Secure Development

Benjamin Whelan

Fontys ICT

Cyber Security

Contents

# Version

|  |  |  |
| --- | --- | --- |
| Version (v) | Date | Changes |
| 1 | 02/09/2022 | Introduction, Important Info and Blockchain CLI added |
| 2 | 12/09/2022 | Blockchain CLI completed |
| 3 | 21/09/2022 | Blockchain Web Client Documentation |

# Introduction

This document serves as an overview of my specialization project. For key points related to my specialization project, please see the portfolio. The goal of the project is to develop a fully functional and secure blockchain. There are several subprojects to build up to the final release version of the program:

* Blockchain Command-Line Interface
* Blockchain Basic Web Version
* Blockchain Advanced Vue.js Version
* WebSocket’s for final implementation

# Important Info (readme)

All source code for the program is on my public GitHub repository: <https://github.com/dainank/fontys-blockchain>

The file structure for the code is explained on the parent README there. The wiki will have more detailed information if needed.

# Blockchain Command-Line Interface (CLI)

The objective of this subproject is to get to grips with basic blockchain functionality in preparation for a full stack blockchain program in later sprints. There are several different iterations of this CLI thus I will walk through each one individually. For a more developer orientated quick start and explanation, see the readme in the following repository:

<https://github.com/dainank/fontys-blockchain/blob/main/blockchain-cli/README.md>

## Blockchain

Graphical user interface

Description automatically generated with medium confidenceThis is the most basic and barebones implementation of a blockchain. We are utilizing the ‘crypto’ API and on launch create a new blockchain (which is never saved anywhere and destroyed on program exit). The new blockchain will initially contain only a ‘Genesis Block’, the first block in the chain. The process of the generation of this initial block has completed when *creating blockchain* has been logged:

It has no previous hash (since no blocks came before) and no significant data (only that it is the genesis block). It does however have its own hash and an index for reference as well as a timestamp: Text

Description automatically generated

These two dummy blocks are mined and added to the chain, as represented by the respective log messages. At this point the mining stops since I have hardcoded the program to only mine two blocks after the genesis, thus the entire chain is now logged as shown in the image below:

Text

Description automatically generated

This blockchain is extremely vulnerable to attacks for several reasons. Firstly, no ‘nonce’ is required (see [README](https://github.com/dainank/fontys-blockchain/blob/main/README.md)), since the blocks do not require their hashes to start with a particular amount of zeroes at the start. This means that it is extremely easy for a block to be mined and therefore a fraudulent chain can be created incredibly quickly with minimal resources. To prevent this, we can require zeroes to be visible to the beginning of a hash for the hash to be valid. This concept is also known as proof of work since it requires the miners to utilize significant computer power to crack problems and find that hash that still contains all the data but also starts with the required number of zeroes.

The source code for this basic program can be found here:

<https://github.com/dainank/fontys-blockchain/blob/main/blockchain-cli/src/blockchain.ts>

## Blockchain with Proof of Work

As already stated in the title, this iteration of the CLI is more like a professional blockchain in that the hashed values of the blocks require zeroes, thus need time and effort to be mined making it harder for criminals to defraud the chain. The function that calculates this is shown below:

Text

Description automatically generated

Text

Description automatically generated

To further expand on this project, I added user options:

* Mine block
  + Simply mining a block with a random data (normally this data would consist of transaction info).
* View blockchain
  + Printing the entire blockchain out in JSON format.
* Exit program
  + Exiting the program gracefully (no need for CTRL+C).

Below is a demonstration of the program in action.

Text

Description automatically generated

The user is prompted with the three options. Let us view the blockchain currently:



As you can see, a ‘genesis block’ has been created, the first block of the chain which has no previous hash, but all the other parameters are there. There are no other blocks in the chain since we have not mined any yet. Let us mine a block now and then view the chain again:

Text

Description automatically generated

A new block has been mined that contains the previous hash but also a valid unique hash for itself with the expected number of zeroes at the front, thus proving the work done and validating it for the blockchain. We can mine several blocks if desired, but that is as far as this demo goes:

Text

Description automatically generated with low confidence

If you would like to test this program out for yourself, please download the binaries from the repository here: <https://github.com/dainank/fontys-blockchain/releases/tag/cli>

# Blockchain Vanilla Web (UI) + Additional Standalone Node Client

The goal of this subproject is to now layer the previous basic blockchain code with an interactable user interface. This will lay down the foundations for the final project, which will have a fully functional secure web client to initiate mining and view the blockchain as well as manage the currencies related to it. The key functionalities within the scope of this project include:

* Interactable basic user interface
* Browser & Node support for all blockchain functionalities
* Add transactions
* View transactions
* Add new block with pending transactions

## Proof of Work Walkthrough

When starting the program, the first genesis block is created: Graphical user interface, text, application, chat or text message

Description automatically generated

Once this is complete, the status panel will indicate that it is ready to receive transactions: Graphical user interface

Description automatically generated with low confidence

We can add the following transaction to the blockchain:

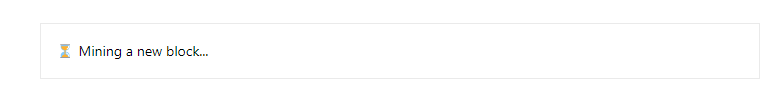
Graphical user interface

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with low confidence

At this point we can confirm the transactions and validate them now by finding a legal block to add to the blockchain, containing these transactions. At this point only the genesis block is present on the chain:

Graphical user interface, text, application, chat or text message

Description automatically generated

After a short pause, the next block is found and added to the blockchain, thus validating the transactions and saving it forever in the ledger. Graphical user interface, application, Word

Description automatically generated

We can also add multiple transactions in one block:

Graphical user interface, application

Description automatically generated

Let’s generate the block storing these transactions:

Graphical user interface, application, Word

Description automatically generated

In production, real blockchains often store many transactions within a single block for efficiency reasons.

## Look Into the Hash Generation

The first step is to convert the string of concatenated data into an array of unsigned 8-bit integers (a byte stream). We do this with the following method: 

The example in the docs shows very well what this exactly does: <https://developer.mozilla.org/en-US/docs/Web/API/TextEncoder>

Through the array from method, we can copy the contents of the byte stream into an actual array: 

We can now convert this hash into a string of hexadecimal values:

