Newcastle University School of Computing

Investigating Hyperledger Fabric as a Platform for Blockchain Based E-Voting

Luke Powlett
140275937
BSc Computer Science (with Industrial Placement)
Supervised by Dr Feng Hao
Word Count: 13,130

April 2018

"It is enough that the people know there was an election. The people who cast the votes decide nothing. The people who count the votes decide everything"

Joseph Stalin[1]

[1] Reported from Stalin in Memoirs of the Former Secretary of Stalin (Bazhanov, 2002)

Abstract

Due to its properties including transparency, verifiability, immutability and decentralisation, Blockchain has generated a huge amount of interest; both as a technology in its own right, and due to its potential when integrated into a wide range of existing industries. Hyperledger Fabric, IBM's leading contribution to the Linux Foundation's open-source Hyperledger Project, brings us a permissioned version of the traditional permission-less Blockchain with a focus on extending Blockchain's adoption away from the cryptocurrencies that brought it to existence, toward more industry focused implementations. In this dissertation, I investigate Hyperledger Fabric's suitability in the cutting-edge area of Electronic Voting.

Declaration

I, Luke Powlett, declare that this dissertation represents my own work except where otherwise stated.

Acknowledgements

I would firstly like to thank Dr Feng Hao for supervising me throughout this dissertation project, providing an invaluable expertise in Cryptography and more specifically the field of Electronic Voting. I would also like to thank the IBM Blockchain team at IBM Hursley for my initial introduction to Blockchain and Hyperledger Fabric; specifically Anthony O'Dowd (STSM, IBM Blockchain) for taking his time to personally talk me through my early project ideas around Hyperledger Fabric and run through the implementations of some of IBM's Hyperledger Fabric early demonstrations.

TABLE OF CONTENTS

Acronyms/Abbreviations

HLF	Hyperledger Fabric
TA	Tallying Authority
E2E	End-to-End
DRE	Direct-Recording Electronic
DRE-i	DRE with integrity
DRE-ip	DRE with integrity & privacy
P2P	Peer-to-Peer
OVN	Open Vote Network
PoW	Proof-of-Work
PoS	Proof-of-Stake
MSP	Membership Service Provider
Tx	Transaction
DoS	Denial of Service
TPS	Transactions Per Second

[PAGE INTENTIONALLY LEFT BLANK]

PART 1 - BACKGROUND

Chapter 1

Introduction

This chapter provides a high-level introduction to the project, including my motivation for such a project and outlining the key concepts and technologies upon which the entirety of the project and dissertation are fundamentally built.

1.1 Background and Motivation

On Election Day, we head to our nearest polling station, collect our ballot paper, mark our vote, and then insert our completed ballot into the ballot box. At this point, we lose sight of the ballot, and we trust that our vote will be counted by the election authority; we don't have any real assurance that the vote we cast *will* be counted, but we rely on the integrity of the tallying authority to ensure a fair election. Where the central authority cannot be trusted, though, this lack of assurance can provide a substantial dispute in the aftermath of an election.

Since the first country-wide use case of Internet voting in Estonia (Madise & Martens, 2005) back in 2005, governments across the world have experimented with the adoption of electronic voting systems, whether that be Internet based (allowing voting from a remote location) or using Direct-Recording Electronic (DRE) machines at a polling station. However, these implementations still require trust in a central authority to correctly tally the votes, also providing another, in some cases easier, opportunity for any corrupt authority to tamper with the electronic vote record.

In general, Blockchain's provide a distributed, public ledger of all transactions that have been executed on the network, with an identical replica of the Blockchain state being shared among its peers. Rather than trusting a central authority to validate the transactions (i.e. count the votes), each transaction is verified by consensus of a majority of participants on the network, and so is essentially self-enforcing (as opposed to enforced by a central authority). In addition, once a transaction is written to the Blockchain, it is immutable, and so can never be modified or erased, leading to a certain and verifiable record of every transaction ever made on the network. I believe that these characteristics of Blockchain technologies make it an ideal area to explore for the implementation of an improved Electronic Voting system.

In 2016, The Economist and Kaspersky ran a competition for US and UK academic institutions to improve the security of E-Voting. I will review 2 of the top proposals, both of which make use of Blockchain: Votebook (Kirby, Masi, & Maymi, 2016) – a design based on a permissioned ledger; and Removing Trusted Tallying Authorities (McCorry, Toreini, & Mehrnezhad, 2016) – proposing a 'Digital Voting' protocol which can be run on the popular existing Ethereum (Ethereum, 2017) Blockchain. The findings of these proposals, and particularly their respective concerns, will form the basis of my investigation.

1.2 Aim and Objectives

The overarching aim of my project is to explore the suitability of Hyperledger Fabric (HLF) as a platform for an E-Voting system, which can be broken down into the following 3 objectives:

I. Perform a critical feature analysis of HLF to ensure it is a feasible platform on which to develop an electronic voting system, in comparison with other popular existing Blockchains
II. Develop a proof-of-concept E-Voting system running on HLF to explore the practical implementation of the suggested system
III. Review of feature analysis findings and proof-of-concept implementation to give a final critical overview of HLF's suitability

1.3 Key Sources

For the projects background - The project builds upon the findings of the Blockchain E-Voting Challenge (discussed in section 2.3), specifically 2 of the competitions leading proposals: Removing Trusted Tallying Authorities (McCorry, Toreini, & Mehrnezhad, 2016) and Votebook (Kirby, Masi, & Maymi, 2016). Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting (Hao, et al., 2017) provides the project with a good grounding in the security requirements of an E-Voting system.

In terms of exploring the architecture of the Hyperledger Fabric – Rethinking Permissioned Blockchains (Vukolic, 2017) and Architecture of the Hyperledger Fabric (Cachin, 2016) provide an in-depth insight into the architecture of the first official Version 1.0 release of HLF and the architecture of the initial HLF proposal respectively.

For the proof-of-concept implementation – The Hyperledger Projects online tutorials covering building a Fabric Network (http://hyperledger-fabric.readthedocs.io/en/release-1.1/tutorials.html) and developing a 'Business Network' to run on the fabric using Hyperledger Composer (https://hyperledger.github.io/composer/latest/tutorials/tutorials) provide a solid grounding.

1.4 Structure of this Dissertation

The dissertation is divided into 3 Parts, with a total of 7 Chapters:

PART 1 - BACKGROUND

Chapter 1 - Introduction

I begin by introducing the project, providing a high-level overview of both E-Voting and Blockchain, before presenting the projects Aim and Objectives and discussing some key sources used throughout the project and dissertation. This should set the context and provide a good grounding in the fundamental building blocks and desired outcomes of the project, on which I build throughout the remainder of the dissertation.

Chapter 2 – Background and State of the Art

Here I review the background of the projects two core focuses – E-Voting and Blockchain, before exploring the relatively new work combining the two, discussing the state of the art in Blockchain E-Voting. This allows me to identify the strength and weaknesses of existing work in the field, making the most of the strengths and the weaknesses highlighting areas in which my project should aim to improve.

PART 2 – WHAT I'VE DONE

Chapter 3 – Hyperledger Fabric

The first objective of my project is to explore Hyperledger Fabric's functionality and architecture to discuss it's suitability as an E-Voting platform from a theoretical perspective. Chapter 3 documents my exploration into the architecture of HLF, as well as the tools available to aid in the development of applications to be run on the fabric, all with the end context of an E-Voting system. This aims to provide a solid, in depth understanding of the fabric and contributes substantial evidence toward the feasibility of HLF as a platform for secure E-Voting.

Chapter 4 – In Theory: A Secure E-Voting System for Blockchain

Building on the exploration of HLF's architecture in the previous chapter, Chapter 4 proposes a full secure E-Voting procedure within the full context of a UK National Election. This again emphasises HLF's suitability in terms of the projects aim, providing motivation and a solid grounding for the practical implementation to come in the following Chapter.

Chapter 5 – In Practice: A Proof-of-Concept

Having fully explored and proposed a HLF based E-Voting system in the preceding chapters, this Chapter further explores the tools required to develop HLF applications, before presenting the design and implementation of Votechain: my proof-of-concept implementation of the E-Voting system proposed in Chapter 4. This provides a tangible representation of the end-to-end verifiable E-Voting system to demonstrate the projects aim, as well as highlighting some issues faced with the current fabric release.

PART 3 – EVALUATION

Chapter 6 – Hyperledger Fabric as a Platform for Blockchain Based E-Voting

Drawing closer to the end of the dissertation, Chapter 6 reviews my investigation into HLF as a platform for secure Blockchain based E-Voting and my proof-of-concept E-Voting system running on the latest version of HLF, first in terms of each of Hao *et al's* 6 security recommendations for E-Voting systems (as discussed in Chapter 2), before conducting an evaluation of the projects findings in terms of the projects main objectives. This discusses both the positive findings of the project, as well as some of the struggles and concerns faced in fully exploring the projects aim.

Chapter 7 – Future Direction

The final Chapter of the dissertation proposes further investigation and research to follow on from the findings and results of this project, encompassing both areas the project couldn't cover as a result of this project and Hyperledger Fabric's current constraints, further testing to explore the proposal at

a true scale closer to that of a national election, and utilising some of Hyperledger Fabric's customisable features to explore and optimise the fabric network, specifically in terms of an E-Voting context.

I close the dissertation with a conclusion of the project and its results as a whole, briefly discussing my experiences throughout the project.

Chapter 2

Background and State of the Art

In this chapter I will introduce and discuss the key concepts on which my project is founded and the state of the art in the Blockchain and E-voting field's.

2.1 An Introduction to E-Voting

In the research of e-voting, systems have generally been broken down into two categories: smaller boardroom scale elections and larger national scale elections. In this dissertation, I will be focusing on the latter.

Traditional paper based elections require a trusted tallying authority (TA) to count the ballots. This may be accepted where the TA is a trusted authority (as it is here in the UK), but in many countries with less stable authorities, the outcome of an election is left open to scrutiny.

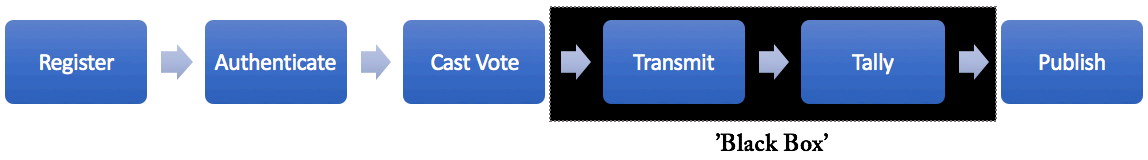
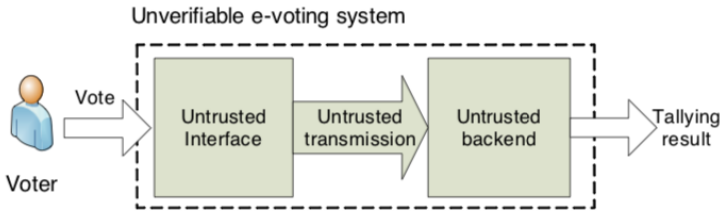


Figure 1 – A traditional voting process

Figure 1 breaks down the steps involved in a traditional voting system. In a paper ballot based election, once we hand over our ballot paper, our ballot enters a 'black box', where we assume the vote will a) reach the tallying authority and b) be correctly counted. Figure 2 further breaks down this black box to show the points of vulnerability in an unverifiable e-voting system.

Figure 2 –Taken from Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting (Hao, et al., 2017)



The state of the art in e-voting is the provision of an end-to-end (E2E) verifiable voting mechanism. This means providing a solution in which the voter can verify that the vote was cast as expected at any stage throughout this 'black box'. That is the vote cast is the same as the vote intended, the vote recorded is the same as the vote cast, and the vote tallied is the same as the vote recorded.

Many modern e-voting solutions use a Direct-Recording Electronic (DRE) machine for the recording of votes. Hao *et al* present the DRE with Integrity (DRE-i) and DRE-i with privacy (DRE-ip) protocols. Whilst these protocols are E2E verifiable, they still rely on a TA for the counting of votes. In this dissertation, I am investigating a 'self-enforcing' voting mechanism – that is,

an E2E verifiable system capable of delivering a trusted election outcome **without** the need for a trusted TA.

In the report, Hao *et al* outline some key generally assumed security requirements for a secure e-voting system. I will use the same assumptions throughout my own project:

> *(1) User enrolment: Only eligible users can be enrolled in the voter registration.*
> *(2) User authentication: Only authenticated voters are allowed to vote during the election.*
> *(3) One-man-one-vote: Each authenticated voter is allowed to vote just once.*
> *(4) Voting privacy: Voting happens in a private space that no one else can observe.*
> *(5) Anonymity: The voting machine that is used does not know the real identity of the voter.*
> *(6) Public Bulletin Board: There is a publicly readable, append-only bulletin board (e.g. a mirrored public website) on which the legitimate voting system can publish audit data for verification (the authenticity of data can be ensured by the use of digital signatures).*

(Hao, et al., 2017)

2.2 An Introduction to Blockchain

In 2009, an unknown person or group, 'Satoshi Nakamoto', released a whitepaper (Nakamoto, 2009) titled 'Bitcoin: A Peer-to-Peer Electronic Cash System'. This paper describes what went on to become a revolutionary technology which laid the foundations of Bitcoin - the world's first peer-to-peer (P2P) and fully decentralised cryptocurrency – The Blockchain.

2.2.1 Bitcoin

The abstract of Nakamoto's paper gives a fantastic summary of the principals that make Blockchain (the technology underpinning Bitcoin) such a powerful technology:

> *"[Bitcoin] A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem[2] using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work."*

2.2.2 Blockchain

The paper goes on to explain in depth the workings the Blockchain; the core principals of which I shall explain as follows:

- **Distributed Public Ledger** – A Blockchain can essentially be thought of as a distributed database of records, or a shared public ledger of all transactions executed on the network
- **Verified by Consensus** – In Nakamoto's Bitcoin Blockchain, and many other popular Blockchain implementations since (Ethereum (Ethereum, 2017), for example), the requirement for a trusted third party to verify transactions and prevent the double-spending problem has been solved through decentralised consensus – that is the standard & rules by

[2] Double-spending problem – Ensuring that any asset/representation of value cannot be 'spent' more than once

which every node exchanges information, the mathematical rules for all nodes to agree on the integrity of the data (often referred to as Proof-of-Work (PoW)) and the payment incentive to support the consensus model. PoW and other methods of reaching consensus will be explored further in Chapter 3.

- **Immutable** – Once a transaction has been written to the Blockchain, it cannot be modified or removed. In a case where reversing a transaction would be desirable, this must be done by making a separate transaction, where both transactions are visible on the Blockchain.
- **Verifiable** – Through its hashing mechanism, which uses a Timestamp server (explained more thoroughly in Nakamoto's paper) where every new timestamp includes the previous timestamp in its hash, forming a chain where each new timestamp reinforces the integrity the preceding timestamps (see Figure 3). This means that each timestamp proves that the data in the block must have existed at the time.

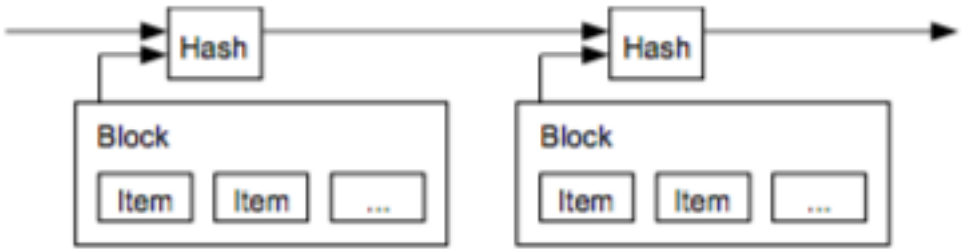


Figure 3 – A Timestamp Server. Taken from Bitcoin: A Peer-to-Peer Electronic Cash System (Nakamoto, 2009)

2.2.3 Suitability for E-Voting

In an election, we want to ensure that each voter is only able to make one vote (the double-spending problem), we want to remove the requirement for a trusted third-party authority (P2P consensus), we want to ensure that the votes cannot be modified/removed once cast (immutability), and we want to ensure that every vote can be verified (verifiable). It is these properties of the Blockchain that I believe make it a great platform on which to build an e-voting system.

2.3 Blockchain E-Voting

In 2016, Kaspersky sponsored a competition[3] hosted by The Economist, challenging academic institutions from the USA and the UK to address the security issues present in the e-voting systems available at the time. Of the 3 prize winning submissions, 2 utilised Blockchain in their implementation. I will discuss both proposals, taking inspiration from both the security enhancements provided and the concerns raised to inform my own proposal in this dissertation project.

2.3.1 Removing Trusted Tallying Authorities (McCorry, Toreini, & Mehrnezhad, 2016)

This 2016 paper, written by a team of Newcastle University PhD students, documents the results of a challenge by The Economist and Kaspersky to design a Blockchain system for digital voting. Their

[3] The original The Economist site for this competition (http://www.economist.com/whichmba/mba-case-studies/cyber-security-case-study-competition-2016) requires authorised access. Kaspersky's advertisement for the competition can be found at https://www.kaspersky.co.uk/blog/economist-kaspersky-contest/7656/ (Kaspersky, 2016). The competition results are discussed in the following article https://www.businesswire.com/news/home/20161208005130/en/Blockchain-Technology-Secure-Digital-Voting-Systems-Kaspersky (Bettencourt, 2016).

proposal demonstrates the Open Vote Network (OVN), a 'decentralised Internet voting protocol, [which] can be run over Ethereum's Blockchain today' (McCorry, Toreini, & Mehrnezhad, 2016).

The OVN protocol is a decentralised 2-round voting scheme, based on Hao *et al's* proposal (Hao, Ryan, & Zielinski, Anonymous Voting by 2-Round Public Discussion, 2010). The original protocol assumes an authenticated public channel available for every voter, with a public bulletin board being the common suggestion to realise such a channel. The practical implementation of such a secure bulletin board, however, was a remaining challenge. McCorry *et al's* proposal, OVN, suggest that the properties of the Blockchain hold the key to resolving this problem.

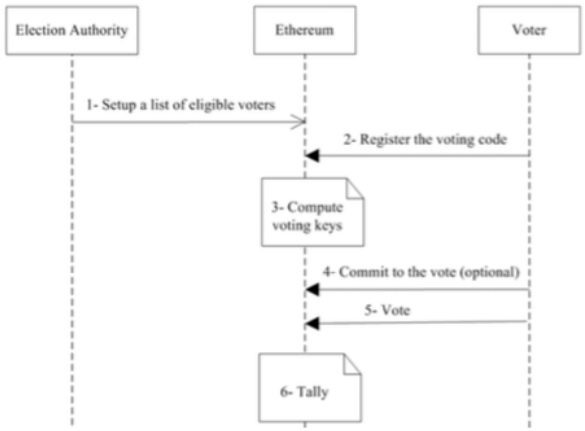


Figure 4 - Sequence diagram of the proposed Implementation. Taken from Removing Trusted Tallying Authorities (McCorry et al)

OVN implements McCorry *et al's* version of the 2-round voting scheme using 'Smart Contracts'[4] on the Ethereum Blockchain (Ethereum, 2017) in Ethereum's native language – Solidity (Ethereum). Their proof-of-concept (outlined in Figure 4) uses 3 different user views; a Voter page where the voter can register and subsequently vote, an Election Authority page where the Election Authority can setup the election environment and initiate/terminate proceedings, and a Live Feed page where anyone is able to view the Blockchain state (allowing voters to verify that their votes have been recorded).

In Ethereum, any computations performed, for example any transactions involved in running a smart-contract, are paid for using the notion of 'gas', which is paid for using Ether (Ethereum's native cryptocurrency). Each block added to the chain has a gas limit which corresponds to the maximum amount of allowed computation. This gas limit, and Ethereum's PoW algorithm, posed several severe concerns to the suitability of Ethereum, or any PoW consensus based Blockchain, for an E-Voting system, especially on a national scale.

Concerns

For a test election with 40 voters (McCorry, Toreini, & Mehrnezhad, 2016)

- 125 Ethereum transactions were required
- With 1 block being generated every ~12 seconds
    - Only 6 voters could register for the election
    - Only 1 vote can be cast per block using the Ethereum network
        - Only 5 votes could be cast per minute (not scalable for a national election)
- Average cost per voter: £0.69 (lower than typical UK election cost – see Appendix I: Cost Analysis of UK Election's)
- Total election cost: £27.72
- For these reasons, McCorry *et al* propose OVN for small-scale 'boardroom' style elections, with DRE-i and DRE-ip to be implemented similarly for large-scale Internet voting and polling station voting respectively.

Ethereum follows the original Bitcoin (Nakamoto, 2009) Blockchain design in being a permission-less ledger, meaning it is designed for *anyone* to be able to be a part of the Blockchain network. PoW

[4] Smart Contract – A program to directly control the transfer of value between participants through given conditions

consensus requires intensive computing, and thus cost, to limit permission to modify the Blockchain. An election is not a permission-less event (voters must be eligible and registered to vote).

I believe that a Permissioned[5] Proof-of-Stake (PoS) based Blockchain would be a more suitable choice for an election environment. This idea forms the basis of the investigation into Hyperledger Fabric as a Blockchain platform, and will be further discussed in Chapter 3.

2.3.2 Votebook (Kirby, Masi, & Maymi, 2016)

The first-place proposal of the competition, Votebook, '*borrows the most important ideas from blockchain technology but in a slightly altered format, called a "permissioned blockchain."*' (Kirby, Masi, & Maymi, 2016). Hyperledger Fabric, which will be discussed in much greater detail in Chapter 3, is built on the concept of a permissioned ledger Blockchain, and so Kirby *et al's* proposal aligns much more closely with my project than that of McCorry *et al.*

Kirby *et al* propose a network of 'state-identified voting machines', with each of the voting machines acting as a node on the Blockchain network. Each node (I use the terms 'node' and 'voting machine' interchangeably) on the network must have permission to make changes to the Blockchain prior to the election. Each of the voting machines then generate a private-public key pair, which is shared with the central election authority. The election authority can then compose a table of each node's public key, which is distributed to every node on the network (to be used in verification of any node's incoming block's). During the election, each node will collect user votes, continuously organising them into a Block containing voters and their corresponding votes, a hash of the previous block, and digital signature (encryption of a hash digest of the new Block), as demonstrated in Figure 5. Each Block is then proposed to the Blockchain network in accordance with a pre-defined time-based protocol, with each receiving node verifying the new Block using the shared table of public key's. If the Block is verified, it will be added to the Blockchain.

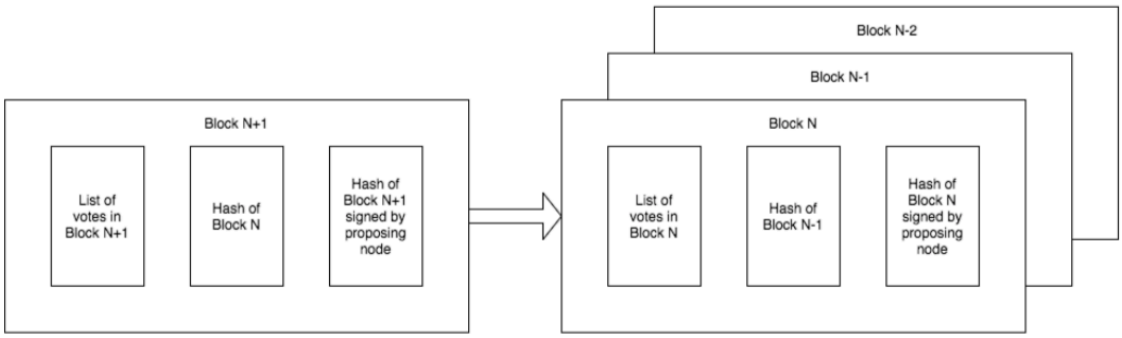


Figure 5 - Demonstration of the proposed Votebook Blockchain. Taken from Votebook (Kirby, Masi, & Maymi, 2016)

Votebook proposed a primitive version of a permissioned Blockchain by using a central election authority to oversee the verification of each voting machine on the network, and the collection and distribution of each node's public key. I believe that in an election environment, the concept of the traditional open-to-all permission-less Blockchain is redundant; elections are not permission-less – each voter must be both eligible and registered to vote, each polling station and every person counting the votes must be must be authorised, which is why I will base my proposal on a permissioned Blockchain. However, in Votebook, the authorisation of each voter takes place outside of Votebook,

[5] Note that at the time of the competition in 2016, permissioned Blockchain's were in the early stages of exploration/prominence.

which I believe still places too much trust in the central election authority – an issue I hope to address in my own proposal.

2.3.4 Conclusion

From analysis of the proposals above, I suggest a system which

- Makes full use of the core features of the Blockchain (as discussed in 2.2 An Introduction to Blockchain)
- Does not require the intensive computing resource of PoW consensus
- Can be permissioned to the voter level to be fully suitable for the election environment

PART 2 – WHAT I'VE DONE

Chapter 3

Hyperledger Fabric

The Hyperledger Project (Linux Foundation) is an umbrella collaborative effort aiming to create an enterprise-grade, open-source distributed ledger, tailored specifically with a focus on using Blockchain for industrial use cases. Established by the Linux Foundation in early 2016, the Hyperledger Project has over 150 members (see Figure 6 below), spanning a variation of industries, including IBM, Intel and J. P. Morgan.

Figure 6 - Hyperledger Project Members and Associates as of 27/03/18 (Hyperledger Project, 2017)

Hyperledger Fabric (Hyperledger Project, 2017) is IBM's leading contribution to the Hyperledger Project, with the first 'v.05-developer-preview' released in June 2016, and the first official HLFv1 released in July 2017. In this chapter, I will discuss my research into HLF, first giving a high-level overview of its place in the Blockchain market, then diving further into the lower level implementation details of HLF[6], before finally looking at how HLF compares with other Blockchain implementations, specifically focusing on its suitability to an E-Voting system.

---

[6] Here I acknowledge the work of IBM Research, specifically Marko Vukolic (Vukolic, Hyperledger Fabric, 2017) (Vukolic, Rethinking Permissioned Blockchains, 2017) (Vukolic, Sousa, & Bessani, A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform, 2017) and Christian Cachin (Cachin, 2016), as key sources for this section.

3.1 An Introduction to Hyperledger Fabric

On a personal level, I was first introduced to Hyperledger Fabric (and the whole world of Blockchain for that matter) in March 2017, a few months prior to the initial developer preview release, at an IBM internal presentation during a Software Development placement at IBM's largest UK Research & Development lab in Hursley, Winchester. Even at this early stage, it was clear that IBMs key focus was on moving away from cryptocurrency-centric Blockchains toward developing a Blockchain solution tailored for industry.

Like other Blockchain technologies, it has a distributed ledger and allows participants to manage their transactions. Also, similar to Smart Contracts on the Ethereum Blockchain, HLF is capable of running distributed applications called Chaincodes. Where HLF diverts from most other popular Blockchains is that it is private and permissioned; so where Blockchains like Bitcoin and Ethereum are permission-less - allowing unknown participants to conduct transactions, using PoW protocols to validate - participants in a HLF network are enrolled through a Membership Service Provider (MSP). I believe that this property of HLF lends itself well to an E-Voting use case, where participants (polling stations/voters) should not be completely anonymous as they must at least be registered polling stations/voters. With a permissioned ledger like HLF, we can ensure only authorised machines/users are able to participate in the network, whilst ensuring that anonymity of the voter is maintained in the Chaincode application logic.

3.2 Architecture

The ledger protocol of the fabric is run by two kinds of peers, both requiring permission to participate in the network.

- Validating Peers – Nodes on the network responsible for running the consensus protocol, validating transactions and maintaining the shared state of the ledger
- Non-Validating Peers – Nodes on the network that act as a proxy to connect clients who issue transactions to validating peers – they do not execute transactions but can verify them

Prior to the release of HLFv1, all permissioned Blockchains (including HLFv0.6) followed an ORDER → EXECUTE architecture. This means that smart contract transactions are totally ordered following consensus on input to the state machine. However, the long execution latency of this sequential execution of smart contracts can block other smart contracts from being executed, creating a potential performance bottleneck, or even a smart contract DoS attack by creating infinite loops within the contract code. These Blockchains have generally coped with the heavy computation requirements by requiring nodes to pay for each step of the computation (i.e. the notion of Gas in the Ethereum Blockchain).

3.2.1 HLFv1 Transaction Flow

However, the release of HLFv1 brought a re-architected Fabric, now following an EXECUTE → ORDER → VALIDATE pattern. This means that Chaincodes are executed *before* applying the consensus protocol, with consensus then being used to agree on the propagation of versioned state-updates. Figure 7 below illustrates the full transaction flow in HLFv1. Application developers can specify both the Chaincode and Endorsement Policy (Validation code) of the application.

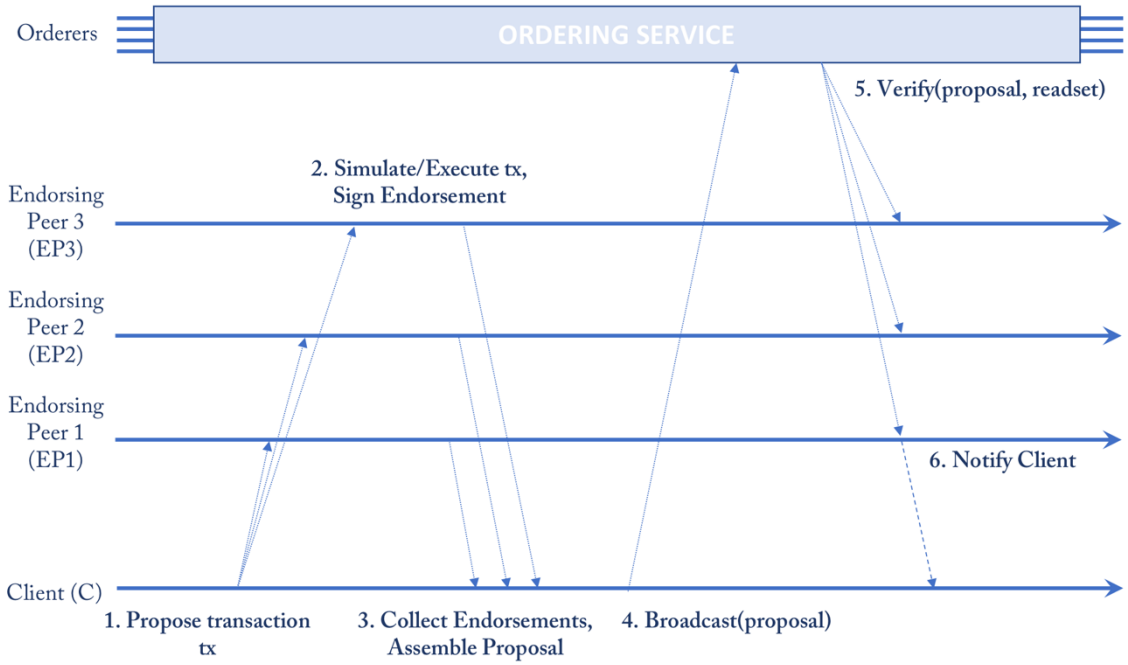


Figure 7 - Hyperledger Fabric V1 Transaction Flow

1. **Propose Transaction tx –** Transaction 'tx' is proposed to the network

```
<PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>
```

2. **Execute tx –** The tx Chaincode is executed by each endorsing peer and the peer signs its endorsement (but any state change is not written to the fabric yet so you could say tx is just simulated). Readset contains all keys read by the Chaincode during execution (a set of existing state). Writeset contains all keys written by the Chaincode along with their new values (a set of state updates).

```
<TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>
```

3. **Collect Endorsements –** A proposal to the network requires K out of N endorsements to successfully endorse tx (where K and N can be set out by the application developers in the applications Endorsement Policy).

4. **Broadcast Proposal –** The endorsed proposal is sent to the ordering service and ordered in line with the applications consensus protocol (HLF features a 'pluggable' consensus, discussed further in 3.2.1 Consensus).

5. **Verify –** All peers on the network validate the state change in accordance with the applications Endorsement Policy.

6. **Notify Client –** The proposing client is notified of the outcome of its proposal.

To summarise, Chaincodes (Smart Contracts) are *not* executed sequentially, improving both the performance and scalability of the network; not *all* peers are required to execute all Chaincodes, again improving scalability; and finally, due to the introduction of deterministic Endorsement Policies, non-deterministic Chaincode execution is not an issue, increasing consistency in the network, as well as

allowing for Chaincodes to now be written in general purpose languages (e.g. Go/Java) where previously this was not possible due to their non-deterministic nature.

3.2.2 Consensus

One of the key principals of Blockchain is the assurance of a transactions validity and finality without the need for a trusted third party. Blockchains like Bitcoin and Ethereum reach consensus on any transactions to be written to the Blockchain by incentivising nodes on the network to 'mine' new blocks. This is achieved by completing resource intensive Proof of Work algorithms (complex mathematical challenges). The first node to solve the challenge shares the result with other nodes, who in turn check both the result of the challenge and the validity of the transaction. Ethereum requires nodes to pay for every stage of a smart contract's computation using the notion of 'Gas' (the financial value of the computation required).

HLF (Vukolic, Sousa, & Bessani, A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform, 2017) and Ripple (Schwartz, Youngs, & Britto, 2014) are two popular alternative Blockchains that move away from compute-intensive PoW consensus toward Proof of Stake (PoS) consensus mechanisms based on algorithmic Practical Byzantine Fault Tolerant (PBFT) State Machine Replication. So where permission-less Blockchains enable any anonymous peer to create new blocks (having solved a mathematical puzzle), permissioned Blockchains rely on a set of trusted peers tasked with creating new blocks, where the order in which blocks are added to the ledger is determined by a BFT protocol.

3.3 Hyperledger Composer

Hyperledger Composer (Hyperledger Project, 2018) is a framework for developing applications to run on HLF. Being targeted primarily at industrial use cases, Hyperledger developed the notion of 'Business Networks' - essentially a way to define a Blockchain application for HLF for any given industry or use case. It focuses on 3 key concepts; Assets (tangible or intangible goods/services), Participants (members of the network) and Transactions (the mechanism by which participants interact with assets – essentially the smart contract logic).

I believe that this idea of Assets, Participants and Transactions is perfect for an e-voting application. If we think of the Business Network as our election, we have voters, candidates, ballots, registrations and votes:

- Assets
    - Ballot
- Participants
    - Voter
    - Candidate
- Transactions
    - Registration
    - Voting

This will form the basis of proof-of-concept design, discussed further in chapters 4 & 5.

3.4 Comparison with Other Blockchains

Table 1 - Blockchain Feature Comparison

	Bitcoin	Ethereum	Hyperledger Fabric
Year	2009	2014	2017
Distributed Applications	❌ Hard Coded Cryptocurrency	✅ Smart Contracts	✅ Chaincodes
Language	Limited Scripting Language	Domain-Specific (Solidarity)	Multiple General-purpose Languages (e.g. Go, Java)
Consensus	Proof-of-Work	Proof-of-Work	Proof-of-Stake (Modular/Pluggable)
Independent of Cryptocurrency	❌ BTC	❌ ETH	✅
Permissioned	❌	❌	✅
Multiple Deployments	❌	❌	✅
REST API Support	❌	❌	✅

Where most popular Blockchains up until this point have focused around the original cryptocurrency use case which gave rise to the Blockchain technology, HLF aims to move away from cryptocurrencies toward extending Blockchain toward more industry focused use cases. With this in mind, HLF implements a permissioned Blockchain, where peers on the network are 'trusted', allowing for a significantly less computationally intensive PoS consensus mechanism (over the more common PoW consensus seen in Bitcoin and Ethereum). Table 1 above provides a taxonomy of some of HLF's core functionality alongside 2 of the biggest players in the Blockchain arena.

3.5 Scalability

To summarise what we have covered so far, HLF's architecture is made up of 3 main components: peer nodes, ordering nodes and client applications. The identities of these components are issued from certificate authorities. A peer in HLF can have one or both of 2 roles: committing nodes maintain the ledger by committing transactions, and endorsing nodes execute chaincodes and endorse the result. The ordering nodes are responsible for deciding the order of transactions in a block which is to be committed to the ledger. The ordering service used is pluggable, with a byzantine fault tolerant ordering service provided by default.

Hyperledger have not released any official figures for the performance of HLF, and with version 1.1 only recently released ordering services are still in the early stages of testing and developing. A report (Vukolic, Sousa, & Bessani, A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform, 2017) by IBM proposes a Byzantine Fault Tolerant ordering service stating 'Our preliminary evaluation, both on a local cluster and in a geo-distributed setting, show that HLF

with BFT-SMART ordering service can achieve up to 10k representative transactions per second and write a transaction irrevocably in the blockchain in half a second, even with consensus nodes spread through different continents'. When transmitting blocks of 400kb in a 10-node cluster, their ordering service reached 'a peak throughput of approximately 2200 transactions/second'. This more than doubles Ethereum's theoretical peak of 1000tps (Transactions Per Second) and massively shadows Bitcoins peak of 7tps. However, performance is not discussed past 10 nodes and every node added up to 10 results in significant performance degradation.

In another paper discussing the performance of different Blockchain Networks (Scherer, 2017), Scherer's tests using HLFv1 paint a similar picture:

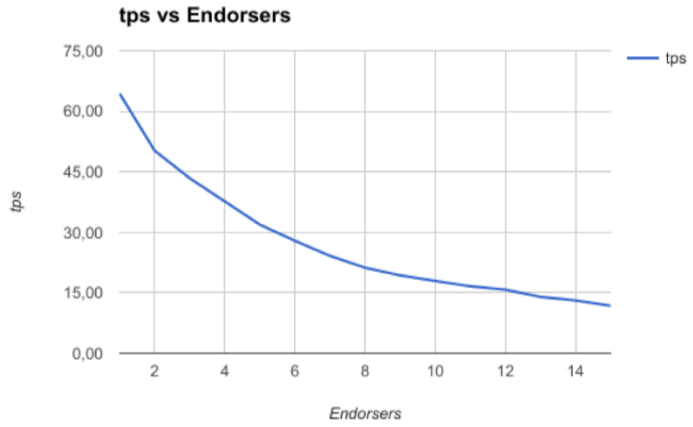


Figure 8 -TPS results of a test on HLFv1 with increasing no of endorsing peers

From his tests, Scherer argues that the service may 'involve as many as $O(n^2)$ messages per endorser, where n is the number of endorsers in the channel'. This current feasible limit of only a handful of endorsers per channel does not necessarily represent poor scalability on the part of HLF though, as HLF does not limit the number of channels which can be running on the Blockchain.

Participants in a HLF network interact in a manner that ensures their transactions are restricted to the participants on their channel, where the channel can run an independent instance of the shared ledger. This means that we could test the TPS throughput of our personal network to find the optimal configuration, and then group our endorsing peers into channels of, say, machines in a polling station, or polling stations within a constituency, depending on the optimal setup. This effectively splits one national scale Blockchain into many smaller private Blockchains, with each channel sharing its own ledger amongst its peers.

In an election scenario where each constituency is responsible for maintaining its own electoral register and the candidates standing, many smaller ledgers (as opposed to one national ledger) sounds like a sensible implementation, whilst also benefitting from reducing the number of peers and its corresponding performance benefits, as increasing the number of peers on a channel will only affect the performance of that channel.

An alternative implementation may be to separate our committing and endorsing peers, so rather than have a voting machine both commit its own transactions and endorse the transactions of other voting machines, the voting machines could act solely as committing peers, and then have 10-15 powerful machines spread around the UK acting as endorsing peers.

Hyperledger Fabric would also be suitable to smaller board-room style voting scenarios, where each participant could act as an endorsing peer, again creating a more decentralised ledger.

3.6 Reflection

This chapter concludes my exploration and critical feature analysis of HLF as a potential platform on which to develop a secure Blockchain based E-Voting system in line with the first of my 3 project objectives. Having found only positive reasons to suggest improvement on previous Blockchain E-Voting implementations, my next step will be to detail a proposed HLF based E-Voting implementation, supported by my own small-scale proof-of-concept E-Voting system.

Objective Reflection

I. Perform a critical feature analysis of HLF to ensure it is a feasible platform on which to develop an electronic voting system, in comparison with existing Blockchain and popular non-Blockchain E-Voting systems

Chapter 4

In Theory: A Secure E-Voting System for Blockchain

In this chapter, I discuss my proposed implementation for a secure E-Voting system running on Hyperledger Fabric, working within the context of a UK Election.

4.1 UK Election Process

As a background on the environment an E-Voting system operating as part of a UK election would be working within, the following figures are taken from the 2015 UK General Election (UK Political Info, 2015) which represented a higher than average turnout.

Total Turnout:	30,513,068
Total Electorate:	46,425,386
Turnout Percentage:	~66%
Seats:	650
Candidates:	3,971
Poll Open Hours:	13
Final Count (hrs from poll close):	~15.5 hours (notably late)

Note that the total turnout and candidates presented here are representative of the quantities of voters and candidates respectively which any proposed system should be capable of handling in a timescale similar to previous elections, where final counts are usually released in the early hours of the morning. Further, any new E-Voting system should integrate with the current electoral process and infrastructure as much as possible to cause minimal impact. Figure 9 illustrates the current UK Election process:

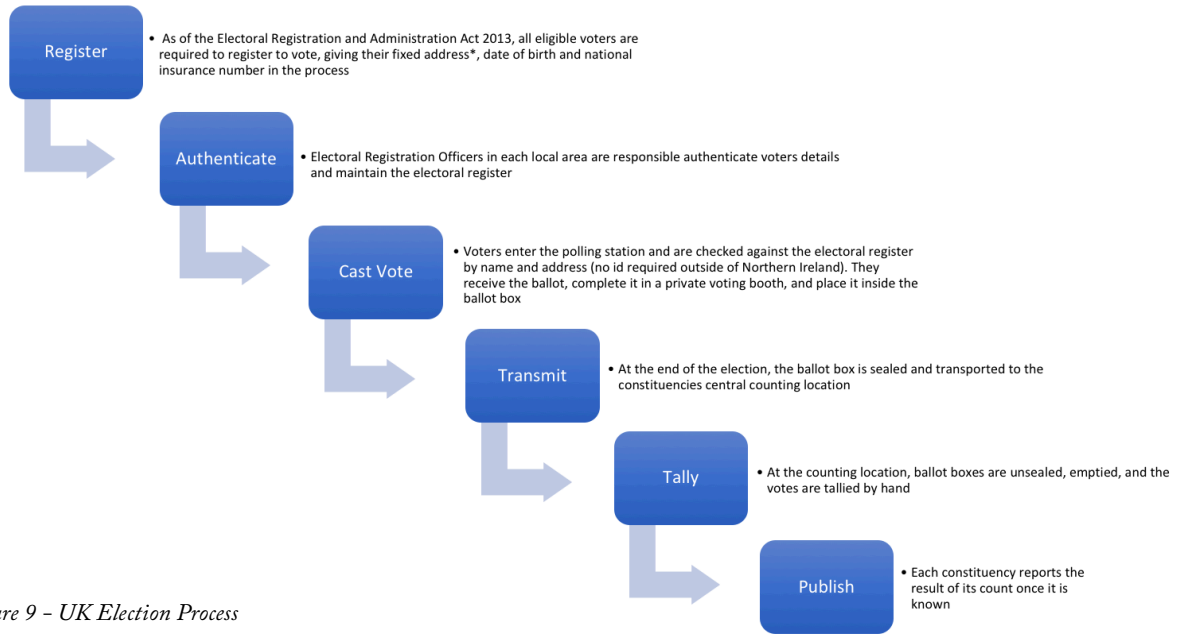


Figure 9 - UK Election Process

4.2 Votechain Election Process

There are several ways in which HLF could be implemented in the context of a UK Election discussed in Chapter 3.5 – Scalability. Here, we assume that each constituency is running as a channel on a single national-scale HLF Blockchain. Figure 10 illustrates the whole election process using our E-Voting system – which we call Votechain:

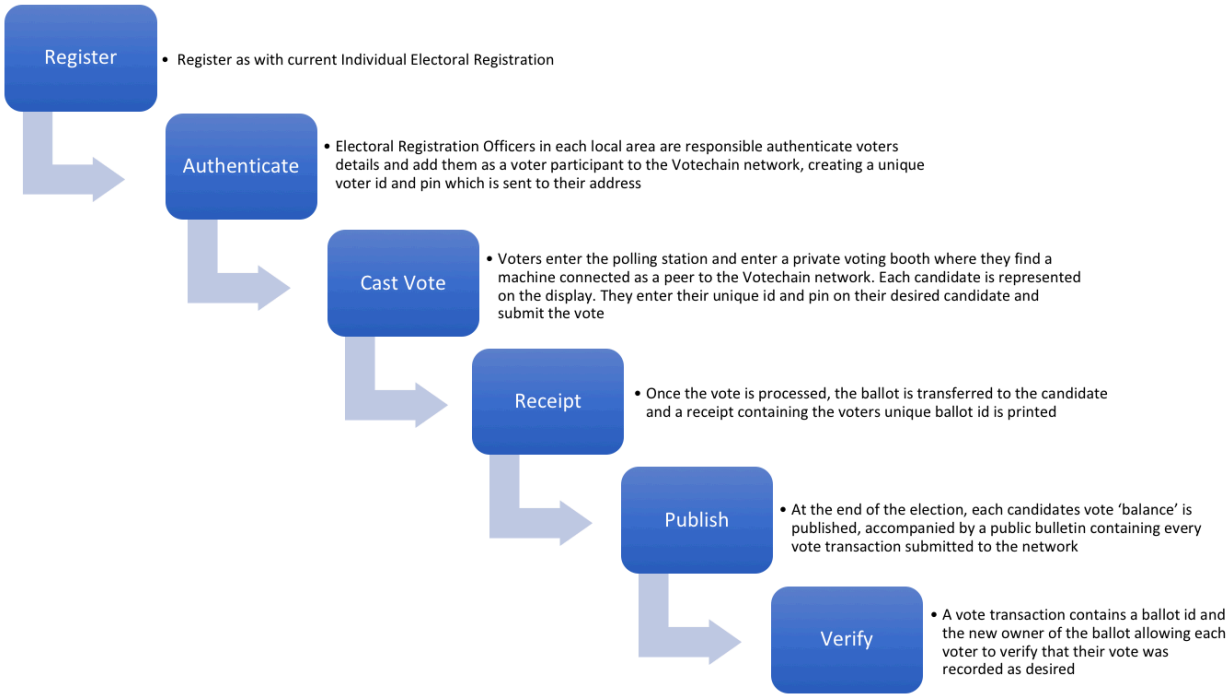


Figure 10 - Proposed Votechain Election Process

Each polling station contains several voting machines, each of which is set up with a Hyperledger Fabric environment capable of running the Votechain network. The IP addresses of each of the voting machines is added to Votechain's connection profile file. Connection Profiles contain the details (names, IP addresses, etc.) of all organisations, peers and orders. When the Business Network install command is run by the network administrator (the election authority in our case), the business network is distributed to each of the peers.

The Electoral Registration Officers (EROs) of each constituency create a new Candidate participant for each of the candidates running in their constituency.

Each voter registers to vote through the same channels they do currently. The EROs in each constituency authenticate voter details and create a new Voter participant for them. Votechain's smart contract ensures 1 new Ballot asset is created for each Voter participant. This generates a unique Voter ID and PIN, both of which are posted to the voters registered address.

On election day, voters enter the polling station where they find a Voting Machine (connected as an Endorsing peer to the Votechain network). Each Candidate in their constituency is represented on the display. The voter enters their Voter ID and PIN next to their candidate of choice and then submit their vote. If the transaction is accepted, ownership of the ballot is transferred to the Candidate, and a receipt is printed out for the Voter containing their Ballot ID.

Finally, we have a public bulletin board in the form of a publicly available web page. The web page contains a list of every Vote transaction on the Blockchain, with each transaction containing the Ballot ID and new owners ID, allowing the Voter to ensure that their Ballot was transferred to the desired candidate. Note that transactions **do not** contain any information on the original owner of the Ballot, thus ensuring voter anonymity. Figure 11 illustrates the whole Votechain Election process in greater detail.

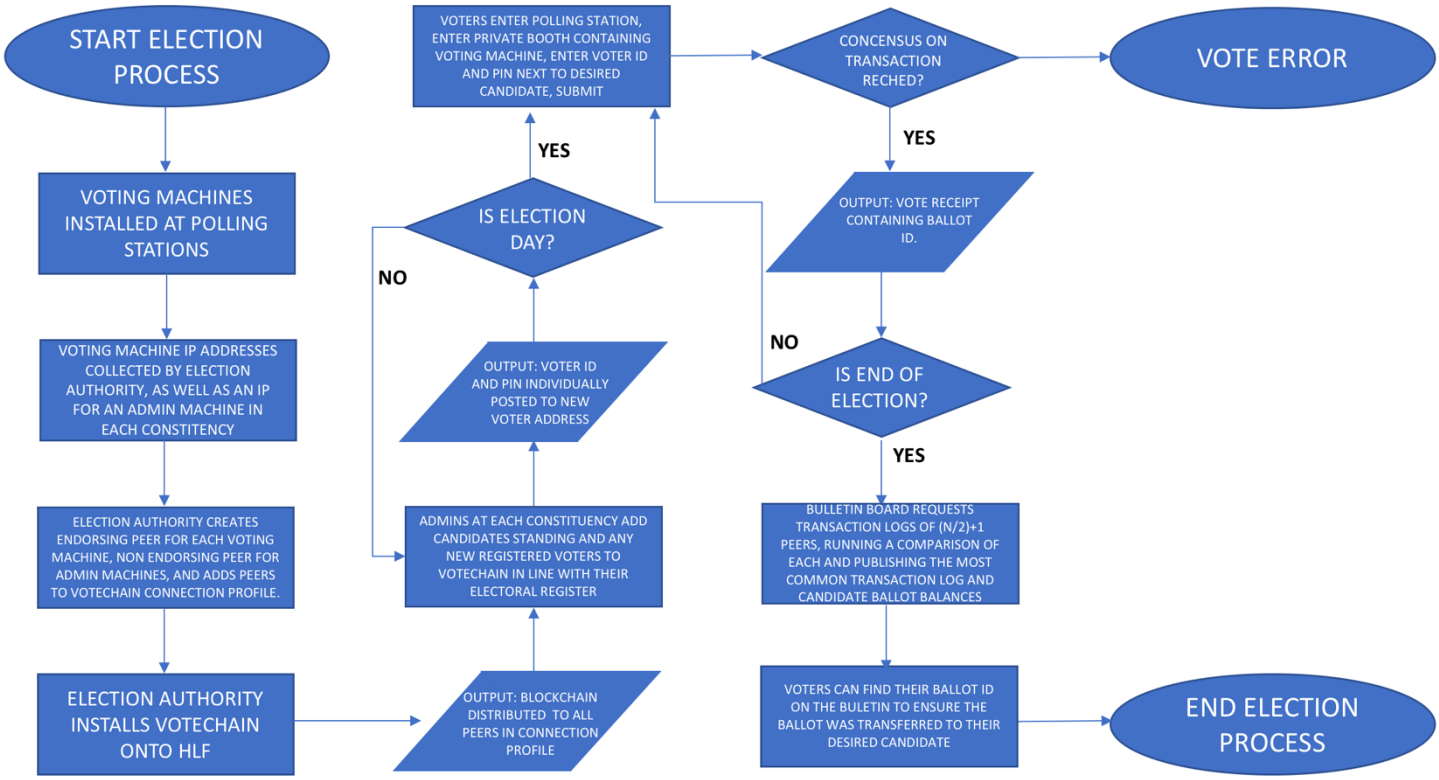


Figure 11 - Votechain Election Flowchart

The bulletin board must query the Blockchain to retrieve all Vote transactions. For votes to be end-to-end verifiable, we must ensure that the transactions displayed on the bulletin board will not be affected by a malicious peer returning falsified transactions. To achieve this, we request the transaction logs from (n/2)+1 peers, where n is the number of peers on the network. Each of the logs is compared. If there are no malicious peers, all of the logs will be identical as they should all share the same ledger. Assuming no clear abnormalities are flagged up here (i.e. many logs are different), the most common log is used for the bulletin board.

Chapter 5

In Practice: Votechain – A Proof-of-Concept

Up until this point in my investigation into Hyperledger Fabric as a platform for E-Voting, I have focused purely on HLF's functionality and suitability from only a theoretical perspective. The exploration up until this point in the preceding chapters have given me enough of an understanding of HLF to suggest it could offer improvement on previous Blockchain based E-Voting solutions, and so now I think it makes sense for me to explore HLF in more of a practical capacity.

This chapter details 'Votechain' – my small-scale proof-of-concept implementation of the E-Voting system proposed in Chapter 4, designed to be run on the latest version[7] 1.1 of the fabric: HLFv11 (Hyperledger Project, 2018).

5.1 Developing Apps for Hyperledger Fabric

As mentioned earlier in the dissertation, my first introduction to HLF came whilst on placement at IBM, where I started to read developer guides for what was at the time the v0.6 beta of the fabric. This was obviously fairly early in the development of HLF, so there really wasn't a whole lot of information available, but of what I could find most of the guides were for writing Chaincode apps in Go (Go, 2018) using the Go SDK for Hyperledger Fabric (Hyperledger Project, 2018). I experimented with that for a while, but with little documentation (or personal knowledge!) I didn't make much tangible progress.

I then read an article about their new tool, Hyperledger Composer (Hyperledger Project, 2018). With HLF being designed around using Blockchain for industry, Hyperledger Composer aimed at drastically simplifying and speeding up the development process of 'Business Networks' – essentially applications composed of Assets, Participants and Transactions (see Figure 11 below) – deployable onto an instance of the fabric.

It doesn't take long to get a good enough grasp on the structure of a Hyperledger Composer application and the syntax of the files to be able to start developing your own Business Network, and they even have the Composer Playground (Hyperledger Project, 2018) as an introduction to start exploring and developing your business network ideas.

With version 1.1 of HLF now released, Fabric Composer now seems to be the suggested tool for developers looking to start building applications for HLF, and so, especially given the time constraint of this project, I think it makes sense to focus on the use of Hyperledger Composer to build my E-Voting Proof-of-Concept.

[7] As of the start of my proof-of-concept implementation in March 2018

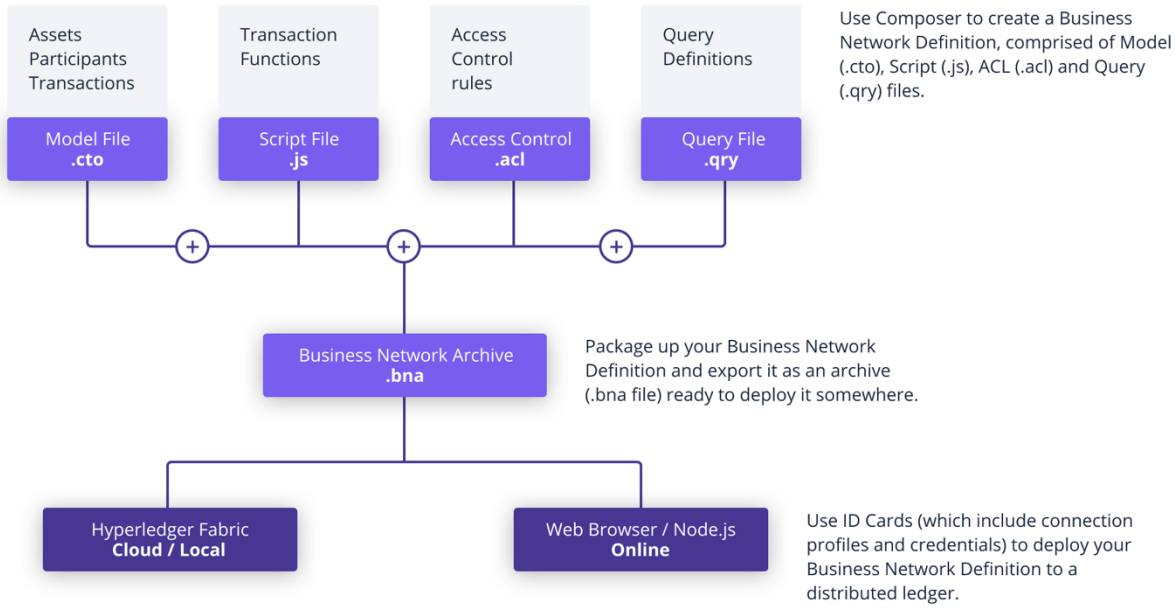


Figure 12 - Hyperledger Composer Application Structure (Hyperledger Project, 2018)

As a quick overview of my development environment, I am running Mac OS High Sierra with an instance of Hyperledger Fabric's HLFv11 deployed locally. For more information, see my guide to running Votechain locally (Appendix II) where I also cover the tools I used in Votechain's development.

5.2 Design

I am developing a proof-of-concept E-Voting system (referred to as 'Votechain' from here on) both to get hands on experience with developing apps for HLF, aiding in my investigation of the technologies suitability for an E-Voting system; and to accompany/aid in the understanding of my proposal for a secure E-Voting system built on HLF (as presented in Chapter 4).

What I aim to build is essentially a smaller-scale version, with all of the useful features of Blockchains and HLF, initially to be run on one local machine running an instance of HLF as a demonstration of HLF running an E-Voting system (essentially a centralised E-Voting system), before attempting to deploy Votechain onto a handful of peers do demonstrate Votechain as a more decentralised E-Voting system, working in the full desired context of a distributed ledger.

Following the structure of a Hyperledger Composer application (as above in Figure 11), the proposed structure of Votechain would be as follows:

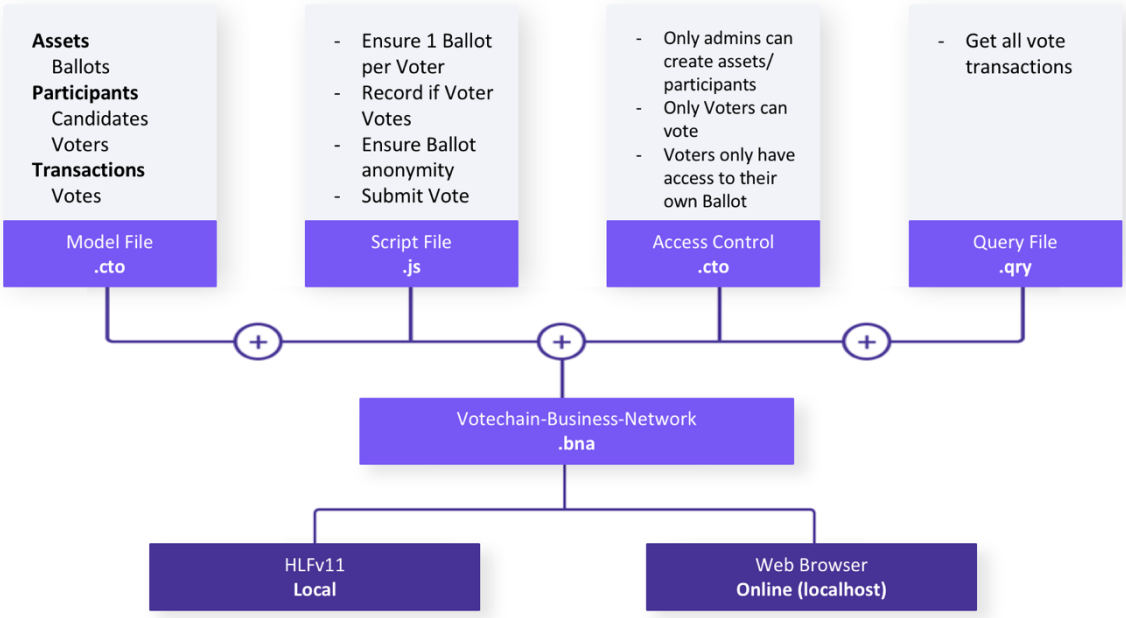


Figure 13 – Proposed Votechain Business Network Structure

The front end of Votechain is purely for the demonstration of the Business Network in its actual E-Voting context, for which I plan on using 3 dashboards:

1. **Admin Dashboard** – Set up and administration of the election (i.e. creating candidates/voters, launching the election, viewing results, providing a place to audit the network etc.)
2. **Voter Dashboard** – The main interface for machines in polling stations, through which voters cast their votes
    a. *Note that as the proof-of-concept is only deployed locally onto a 1 machine, 1 peer fabric. There is only one Endorsing Peer, and therefore no real consensus as each transaction is self-endorsed. In the full-scale system described in Chapter 4, each polling station voting machine would become an Endorsing Peer for the Fabric's consensus mechanism*
3. **Public Bulletin** – A publicly viewable page containing an un-manipulated direct response of all vote transactions on the Blockchain, allowing for voters to ensure their ballot was cast as expected, but still maintaining voter anonymity. This bulletin should **not** be affected by malicious peers

5.3 Implementation

Here I describe the implementation of my proof-of-concept E-Voting system – Votechain – capable of running on Hyperledger Fabric version 1.1 (HLFv11).

The Votechain system has 2 main components; the Votechain Business Network (the application which is running on the HLF Blockchain) and the Votechain dashboards (the front-end interfaces allowing users to interact with the Blockchain, communicating with the Business Network through a REST API.

5.3.1 Votechain Business Network

As discussed in Chapter 5.1 – Developing Apps for Hyperledger Fabric, a Hyperledger Composer Business Network is composed of

- A file modelling the network in terms of its assets, participants and transactions
- A file containing the transaction logic for the network
- A file defining the permissions of participants in terms of their access to assets and transactions
- An optional file containing personalised queries to retrieve data from the network

Hyperledger Composer is developed to be able to quickly define global-scale supply chain business networks, containing complex relationships between assets, participants and transactions. In comparison, defining Votechain as a Business Network is quite simple and intuitive. As a brief outline, using the idea of a UK national election:

- We have only 1 very simple type of asset – a Ballot
- We have only 2 types of participant – Candidates and Voters
- We have only 1 transaction – a Vote

Model[8]

Model files are written in Hyperledger Composer Modelling Language[9] (CTO).

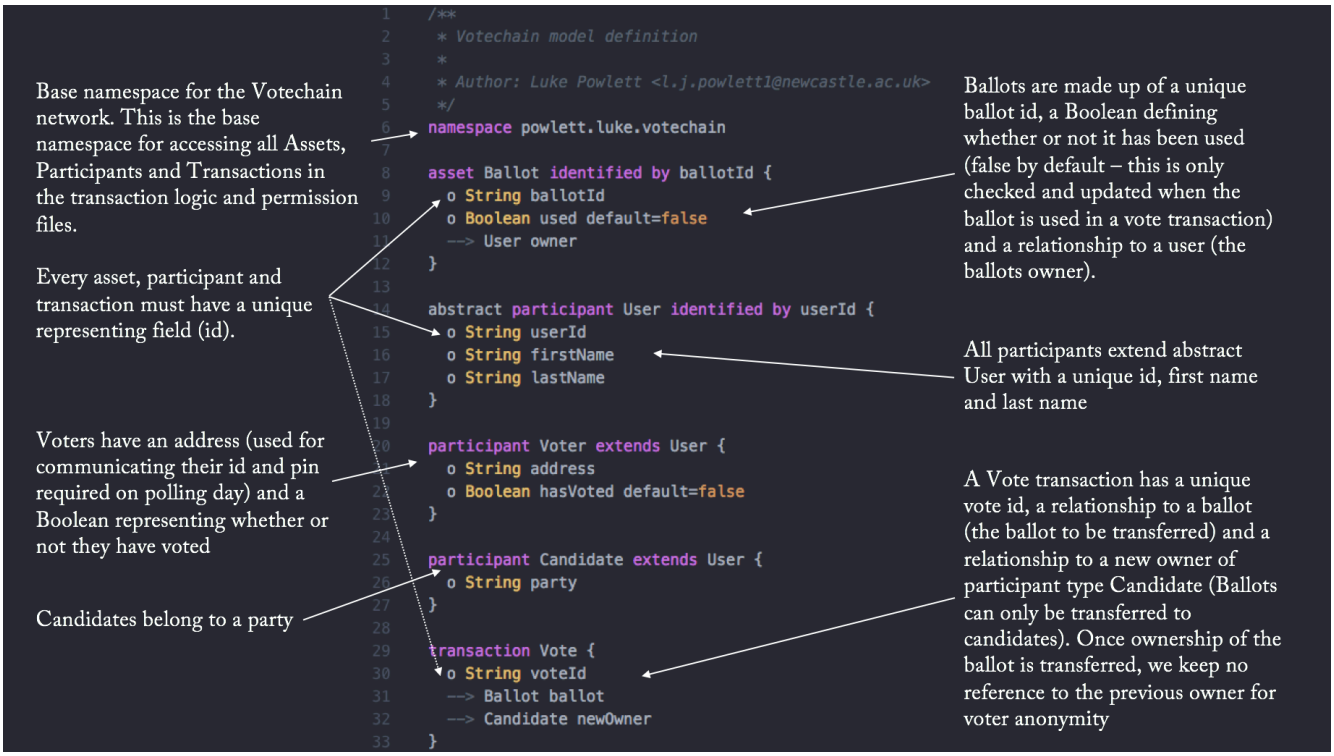


Figure 14 - Votechain model file

[8] File Path: votechain-network/models/powlett.luke.votechain.cto
[9] CTO Reference - https://hyperledger.github.io/composer/latest/reference/cto_language.html

Transaction Logic[10]

Transaction files are written in JavaScript following Hyperledger Composer's Transaction Processor Functions[11].

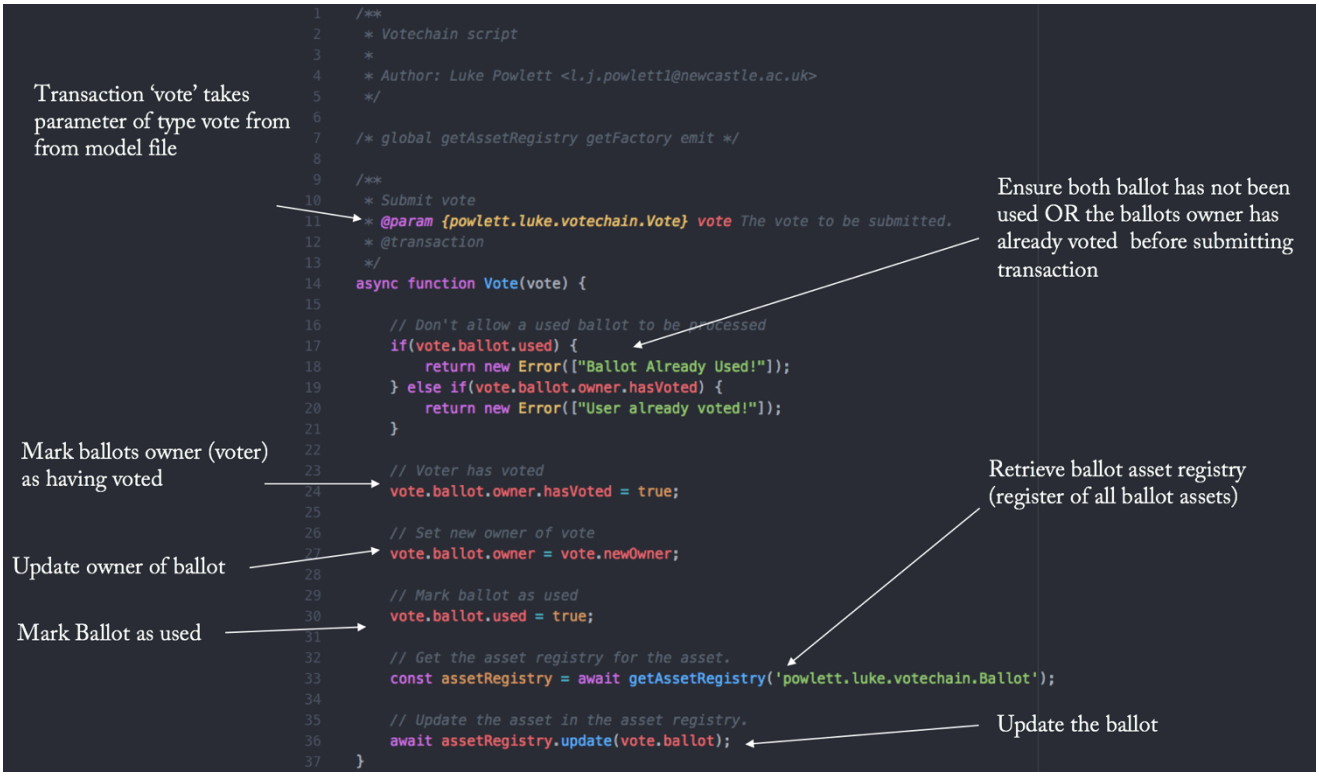


Figure 15 - Vote Transaction Processor Function

The transaction processor functions of a business network provide the core logic which all transactions must follow. Combined with the model and permission files, building the Business Network converts this into Chaincode – HLF's equivalent of a Smart Contract.

[10] File Path: votechain-network/lib/script.js
[11] TP Function Reference - https://hyperledger.github.io/composer/latest/reference/js_scripts.html

Permissions[12]

Permission files are written in Hyperledger Composer Access Control Language[13] (ACL).

Figure 16 - ACL Permission File

Creating a Business Network Archive

Once the Business Network has been defined, the next stage is for it to be packaged into a deployable Business Network Archive (.bna) file. Using the Composer command line tools (Hyperledger Project, 2018), running the '`composer archive create`' command will package the Business Network in line with the directories package.json file (Appendix III), in our case creating the file 'votechain-network@0.0.2.bna', in the projects root directory.

[12] File Path: votechain-network/permissions.acl
[13] ACL Reference - https://hyperledger.github.io/composer/latest/reference/acl_language

Installing Votechain

Now that the business network has been packaged, it can be deployed onto an instance of HLF. Any modifications to the fabric require a 'PeerAdmin' identity card[14]. The business network can be installed by running the composer network install command from the business network's root directory, passing the PeerAdmin card and the .bna file.

A connection.json file in the same root directory can be used to specify any non-default configuration settings for your specific fabric network setup, including the locations of all peers on the fabric on which the business network should be deployed. As the proof-of-concept is only running the network with one peer on one machine, this is not required. Chapter 7: Future Direction discusses my challenges around setting up a multi-peer HLF network.

Once the business network has been installed onto the fabric, the composer network start command can be issued to start the Votechain network.

Generating REST API

For interfacing between the web front end, business network and Blockchain, we create a bespoke REST API based on our business network, configured by issuing the composer-rest-server command. For the proof-of-concept, we are hosting the API locally. Where we have multiple users, the API can be run in multiple-user mode, with access to the being authenticated using OAUTH2.0[15].

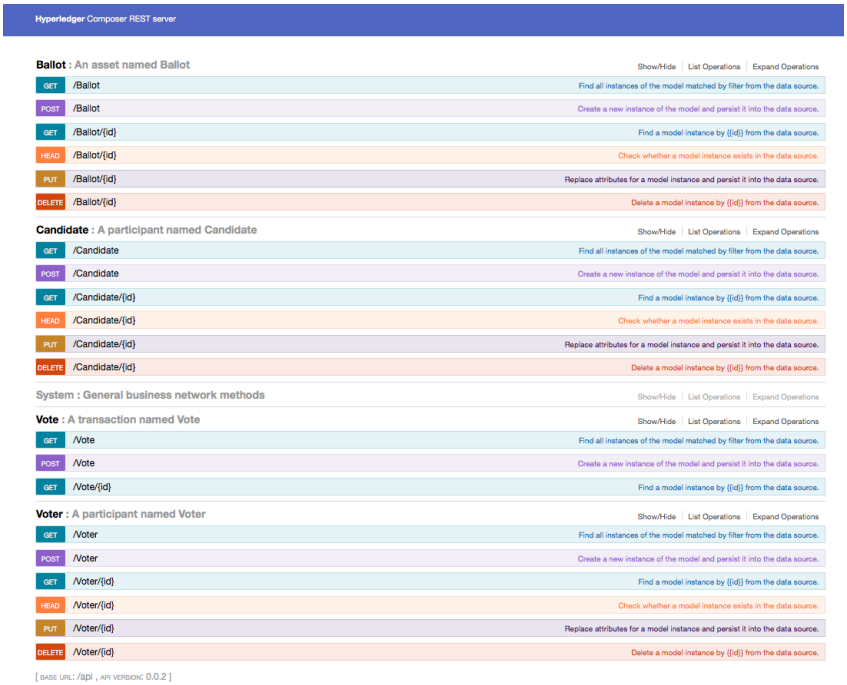


Figure 17 - Composer REST API for Votechain

[14] See https://hyperledger.github.io/composer/latest/reference/composer.card.create.html
[15] See https://hyperledger.github.io/composer/latest/tutorials/google_oauth2_rest

5.3.2 Dashboards

Now that we have Votechain running as a business network on an instance on the fabric, we need a way to interact with the Blockchain. As discussed in the design section of our system, there are 3 core interfaces required for our election environment; an admin dashboard for the administration of the election, a voter dashboard through which votes are cast, and a public bulletin board providing a list of every vote transaction, providing voters with a way of verifying their vote was cast as expected. All of the interfaces are web pages hosted locally on the machine.

Below I present a breakdown of each interface with the context of an election containing 3 Candidates and 3 Voters. The interfaces communicate with the business network through HTTP requests to the deployed REST API as discussed in section 5.3.1.

Admin Dashboard

The Admin dashboard has 2 views: one for the administration of Candidate participants on the network and one for the administration of Voter participants on the network. Figure 17 (below) presents the Candidate View of the Admin dashboard. The Admin dashboard would be run by an Admin peer on the fabric network.

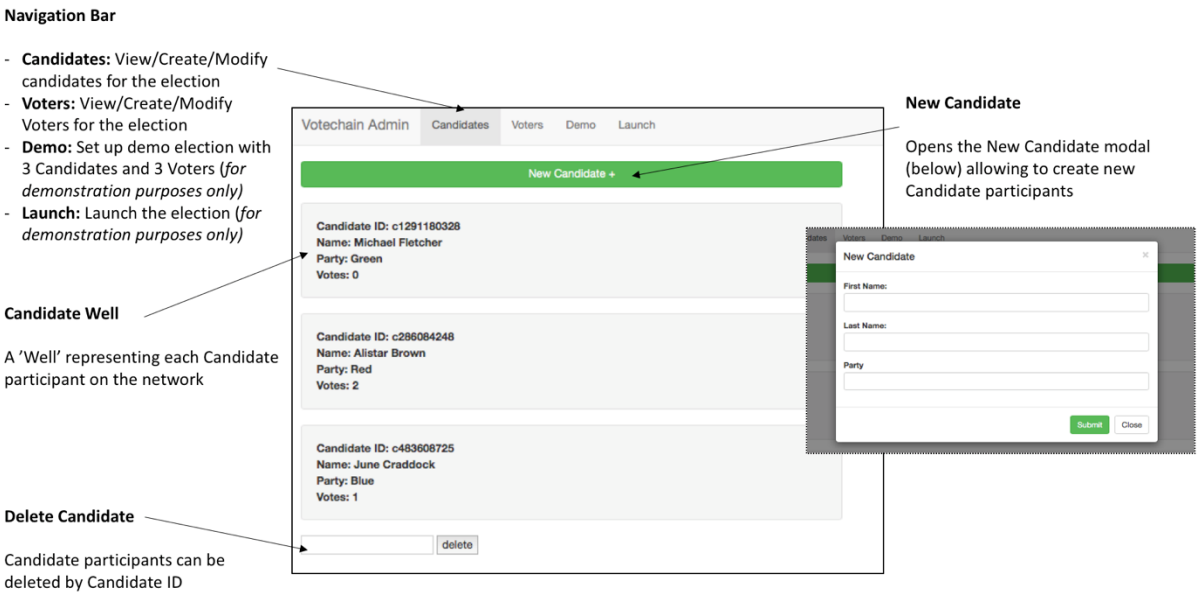


Figure 18 - Admin Dashboard (Candidate View)

Figure 18 (below) presents the Voter view of the Admin dashboard. On creation of a new Voter participant, 1 Ballot asset is also created and assigned to the Voter by Voter ID.

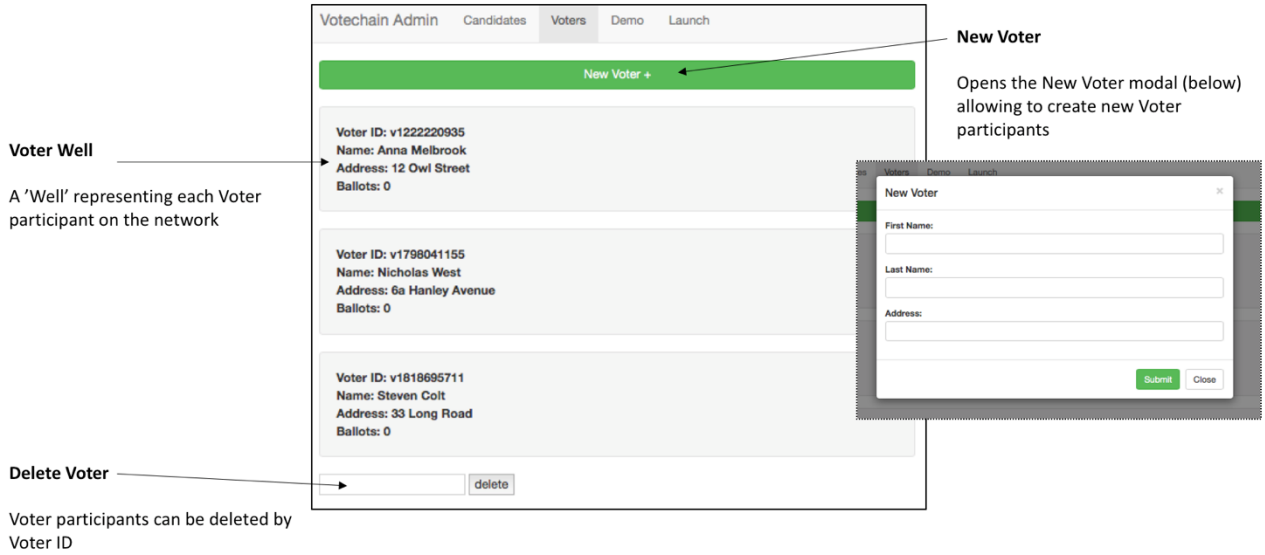


Figure 19 - Admin Dashboard (Voter View)

Vote Machine Dashboard

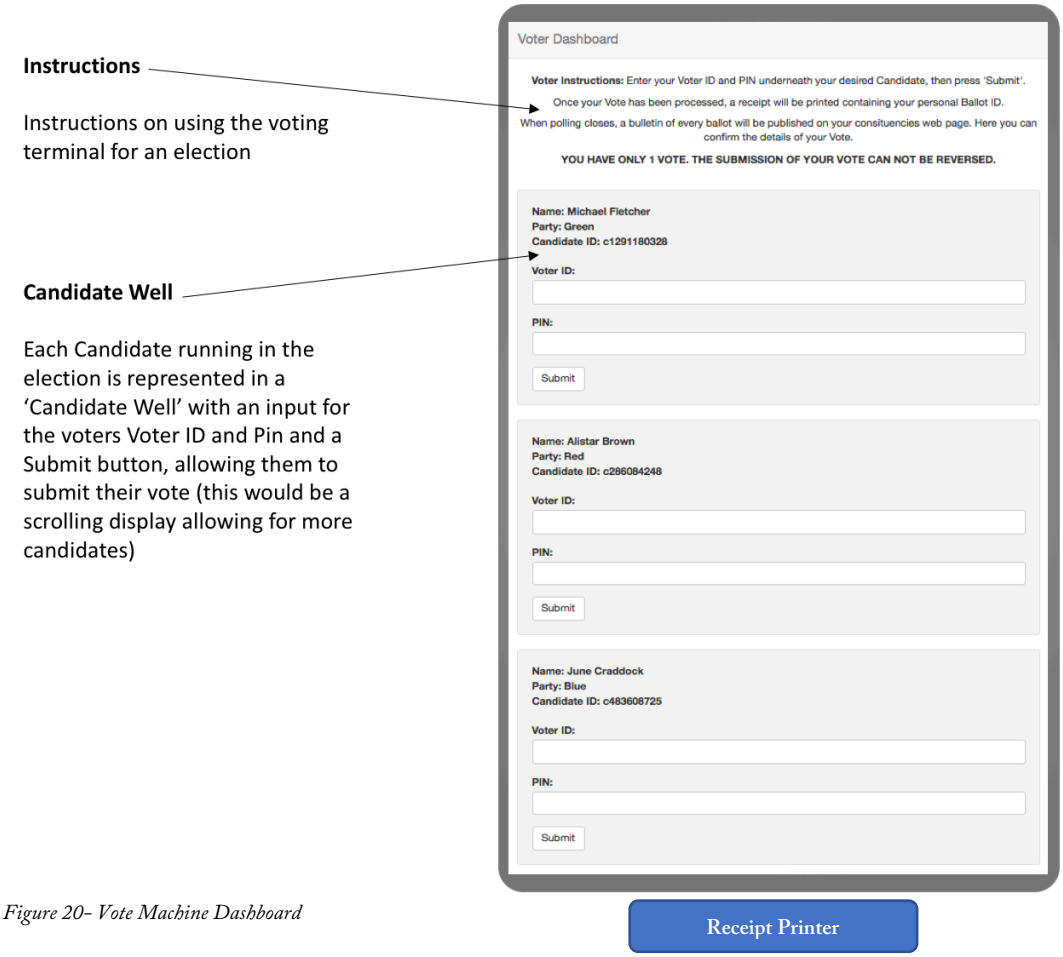


Figure 20- Vote Machine Dashboard

Figure 19 (above) presents the Vote Machine dashboard, through which Voters would be able to cast their votes on Polling Day. By Polling Day, each registered Voter would have received their unique Voter ID and PIN. Using these credentials, they are able to submit their Ballot to their desired candidate. On submission of the vote, the Ballot ID is printed out as a receipt to the Voter.

Public Bulletin

At the end of polling and once all vote transactions have been recorded on the fabric, a transaction log of all Vote transactions on the fabric is published online. Figure 20 (below) presents the Public Bulletin.

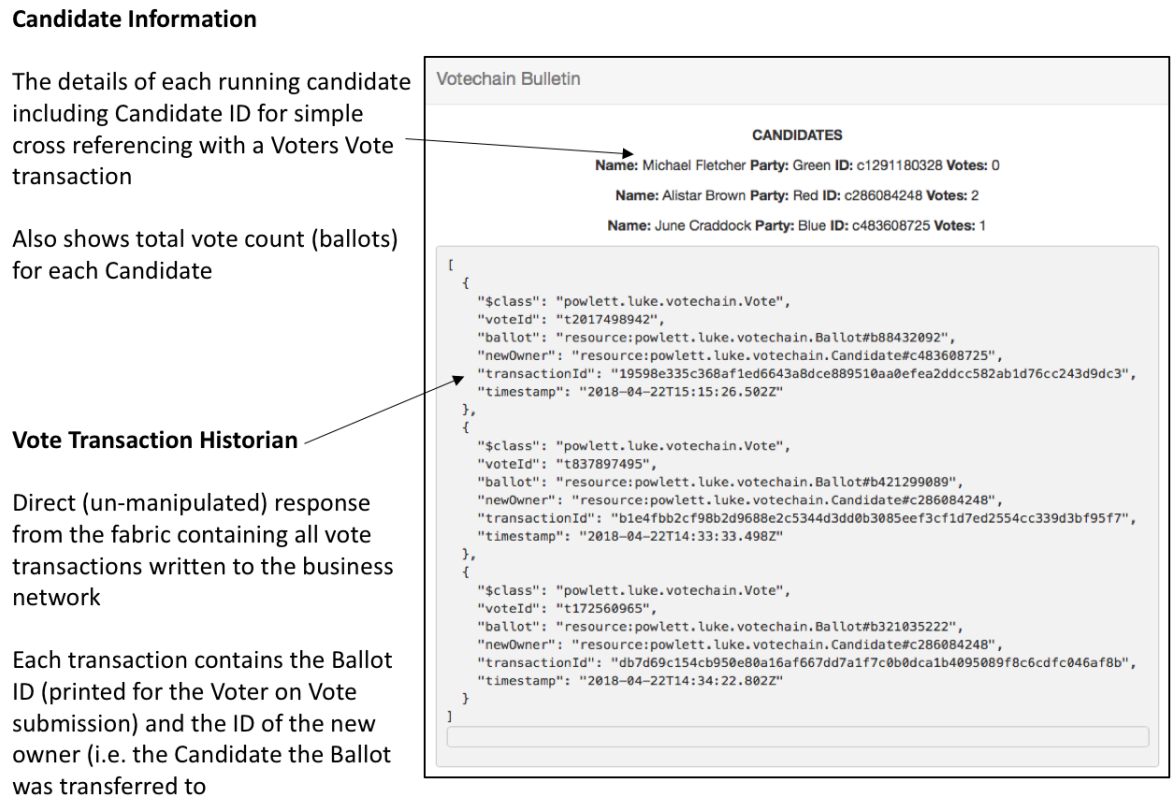


Figure 21 - Public Bulletin

For the Vote transactions to be End-to-End verifiable, we must endeavour to ensure that a maliciously acting peer on the network is not able to alter the bulletin board. To achieve this, we request the Vote transaction log of all peers on the network and calculate a Hash of the JSON response. Because the transaction log response is ordered by Transaction ID, and we know all Peers operating correctly on the network should have an identical transaction log (from the shared ledger), rather than check through and compare all transactions in all responses, we can quickly calculate the hash of each response and compare the results, publishing the most common transaction log (assuming less than (n/2) responses, where n is the number of peers, give different hash results) and report the peers with different transaction logs for further investigation. This means that, as long as at least half of the peers on the network are acting correctly (i.e. not compromised by a malicious attacker), the transaction log on the Public Bulletin will be the correct transaction log. Figure 21 (below) shows the transaction log responses and the calculated hash of each. The first 2 responses are from correctly acting peers, showing identical transaction logs and thus identical hash results. The 3rd response has a modified log, which results in a different hash.

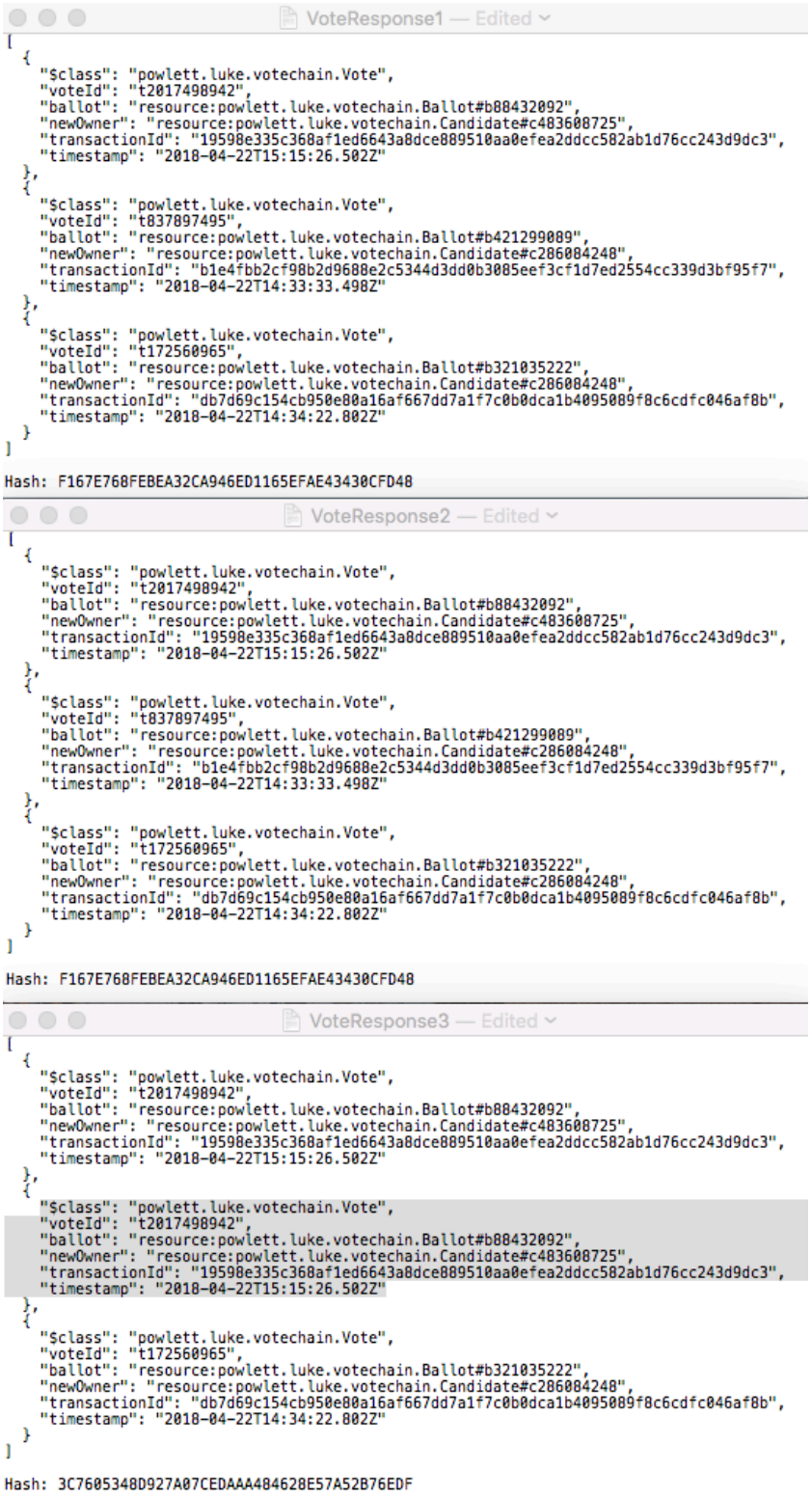


Figure 22 - Transaction logs of 3 peers, where response 3 is a malicious peer

5.4 Reflection

This chapter concludes my exploration of the practical implementation of a Proof-of-Concept E-Voting on Hyperledger Fabric. I have successfully implemented a prototype of the system proposed in Chapter 4, capable of running one peer locally on one machine. All of the projects code, as well as instructions for running Votechain locally on Ubuntu or Mac OS can be found on the projects GitHub[16].

I did face several challenges when trying to extend the Fabric Network to more than one peer in an attempt to fully demonstrate the E-Voting system running with in its true distributed ledger environment. These challenges and HLF's suitability as a platform for a secure E-Voting system are discussed and evaluated in Part 3 – Evaluation.

Objective Reflection

II. Develop a proof-of-concept E-Voting system on HLF to explore the practical implementation requirements of the suggested system.

[16] https://github.com/LukePowlett/Votechain

PART 3 – EVALUATION

Chapter 6

Hyperledger Fabric as a platform for Blockchain based E-Voting

This project has been an investigation into Hyperledger Fabric as a platform for a secure Blockchain based E-Voting system. I've explored the topics of Voting and E-Voting, aiding in the design of my proposed system. I've researched and conducted a feature analysis of Hyperledger Fabric to provide a grounded reasoning for HLF's basic suitability for an E-Voting system, again aiding in the design of my proposed system. Finally, I've utilised Hyperledger Composer to implement a Proof-of-Concept prototype of my proposed E-Voting system, Votechain, capable of running on an instance of Hyperledger Fabric version 1.1.

In this Chapter, I will first evaluate Votechain against the generally assumed security requirements of an E-Voting system as discussed in Chapter 2, before a review and evaluation of my findings on Hyperledger Fabric in terms of the project objectives presented in the first chapter.

6.1 Hyperledger Fabric E-Voting Security Evaluation

In section 2.1, I presented Hao *et al's* generally assumed security requirements for an E-Voting system which I aimed to assume in my own project. Here, I review and evaluate Votechain's implementation against each of the 6 requirements. To recap:

> *(1) User enrolment: Only eligible users can be enrolled in the voter registration.*
> *(2) User authentication: Only authenticated voters are allowed to vote during the election.*
> *(3) One-man-one-vote: Each authenticated voter is allowed to vote just once.*
> *(4) Voting privacy: Voting happens in a private space that no one else can observe.*
> *(5) Anonymity: The voting machine that is used does not know the real identity of the voter.*
> *(6) Public Bulletin Board: There is a publicly readable, append-only bulletin board (e.g. a mirrored public website) on which the legitimate voting system can publish audit data for verification (the authenticity of data can be ensured by the use of digital signatures).*

(Hao, et al.)

User Enrolment

Only eligible users can be enrolled in the voter registration

HLF is a permissioned Blockchain. Where many other popular Blockchains are built on the idea of full decentralisation and anonymity, HLF provides what I would described as a quasi-decentralised private ledger, where the ledger is shared only among permissioned peers in accordance with a central authority. Further, Hyperledger Composer allows for a permission file to be used in the building of the business network (essentially the application running on the HLF) specifying which levels of user can perform actions in relation to both the fabric and the business network. In an E-Voting context, this means that the election authority have a permission level allowing them to perform modifications to the Blockchain (i.e. installing the business network, creating new voter participants, etc). Assuming the case of a UK National Election, the election authority has the electoral register with the required details of every eligible user, or voter, to be added as a voter participant to our business network.

User Authentication

Only authenticated voters are allowed to vote during the election

With HLF business networks focusing on networks of assets, participants and transactions, we specify that each Voter participant is represented by a unique Voter ID and PIN (which they will possess prior to the election). The combination of these credentials allows a Voter to start a Vote transaction.

One Man, One Vote

Each authenticated voter is allowed to vote just once

HLF's version of a smart-contract, Chaincode, implements the 'rules' of the Blockchain network. In our business networks Vote transaction (which is converted to Chaincode when deployed to the fabric), we specify rules that ensure each Voter participant is only able to submit one Vote transaction, namely that each Voter participant is issued only one Ballot asset, of which ownership is transferred to the desired candidate on execution of the Vote transaction.

Voter Privacy

Voting happens in a private space that no one else can observe

This is of course fully dependent of the set-up of the election, but the fabric network can be deployed dedicated voting machines inside a private booth inside a polling station.

Anonymity

The voting machine that is used does not know the real identity of the voter

To authenticate the Voter, the voting machine does need to check the Voter ID and PIN provided, with the Voter ID being linked to the Voter participant and thus the Voter's personal details. However, the Vote transaction removes the Voter's ID from the cast Ballot asset, replacing it with that of the voted for Candidate.

Public Bulletin Board

There is a publicly readable, append-only bulletin board (e.g. a mirrored public website) on which the legitimate voting system can publish audit data for verification (the authenticity of data can be ensured by the use of digital signatures)

To allow Voter's to check their Votes have been cast as desired, we provide a web based Public Bulletin Board containing a direct, un-manipulated transaction log of every Vote transaction on the Blockchain. To ensure that votes are End-to-End verifiable, we must ensure that this public bulletin will not be affected by a malicious node reporting falsified transactions that aren't on the Blockchain. We achieve this by requesting and computing the hash of the Vote transaction log of every peer on the Blockchain, then selecting the most common. The idea of this is that, assuming peers are acting correctly, every transaction log should be identical, and so their hashes should be identical. We can easily compare the hashes of the transaction logs to identify any outliers.

This ensures that the public bulletin board will be accurate as long as less than (n/2) peers are maliciously falsifying transaction logs (where n is the number of endorsing peers on the Blockchain).

6.2 Evaluation Against Project Objectives

In the first Chapter of this dissertation, I outlined my 3 core objectives of the investigation. To recap:

I. Perform a critical feature analysis of HLF to ensure it is a feasible platform on which to develop an electronic voting system, in comparison with other popular existing Blockchains
II. Develop a proof-of-concept E-Voting system running on HLF to explore the practical implementation of the suggested system
III. Review of feature analysis findings and proof-of-concept implementation to give a final critical overview of HLF's suitability

Here, I review and evaluate the findings of Objective's I and II, providing a final critical discussion on HLF's suitability as a platform for a secure Blockchain based E-Voting system, in line with Objective III.

Feature Analysis

Perform a critical feature analysis of HLF to ensure it is a feasible platform on which to develop an electronic voting system, in comparison with existing Blockchain and popular non-Blockchain E-Voting systems

Elections aren't permission-less. We must register as voters to be eligible to vote and we cast our votes in registered polling stations. For this reason, in terms of an E-Voting scenario, a permissioned Blockchain like HLF seems to be a natural choice over the more common permission-less Blockchain's. This does come with the loss of full decentralisation as we still require an Election Authority to oversee the issuing of permissions to each peer, but one of the core features of the Blockchain is the verifiability of transactions, so by publicly publishing a verifiable bulletin board containing each transaction we can create a verifiable, and much more transparent, election process – certainly more so than we see in a UK Election today.

One of the key concerns raised in McCorry *et al's* E-Voting implementation (McCorry, Toreini, & Mehrnezhad, 2016) on the Ethereum Blockchain was the computation cost incurred by the Proof-of-Work consensus mechanism (paid for using the notion of 'gas' on Ethereum) which posed significant costs when scaling up to a national election. Being a permissioned Blockchain, HLF's use of Proof-of-Stake consensus mechanisms removes the high computation costs on the basis the each of the peers on the network are 'trusted' peers. Further, the consensus mechanism is fully pluggable which means that it could be tailored to the requirements of the individual implementation to maximise efficiency/security.

The biggest concern I found with HLF in terms of its suitability for an E-Voting system is the complete lack of official published information on the performance of HLF in terms of scalability. Unofficial sources (discussed in Chapter 3.5) indicated that increasing the size of the network resulted in a significant performance bottleneck. Further investigation would be required here to fully determine HLF's suitability for a national scale election, but current indications certainly suggest that at this time it should at least be fully suitable for smaller elections where the number of endorsing peers remains in the single digits. There is a silver lining with HLF in terms of its scalability, though. HLF supports multiple channels running on one Blockchain, where each cannel has access only to its own ledger and thus each peer in the channel only endorses the transactions of other peers in the

channel. Theoretically, this means that once we the know scale limits of our system, we can segregate the Blockchain into multiple channels to maintain performance, even at a much larger scale.

Proof-of-Concept

Develop a proof-of-concept E-Voting system on HLF to explore the practical implementation of the suggested system

In the development of my proof-of-concept E-Voting system, I found Hyperledger Composer's focus on a Business Network made up of Assets, Participants and Transactions to be particularly apt for the projects E-Voting context. While an Election wouldn't necessarily fit to be defined as a 'Business Network', an Election fits perfectly into these 3 categories, being comprised of a Ballot (our Asset) being traded in the form of a Vote (our Transaction) between a Voter and a Candidate (our Participants). Further, HLF's support for REST API's makes simple work of integrating your applications with the Blockchain.

The biggest challenge I faced in the practical implementation of the proof-of-concept Votechain system wasn't the development of the Votechain 'Business Network' – or the actual application running on the fabric - but the setting up of a fabric network with multiple peers. There is a lot of official documentation available on building applications and writing Chaincode for HLF, but I found the documentation around building the actual fabric network on which the application could be deployed to be much less comprehensive; and with the first official release of HLF, Version 1.0, being released just last summer, it's not too much of a surprise to find areas where the provided documentation wasn't sufficient for a relative new-comer to HLF development. The main aim of this project, though, was to investigate HLF's suitability for an E-Voting application – my proposal and what I have developed is an end-to-end verifiable E-Voting Business Network, using the Hyperledger Project's own Hyperledger Composer tools, which is deployable onto any running fabric network, and which I demonstrate on one peer running an instance of the latest version of the fabric. Whilst ideally I would have been able to demonstrate my system running in its intended distributed environment with multiple peers, I believe the fact that this goes beyond the projects core investigative focus, and so I discuss the extension of my proof-of-concept onto a multiple peer fabric network in the next chapter looking at the possible future direction of this project.

Objective Reflection

III. Review of feature analysis findings and proof-of-concept implementation to give a critical view of HLF's suitability

Chapter 7

Future Direction

Now that I have investigated the implementation of an E-Voting application capable of running on a HLF Blockchain, demonstrated on one single peer running an instance of the fabric, the logical next phase of the investigation would be to move on to building an extended fabric network with multiple peers to demonstrate and test the E-Voting system working in its intended distributed Blockchain structure, with the ledger shared between multiple peers.

Having reached the end of my Proof-of-Concept implementation (the Votechain application to be run on a fabric network), I began exploring extending my 1 peer fabric network to multiple peers across multiple devices. However, I found the current level of documentation available on building a fabric network across multiple hosts to be not nearly sufficient for me to be able to extend the fabric network, especially given the constraints of this undergraduate level project. From my own exploration and experimentation from unofficial sources, the main barrier to extending the fabric network is HLF's current Docker based architecture. HLF relies on a Docker based architecture where all of the components of the fabric network run in separate containers, with each container communicating over a local network which each container attaches itself to. This means that when we try to extend the fabric network across multiple hosts, the containers aren't able to communicate with one another. There are suggestions that this can be achieved by setting up a multi-node Docker Swarm, but this is not part of the official Hyperledger Fabric documentation. The official documentation[17] only focuses on building and deploying a multi-peer HLF network on one single machine for demonstration purposes.

The main intention of this project is to explore the suitability of HLF as a platform for an E-Voting application – I have explored HLF and implemented an E-Voting application capable of running on a fabric network. Given this and the comments above, the actual building of a multi-peer, multi-host fabric network is outside the main scope of my project, and so I propose this as the next logical future direction leading on from this project.

This would then open the door to conducting tests to further explore HLF's performance, specific to our E-Voting context, when operating with an increased number of peers. As discussed in Chapter 6, Hyperledger Project or IBM themselves don't yet seem to have released an official figures regarding the performance of the fabric with regards to scale. By extending the fabric network, we could begin investigating the optimal performance set up for the E-Voting HLF network, specifically in relation to segregating the Blockchain into multiple channels (as discussed in Chapter 6).

Assuming that further investigation maintains HLF's suitability as a platform for secure Blockchain E-Voting, HLF's pluggable consensus mechanism would also the create a further direction of exploration, evaluating the available PoS consensus mechanisms and/or proposing a new mechanism tailored to our E-Voting use case with the desired balance between performance and security.

[17] See: http://hyperledger-fabric.readthedocs.io/en/release-1.1/build_network.html

Closing Words

Since my first real introduction to Blockchain and Hyperledger Fabric last year on placement at IBM, I've been thoroughly interested in the technology, and have really enjoyed and taken a lot from deep exploration into Blockchain that the project has afforded me.

On the whole, I am very happy with the outcome of my project. As Hyperledger Composer Business Networks support multi-peer fabric networks, there is no reason to suggest that Votechain would have a problem running on at least a basic small multi-peer fabric network. Whilst demonstrating my Votechain system working across multiple peers on multiple machines would have been the perfect end to the project, I think the fact that I can still demonstrate a verifiable E-Voting system running on a one peer fabric instance is a very positive outcome, and one that has been able to satisfy all 3 of my projects objectives and the overall project aim.

[PAGE INTENTIONALLY LEFT BLANK]

Appendix

Appendix I: Cost Analysis of UK Elections

Year	Type	Total Cost (±£1m)
2017	'Snap' Election	£140,000,000
2016	EU Referendum	£142,000,000
2015	General Election	£123,000,000
2010	General Election	£113,000,000

Taking 2017's Electorate of 46,425,486 and the average cost of the last 4 national elections/referendums (above) of £129,500,000, we get an average cost per voter of ~£2.79 (Mason, 2017) (UK Parliment, 2017).

Appendix II: Votechain Local Installation Instructions

Note that the projects source code can be found at https://www.github.com/LukePowlett/Votechain as well as being submitted as supplementary material alongside this dissertation.

(Adapted from official Hyperledger Composer tutorials at https://hyperledger.github.io/composer/latest/tutorials/tutorials.html)

Installing Pre Requisites

- Min OS: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (version 9 not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x

Ubuntu

```
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
chmod u+x prereqs-ubuntu.sh
```

This script briefly uses sudo in its execution so you will be prompted for your password
`./prereqs-ubuntu.sh`

Mac OS

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

When you hit enter you should see the pop-up below, prompting you to install git. Press the Get Xcode button to install the full Apple Xcode IDE, including a C++ compiler, used to install native Node.js modules.

Create your bash profile
```
touch .bash_profile
```

Run original command
```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

Close and re-open terminal

Install node
```
nvm install --lts
nvm use --lts
node --version
```

Install docker following instructions at:
https://docs.docker.com/docker-for-mac/install/

Installing Hyperledger Fabric

CLI Tools
```
npm install -g composer-cli
npm install -g composer-rest-server
```

In [dir] of choice (e.g. ~/HLF)
```
mkdir [dir]/fabric-tools && cd [dir]/fabric-tools

curl -O https://raw.githubusercontent.com/hyperledger/composer-
tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz
tar -xvf fabric-dev-servers.tar.gz

cd [dir]/fabric-tools
./downloadFabric.sh
```

Running Votechain

In [dir] (same [dir] fabric tools installed)

```
git clone https://github.com/LukePowlett/Votechain.git
cd votechain-network
sh ./firstVotechainStart.sh
```

Accessing dashboards

Set up local web host root dir (e.g using XAMPP/MAMP htdocs root):
[dir]/votechain-network/votechain-web

Go to http://localhost for links to 3 dashboards

Appendix III: Votechain package.json file

```
{
  "name": "votechain-network",
  "version": "0.0.2",
  "description": "Proof-of-Concept E-Voting System",
  "scripts": {
    "prepublish": "mkdirp ./dist && composer archive create --sourceType dir --
sourceName . -a ./dist/votechain-network.bna",
    "pretest": "npm run lint",
    "lint": "eslint .",
    "test": "nyc mocha -t 0 test/*.js && cucumber-js"
  },
  "author": "Luke Powlett",
  "email": "l.j.powlett1@newcastle.ac.uk",
  "license": "Apache-2.0",
  "devDependencies": {
    "composer-admin": "^0.19.0",
    "composer-cli": "^0.19.0",
    "composer-client": "^0.19.0",
    "composer-common": "^0.19.0",
    "composer-connector-embedded": "^0.19.0",
    "composer-cucumber-steps": "^0.19.0",
    "chai": "latest",
    "chai-as-promised": "latest",
    "cucumber": "^2.2.0",
    "eslint": "latest",
    "nyc": "latest",
    "mkdirp": "latest",
    "mocha": "latest"
  }
}
```

Figures

Bibliography

Bazhanov, B. (2002). *Memoirs of the Former Secretary of Stalin.* Moscow: III Tysiacheletie.

Bettencourt, J. (2016, 12 08). *Can Blockchain Technology Secure Digital Voting Systems? Kaspersky Lab Challenged 19 Universities with Protecting e-Voting from Cyberattacks in Competition; Awards Three Top Finalists*. Retrieved 02 08, 2018, from Business Wire: https://www.businesswire.com/news/home/20161208005130/en/Blockchain-Technology-Secure-Digital-Voting-Systems-Kaspersky

Cachin, C. (2016). *Architecture of the Hyperledger Blockchain Fabric.* IBM Research. Zurich: IBM.

Ethereum. (2017). *Ethereum - Blockchain App Platform*. Retrieved 11 22, 2017, from https://ethereum.org

Ethereum. (n.d.). *Develop*. Retrieved 02 08, 2018, from Solidity: http://solidity.readthedocs.io/en/develop/

Go. (2018, 03 26). *GoLang Home*. Retrieved 04 2018, from Go: https://golang.org

Hao, F., Kreeger, M. N., Randell, B., Clarke, D., Shahandashti, S. F., & Lee, P. H.-J. (2017). *Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting.* JETS.

Hao, F., Ryan, P., & Zielinski, P. (2010). *Anonymous Voting by 2-Round Public Discussion.* IET Information Security.

Hastings, N., Peralta, R., Popoveniuc, S., & Regenscheid, A. (2011). *Security Considerations for Remote Electronic UOCAVA Voting* . National Institute of Standards and Technology, Information Technology Laboratory, Gaithersburg.

Hyperledger Project. (2017, 03). *Hyperledger Fabric V1 Github*. Retrieved 11 23, 2017, from https://github.com/hyperledger/fabric

Hyperledger Project. (2018). *Composer*. Retrieved 03 31, 2018, from Hyperledger Docs: https://hyperledger.github.io/composer/

Hyperledger Project. (2018, 03). *Composer Playground*. Retrieved 03 2018, from IBM Bluemix: https://composer-playground.mybluemix.net

Hyperledger Project. (2018, 04). *Go Fabric SDK*. Retrieved 04 2018, from Github: https://github.com/hyperledger/fabric-sdk-go

Hyperledger Project. (2018, 03). *HLFv11 Github*. Retrieved 03 2018, from Github: https://github.com/hyperledger/fabric/tree/release-1.1

Hyperledger Project. (2018). *Hyperledger Composer Command Line*. Retrieved 04 23, 2018, from Hyperledger Docs: https://hyperledger.github.io/composer/latest/reference/commands.html

Hyperledger Project. (2018, 03). *Welcome to Hyperledger Composer.* Retrieved 04 2018, from Hyperleger Tutorials: https://hyperledger.github.io/composer/latest/introduction/introduction

Kaspersky. (2016, 09 15). *Can you make voting safer?* Retrieved 02 08, 2018, from Kaspersky: https://www.kaspersky.co.uk/blog/economist-kaspersky-contest/7656/

Kirby, K., Masi, A., & Maymi, F. (2016). *Votebook.* New York University.

Krimmer, R. (2006). Electronic Voting 2006. *2nd International Workshop Co-organized by Council of Europe, ESF TED, IFIP WG 8.5 and E-Voting.CC.* Bregenz: Gesellschaft für Informatik.

Linux Foundation. (n.d.). Retrieved from Hyperledger Project: https://www.hyperledger.org

Madise, U., & Martens, T. (2005). *E-voting in Estonia 2005. The first practice of country-wide binding Internet voting in the world.* National Electoral Committee; Tallinn University of Technology. Estonia: National Electoral Committee.

Mason, R. (2017, 09 13). *Theresa May's snap election cost taxpayers £140m.* Retrieved 04 29, 2018, from The Guardian: https://www.theguardian.com/politics/2017/sep/13/theresa-mays-snap-election-cost-taxpayers-140m

McCorry, P., Toreini, E., & Mehrnezhad, M. (2016). *Removing Trusted Tallying Authorities.* Newcastle University, Computing Science. Newcastle-upon-Tyne: Newcastle University.

Nakamoto, S. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System.*

Ripple. (n.d.). *Ripple*. Retrieved 03 29, 2018, from https://ripple.com/
Scherer, M. (2017). *Performance and Scalability of Blockchain Networks and Smart Contracts.* UMEA University.
Schwartz, D., Youngs, N., & Britto, A. (2014). *The Ripple Protocol Consensus Algorithm.* Ripple Labs Inc, San Francisco.
UK Government. (2013). *Electoral Registration and Administration Act 2013.* UK Legislation Act.
UK Parliment. (2017, 09 13). *Cabinet Office Consolidated Fund Standing Service:Written statement - HCWS130*. Retrieved 04 29, 2018, from UK Parliment: https://www.parliament.uk/business/publications/written-questions-answers-statements/written-statement/Commons/2017-09-13/HCWS130/
UK Political Info. (2015, 05). *2015 General election results summary*. Retrieved 04 18, 2018, from UK Political Info: http://www.ukpolitical.info/2015.htm
Vukolic, M. (2017). Hyperledger Fabric. *Swiss Blockchain Summer School.* Zurich: IBM.
Vukolic, M. (2017). *Rethinking Permissioned Blockchains.* IBM Research. Zurich: IBM.
Vukolic, M., Sousa, J., & Bessani, A. (2017). *A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform.* IBM Research, Zurich.