	Running Potentially Malicious Code on Link Preview Servers
Discord	No	Up to 15 MB	No	No	No	No	No
Facebook Messenger	No	Unlimited!	**Yes**	No	No	No	**Yes**
Google Hangouts	No	Up to 20 MB	No	No	No	No	No
iMessage	Yes	No copies	No	No	No	No	No
Instagram	No	Unlimited!	**Yes**	No	No	No	**Yes**
LINE	Yes	Up to 20 MB	No	No	**Yes (Fixed!)**	**Yes**	No
LinkedIn	No	Up to 50 MB	**Yes (Through JavaScript)**	No	No	No	**Yes**
Reddit	No	No copies	No	**Yes (Fixed!)**	**Yes (Fixed!)**	No	No
Signal	Yes	No copies	No	No	No	No	No
Slack	No	Up to 50 MB	No	No	No	No	No
Threema	Yes	No copies	No	No	No	No	No
TikTok	No	No copies	No	No	No	No	No
Twitter	No	Up to 25 MB	No	No	No	No	No
Viber	Yes	No copies	No	**Yes**	No	No	No
WeChat	No	No copies	No	No	No	No	No
WhatsApp	Yes	No copies	No	No	No	No	No
Zoom	No	Up to 30 MB	No	No	No	No	No

Figure 1.5: Security and link preview comparison in different IM apps

1.6.3 Few privacy conflicts in some famous IM Apps:

E2EE in Viber:

In late April, security researchers discovered that the popular mobile messaging app, Viber, sent video and images without encrypting them (the equivalent of scrambling the message so only the intended receiver can descramble and thus read) first. To make matters worse, the app also stored the messages online on a publicly available server, making it possible for private photos and messages to be accessed by anyone with enough determination and know how. The company has now solved the issue, but the potential damage done is still hard to determine[24].

Snapchat:

Snapchat, the popular messaging platform that promises to erase each message after it is viewed, recently settled with the Federal Trade Commission regarding a similar infraction. The reason: its messages didn't actually disappear as often as promised[24].

1.7 Conclusion:

After identifying/knowing what is the nature of instant messaging Apps, their history, how they work and how they can be secured as we mentioned some security measures and methods. We also went through some security problems, inconveniences and potential threats which lead us to the conclusion that the users data and privacy are threatened while using these IM apps. That's why a solution or an alternative must be found to grantee people's privacy and safety without denying them the luxury of such services.

Chapter 2

Decentralized applications to secure users data

2.1 Introduction

In the previous chapter, we saw some inconveniences about the IM apps centralized systems and some security issues that modern applications and companies are facing. For instance the WhatsApp snooping scandal that over 1400 people globally spied upon using a surveillance technology developed by Israel-based NSO Group, or Google is suspected to collect personal data without users permissions. Researchers suggested a new approach or infrastructure to deal with data privacy and try to eliminate the risks that come with centralized infrastructures.

The decentralized infrastructure suggested being the best alternative to maintain good security for the user's data.

2.2 Decentralized Systems

In a decentralized system, we don't have a central entity to receive or respond to requests. They use multiple central nodes, each of which usually stores a copy of the resources users can access.

The decentralized systems can be vulnerable, however, its architecture is more tolerant to failure, because when one central node fails, the other can continue to provide data to

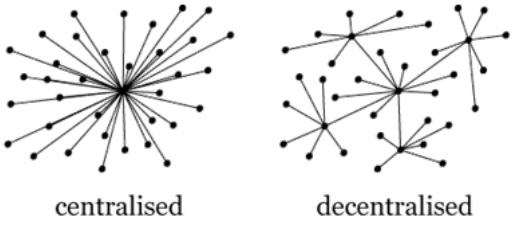

Figure 2.1: Centralized and decentralized systems

the users.

2.3 Data security and decentralized systems

Out from the centralized system to the decentralized, some legitimate questions appear about the privacy, data security, and where to store the data. Several researches responded to these questions, one of them was decentralized data storage or decentralized cloud storage, also End-To-End encryption is here to secure communication between two ends(host) without letting the third parties access the data.

2.4 BlockChain technology

Dealing first with the main technology to build a real or concrete decentralized application. It's one of the most trending technologies due to the cryptocurrency Bitcoin.

History of blockchain

Blockchain was Created by NAKAMOTO SATOCHI the concept of decentralized of blockchain and first successful decentralized digital currency emerged with the Bitcoin whitepaper in late 2008 [3]. The first successful cryptocurrency and blockchain application was released in 2009, combining concepts of public key cryptography with a consensus algorithm known as proof-of-work. The main goal of this invention is "we don't need bank anymore".

According to Satoshi, the problem is *trust*, he said that to participate in the world economy, trust in the third party like banks and governments is an unnecessary burden.

The author of the first article wanted to remain anonymous and therefore nobody knows Satoshi Nakamoto to this day. A few months later, an open source program implementing the new protocol, starting with the first block GENESIS with 50 coin in it. Anyone can install this open source program and use it, and become the part of P2P Bitcoin network.

These days the value of Bitcoin still growing, and the technology of Blockchain evolve to find new kind of applications far from the field of finance.

2.4.1 Definition of blockchain

Blockchain are often defined as an immutable distributed digital ledger, which is secured using cryptography, replicated among the peer nodes within the P2P network, and uses a consensus mechanism to agree upon the transaction, whereas control is decentralized [3]. We can say that a blockchain is a data structure that makes it possible to create a digital ledger of data to share it among a network of independent parties[22]. There are different type of blockchain.

- **Public blockchains:** A large distributed networks that are run through a native token such as Bitcoin [22] and ethereum. In this kind of blockchain anyone can join the network, write and read. The data in a public blockchain are secured as it impossible to modify once they have been validated on the blockchain.
- **Private blockchain:** Private blockchain tends to be smaller and utilizes a token. Their membership is limited and controlled [22]. In this type of blockchain, There are one or more entities that control the network, and this leads to resilience on third parties. Hyperledger Fabric is a perfect example of private blockchain.
- **Permissioned blockchain:** Permissioned blockchain controls roles that individuals can play within the network. they're still large and distributed systems that use a native token. Their core code may not be open source [22].

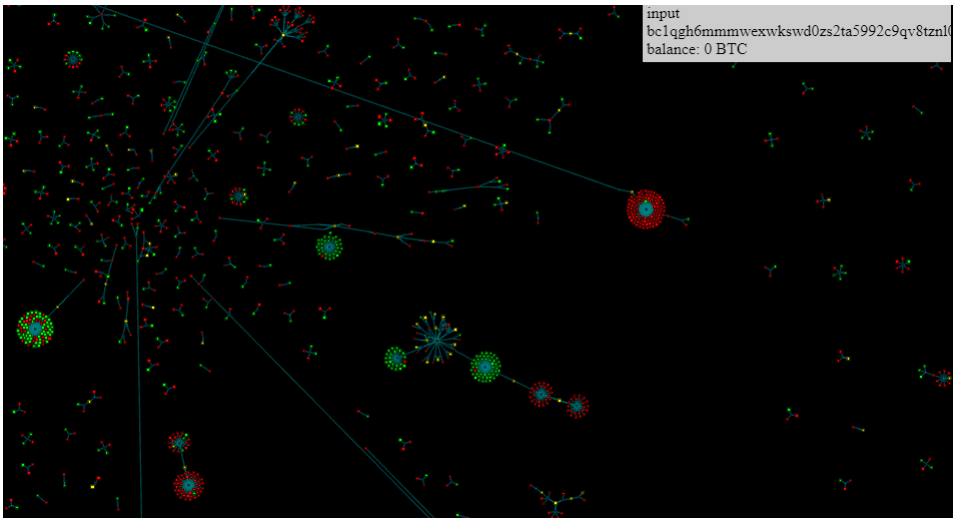

Figure 2.2: The structure of bitcoin blockchain network:dailyblockchain.github.io

2.4.2 The structure of blockchain

A blockchain consists of a collection of data (a block) linked to the previous block. How are they linked? A block contains data, and each block refers to the block preceding it, thus they are linked just as a chain link would be connected to the chain link before it [13].

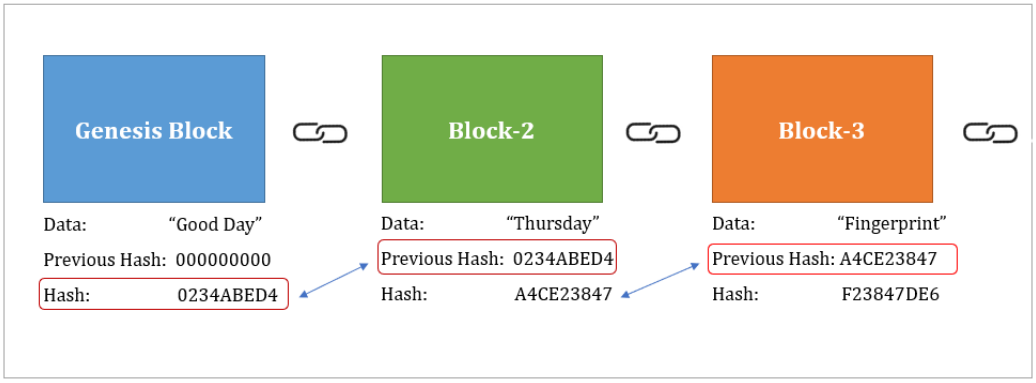

Figure 2.3: Blockchaine structure

The blockchain are composed of three core parts:

- Block : Contains a list of transaction recorded into a ledger over a given period. The size, period, and triggering event blocks is different for every blockchain.

Not all blockchains are recording and securing a record of the movement of their cryptocurrency as their primary objective. But all blockchains do record the movement of their cryptocurrency or token. Think of the transaction as simply being the recording of data. Assigning a value to the block (such as in a financial transaction) is used to interpret what that data means[22].

- **Chain:**

A hash that links one block to another, mathematically "chaining" them together. This is one of the most difficult concepts in blockchain to comprehend. It's also the magic that glues blockchains together and allows them to create mathematical trust. The hash in blockchain is created from the data that was in the previous block.
The hash is a fingerprint of this data and locks blocks in order and time.
Although blockchains are a relatively new innovation, hashing is not. Hashing was invented over 30 years ago. This old innovation is being used because it creates a one-way function that cannot be decrypted. A hashing function creates a mathematical algorithm that maps data of any size to a bit string of a fixed size. A bit string is usually 32 characters long, which then represents the data that was hashed. The Secure Hash Algorithm (SHA) is one of some cryptographic hash functions used in blockchains. SHA-256 is a common algorithm that generates an almost-unique, fixed-size 256-bit (32-byte) hash. For practical purposes, think of a hash as a digital fingerprint of data that is used to lock it in place within the blockchain.

- **Network:** It's network composed of nodes . Each node contains a complete record of all transactions, these nodes have different locations all over the world.

2.4.3 Features of blockchain:

In this section we are going to present the main features of blockchain:

- **Secure:** It is really impossible for anyone to tamper with transactions or ledger records present in the blockchain, which makes it more secure, so it is seen as a reliable source of information [21]

- **Distributed:** All records are stored in each node of the network, if one node goes

down, it will not affect the network or the other nodes, because they are globally distributed all the nodes.

- **Automated operations:** Operations are fully automated through software. Private companies are not needed to handle operations, which is why there is no mediation required to carry out the transactions, and trust is assured, so people can carry out their own transactions [21].
- **Open source:** Blockchain is an open source technology. All the operations are carried out by the open source community [21].
- **Flexibility**: One of the assets of this technology, it can be programmed, using the basic programming concepts and semantics, which make it very flexible.

2.4.4 How blockchain works:

In the blockchain world, consensus is the process of developing an agreement among a group of commonly mistrusting shareholders. These are the full nodes on the network. The full nodes are validating transactions that are entered into the network to be recorded as part of the ledger.

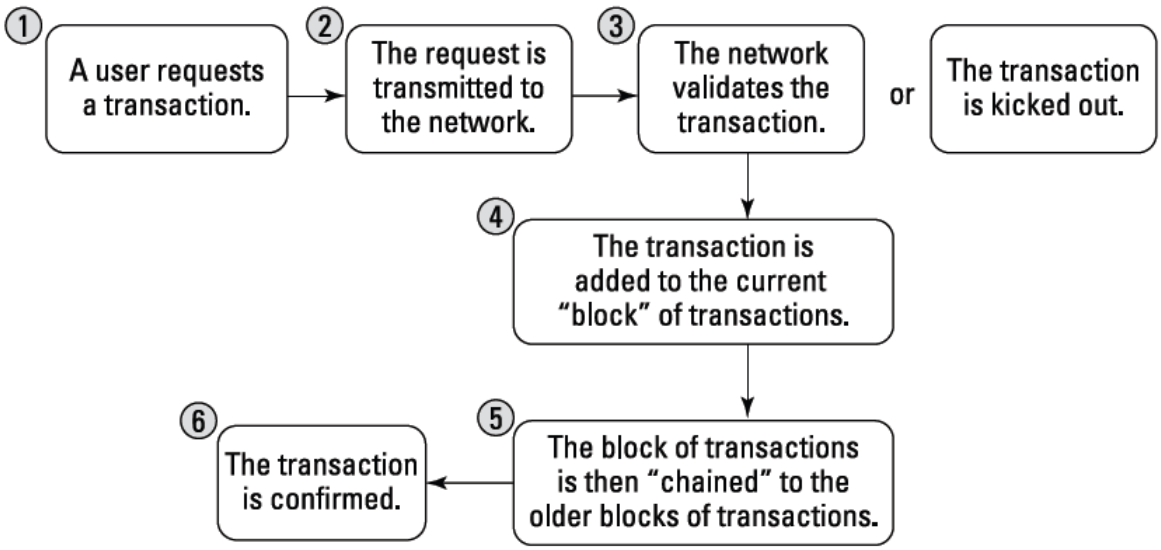

Figure 2.4: How blockchain work [22]

this how it work:

- one user requests a transaction.
- the request is diffused in p2p networks(blockchain network) composed with many nodes.
- the network validate the transaction.
- once the validation of the transaction is completed, the transaction is added to the current block of transactions.
- the new block is then chained to the older blocks.
- Finally the transaction will be completed successfully.

Each blockchain has its own algorithms for creating agreement within its network on the entries being added. There are many different models for creating consensus, because each blockchain is creating different kinds of entries. Some blockchains are trading value, others are storing data, and others are securing systems and contracts [22].

2.4.5 Types of consensus algorithms

we will present the most used algorithms or protocols in major blockchain platform

- **Proof of Work:** PoW uses a hash function to create conditions that allow a single/individual participant to announce their conclusions about the submitted information, which are then verified by all the other system participants. The hash function has a parameter that ensures false information will fail to compute in an acceptable way to prevent any false conclusions [22].
- **Proof of Stake:**Proof of Stake (POS) uses a randomized process to figure out who gets a chance to produce the next block. Blockchain users can lock up their tokens for a certain time to become a validator. After becoming a validator, users can be able to produce blocks. Validators can also be selected based on the design of blockchain. Generally, the user who owns the biggest stake or coins for the longest period of time has better odds of creating a new block [19]

- **Proof of Elapsed Time:** Proof of Elapsed Time process randomly and fairly decides the producer of a new block based on the time they have spent waiting. For this purpose, the mechanism provides a random wait time for each user and the user whose wait time finishes the earliest will produce a new block. This consensus mechanism only works if the system can verify that no users can run multiple nodes and the wait time is truly random [19].

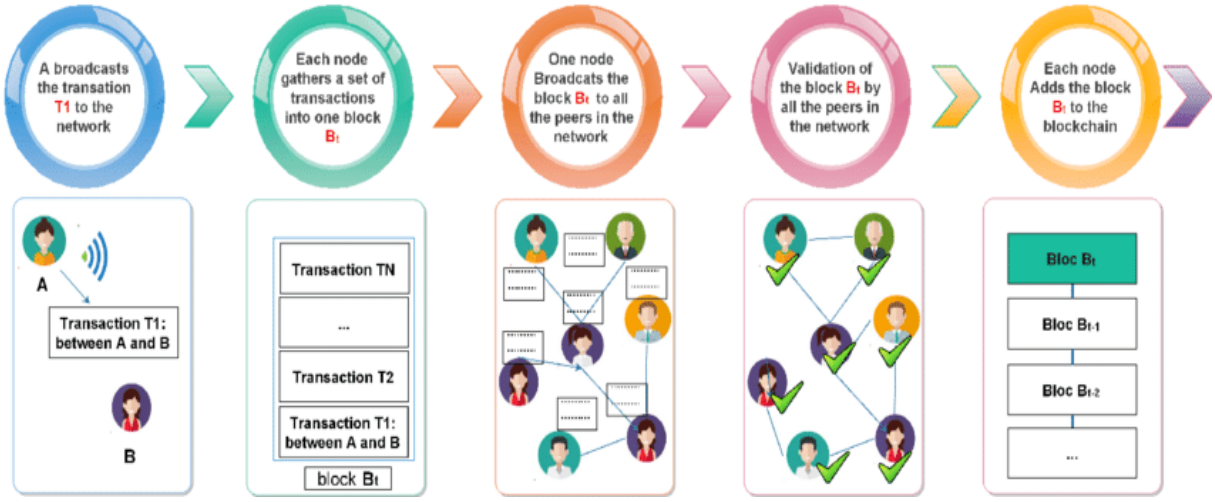

Figure 2.5: How blockchain work[20]

Smart Contract

Smart contracts are at the core of blockchain technology. These are programs written in Turing-complete programming languages and are capable of self-verifying and self-executing agreements that can function autonomously or without any external intervention.

A smart contract is a piece of code that is stored in the blockchain, is triggered by the blockchain transactions, and which then reads or writes data from/to that blockchain's database.[21].

2.4.6 Cryptocurrencies and Cryptography

Blockchain technology is the fundamental pillar of cryptocurrency. In this section we will talk about cryptography, cryptocurrency, and how these two work together. But first

let us talk about the global currencies like dollar, euro and dinar. these currencies have some characteristics like:

- they owned by different governments.
- they are controlled by a central entity like banks.
- They include various security properties to prevent counterfeiting and cheating[21].

cryptocurrency

A cryptocurrency is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology a distributed ledger enforced by a disparate network of computers. The most noticeable feature of cryptocurrencies is that they are generally not released by any central authority, rendering them theoretically free from the governments interference or manipulation [17].

The main features of cryptocurrency are as follows:

- It is a digital asset used as a medium of exchange [21].
- All the transactions are secured by cryptography.
- Alternative for our traditional currencies.
- Cryptocurrency is digital money, while the underlying technology that enables the moving of digital coins or assets between individuals is called blockchain [21].

Cryptography :

The main functions of cryptography used in cyptocurrencies and blockchain are: The hash function and Digital signature. let examine these two:

Hash function: A cryptographic hash function is an algorithm that takes an arbitrary amount of data input and produces a fixed-size output of enciphered text called a hash value, or just "hash." That enciphered text can then be stored instead of the password itself, and later used to verify the user. [33].

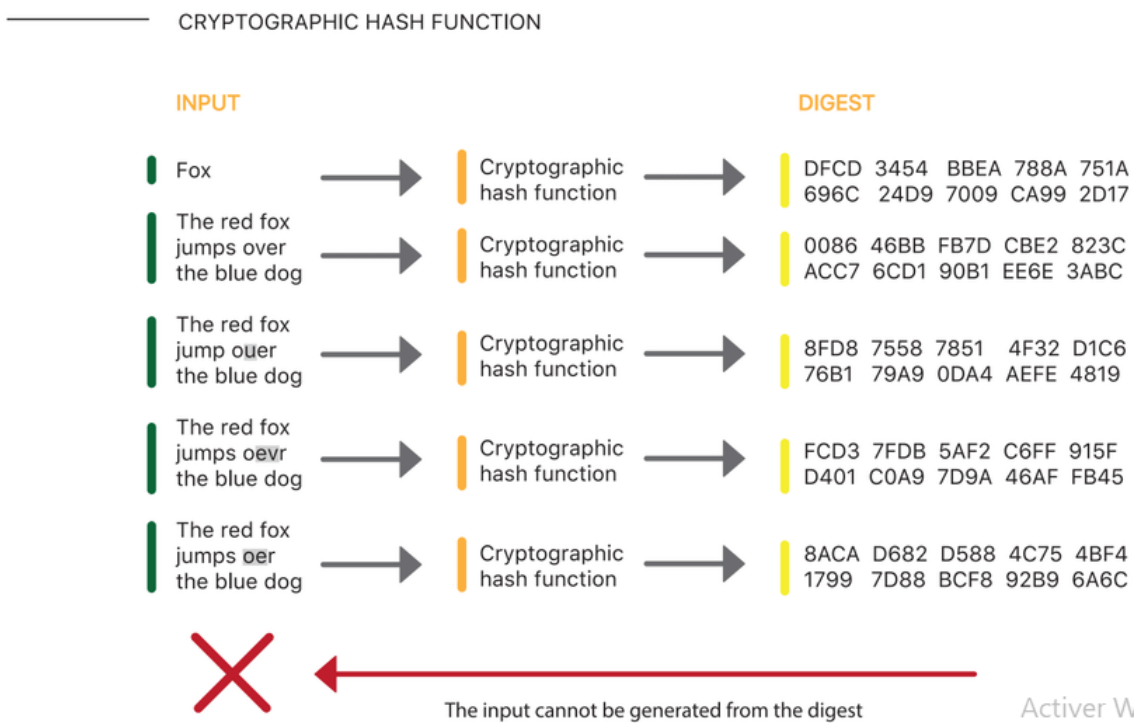

Figure 2.6: Hash function work

We gave "Fox", a string, as an input and we applied a hash function to it. It gives a specific fixed hash outcome, which is also known as a digest hash sum.let's say we pass another string"**The red fox runs over the blue dog**" and apply a hash function to it. It gives us a specific hash sum.This time we changed the letter **V** from the word **'over'** by a letter **'u'** and apply the hash function, we see that the output changed.

Digital signature: A digital signature (DS) is the detail of an electronic document that is used to identify the person transmitting data. DS makes it possible to ascertain the non-distortion status of information in a document once signed and to check whether or not the signature belongs to the key certificate holder. The detail's value is a result of the cryptographic transformation of information through the use of a private and public key [11].

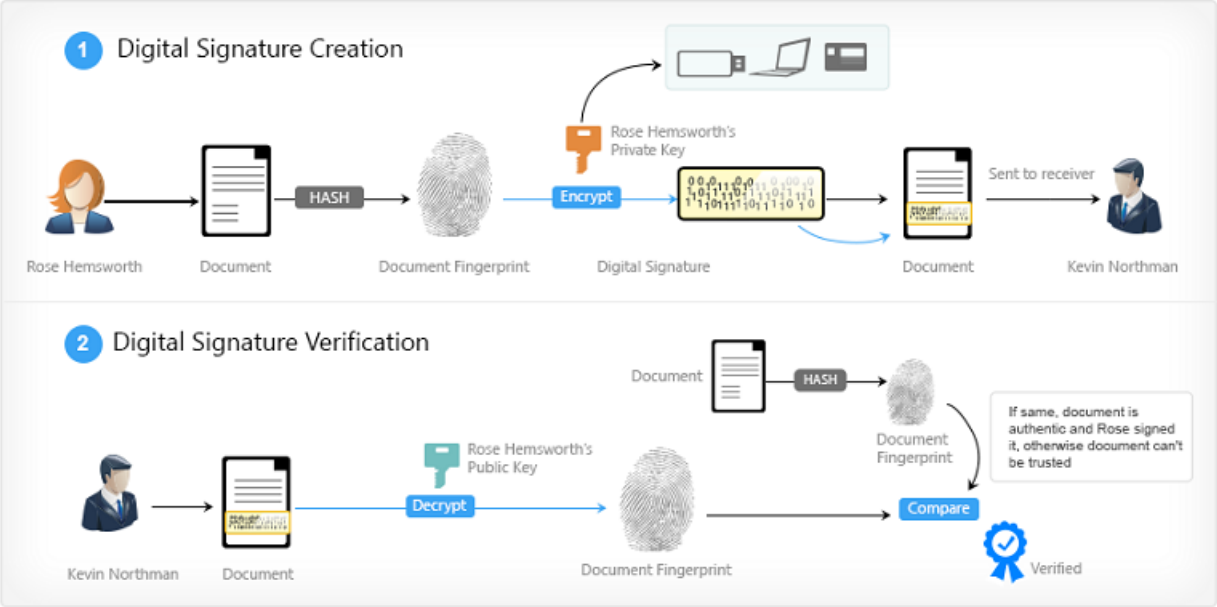

Figure 2.7: Digital signature

Bitcoin vs Ethereum

The most trending cryptocurrencies in the present day, bitcoin and Ethereum take the podium. It's important to understand that there are two categories of digital currencies: the cryptocurrencies (like Bitcoin, Dogecoin, and Litecoin...), and tokens (like Ethereum, Filecoin, storj ...)

Bitcoin emerged in 2009, it was the first cryptocurrency back then. Ethereum is a far more recently developed in 2015.

In the time between Bitcoin and Ethereum's release, a lot of cryptocurrencies emerge, however they were limited to trying to improve on aspects of Bitcoin's performance like improving security and increase the speed of transactions.

	bitcoin	Etherreum
inventor	Satoshi Nakamoto	Vitalik Buterin,Joseph Lubin, Gavin Wood
category	currency	Token
small unit	1 Satoshi = 0.00000001 BTC	1 Wei = 0.000000000000000001 eth
Time to produce new token	every 10 minutes	every 20 seconds
utility	Used for the purchase of goods and and services, as well as to storing value	Used to create dApps (decentralized applications) on the Ethereum blockchain.
price	1BTC= 37,960.00 $	1 ether=2,780.83$
Objective	Bitcoin is currency created to with gold and other currencies	Ethereum is a token capable of to facilitate smart contracts (for example, a lawyer's contract)

TABLE 1 – comparison between Bitcoin and Ethereum

2.5 Peer to Peer (P2P) architecture

A P2P system is a distributed system in which the peers (nodes) are relatively autonomous and can join or leave the system anytime. By distributing data storage, processing and bandwidth across autonomous peers, P2P systems can usually scale up to a very large number of peers. They have been successfully used for sharing computation, e.g., Seti@home [Anderson et al., 2002] and Genome@home [Larson et al., 2003a], [Larson et al., 2003b], internet services, e.g., P2P multicast systems [Bhargava et al., 2004], or data, e.g., Gnutellal [26].

P2P overlay Peer to peer system are built on a P2P overlay, and the overlay is built on top of the physical network (typically the Internet)[26]. The peer to peer overlay enables participants to find other peer not only by IP addresses, but also by a specific logical identifier. We consider three peer to peer architectures: structured, unstructured, super peer.

structured:

In a structured P2P network users directly participate to form a structured overlay or use services provided by a third-party structured overlay. Any query in the system can be resolved using the structured overlay in a bounded number of steps.these systems are structured because they maintain a close coupling between the network topology and the location of data via a hash table (DHT). This hash table is used to precisely define the data placement and lookup operations [23].

Super peer (semi-Structured)

A semi-structured P2P network is formed by a subset of all the users in the system (super peers), which take responsibility for storing the index and managing other users. The superpeers are responsible for providing the interface to the rest of the users for carrying out different system operations. Users' participation for providing super peer service can be voluntary or incentive based [23].

Unstructured:

An unstructured P2P network is formed when the logical links among participating nodes are established randomly. In these systems, no user maintains an index, and system operations are usually carried out using flooding or gossip-like communication between users. Unstructured systems are designed more specifically. for heterogeneous and distributed environments where maintaining strict restrictions on control data placement and network topology is not possible [23].

2.6 Applications based on blockchain and p2p

After the success of blockchain in the economic domain and e-commerce, new ideas emerged to apply this technology in different domains. the main idea is to give the users complete control over their data, and not rely on a central entity.

The blockchain technology allows an approach to creating secure applications, some academic research proposed partial use of blockchain in decentralized online social network(DOSN), Tawki is the best example for this.

In the following, we have selected some applications, related to our theme. We will explain how these applications have taken advantage of blockchain technology and the peer-to-peer network and some other technologies.

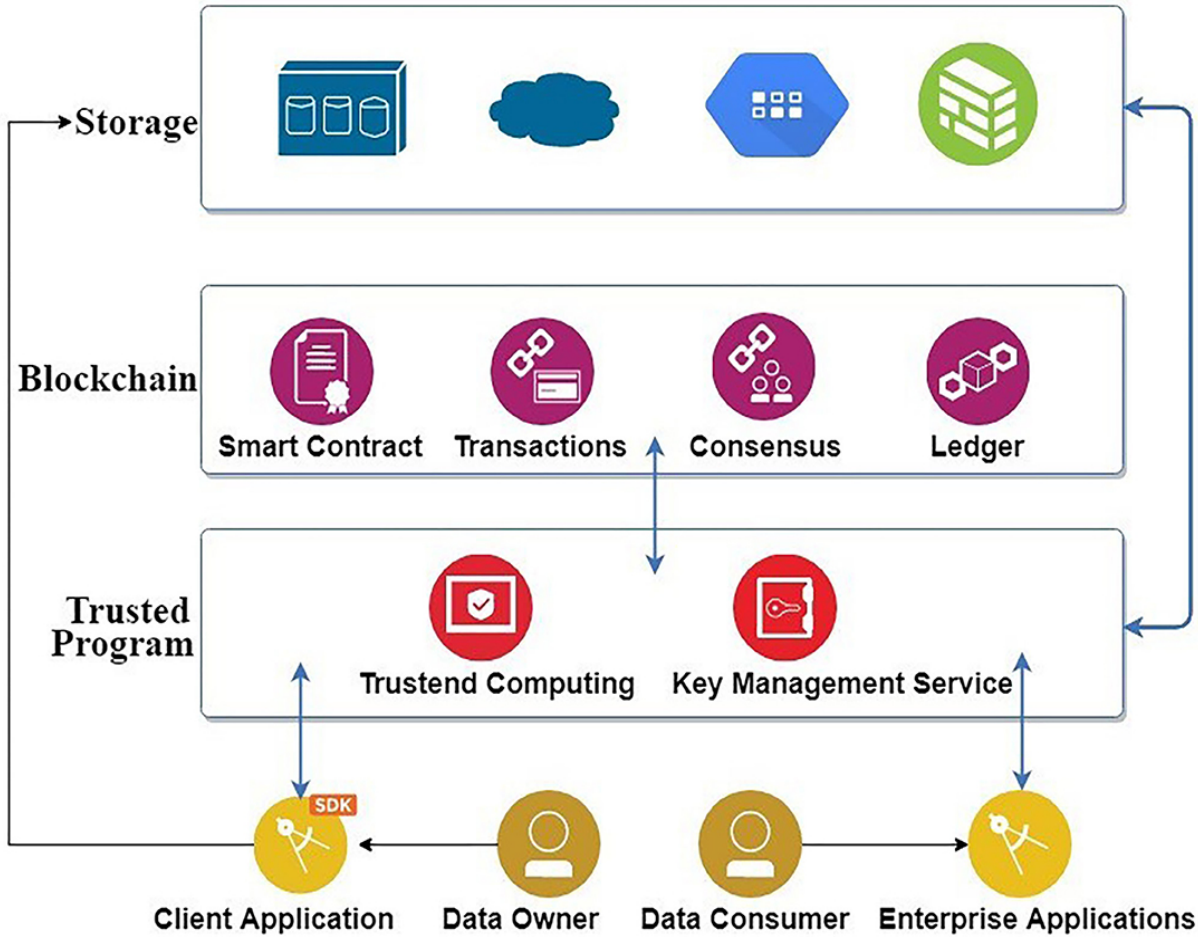

Figure 2.8: Architecture of DOSN

Tawki

Tawki is a decentralized service architecture for social communication. Users in Tawki remain in full control of their data, which is stored and managed by personal data storage, each data storage is accessible via Tawki API, which allows users to send and request data to and from other user's personal data storage[32].

Tawki utilize blockchain technology to manage identities in a decentralized and distributed fashion, building on a blockchain based identification mechanism that links self-hosted data storage to a distributed, secure and human-readable identifier. Tawki poses an alternative to current centralized or federated social networks [32].

To achieve complete control of data, users' content is stored in a personal data storage to ensure full control over the content(server), this back-end server does not only offer storage but also adds REST API for interaction with the client and exchange messages with other user's personal data storage. In contrast to client devices, the personal data storage server has to provide high availability to be accessible by other contacts who retrieve public or shared information and messages.[32]

In order to verify the validity of interactions with the server, for instance in server-to-server communication, a key pair is generated. The respective public key is stored and linked to the user on a distributed ledger. Server responses are signed using the private key and therefore verifiable after retrieving the key from the ledger[32].

The main key aspect in social network is discovering and connecting with friends and other contact. Tawki provided human readable names for identifying users.

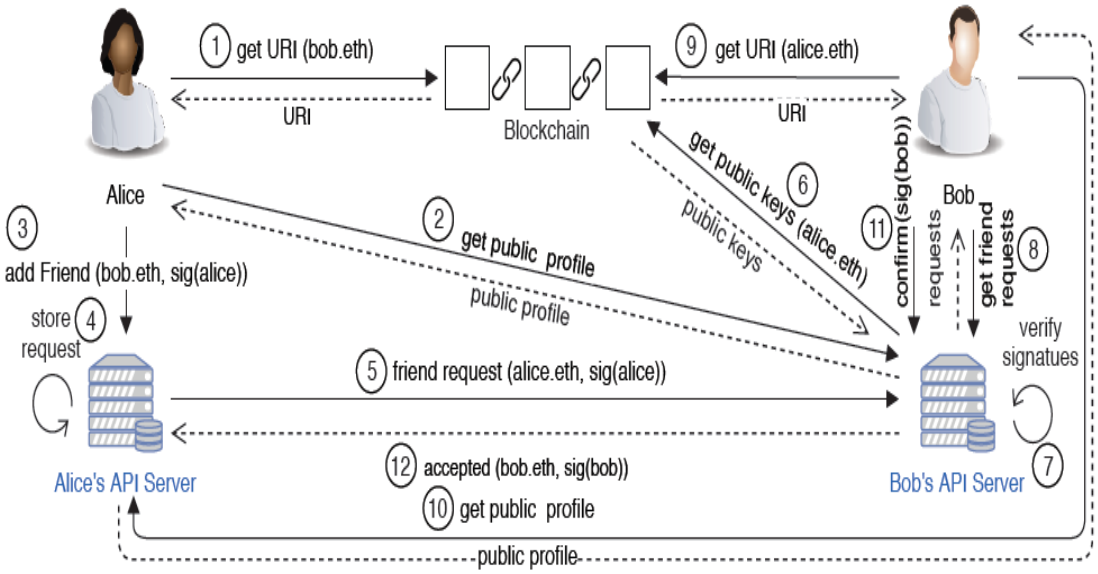

Figure 2.9: How friends request work in Tawki [32]

JAMI

Jami is a messaging application born out of a desire to create a solution to preserve privacy, previously called Ring, Jami is free software, that allows users to exchange text, sharing files and making a phone calls. Some features added recently like video conferencing, screen share, and streaming. All these features and more a completely secure, due to the distributed architecture, there is no central entity that stores data(email address, phone number ...).

Technologies behind Jami.

Users are identified internally by Jami with a forty-character string called a hash. This hash is different for each user, but it is very difficult for humans to remember. User name can be associated with the hash to increase the usability and security of Jami, allowing users to recognize themselves on the platform and know who they are talking to at all times.

Usernames must be managed by some sort of authority to ensure that they remain unique and that there is no ambiguity about the hash associated with them. JAMI use smart contracts of the Ethereum blockchain. These smart contracts have the advantage of being able to handle the management of unique usernames while remaining distributed.

Another technology that has made Jami robust is **OpenDHT**. However, fist lets understand what is Distributed Hashed Table: **DHT** is a class of distributed systems that allow access, from any node in the network, to a shared dictionary of key:value pairs whose data is distributed among the participants. Currently, the most popular DHT networks such as Mainline DHT (BitTorrent) are used for peer-to-peer file sharing. In these networks, the key is the identifier of the torrent file, also called "Magnet link", and the values are the IP addresses of the seeders, i.e. the clients sharing the torrent[12]. Inspired from **DHT** library, **OpenDht** is a lightweight and robust DHT network project in C++ offering an easy-to-use interface for application developers. **OpenDHT** offers the ability to store any type of data, not just IP addresses with a per-value limit of 64KB. It also provides a listen function, allowing a node to be informed of changes in values for a key.

Jami gives a complete peer-to-peer connection to take benefits of speed and size of transfers and take advantage of the bandwidth made available by their ISP and send files at the maximum possible speed or make video calls in full HD quality.

When devices communicate on the Jami network, they first send each other the set of public and local IP addresses they can be reached at via OpenDHT. This is the technology that allows the devices to find each other on the Internet and establish a first connection without knowing each other's IP addresses beforehand. Jami then uses Interactive Connectivity Establishment (ICE) to find the most direct route to create a link between the two peers using their respective addresses [15]

Stacks

previously called blockstack, in 2017, blockstack is a decentralized computing platform that aims to put users in control of their data and identity. Blockstack seeks a new infrastructure for the internet where users choose which data to share and whom to share it with. Through their platform design, applications developers can't access user data, users can choose who stores their data, and gives permission to read or write the data that are decided by them.

How Stacks work

This platform has multiple technologies involved inside, that makes it adequate for creating decentralized applications, without relaying in central entity.

Stacks API works on the application layer. It uses the underlying transport layer protocol TCP and UDP. It allows developers to build decentralized applications on top of the blockstack architecture, where user have complete control of their digital identity and the data associated with it[30]. There are three main components:

- **Blockchain** Stacks blockchain is actually composed of virtual chains built on top of the Bitcoin blockchain. It leverages the inherent security and trust Bitcoin blockchain to provide additional functionality such as Blockchain Naming System (BNS) and providing digital identity [30].

- **Peer to peer Network** Stacks has implemented a peer to peer network (Atlas) for the discovery of data. They have separated actual data storage from discovery. Atlas is only used to store pointers to the data, which is stored elsewhere on Gaia hub [30].

- **File Storage System (Gaia hub)**Stacks has treated cloud storage (Amazon S3, Google cloud storage[GCS]), etc.) as dumb storage area where encrypted user data is stored. Users don't have to trust the storage provider because they have complete control on their data. Only the owner can decrypt this data with their private key and storage providers will be left with data blobs only[30].

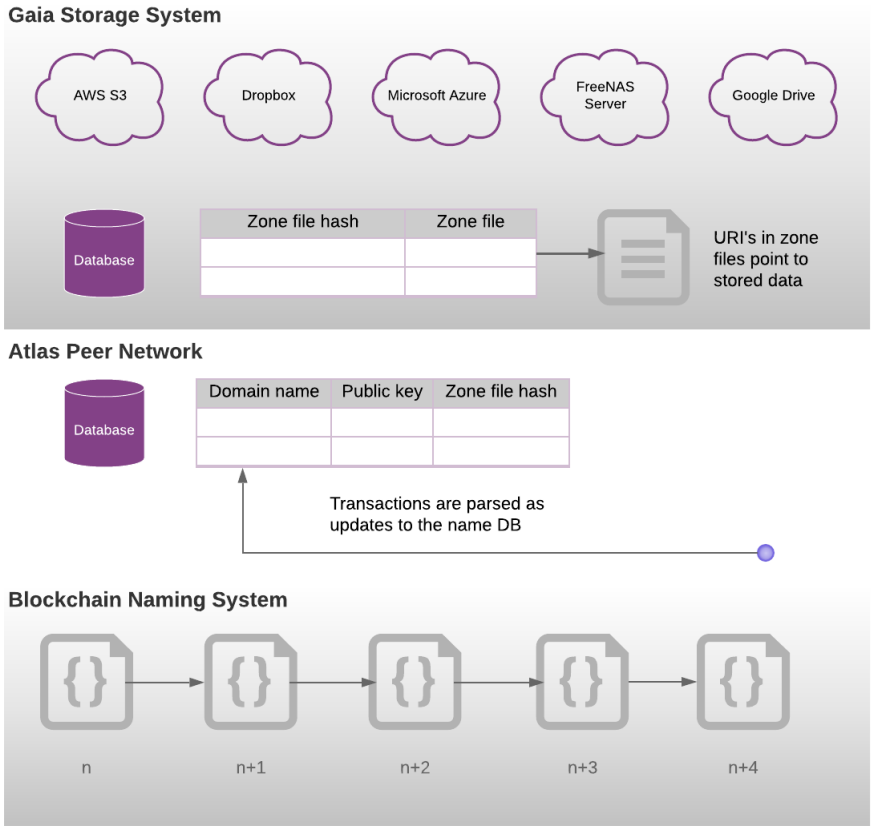

Figure 2.10: Stacks architecture

Gaia storage system

Gaia stores data as a simple key-value store. When an identity is created, a corresponding data store is associated with that identity on Gaia. When a user logs into a DApp, the authentication process gives the application the URL of a Gaia hub, which then writes to storage on behalf of that user[30].

The Stacks blockchain stores only identity data. Data created by the actions of an identity is stored in a Gaia Storage System. Each user has profile data. When a user interacts with a decentralized App that application stores application data on behalf of the user. Because Gaia stores user and application data off the blockchain, a Stacks DApp is typically more performant than DApps created on other blockchains.

Gaia's approach to decentralization focuses on users control of data and its storage. Users can choose a Gaia hub provider. If a user can choose which Gaia hub provider to utilize, then that choice is all the decentralization required to enable user-controlled applications[30]. By default stacks, use Gaia hub to store data user' encrypted by the user public key. With this, no one can see what inside the hub even the cloud provider.

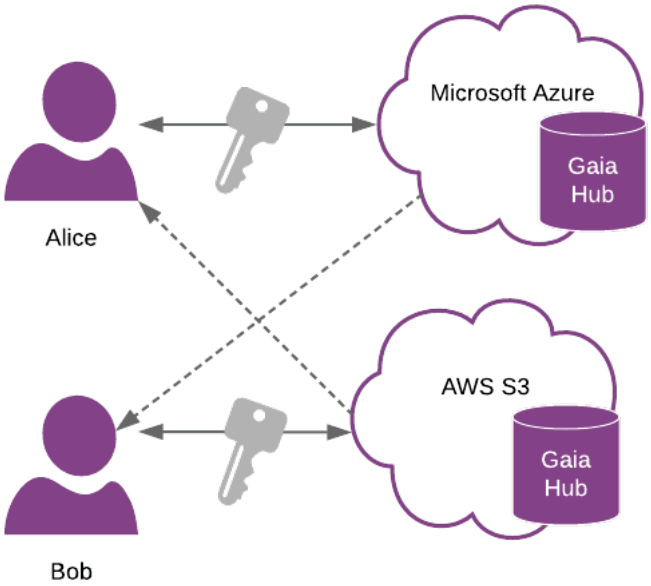

Figure 2.11: Interaction between Gaia hub and client

Atlas P2P network

Atlas is a P2P network that based on gossip protocol, Atlas has the ability that each peer can keep a record of the existing peer on the network, and each peer try to store a copy of all data to prevent failure and tolerate a high level of scalability.

Another particularity of Atlas, that the network is divided into two-part, the first one is responsible to store the data of the blockchain (blocks and transactions), while the second one stores the data of those hashes. As a protection mechanism, the data stored in the second section is cryptographically related to avoiding manipulation [7]

The Stacks Blockchain

Stacks solves the bootstrapping of trust problem by leveraging the most secure public blockchain (currently Bitcoin). More specifically, stacks binds (human-readable) domains to public keys to establish ownership of domains. These domains have associated data records as well. These small bindings are stored on the blockchain and are tamper resistant. The actual payload from the data records is stored outside of the blockchain, because blockchains have limited storage space and are not meant to be used as general-purpose databases.

Blockchains don't have central points of trust or control, and stacks nodes can use the stacks blockchain to boot up and connect to the new decentralized internet without relying on any remote servers. stacks's blockchain is built using a technology that we introduced earlier, called virtualchains. Virtualchains are to blockchains what virtual-machines are to physical computers. By using virtualchains for implementing a blockchain, Blockstack can survive the failures of underlying blockchains. Therefor, blockchains will come and go, but the apps built on top need to outlast the underlying blockchains [10].

2.7 Conclusion

In this chapter we saw the trending technologies that helps users to have more control over their data, we talked about Blockchain technology and it's features, we also talked about peer-to-peer architecture, and finally we mentioned some applications like Jami and Tawki based on these technologies.

Chapter 3

Conception

3.1 Introduction

In order to achieve our goal which is making a secured messenger app, We will propose an application that should fulfill our desire. A decentralized messaging application based on *Blockchain*. In our research, we found a way to secure the users data which should be stored in a safe cloud storage system.

We will propose a conception where we minimize the amount of information that can be found, leaked or concluded about the user's interactions, with whom has he been interacting or the content of the messages. Then, we shall specify the data flow (data structure) and the users interactions with the app and the blockchain.

3.2 The application's global architecture:

The goal of this application is to allow users to exchange messages, keep in touch and communicate as they would while using other messaging applications with equal performance but in a safer manner in a way where they guarantee that no one is sniffing on their personal data nor their conversations. No third party will be able to know the user's interactions or favorite contacts, also the time you spend online chatting will remain a secret. The system guarantees users that no one is trading their data which is the least thing anyone would want and a very important feature.

So next, we'll represent the architecture of the app as it will be including blockchain and cloud technologies to allow each user to store his data in his own space.

3.2.1 Contact adding

Case 01: the users know each other's ID/username

If the two users are friends in real life or somehow they know each others ID/username, they can add each other to their contact's list without the interference of the app or any other third party. Thus, no one in the world can know that these two users are in contact.

Case 02: the two users don't know each other

In the case where user1 wants to add user2 but they don't know each other, we will need the interference of a third party which is the app just to inform the users about each other's existence. So, user1 will tell the app that he wants to interact with user2. The app in turn, will inform user2 that someone wants to interact with him under the specific ID of user2. Straight away after user2 gets this notification he'll choose whether he wants to add user1 to his contact list or not.

Now in case user2 want in fact to add user1 to his contact list, he'll do so without informing the app about it. So the app won't actually know any detail or know for sure if they are in contact or how many times they contacted each other...etc.

3.2.2 Messages exchanging

Case 01: Users are not in each others contact list

If a user wants to send a message to another user but he doesn't have him as a contact he should go throw the step of adding him as a contact which we mentioned previously in the last section.

Case 02:Users are in each others contact list

In this case where Alice is messaging BOB, There are two sides to the story:
Alice's side (the sender):
Alice shall write Bob a message, encrypt it with his public key, store the message in an

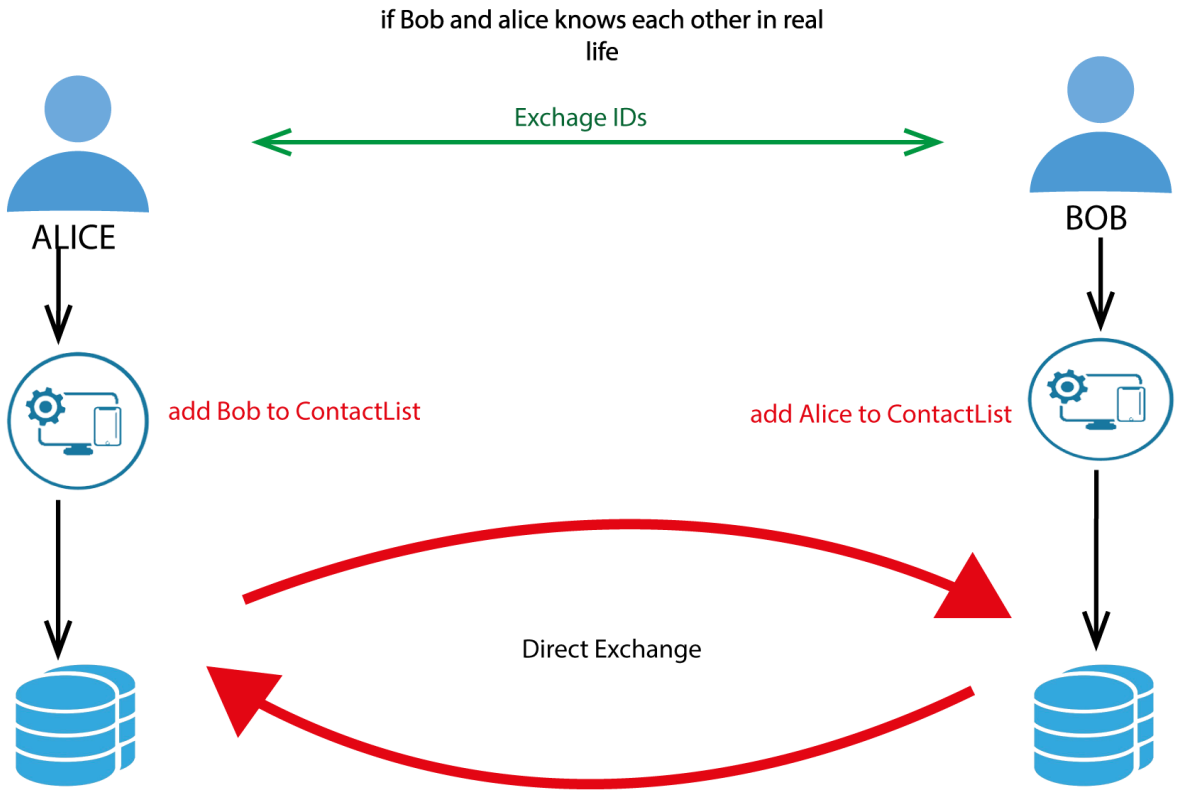

Figure 3.1: How users add each other

index specified for messages meant for the contact "BOB" and encrypted with a mutual key only Bob and Alice know, This index with the rest of the "indexes of sent messages" is saved in Alice's personal cloud storage space.

Alice has multiple indexes of "sent messages", Each one is meant for a specific contact.

The index of a specific contact will be encrypted with a mutual key.

Bob's side (the receiver): When Bob is online he'll consult the index of "sent messages" in Alice's storage space, now Bob shall retrieve it's content, decrypt it with the mutual key that only he and Alice know. Bob will fetch for new messages, decrypt them with his private key and read them.

Now even if Bob somehow retrieves someone else's index of messages he won't be able to see it's content as he does not have the mutual key between Alice and that person.

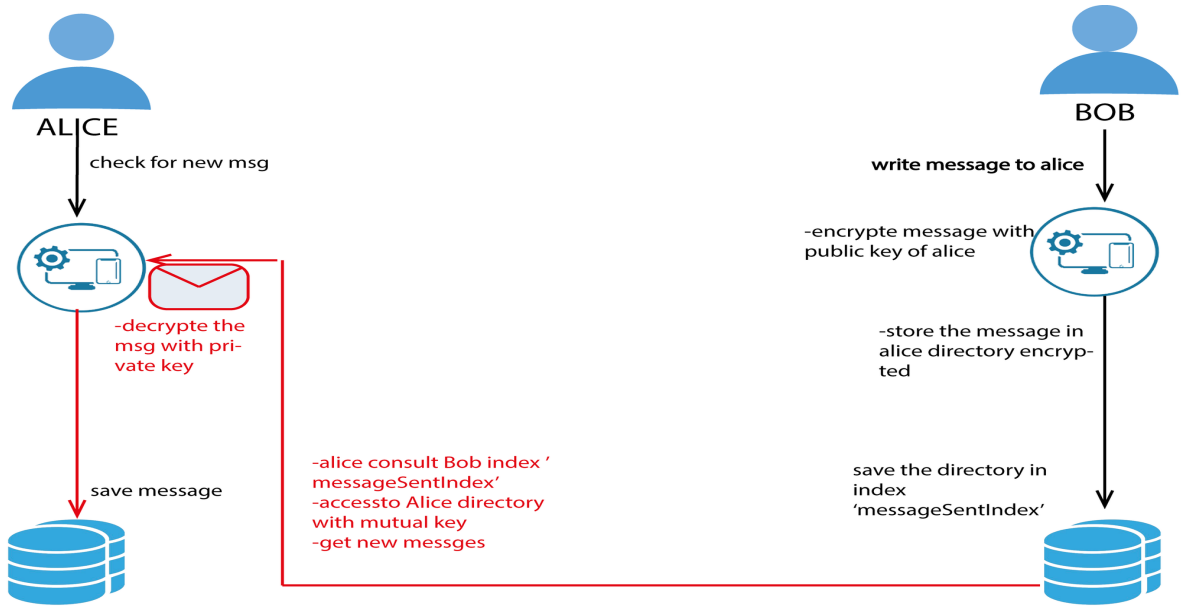

Figure 3.2: Figure of message exchange between the sender and receiver

3.2.3 Mutual Key exchanging

For them to exchange the mutual key safely, Let's say that user1 wants to add user2, he will generate a symmetric key encrypt it with the public key of user2 and store it in an index for him, user 2 now can access it, decrypt it and start using it to access the "sent messages index" where he can find new messages meant for him.

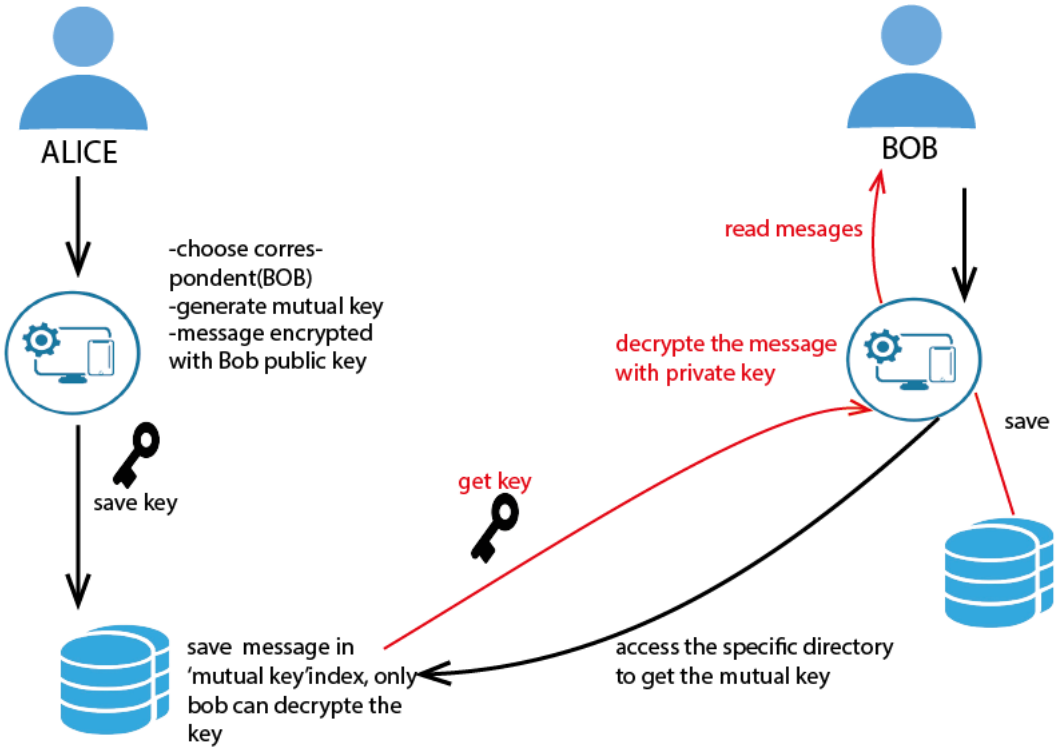

Figure 3.3: Figure on how users exchange their mutual key

3.3 Application features :

As we already mentioned our app is a messaging app like any other similar app and it should at least have the very basics of messaging.
The main feature of our app is:

Text Messaging

3.4 Data structures and access to messages:

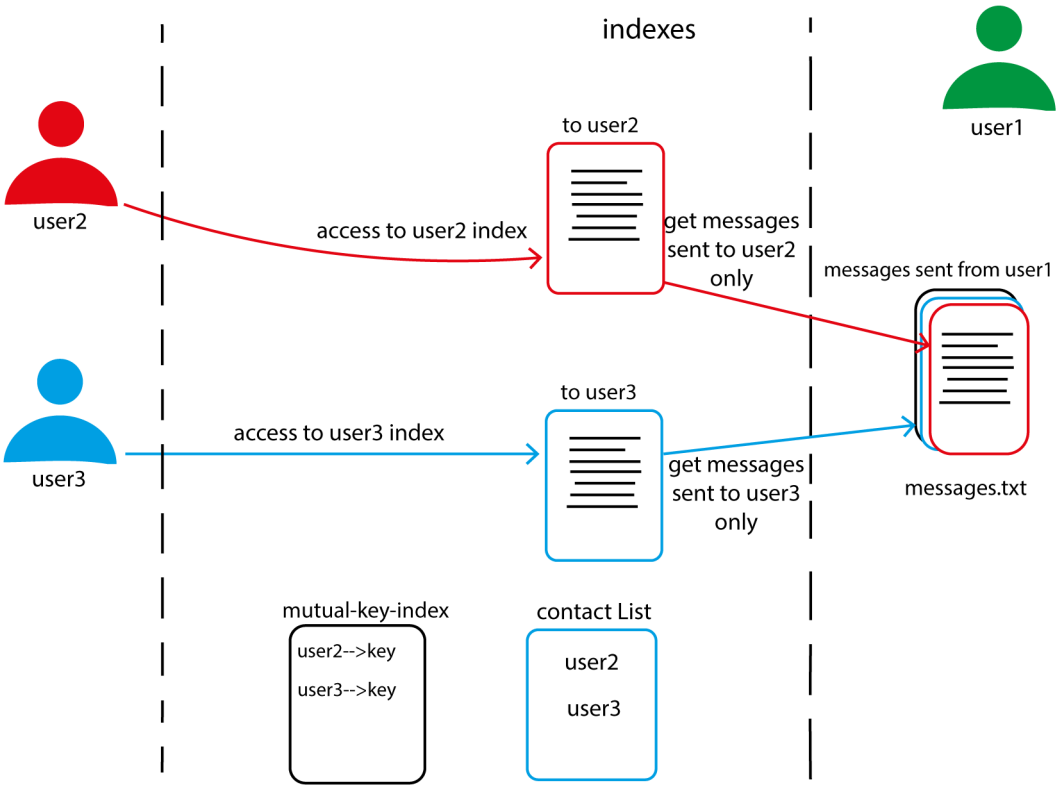

Figure 3.4: Data structures and access to messages

3.5 Modeling:

This section will be devoted to the modeling of our application, for that, we have chosen the UML (Unified Modeling Language) for its various advantages (modularity, abstraction, coherent structuring of features and data, variety of diagrams...).

3.5.1 Functional view :

After identifying the actors whom will be interacting with the system, in order to be able to precisely establish the boundaries of the system. With that in mind, we will identify some diagrams. Then, we'll specify the functional point of view by detailing the different ways in which the actors can use the system.

Identifying the actors in our system:

- **internet user:**A simple internet user who can browse the welcome page and sign up to the app to have access to its features.
- **App user:**It is the consumer who benefits from the services and features offered by the app.

3.5.2 Dynamic view :

In this part we will clarify the dynamic view by using sequence diagrams.

Sequence diagrams:

It sequentially represents the course of processing and interactions between the elements of the system and / or its actors.

realization of Sequence diagrams:

We will now present some sequence diagrams which will clarify how our system should work.

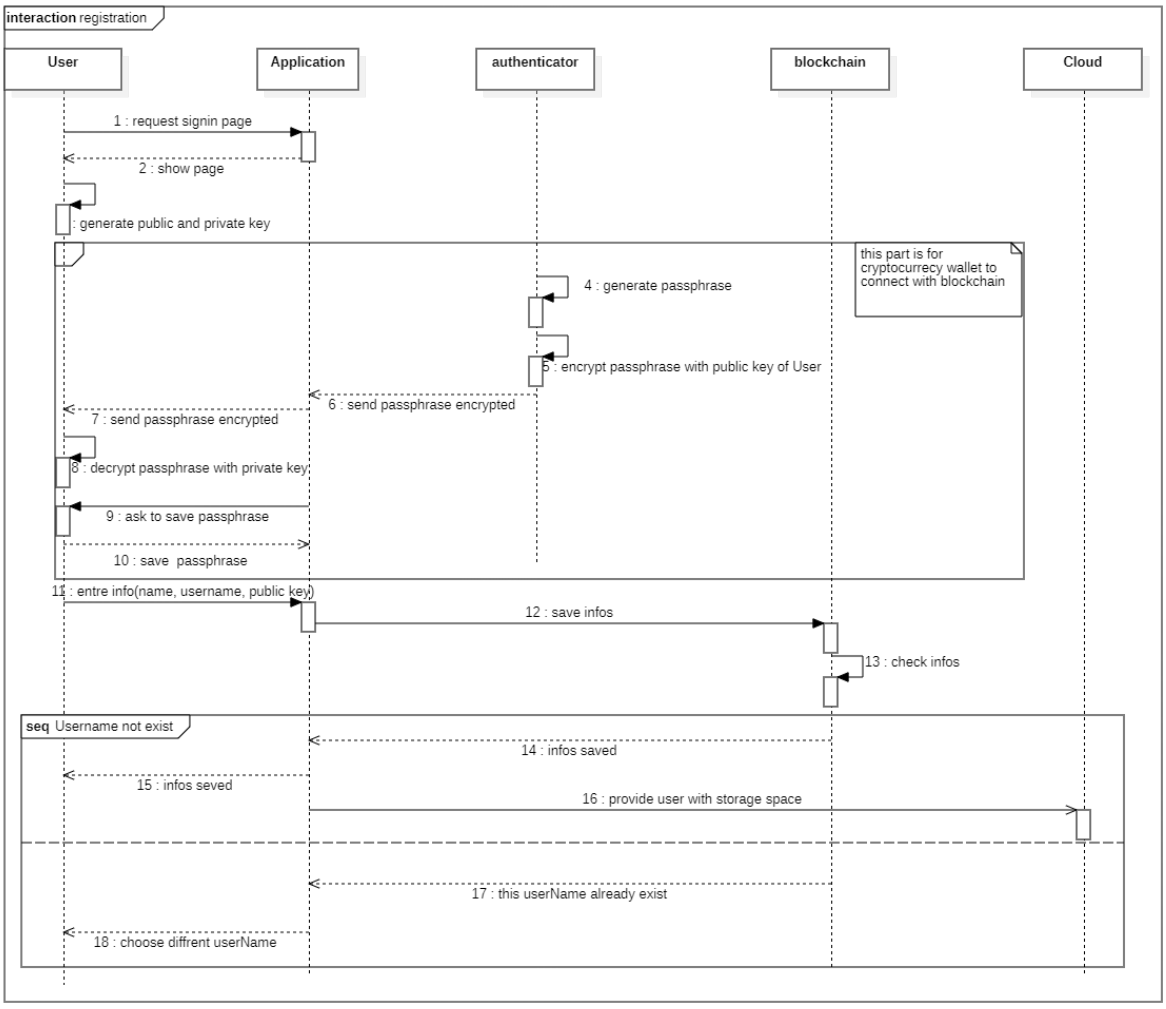

Figure 3.5: Sequence diagram of the registration process

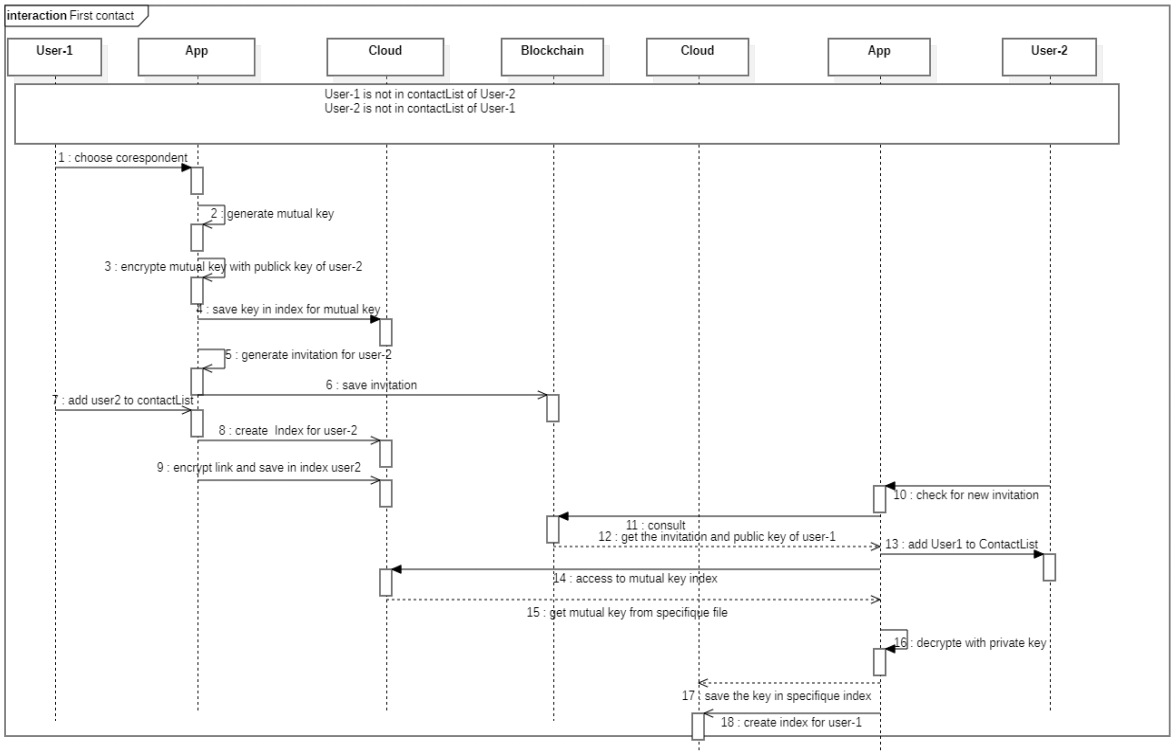

Figure 3.6: Sequence diagram of the first contact between two users

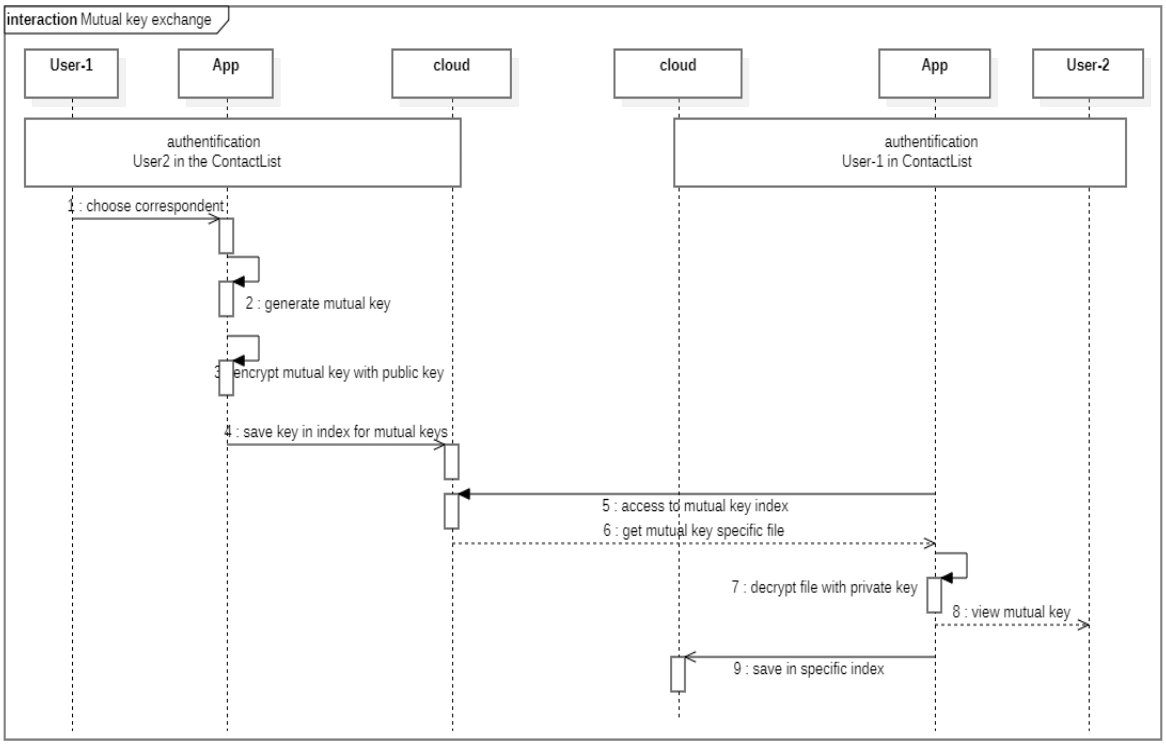

Figure 3.7: Sequence diagram of mutual key exchange between two contacts

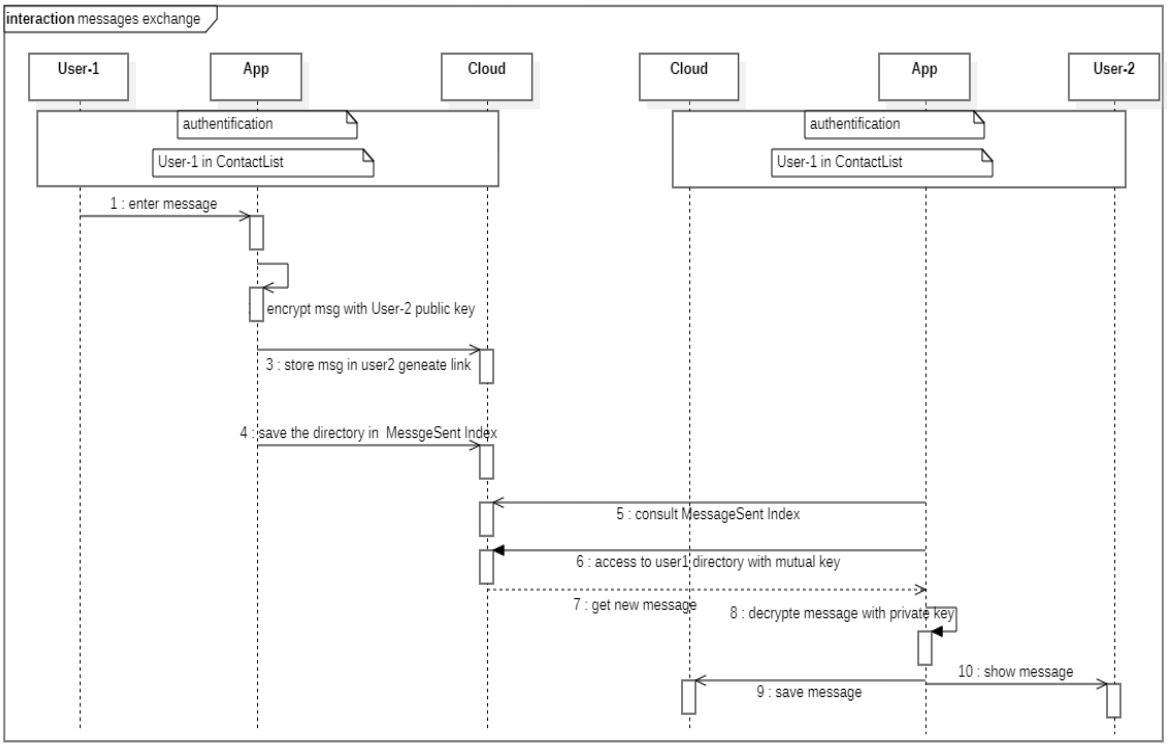

Figure 3.8: Sequence diagram of message exchange between two users

3.6 Conclusion

In this chapter, we detailed our proposed architecture of a secured messenger application.

We modeled an app that is peer to peer uses blockchain and cloud storage technologies. We also clarified the data structure, the actors along with their roles and interactions in detail. Now, we shall represent the implementation details.

Chapter 4

Implementation and realization

4.1 Introduction

In this chapter, we aim to realize our proposed application. You may notice that our application is designed according to the Stacks platform: identities and routing using Blockchain and Cloud base storage. In fact, we aimed to implement our application on top of this emerged system. The first part of this chapter is to explain how we tried to use Stacks API to realize our application. This includes the used tools and the successfully implemented parts. Unfortunately, several technical issues made it impossible to achieve our goal in the limited time that we had to finish the project. So, we opt for a simulation of the entire system to prove the efficiency of our proposition. The simulation environment and results are also exposed in the second half of this chapter.

4.2 Work environments

Hardware environment

For this project, we used a HP PRO-BOOK laptop with Intel i-5 and 8GB RAM, with Windows 10 system 64bit as an operating system, HP Notebook with Intel i-5 and 8GB RAM, with Windows 10 system 64bit as operating system.

Software environment

we used different tools and software to maintain this project, conception software, graphic software and more...

Visual studio code

Visual studio code is an integrated development environment(IDE), created by Microsoft for Windows, macOS, and Linux. it's a code source editor that can work with multiple programming languages like Java, JavaScript, Go, Node.js, Python and, C++. Visual studio can be extended with extensions, available in a central repository and it offers different other functionality like choosing to encode character (ANSI, UTF-8, UCS-2) and offer a good syntax color to be understandable.

Adobe Illustrator:

Adobe Illustrator(AI) is vector-based graphic software developed by adobe. Illustrator is used to creating a variety of digital and printed images like cartoons, logos, charts, diagrams, and others...

Illustrator also makes it possible to manipulate text, images in many ways, illustrator provides a bunch of tools for creating postcards, posters, and other visual designs which use text and images together[6].

StarUML:

StartUML is UML modeling software, that allows access to multiple diagrams and use each diagram to suit any kind of modelisation. We used StarUML to make our conception part.

Github:

Is mostly, a hosting platform for coders and developers, the cloud-based service allows coders and developers to effectively manage and

maintain open-source programming projects while collaborating with each other [34].

Adobe XD:

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows[5].

Figma:

Figma is a vector graphics editor and prototyping tool which is primarily web-based, with additional offline features enabled by desktop applications for macOS and Windows. The Figma Mirror companion apps for Android and iOS allow viewing Figma prototypes in real-time on mobile devices[16].

programming languages

HTML/CSS:

HTML (Hyper Text Markup Language) created in 1991 by Tim Berners-Lee, Robert Cailliau, and basically used for creating and managing web pages, and web applications. It's managing the way the content of their web pages is displayed on a screen, through the browser. It is based on a system of tags. HTML evolves over time and now the last version of HTML5 gives a lot of features that help developers.

CSS(Cascading Style Sheets) CSS allows creating good looking web pages. We will use this language to make HTML pages more attractive and give them a special style.

Javascript:

Javascript is the most popular programming language for the front-end, it is widely used to create an interactive front-end web application, it's also used on the back-end side. It has multiple libraries and frameworks that help to create a different types of applications.

PHP:

Created in 1994 by Rasmus Lerdorf, Php is a scripting language generally used on the server-side, it is suited for web development and can be embedded into HTML.

MySQL:

MySQL is an open-source relational database management system (RDBMS) on SQL – Structured Query Language. It offers two different editions: MySQL open-source community and the proprietary Enterprise Server. Association with a scripting language like PHP or Perl, it is possible to create websites that will interact in real-time with a MySQL database rapidly.

4.3 Libraries and frameworks:

4.3.1 React.js

React.js is a free open-source front-end JavaScript library, created by Facebook in 2013, it is made for building user interface and UI(User Interface) components. React allow developers to create web application without reloading pages, it has several advantages one of them is the simplicity and extensibility, it can be used to create a simple application to complex web applications [27].

4.3.2 Node.js:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser[25].

4.3.3 Jquery.js

Jquery is a javascript library, it makes things like HTML document traversal and manipulation, event handling, animation, and ajax easy to use that works in multiple browsers, it also provides developers to create plugins.

4.3.4 Stacks.js:

Stacks.js is a library which provides a programming interface by Stacks platform to help developers make reliable, trusted and secured decentralized applications. We used from their library the following features:

- The authentication system which is based on blockchain technologies. The system manages the users profile creation and identities.
- The storage system (Gaia cloud storage) which is a decentralized cloud storage.

Authentication/(SignUp,Login and logout):

To authenticate a user, we use a function called $redicrectToSignIn()$ which will redirect the user to the Stacks Blockchain authentication page where he can chose to $Log-In$ or create a new profile if he doesn't have one already. To logout we use a function called $signUserOut()$ which redirects the user back to the login page.

```
1  import React from "react"
2  import { UserSession, AppConfig } from "blockstack"
3
4
5  const appConfig = new AppConfig(["store_write"])
6  const userSession = new UserSession({ appConfig: appConfig })
7
8  class Login extends React.Component {
9
10     handleSignIn = () => {
11       this.props.userSession.redirectToSignIn()
12     }
13
```

Figure 4.1: SignIn/Login function

```
24    //handle signout
25    handleSignout = () => {
26      this.props.userSession.signUserOut(window.location.origin)
27    }
28
```

Figure 4.2: SignOut function

Reading and writing Data on Gaia's decentralized cloud storage system:

Apps built with the Stacks blockchain store off-chain data using a storage system called Gaia. Whereas public transactional metadata is best stored on the Stacks blockchain, user application data can often be stored more efficiently and privately in Gaia storage. Storing data off of the blockchain ensures that Stacks applications can provide users with high performance and high availability for data reads and writes without introducing central trust parties[31].

We used the $putFile()$ and $getFile()$ functions to interact with the system, one is for reading data from the storage while the other is for writing data/storing it.
The $getFile()$ function is used alongside another function called $loadUserData()$ which loads the current logged in user's data. Here is an example where we store a sent message as an array in a Json file and read it back:

```
41  handleAddMessageClick = e => {
42    e.preventDefault()
43    const newMessage = {
44      id: this.state.messages.length + 1,
45      content: this.state.newMessage,
46      time : this.state.curTime,
47
48    }
49    const messages = [...this.state.messages]
50    messages.push(newMessage)
51    const options = { encrypt: true }
52    this.props.userSession
53      .putFile("messages.json", JSON.stringify(messages), options)
54      .then(() => {
55        this.setState({
56          messages,
57          newMessage: "",
58        })
59      })
60  }
```

Figure 4.3: Writing data to the cloud storage as a Json file

```
async fetchData() {
  const options = { decrypt: true };
  const file = await this.props.userSession.getFile("messages.json", options);
  let messages = JSON.parse(file || "[]");
  this.setState({
    messages,
    user: new Person(this.props.userSession.loadUserData().profile)
  });
}
  render() {
    const { user } = this.state;
    return (
```

Figure 4.4: Reading data from cloud storage as a Json file

While writing and reading data from the Gaia decentralized cloud storage, you have the option to save your data publicly or you can choose to crypt it with the possibility to specify the key. For an example, it can be saved for a specific user and crypted with his public key so that only he can decrypte and it.

4.3.5 Different Scenarios and graphic presentation of the stacks App

Figure 4.5: Front Page/Login Page

After pressing the login button, the user will be redirected to the sign-in page with the option to sign-up if he is new.

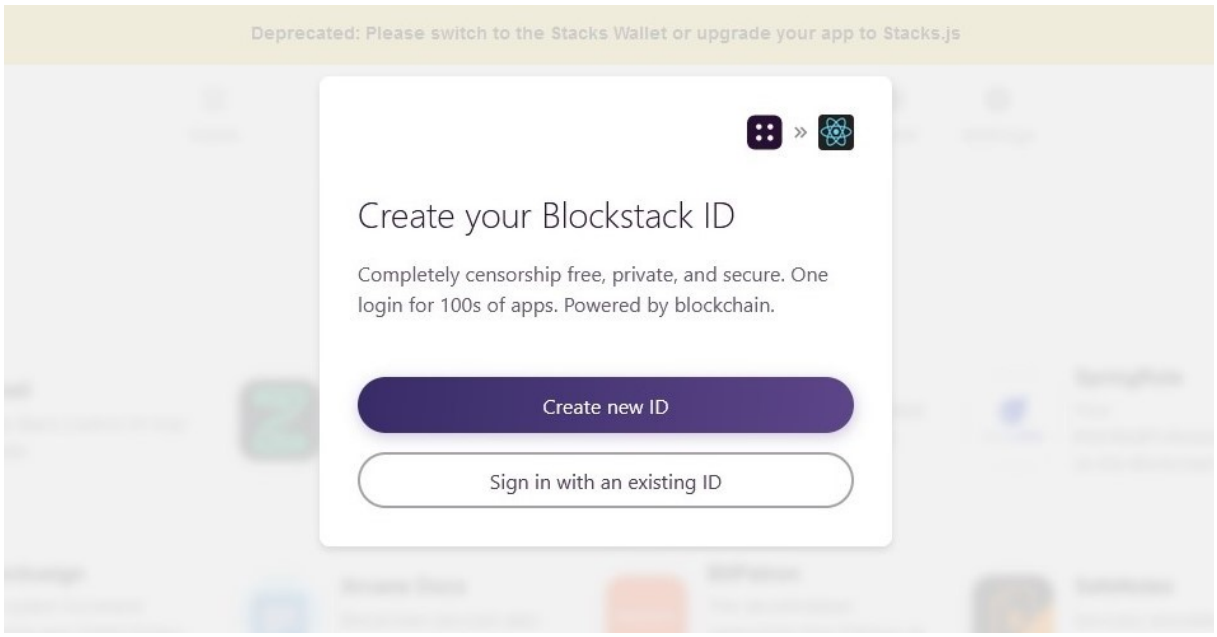

Figure 4.6: Sign-In/sign-Up options

If the user already has an ID and a security key he can simply log-in.

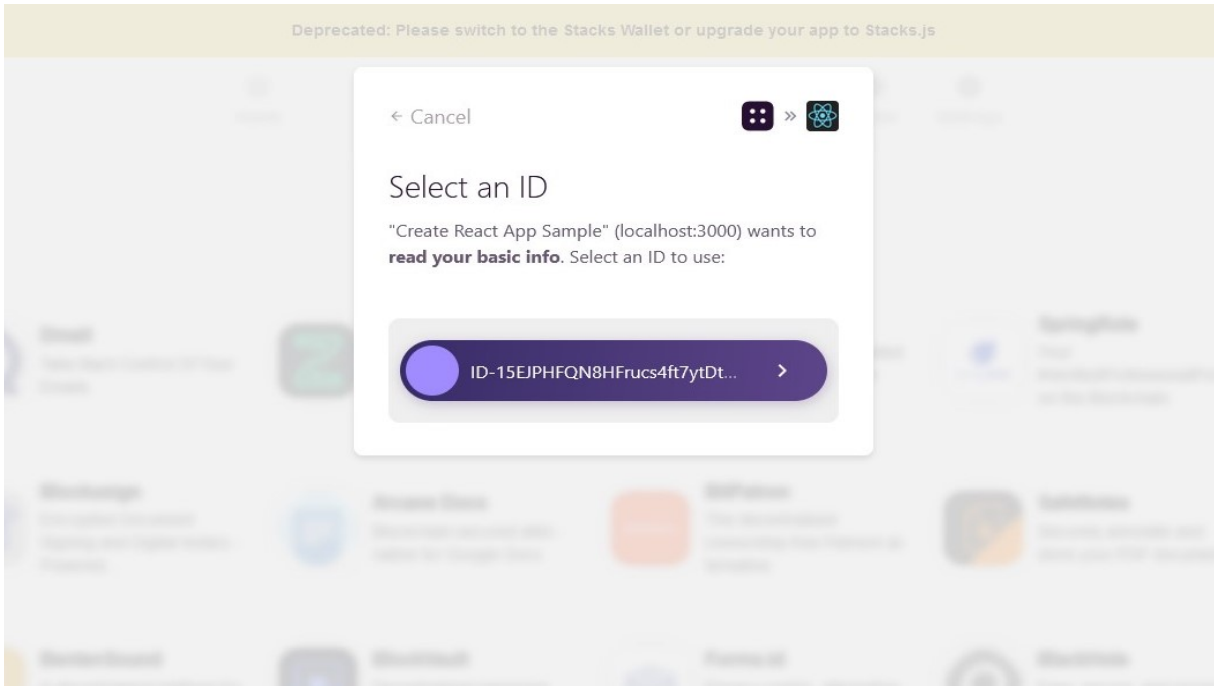

Figure 4.7: User log-in with an existing ID

If the user is new, he will have to go through the following steps to create a new ID:

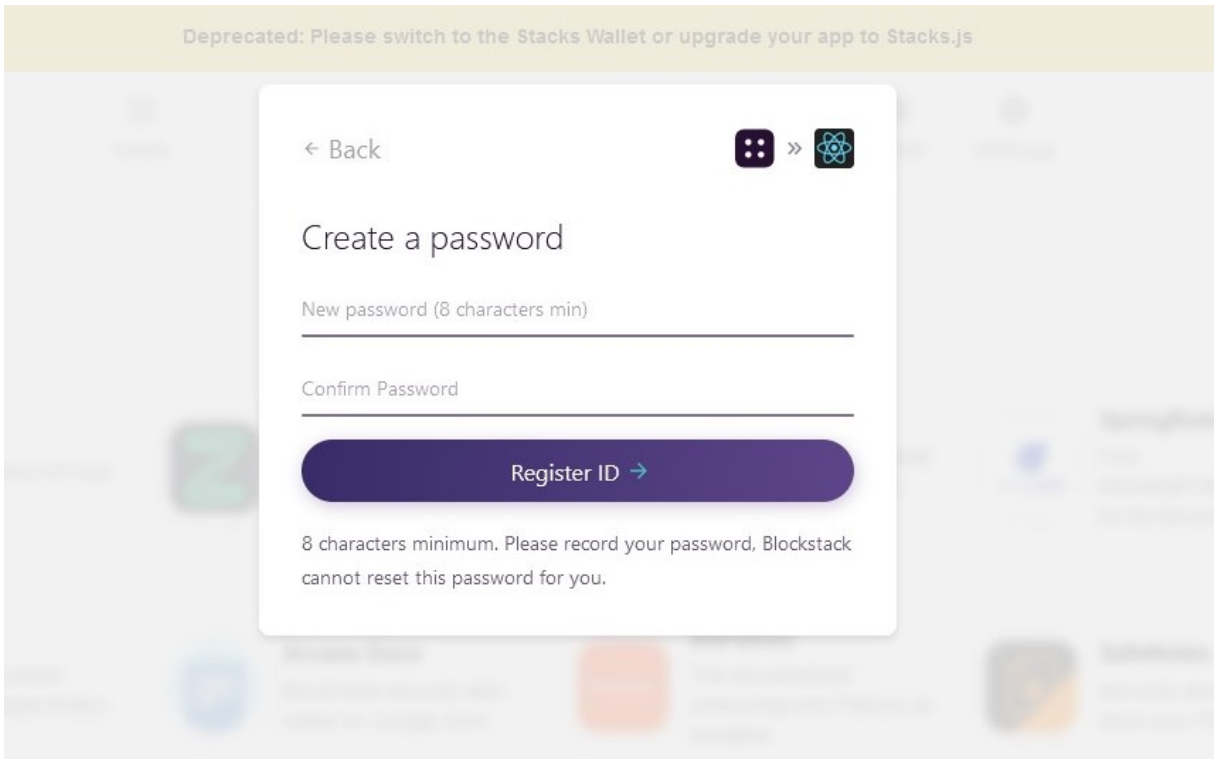

Figure 4.8: Sign-Up step 1 (Create a Password)

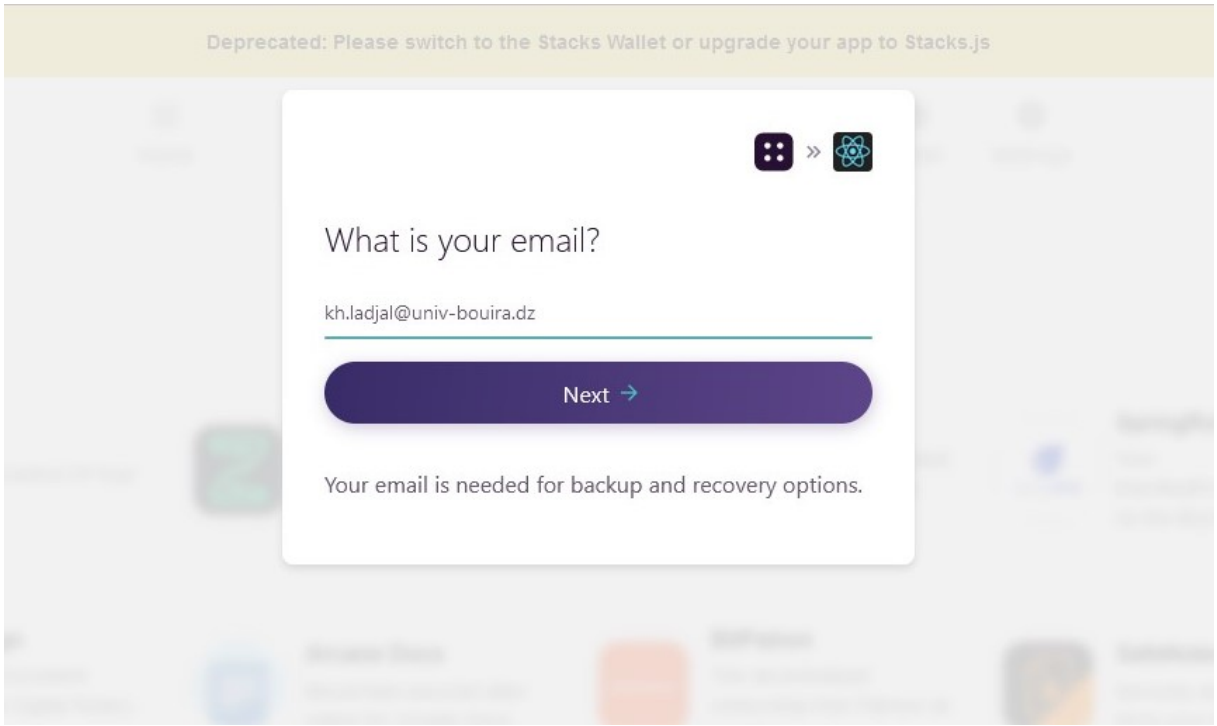

Figure 4.9: Sign-Up step 2 (ADD email)

Every new user will be offered a recovery/secret key which is very important. This key is his only way to confirm his identity. Later on, he will be asked to save it and to be

very careful with it for security measures.

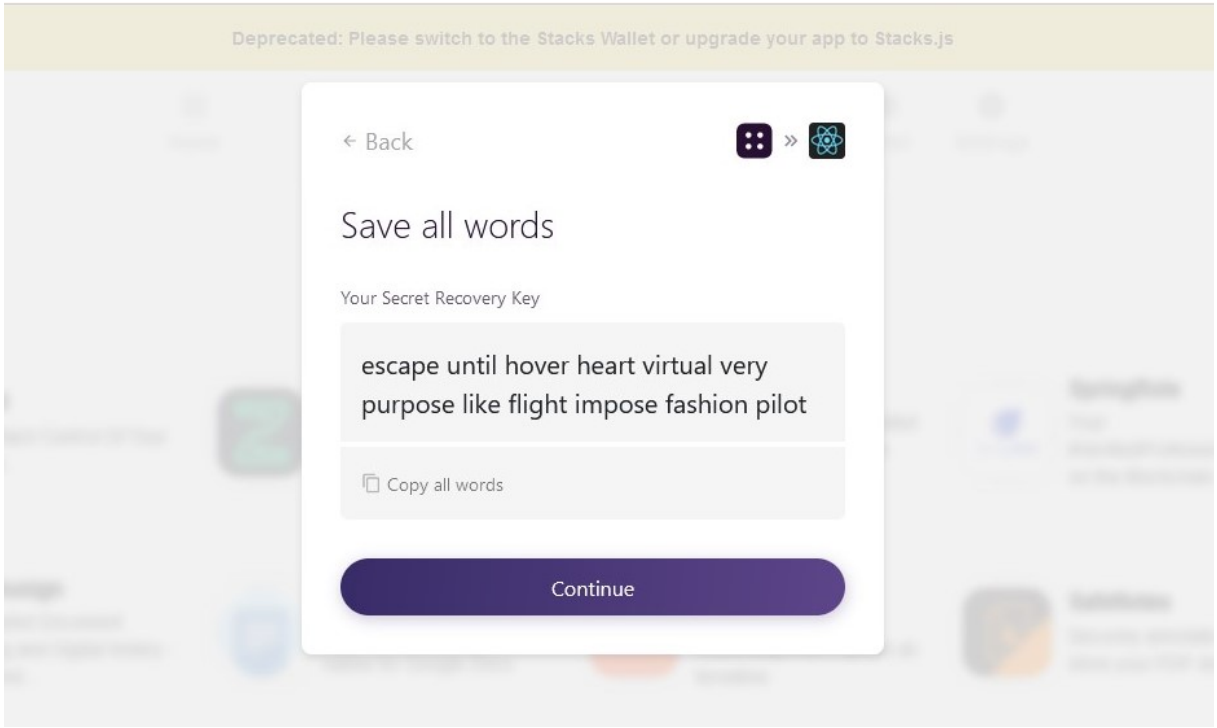

Figure 4.10: Sign-Up step 3 (Save recovery key)

After creating a new account with a unique ID, it's better for the user to proceed to $https://browser.blockstack.org/profiles$ and complete his account by adding an avatar and a user name.

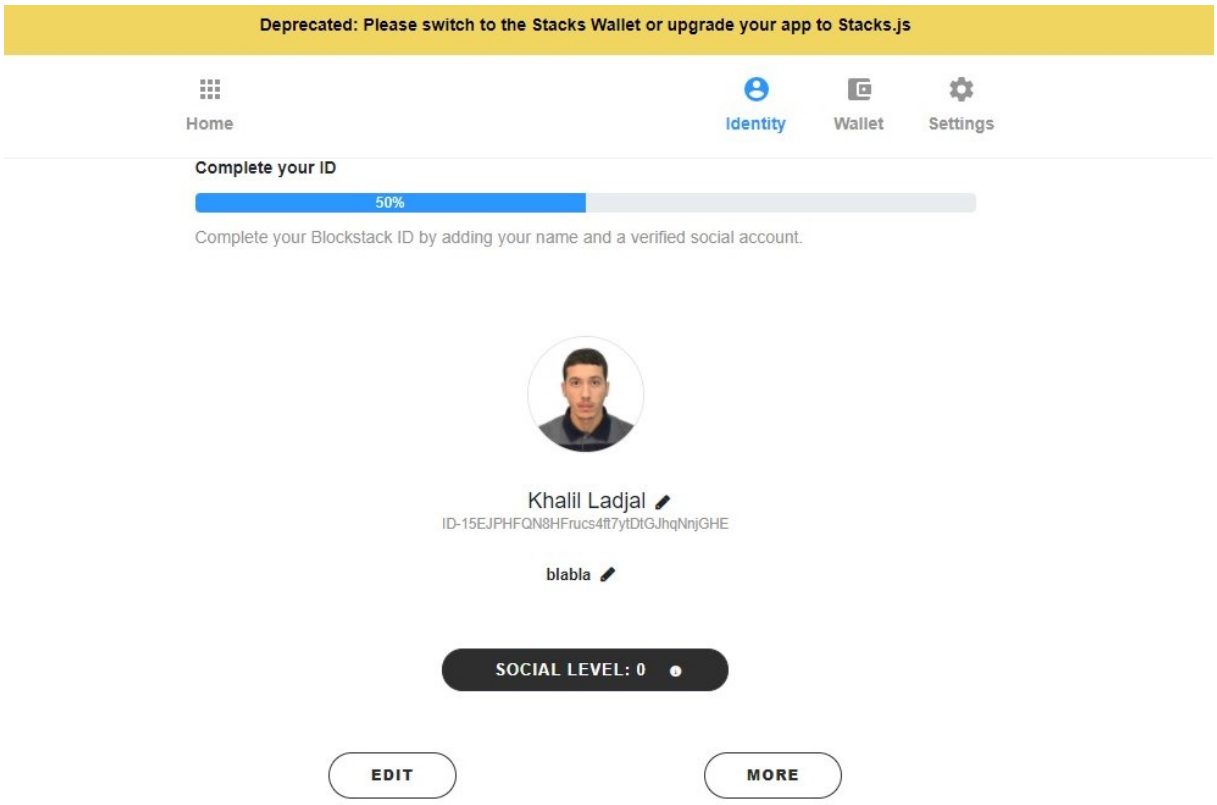

Figure 4.11: Editing/completing a stacks profile

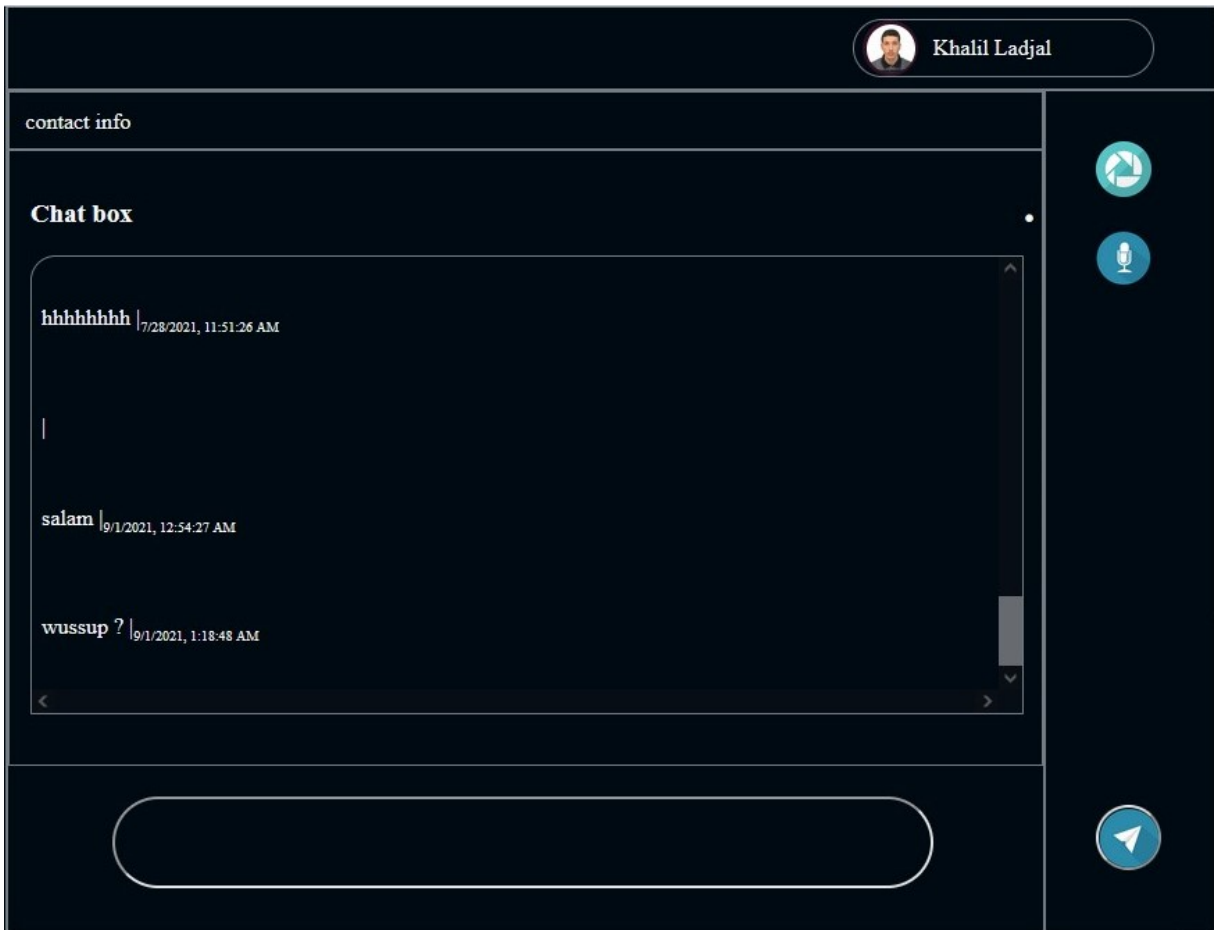

Figure 4.12: App interface as a Logged In user

End of progress duo to multiple issues

Unfortunately, we were not able to finish the live Stacks Decentralized Messenger App to depose it online as we initially planed duo to multiple issues and obstacles, these issues were at most related to the Stacks platform as they added new updates and made a lot of changes in a very short time. These changes were not expected they included changes to the Stacks.js library itself. This made it very hard for us to adapt. One more reason is that, the Stacks community is very small as it's a very recent project. We concluded that the implementation of the whole system needs more time and technical master of the Stacks technology. Therefor, we jumped into making a simulation to be run locally and demonstrate how the initial conception plan should look and operate.

4.4 Scenarios and presentation of our simulation

Forthwith and as we mentioned in the last section, we weren't able to stick to our initial plan. Thus, we made a simulation app as an alternative in order to elucidate & demonstrate the architecture behind our project, the goal of this simulation is to explain in detail how the architecture should be in blockchain and decentralized cloud storage. Now we shall break each step of the simulation.

4.4.1 First step: Sign in

In order for new users to utilize a new application on the Stacks platform, they need to sign in or register in blockchain, where Bitcoin-Blockchain is used to save data. It is for a fact that blockchain is a type of distributed database, that keeps data in blocks.
So in our case, we took the database aspect of the blockchain, so as to all the necessary pieces of information of the new clients will be saved in one database(name, user name, public key...).

Inside this database, we find all information about the clients like name, user name, a public key unique for each user, and others. After this, a new database is created for the new client with his name that contains messages table, contact list table and mutual key table.

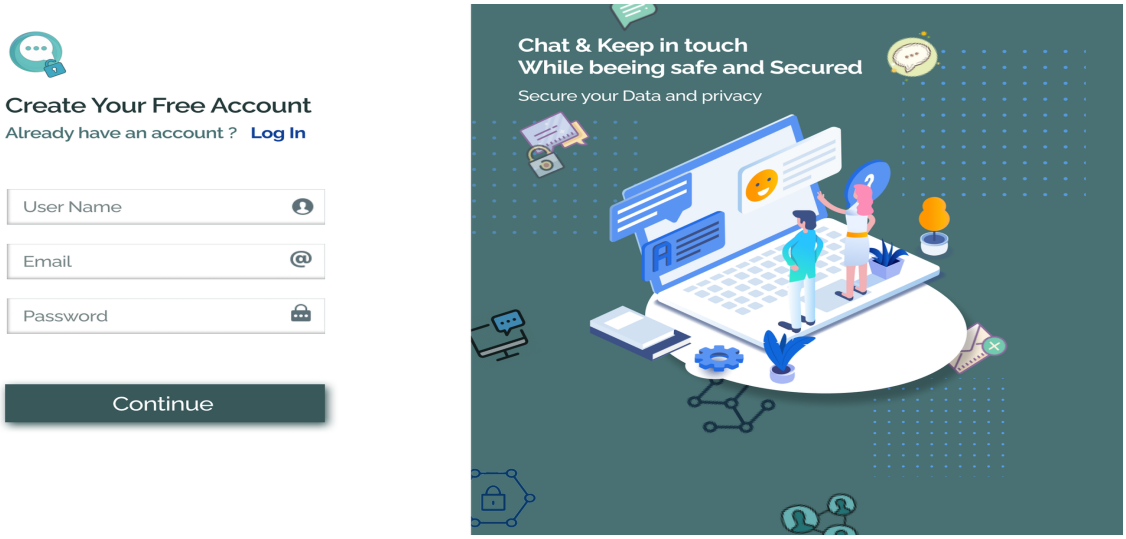

Figure 4.13: SignIn page

4.4.2 Second step: Login

Login part is quite simple, the same thing happens in the Stacks platform this part is to check if the information entered is correct and exists in the blockchain database, if it is then the client has access to his storage and begin to use the application.

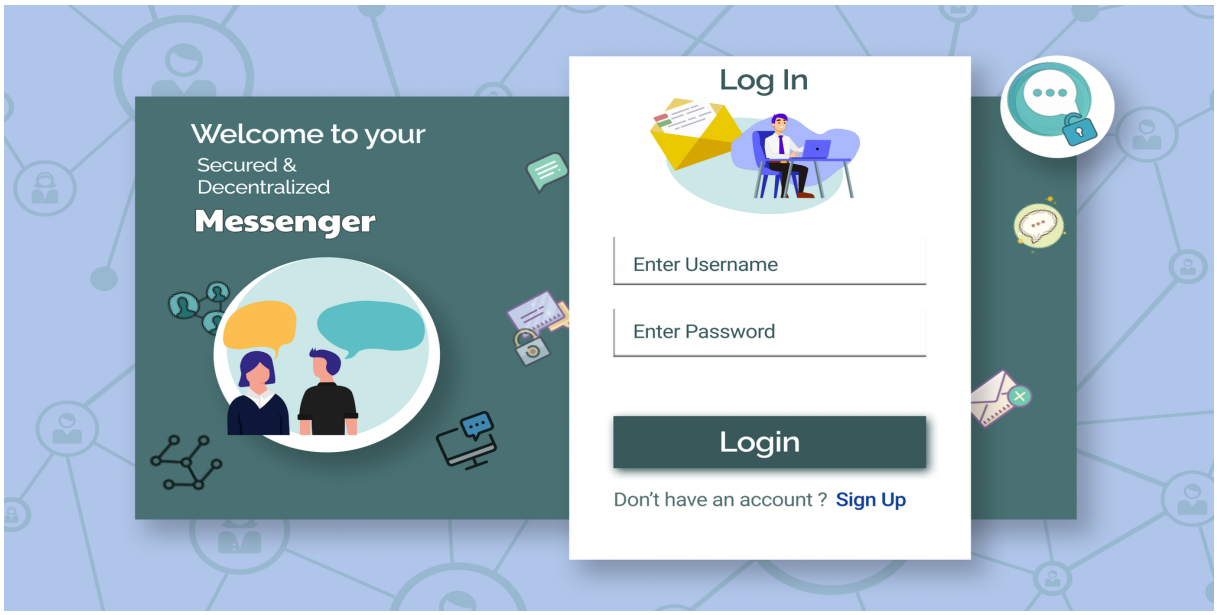

Figure 4.14: login page

4.4.3 Third step: contact request

In order for a client to communicate with another client, let's say Bob wants to communicate with Alice.

- Bob will search for Alice's profile in the blockchain. Once he finds the profile, and upon pressing the (add contact button) a contact request will be sent to Alice while a mutual key will be generated by Bob, this key will be stored in his mutual key index.
- If the contact request is accepted by Alice, then an index with Alice's name will be created in Bob's personal space, this index contains all the messages meant for Alice, simultaneously the contact Alice will be added to Bob's contact list. Same thing happens in Alice's side.

  For the mutual key, it will be sent as an encrypted message with the public key of Alice and it will be put in the index of mutual keys on the side of Alice.

After this, both users can exchange messages as shown next:

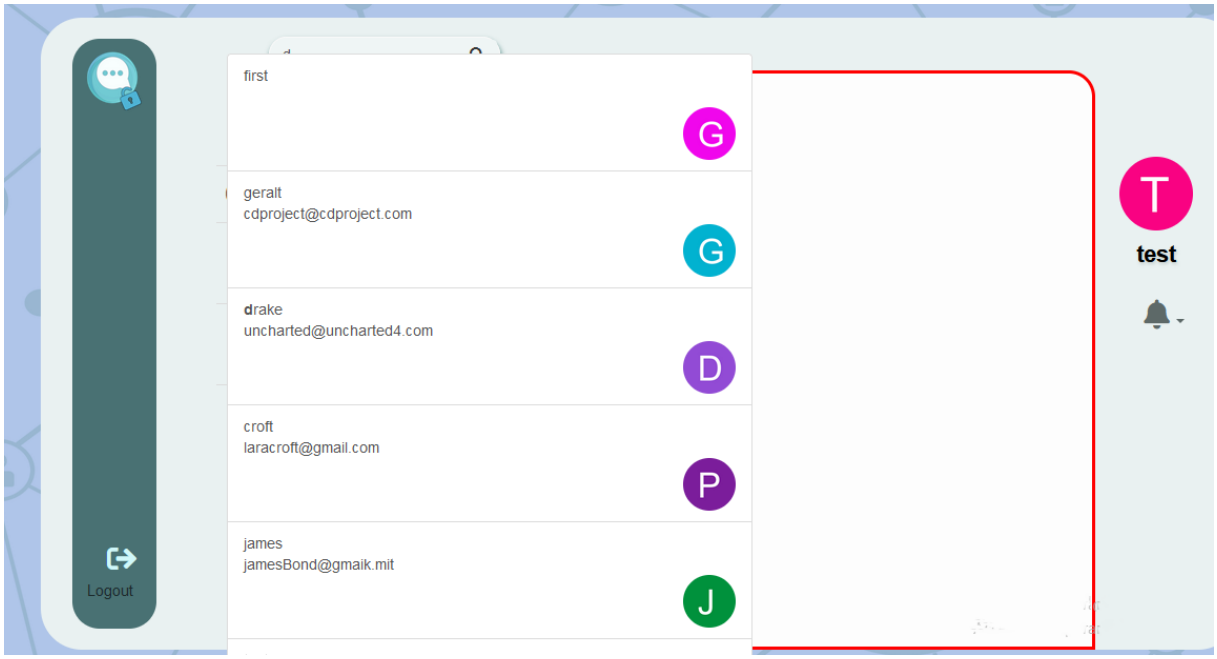

Figure 4.15: Search for other contact

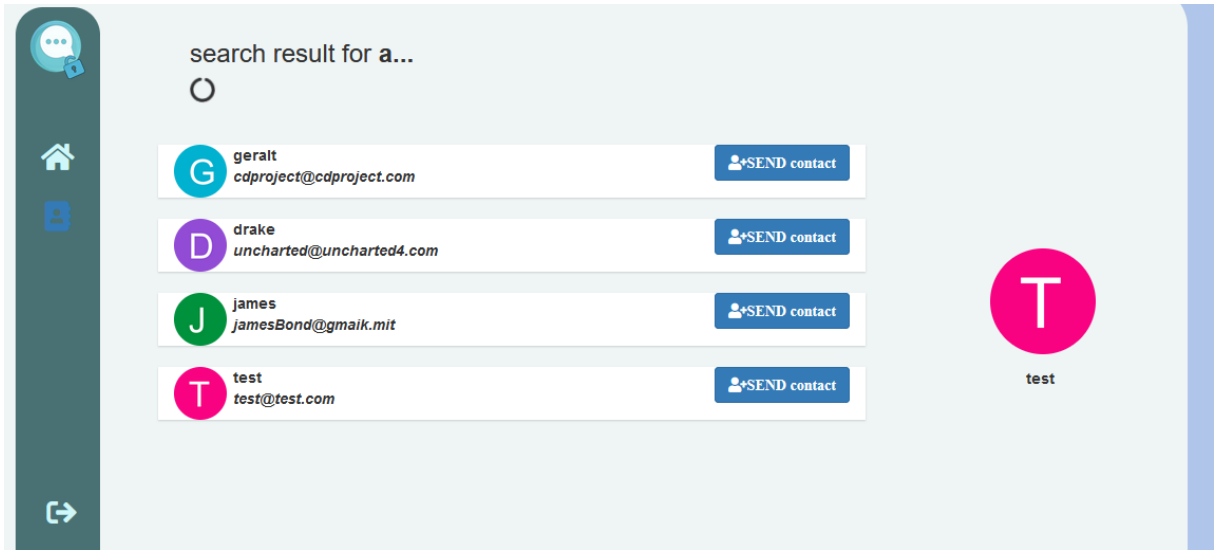

Figure 4.16: Search result and send request button

4.4.4 Forth step: messages exchange

We shall continue using Bob and Alice in our demonstration.
Now let's say that Bob wants to send a message to Alice. for the message to be correctly sent , the following steps must be followed.

- Bob choose Alice as his corespondent.
- Bob submits his message.
- After that, the message will be encrypted by Alice's public key and stored. At the same time, a link will be generated and stored in Alice's index, this index is encrypted by the mutual key that they both share.

Now we'll see how Alice receives these messages from Bob in details.

- Access to the index that contains her name, decrypt the file with the mutual key.
- Access to all messages meant for her form Bob (encrypted messages).
- Decrypt the messages with her private key, and read them.

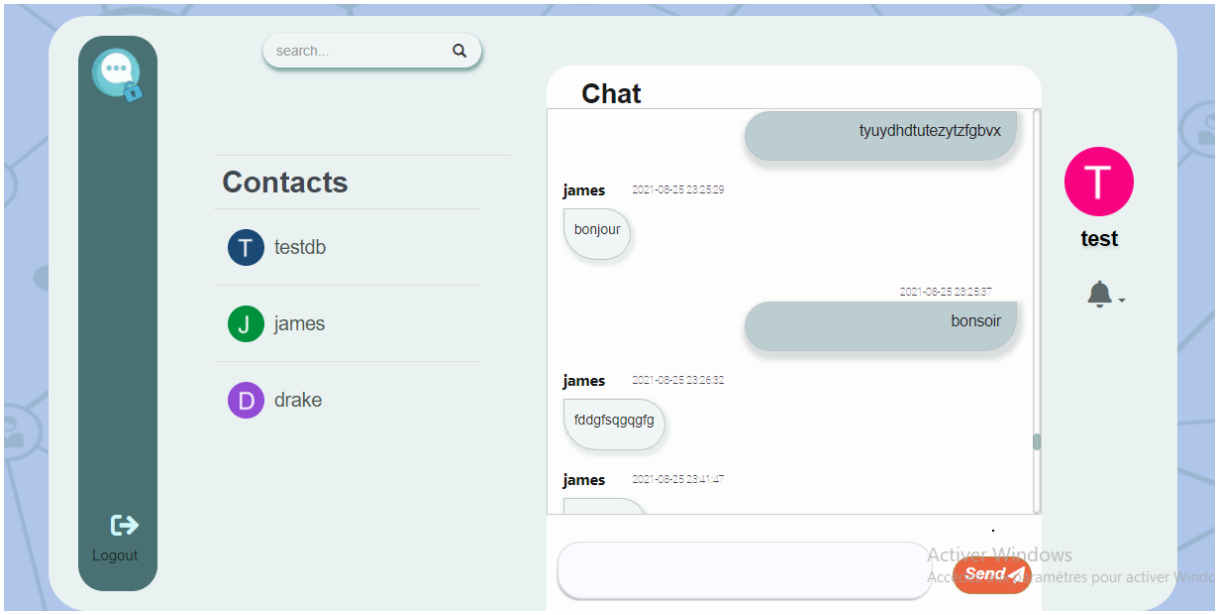

Figure 4.17: Messages exchange

4.5 Conclusion

In this last chapter we tried to implement what we researched and studied in the previous chapters as well as bringing our conception into life. We managed to make a decentralized messaging app on the Stacks platform, although we stopped mid progress and didn't finish it duo to many issues we managed to make another app anyway which was a simulation app to showcase all the necessary objectives that we initially wanted as we demonstrated how our conception should look and operate with screenshots from our final work. We simulated both the blockchain ledger and the cloud storage services with the different data structures proposed in the conception.

General conclusion

At the end, we really hope that we managed to achieve or deliver the objective we desired and set ourselves at the beginning: the making of a secured messenger application based on decentralization technologies (on Block-chain technology along with decentralized cloud storage) allowing users to exchange messages safely, while maintaining full control of their data and identities. We hope that this research and application will be beneficial, profitable and useful in a way, as it brings something new giving its users a good experience while using a reliable and trust worthy messenger if we take into count the provided security and privacy.

In this brief, we have presented the application functionalities (message exchange, adding contacts,... Etc.) modeled with the UML language plus other explained images according to the conception and design we have chosen. For the development of our application we tried taking advantage of the Stacks platform and its offered features (securing interactions and user identities, with total control for each user over his own data). Even though we failed to finish a Stacks app on time, we managed to deliver a simulation of the Block-chain technology, which is a revolutionary technique that is secure, transparent and operates without a central control entity, to secure access to data in cloud storage, the authentication and registration processes, ensure decentralized name distribution and full identity management.

This work has led us to persistently confirm that it is possible to explore entirely new area of studies a completely new domain to both of us. A Work that also allowed us to learn and acquire knowledge in the field of messaging applications, security and decentralized technologies in particular those of Block-chain, and to enrich our knowledge

in making a research, software design and development too.

We know for a fact that our project can be improved a lot more as we only implemented the basics of a messenger application. We plan to complete the implementation on the Stacks platform. Then, the following features can be improved:

- Adding the possibility to share other types of files (audio, video ,... Etc.).
- Adding the option to like and react to other users messages.
- Adding the block feature.
- Adding the possibility to create group chats.
- Adding voice calls with video calls would be the cheery on top of the cake.

Bibliography

[1] "Chintan Bhatt" "Ruturaj Shelat" "Nirav Patel ". ""p. 'Extensiblc Messaging and l'resence protocol (XMPP): Core""". **in**: *"" ""* ("2015"), **page** 4.

[2] *"The 50-year history of email"*. URL: `https://www.thinkautomation.com/histories/the-50-year-history-of-email/`.

[3] "A Survey on Blockchain Technology Concepts, Applications, and Issues". **in**: *SN Computer Science* ().

[4] Resul Kara Abdullah Talha Kabakus1. "Survey of Instant Messaging Applications Encryption Methods". **in**: *European Journal of Science and Technology* 2.4 (june 2015), **pages** 112–117.

[5] *Adobe XD*. **june** 2021. URL: `https://en.wikipedia.org/wiki/Adobe_XD`.

[6] *American graphics Institute*. URL: `https://www.agitraining.com/adobe/illustrator/classes/what-is-adobe-illustrator`.

[7] Antonio. *What is Blockstack?* **april** 2021. URL: `https://academy.bit2me.com/en/what-is-blockstack-stx/`.

[8] Madelyn Bacon. *"end-to-end encryption (E2EE)"*. July 2015. URL: `https://searchsecurity.techtarget.com/definition/end-to-end-encryption-E2EE`.

[9] Trushar Barot **and** Eytan Oren. *"A Brief History of Chat Apps"*. November, 2015. URL: `https://towcenter.gitbooks.io/guide-to-chat-apps/content/introductionthe_dawn_of/a_brief_history.html`.

[10] *Blockstack Whitepaper: A New Decentralized Internet*. **november** 2018. URL: `https://blog.blockstack.org/blockstack-whitepaper-part-1/`.

[11] *Digital Signature in Blockchain - DZone Security*. **july** 2018. URL: https://dzone.com/articles/digital-signature-2.

[12] Adrien Béraud dit **andothers**. *OpenDHT, table de hachage distribuée de nouvelle génération*. **january** 2018. URL: https://blog.savoirfairelinux.com/fr-ca/2015/opendht-une-table-de-hachage-distribuee-au-coeur-de-ring/.

[13] Elad. Elrom. *The Blockchain Developer A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects*. eng. 1st ed. 2019. Berkeley, CA: Apress, 2019. ISBN: 1-4842-4847-3.

[14] "Robert E. Endeley". ""End-to-End Encryption in Messaging Services and National Security—Case of WhatsApp Messenger"". **in**: *"Journal of Information Security"* "9".95-99 ("2018"), **page** 5.

[15] *Establishing P2P connections with Jami*. **june** 2019. URL: https://jami.net/establishing-peer-to-peer-connections-with-jami/.

[16] *Figma*. **july** 2021. URL: https://fr.wikipedia.org/wiki/Figma.

[17] Jake Frankenfield. *Cryptocurrency*. **may** 2021. URL: https://www.investopedia.com/terms/c/cryptocurrency.asp.

[18] DAN GOODIN. *The Privacy Problems with Mobile Messaging Apps*. Jun 03, 2014. URL: https://www.mcafee.com/blogs/consumer/mobile-and-iot-security/viber-app-sends-data-unencrypted/.

[19] Joshinav. *8 blockchain consensus mechanisms you should know about*. **april** 2019. URL: https://www.allerin.com/blog/8-blockchain-consensus-mechanisms-you-should-know-about.

[20] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah **and** Hicham Lakhlef. *Internet of things security: A top-down survey*. **march** 2018. URL: https://www.sciencedirect.com/science/article/abs/pii/S1389128618301208.

[21] Kirankalyan Kulkarni. *Learn Bitcoin and Blockchain: understanding Blockchain and Bitcoin architecture to build decentralized applications*. Packt Publishing Ltd., 2018.

[22] Tiana laurence. *Blockchain for dummies*. 2018.

[23] Francisco López-Fuentes. "Decentralized Online Social Network Architectures". **in**: (**january** 2019), **pages** 85–100.

[24] By McAfee. *Study shows which messengers leak your data, drain your battery, and more*. 10/26/2020. URL: https://arstechnica.com/information-technology/2020/10/study-shows-which-messengers-leak-your-data-drain-your-battery-and-more/.

[25] *Node.js*. **september** 2021. URL: https://fr.wikipedia.org/wiki/Node.js.

[26] Esther Pacitti, Reza Akbaranian **and** Manal El-Dick. 2012, **page** 421.

[27] Nitin Pandit. *What and why react.js*. URL: https://www.c-sharpcorner.com/article/what-and-why-reactjs.

[28] ICRC the engine room. *Messaging Apps*. updated in January 2017. URL: https://library.theengineroom.org/messaging-apps/.

[29] operators of federated services **and** developers of software programs that use the XMPP standard. *"Manifesto, A Public Statement Regarding Ubiquitous Encryption on the XMPP Network"*. Last Updated: 2014-03-21. URL: https://github.com/stpeter/manifesto/blob/master/manifesto.txt.

[30] Sidra. *The Blockstack Internet*. **february** 2019. URL: https://medium.com/block360-labs/the-blockstack-internet.

[31] stacks. *Gaia,Decentralized storage architecture for off-chain data*. URL: https://docs.stacks.co/build-apps/references/gaia.

[32] Martin Westerkamp, Sebastian Göndör **and** Axel Küpper. "Tawki: Towards Self-Sovereign Social Communication". **in**: (**january** 2019). DOI: 10.13140/RG.2.2.23522.99527.

[33] *What are cryptographic hash functions?: Synopsys*. **january** 2019. URL: https://www.synopsys.com/blogs/software-security/cryptographic-hash-functions/.

[34] Emma Witman. *What is GitHub? How to start using the code hosting platform that allows you to easily manage and collaborate on programming projects*. **june** 2021. URL: https://www.businessinsider.fr/us/what-is-github.

[35] network world. *Secure SIP protects VoIP traffic.* MAY 1, 2006. URL: https://www.networkworld.com/article/2311252/secure-sip-protects-voip-traffic.html#:~:text=Secure%5C%20SIP%5C%20is%5C%20a%5C%20security,communications%5C%20from%5C%20eavesdropping%5C%20or%5C%20tampering..

ملخص

في أيامنا هذه ، أصبحت منصات الاتصال وخاصة تطبيقات المراسلة بجميع أشكالها جزءًا أساسيًا في حياتنا ، ليس مجرد رفاهية بل ضرورة ، ومع النمو الهائل لهذه التطبيقات بين جميع اصناف المجتمع، بدأ الناس يقلقون بشأن خصوصيتهم وحماية بياناتهم ، حيث تقوم غالبية شركات تطبيقات المراسلة هذه بإدارة وتخزين بيانات المستخدم باستخدام وحدة مركزية ، مما يهدد البيانات المختلفة ويعرضها للخطر. تحتوي هذه البنى المركزية على العديد من نقاط الضعف التي أدت إلى العديد من التسريبات في بيانات المستخدمين ، والمشكلات والتعارضات فيما يتعلق بسلامة المستخدمين وخصوصيتهم. كل هذا حفز مجتمع البحث على البحث عن بديل هو بنى غير مركزية بهدف رئيسي وهو توفير السلامة مع منح المستخدمين السيطرة الكاملة على بياناتهم ،علاقاتهم وأنشطتهم.

يهدف مشروعنا الى تطوير برنامج مراسلة آمن لامركزي جديد يعتمد على تقنية Block-chain ، وهي تقنية ثورية لا تتطلب وحدة مركزية للإدارة والتحكم في كل شيء ، لتأمين الوصول إلى البيانات في مساحة التخزين ومصادقة الدخول والتسجيل والإدارة المثالية لهوية المستخدمين ، من أجل ضمان الامن المعلوماتي. لذلك ، سوف نقدم منصة. لذلك ، سوف نقدم منصة Stacks التي اخترناها في البداية لتطوير تطبيقنا عليها مع الاستفادة من وظائفها بالإضافة إلى تمثيل تطبيق محاكاة و الذي سيوضح كيف يجب أن يعمل برنامج المراسلة الامن و الغير المركزي اللذي اردنا الوصول اليه. سبب تطويرنا لتطبيق محاكاة هو أننا لم نكن محظوظين بما يكفي من الوقت لإنهاء تطبيقنا على منصة Stacks .

كلمات مفتاحية

تطبيقات المراسلة ، خصوصية ، حماية البيانات ، الغير مركزي ، المركزية . . .

Abstract

De nos jours, les plates-formes de communication, en particulier les applications de messagerie, sont devenues un élément essentiel de nos vies, non seulement un luxe mais aussi une nécessité, et avec l'énorme croissance de ces applications parmi tout le monde, les gens ont commencé à s'inquiéter de leur vie privée et de la sécurité de leurs données,comme la majorité de ces applications de messagerie, les entreprises gèrent et stockent les données des utilisateurs de manière centralisée, ce qui les menace et les met en danger. Ces archi-

tectures centralisées présentent de nombreuses fragilités qui ont conduit à de nombreuses fuites de données d'utilisateurs, à des problèmes et à des conflits concernant la sécurité des utilisateurs et leur vie privée. Tout cela a motivé la communauté des chercheurs à chercher à développer une alternative, à savoir des architectures décentralisées dont l'objectif principal est de fournir une sécurité tout en donnant aux utilisateurs un contrôle total sur leurs propres données, relations et activités

L'objectif de ce travail est de développer une nouvelle application de messagerie sécurisé décentralisé basé sur la technologie de la "Blockchain" et le stockage décentralisé, des technologies révolutionnaires qui ne nécessitent pas d'entité centrale pour tout gérer et contrôler. Par conséquent, nous présenterons la plateforme Stacks que nous avons choisie initialement pour le développement de notre application en profitant de ses fonctionnalités ainsi qu'en représentant une application de simulation qui démontrera comment un messager décentralisé devrait fonctionner. la raison pour laquelle nous avons développé une application de simulation est que nous n'avons pas eu de chance car nous n'avons pas eu assez de temps pour terminer notre application sur la plateforme Stacks.

Mots clé: securité, Stacks, Blockchain, architecture centralisé, application de mesagerie, stockage décentralisé ...

Resume

Now days, Communication platforms especially Messaging applications with all their varieties have become an essential part in our lives, not just a luxury but a necessity , and with the enormous growth of these apps among everyone, people have started to worry about their privacy and the safety of their data, as the majority of these messaging applications companies manage and store user data centrally, which threatens and puts it at risk. This centralized architectures have many vulnerabilities which led to many leaks in users data, issues and conflicts regarding the safety of users and their privacy. all of this have triggered and motivated the research community to look to develop an alternative which is decentralized architectures with the main purpose of providing safety while giving users full control over their own data, relationships and activities.

The aim of this work is to develop a new decentralized secured messenger based on Block chain technology, which is a revolutionary technology that doesn't require central entity to manage and control everything, to secure data access in the storage space, authentication and registration processes and perfect identity management, in order to guarantee the users information. Therefore, we will present the Stacks platform that we have chosen initially for the development of our application taking advantage of its functionalities as well as representing a simulation application which will demonstrate how a decentralized messenger should function. The reason we developed a simulation app is that we weren't lucky as there wasn't enough time for us to finish our app on the stacks platform.

Key words: Messaging Applications, centralized architectures, decentralized architectures, block-chain, security, privacy ...