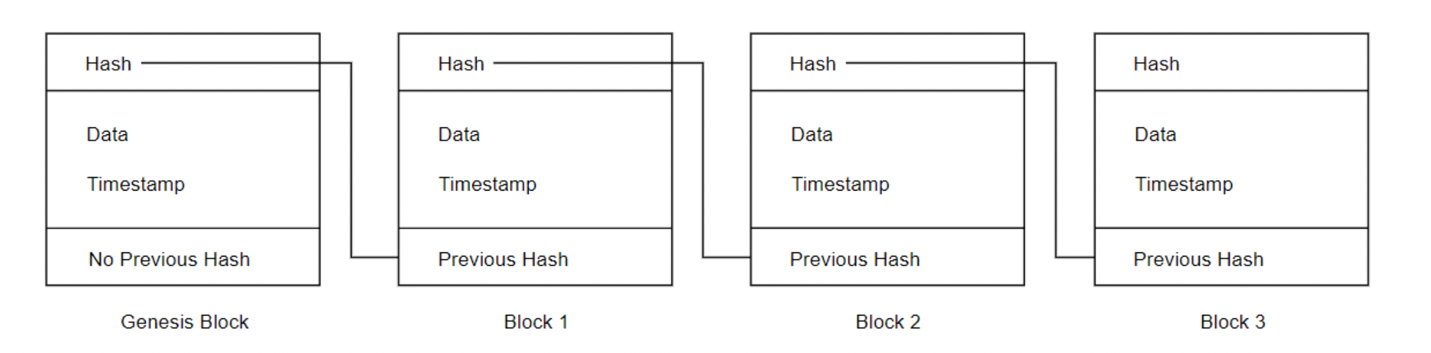
# CHAPTER 2: Introduction

## Overview

Sending messages over insecure channels has always been a security issue. Although there are many technologies to encrypt messages, but there are still ways to attack messages, such as: Eavesdropping, MITM, EFAIL. In many countries, such as Palestine, China and South Korea, citizens are persecuted by the government or the occupation. In addition, traditional applications maintain their data in a centralized database, which is also a problem. Our app ensures decentralization, immutability, censorship resistance and data security. data sent via Users are added directly to the blockchain and a global copy of the data is created in each node. Only legitimate users can do this. This data is accessed using a private key in the blockchain. It removes the need for a trusted intermediary. system complete Decentralizes and allows users to exchange messages securely.

**Disadvantage of central-based communication apps**

1. Central chatting apps collects meta information about the user
2. Confidentiality of a user can be compromised on the request of government
3. Single Point of Failure (SPF): If a single node fails then whole application can be compromised.

****

Blockchain is a shared, distributed ledger that facilitates the process of recording transactions and tracking assets in a network. It provides zero intermediary or 3rd party also achieved the security basics, confidentiality integrity and availability. The blockchain Network is collection of nodes and each node has a local copy of the blockchain

The benefits of blockchain-based communications and data storage technology are numerous: a. the stored data can be encrypted making it impervious to theft

b. the data is stored in a distributed manner, eliminating single-point failures

c. communications can be end-to-end encrypted making them spy-proof n

e. transactions and communications can be subject to incorruptible rules enforced by the blockchain itself

f. the system maintains a tamper and repudiation-proof history of past transactions and communications.

How to build the decentralize chat app

Ethereum is a platform allow the developers to develop dapp using smart contract,

Smart contract: an agreement, contract, or set of instruction that is deployed on a decentralized blockchain

* Cannot be altered “immutable”
* Automatically executed
* Every on sees the terms of agreement

Anyone can join the network can send transaction through the network and paying small amount of fees the fees may vary from one transaction to another. because it depends on the number of transactions sent in the same time, so paying fees for every message may not suitable for some users but for those who wants a high level of security our project is there right chouse.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

Messaging apps now have more global users than traditional social networks—which mean they will play an increasingly important role in the distribution of digital information in the future. Today, it’s common place for offices to use a messaging app for internal communication in order to coordinate meetings, share pitch decks, and plan happy hours. And with the latest bot technology, chat apps are becoming a hub for employees to do work in their apps without leaving the chat console. For many people, chat apps are a given part of their workday.

## 2.2 Historical Overview

### 2.2.1 INSTANT MESSAGING: CHILD OF THE 90’S

Chat apps (and their subsets, chat rooms) bring to remembrance images of the 1990s, with its dial-up internet and classic sitcoms, however, commercial chat apps date back to the 1980s. CompuServe released CB Simulator in 1980, and 1985 brought the launch of Commodore’s Quantum Link (also known as Q-Link). An online service, it allowed multiuser chat, email, file sharing, and games.

If Q-Link sounds familiar, that’s because it is: in 1991, the company changed its name to America Online (AOL). But AOL wouldn’t launch its signature product, AOL Instant Messenger (AIM), until 1997. In the meantime, the Vodafone GSM network enabled the first SMS in 1992. And in 1996, ICQ launched as the first widely-adopted instant messaging platform.

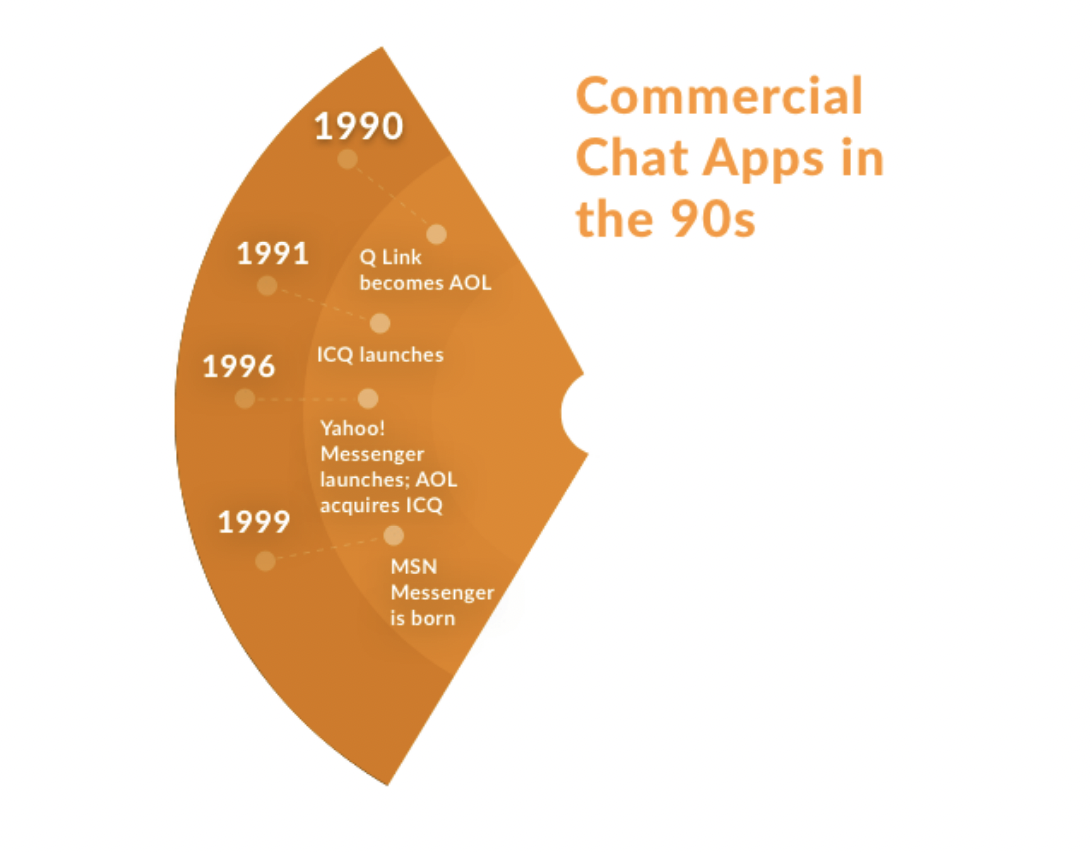


Figure 2.1: Commercial chat apps in 90’s

### 2.2.2. COMMERCIAL CHAT APPS IN THE 10s

### 

Figure 2.2: Commercial chat apps in 10’s

With the inception of smart phones, chat apps continued to thrive; in 2013, chat apps finally surpassed SMS in message volume. By 2015, WhatsApp alone hosted 30 billion messages per day; SMS logged only 20 billion. And in the summer of 2016, Facebook Messenger hit one billion users.

### 2.2.3. THE SLOW GROWTH OF ENTERPRISE CHAT APPS

### Despite developing around the same time, the history of enterprise chat apps is markedly different than the story of their consumer-facing counterparts. The very first enterprise chat apps do not enjoy the same place in our collective memory as AIM and ICQ.

### Early contender Yammer launched in 2008; Microsoft acquired the platform in 2012. Clear space began in 2006, rebranding several times until its rebirth as Jive six years later. Other enterprise chat programs, usually integrated with other social features like blogs and wikis, blipped in and out of existence.

### None of this initial crop of enterprise apps proved a runaway success, and many theories exist as to why later programs have overshadowed them. One thing is certain: these programs predated the rise of smart phones, and mobility certainly fomented the creation of second-generation commercial apps like WhatsApp and Snapchat.

### 2.2.5 INDUSTRY CHALLENGES

### 1. FRAGMENTATION: The social media landscape is entering a period of hyper fragmentation that may be a challenge to publishers: Facebook, Twitter, and Instagram continue to loom large, but social media managers can now launch official channels on roughly 10 chat apps with over 50 million monthly, active users each.

### 2. ANALYTICS: For organizations accustomed to robust, real-time data, the lack of good analytics tools for messaging apps remains a major deterrent to adoption. The challenge is twofold: Strong analytics dashboards take time to build, and many messengers are privacy centric by nature.

### 2.2.6 INDUSTRY OPPORTUNITIES

### 1. HIGHER ENGAGEMENT: Since many chat apps provide publishers with push notifications or chatbot experiences (programmable robots that converse with users), they can deliver significantly higher engagement rates.

### 2. AUDIENCE DEVELOPMENT: With billions of active users across multiple major chat apps, there is the opportunity in building large audiences fairly quickly on several platforms.

### 3. A CHANCE TO CONNECT WITH USERS IN A NEW WAY: Messaging apps offer a host of features not unavailable on social networks or other platforms. Programmers can creatively leverage these tools to socialize in new ways.

## 2.3 Related Work

### 2.3.1 WHAT THE FUTURE HOLDS

Predictions are always prone to inevitable ridicule and failure, but there are some that are worth making.

As government snooping, personal privacy, and security become issues for many people

globally, those living in countries where these are particular concerns will increasingly

look for platforms that enable them to both communicate securely and receive accurate

information, unfiltered by government censors.

This is the era to launch Dchat as it focuses on audience in whole the world

## 2.4 Summary

# Dchat is decentralize app that leverages on blockchain technology allowing its users communicate and share text. It offers a wonderful experience for keeping in touch with people you know in decentralized way.

# 

# CHAPTER 3: REQUIREMENTS, ANALYSIS, AND DESIGN

## 3.1 Overview

Communication is a means for people to exchange messages. It has started since the beginning of human creation. Distant communication began as early as 1800 century with the introduction of televisions, telegraphs and then telephony. Interestingly enough, telephone communication stands out as the fastest growing technology, from fixed lines to mobile wireless, from voice call to data transfer. The emergence of computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much efforts has been drawn towards consolidating the device into one and therefore indiscriminate the service. Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years, but the acceptance was quite recent. the user application runs on any PC on the network. To start chatting the user should create wallet and enter the wallet address and private key to start private chatting.

## 3.2 Requirements Specifications

# Table 3.1: Requirement Specification

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RQ.ID | NAME | FUNCTIOANL | NONFUNCTIOANL | DESCRIPION | PRIORITY | ACTOR |
| 1 | User registration | Yes |  | Functionality for user to create account | high | user |
| 2 | Login | Yes |  | Functionality for user to get access | high | user |
| 3 | Add friend | yes |  | Functionality to be friend with each other | Medium | User |
| 4 | Friend list | Yes |  | Functionality to see list of friends | High | User |
| 5 | Send message | Yes |  | To send message | High | user |
|  | Privacy | yes |  | User privacy | High | network |
|  | Performance | yes |  | Application performance must be better | High | network |

## 3.3 Use case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| Level 0 | Level 1 | Level 2 | actor |
| Chat application | Authentication system | Register Login | user |
| Chat application | contact form | Friend list, Add friend | user |
| Chat application | Chat form | Send message | user |

# 

### 3.3.2 Authentication Service

# 

# Figure 3.1: Use Case Diagram of Authentication Service

### 3.3.3 Contact Form

Figure 3.2: Use Case Diagrams of Contact Forms

### 3.3.4 Chat Form

**Figure 3.3: Use Case Diagram of Chat Form**

# 

## 3.4 System Design

### 3.4.1 User Interface Design

### 

**Figure 3.4: introduction page**

### 

**Figure 3.5: registration page**

# 

**Figure 3.6: friend list page**

# CHAPTER 4: IMPLEMENTATION

## 4.1 Overview

This document provides the requirement for the design and implementation of a chat application. This project will create a chat application and allow users to be able to chat with themselves. Instant messaging solution will be proffered so that users will be able to communicate seamlessly and ensuring that even a novice can use this chatting application, thereby ensuring it is not too complex, so that it can cut across a wide range of audience.

## 4.2 Project Scope and features

1. Dchat application is going to be a text communication software; it will be able to communicate between two computers using blockchain technology.
2. There is limitation on live chat as it does not support audio communication and media.
3. The easy usability breaks the complexity syndrome.

## 4.3 Methodology

### 4.3.1 Project Workability

### The user interacts using GUI. The GUI operates in four forms, which are contacts forms, introduction form, network form and chat forms. The contacts forms contain the list of all friends and network form contain the registration fields and the chat form will be used to chat with friends.

### 4.3.2Constraints & Limitations

### The system must be connected to the internet. This app does not have audio and video calling system. This app cannot send media.

### 4.3.5 Project risk management

### Identifying the necessary threat of the project either internal or external will help a great deal in ensuring the success of the project. To guide again risk significantly much brainstorming was carried out to factor in all facets of the project details.

## 4.4 Tools and Technology

**Software Analysis**

Brainstorming, Mind-Map

**Programming Language**

python, solidity, JavaScript, bootstrap, Json,

# REFERENCES

Decentralization is a method of organizing different parts of a system, to ensure that no single entity holds too much power. In the context of security, decentralization can be beneficial as it reduces the single point of failure. This means that without one centralized entity, it becomes harder for malicious actors to compromise the system, as it is spread over multiple entities.

Decentralized systems tend to be more secure as they have geographically diverse nodes and make it more difficult to access the entirety of the information in the system. Data is often spread across multiple locations, making it impossible to access the entire system from one single point of attack. Attacks that target one node or data points do not undermine the entire system since data can continue to be accessed from other points.

Additionally, decentralization introduces a robust layer of security due to cryptographic methods such as public key infrastructure, digital signatures, and cryptographic hashing. These cryptographic methods are used to ensure the integrity and secure transmission of data across the network. Such techniques allow users of the system to verify the data being exchanged is valid and from the correct source.

Lastly, decentralization encourages users to be more aware of their systems’ security. As users of the system are considered individually, each node can be responsible for their own security and thus secure the entire system as a unit. Users also have greater control over their data, allowing them to take further measures to ensure their data is secure.

Overall, decentralization provides an extra layer of security, allowing systems to be more secure, resilient, and reliable. By spreading and protecting data across the network, the risk of malicious actors is greatly reduced.

A smart contract is a computer protocol (a set of instructions) that enables two or more parties to securely exchange information and/or assets without the need for a third party. Smart contracts are usually built on top of a blockchain technology like Ethereum. They operate by capturing and executing the terms of a contract using a programmable code that is stored, replicated, and supervised by a network of computers on the blockchain.

Smart contracts are similar to traditional contracts except they are self-executing, meaning that they automatically manage and enforce the terms of the contract instead of relying on a third party. For example, if a buyer and seller agree to exchange goods for payment, the buyer can transfer funds to a designated address and the smart contract will verify the funds and release the goods to the seller. This eliminates the need for third parties or intermediaries, so transactions are faster and more secure. Smart contracts are also transparent and immutable, meaning that they cannot be changed arbitrarily and all parties have access to the same information.

# Testing Validation and Discussion:

# Conclusion:

# References:

URL to how the addresses are generated

https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch04.html

Elliptic Curve Cryptography Explained

Elliptic curve cryptography is a type of asymmetric or public-key cryptography based on the discrete logarithm problem as expressed by addition and multiplication on the points of an elliptic curve.

The elliptic curve cryptography (ECC) **does not directly provide encryption** method. Instead, we can design a **hybrid encryption scheme** by using the **ECDH** (Elliptic Curve Diffie–Hellman) key exchange scheme to derive a **shared secret key** for symmetric data encryption and decryption.

* **calculateEncryptionKey**(pubKey) --> (sharedECCKey, ciphertextPubKey)
  1. 1.

Generate **ciphertextPrivKey** = *new* ***random*** *private key*.

* 1. 2.

Calculate **ciphertextPubKey** = ciphertextPrivKey \* G.

* 1. 3.

Calculate the ECDH shared secret: **sharedECCKey** = pubKey \* ciphertextPrivKey.

* 1. 4.

Return both the **sharedECCKey** + **ciphertextPubKey**. Use the **sharedECCKey** for symmetric encryption. Use the randomly generated **ciphertextPubKey** to calculate the decryption key later.

* **calculateDecryptionKey**(privKey, ciphertextPubKey) --> sharedECCKey
  1. 1.

Calculate the the ECDH shared secret: **sharedECCKey** = ciphertextPubKey \* privKey.

* 1. 2.

Return the **sharedECCKey** and use it for the decryption.

The above calculations use the same math, like the **ECDH** algorithm (see the [previous section](file:////asymmetric-key-ciphers/ecdh-key-exchange)). Recall that EC points have the following property:

* (***a*** \* **G**) \* ***b*** = (***b*** \* **G**) \* ***a***

Now, assume that ***a*** = privKey, ***a*** \* **G** = pubKey, ***b*** = ciphertextPrivKey, ***b*** \* **G** = ciphertextPubKey.

The above equation takes the following form:

* pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey = **sharedECCKey**

This is what exactly the above two functions calculate, directly following the **ECDH key agreement** scheme. In the hybrid encryption schemes the encapsulated **ciphertextPubKey** is also known as "**ephemeral key**", because it is used temporary, to derive the symmetric encryption key, using the ECDH key agreement scheme.

the **encryption key** (derived from the public key) and the **decryption key** (derived from the corresponding private key) **are the same**. This is due to the above discussed property of the ECC: pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey. These keys will be used for data encryption and decryption in an integrated encryption scheme. The above output will be different if you run the code (due to the randomness used to generate ciphertextPrivKey, but the encryption and decryption keys will always be the same (the ECDH shared secret).

The above demonstrated mechanism for generating a shared ephemeral secret key, based on a ECC key pair, is an example of **KEM** (key encapsulation mechanism), based on the ECC and ECDH.

Src =https://cryptobook.nakov.com/asymmetric-key-ciphers/ecc-encryption-decryption