**COVIBLOCK: A MERKLE DAG AND BLOCKCHAIN**

**IMPLEMENTATION FOR COVID-19 RECORDS**

A Capstone Project Presented to the Graduate Program

College of Engineering and Technology

Pamantasan ng Lungsod ng Maynila

In Partial Fulfillment of the Requirements for the Degree

Master in Information Technology

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

By

Jennifer L. Fadriquela

Dr. Khatalyn E. Mata

Adviser

August 2021

**APPROVAL SHEET**

The capstone project hereto titled

**COVIBLOCK: A MERKLE DAG AND BLOCKCHAIN**

**IMPLEMENTATION FOR COVID-19 RECORDS**

prepared and submitted by Jennifer L. Fadriquela in partial fulfilment of the requirements for the degree of Master in Information Technology has been examined and is recommended for acceptance and approval for **ORAL EXAMINATION**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DR. KHATALYN E. MATA**

Adviser

PANEL OF EXAMINERS

Approved by the Committee on Oral Examination

with a grade of \_\_\_\_\_\_\_\_\_ on \_\_\_\_\_\_\_.

**PROF. MANUEL L. OCAMPO**

Panel Chair

Chairman

**PROF. EDGARDO S. DAJAO** **PROF. DAN MICHAEL A. CORTEZ**

Panel Member Panel Member

Member Member

Accepted and approved in partial fulfilment of the requirements for the degree of

Master in Information Technology

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DR. JOSEPH BERLIN P. JUANZON** **ENGR. JUAN C. TALLARA JR.**

Director, CET Dean

Graduate Program Graduate Program

**TABLE OF CONTENTS**

[TITLE PAGE](#_heading=h.2s8eyo1) i

APPROVAL SHEET ii

TABLE OF CONTENTS iii

LIST OF FIGURES iv

LIST OF TABLES v

INTRODUCTION 6

1.1 Background of the Study 6

1.2 Statement of the Problem 8

1.3 Objective of the Study 8

1.4 Scope and Limitation 8

1.5 Significance of the Study 9

REVIEW OF RELATED LITERATURE 10

2.1 Related Literature 1

2.2 Related Studies 12

THEORETICAL FRAMEWORK 20

3.1 Conceptual Framework 30

METHODOLOGY 33

4.1 Requirement Modelling 34

4.2 Quick Design 42

LIST OF REFERENCES 50

**LIST OF FIGURES**

Figure 3.1 – Diagram of Proposed Solution 20

Figure 3.2 – Hashing Process 21

Figure 3.3 – Generic Blockchain Transactions 23

Figure 3.4 – Merkle Tree Implementation using hashes 25

Figure 3.5 – DAG Illustration 26

Figure 3.6 – Merkle DAG implemented on a file system 27

Figure 3.7 – OpenPGP Two-way process 28

Figure 3.8 – Smart Contract comparison 29

Figure 3.9 – Conceptual Framework 31

Figure 4.1 Prototype Model Phases and Process 33

Figure 4.2 – Sample Rapid Antigen Test Result 35

Figure 4.3 - Sample Real-Time Polymerase Chain Reaction Test Result 36

Figure 4.4 - Sample Covid Vaccination Certificate 37

Figure 4.5 – Merkle DAG representing sample records 40

Figure 4.6 – Context Diagram 42

Figure 4.7 – Data Flow Diagram 43

Figure 4.8 – Proposed Use Case Diagram 44

Figure 4.9 – Transactional Operation Diagram 45

Figure 4.10 – Key Generation Process Flowchart 45

Figure 4.11 – Third party access to patient file Flowchart 46

Figure 4.12 – File Uploading Flowchart 47

Figure 4.13 – File Retrieval Flowchart 48

**LIST OF TABLES**

Table 2.1 – SHA-1 and SHA-2 Comparison 12

**Chapter One**

**INTRODUCTION**

* 1. **Background of the Study**

With the advancement of computer technology, electronic documentation and the use of electronic medical records have become more feasible. Medical records on a shared computer network that are read and written electronically on a relational database using a graphic user interface are referred to as electronic medical records. In the study entitled “A comparison of electronic records to paper records in mental health centers” (Tsai and Bond, 2007), they looked at three mental health facilities that had recently switched from paper to electronic medical records. Electronic records' documentation was shown to be more thorough and retrievable than paper records. As per the study, this finding can be a factor to take in when making treatment decisions.

In the study entitled “Perceived Benefits of Implementing and Using Hospital Information Systems and Electronic Medical Records” (Khalifa, 2018), they pointed out six ways EMRs could enable data accessibility and care organization: improving access to data during patient encounters, improving processes workflow, managing information overflow to clinicians, enhancing medical decision-making process care plans, supporting operational processes and improving financial data accessibility. They also emphasized that when a computer was used to retrieve patient information, physicians earned higher overall patient satisfaction rates, and when a computer was used to enter patient information, physicians received identical satisfaction rates.

The current technological advancements in the Philippines has yet to be manifested in its healthcare system. Though there were efforts from the government to adopt various modern tools, we are still miles behind other countries. On a study entitled “Barriers to the Adoption of Electronic Medical Records in Select Philippine Hospitals: A Case Study Approach” (Ebardo and Celis, 2019), identified barriers such as weak infrastructure, technology complexity and poor interface design of applications have made it difficult for various health organization to progress. Another study entitled “Barriers to Electronic Health Record System Implementation and Information Systems Resources: A Structured Review” (Gesulga et.al, 2017), they determined another set of barriers to the adoption of EMRs in the Philippines namely: User resistance, lack of education and training, and concerns arising from data security. In a paper entitled “Identifying Healthcare Information Systems Enablers in a Developing Economy” (Ebardo and Tuazon, 2019), they discussed how the integration of existing information systems to be “paper-less” can produce potential savings. This is crucial given that the Philippines is still a developing country and has budget constraints to health systems.

Ospital ng Makati, a government-funded hospital, is one of the main facilities for Covid patients in the City of Makati. Being a public hospital, it is expected to handle larger demographics compared to its private counterparts. Since the start of the pandemic, it is one of the primary health care facility in the city that facilitates Covid-19 test results and vaccination.

Presently, the hospital has no in-house molecular laboratory. They send requests to third-party laboratories in executing those tests. After successfully doing the tests, those laboratories will forward their results back to the hospital. These results are then given physically to the patient while the hospital keeps a copy in their archive. For vaccination, the City of Makati has an online web portal to assist with scheduling. This online system aims to lessen the crowd going to the vaccination site as they are guaranteed of a slot as opposed to other cities that implement a “first come first serve” basis. The proof of vaccination is a physical certificate and the patient being tagged as fully vaccinated in the system.

Other cities also had the same effort of putting up systems to cater the pandemic needs. Manila, Mandaluyong and Taguig also have their own sets of application. Though the motivation is good, the issue of how to unify these applications have risen and calls for having a unified system on a national level are being thrown in various media outlets.

* 1. **Statement of the Problem**

At present, Ospital ng Makati is using a system for keeping Covid Test Results while vaccination records are maintained on a separate system used by the entire City of Makati. The patients only receive physical copies of these records as proof of execution.

Current setup for organizing and managing these records still has shortcomings. One aspect that the researcher can improve is the strategy for storing records since this is still stored or archived physically both by the hospital and patients.

* 1. **Objectives of the Study**

This study aims to develop an alternative platform to store Covid-19 related records for Ospital ng Makati.

Specifically, the study seeks to address the following objectives:

1. To develop an application that will decentralize storage of Covid-related files.
2. To provide an alternative way to minimize record tampering of uploaded files in CoviBlock by applying concepts of Merkle DAG and blockchain.
3. To secure uploaded files in CoviBlock by using asymmetric key encryption scheme.
   1. **Scope and Limitations**

The study will be focusing on developing an application for management of Covid-related records for Ospital ng Makati. Since there are privacy regulations concerning health information, the researcher will use dummy data and instead will probe more on the processes on how these records are archived or managed.

The study will exclude the test results generation and vaccine management. Thus, it is more focused on how the result or outputs of these processes. The study assumes that outputs are already generated in computer readable format such as images (.png, .jpg) or documents (.pdf).

The study will only be concerned on two types of records: Test Results and Vaccine Certificates. The researcher will concentrate on developing an alternative storage system and accessibility strategy for patients and other verifying party.

* 1. **Significance of the Study**

Results obtained from the study will benefit the following stakeholders:

**Patients**. Above all, patients will greatly benefit on this application. Various regulations and laws have been implemented to ensure people are not spreaders or vaccinated. Currently, there are no unified way in getting and presenting these records are proof. More so, bad actors are using this pandemic to make money out of tampering records. The application will help solve the woes of patients in terms on ease of access and portability of their records. They will also have full autonomy of said records.

**Medical Personnel**. The application will help medical workers to focus on their medical line of duty and alleviating various admin jobs.

**Third Party Validators.** As mentioned above, records tampering has become rampant. Businesses or employers requiring such records can now be protected of this illegal activity.

**Chapter Two**

**REVIEW OF RELATED LITERATURE**

This chapter covers studies and other literatures carried out by foreign and domestic researchers that have a significant impact on the variables investigated in this study. These studies focus on several factors that will help with the research's development. Literatures mentioned here will be of different sources: books, journals, articles, electronic materials such as PDF or E-Book, and other existing thesis and dissertations, foreign and local. Their inclusion will be considered supplemental in developing the proposed solution of this study.

**2.1 Related Literature**

**SHA-256**

A cryptographic hash function is an algorithm that takes input strings of arbitrary (typically very large) length and maps these to short fixed length output strings. SHA-256 is a cryptographic hash function introduced in 2002 in a paper titled “FIPS-180-2: Secure Hash Standard (SHS)” by National Institute of Standards and Technology. SHA-256 is constructed from MD(Merkle-Damg˚ard) -construction and Davis-Meyer mode. The compression function of SHA-256 has 64 rounds, two kinds of non-linear functions, cyclic rotations, and round-dependent constants. The hash value calculated by SHA-256 is 256 bits long. The function obtained from the compression function of SHA-256 by removing the feed-forward operation of the Davis-Meier mode is invertible (Yoshida and Biryukov, 2005).

SHA-256 is a hash function that is based on the well-known Davies-Meyer construction of hash functions (Menezes et.al, 1997). The variable-length message *M* is divided into 512-bit blocks *M0, M1,...,Mn−1*. The 256-bit hash value *Vn* is then computed as follows:

*V0 = IV ; Vs+1*= compress(*Vs, Ms*) = *EMs (Vs) + Vs* for 0 ≤ s < n,

where compress is the compression function, *IV* is a fixed initial value and *EK(X)* is the block cipher, SHACAL-2. The function *EK(X)* is an iterated design that only uses simple operations on 32-bit words. The 256-bit input *Vj* is loaded into 8 registers *(A, B, C, D, E, F, G, H)* and the 512-bit message block is divided into 16 words of 32 bits *(W0 ...W15)* and these words are expanded to a sequence of 64 words through the message schedule:

*σ0(X) = ROTR7(X) ⊕ ROTR18(X) ⊕ SHR3(X);*

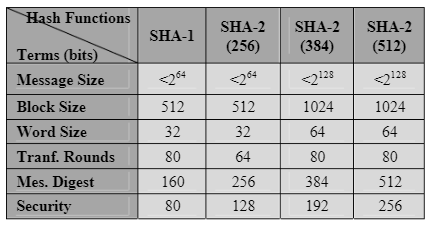
*σ1(X) = ROTR17(X) ⊕ ROTR19(X) ⊕ SHR10(X);*

*Wt = σ1(Wt−2) + Wt−7 + σ0(Wt−15) + Wt−16*

where *ROTRn* is right rotation by *n* bits. SHACAL-2 encrypts the initial value using this sequence as a key.

SHA-256 has better security from its predecessor family – SHA-1. Most features of the basic components of SHA-2 seem to provide a better security level than for preceding hash functions, even though the relative number of rounds is somewhat lower than for SHA-1 for instance, and though the selection criteria and security arguments for some design choices are difficult to reconstruct from the specification, in the absence of any public design report (Gilbert and Handschuh, 2004).

Sklavos and Koufopavlou (2003) cited a comparison of SHA-1 vs. three SHA-2 hash functions. These three hash functions differ in terms of the block size and words of data that are used during hashing. They also asserted that hash functions of the SHA-2 family differ most significantly in the number of security bits that are provided for the hashed input data block. Security is directly related to the message digest length.



**Table 2.1 – SHA-1 and SHA-2 Comparison**

In their paper, they presented above table which presents the basic properties of all four secure hash functions.

**Merkle Trees**

In 1989, Ralph Merkle introduced the Merkle tree in his paper “A Certified Digital Signature”. The Merkle tree is a tree constructed bottom-up. More precisely, the tree discussed in this paper is a full binary tree and constructed from the bottom-up. Assume that the height of the tree is *hm*, and the tree owns 2hm data blocks *xi* and *yi=hash(xi),i∈[0,2hm−1]*, where *yi* is a leaf node value of the Merkle tree. Each value of the parent node is the hash of the concatenation of its children, *yparent=hash(yleft|yright)*, where | refers to concatenation. Below is a pseudocode format of the Classic Merkle Tree Traversal algorithm:

1. Set *leaf* = 0.

2. Output:

• Compute and output *leaf* with *LEAFCALC(leaf)*

• For each *h* ∈ [0,*H* − 1] output {*authh*}.

3. Refresh Auth Nodes:

For h such that 2h divides leaf + 1:

• Set authh be the sole node value in stackh.

• Set *startnode* = (*leaf* + 1 + 2h) ⊕ 2h.

• *stackh.initialize(startnode,h)*.

4. Build Stacks:

For all *h* ∈ [0,H − 1]:

• stackh.update(2).

5. Loop

• Set *leaf* = *leaf* + 1.

• If *leaf* < 2*H* go to Step 2

A Logarithmic Merkle Tree Traversal was proposed by M. Szydlo (2003). The main idea of the improved algorithm is, to reduce the memory requirements, by reducing the number of active treehash instances during the signature generation.. Here is the pseudocode:

1. Set *leaf* = 0.

2. Output:

• Compute and output leaf with *LEAFCALC*(*leaf*)

• For each *h* ∈ [0,*H* − 1] output {*authh*}.

3. Refresh Auth Nodes:

For *h* such that 2*h* divides *leaf* + 1:

• Set *authh* be the sole node value in *stackh*.

• Set *startnode* = (*leaf* + 1 + 2*h*) ⊕ 2*h*.

• *stackh.initialize(startnode,h).*

4. Build Stacks:

Repeat the following 2*H* − 1 times:

• Let *lmin* be the minimum of {*stackh.low*} for all *h* = 0,...,*H* − 1.

• Let focus be the least *h* so that *stackh.low* = *lmin*.

• *Stackfocus.update*(1).

5. Loop

• Set *leaf* = *leaf* + 1.

• If *leaf* < 2*H* go to Step 2.

In Fractal merkle tree representation (Micali et.al, 2003) and traversal, the goal is to split up the merkle tree in subtrees and to save and compute these subtrees, instead of single nodes. Below is the pseudocode:

1. Set *leaf* = 0.

2. Output:

• Compute and output leaf with *LEAFCALC*(*leaf*)

• For each *j* ∈ [0,*H* − 1] output {*authj*}.

3. Next Subtree:

For each *i* for which *Existi* is no longer needed, i.e., for *i* ∈ {1, 2,...,*L*} with *leaf* = 1(*mod*2*hi*):

• Set *Existi* = *Desirei*.

• Create new empty *Desirei* (if *leaf* + 2*ih* < 2*H*).

4. Grow Subtrees

For each *i* ∈ {1, 2,...,*h*}: Grow tree *Desirei* by applying 2 units to modified treehash (unless *Desirei* is completed)

5. Increment *leaf* and loop back to step 2 (while *leaf* < 2*H*).

**Distributed Hash Tables**

Distributed Hash Tables (DHTs) are widely used to coordinate and maintain metadata about peer-to-peer systems. For example, the BitTorrent MainlineDHT tracks sets of peers’ part of a torrent swarm. Kademlia was introduced in a paper titled “Kademlia: A peer-to-peer information system based on the xor metric” (Maymounkov and Mazieres, 2002). It is a DHT which provides:

1. Efficient lookup through massive networks: queries on average contact dlog2(n)e nodes. (e.g. 20 hops for a network of 10, 000, 000 nodes).

2. Low coordination overhead: it optimizes the number of control messages it sends to other nodes.

3. Resistance to various attacks by preferring long-lived nodes.

4. Wide usage in peer-to-peer applications, including Gnutella and BitTorrent, forming networks of over 20 million nodes.

In “Democratizing content publication with coral” (Freedman et.al, 2004), it examined Coral DSHT as an extension of Kademlia in three particularly important ways:

1. Kademlia stores values in nodes whose ids are “nearest” (using XOR-distance) to the key.

2. Coral relaxes the DHT API from get\_value(key) to get\_any\_values(key) (the “sloppy” in DSHT).

3. Additionally, Coral organizes a hierarchy of separate DSHTs called clusters depending on region and size.

Another approach, S/Kademlia DHT (Baumgart and Mies. 2007) extends Kademlia to protect against malicious attacks in two particularly important ways:

1. S/Kademlia provides schemes to secure NodeId generation, and prevent Sybill attacks

2. S/Kademlia nodes lookup values over disjoint paths, in order to ensure honest nodes can connect to each other in the presence of a large fraction of adversaries in the network.

Xie (2003) discussed how DHTs are implemented in P2P systems in his paper “P2P Systems based on Distributed Hash Table”. Files are associated to keys (produced by hashing the file name); each node in the system handles a portion of the hash space and is responsible for storing a certain range of keys. After a lookup for a certain key, the system will return the identity (e.g., the IP address) of the node storing the object with that key. The DHT functionality allows nodes to put and get files based on their key and has been proved to be a useful substrate for large distributed systems and a number of projects are proposing to build Internet-scale facilities layered above DHTs. In DHTs, each node handles a portion of the hash space and is responsible for a certain key range. Routing is location-deterministic distributed lookup. The most important enhancements are deterministic locating and load balance.

• No global knowledge

• Absence of single point of failures

**Blockchain**

Blockchains are a type of distributed ledger written by decentralized and usually anonymous groups of agents rather than known centralized parties. This novel method of recordkeeping has introduced two economic innovations that overcome the two limitations of competition among centralized ledgers. There is free entry of record-keepers: any agent may write on the ledger so long as they follow a certain set of established rules. Furthermore, information on an existing blockchain is portable to a competing one. A software developer can propose to “fork off” an existing blockchain to establish one with different policies while retaining all the information contained in the original blockchain. Fork competition eliminates the inefficiencies arising from switching costs in centralized record-keeping systems (Abadi and Brunnermeier, 2018).

On an article “Blockchain Technology Overview” (Yaga et.al 2018), they mentioned four key characteristics of this technology:

• Ledger – the technology uses an append only ledger to provide full transactional history. Unlike traditional databases, transactions and values in a blockchain are not overridden.

• Secure – blockchains are cryptographically secure, ensuring that the data contained within the ledger has not been tampered with, and that the data within the ledger is attestable.

• Shared – the ledger is shared amongst multiple participants. This provides transparency across the node participants in the blockchain network.

• Distributed – the blockchain can be distributed. This allows for scaling the number of nodes of a blockchain network to make it more resilient to attacks by bad actors. By increasing the number of nodes, the ability for a bad actor to impact the consensus protocol used by the blockchain is reduced.

Blockchain is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger. With a previous block hash contained in the block header, a block has only one parent block. It is worth noting that uncle blocks(children of the block’s ancestors) hashes would also be stored. The ﬁrst block of a blockchain is called genesis block which has no parent block (Zheng et.al, 2017).

In the article of Monrat et.al (2019) titled “A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities”, they identified six comparison perspectives when comparing blockchain networks:

1. Consensus Determination - All the nodes can participate in the consensus process in the public blockchain such as Bitcoin, while only a few selected set of nodes are being responsible for confirming a block in the consortium blockchain. In the private blockchain, a central authority will decide the delegates who could determine the validated block.

2. Read Permission - Public blockchain allows read permission to the users, where the private and consortium can make restricted access to the distributed ledger. Therefore, the organization or consortium can decide whether the stored information needs to be kept public for all or not.

3) Immutability - In the decentralized blockchain network, transactions are stored in a distributed ledger and validated by all the peers, which makes it nearly impossible to modify in the public Blockchain. In contrast, the consortium and private Blockchain ledger can be tampered by the desire of the dominant authority.

4) Efficiency - In the public blockchain, any node can join or leave the network which makes it highly scalable. However, with the increasing complexity for the mining process and the flexible access of new nodes to the network, it results in limited throughput and higher latency. However, with fewer validators and elective consensus protocols, private and consortium blockchain can facilitate better performance and energy efficiency.

5) Centralized - The significant difference among these three types of Blockchain is that the public blockchain is decentralized, while the consortium is partially centralized and private blockchain is controlled by a centralized authority

**Asymmetric Encryption**

Asymmetric encryption schemes are usually used only for secretly transmitting a session key of a symmetric encryption scheme for message encryption. In fact, the hybrid usage of asymmetric and symmetric encryption schemes is very common in practice (Fujisaki and Okamoto, 2011)

Goldwasser and Micali (1984) discussed the symmetric (aka private-key) encryption scheme as follows. Given by a pair of algorithms, *Π* = (*E,D*), where for every sufficiently large *k* ∈ *N*,

• *E*, the encryption algorithm, is a probabilistic polynomial-time (in *k*) algorithm that takes secret key *a* ∈ KSP and message *x* ∈ MSP, draws coins *r* uniformly from coin space COIN, and produces ciphertext *y := Ea(x;r).* This experiment is written as *y* ← *Ea(x).* The key, message, and coin spaces, KSP, MSP and COIN, are uniquely determined by *k*.

• *D*, the decryption algorithm, is a deterministic polynomial-time (in *k*) algorithm that takes secret key *a* ∈ KSP and ciphertext *y* ∈ {0, 1}∗, and outputs message *x := Da(y).*

We require that a symmetric encryption scheme should satisfy the correctness condition: For every sufficiently large *k* ∈ *N*, every *a* ∈ KSP and every *x* ∈ MSP, we always have *Da(Ea(x)) = x*.

Bellare et.al (1998)detailed the asymmetric (aka public-key) encryption scheme. Given by a triple of algorithms, *Π = (K, E,D)*, where for every sufficiently large *k* ∈ N:

• *K*, the key-generation algorithm, is a probabilistic polynomial-time (in *k*) algorithm which on input *1k*outputs a pair of strings, (*pk,sk*), called the public and

secret keys, respectively. This experiment is written as (*pk,sk*) ← *K*(1*k*).

• *E*, the encryption algorithm, is a probabilistic polynomial-time (in *k*) algorithm

that takes public key pk and message *x* ∈ *MSP*, draws coins *r* uniformly from coin

space COIN, and produces ciphertext *y := Epk(x;r).* This experiment is written as *y ← Epk(x).* The message and coin spaces, MSP and COIN, are uniquely determined by *pk*.

• *D*, the decryption algorithm, is a deterministic polynomial-time (in *k*) algorithm that takes secret key *sk* and ciphertext *y* ∈ {0, 1}∗, and returns message *x := Dsk(y).*

We require that an asymmetric encryption scheme should satisfy the following correctness condition: For every sufficiently large *k* ∈ *N*, every (*pk,sk*) generated by *K*(1*k*)

and every *x* ∈ *MSP*, we always have *Dsk(Epk(x)) = x*.

**Chapter Three**

**THEORETICAL FRAMEWORK**

**Present State of Covid-19 Tests and Vaccine Certificate Storage**

Documents and certificates given out by various units (private and public) for Covid-19 related tests such as rRt PCR and Cartridge – Based PCR are still on paper-form. There are some units that store the results in their server and can be accessed online thru their website. Same is true with giving out vaccine certificates. Primary providers of vaccines are Local Government Units (LGUs) and they vary in implementation. Some only give out physical copies (certificates, cards) and others have virtual copies on their websites stored on their servers. There is a disconnect on a unified tracking of all these documents and might result to issues when these documents will be used on different areas of the Philippines. The usual proposition to solve this is to create a unified website that will be hosted in a central server.

**Proposed Documents Storage Structure**

**Figure 3.1 – Diagram of Proposed Solution**

Above is a summarized approach in solving the problem in document storage. The main components of this application will be the IPFS for file storage and blockchain to record the logs of transaction being done in the system.

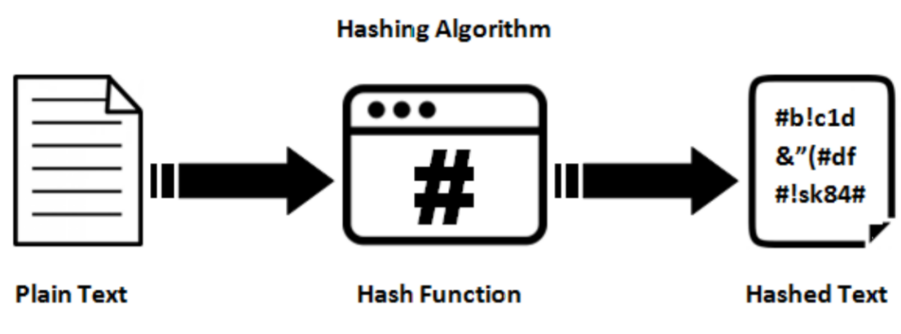
The next sections will discuss the different algorithms and frameworks to be used in order to achieve the proposed solution.

**Cryptographic Hash Functions**

A cryptographic hash function is an algorithm which maps data of any size (often called the "message") to a bit array of a fixed size ("hash value", "hash", or "message digest"). It is a one-way function, that is, a function which is practically infeasible to invert or reverse the computation. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.

The ideal cryptographic hash function has the following main properties:

* it is deterministic, meaning that the same message always results in the same hash
* it is quick to compute the hash value for any given message
* it is infeasible to generate a message that yields a given hash value (i.e. to reverse the process that generated the given hash value)
* it is infeasible to find two different messages with the same hash value
* a small change to a message should change the hash value so extensively that a new hash value appears uncorrelated with the old hash value



**Figure 3.2 – Hashing Process**

Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value.

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. In theoretical cryptography, the security level of a cryptographic hash function has been defined using the following properties:

* Pre-image resistance

Given a hash value h, it should be difficult to find any message m such that h = hash(m). This concept is related to that of a one-way function. Functions that lack this property are vulnerable to preimage attacks.

* Second pre-image resistance

Given an input m1, it should be difficult to find a different input m2 such that hash(m1) = hash(m2). This property is sometimes referred to as weak collision resistance. Functions that lack this property are vulnerable to second-preimage attacks.

* Collision resistance

It should be difficult to find two different messages m1 and m2 such that hash(m1) = hash(m2). Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for pre-image resistance; otherwise collisions may be found by a birthday attack.

**Blockchain**

In 2008, Satoshi Nakamoto released a whitepaper titled “Bitcoin: A peer-to-peer electronic cash system”. This paper proposed a system for electronic transactions which uses a peer-to-peer network. Participating nodes in the network utilize Proof-of-Work to record public history of transactions.

At its most basic level, blockchain technology permits a network of computers to have a consensus on the true status of a distributed ledger at regular intervals. Blockchain network users submit potential transactions to participating nodes. The network will then choose a publishing node to update the pending transaction. Once this is done, transaction will be propagated to non-publishing nodes. Transactions are logged chronologically – with information being passed from the first transaction (or blocks) up to the last. This repetitive process forms an immutable chain on which all blocks are interconnected with each other.

Transactions are inserted to the blockchain when a publishing node creates a block. A block may represent various types of data from simple texts to complicated ones such as digital rights or intellectual property. It is divided into two parts, header and body. Header contains metadata and body is for the actual data being persisted in the blockchain. Below is a typical specification of these 2 parts:

Block Header

* Previous block header’s hash value
* Hash representation of block data
* Timestamp
* Size of the block
* Nonce value. In Bitcoin and other Proof-of-Work blockchains, this is a number manipulated by the publishing node to solve the hash puzzle.

Block Data

* Actual data (text, files)

**Figure 3.3 – Generic Blockchain Transactions**

Figure 3.3 shows how blockchain works given we have a simple data of text. The first block is called the genesis block and is automatically generated upon the chain’s creation. This genesis block will be the seed and considered as reference of all blocks going forward. Blocks are linked through each block containing the hash value of the previous block’s header, thus creating the chain. In case a previously published block was changed, it will have a different hash. This will create a domino effect on all subsequent blocks to also have a different hash because they contain the hash of the altered block.

An essential part of the blockchain is identifying which user will publish the next block or become the next publishing node. This is solved by implementing a consensus model. The common model used is to compete on who will publish it and winning an incentive in doing so.

Once a user joins a blockchain network, they agree to the initial state of the system. This is recorded in the only pre-configured block or the genesis block. Each blockchain network have a genesis block on to which all subsequent blocks would reference to. Each block must be valid and can be validated independently by each blockchain network user.

The following properties of a consensus model are:

* The initial state of the system is agreed upon (e.g., the genesis block).
* Users agree to the consensus model by which blocks are added to the system.
* Every block is linked to the previous block by including the previous block header’s hash digest (except for the first ‘genesis’ block, which has no previous block and for which the hash of the previous block header is usually set to all zeros).
* Users can verify every block independently

**InterPlanetary File Storage (IPFS)**

IPFS is a distributed platform for storing and retrieving files, websites, applications and data. It has rules that regulate in what manner data and content move around on the network. These rules are comparable in nature to Kademlia, the peer-to-peer distributed hash table (DHT) that is widely known for its use in the BitTorrent protocol.

IPFS is essentially a peer-to-peer system for retrieving and sharing IPFS objects. An IPFS object is a data structure with two fields:

* Data: a blob of unstructured binary data of size < 256 kB.
* Links: an array of Link structures. These are links to other IPFS objects. Links have 3 sub-parts:
  + Name: the name of the Link.
  + Hash: the hash of the linked IPFS object.
  + Size: the cumulative size of the linked IPFS object, including following its links.

IPFS builds a Merkle DAG, a blend of a Merkle Tree and a Directed Acyclic Graph (DAG).

A Merkle tree summarizes all the transactions in a block by producing a digital fingerprint of the entire set of transactions, thereby enabling a user to verify whether or not a transaction is included in a block. Merkle trees are created by repeatedly hashing pairs of nodes until there is only one hash left (this hash is called the Root Hash, or the Merkle Root). They are constructed from the bottom up, from hashes of individual transactions (known as Transaction IDs). Each leaf node is a hash of transactional data, and each non-leaf node is a hash of its previous hashes. Merkle trees are binary and therefore require an even number of leaf nodes. If the number of transactions is odd, the last hash will be duplicated once to create an even number of leaf nodes.



**Figure 3.4 – Merkle Tree Implementation using hashes**

A directed acyclic graph (DAG) is a conceptual representation of a series of activities. The order of the activities is depicted by a graph, which is visually presented as a set of circles, each one representing an activity, some of which are connected by lines, which represent the flow from one activity to another. Each circle is known as a “vertex” and each line is known as an “edge.” “Directed” means that each edge has a defined direction, so each edge necessarily represents a single directional flow from one vertex to another. “Acyclic” means that there are no loops (i.e., “cycles”) in the graph, so that for any given vertex, if you follow an edge that connects that vertex to another, there is no path in the graph to get back to that initial vertex.



**Figure 3.5 – DAG Illustration**

A Merkle DAG is a DAG where each node has an identifier, and this is the result of hashing the node's contents — any opaque payload carried by the node and the list of identifiers of its children — using a cryptographic hash function like SHA256. This brings some important considerations:

* Merkle DAGs can only be constructed from the leaves, that is, from nodes without children. Parents are added after children because the children's identifiers must be computed in advance to be able to link them. Every node in a Merkle DAG is the root of a (sub)Merkle DAG itself, and this subgraph is contained in the parent DAG.
* Merkle DAG nodes are immutable. Any change in a node would alter its identifier and thus affect all the ascendants in the DAG, essentially creating a different DAG. Take a look at this helpful illustration using bananas (opens new window)from our friends at Consensys.
* Merkle DAGs are similar to Merkle trees, but there are no balance requirements, and every node can carry a payload. In DAGs, several branches can re-converge or, in other words, a node can have several parents.

Identifying a data object (like a Merkle DAG node) by the value of its hash is referred to as content addressing. Thus, we name the node identifier as Content Identifier, or CID.

**Figure 3.6 – Merkle DAG implemented on a file system**

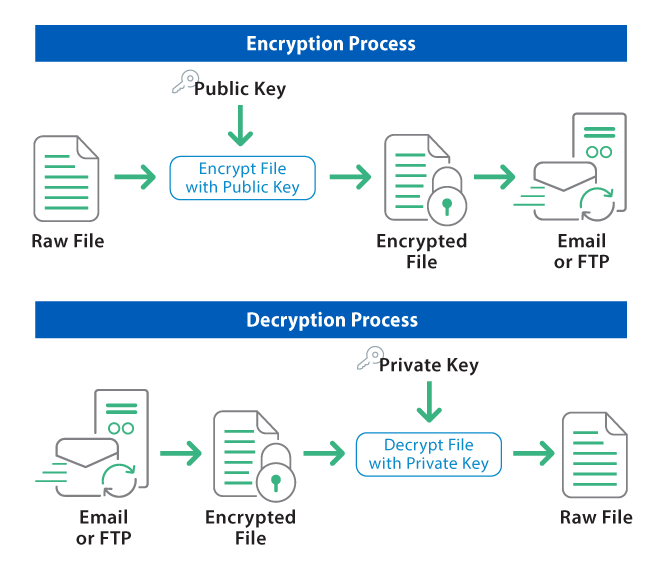
**Open Pretty Good Privacy**

OpenPGP is a key-based encryption method used to encrypt files so that only their intended recipient can receive and decrypt them. OpenPGP is used widely to secure e-mail communications, but its technology can also be applied to FTP.

OpenPGP works by using two cryptographic keys to secure files. A Public Key is used to encrypt the file so that only its corresponding Private Key can decrypt it.

The following is a step-by-step process of how OpenPGP Mode works with FTP.

1. The file to be uploaded is encrypted using a Public Key that the file's intended recipient has previously provided.
2. The encrypted file is uploaded to the FTP server.
3. The intended recipient retrieves the file from the FTP server.
4. Using the Private Key (which together with the Public Key used to encrypt the file initially comprises the Key Pair), the intended recipient decrypts the file and accesses its contents.

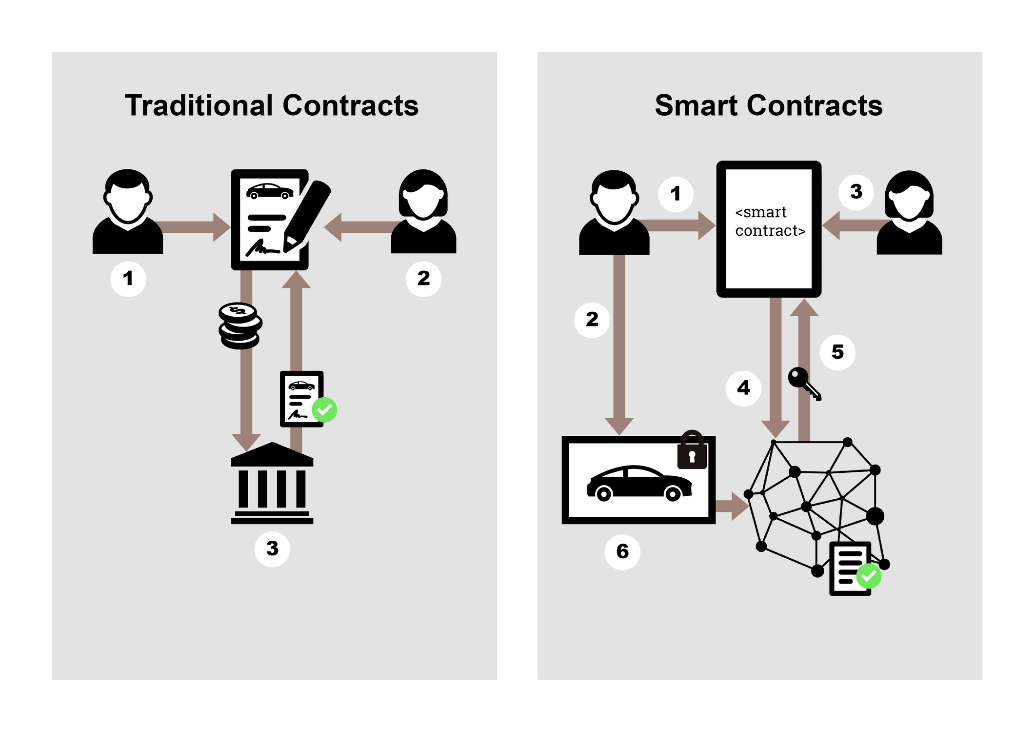


**Figure 3.7 – OpenPGP Two-way process**

**Smart Contract**

Smart Contract is a term used to describe computer code that automatically executes all or parts of an agreement and is stored on a blockchain-based platform. The code itself is replicated across multiple nodes of a blockchain and, therefore, benefits from the security, permanence and immutability that a blockchain offers. That replication also means that as each new block is added to the blockchain, the code is, in effect, executed. If the parties have indicated, by initiating a transaction, that certain parameters have been met, the code will execute the step triggered by those parameters. If no such transaction has been initiated, the code will not take any steps. Most smart contracts are written in one of the programming languages directly suited for such computer programs, such as Solidity.

If two individuals, Alice and Bob, do not know each other they also do not trust each other. When they want to make an agreement, they usually need a trusted third party that acts as intermediary. This intermediary verifies the transaction and can also enforce an action that was written into the agreement. With a smart contract in a blockchain, there is no need for a trusted intermediary because the clearing and settlement is automatically executed and enforced via blockchain technology.



**Figure 3.8 – Smart Contract comparison**

The following example in Figure shows the process of selling and buying a car between Alice and Bob. It also indicates the difference between a traditional contract and a smart contract. The comparison was originally created by BlockchainHub, this Figure was adopted by Braincept AG.

Traditional Contracts

1. Bob would like to sell his car.
2. Alice would like to buy a car.
3. A third party (intermediary) enables the trust that is needed in order to transfer the ownership of the car. Mostly different intermediaries are needed: motor vehicle registration authority, notary, insurance company. All middlemen take fees.

Smart Contracts

1. Bob would like to sell his car. He defines in a smart contract the conditions by which he will sell the car and signs the contract with his private key.
2. Bob leaves his car locked with a smart lock in his garage. The car has its own blockchain address and the smart lock is controlled by a smart contract.
3. Alice would like to buy a car. She finds Bobs car on an internet platform and signs the smart Bob’s contract with her private key. She adds the agreed amount from her blockchain address to Bob’s blockchain address.
4. As soon as the smart contract is executed the whole blockchain network will check if Bob is the real owner of the car and if Alice has enough money to buy the car.
5. If all peers in the blockchain network agree on the same state, it means that all conditions in the smart contract are met. The access code for the smart lock will be transferred to Alice and the blockchain address of the car will be registered to Alice. Bob will get the defined amount of money in his blockchain address.
6. Alice will be able to open the smart lock with her private key.

**Conceptual Framework**

This section aims to demonstrate the overview of the final product of this research. It identifies relevant variables, inputs, mappings and other components and how they will interact with each other. This includes all the underlying concepts and their associated mappings based on the system’s use.



**Figure 3.9 – Conceptual Framework**

The users of the proposed application will be patients, medical workers or other third-party requiring the patient to present a Covid-19 Test Result or Vaccine Certificate. The users will access the same application but with different levels of access depending on their role.

The input are the medical documents and distribution key. There will be different types of keys which will be discussed on Chapter 4. These keys will be used to authenticate and unlock or lock the files.

Once all required inputs are provided, the file will now go thru the necessary steps to access it. Depending on the type of transaction (insert a new file or retrieval), the keys provided should have enough privilege for it to succeed. The file hash will be then stored in the blockchain after going thru smart contracts. Once the blockchain successfully updated the network, provided file will now become an immutable component of both IPFS and blockchain network.

**Chapter Four**

**METHODOLOGY**

This chapter provides an overview of the strategies used to attain the study's goals. It describes the study's respondents as well as the research instruments that were used. It then goes on to explain the data collection strategies that contributed in the completion of the study endeavor.

**Figure 4.1 Prototype Model Phases and Process**

Figure 4.1 illustrates Prototype Model used by the researcher in developing the proposed study entitled “CoviBlock: A Blockchain and Peer-to-peer Platform for Covid-19 Test Results and Vaccine Records” which is under the family of System Development Life Cycle (SDLC). Prototyping was used to ensure faster turnaround time on each phase while addressing client’s requirements and feedbacks. This model also enables the researcher and client to have discussions in between development cycles.

The next sections of this chapter will discuss the phases of the used model.

**4.1 Requirements Modeling**

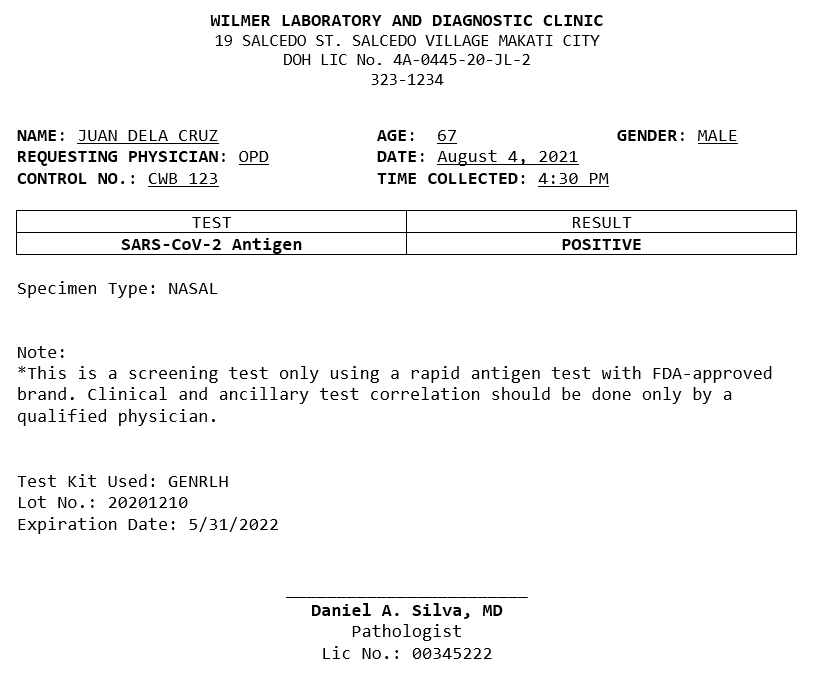
The Prototype Model starts with outlining the requirements. The researcher will conduct an initial investigation to determine the purpose and utilization of the application coupled with the nature and scope of the study. It is also in this stage that the researcher requested permission from medical unit authorities and other parties to conduct the study and all relevant data and information were examined.

Fact-finding was used via interviews and probing of processes to build a logical model of the application. With these interviews, the researcher was able to piece out a picture of transactions involved and analyzed them against the proposed solution. This information will also enable the researcher to identify critical decisions geared toward implementing the application.

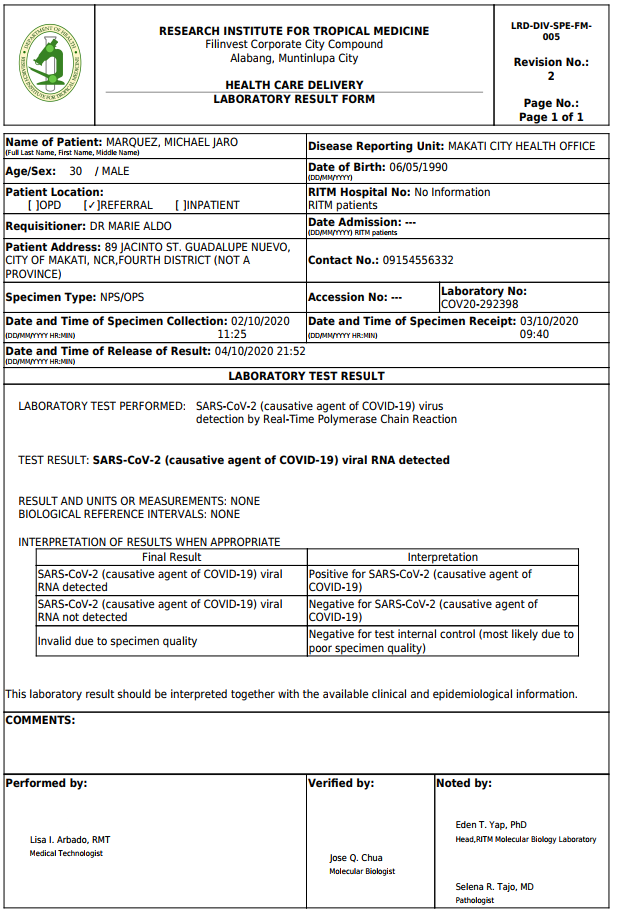
At present time, Ospital ng Makati does not have its own molecular laboratory to conduct independent Covid tests. Instead, they forward test requests to their partner third party laboratories. Once requested by the patient from the hospital, they will either execute the tests in-house or hand-over the execution to the third-party lab. After results comes out, the hospital will furnish a physical copy to the patient whilst maintaining a softcopy in their archive.

For vaccination records, Makati citizens are encouraged to register online via the web portal. This will ensure a scheduled slot on a specified date. On the day of vaccination, patient will be checked up by a physician to ensure he is fit for vaccination. The physician’s findings are logged on the system. Upon issuing a go signal, patient can now be vaccinated. After vaccination, vaccination site will sign a vaccination card while tagging the patient in their system as fully vaccinated.

The study will be focused on two Covid-related records: Test Results and Vaccine Certificate. Mocked test data will be used and will only be for the purpose of this research. This is due to various privacy regulation such as Health Insurance Portability and Accountability Act (HIPAA). This is a United States created health law adopted by medical facilities in the Philippines.

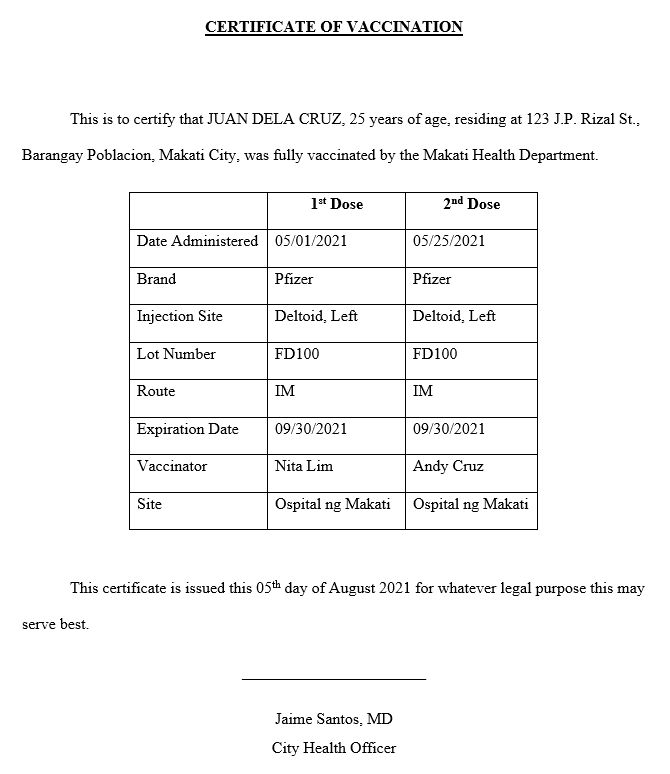


**Figure 4.2 – Sample Rapid Antigen Test Result**

****

**Figure 4.3 - Sample Real-Time Polymerase Chain Reaction**

**(RT-PCR) Test Result**



**Figure 4.4 - Sample Covid Vaccination Certificate**

Once above files are generated from existing system or printed by medical volunteers or workers, it is now ready to be consumed by the application.

Below are requirements grouped by specific role:

Patient

* Register and Login – register to gain access to the system
* Grant Access – Grant access to file requestors
* View Own Record – Get access to own results/certificate

Verifying Third Party

* Register and Login – register to gain access to the system
* View Patient Record – retrieve and view patient record if was given access

Physician/Medical Unit

* Register and Login – register to gain access to the system
* Upload record – upload a record for the patient

**4.1.1 Data Storage Scheme**

Since the study is primarily concerned on how medical records will be stored, this section will discuss the different schemes that will be used in the application. This will involve simulation, graphical visualizations and detailed discussions.

**4.1.1.1 IPFS – Merkle DAG**

The algorithm used in IPFS to manage content and assets is Merkle DAG. Suppose we want to upload 2 vaccine certificates. For brevity, we will use a small size text file to better illustrate the process. The default chunk size of IPFS is 256Kb but in this example we will reduce it to 32Kb to have appropriate representation using small sample files.

File 1

Name: cert\_allen\_smith.txt

Size: 86 bytes

Content:



File 2

Name: cert\_john\_doe.txt.txt

Size: 83 bytes

Content:



Generated Details for cert\_allen\_smith.txt:

|  |  |  |
| --- | --- | --- |
| Node Type | Size (Bytes) | Hash |
| Root | 0 | [QmZkJLp7PJGMc3mMSxTeLtyQCRqZ5CudGdjPB3jjTSFaoX](https://cid.ipfs.io/#QmZkJLp7PJGMc3mMSxTeLtyQCRqZ5CudGdjPB3jjTSFaoX) |
| Links | 32 | [QmdsyzBk5nWmC7a92gaAuRHxWTQu6e4wwyv2bVmZtF7mcq](https://cid.ipfs.io/#QmdsyzBk5nWmC7a92gaAuRHxWTQu6e4wwyv2bVmZtF7mcq) |
| Links | 32 | [QmPsFk9hcP4WmN96r8mXYjV5rKCNNb94c95jfqLBNZvigT](https://cid.ipfs.io/#QmPsFk9hcP4WmN96r8mXYjV5rKCNNb94c95jfqLBNZvigT) |
| Links | 22 | [QmVVrfBPAnF5DC1DXDZH2yftW6MEoCSKXEQEbY5LKfFzAt](https://cid.ipfs.io/#QmVVrfBPAnF5DC1DXDZH2yftW6MEoCSKXEQEbY5LKfFzAt) |

Generated Details for cert\_john\_doe.txt:

|  |  |  |
| --- | --- | --- |
| Node Type | Size (Bytes) | Hash |
| Root | 0 | [QmanmTVLostTHeeLiz8vr99QDWmVbmbd53rSA2iFoDcmXu](https://cid.ipfs.io/#QmanmTVLostTHeeLiz8vr99QDWmVbmbd53rSA2iFoDcmXu) |
| Links | 32 | [QmdsyzBk5nWmC7a92gaAuRHxWTQu6e4wwyv2bVmZtF7mcq](https://cid.ipfs.io/#QmdsyzBk5nWmC7a92gaAuRHxWTQu6e4wwyv2bVmZtF7mcq) |
| Links | 32 | [QmTfDsTDe3nVu7b3hij43R3mBzyhJZgVm9eFBewVb5FfKV](https://cid.ipfs.io/#QmTfDsTDe3nVu7b3hij43R3mBzyhJZgVm9eFBewVb5FfKV) |
| Links | 32 | [QmRF3DNTkA43a7AG26uva4n7pgR22ctz6PjZW4KMuN5Cvu](https://cid.ipfs.io/#QmRF3DNTkA43a7AG26uva4n7pgR22ctz6PjZW4KMuN5Cvu) |

We can now map out the links with their respective roots. Notice that link “Qmdsy” is referenced by both root objects.



**Figure 4.5 – Merkle DAG representing sample records**

**4.1.1.2 Blockchain**

The main purpose of using a blockchain is to validate whether a given or requested CID is authentic in the context of the system. After a doctor uploads a record in IPFS, the generated IPFS CID will then be logged to the blockchain. Blockchain validation will then be used as a proof that a CID exists in the context of CoviBlock. This will prevent illegal tampering or modification of records.

Taking our sample files from 4.1.1.1, we will create a blockchain of transactions given the files were already uploaded to IPFS and CIDs are generated. JSON Objects will be used as format of the payload.

Object for cert\_allen\_smith.txt:

{

“CID”: “[QmZkJLp7PJGMc3mMSxTeLtyQCRqZ5CudGdjPB3jjTSFaoX](https://cid.ipfs.io/#QmZkJLp7PJGMc3mMSxTeLtyQCRqZ5CudGdjPB3jjTSFaoX)”,

“MedPerson”: “Dr. Ramon Cruz”,

“LicenseNum”: “123-456”,

“DateTime”: “23/07/2021 14:00:00”

}

Object for cert\_john\_doe.txt:

{

“CID”: “[QmanmTVLostTHeeLiz8vr99QDWmVbmbd53rSA2iFoDcmXu](https://cid.ipfs.io/#QmanmTVLostTHeeLiz8vr99QDWmVbmbd53rSA2iFoDcmXu)”,

“MedPerson”: “Dr. Erik Lim”,

“LicenseNum”: “122-322”,

“DateTime”: “23/07/2021 09:00:00”

}

**4.2 Quick Design**

After identifying the requirements, a design of the proposed application is created. This is not a detailed design with complete technical specifications but a simplified one with critical aspects of the solution. This phase will give a bird’s eye view to the client of the application.

**4.2.1 Context Diagram**

**Figure 4.6 – Context Diagram**

The context diagram shown in Figure 4.6 summarizes the application on inputs and outputs of the system and targeted users. On general, users of the application will be required to provide public/private keys and raw files to be stored. It is now the application will trigger and execute various processes to upload, encrypt/decrypt, or release files. Note that this is a general illustration of inputs and outputs. Next sections of this chapter will discuss the mentioned processes on this diagram.

**4.2.2 Data Flow Diagram**

**Figure 4.7 – Data Flow Diagram**

Figure 4.7 illustrates how various types of users receives and provides information to the application and how the application provides and receives data from users. This also mentions the executing process to generate the data.

It is important to note that authorized medical personnel are the only ones allowed to upload files. Patients will have to generate private and public keys for their files to be uploaded or requested. These keys are crucial for a patient file to be encrypted or decrypted. Third parties can request for patient files and will be granted access to view decrypted files.

**4.2.3 Use Case Diagram**

**Figure 4.8 – Proposed Use Case Diagram**

The suggested application's development is not solely dependent on the system's functionality. It also depends on the workflow procedure that needs to be identified, implemented, and followed. The components of the proposed application “CoviBlock: A Merkle Dag and Blockchain Implementation for Covid-19 records”, is demonstrated in Figure 4.x and utilized a Use Case Diagram. The patient, being the central user of this system will provide appropriate keys with reference to the executing process. These in turn can trigger uploading or granting of view access to either medical unit or a third party.

**4.2.4 Transactional Operation Diagram**



**Figure 4.9 – Transactional Operation Diagram**

Figure 4.9 illustrates the operations that exist in the proposed application. It is divided according to the users triggering the process (A. B. C). The crucial process of generating the private and public keys will be prompted by the patient. Without these keys, medical personnel cannot upload files which in turn, the third parties will not be able to request any files.

**4.2.5 System Flowchart of the Proposed Application**



**Figure 4.10 – Key Generation Process Flowchart**



**Figure 4.11 – Third party access to patient file Flowchart**



**Figure 4.12 – File Uploading Flowchart**



**Figure 4.13 – File Retrieval Flowchart**

For the key generation process illustrated in figure 4.10, once the user generates keys it will then go to separate storages. Public key will be saved to the application database while private key will be the user’s responsibility to store securely. Figure 4.11 details how a third party can request for files. Logs from the blockchain will be displayed to the third party that has information about files available for viewing. They will choose the file they want to access, and a request will be sent to the patient. The application will notify the patient that a request is sent to access their files and will be asked to provide keys. It will be patient’s discretion if they will grant the request. If patient agrees, they will provide the private key to be used in decrypting the file requested.

Figure 4.12 illustrates how the application will handle uploading of files. Authorized medical personnel will trigger the upload. The application will check if the patient being referenced by the record has an existing public key. If yes, it will proceed on encrypting it using the key and uploading the encrypted file to IPFS. IPFS will generate a hash of the uploaded file. This hash will then be stored as a transaction in the blockchain. Figure 4.13 shows how the application will handle retrieval of files. The patient will provide the private key to enable decryption of files retrieved from IPFS.

**LIST OF REFERENCES**

Tsai, Jack and Bond, Gary (2007). *A comparison of electronic records to paper records in mental health centers*

Khalifa, Mohamed (2018). *Perceived Benefits of Implementing and Using Hospital Information Systems and Electronic Medical Records*

Ebardo, Ryan and Celis, Nelson (2019). *Barriers to the Adoption of Electronic Medical Records in Select Philippine Hospitals: A Case Study Approach*

Ebardo, Ryan and Tuazon, John Byron (2019). *Identifying Healthcare Information Systems Enablers in a Developing Economy*

Gesulga, Jaillah Mae; Berjame, Almarie; Moquiala, Kristelle Sheen; Galido, Adrian (2017). *Barriers to Electronic Health Record System Implementation and Information Systems Resources: A Structured Review*

Merkle, Ralph (1989). *A Certified Digital Signature*

Benet, Juan (2014). *IPFS - content addressed, versioned, P2P file system (draft 3)*

Maymounkov, Petar and Mazieres, David (2002). *Kademlia: A peer-to-peer information system based on the xor metric*

Freedman, Michael J.; Freudenthal, Eric; Mazieres, David (2004). *Democratizing content publication with Coral*

Baumgart, Ingmar and Mies, Sebastian (2007). *S/kademlia: A practicable approach towards secure key-based routing.*

National Institute of Standards and Technology (2002). FIPS-180-2: *Secure Hash Standard (SHS)*

Yoshida Hirotaka and Biryukov, Alex (2005). *Analysis of a SHA-256 Variant*

Gilbert Henri and Handschuh, Helena (2004). *Security Analysis of SHA-256 and Sisters*

Menezes, Alfred; van Oorschot, Paul; Vanstone, Scott (1997). *Handbook of Applied Cryptography*

Sklavos, Nicolas and Koufopavlou, Odysseas G. (2003). *On the hardware implementation of the SHA-2 (256, 384, 512) Hash functions*

Szydlo, Michael (2003). *Merkle tree traversal in log space and time*

Micali, Silvio; Jakobsson, Markus; Leighton, Tom; Szydlo, Michael (2003). *Fractal merkle tree representation and traversal.*

Xie, Ming (2003). *P2P Systems based on Distributed Hash Table*

Szabo, Nick (1997). *Smart contracts: formalizing and securing relationships on public networks.*

Yaga, Dylan; Mell, Peter; Roby, Nik; Scarfone, Karen (2018). *Blockchain Technology Overview*

Abadi, Joseph and Brunnermeier, Markus (2018). *Blockchain Economics*

Zheng, Zibin; Xie, Shaoan; Dai, Hong-Ning; Chen, Xiangping (2017). *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*

Monrat, Ahmed Afif; Schelén, Olov; Andersson, Karl (2019). *A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities*

Fujisaki, Eiichiro and Okamoto, Tatsuaki (2011). *Secure Integration of Asymmetric and Symmetric Encryption Schemes*

Bellare, Mihir; Desai Anand; Pointcheval, David; Rogaway, Phillip (1998). *Relations among notions of security for public-key encryption schemes*

Goldwasser, Shafi and Micali, Silvio (1984). *Probabilistic encryption*