PROJECT STATUS REPORT

Use Block Chain Database for Medical Record Storage

Da Gong

dagong@usc.edu

I. Introduction

The goal of this project is to build a prototype of block chain database which can deployed on local or remote Linux-based server. The database would allow users to add any medical record, checkup certain medical record and check certain patient's all medical records. Each user must have their unique token key to access the database. This database could prevent data leakage, data misuse or data falsify. This project will prove that the use of block chain or NFT (Non-Fungible Token) techniques can be used as a new way to protect important data like medical records.

To illustrate the idea of using block chain to store medical records, the project will deploy a simulate database build using python. People with permission, in this project having private key will be able to access to the data. The database will also show how hard for people without permission to break the block chain database. This project will simulate certain brute force way to decipher this database to simulate potential attack to this database and show how long time the attacker needs to crack down the security provided by block chain technic.

Keywords: *Database Management, Block Chain Database, Database Access, Data Integrity*

II. Background

"Health facilities must control and manage records according to the legislation promulgated by government to enable healthcare workers to have timely access to accurate and reliable patient information. The legislative provisions in Section 13 of the National Archives and Records Service of South Africa Act, 1996 (Act 43 of 1996) are aimed at promoting sound records management and thereby promoting accountability and better service delivery."

-- Malebona Precious Matsoso, Director General of the National Department of Health

Medical Record Storage has always been a challenge for the Modern Health System. With the increase population, many health facilities must use electric database system to storage medical records instead of the traditional paper storage in the old days. Since the facilities must enable their workers to have not only timely but also accurate access to patient medical records, the database system must connect to the Internet or the Intranet system for records access. Thus, many concerns about patients' privacy, data misuse by certain entities and potential data leak have become a huge challenge for the database managers. In order to solve these problems, the new NFT-based block chain database can replace the traditional database.

Block Chain Database is a new application of Non-Fungible Token (NFT) technic. It is a decentralized database used to store data by blocks and the whole database is a chain of blocks. It shares certain characteristics. The first one is decentralized. This means that the data stored in it can be dispersed to many locations. The second one is immutability. Data in it is append-only which means that annotated data is the only way to fix old data and all data is trackable. The third one is that it belongs not to certain entity. But instead, it belongs to a group of people. Using its characteristics, the block chain database would be perfectly fit for storing medical records data.

The structure of the block chain database is linked data nodes. In each data node, it stores the metadata, which is the medical records in our project, transactions which shows the record has been transferred or not, previous hash code and current hash code to ensure and validate the integrity of data nodes before it. This structure ensures the immutable of data its stored. If one data node's content got changed, the other

data nodes will not accept that change using the hash keys they stored.

Though the block chain database is a good choice for nowadays information storage, it still has certain shortages like any actions to a large block chain database would be expensive. However, since we are protecting sensitive medical records, expensive actions are acceptable since security is our top priority.

IV. Objective/Goal Statement

- Implementing block chain database for medical records storage using python package BigchainDB.
- Make the database running as a server and allow user to use APIs for endpoints checkup.
- User would be able to add medical records of certain patient to the database, checkup certain medical record or checkup all medical records for certain patients via having the token ready.
- The database will validate every action user made. If the database got modified or lose integrity, the system will report immediately.
- In this project, there will be several simulate attacks using brute force on the database to check the securities of this database system.

V. System Description

- Since the database is implemented using python, it will be built using objective programming which will create two classes.
- The first class is called Block which represents one data block in the database with following parameters:
    - Data: Specific data, the medical records we want to store, json file format is prefer here.
    - Index: A number represents index, an index for the specific order in the chain
    - Previous hash: A hex hash generated number, works as a pointer to the previous data block
    - Timestamp: A timestamp use to represents the time
    - Nonce: A nonce number (normally one), meaning a number that can only be used once.

  The second class is called Blockchain which is the database itself. It has following parameters:
    - Chain: The python list type instance to store all the data.
    - New data: the new medical records to be added into our database
    - Difficulty: the number to set the complexity of hex hash code. Increase the difficulty will increase the security of database by increasing the difficulty using brute force to crack the hash code or private key.

  It also has following functions for users:
    - Add_new_data: add json format medical record to add list
    - Mine: add contents in add list to our linked database
    - Last_block: easy to access the last block.
    - Get_block_data_use_key: use hash code to access certain data block quickly
    - Proof_of_integrity: inner function automatically run checking the integrity of the entire database.
    - Modify_data_block: use hash code to modify certain data block. This operation would be expensive except have full access to the database (having admin privilege and all hash keys).
- The result of this project would be work in a webpage form. User can choose certain identity from admin, doctor, patient. Each identity will have different access to the database. Then the user can manipulate or access the database using buttons build in the webpage.

VI. Initial Protype Implementation

```
blockchain = Blockchain()
print(blockchain.chain)
for i in range(1, 10):
    data = {
        'name': 'Mining Block {}'.format(i),
    }
    blockchain.add_new_data(data)
    blockchain.mine()
print(blockchain.chain)
```

```
[Block: {'index': 0, 'data': [], 'timestamp': 1668365771.0090134, 'previous_hash': 0, 'nonce': 0, 'hash': '61f0b4620df4a3ad5cd5
d6eac355d18166cc95fb8c33bb232246e951b7c5dda6'}]
Mining block 1
Difficulty: 1
Computed hash: 0977e131c2ead08661ada01ed654fe84148a8139a899b8eb1126da495c8bf9b2
Nonce: 9
Elapsed time: 0:00:00.000126
Mining block 2
Difficulty: 1
Computed hash: 03844473f67dba91530ec7db69497bf6c28e7f10f66ffd7eac30832cbf0339bb
Nonce: 16
Elapsed time: 0:00:00.000372
Mining block 3
Difficulty: 1
Computed hash: 08d6f6efc643703d3e163c6f738e8b34ad06d9ce1bfc8693f5362b05de8f75ff
Nonce: 24
Elapsed time: 0:00:00.000443
Mining block 4
Difficulty: 1
Computed hash: 0dc73f15a4fbc9b421ae16027d378cdd37558515db10d826785e89d34e04c982
Nonce: 1
Elapsed time: 0:00:00.000065
Mining block 5
Difficulty: 1
Computed hash: 05f18e5b290b267163069ee17d161ec7278907b6b23fe20472de6ac00b14f79b
Nonce: 2
Elapsed time: 0:00:00.000064
Mining block 6
Difficulty: 1
Computed hash: 0e7fec8fc072b694a9a0d6f4b70472b93941693d1e4321e446c89e0269d83b05
Nonce: 1
Elapsed time: 0:00:00.000055
Mining block 7
Difficulty: 1
Computed hash: 082ab3aed6efeba24cd4979ce86add7ee9ffba611ddfcb1edf8ec0faf823bd78
Nonce: 13
Elapsed time: 0:00:00.000131
Mining block 8
Difficulty: 1
Computed hash: 0a92bf1387e6a686ec5c1c1bea95dd36b4796df43b875e1c7531408fd8a538a7
Nonce: 1
Elapsed time: 0:00:00.000190
Mining block 9
Difficulty: 1
Computed hash: 06041a5c18f3a68ba71739693fd1e2a68db276a5e037853b2d1f4710a530ab22
Nonce: 27
Elapsed time: 0:00:00.000217
```

- The user would be able to add block into the database. Each block will generate an hash code automatically for validating.

```
[Block: {'index': 0, 'data': [], 'timestamp': 1668365771.0090134, 'previous_hash': 0, 'nonce': 0, 'hash': '61f0b4620df4a3ad5cd5
d6eac355d18166cc95fb8c33bb232246e951b7c5dda6'}, Block: {'index': 1, 'data': [{'name': 'Mining Block 1'}], 'timestamp': 16683657
71.0090134, 'previous_hash': '61f0b4620df4a3ad5cd5d6eac355d18166cc95fb8c33bb232246e951b7c5dda6', 'nonce': 9, 'hash': '0977e131c
2ead08661ada01ed654fe84148a8139a899b8eb1126da495c8bf9b2'}, Block: {'index': 2, 'data': [{'name': 'Mining Block 2'}], 'timestam
p': 1668365771.0090134, 'previous_hash': '0977e131c2ead08661ada01ed654fe84148a8139a899b8eb1126da495c8bf9b2', 'nonce': 16, 'has
h': '03844473f67dba91530ec7db69497bf6c28e7f10f66ffd7eac30832cbf0339bb'}, Block: {'index': 3, 'data': [{'name': 'Mining Block
3'}], 'timestamp': 1668365771.010011, 'previous_hash': '03844473f67dba91530ec7db69497bf6c28e7f10f66ffd7eac30832cbf0339bb', 'non
ce': 24, 'hash': '08d6f6efc643703d3e163c6f738e8b34ad06d9ce1bfc8693f5362b05de8f75ff'}, Block: {'index': 4, 'data': [{'name': 'Mi
ning Block 4'}], 'timestamp': 1668365771.010011, 'previous_hash': '08d6f6efc643703d3e163c6f738e8b34ad06d9ce1bfc8693f5362b05de8f
75ff', 'nonce': 1, 'hash': '0dc73f15a4fbc9b421ae16027d378cdd37558515db10d826785e89d34e04c982'}, Block: {'index': 5, 'data':
[{'name': 'Mining Block 5'}], 'timestamp': 1668365771.010011, 'previous_hash': '0dc73f15a4fbc9b421ae16027d378cdd37558515db10d82
6785e89d34e04c982', 'nonce': 2, 'hash': '05f18e5b290b267163069ee17d161ec7278907b6b23fe20472de6ac00b14f79b'}, Block: {'index':
6, 'data': [{'name': 'Mining Block 6'}], 'timestamp': 1668365771.010011, 'previous_hash': '05f18e5b290b267163069ee17d161ec72789
07b6b23fe20472de6ac00b14f79b', 'nonce': 1, 'hash': '0e7fec8fc072b694a9a0d6f4b70472b93941693d1e4321e446c89e0269d83b05'}, Block:
{'index': 7, 'data': [{'name': 'Mining Block 7'}], 'timestamp': 1668365771.010011, 'previous_hash': '0e7fec8fc072b694a9a0d6f4b7
0472b93941693d1e4321e446c89e0269d83b05', 'nonce': 13, 'hash': '082ab3aed6efeba24cd4979ce86add7ee9ffba611ddfcb1edf8ec0faf823bd7
8'}, Block: {'index': 8, 'data': [{'name': 'Mining Block 8'}], 'timestamp': 1668365771.0110078, 'previous_hash': '082ab3aed6efe
ba24cd4979ce86add7ee9ffba611ddfcb1edf8ec0faf823bd78', 'nonce': 1, 'hash': '0a92bf1387e6a686ec5c1c1bea95dd36b4796df43b875e1c7531
408fd8a538a7'}, Block: {'index': 9, 'data': [{'name': 'Mining Block 9'}], 'timestamp': 1668365771.0110078, 'previous_hash': '0a
92bf1387e6a686ec5c1c1bea95dd36b4796df43b875e1c7531408fd8a538a7', 'nonce': 27, 'hash': '06041a5c18f3a68ba71739693fd1e2a68db276a5
e037853b2d1f4710a530ab22'}]
```

- The user with admin identity can view the database as a whole and print it out.
- For now, all the functions are experimenting in the Jupyter notebook. In the future, the functions can be easily performed using button and UI provided by the result webpage.
- In the future, user can choose different identity and feel the difference between using different identities in accessing the database.

- The Github link of this project is: https://github.com/DaG0ng/Block_Chain_Database

VII. Discussion of future work

- Import different identity with different access to the database
- Import more functions for different users to access the database and modify the database
- Put all codes behind the webpage and allow user without programming skills to manipulate the database.
- Show that with the set of difficulty increase, how hard it could be for attackers to crack the hash code by brute forcing.

VIII. Related Research

- The block chain database is non-SQL type database which means it can store any form of data. Medical records can save as one block's message of the block chain.
- The python package BigchainDB provide an all-in-one node docker for testing the block chain database.
- In the image created by the docker, users can test add records or checkup records using predefined APIs.
- The user would be able to test out and feel the characteristics of block chain database using the built database server and considering deploy a block chain database server for business use in the future.

VIIII. Approach/Project Scope

A. In Scope:

This project is divided into four steps in order to build a block chain database server storing medical records.

- Firstly, generating random medical records for unique patients with unique id.
- Secondly, implementing the database server using packages and add each medical record as one block of the block chain database.
- Thirdly, running the database on local host as a server for user checkup.
- Lastly, testing the add and checkup operation APIs working or not.

B. Out of Scope:

- Medical Records will be generated through simulation since this is only a test of block chain database.
- Implementation of the database must be on Linux-based system like Mac OS. Windows system might not be able to run the database server.

VI. References

1. *National guideline for filing, archiving and disposal of patient records in primary health care facilities*. MEDBOX. (n.d.). Retrieved September 22, 2022, from https://www.medbox.org/document/national-guideline-for-filing-archiving-and-disposal-of-patient-records-in-primary-health-care-facilities#GO

2. *BigchainDB package Documentation Page*. http://docs.bigchaindb.com/en/latest/index.html

3. *How to build a regex engine in python | by Lorenzo Felletti | Python in ... (n.d.). Retrieved November 14, 2022, from https://python.plainenglish.io/how-to-build-a-regex-engine-in-python-c06472bdb2f9*