International Journal of Digital Crime and Forensics
Volume 13 • Issue 5 • September-October 2021

Secure Storage and Sharing of Visitor Images Generated by Smart Entrance on Public Cloud

Rajashree Soman, Jain University, Bengaluru, India
Sukumar R., Jain University, Bengaluru, India

ABSTRACT

Visitor validation at entrance generates a large number of image files that need to be transmitted over to cloud for future reference. The image data needs to be protected by active and passive adversaries from performing cryptographic attacks on these data. The image data also needs to be authenticated before giving it for future use. Focusing on reliable and secure image sharing, the proposed method involves building a novel cloud platform, which aims to provide a secure storage in the public cloud. The main objective of this paper is to provide a new way of secure image data storage and transmission on cloud using cryptographic algorithms. To overcome the flaws in current system, a novel method using BigchainDB, which has advantages of blockchain technology and traditional database, is proposed for storing attributes of image.

KEYWORDS

AES, Applied Cryptography, BigchainDB, Blockchain, SHA

INTRODUCTION

In the present-day environment visitor validation at the entrance of home or office is very essential to prevent unauthorised persons from entering the building. The traditional methods have limitation where identity cards can be misused, and data can be manipulated causing security issues. Bitcoin (Nakamoto & Bitcoin, 2008) is a revolutionary idea in the cyber space. The real gem technology was not the bitcoin, but the distributed technology used by bitcoin known as Blockchain (Crosby, Pattanayak, Verma, & Kalyanaraman, 2016), (Zheng, Xie, Dai, Chen, & Wang, 2018). The use of bitcoin was limited to financial purposes while Blockchain technology has a wide scope. The focus is on the application of Blockchain in the area of Identity and Access Management (IAM)(Witty, Allan, Enck, & Wagner, 2003). Self-Sovereign Identity will reshape the future of IAM and Blockchain is the technology that will propel digital identity to self-sovereign identity (SSI). SSI is a model for managing digital id in which an individual has sole ownership to control their identity information. No third party should have access to this information. At the same time, the user should be able to interact with the digital world using this digital id without any difficulty. It basically removes the dependency on third party services which are responsible for verifying the user id credentials in a normal authentication system. In Section II different authentication mechanism and evolution of the identity management system is explained. Section III deals with the concepts involved in Blockchain and explains how it can be used to solve the self-sovereign id issue. A comparison between various Blockchain technologies is done in section IV. Section V describes proposed methodology to secure files using blockchain. (Bhardwaj et al., 2020) specifies usage of blockchain technology for various bigdata applications.

DOI: 10.4018/IJDCF.20210901.oa4

AUTHENTICATION TECHNIQUES

Authentication (Needham & Schroeder, 1987) is the process of proving the identity of a person or entity. It is ascertaining whether someone is truly the one who that person is claiming to be. Entering a username and password while logging to a website is the most common example of authentication process. While this may be one of the most common and simplest, this is not the safest approach.

Authentication Factors Knowledge

There are mainly three common factors (Gibson, 2011) used for authentication and possession.

- The first factor is basically a key like a password or a pin. Ideally it is a secret key that is known only to the user. It is the most common factor and the least secure.
- The second factors refer to some item like a smartcard or a token generator. It improves security considerably, but user may run into trouble if someone is able to get hold of this item or somehow, the item is last. Therefore, it is not fully safe.
- The third factor is Inheritance which is one of the safest factors. It is hard to replicate and impossible to forget or lose. This factor refers to the identity of a user like fingerprint, Iris, Voice etc. are used for authentication. Only way to break this system is to capture the signature and feed into the system which is based on digital signature.

Even though these are the main factors of authentication there are also other factors coming up as the technology is getting advanced like authentication based on the parameters like the location and the activities of the user. Using a single factor for authentication is no longer secure for the current world. Since no single authentication factor is impenetrable, we end up combining two or more of these factors for authentication known as the multi-factor authentication or MFA (Ometov et al., 2018).

BLOCKCHAIN BASED SOLUTION

In this section analyses various terms related to Blockchain which will form the fundamentals of the SSI Management Model.

Core Concepts Blockchain

Blockchain is the brainchild of Satoshi Nakomoto. It was originally developed as the backbone technology of Bitcoin in 2001. Blockchain is an incorruptible digital ledger of time-stamped data which is not maintained by a single entity but by a collection of nodes which forms the network. The distributed and incorruptible nature of Blockchain is what makes it interesting. Blockchain consists of several blocks of data chained together. Each block has a record of its own hash as well as the hash of the previous block, which makes it very difficult to modify this ledger. If an active adversary tries to modify the contents of a block maliciously, the entire chain will lose its integrity and will not be considered valid. This is because a block contains the hash of the previous block. So if an adversary tries to change the contents of a block then its hash will change, then when the next block compares this hash with the hash stored by that block, it will fail. To add a new block to the chain, it must be approved by more than 50% of the nodes in the network. This is done using a consensus algorithm. There are many consensus algorithms used, among which the most widely used is proof-of-work. Ethereum, Hyperledger Fabric, IBM Blockchain are some of the most common Blockchain platforms available.

Hashing simply refers to the process of generating a hash value from a string of text. Basically a hashing function takes an input string of any length and returns an output string of fixed length. Ideally, the output string should be unique for the input string, meaning no other string should produce the

same hash. Hashing offers protection to data against tampering. Any small change to the data will make a big difference in its hash. Cryptographic hash functions are special class of hash functions which has some special properties which makes them suitable in the field of cryptography. Blockchain uses hashing to ensure that the chain is tamperproof. Each block contains the hash of the previous block as well as its own hash. Every block stores its own hash as the root of the Merkle tree(Cosset, 2017).

In a **Merkle Tree**, every non-leaf node represents the hash of its children and the leaf node represents data. A pair of nodes are repeatedly hashed until there is only one node left which turns into the root of the tree. This top hash represents the hash of the entire tree and is called the root hash. A Merkle tree (Buchmann, Dahmen, & Schneider, 2008) takes 'n' number of hashes and returns a single hash. Merkle trees are used for efficient and secure verification of large datasets which make perfect sense to be used in a blockchain. Since blockchains can get pretty big in size, Merkle trees provide the perfect solution for storage and verification of integrity of the chain. If the root hash is public, then anyone can verify if some data is consistent with the root hash by checking only a chunk of hashes rather than the entire set of hashes. This is known as Merkle proof.

Consensus algorithms (Mingxiao, Xiaofeng, Zhe, Xiangwei, & Qijun, 2017) are used to add a block to the blockchain. **Proof-of-Work (PoW)** is one of the most widely used consensus algorithm. Ethereum(Wood, 2014) and Bitcoin uses PoW with some differences in implementation. This algorithm gives the right of adding a new block to the chain to a node who is able to solve a complex mathematical problem first. The nodes in the chain compete with others nodes to solve this complex problem and are called the miners. PoW takes the workload as a safeguard. If anyone wants to tamper with the blockchain they should have more than 50% of computing power of all the nodes in the chain.

Smart contracts (Bhardwaj et al., 2020; Kumar, Kumar, & Panduranga, 2013) are pieces of code which are self-executing, self-verifying and tamper resistant. Smart contracts will be holding some logic to do some operations, and once they are deployed on the blockchain, no one would be able to change the nature of this operations. A smart contract that has logic to issue movie tickets when user pays certain amount of money and once this smart contract is deployed on to the blockchain no other user would be able to get a movie ticket without actually paying for it. The important thing to note here is that users does not need a central authority to make sure that they have paid the specified amount of money and also they get movie tickets if you have paid for it. This removal of the third party is achieved by combining the features of blockchain and smart contracts. The deployed smart contracts on the blockchain will be having a unique address associated with it using which anyone can trigger this contract. Solidity is the most common language using which smart contracts are written.

Zero Knowledge proof (Van Amersfoort) agreement is a method by which one party is able to verify the claims of another party without seeing the actual data. Zero-Knowledge proof can prove something is true to another party. The other party gets no additional information except for the fact that the claim is true. For example proving that a user has 2 different coloured balls without telling the other party the actual colours of the balls or user can claim that he/she are above the age of 18 without specifying the age. Zcash is a well-known blockchain project which has implemented Zero-Knowledge proof.

BLOCKCHAIN PLATFORMS

Blockchain serves as the backbone of the SSI management solution that we identified. Developing a blockchain application from scratch can be a tough job but blockchain platforms make the job of developers easier. However, there are legions of blockchain platforms available, one should be careful in choosing the platform. Different platforms maybe offering different utility. This paper tries to put bird eye on some of the most popular and easy-to-use blockchain platforms.

Ethereum

Ethereum (Buterin, 2013)is a global, open blockchain platform that enable users to build and deploy decentralized applications seamlessly. The crypto currency that fuels Ethereum is called Ether. The main difference between the Ethereum and bitcoin was that, Ethereum is programmable, which means developers can use it to build decentralized applications. It is based on Ethereum Virtual Machine which enable a run-time environment to run smart contracts. It is a permission-less ledger which uses proof-of-work as the consensus algorithm. Ethereum community is one of the largest and the most active blockchain community, which means the developers will be having huge support and guidance by the community.

Hyperledger Fabric

Hyperledger Fabric(Cachin, 2016) is a blockchain project which comes under the Hyperledger project, which is an umbrella project of open source blockchain and related tools, started by the Linux Foundation. It has a modular architecture and allows plug and play components such as consensus and membership services which makes it a versatile blockchain platform. It supports private transactions and confidential contracts which makes it ideal for use within private enterprises. It is a permissioned ledger which requires all the participants to have known identities.

Stellar

Stellar() is an open source distributed ledger that facilitates cross-platform asset transfer without incurring additional middleman fees. It uses its own consensus protocol knows as Stella Consensus Protocol (SCP) which adds a lot of flexibility in reaching consensus between the nodes. It can function as both public and private ledger according to the requirement and supports smart contracts.

Openchain

Openchain(Baliga, 2016) is an open source distributed ledger which is aimed at organizations that issue and manage digital assets. It uses a Portioned Consensus algorithm where every Openchain instance has only one authority validating transactions. Different transaction will be validated by the corresponding authorities, which means this network doesn't require any miners. In this blockchain transaction process is free of cost as there are no miners and is one of the most efficient platforms.

BigchainDB

BigchainDB (McConaghy et al., 2016) is different from the other blockchains. It starts as distributed big data database and then adds the features of blockchain to it which include decentralized control, immutability, and the transfer of digital controls. It does not user Merkle Tress, rather each write is recorded on the blockchain. It supports both public and private networks and has no native currency, which means any asset, token or currency can be used. BigchainDB runs on any flavour of Linux operating system and depends on MongoDB or RethinkDB.

PROPOSED METHODOLOGY

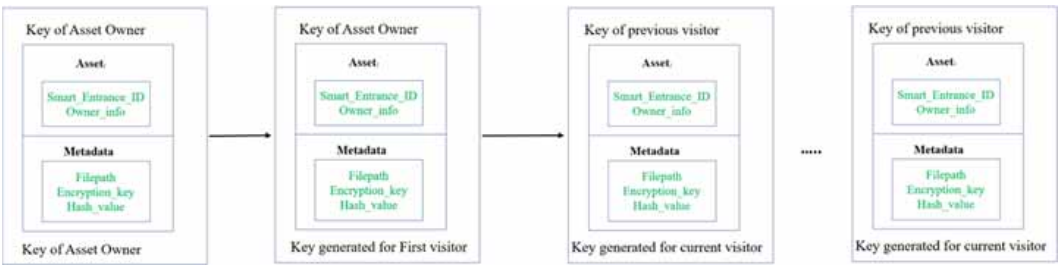
In the current scheme the data is stored as incoming timestamp, image file for all users who are entering the building. The data (outgoing time stamp, image file) is not recorded or stored. In the scenario where large number of visitors are entering the building and some issue (like theft of resources) needs to be detected at the time of detection the users who had already entered and left the building are likely to have caused the problem. The visitors who are still in the building can be scanned manually the data to be encrypted, stored, and analysed becomes much lesser. This also helps in reducing the cloud storage space usage. Following section explains the proposed methodology for Secure File Storage & Sharing on Cloud Using blockchain:

1. Secure Login: The user must be authenticated before the whole process occurs.
2. File upload: The upload process mainly involves three steps:
   a. Step 1: Image of the visitor with timestamp needs to be taken.
   b. Step 2: Password and hash of the file will be generated.
   c. Step 3: Encrypt the file using generated password.
   d. Step 4: Add file path, user password, and filename with hash to BigchainDB for that user.
3. File Download: The user will download the file with the downloadable link which is sent via the mail.
4. File Decryption: The decryption process mainly involves these steps:
   a. Step 1: Access the BigchainDB for recovering password after validation of user credentials
   b. Step 2: Find hash of file compare with hash stored in BigchainDB
   c. Step 3: Send the download link through chat, that gives decrypted file.

In this proposed solution blockchain will be created for the files uploaded by each user in which each block has data with respect to each image file where timestamp, password, file path and file hash along with device id which generated the image will be maintained as shown in Figure 1.

The data of a visitors captured at the smart entrance should be immutable (cannot be modified or deleted). The traditional database does not provide immutable property. Even though blockchain provides immutable property it has high latency. BigchainDB which is formed for providing immutable property(Pandey, Kumar, Singha, Gayathri, & Kumar, 2020) with low latency(McBee & Wilcox, 2020) can be used for smart home validation application. For creating a block key should be generated as shown in Figure 2.

Figure 1. Blocks of BigchainDB



An asset for smart door(Smart_Door_Data) which consists of an ID for door address and owner e-mail address value key pair and a description field can be created as shown in Figure 3.

A metadata (Visitor_Data) is generated every time when a visitor data is received at the cloud. Metadata consists of following details in value key pair (shown in Figure 4):

- Image_file_path: Path of the image as each image might be stored in daily basis its path may be different folder, so this variable holds the path of visitor.
- Image hash value: SHA256 value of the image used for authentication purpose.
- File encryption password: Password used for encrypting file. AES with CBC mode of operation used for encrypting the image
- Time stamp: Indicate the time when the image was captured (indicating the entry of visitor at that point of time)

Figure 2. Generation of Key

```
>>>
>>> from bigchaindb_driver.crypto import generate_keypair
>>> alice, bob = generate_keypair(), generate_keypair()
>>>
>>> alice
CryptoKeypair(private_key='DQc4oMUUdzWZEaG5Fjhzst66K1Jn68wo2ccPgGGjfL
J9', public_key='2MdBN7HTKd4k2hu1oMnzHz8La3YwWyNpCeHZppX9ewgT')
>>> bob
CryptoKeypair(private_key='HhYJqRocG8J7M9DH42KPqpTB5DeJuGMfi3VmwwcCj7
tj', public_key='DLKuWc6WdkijH8ZkHmtGo4SWqJa6hk8jtS6ksmFtQ1xE')
>>>
>>>
```

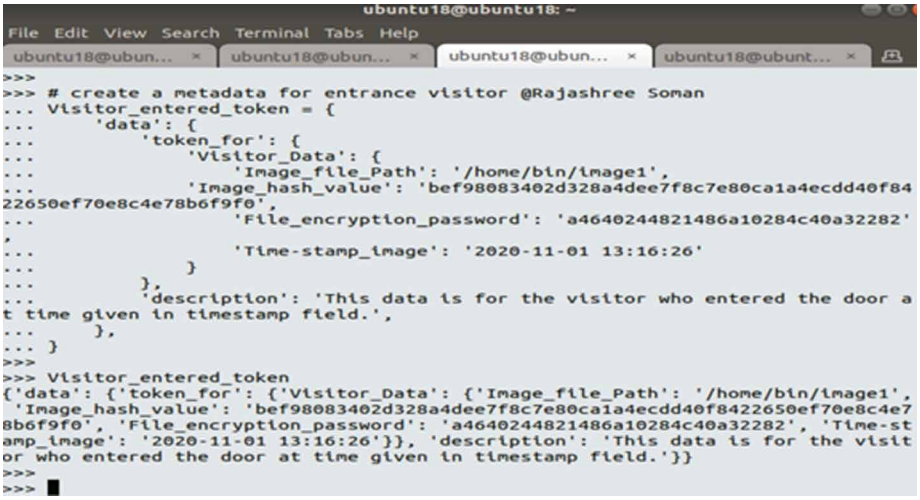
Figure 3. Creation of Assets in Bigchaindb

```
>>>
>>>
>>> Visitor_Data_token = {
...     'data': {
...         'token_for': {
...             'Smart_Door_Data': {
...                 'Smart_Door_ID': 'LR35902',
...             'Mail_Addr': 'srajashrees@gmail.com'
...             }
...         },
...         'description': 'This door image data is owned by the owne
r address defined in Owner_mail_Address.',
...     },
... }
>>>
>>> Visitor_Data_token
{'data': {'token_for': {'Smart_Door_Data': {'Smart_Door_ID': 'LR35902
', 'Mail_Addr': 'srajashrees@gmail.com'}}, 'description': 'This door
image data is owned by the owner address defined in Owner_mail_Addres
s.'}}
>>>
>>>
```

When the transaction is created it requires two keys to create a block in Bigchaindb. The initial block creation uses same key of the owner twice to indicate an asset belonging to owner. Then each transaction is created by using previous block key and new key which is generated with visitor data in metadata. Each transaction therefore consists of asset data, metadata signed by two keys (previous block key, new key) as shown in Figure 5.

For file upload option user will be asked with the password which will be used for generating a secret key. The secret key will be used for encrypting the file. This key will also be stored in block of the bigchaindb along with file path and hash of that file. These values (secret key and hash value) will be used for decryption of file and authentication of decrypted file. The encryption process implementation mainly involves three phases

Figure 4. Creation of Metadata for visitor in BigchainDB



1. Generation of keys
2. Encrypting the file
3. Creating a block and secret key with file path and adding that block to the current block chain.

Generation of Key

For generating the key along with the user entered password a random variable generated by using inbuilt /dev/urandom function. The implementation of key_gen.c is shown in Figure 6.

The results for two files with one-character change is shown in Figure 7. The keys generated for two files show the existence of randomness.

Encrypting the file using above generated password:

Figure 5. Creation of transaction based on metadata

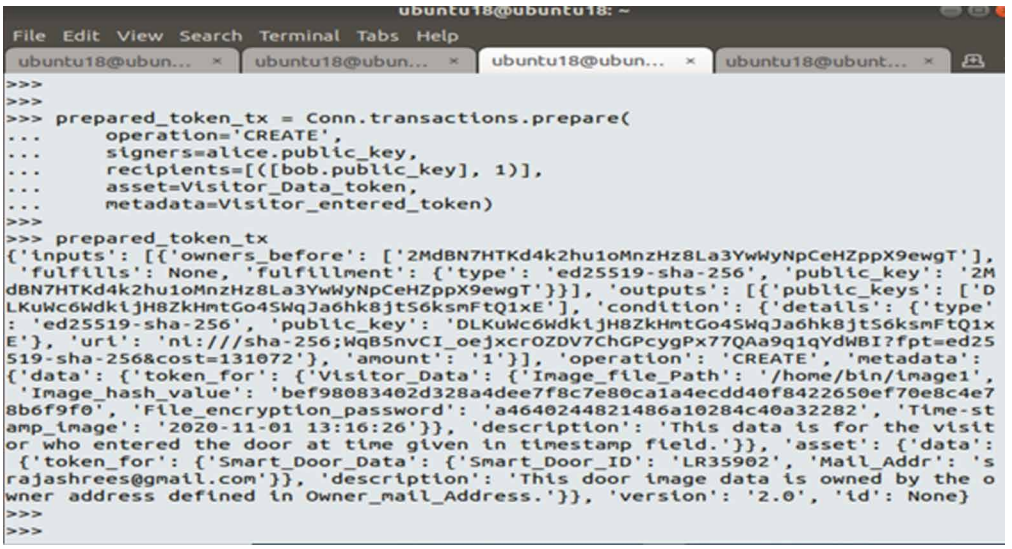


Figure 6. c-code for key generation

```
printf("Enter the password maximum 16 charecters\n");
scanf("%s",password);

p=key_u;

random = fopen("/dev/urandom", "r");
for (i = 0; i<16; i++)
{

        fread(key, sizeof(unsigned char) , 1, random);
        memcpy(p,key,sizeof(unsigned char));
        p+=1;
}
for(i=0;i<16;i++)
{

        *key=*(key_u+i)&*(password+i);
        result[i]=*key;
        printf("%x",result[i]);
}
```

In this proposed method files are encrypted using a symmetric block cipher AES algorithm with CBC mode of operation. Advanced Encryption Standard (AES)(Frankel, Glenn, & Kelly, 2003) is a popular, symmetric, block cipher scheme. Compared to its predecessors, AES offers better security, as well as performance. This is because AES employs a series of related operations known as a substitution-permutation network. There are different variants of AES (128/192/256), depending on the key sizes. Among its variants, AES-256 even qualifies itself as a quantum-resistant scheme. To date, AES has not been subject to any cryptanalytic attacks. Nonetheless, the security of AES lies in its correct implementation and proper key management.

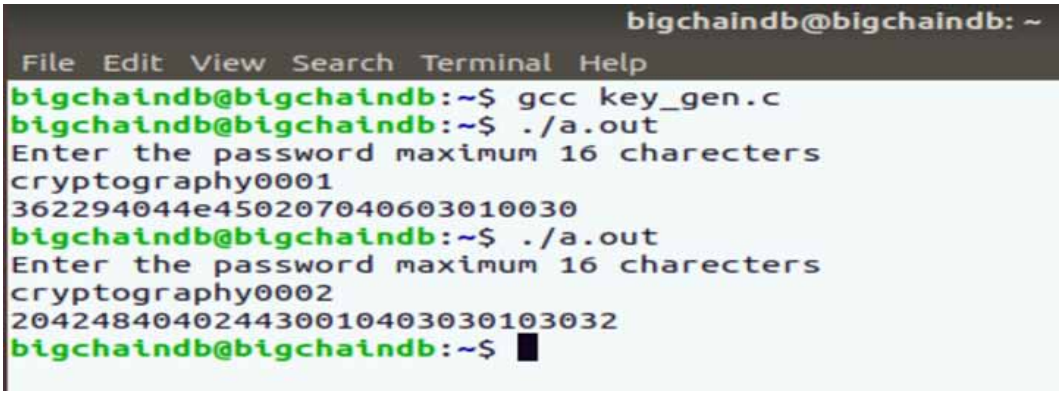
AES achieves confusion and diffusion by employing the substitution-permutation network. It involves a series of interrelated operations like the substituting inputs by specific outputs and shuffling of bits. AES-128 considers 128-bits as 16-bytes, which are arranged in the form of a matrix for further processing. Unlike its predecessors, the number of rounds in AES varies with key-size. The relation between key-size and the number of rounds is as follows,

AES Encryption (Working Principle)

The encryption process comprises of following four sub-processes,

1. Byte substitution or sub-bytes: It is a non-linear substitution step, wherein the 16-bytes input is substituted using a fixed S-box table. The result of this step is a 4x4 matrix. It is the simplest of substitution of each byte, which is used to provide confusion. Each byte of the state is replaced

Figure 7. Output key



by a byte indexed by row (left 4-bits) and column (right 4-bits). For example, 95 is replaced by a byte in the 9th row and 5th column. The S-box is constructed using a well-defined transformation of values in Galois Field (GF). The Byte-substitution operation involves 16 independent byte to byte transformations.

2. Shift Rows: Each row of the 4x4 matrix is shifted to the left. This shifting operation permutes the bytes to provide diffusion. The circular byte operation for each row is performed as follows,
3. Mix Columns: Each byte in every column of the 4x4 matrix is replaced by a new value which is dependent on all the 4-bytes in the column. Each column of the 4x4 matrix is processed separately. Like Shift-rows, Mix-Columns is also used to provide diffusion to the cipher. This operation is effectively a matrix multiplication in Galois Field (GF) using prime polynomial, $m(x) = x^8 + x^4 + x^3 + x + 1$. It is to be noted that this operation is not carried out in the last round.
4. Add Round Key: In this subprocess, 16-bytes of the 4x4 matrix is considered as a 128-bits in this round. It is XORed with a 128-bits of round key. A round key word is added to each state of the column matrix. This operation is a matrix addition. In the last round, the output of this operation is a cipher text. Otherwise, the result of this operation is an input to the next round. During decryption, the XOR operation performed here, itself serves as the inverse.

Encryption process is done using openssl() command using pbkdf2 option which does encryption based on password provided. In this proposed method keys are randomly generated and used for encryption process. This key along with encrypted file is stored in cloud along with the hash value of the file before encryption for verifying file after decryption. The hash value of the file is found using sha256sum(Kelly & Frankel, 2007) as shown in Figure 8. Decryption of file also done using openssl with the password retrieved from the blockchain for that file. Once the file is decrypted the hash value of the file is generated to compare with the original value there by authenticating it against the modifications if at all occurred.

When a user wants to download a file, authentication happens. Once the user is authenticated, a download link will be generated after the process of decryption is done using the stored password and decrypted file is generated. Decryption process implementation is also done using an inbuilt AES module supported by openssl. Both decryption and encryption process is shown in Figures 9 and 10 using openssl command.

Even though the encryption and decryption processes appear to be closely related, they need to be implemented separately.

Figure 8. Authentication using SHA

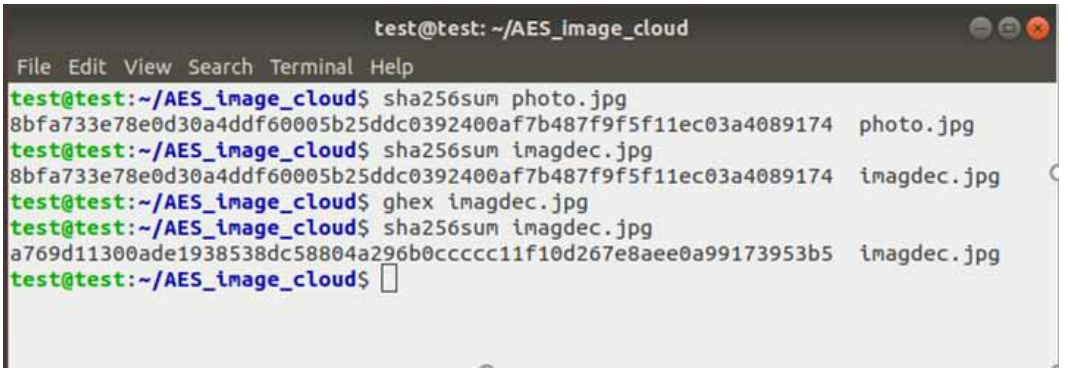
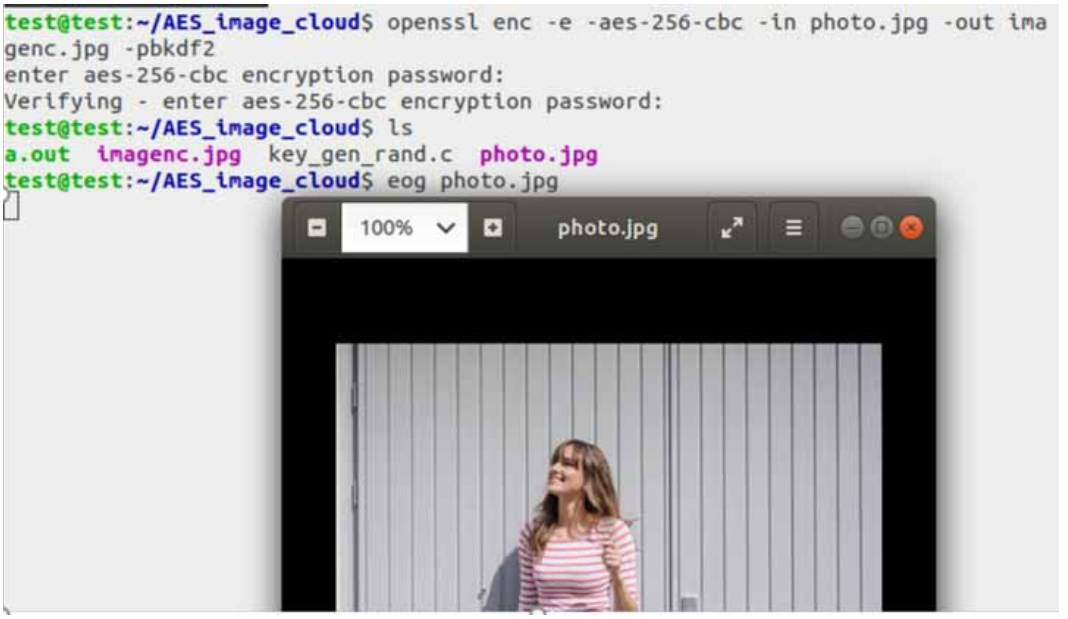


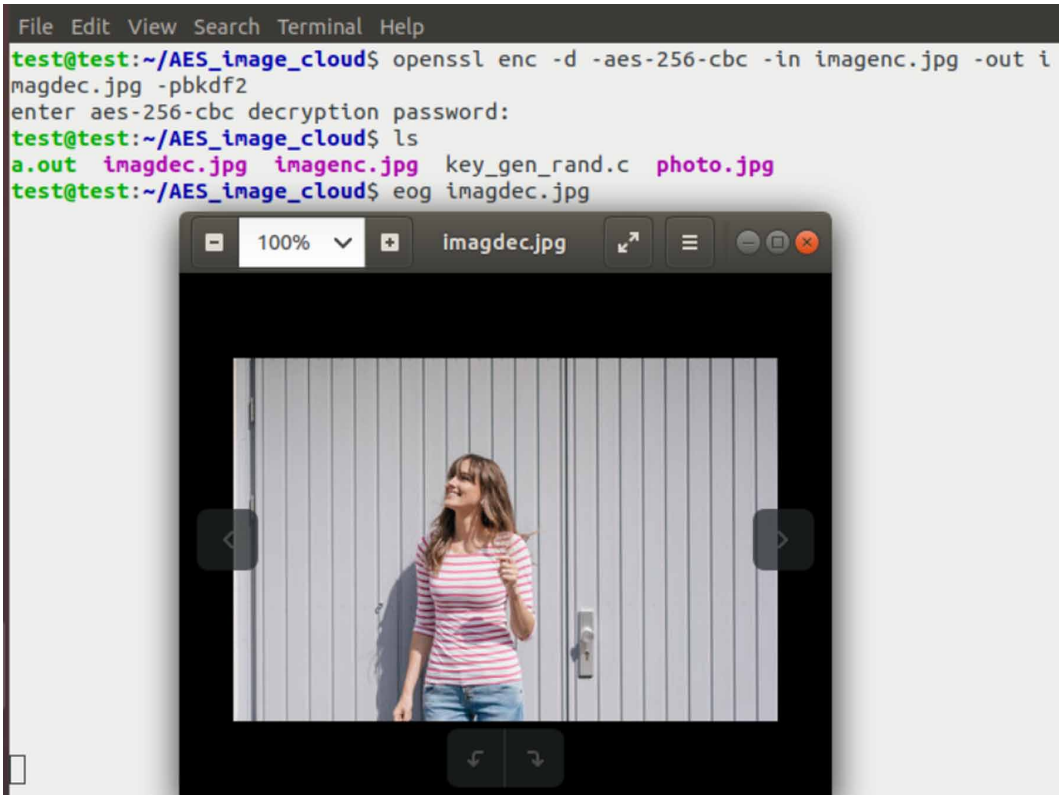
Figure 9. Encryption of file using openssl AES-256-CBC



CONCLUSION

The visitor validation is very important process in smart home or office. Each of the visitor information image needs to be stored in cloud as it might be required for future analysis in case of accidental or intentional cybercrimes. In this paper an authenticated encryption of visitor image files is proposed where images are stored securely in encrypted format in the cloud and retrieved. For storing sensitive information like password for decryption, hash value of file for authentication with respect to each image blockchain technology is used. The filename entered by user is acting like seed which is used to generate the key. The hash value ensures that there is no tempering of image data, if any of such event has occurred then hash value will not match there by indicating a modification done by adversaries to original image. The limitation with our proposed methodology is, it can only indicate that image data has been tampered but it cannot predict the amount of changes on the actual image data. Since the quantity of changes cannot be determined it will not be possible to retrieve the original data.

Figure 10. Decryption of file using openssl



REFERENCES

Baliga, A. (2016). The blockchain landscape. *Persistent Systems, 3*(5).

Bhardwaj, A., Shah, S. B. H., Shankar, A., Alazab, M., Kumar, M., & Gadekallu, T. R. (2020). Penetration testing framework for smart contract Blockchain. *Peer-to-Peer Networking and Applications*, 1–16.

Buchmann, J., Dahmen, E., & Schneider, M. (2008). *Merkle tree traversal revisited.* Paper presented at the International Workshop on Post-Quantum Cryptography.

Buterin, V. (2013). *Ethereum white paper*. GitHub repository. EOS. IO technical white paper v2.

Cachin, C. (2016). *Architecture of the hyperledger blockchain fabric.* Paper presented at the Workshop on distributed cryptocurrencies and consensus ledgers.

Cosset, D. (2017). *Blockchain: what is in a block?* https://dev. to/damcosset/blockchain-what-is-in-a-block-48jo

Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, *2*(6-10), 71.

Frankel, S., Glenn, R., & Kelly, S. (2003). RFC3602: The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC Editor.

Gibson, D. (2011). *Understanding the three factors of authentication.* Pearson IT Certification.

Inc., OpenSSL Foundation. (n.d.). *OpenSSL.* www.openssl.org/docs/man1.1.1/man1/openssl-enc.html

Kelly, S., & Frankel, S. (2007). *RFC 4868-Using HMAC-SHA-256*. Academic Press.

Kumar, S. N., Kumar, H. S., & Panduranga, H. (2013). *Hardware software co-simulation of dual image encryption using Latin square image.* Paper presented at the 2013 fourth international conference on computing, communications and networking technologies (ICCCNT).

McBee, M. P., & Wilcox, C. (2020). Blockchain technology: Principles and applications in medical imaging. *Journal of Digital Imaging*, 1–9.

McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., . . . Granzotto, A. (2016). *BigchainDB: a scalable blockchain database.* White Paper, BigChainDB.

Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE.

Nakamoto, S., & Bitcoin, A. (2008). *A peer-to-peer electronic cash system.* Bitcoin. https://bitcoin. org/bitcoin. pdf

Needham, R. M., & Schroeder, M. D. (1987). Authentication revisited. *Operating Systems Review*, *21*(1), 7–7.

Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., & Koucheryavy, Y. (2018). Multi-factor authentication: A survey. *Cryptography*, *2*(1), 1.

Pandey, A., Kumar, A., Singha, A., Gayathri, N., & Kumar, S. R. (2020). 4 Blockchain Databases 2. *Blockchain, Big Data and Machine Learning: Trends and Applications*, 97.

Van Amersfoort, J. (n.d.). *Hand-out Zero-Knowledge proofs*. Academic Press.

Witty, R. J., Allan, A., Enck, J., & Wagner, R. (2003). *Identity and access management defined. Research Study SPA-21-3430*. Gartner.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper, 151*(2014), 1-32.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, *14*(4), 352–375.

Rajashree Soman received the Bachelor of Engineering degree from Visvesvaraya Technological University, Belagavi, India, in 2003. She is pursuing Ph.D. degree from the Jain University, Bangalore, India, from 2017; currently she is working as Assistant Professor in PES University Bangalore, India. Her current research interests include Internet of Things, Cryptography, and IP security. Rajashree has coauthored 4 papers in peer reviewed conferences.

R. Sukumar completed his B.E. in ECE from Madurai Kamaraj University in 1992, M.E. in CSE from Manonmanium Sundaranar University in 2004, and Ph.D. from Anna University in 2010. He has over 20 years of teaching experience from undergraduate to Ph. D level. He is also a recognized PhD supervisor of Anna University Chennai and Jain University Bangalore. Already, three candidates have completed Ph.D under his supervision and six scholars are pursuing Ph.D. He is also serving as member of Board of Studies and various committees. He has over 25 high impact factor international journal publications and 20 international conference publications. He is currently serving as Professor at School of Engineering & Technology, Jain University, JGI Global Campus, Bangalore, India. His research areas of interest include Cryptography & Network Security, Sensor Networks, Cloud and IoT.