Blockchain Project Report

Konrad Romański
Mikołaj Poniży

June 14, 2023

Contents

1 Introduction

Blockchain technology has revolutionized the way we think about financial transactions and data integrity. This report takes a closer look at a simple blockchain project, which serves as a prototype for a decentralized transaction system. The project leverages the power of various technologies such as NestJS, a progressive Node.js framework, TypeScript, a statically typed superset of JavaScript, PostgreSQL, an open-source object-relational database system, Swagger for API documentation, and npm for package management. The consensus algorithm used in this project is the Proof of Work (PoW) method, which is a cornerstone of many existing cryptocurrencies. This report provides a comprehensive understanding of the project's structure, functionality, and underlying technologies.

2 Technologies Used

2.1 NestJS

NestJS is a progressive Node.js framework for building efficient and scalable server-side applications. It uses modern JavaScript, is built with TypeScript, and combines elements of OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming). More details can be found at `https://github.com/nestjs/nest`.

2.2 TypeScript

TypeScript is a statically typed superset of JavaScript that compiles to plain JavaScript. It offers classes, modules, and interfaces to help build robust components.

2.3 PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system. It is used in this project to store the blockchain data. There is a table Block in the db.

2.4 Swagger

Swagger is used for API documentation. It helps to understand the structure and usage of the API endpoints. Here is used as a basic frontend - UI for the application.

2.5 npm

npm (Node Package Manager) is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command-line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.

3 Approach

The approach used in this project is the Proof of Work (PoW) method. PoW is a consensus algorithm used in blockchains that requires a certain amount of effort to be expended before a new block is added to the blockchain. This effort is in the form of computational work done by the system to solve a complex mathematical problem. The solution to the problem is difficult to find but easy to verify. This asymmetry ensures the security and integrity of the blockchain by making it prohibitively expensive and time-consuming for malicious actors to alter the blockchain.

In the context of this project, the PoW concept is implemented in the mining process of new blocks. When a new transaction is made, it is added to a block. This block needs to be mined, i.e., a specific number (nonce) needs to be found that, when hashed with the block's content, produces a hash that meets certain conditions (e.g., a specific number of leading zeros). This process requires significant computational power, hence the term "proof of work". Once the nonce is found, the block is added to the blockchain. The difficulty level for mining a block is set to 4, which means the hash of a block must start with four zeros.

4 Endpoints

The project has several endpoints which include:

- GET /blocks: This endpoint is used to get all the blocks in the blockchain.
- POST /transaction: This endpoint is used to add a new transaction to the blockchain.
- GET /validate: This endpoint is used to validate the blockchain.
- GET /key: This endpoint is used to get a pair of cryptographic keys.

5 Transaction Signing

Transaction signing is a crucial aspect of blockchain technology as it provides a secure way of confirming the identity of the participants in a transaction. It involves the use of cryptographic keys to sign and verify transactions. In the context of this project, the Elliptic Curve Digital Signature Algorithm (ECDSA) is used.

The process of transaction signing in this project can be broken down into the following steps:

1. **Key Generation:** A pair of cryptographic keys (private and public) is generated for each participant in the transaction. The private key is kept secret by the participant, while the public key is shared with others. Here the key generation can be done using the proper endpoint.

2. **Transaction Creation:** When a participant wants to send a transaction, they create a transaction object that includes the amount, sender's public key, and receiver's public key. This transaction object is then hashed to create a transaction hash.

3. **Signing the Transaction:** The sender uses their private key to sign the transaction hash. This creates a digital signature, which is a unique piece of data that can only be produced by someone who possesses the private key. In our project hashing with signing is done by the server. - the 'sign' function of the elliptic curve library is used to sign the transaction hash.

4. **Verifying the Transaction:** Anyone who receives the transaction can verify its authenticity by using the sender's public key to check the digital signature. If the signature is valid, it means that the transaction was indeed created by the sender and has not been tampered with. We are using 'verify' function of the elliptic curve library to verify the signature.

5. **Adding the Transaction to the Blockchain:** If the transaction is valid, it is added to a block. Once the block is mined (i.e., a nonce is found that satisfies the Proof of Work condition), the block, along with the transaction it contains, is added to the blockchain. In case of keys mismach - signed transaction cannot be validated using sender's public key - the transaction is not added and the proper message is returned to the sender/

This process ensures the integrity and non-repudiation of the transactions. The use of private and public keys allows for secure identification of participants, and the digital signature ensures that the transaction cannot be altered once it is created. However, it is important to note that this implementation of transaction signing is basic and serves only to illustrate where it could be if the application will be under further development. In a real-world application, additional measures such as key management and protection against various attacks would need to be implemented.

6 Blockchain Structure

The blockchain structure in this project consists of blocks, and each block contains multiple transactions. Each block is linked to the previous block through the previousHash field. The structure of the blockchain is as follows:

```
[
  {
    "id": 12,
    "timestamp": "2023-06-14T22:00:47.865Z",
    "nonce": "9493060",
    "previousHash": "",
    "transactions": [
      {
        "amount": 0,
        "sender": "0425044fc43c7279ea5c79b5deea3a004521e0906226cd59bf13f8ad088f4
        "receiver": "047400ec9bbf99eeddda95621646d2eacbfefda6c8aba0f563e202943db
        "hash": "16ef80c0b19854791fb0dcda0b33258e7b756c523b3975c824a22dcbc3609c0
      },
      ...
    ],
    "hash": "00001342cccdadb4696fdeaa511bd3b5db0e16c206c3a2c4bebba7699b89e25d"
  },
  ...
]
```

7 Validation

Validation is a critical aspect of any blockchain as it ensures the integrity and consistency of the data stored within. In this project, a validation function is implemented to check the integrity of the blockchain. It verifies that the hashes of the blocks are correct and that the blocks are properly linked, ensuring that no block has been altered or inserted illegitimately. The validation function iterates over the entire blockchain, checking the hash of each block and comparing the 'previousHash' field of the current block with the hash of the previous block. If any discrepancies are found, the blockchain is invalid. This validation process can be triggered through the '/validate' endpoint.

8 Conclusion

This blockchain project serves as a testament to the power and versatility of blockchain technology. While simple in its design and functionality, it encapsulates the core principles of decentralization, data integrity, and consensus. It demonstrates how transactions can be securely signed and validated, how blocks are mined using the Proof of Work algorithm, and how data integrity is maintained through continuous validation. However, it is important to note that this project is a basic implementation and serves as a stepping stone towards more complex and robust blockchain systems. Future enhancements could include the implementation of a more sophisticated consensus algorithm, a more secure transaction signing process, a comprehensive key management system, and additional features such as smart contracts.

9 Future Work

While the project provides a solid foundation for a blockchain system, there are several areas where it could be expanded for further development.

9.1 Consensus Algorithm

The current system uses a simple Proof of Work consensus algorithm. Future work could explore other consensus mechanisms like Proof of Stake, Delegated Proof of Stake, or Byzantine Fault Tolerance, each having its own advantages and trade-offs.

9.2 Security Enhancements

While the project includes basic transaction signing, future work could focus on enhancing the security of the system. This could include more advanced cryptographic techniques, multi-signature transactions, or even quantum-resistant algorithms.

9.3 Smart Contracts

The addition of smart contracts - self-executing contracts with the terms of the agreement directly written into code - could greatly enhance the functionality of the system, allowing for more complex transaction types and decentralized applications.

9.4 Scalability Improvements

As with many blockchain systems, scalability can be a challenge. Future work could explore techniques for improving the scalability of the system, such as sharding or off-chain transactions.