DLSU Research Congress 2022
De La Salle University, Manila, Philippines
July 6 to 8, 2022

A Confidentiality and Access Control Framework for Electronic Health Records in a Public Blockchain

Maria Patricia Javier[1], Earth Wendell Lopez[1], Gabriel Luis Marcelo[1], and Katrina Ysabel Solomon[2,*]
[1] *College of Computer Studies, De La Salle University*
[2] *Computer Technology Department, College of Computer Studies, De La Salle University*
**Corresponding Author: katrina.solomon@dlsu.edu.ph*

Abstract: The current state of Electronic Health Records (EHRs) is faced with both confidentiality breach and accessibility problems. EHR systems that offer good accessibility typically have mediocre security that can be vulnerable to malicious attacks that might lead to sensitive healthcare data being exposed and accessible to unauthorized users. On the other hand, while existing state-of-the-art security mechanisms for EHR systems protect the privacy and confidentiality of EHRs against malicious attacks, the drawback comes with the difficulty in accessing the records of patients as well as in the sharing of such information to each patients' respective healthcare providers and their fellow peers. Integrating blockchain technology to EHRs can improve both accessibility and security with the system. However, most public blockchain implementations still have their respective limitations especially when it comes to the preservation of privacy and confidentiality of medical data inside EHRs during data sharing. This study proposes a framework to address the said limitations by utilizing cryptography and access control functionalities to an existing public blockchain as a means to strengthen the privacy and confidentiality of healthcare data, specifically immunization records, prior to sharing. The framework aims to implement a more robust access control mechanism to limit access to EHRs solely to the authorized subjects by utilizing a smart contract stored within the blockchain. Additionally, confidentiality of EHRs is strengthened through the implementation of private-key cryptography among three known encryption algorithms namely, AES, Triple DES (3DES), and Blowfish. The framework is tested on a personal blockchain environment through the Ganache application.

Key Words: blockchain; confidentiality; access control; electronic health records; private-key cryptography

1. INTRODUCTION

Blockchain is a system that acts as a digital ledger that is able to record information in a distributed fashion without the need for a third party or a central authority. Blockchains have the unique property of being tamper-proof due to the way it is designed such as not being able to change the details of a transaction once it is published and giving the ability for users to agree when a transaction is valid by reaching a consensus (Yaga, Mell, Roby, & Scarfone, 2019). Blockchain grew in popularity together with the rose of the cryptocurrency, Bitcoin.

The blockchain technology is now being researched not just for cryptocurrency but towards other applications as well such as smart contracts, Internet of Things (IoT), and for medical purposes. Blockchain can have many use cases particularly in the field of healthcare; one such case is the secure sharing of medical data between the involved parties.

Medical data may include important information about an individual such as diagnoses, medications, allergies, and more. These data can be stored in an Electronic Health Record (EHR) in order to store it in a digital manner which can improve the efficiency of sharing data among a medical facility (Fusion Practice, 2021). However, EHRs do not come without negatives as it is implemented in a centralized manner and in order to protect the data, security towards EHRs have been strict to the point that the data is generally inaccessible to patients (Vora, et al., 2018). Systems that offer good accessibility to their records tend to not have great or strict security which can cause vulnerability to attacks that could lead to data breaches.

Integrating blockchain to EHRs has the ability to make the sharing of medical data accessible and secure (Shahnaz, Qamar, & Khalid, 2019). Blockchain-based EHRs are implemented in a decentralized manner which eliminates the aforementioned issues of traditional EHRs with the additional benefit of utilizing the strengths of the blockchain technology making sure that the security, accessibility, and integrity of data is upheld (Azaria, Ekblaw, Vieira, & Lippman, 2016). However, the current state of blockchain-based EHRs still has limitations on functionality when it comes to secure data sharing of EHRs. The limited functionality of blockchain-based EHRs can cause accidental or malicious access towards the health records that can cause a breach of confidentiality. Providing confidentiality and proper access control are among the most critical security improvements needed in this domain as identified in other research works such as those by Dubovitskaya, Xu, Ryu, Schumacher, & Wang (2017) and Xia, et al. (2017). Recently, Jung & Agulto (2021) proposed the use of Software-Defined Networking Controller (SDNC) to address the aforementioned issues, specifically to monitor and track the spread of COVID-19 through users' smartphones.

The framework is created to strengthen the privacy and confidentiality of healthcare data prior to sharing. It utilizes cryptographic libraries for its implementation of private-key cryptography with different algorithms, namely Advanced Encryption Standard (AES), Triple Data Encryption Algorithm (3DES), and Blowfish for the improvement of the confidentiality of the EHRs. A personal blockchain environment is created with the software Ganache in which the framework is tested upon. The smart contract in the blockchain are created with Remix, an Ethereum Integrated Development Environment (IDE) for the access control mechanisms of the framework.

2. METHODOLOGY

2.1 Framework Overview

The framework is adapted from the work of Vora, et al. (2018) only with data stored in the blockchain itself. Transactions are created through the blockchain in which data or information are stored using a smart contract. The implementation of the framework contains a smart contract that is used to access and interact with the blockchain, and is divided into different components, namely: Access Permission and Records Retrieval. The communication between these components is illustrated in Fig 1. The framework emphasizes confidentiality through private-key cryptography to be applied on the EHRs and access control through detailed permissions and the ability to invoke and revoke said permissions. The summary of the function of each module and submodule is listed in Table 1 and discussed in detail thereafter.

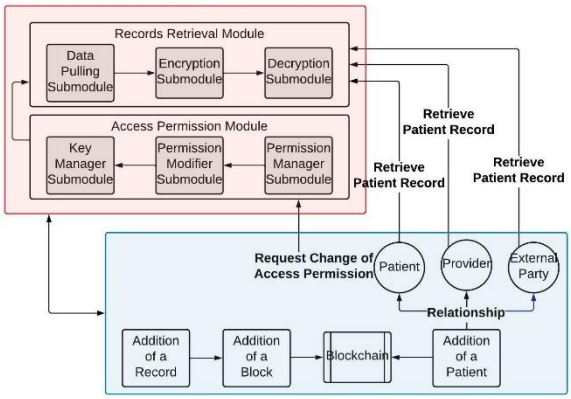

Fig 1. Architecture of the Proposed Framework

Table 1. Summary of the Functions of Each Module

Access Permission Module	
Permission Manager Submodule	Verifies that the requestor is the owner of the record being asked to modify the permissions via the smart contract

Permission Modifier Submodule	Validates the request to change access permission and uses the smart contract to apply the invocation/revocation change as well as save the recipient's address
Key Manager Submodule	Generates a key shared via the smart contract upon the update of an access permission to be used for both the encryption and decryption of a shared record when viewed by a recipient
Records Retrieval Module	
Data Pulling Submodule	Retrieves the requested record by either the owner/recipient from the blockchain
Encryption Submodule	Encrypts the retrieved record using the key generated upon the updated access permission and cryptographic algorithm set by the developer
Decryption Submodule	Decrypts the ciphered record using the provided key for the recipient for their access to the record and returns the record in plaintext

Invoke and revoke of permissions as well as key generation happen in the Access Permission Module. If the owner is the one retrieving the record, then the owner will be directly serviced by the Records Retrieval Module. On the other hand, if the owner intends to share a record, then the process starts with an owner's request for change of permission and generates the original access permission of the requested record. The original access permission will be updated by creating a new transaction for the given access permission. The process then proceeds to the generation of an encryption key.

Responsible for fetching the requested information by either the owner or recipient being shared to is the Records Retrieval Module. The primary functions of this component are pulling the data from the blockchain, encrypting the record according to the algorithm set, and decrypting the record for the recipient to receive, respectively. The module allows for both types of users to retrieve the record, varying mostly upon the need to securely provide a record for the intended recipient before completely accessing the said information. For the owner, a checking of Access Permissions for the record will first be verified where a confirmation will result in the retrieval of the record and the owner receiving it in plaintext.

For recipients given access permission by the owner, the process also begins with the receiving of the recipient's request, verifying access permission, and pulling the requested record from the blockchain. The pulled plaintext record is forwarded to the Encryption submodule where it is to be ciphered with the owner's key in accordance with the specified encryption algorithm before being sent to the recipient. The recipient then, with the shared key, will have to decipher the encrypted record through the Decryption submodule to receive the information again but in plaintext.

The implementation of the framework was done in a private test blockchain using the software Ganache (Truffle Suite, 2021). Ganache is an Ethereum simulator that can make development for Ethereum applications easier, faster, and safer. It can provide a local blockchain that allows for developers to develop, deploy, and test projects and smart contracts such as Solidity in a safe environment and without the need to pay for gas fees provided by Ethereum transactions making it cost-effective. The smart contract needed by the blockchain were written using the Solidity (Ethereum, 2015) programming language.

3. RESULTS AND DISCUSSION

3.1 Algorithm for Cryptographic Functions

The implemented framework allows for initialization of a private-key cryptographic algorithm to use among AES, 3DES, and Blowfish. As shown in Fig 2, the algorithm is chosen prior to the contract deployment which is represented in integer value 0 (AES), 1 (3DES), or 2 (Blowfish). The three algorithms were chosen among several available algorithms based on their performance as analyzed by Ghosh, Parmar, Shah, & Samdani (2018). Once the contract is deployed with the chosen algorithm, this will be applied for all needed cryptographic functions such as record encryption and decryption for record sharing.

AES, which is also known as the Rijndael algorithm is a symmetric block cipher that uses 128, 192, and 256-bit keys with a block length of 128 bits to encrypt data. It is selected by the U.S. National Institute of Standards and Technology (NIST) as the standard symmetric key encryption algorithm. 3DES is another block cipher that encrypts data with Data Encryption Algorithm (DES) three times with a different key each time. 3DES uses 56-bit keys instead of 64 because of the need to skip every eighth bit for parity check bits and 3DES uses three of these 56-bit

keys to form a single 168-bit key. Blowfish is also a symmetric encryption algorithm that uses a 64-bit block size and is considered much faster than DES. These algorithms are used and implemented using the Cryptography library in python, specifically *Crypto.Cipher* and *Crypto.Random*. The default algorithm set is the AES while users have the ability to change it into the two other algorithm options 3DES and Blowfish. All keys regardless of the algorithm chosen are created with 32 bytes length.

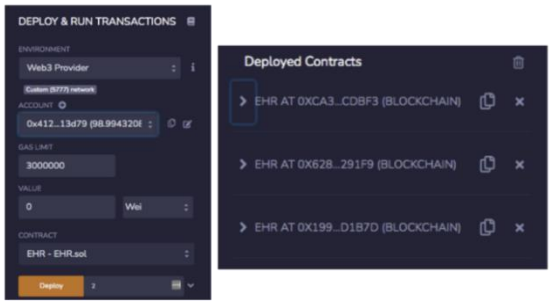

Fig 2. Encryption Algorithm Initialization

3.2 EHR Access Control

The invoking or revoking of permission of a certain record for a specific address can only be initiated by the owner of the record. "R" represents "Read Access" when access permission is invoked, and "N" represents "No Access" when access permission is revoked. Once permission is invoked, as shown in Fig 3, a key corresponding to the chosen algorithm is created which the recipient will need to open the shared record in readable format. If previously invoked permission of a shared record gets revoked, as shown in Fig 4, the key is then invalidated and removed from key storage which leaves the past receiver no way of converting the record in readable format.

```
Input recipient wallet: 0x41208047279fF544e4E5D77DfD125B81b1113d79
Input recordID: 1
Input access code (R or N): R

Access Permission Invoked Successfully.
Access Permission created:

Access ID: 6
Recipient: 0x41208047279fF544e4E5D77DfD125B81b1113d79
Record ID: 1
Updated Access Permission: R
AES Key: 0x5290bd7c6fd7ea971fa64d005b24826b0562f208772685e7ef4e0f8141b5c414
```

Fig 3. Invoking of Access Permission

```
Access Permission Revoked Successfully.
Access Permission created:

Access ID: 11
Recipient: 0x41208047279fF544e4E5D77DfD125B81b1113d79
Record ID: 1
Updated Access Permission: N
```

Fig 4. Revoking of Access Permission

3.3 EHR Encryption

Once a record is pulled from the blockchain, the access permission of the record retriever is checked as shown in Fig 5 represented by the Access Permission Value. If the retriever is the owner, then the record will directly be output in plaintext. If the retriever, however, is a recipient of the record, the record is first encrypted so that the recipient needs to provide a key first to be able to output the record in plaintext. Otherwise, the record is encrypted and is not readable as shown in Fig 5.

```
Input recordID to retrieve: 2

Access Permission Value: 2

Encrypted record: b'\xe9\xc7\xb4\x93_\x82\xa9De\xb6{}P~+k\xe5u\xb3y\x0b\xa
13\xd0\xa5w\xa7\xbd\x8b\x85\x89P~\x15\xd3ew\x99iK\x01\x1c\x9e\x85u\x89K\x1
```

Fig 5. Sample Encrypted EHR

3.4 EHR Decryption

Decryption of a record checks the access permission of the user accessing the record. An access permission of "O" indicates that the retriever of the record is its owner thus the record is outputted in plaintext once the record ID is provided as shown in Fig 6. If the retriever, on the other hand, is the recipient of a record, the key for decrypting the requested record is asked as input and only upon inputting the right and valid key will the record be outputted in plaintext as shown in Fig 7.

```
Input recordID to retrieve: 2

Access Permission Value: 1

Record ID: 2   Date: 2021-01-30
Patient ID: 0xbf74F2a2F739E91Edfdb7AA893c715A2839
Vaccine Code: 83    Description: Hep A  ped/adol
```

Fig 6. Decrypted Record Retrieval by Owner

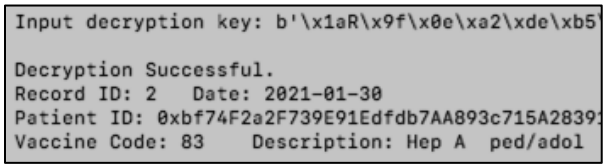

Fig 7. Decrypted Record Retrieval by Recipient

3.5 Key Management

With the utilization of cryptography for EHRs upon sharing, a functionality is created to regulate the storage of keys used for the encryption and decryption. As shown in Fig 8, each key generated is referenced via a key ID that is mainly used for queries to validate the legibility of the provided private key especially in the case of a third-party recipient that has been provided access to a certain EHR.

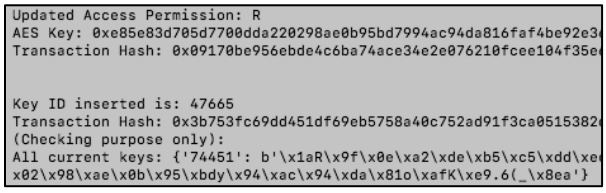

Fig 8. Cryptographic Key Management

4. CONCLUSIONS

The capability of the proposed framework was validated through the implementation on a private test blockchain. The test cases conducted proved that a more robust confidentiality and access control mechanisms can be used to further improve the security of handling EHRs. Allowing the use of various private-key cryptographic algorithms will enable future systems that will adapt this framework to be more flexible and tailored to their respective organizations' compliance and regulatory requirements. As for the access control functionality, the framework ensures that the EHRs will only be available to authorized parties aside from the owner of the record itself. That way, sensitive information found in EHRs will not be immediately disclosed to entities that should not have permission to access these records in the first place.

Since the framework uses public-key cryptography, specifically AES, 3DES, and Blowfish as cryptographic algorithm options for the user, we recommended to consider other cryptographic algorithms that provide security against emerging and more recently dated threats as well as implementing private-key cryptography. We also recommend evaluating different algorithms when choosing what to include as choices for implementing the cryptographic functions in terms of the security they can provide on the framework as an addition to the provided security of blockchain technology so that chosen algorithms become more fit to the system or application that the framework will be used for.

5. REFERENCES

Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using Blockchain for Medical Data Access and Permission Management. *2016 2nd International Conference on Open and Big Data (OBD)*, (pp. 25-30).

Dubovitskaya, A., Xu, Z., Ryu, S., Schumacher, M., & Wang, F. (2017). Secure and Trustable Electronic Medical Records Sharing using Blockchain. *AMIA ... Annual Symposium proceedings. AMIA Symposium*.

Ethereum. (2015). *Solidity*. Retrieved from Solidity Documentation: https://docs.soliditylang.org/en/v0.8.13/

Fusion Practice. (2021, May 21). *EHR (electronic health record) Definition*. Retrieved from Practice Fusion: https://www.practicefusion.com/blog/ehr-vs-emr/#EHRDefinition

Ghosh, S., Parmar, H., Shah, P., & Samdani, K. (2018). A Comprehensive Analysis Between Popular Symmetric Encryption Algorithms. *2018 IEEE Punecon*, (pp. 1-7).

Jung, Y., & Agulto, R. (2021). A Public Platform for Virtual IoT-Based Monitoring and Tracking of COVID-19. *Electronics, 10*(1). doi:https://doi.org/10.3390/electronics10010012

Shahnaz, A., Qamar, U., & Khalid, A. (2019). Using Blockchain for Electronic Health Records. *IEEE Access*, 147782-147795.

Truffle Suite. (2021). *Ganache*. Retrieved from Truffle Suite: https://trufflesuite.com/ganache/index.html

Vora, J., Nayyar, A., Tanwar, S., Tyagi, S., Kumar, N., Obaidat, M., & Rodrigues, J. (2018). BHEEM: A Blockchain-Based Framework for Securing Electronic Health Records. *2018 IEEE Globecom Workshops (GC Wkshps).* Abu Dhabi.

Xia, Q., Sifah, E., Asamoah, K., Gao, J., Du, X., & Guizani, M. (2017). MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain. *IEEE Access*, 14757-14767.

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2019). Blockchain technology overview. *arXiv preprint arXiv:1906.11078*.