FINAL YEAR / UNDERGRADUATE PROJECT 2019/20

School of Electronic Engineering and Computer Science

FINAL 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: TBA

ABSTRACT

Over the past ten years, blockchain has gained widespread attention throughout the world. It is being tested in several use cases in various fields, providing reliability, security and byzantine fault tolerance. This paper will introduce a blockchain-based email system, *BlockMail*, which takes advantage of blockchain technology to provide a network of intercommunicating nodes which exchange mail. 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, leading to a highly-available, stable system.

The project applies various features from the field of cryptography to provide key generation and hashing, distributed systems for blockchain distribution and management, and networking for inter-node and frontend-node communications.

ACKNOWLEDGEMENTS

The support of my Project Supervisor, Professor Steve Uhlig has proven invaluable throughout the project, providing advice, technical input, and showing an incredible willingness to help. Thank you.

Also, to my friends, family, and colleagues at work, I would like to say thank you for your understanding, support and patience from the beginning to the end of this project – it would not have been possible without you.

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 unexpectedly.

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

1.4 OBJECTIVES

The tables below outline the attainment of objectives, based on those outlined in the Interim Report. Where necessary, comments have been given in order to explain how the objectives where met. Where they were not met, an explanation follows to outline why this is the case.

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

Objective	Met?	Comments
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.	Yes	All key generation happens with client-side JavaScript. The only key exposed to the network is the public email encryption key and the address public key.
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).	Yes	Private keys both have ".key" extension. The email address has extension ".addr".
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.	Yes	Number reduced to 4 for testing and demonstration.
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.	Yes	Blocks contain all emails received in the block time period.
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.	Yes	Yes, although a design change has been made. Rather than blocks being elected, individual emails are broadcast to the entire BlockMail network.
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".	Yes	Once again, number reduced to 4 for testing and demonstration.

• This leads to exponential email broadcasting.		
Anybody may download the BlockMail node client and contribute to the network.	Yes	Using the “Become a Node” link on the website.
Emails are to be encrypted with RSA-2048. • This will be facilitated by the second private key.	Yes	
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.	Yes	
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.	Yes	
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.	Yes	
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.	Yes	

1.4.2 OPTIONAL

Objective	Met?	Comments
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.	No	Deemed unnecessary, as when the network expands, load will be spread naturally.
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.	Yes	
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.	TBA	
An option to deactivate “accounts”.	TBA	

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 system at a high level.
- ***Presentation*** – This section will provide a place to introduce the design of the system. I will provide code snippets of key parts of the system and an overview of the interfaces.
- ***Evaluation*** – I will give an overview of the strengths and weaknesses of the project, and will explain what could be done in the future to produce a more effective solution.
- ***Conclusion*** – Here I conclude the report, exploring whether its aim has been met and discuss its success along with potential future applications.
- ***Appendices*** – Various sections containing facts and figures that support the report.
- ***References*** – A bibliography of all resources (literature, websites, etc…) used during the writing of this 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.

BlockMail makes use of an alternative system. Rather than considering processing power, it adopts a model which commits a block to the blockchain at a given time interval (for example, 60 seconds). Since a double-spend attack is not beneficial in the case of BlockMail, the Proof-of-Work model is not required, but a Proof-of-Stake model is adopted – in this case, using block age to decide whether or not to publish 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, other factors 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 concerning 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.

2.1.4.1 Analysis of The Definition

Below is an analysis of the primary constituents of the definition:

- "Decentralized" – No central control point (eg: server).
- "Cryptographically linked" – Every hash of each block depends on the last. Provides the immutability blockchain.
- "Digitally signed" – Prevents accidental or unauthorised modification of blocks and transactions.
- "Asset transactions" – The information flowing around the network. In BlockMail's case, this is emails.
- "Consensus model" – The agreement model which governs the reliability of the network. Ensures that no one node can take over and trick the network.

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 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, to create provably valid blocks.

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 does not use "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”, relates 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 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, for example, 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 blockchain structures 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 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 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.
- Ease of expansion.

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

2.3.2.1 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.

2.3.2.2 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 joining 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 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 responsively, 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

3.2.1.1 Peer-to-Peer Network: State Transition Diagram

On the next page, there is a State Diagram which highlights the overall operation of the BlockMail node client. Although it is at quite a high level, it is accurate in the fact that it gives an overview of all operation of the client.

The dashed lines in the diagram have been used to represent concurrency. All inner states with pseudo start state indicators (filled circle) indicate that they are run as soon as the client is launched. The only time the program will exit is in the case that the user closes the program, or that it fails to validate its local time with an NTP server.

The "Server" section of the diagram highlights the four main virtual servers which exist in BlockMail, each handling a specific task, as follows:

- ***FrontendCommsServer*** – Waits for communication from the frontend to serve various blockchain-related services.
- ***BroadcastServer*** – Waits for incoming mail from other nodes, originating from BroadcastClient on that neighbour.
- ***SyncServer*** – Waits for incoming blockchain data from other nodes, originating from SyncClient on that neighbour.
- ***DiscoveryServer*** – Handles new connections to the network and establishes neighbours.

The same applies to the "Client" subset:

- ***BroadcastClient*** – After a new email is received to a node, broadcasts it to all neighbour nodes.
- ***SyncClient*** – When a new node joins the network, send this nodes copy of the blockchain to that new node.
- ***DiscoveryClient*** – Contacts DiscoveryServer's on other nodes to exchange information.

Finally, there are some helper classes:

- ***Mail*** – Handles the creation of new mail objects and their commitment to the blockchain. invokes BroadcastClient to send the mail to other nodes.
- ***Time*** – Checks client time against an NTP server to ensure in sync, and ensures that blocks are created at the given block interval.
- ***Block*** – Creates, hashes, and commits finished blocks to the blockchain.

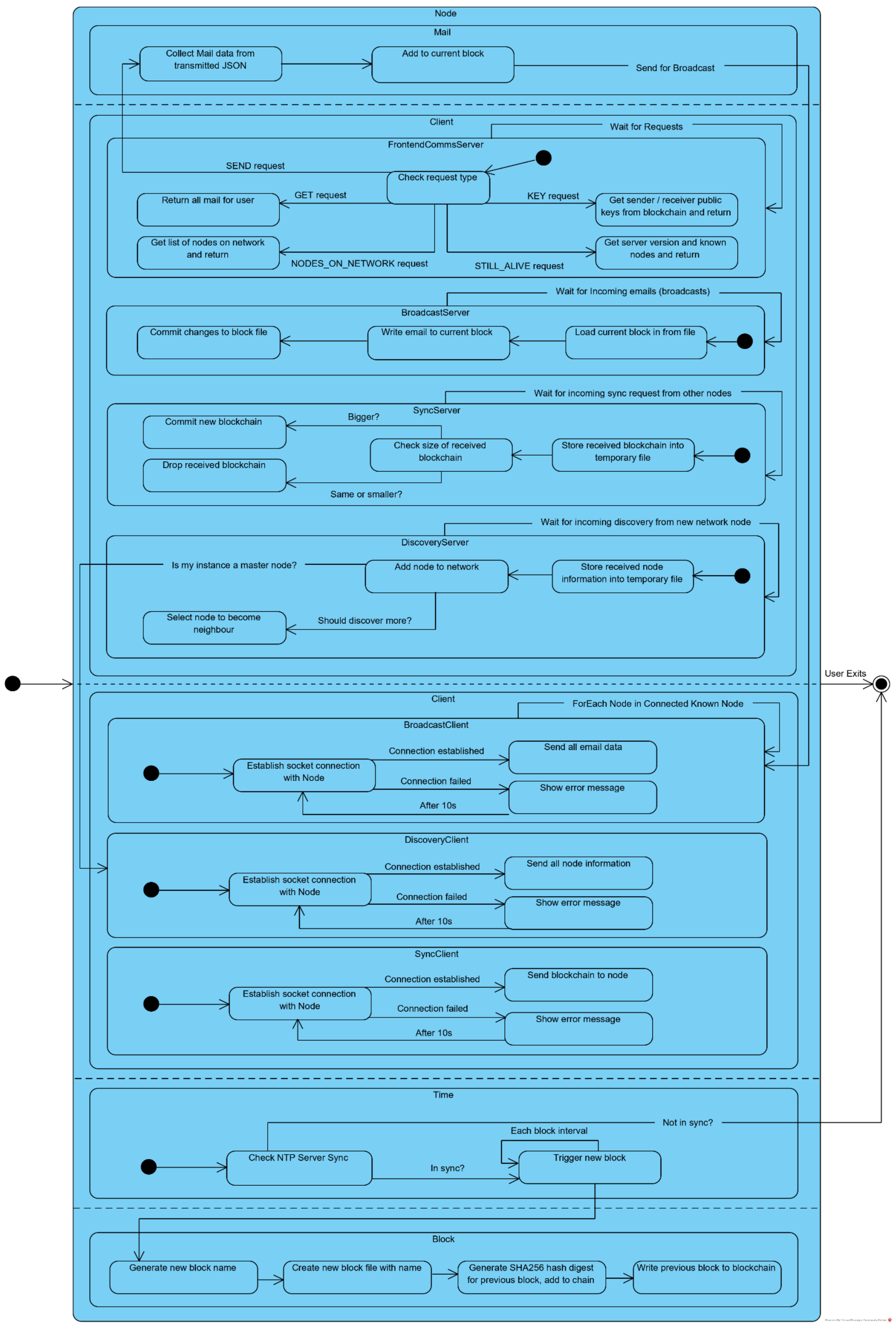

Node
Mail
Collect Mail data from transmitted JSON
Add to current block
Send for Broadcast
Client
Wait for Requests
FrontendCommsServer
SEND request
Check request type
GET request
KEY request
Return all mail for user
Get sender / receiver public keys from blockchain and return
Get list of nodes on network and return
NODES_ON_NETWORK request
STILL_ALIVE request
Get server version and known nodes and return
Wait for Incoming emails (broadcasts)
BroadcastServer
Commit changes to block file
Write email to current block
Load current block in from file
Wait for incoming sync request from other nodes
SyncServer
Bigger?
Commit new blockchain
Check size of received blockchain
Store received blockchain into temporary file
Drop received blockchain
Same or smaller?
Wait for incoming discovery from new network node
DiscoveryServer
Is my instance a master node?
Add node to network
Store received node information into temporary file
Select node to become neighbour
Should discover more?
User Exits
Client
ForEach Node in Connected Known Node
BroadcastClient
Connection established
Send all email data
Establish socket connection with Node
Connection failed
Show error message
After 10s
DiscoveryClient
Connection established
Send all node information
Establish socket connection with Node
Connection failed
Show error message
After 10s
SyncClient
Connection established
Send blockchain to node
Establish socket connection with Node
Connection failed
Show error message
After 10s
Time
Not in sync?
Each block interval
Check NTP Server Sync
In sync?
Trigger new block
Block
Generate new block name
Create new block file with name
Generate SHA256 hash digest for previous block, add to chain
Write previous block to blockchain

Figure 1 – State Transition Diagram, BlockMail Node

3.2.1.2 Peer-to-Peer Network: Deployment Diagram

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

The main interface for the user is the web interface, which is used to send, receive and view mail. It also serves other purposes such as viewing an overview of the operating network and creating accounts. This interface communicates with master nodes, which in turn communicate with standard nodes on the BlockMail network. Communication between the frontend and nodes is facilitated by the lightweight and widely supported WebSocket protocol for completing 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 Protocols. These protocols allow nodes 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.

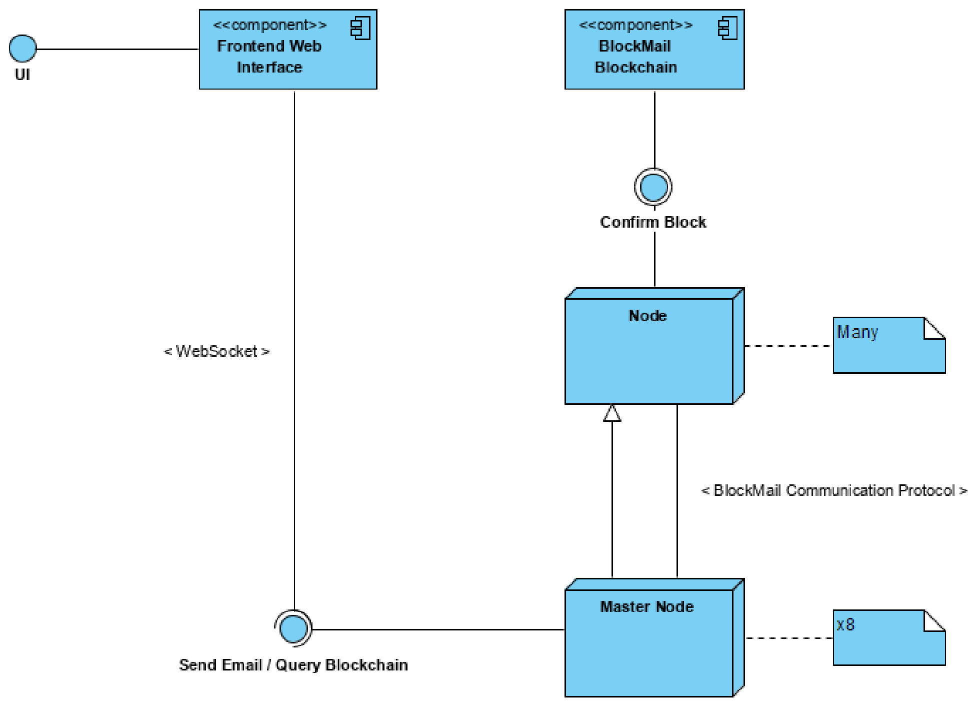

UI
<<component>> Frontend Web Interface
<<component>> BlockMail Blockchain
Confirm Block
Node
Many
< WebSocket >
< BlockMail Communication Protocol >
Master Node
x8
Send Email / Query Blockchain

Figure 2 – Deployment Diagram, BlockMail Network

3.2.2 METHODS OF OPERATION

3.2.2.1 Network Propagation

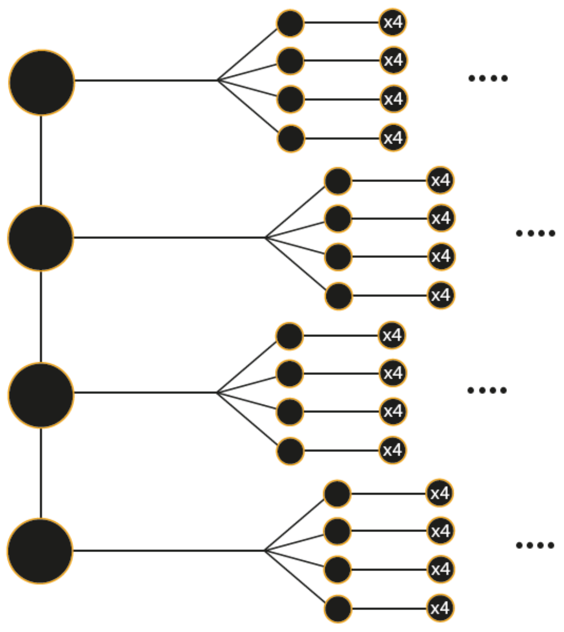


Figure 3 – Network Propagation Diagram

Figure 3 outlines the basic node structure of the BlockMail network. The leftmost (large) circles indicate "Master Nodes". These serve as the primary entry point for the system and are only required to get the network up and running. All of the master nodes are directly connected. Although the network can function with only one master node, in the interests of security and reliability, it is best to have at least two.

The smaller nodes and their children are just "standard" nodes on the network. Each of these nodes contacts the master nodes to inform them that they have connected.

The general propagation pattern is as follows: every node on the network knows 4 others, so by proxy knows 16, 64, 256, … 2^n.

3.2.2.2 Encryption Process

Figure 4 shows the method of encryption for mail.

Firstly, the public keys for the receiver and sender addresses are looked up in the blockchain. The key is the first entry in the blockchain for each address. Mail is encrypted with both keys so that both parties can read it. The email subject and body is then encrypted with both keys, and sent to a node to be hashed, and committed to the blockchain.

After this point, the email is broadcast to all other nodes on the BlockMail network.

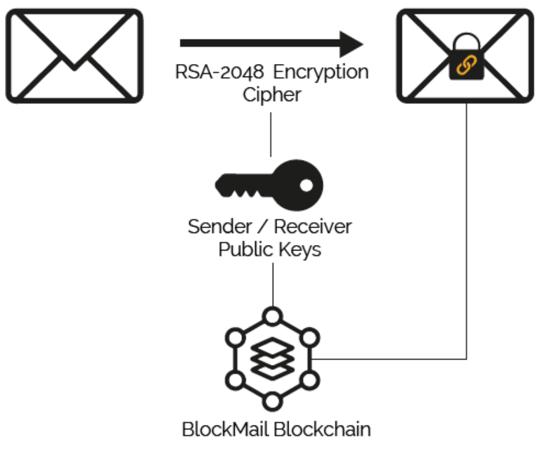


Figure 4 – Encryption Diagram

CHAPTER 4: PRESENTATION

4.1 LANGUAGE AND PLATFORM RATIONALE

BlockMail's backend (i.e. the node software) has been developed using Python 3, with a selection of libraries which assist in blockchain distribution, node communication, and block validation (see section 5.2). 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 responsively, and be compatible with any device that supports JavaScript".

4.2 IMPLEMENTATION

4.2.1 LIBRARY SELECTION

This will outline the modules used in both Python (backend) and JavaScript (frontend).

4.2.1.1 Python

- **`socket`** – Facilitates the TCP socket protocol used for intercommunication between nodes on the BlockMail network.
- **`threading`** – Facilitates parallelisation of the constituent server and client portions of each BlockMail node.
- **`json`** – Allows for interpretation of JSON objects in Python, both in communication from frontend services and for processing blocks and the blockchain itself, which are in JSON format.
- **`asyncio`** – Used in conjunction with `websockets` to provide threaded operation. The **`threading`** library packaged with Python is not supported.
- **`datetime`** – Used for processing date and time related objects, such as mail timestamps. Also used with time synchronisation.
- **`time`** – Provides various delays throughout the program.
- **`random`** – Used when selecting nodes to become neighbours.
- **`os`** – Used for various operating system file operations, primarily when working with block files and the blockchain itself.
- **`shutil`** – Similar to `os`. Provides the ability to copy files.
- **`ntplib`** – Used to contact NTP servers to ensure that client time is synchronised.
- **`tempfile`** – Handles the management of temporary files which are created when nodes communicate with each other over their TCP sockets.
- **`hashlib`** – Generates the SHA256 hash digests for each block before committing to the blockchain.
- **`websockets`** – Allows for communication with frontend services, which use the lightweight JavaScript WebSocket protocol.
  - Aymeric Augustin, Available at: https://pypi.org/project/websockets/
- **`ijson`** – Facilitates progressive reading in of JSON files to prevent crashes due to large files.
  - Rodrigo Tobar, Ivan Sagalaev, Available at: https://pypi.org/project/ijson/

4.2.1.2 JavaScript

The below modules are NodeJS[17] modules, which have been converted into client available scripts using a program called Browserify[18].

- **`elliptic`** – Facilitates the client-side generation of ECDSA SECP256K1 address keypairs, used for "logging in". Also allows for validation of keys.
  - Fedor Indutny, Available at: https://www.npmjs.com/package/elliptic
- **`node-rsa`** – Facilitates the client-side generation of RSA keys, encryption and decryption of RSA-encrypted data using the provided keys.
  - Sergey Vychegzhanin, Available at: https://www.npmjs.com/package/node-rsa

4.2.2 KEY CODE SNIPPETS

4.2.2.1 Connection Handler (ServerConnection)

The connection handler is always running on each BlockMail node and waits for a request to either the Discovery Server, Synchronisation Server, or Broadcast Server. Upon receiving a valid request, the request body is directed to the appropriate server for processing. The following two functions handle the bulk of the computation in this area.

```
def run(self):
        while True:
            data = self.__connection.recv(RECV_SIZE)  # Maximum data stream size of 256b
            decoded_data = data.decode("utf-8")
            if data:
                if self.__expected_size == -1:  # Expecting new data stream?
                    hex_length = decoded_data.split("{")[0]
                    self.__expected_size = int(hex_length, 16)  # Convert hex to integer
                    self.__expected_size_counter = self.__expected_size
                    decoded_data = decoded_data[len(hex_length):]  # Cut size off front
                    self.createTempFile()
                print(f"[{self.__server_type}] Expecting {self.__expected_size_counter}
                     more bytes of data...")
                self.processIncomingData(decoded_data)  # Store incoming data in file
                if self.__expected_size_counter > RECV_SIZE:
                    self.__expected_size_counter -= RECV_SIZE
                else:  # All data in stream received?
                   # Data stream complete, so reset expected_size (Ready for more data)
                    self.__expected_size = -1
                    self.__tmp_file.close()
                    print(f"[{self.__server_type}] All Data Received!")
                    self.directToCorrectServer()

    def directToCorrectServer(self):
        if self.__server_type == "DISCOVERY":
            DiscoverySever(self.__tmp_file_name)
        elif self.__server_type == "BLOCKCHAIN":
            SyncServer(self.__tmp_file_name)
        elif self.__server_type == "BROADCAST":
            BroadcastServer(self.__tmp_file_name)
```

The *run* method is executed in a thread that is created when each instance of each of the servers described above is created (on node launch). Its primary purpose is to await incoming data and reconstruct TCP segments, storing the result in a temporary file. After the data is combined into a valid format, the system calls the *directToCorrectServer* method, which looks at the *server_type* variable passed during instantiation, which identifies the part of the system the data is intended for.

By using a common class for all servers, it greatly reduces the code required to run the node. Ideally, this would have been implemented as a parent class for each server, however, limitations with the socket module in python prevent the passing of connection instances using a normal class hierarchy.

4.2.2.2 Blockchain Synchronisation Mechanism

Server Side

The below is executed upon receiving a blockchain file from another node.

Firstly, it checks whether this is the first synchronisation (ie: the node is new). If so, it simply accepts the blockchain and initiates a new block. Otherwise, if the received blockchain is larger than the current held by the node, it is replaced. In either case, this is followed by a new block being created.

```
def checkBlockchainUpdate(self):
    if os.path.exists("blocks/blockchain.chain"):  # Updating an existing blockchain?
        # Is the blockchain received bigger than the current one I hold?
        if os.stat(self.__tmp_file_name).st_size >
        os.stat("blocks/blockchain.chain").st_size:
            print("[BLOCKCHAIN] Updating Blockchain...")
            # Replace the blockchain (copy - overwrite)
            shutil.copy(self.__tmp_file_name, "blocks/blockchain.chain")
            Block(True)  # Create a Block in sync mode
            print("[BLOCKCHAIN] Blockchain Successfully Updated!")
        else:
            print("[BLOCKCHAIN] Received the Same or Outdated Blockchain.
                  No Changes Made.")
    else:  # Or first chain received
        print("[BLOCKCHAIN] Synchronising Blockchain...")
        shutil.copy(self.__tmp_file_name, "blocks/blockchain.chain")
        Block(True)  # Create a Block in sync mode
        print("[BLOCKCHAIN] Blockchain Successfully Synchronised!")
```

Client Side

This code is executed when a neighbouring node "turns on".

Simply, it reads in the node's entire blockchain and packages it into a form that can be sent over the TCP Synchronisation socket established earlier.

```
def sendBlockchain(self, s):
    if os.path.exists("blocks/blockchain.chain"):
        print(f"[BLOCKCHAIN] Sending blockchain to {self.__host}...")
        # Open this node's copy of the blockchain for reading
        blockchain_file = open("blocks/blockchain.chain", "r")
        read_file = blockchain_file.read()
        file_size = hex(len(read_file))  # Get file size in hex
        s.sendall(bytes(file_size + read_file, encoding="UTF8"))  # Send data
        blockchain_file.close()
```

4.2.2.3 Mail Broadcast Mechanism

Server Side

Below is the primary method responsible for adding new transactions (emails) to the network.

Note: this is not the initial ingest point for mail, but is executed when a neighbouring node broadcasts a received email.

First of all, the current block file is read in, along with the contents of the mail received. It is then added to the “mail” attribute of the block, and written back. After this, the normal process of block commitment to the blockchain will occur at the specified interval.

```
def addToBlock(self):
    block_num = Block.current_block_name
    print("[BROADCAST] New Email Received, Updating...")
    block_read_in = json.load(open(f"blocks/{block_num}.block", "r"))  # Working block
    temp_mail_file = json.load(open(f"{self.__tmp_file_name}", "r"))  # Received mail
    block_read_in["mail"].append(temp_mail_file)
    block_file = open(f"blocks/{block_num}.block", "w")
    json.dump(block_read_in, block_file)  # Update block with new mail
    block_file.close()
    print("[BROADCAST] Block Successfully Updated!")
```

Client Side

The code below is executed when an email is sent by a user. It packages the mail JSON received from the frontend into a string to be sent over the TCP Broadcast Socket established earlier.

```
def broadcastTransaction(self, s):
    mail_to_send = json.dumps(self.__mail)  # Convert to string ready for send
    file_size = hex(len(mail_to_send))  # Get file size in hex
    print(f"[BROADCAST] Broadcasting Email...")
    s.sendall(bytes(file_size + mail_to_send, encoding="UTF8"))  # Send data
```

4.2.3 KEY DATA STRUCTURES

All structures below are in JSON, which is easy to work within both JavaScript (in which it is native) and Python, where the structure can be treated as a dictionary. The below are in an order of hierarchy, each being a part of the preceding.

4.2.3.1 Mail

Every email sent takes the following structure. This also applies for when the user signs up and the user's public key is added. In such a case, the *recv_addr* will become "0x0" and *body_receiver* will contain the public key.

```
{"send_addr": "",
 "recv_addr": "",
 "subject_receiver": "",
 "subject_sender": "",
 "body_receiver": "",
 "body_sender": "",
 "datetime": "",
 "origin_node": ""}
```

- **send_addr** – The address of the sender.
- **recv_addr** – The address of the receiver.
- **subject_receiver** – The encrypted subject, using the receiver's public key.
- **subject_sender** – The encrypted subject, using the sender's public key.
- **body_receiver** – The encrypted body, using the receiver's public key.
- **body_sender** – The encrypted body, using the sender's public key.
- **datetime** – The date / time the email was sent.
- **origin_node** – The node which initially processed the email.

4.2.3.2 Blocks

A new block will take the following structure, and is generated at the block interval. This is contained in a file with the name: *BLOCK_NUM.block*

```
{"block": "b0",
 "mail": [ {"send_addr": "",
            "recv_addr": "",
            "subject_receiver": "",
            "subject_sender": "",
            "body_receiver": "",
            "body_sender": "",
            "datetime": "",
            "origin_node": ""
           },
            ...
         ]
}
```

- **block** – The identifier for the block.
- **mail** – A collection of all mail in the current block.

4.2.3.3 Blockchain

The blockchain data structure is at the top of the data structure hierarchy. It is a collection of committed blocks, each labelled with the block identifier.

```
{"b0" : {"block": "b0",
         "mail": [ {"send_addr": "",
                    "recv_addr": "",
                    "subject_receiver": "",
                    "subject_sender": "",
                    "body_receiver": "",
                    "body_sender": "",
                    "datetime": "",
                    "origin_node": ""
                    },
                     ...
                  ],
          "hash_digest": ""
         },
         ...
}
```

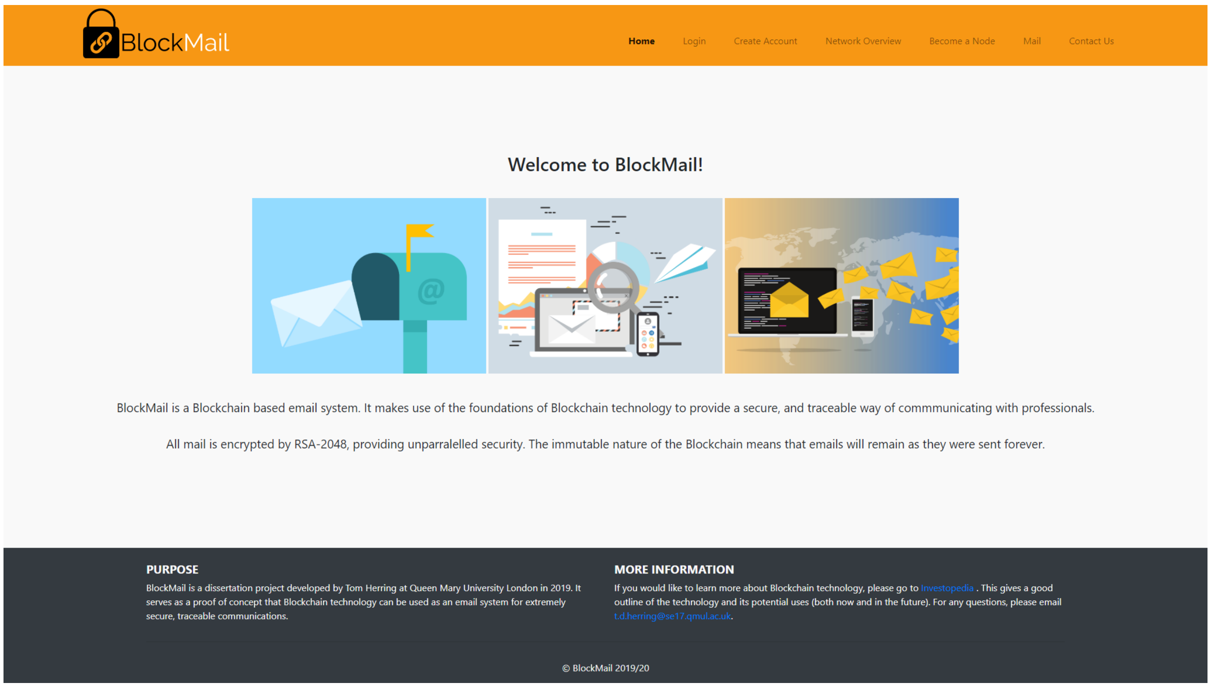
In each block, there is one additional attribute added at the commit stage:

- **`hash_digest`** – The SHA256 digest of the entire block.

All other elements remain the same from the block and mail structures.

4.3 FRONTEND INTERFACE

4.3.1 DESIGN CHOICES



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 RSA-2048, providing unparralelled security. The immutable nature of the Blockchain means that emails will remain as they were sent forever.

PURPOSE

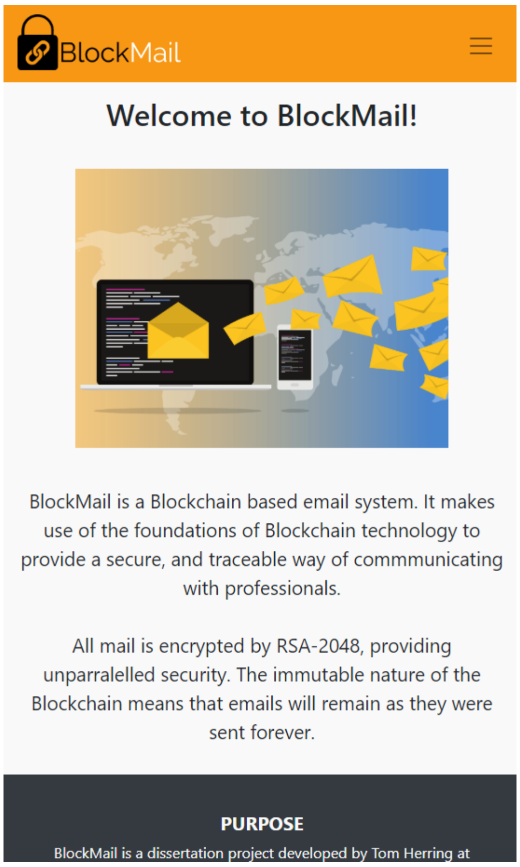
BlockMail is a dissertation project developed by Tom Herring at Queen Mary University London in 2019. It serves as a proof of concept that Blockchain technology can be used as an email system for extremely secure, traceable communications.

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 2019/20

Figure 5a – Desktop 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 RSA-2048, 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

Figure 5b – Mobile Frontend Interface

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

Figure 5a 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 5b 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.

Since submission of the interim report, the colour scheme has been adjusted to have a grey footer and light grey background. This makes the interface far less straining on the eyes.

4.3.2 INTERFACE WALKTHROUGH

The diagram to the below gives a brief overview of the frontend interface structure. Every page is visible from each other page, and the user is put on the "Home" page when they first load the site. The function of each page is as follows:

- ***Home*** – Outlines the function of the site briefly.
- ***Login*** – Used by users to access their mailbox (still encrypted until decrypted using their mail private key).
- ***Create Account*** – Generates key pairs for login and mail encryption / decryption, and allows for users to download them.
- ***Mail*** – The main interface with the blockchain. Allows users to send, receive and decrypt mail.
- ***Contact Us*** – A page to get in contact with me.
- ***Become a Node*** – Allows a site visitor to download the node client.
- ***Network Overview*** – A overview of the network operations and all of the nodes contributing to it.

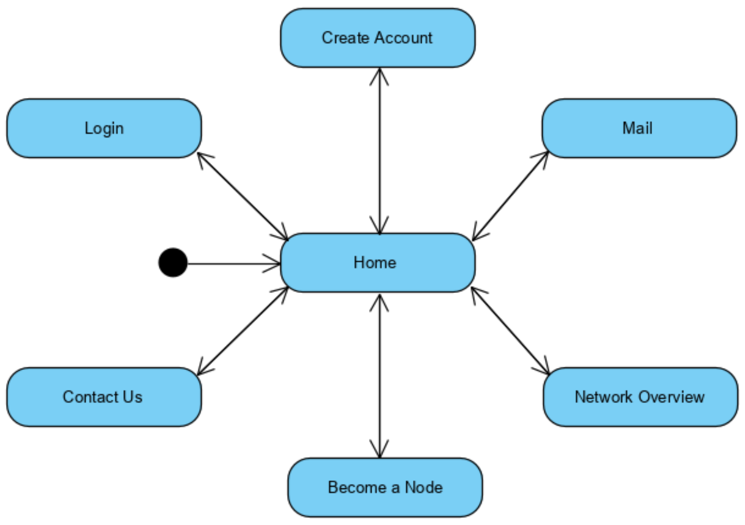
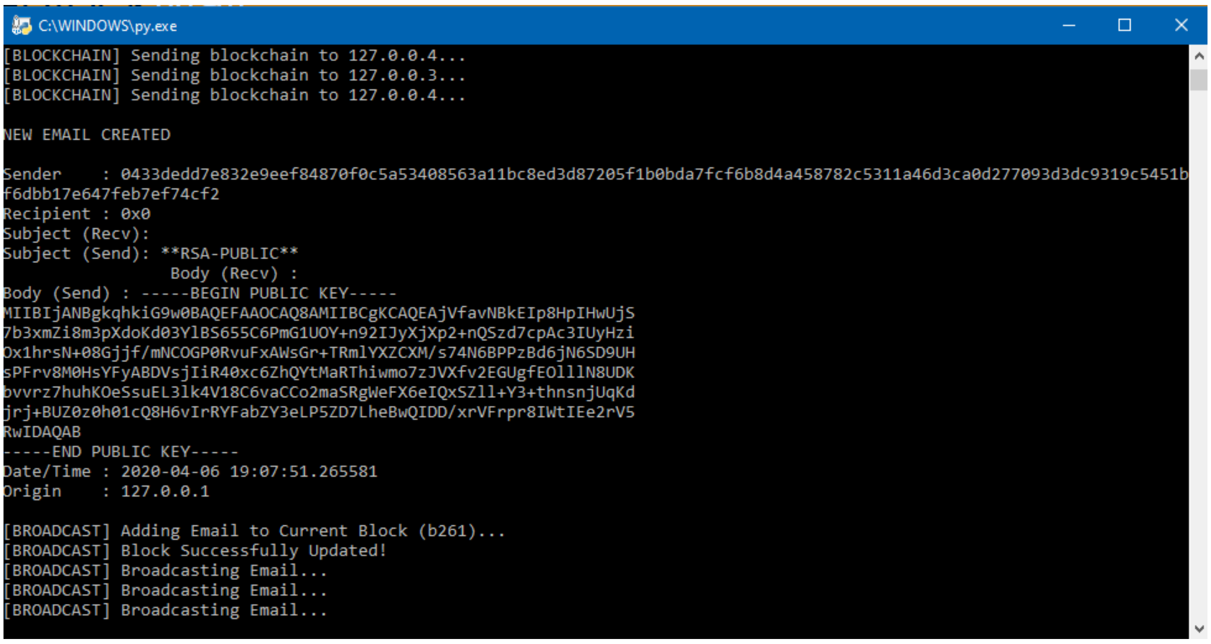


Figure 6 – Interface Navigation Diagram

4.4 BACKEND INTERFACE

Since the majority of users will be interacting with the frontend rather than the backend of BlockMail, a command-line based node client was chosen to be the best path forward. The interface provides only essential information to the user running it, allowing them to debug and view communications between other nodes. *Figure 7* shows an example of what a user running a node may expect to see during its operation.



```
[BLOCKCHAIN] Sending blockchain to 127.0.0.4...
[BLOCKCHAIN] Sending blockchain to 127.0.0.3...
[BLOCKCHAIN] Sending blockchain to 127.0.0.4...

NEW EMAIL CREATED

Sender    : 0433dedd7e832e9eef84870f0c5a53408563a11bc8ed3d87205f1b0bda7fcf6b8d4a458782c5311a46d3ca0d277093d3dc9319c5451b
f6dbb17e647feb7ef74cf2
Recipient : 0x0
Subject (Recv):
Subject (Send): **RSA-PUBLIC**
                 Body (Recv) :
Body (Send) : -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjVfavNBkEIp8HpIHwUjS
7b3xmZi8m3pXdoKd03YlBS655C6PmG1UOY+n92IJyXjXp2+nQSzd7cpAc3IUyHzi
Ox1hrsN+08Gjjf/mNCOGP0RvuFxAWsGr+TRmlYXZCXM/s74N6BPPzBd6jN6SD9UH
sPFrv8M0HsYFyABDVsjIiR40xc6ZhQYtMaRThiwmo7zJVXfv2EGUgfEOlllN8UDK
bvvrz7huhKOeSsuEL3lk4V18C6vaCCo2maSRgWeFX6eIQxSZll+Y3+thnsnjUqKd
jrj+BUZ0z0h01cQ8H6vIrRYFabZY3eLP5ZD7LheBwQIDD/xrVFrpr8IWtIEe2rV5
RwIDAQAB
-----END PUBLIC KEY-----
Date/Time : 2020-04-06 19:07:51.265581
Origin    : 127.0.0.1

[BROADCAST] Adding Email to Current Block (b261)...
[BROADCAST] Block Successfully Updated!
[BROADCAST] Broadcasting Email...
[BROADCAST] Broadcasting Email...
[BROADCAST] Broadcasting Email...
```

Figure 7 – Account Creation on Node Client

CHAPTER 5: EVALUATION

5.1 TESTING

Several tests were carried out in order to determine whether the system is working as expected. They have been broken down into both frontend tests and backend tests. Comments are included where there is notable information relating to how the test was carried out, or how the test was able to pass.

5.1.1 FRONTEND

Test	Passed?	Comments
Only someone who possesses the wallet public AND private ECDSA key can log-in and send mail.	Yes	
Even if the EC keys are valid, only addresses that are registered in the BlockMail blockchain can log in.	Yes	Checked against blockchain when login requested.
A user can only decrypt their mail using their RSA private key corresponding to the account.	Yes	
Key generation happens within a reasonable amount of time.	Yes	Depends on Processing power, but tested on mobile devices and PCs, no greater than 4 seconds.
Duplicate login / decryption keys cannot be generated.	Yes	Considered mathematically impossible due to key size.
The interface is responsive (for both desktop and mobile).	Yes	Website uses Bootstrap.

5.1.2 BACKEND

Test	Passed?	Comments
All nodes hold the most up to date version of the blockchain at all times (except if synchronising with the network).	Yes	Mail broadcast allows for updating of blocks and blockchain.
When a new node joins the network, it correctly downloads and constructs the received blockchain.	Yes	
If, during synchronisation, different sized blockchains are received from nodes, the node will select the largest (in terms of bytes).	Yes	Tested with a number of different nodes with different blockchain sizes.
If a node cannot connect to a server on another node, it will automatically try again in 10 seconds.	Yes	
The client will not start if another instance is running on the host (with the same IP), or another application is using the required ports.	Yes	Shows an error.
The client will not start if it fails to synchronise with an NTP server.	Yes	Shows an error telling user to synchronise their local time.
Can handle large JSON files without significant slow-down or crashing.	Yes	Used ijson Python library for progressive read-in.
The network is scalable.	Yes	Tested with 16 node clients.

5.2 CHALLENGES FACED

5.2.1 DISTRIBUTION AND REPLICATION OF THE BLOCKCHAIN

As blockchain systems are distributed, it is natural for considerations to be made surrounding the distribution and replication of the blockchain across multiple nodes as they join and leave the network. This was a particular challenge as thought had to go towards the amount of overhead required to facilitate synchronisation.

In the case of BlockMail, the option chosen was as follows:

1. Upon node start, it connects to its neighbouring nodes.
2. Each of the neighbouring nodes, send their copy of the blockchain to the node.
3. The node evaluates the size of the received blockchain and chooses the largest.

Among others, an alternative approach explored was to package a copy of the most up to date blockchain with the node client, so that when downloaded from the website, the node would be "ready to go". There are several disadvantages to this approach, however:

- It introduces a single point of failure being the web server storing the most up to date blockchain. If this was modified in an unauthorised manner it could lead to an invalid blockchain being written, with the possibility of email modification or deletion.
- It requires an additional channel of communication between the frontend and backend to keep the blockchain updated.
- If downloaded and not run immediately, the blockchain packaged with the node client quickly becomes outdated.

5.2.2 MANAGEMENT OF THE EXPANDING BLOCKCHAIN

BlockMail is by nature scalable. Therefore, considerations had to be made when deciding how to work with large JSON files as they expand.

In the BlockMail node client, a module called `ijson` was used to facilitate this. It allows for progressive reading of JSON files rather than loading the entire file into memory as the pre-packaged `json` module does. The module treats JSON files as a stream of information, rather than parsing the entire file and then storing it into memory. This is required as reading a huge JSON file in (possibly of 100MB+) would lead to huge slowdowns and crashes as nodes become overwhelmed.

5.3 POINTS FOR IMPROVEMENT

5.3.1 DATABASE INSTEAD OF JSON

Using a database rather than JSON would have made the management of the blockchain easier since the considerations mentioned in 6.2.2 would not be necessary. This is because adding a new email to the blockchain would no longer require reading in the entire blockchain, but would simply only need a database write operation – something that is much more efficient.

A further benefit is that some relational database management systems, such as SQLite, allow for caching of data. This could be incredibly useful, for example when frequently receiving requests to get all mail for an address.

On the other hand, problems could arise with sharing the database, as it would have a much larger file size than a simple JSON, which, as a type of semi-structured data, only requires the length of the text in bytes to be stored. A side-effect of this could be undesirable download times for new nodes joining the BlockMail network.

5.3.2 A BETTER TIME SYNCHRONIZATION MECHANISM

In BlockMail, the time is synchronized with a Network Time Protocol (NTP) server. Although this allows the BlockMail network to operate, it introduces an external system to be relied upon. If this system were to go offline, no nodes would be able to start. Furthermore, if the system became inaccurate (either intentionally or unintentionally), it would lead to all nodes on the network being out of sync.

A better synchronization method would not introduce external factors and would solely be handled internally to the network. For example, many BlockMail controlled reference time servers being averaged to provide a reliable and accurate time.

CHAPTER 6: CONCLUSION

6.1 WAS THE AIM MET?

The primary aim of the project was to *explore an alternative application of blockchain networks, by creating a decentralized email system with a direct focus on reliability, traceability, and security* (1.3).

The decentralized nature of the BlockMail network means that the system is reliable, by ensuring no reliance on any one node. If a node goes offline, the network will continue to function normally, simply exchanging mail between the remaining nodes. If the node comes back online, it will simply update with no detriment to the operation of the network.

In terms of traceability, the immutable nature of the blockchain data structure means that all messages can be traced back to one address. This can be proved since an adversary can't modify an email in transit, as to do so after commitment to the blockchain would invalidate it after that point.

Finally, security has been achieved through the use of Elliptic Curve Cryptography to secure email addresses, and RSA-2048 to encrypt mail itself. Overcoming either of these would take millions of years given current computing power.

6.2 APPLICATIONS OF BLOCKMAIL

As the world continues to evolve, and security and high-availability are placed in the spotlight, it has become clear that going forward it is a possibility that this application of blockchain technology is viable, and could be implemented on a larger scale.

Possible applications in the future include:

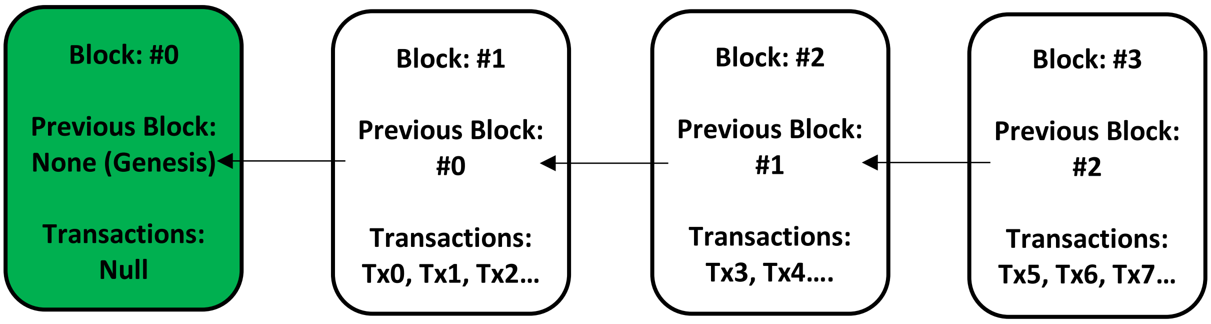
- Private communications between parties in the government sector.
- Situations where anonymity is required.
- Communications where there needs to be a paper trail.

Of course, there is also the possibility of this (and other) blockchain-based technologies becoming much more widespread, and replacing conventional existing technologies. However, this looks rather unlikely in the short term due to the large operational difficulties that arise when working with blockchain.

6.3 CLOSING STATEMENT

Overall, I am extremely happy with the outcome of the project, being the largest and most difficult software engineering task that I have undertaken to date. It has allowed me to apply my knowledge developed throughout my degree, particularly that gained in the Internet Protocols and Applications and Security Engineering modules. Moreover, the project has pushed me as a Computer Science undergraduate, providing me with something to show my knowledge and ability to potential future employers.

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].
[17] Node.js. (2020). About | Node.js. [online] Available at: https://nodejs.org/en/about/ [Accessed 15 Mar. 2020].
[18] Browserify.org. (2020). Browserify. [online] Available at: http://browserify.org/ [Accessed 15 Mar. 2020].