BITS F463 CRYPTOGRAPHY TERM-PROJECT
TITLE: HOSPITAL MANAGEMENT SYSTEM USING BLOCKCHAIN TECHNOLOGY

Problem Statement:

The Healthcare sector gathers a lot of medical data corresponding to the patients. The security of this data is of most importance because it can be misused, and it has been proved that medical data can reveal an individual's identity. Suppose there is a common database, and anyone in the hospital/institution has access to write any data. In that case, there is a chance of misusing it and tampering with the data in the database. Also, when multiple hospitals share the data through a shared database, tracking the data modifications and ensuring a secure data entry becomes difficult. Hence, we need an authentication system and secure data entry while providing data visibility. Also, we need to keep track of the data and history of transactions.

Solution: Blockchain

Introduction to Blockchain technology:

- Blockchain is a shared, decentralized, immutable ledger that helps keep track of transactions and assets.
- The data is stored and shared across the network. Modification to the data needs to be changed throughout the network, and only after the acceptance of the nodes. Additionally, the chaining makes it almost impossible to modify the existing records.
- Hence, the data can be viewed throughout the network while maintaining data security.
- The new record is always added at the end, ensuring the correct track of the history as the new blocks cannot be placed in the middle.

How blockchain technology can be used to solve the above-mentioned problem:

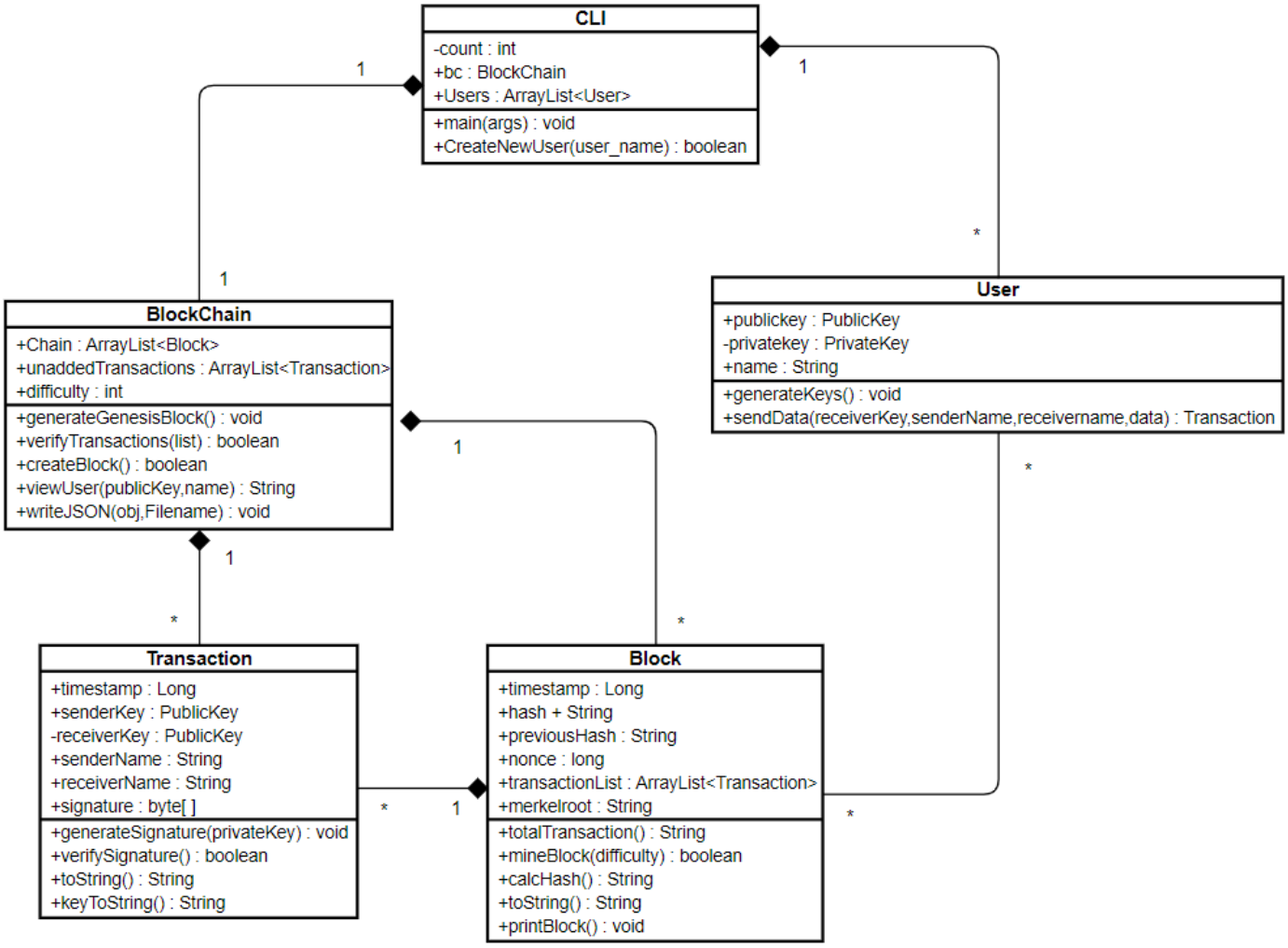
- Blockchain can be used to store the health records of the patients securely.
- We can keep track of the history of the medical reports of a patient.
- If multiple hospitals share the data, the blockchain data would be present throughout the network, making it safe from illegal modifications and provides secure access.
- The doctors can see the patient's medical history while maintaining the data securely, and new records can be added securely to the existing history of records of the patients.

Implementation of zero-knowledge proof:

We have used the standard procedure of verifying the signatures using ECDSA used in many blockchain applications including cryptocurrency. Elliptic Curve Digital Signature Algorithm (ECDSA) is a Digital Signature Algorithm (DSA) that uses keys derived from elliptic curve cryptography (ECC). Private data is chosen and applied with the ECDSA algorithm which

produces a signature that can be used to verify using a “verify” function on signature instance of ECDSA algorithm.

Functions and Classes used:



Screenshots of the application:

The application can be started by running commands

1. javac -cp ".;gson-2.6.2.jar;bcprov-jdk15to18-165.jar" CLI.java
2. java -cp ".;gson-2.6.2.jar;bcprov-jdk15to18-165.jar" CLI

We get a screen similar to the below screenshot

```
PS C:\Users\Kedar\OneDrive\Desktop\code_final\Crypto> java -cp ".;gson-2.6.2.jar;bcprov-jdk15to18-165.jar" CLI
timestamp: 1650898643976
hash: 0000bde16e742ebb171538a67a3e04885647e55c8c71b677ae224d3e5c2b9846
prevHash: 0
nonce: 45674


Timestamp: 1650898643976
Sender name: genesis
Receiver name: genesis
This is genesis block


----------------End of Transaction----------------
Welcome to the blockchain system
Current User: 'Default_User'
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option :
```

Create Users:

```
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option : 2
Enter User name : kedar
User successfully created
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option : 2
Enter User name : abhinav
User successfully created
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option :
```

Enter data:

```
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option : 1
Enter sender's user name : kedar
Enter receiver's user name : abhinav
Enter data : hi abhinav 1
DEFAULT LOGIC: A block is created when there are 4 transactions to be added
Options: (1: Add) (2: Add by creating block)
Enter option : 1
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option :
```

After 4 transactions, a block is created.
We showed the transactions which are stored in the block

```
Options: (1: Add) (2: Add by creating block)
Enter option : 1
----------------CREATING A BLOCK WITH BELOW TRANSACTIONS----------------
timestamp: 1650898881836
hash: 0000d3a13de23cd51b56b5771c7f5759a4b8ec8c9c05ca406033b39003f69fb0
prevHash: 0000bde16e742ebb171538a67a3e04885647e55c8c71b677ae224d3e5c2b9846
nonce: 86592


Timestamp: 1650898812432
Sender name: kedar
Receiver name: abhinav
hi abhinav 1


----------------End of Transaction----------------


Timestamp: 1650898852952
Sender name: abhinav
Receiver name: kedar
hi kedar 2


----------------End of Transaction----------------


Timestamp: 1650898869524
Sender name: kedar
Receiver name: abhinav
good to hear it 3
```

```
good to hear it 3


----------------End of Transaction----------------


Timestamp: 1650898881804
Sender name: abhinav
Receiver name: kedar
thanks 4


----------------End of Transaction----------------
----------------BLOCK CREATION COMPLETED----------------
Data added!
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option :
```

The created blocks data is also stored in a JSON file
The JSON file contains
First block:


```
{
  "difficulty": 4,
  "Chain": [
    {
      "timestamp": 1650898643976,
      "hash": "0000bde16e742ebb171538a67a3e04885647e55c8c71b677ae224d3e5c2b9846",
      "previousHash": "0",
      "nonce": 45674,
      "transactionList": [
        {
          "timestamp": 1650898643976,
          "senderName": "genesis",
          "receiverName": "genesis",
          "data": "This is genesis block"
        }
      ],
      "merkelroot": "MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAENQnldxQP+CE42t16pVxqQiKTicsvxkwuLLoN9H+tS0pwLD+3Prd4ccHEILZXZgcOMEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgA
    },
```


Followed by the newly added blocks. The block added in this example is:


```
    {
      "timestamp": 1650898881836,
      "hash": "0000d3a13de23cd51b56b5771c7f5759a4b8ec8c9c05ca406033b39003f69fb0",
      "previousHash": "0000bde16e742ebb171538a67a3e04885647e55c8c71b677ae224d3e5c2b9846",
      "nonce": 86592,
      "transactionList": [
        {
          "timestamp": 1650898812432,
          "senderName": "kedar",
          "receiverName": "abhinav",
          "data": "hi abhinav 1"
        },
        {
          "timestamp": 1650898852952,
          "senderName": "abhinav",
          "receiverName": "kedar",
          "data": "hi kedar 2"
        },
        {
          "timestamp": 1650898869524,
          "senderName": "kedar",
          "receiverName": "abhinav",
          "data": "good to hear it 3"
        },
        {
          "timestamp": 1650898881804,
          "senderName": "abhinav",
          "receiverName": "kedar",
          "data": "thanks 4"
        }
      ],
      "merkelroot": "MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEc3BRmdu0al2RIpwn42ETV/vgAfrPBA7Oxa9x8uG6A2jXaml9PNEcffyQm29RoSNsMEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgA
    }
  ],
  "unaddedTransactions": []
}
```


Transactions corresponding to the users can be viewed by selecting the option “View User’s data,” which prints all the transactions, including transactions sent and received by the user and the transaction's timestamp. Here, I have typed user “kedar” and below is the output:

```
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option : 3
Enter User name : kedar
--------PRINTING THE RECEIVED DATA--------
User name: kedar
Public key:
EC Public Key [83:ad:bc:84:72:c4:d1:d7:55:d3:19:f3:61:4a:62:40:bf:7f:d4:30]
            X: 73705199dbb46a5d91229c27e3611357fbe001facf040ece
            Y: c5af71f2e1ba0368d76a697d3cd11c7dfc909b6f51a1236c

Sent data: hi abhinav 1
to abhinav
with EC Public Key [60:8c:a1:06:72:68:07:54:fa:59:1a:3e:50:a8:4d:e1:e4:87:b7:cd]
            X: 53cba9fce34aab6822ea2c643c8677e9c8010b2437a7591
            Y: 12d431f4d2e27ceff93b00b16ab78adc3c1f279746136399

at timestamp 1650898812432
Received data: hi kedar 2
from abhinav
with EC Public Key [60:8c:a1:06:72:68:07:54:fa:59:1a:3e:50:a8:4d:e1:e4:87:b7:cd]
            X: 53cba9fce34aab6822ea2c643c8677e9c8010b2437a7591
            Y: 12d431f4d2e27ceff93b00b16ab78adc3c1f279746136399

at timestamp 1650898852952
Sent data: good to hear it 3
to abhinav
with EC Public Key [60:8c:a1:06:72:68:07:54:fa:59:1a:3e:50:a8:4d:e1:e4:87:b7:cd]
            X: 53cba9fce34aab6822ea2c643c8677e9c8010b2437a7591
            Y: 12d431f4d2e27ceff93b00b16ab78adc3c1f279746136399

at timestamp 1650898869524
Received data: thanks 4
from abhinav
with EC Public Key [60:8c:a1:06:72:68:07:54:fa:59:1a:3e:50:a8:4d:e1:e4:87:b7:cd]
```

We can also add blocks with less than four transactions by choosing the option “Add by creating block.” Below is the example:

```
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option : 1
Enter sender's user name : kedar
Enter receiver's user name : abhinav
Enter data : Hello abhinav 5
DEFAULT LOGIC: A block is created when there are 4 transactions to be added
Options: (1: Add) (2: Add by creating block)
Enter option : 2
----------------CREATING A BLOCK WITH BELOW TRANSACTIONS----------------
timestamp: 1650899418139
hash: 00004d26846e2205b5fe187e760d4e74a7ba184cf436a073e90ca4d7e7f5b073
prevHash: 0000d3a13de23cd51b56b5771c7f5759a4b8ec8c9c05ca406033b39003f69fb0
nonce: 275914


Timestamp: 1650899418139
Sender name: kedar
Receiver name: abhinav
Hello abhinav 5


----------------End of Transaction----------------
----------------BLOCK CREATION COMPLETED----------------
Data added!
Options (1: Enter data) (2: Create New User) (3: View User's data)
Choose your option :
```

hello.json gets:


```
  },
  {
    "timestamp": 1650899418139,
    "hash": "00004d26846e2205b5fe187e760d4e74a7ba184cf436a073e90ca4d7e7f5b073",
    "previousHash": "0000d3a13de23cd51b56b5771c7f5759a4b8ec8c9c05ca406033b39003f69fb0",
    "nonce": 275914,
    "transactionList": [
      {
        "timestamp": 1650899418139,
        "senderName": "kedar",
        "receiverName": "abhinav",
        "data": "Hello abhinav 5"
      }
    ],
    "merkelroot": "MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEc3BRmdu0al2RIpwn42ETV/vgAfrPBA7Oxa9x8uG
  }
```