Blockchain: An Exploration into Basic and Alternate Applications

Adam Spanier and Jose Bibiano-Chavez

1. Introduction

While blockchain technology and applications currently exhibit fad-like characteristics based on a high level of misunderstandings surrounding blockchain primitives, the basic concepts espoused by blockchain continue to prove themselves generally robust. Though blockchain applications, in most current iterations, trend toward concepts like cryptocurrencies and Non-Fungible Tokens, there still remain a large number of potential applications suitable for blockchain technologies and the benefits yielded by their subsequent use.

In general, these potential applications derive themselves from the same basic cryptographic primitives that make blockchain useful. Most notably, cryptographically secure hashing algorithms and digital signatures [1] provide, “some way to control supply and enforce various security properties to prevent cheating” [1]. Such properties find themselves beneficial, not only in cryptocurrencies, but also in any application where security properties like integrity and confidentiality must be guaranteed, verified, and validated. In particular, cryptographic hashing algorithms help ensure the validity and integrity of a given blockchain, while digital signatures “seal” the chain.

While some proponents of blockchain may advocate the inherent advantages of its use, there still remain several issues surrounding the use of blockchain. The most significant issue concerning all blockchain applications yet remains the *double-spending* problem. Double spending is the practice of, “signing-over the same coin to two different users” [3]. In order to mitigate such a practice, Satoshi Nakamoto originated the idea of Bitcoin as a means to reduce the double-spending issues via the use of peer-to-peer networking and timestamp validations [2].

While double-spending mitigations find themselves currently applied broadly across most blockchains, the need for consensus among peer-to-peer nodes results in a secondary issue. Given a non-deterministic consensus based on algorithms used in blockchains like Bitcoin and Ethereum, double-spending can still be accomplished should the same coin be signed over before consensus can be achieved. Called the *Blockchain Anomaly* by [4], such an issue arises when, “existing blockchains do not ensure consensus safety deterministically; there is no way for Bob to make sure that Alice actually sent him coins without Bob using an external mechanism” [4]. By capitalizing on inherent characteristics of the peer-to-peer architecture, double-spending can still be accomplished, even if patent double-spending migrations exist.

The final, and possibly most problematic, issue concerning blockbusting lies in trust. While [2] indicates that purely trust based models do not adequately serve digital blockchain applications as a third party must validate transactions, trust within the operation of the blockchain system itself must still be assured. While Nakamoto [2] does mitigate the need for third party involvement in all transactions, he mitigates blockchain issues with the “51%” rule. This creates the potential for blockchains to be hijacked and replaced with invalid transactions.

As such, many benefits can still be gleaned from the application of blockchain in alternative designs. One such design explores blockchain in Farming [5], [6] proposes the use of blockchain in Human Resources Management, [7] extrapolates a blockchain system for Supply Chain Management, and a novel use of blockchain in medical record integrity and sharing is outlined in [8].

This project proposes the use of a novel blockchain application within an intranetwork

environment such that blockchain principles can be applied to internal packet sharing, management, detection, and identification. The goal of this research is to: (1) research blockchain fundamentals, (2) understand blockchain basics, (3) outline blockchain concepts, (4) analyze blockchain strengths and weaknesses, (5) propose a potential intranetwork based blockchain within a Software Defined Network, and (6) observe the outcomes.

2. Objectives and Methodology

2.1. Overview

The fundamental principal involved in blockchain lies in blocks of data which are verified by all nodes within the network. Blockchain is designed to be decentralized, transparent, and tamper proof [9]. We will go over how blockchain achieves each of these functionalities. A block is represented as a hash value, mining is the action of an individual finding this hash value. The block consists of four main values. First is the current transaction being recorded in the block [9]. Second is the nonce, a number that is used only once [9]. Third is the timestamp of the current transaction. Lastly, the block includes the hash of the previous transaction in the block [9]. Due to the decentralized nature of blockchain, to add a new block, all nodes within the network must first validate and agree the block is legitimate achieving decentralization [9]. Without consensus the block is discarded, thus blockchain achieves to be tamper-proof. Each node of the network will have the same copy of the transactions which leads to transparency.

Popular crypto currencies that run on block chain, like bitcoin, use unstructured peer-to-peer (P2P) networks based on TCP as the main basis of the communication network [9]. An unstructured P2P network organizes clients and utilizes various techniques to locate peers with possible data. Network nodes maintain at least 8 non-encrypted connections with a maximum of 125 [9]. When peers connect, the socket is initiated by an application layer handshake. This handshake will include synchronization time, IP addresses, and protocol versions. Peer selection by the node is random and a new peer is selected after a set amount of time. Also, every 30 minutes a message is sent to keep the connection open until a new node must be chosen. This will help avoid vulnerabilities where the attackers can create inconsistencies between the views of the network [9]. Using an unstructured network helps the block chain quickly distribute information to every part of the network.

2.2. Objectives

This research intents to reach the following objectives:

- Research and document the fundamental principles involved in blockchain
- Asses common functions and problems relative to blockchain implementations
- Observer existing alternate implementations of blockchain
- Design and model a theoretical blockchain based network communication protocol
- Attempt to implement a basic prototype for the proposed model.

2.3. Methodology

To reach the above stated objectives, the following procedure will be followed: (1) research documentation will be gleaned from journal and textbook publications, (2) data concerning blockchain will be synthesized and documented, (3) a case will be made for the implementation of a blockchain based network protocol, (4) basic requirements will be established, (5) a model proposed, and (6) a prototype developed.

3. Background

At the core of any blockchain lie two cryptographic bulwarks: (1) cryptographically secure hashing algorithms and (2) digital signatures [1]. based on these two primitives alone, a basic blockchain can be built in the form of a hash pointer based linked list [1]. Upon this linked list, called the Blockchain [1], other elements are incorporated. These elements include but are not limited to: (1) the Proof of Work mechanism [2], (2) timestamping [2], and (3) consensus b methods [4]. As a means to better understand alternate applications and potential techniques, a brief background into these basic functions is carried out below.

3.1. Basic Functions

3.1.1. Cryptographically Secure Hashing Algorithms

A hashing algorithm is any algorithms that takes in a variable number of bits *k*, produces a fixed number of bits *n*, and is computationally efficient [1]. While this definition is useful, more definition is needed for a hashing algorithm to be considered *cryptographically secure.* According to [1], a hashing algorithm must exhibit three (3) distinct characteristics to be considered *cryptographically secure:* The algorithm must (1) be collision-resistant, (2) exhibit hiding, and (3) be puzzle-friendly.

A *collision* is said to occur when two different inputs m_1 and m_2 result in the same hashed output h_1[1]. For a hashing algorithm to be c*ollision-resistant*, the algorithm need not be collision free. In fact, due to the nature of hashing algorithms producing a fixed bit output from a potentially infinite input space, there must, in theory, be an infinite number of possible collisions. Rather, *collision-resistant* simply indicates that collisions are *difficult* to discover. For an output space of 256 bits, the calculation of 2^{128} outputs should, with a probability of 99.8% [1], detect a collision, but, even with a sufficiently fast computer, such a number of calculations (3.402823669×10^{38}) impedes the feasibility of such a process. Due to the inherent difficulty of this process, a hashing algorithm like SHA-256 is considered *collision-resistant.*

Hiding indicates the level of randomness in the output space of a given hashing algorithm [1]. Should the output maintain the appearance of random distribution, no information can be gleaned from the input of the algorithm. In this case, the algorithm is said to be *hiding.*

Puzzle-friendly refers to the *birthday paradox* and aims to ensure that, at the *very least*, $\sqrt{2^n}$ calculations must be carried out on an *n* bit output to find a collision [1].

3.1.2. Digital Signatures

A digital signature serves as a means to sign digital information much like analog, hand-written signatures are used to seal the consent of an individual to a given document. In [1], a digital signature must exhibit two primary traits: (1) only the entity signing can make the signature and (2) the signature *must* be tied to a particular document.

In order to formalize these properties, digital signature schemes are used to both sign documents and validate signatures. To sign a document, a public/private key pair must be generated, and the subsequent secret key is used to generate a randomized output based on the document being signed. This output indicates that: (1) the holder of the secret key used the secret key to sign a very specific document. Any changes to either the secret key or the document will result in a different signature. To validate the document, the process need only be replicated, and the resulting signature compared to the signature associated with the document [1].

3.1.3. The Block Chain

In *Bitcoin and Cryptocurrency Technologies [1]* a block chain is described as a linked list using hash pointers. The important distinction between a simple linked list and a blockchain lies in the use of hashed pointers versus simple memory location pointers. [1] defines a hash pointer as, “a pointer to where data is stored together with a cryptographic hash of the value of that data at some point in time” [1]. This data structure supplies a means to fix the state of a given system at a fixed point in time. Because of the hashing carried out in the linked list, the integrity of each value can be assured based on a comparison between the value and the hash. While this doesn’t ensure the integrity of a given link, it does provide the valuable first step to sealing a blockchain.

3.1.4. Proof-of-Work

The Proof-of-Work system involves the use of some function to prove that sufficient energy and computations have been carried out to generate or “mine” a given block. According to Satoshi’s White Paper, “the proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the has begins with a number of zero bits” [2]. In such a case, the work needed to *find* the correct value is exponentially harder than the amount of work needed to *verify* such a value. This effort ensures that blocks aren’t simply mined in one chuck, thus rendering such a block chain application useless. Further, the number of zeros can be augmented to increase the difficulty of mining given the number of nodes mining on the network. In [2], this proof-of-work is also the means by which network consensus is established via the “one-CPU-one-vote” model.

In practice, the proof-of-work system eliminates the need to pick a random node to propose the next block. Instead, it allows all the nodes to compete independently of each other for the right to propose the next block [1]. This streamlines the production of blockchain nodes, while eliminating a centralized mechanism for randomized propositions.

3.1.5. Timestamp Servers

While hashing and digital signatures go a long way to protecting the integrity of a given blockchain, time stamping finalizes the linear nature of a given block chain by fixing *in time* the creation of transaction documentation [1]. In general timestamping services, clients send information to timestamp servers. “When the server receives a document, it signs the document together with the current time … and issues a ‘certificate’ with this information” [1]. By assuring the linear creation of blockchain elements, a linear progression of transactions can be generated such that a coin cannot be respent after being signed to another user.

3.1.6. Consensus

During decentralized mining of blockchain blocks, the concurrent mining of the same block is likely to occur. In order to determine which node legitimately keeps the block, consensus algorithms, specifically *distributed consensus algorithms*, are used to determine which node, given a collision, is the rightful owner of a mined block. Each consensus algorithm, “guarantees a total order … so that the chin does not end up being a tree” [4].

According to [1], a *distributed consensus protocol* is a protocol over *n* nodes, each with an input value, some that are malicious or flawed, and exhibits the following properties: (1) it must eliminate dishonest nodes and terminate with all honest nodes, and (2) the value at the core of the consensus algorithm must be derived from an honest node.

Consensus issues continue to plague a number of decentralized blockchain implementations that propagate discovered blocks via a non-deterministic methodology. Most notably, the Blockchain Anomaly facilitates the possibility of a double spending attack in a network where the consensus algorithm is non-deterministic, and the resulting consensus is susceptible to network interference, failures, and anomalies [4].

3.2. Common Problems

Due to the inherently novel nature of many blockchain applications, there yet remain a significant number of common problems that plague most blockchain systems. The three most considerable problems include: (1) double-spending, (2) consensus failures, and (3) trust. By analyzing these issues to a greater depth, a better understanding can be achieved such that more potential applications can be gleaned.

The first issue is the double-spending issue. Double spending is one of the most important problems that could arise when using blockchain. The problem is the end user is trusting an anonymous and unstructured network to create and conduct secure transactions. An end user must be confident that the goods being verified or traded are only done so as intended, usually this means once. Current blockchain networks mitigate this problem by using strong consensus algorithms that help ensure blocks are vetted before being added to the chain [10].

Second is the consensus issues. How do all nodes within the network come to a consensus that the data should be accepted? As mentioned previously, in cryptocurrency we rely on the strong consensus algorithms [10]. For example, we have proof of work and proof of stake. The proof of work consensus algorithm is used mainly for validating new transactions within the blockchain [9]. Currently bitcoin uses Proof of work, in the case of bitcoin, the consensus algorithm requires the hash to contain a certain number of zeros at the beginning to be

considered valid [9]. The miner accomplishes this by using different nonce values to produce the correct value for the consensus algorithm. Once the value is produces it is broadcasted to the whole network and the block is vetted [10]. These preceding zeros make is extremely difficult and computationally expensive to provide a valid proof of work [9]. Proof of stake will randomly assign the node that will mine or validate block transactions by the number of coins that are held. The more tokens a miner owns the more changes they will be assigned a coin to mine or validate. Proof of stake can lead to other increase chance of 51% attacks, which also incentivizes attackers to keep coins rather than use them [11].

The final and the most important issue is trust. As a network node, you must decide if a new node will be trusted or not. Within cryptocurrencies the trust issue is mitigated by a strong consensus algorithm [9]. Since all nodes are anonymous there is no central authority providing permission to interact with the blockchain network. Any node can connect and start sending and receiving data from other nodes. The strong consensus algorithm will approve legitimate data while denying all illegitimate data. In turn, the end users of the data are incentivized to continue producing legitimate data [9]. In alternate applications of blockchain it will be important for developers to find a good way to trust new nodes of the network. In a paper published on deploying blockchain in a software defined network (SDN) a private key is used to ensure the nodes are to be trusted [12].

4. Alternate Implementations

The idea and implementation of Blockchain is cutting-edge. With the level of disruption blockchain has caused to the financial sector, there are many researchers looking into novel applications. One example of two converging new technologies is the idea of blockchain enabled vehicle charging infrastructure [13]. The researchers propose an app that will help end users find and purchase electricity to recharge their electric vehicle. The transactions will be recorded within the blockchain network. With this data the developer aims to enable the government and the private sector to act on much needed data. The data and network will help decentralize electric charging services [13]. More availability of electrical charging stations will allow more users to purchase electric vehicles. More data on how these stations are used and the decentralized aspect of the app will allow anyone who can provide charging stations to join the service.

Another possible application of blockchain is in the IoT space. IoT devices are in hard-to-reach places and usually insecure. With the implementation of blockchain it is possible to mitigate some of the vulnerabilities found within different types of IoT networks [13]. Researchers have proposed a network of IoT enabled vehicles that utilize blockchain to manage trust within a supply chain. Recent high profile supply chain attacks provide evidence of the need to secure the supply chain. With the blockchain enabled network regulators can then track the supplier, logistics, and destination. The model is specified for monitoring cooling, but this model can also be used in other industries.

5. Blockchain Intranetwork Model (BIM)

5.1. Introduction

As noted above, the strengths of blockchain lies in its ability to provide integrity and authentication via any linear progression of transactions. While observing possible alternative applications for such a system, network protocol transactions, notably at the L2 and L3 levels, exhibit many of the same application characteristics that best serve blockchain implementations. During Level 2 and 3 transactions within a network, mac addresses and the Internet Protocol are utilized to identify the endpoints of a transaction, and, upon the creation of a successful connection, split data into finite blocks of data and relay them to the proper location [14]. By considering such a set of transactions of data between clients similar to the transactions between blockchain users, a direct connection can reasonably be made between network communications and the transactions occurring via most crypt currencies.

With these similarities in mind, a model can theoretically be developed by which the communications between clients within a L2/L3 network can connect and share data via the implementation of a fundamental blockchain architecture. This Blockchain Intranetwork Model (BIM) is hypothesized to aid in three (3) significant areas: (1) non-repudiation, (2) malware and intrusion identification, and (3) proxy attacks, all of which remain predominant issues in network communications.

By using the basic function of digitally signing each packet of information, all packets can then be traced back to the originating point. Because each client has its own secret key that identifies itself to other clients, signing each packet as a function over the data within allows each packet to be triangulated to its owner. Not only would this allow for easy identification of origination of any set of packets, but it can also provide a much faster means by which to squelch identified malware attacks as they proliferate over the network. As such, Intrusion Detection Systems embedded into blockchain networks can monitor packets as they pass, not by the information within, but rather by the hash and the signature associated. This not only reduces computational efforts expended by the IDS, but also allows for more accuracy in targeting infected clients and identifying known malware proliferation pattern hashes.

Due to the inherent nature of collision resistance within a hashing algorithm, each packet and subsequent set of packets should form, in and of themselves, a unique set of linear hashes. When combined, these hashes can be observed and recorded as either normal traffic or malware. As a system comes under attack, and the danger is recognized, the packet proliferation pattern can be recorded and subsequently used to identify the same threat on the local network. Further, these patterns can be collected and curated into a heuristic database that each network can check for the most up to date malware proliferation pattern hashes.

Where blockchain implementations facilitate authenticated encryption and integrity verification within any system into which they are embedded, any packet that has been intercepted and subsequently modified within a BIM network can be recognized. By observing the value, the hash, and the signature of a given packet, any modifications of such a packet would generate inaccuracies when validated against a checking mechanism. While a centralized validation mechanism can be used to validate client conversations, the brunt of the work can be carried out locally within a client. Each client must possess the public key of the client to which they are communicating. By using this key to validate the resulting hash of the block, the client itself can ensure that the packets being received have not been modified.

5.2. Blockchain Requirements

In order to ensure that a Blockchain architecture is truly implemented within the proposed BIM model, a set of core blockchain requirements must be defined. Within the scope of this research, the Core Blockchain Requirements (CBR) include the following: (1) a blockchain must be used to facilitate packet transfer between clients, (2) each packet must contain the value, the hash of the value, a pointer to the previous packet, and a valid signature for the packet, (3) a local, intranetwork timestamping server will facilitate the linear progression of intranetwork conversations. The BIM proposed in this research must adhere to the below stated CBR so that, at the minimum, it's implementation can prove in a general sense that blockchain implementations in network protocols are *actually* useful.

As the namesake of the process, it is fundamental that a blockchain is used in the implementation of this BIM. As defined by [1], a blockchain is a "linked list using hash pointers." This blockchain, called the Concurrent Communications Blockchain (CCB), will be implemented in such a way as to satisfy the requirements laid out in [1] so as to meet the definition of a blockchain.

Within the CCB, each packet will be modeled in such a way that the internal blocks conform to a standard implementation of blockchains. [1] indicates that each block must contain, "data as well as a pointer to the previous block in the list" [1]. So as to conform with standard blockchains, this pointer will be replaced with the hash pointer of the previous block. This means that each pointer within the packet structure provides both the value of the preview block, but also the hash.

As the final requirement, a local intranet, timestamping server will be implemented within the internal network. The purpose of such a server is to create a ledger of linear transactions within a blockchain [1][2]. In the BIM proposed, the timestamp ledger would overflow quickly, and cause throughput bottlenecks should every packet generated be required to pass through the timestamp server. As such, the internal timestamping will require each *new* intraclient conversation to be timestamped and recorded in the ledger so as to validate the linear progression of all communications in the network. While this doesn't adhere to the blockchain model in the *strictest* sense, it creates a means by which all CCB's can be tracked back to the source.

5.3. Network Requirements

To best facilitate a network protocol based on blockchain, a set of basic network definitions must be established. As previously mentioned, the proposed BIM will exist at the L2 and L3 levels of the OSI model [14]. At L2 in the OSI hierarchy, intranetwork addressing via mac is used to identify local clients *inside* the gateway of the network [14]. Level 3 of the OSI hierarchy sees the broader addressing of clients and the division of data into packets for transfer. As such, the network into which the proposed BIM will be integrated must (1) exhibit basic L1-L3 OSI characteristics, and (2) conform to OSI Standard protocols for the transfer of data.

While the potential remains for a "from scratch" BIM that manages internal and external addressing and incorporates these addresses into the basic blockchain architecture, the BIM proposed in this research will instead be implemented *on top* of existing OSI protocols. While better implementations may exist outside of OSI standards, it is hypothesized that the use of

existing protocols will help adaptability and uptake of a new BIM system into legacy network applications.

5.4. Model

5.4.1. Overview

The BIM model is divided into two (2) distinct keying functions, each with a corresponding ledger, blockchains associated with inter-client communications, and several gateway functions. The two keying functions are: (1) the Authentication Process and (2) Client Key Mining. To carry out these functions the Gateway must: (1) initialize a one-time network-based hash for distribution to clients, (2) timestamp successful login blocks, (3) keep a Login Ledger Blockchain, (4) manage a Concurrent Communications Ledger, (5) determine Gateway Acceptance Parameters.

5.4.2. Authentication Process

The authentication process occurs when a new client attempts to join the network. Authentication is achieved by applying a SHA-256 hash to the concatenation an out of band password (distributed by the system administrator) with the automatically received network hash. The resulting hash *must* match the exact expected login hash validated by the gateway. Further, the login hash will always maintain at least 10 leading zeros. This subsequently small number makes brute force authentication mining computationally infeasible.

Upon successful authentication, the Gateway carries out the Timestamping Process. This occurs when a Gateway generated timestamp is concatenated to the incoming login hash and the client MAC address and then hashed through the SHA-256 algorithm. This resulting Login Block is sent back to the client in anticipation of the Client Key Mining Process and added to the Login Ledger.

The Login Ledger is a simple blockchain responsible for maintaining a record of successful login attempts. Due to the architecture of the Login Block, the time, hash, and client MAC are each recorded for validation at a later time.

This authentication process capitalizes on the differentiation between Proof of Work calculations and the validation processes that verify them. In this instance, should a malicious attacker wish to enter the network, the password would be considered a nonce that would need to be “mined” in order to generate a subsequently small hash. Further, even, should the miner achieve a hash with 10 leading zeros, the mismatch of the remaining 256 bits will cause an authentication denial. This process increases the difficulty of a brute force attack by a magnitude.

5.4.3. Client Key Mining

Upon the successful authentication and Login Ledger addition, a new client must carry out a Proof of Work to “prove” its entry into the network. This Client Key Mining process follows a standard Proof of Work format: a nonce is appended to the Login Block generated in the Authentication Process and subsequently hashed. This resulting hash is verified against the *Acceptance Parameter* automatically provided by the Gateway. When the resulting hash is less than the Acceptance Parameter, the resulting block is hashed and added to a Client ID Ledger.

The Client ID Ledger is a simple block chain that keeps a record of each unique client ID used in the Concurrent Communications Blockchain process outlined below. By mining this ID, each client proves by work its entrance to the network.

5.4.4. Acceptance Parameter

The *Acceptance Parameter* is a value calculated by the Gateway at a regular interval based on the average amount of traffic flowing across the network at a given point in time. This Acceptance Parameter will increase the difficulty of Client Key Mining as the traffic increases. As such, the Acceptance Parameter can be seen as a function of leading zeros over the average amount of packets flowing at the time of its calculation. The *Acceptance Parameter* is sent automatically to each successfully authenticated client.

Where n is the current number of users, the *Acceptance Parameter* is calculated as follows:

$$AP = \frac{2^n}{n^2}$$

5.4.5. Network Hash Generation

Each BIM enabled network will, at its inception, generate a semi-constant *Network Hash*. This value is generated via a SHA-256 hash over the Gateway Serial and MAC. Upon the initialization of the Authentication Process, this *Network Hash* is automatically sent to each client for use in Authentication. This Network Hash will change only when a new Gateway is embedded into the system.

5.4.6. Concurrent Communications Blockchain (CCB)

Upon successfully entering the network via the Authentication Process and Client Key Mining, each client can begin internal communications with other clients on the network. Each new conversation formed between clients will, after a successful TCP handshake, pass the session key to the Gateway for Timestamping. When the Gateway receives the session key, it concatenates a timestamp onto the key and hashes the resulting value. This hash is then added to the Concurrent Communications Ledger (CCL).

The CCL maintains a record of all unique conversations carried out between any two clients in time. This creates the unique ability for each conversation to be tracked back and validated at any point in time from the Gateway. While the CCL doesn't keep track of all packets generated in the conversation, it keeps a log of all generalized "transactions" between all clients.

After timestamping, the two clients will begin passing packets. These packets build a blockchain. The blockchain is modeled as a simple linked list of hashed pointers. Each packet contains: (1) the hashed pointer to the previous packet, (2) the hashed value of the packet data, (3) the packet data, and (4) a valid signature of the sender.

As these packets proliferate, possible functions can be built to monitor common hash proliferation packets. Due to the nature of each hashed transaction, malware proliferation should exhibit obvious hashes based on the values included in the hashed packet.

6. BIM Prototype

6.1. Implementation Overview

In choosing a code base for our implementation, we decided to work with Python. Our choice is derived from both familiarity with the syntax, as well as the existence of relevant code libraries. By working with Python, the ease of demonstrating a workable model in the timeframe allotted also increased the probability significantly of achieving some version of a prototype for demonstration purposes.

To demonstrate the model defined above, the processes involved were coded into a number of functions and classes and subsequently added together to form a linear, command line model demonstrating, at the very least, the most basic attributes described above. The prototype includes: (1) network hashing functions, (2) network key mining functions, (3) authentication processes, (4) client key mining functions, (5) acceptance parameter calculations, (6) Login Ledger extension and printing, (7) Client ID Ledger extension and printing, (9) the initialization and verification of the Concurrent Communications Ledger, (10) the initialization and validation of the Concurrent Communications Blockchain, and (11) substructures like loops and variables that model login attempts and client communications.

To run the demonstrations, two files must be executed separately. For the network password mining processes, authentication, timestamping, login ledger processes, and client ID processes, the mainDriver.py file must be executed. This file carries out the algorithms designed and the various supporting functions necessary. To run the Concurrent Communications Blockchain demonstration, the concurrentCommuncationsBlockchain.py file must be executed.

6.2. Source Code Overview

One of the more fundamental characteristics of this design is the means by which authentication occurs. Rather than choose a password that is subsequently hashed and stored for future authentications, the password for network authentication must be mined. This mining procedure is carried out by choosing a random, large integer, adding the network hash (derived from network characteristics) and checking for sufficient leading zeros. This number of leading zeroes can be set by administrators based on the security needs of the network. Should the leading zeroes be significant, network password mining can take a significant amount of time, however, this time is exponentially augmented when an attacker attempts to brute force authentication. The network must only find one hash that meets the number of leading zeroes required. An attacker, however, must find the *exact* hash discovered by the network password mining process. A random large integer is chosen so as to ensure the attacker does not know where to start the nonce. This necessitates every hacking attempt either choose a random integer as a nonce or start at zero and count up to find the correct hash. Password mining occurs in the minePassword.py code file.

Upon successfully mining a network password, authentication, timestamping, and login block generation are handled in the authenticate.py code file. In order to check authentication, the user enters the password mined above. This password is kept by the system administrator and distributed based on security protocols. Upon the submission of this password, the network concatenates the password with the network hash, hashes the result and checks the resulting hash with the authorization hash, stored from the mining process. Should the two match,

authentication is accepted. It is important to note, *the mined password IS NOT KEPT on the network in any form*. Due to the hashing calculation, the network only need store the resulting hash from the mining process. In this way, shadow files derived directly from passwords aren't needed. From this process, a Login Block is generated and passed into the Login Ledger process and the Client Key Mining process.

Upon a successful login, an acceptance parameter is calculated and passed with the Login Block back to the client mining process. This process occurs in the clientKeyMining.py file. Based on the number of current users, this acceptance parameter will force the client to mine harder and harder puzzles in order to generate an entrance ID. Upon the successful mining of an acceptable Client ID, the ID is added to the Client ID blockchain, These processes are carried out in the mainDriver.py code file.

Finally, the Concurrent Communications Ledger (CCL) and Blockchain (CCB) is demonstrated in the concurrentCommuncationBlockchain.py code file. This portion of code models a three client environment, the creation of a session, the CCL, and the sending of packets via the CCB. It is important to note that this file is simply a *demonstration* in real time of the concepts. While the use of TCP and L3 protocols is implied, no actual networking infrastructure was used for host to host communications.

A number of supporting code files were implemented to support the above described code. Blockchain.py outlines the blockchain and block classes used to build the blockchains and client.py is used to model clients via an ID and timestamp.

6.3. Code Repository

Source Code: https://github.com/Jbibianochavez/CYBR8410_Project/tree/main

6.4. Testing

During testing, the codebase was analyzed for both successful and unsuccessful login attempts. Failed login attempts reveal little data, while successful attempts are logged in the login ledger with MAC, Serial, and Hash values. Currently the test bed models the mining of the network password, 5 login attempts, displays the login and client ID ledger, and demonstrates the interactions of two clients within the context of the CCB in a network with three clients present.

7. Conclusion

Fundamental blockchain applications and their underlying primitives currently exhibit a great deal of refinement and sophistication. Cryptocurrenices and cryptocurrency exchanges are capitalizing on this fact by creating infrastructures around the agreed upon, well developed functionalities that currently frame most cryptocurrencies. While the majority of blockchain applications fall into the financial sector via cryptocurrency, many researchers are attempting to find alternate applications of blockchain technologies. These alternatives fall in a range of industries including medical, agricultural, and industrial applications.

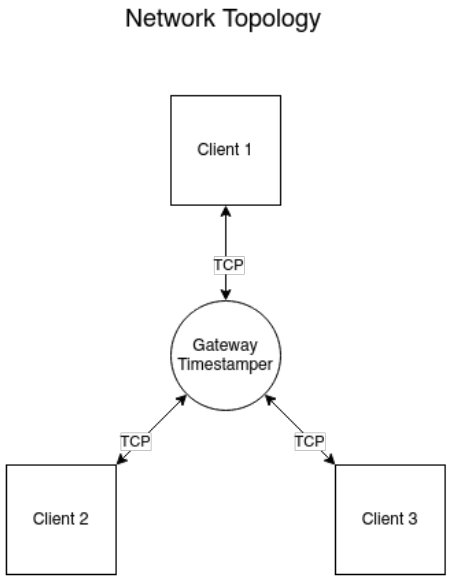
In an attempt to develop yet another alternative application to blockchain technology, this research developed blockchain internetworking model that offers a potential for more secure, predictable small network communications. This model appears adequate for application in small business or local area networks. By utilizing blockchain technologies to better fortify

authentication, identification, and internal communication routines, the model developed offers a novel approach to intranetwork computing.

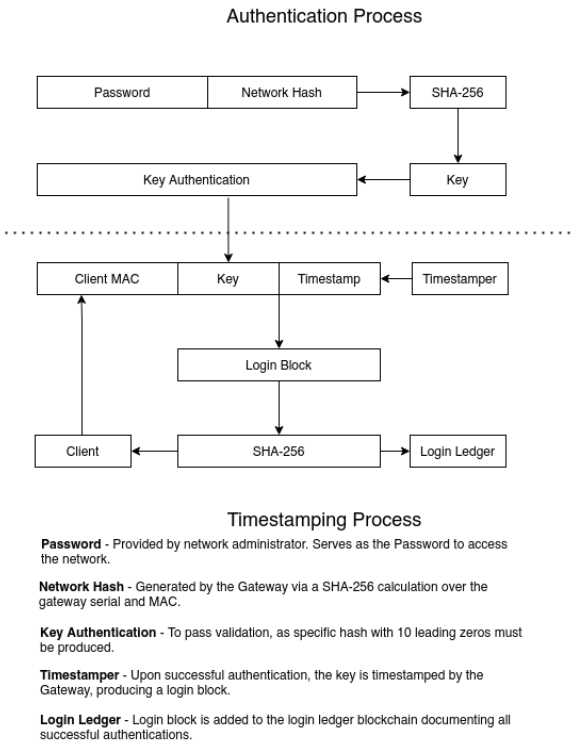
While further research is needed to understand just how robust the model may be, in the earliest phases, the model exhibits interesting characteristics that could potentially benefit network communications.

8. **Figures**

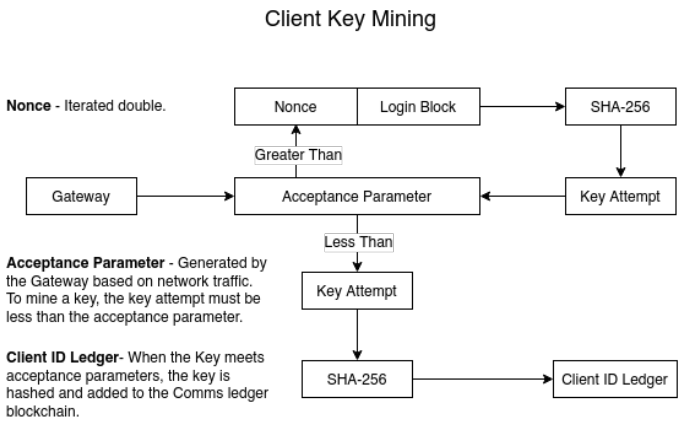
8.1. Network Topology



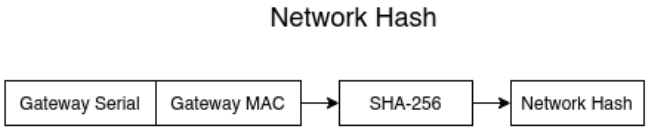
8.2. Authentication and Timestamping Processes



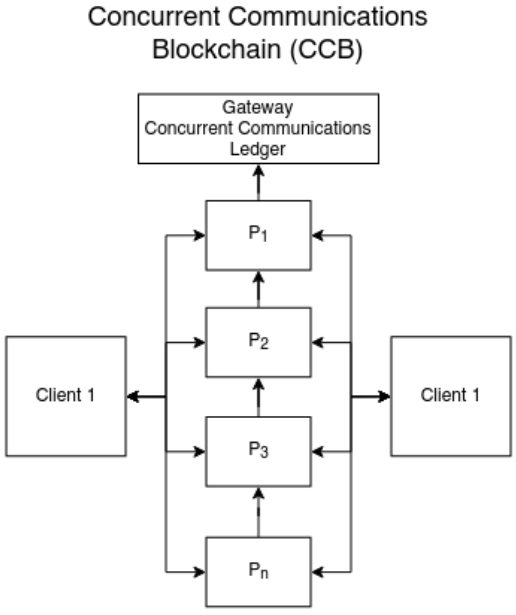
8.3. Client Key Mining



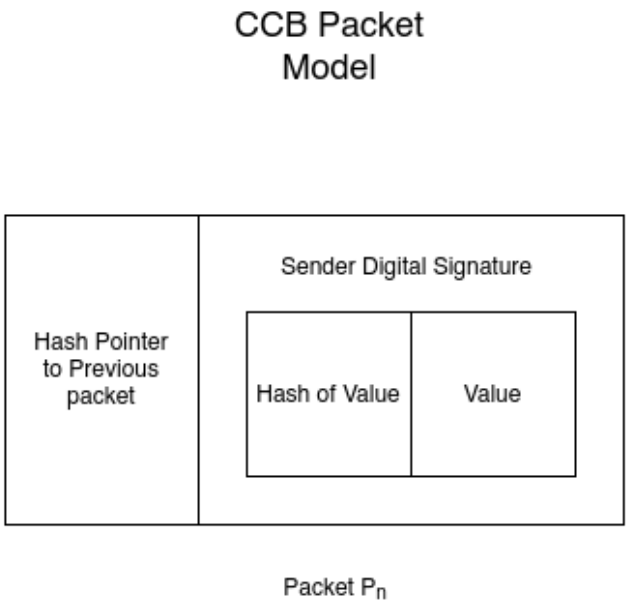
8.4. Network Hash



8.5. Concurrent Communications Blockchain (CCB)



8.6. CCB Packet



9. References

1. Narayanan, A. (2016). Bitcoin and cryptocurrency technologies: A comprehensive introduction. Princeton University Press.

2. Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

3. Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending fast payments in bitcoin. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). Association for Computing Machinery, New York, NY, USA, 906–917. DOI:https://doi.org/10.1145/2382196.2382292

4. Natoli, C., & Gramoli, V. (2016). The blockchain anomaly. 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA). https://doi.org/10.1109/nca.2016.7778635

5. Talreja, R., Chouksey, R., & Verma, S. (2020). A study of blockchain technology in Farmer's Portal. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). https://doi.org/10.1109/icirca48905.2020.9182969

6. Dina Salah, Maha Hafez Ahmed, and Kamal ElDahshan. 2020. Blockchain Applications in Human Resources Management: Opportunities and Challenges. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 383–389. DOI:https://doi.org/10.1145/3383219.3383274

7. Tan Guerpinar, Gilberto Guadiana, Philipp Asterios Ioannidis, Natalia Straub, and Michael Henke. 2021. The Current State of Blockchain Applications in Supply Chain Management. In *2021 The 3rd International Conference on Blockchain Technology* (*ICBCT '21*). Association for Computing Machinery, New York, NY, USA, 168–175. DOI:https://doi.org/10.1145/3460537.3460568

8. Sihua Wu and Jiang Du. 2019. Electronic medical record security sharing model based on blockchain. In Proceedings of the 3rd International Conference on Cryptography, Security and Privacy (ICCSP '19). Association for Computing Machinery, New York, NY, USA, 13–17. DOI:https://doi.org/10.1145/3309074.3309079

9. Conti, Mauro, et al. “A Survey on Security and Privacy Issues of Bitcoin.” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018, pp. 3416–3452., https://doi.org/10.1109/comst.2018.2842460.

10. Yang, Xinle, et al. “Effective Scheme against 51% Attack on Proof-of-Work Blockchain with History Weighted Information.” *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, https://doi.org/10.1109/blockchain.2019.00041.

11. Nair, P. Rajitha, and D. Ramya Dorai. “Evaluation of Performance and Security of Proof of Work and Proof of Stake Using Blockchain.” *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021, https://doi.org/10.1109/icicv50876.2021.9388487.

12. Alharbi, Talal. “Deployment of Blockchain Technology in Software Defined Networks: A Survey.” *IEEE Access*, vol. 8, 2020, pp. 9146–9156., https://doi.org/10.1109/access.2020.2964751.

13. Malik, Sidra, et al. “Trustchain: Trust Management in Blockchain and IOT Supported Supply Chains.” *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, https://doi.org/10.1109/blockchain.2019.00032.

14. Miller, Rachelle. 2021. “The OSI Model: An Overview” Sans Technology Institute. https://sansorg.egnyte.com/dl/oyHUTHkIzn