Blockchain Technologies and Quantum Computation project
REPORT

Piotr Szymański, 141322

POZNAŃ 2023

Contents

Chapter 1

Introduction

1.1 Objective

The objective of the project was to implement a blockchain structure in a programming language of self choice. The implementation should allow user to use basic actions, such as: showing, adding the blockchain etc.

1.2 Blockchain implementation

1.2.1 Language

Python programming language was chosen to implement the blockchain. It was chosen due to author's large experience with it. It has also many highly accessible libraries proven to be useful while completing the task.

1.2.2 Data safety

To ensure the integrity and security of the data some external libraries shown in figure 1.1 were used.

```
20  from Crypto.Hash import SHA
21  from Crypto.PublicKey import RSA
22  from Crypto.Signature import PKCS1_v1_5
```

FIGURE 1.1: Modules used in the project

After creating a client, new pair of public and private key as well as signature is generated. On top of that a user has to log in with unique credentials to access the blockchain. The *Client* class as well as generation of the keys and signatures are shown in figure 1.2.

```
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)
```

FIGURE 1.2: Generating the key pair and signature

1.2.3 Consesus mechanism

To ensure the validity of the blocks proof-of-work was implemented. Proof-of-work is stored in *Block* class. It is used as verifier if the computations were indeed made and to ensure the block is valid. The computation that was picked for this program goes as follows:

1. check the hash of the last block
2. add nonce as a string at the end of the hash
3. hash the 2^{nd} step result
4. repeat steps 2 and 3 until you find a resulting hash that has an agreed numer of "1"-s in front of it

The simple implementation of above algorithm is shown in figure 1.3. The implementation is based on brute force approach. The difficulty variable indicates how many "1"-s at the begining are we looking for.

```
178    def mine(message, difficulty):
179        assert difficulty >= 1
180        prefix = '1' * difficulty
181        k = 0
182        global pp
183        for i in range(100000):
184            digest = sha256(str(hash(message)) + str(i))
185            if digest.startswith(prefix):
186                print ("after " + str(i) + " iterations found nonce: "+ digest)
187                pp = str(i)
188                return digest
189        return -1
```

Figure 1.3: Implementation of the algorithm

1.2.4 User interface and avaible actions

Simple user interface was prepared. After logging in it allows user to do following actions:

1. Add block/mine
2. Display blockchain
3. Add/display transaction

The results of the listed actions are shown below

```
What do you want to do [1]-mine/add block [2]-display: [3]-add transaction 1
after 990 iterations found nonce: 11117e96e2907402e8ba43f87dc7eda8928d660ac01c111577f5519023e5d534
you can now add block [press enter]
```

Figure 1.4: Adding an extra block

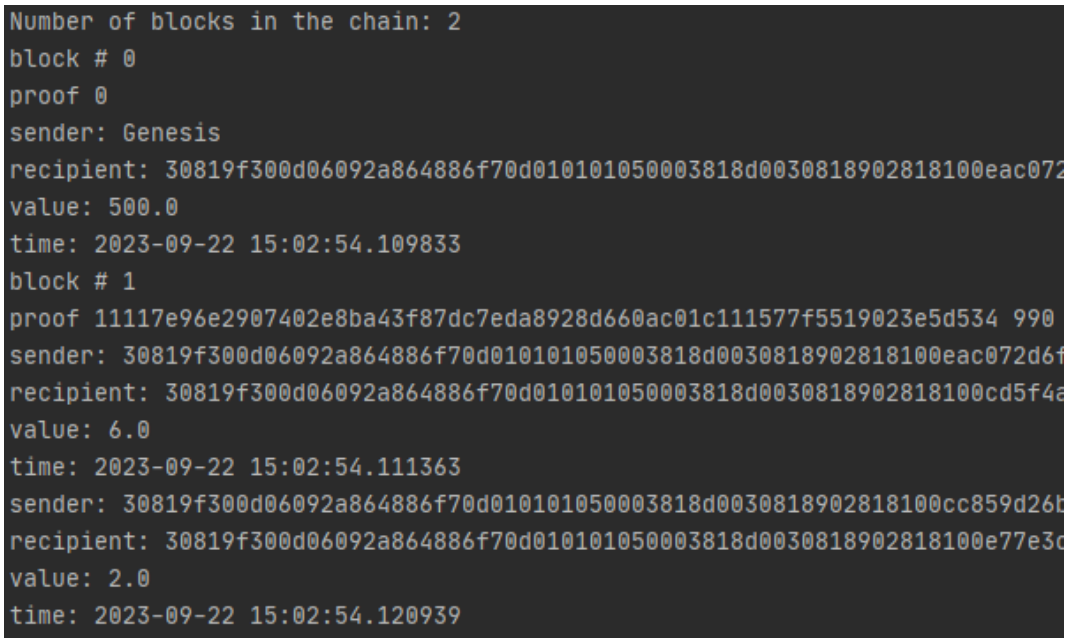

Figure 1.5: The display of the blockchain after adding 1 block to it

Figure 1.6: The display of the transactions after adding 1 transaction

Chapter 2

Conclusions

2.1 Goal and effectiveness

The goal was to prepare a project that shows how does blockchain work and allow user to take some actions with it. The goal was achieved and a working program was made. Although some upgrades could be made. The effectiveness of the mining is not satisfactory. The brute force approach in my case was just a tool to show how does mining work but for larger computations f.e. looking for hash that has twenty "1"-s at the begining I couldn't reach the resolution in measured time.

2.2 Safety

Each new block connects to all the blocks before it in a cryptographic chain in such a way that it is nearly impossible to tamper with. All transactions within the blocks are validated and agreed upon by a proof-of-work mechanism. Other part of safety lays in RSA. Keys of enough length should be picked not to expose the safety of the program. As the project was meant to show the mechanisms of blockchains the author did not implement authentication mechanism of correct safety. The logging feature is simple and to use it in safer mode, some adjustments have to be made regarding logins and passowrds.

2.3 Conclusion

The project was very interesting way to learn about blockchain. The author had almost no knowledge about blockchains before classes and project. At the end of the project he learnt a lot about the blockchain technology and this could prove to be very valuable in the future.

This report is a shorter, written version of the recorded discussion which can be watched on youtube: `https://www.youtube.com/watch?v=UelioKfV9S0&t=6s&ab_channel=czymansky`

Poznan University of Technology
Faculty of Computing and Telecommunication
Institute of Computing Science

Typeset using LaTeX.