

CS 302 Bonus Project - Hashing

...

Made by: Daniel Enriquez, Erik Marsh, Bruce Choe

What it does

- Generates a public and a private key
- Makes a blockchain of data value integer and stores it onto a map
- This is then printed out into a file when chosen
- Ensures permissions to either access or edit the data
- Anyone with the public key can view the data
- Anyone with a private key can add to the data
- In our given project, when choosing an option manually enter in your randomly provided key every time you want to do a certain operation
- This shows the concept of different accessing in and out of the chain

Why blockchain?

- It uses hashing in order to store the pointer to the next location in the chain
- This is because when one data value is changed within the chain, it changes all of the hashes
- This makes it so we can check if the blockchain has ever been changed without proper permissions
- It is mainly meant for secure access to data that should not be manipulated
- Hashing is also $O(1)$ time complexity by definition
- Accessing the hash map is also a $O(1)$ time complexity, this allows for easy accessing and printing of the map
- The hashmap stores a pair of data and a hash signature
- All of the code provided also works with any data type as it is templated

Terminal output (Saved as log.pdf)

For demonstration purposes, we'll generate random public and private keys for you.

Your keys:

Public Key: 191549

Private Key: 4960183

(1) Add a block to the Block Chain

(2) View the Block Chain

(3) Check if Block Chain has been manipulated

(4) Write the existing Block Chain to the file

(5) Exit

Enter in your option: 1

Enter in your private key: 4960183

Permission granted

Add an integer value to the Block Chain: 123

Adding value 123 to the Block Chain at location 1

Block mined: 00002c61d02a2fe9a2097aelf9ba42443adabc697e8c9353b365ba62bf93842d

Your keys:

Public Key: 191549

Private Key: 4960183

(1) Add a block to the Block Chain

(2) View the Block Chain

(3) Check if Block Chain has been manipulated

(4) Write the existing Block Chain to the file

(5) Exit

Enter in your option: 1

Enter in your private key: 4960183

Permission granted

Add an integer value to the Block Chain: 456

Adding value 456 to the Block Chain at location 2

Block mined: 00008ae24969982adfabb06be810cb9dda667cd4bd4cf93540ac591dbc954fc

Terminal output continued:

```
Your keys:
Public Key: 191549
Private Key: 4960183
(1) Add a block to the Block Chain
(2) View the Block Chain
(3) Check if Block Chain has been manipulated
(4) Write the existing Block Chain to the file
(5) Exit
Enter in your option: 2

Printing out map according to lowest to greatest hash signature
Hash: 00002c61d02a2fe9a2097aelf9ba42443adabc697e8c9353b365ba62bf93842d With Data: 123
Hash: 00008ae24969982adfabb06be810cb9dda667cd4bd4cf93540ac591dbc954fc With Data: 456

Your keys:
Public Key: 191549
Private Key: 4960183
(1) Add a block to the Block Chain
(2) View the Block Chain
(3) Check if Block Chain has been manipulated
(4) Write the existing Block Chain to the file
(5) Exit
Enter in your option: 3

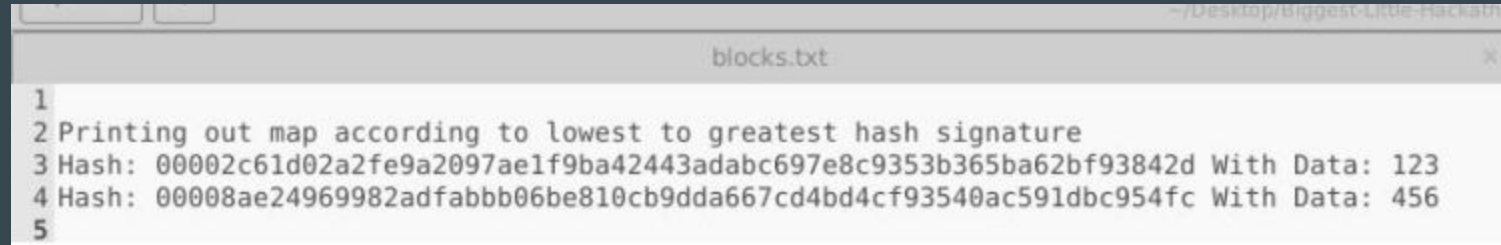
Block Chain is valid

Your keys:
Public Key: 191549
Private Key: 4960183
(1) Add a block to the Block Chain
(2) View the Block Chain
(3) Check if Block Chain has been manipulated
(4) Write the existing Block Chain to the file
(5) Exit
Enter in your option: 4

All of the blocks are now written into the file blocks.txt

Your keys:
Public Key: 191549
Private Key: 4960183
(1) Add a block to the Block Chain
(2) View the Block Chain
(3) Check if Block Chain has been manipulated
(4) Write the existing Block Chain to the file
(5) Exit
Enter in your option: 5
```

Text file



A screenshot of a text editor window. The title bar at the top shows the file path `~/Desktop/biggest-Little-Hacker`. The window title is `blocks.txt`. The text content is as follows:

```
1
2 Printing out map according to lowest to greatest hash signature
3 Hash: 00002c61d02a2fe9a2097aelf9ba42443adabc697e8c9353b365ba62bf93842d With Data: 123
4 Hash: 00008ae24969982adfabb06be810cb9dda667cd4bd4cf93540ac591dbc954fc With Data: 456
5
```