CSCI 531 - Applied Cryptography Project

Chirayu Agarwal (caagarwa@usc.edu), Rishit Saiya (rsaiya@usc.edu)

1. Introduction

Technology has changed the character and speed of our life in the modern digital era. It has also been used as an asset for a long time and has long played a vital role in business, which has increased the dependency of organizations on it. Technology has an impact on every industry, and the healthcare industry is no exception [1]. As part of this shift, electronic health records have largely taken the role of traditional medical records. Rising cyberthreats are a result of technological innovation. The nature of medical records is delicate. The integrity, privacy, and confidentiality of PHI (Public Health Information) are also protected by a number of laws [2].

One of the important pieces of law is the HIPAA (Health Insurance Portability and Accountability Act), which applies to healthcare providers, medical professionals, business partners, etc. In order to combat cyber risks, audit management has evolved into one of the most crucial components of an all-encompassing cyber security architecture. Knowing who has accessed the system and what actions have been taken on these records are thus of the utmost significance.

In reality, when any subject tries to enter the system, it is essential to capture the logs. To begin with, the log files may be quite helpful in determining culpability. Furthermore, having sufficient logs would be quite beneficial for incident management. We may learn about the actions taken by various entities using log information, and this knowledge will be crucial in reconstructing the states of health information data [3]. Additionally, it might be utilized as legal proof in cases of medical negligence.

2. Goals and Objectives

The goals and objectives encapsulating this audit system are:

Privacy and Confidentiality of Data: Health information audit records have been kept private and confidential. By using a variety of cryptographic primitives, information has been protected such that unauthorized parties cannot access sensitive data in an unlawful way.

Queries: The records of health information are accessible to and searchable by authorized entities.

Immutability: Existing audit records cannot be changed by any organizations without being noticed. That will be automatically detected and reported if someone does it.

Decentralization: In this safe audit system for electronic health records, many entities can provide immutability.

This system is restricted to five patients and two audit businesses, as stated in the project document.

Patients are: `patient1, patient2, patient3, patient4` and `patient5.`
Two audit companies are: `audit1` and `audit2`.

It is assumed that audit data is sent to this system in an unencrypted (clear) format. The audit record includes the date and time of the logged event, the unique-id (security identification) of the user who entered the system, the patient ID whose health information records were viewed, and the type of action taken on the data, such as creation, copying, displaying, etc.

2. System Architecture

The technology initially enables patients to create accounts as well as audit businesses in the event that accounts and documents are missing. The system has maintained the separation between patients and audit firms. For each of the five patients, we have credentials—individual IDs and passwords. These credentials serve as the foundation for authentication. Along the same lines, there are special IDs and passwords for two audit organizations that also help to distinguish the auditors from one another. Two dictionaries are defined in order to hold credentials. Both are used for audits and patients, respectively. For identifying reasons, password-based authentication is used.

The system starts by taking input. These inputs are verified by the authentication tool. If the credentials match, only the authentication tool allows people access to the system. If credentials don't match, users won't be able to access the system. Authentication function is implemented in a way that necessitates successful authentication before beginning any system action.

After verification, we have a server where the auditing companies may submit their audit records of electronic health information. Audit enterprises have the responsibility of receiving the records, including the audit records, and then tracking and keeping an eye on these audit health data. The audit record is fed to the server, as stated in the project sheet. This input takes the form of a file, and it contains unencrypted data. Since the information is in plaintext, malevolent actors can access it without authorization. As a result, it is susceptible to manipulation, alteration, or modification for the benefit of an unauthorized person.

The system only allows patients to upload their health information records through audit companies in order to solve a few of these issues. The file's audit data is encrypted using the symmetric key method AES (Advanced Encryption Standard). The RSA public-key encryption technique is then used to produce public and private keys for all patient records in order to facilitate key management and interchange [8]. A set of the public and the private key is produced for each of the five patients. They aid in keeping data safe both during storage and during transit.

After that, Merkle Tree is used, a crucial component of the blockchain, to implement a couple of the key features of this system [5]. The technology enables the Merkle Tree to be generated by audit firms. The Merkle tree considerably decreases the amount of memory and computation needed for verification while providing efficient assurance of the accuracy and validity of audit health information [6].

In actuality, the transmission of a little quantity of data between the records is sufficient for the confirmation of verification. Additionally, it helps distinguish between legitimate and fake health data. Audit enterprises alone are responsible for the necessary Merkle tree actions. Audit enterprises must authenticate themselves before doing any Merkle tree-related actions.

Health information records may be searched for using the Merkle tree, and if any are found, they are presented using the evidence of inclusion method. Any one of the five patients can check to see if the EHR system contains their health audit record. The patient asks the server for their health records to start this procedure off. The patient verifies whether or not his or her record is stored on the server.

The server verifies whether or not the record is present. If it is, the server provides the patient with the record. The patient additionally confirms the inclusion proof in order to have confidence in the records. By calculating the requisite hashes, the inclusion proof is established. The authenticity of the record is then checked to ensure that no malicious actor has altered the current record.

Due to the dynamic nature of the operation, we may anticipate that the records will be updated when the auditing companies and patients make adjustments. As activities increase, these health audit reports will change over time. Due to this, it is extremely important to ensure that records are reliable and uncontaminated.

Merkle tree's attribute is used in this case to help the system determine the consistency of the health audit reports. We may compare a group of old files with the most recent versions in order to ensure that health information records are consistent. We can also check to see if any new material or health records have been included. The system makes sure that records are unchangeable and consistent at every level by continuously recording and monitoring changes.

3. Goals Achieved

Identification and Authorization: The system is set up such that users without the necessary credentials are unable to access it. For patient health audit records and audit companies, credentials based authentication has been established (unique user-ids and passwords).

Privacy: To maintain the privacy and secrecy of the health audit records, cryptographic techniques have been deployed. AES (Advanced Encryption Standard) symmetric algorithm. (RSA) algorithm's public key distribution and management has been deployed in order to improve these processes.

Queries: Authenticated and permitted entities may query the records because of how the system is set up. You can retrieve the health audit records if your credentials are valid and you have the necessary authorization. The validation proof is used to operate and safeguard the entire process. Additionally, records that are encrypted must first be decrypted in order for authorized users to access them. The system features a feature that decrypts the encrypted file kept on the server using AES.

Decentralization: In this system, control is not confined to a single organization. The necessary operations can be carried out by a number of organizations. The processes that check the veracity of the data and provide verification proof make sure the system is in good working order and that audit logs for health information are accessible. These systems control and manage this capability.

Immutability: This system forbids making unauthorized changes to the records. The application of the Merkle Tree enforces it. Changes, modifications, and deletions are recognized and recorded at every level thanks to consistency and validity proofs. Additionally, we save the evidence for change tracking and monitoring.

4. Cryptographic Components

This system makes use of a number of cryptographic components to achieve the project's objectives.

Those components are as follows:

1. AES algorithm (CBC mode)
2. RSA (Public key cryptography)
3. SHA-256 (Hashing)
4. Merkle Tree

Symmetric key encryption is a crucial and essential component. AES (Advanced Encryption Standard) is one of the most often utilized algorithms in many different systems. However, there are still further choices, like 3DES, IDEA, and blowfish. However, AES performs remarkably well when measured in terms of enhanced security and resilience to various assaults.

As a result, health audit records are encrypted using AES. It encrypts data in blocks of 128 bits apiece and is essentially a block cipher. It uses a substitution-permutation network to perform a series of various operations, including data shuffles and replacements. There are several ways to put AES into practice. These modes include counter mode, cipher feedback mode, cipher block chain mode, output feedback mode, and cipher block chain mode. CBC mode is the one being used for the audit system [11].

After executing XOR with the initial block of plaintext in CBC mode, the algorithm is then given the previous cipher block as input. Although decryption can occur simultaneously with encryption in CBC mode, encryption proceeds progressively. CBC mode was chosen since it is quite effective and has a stronger resistance to cryptanalysis.

The public key encryption technique is a further cryptographic scheme that has been incorporated into the system. Here, the RSA Algorithm is used. A set of public and private keys have been created for each of the patient records using RSA methods. This algorithm has been employed for key management and distribution [9].

AES key encryption and decryption are carried out using public and private keys, respectively. An extremely big number is challenging to factorize, which is essential to the RSA concept. Multiplying two big prime numbers gives one of the two numbers used to make the public key. The same two prime numbers are also used to construct the private key.

RabinMiller routines have been utilized, which can assess if a given number is likely to be a prime number or not, for the implementation of RSA key creation. Additionally, functions to create prime numbers, the modular universe, the largest common factor, and other functions that are related to creating both public and private keys have been used,

The SHA algorithm is another cryptographic element that has been used. The major purpose of this is to preserve integrity. The primary application of this approach is for Merkle tree implementation. SHA 256 is fundamentally a member of the Secure Hash Algorithm (SHA) 2 family of algorithms.

The number 256 has a distinct meaning given that it reflects the final hash digest result, which is always 256 bits regardless of how large the plaintext or cleartext is (a hashing property). It utilizes the initializing the buffer, compressing the functions, padding bits, and padding length of the SHA 256 algorithm.

The Merkle Tree is the foundation of this system, to move forward. Hash tree and the Merkle tree are similar terms. It is a type of data structure that can be applied to synchronization and data verification. It uses hash functions to maintain data inclusion, integrity, and verification. The Merkle tree's essential component is the hash function.

Large amounts of data can be efficiently and effectively mapped because of the tree's structure, and any changes - no matter how minor - may be quickly found and reported. With the aid of the Merkle tree, we can identify the location of a data change, check to see if the data is consistent with the root hash, and optimize the entire process as we won't have to go over the entire data structure.

We can learn the necessary details about the structure by looking at only a small portion of it. The entire data/content is fingerprinted using the root hash. The distribution system, which is necessary for this project, handles it fairly effectively. It verifies the consistencies remarkably well.

Each leaf node in the Merkle Tree represents a distinct data element. All patient records are the individual data elements for this system. The initial data pieces (health audit records) are first hashed using the merkle tree hash algorithm (SHA-256) to produce the leaf nodes of the tree, which is how a merkle tree is built.

The parent nodes of the leaf nodes are then created by hashing together the resulting hashed data in pairs. This procedure keeps going until it produces the Merkle root, which is a single hash. In order to offer immutability as well, Merkle tree is being used. The only thing we need to do to verify any changes in the records is to check the hash pointer of the genesis/root block. If the patient audit record is changed, then the hash pointer in its parent node and, subsequently, the

root node's hash pointer must also be modified [14]. This requires a lot of computation. When confirming the existence of content (health audit data), the Merkle tree is extremely important. It accomplishes that in a very effective and efficient way.

All of the data in the tree need not be revealed or calculated. We can determine whether or not a specific health audit record is there by computing the few necessary nodes from the desired leaf to the merkle root. As the decentralization that this system is designed to support. There is a good likelihood that several adjustments will take place simultaneously. In light of this, maintaining consistency becomes essential means that everything from the previous edition of the health audit records was included in the current version [15]. Additionally, the hierarchy should be preserved, with new records coming after older ones. Merkle Trees help to effortlessly fulfill all of these needs.

5. Prototype Implementation:

Some of the modules/libraries used to develop this system are as follows:

1. `Crypto.Cipher`
2. `AES: For Symmetric key Cryptography Crypto`
3. `Random`
4. `sys`
5. `os`
6. `math:Hashlib`
7. `sha256`: For Hashing (Integral part of Merkle Tree)

One file for every patient is created. In total 5 files (`patient<i>.txt`). This file includes patient ID, date of access and ID of user which is unique for every user accessed the system and the action that was taken on that.

Example: For `patient1`:**`d:04/20/2021;pid:486541;uid:987651;a:create;`**

Similarly, five files have been utilized. To implement various functionalities, a variety of functions have been used. The functionalities that have been implemented support encryption and decryption. Public and private keys are generated using the RSA algorithm using encryption files, decryption files, the padding function for `AES` (because it is a block cipher), `rabinmiller`, `generateprime`, `gcd` and `modular inverse` functions. The `consistency` function, `authenticity` function, and `verify inclusion` function were added to the `class merkletree` that was created to satisfy the various reasons outlined earlier in this report.

Some of the functions are shown as follows:

```python
def encrypt(message, K):
    message = pad(message)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(K, AES.MODE_CBC, iv)
    return iv + cipher.encrypt(message)

def decrypt(ciphertext, K):
    iv = ciphertext[:AES.block_size]
    cipher = AES.new(K, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext[AES.block_size:])
    return plaintext.rstrip(b"\0")
```

These are the functions of encryption and decryption of plaintext using key K. The keys are generated by the function `generateKey`. A snippet of code is as follows:

```python
def generateKey(keySize):
    p = generatePrime(keySize)
    q = generatePrime(keySize)
    n = p * q
    while True:
        e = random.randrange(2 ** (keySize - 1), 2 ** (keySize))
        if gcd(e, (p - 1) * (q - 1)) == 1:
            break
    d = findModInverse(e, (p-1) * (q-1))
    publicKey = (n, e)
    privateKey = (n, d)
    return (publicKey, privateKey)
```

As we can see here, `publickKey` and `publicKey` are generated using `n, e, d` parameters of RSA.

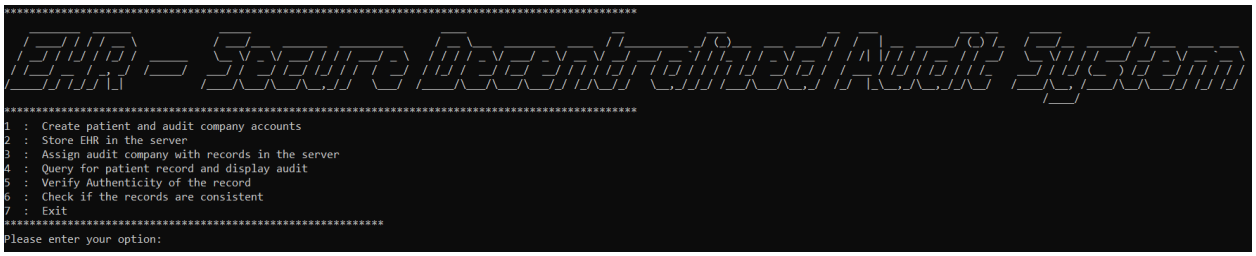
The `generatePrime` function is as mentioned below:

```python
def generatePrime(keySize):
    while True:
        num = random.randrange(2**(keySize-1), 2**(keySize))
        if isPrime(num):
            return num
```

Adding to the `RabinMiller` test [7] is done as follows:

```python
def rabinMiller(num):
    if(num % 2 == 0 or num < 2):
        return False
    if num == 3:
        return True
    s = num - 1
    t = 0
    while (s % 2 == 0):
        s = s // 2
        t += 1
    for trials in range(5):
        a = random.randrange(2, num - 1)
        v = pow(a, s, num)
        if v != 1:
            i = 0
            while v != (num - 1):
                if(i == t-1):
                    return False
                else:
                    i = i + 1
                    v = (v**2) % num
    return True
```

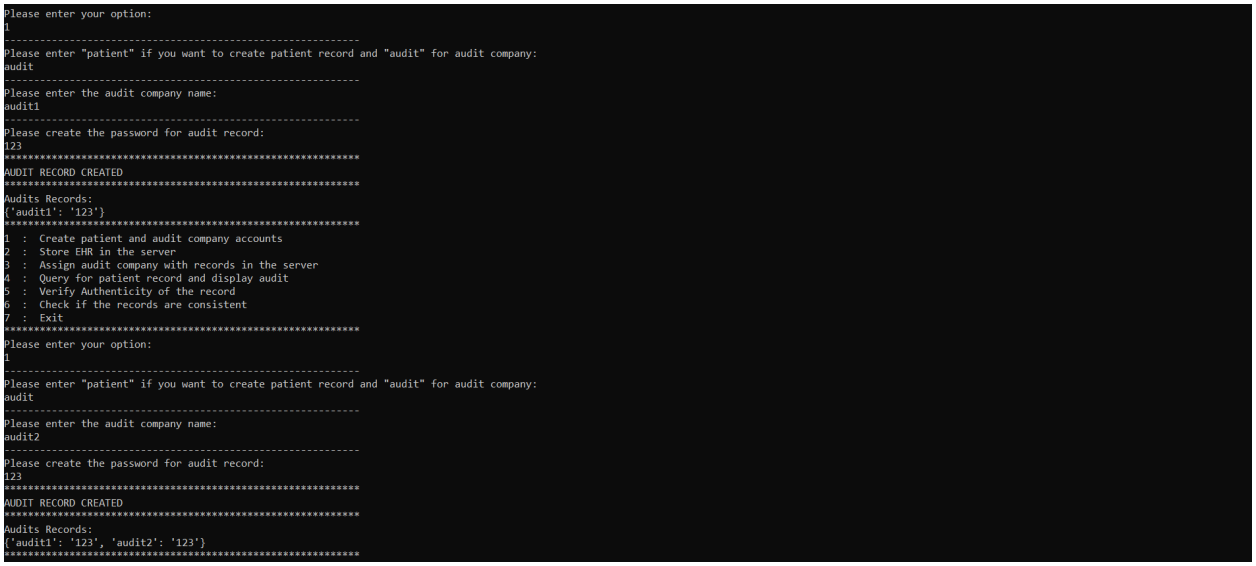
The Start Screen of the system:



Starting with the first choice, which serves to create patient records and audit the company's account if none exist, we proceed.

We must type patient when creating patient accounts. Similar to that, we must type audit if we wish to run an enterprise audit. The patient's name must then be entered. `patient1` is given as input here. The system now requests a password, which must be entered. The password is entered as 123. We therefore obtain a first set of patient credentials that are kept in a dictionary. In the same way, five patient accounts with IDs and passwords are created. The message "`PATIENT RECORD EXISTS`" will be displayed if the account already exists and someone tries to create a new account with the patient name that already exists.

The system similarly enables the creation of records for audit companies. We must first select Option 1 before typing Audit. The system allows you to enter the name of the audit enterprise after typing audit. Following name entry (`audit1` in this case), the system prompts for a password. It generates the account for the audit business after receiving the password. For the second audit enterprise, the account was created in a similar manner. These audit enterprise accounts are kept in a unique dictionary designed specifically for audit firms. We currently have five patients and two audit firms with accounts.



Moving on, the second choice involves keeping electronic health records on a server. The system has been designed such that only authenticated audit companies are permitted to keep patient records on the server.

The system first prompts the user to select the audit company that will create and upload a patient record. The audit company's name is the first question the system asks. It requests a password after accepting the audit company's account name. Once the verification has been completed successfully, the audit company is free to create and upload the records to the server. The system then asks which patient these operations should be performed on. `patient1` in this case. After taking input and doing verification, it takes input in the form of a file before successfully uploading the patient's record to the server. The system converts the audit records' plain text files to encrypted files in the background while also generating keys. This helps to protect records' confidentiality while they are in transit and storage. (The file contains patient-id, unique-id of the user and date at which the user accessed health records).

Similarly, these operations have been carried out for all the patients i.e. `patient1`, `patient2`, `patient3`, `patient4`, `patient5`. Audit companies `audit1` and `audit2` are the authorized audit companies that carried out the record generation and upload operation.

```
Please enter your option:
2
------------------------------------------------------------
Please enter the audit company name to create and upload records:
audit2
------------------------------------------------------------
Enter the password:
123
************************************************************
audit2 IS AUTHENTICATED AND CAN NOW UPLOAD THE RECORDS.
************************************************************
Please enter the patient record name to be uploaded:
patient2
************************************************************
patient2 RECORD NAME IS SUCCESSFULLY UPLOADED TO audit2 AUDIT.
************************************************************
```

Next, we go on to the third option, which is to assign records to audit companies from the server. The mechanism here enables the approved audit firms to make Merkel records for the patients. The system then requests that the audit company be given access to patient records. After incorporating all the inputs, a Merkle record containing left, right, and Merkle root is created.

```
Please enter your option:
3
------------------------------------------------------------
Please enter the audit company name to create merkle records:
audit1
------------------------------------------------------------
Enter the password:
123
************************************************************
audit1 IS AUTHENTICATED AND CAN CREATE THE MERKLE RECORD.
************************************************************
Please assign patient record name to the audit company: (Syntax: [P1,P2])
[patient1,patient2]
************************************************************
Merkle Tree:  {'7550740bb08d0b177881ae6b1042e8ef2f29dc2995a582bf62bba1f065652a22': 'e5835c223bb999eabfafc602bcd8cfd40f33eab7d141b8d7f6f5299b5ce0de38', 'e5835c223bb999eabfafc602bcd8cfd40f33eab7d141b8d7f6f5299b5ce
0de38': '7550740bb08d0b177881ae6b1042e8ef2f29dc2995a582bf62bba1f065652a22', '369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995': '369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995'}
Left Child:  ['7550740bb08d0b177881ae6b1042e8ef2f29dc2995a582bf62bba1f065652a22']
Right Child:  ['e5835c223bb999eabfafc602bcd8cfd40f33eab7d141b8d7f6f5299b5ce0de38']
Root: 369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995
************************************************************
```

Moving on to the Fourth and Fifth choices, which are the Patient Records Query and Display choices, which show an audit and authenticity check of the records. Patients can request system audit data by using the system's querying functionality. Authentication comes first. Authentication is required before any patient can query the data. Queries are only permitted following successful authentication. Following that, the audit record's existence is verified. The system's database is checked. Upon discovery, a notice is displayed stating that the system was examined, and the record was obtained.

Following the presentation of further hash values for relevant nodes, plain text versions of the records' data are then shown.Since the data is being stored encrypted, a decryption process is being done in the background. The information in the record includes the date, the patient's ID, and the user's security ID. The system enables patients to confirm the validity of records. To know if a record is immutable is helpful. The patient can trust the records after doing the necessary verification.

```
Please enter your option:
4
------------------------------------------------------------
Please enter the patient record name:
patient1
------------------------------------------------------------
Enter the password:
123
************************************************************
THE PATIENT RECORD NAME EXISTS:  ['e5835c223bb999eabfafc602bcd8cfd40f33eab7d141b8d7f6f5299b5ce0de38', '369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995']
THE PATIENT RECORD:  d:04/20/2021;pid:486541;uid:987651
************************************************************
```

```
Please enter your option:
4
------------------------------------------------------------
Please enter the patient record name:
patient2
------------------------------------------------------------
Enter the password:
123
************************************************************
THE PATIENT RECORD NAME EXISTS:  ['7550740bb08d0b177881ae6b1042e8ef2f29dc2995a582bf62bba1f065652a22', '369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995']
THE PATIENT RECORD:  d:04/21/2021;pid:486542;uid:987652
************************************************************
1  :  Create patient and audit company accounts
2  :  Store EHR in the server
3  :  Assign audit company with records in the server
4  :  Query for patient record and display audit
5  :  Verify Authenticity of the record
6  :  Check if the records are consistent
7  :  Exit
************************************************************
```

```
Please enter your option:
5
------------------------------------------------------------
Please enter the patient record name:
patient1
------------------------------------------------------------
Enter the password:
123
------------------------------------------------------------
HASH VALUES FOR VERIFYING AUTHENTICATION:
7550740bb08d0b177881ae6b1042e8ef2f29dc2995a582bf62bba1f065652a22
369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995
************************************************************
True. THE PATIENT RECORD IS AUTHENTIC.
************************************************************
```

```
Please enter your option:
5
------------------------------------------------------------
Please enter the patient record name:
patient2
------------------------------------------------------------
Enter the password:
123
------------------------------------------------------------
HASH VALUES FOR VERIFYING AUTHENTICATION:
e5835c223bb999eabfafc602bcd8cfd40f33eab7d141b8d7f6f5299b5ce0de38
369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995
************************************************************
True. THE PATIENT RECORD IS AUTHENTIC.
************************************************************
```

Likewise all the patient records have been verified.

Moving on to the final and most crucial step which is to examine the consistency of records. Record consistency in a distributed and decentralized context is a difficult undertaking. As patients and auditing organizations make modifications, audit records are updated. It is crucial to verify whether or not the records are consistent.

We compare the old and new versions of files to see if the previous version is still present in the new version, if the order is the same, and if new information has been added after the old. The system keeps track of all the modifications during the process and offers immutability evidence for all system data.

```
Please enter your option:
6
------------------------------------------------------------
Please enter the audit company name:
audit1
------------------------------------------------------------
Enter the password:
123
************************************************************
audit1 IS AUTHENTICATED AND CAN CREATE THE MERKLE RECORDS.
************************************************************
Please assign patient record names to the audit company:
Syntax:
[P1,P2]
[P1,P2,P3,P4]
------------------------------
[patient1,patient2]
[patient1,patient2,patient3,patient4]
************************************************************
YES, THE RECORDS ARE CONSISTENT
HASH VALUES:  ['369a8c28ee5927481ec34fb896910f3724dd00a1bc32eb51c891bbc2ee4a6995', 'f02018132f462123c856ab2584bc4aeeaff19cf6147819a9020b2646a695a972', 'ca5d2f76b9de5913b1ed2c80293c8a1ffe97851499b04fffc64432102af
c6d38']
************************************************************
```

The last option is to exit the program.

```
Please enter your option:
7
************************************************************
EXITING THE SYSTEM. THANK YOU.
************************************************************
```

6. Assumption and System Limitation:

File path needs to be defined in the code for its execution and every step is dependent on the previous step. Only after making all 5 patient records and 2 audit enterprises, other options (2-6) will work. The system has been implemented on a single machine.

The implementation of message exchange involves reading from or writing to a file. Five files in all are created. For every patient, one. This file contains the user's unique ID, the patient ID, and the date on which the record was accessed. Password-based authentication is implemented to authenticate patients and audit enterprises [13]. However, the system should have two-factor authentication to have robust security. For this project, only one factor of authentication which is a password is implemented. For a better implementation, a second factor (like OTP, fingerprint, etc) could be employed.

Socket programming is not utilized for network operations. In true sense, there should be comprehensive socket programming to connect between client and server and exchange data between them. Ideally, TLS 1.2 or TLS 1.3 should be implemented for the system where data is secure in transit. Proper authentication, confidentiality and integrity can be comprehensively ensured by TLS 1.2 or TLS 1.3. In this system, there is no mechanism for a digital signature certificate, thereby, the client has not adequate provisions to verify the server with high assurance.

During transit, content integrity is something that needs to be enforced in true sense. With TLS, it could have been done effectively. This system uses CBC mode for performing encryption and decryption which is reasonably good and does pretty well in achieving the objectives. However, counter mode based encryption could do even better in attaining the desired objectives with a higher degree of assurance, efficiency, performance and security.

Ideally, TLS 1.2 or TLS 1.3 should be implemented for the system where data is secure in transit. Proper authentication, confidentiality and integrity can be comprehensively ensured by TLS 1.2 or TLS 1.3. In this system, there is no mechanism for a digital signature certificate, thereby, the client has not adequate provisions to verify the server with high assurance. The strict enforcement of content integrity during transit is necessary.

With TLS, it could have been done effectively. This system uses CBC mode for performing encryption and decryption which is reasonably good and does pretty well in achieving the objectives. However, counter mode based encryption could do even better in attaining the desired objectives with a higher degree of assurance, efficiency, performance and security.

7. Conclusion

In conclusion, there are benefits and drawbacks to the decentralized audit system. On the plus side, it offers better accuracy and efficiency along with more accountability and openness [10]. Decentralization aids in the distribution of authority and power, which lowers the possibility of corruption and manipulation. The adoption of blockchain technology also guarantees data immutability and integrity [4]. However, the decentralized audit approach has significant disadvantages as well. The possible lack of uniformity and consistency among many platforms and systems is a significant problem. The potential for security lapses and hacker attempts, which could jeopardize private financial information, is another worry. Decentralized system utilization can also be challenging and complicated, necessitating particular training and experience [12].

Overall, the decentralized audit system holds the promise of revolutionizing the auditing process and elevating public confidence in financial institutions. However, it also necessitates giving considerable thought to and managing any potential dangers and difficulties. The advantages and disadvantages of each new technology or system must be carefully considered before being put into use.

8. References

[1] Charalampos Stamatellis, Pavlos Papadopoulos, Nikolaos Pitropakis, Sokratis Katsikas, William J Buchanan, A Privacy-Preserving Healthcare Framework Using Hyperledger Fabric, arXiv - CS - Cryptography and Security, 2020

[2] D Tith, JS Lee, H Suzuki, W Wijesundara, N Taira, T Obi, N Ohyama, Application of Blockchain to Maintaining Patient Records in Electronic Health Record for Enhanced Privacy, Scalability, and Availability, Healthcare Informatics Research 26 (1), 3-12, 2020.

[3] Diver, Sorcha. ”Information security policy-a development guide for large and small companies.” Sans Institute (2007): 1-37.

[4] M. M. Madine et al., Blockchain for Giving Patients Control Over Their Medical Records, in IEEE Access, vol. 8, 2020

[5] Li, Hongwei, et al. "An efficient merkle-tree-based authentication scheme for smart grid." IEEE Systems Journal 8.2 (2013): 655-663.

[6] Xu, Jian, et al. "Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures." Journal of Network and Computer Applications 107 (2018): 113-124.

[7] Arnault, François. "Rabin-Miller primality test: composite numbers which pass it." mathematics of computation 64.209 (1995): 355-361.

[8] Dongjiang, Li, Wang Yandan, and Chen Hong. "The research on key generation in RSA public-key cryptosystem." 2012 Fourth international conference on computational and information sciences. IEEE, 2012.

[9] Nagar, Sami A., and Saad Alshamma. "High speed implementation of RSA algorithm with modified keys exchange." 2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT). IEEE, 2012.
[10] Menachemi, Nir, and Taleah H. Collum. "Benefits and drawbacks of electronic health record systems." Risk management and healthcare policy (2011): 47-55.
[11] Belchior, Rafael, Miguel Correia, and André Vasconcelos. "Towards secure, decentralized, and automatic audits with blockchain." (2020).
[12] Du, Yuefeng, et al. "Enabling secure and efficient decentralized storage auditing with blockchain." IEEE Transactions on Dependable and Secure Computing 19.5 (2021): 3038-3054.
[13] Tian, Guohua, et al. "Blockchain-based secure deduplication and shared auditing in decentralized storage." IEEE Transactions on Dependable and Secure Computing 19.6 (2021): 3941-3954.
[14] Chen, Yu, et al. "PGC: decentralized confidential payment system with auditability." Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. Springer International Publishing, 2020.
[15] Fan, Kuan, et al. "Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT." Future Generation Computer Systems 110 (2020): 665-674.