Proof of university certificate using blockchain technology

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Electrical and Automation Engineering
Valkeakoski

Spring 2018

Kaltyshev Mikhail

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and Automation Engineering
Valkeakoski

Author Kaltyshev Mikhail **Year** 2018

Title **Proof of university certificate using blockchain technology**

Supervisor(s) Timo Viitala

ABSTRACT

Blockchain technology is a global breakthrough with major consequences, which will qualitatively change many areas of life. This thesis explains why this technology is unique and how it can be applied in spheres other than the Bitcoin.

This thesis is divided into two parts – a theoretical and a practical one. In the theoretical part, the concepts of the Blockchain technology are explained including its working principles, the most important technological solutions and advantages over classical systems.

The practical part uses the knowledge from the theoretical part in the development of an application based on the Multichain platform version 1.0.4. This application is designed to collect the certificates of students from universities around the world in a single trusted ledger based on the blockchain to drastically reduce the forgery of these documents.

It is important to understand that this version of the program is a beta version and requires further development to develop into a full-fledged software solution in the future.

Keywords Blockchain, Multichain, cryptography.

Pages 33 pages including appendices 40 pages

CONTENTS

LIST OF FIGURES

Appendices

ABBREVIATIONS USED

API	Application Programming Interface
OS	Operating System
POW	Proof of Work
POS	Proof of Stake
RPC	Remote Procedure Call
SHA	Secure Hashing Algorithm
HTTP	Hyper Text Transfer Protocol
TLS	Transport Layer Security
P2P	Peer-to-Peer
DDOS	Distributed Denial of Service
IP	Internet protocol
PBFT	Practical Byzantine Fault Tolerance

1 INTRODUCTION

In 2009, an anonymous hacker (or a group of hackers), under the pseudonym of Satoshi Nakamoto, created the first digital currency called *Bitcoin* explained in the white paper called "Bitcoin: A Peer-to-Peer Electronic Cash System". The most innovative in this virtual currency was the principle of performing the transactions between people. Payments which are done using the Bitcoin are recorded in a public ledger and stored on many Bitcoin user's computers. In this system, money was only a tool of accounting, a method of abstracting value, assigning property and providing funds for transactions. Historically, money was used to fulfil these functions. Cash is hard enough to copy, so there is no need for the full record of who owns a certain part of the money supply. Nevertheless, creating a table with the information of who owns the certain amount of money makes coins and bills unnecessary. Bitcoin completed the conversion from physical to digital assets by setting up a single universal digital register, called blockchain (Nakamoto, 2008).

The blockchain is a reliable way of storing data about transactions, contracts, basically everything that needs to be written down and verified. Members of the network are anonymous persons, called nodes. The communication in the network utilize cryptography to identify the sender and recipient without disclosure of the sensitive information. Initially, the blockchain was associated only with the Bitcoin crypto currency, but nowadays the situation changes dramatically. The blockchain is ready to fundamentally change the financial system of the state currencies and simplify the work of medium and large businesses. Different financial organizations are in process of implementation and testing of the blockchain for digital records, tracing and managing the transactions. (Swan, 2015)

Blockchain is a new and rapidly developing technology. There are many terms that are not familiar to a person for the first time encountered with such technology. In consequence of this, a dictionary of the most important terms created by BlockchainHub (n.d.) was added to understand the processes and concepts of the Blockchain:

Bitcoin – first created digital currency which runs across the World Wide Web.

Block- record of recent transactions added to the end of the blockchain.

Block signature – digital sign which verifies who created a new block.

Cryptographic Hash Function – is a mathematical mechanism which gets an input (any kind of digital data; a text, video or picture) and produces a single fixed length output.

Full node – computer or server connected to the blockchain network and has an exclusive permission to perform actions in the network.

Genesis block– the first block created in the blockchain.

Mining – the process handled in blockchain to add and confirm new transactions in the block.

Node – any computer or server connected to the blockchain network.

Permissioned blockchain – type of the blockchain which allows to perform certain actions only to identifiable participants.

Transaction fee – small payment per transaction to make sure it will be included in the next block and confirmed.

2 PROJECT CONCEPT

The problem of university certificate forgery is distributed around the world. Bear & Ezell (2012, p. 1), participating in diploma mills investigation, found the following:

One international diploma mill, with offices in Europe and the Middle East and mailing addresses in the UK, run by Americans, has sold more than 450,000 degrees—bachelors, master's, doctorates, medicine, and law—to clients worldwide, who did nothing more than write a check. Their revenues exceeded US$450,000,000.

To prevent the expansion of fraudulent diplomas, the solution based on the blockchain proof-of-existence concept could influence the situation. The essence of the application's idea is to declare the fact of the university issuing a diploma about the graduation to the blockchain and give the possibility to confirm the diploma by the system at the request of the employer, state organization or any other institution.

The workflow of the application could be described as follows:
When the new university representative accesses the application, the registration request must be submitted for university authorization. After carrying out the necessary checks, the login and password are provided for the use of the system. Every university connected to the system fills in a form with the name, student ID number, diploma ID and upload a scan copy of the degree certificate. The document is hashed and timestamped,

the form information is encoded and converted to a hexadecimal representation and published to the blockchain. As soon as transaction is confirmed by the system, the diploma information is visible in application. Any further forgery attempt causes a discrepancy between the hash functions of the original document and the document with unauthorized changes.

A student, graduating from an educational institution, receives a copy of the original document and the information stored in the system. It includes a hash function that acts as a digital signature and a sequence number in the system, the transaction number, and the time when the document was loaded into the system. If it is necessary to confirm the passage of the studies at the university, this information can be provided to the employer for further verification.

There are two ways to verify the authenticity of the diploma:

- database search using a hash function
- upload of the document to the system

First method involves entering a hash function in the search field. The system looks for matches in the database and outputs the results in the form of brief information and a link to the extended version with the complete report. Second method involves document upload into the system and forming a hash function. Next, the system compares the hash functions of existing documents in the blockchain and the hash function of the uploaded file. If a match is found, the user will be provided with a full report. The graphical scheme of the application workflow is shown in Figure 1.

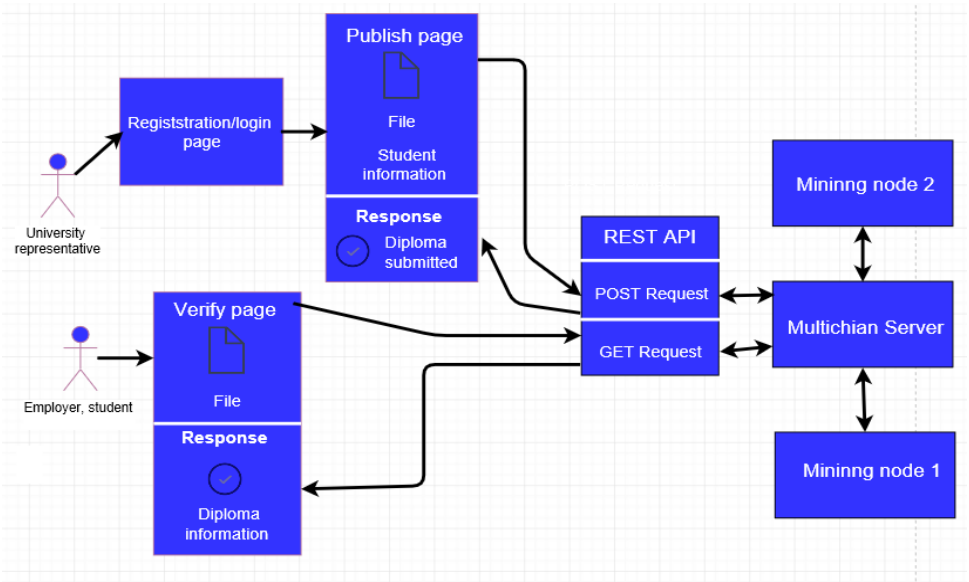


Figure 1. Application workflow

3 BLOCKCHAIN: THE NEW TECHNOLOGY OF TRUST

3.1 Decentralised networks

Decentralization is one of the most important concepts needed to have a clear picture about the technology of the blockchain (Imran, 2017, p.557).

There are two definitions of the blockchain:

- Database distributed between participants(nodes) of the network;
- Chain of blocks containing the information about the transactions;

Both definitions are relevant, but do not give a proper understanding. It is necessary to compare this technology to the computer network architectures which exist in the modern IT market (Imran, 2017). There are three types of architectures:

- Client-server network
- Decentralized networks
- P2P network

The schemes of the architecture types mentioned above can be seen in Figure 2.

Figure 2. Three kinds of networking (Sanjeeb, 2017).

Networking in the first way implies centralized control of everything: applications, data, access. All system logic and information are hidden inside the server which reduces the processing costs. This method has become the most widespread today. However, if the valuable information is stored on one or several computers at home or at the enterprise then nothing guarantees its safety. Such database can be hacked, changed, deleted, the computer may fail or the server with the information can undergo a DDOS attack. Even a simple power outage in the server room will block the access to the database (Imran, 2017, p. 35).

The second type has a database present on the computers or servers connected to the network and those are located in different geographical regions. The network participants do not have a data communication links (Imran, 2017, pp.36 & 98).

The P2P network do not have a main device and all participants have equal operating rights. In this model each user is not only a consumer, but the service provider (Imran, 2017, pp. 20 & 113).

3.2 Principles of Blockchain operations

In essence, the blockchain can be seen as a chain of blocks and it can be compared to a puzzle. A block is an array of data, containing information about the transactions that entered the network after the creation of the previous block. Each new data block is attached to the previous one by means of complex mathematical and cryptographical algorithms, which makes it possible to consolidate the blocks. Each block is represented by a base and a header. The base of the block is a certain order of information records. The header is the key for the previous block sequence and a hash function. To create a new block, system needs to calculate its cryptographic fingerprint (hash) that satisfies certain conditions. All transactions in the block are represented as rows in the hexadecimal format (raw transaction format), which are hashed to obtain transaction identifiers (txid) (Imran, 2017, pp. 19-23). An example of the blockchain structure is shown in Figure 3.

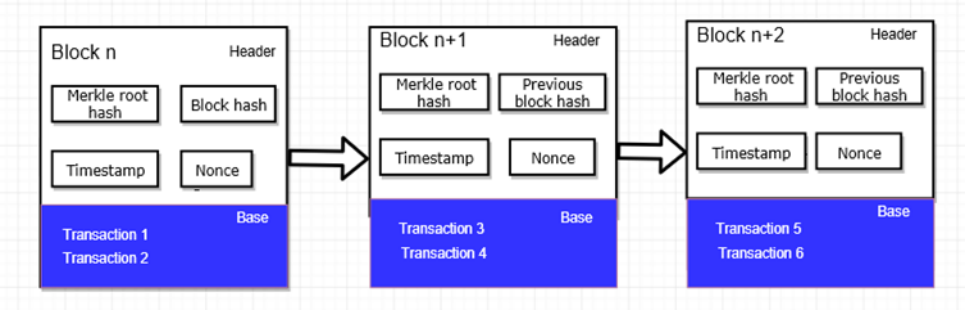


Figure 3. Structure of the blockchain

The system is distributed among all computers (nodes) connected to the network where each computer stores the copy of the blockchain information. Any attempt to interfere into the network is impossible because it requires disabling all system computers or gaining the control of 51% of nodes (51% attack). Each member of the network has equal rights and the capacity of the system is determined by the number of users (Imran, 2017, pp. 16 & 574).

There are two sides to the coin here: openness and security. Information about each block is available for any participant of the system, allowing to track the entire chain of information data with all accuracy and reliability.

Therefore, users do not have to guess how reliable the information in the system because the blockchain itself is the guarantee of these properties. The slightest change in the data affects the resulting hash. This is one of the main rules of the Blockchain technology (Imran, 2017, pp. 24-25).

3.3 **Cryptography**

This science uses mathematics for encryption and decryption of information. The methods and algorithms developed by the cryptographical scientists allow sensitive information storage and distribution through the unreliable channels in the way that it can be read only by the intended user (Imran, 2017, p. 51).

Cryptography is the cornerstone of the blockchain, which ensures the operation of the system. The architecture of the blockchain assumes that the trust between the participants of the network is based on the principles of mathematics and economics. Cryptography also guarantees security, which is based on the transparency and verifiability of all operations, rather than on the industry's traditional perimeter security system. Various cryptographic techniques guarantee the invariability of the transaction log of the blockchain, resolve the authentication task and control the access to the network and data in the blockchain (How Does Cryptography Protect the Blockchain?, 2017).

3.3.1 SHA-256 hashing function

Information reliability and security of the blockchain is maintained on cryptographic hashes. Hashing is the process of converting an array of input data of arbitrary length into an output of bit string of fixed length. For example, a hash function can take a string with any number of characters and get a string with a strictly defined number of characters (digest) at the output (Imran, 2017, pp.86-87).

The Blockchain utilizes SHA-256 hashing function as the main principle of operations with new blocks. The SHA-256 hash function is based on the Merkle–Damgård construction where the initial value after the addition is divided into blocks and each block is spitted into 16 words. Each block of the message runs through the algorithm with 64 interventions. At each round, the conversion function from the words entering the block is specified. Two words from the message are converted by this function. The results are summarized and the result is a hash value. To process the next block, the result of processing the previous block is used (Imran, 2017, p. 89).

For example, the name Mikhail after the SHA-256 hash function converted using the online generator will look like this:

3F2DB9A5D690C31E01FD7748766C0E870E15447A32839EA13996D8B98
96DD20D.

Hash functions in the blockchain guarantee the inability to change information in previous blocks. Each new block of transactions refers to the hash of the previous block in the registry. The hash of the block itself depends on all the transactions recorded in the block, but instead of sequentially passing the hash functions of each transaction, the Merkle tree is used. The Merkle tree is a binary tree where all transactions are collected in one hash value. The end nodes are transaction hashes and internal vertices are the results of adding the values of the associated vertices. An example of such a tree is shown in Figure 4 with three transaction hashing process. Thus, hashes are used as a replacement for pointers in conventional data structures. For a given data set, the hash function produces one hash that have extremely important properties. There are no two different sets of information that would give the same hash function. Additionally, even the smallest change in the data set influences the output result of the hashing. Due to the use of hashes, the general condition of the blockchain system can be expressed by a single number: the hash of the newest block (Shaan, 2017).

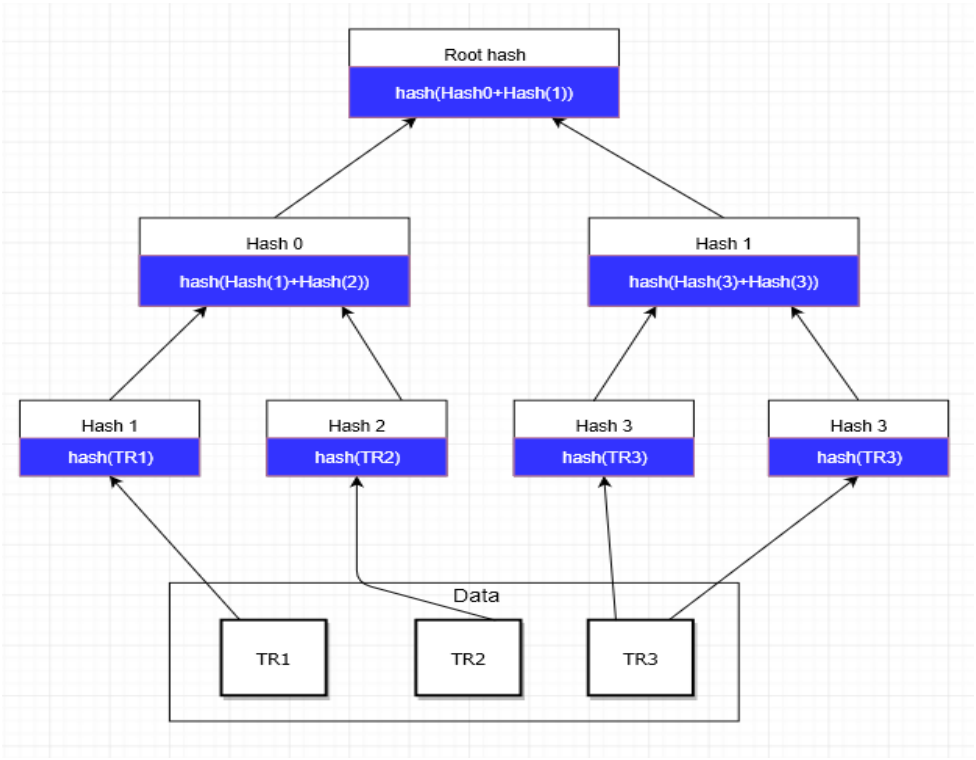


Figure 4. Example of a Merkle tree

The tree is constructed, according to the Shaan (2017), as follows:

1. Transactions, arranged in the block (hash (TR1), hash (TR2), hash (TR3)), are hashed.

2. The sum of previous hashes is hashed, e.g. hash (hash (TR1) + hash (TR2)). Since the Merkle tree is binary, the number of elements on each iteration must be even. Therefore, if the block contains an odd number of transactions, the latest one is duplicated and formed with itself: hash (hash (TR3) + hash (TR3)).
3. Next, the hashes are recalculated from the sum of the previous hashes. The process is repeated until a single hash (root hash) is obtained. It is a cryptographic proof of the integrity of the block. The value of the root hash is fixed in the header of the block.

3.4 **Consensus algorithms**

A consensus algorithm is a well-proven solution for many distributed systems (for example, NoSQL databases). It is a process where distributed nodes are trying to reach an agreement on the ultimate data state. The task of consensus where nodes can behave "in a bad way" was first formulated in the 80s of the last century and methods for its solution appeared in the late 90's. It is called a Byzantine consensus mechanism. Byzantine general's problem is one of the most popular cryptographical tasks where network users are located at a distance and coordinated from a common control centre. The main difficulty is that any participant of the network, even the control centre, may become a traitor. (Imran, 2017, pp. 12-13).

The idea of Byzantine consensus could be described as follows:
Byzantium on the eve of the battle. The army of the Byzantines consists, for example, of five legions located at a sufficient distance from each other which makes direct communication impossible. Each general obtains tasks to attack or retreat from control center. The task is clear, but there is no guarantee that there are no traitors who will carry out the order on the contrary way. Moreover, there is no guarantee that the control center is not a traitor, having sent various orders to different generals. The principle of consent is that all the faithful generals come to the same decision, excluding the data received from the traitor (Lamport, Pease & Shortak, n.d). The principle of information exchange is as follows:

- Each of the five generals sends each other a message about the number of people in the legion. The traitor, for disinformation purpose, sends the different information to each general.
- Each general forms a block where four digits received are included. Additionally, the information about the original data source is mentioned. This block is sent to other generals.
- In the end, each general has five blocks with numbers of people in legions. It is logical that four generals will have the same information about state of legions and one will have discrepancies.

The honest leaders continue execution of the order, excluding the traitor. There is one serious flaw in Byzantine consensus mechanism; the generals

know the source of the information comes from meaning that there is no anonymity (Lamport, Pease, & Shortak, n.d).

The blockchain offers a completely opposite solution. The participants of the network are unknown to each other in advance and have a freedom of choice to connect or disconnect from the network. At the same time, being a decentralized system, the blockchain has certain properties: censorship resistance (no one can prohibit the participation in the blockchain network) and objectivity (there is no need to trust certain authoritative source to determine the current version of the transaction log). The root of trust is in the blockchain itself (Imran, 2017).

Therefore, many different algorithms were developed to suit the blockchain system operation. There are two main consensus mechanisms distinguished: proof-of-work and proof-of-stake algorithms. Moreover, new consensus mechanisms for narrowly focused activities are actively developing, for example, the Practical Byzantine Fault Tolerance consensus algorithm. In the next chapters these algorithms are described in detail.

3.4.1 Proof of Work

In 1993, researchers Moni Naor and Cynthia Dwork offered the idea that access to a network resource would become possible only when performing a certain complex task. Three years later, Adam Beck realized this idea in the project called Hashcash, created to protect electronic systems from spam. The term "Proof of Work" appeared three years later and was described as an algorithm designed to protect servers from DDoS attacks (Dwork & Naor, n.d).

The POW consensus mechanism is similar to the reporting in the office. Employees regularly make reports for verification, confirming the fact of fulfilment of the specific task. Without this verification, the salary would not be paid since the result of assignment is not proved. The main idea of this consensus mechanism is that the nodes of the network, which are confirming the transactions, perform a complex computational work and the result can be easily and quickly verified by other nodes of the network. The first node, which completed all the necessary calculations receives a reward from the blockchain. All nodes are competing to be the very first node receiving the reward (Imran, 2017, pp. 129-134).

The algorithm of Proof of Work according to Imran (2017) works as follows:

- Transactions inside the blockchain are randomly grouped into separate blocks
- Miners confirm transactions, solving a complex task which requires powerful computational equipment
- Miner (or pool for mining), who first solved the problem, receives a reward for disclosing the block

- Confirmed transactions are added to the block

3.4.2 Proof of Stake

An idea of a Proof of Stake consensus algorithm was announced in 2011 in online forum called bitcointalk.org. The main idea is that the validators are not miners with more powerful equipment, but the users who own a large share of the internal coins of the network. In POS mining, the user who owns a larger stake becomes the creator of the new block, confirms transactions and gets a remuneration. (QuantumMechanic, 2011)

The Proof of Stake algorithm, proposed by QuantumMechanic (2011), works as follows:

- The user selects a crypto currency that uses the POS
- Buys coins and transfers assets to the crypto wallet
- User downloads a special program-miner
- Waits for the activation of the wallet and runs the program on PC
- The program performs the necessary calculations. After a fixed number of days, the earned funds arrive at the wallet

3.4.3 Practical Byzantine Fault Tolerance

The PBFT protocol is based on the fact that the leader node is honest, because the malicious actions freezes the work of entire system causing harm to all participants. If the leader node behaves in a way that contradicts with the system internal rules, there is a mechanism of changing the leader node in the protocol. When there is no progress within a given period, each node has the right to send a request to change the leader node. The decision to change the leader node is achieved by the overwhelming majority of nodes and the next leader is chosen from previously agreed list (Castro & Liskov, n.d.).

The consensus process according to Castro & Liskov. (n.d.) has three stages:

- Preliminary preparation. At this stage, the leader node declares the creation of a new block and sends out a preliminary message.

- Preparation. When message is received, each node checks the information in the block for correctness and sends a message of the block readiness forward to the other nodes.

- Implementation. When messages about the readiness are received from the vast majority, each node sends out a message about the execution of transactions in the block. Finally, each node expects a message about the execution from the vast majority of nodes

which would mean that the information proposed by the leader was accepted.

3.5 **Types of blockchains**

This chapter discuss the existing types of the blockchains, their key features and main differences.

3.5.1 Public blockchain

A public blockchain can be read by any user connected to the network. Every user has the right to form transactions. Transactions are protected by cryptographic consensus algorithms: Proof-of-Work or Proof-of-Stake. The public blockchain is controlled by the community of network members. All the users ensure the integrity of the network and the convenience of work. Network performance is achieved through protocol updates that prevent harmful changes. Therefore, the system allows to create blockchains with low maintenance expenses (Lee & Deng , 2017, pp. 439-441).

Main advantages are according to Byterin (2015) of public blockchain:

- Protection from the intervention of developers. The creators of the system can not affect it in any way and change the code or data at own discretion.
- Powerful network effect. Such an environment helps the developer to easily gather an extensive user base around the application because users quickly discover newly created ones.
- Security requires a small amount of funds. The attack on such system requires powerful computational resources which makes the attack simply unprofitable for intruders.

3.5.2 Private blockchain

Private is a type of the blockchain where the creation of blocks is centralised and all the rights to conduct such operations belong to one or multiple organisations. Common user can only read the information to conduct the audit of the system. Management of databases is delegated to the trusted nodes of the blockchain (Lee & Deng , 2017, pp. 381-382).

Private blockchains have certain advantages. First, low transaction costs since the validation is performed by trusted and high-performance nodes instead of thousands of user devices, as in the case with the public networks. Second, the blockchain can be configured in a way that the transactions per second index will be significantly higher than the public networks index. The only limitation in this case is the throughput of the

weakest node in the network. Another advantage of private blockchain is greater control of the system by the company. The bottom line is that private blockchain allows, for example, quicker update of functionality and development of the system according to the specific needs. Therefore, it is attractive for institutions working with registries and accounting systems since it forms a controlled and predictable environment in comparison with the publicly accessible (Byterin, 2015).

3.6 **Blockchain changing the world?**

Even though the blockchain technology appeared not such a long time ago, it has started to become embedded into projects around the world. There are several reasons why this will all lead to a success by Williams (2017):

- Decentralization. There is no server in the chain. Every participant is a server. These concept guarantees stronger stability of the application. The attempt to influence the work of the service requires bigger resources than in the centralized systems
- Transparency. Information about transactions, contracts and actions is kept publicly available
- Theoretical unboundedness. The blockchain can be supplemented with entries to the infinity. Therefore, it is often compared to a supercomputer
- Reliability. The conservation of new data requires consensus of the nodes. This allows to filter operations and record only legitimate transactions
- Trust in the system is mathematically approved

3.7 **Areas of blockchain implementation**

When Bitcoin appeared, the underlying technology was not considered for the use in other areas. Nowadays, it is used in different spheres which are not related to finances. The list of applications shown in the following contains the projects that have already been created and are in demand among people (Rosic, 2017):

SMART CONTRACTS

Blockchain technology allows to create the distributed applications where the code is executed on the machines of network participants. The principle of the work of contracts is very simple:

If the defined event happens, some action automatically performs.

This technology allows to make contracts in the form of executable code, which will be executed in any case; it is protected from human factors and violations of rules.

IDENTIFICATION

There are services exist in the field of identification and verification of access to the web resources. Those are created to offer a digital analogue of ID card.

VOTING

The principle of elimination of a control device between network participants allows implementing a fair voting model based on the blockchain where fraud is impossible.

DOCUMENTATION VERIFICATION

Verification of documents is an ingenious way of using blockchain as a publicly-certified timestamp. Companies, universities and public institutions can now secure their electronic content and secure the authorship by registering the content in the blockchain.

The above examples of applications by Rosic (2017) are only a small fraction of possible implementations.

3.8 **Multichain blockchain platform**

Multichain is a multipurpose platform for the creation and deployment of the private blockchains based on the Bitcoin's blockchain. The project aims to offer a flexible implementation of the permissioned blockchain for the corporate clients (Greenspan, 2015).

Multichain 1.0.4 has following specifications according to Greenspan (2015):

- Thousand transactions per minute performance speed;
- Access to all applications and libraries previously used to perform operations with crypto currency;
- Simplicity of operations because the usage of the distributed ledger.
- Multiple API functions for the development.
- Flexible node permission configuration
- Possibility to run multiple blockchains running on the same server
- Ability to set parameters (valid transaction types, confirmation time, minimum quantity, transaction speed and size limits).
- Consensus mechanism is distributed between identified validator units in a cyclical principle.

Multichain has an important function called stream. Stream is a tool which stores all information about the transaction in the ledger. It offers a functionality for blockchain where the general data retrieval, timestamping and archiving is more important rather than the transfer of assets between network participants. Generally, any number of streams can be utilised simultaneously. Each stream acts as an independent collection of items (Greenspan, 2015). Each item has the following properties:

- Single or multiple publishers
- Keys for easy data retrieval
- Data formats ranges from texts to the heavy files of raw binary
- Timestamping declares when the block was created

Multichain platform utilizes the consensus of Practical Byzantine Fault Tolerance, but with some changes and adaptations. Instead of having multiple nodes for each block validation process, there is only one main validator per block and it works in a round-robin algorithm . In the round--robin, permitted miners must rotate after creation of certain number of blocks in order to maintain a valid blockchain (Greenspan, 2015).

4 PROJECT IMPLEMENTATION

As from this chapter, the practical part of the thesis project is examined with a detailed description of the steps for creating the application and a demonstration of its functionality.

4.1 Development environment

The creation of a new software project requires a suitable development environment which will allow to code, configure and run the project as planned. Linux is an OS ideally suited for software project elaboration. It offers a convenient console interface, flexible customization for the needs of the project and it can act as an operating system for the server part. Ubuntu was chosen from other Linux distributives because of the various repositories and libraries available and lots of reference materials for the elimination of errors in the program.

4.2 Server set up and OS update

First, it was decided to use Digital Ocean’s public hosting servers. The purchase and setup of a private server would have been more correct, but it would have increased the costs and slowed down the time of development.

Second, the Multichain server requirements were clarified from the official website. Compliance with the server requirements stated by the software developer guarantees the correct operation of the system. The requirements were Linux: 64-bit OS that supports Ubuntu version 12.04 and later, minimum 512 MB of RAM and 1GB of disk space.

The server setup takes a few steps:

- Create a droplet
- Choose OS of the server
- Choose a datacentre country
- Create name

Other processes were made automatically by the hosting provider. There were three servers created for the application development. An **Interface** server was responsible for the user interface, the **MultichainServer** handled the publishing and verification events from the user interface and the **MiningNode** participated in the confirmation of transactions. The list of servers is shown in the Figure 5.

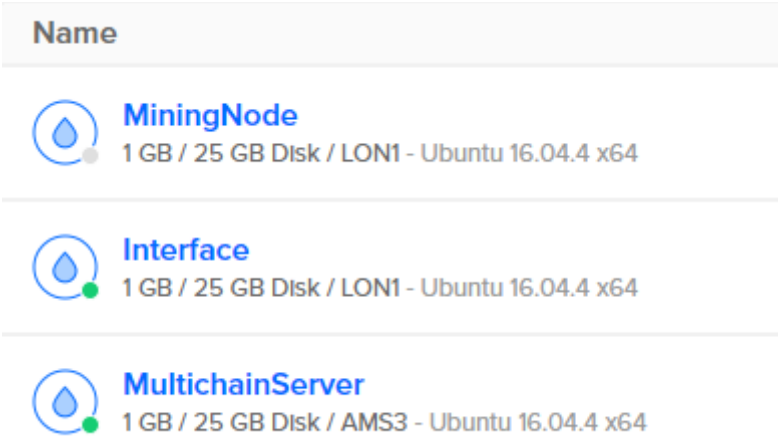


Figure 5. Servers created

Before the installation of libraries and programming of the application, each of the servers was updated to the latest version of OS. The update command for downloading the newest libraries is shown as follows:

apt-get update

After the update was completed, the upgrade command was executed to start the installation of the updates. Upgrade command:

apt-get upgrade

Once the creation of the servers and the operating system update were performed, the installation of the necessary libraries and repositories were carried out.

4.3 Multichain server installation and configuration

In this chapter, the full installation of the Multichain platform is described including the installation of the firewall permissions, carrying out the RPC protocol configurations, and connecting and creating an additional node on another server.

4.3.1 Multichain repository download, configuration and initialisation.

To download an archive with the official version of the Multichain repository and to unpack it, the command from Figure 6 was executed.

```
root@Interface:~# wget https://www.multichain.com/download
/multichain-1.0.4.tar.gz
root@Interface:~# tar -xvzf multichain-1.0.4.tar.gz
```

Figure 6. Download command of Multichain repository

Foremost, a new blockchain was created and named as **chain1**. The next step was the initialization of the system including mining the genesis block, illustrated in Figure 7.

```
root@Interface:~# multichain-util create chain1
root@Interface:~# multichaind chain1 -daemon
```

Figure 7. Blockhain initialisation

The output of the commands should state that the genesis block was mined and the node is running as shown in Figure 8.

```
Starting up node...

Looking for genesis block...
Genesis block found

Node ready.
```

Figure 8. Genesis block sucessfully created

The next stage was the creation of a stream. The working principle of a stream was explained in the Chapter 4.8. The command to create the first blockchain stream is shown here where **stream1** is the name of the stream which was created and used further for requests from the client server, the **false** means that the stream can only be used by the servers with

permission to publish and retrieve data in order to prevent unauthorised and spam information in the blockchain.

create stream stream1 false

Subsequent changes were performed in the interaction mode. This is the specialized mode of the system to execute commands directly to **chain1** and **stream1**. It was activated by typing the following command:

multichain-cli chain1

In order to be able to receive further information from the stream, a subscription was required. By typing the following commands, the stream subscription and test publish transaction were executed:

subscribe stream1
publish stream1 key1 736f6d65206f746865722064617461

The list of success transactions can be displayed via the following command:

liststreamitems stream1

If the test transaction is visible in the system, development could proceed.

To check the extended information on the number of mined blocks, node address and ports for the connection of additional miners to the blockchain, the following command was used:

getinfo

At this stage, the launch of the blockchain was completed and the transition was made to the RPC protocol and Firewall configuration.

4.3.2 RPC connection and Firewall settings

Remote Procedures Call mechanism is a key technology used in building distributed applications. It allows the application to execute procedures on another machine in the network. In this project, RPC is used to transfer data and commands from program running on user interface side thought API server and transfer it to the blockchain server.

To allow RPC connection, IP address of the user interface server must be added to the multichain.conf file which can be found via the path ~/.multichain/chain1. IP address could be found from the in the personal account on the hosting provider's website. Before committing any changes to the configuration files, blockchain should be stopped with the command bellow:

multichaind chain1 stop

The third line of the Figure 9 is allowing connection trough RPC protocol only to the defined IP address. There are two other lines of configuration in the file which are the login and the password for the RPC connection. Those were used in Chapter 5.

```
rpcuser=multichainrpc
rpcpassword=Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp
rpcallowip=167.99.201.254
```

Figure 9. RPC access list configuration

When all the changes were committed to the settings, the blockchain was started by the following command:

nohup multichaind chain1 -daemon &

Furthermore, the server firewall rule needs to be configurated. This was done by creating a rule and adding a RPC port number that receives requests from the client server.

First, the default RPC port can be found via command bellow:

cat params.dat

The right line is shown in the Figure 10:

```
default-rpc-port = 2754
```

Figure 10. Default RPC port

This port number was entered to the server's firewall rules panel. See the example of the rule in the Figure 11.

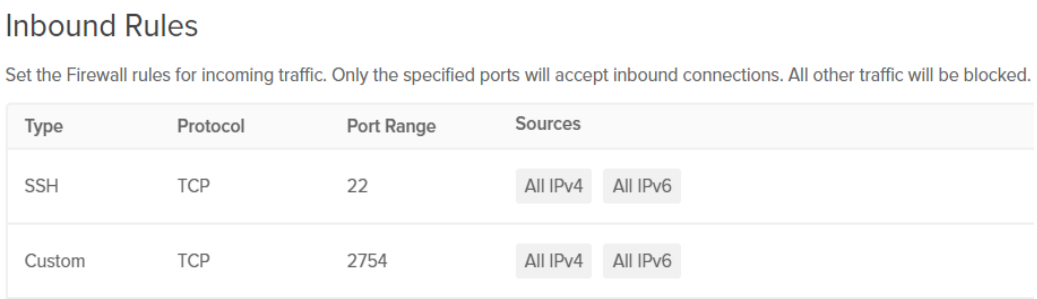


Figure 11. Firewall rule

4.3.3 Subsidiary mining node connection

The last step of the Multichain server setup part was the connection of the second full node to the mining process. Main node connection address can be check in the blockchain interactive mode with the **getinfo** command.

Afterwards, actions for download and installation of the Multichain library were made on the third server called MiningNode as it was described in Chapter 4.3.1. When the installation was completed, the connection request was sent to the main server and the mining permission grated.

First, the following line was executed:

multichaind chain1@167.99.212.17:2755

The output of the command should display the node's wallet address as it is shown in Figure 12.

```
Retrieving blockchain parameters from the seed node 167.99.212.17:2755 ...
Blockchain successfully initialized.

Please ask blockchain admin or user having activate permission to let you connect and/or transact:
multichain-cli chain1 grant 1DWzGhBxNNnXEAtskfUxfJzvUCCWxcTNxfNbwT connect
multichain-cli chain1 grant 1DWzGhBxNNnXEAtskfUxfJzvUCCWxcTNxfNbwT connect,send,receive
```

Figure 12. Subsidiary node connection

Back to the MultichainServer, connection and mining permission were granted to start collaborative mining. See the commands bellow to enter interaction mode and give permissions:

multichain-cli chain1

grant 1DWzGhBxNNnXEAtskfUxfJzvUCCWxcTNxfNbwT connect
grant 1DWzGhBxNNnXEAtskfUxfJzvUCCWxcTNxfNbwT mine

Running the next command maximized the degree of miner randomness to ensure the integrity of the system:

setruntimeparam miningturnover 1

After a couple of minutes, a new node was added to the system and blocks were signed by one of two permissioned nodes.

Addition of extra node increases the efficiency of the system. If the node on the Multichain server stops responding to the host, a spare node is included in the work. When the operation of the main node is normalized, the work continues in the normal mode with round-robin switching.

4.4 Technology set up for interface server

This chapter describes the installation of the database, API and HTTP Server libraries for the client side of the application

4.4.1 Apache2 installation

Apache2 is the HTTP server client for the subsequent development and processing of the application interface. See the installation command in the Figure 13.

```
root@Interface:~# sudo apt-get install apache2
```

Figure 13. Apache2 installation

Before the Apache server the activation, the firewall rules were configurated to allow outside access to the web ports. See commands in the Figure 14.

```
root@Interface:/# sudo ufw allow 'Apache Full' 'Apache'
 'OpenSSH' 'Apache Secure'
```

Figure 14. Apache2 firewall adjustment

After the completion of previous steps, Apache server was up and running. System status can be checked with code from the Figure 15.

```
root@Interface:~# sudo systemctl status apache2
```

Figure 15. Apache2 system restart

The output should state that system is active. If the status is the same as in the Figure 16, this means that the system is properly configured and ready for use.

```
 Active: active (running)
```

Figure 16. Apache2 active status

4.4.2 PHP libraries and MySQL installation

Client side uses PHP programming language to perform mailing functions, API calls, database and user management. To ensure the operation of these functions the installation of certain libraries was required. The following commands from the Figure 17 has downloaded necessary PHP7 libraries.

```
root@Interface:~# apt-get -y
install php7.0 libapache2-mod
-php7.0
```

Figure 17. PHP7 installation command

Next, MySQL database and additional libraries were installed as shown in the Figure 18. It was used to create a database and configure user access levels.

```
root@Interface:~# apt-get -y install php7.0-mysql php7.0-
curl php7.0-gd php7.0-intl php-pear php-imagick php7.0-im
ap php7.0-mcrypt php-memcache  php7.0-pspell php7.0-recod
e php7.0-sqlite3 php7.0-tidy php7.0-xmlrpc php7.0-xsl php
7.0-mbstring php-gettext
```

Figure 18. MySQL database and libraries installation commands

To configure the MySQL, PHPMyAdmin package is installed as shown in the Figure 19. There was a possibility to configure MySQL database from terminal, but for the for greater visibility and ease of operation PHPMyAdmin interface was used.

```
root@Interface:~# sudo apt-get
install phpmyadmin php-mbstring
 php-gettext
```

Figure 19. PHPmyadmin installation

All the previously installed libraries were activated automatically. The only manual activation was needed for **mcrypt** and **mbstring** extensions. See code in the Figure 20.

```
root@Interface:~# sudo phpenmod mcrypt
root@Interface:~# sudo phpenmod mbstring
```

Figure 20. PHPMyAdmin and needed libraries installation command

Next, restart of the Apache server was compulsory to apply the changes with the code in the Figure 21.

```
root@Interface:~# sudo systemctl restart apache2
```

Figure 21. Apache2 restart command

Last step was the installation of the library called Composer. Composer downloads obligatory dependencies for the applications written in PHP. It was used in the Chapter 4.4.3 while creating RESTful API. It was installed with command shown in the Figure 22.

```
root@Interface:~# php composer-setup.php --install
-dir=/usr/local/bin --filename=composer
```

Figure 22. Composer installation

4.4.3 REST API using Slim framework

With an eye to carry out communication with Multichain server in the form of publishing, verifying and retrieving data, the RESTful API was implemented.

The RESTful API defines a set of functions to make requests and receive responses. The interaction is proceeded via the HTTP protocol. The advantage of this approach is the wide distribution of the HTTP protocol, so the REST API can be used in almost any programming language. There are many open source frameworks for creating RESTful API. After the analysis of the possible options, Slim framework was chosen because of being lightweight, fast and written in PHP.

This framework installs the required libraries and repositories for handling the API requests, but actual API calls were written by the thesis author. The command bellow creates a new folder with the API files:

composer create-project slim/slim-skeleton multichainapi

The API workflow is shown in the Figure 23. When the API receives a request for publication, it sends a POST request to the server using RPC protocol. The Multichain server processes it and adds the information to the blockchain. Afterwards, Multichain server sends back a response to the RESTful API with a response of successful publication.

In case when the user sends a verification request, the API sends GET call to the Multichain server and receives the response with the data stored in blockchain.

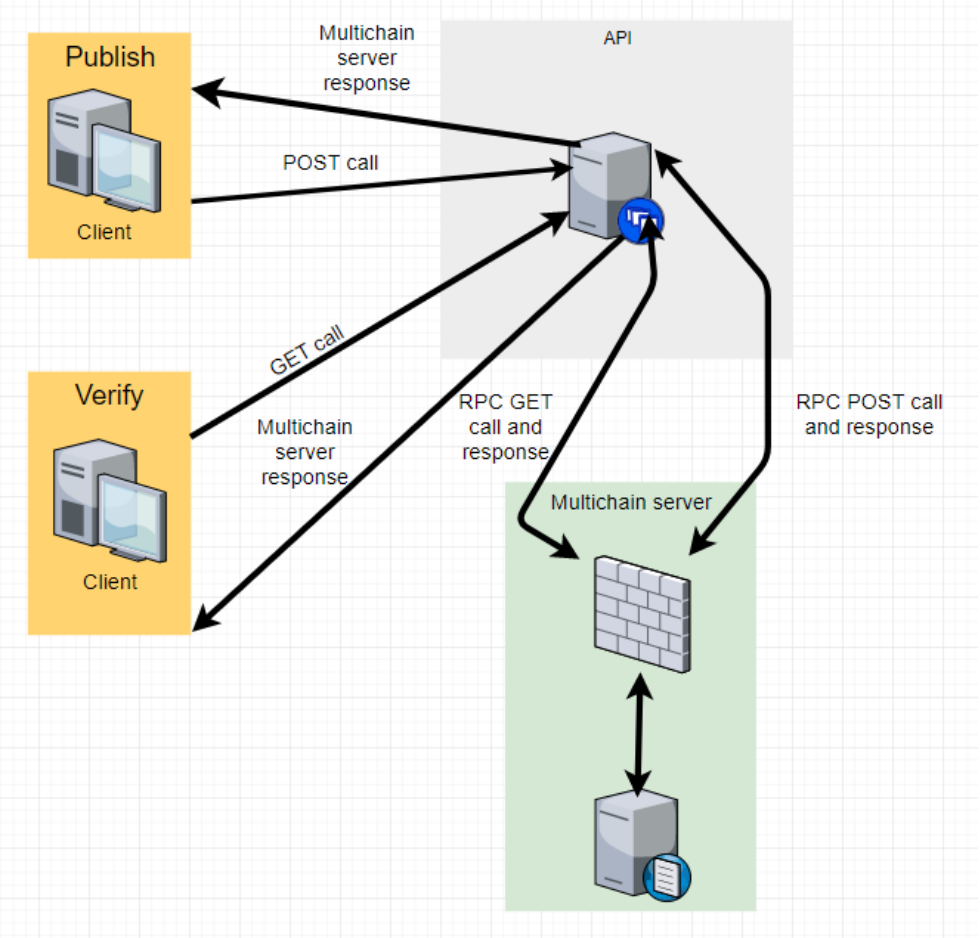


Figure 23. RESTful API workflow.

Full API code and the examples of requests are shown in the Appendices 1 and 8.

4.5 **Registration and login system**

As it was stated in Chapter 2, there were two types of users filtered by functionality accessibility levels. Students and employers were only allowed to verify documents and use search database, while registered users get the additional approach to the publish function. The installation of this system is shown in this chapter.

4.5.1 Database set up and configuration

MySQL database is used for connection of new universities to the system and further login.

To create the database and establish the connection to the user interface, following steps were completed:

- Access of the admin interface on the www.167.99.201.254/phpadmin address. Login into the system was required to commit changes.
- Creation of new database by running the script from the Figure 24.

```
CREATE DATABASE accounts;

CREATE TABLE `accounts`.`users`
(
    `id` INT NOT NULL AUTO_INCREMENT,
    `first_name` VARCHAR(50) NOT NULL,
     `last_name` VARCHAR(50) NOT NULL,
    `email` VARCHAR(100) NOT NULL,
    `university` VARCHAR(100) NOT NULL,
    `phonenumber` VARCHAR(100) NOT NULL,
    `hash` VARCHAR(32) NOT NULL,
    `active` BOOLEAN NOT NULL DEFAULT 0,
PRIMARY KEY (`id`)
);
```

Figure 24. MySQL database creation script

- The PHP script coded in registration folder for connection to the database as it shown in the Figure 25.

```
<?php
$host = 'localhost';
$user = 'root';
$pass = 'password';
$db = 'accounts';
$mysqli = new mysqli($host,$user,$pass,$db)
or die($mysqli->error);
?>
```

Figure 25. Database connection to the interface

When these steps were completed, the system was ready to register new users and verify information on login.

4.5.2 Registration and mailing

The registration process includes sending information about the university and obtaining data for the login.

The code in the Figure 26 displays how the information is collected and redirected to the mailing service called PHPmailer.

```php
$nemail = $_REQUEST['nemail'] ;
$nemail = $_POST['nemail'] ;
$fristname = $_REQUEST['firstname'] ;
$lastname = $_REQUEST['lastname'] ;
$firstname = $_POST['firstname'] ;
$lastname = $_POST['lastname'] ;
$organisationname = $_REQUEST['organisationname'] ;
$businessid = $_REQUEST['businessid'] ;
$organisationname = $_POST['organisationname'] ;
$businessid = $_POST['businessid'] ;
$phone = $_REQUEST['phone'] ;
$phone = $_POST['phone'] ;
```

Figure 26. Information parsing and redirecting.

PHPMailer is an e-mail sending library. When the registration request is submitted, it collects the information and sends it to the application administrator's e-mail.

```php
$mail = new PHPMailer();
$mail->CharSet =  "utf-8";
$mail->IsSMTP();
$mail->SMTPAuth = true;
$mail->Username = "mailingbox@gmail.com";
$mail->Password = "password";
$mail->SMTPSecure = "tls";
$mail->Host = "smtp.gmail.com";
$mail->Port = "587";
$mail->setFrom('mailingboxn@gmail.com', 'University Blockchain');
$mail->AddAddress('receiversinbox@gmail.com', 'receivers name');
```

Figure 27. PHPmailer mailing script.

The code presented in the Figure 27 establishes the connection to the mailing account (mailingboxn@gmail.com) using TLS connection and sends the email to the application administrator's e-mail (receiversinbox@gmail.com). When information is received, it is verified by the administrator. If it is relevant and correct, a new user account is created and login information sent to the university representative.

4.5.3 Login

Before proceeding with the activities related to the publication of the diploma, user must log in to the system. The code in the Figure 28 creates a session for each user who is trying to visit the page to load the diploma. The system checks whether the user has logged in and looks for matches in the database. If the system has not confirmed the login, there is an

automatic redirect to the page with the necessity to login or register. In another case, the redirect goes to the client profile page and from there user can procced to the page with the publish form.

```
session_start();
if ( $_SESSION['logged_in'] != 1 ) {
  $_SESSION['message'] = "You must log in before publishing!";
  header("location:http://167.99.201.254/registration/error.php");
}
else {
    $first_name = $_SESSION['first_name'];
    $last_name = $_SESSION['last_name'];
    $email = $_SESSION['email'];
    $active = $_SESSION['active'];
}
```

Figure 28. User login check.

The appearance of the login, error, user profile and registration pages, full PHPMailer code and example of registration email could be found in Appendices 2-7.

5 DIPLOMA PUBLISH AND VERIFICATION DEMOSTRATION

This chapter of the thesis demonstrates the work of publishing and verification services. Additionally, the communication between the client and the server part is demonstrated.

5.1 Regular user without publishing permission

A regular user such as an employer or a student can produce two types of actions on the website: automatic verification and database search using the search line.

Automatic verification occurs by uploading the original file that was originally added to the blockchain to the webpage shown in the Figure 29. It is important to understand that the file should not have any, even the slightest changes, as it is regarded by the system as a different document because the hash function of the originally uploaded document would not match the hash function of the document uploaded for verification.

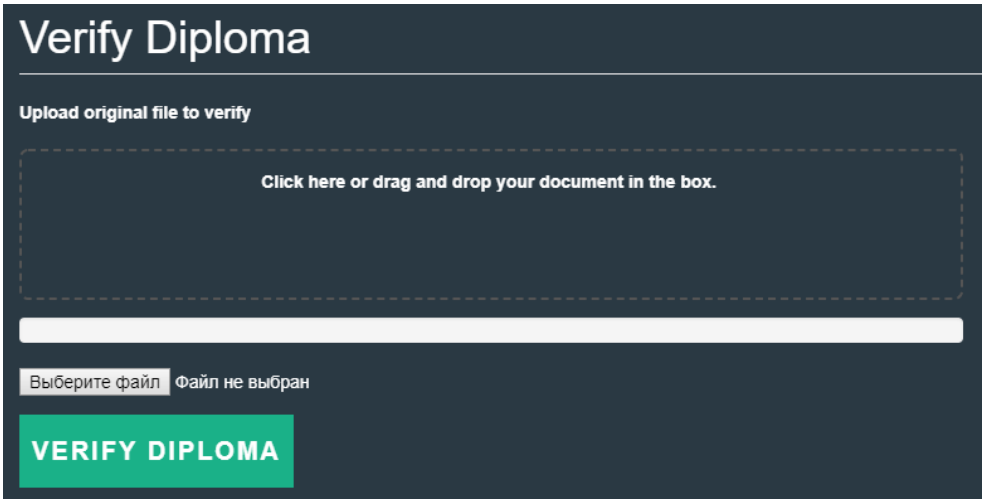


Figure 29. Verification page appearance

When a file is uploaded to the website, the file is hashed using CryptoJS library (version 3.1.2) which proceeds SHA-256 cryptographical algorithm. Further, a request is sent to the Multichain server through the API with the indication of the received hash and further search in the blockchain.

The request is sent to the server via the RPC protocol. The third line of the code in the Figure 30 includes IP of the RPC and the port the server is listening for the requests, username and password for login. After the authorization of the request has been executed, the system looks for the matches in the blockchain stream for the sent hash.

```
$application->get('/verify/{signature}', function (Request $request, Response $response) {
$signature = $request->getAttribute('signature');
$client = new MultichainClient("http://167.99.212.17:2754",
'multichainrpc', 'Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp', 3);
$data = $client->setDebug(true)->executeApi('liststreamkeyitems', array("poe", $signature));
$data = array_reverse($data);
$returndata = array();
```

Figure 30. Verification request to the API

If a match is found, the user receives a table shown in the Figure 31 with detailed information about the diploma. It includes the name, surname and identification number of the student, the number of the diploma and information on the confirmation of the document by the blockchain.

Data	Value
signature	4e71e2fb1a34b036dda206e50832cdbe40e45cc0480f41d7c9c6a4339eace563
transaction_id	608aeb59676c5ce2ef40ab74b36cd3d31ca5f31b6db8e1da6fe51c4d60127de6
confirmations	1
blocktime	1:35 PM on Wednesday 2nd May 2018 (Timezone EEST)
name	Kaltyshev Mikhail
studentid	STUDENT368743S
message	HK8983678HNNK
recorded_timestamp_UTC	1525268133
readable_time_UTC	1:35 PM on Wednesday 2nd May 2018 (Timezone EEST)

Figure 31. Verified diploma information

If by some coincidence the original document has been lost, but the transaction number has been saved, user can make a search through the database using it. The application sends a request to the server to retrieve all published diplomas of the system by means of the code in the Figure 32.

```
$application->get('/latest/published/{count}', function (Request $request, Response $response)
```

Figure 32. GET request from published diplomas

The database page contains a table with the list of uploaded diplomas. See the Figure 33.

search

Document Digest	Timestamp	Document ID	Name of Diploma owner
e88e49ae1f56068b10dc16ace28ae082a2b109327c13c6c49ea2a2794b2e95d1	2018-05-05 17:57:58	GHJG580985BNMIUIU	Student1 Name1
41bd47030c8ff5bb9dd7391f8104b7d01fe1d414fb7ebcb64fdb9b057eafd9ba	2018-05-02 14:19:48	GDYTUXS6786876S	Kaltyshev Mikhail
4e71e2fb1a34b036dda206e50832cdbe40e45cc0480f41d7c9c6a4339eace563	2018-05-02 13:35:33	HK8983678HNNK	Kaltyshev Mikhail

Figure 33. Database search page appearance

5.2 **Registered user with publishing permission**

The second level of users has access to the same list of the functions described in Chapter 5.1. Additionally, the access to the function of uploading a diploma to the system is granted. The publish form is shown in the Figure 34.

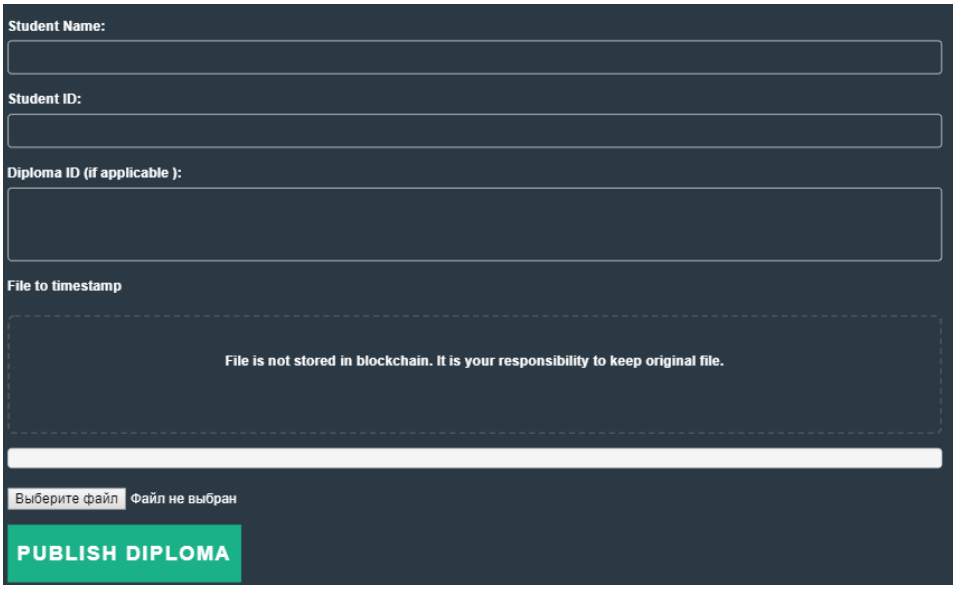


Figure 34. Publish form page appearance

On the server side, the process is programmed as it shown in the Figure 35. After filling out the form and uploading the diploma, the system collects data and sends it via the API into the blockchain for publishing. The RPC authorization process described earlier is performed and the data is published in the blockchain.

```
$application->post('/publish/{signature}', function (Request $request, Response $response) {

    $signature = $request->getAttribute('signature');
    $client = new MultichainClient("http://167.99.212.17:2754",
    'multichainrpc', 'Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp', 3);
    $data = $request->getParsedBody();
```

Figure 35. POST request to publish diploma

If the information is successfully published, the user receives confirmation in form of a table as in the Figure 36.

Data	Value
url	http://167.99.201.254/poe/details.php?signature=41bd47030c8ff5bb9dd7391f8104b7d01fe1d414fb7ebcb64fdb9b057eafd9ba
signature	41bd47030c8ff5bb9dd7391f8104b7d01fe1d414fb7ebcb64fdb9b057eafd9ba
transaction_id	3af9e6c1f80ff67eea23fccf694bf4f60efb4e0794a8a1c549d4eeefa0ad3675
confirmations	0
blockhash	NA
blocktime	NA
name	Kaltyshev Mikhail
email	STUDENT36874CJ
message	GDYTUXS6786876S
timestamp	2:19 PM on Wednesday 2nd May 2018 (Timezone EEST)

Figure 36. Successful upload confirmation

6 FUTURE IMPROVEMENTS

This version of the software is uncompleted and needs further development in the future. Possible improvements are listed below:

1. Storage of files in the blockchain

At the moment, files are not loaded to the blockchain and only the hash of a document is stored. It is possible to implement this using the Multichain platform, but it will slow down the work of network nodes which are confirming the transactions and creating new blocks. To perform this task, it is necessary to select the optimal document size and document storage algorithms in blockchain.

2. Improvement of safety.

Basic security matters in this application version do not guarantee the absence of unauthorized publications in the blockchain by hackers getting access into a university account. Adding a two-factor authorisation (2FA) mechanism and electronic identification can significantly improve this parameter.

3. Use of public blockchains.

In the future, the use of public blockchain projects such as Ethereum can be considered. In contrast to private blockchains, in theory, the degree certificate can be stored forever in the public network. However, this significantly increases the cost of the system since in public networks each transaction costs a certain amount.

7 CONCLUSION

The main objective of this thesis project was to get the reader familiar with the blockchain technology in the theory part and to show the reader the step-by-step process of developing and running an application that proves the existence of university graduation certificates in empirical part.

The outcome of this thesis project was an application based on the Multichain platform. This version of the program includes a fully configured private blockchain server with second node mining on a different server. Additionally, the RESTful API was created for communication between the user interface and the blockchain server for request processing and information exchange. Moreover, a user-friendly interface was developed. The concept of the application that was presented at the beginning of the thesis was fully implemented.

It should be noted that the program in this thesis covered one possible application. An example of other applications can be the use of it to

confirm the possession of intellectual property. Files (music, drawings, texts) can be uploaded and hashed by the system. Additionally, the author's name, the creation date and a short description of the file content. Moreover, this program can be used to protect luxury goods such as clothes, expensive perfumes, jewellery and watches. Each of these items has a manufacturing certificate which can be written and stored in the blockchain to prevent forgery and copying.

The biggest difficulty at the time of writing the thesis was the lack of extensive experience in coding and configuring applications working with the use of the REST API and RPC protocol. A lot of time was spent solving issues related to the application's access settings to the remote server's ports. It was important to make sure that the application could send and receive requests from the blockchain server.

Developing this application was a complex and interesting challenge. The author of this thesis had some previous experience in creating web applications, but had only theoretical knowledge about the function of blockchain systems. This project greatly improved the level of practice of the author in web development and gave him invaluable experience in working with the blockchain technology.

REFERENCES

BlockchainHub. (n.d.). *Blockchain Glossary for Begginers.* Retrieved from https://blockchainhub.net/blockchain-glossary/

Bear, J., & Ezell, A. (2012). Degree Mills: The Billion-Dollar Industry That Has Sold Over a Million Fake Diplomas. New York: Prometheus Books.

Satoshi, N. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash system.* Retrieved from https://bitcoin.org/bitcoin.pdf

Swan, M. (2015). *Blockchain: Blueprint for a New Economy* (1 edition). Sebastopol: O'Reilly Media.

Imran, B. (2017). Mastering Blockchain: Deeper insights into decentralization, cryptography, Bitcoin, and popular Blockchain frameworks. Packt Publishing.

Dzone. (2017). *An Introduction to Blockchain.* Retrieved May 16, 2018, from https://dzone.com/articles/how-blockchain-technology-works-and-gives-maximum

Invest in Blockchain. (2017). *How Does Cryptography Protect the Blockchain?* Retrieved from https://www.investinblockchain.com/how-does-cryptography-protect-blockchain/

Shaan, R. (15 December 2017). *Merlke Trees*. Retrieved from https://hackernoon.com/merkle-trees-181cb4bc30b4

Lee, D. K., & Deng, R. H. (August 18, 2017). *Handbook of Blockchain, Digital Finance, and Inclusion, Volume 1: Cryptocurrency, FinTech, InsurTech, and Regulation*. London: Academic Press; 1st edition.

Lamport, L., Pease, M., & Shortak, R. (n.d.). *The Byzantine Generals Problem* [PDF]. SRI International. Retrieved from https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

Dwork, C.,& Naor, M. (n.d.). Pricing via processing or combatting junk mail [PDF].

QuantumMechanic. (11 July 2011). Proof of stake instead of proof of work [Online forum comment]. Retrieved from https://bitcointalk.org/index.php?topic=27787.0.

Castro, M., & B., Liskov. (n.d.). *Practical Byzantine Fault Tolerance* [PDF]. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge. Retrieved from http://pmg.csail.mit.edu/papers/osdi99.pdf

Byterin, V. (2015, August 7). On Public and Private Blockchains [Web log post]. Retrieved May 8, 2018, from https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Williams, S. (2017, December 11). 5 Big Advantages of Blockchain, and 1 Reason to Be Very Worried [Web log post]. Retrieved May 8, 2018, from https://www.fool.com/investing/2017/12/11/5-big-advantages-of-blockchain-and-1-reason-to-be.aspx

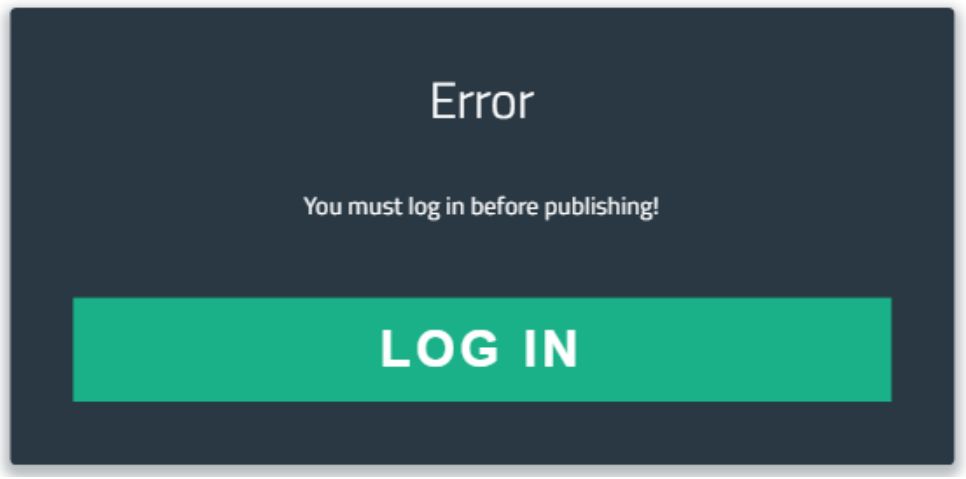
Rosic, A. (2017). 17 Blockchain Applications That Are Transforming Society [Web log post]. Retrieved May 8, 2018, from https://blockgeeks.com/guides/blockchain-applications/

Greenspan, G. (2015). *MultiChain Private Blockchain — White Paper* [PDF]. Retrieved from https://www.multichain.com/download/MultiChain-White-Paper.pdf

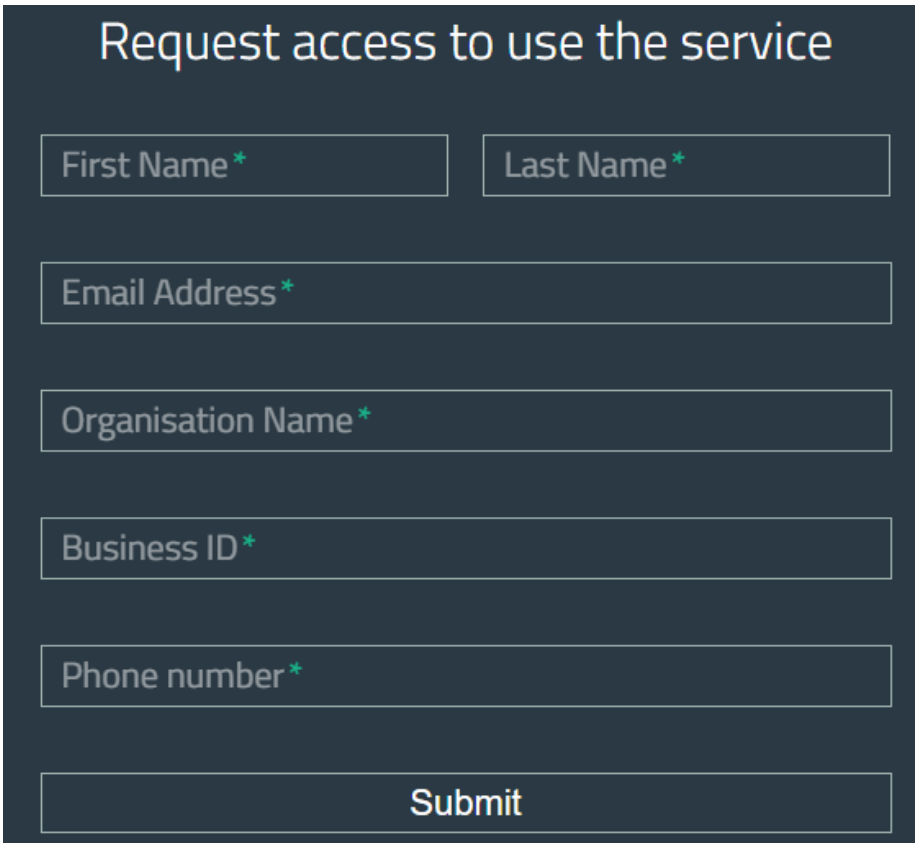
Examples of API requests and responses Appendix 1

Call code	Response
POST 167.99.201.254/ui/api/v1/publish/{hash function signature}	{"url": "http://167.99.201.254/poe/details.php?signature=hash function signature", "signature":"hashfunctionsignature", "transaction_id":"assigned transaction number", "confirmations": 0, "blockhash": "NA", "blocktime": "NA", "studentname": "Full Name", "studentid": "Student Number", "diplomaid": "Diploma ID", "timestamp": "Date and time"}
GET 167.99.201.254/ui/api/v1/verify/{hash function signature}	{"url": "http://167.99.201.254/poe/details.php?signature=hash function signature", "signature":"hashfunctionsignature", "transaction_id":"assigned transaction number", "confirmations": number of node confirmations, "blockhash": "block hash", "blocktime": "time when block was created", "studentname": "Full Name", "studentid": "Student Number", "diplomaid": "Diploma ID", "timestamp": "Date and time"}

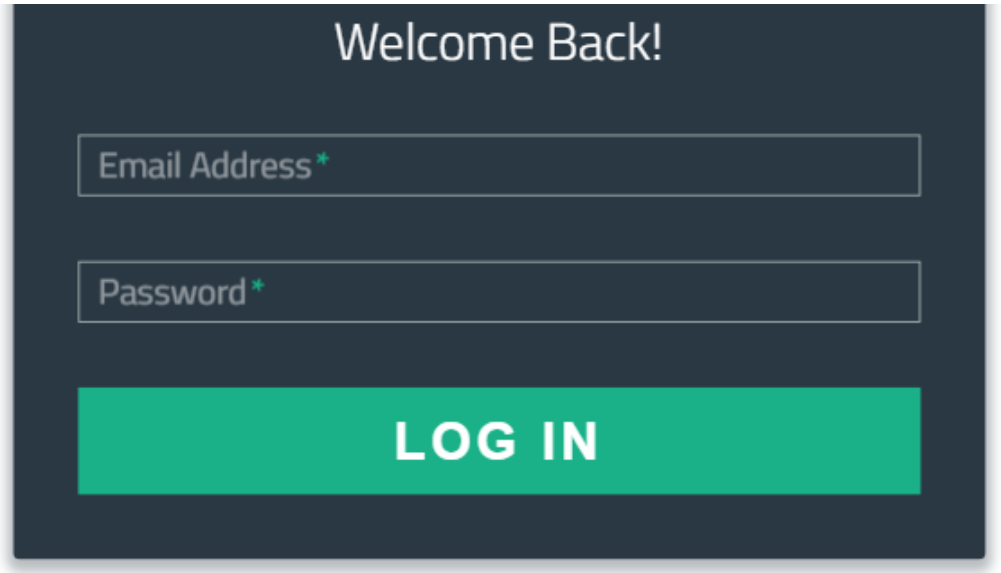
Appearance of the error page Appendix 2



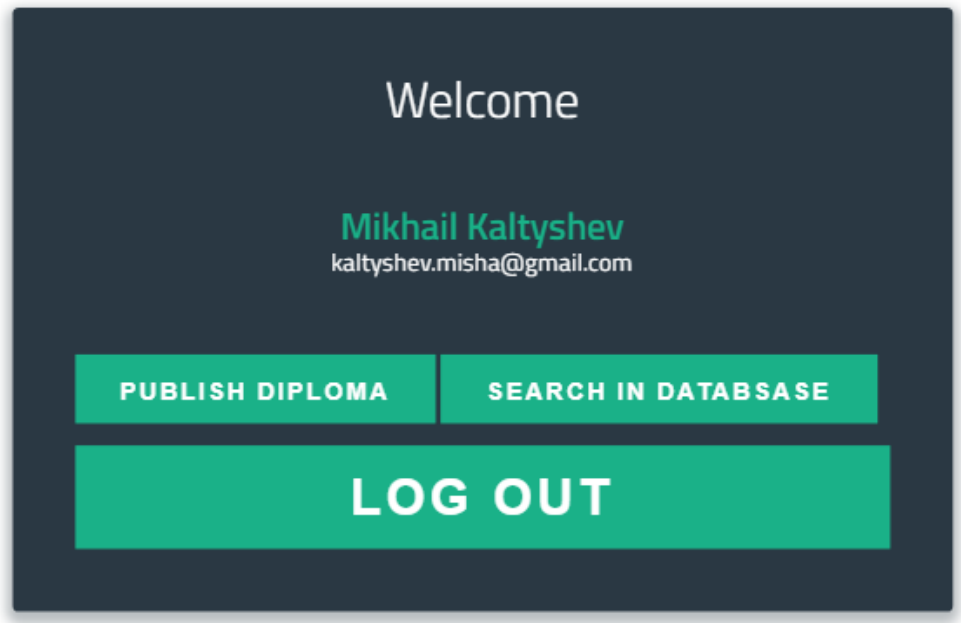
Registration from appearance Appendix 3



Login page appearance Appendix 4



User profile appearance Appendix 5



Registration request Appendix 6

New registration request!

University Blockchain

First name: Mikhail
Last name Kaltyshev
Email: registerme@system
Organisation name HAMK
Business ID 13371488
Phone +888999998897897

PHPMailer code Appendix 7

```php
<?php
ini_set('display_errors', 1);
 require_once('PHPMailer/PHPMailerAutoload.php');
 $nemail = $_REQUEST['nemail'] ;
 $nemail = $_POST['nemail'] ;
 $fristname = $_REQUEST['firstname'] ;
 $lastname = $_REQUEST['lastname'] ;
 $firstname = $_POST['firstname'] ;
 $lastname = $_POST['lastname'] ;
 $organisationname = $_REQUEST['organisationname'] ;
 $businessid = $_REQUEST['businessid'] ;
 $organisationname = $_POST['organisationname'] ;
 $businessid = $_POST['businessid'] ;
 $phone = $_REQUEST['phone'] ;
 $phone = $_POST['phone'] ;
    $mail = new PHPMailer();
    $mail->CharSet =  "utf-8";
    $mail->IsSMTP();
    $mail->SMTPAuth = true;
    $mail->Username = "mailingbox@gmail.com";
    $mail->Password = "password";
    $mail->SMTPSecure = "tls";
    $mail->Host = "smtp.gmail.com";
    $mail->Port = "587";
    $mail->setFrom('mailingboxn@gmail.com', 'University Blockchain');
    $mail->AddAddress('receiversinbox@gmail.com', 'receivers name');
    $mail->Subject = "New registration request!";
    $mail->Body = "
<br /> First name: $firstname <br>
Last name $lastname <br>
Email: $nemail<br>
Organisation name $organisationname<br>
Business ID $businessid<br>
 Phone $phone <br>";
    $mail->IsHTML(true);
     if($mail->Send()) {
   echo "<script>alert('Thank you for contacting us! We will soon get in touch with you')</script>";
   }
?>
```

REST API code Appendix 7

```php
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;
use be\kunstmaan\multichain\MultichainClient;
use be\kunstmaan\multichain\MultichainHelper;

if (PHP_SAPI == 'cli-server') {

    $url  = parse_url($_SERVER['REQUEST_URI']);
    $file = __DIR__ . $url['path'];
    if (is_file($file)) {
        return false;
    }
}
require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/../src/dependencies.php';
require __DIR__ . '/../src/middleware.php';
require __DIR__ . '/../src/functions.php';
require __DIR__ . '/../src/routes.php';
$config = require __DIR__ . '/../src/settings.php';
$application = new \Slim\App($config);
$application->post('/publish/{signature}', function (Request $request, Response $response) {

    $signature = $request->getAttribute('signature');
    $client = new MultichainClient("http://167.99.212.17:2754",
    'multichainrpc', 'Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp', 3);
    $data = $request->getParsedBody();
    if(isset($data['name']))
        $name = filter_var($data['name'], FILTER_SANITIZE_STRING);
    else
        $name = "Unknown";

    if(isset($data['studentnumb']))
        $studentnumb = filter_var($data['studentnumb'], FILTER_SANITIZE_STRING);
    else
        $studentnumb = "Unknown";

    if(isset($data['message']))
        $diplomaid = filter_var($data['message'], FILTER_SANITIZE_STRING);
    else
        $diplomaid = "NA";
```

Part one

```
    $dataArray = array("signature" => $signature,"name" => $name,
    "studentnumb"=> $studentnumb, "message"=>$diplomaid);
    $dataJSON = json_encode($dataArray);
    $dataBase64 = base64_encode($dataJSON );
    $dataHex = bin2hex($dataBase64);
    $returndata = array();
    $tx_id = $client->setDebug(true)->executeApi('publish', array("stream1", $signature, $dataHex));
    $longUrl = $_SERVER['HTTP_HOST']."/ui/details.php?signature=".$signature;
    $shorUrl = shortUrl($longUrl);

    $block_info = $client->setDebug(true)->executeApi('getwallettransaction', array($tx_id));
    $confirmations = $block_info['confirmations'];
    if($confirmations == 0){
        $blockhash = "NA";
        $blocktime = "NA";
    }
    else{
        $blockhash = $block_info['blockhash'];
        $blocktime = $block_info['blocktime'];
    }

    $returndata['long_url'] = "http://".$longUrl;
    $returndata['signature'] = $signature;
    $returndata['transaction_id'] = $tx_id;
    $returndata['confirmations'] = $confirmations;
    $returndata['blockhash'] = $blockhash;
    $returndata['blocktime'] = $blocktime;
    $returndata['name'] = $name;
    $returndata['studentnumb'] = $studentnumb
    $returndata['message'] = $diplomaid;
    $returndata['timestamp'] = date('g:i A \o\n l jS F Y \(\T\i\m\e\z\o\n\e \E\E\S\T\)', time());
    return $response->withJson($returndata)->withHeader('Content-Type', 'application/json');
);

application->get('/verify/{signature}', function (Request $request, Response $response) {
signature = $request->getAttribute('signature');
client = new MultichainClient("http://167.99.212.17:2754",
multichainrpc', 'Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp', 3);
data = $client->setDebug(true)->executeApi('liststreamkeyitems', array("stream1", $signature));
data = array_reverse($data);
returndata = array();
    foreach($data as $key => $value){
```

Part two

```
        $d = array();
        $d['signature'] = $signature;
        $d['transaction_id'] = $value['txid'];
        $d['confirmations'] = $value['confirmations'];
        $d['blocktime'] = date('g:i A \o\n l jS F Y \(\T\i\m\e\z\o\n\e \E\E\S\T\)', $value['blocktime']);
        $meta_data = json_decode(base64_decode(hex2bin($value['data'])));
        $d['name'] = $meta_data->name;
        $d['studentnumb'] = $meta_data->studentnumb;
        $d['message'] = $meta_data->diplomaid;
        $d['recorded_timestamp_EEST'] = $value['blocktime'];
        $d['readable_time_EEST'] = date('g:i A \o\n l jS F Y \(\T\i\m\e\z\o\n\e \E\E\S\T\)', $value['blocktime']);
        $returndata[$key] = $d;
    }

    return $response->withJson($returndata)->withHeader('Content-Type', 'application/json');

});

$application->get('/latest/published/{count}', function (Request $request, Response $response) {
    $count = $request->getAttribute('count');
    $client = new MultichainClient("http://167.99.212.17:2754",
    'multichainrpc', 'Fd2DuQcEdGLLaLdizpAVQpCjq3i4Yva4Y9x7oTLC5xnp', 3);
    $data = $client->setDebug(true)->executeApi('liststreamitems', array("stream1"));
    $data = array_reverse($data);
    $returndata = array();
     foreach($data as $key => $value){
        $d['signature'] = $value['key'];
        $d['confirmations'] = $value['confirmations'];
        $d['blocktime'] = date('Y-m-d H:i:s', $value['blocktime']);
        $meta_data = json_decode(base64_decode(hex2bin($value['data'])));
        $d['name'] = $meta_data->name;
        $d['message'] = $meta_data->diplomaid;
        $returndata[$key] = $d;
    }

    return $response->withJson($returndata)->withHeader('Content-Type', 'application/json');

});

$application->run();
```

Part three