Privacy-First Infrastructure Design for Hospital Record Management using GDPR Compliant Blockchain

Ankita Sharma
Brown University

James Rolfe
Brown University

Abstract

The blockchain datastructure has caught the attention of many. By design, it is resistant to modification, tamper proof, and transparent. It uses decentralized consensus to maintain an append-only ledger that everyone can see. While the blockchain has made great strides integrating into our lives and enabling the existence of cryptocurrency, its fundamental properties may prevent its usage with policies like the General Data Protection Regulation coming into affect that mandate that data stored by the data processor should be subject to many requirements such as the 'right to erasure' in which data must be deleted. In this paper, we present a GDPR compliant infrastructure design for record management in which the primary data stores are blockchains. We chose to apply this design to create a generic hospital record managing application because the hospital industry is often the victim of data breaches. Our application and resulting demo is fully GDPR compliant and publicly verifiable (via a public blockchain) while maintaining reasonable privacy for its data subjects.

1 Introduction

In order for a blockchain to accommodate requests like the 'right to erasure', private data cannot be stored on a blockchain since deleting data confirmed on a blockchain breaks the fundamentals of a blockchain. Instead, we store this data, encrypted, some where off the chain and use public blocks on the blockchain as a pointer to tell us where to find the data. In other words, blocks on the public chain would contain anonymous metadata associated with the transaction and contain a link to where encrypted user data is stored. This link could be thought of as a pointer or key in a database. By storing this data off the chain, we retain the fundamental properties of the blockchain by never running into a situation where a block needs to be removed or the entire blockchain reconstructed. However, to further extend this idea and to apply the blockchain's tamper proof properties to the data, which is desirable, the user data that is stored off the chain can also be stored in a separate, private blockchain. If the user wishes to remove their data (right to erasure) the entire private blockchain can be deleted and the pointer in the database would point to nothing. This multi-dimensional design satisfies blockchain properties at the meta and granular per user level. If a user removes their data the main blockchain can continue to operate and function as if nothing happened.

We decided to apply these idea to the healthcare sector. The motivation for this decision arises from our own observations and experiences. The healthcare field often lags behind in its technical readiness; as an example, the use of paper medical records and fax as a means for transferring medical records is still prevalent. Furthermore, multiple hospitals have been fined for GDPR non-compliance; lacking the infrastructure to prove and guarantee that secure medical information is not disclosed arbitrarily. The penalties these hospitals face are non-trivial. Just last year (2018), the central hospital of Barreiro Montijo in Portugal was fined 400,000 euros for being unable to prove GDPR compliance [1]. For these reasons, we modeled the construction of a GDPR compliant blockchain for organizations like hospitals that store personal records.

We wanted our proof-of-concept to target the following potential problems that pertain to the current healthcare ecosystem:

1. Unrestricted access to patient data
2. Malpractice through patient record tampering

[1] https://iapp.org/news/a/first-gdpr-fine-in-portugal-issued-against-hospital-for-three-violations/

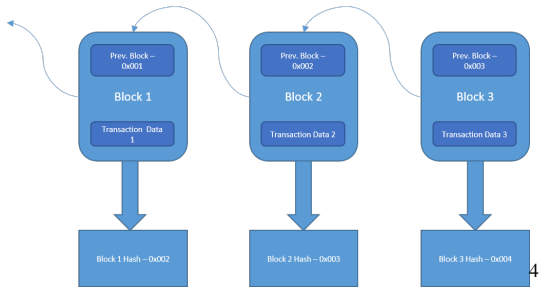
3. Patient cheating *e.g.* getting Opiates prescribed in 10 different hospitals

4. Slow and insecure removal and transfer of medical records

Our implementation does not make use of a full fledged network of distributed nodes. However, our implementation can easily be extended to using legitimate blockchain implementations like Bitcoin[2] or the XRP ledger.[3]

2 Background

A blockchain is a network of blocks containing all the transaction history. Each block in the blockchain contains an index, timestamp, hash of the previous block, transaction data, and a nonce for proof of work calculations. The hash of each block is based on the hash of the previous block as well as the data of the current block.



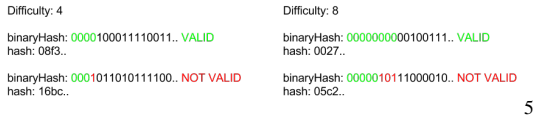
4

A new transaction is first added to a list of unconfirmed transactions. Mining is the process of adding new blocks to the blockchain by confirming transactions. Mining produces a proof of work which is used to confirm each transaction and check that a node has performed the necessary calculations to confirm a block by brute forcing the solution to a particular output of an encrypted hash function. To perform proof of work, a hash must be produced for the given block. Then the hash is verified by checking that the hash produced using the current nonce of the block has the correct amount of leading zeroes. The nonce field in the block can be adjusted in order to generate the desired hash and the amount of leading zeroes can be configured to customize the difficulty of mining (more zeros leads to higher difficulty). All of the other fields in the block must stay the same.

[2]https://bitcoin.org/en/

[3]https://xrpl.org/

[4]https://medium.com/@venkinarayanan/how-blockchain-works-b0a62ca2fca1



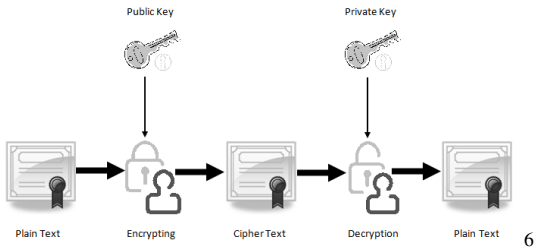
5

For each degree of difficulty, the average time to perform proof of work doubles. The difficulty value is updated to ensure that blocks are added to the blockchain at consistent intervals. In other words, it would take less time between the addition of new blocks as new nodes join the network if the difficulty remained the same. The difficulty value reflects the amount of computing power in the network and so as more nodes enter the network, the difficulty should increase.

The data stored on the blockchain is tamper-proof due to the use of encryption and digital signatures because if a block (other than the last block) was modified then the previous hash field of the next block would be incorrect. The blockchain is ideal for scenarios that entail mutually untrusted parties to collaborate without relying on a central authority.

3 Design

Standard asymmetric encryption serves as the basis of our design. Each user in the system is associated with a unique public and private key pair.



6

The patient information interacting entities include the public blockchain, a patient card, and the hospital database. The public blockchain is responsible for storing unique patient registrations in our ecosystem. When a patient registers with any hospital in our ecosystem for the first time, this registration is recorded on the public blockchain. Registration entails the generation of a unique public and private key pair for the patient. The public key is stored as a transaction on the public blockchain and the private key is returned to the patient in the form of a patient card. The patient card can be thought of as storing capabilities or secure tokens

[5]https://lhartikk.github.io/jekyll/update/2017/07/13/chapter2.html

[6]https://www.ssl2buy.com/wiki/what-is-a-public-and-private-key-pair

that grant access to a patient's medical records. Each hospital has their own database which stores a patient's encrypted medical records.

Here is a break down of these components:

Patient Information Interacting Entities

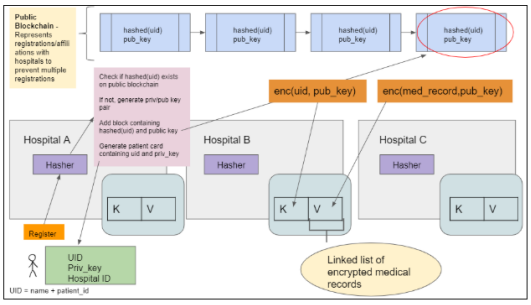
- Public Blockchain
  - Hospital hash(UID)
  - Patient pub_key
- Card
  - UID
  - Priv_key
  - Hospital_ID
- Hospital DB (private blockchain)
  - K - Encrypted(UID, pub_key)
  - V - Encrypted(linked list of medical records, pub_key)

Our ecosystem contains physicians, hospitals, and patients with the following properties:

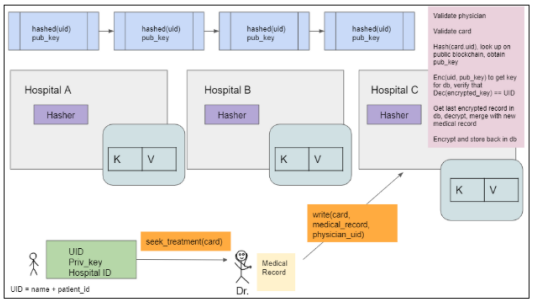
Ecosystem

- Physician
  - Name
  - Physician_id
  - Hospitals
- Patient
  - Name
  - Patient_id
  - Card
- Hospital
  - DB
  - Registered Physicians
  - Blockchain

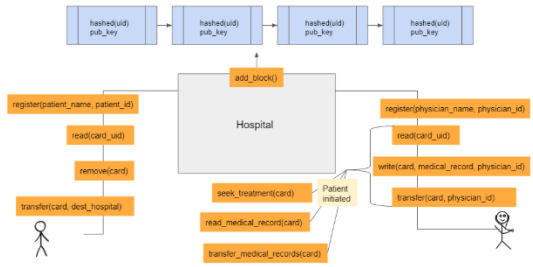
Each hospital in our ecosystem interacts with the public blockchain and each hospital stores a patient's encrypted medical records. The hospital that stores a patient's encrypted medical records corresponds with the hospital listed on the patient's card. Each hospital validates the patient's card when processing a request. If a patient wishes to transfer their medical records to another hospital, that field of their card is updated so that they can continue seeking treatment seamlessly. Below lays out how a patient would register with a hospital in our implementation. Note that each hospital employs a special hash function to disassociate the id on the card with the information stored on the blockchain to guarantee patient privacy by making the public metadata stored on the blockchain unidentifiable.



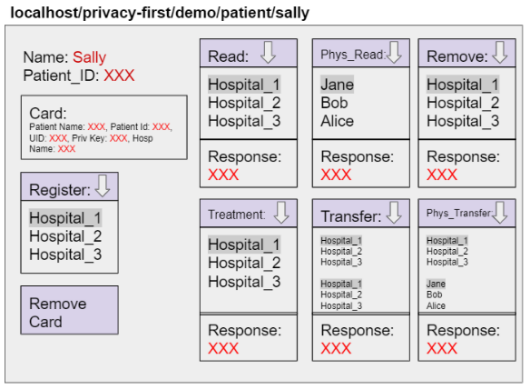
If a patient would like to seek treatment, a patient would first need to present their card to a physician. The card enables the physician to make a successful write request to the hospital server. The hospital server employs the unique hash function on the id listed on the patient card to look up the public key for that user. The public key allows the hospital to find the key where patient information is stored in its database; the key being the encrypted patient_id. The hospital can verify that the card has not been tampered with by also decrypting the encrypted key with the private key listed on the card; if this does not equal the patient_id, the request is rejected. A successful write request will store the encrypted medical record in the hospital database. This flow is mapped out below.



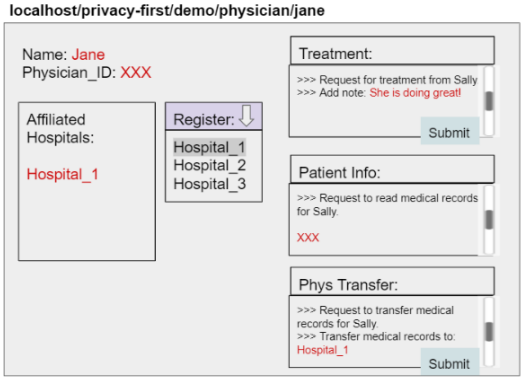
We defined the following API endpoints. The API endpoints that entail a physician reading, writing, or transferring medical records are patient initiated in which a card would need to be presented in order for the physician to successfully make a request.



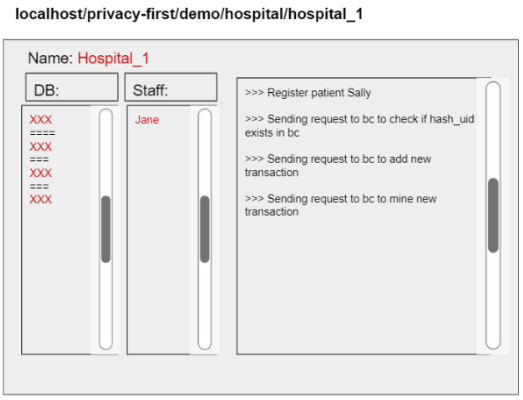
A patient can make the following requests with their card and any updates to the patient blockchain that is stored in the hospital database is treated as a transaction that is automatically signed by the requester. –
We thought a bit about how an end-user would use our application and mocked out the following web portals for a patient, physician, and hospital.



Portal for Patient Sally



Portal for Physician Jane



Portal for Hospital_1

We envision future iterations to include translating a patient card to a QR code that a physician could scan in order to make any of the patient initiated requests like transfer/read/write. Also, we may have applications using our infrastructure design implement digital cards (as a phone application) for ease of use. The underlines the fact that the 'card' just needs to be something unique that can hold a private key and tethered in some way to the user.

–

Our system makes the following assumptions:

1. There is no collusion between the physician and the hospital database admin
2. There is no collusion between the patient and the hospital database admin
3. There is no mutual trust among physicians and hospitals
4. Two-factor authentication can prevent identity theft if a patient's card is stolen or lost
5. Since a patient card is associated with immutable information on the public blockchain, an alternative recovery mechanism for continued interaction in our ecosystem would need to be thought about if a patient's card is stolen or lost
6. A patient can only be active with one hospital at a time. However, it would take a matter of seconds to transfer one's records to another hospital using in our system
7. Information is sent over encrypted wires to prevent man in the middle attacks

4 Implementation

To scope down the implementation of our project and to focus on creating a meaningful proof-of-concept, we defined a configuration file that specifies the entities involved and their contact information (i.e. ip address and port number) and decided to use a single node in our blockchain network - which can be scaled up in future iterations.

We created proxy clients and servers for each of the entities involved (i.e. hospital_proxy_server.py, patient_proxy_client.py, blockchain_proxy_server.py, physician_proxy_client.py) that are responsible for message processing, invocation of internal API calls, and spinning up a flask server for the creation and updating of the entity's web portal.

We created a DNS like service that essentially parses our configuration file to simplify the lookup of other entities in the ecosystem.

There were several interesting design decisions that needed to be made to create clean interfaces and to enable the interoperability of all entities in our ecosystem.

Our implementation, in total, is about 4,500 lines of code. The end result, that we encourage you to play around with, entails running a startup script that will kick-off all of the required proxy clients/servers and

navigating to the portal's of each of the entities to see how requests are made and processed using our service. We felt that this experience would help demonstrate the different perspectives and users of our service.

The main challenges that arose for us was accommodating the knowledge gap since we are far removed from the medical field. This required a bit of research and investigation to make reasonable yet realistic assumptions.

5 Evaluation

We achieved what we set out to address by testing that our system can accommodate GDPR requirements, most specifically the 'right to erasure' requirement. A patient is able to request to remove their data from a hospital and their data is indeed removed. We were able to confirm this with the prototype we built. However, this was not the only problem that we set out to solve. We also wanted to tackle unrestricted access to patient data, malpractice through patient record tampering, patient cheating, and the slow and insecure removal and transfer of medical records. We believe we solved these problems as well.

1. Unrestricted access to patient data - A physician is unable to read patient data or update a patient's medical history without patient consent. We were able to test this by confirming that when a physician obtains a patient's card and has updated a patient's medical history, for instance, they are unable to submit more updates for the patient since the card provided is immediately deactivated after the request to write to a hospital's database is made.

2. Patient record tampering - A patient's data cannot be tampered with since all patient data is encrypted, all updates are signed by the requester, and in order for anything meaningful to happen with patient data, the patient card must be supplied. We mentioned above that we can solve a patient losing their card with two factor authentication, but the patient card indeed stores the capabilities that grant access to a patient's medical records.

3. Patient cheating - A patient is unable to register or seek treatment from multiple hospitals at the same time since the public blockchain maintains a record of all patient registrations and each hospital consults the public blockchain before accommodating or processing any requests.

4. Slow and insecure removal and transfer of medical records - These two requests happen in a matter of seconds. The transfer of medical records is seamless since the public blockchain serves as a mechanism for all participating hospital entities to consult and utilize for the duration of a patient's existence. Since each hospital uses the same hash function to disassociate a user id from the id stored on the public blockchain, each hospital is able to obtain the public key for each patient given the patient's card.

To make our system more efficient, we presume utilizing other data structures like a Merkle Tree for our blockchain implementation may provide a performance benefit when finding the block for a given hashed user id.

6 Conclusion

Our infrastructure design accomplishes both GDPR compliance and the use of a blockchain for the primary data storage (through linking). This infrastructure could be tailored to any application that has records and corresponding accounts (most applications). We showed through our hospital record management application that applications using our design can enable their users with full control over their data without fear that an action on their data be made unnoticed and without their consent.

7 Acknowledgements

A huge thank you to our professor Malte Schwarzkopf for helping us make our idea come to fruition with the willingness to provide us with feedback and guidance throughout the process.