FINAL YEAR / UNDERGRADUATE PROJECT 2019/20

School of Electronic Engineering and Computer Science

INTERIM REPORT

Programme of Study: BSc Computer Science

Project Title: BlockMail – A blockchain-based Email System

Supervisor: Professor Steve Uhlig

Student Name: Thomas Daniel Herring

Date of Submission: 28/11/19

ABSTRACT

Over the past ten years, blockchain has gained widespread attention throughout the world. It is being tested in a number of use cases in various fields, providing reliability, security and fault tolerance. This paper will introduce a blockchain-based email system, *BlockMail*, which takes advantage of blockchain technology in order to provide a network of intercommunicating nodes. The overall aim is to lead to the elimination of trust in central parties, and migrating this trust to the implementation of the blockchain itself.

CONTENTS

CHAPTER 1: INTRODUCTION

A blockchain is an immutable, continuously growing distributed ledger which is made up of "blocks" containing transactions. These "blocks" are linked together using cryptographic hashes of their previous blocks. All blocks stem from a genesis block, which is the start of the blockchain (see appendix A). They are decentralized systems with extremely high Byzantine fault tolerance[1] – that is, they are still able to operate properly even if some of the nodes fail, or are modified in an unexpected way.

1.1 BACKGROUND

With the launch of Bitcoin in 2008, following Satoshi Nakamoto's report: "Bitcoin: A Peer-to-Peer Electronic Cash System"[2], the world saw the first real application of blockchain networks. Many other cryptocurrencies followed, driven by its success. Over the coming years, the benefits of blockchain technology surfaced, which lead to the development of many other non-monetary applications of blockchain.

The possibility of a blockchain-based email system is of particular interest as it removes the trust which users place in central parties as they do when using email services from a conventional provider. By using blockchain, its distributed nature and block signing mean that users would instead place their trust in the inherent security that is delivered by this type of system. As the system gains nodes, it would become more reliable and resilient per node added.

1.2 PROBLEM STATEMENT

Although there are a couple of other attempts at blockchain-based email systems: Cryptamail[3] and Ledgermail[4], both are nearly completely undocumented. The latter seems to be somewhat active but makes use of a Private blockchain, since it "requires an invitation and must be validated by either the network starter or by a set of rules put in place by the network starter" (Jayachandran, 2017)[5]. Moreover, it has a "Pricing" section on its website, indicating that it costs money to use.

The above clearly shows that there is a gap for a free, open, "Public" blockchain-based email system. A "Public" blockchain would ensure full transparency of emails (transactions) that pass through the network and will allow for analysis by node and by the user.

1.3 AIM

This project aims to explore an alternative application of blockchain networks, by creating a decentralized email system with a direct focus on reliability, traceability, and security. The primary goal is to understand how one may apply the reliability of such a network to create a system that is, theoretically, interference proof, and to create a series of interlinking applications which will demonstrate this.

It will pull on aspects from elliptic curve digital signature algorithms (ECDSA), Rivest–Shamir–Adleman (RSA) public-key cryptography, and will also heavily feature networking (in particular, from the distributed systems field) in order to facilitate node communication, discovery and blockchain distribution. The system should be usable on any device with internet connectivity, as well as a web browser which supports JavaScript.

1.4 OBJECTIVES

There are various mandatory and optional requirements for this project. Depending on time constraints, some, or possibly all of the optional requirements may be implemented in order to make a more complete, usable final system.

From the problem definition, the majority of objectives have remained the same. Some descriptions have been amended, and a few additional mandatory and optional objectives have been added.

1.4.1 MANDATORY

- Operations which involve cryptographic functions (for example key generation and key validation) should take place on the client-side as to ensure private keys are not exposed to any server at any time. This will ensure that the user can be sure that their mail can only be read by them, and anyone else who possesses their public / private key pair.
    - Users will "sign-in" with their ECDSA Public and Private keys.
- Every user will have a total of 3 keys. Each of these will be provided in the form of .key files.
    - ECDSA Public Key (Logging in).
    - ECDSA Private Key (Logging in).
    - RSA-2048 Private Key (Decrypting mail).
- A series of master nodes are to exist. These will be hardcoded into the BlockMail node client and will be responsible for processing emails, or passing the emails on to other nodes for processing if they are not selected to do so. There will be 8 of these, to begin with, although this is easily expandable. For testing, the number will be reduced.
- Individual emails will be separated into "Blocks". Blocks will be generated at a fixed time interval and will contain all emails sent during that said interval. This interval will start low and will increase as network usage expands.
- Each email is to be broadcast to the entire network. At the end of the fixed time interval, each node will measure the size of all other nodes block (size being the number of emails in the block) and will elect the block with the biggest size to be added to the BlockMail Blockchain. In the event two have the same size, the first will be added.
- Every node on the network will know 8 other nodes. This may be reduced for testing.
    - Upon connecting to the network, the new node will contact master nodes in order to be directed to other nodes which will become the new nodes "neighbours".
    - This leads to exponential email broadcasting.
- Anybody may download the BlockMail node client and contribute to the network.
- Emails are to be encrypted with RSA-2048.
    - This will be facilitated by the second private key.
- Email addresses will be using elliptic curve cryptography. Specifically, ECDSA SECP256K1 will be used, as this is what Bitcoin uses, so there is already large amounts of information available regarding it.
- The user interface will be web-based. This means that any device with a web browser with JavaScript compatibility will be able to use the system.
- The first transaction of every address on the network is to be that which contains its public key.
    - When sending an email to an address, the recipients public key will be looked up, in order to encrypt the email so only they may view it.
- The front-end web interface should be responsive. This will ensure that it functions correctly on a large variety of devices.
    - This will be achieved using bootstrap and CSS media queries effectively.

1.4.2 OPTIONAL

- A load balancing system which will ensure that less powerful nodes do not get overwhelmed with emails from the network.
    - A page which allows the user to see the load on each network node.
- A page which allows the user to view the live Blockchain.
    - Every email on the network (will be encrypted thus unreadable).
    - The number of times each node has been "awarded" the win for the largest block.
    - Each email within the blocks.
    - Every email sent and received from each address.
- A "confirmation" system. This would report the number of blocks which have passed since each email. With each block passed, the validity of the email becomes more so.
    - An option upon sending the email to choose the number of required confirmations before displaying to the recipient.
- An option to deactivate "accounts".

1.5 REPORT STRUCTURE

Further to this Introduction, in which I outlined the project's background, aims, objectives, and research questions, there are a number of other sections in this report:

- **Introduction –** This section gives an overview of the project and outlines its key aims and objectives.
- **Literature Review** – I will evaluate, critically analyse, and present conclusions from my research leading up to this report. I will include reviews of papers relating to characteristics and applications of blockchains, security in blockchain systems, and existing applications.
- **Analysis and Design** – Here I will outline the project requirements and provide some UML diagrams which describe the proposed system.
- **Risk Assessment** – Here I will summarize the risks that may occur in the course of development of this project, their likelihood, and will identify how they may negatively impact its outcome. I will also draft preventative measures for the aforementioned risks.
- **Presentation** – This section will provide a place to introduce the current design of the system, and recognise potential future development. I will provide code snippets of key parts of the system that have been developed so far.
- **Evaluation** – I will give an overview of the strengths and weaknesses of the project, and will explain what could be done in the future in order to produce a more effective solution. I will also consider different viewpoints on blockchains (and similar systems) and will convey these.
- **Conclusion** – Here I will provide a summary of the key points discussed in the report. I will explain how I believe the solution is effective and will back this up with the appropriate literature.
- **Appendices** – Various sections containing facts and figures that support the report.

CHAPTER 2: LITERATURE REVIEW

2.1 CONCEPTUALIZING BLOCKCHAINS: CHARACTERISTICS & APPLICATIONS - SULTAN, K; RUHI, U; LAKHANI, R[6]

2.1.1 INTRODUCTION

Karim Sultan, Umar Ruhi, and Rubina Lakhani published this paper in 2018 following the "recently gained widespread attention by media, businesses, public sector agencies, and various international organizations". With the continuous development of Bitcoin and other crypto-currencies, multitudes of people became interested in the underlying technology which facilitates their operation. Questions were beginning to arise about alternative applications of the blockchain, therefore this trio of researchers from the University of Ottawa, Canada, deemed it appropriate to provide a "discussion and classification of current and emerging blockchain applications".

Although the paper contains a wide range of concepts, in this review I will focus on five main points: The Conception of Blockchain, Trust vs Consensus, Types of Blockchain, The definition of Blockchain, and Blockchains as Trusted-Service Applications.

2.1.2 THE CONCEPTION OF BLOCKCHAIN AND ITS FEATURES

The "multi-disciplinary" nature of Blockchain required someone with vast knowledge in various fields to convey such a detailed description of their ideas. Satoshi Nakomoto's coining of Blockchain in his 2008 paper, *Bitcoin: Peer-to-Peer Electronic Cash System* served as proof of his undeniable intelligence. In their report, Sultan, Ruhi, and Lakhani outline four key areas of "software engineering, distributive computing, cryptographic science, and economic game theory" that Blockchain technology pulls from.

A key part of this report is the author's discussion of the Proof-of-Work Model, developed by Nakomoto. This is a concept which provides the true security of crypto-based applications of blockchain, by protecting against a double-spend attack. Such an attack is described as when a user "attempts to create a race condition in which they spend the same virtual assets twice". By using a random number (a so-called "nonce") to assign an unconventional cost of computing power to process transactions, Nakomoto was successfully able to prevent these attacks from occurring.

An alternative approach will be used by BlockMail: Proof-of-Stake. In such a model, rather than using computing power to compete for the block to be added to the chain, various aspects such as the block size or age are used – this is the "stake". In BlockMail, the stake will be defined by the size of the block; that is, the block with the biggest size at the end of a time interval will be elected to be added to the chain.

2.1.3 TRUST VS CONSENSUS

Quoting President Ronald Regan, the researchers draw parallels between blockchain and his famous proverb: "trust, but verify". Described as "trustless", in blockchains, no one node trusts another. Instead, they are backed up by mathematically secure hashing algorithms which provide the basis for the security of the network. It is this concept entirely which provides the unbeaten Byzantine Fault Tolerance of such systems.

The decentralized nature of Blockchain is at its "core" and is provided by keeping a "copy of the database file" on all participating nodes. Although the nodes are open to receiving information from each other, there is a "consensus algorithm" which provides the validation and authorization that is required to add transactions to the end of the chain. Of course, there are other factors that contribute to such systems, however, effective distribution and decentralization are important in ensuring near-perfect reliability.

The writers go on to discuss a concept discussed earlier in this report, that "traditional transaction models rely on a central authority to act in the clearinghouse role". Simply put – in traditional systems, users of such systems depend on the "central authority" to "remain honest while verifying and clearing transactions". In a blockchain, reliance is placed in no central party, thus removing the possibility of unexpected manipulation or disruption of transactions. If one node did attempt to do this, it would simply become marked as rogue by the network, and as a result, would be ignored.

2.1.3 TYPES OF BLOCKCHAIN

Earlier in my report, I briefly stated that there are two types of blockchain, one being Public, and the other Private. Although this is partly true, Sultan, Ruhi, and Lakhani add a further "Hybrid" (or "Consortium") blockchain.

In their paper, Sultan, Ruhi, and Lakhani describe Public blockchains as "open to all to participate in", and Private blockchains as using "privileges to control who can read and write to the blockchain". Moreover, they introduce a "Hybrid" blockchain which is only public to a "privileged" group, but copies of the blockchain are only "distributed among entitled participants".

This discussion is incredibly interesting in relation to BlockMail. By creating a public blockchain, it means that it can be viewed by everyone, however, encrypting the mail ensures that it can only actually be read by the intended recipient. This provides a type of blockchain implementation that does not definitively fall into any of the three types discussed in the literature, but rather defines an entirely new category: "semi-public". That is, although the underlying transactions and operation of the blockchain are public (and can be viewed by anyone who wishes to do so), each transaction's actual contents are only readable by the "privileged" user who possesses the key required to open them.

2.1.4 THE DEFINITION OF BLOCKCHAIN

As a type of system that has seen the application to crypto-currencies more than anything else, it is important to note that many definitions given for blockchain are specific to that application. For example, the definition provided by Coinbase (a crypto-currency exchange) "does not account for the fact that blockchains can be reused for other cryptocurrencies and industry applications independently". Therefore, the writers of this paper felt it appropriate to create a comprehensive and cohesive definition that could be used when talking about a range of applications of blockchain. The following is provided in the literature:

Blockchain *– "A decentralized database containing sequential, cryptographically linked blocks of digitally signed asset transactions, governed by a consensus model."*

This definition provides an excellent outline of the primary “constituents” that make up blockchain networks: decentralization, blocks, cryptography, and consensus models. Going forward, this will be assumed to be the definition of blockchain throughout the project.

2.1.5 ARE BLOCKCHAINS SUITABLE IN TRUSTED-SERVICE APPLICATIONS?

As clear supporters of the technology, the three authors of this report believe that blockchain-based systems are “the future”, stating that it facilitates “highly specialized applications for any purpose imaginable”.

In systems that involve governments, or require extensive security, it is highlighted that it is important to consider both “access” and “scope”. They in particular talk about blockchain in health care, providing an example in which health records are stored in a blockchain, which has a scope that is “private to health care partners”. They contrast this with a further example for the real estate industry which needs to be “open and transparent to the public”.

The trio concludes that their envisioned use cases are “poised to create a global decentralized yet trusted value ecosystem that can lead to exciting new economic opportunities in the public and private sectors alike”.

2.1.6 CONCLUSION

This paper provides some extremely useful insight into blockchain and its potential uses. From these, it is possible to draw similarities to BlockMail and to progress in the outline of the project.

Through this paper, I have identified a further type of blockchain (“semi-public”), and have also obtained a new definition of blockchain that covers all applications and can be applied going forward. Furthermore, the discussion about Trusted-Service Applications is very intriguing, as it provides some clarity from another party about usage in security-focused applications, and has caused thoughts to arise surrounding the potential application of blockchain in such an application.

Although there are many positives surrounding the paper, it is also important to identify its shortcomings. For example, despite giving a detailed discussion of Proof of Work, the researchers only briefly mention other methods of preventing double-spending attacks. It would have been useful to see a bit more written about these in order to understand how blockchain networks can be customised beyond Satoshi Nakomoto’s specifications.

To Summarize, I feel that this paper provides an excellent introduction to blockchain technology, slowly stepping through the origins, to advanced and speculative aspects relating to possible future applications. It has a high density of information that has allowed me to analyse various different parts in significant depth and detail and has allowed me to compare whether BlockMail is in line with what people “want” from blockchain.

2.2 (INVITED PAPER) ON THE SECURITY OF BLOCKCHAIN CONSENSUS PROTOCOLS – DAS, S; KOLLURI, A; SAXENA, P; YU, H[7]

2.2.1 INTRODUCTION

Similarly to the first paper in my review of the literature, this one was also published in the height of Blockchain media coverage and speculation. However, it takes a far more technical approach and dives into a detailed analysis of the security of blockchain technology – a key part of the BlockMail project. From it, I hope to ascertain some details of the actual workings of security in existing (larger) blockchain-based networks, so that, I can apply these to the implementation of my project.

The core points of focus I would like to take are: The Nakomoto Consensus and the Threat Model and Security Properties of Blockchain Consensus Protocols.

2.2.2 THE NAKOMOTO CONSENSUS AND THE THREAT MODEL

Developed by the previously mentioned Satoshi Nakomoto, the Nakomoto Consensus is the key security protocol used on the Bitcoin network. It relies on a set of "miners" that are connected via a peer-to-peer network, on which they "agree on the state of a globally distributed ledger of transactions periodically". The report goes on to discuss how transactions are broken into "sets of constant size called 'blocks'". More generally, it continues: "the essence of the blockchain consensus protocol is to reach agreement on the total order of a common subset of blocks seen by all the honest miners" – in layman's terms, all miners that are good work together to check on each other, in order to create blocks that are provably valid.

The Threat Model plays a key role in the Nakomoto Consensus. This paper defines "honest peers" and (implies) "dishonest" peers. The so-called "honest peers" are those who "follow the prescribed protocol". It continues to explain how Nakomoto's protocol makes three assumptions that "strikingly differ from prior literature", summarized these are:

a. New "honest peers" can broadcast a block publicly to other honest peers within a delay (δ).
b. The computational power of the blockchain is approximately known. A known fraction of this is assumed to be malicious (f).
c. All nodes have a source of randomness that is non-biased. The trusted setup phase creates a "genesis block".

Although BlockMail will not be using a "miner" based consensus algorithm, the above are all principles that can be applied to it, both for analysis and operation. By implementing the detection and handling of dishonest nodes, the system would be more secure from attack. Point (b) would provide transparency for users in being able to see an estimation of the network that is compromised. Moreover, point (a) describes the same broadcast mechanism discussed in the Introduction.

2.2.3 SECURITY PROPERTIES OF BLOCKCHAIN CONSENSUS PROTOCOLS

The literature defines three key factors that contribute to the security of a blockchain consensus protocol: "Stability", "Agreement", and "Fairness".

The first, “stability”, is in relation to confirmed blocks. The literature gives the following: “the set of confirmed blocks at output time t_1 is a subset of the set of confirmed blocks at time t_2 w.h.p, if $t_2 > t_1$”. Essentially, each new block contains (by linking to it in the chain) its previous.

The next, “agreement”, states that if C_1 and C_2 are the “set of confirmed blocks reported by any two honest miners”, then, with high probability, “either $C_1 \subseteq C_2$ or $C_2 \subseteq C_1$” and “the blocks in $C_1 \cap C_2$ are ordered identically by both miners”. Simplified, this means that each node must agree on the set of confirmed blocks, and the intersection of the confirmed blocks must be in the same order (as the order in blockchains is key).

Finally, the definition of “fairness” is given as “There is a negligible probability that the fraction of blocks proposed by the adversary in the set of confirmed blocks, over any time interval $t > c * \delta$, for some constant *c*, is more than f”. Breaking this down:

- δ = Block broadcast delay (see 2.1.2, a.).
- f = The fraction of the network known to be malicious (see 2.1.2, b.).
- c = Block requirement (time).

In this section, the writers go on to discuss how a lower value of *c* is preferable, as they “sample from the computational power distribution in the mining network frequently”. Although computational power will have no direct effect on the BlockMail network as Proof-of-Stake will be used, a low value of *c* will still be chosen in order to achieve a similar goal – timely delivery of mail. A high value of *c* would mean that users would have to wait 1 minute, 2 minutes or 10 minutes, for example. However, if *c* was something like 10 seconds, mail would be delivered within a much more reasonable time (and would get much more quickly confirmed).

2.2.4 CONCLUSION

Although in this report there were fewer points that I felt were appropriate to review, the ones which I did select have allowed me to understand in-depth how different constants can affect a blockchain network. This will give me the ability to ensure that during development I make the correct design choices in order to ensure an efficient and secure network. I have been able to draw comparisons between my vision for BlockMail and Nakomoto’s visions for Bitcoin given his consensus algorithm.

As with the first review, it would have been interesting to see a little more information regarding Proof-of-Stake, but as a lesser utilised approach this is to be expected. It also is not entirely necessary as the theory behind both Proof-of-Work consensus algorithms and Proof-of-Stake ones are very similar. Furthermore, it may have been useful to demonstrate a lot of the concepts described using diagrams, especially in terms of the Threat Model.

Overall, the report has provided me with a good technical insight into blockchains and has allowed me to discover some more specialist terminology that will certainly help when going forward with the report(s). It has also demonstrated how to ensure the security of the blockchain network and has evoked thought into potential methods that could be used to do so.

2.3 A SUPERFICIAL OVERVIEW OF BLOCKCHAIN AND EXISTING APPLICATIONS

With new applications of blockchain emerging daily, it is apparent that there are a large variety of possible uses for the technology. Although none correlate completely with BlockMail, there are various different existing applications which have been drawn from to form a complete solution.

2.3.1 IBM VERIFY CREDENTIALS

IBM Verify Credentials[8] is a system that is currently in the alpha stages of development which takes a blockchain-based, "decentralized approach to identity management". In their video, *Digital identity management: How much of your personal information do you control?*[9] IBM discusses how current identity systems "aggregate" personal information into "centralised systems of records". By opting for a decentralised blockchain system, IBM has removed "identity mediators" – being the companies who store personal information in their own databases. Not only does IBM seek to "offload" data management from companies using their approach, but they also seek to use blockchain to create a more secure and controlled identity management system. Although the system has a much wider scope than this project (aiming to replace identity management for many applications), it provides some key foundations to build upon:

- In the future it is a possibility that no systems which control and store data may be under the control of a central party – this promotes trust via design. Blockchain is one way of achieving this.
- The removal of the *right of access*[10]. This may no longer be necessary as a blockchain system would spread the data throughout the world in an immutable way.
- High availability, backup, and redundancy through distribution.
- Easy expandability.

2.3.2 SMART CONTRACTS AND THE ETHEREUM BLOCKCHAIN

Traditionally, a contract is "an agreement between two parties creating a legal obligation for both of them to perform specific acts"[11]. A "Smart Contract", however, is defined by Jake Frankenfield as "a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code"[12]. This concept of a "smart contract" was first identified and introduced by Nick Szabo in his 1994 paper, *Smart Contracts: Building Blocks for Digital Markets*[13]*.* The first expansive implementation of this type of electronic contract was by the Ethereum Blockchain[14], which provides a way to place code on the blockchain that can be executed by sending a transaction containing instructions to a smart contract address, as well as a "gas" fee. The main advantage in such a system comes with the immutability of the blockchain – being that it creates an unchangeable transaction (and code) that can be represented in any way which the contract writer desires.

Similarly to the Ethereum implementation of blockchain, BlockMail aims to create a system which means that emails cannot be unsent or modified in any way; as smart contracts cannot be changed. Such a feature could be very useful in government sectors or security agencies, which rely on a paper trail. By implementing technology where an item could not be changed, it removes the need to question validity – instead, it can simply be assumed.

2.3.2 EXISTING BLOCKCHAIN BASED EMAIL SYSTEMS

CryptaMail

CryptaMail[3] seems to be the first attempt at a blockchain-based email system. Although the project is seemingly stagnant and non-operational, it is important to note as it takes quite a different approach to BlockMail. Instead of using its own blockchain, it implements the NxtCoin blockchain[15], which provides a large variety of support. In this case, "encrypted messages", "account properties", "alias registration", and "asset exchange" are of particular interest.

By using a pre-existing blockchain, CryptaMail is limited to the features which NxtCoin currently offers. Blockchain will implement a bespoke blockchain which means that the features can be adapted to suit, and can be upgraded in the future.

LedgerMail

As a very closed off implementation, LedgerMail[4] takes an approach to make the service a product, seeking to make money off the technology. The problem with privatising a blockchain like this is that it defeats the key feature of blockchain technology, in that a private implementation "does not offer the same decentralized security as its public counterpart"[16]. Since all "nodes" on a private blockchain are under the direct control of the entity running it, a so-called "middleman" may interfere with the chain itself – the exact problem that the technology seeks to avert.

Unlike LedgerMail, BlockMail aims to implement a public blockchain with encryption for mail contents. This means that the system would benefit from the full feature set of blockchains whilst maintaining the full distributed nature of the system. As discussed in section 2.1.3, the combination of these two aspects would create a "semi-public" blockchain.

CHAPTER 3: ANALYSIS AND DESIGN

3.1 REQUIREMENTS ANALYSIS

3.1.1 FUNCTIONAL

- The network must be open to join by anybody who wishes to do so, both in terms of becoming a node and using the system.
- Email content must be encrypted using a cypher that is near impossible to solve. Only the person with the Private Key (intended recipient) must be able to read mail.
- The network must remain operational even if one (or more) nodes become "dishonest" (see 2.1.2).
    - Must have high Byzantine Fault Tolerance.
- Public / Private Key pair generation must happen on the client-side, as to ensure keys are not exposed to any server at any time.
- The blockchain must use a Proof-of-Stake mechanism to confirm blocks.
- The network must include error checking in order to ensure that malformed data does not get added to the blockchain.
- The network must work on a consensus protocol, to provide an environment in which no number of nodes can take control.
- It must not be possible to modify, delete, or otherwise change emails that have been sent after their confirmation by the BlockMail network.
- The system must contain a series of "master nodes" which provide the first point of contact for new nodes joining the network.
- Addresses must be generated using ECDSA (SECP256K1) to ensure that all are unique.
- "Sign In" operations must take place by validating private and public keys on the client-side. A further check will happen on each node when sending mail to ensure addresses have not been spoofed.
- The network should be able to handle a large number of concurrent connections.

3.1.2 NON-FUNCTIONAL

- The frontend must behave in a responsive manner, and be compatible with any device that supports JavaScript.
- The system should feature an easy to use interface.
- Colours should be carefully selected to create a human-computer interface that allows those who may be partially visually impaired may benefit from the system.
- The system should be free to use.

3.2 DESIGN

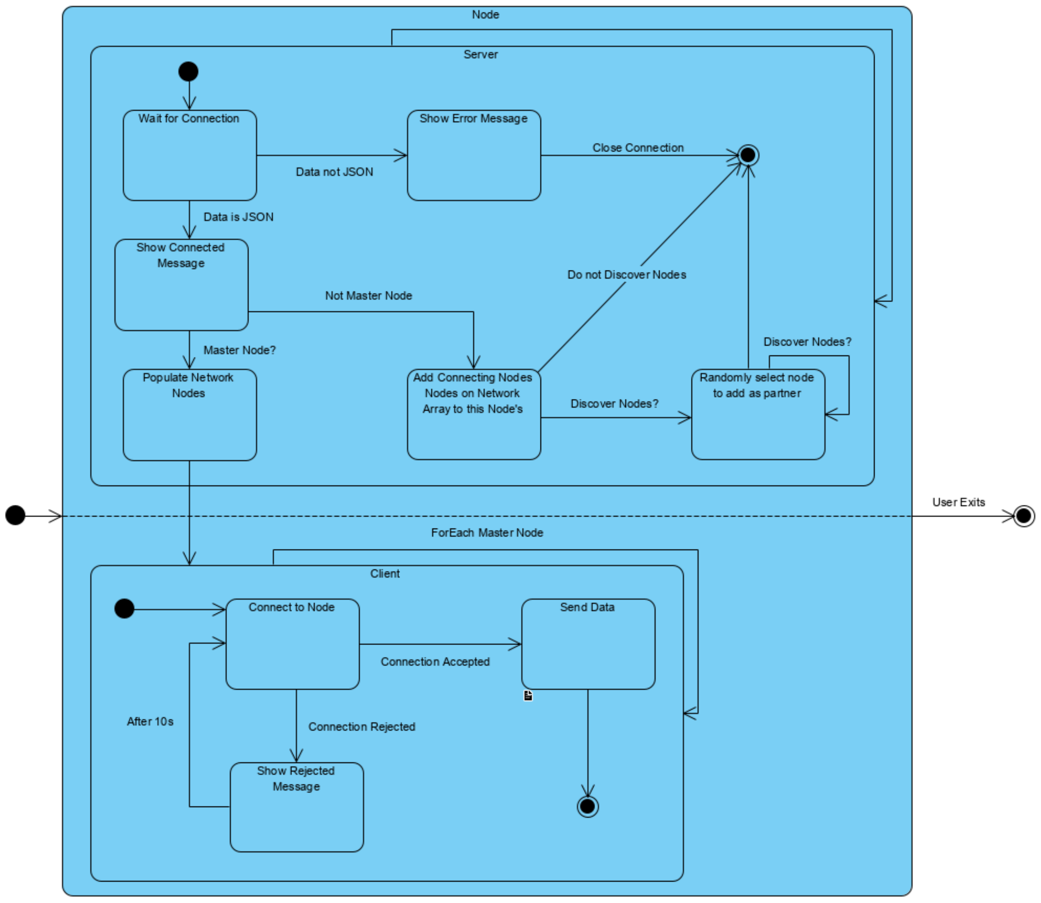
3.2.1 UML DIAGRAMS

Peer-to-Peer Network: State Transition Diagram

Below is a high-level State Transition Diagram for the P2P network that BlockMail will use. It models the code in its current state. Efforts have been made to use UML standards, although, due to the nature of the network this has not been entirely possible (many instances of the same code communicating with each other). However, hopefully, it will provide some insight into the intended workings of the software.

Each node in the Peer-to-Peer network has both a Client and Server class which are responsible for sending and receiving data to/from other nodes. The dotted line indicates concurrency (both the client portion and the server portion run at the same time, as is a feature of Peer-to-Peer networks). In the Client portion, the node attempts to establish connections to the master nodes on the BlockMail network. It will retry the connection every 10 seconds if unsuccessful. If a connection is established pieces of data are sent to the peer in order for the node to become a proper member of the network.

The Server portion constantly waits for incoming connections from incoming peers. It checks that the data received is in the correct (JSON) format, and if so, accepts the connection. It then checks whether it is a master node itself, and establishes a client connection to the client. If not a master node, then the array of all networked nodes is updated with the connecting node's host address. If the server instance has not reached its quota of known nodes, it chooses one to add. Otherwise, it closes the connection.



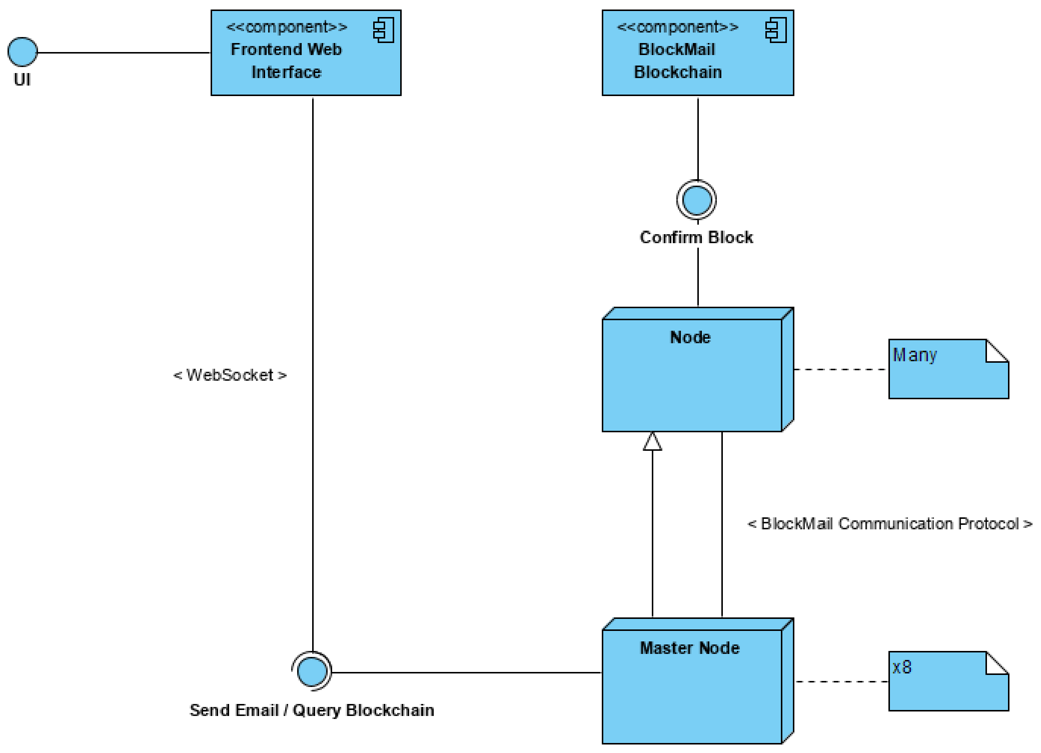
Peer-to-Peer Network: Deployment Diagram

The below diagram gives a basic overview of the expected deployment structure of the network.

The main interface for the user is the web interface, which they will use to send, receive and view mail. This interface communicates with master nodes, which in turn communicate with standard nodes on the BlockMail network. Communication between the frontend and nodes happens using the lightweight WebSocket protocol for basic requests such as sending mail and querying the blockchain for received mail.

All nodes on the BlockMail network (master's and standard) communicate with each other using the proprietary BlockMail Communication Protocol. This protocol allows them to share transactions and other information with each other.

The final part of the diagram indicates how the nodes interface with the BlockMail blockchain, confirming blocks if elected to add theirs to the chain.


UI
<<component>> Frontend Web Interface
<<component>> BlockMail Blockchain
Confirm Block
Node
Many
< WebSocket >
< BlockMail Communication Protocol >
Master Node
x8
Send Email / Query Blockchain

CHAPTER 4: RISK ASSESSMENT

Risk	Likelihood Rating	Impact Rating	Impact Description	Preventative Actions
Overestimation of time for the project.	Medium.	High.	May not deliver a complete project.	Set out strict time limits using tools such as GANTT charts. Continuously have progress reviews with supervisor.
Computer failure.	Low.	High.	Could lead to loss of work and non-submission.	Ensure that frequent backups are taken. Have a secondary PC that can be used in case of emergency.
Underestimation of the project cost.	Medium.	Medium.	May not be able to buy certain pieces of hardware or software that are required for the project.	Construct a clear outline of hardware and software requirements prior to beginning implementation.
Implementation of unnecessary features.	Medium.	Medium.	May clutter code and waste time that could be spent in other areas.	Ensure that features meet requirements without being over-engineered.
Poor requirements outline.	Medium.	Medium.	A poor definition of requirements at the start of the project may lead to time being wasted during the development phase, as requirements are gathered.	Produce a detailed set of requirements, which provide a modular style feature-set.
Lack of experience in the language(s) used for the solution.	Medium.	Medium.	May produce a sub-par solution with poor programming style.	Prepare for using languages by completing tutorials and researching modules to be used.
Unforeseen illness.	Low.	Medium.	An illness could lead to some time lost during development.	Maintain good personal hygiene.
Plagiarism.	Medium.	High.	May result in mark being zeroed.	Be informed about how to reference, and how to ensure that others are given proper credit for their work. If not sure – ask.

CHAPTER 5: PRESENTATION

5.1 LANGUAGE AND PLATFORM RATIONALE

BlockMail's backend (i.e. the node software) will be developed using Python 3, with a selection of libraries which will assist in blockchain distribution, node communication, and block validation. Python 3 was selected due to its versatile nature and the large selection of libraries which are available for it. As an Object-Oriented language, classes provide many features that will be key for node communication, as many concurrent connections will be established at one time – these will be reflected as instances. Experience with Python was also a significant influential factor in my decision to use it.

The Frontend of BlockMail is to be developed using HTML with Bootstrap, CSS, and JavaScript for logic involving communicating with nodes on the network to obtain and send mail. I chose the above as they will facilitate operation on a large number of devices, thus fulfilling the requirement "The frontend must behave in a responsive manner, and be compatible with any device that supports JavaScript".

5.2 SOURCE CODE AND EXPLANATIONS

Code given here is in no way final, but simply provides an insight into the current stage of development. This may change significantly by project submission (although unlikely).

5.2.1 PEER-TO-PEER NETWORK

The Peer-to-Peer (P2P) network part of the BlockMail infrastructure ensures the actual operation of the blockchain. It will deal with node discovery, blockchain distribution, and contains the actual Proof-of-Stake algorithm. Some of these parts are complete, but it is still a work-in-progress so is not a true reflection of the final node software.

Imports and Constants

Code Snippet

```python
import socket
import threading
import json
import time
import random

VERSION = "1.0"
# Blockchain node discovery port. DO NOT CHANGE.
DEFAULT_DISCOVERY_PORT = 41285
MASTER_NODES = ["127.0.0.1", "127.0.0.2", "127.0.0.3", "127.0.0.4"]
# Used only for master nodes to keep track of and assign "known nodes".
NODES_ON_NETWORK = []
KNOWN_NODES = []
SERVER_IP = "127.0.0.1"
```

Explanation

This part of the program imports all of the modules that are to be used in the Peer-to-Peer network section, and defines constants to be used throughout. Each module and constant is explained below:

Modules

- **socket** – Provides the primary P2P networking interface to communicate between nodes.
- **threading** – Facilitates concurrency in order to ensure non-blocking I/O. This is required as one node may connect to many others (or receive many other connections) at the same time.
- **json** – Used to parse JSON strings that are passed between nodes.
- **time** – Used for connection timeouts.
- **random** – Used for random number generation, specifically for partner node selection.

Constants

- **VERSION** – The version of the node software.
- **DEFAULT_DISCOVERY_PORT** – The port on which nodes communicate with each other.
- **MASTER_NODES** – An array of all master nodes on the network.
- **NODES_ON_NETWORK** – An array of all nodes on the network.
- **KNOWN_NODES** – The partner nodes of this node.
- **SERVER_IP** – The IP that the server socket is to be created on.

NodeServer Class

Code Snippet

```
class NodeServer:
  """Controls communication between nodes on the BlockMail network
     along with NodeNodeClient.
     Discovers neighbouring nodes.\n
     Takes two arguments:
       host - IP to host server on.
       port - Port to host server on. """

  def __init__(self, host=SERVER_IP, port=DEFAULT_DISCOVERY_PORT):
    self.__host = host
    self.__port = port
    self.__json_data = ""
    # Create Server thread to avoid blocking.
    thread = threading.Thread(target=self.establishSocket)
    thread.start()  # Start the thread.

  def establishSocket(self):
    # Open a new socket, s.
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
      try:
        # Bind socket to host and port passed to class.
        s.bind((self.__host, self.__port))
      except WindowsError as e:
        if e.winerror == 10048:  # 10048 - Port blocked error.
          print("Port already in use. Please check port usage by
                 other applications, and ensure that the port is
```

```
                  open on your network.")
          exit()
        else:
          print(e)
      s.listen()
      print(f"\n[Node Discovery] Listening on {self.__host}
              :{self.__port}...")
      self.acceptConnection(s)  # Accept connections.

  def acceptConnection(self, s):
    connection, address = s.accept()  # Accept data.
    with connection:
      while True:
        data = connection.recv(1024)  # Maximum data stream size of 1024b.
        if data:
          print(f"\nPeer connected - {str(address[0])}:{str(address[1])}")
          self.__json_data = self.isJson(data)
          if self.__json_data:  # Is data in correct format?
            print(f"\n{self.__json_data['origin_host']}:{str(address[1])}
                    - Peer Connected.")
            if SERVER_IP in MASTER_NODES:
              # Populate NODES_ON_NETWORK on initial node connection
              # to Master Nodes.
              self.populateNodesOnNetworkMasters()
            else:
              for node in self.__json_data["nodes_on_network"]:
                if node not in NODES_ON_NETWORK:
                  NODES_ON_NETWORK.append(node)
              # Should I be looking for more nodes?
              while self.discoverNodes():
                self.chooseNodesToAdd()
          else:
            print("Not a JSON!")
        break
      print(f"\n{self.__json_data['origin_host']}:{str(address[1])}
              - Connection Closed.")
      self.acceptConnection(s)

  def discoverNodes(self):
    # Should I be discovering more nodes?
    if len(KNOWN_NODES) < len(MASTER_NODES):
      return True
    return False

  def chooseNodesToAdd(self):
    # Randomly choose node to become neighbour.
    node_index = random.randint(0, len(NODES_ON_NETWORK)-1)
    if NODES_ON_NETWORK[node_index] not in KNOWN_NODES
    and NODES_ON_NETWORK[node_index] != SERVER_IP:
      # Add node to known nodes.
      KNOWN_NODES.append(NODES_ON_NETWORK[node_index])
      print(f"\nNew Neighbour Found! Neighbouring Nodes: {KNOWN_NODES}")

  def populateNodesOnNetworkMasters(self):
    # Node not already in NODES_ON_NETWORK?
    if self.__json_data["origin_host"] not in NODES_ON_NETWORK:
      # Add the node to array containing all nodes on network.
      NODES_ON_NETWORK.append(self.__json_data["origin_host"])
      print(f"\nNew Node Joined the Network: {self.__json_data
              'origin_host']}")
      NodeClient(host=self.__json_data["origin_host"])
  def isJson(self, string):
    try:
      json_data = json.loads(string)  # Try to parse JSON.
    except ValueError:
      return False  # If ValueError - not JSON.
    return json_data
```

Explanation

The code given here provides the Server-side part of the Peer-to-Peer network connection. Each function is explained below:

__init__(self, host=SERVER_IP, port=DEFAULT_DISCOVERY_PORT)

Sets up the default variables, and creates threads for each instance to avoid blocking.

establishSocket(self)

Checks whether the port is available on the network and if so creates a socket. If the port is unavailable the user will be informed via an appropriate message.

acceptConnection(self, s)

This is the primary function of the class. It serves to receive data, process it, and saves it where needed. If the node is a Master Node, `populateNodesOnNetworkMasters()` is called. Otherwise, it collects IP addresses of incoming connections, and if they are valid, collects the incoming nodes' `NODES_ON_NETWORK` array, and for all entries not existing, stores them in its own `NODES_ON_NETWORK` array. Should the node need to discover more neighbour nodes, it will then randomly choose which nodes to become neighbours with. Recursively called to "always listen".

discoverNodes(self)

Returns True or False depending on whether the node should be looking for more neighbours.

chooseNodesToAdd(self)

As described in the section about `acceptConnection(self, s)`, this function decides which neighbours should be added by randomly choosing them and adding them to the `KNOWN_NODES` array accordingly.

populateNodesOnNetworkMasters(self)

If the node is a master node, then this function is called upon peer connection. It will update the `NODES_ON_NETWORK` array by adding the host address of the incoming connection. It will then establish a client connection to that host.

isJson(self, string)

Validates if a given string is JSON, and if so, returns True. Otherwise, returns False.

NodeClient Class

Code Snippet

```
class NodeClient:
  """Controls communication between nodes on the BlockMail network
     along with NodeServer.
     Sends node information to peers.\n
       host - IP of node to connect to.
       port - Port of node to connect to."""

  def __init__(self, host, port=DEFAULT_DISCOVERY_PORT,
               hop=False, server_json=""):
    self.__host = host
    self.__port = port
    self.__hop = hop
    self.__server_json = server_json
    # Create Server thread to avoid blocking.
    thread = threading.Thread(target=self.establishSocket)
    thread.start()  # Start the thread.

  def establishSocket(self):
    print(f"\n{self.__host}:{self.__port} - Connecting...")
    # Open a new socket, s.
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
      try:
        # Attempt to establish connection at given host and port.
        s.connect((self.__host, self.__port))
        data_to_send = {"version": VERSION,
                "origin_host": SERVER_IP,
                "origin_port": DEFAULT_DISCOVERY_PORT,
                "nodes_on_network": NODES_ON_NETWORK}
        s.sendall(bytes(json.dumps(data_to_send), encoding="UTF8"))
      except ConnectionRefusedError:
        print(f"\n{self.__host}:{str(self.__port)} - Connection
                refused (node may be offline). Retrying in 10 seconds...")
        time.sleep(10)  # Wait for 10 seconds
        self.establishSocket()  # Retry
```

Explanation

The code given here provides the Server-side part of the Peer-to-Peer network connection. Each function is explained below:

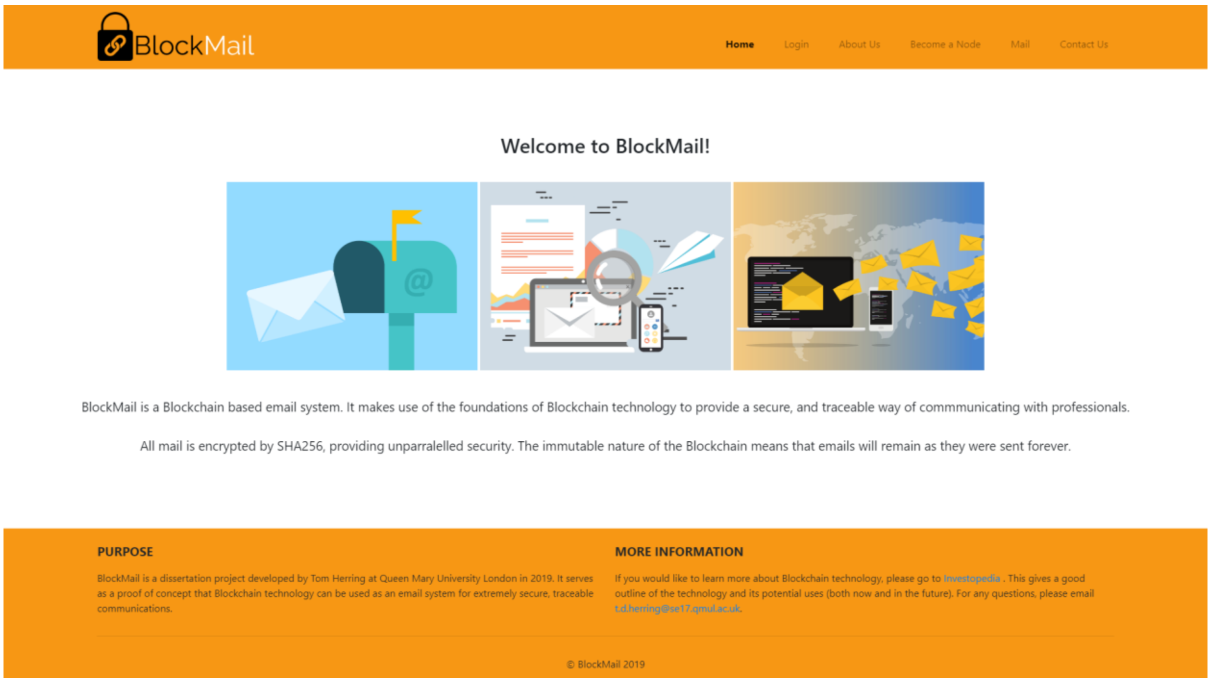
__init__(self, host=SERVER_IP, port=DEFAULT_DISCOVERY_PORT)

Sets up the default variables, and creates threads for each instance to avoid blocking.

establishSocket(self)

Attempts to set up a socket to connect to the given host and port. If successful, send node information in JSON format, including: version, host, port, and `NODES_ON_NETWORK` array. If the connection is refused, the user will be warned, and a connection will be reattempted every 10 seconds from thereon.

5.3 FRONTEND INTERFACE



Welcome to BlockMail!

BlockMail is a Blockchain based email system. It makes use of the foundations of Blockchain technology to provide a secure, and traceable way of commmunicating with professionals.

All mail is encrypted by SHA256, providing unparralelled security. The immutable nature of the Blockchain means that emails will remain as they were sent forever.

PURPOSE

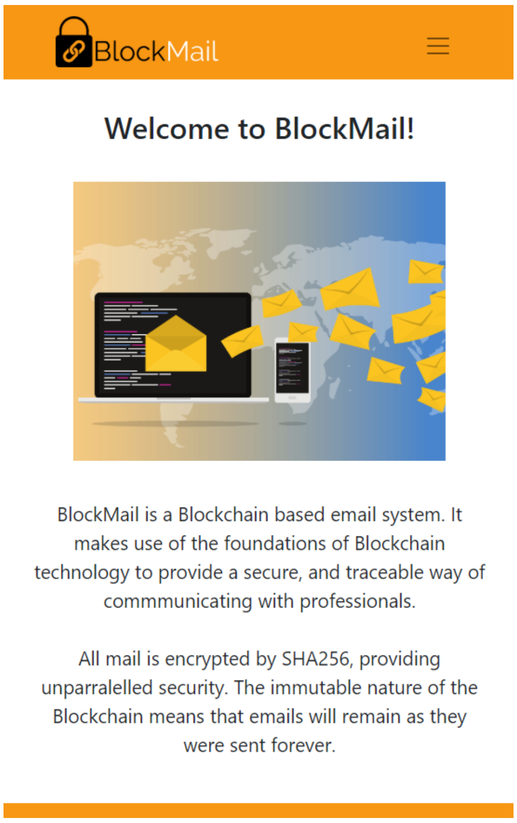
BlockMail is a dissertation project developed by Tom Herring at Queen Mary University London in 2019. It serves as a proof of concept that Blockchain technology can be used as an email system for extremely secure, traceable communications.

MORE INFORMATION

If you would like to learn more about Blockchain technology, please go to Investopedia . This gives a good outline of the technology and its potential uses (both now and in the future). For any questions, please email t.d.herring@se17.qmul.ac.uk.

© BlockMail 2019

Figure 1



Welcome to BlockMail!

BlockMail is a Blockchain based email system. It makes use of the foundations of Blockchain technology to provide a secure, and traceable way of commmunicating with professionals.

All mail is encrypted by SHA256, providing unparralelled security. The immutable nature of the Blockchain means that emails will remain as they were sent forever.

Figure 2

The two figures here demonstrate the current homepage for the primary user interface for BlockMail. A distinct colour choice of orange and black has been selected in order to create a unique brand identity.

Figure 1 shows a normal desktop environment (1920x1080). The website also displays properly on higher resolution displays. Although simple it gives a quick but thorough introduction to the project and outlines its features.

Figure 2 presents the responsive design of the user interface. The menu collapses down into a mobile-friendly style, and images are adjusted respectively. This is to be achieved using Bootstrap as well as custom CSS.

It is important to note that this design is not final, but just provides some guidance as to what the final project may look like. As with all aspects, it is subject to change as development progresses.

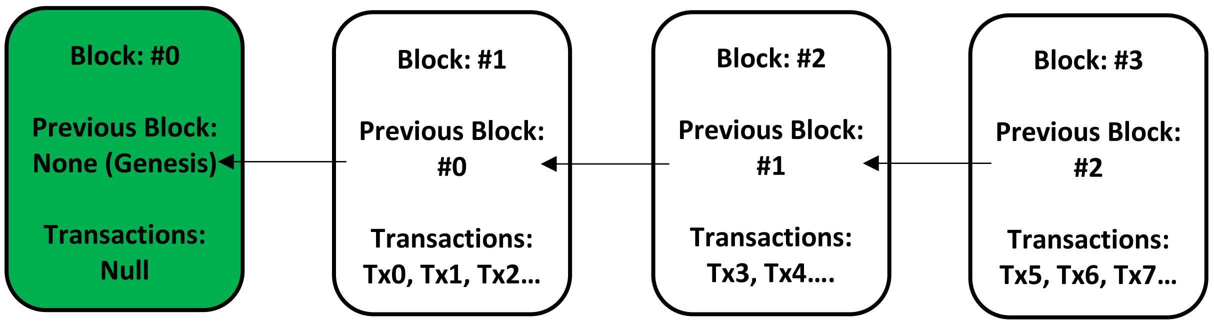
CHAPTER 6: EVALUATION

To be completed in Final Report.

CHAPTER 7: CONCLUSION

To be completed in Final Report.

APPENDIX A – BLOCKCHAIN DIAGRAM



* Information showed per block here is not actually reflective of a true block, which would likely contain much more information (such as a hash digest, timestamp, etc…). This is simply provided as a means to demonstrate the primary concept of the blockchain.

REFERENCES

[1] Lamport, L; Shostak, R; Pease, M. (2019). [online] Available at: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf [Accessed 18 Nov. 2019].
[2] Nakamoto, S. (2008). [online] Bitcoin: A Peer-to-Peer Electronic Cash System. Available at: https://bitcoin.org/bitcoin.pdf [Accessed 5 Nov. 2019].
[3] CryptaMail. (2019). CryptaMail. [online] Available at: http://www.cryptamail.com/ [Accessed 10 Nov. 2019].
[4] Ledgermail.io. (2019). World's first email service on Blockchain | Best Blockchain based Encrypted Email Service. [online] Available at: https://ledgermail.io/ [Accessed 10 Nov. 2019].
[5] Jayachandran, P. (2017). The difference between public and private blockchain - Blockchain Pulse: IBM Blockchain Blog. [online] Blockchain Pulse: IBM Blockchain Blog. Available at: https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain/ [Accessed 6 Nov. 2019].
[6]Sultan, K; Ruhi, U; Lakhani, R (2018). [online] Conceptualizing Blockchains: Characteristics and Applications. Available at: https://arxiv.org/ftp/arxiv/papers/1806/1806.03693.pdf [Accessed 14 Nov. 2019].
[7] Das, S; Kolluri, A; Saxena, P; Yu, H (2018). [online] (Invited Paper) on the Security of Blockchain Consensus Protocols. Available at: https://www.researchgate.net/publication/329437697_Invited_Paper_on_the_Security_of_Blockchain_Consensus_Protocols_14th_International_Conference_ICISS_2018_Bangalore_India_December_17-19_2018_Proceedings [Accessed 16 Nov. 2019].
[8] Ibm.com. (2019). Blockchain for digital identity - IBM Blockchain. [online] Available at: https://www.ibm.com/blockchain/solutions/identity [Accessed 25 Nov. 2019].
[9] IBM. YouTube.com. (2019). [online] Available at: https://www.youtube.com/watch?v=H69l_trRArU [Accessed 25 Nov. 2019].
[10] Ico.org.uk. (2019). Your right of access. [online] Available at: https://ico.org.uk/your-data-matters/your-right-to-get-copies-of-your-data/ [Accessed 25 Nov. 2019].
[11] LegalMatch Law Library. (2010). What is a Contract?. [online] Available at: https://www.legalmatch.com/law-library/article/what-is-a-contract.html [Accessed 25 Nov. 2019].
[12] Frankenfield, J. Investopedia. (2019). Smart Contracts: What You Need to Know. [online] Available at: https://www.investopedia.com/terms/s/smart-contracts.asp [Accessed 25 Nov. 2019].
[13] Nick Szabo. (2019). Smart Contracts: Building Blocks for Digital Markets. [online] Available at: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html [Accessed 25 Nov. 2019].
[14] Buterin, V. (2019). A Next Generation Smart Contract & Decentralized Application Platform. [online] Available at: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf [Accessed 25 Nov. 2019].
[15] Nxtwiki.org. (2019). Whitepaper:Nxt - Nxt Wiki. [online] Available at: https://nxtwiki.org/wiki/Whitepaper:Nxt [Accessed 25 Nov. 2019].
[16] Medium. (2019). Busting the Myth of Private Blockchains. [online] Available at: https://media.consensys.net/busting-the-myth-of-private-blockchains-9ae0ed058b0d [Accessed 26 Nov. 2019].