

T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

**BLOCKCHAIN BASED
SECURE MESSAGING APPLICATION**

MUHAMMED BEDIR ULUÇAY

**SUPERVISOR
PROF. DR. İBRAHIM SOĞUKPINAR**

**GEBZE
2023**

**T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT**

**BLOCKCHAIN BASED
SECURE MESSAGING APPLICATION**

MUHAMMED BEDİR ULUÇAY

**SUPERVISOR
PROF. DR. İBRAHİM SOĞUKPINAR**

**2023
GEBZE**



GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 18/01/2023 by the following jury.

JURY

Member

(Supervisor) : Prof. Dr. İbrahim Soğukpınar

Member : Prof. Dr. Yusuf Sinan Akgul

ABSTRACT

Sending messages has always been a security problem in relation to unsecured channels. Even though there are a number of techniques to encrypt messages, there are still opportunities to attack messages. Furthermore, the traditional application manages their data on a centralized database that is also a concern. In this graduation project the application which ensures decentralization, immutability, censorship resistance and data security. The data that users transmit will be immediately added to the blockchain, creating a global copy of the data in every node. By using their private key on the blockchain, only the authorized users can access that data. It does away with the necessity for reliable middlemen. Users are able to securely send messages thanks to the system, which is completely decentralized.

Keywords: Block Chain, Decentralized, Cryptograph, Message Send/Receive.

ÖZET

Mesaj göndermek, almak ve iletmek güvenli olmayan kanallarla ilgili olarak her zaman bir güvenlik sorunu olmuştur. Mesajları şifrelemek için bir takım teknikler olsa da mesajlara saldırmak için hala fırsatlar vardır. Ayrıca geleneksel uygulama, verilerini merkezi bir veri tabanında yönetir ve bu da endişe vericidir. Bu bitirme projesinde merkeziyetçilik, değişmezlik, inkar edilememezlik ve veri güvenliğini sağlayan uygulama oluşturacağız. Kullanıcıların传递 veriler anında blok zincirine eklenecek ve her düğümde verilerin küresel bir kopyası oluşturulacaktır. Blok zincirindeki özel anahtarlarını kullanarak, yalnızca yetkili kullanıcılar bu verilere erişebilir. Aracılara olan ihtiyacı ortadan kaldırır. Tamamen merkezi olmayan sistem sayesinde kullanıcılar güvenli bir şekilde mesaj gönderip alabilmektedir.

Anahtar Kelimeler: Block Chain, Dağıtık sistemler, Criptografi, Mesaj Gönderme/Alma.

ACKNOWLEDGEMENT

Thank you to the Prof. Dr. İbrahim Soğukpınar for giving me the opportunity to be curious and learn a new important technology.

Muhammed Bedir ULUÇAY

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or	
Abbreviation	Explanation
P2P	Peer-to-peer

CONTENTS

Abstract	iv
Özet	v
Acknowledgement	vi
List of Symbols and Abbreviations	vii
Contents	ix
List of Figures	x
1 INTRODUCTION	1
1.1 Motivations And Aim	2
2 BLOCKCHAIN	3
2.1 Blockchain	3
2.1.1 Advantages	3
2.1.2 Blockchain Properties	4
3 RELATED WORK	5
3.1 Similar Projects	6
4 IMPLEMENTATION	7
4.1 Project Design	7
4.2 Class Diagrams	8
4.2.1 Blockchain Class Diagram	8
4.2.2 Network Class Diagram	8
4.2.3 Other Classes	10
4.3 Project Implementation	10
4.3.1 Communication network setting	10
4.3.2 Creating a wallet	11
4.3.3 Creating a Blockchain Database	11
4.3.4 Printing a Blockchain Database	11
4.3.5 Key Generation	12
4.3.6 Send Message	12

4.3.7	Encrypted Message	13
4.3.8	Decrypted Message	13
4.4	Ethereum Smart Contract Project Implementation	15
4.4.1	Tools	15
4.4.2	Deploy Ethereum Network	16
4.4.2.1	Adjust Settings	16
4.4.2.2	Deploy Contract Command	16
4.4.2.3	Ethereum Network - Etherscan	17
4.4.2.4	Send Transaction	17
4.4.2.5	Check Transaction	18
5	Local Blockchain Implementation	19
5.1	Usage Of Program	20
6	Conclusions	24
	Bibliography	25

LIST OF FIGURES

1.1	Blockchain Structure	1
1.2	Block Structure	1
3.1	Diagram	5
4.1	Message Path	8
4.2	Blockchain Class Diagram	9
4.3	Network Class Diagram	9
4.4	Other Classes	10
4.5	Network Port Command	10
4.6	Wallet Creation Command	11
4.7	Badger Blockchain Database Command	11
4.8	Printing Chain Command	11
4.9	Printing Chain Command	12
4.10	Message Sending Command	12
4.11	Printing Chain Command	13
4.12	Printing Chain Command	14
4.13	Deploy Settings	16
4.14	Deploy Settings	16
4.15	Deploy Command	16
4.16	Ethereum Network	17
4.17	Transaction	17
4.18	Check Transaction	18
5.1	Command	19
5.2	Image 1	20
5.3	Image 2	20
5.4	Image 3	21
5.5	Image 4	21
5.6	Image 5	22
5.7	Image 6	22
5.8	Image 7	23
5.9	Image 8	23

1. INTRODUCTION

Blockchain was designed to make transactions more secure. It is an inherent approach to achieve confidentiality, data integrity, authorization and relying on cryptography to achieve tamper-resistance. It can be used to achieve the secure communication and integrity of data. It provides the feature of zero participation of 3rd party in the transaction. All network participants must reach a consensus to validate transactions in a secure way, and previous records cannot be changed. A very high cost must be spent if someone wants to alter previous records. External attackers would have to gain access to every computer in the network that hosts the blockchain database at the same time to manipulate it, which is as practically impossible. ??.

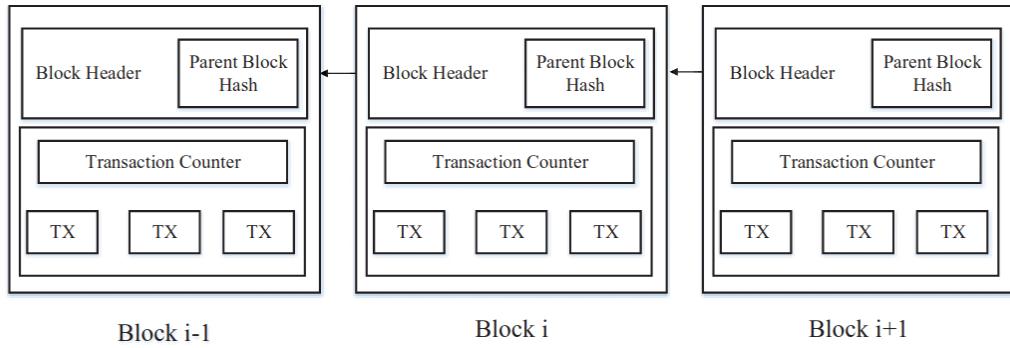


Figure 1.1: Blockchain Structure

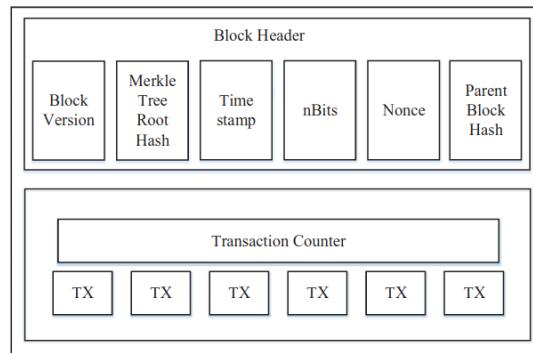


Figure 1.2: Block Structure

1.1. Motivations And Aim

Due to its enormous commercial potential and use in a variety of applications, including cryptocurrencies, blockchain has drawn a lot of interest from investors and engineers. P2P systems help solve a number of challenges beyond the scope of traditional client-server methods, but these qualities also bring new concerns, such as how to establish trust relationships in P2P networks.

Secure messaging is important for protecting sensitive information from unauthorized access and cyber attacks. However, many individuals and organizations still use unsecured platforms, which can lead to data breaches and other serious consequences. It is important to use secure messaging platforms to protect personal and professional communications.

The aim of this article is to present how a messaging application can be implemented using blockchain technology and cryptography.

2. BLOCKCHAIN

2.1. Blockchain

Blockchain is a distributed ledger technology that enables secure and transparent record-keeping. It is the underlying technology behind cryptocurrencies like Bitcoin, but it has many other potential uses as well. A blockchain is a decentralized system that allows multiple parties to share and maintain a single, tamper-proof ledger. Each block in the chain contains a record of multiple transactions, and once a block is added to the chain, the information it contains cannot be altered. This creates a secure and transparent system for recording and tracking transactions, without the need for a central authority or intermediary. Blockchain technology has the potential to revolutionize industries from finance and banking to supply chain management, voting systems, and more. It can be used to create more efficient, secure and transparent systems, enabling the secure and transparent recording of anything that has value.

2.1.1. Advantages

- Decentralization: Blockchain technology eliminates the need for a central authority or intermediary, which makes it more secure and transparent.
- Immutable: Once a block is added to the chain, the information it contains cannot be altered, ensuring the integrity of the data.
- Transparency: Blockchain allows all parties to access and view the same information, making it transparent and easy to audit.
- Security: Blockchain uses advanced cryptography to secure transactions, making it resistant to hacking and fraud.
- Cost-effective: Blockchain can reduce the costs associated with middlemen and intermediaries, making transactions cheaper and more efficient.
- Automation: Smart contracts, which are self-executing contracts with the terms of the agreement written into the code, can automate complex business processes.
- Faster transaction: Blockchain-based transactions are faster and more efficient than traditional methods.

- Increased Efficiency: Blockchain technology can be used to improve efficiency and reduce costs in various industries, such as supply chain management, health-care, and real estate.
- Traceability: Blockchain allows tracking of products and assets throughout their lifecycle, enabling transparency and traceability.
- Increased Accessibility: Blockchain technology enables financial services to be extended to those who were previously unbanked or underbanked.

2.1.2. Blockchain Properties

- **Public key infrastructure:** It uses public/private key encryption and data hashing to store and exchange data safely.
- **Distributed ledgers:** There is no central authority to hold and store the data, it removes the single point of failure
- **Peer to peer Network:** The communication is based on the P2P network architecture and inherits the decentralized characteristics.
- **Cryptography:** Numerous cryptographic methods, including hash functions, Merkle trees, public and private keys, are used by blockchain. It is challenging to modify the blockchain; in order to do so, more than 51 percent of the participants must be simultaneously attacked.
- **Consensus Algorithm:** The protocols that the network nodes adhere to in order to validate the distributed ledger. Consensus algorithms are made to ensure dependability in a network with several faulty nodes. The transactions are verified by a consensus among the nodes. Performance, scalability, latency, and other Blockchain factors are significantly impacted by the consensus algorithm chosen. Consensus algorithms must be robust to errors.
- **Transparency:** Every transaction is accessible to anybody with system access. An entry is immediately discarded if it cannot be confirmed. As a result, the data is completely transparent. A blockchain's nodes are identified by their distinct addresses. A user has the option to give other users identification documentation.

3. RELATED WORK

Modern messaging applications are centralized, which opens up a wide range of possible technological problems. Data demonstrates that network issues and frequent breakdowns are commonplace in apps like WhatsApp and Facebook Messenger. The enormous, centralized operational system is the primary cause of catastrophic accidents. It's challenging to maintain the seamless operation of a centralized system with hundreds of millions of users. For instance, a modest app update frequently throws a centralized messaging system into disarray. Because a centralized data processing system manages the update procedure, the entire network must concurrently update its mechanism.

In terms of network topology, blockchain messaging apps are the direct opposite of centralized messengers. To enable user communication through blockchain transactions, these applications make use of blockchain technology. Distributed ledger technologies, or blockchains, are fundamentally a decentralized network architecture. Blockchain messaging applications employ several autonomous network nodes rather than a single centralized system to process traffic.

Blockchain messengers are substantially more reliable because to their decentralized software structure. Transaction processing is distributed among several nodes, thus the network is not overloaded. Additionally, because blockchain apps don't rely on centralized servers, they may be more secure. The network won't be destroyed by a few nodes being hacked. Blockchain technology also emphasizes protecting user privacy. Blockchain messaging applications don't save user information or previous messages on corporate servers. 4.1.

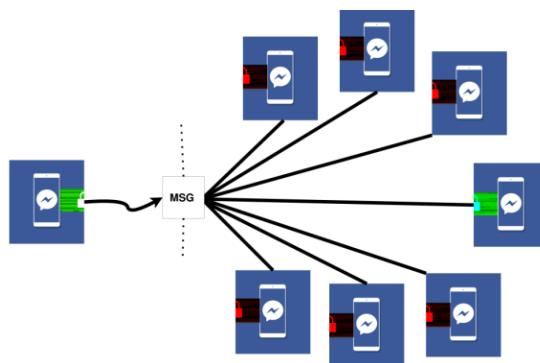


Figure 3.1: Diagram

3.1. Similar Projects

- Dust: One of the First Blockchain Messaging Apps
- Status: Status is a highly innovative Ethereum-based blockchain messenger that provides users with multiple blockchain features. Users can communicate with each other through the app's secure messaging feature.
- Wispr: Wispr uses blockchain technology to provide users with complete anonymity. This blockchain messenger app enables users to engage in texting, voice calls, video calls, and data transfer.
- E-Chat: It is possible to use IPFS technology and a peer-to-peer messaging service with the e-Chat software. No central data storage is provided, unlike other typical messaging networks.
- BChat: The decentralized application is implemented on Ethereum blockchain network with ReactJS.

4. IMPLEMENTATION

4.1. Project Design

The design plan for the blockchain-based secure messaging application will begin with a thorough analysis of the current state of messaging security and the ways in which blockchain technology can be utilized to address these issues. This will include researching existing blockchain-based messaging platforms and identifying the strengths and weaknesses of each.

In order to use the blockchain-based secure messaging application, users will first need to have a wallet address on the blockchain network that the application is built on. This wallet address will be used to store the data associated with the messages, as well as any cryptocurrency or tokens that may be used within the application. Users can create a wallet address by using a wallet software or by signing up to a cryptocurrency exchange that support creating a wallet.

In addition to the wallet address, users will also need two keys for encryption and decryption to use the messaging system. One key will be the user's private key, which will be used to encrypt messages before they are sent. The other key will be the recipient's public key, which will be used to decrypt messages when they are received. This is known as asymmetric encryption, it ensures that only the intended recipient can read the message, as the private key is only known by the user, and the public key is shared with the intended recipient.

Users will also have the option to create a unique private key for each conversation, this will ensure that the data is only accessible by the intended recipient, and it will increase the level of security and privacy.

The wallet address and the encryption keys are essential for the secure messaging application, they are used to store and encrypt the data, and it ensures that the data is only accessible by the intended recipients.

The only data that is stored on the blockchain is the encrypted message, and the wallet address of the sender and the recipient. This ensures that the data is tamper-proof and immutable, and it can be accessed by anyone with the correct wallet address and the private key to decrypt the message.

The keys used for encryption and decryption are not stored on the blockchain, they are generated and stored by the users on their devices, and the data stored on the blockchain is the encrypted message, and the wallet address of the sender and the recipient.

Message Path:

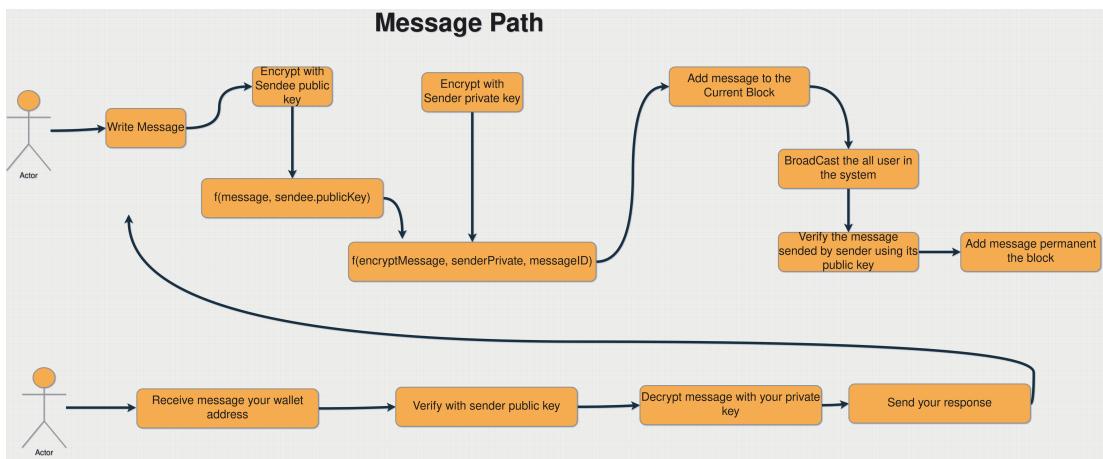


Figure 4.1: Message Path

Asymmetric encryption algorithms use two keys, a public key and a private key. The public key is used to encrypt the message and the private key is used to decrypt it.

When a user sends a message, they will use the recipient's public key to encrypt the message. The encrypted message is then sent to the recipient, who uses their own private key to decrypt the message. The private key is generated by the user and kept private, it's not stored on the blockchain.

4.2. Class Diagrams

4.2.1. Blockchain Class Diagram

4.2.

4.2.2. Network Class Diagram

4.3.

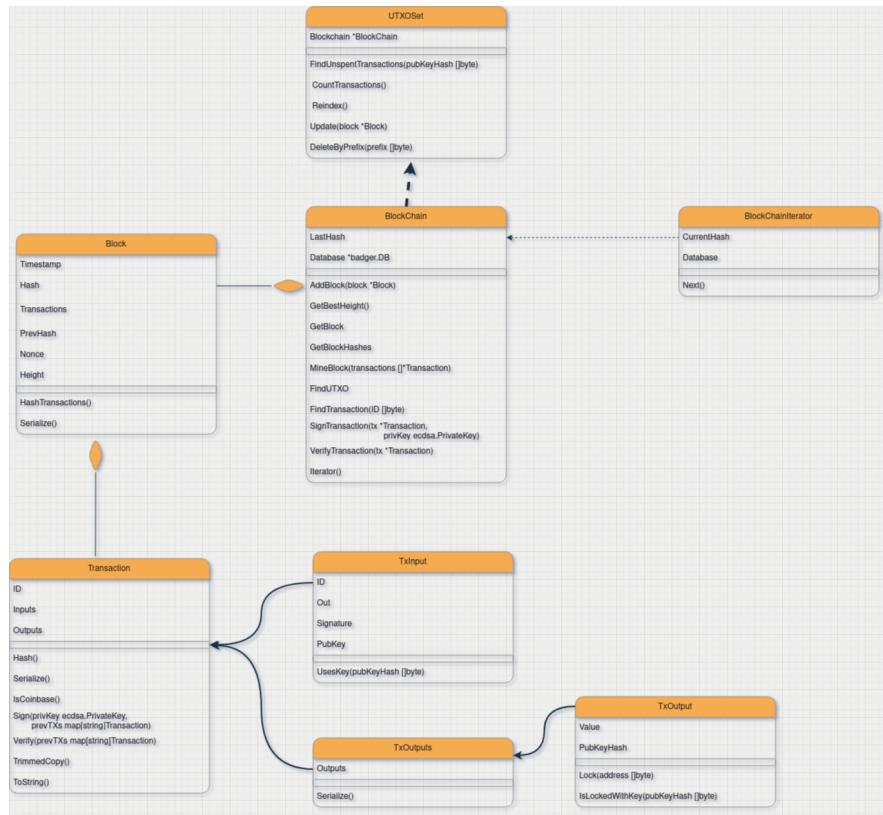


Figure 4.2: Blockchain Class Diagram

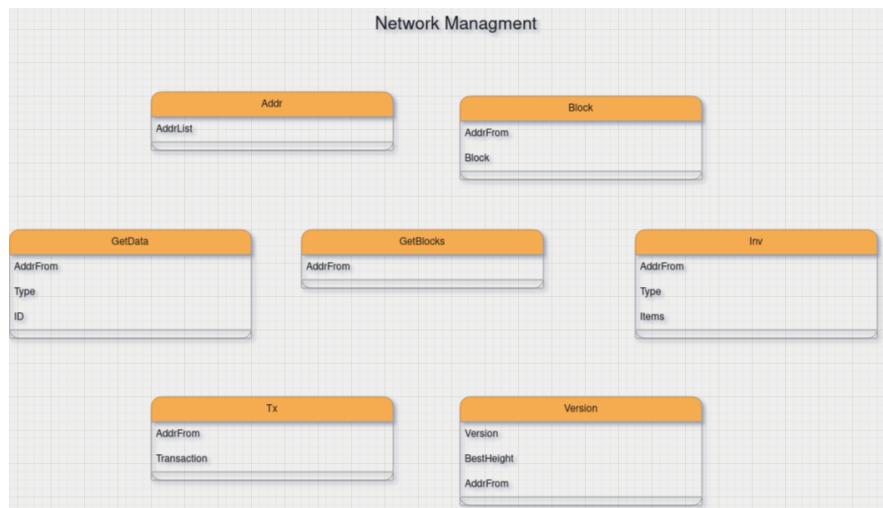


Figure 4.3: Network Class Diagram

4.2.3. Other Classes

4.12.

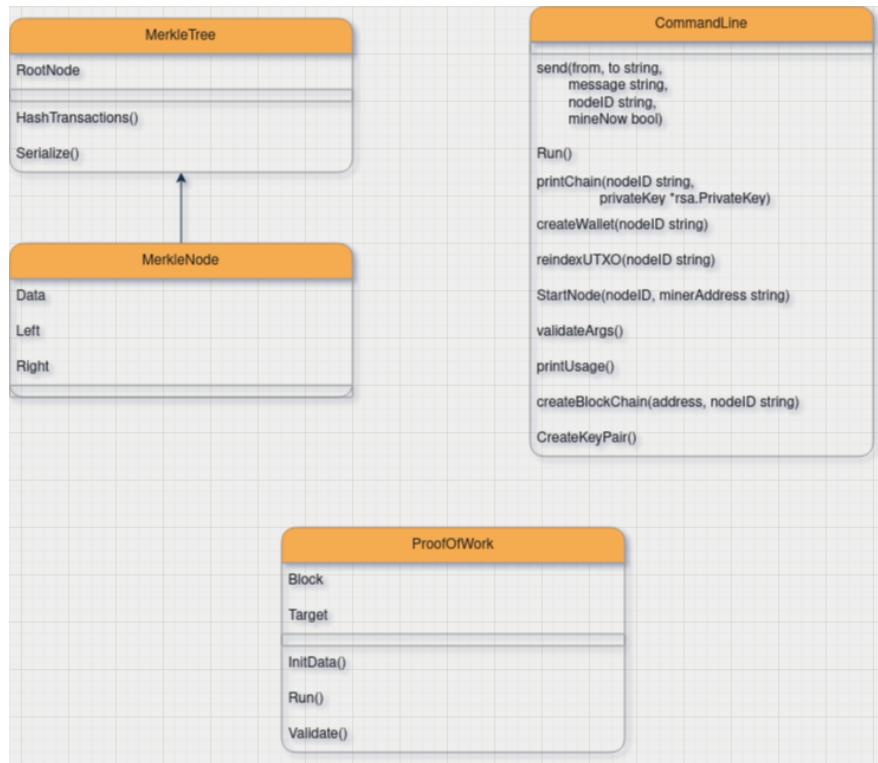


Figure 4.4: Other Classes

4.3. Project Implementation

4.3.1. Communication network setting

It need a port to communicate the other node registered to the blockchain.

```
● mbulucay@mbulucay:~/Blockchain/p2pChatApp$ export NODE_ID=3000
○ mbulucay@mbulucay:~/Blockchain/p2pChatApp$ 
```

Figure 4.5: Network Port Command

4.3.2. Creating a wallet

User creating a wallet address for yourself to communicate the others

```
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createwallet
New address is: 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$ ls
# mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ ls
wallets_3000.data
# mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$
```

Figure 4.6: Wallet Creation Command

4.3.3. Creating a Blockchain Database

User creating a wallet address for yourself to communicate the others

```
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createblockchain -addr
1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
2023/01/15 12:38:07 Replying from value pointer: {Fid:0 Len:0 Offset:0}
2023/01/15 12:38:07 Iterating file id: 0
2023/01/15 12:38:07 Iteration took: 10.503us
000098075eedfcfd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
Genesis created
Finished!
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$ ls
# mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ ls
blocks_3000 wallets_3000.data
# mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$
```

Figure 4.7: Badger Blockchain Database Command

4.3.4. Printing a Blockchain Database

Printing the created database with genesis block

```
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go printchain
2023/01/15 12:39:41 Replying from value pointer: {Fid:0 Len:42 Offset:1039}
2023/01/15 12:39:41 Iterating file id: 0
2023/01/15 12:39:41 Iteration took: 9.948μs
Hash: 000098075eedfcfd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
Prev. hash:
PoW: true
--- Transaction c2cd3044f058aed47c2cf7e7df4780540b35339b676b8e7071ba8ffd730cd969
:
Input 0:
TXID []:
Out -1:
Signature :
PubKey 4669727374205472616e73616374696f6e2066726f6d2047656e65736973:
Output 0
Value Thank you for your services 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
Script: u0000q c0BgK)[0~0
# mbulucay@mbulucay:~/Blockchain/p2pChatApp$
```

Figure 4.8: Printing Chain Command

4.3.5. Key Generation

Creating 2 key one for public other one is private

```
• mbulucay@mbulucay:~/Blockchain/p2pChatApp/keyGenerator$ go run generate.go
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAOB0Arfm+stb/WYP+FcovRz5ScLlIMekXt6TClQIfIwrYYDq0
1Ead7kh8Y0ZNGxyOA54X0vouD3heLnqSd9krVAmJhmNqUEu4K1++2yV6EcwogUn
BXD8FuVY1jyT0PT3DBadpGnmIANhlUhll+upRMTYrD7AMlCTKAL5lpYfRF7/WA4k
9G2XHUbP0k7cS+brviXcsK9IRJsH+xVVV0hU5npbj4eE/MUGCgPZ4gv2cCuvXrf
pKsGt+77rHvlgrYVb52+4dhBRLrlmeaSYiIDpGko5yFZII9AWXdnLZ074PbI3JVG
VVSRxXvmv/aR5ra/owvgSKp1gkZMBxeqRhW14QIDAQABoIBAGVFJ7aBKEX4jytt
KnBKAAbWmYxGNZ885wRvrJgrCipN1bsF9kax0lpIlZxFm9Ecjk0mWiuFKHLivPPb
G3KMiaQMPAbUR8CpCa7QbF8TCighHjdsv6b21/83Lzrn3Z4LduJc8bu29qlcrB
/wxtXG1lc9ZLdrfMEaxI+keJrjLksueEGWI2tCTExez9MA2U0sagcedbCBhaOs
dz2ZqEqYQvM0Pc0NQJXKE/v8GlhchW+jpp4of3CHEqOrs4TNP/zRggAiZnq4a6IC
JqMZ7KyTfaS9Q0zdFKHn4CcBzL+4isSqcBUqw24KSl6Hke+zHxyAxZ6eMr0sD+vW
XerFowEcgYEAA5dd12gBnawHuuHFsvUnbpXegulpCCPa4DdjmXeYfLd2zeMkzVqjg
g5pVWhi3t5sXTjQ7CB3NYMEVk7TFdWh8Eghkw1KiH5basAFsgPKOEI2tVMNSQdtT
Ghau9Bnk008RaXUSQDx00BamHWX31x21Msru00ZGe90A39WavjPj5/ECgYEA5ym
fsPsBN7ooaWVzq3UBxUv4VNINXhcdBW05gUyluxfcKxwDmkED06JrrfTw5lgKw3
INIza0rNmLcza50haevGHeHuFMtab+m0AsivwUPMQkXvGxzgYjVqYFTQG1Fer5z
Ez3ehawPVL4SvnLZhjGFAaTjMprKAG5ZCvKHPEcgYEAv0D0RtuSw4peBGLpczLz
pvzVuBkOpU8jj/CswSOYdecDLgMCot/th4dJ+r0o4U9MmY+Ah5A4oz15zSg3n
sGmEDBFo0/l7k8JHyZwK1V2ogIzW71nu6/SSJ/WTj3ddxr031l2Q4FdQgcs4XG1
c55JqgEXMTkI0+nM9duswfEcgYEAhubemvXxHSCRqMnsdrdJplmYRP2QD7Zdq3WC
GysrVxR3s/Bja/IMX3W+UsUxmhmjSX92Rx2bnWr7tCozM0zZAVL70Rbs1WNH3vcC
HzKmgj/n10YWEuGxdv0ybJ/PgaaIgZG6h0pD3+2z+nCoPfhfDahqLPvGUgQtvbR7
B8ZIwTEcgYBKKAEbI9wbtr9d2t9zJWkgJp0KblexzwzOViCzq2bOK33CR5yLRM2U
3gr5v8T6r5dASU1chC7Qxj3cTaQGT0g5bNrXfcz+ALIECHDRDX//NYZNqaGTdSas
CAUBohsmNnbDAu2e62n6l3IVyJyEQF2ubqDch83e3Mx7ypSGziJ2Uw==

-----END RSA PRIVATE KEY-----

-----BEGIN RSA PUBLIC KEY-----
MIIBCgkCAQEAOB0Arfm+stb/WYP+FcovRz5ScLlIMekXt6TClQIfIwrYYDq01Ead
7kh8Y0ZNGxyOA54X0vouD3heLnqSd9krVAmJhmNqUEu4K1++2yV6EcwogUnBXD8
FuVY1jyT0PT3DBadpGnmIANhlUhll+upRMTYrD7AMlCTKAL5lpYfRF7/WA4k9G2X
HUbP0k7cS+brviXcsK9IRJsH+xVVV0hU5npbj4eE/MUGCgPZ4gv2cCuvXrfpKsg
t+77rHvlgrYVb52+4dhBRLrlmeaSYiIDpGko5yFZII9AWXdnLZ074PbI3JVGvSR
xXvmv/aR5ra/owvgSKp1gkZMBxeqRhW14QIDAQAB
-----END RSA PUBLIC KEY-----
```

```
• mbulucay@mbulucay:~/Blockchain/p2pChatApp/keyGenerator$ ls
generate.go  pubkey.pem  pubkey.pem
• mbulucay@mbulucay:~/Blockchain/p2pChatApp/keyGenerator$
```

Figure 4.9: Printing Chain Command

4.3.6. Send Message

Sending message to one wallet address to another wallet address

```
• mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go send -from 1BfoaDYMgs7Vv1ZCqY6V7UJT6D4wyLX7MX -to 1BfoaDYMgs7Vv1ZCqY6V7UJT6D4wyLX7MX -message "NODE 3000 den merhabalar" -publickey ./userKeys/user_1/pubkey.pem -mine
2023/01/18 12:42:42 Replying from value pointer: {Fid:0 Len:42 Offset:1039}
2023/01/18 12:42:42 Iterating file id: 0
2023/01/18 12:42:42 Iteration took: 7.675us
000029767e5781a09c0064785e21589758c0328646dbad97ae39985a8f4e5e1
Success!
• mbulucay@mbulucay:~/Blockchain/p2pChatApp$
```

Figure 4.10: Message Sending Command

4.3.7. Encrypted Message

Printing the send command transaction

```
● mbutucay@mbutucay:~/Blockchain/p2pChatApp$ go run main.go send -from 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX -to 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX -message "NODE 3000 den merhabalar" -publickey ./userKeys/user_1/pubkey.pem -mine
2023/01/15 12:42:42 Replaying from value pointer: {Fid:0 Len:42 Offset:1039}
2023/01/15 12:42:42 Iterating file id: 0
2023/01/15 12:42:42 Iteration took: 7.675μs
000029767e5781a09c0064785e21589758c0328646dbad97eae39985a8f4e5e1
Success!
● mbutucay@mbutucay:~/Blockchain/p2pChatApp$ go run main.go printchain
2023/01/15 12:43:10 Replaying from value pointer: {Fid:0 Len:42 Offset:2975}
2023/01/15 12:43:10 Iterating file id: 0
2023/01/15 12:43:10 Iteration took: 7.901μs
Hash: 000029767e5781a09c0064785e21589758c0328646dbad97eae39985a8f4e5e1
Prev. hash: 000098075eedcfcd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
PoW: true
--- Transaction 06b52fed961027fc975ae098360a92b6b9db29c43f2a6d06b7fc7d80bff0332:
    Input 0:
    TXID []:
    Out -1:
    Signature :
    PubKey 32613637303130346136626137383839626133653566336630395393138656662376237646432663739313530663032:
        Output 0
        Value Thank you for your services 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
        Script: u0000q c0BgK)[0..0
--- Transaction ed6b920b2f21f142b1584aa7f704390a8f9972b54f679f927d9b119829f48736:
    Output 0
    Value 00000-SC00000b0) HDM00~KAZ|-00*%V;w
0;9000E0000b0A0@0^w000Q}Ahk_0200c000Dy000|-0u000m0Tp0EU050000A00;00PW0D0N0^00CT#00N000]00N0lsk>B0040000000500p00Y0004000
    Script: u0000q c0BgK)[0..0

Hash: 000098075eedcfcd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
Prev. hash:
PoW: true
--- Transaction c2cd3044f058aed47c2cf7e7df4780540b35339b676b8e7071ba8ffd730cd969:
    Input 0:
    TXID []:
    Out -1:
    Signature :
    PubKey 4669727374205472616e73616374696f6e2066726f6d2047656e65736973:
        Output 0
        Value Thank you for your services 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
        Script: u0000q c0BgK)[0..0
```

Figure 4.11: Printing Chain Command

4.3.8. Decrypted Message

Printing the send command transaction

```

TXID []:
Out -1:
Signature :
PubKey 326136373031303461366261373838396261336535663366303935393138656662376237646432663739313530663032:
    Output 0
    Value Thank you for your services 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
    Script: u000q`c0BgK)[@.0
--- Transaction ed6b920b2f21f142b1584aa7f704390a8f9972b54f679f927d9b119829f48736:
    Output 0
    Value 0000-SC0000b0) HDM00-kAZ-00*%V;W
    Script: u000q`c0BgK)[@.0
Hash: 000098075eedcfcd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
Prev. hash:
PoW: true
--- Transaction c2cd3044f058aed47c2cf7e7df4780540b35339b676b8e7071ba8ffd730cd969:
    Input 0:
    TXID []:
    Out -1:
    Signature :
    PubKey 4669727374205472616e73616374696f6e2066726f6d2047656e65736973:
        Output 0
        Value Thank you for your services 1BfoaDYMgs7VV1ZCqY6V7UJT6D4wyLX7MX
        Script: u000q`c0BgK)[@.0
● mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go printchain -privatekey ./userKeys/user_1/privkey.pem
2023/01/15 12:44:24 Replaying from value pointer: {Fid:0 Len:42 Offset:2975}
2023/01/15 12:44:24 Iterating file id: 0
2023/01/15 12:44:24 Iteration took: 8.636µs
Hash: 000098075e75781a09c0064785e21589758c0328646dbad97eae39985a8f4e5el
Prev. hash: 000098075eedcfcd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
PoW: true
--- Transaction 06b52fed961027fc975ae098360a92b6b9db29c43f2a6d06b7fc7d80bff0332:
    Input 0:
    TXID []:
    Out -1:
    Signature :
    PubKey 326136373031303461366261373838396261336535663366303935393138656662376237646432663739313530663032:
        Output 0
        Value:
        Script: u000q`c0BgK)[@.0
--- Transaction ed6b920b2f21f142b1584aa7f704390a8f9972b54f679f927d9b119829f48736:
    Output 0
    Value: NODE 3000 den merhabalar
    Script: u000q`c0BgK)[@.0
Hash: 000098075eedcfcd8ddb0b9bec9f66ed6c2f8bb22650783d628d145f5c596d4f
Prev. hash:
PoW: true
--- Transaction c2cd3044f058aed47c2cf7e7df4780540b35339b676b8e7071ba8ffd730cd969:
    Input 0:
    TXID []:
    Out -1:
    Signature :

```

Figure 4.12: Printing Chain Command

4.4. Ethereum Smart Contract Project Implementation

4.4.1. Tools

- Ethereum Network: A platform for decentralized applications Ethereum is an open source, public, blockchain based distributed computing platform and operating system featuring smart contract functionality. It supports a modified version of Nakamoto consensus.
- Solidity: Solidity is an object-oriented, high-level language for implementing smart contracts. Smart Contracts are programs which govern the behaviour of accounts within the Ethereum state.
- Hardhat: Hardhat is a development environment for Ethereum software. It consists of different components for editing, compiling, debugging and deploying your smart contracts and dApps, all of which work together to create a complete development environment.
- Goerli Etherscan: Goerli Etherscan is a blockchain explorer for the Goerli testnet, which is a test network for the Ethereum blockchain. It allows users to view and search for transactions, blocks, and contracts on the Goerli testnet.
- MetaMask: MetaMask is a browser extension and mobile app that allows users to interact with the Ethereum blockchain. It acts as a digital wallet and enables users to securely store, manage, and send Ethereum and other Ethereum-based tokens. MetaMask also allows users to interact with decentralized applications (dApps) built on the Ethereum blockchain, such as decentralized exchanges, lending platforms, and prediction markets.
- Infura: Infura is a hosted Ethereum node cluster that lets your users run your application without requiring them to set up their own Ethereum node or wallet.

4.4.2. Deploy Ethereum Network

We are building a basic chat smart contract to send message with solidity.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract p2pChatApp{
    struct Mesaj{
        address from;
        string content;
    }
    mapping(uint => Mesaj) m_mesaj;
    mapping(address => Mesaj[]) public adres_box;
```

Figure 4.13: Deploy Settings

4.4.2.1. Adjust Settings

Adjust the setting for the deploy requirements. The keys are unique for the accounts so don't share with the others.

```
RPCURL="https://goerli.infura.io/v3/<your infura key>"
ETHERSCAN_API_KEY="<your etherscan key>|
PRIVATE_KEY="<your metamask private key>"
```

Figure 4.14: Deploy Settings

4.4.2.2. Deploy Contract Command

Deploy Contract has return the contract deploy address then we can the ethereum network.

```
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL GITLENS
• mbulucay@mbulucay:~/Blockchain/deploy/p2pChats$ npx hardhat clean
• mbulucay@mbulucay:~/Blockchain/deploy/p2pChats$ npx hardhat compile
    Compiled 2 Solidity files successfully
• mbulucay@mbulucay:~/Blockchain/deploy/p2pChats$ npx hardhat run ./scripts/deploy.js --network goerli
Lock with 1 ETH and unlock timestamp 1705319045 deployed to 0xd108f3e7D592e4b07094348C071503588FD960DE
• mbulucay@mbulucay:~/Blockchain/deploy/p2pChats$
```

Figure 4.15: Deploy Command

4.4.2.3. Ethereum Network - Etherscan

Check the Ethereum test network for the return contract address

Figure 4.16: Ethereum Network

4.4.2.4. Send Transaction

Sending a transaction on to the ethereum network with the remix-ide (Remix is an Integrated Development Environment (IDE) for writing, testing, and deploying smart contracts on the Ethereum blockchain). The metamask extension account has need to be some ether to sending transaction one account to another. Because the sending messages or transactions have cost.

We need to connect the metamask with ide. In remix choose Envirement ; Wallet Connect; and pick the account which you want to send message.

Figure 4.17: Transaction

4.4.2.5. Check Transaction

Check the Transaction which we send. On the ethereum test network with etherscan.

② Transaction Hash: 0xa96dcdfcd7f0d48ea981e8f3dca1a95064501023947f3a05a2fd77c99c3d6a1c ⓘ

② Status: Pending

② Block: (Pending)

② Time Last Seen: 13 days 00 hr 00 min 24 secs ago (Jan-15-2023 11:52:44 AM)

② From: 0x9f1fb796a96214ac36a7e8427e3c024060282bf1 ⓘ

② Interacted With (To): Contract 0xd1d8f3e7d592e4b07094348c071503588fd960de ⓘ

② Value: 0 Ether (\$0.00)

② Max Txn Cost/Fee: 0.0000393872502678333 Ether (\$0.00)

② Gas Price: 0.000000002500000017 Ether (2.500000017 Gwei)

② Gas Limit & Usage by Txn: 157549 | (Pending)

② Gas Fees: Max: 2.500000028 Gwei | Max Priority: 2.5 Gwei

② Other Attributes: Txn Type: 2 (EIP-1559) | Nonce: 5 | Position In Block: (Pending)

② Input Data: 0x" followed by a large hex string. Note: This message comes from the remix ide

View Input As ▾

Figure 4.18: Check Transaction

5. LOCAL BLOCKCHAIN IMPLEMENTATION

It's worth mentioning that building a blockchain network from scratch could be a challenging task, and it requires a good understanding of the underlying technology and its implementation. I build an local blockchain network using go programming language and badger database. Instead of using just tool for the make a program. Here is the example usage of program.

```
The program can createing wallet. With the unique address on the ./tmp folder for the each user
{
    $ go run main.go createwallet
}

The program can list all the wallets in the ./tmp folder
{
    $ go run main.go listwallets
}

The program can send the message to the address
{
    $ go run main.go send -from FROM_ADDRESS -to TO_ADDRESS -message CONTENT -mine -publickey PUBLIC_KEY_FILE_PATH
}

The program can create the blockchain
{
    $ go run main.go createblockchain -address ADDRESS
}

The program can print the blockchain
{
    $ go run main.go printchain -privatekey PRIVATE_KEY_FILE_PATH
}

The program can create the new node as miner
{
    $ go run main.go startnode -miner ADDRESS
}
```

Figure 5.1: Command

5.1. Usage Of Program

```

mbulucay@mbulucay:~/Blockchain/p2pChatApp$ ./p2pChatApp 60654
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ export NODE_ID=30800
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createwallet
New address is: 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createblockchain
Blockchain -address 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
0023/01/15 14:07:51 Iterating free value pointer: (Fid:0 Len:0 Off:0)
0023/01/15 14:07:51 Iterating file id: 0
0023/01/15 14:07:51 Iteration took: 0.174us
0007f2fd38163dbdf43f5240efffb92795b3080013b3a5c20c4b076c34c3e4
generics created
genesis created
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go printtchain
in
0023/01/15 14:08:49 Iterating from value pointer: (Fid:0 Len:42 Off:0x1030)
0023/01/15 14:08:49 Iterating file id: 0
0023/01/15 14:08:49 Iterating took: 17.243us
hash: 0007f2fd38163dbdf43f5240efffb92795b3080013b3a5c20c4b076c34c3e4
Prev. hash:
PwN: true
... Transaction 930ca17c019da6d3d3d859a6e615ef7a7b53a61f11dcbb002e807
0391ba0d5f42c9:
    Input 0:
        TXID []:
        Out -1:
        Signature :
            PubKey 4669727374205472616e736163746996f6e2066726f6d2047656e65
736973:
        Output 0
            Value Thank you for your services 18whArzkMn3GeZEY9ddLT4MY
HYENQGLb5RS
        Script: W *****l2*****l
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ 

mbulucay@mbulucay:~/Blockchain/p2pChatApp$ export NODE_ID=50800
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createwallet
New address is: 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ cd tmp/
mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ cp -R blocks_3000/ blocks_5000
mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ cd ..
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ 

```

Figure 5.2: Image 1

```

mbulucay@mbulucay:~/Blockchain/p2pChatApp$ ./p2pChatApp 60654
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ export NODE_ID=30800
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createwallet
New address is: 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createblockchain
Blockchain -address 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
0023/01/15 14:07:51 Iterating free value pointer: (Fid:0 Len:0 Off:0)
0023/01/15 14:07:51 Iterating file id: 0
0023/01/15 14:07:51 Iteration took: 0.174us
0007f2fd38163dbdf43f5240efffb92795b3080013b3a5c20c4b076c34c3e4
generics created
genesis created
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go printtchain
in
0023/01/15 14:08:49 Iterating from value pointer: (Fid:0 Len:42 Off:0x1030)
0023/01/15 14:08:49 Iterating file id: 0
0023/01/15 14:08:49 Iterating took: 17.243us
hash: 0007f2fd38163dbdf43f5240efffb92795b3080013b3a5c20c4b076c34c3e4
Prev. hash:
PwN: true
... Transaction 930ca17c019da6d3d3d859a6e615ef7a7b53a61f11dcbb002e807
0391ba0d5f42c9:
    Input 0:
        TXID []:
        Out -1:
        Signature :
            PubKey 4669727374205472616e736163746996f6e2066726f6d2047656e65
736973:
        Output 0
            Value Thank you for your services 18whArzkMn3GeZEY9ddLT4MY
HYENQGLb5RS
        Script: W *****l2*****l
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ 

mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go send -from 18BVUqqzZ8LP9DNHerrn0gUewrwdop97VnA -to 18whArzkMn3GeZEY9ddLT4MYEhQG
-Lockchain -RECEIVER send MESSAGE "USER1" -nme -publickey
2023/01/15 14:10:43 Iterating from value pointer: (Fid:0 Len:42 Offs:0x1030)
2023/01/15 14:10:43 Iterating file id: 0
0007f5cc6b4569db0e02d0a6d03cff87b70aae3d3fc3ff57228bb71ff
success!
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ 

mbulucay@mbulucay:~/Blockchain/p2pChatApp$ export NODE_ID=50800
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ go run main.go createwallet
New address is: 18whArzkMn3GeZEY9ddLT4MYEYhQhLbh5RS
mbulucay@mbulucay:~/Blockchain/p2pChatApp$ cd tmp/
mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ ls
blocks_3000 blocks_5000 wallets_4000.data wallets_5000.data
blocks_4000 wallets_3000.data wallets_3000.data
mbulucay@mbulucay:~/Blockchain/p2pChatApp/tmp$ 

```

Figure 5.3: Image 2

Figure 5.4: Image 3

Figure 5.5: Image 4

Figure 5.6: Image 5

Figure 5.7: Image 6

Figure 5.8: Image 7

Figure 5.9: Image 8

6. CONCLUSIONS

A blockchain-based secure messaging application is a decentralized and secure way to send messages. By using blockchain technology, it ensures that the data is tamper-proof, immutable, censorship resistant and accessible only by authorized users.

The project design plan for the application involves a thorough analysis of the current state of messaging security and the ways in which blockchain technology can be utilized to address these issues. The implementation of the application includes the use of an appropriate blockchain platform, a user-friendly interface, and a peer-to-peer messaging system that is based on asymmetric encryption.

Users need a wallet address on the blockchain network and a private key to encrypt and decrypt the messages, this ensures that only the intended recipient can read the message, and it increases the level of security and privacy. The application will be thoroughly tested for security and performance, and it will be deployed and made available to users.

Overall, building a blockchain-based secure messaging application is a challenging task that requires a good understanding of the underlying technology and its implementation. However, the benefits of a decentralized and secure messaging system make it a worthwhile endeavor. With the increasing demand for secure communication, a blockchain-based secure messaging application can provide a valuable service for users and businesses alike.

BIBLIOGRAPHY

- [1] Authors: Zibin Zheng¹ , Shaoan Xie¹ , Hongning Dai ² , Xiangping Chen ⁴ , and Huaimin Wang³
Title: An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends
Date: June 2017
Page: 557-564
Number: 978-1-5386-1996-4/17
- [2] Authors: Sourabh¹, Deepanker Rawat², Karan Kapkoti³, Sourabh Aggarwal⁴, Anshul Khanna⁵
Title: A Decentralized Chat Application
Date: 05 — May 2020
Page: 2775-2779
Number: 9001:2008
Volume: 07
- [3] Authors: Kahina Khacef , Guy Pujolle
Title: Secure Peer-to-Peer communication based on Blockchain
Date: 15 March 2019
Volume: 927
DOI: 10.1007/978-3-030-15035-8_64
- [4] Authors: Vikas Bhardwaj, Raja Ram Sharma, Akhilesh Kumar
Title: Blockchain Technology: Chat Application
Date: July 2021
Page: 7260 - 7268
Volume: 12
- [5] Authors: Satoshi Nakamoto
Title: Bitcoin: A Peer-to-Peer Electronic Cash System
Date: 2008
- [6] Authors: Abdul Ghaffar Khan, Sana Basharat, Muhammad Usama Riaz
Title: Analysis of asymmetric cryptography in information security based on computational study to ensure confidentiality during information exchange
Date: October 2018
Page: 992 - 999

- [7] Authors: Stearns, Mahzad
Title: A Decentralized Approach to Messaging Using Blockchain Technology
Date: May 2019
- [8] Authors: DR. Wood Founder, Ethereum & Ethcore
Title: A Secure Decentralized Generalized Transaction ledger
Date: 08 - Oct- 2018
- [9] Github Links:
<https://github.com/TristanBilot/blockchain-chat-app>
<https://github.com/nhatminh12369/etherchat>
<https://github.com/ethereum/go-ethereum/>
- [10] Tutorial Links:
https://www.youtube.com/watch?v=hYip_Vuv8J0t=261s
<https://www.youtube.com/watch?v=bBC-nXj3Ng4t=1065s>
https://www.youtube.com/watch?v=ZEApLtE8KkElist=PLxz5ldaTYSOUmhECFNN-WfGeJrzlnXn_a
<https://www.youtube.com/watch?v=yubzJw0uiE4t=261s>
- [11] Other Links:
<https://medium.com/adamant-im/how-decentralized-blockchain-messenger-works-b9932834a639>
<https://medium.com/@BeFastTV/top-blockchain-messaging-apps-crypto-messengers-28893e5f908f>
<https://hal.archives-ouvertes.fr/hal-02180329/document/>
<https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html>
<https://remix.ethereum.org>
https://www.researchgate.net/publication/328160285_Survey_of_Consensus_Proocols
<https://blog.harmony.one/peer-discovery-in-harmony-network/>