

**Capstone Project Phase B**

**Blockchain private key security**

**Project number** - 23 - 1 - D - 16

Supervisor: Mr. Alex Keselman

Karin Shpigelman - [karinsh97@gmail.com](mailto:karinsh97@gmail.com)

Nitzan Shani - [nitzan963@gmail.com](mailto:nitzan963@gmail.com)

**Table Of Contents**

[**1 Introduction**](#_2vmlapckhcd1) **3**

[**2 Background**](#_14fnl8qre6zn) **4**

[2.1 Cryptocurrency and Blockchain](#_arcsr0mk6ofo) 4

[**3 Engineering process**](#_8w3j2p2f5mto) **7**

[3.1 The Process](#_3s9mecswpj1k) 7

[**4 Product**](#_42ub6eha3n1l) **9**

[4.1 Proposed Solution](#_o1jp8fqva3uj) 9

[4.2 Implementation Process](#_4due2c3pt86i) 11

[4.3 Used Technology](#_6a5lisfidx1p) 12

[4.4 Architecture Diagram](#_7j7vs62mfkup) 13

[4.5 Activity Diagram](#_k6bzqmcfuv1m) 13

[4.6 Web Screens](#_8klzg2qc1sb4) 15

[**5 Evaluation/Verification Plan**](#_k1wr4mp47svy) **15**

[**6 Results and Conclusions**](#_9373z8relty6) **16**

[**7 User Documentation**](#_8u8v34jqke3) **17**

[7.1 General Description](#_tujbcft5pjtt) 17

[7.2 Usage Scenarios](#_qque6hw4zdj) 18

[**8 Maintenance Guide**](#_fpht3wxyt5x3) **19**

[8.1 Package Diagram](#_romem9yx5kta) 19

[**9 User Help**](#_9n13oe4uz5gw) **19**

[**10 References**](#_nlgo8dyk6gow) **21**

**Abstract**

In this project, we present a new approach for securing private key in cryptocurrency exchanges and large-scale projects based on blockchain technology.The goal of the project is to reduce the risk of unauthorized access to the private key for the administrator wallet, without the need for human intervention. The proposed algorithm splits the private key into smaller parts and stores them in blocks, which are then distributed across multiple servers. This approach aims to mitigate the risk of unauthorized access to the private key and makes it more difficult for hackers to access and compromise the assets associated with it. The algorithm is demonstrated using a web application, which executes a transaction using the algorithm.

# **1 Introduction**

We may be at the beginning of a new revolution. These days we are witnessing plenty of changes in the finance world. When talking about money transfers or investment services, a third party usually confirms and performs the transfer between people or invests the money.

There are some major problems in this system like inflation, slow transfer time, high commission and the main problem is the third party which in most cases is the bank, the government, or a person that controls the money. Blockchain is a peer-to-peer version of electronic cash allowing secure and transparent transactions without the need for a trusted third party, such as a bank. This eliminates the need for costly intermediaries and makes transactions faster and more efficient. Each Blockchain account owner has a private key, so whoever has the key is the account's owner and can perform transfer operations, etc.

The main problems with private keys in the blockchain are that they are vulnerable to being lost or stolen and they can be difficult to manage and secure.

One of the main challenges facing cryptocurrency exchanges, individuals, and large-scale projects based on blockchain is security. Because cryptocurrencies are digital and decentralized, they are vulnerable to hacks and other forms of cyber attacks. This has led to numerous incidents of exchanges being hacked and users losing their funds. For example in December 2021 Hackers withdrew [$196 million](https://www.coindesk.com/business/2021/12/05/crypto-exchange-bitmart-hacked-with-losses-estimated-at-196-million/) of cryptocurrency from the crypto exchange BitMart by stealing private keys that opened two wallets.

Many individuals continue to use cryptocurrency exchanges as a means of storing and trading their digital assets due to the convenience and ease of conducting transactions through these platforms.

Cryptocurrency exchanges as a system automating transfers of a lot of valuable assets and having access to the funds locked in all the involved smart contracts, require a security layer solution for minimizing the risk of bad actors obtaining access to the private key to the admin wallet. Cryptocurrency exchanges often implement a combination of technical and operational measures to safeguard private keys, while individuals may be more vulnerable to losing their private key or having it stolen by hackers, due to a lack of resources or expertise. In this study, we will review various methods for securing private keys and propose an algorithm for doing so. By implementing effective technical and operational measures to protect private keys, the risk of hacking and theft of billions of dollars stored in cryptocurrency exchanges can be significantly reduced.

The algorithm we suggest is automated Private key storage, which involves dividing the key into several parts, storing each part on a different server, and using MPC to reconstruct the key when needed. We will examine the requirements and corresponding solutions for private key storage.

# **2 Background**

In this section, we will provide an overview of the concepts and technologies that we are using in this project.

## **2.1 Cryptocurrency and Blockchain**

*2.1.1 Cryptocurrency*

Cryptocurrency is a digital currency that uses cryptography for security and is decentralized, meaning it is not controlled by any government or other central authority.

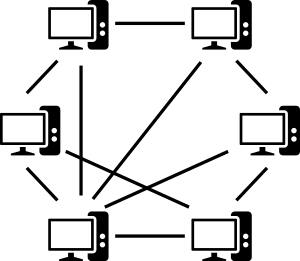
Cryptocurrencies are typically designed to be limited in supply and to operate independently of a central bank, making them immune to government manipulation or interference. They are also often anonymous, allowing users to transact without revealing their identity. Some of the most well-known cryptocurrencies include Bitcoin, Ethereum, and Litecoin.

*2.1.2 Blockchain*

Blockchain was created by Satoshi Nakamoto in 2008 based on previous work by Stuart Haber, W. Scott Stornetta, and Dave Bayer. It was first introduced as the underlying technology behind the cryptocurrency Bitcoin. It has since been adopted by many other cryptocurrencies and has also been used in a variety of other applications.

Blockchain consists of continuously growing lists of transaction records, called blocks. Blocks are securely linked together using cryptography. Each block contains transaction data, a timestamp, and a cryptographic hash of the previous block. The timestamp proves that the transaction data existed when the block was created. Since each block contains information about the previous block, they form a chain (linked list), with each additional block linking to the ones before it. any attempt to modify a block would require modifying all subsequent blocks in the chain, which would require a significant amount of computing power.  
The transparency and immutability of the blockchain allow transactions to be tracked and verified, providing a high level of security and trust.

Blockchains are typically managed by a peer-to-peer (P2P) computer network, whereby two individuals interact directly with each other, without the need for a trusted authority or a central server (third party).



*figure 3: peer to peer*

*2.1.3 Public Key*

A public key allows you to receive cryptocurrency transactions. While anyone can send transactions to another account, using its public key, you need the private key of your account to prove that you are the owner of the cryptocurrency received in the transaction. You can have any number of public keys connected to a single private key. Unlike the private key, the public key can be shared publicly without compromising the security of the user's funds.

*2.1.4 Private Key*

A private key gives you the ability to prove ownership or spend the funds associated with your public address. A private key can take many forms: long binary code, hexadecimal code, QR code, mnemonic phrase.   
Regardless of its form, a private key is an astronomically large number. While you can generate a public key with a private key, doing the opposite is practically infeasible because of the one-way function. It is important to back up the private key in case it is lost or stolen and never share your private key with anyone. If a private key is lost, stolen, or otherwise compromised, the associated digital assets may be at risk of being accessed or transferred without the owner's consent. It is therefore essential that private keys be kept secure and protected at all times.

*2.1.5 Transaction*

Transaction is a record of an exchange of value between two or more parties. This can include the transfer of cryptocurrencies, the execution of a smart contract, or the recording of information on the blockchain.

Transactions on a blockchain network are validated and recorded on the distributed ledger using a consensus mechanism, such as proof of work or proof of stake. Once a transaction is added to a block and the block is added to the blockchain, it becomes part of the permanent and tamper-evident record of transactions on the network.

*2.1.6 Submit Transaction to Blockchain*

The process of accepting a transaction by the blockchain involves several steps to ensure that it is properly formatted and follows the rules of the blockchain. First, the transaction is verified to ensure that it is properly formatted and that it follows the rules of the blockchain. Next, it is broadcast to the rest of the network and propagated to other nodes, where it is validated by those nodes to ensure that it is valid and follows the rules of the blockchain. If the transaction is valid, it is then added to a block of transactions, which is broadcast to the network and added to the blockchain by other nodes if it is valid. By following this process, the blockchain can ensure that only valid transactions are added to the blockchain, which helps to maintain the integrity of the network.

# **3 Engineering process**

In Part A of the project, we gained an understanding of the cryptographic and cryptocurrencies field and its relevance to our project. We also addressed the problem of private key security and reviewed various solutions. In Part B of the project, we will design and build a web application to demonstrate the algorithm.

## **3.1 The Process**

In the initial phase of our research process, we defined the research question that we sought to address through our work. This involved identifying the main problem that we wanted to address and formulating a specific research question.

Subsequent to this initial phase, we conducted a literature review in order to gain a deeper understanding of the relevant fields of study. This involved reading and reviewing academic articles and other materials written by experts in these fields, in order to gain a comprehensive understanding of the current state.

Following the initial phase of our research process, we defined the specific requirements. This involved identifying the specific goals and performance criteria that the algorithm needed to meet, such as the required level of security and scalability.

Based on these requirements, we designed the overall structure and functionality of the algorithm, including the specific protocols that would be used and UML diagrams.

As part of the algorithm development process, we conducted extensive research on the management and storage of private keys, as well as various cryptography methods.

Once the design of the algorithm was complete, we developed a testing plan in order to evaluate its performance and functionality. This may have involved identifying specific test cases and test data to be used, and outlining the specific methods and techniques that would be used to evaluate the algorithm.

The algorithm can provide improved security compared to traditional cryptographic algorithms. This is because it relies on the cooperation of multiple parties, and does not rely on the assumption that a single party is trustworthy.

One of the main advantages of the algorithm is that it increases the difficulty for hackers to access the keys. This is because, in the system, the private key is divided into several smaller pieces, which are stored on separate servers. To access the private key, the parties must be combined and decrypted using a specific process. As a result, a hacker would need to attack multiple parties across different systems and locations in order to access the private key, rather than just targeting a single server or individual.

Even if an attacker were able to gain access to one of the validators, they would only be able to access a portion of the private key and would not have the full key needed to access the assets. Additionally, by storing each character of the private key twice in two different validators, the system is able to protect against data loss or corruption. If one of the servers fails or is compromised, the private key can still be reconstructed from the remaining validators. It also has the potential to improve scalability. The algorithm can be designed to scale up to a large number of servers, making it suitable for use in large-scale systems. The more parties involved, the more secure the system is.

*3.2.1 The Challenges*

In the research process, three primary challenges were encountered, specifically in relation to secure channel communication, single component failure, and synchronization. To address these challenges, solutions were integrated into the algorithm. These solutions include the implementation of encryption, the distribution of servers across various locations and cloud providers, and the adoption of the Master-slave replication method.

The establishment of secure channels of communication is to mitigate the risk of man-in-the-middle attacks. The attacker could intercept the communication between the servers and modify the private key or the way it is divided and distributed. This would allow the attacker to gain access to the private key and compromise the assets associated with it, despite the algorithm's security measures. To mitigate the risk of these attacks, we implement additional security measures - symmetric encryption. To ensure the integrity and confidentiality of the communication channel between the servers.  
To ensure secure transmission of the private key components, we employed AES-256-CBC encryption. This encryption algorithm provides a high level of security by employing a 256-bit key and utilizing the CBC mode of operation. By utilizing AES-256-CBC encryption, we aimed to safeguard the confidentiality and integrity of the transmitted private key parts, mitigating the risk of unauthorized access or tampering during transmission.

Another potential vulnerability occurs when a system has a single component that if it fails, it will cause the entire system to fail. It would occur if all the servers that store the private key are located in the same location or hosted by the same cloud provider. If the location or cloud provider is compromised, an attacker could access all the servers and steal the private key, compromising the assets associated with it. To mitigate the risk it is important to distribute the servers across multiple locations and cloud providers. In our algorithm, we have implemented a fault-tolerant mechanism to ensure the resilience of the system in the event of server failure. Specifically, we have incorporated a backup strategy that ensures redundancy across multiple servers. This approach guarantees that if one server fails, the required data or functionality can still be seamlessly accessed or executed from the other available servers. By implementing this fault-tolerant mechanism, we aim to maintain the continuity and reliability of the system even in the presence of server failures.

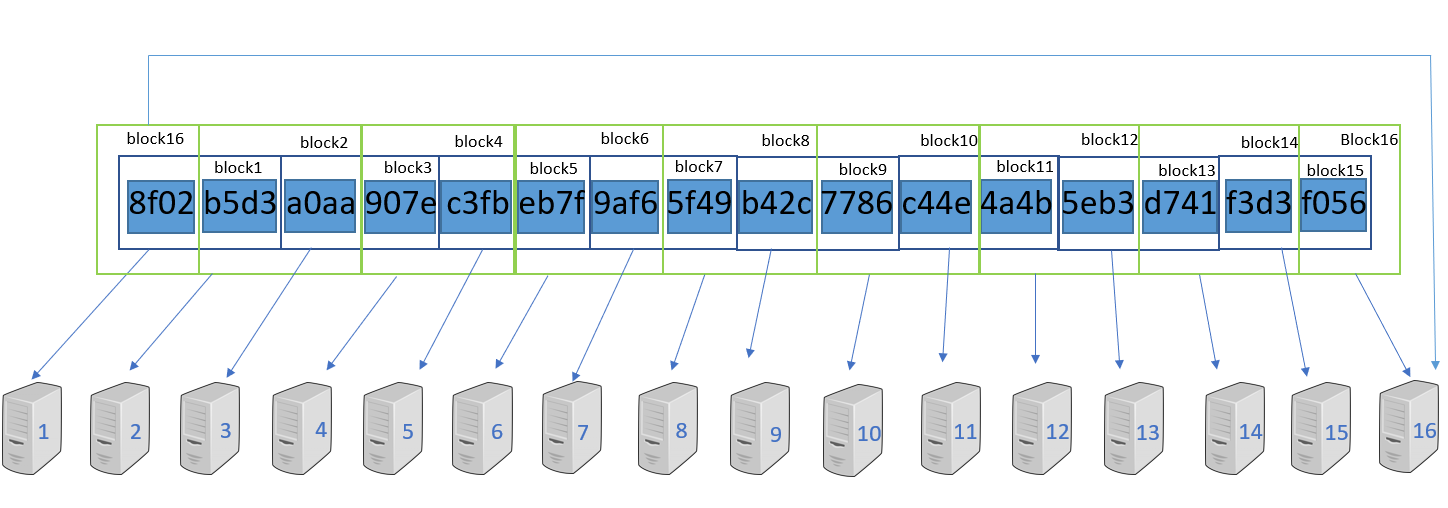
Synchronization is another challenge that needs to be addressed. The servers need to be kept in sync with each other in order to ensure that the private key can be reconstructed when needed. We plan to use the Master-slave replication method in which the primary server is tasked with managing and distributing the private key, while the secondary servers are responsible for maintaining the key partitions.  
We employed two synchronization mechanisms in our implementation. The first approach involves the gateway sending a request and waiting for a response for a specified duration of 5000 milliseconds. Alternatively, the gateway can wait until all servers have responded, whichever occurs first. This synchronization strategy ensures that the gateway proceeds with further processing based on the earliest received response or upon the completion of responses from all servers, thereby optimizing the synchronization process in the system.

# **4 Product**

## **4.1 Proposed Solution**

We have discussed the main problems of private key security. and the consequences of revealing or losing the private key. In this part, we will present a solution that may help cryptocurrency exchanges and individuals as well.  
Our proposed solution is a way to divide and distribute a private key across multiple servers for improved security. It contains several components and includes the following steps to implement:

1. The following algorithm is suitable for 16 servers, although it could be easily managed for more or fewer servers, according to security needs.
2. The 64-character (256-bit) private key is divided into 16 smaller parts of 4-character. These smaller parts are stored in 8-character blocks, with each block containing 2 of the smaller parts. The first block, containing the first 8-character of the private key, is stored in the first server. The second block, containing the last 4-character from the first block and next 4-character of the private key, is stored in the second server. This process continues until all 16 smaller parts are stored in 8-character blocks and distributed across 16 servers. This ensures that each bit of the private key is stored twice in two different servers, adding an extra layer of security to the key. In case one of our servers fails we can still access the private key. By storing the private key this way, a potential hacker will have to hack all 16 servers in order to access the private key and compromise the assets associated with it, which becomes much more difficult.

Example of the divided and stored private key:

1. Each part will be encrypted using symmetric encryption and sent to a different server.
2. To improve security, the private key can be divided into several parts and stored on different servers located in different locations and companies, such as AWS and Google's servers. For example, servers 1 to 8 could store parts of the private key on AWS servers, while servers 9 to 16 could store other parts of the key on Google's servers. This ensures that if one company's servers are compromised, the attacker would still not have access to the entire private key.
3. Upon completion of a transaction, a new random key will be employed to encrypt each partition of the divided private key.

Upon initiating a withdrawal, the main computer initiates a request to each of the servers to retrieve their respective portions of the private key. Once receiving responses from all servers, the individual portions of the private key are combined to form a composite key. This composite key is then decrypted using symmetric decryption to reveal the true private key, which is then utilized to sign the transaction. The signed transaction is then transmitted. Subsequent to the transmission of the transaction, the private key is re-encrypted using a different random key and symmetric encryption.

## **4.2 Implementation process**

The development process of the system involved a systematic approach that began with the independent development of each component. This allowed for focused attention on the specific functionalities and requirements of each part before merging them into a unified system.

The first component, the user interface, was implemented using JavaScript. This component serves as the front-facing aspect of the system, providing a visual interface for users to interact with and input their data. The user interface was designed to be intuitive and user-friendly, in order to demonstrate in the best way how the algorithm works.

The gateway component serves as a critical intermediary between the user interface and the validators. It acts as a central hub that houses the algorithm's implementation and facilitates the seamless flow of data and operations. The development of the gateway involved in-depth planning and coding to ensure robust functionality and efficient execution of the algorithm. Furthermore, the gateway component also establishes and manages the connection with the validators, allowing for the exchange of data and results. This integration ensures a smooth and reliable interaction between the algorithm, validators, and the overall system.

## **4.3 Used Technology**

*5.3.1 Web3 and Ethers Libraries*

web3 and ethers are collections of libraries, which allow interaction with a local or remote Ethereum node, using HTTP. It is a JavaScript library that allows developers to interact with the Ethereum blockchain through their web applications.

Ethers.js is more lightweight and developer-friendly. Ethers.js is compatible with both Node.js and web browsers.

Ethers.js is also a more powerful and flexible library, it has a lot of functionalities that are missing from web3.js. it also has a lot of built-in functionalities like signing, contract deployment, contract interactions, and more.

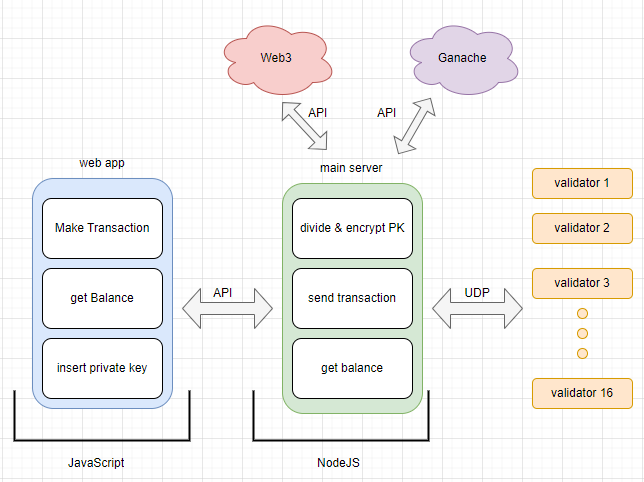
We will use Ethers.js in order to create and sign the transaction and send it to the blockchain nodes.

*4.3.2 Ganache*

Ganache is a development tool and a personal Ethereum blockchain emulator that allows you to create a local blockchain environment for testing and development purposes. It provides a set of pre-funded accounts that you can use for transactions and interactions with smart contracts.

Ganache is a development tool and a personal Ethereum blockchain emulator that allows you to create a local blockchain environment for testing and development purposes in decentralized applications in a controlled and efficient environment. With Ganache, we set up a local blockchain network on our machines, emulating the behavior of the Ethereum network. This local network provides a sandboxed environment where we test and simulate without incurring any costs or affecting the live blockchain. Ganache comes equipped with a set of pre-funded accounts, enabling us to simulate transactions and interactions with smart contracts. It offers features such as fast block mining, deterministic behavior, and a user-friendly interface for inspecting and debugging transactions.

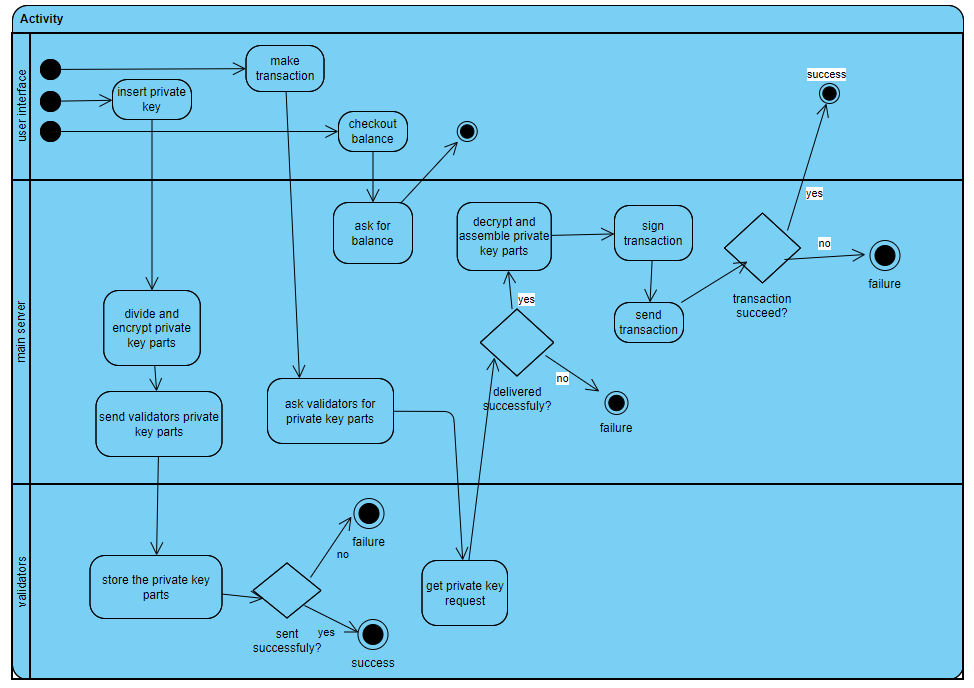
## **4.4 Architecture Diagram**

****

In the architecture diagram, we can observe the web app which serves as the user interface, allowing users to interact with the system. The main server, which is responsible for executing the logic of the algorithm, is also depicted. In order to protect the confidentiality of private keys, we used symmetric encryption algorithms. The remaining servers are utilized solely for storage purposes.  
The web application to be implemented in the deployment will be that of a cryptocurrency exchange. It should be noted that this web application is being developed solely for demonstration purposes.

## 

## **4.5 Activity Diagram**



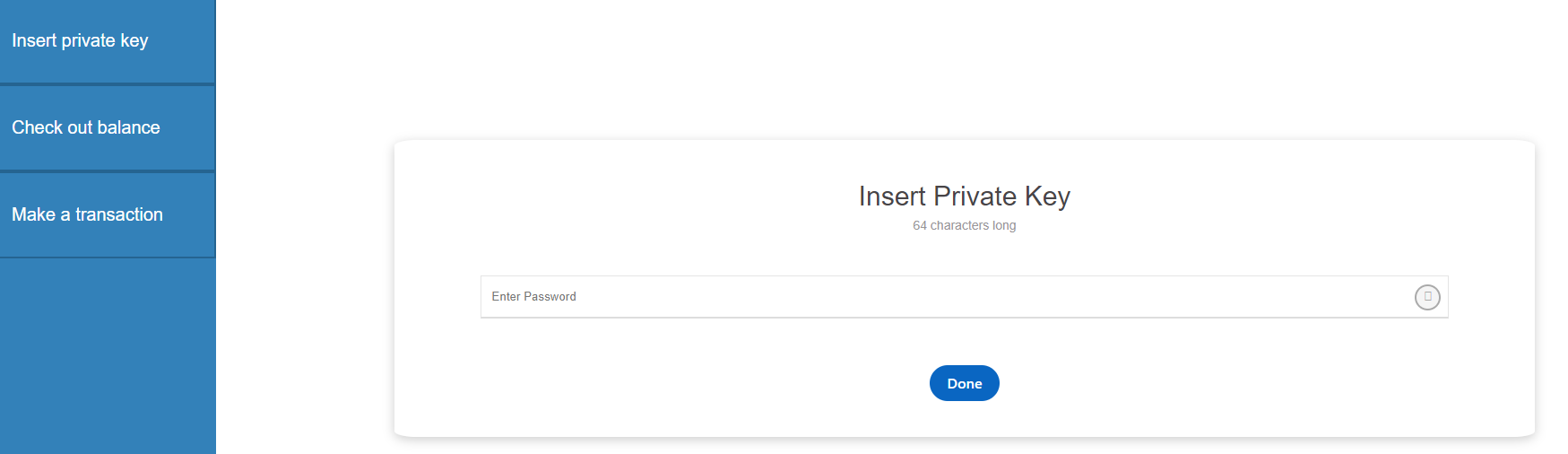
In the system, users have the ability to perform three distinct actions.

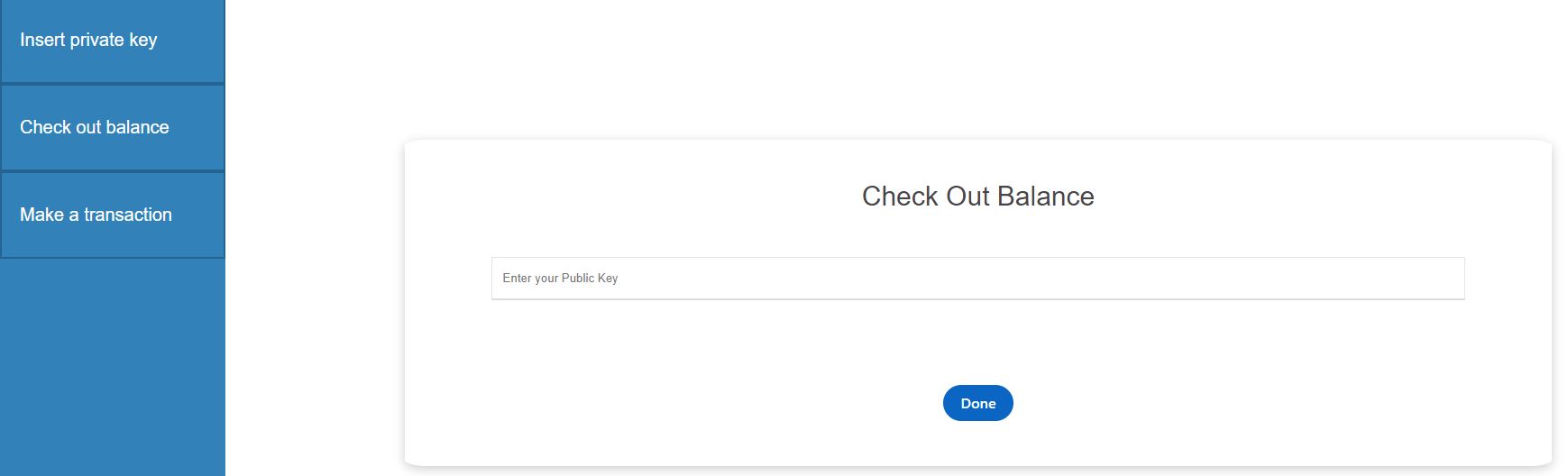
The first action is "insert private key," where the user provides their private key. Upon receiving the private key, the system divides it into smaller parts and distributes these parts across 16 different servers for enhanced security and redundancy.

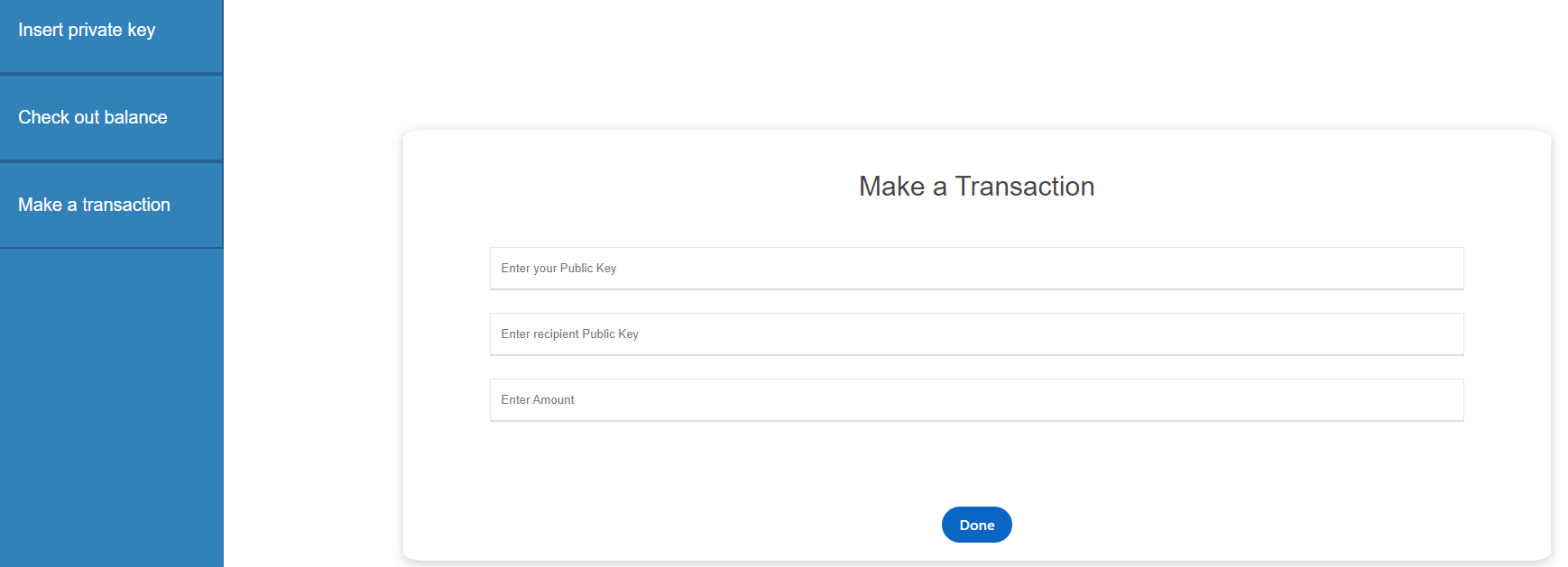
The second action is "make a transaction." In this scenario, the user provides their public key, the recipient's public key, and the desired amount to transfer. The main server, upon receiving this information, initiates a request to the validators, requesting them to send their respective parts. Once all the necessary parts are obtained, the main server combines the parts to reconstruct the complete private key, signs the transaction using the private key, and sends the signed transaction to the blockchain network for processing and recording.

The third action is "get balance," where the user provides their public key. The main server, upon receiving this request, interacts with the blockchain network (specifically, the ganache network) to retrieve the balance associated with the provided public key. This allows the user to obtain information about their account balance in the system.

## **4.6 Web Screens**

**

**

**

# **5 Evaluation/Verification Plan**

The web application serves as a test platform to evaluate the functionality and effectiveness of the proposed algorithm. It allows us to simulate real-world scenarios and test the algorithm's ability to handle different types of transactions, security breaches, and other potential issues. This allows us to identify any bugs, weaknesses, or areas for improvement, and make any necessary adjustments before implementing the algorithm in a live environment. Additionally, it enables us to demonstrate how the algorithm works and explain how it improves the security of the system. In our evaluation of the system, we assess the system's ability to create valid transactions. This will include generating test transactions using a variety of different inputs and configurations, and verifying that they are correctly signed and accepted by the blockchain system. To do this, we use a test Ethereum blockchain environment and create test transactions using the system. Once the transactions have been submitted, we verify that they are correctly signed and accepted by the blockchain. This will involve checking that the transactions are properly formatted and that they include the correct signatures, as well as verifying that the transactions are recorded on the blockchain as expected. By testing the system's ability to create valid transactions, we can ensure that the private key is functioning correctly and that the system can sign transactions as expected. This is an important step in the evaluation process.

some of the test cases we implemented:

| Test number | Test case | Expected result |
| --- | --- | --- |
| 1 | Dividing the private key | division should be according to the algorithm |
| 2 | Encrypting and decrypting the private key parties | each part should be encrypted |
| 3 | Sending the private key parties to the correct servers | each server should have its part |
| 4 | making a new transaction | the transaction is accepted in blockchain |
| 5 | making a new transaction in case one of the servers is down | the transaction is accepted in blockchain |
| 6 | making sure the transaction is recorded in blockchain | we will see the current balance |

# **6 Results and Conclusions**

The results of the algorithm implementation yielded positive outcomes in terms of private key storage and reconstruction. Through the use of distributed servers and the application of secure cryptographic techniques, the algorithm successfully mitigated the risk of unauthorized access to the private key. The division of the private key into multiple parts and their storage on different servers enhanced the security and resilience of the system. Moreover, the use of encryption algorithms, such as AES-256-CBC, ensured the secure transmission of the private key parts. The verification process demonstrated the accuracy and integrity of the reconstructed private key, as validated through consistency checks and cryptographic verifications. These results underscore the effectiveness of the algorithm in providing a secure and reliable approach to private key storage. In conclusion, the algorithm offers a robust solution for private key storage, protecting sensitive cryptographic assets while ensuring accessibility and resilience in distributed environments.

# **7 User Documentation**

## **7.1 General Description**

The system is designed as a secure and distributed infrastructure for private key storage and management. It employs a multiparty computation (MPC) approach, where the private key is divided into multiple parts and distributed across different servers. This decentralization enhances security by reducing the risk of unauthorized access to the complete key. The system utilizes advanced encryption algorithms, such as AES-256-CBC, to ensure secure transmission and storage of private key parts.

The system consists of several components, including a main server (gateway) responsible for coordinating the private key reconstruction process. Additionally, multiple servers are employed to store the individual parts of the private key securely. Communication channels between these servers are established using secure protocols, such as SSL/TLS or IPSec, to protect data in transit.

The system caters to two primary user groups: developers deploying the system in cryptocurrency exchanges and private users seeking enhanced security measures.

Developers who implement the system within cryptocurrency exchanges are proficient in software development and possess a deep understanding of blockchain technology. Their role involves integrating the system into the exchange's infrastructure, configuring it to meet specific requirements, and ensuring seamless compatibility with existing components. These developers aim to enhance the security of private key management within the exchange, safeguarding user assets and mitigating the risk of unauthorized access.

Private users form another key user group of the system. These individuals are concerned about the security of their cryptocurrency holdings and seek robust measures to protect their private keys. By utilizing the system, private users can benefit from advanced security features such as distributed key storage and encryption. They have the opportunity to secure their private keys across multiple servers, reducing the risk of key compromise and bolstering the overall security of their digital assets.

In summary, the system caters to both developers deploying the system in cryptocurrency exchanges and private users seeking heightened security for their private keys. Developers aim to enhance the security posture of exchanges, while private users desire greater protection for their cryptocurrency holdings.

## **7.2 Usage Scenarios**

The system supports several usage scenarios that cater to different user actions and objectives. The following scenarios highlight common interactions within the system:

Insert Private Key:

In this scenario, the user initiates the process of inserting a private key into the system. This private key can be generated externally or obtained from an existing cryptocurrency wallet. By inserting the private key into the system, the user ensures that it is securely stored and protected using the system's advanced security mechanisms.

Make Transaction:

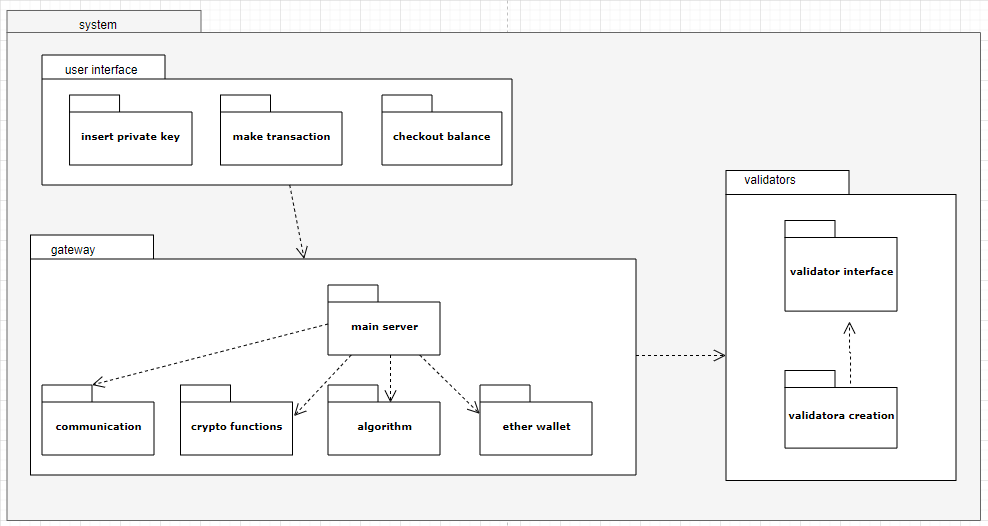
Once the private key is securely stored in the system, users can proceed with making cryptocurrency transactions. This scenario involves initiating and authorizing transactions using the stored private key. The system facilitates the signing of transactions using the private key, ensuring their integrity and authenticity before broadcasting them to the blockchain network.

Check Out Balance:

Users can also utilize the system to check the balance of their cryptocurrency holdings. This scenario allows users to retrieve information about their account balances and transaction history associated with the stored private key. By leveraging the system's capabilities, users can conveniently monitor their cryptocurrency assets and stay updated on their account balances.

# **8 Maintenance Guide**

## **8.1 Package Diagram**



# **9 User Help**

To utilize the system, you need to follow these steps:

Download the Projects: Obtain the frontend, backend, and validators projects, which are available for download. Ensure that you have the necessary files and folders for each project.

Open Projects in VSCode: Open each project in a separate instance of Visual Studio Code (VSCode). This allows you to work on each project independently and simultaneously.

Backend Setup:

Navigate to the backend project in the file explorer within VSCode.

Open a terminal within the backend project.

Run the command "npm install" in the terminal to install the required dependencies.

After the installation is complete, run the command "npm run dev" in the terminal. This will start the backend server.

open another terminal and install ganache:  
‘npm install -g ganache-cli’

Afterward run ganache:  
‘npx ganache-cli --deterministic --account\_keys\_path accounts.json --port 8545’

Frontend Setup:

Switch to the frontend project in the file explorer within VSCode.

Open a terminal within the front-end project.

Run the command "npm install" in the terminal to install the necessary dependencies.

Change the directory to "src/backend/" using the command "cd src/backend/" in the terminal.

Run the command "node server.js" to start the frontend server.

Validators Setup:

Open the validators project in a separate instance of VSCode.

Open a terminal within the validators project.

Run the command "npm install" to install the required dependencies.

Execute the command "node utils.js" to run the validators.

the ganache will generate private and public keys.  
you can use them or use these:  
public key: 90F8bf6A479f320ead074411a4B0e7944Ea8c9C1  
private key: 4f3edf983ac636a65a842ce7c78d9aa706d3b113bce9c46f30d7d21715b23b1d  
public key account to send transactions to: FFcf8FDEE72ac11b5c542428B35EEF5769C409f0

in each account its initialized to be 100 ETH

By following these steps, you can set up the frontend, backend, and validator projects for the system. This allows you to run the necessary servers and perform the required operations for using the system effectively.

# **10 References**

1. cryptopedia website (2022): https://www.gemini.com/cryptopedia/public-private-keys-cryptography
2. Daniel Escudero (2022) “An Introduction to Secret-Sharing-Based Secure Multiparty Computation”
3. Giancarlo Giudici, Alistair Milne, Dmitri Vinogradov (2019) “Cryptocurrencies: market analysis and perspectives”
4. Peter M. Krafft, Nicolás Della Penna, Alex “Sandy” Pentland (2018) “An Experimental Study of Cryptocurrency Market Dynamics”
5. Satoshi Nakamoto (2008) “Bitcoin: A Peer-to-Peer Electronic Cash System “
6. Sinergija University, Faculty of computing and informatics, Bijeljina, BiH (2019) “COMPARATIVE ANALYSIS OF CRYPTOCURRENCY WALLETS VS TRADITIONAL WALLETS “
7. Usman W. Chohan, MBA, PhD (2022) “The Problems of Cryptocurrency Thefts and Exchange Shutdowns”
8. Magazine by coin telegraph website: https://cointelegraph.com/magazine/crypto-exchange-hacks/

figures:

figure 1:<https://www.hedgewithcrypto.com/cryptocurrency-exchange-hacks/>

figure 2: <https://www.zdnet.com/article/2019-saw-more-cryptocurrency-hacks-than-any-other-year/>

figure 3: <https://en.wikipedia.org/wiki/Peer-to-peer>

figure 4: <https://stakey.club/en/verifying-digital-signatures/>

figure 6:

<https://securityboulevard.com/2020/11/symmetric-encryption-algorithms-live-long-encrypt/>

figure 7:

<https://cheapsslsecurity.com/blog/what-is-asymmetric-encryption-understand-with-simple-examples/>

projects git repository - <https://github.com/nitzan933/security-solution-for-private-key-storage>