HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

Designing a Unified Healthcare Data System with Blockchain and Cryptography

Rohan Phanse

Palo Alto High School, Palo Alto, CA

ABSTRACT

The majority of public and private healthcare data systems across the globe are proprietary, making it difficult to coordinate and share data between medical institutions on different systems. To improve the accessibility and shareability of patient data, we propose a Unified Healthcare Data System, built with blockchain and cryptography. Under this system, patients are in control of their data and can choose to share their data with doctors and institutions without traditional barriers. Blockchain is used to ensure data integrity, provide immutability, and allow data ownership while cryptography is used to encrypt patient data and implement permissioned access. In this paper, we provide an implementation built on the NEAR blockchain and expand on the viability, impact, and evolution of this blockchain-based healthcare data system.

Introduction

Healthcare data management and sharing is a challenge globally. In the United States, a technically and economically advanced nation, patients still face significant issues with healthcare data sharing, which contributes to inefficiencies in the healthcare system (Hulsen, 2020). Friction in data sharing is also prevalent in developing nations like India, where the lack of shareability of patient data leads to redundant tests and unnecessarily high healthcare costs (Dhagarra et al., 2019).

Furthermore, patients want more accessibility and security for their healthcare data. According to Pew Research's nationally representative survey on the US healthcare system in 2020, more than two-thirds of adults want their healthcare providers to exchange some health information that federal data sharing policies do not currently require, such as X-ray images or family medical histories (The Pew Charitable Trusts, 2021). The survey adds a majority of Americans want their healthcare data to be both more accessible and better protected.

Because current healthcare systems are built as proprietary systems for security purposes, the consequence is that they also prohibit useful data sharing that patients want.

Background

Patient Challenges

For patients, access to their healthcare data is limited by technology vendors, data management systems, and healthcare providers, impacting their ability to understand and make choices, and potentially, the quality of their care. Patients run into challenges trying to access their own healthcare data and have no visibility into where or how this data is stored. Their data could be stored as paper copies, digitally on a cloud-based server, on-premises of the care provider or via a hybrid solution spanning two or more of these.

Doctor Challenges

Doctors face challenges in the management of patient healthcare data, including lack of access to data from the patient's prior provider and incompatibility of data generated by various systems they use or interact with, along with concerns about privacy and liability issues related to patient data sharing. For example, 70% of Canadian physicians are burdened by the incompatibility of the data generated by the healthcare systems they interact with from other physicians due to lack of data homogeneity (Azarm-Daigle et al., 2015). This results in suboptimal patient care, billing issues, patient frustration and occasionally serious consequences. According to the statistics from the US Department of Health and Human Services and projections made by Kish & Topol, the lack of immediate access to needed health information leads to approximately 20% of preventable medical errors causing 80,000 deaths each year (Kish & Topol, 2015). Additionally, doctors cannot optimize treatment and lifestyle recommendations taking into account disparate sources of healthcare data, even if the patient is willing to share this information.

Healthcare Research

Researchers benefit from access to large datasets. However, the majority of healthcare data is locked away in proprietary systems. Healthcare and other institutions have been reluctant to share this data for research. In 2015, the US government provided $30 billion in incentives to increase data sharing initiatives. However, there was not much progress on data sharing, spurring the US Office of the National Coordinator for Health Information Technology (ONC) to publish a report on this issue of "information blocking," a term they coined for the data hoarding phenomenon (ONC, 2015).

Security

In addition to harboring inefficiencies, current healthcare data systems also have the potential for data abuse. In the current healthcare system, hospitals store and control all patient data. Patients must trust that their healthcare institution is keeping their information confidential and not abusing their data. Hospital data breaches can happen due to insecure storage of data or when the hospital assumes it is storing data securely but does not adhere to or keep up with modern best practices for managing advanced security. For example, in the first half of 2015, hackers breached approximately 100 million patient records (Kish & Topol, 2015). Patients benefit when systems use modern security techniques such as cryptography to securely store their data.

Data Ownership and Transparency

A patient may decide to change doctors and encounter a roadblock when they want their prior health history and data shared with a new doctor. The ONC highlighted in a 2015 Congressional report that patients' access to data is limited by technology vendors and healthcare providers (ONC, 2015). Patients may get frustrated because they realize they do not really own data about their own health and have not much say in related matters, except signing complicated agreements at every healthcare provider they visit.

Patients suffer from the negative effects of the lack of control over their own healthcare data. Older adults are often unable to provide access to their healthcare data and treatment information to their caregiver or children without complex legal paperwork or without ceding more control than they would like to (Crotty et al., 2015). This lack of visibility could be remedied by allowing patients to own their data and instead grant access to these intermediate parties.

Economic Factors

Under current systems, for data to be accessed in any care setting, the healthcare data provider must share the data of one of their patients to other hospitals and institutions who need that data. But Marchibroda points out that the absence of a business case for data sharing, associated costs, and lack of interoperability standards was the reason why there was no investment into building healthcare information exchanges (Marchibroda, 2014). Making data exchange more accessible and widespread is critical to improving patient outcomes, reducing their healthcare costs, and spurring beneficial medical research with life-changing applications. Despite these positives, businesses are economically disincentivized to share data.

Decentralization

The emergence of blockchain, cryptography, and the new field of Web3 have made previously unattainable goals of simultaneous shareability and security possible. Blockchain keeps a digital record of transactions and ensures the immutability of data, which means data cannot be altered or deleted. A blockchain is not owned by any one organization and is decentralized. Web3 refers to an evolution of the current Web technologies built on concepts like decentralization, blockchain, and tokenization.

These technologies allow user identity and data sharing to be secure and managed in a decentralized manner, which has been the biggest challenge in providing patients access to their own data. Decentralized systems differ from centralized platforms in that the systems, network, storage, identity, and other elements are run on multiple nodes by multiple stakeholders, thereby avoiding a single point of failure. With this architecture, decentralized technologies promise to give users ownership over their own data and promote security and privacy. The first application in this space was Bitcoin, a cryptocurrency backed by the blockchain proposed by an individual with the pseudonym Satoshi Nakamoto, which allowed user A and user B to exchange currency without a centralized entity (for example, a bank) either directly or indirectly involved in the transaction, with similar or better levels of security and immutability of transactions as traditional banks (Nakamoto, 2008).

Since then, blockchains have become more general-purpose with the advent of what is known as smart contracts, which allow decentralized apps to be built on the blockchain. Smart contracts are programs on the blockchain, which automatically run when predefined contract conditions are met. Smart contracts have applications in data ownership, and finance, and have tremendous potential to revolutionize a host of real-world industries.

Cryptography

Cryptography is a technique used to confidentially share information (Qadir & Varol, 2019). The goal of cryptography is to allow the contents of a message to be read by the intended recipients, while preventing unauthorized parties from reading the message. In healthcare, cryptography can be used to achieve compliance with the Healthcare Insurance Portability & Accountability Act (HIPAA). HIPAA states that the health information of individuals must be protected against unauthorized access (Nass et al., 2009).

To limit access to healthcare data to authorized parties, symmetric key cryptography can be used. In symmetric key cryptography, one key is used to both encrypt and decrypt messages. A widely used method for symmetric key cryptography is the Advanced Encryption Standard (AES) algorithm, proposed by the National Institute of Standards (Nechvatal et al., 2001). Using an AES key, healthcare data can be securely encrypted, and its access can be granted to authorized individuals by sharing the key.

While this method is secure and works for data of any size, symmetric keys have an important caveat–they must be shared in advance with all intended recipients. If the keys are shared over insecure channels, they may become

compromised. To avoid this dilemma, asymmetric key cryptography can be used to encrypt the symmetric keys, making them safe to send over public channels.

In asymmetric key cryptography, two keys, often designated as public and private, have the property that a message encrypted with one key can be decrypted by the other. The Rivest, Shamir, and Adleman (RSA) algorithm is commonly used to implement asymmetric key cryptography.

The goal of this research is to outline gaps in health care data sharing systems today, propose a solution to address these challenges using blockchain and cryptography, build a working prototype, test and validate the results, and identify limitations and potential evolution.

Methods

To address the challenges in moving the ownership of healthcare data to patients and sharing patient data seamlessly and securely, we propose a Unified Healthcare Data System. In this system, patients will have flexibility in choosing hospitals and providers that will have access to their healthcare data and avoid unnecessary healthcare costs. Since patients own their data, they can make the decision to share it with researchers, which can help grow research datasets and help advance medical science, thereby benefiting patients.

To provide immutability of data and a transparent record of all changes to the network, the Unified Healthcare Data System uses blockchain and cryptography to address the key technical challenges faced by existing healthcare data systems. Blockchain intrinsically provides data ownership and cryptography and provides data security. With these two technologies, this system can enable patients and the doctors and hospitals they allow to read their data. In addition to specifying viewers, patients can add contributors, who are allowed to upload data to the patient's account.

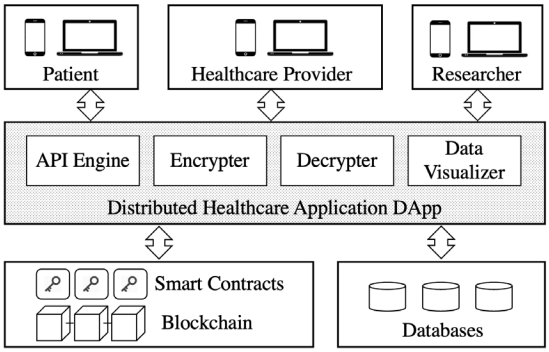

Figure 1. Unified Healthcare Data System Architecture.

We developed the Unified Healthcare Data System as shown in Figure (1). The three types of users, patients, doctors, and researchers, interact with the distributed app (DApp), which runs on many nodes on the decentralized network, and sends and receives data between the blockchain smart contracts and the databases. Patients can add viewers (read data) and contributors (upload data) to their account and can share their data with any medical or research institution on the Unified Healthcare Data System. Doctors and medical institutions, who upload data to their patient's accounts, can manage the healthcare data access for all their patients. Research institutions have dedicated pathways to receive anonymized verified healthcare data from patients. The system is built with three key modules:

1) Smart contracts hosted on the blockchain. The blockchain stores public account data. Smart contract methods handle account creation (storing public keys), setting viewers and contributors, and storing data hashes to expose tampering. This smart contract system was built using Rust on the NEAR blockchain but can also be built with Solidity on the Ethereum blockchain or other smart contract and blockchain systems.

2) Databases for storing encrypted patient health record data on decentralized cloud storage like third party cloud storage or the Interplanetary File System (IPFS). The pointer to the patient's health record consisting of the content IDs for the health record and the hash of the health record is stored on the blockchain.

3) Mobile and Web applications for patients, doctors, and researchers. The application runs the smart contract methods and encrypts and decrypts data on the client side.

Algorithm 1: Add Public Key to Blockchain

```
Data: public key
Result: Public key is added for a new account;
        multiple keys are prevented
if signer does not have a contract account then
    Register contract account for signer
    Add public key to contract account
else
    Report error "Public key already added"
end
```

Algorithm 1. Smart Contract Method for the Addition of a Public Key to the Blockchain.

In asymmetric key cryptography, public keys are shared with other members on the network, while private keys are held locally and confidentially. We create a public directory stored on the blockchain to associate each account with its public key, as shown in Algorithm (1). If the signer, the individual calling this smart contract method, does not have a contract account, one is registered for them, and their public key is added to their account. Users who have already registered an account are prevented from adding multiple private keys to a single account.

Algorithm 2: Validate and Upload Heath Record to Blockchain

```
Data: account ID, data ID, encrypted record
Result: Valid data requests are uploaded to
        blockchain
if signer ID matches account ID or account's
 contributors contain signer then
    if data ID used in records then
        Report error "Data ID already used"
    end
    if data ID length too small or large then
        Report error "Invalid ID length"
    end
    Add encrypted record to contract's records
    Add data ID to contract's list of data IDs
else
    Report error "Unallowed contributor"
end
```

Algorithm 2. Smart Contract Method for the Validation and Upload of a Health Record to the Blockchain.

Additionally, we write smart contract logic for validating and uploading encrypted health records to the blockchain, as shown in Algorithm (2). If the signer is uploading data to their account, or to an account they have been

authorized to contribute to, the smart contract validates their upload permissions. The reason why users must generate random data IDs on the client-side and attach them to their smart contract request is because of the deterministic nature of smart contracts (Alharby et al., 2018). Smart contract methods must produce the same output for any node they are run on, so random values cannot be produced on the blockchain and must be generated on the client.

Algorithm 3: Fetch Record from Blockchain, Decrypt on Client Side

```
Data: data ID, private key
Result: Data is decrypted if user has access
        permission
Fetch user's data IDs from blockchain.
if user's data IDs contain data ID then
    Fetch encrypted symmetric key, encrypted
     data at ID from blockchain.
    Use private key to decrypt symmetric key.
    if decryption fails then
        Report error "Incorrect private key"
    end
    Use symmetric key to decrypt data.
    Display data record on UI.
else
    Report error "No record at given ID found"
end
```

Algorithm 3. Client-Side Method for the Retrieval and Decryption of Encrypted Data from the Blockchain

Because encrypted data is stored on the blockchain, users must decrypt their fetched data, as shown in Algorithm (3). First, the user's list of data IDs, corresponding to the records the user is authorized to access, are checked against the data ID the user wishes to decrypt. If the user's permitted list contains the data ID, the encrypted symmetric key and encrypted data stored at that ID are fetched from the blockchain. Using their private key, the user decrypts the symmetric key, and uses the symmetric key to decrypt their health record. The decrypted record is then displayed on the user interface (UI) for the user.

Implementation

The smart contract implementation shown in the provided GitHub repository includes the core functionality of storing patient data and encrypting data with cryptography (Phanse, 2022). The implementation is written in Rust and JavaScript for the NEAR blockchain, which is a sharded, proof-of-work, layer-one blockchain (NEAR Protocol, n.d.). We use NEAR because of its developer-friendly smart contract development toolkit, testnet (allowing for free smart contract testing), and transaction viewer which displays transaction costs for each smart contract method. The app was built with React.js. This implementation was also possible using Solidity on the Ethereum blockchain or other smart contract and blockchain systems.

```
// Client-side encryption code in JavaScript:

async function uploadEncryptedData() {
// Global variables: toAccount, data
// Fetch intended recipient's public key from
     blockchain
const receiverPublicKey = await
    importPublicKey(JSON.parse(await
    contract.getAccountPublicKey(toAccount)))
// Encrypt data with symmetric key
const initialVector = window.crypto.
    getRandomValues(new Uint8Array(12))
const symmetricKey = await createSymmetricKey
    ()
const keyString = await exportSymmetricKey(
    symmetricKey)
const encryptedData = await
    symmetricKeyEncrypt(symmetricKey,
    initialVector, data)
// Export encrypted buffer as hexadecimal
    string and serialize data as JSON
// Encrypt symmetric key with public key
const encryptedSymmetricKey = await
    publicKeyEncrypt(receiverPublicKey, JSON.
    stringify({ key: keyString, iv:
    bufferToHex(initialVector) }))
// Upload encrypted data and encrypted
    symmetric key to blockchain
contract.uploadData(toAccount,
    encryptedSymmetricKey, encryptedData)
}
```

Listing 1: Client-side data encryption using AES-256 key and RSA-4096 key. Full code at: https://github.com/rohanphanse/healthcare-data-system (Phanse, 2022).

As shown in Listing (1), the smart contract only allows the authenticated patient and approved healthcare provider contributors to upload healthcare records to the patient's account. In this method, the client fetches the recipient's 4096-bit RSA public key from the blockchain, generates a 256-bit AES symmetric key and uses it to encrypt the data to be uploaded, encrypts the symmetric key with the recipient's public key, and uploads the encrypted healthcare data to the blockchain. The result is that only the recipient can decrypt the symmetric key with their private key and thus decrypt and access the healthcare record.

To implement these cryptographic protocols, the Web Crypto JavaScript library is used to generate both symmetric and asymmetric key pairs, use them to encrypt and decrypt messages, export them to the JSON Web Token (JWK) format and as hexadecimal strings, and generate random values to create the seed for the symmetric key, or the initial vector (MDN Web Docs, n.d.). The code calls a method from the contract variable, which is an API that calls smart contract methods on the NEAR blockchain.

```
//Smart contract authentication and upload
    code in Rust:

pub fn upload_data(&mut self, account_id:
    AccountId, data_id: DataId,
    encrypted_symmetric_key: String,
    encrypted_data: String) {
let signer_id = env::signer_account_id();
// Only the user and allowed contributors can
     upload data to the user's account
if signer_id != account_id {
    if let Some(contributors) = self.
      contributors_map.get(&account_id) {
        if !contributors.contains(&signer_id)
     {
            panic!("Unallowed contributor");
        }
    } else {
        panic!("Unallowed contributor");
    }
}
// Validate data ID
if let Some(_) = self.storage_map.get(&
    data_id) {
    panic!("Data ID already used")
}
if data_id.len() > 20 || data_id.len() < 1 {
    panic!("Data ID has invalid length");
}
let data_entry = DataEntry::new(signer_id,
    encrypted_symmetric_key, encrypted_data);
// Upload encrypted record to blockchain
self.storage_map.insert(&data_id, &data_entry
    );
// Add new ID to user's list of data IDs
if self.data_ids_map.get(&account_id) == None
     {
    let set_id: Vec<u8> = format!("d-{}",
    account_id).as_bytes().to_vec();
    self.data_ids_map.insert(&account_id, &
    UnorderedSet::new(set_id));
}
let mut data_ids = self.data_ids_map.get(&
    account_id).unwrap();
data_ids.insert(&data_id);
self.data_ids_map.insert(&account_id, &
    data_ids);
}
```

Listing 2: Smart contract authentication method for user data during upload onto the Unified Healthcare Data System. Full code at: https://github.com/rohanphanse/healthcare-data-system (Phanse, 2022).

Once the health record is encrypted, the uploading process is handled by the Rust smart contract method shown in Listing (2), which is an implementation of Algorithm (2). The signer, the individual calling this method, is first checked to see if their ID is the same as the account they are attempting to upload to, or if it is included in the list of that account's authorized contributors. After the signer's upload permissions are validated, the data ID they are requesting to upload their record at is checked for uniqueness and length, and the encrypted record is uploaded to the NEAR blockchain. To implement persistent storage on the blockchain, the data structures UnorderedMap and UnorderedSet

from NEAR's Rust Software Development Kit (SDK) are used (NEAR Documentation, n.d.). They offer O(1) constant time complexity for lookup, insertion, and deletion, which enables the smart contract execution to be efficient and cost less Gas, or execution fees.

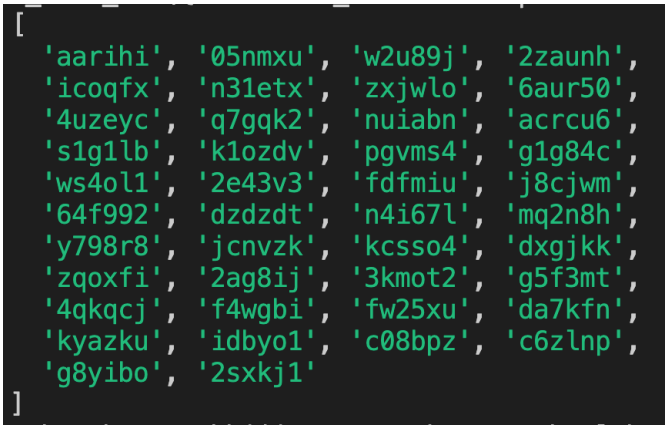

Figure 2: Terminal output of a user's data IDs.

Each patient user will have a data ID for every health record uploaded to their healthcare account. A user will have hundreds of data IDs over their lifetime corresponding to their set of health records. An example of a user's list of data IDs is shown in Figure (2). The terminal output is the result of calling the get_account_data_ids smart contract method using NEAR's Command Line Interface (CLI). A user can call this method on the client to retrieve their data IDs from the blockchain, and request the encrypted records stored at a specific ID. The example IDs shown are composed of six random alphanumeric characters and generated with JavaScript utilities.

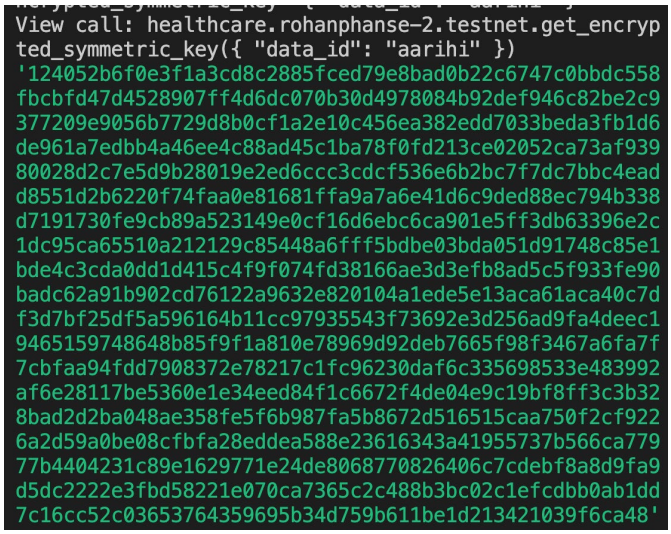

Figure 3: Terminal output of an encrypted symmetric key.

In Figure (3), an example encrypted symmetric key is displayed in the terminal. This text is always 1024-bytes long, because the 4096-bit RSA key encrypts the AES key as a 4096-bit or 512-byte buffer, which is encoded

as a 1024-byte hexadecimal string. The encrypted symmetric key maintains this constant size regardless of the size of the encrypted health record.

```
View call: healthcare.rohanphanse-2.testnet.get_encryp
ted_data({ "data_id": "aarihi" })
'f4d153ed4ae60be0c36a29521890e296'
```

Figure 4: Terminal output of encrypted data stored on the blockchain; for example, an encrypted pointer to a health record.

Finally, an example of encrypted data stored on the blockchain is displayed in the terminal as shown in Figure (4). Using a 256-bit AES key, the size of encrypted data (in bytes) can be calculated by the following formula: $2 * (16 + size)$. The addition of 16 bytes is a result of symmetric key encryption, and the coefficient of 2 results from converting the 256-bit numbers into hexadecimal (base 16) strings, doubling the length.

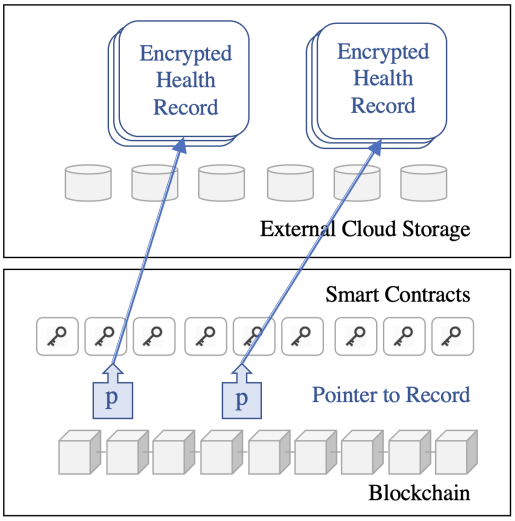

Figure 5: The encrypted health data record is stored on external storage, while the pointer to the health data record is stored on the blockchain.

In the Unified Healthcare Data System, the content IDs and the SHA-256 hash for the health record data constitute the pointer to the encrypted health record and are uploaded onto the blockchain, while the encrypted health record, which is referenced by the content ID, is stored on a long-term low-cost distributed cloud storage system or a decentralized IPFS storage system, as shown in Figure (5). For example, for maintaining three copies of a health record on cloud storage, each with an IPFS content ID of 32 bytes, and a 16-byte hash of this record, we need 108 bytes to create an immutable pointer to the health record to be stored on the blockchain, which enables us to store, manage, and authenticate this health data record. Over a patient's lifetime, even a thousand health records would only need about 100 KB of health record pointer data to be stored on the blockchain.

Evaluation

We tested and successfully validated the Unified Healthcare Data System implementation by running simulated patient, doctor, and researcher interactions and data uploads to this system. We used test data instead of real patient data to avoid privacy issues during the testing phase. We evaluated the cost of smart contract transactions on the blockchain.

Table 1: Measurement of Smart Contract Transaction Costs on the NEAR Blockchain.

Smart Contract Transaction Costs		
Method Name	Cost (NEAR)	Cost (USD)
deploy_contract	0.00136	0.00212
add_account_info	0.00059	0.00092
delete_account	0.00063	0.00098
add_contributor	0.00066	0.00103
remove_contributor	0.00064	0.00100
upload_data	0.00072	0.00112
remove_data	0.00070	0.00109

The transaction costs of each method in the smart contract implementation are presented in Table (1). The data was collected by calling the methods multiple times, recording their transaction cost using the NEAR Testnet Explorer, and averaging the results (NEAR Explorer, n.d.). Deploying the contract, which is 178 KB large, costs $0.00212. The costs of calling the six other smart contract methods range between $0.00092 and $0.00112 for typical data input sizes. These prices demonstrate the low cost of the system when executing the smart contract methods.

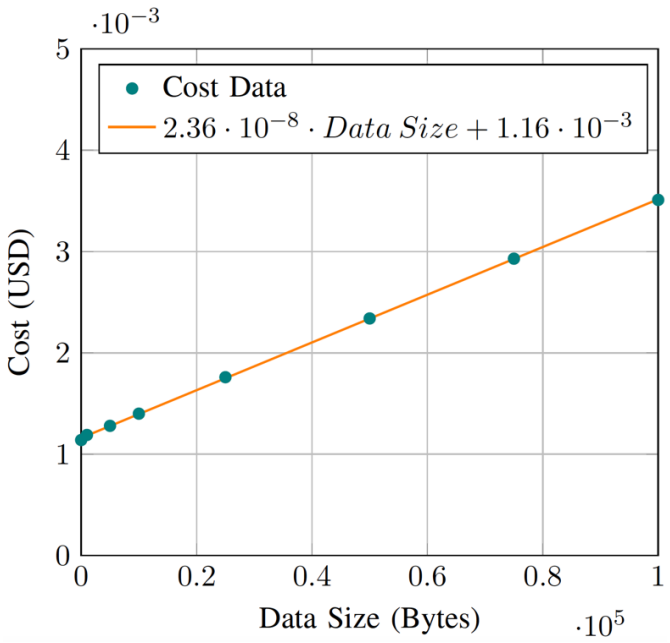

Figure 6: Total cost of calling the upload_data smart contract method to store encrypted record and pointer data on the blockchain for the Unified Healthcare Data System: Total Cost versus Data Size (Bytes).

The results from Figure (6) show that the transaction costs for the upload_data smart contract method scale linearly for data sizes between 0 and 100 KB along the trendline $y = 2.36 \cdot 10^{-8} \cdot \ size \ + \ 1.16 \cdot 10^{-3}$ with $R^2 = 1$. This data was collected using NEAR's transaction viewer on their testnet website, which is a test blockchain network with the capabilities of a real blockchain.

Over a patient's lifetime, if about 100KB of health record pointer data for about a thousand health records are stored on the blockchain, the cost for uploading and storing this data on the blockchain is about $0.0035. If a thousand uploads of 108 bytes are made, the total cost will be $1.16. The overall cost of uploading the lifetime patient health record pointer data to the blockchain is small.

By maintaining the health record data off the blockchain, the costs of implementing the shareability and security of the health records using the blockchain have been kept minimal and this design presents a scalable solution for building a universal and global healthcare data system. This system has been optimized for enabling shareability and security, while maintaining a low-cost implementation.

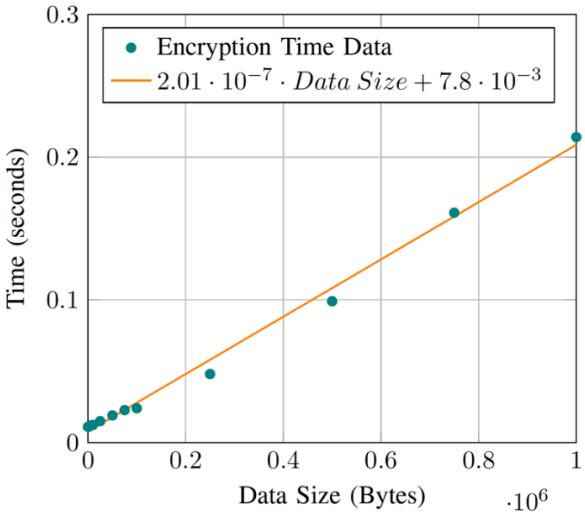

Figure 7: Client-Side Healthcare Data Encryption Time vs. Data Size.

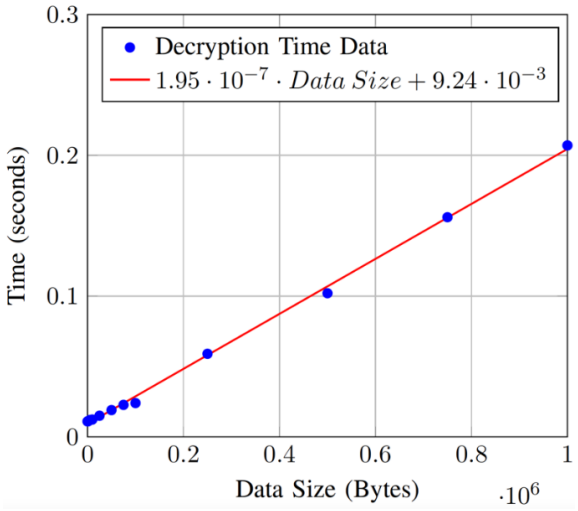

Figure 8: Client-Side Healthcare Data Decryption Time vs. Data Size.

We test the encryption and decryption algorithms for data sizes ranging from 0 to 1 MB. The Encryption Time vs. Data Size graph in Figure (7) has a linear trendline of $2.01 \cdot 10^{-7} \cdot Size + 7.8 \cdot 10^{-3}$ with $R^2 = 0.995$. The Decryption Time vs. Data Size graph in Figure (8) has a linear trendline of $1.95 \cdot 10^{-7} \cdot Size + 9.24 \cdot 10^{-3}$ with $R^2 = 0.999$. These results are consistent with the O(n) time complexity of AES and RSA cryptography algorithms. The results were collected by locally running the JavaScript algorithms on our testing device: a 2015 Macbook 2.2GHz Quad-Core Intel Core i7 with 16GB 1600MHz DDR3 RAM. The times to encrypt and decrypt the healthcare records are shown to be reasonable, presenting a practical and usable client-side solution for the Unified Healthcare Data System.

Conclusion

The proposed Unified Healthcare Data System provides benefits to patients, providers, and research institutions, to enhance shareability and security, while reducing costs in developed countries like the US, and has the potential to create a generational leaping solution to significantly enhance healthcare access for developing countries like India. This system's decentralized implementation and low-cost economics have shown it is capable of scaling to a global level.

Acknowledgements

I would like to thank my research advisor, Jon Stingel, Stanford University, for his mentorship, guidance, and feedback throughout the process of building this system and writing this research paper, and Ms. Roxanne Lanzot, Palo Alto High School, for being an inspiration to me.

References

Alharby, M., Aldweesh, A., & Moorsel, A. van. (2018). Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research . *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*. https://doi.org/10.1109/iccbb.2018.8756390

Azarm-Daigle, M., Kuziemsky, C., & Peyton, L. (2015). A Review of Cross Organizational Healthcare Data Sharing. *Procedia Computer Science*, *63*, 425–432. https://doi.org/10.1016/j.procs.2015.08.363

Crotty, B. H., Walker, J., Dierks, M., Lipsitz, L., O'Brien, J., Fischer, S., Slack, W. V., & Safran, C. (2015). Information Sharing Preferences of Older Patients and Their Families. *JAMA Internal Medicine*, *175*(9), 1492. https://doi.org/10.1001/jamainternmed.2015.2903

Dhagarra, D., Goswami, M., Sarma, P. R. S., & Choudhury, A. (2019). Big Data and blockchain supported conceptual model for enhanced healthcare coverage. *Business Process Management Journal*. https://doi.org/10.1108/bpmj-06-2018-0164

Hulsen, T. (2020). Sharing Is Caring—Data Sharing Initiatives in Healthcare. *International Journal of Environmental Research and Public Health*, *17*(9), 3046. https://doi.org/10.3390/ijerph17093046

Kish, L. J., & Topol, E. J. (2015). Unpatients—why patients should own their medical data. *Nature Biotechnology*, *33*(9), 921–924. https://doi.org/10.1038/nbt.3340

Marchibroda, J. (2014). *Interoperability*. Health Affairs. Retrieved from https://www.healthaffairs.org/do/10.1377/hpb20140811.761828/

MDN Web Docs. (n.d.). *Web Crypto API*. Mozilla. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

Nakamoto, S. (2018). *Bitcoin: A Peer-to-Peer Electronic Cash System.* Bitcoin. Retrieved from https://bitcoin.org/bitcoin.pdf

Nass, S. J., Levit, L. A., & Gostin, L. O. (2009). Beyond the HIPAA Privacy Rule: Enhancing Privacy, Improving Health Through Research. *The National Academies Press*. https://doi.org/10.17226/12458

NEAR Documentation. (n.d.). *Data Storage / Collections*. NEAR. Retrieved from https://docs.near.org/concepts/storage/data-storage

NEAR Explorer. (n.d.). *Explore the NEAR Blockchain*. NEAR. Retrieved from https://explorer.testnet.near.org/

NEAR Protocol. (n.d.). *Learn about NEAR Protocol*. NEAR. Retrieved from https://near.org/about/

Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (2001). Report on the Development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology*, *106*(3), 511. https://doi.org/10.6028/jres.106.023

The Office of the National Coordinator for Health Information Technology (ONC). (2015). *Report to Congress - Report on Health Information Blocking*. Department of Health and Human Services. Retrieved from https://www.healthit.gov/sites/default/files/reports/info_blocking_040915.pdf

The Pew Charitable Trusts. (2021). *Most Americans Want to Share and Access More Digital Health Data*. The Pew Charitable Trusts. Retrieved from https://www.pewtrusts.org/en/research-and-analysis/issue-briefs/2021/07/most-americans-want-to-share-and-access-more-digital-health-data

Phanse, R. (2022). *Healthcare Data System Code Repository*. GitHub. Retrieved from https://github.com/rohanphanse/healthcare-data-system

Qadir, A. M., & Varol, N. (2019). A Review Paper on Cryptography. *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*. https://doi.org/10.1109/isdfs.2019.8757514