Online Voting Protocols with a Blockchain Ledger

Submitted April 2018, in partial fulfilment of the conditions for the award of the degree Computer Science MSci.

Luke de Beneducci

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature: ___________________

Date: _____/_____/_____

Abstract

This dissertation explores current online voting protocols and the viability of an online voting protocol with a blockchain ledger. An exploration of these concepts is undertaken as a software project.

In the course of the literature review, several modern electronic voting protocols have been explored and compared. Blockchain technology has also been studied, examining its key features for use in electronic voting. From this analysis, the key elements of these electronic voting protocols were selected to implement an electronic voting scheme with a blockchain ledger.

The exploratory electronic voting scheme has been created using Python with the Flask framework to create a RESTful web application. Solidity, a language for writing blockchain smart contracts, was used to create an Ethereum smart contract that functions as the ledger and election authority for this voting scheme.

Contents

1 Introduction

Secure and trusted elections are a fundamental component of any functioning democratic institution. From smaller scale private elections to national government elections trusted voting protocols are needed to ensure that people's voices are heard. This leaves an area open for low error rate elections with increased accessibility and a reduced security risk.

One of the main obstacles to online voting lies in creating an underlying level of confidence between voters, the organisation hosting the vote, and the system to be put in place. As seen in the recent French election where electronic voting was halted due to cyber security concerns[37]. Public opinion of the system to be used is a key concern and trust must be built and maintained between the system and the voters. To answer these issues, a system must be built with extreme authentication, security, and vote verification.

The most recent technological innovation with the ability to alter the landscape of electronic voting is the blockchain, an open-faced, distributed, append-only ledger. The original intention of the blockchain, as used in the Bitcoin currency[30], was to remove the centralised role of banks in maintaining a financial transaction database. The technology is currently being re-used by researchers in various fields to solve many other problems, such as blockchain oriented smart cities[21]. The blockchain ledger is maintained by an open-membership, decentralised, peer-to-peer network of computers. Online voting protocols with verifiability rely on a publicly view-able bulletin-board for a consistent display of currently cast votes in order for voters to be able to audit their votes. This bulletin-board is normally a traditional database of cast votes, but, by using the blockchain's global nature and immutable ledger to store voting data, it means that once a vote is verified and cast into the system, it cannot be removed and unverified votes cannot be injected into the system at a later time.

Online voting has the potential to re-imagine smaller scale elections such as those attempted in this feasibility study. With increased trust and security, as well as increased anonymity, smaller groups will have access to efficiently run and extremely secure elections. I feel that voting is a key area of society in both large and small-scale use cases and for these reasons I see the need for further research in this area. This project will focus on the potential for blockchain to increase trust and security in online elections, as well as evaluating the costs of voting via blockchain.

1.1 Aims

The first research aim of this project is to evaluate the existing security protocols for online voting. Comparing them to the eight key electronic voting principles of Bertrand Haas[17] to determine the most suitable protocol for an online election. Following on from this the second research aim of the project is to evaluate the use of blockchain technology to create an immutable vote storing ledger for ease of vote authentication. The software aim of the project is to produce a small-scale trial system implemented as a website to explore whether blockchain technology has a place in online voting systems.

1.2 Objectives

The key objectives of this research are:

- To explore the design of currently available voting protocols, the entities involved in the election, the features they contain, and how secure they are.
- To compare each protocol to the eight principles of an electronic voting platform and select elements from the protocols by contrasting them against each other to determine the most suitable features for a web-based electronic voting platform.
- To examine and understand how blockchain technology works as a distributed append-only ledger and show how this aspect of the blockchain makes it suitable for electronic voting.
- To determine the best front-end languages to build this project. To run a comparison of different available technologies that are suited for back-end security work, a database, and a blockchain ledger.

- To implement as a piece of software an electronic voting client with the appropriate front-end, back-end, blockchain, and hosting technologies.

1.3 Motivation

Online voting has the opportunity to influence many people throughout the world. This is due to (amongst other factors) increased population growth, the spread of democratic voting systems to less developed countries, the need for greater voter participation, and an easier method of access than paper based ballot systems currently employed by countries worldwide.

Current methods of paper voting require an expense of both effort and time on behalf of both the voter and the organizer. If secure, trusted, and reliable electronic voting protocols are developed not only does the population of a democracy benefit due to easier access for expatriates, rural voters, less-able voters, and regular voters. The government also benefits by saving on the overhead monetary and time costs of setting up and manning voting centers and tallying votes.

In the 21st Century we have advanced almost every aspect of our lives to take advantage of the technological development that the world has gone through. Significant work has been put into exploring electronic voting strategies both as electronic vote recording devices at polling locations and as online web-based voting platforms. Direct recording electronic voting machines (DREs) provide some benefits over traditional paper based ballots including: increased speed of vote counting; the use of assistive technology for the less able; increased satisfaction[14]; and reduced overhead in printing and managing ballot papers. DREs do not however, deliver any significant increase in voting speed and have the potential to be tampered with relatively easily[43]. The work of Sarah P. Everett et al.[14] shows that there is no difference in human error rate between DRE and ballot based voting, and voters are still required to perform the somewhat laborious task of visiting a polling location.

Online voting brings with it the benefits offered by DREs with added extras including: the ability to vote anywhere in the world and whilst on the move; increased accessibility for those unable to reach or further away from polling stations; and increased accessibility for those with disabilities. A study on remote voting in the Estonian 2007 parliamentary elections showed that electronic voting increased voter turnout in those that lived further away from polling locations[5]. This shows the unique potential of online voting as a way to increase voter participation in the election process and demonstrates the need for more work in this area. There is also the belief that online voting can impact parties with support among the lower socioeconomic classes and improve support for parties with middle to higher classed supporters. However, a study conducted in Brazil in 2017[28] shows there is no systematic difference in voter choices between offline and online elections. The belief also exists that online voting could widen the digital divide and affect voting by the elderly, due to the perceived view of them as less technologically able. However, a study of senior citizens by Sarah Hyde and Ruth Abbey[20] found that the majority of participants where well "informed and balanced cyber-enthusiasts" who have embraced technological advances especially in their political lives.

Online voting also has the ability to draw in younger voters, often the group least likely to vote in an election, as well as increase overall election attendance. United Kingdom election turnout among registered voters has not been above 75% since 1992 and in 2001 turnout fell to 59% and has not climbed above 70% since[22]. Voter turnout in the USA has only been above 60% four times in the twenty-two presidential elections since 1932 and in 1996 dropped below 50%[6] (with election attendance often up to 20% lower in the years without a presidential election). Online voting has the potential to reverse this downward trend and involve the younger demographics in the voting process again, as shown in a Houses of Parliament report[31], youth participation in the political process has increased as access to technology has grown.

2 Related Work

There have already been use cases of electronic voting in Europe: in France; the Netherlands[23]; Norway[24]; Estonia[36], as well as many other countries inside and outside of Europe. The UK has even attempted small scale local trials of electronic voting[8]. However, all of these studies have understandable security concerns. Putting to one side resistance to external vote tampering by hostile entities, Bertrand Haas has set out eight fundamental requirements which ought ideally to characterise any electronic voting system[17].

- Privacy: No one should be able to associate a particular vote with a voter except for the voter.
- Free Will: Voters should be able to cast their votes of their own free will.
- Reliability: Votes should be transmitted and counted as intended.
- Prevention of Ballot Fraud: Ballots should be protected against fraud.
- Prevention of Ballot Trade: The system should not allow for the selling and buying of votes.
- Accessibility: Any potential voter should be able to vote. In particular discrimination with respect to physical disabilities should be minimised.
- Affordability: Towns and counties should be able to afford it.
- Simplicity and Usability: The process of casting a vote should be understandable and easy enough to be used by anyone.

As regards the question of security in particular, protocols are often expanded from previous designs to generate new protocols with added measures and features. As such there are many electronic security protocols to examine. An early precursor, developed by researchers Cramer et al.[10] in 1996, was the system of Multi-Authority Secret Ballot Elections which laid a strong foundation for electronic voting principles. Following on from this the Civitas protocol[29] by Clarkson, Chong, and Myers proposed in 2008 advanced the system proposed by Cramer et al. to introduce coercion resistance. Also developed in 2008 was Helios, an electronic web-based voting system designed by Ben Adida[1], based on the work of Cramer et al., with variants proposed by Benaloh[2] for elections that have a low coercion risk. Helios was further expanded by Véronique Cortier et al.[9] into a "Helios-like scheme" that integrates "ballot privacy and strong verifiability" called Belenios. This was then expanded upon again by Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo[7] to produce the BeleniosRF system, a new voting scheme which provides receipt-freeness and end-to-end verifiability.

There currently exist several proposals for the use of blockchain technology in electronic voting systems. The Abu Dhabi Stock Exchange has launched its own blockchain voting platform[19] and several companies have proposed such systems including: Votem[26]; FollowMyVote[25]; and Blockchain Voting Machine. These systems function by using traditional voting protocols with the blockchain as an immutable append-only ledger for storing ballots. Patrick McCorry et al.[27] have proposed a system for small scale boardroom style elections without a single trusted authority in which the participants taking part in the vote are responsible for the organisation of the election. Their proposed system takes advantage of the blockchain's inherent peer-to-peer capabilities in order to use the blockchain not only as an immutable ledger but as an agent to implement the voting protocol.

The blockchain as implemented in Bitcoin and Ethereum is not currently a feasible option for large scale national elections with millions of participants due to inbuilt scalability issues and the cost of transactions. Bitcoin only supports on average 7 transactions per second, with a theoretical maximum of 10 transactions per second and each transaction dedicates 80 bytes to a data preamble. Ethereum chooses instead to measure storage and computation with a gas standard and the network sets a limit on the amount of gas that can be used by its users. The average transaction fee for Ethereum ranges from £0.15 to £1.10 whilst the average transaction fee for Bitcoin ranges from £3.70 to £15[3]. Voting on this national scale would require a specially built blockchain designed by the governments or organisations in charge of hosting the election for sole use as a voting system. For these reasons we choose to explore the use of the blockchain as a voting ledger in a proof-of-concept feasibility study.

3 Voting Protocols

Internet voting protocols have the ability to be run as quickly or as slowly as required, with advance or same day voter registration possible. As well as a customisable timeline for the vote casting and tallying stages. For these reasons, election administrators have as much flexibility as they require when determining how to run their election.

Many protocols attempt to address problems along a common theme. These shared problems are explained below:

Coercion Resistance: This aims to address the problem of voter coercion by force or bribery. Protocols with coercion resistance aim to do this by not providing the voter with a method to prove to a coercer which way they have voted. Some protocols offer the voter alternative methods of voting in the case that they are being coerced.
Receipt Freeness: Another method with which to oppose potential coercion. Protocols with receipt freeness do not provide the voter with any information which could generate a receipt to prove which way they have voted.
End to End Verifiability: Allows for individual participants or larger organisations to inspect elements of the voting process in order to confirm the election was not corrupted.

Many voting protocols also contain similar cryptographic primitives. An explanation of these is provided below:

Public Key Cryptography: Also known as asymmetric encryption it makes use of public and private key pairs. A mathematical relationship is required between key pairs for the encryption to function securely. Security of encryption is controlled by two factors: the difficulty of calculating a user's private key from their corresponding public key; and the secrecy of a user's private key. In order to send a secure message, the sender(A) would encrypt their plaintext with the receiver's(B's) public key, creating an encrypted ciphertext that only B can decrypt with their private key.
ElGamal and RSA Cryptography: Two of the most common public key cryptography algorithms in use are ElGamal[13] and RSA[32]. They differ due to the mathematical relationship they use to produce their public and private keys. RSA bases its security on the difficulty of factoring the products of large prime numbers, whereas ElGamal bases its security on the difficulty of calculating discrete logarithms within a large prime modulus (this is known as the discrete logarithm problem).
Homomorphic Encryption: A subset of encryption that allows calculations to occur on encrypted ciphertexts without revealing the plaintext whilst having the same effect on the plaintext underneath.
Proofs: A zero-knowledge proof is a proof such that one party, the prover, can prove that they know some information, z, about an object to another party, the verifier, without revealing any other details about the object except for their knowledge of z.
Signatures: Digital signatures are a method by which a user can sign an object in such a way as to prove that they and only they have sent it. They are based on encryption algorithms such as RSA or ElGamal as public and private keys are required. To create a signature data is first hashed using a hashing algorithm and then this hashed data is signed with the signer's private key. To verify a signature the verifier creates a hash of the original message using the same hashing algorithm. Next, the verifier decrypts the signer's encrypted hash with the signer's public key. Finally, the verifier will compare both hashes in order to confirm that they are equal.
Signatures on Randomisable Ciphertexts: These combine three of the above cryptographic primitives: public key cryptography; proofs; and signatures. They work such that given an encrypted message with a valid signature, one can re-randomise the encryption on the message and adapt the signature to work for the new ciphertext[4].
Mix Nets: A mix net takes in messages from multiple senders and randomises their ordering before forwarding the messages onwards. In order to keep secret who has sent which message.

Many voting protocols are based around similar core ideas and entities. A description of these elements is below:

Voter Registration and Setup: Voters register to vote in an election and are given public and private encryption and signature keys generated by the election administrator or authority. The server's private and public keys are also generated. The voter is then added to the database of registered voters and is able to participate in the election.
Casting a Vote: A voter's choice of vote is encrypted using the chosen method of encryption. The ballot is then signed using the voter's signature key this signature is used so that the voter can verify their vote after the election concludes. The ballot can also contain a proof confirming that the vote is valid. The ballot is finally sent to the ballot box and the publicly viewable bulletin board.
Anonymisation Method: There are a varying number of methods available for anonymising a voter including: mix nets; homomorphic encryption; and signatures on randomised ciphertexts. Protocols can sometimes combine these methods together for increased anonymisation.
Tallying: A property of most protocols is homomorphic encryption allowing for tallying before the ballots are decrypted. The vote is then decrypted using the server's decryption key. This property allows for voter privacy to be maintained during the tallying stage of the election.

3.1 Multi-Authority Secret Ballot Elections

Cramer et al.[10] developed and proposed this protocol for Multi-Authority Secret Ballot Elections in 1996. This protocol takes into account universal verifiability, privacy, and robustness. It was a very large step towards a practical protocol that could be used in real world elections.

Protocol Design:

Preparing a Vote: Before casting a vote the voter must prepare a vote that is masked. This is a randomly prepared vote that is later adapted for use by the voter.

Proof of Validity: For performance reasons witness-indistinguishable proofs are chosen over zero-knowledge proofs. At the time it was thought that zero-knowledge proofs required many iterations to achieve high levels of confidence, this thinking has since then changed with the introduction of practical non-interactive zero-knowledge proofs. Using witness-indistinguishable proofs reduces the effort required by the voter, however, the proofs are weaker. The proof of validity also confirms that the ballot contains a valid vote without revealing any of the details of the ballot.

Casting a Vote: The prepared masked vote is adapted to represent the voter's actual choices.

Authorities & Tallying: When using this encryption scheme having only one authority allows for the potential reveal of votes. To combat this issue the protocol chooses to share votes across multiple authorities. The authorities can then decrypt votes using the public keys supplied to them via the voter's private channel.

Benefits and Drawbacks:

The main benefit to this protocol is that elections can be prepared offline in advance due to the fact that voters are given randomly prepared votes to adapt later. Therefore casting a vote requires less effort on behalf of the user. A drawback to this scheme is that if a number of authorities collude then they can choose to reveal individual votes. The scheme also does not consider the property of coercion resistance or receipt freeness as they had not been theorised when this paper was published. Due to the lack of these features when using this protocol there is the potential for voters to sell their votes or to have their vote choice coerced.

3.2 Civitas

The Civitas protocol was proposed and developed in 2008 by Clarkson et al.[29]. It introduces the concept of coercion resistance into online voting protocols whilst retaining voter privacy and vote verifiability. The Civitas protocol differs slightly in which entities it contains the protocol includes a supervisor for administering an election, a registrar for authorising voters, registration tellers for generating credentials and tabulation tellers to tally votes, as well as having multiple ballot boxes.

Protocol Design:

Election Setup: Each entity in the system posts their relevant keys to the bulletin board in order for them to be used in vote verification. The keys used in a Civitas election registration are RSA keys and the encryption keys are ElGamal. The registrar then posts a list of all registered voters to the bulletin board, containing their IDs and public keys.

Casting a Vote: The registration teller generates the credentials for each voter to use during the voting stage. The voter and registration teller run a protocol that allows the voter to construct a private credential. Voting can then take place immediately or a while after registration has occurred. In order to vote each voter submits their private credential and choice of candidate, both encrypted via ElGamal, along with a proof of a valid ballot.

Coercion Resistance: The key idea in this protocol that allows voters to resist coercion and defeat vote selling is the idea that voters can submit a vote with fake credentials instead of their real credentials. They can then behave as they are demanded. The fake credentials are generated by a voter running a local algorithm.

Revoting: Voters may submit more than one vote per credential if such votes are not allowed votes submitted under duplicated credentials are removed. However, if the election allows for re-votes then the voter must indicate which previous votes are being replaced. The indication comes in the form of a proof that demonstrates knowledge of the credentials and choice used in the previous vote.

Tallying: Tabulation tellers collectively tally the election they retrieve the data from each ballot box as well as public keys from the bulletin board. They then verify votes by checking each vote's proof for correctness. Duplicated votes and votes with invalid proofs are then eliminated from the election. Votes and credentials are then anonymised via the use of a mix-net. Votes submitted with invalid credentials are then removed. Only the final votes are decrypted not the credentials, this means that the final vote count can be publicly tallied.

Benefits and Drawbacks:

The key benefit to this protocol is its coercion resistance this added property pushes Civitas one step closer to being an optimal solution. However, the implementation of this coercion resistance leads to drawbacks in usability.

This protocol introduces coercion resistance by allowing a voter to vote first with fake credentials and then discount that vote by re-voting at a later time with their actual credentials. This impacts the usability of the system as it requires the user to run an algorithm producing fake credentials, and relies on some understanding on the voter's part of how these credentials work. It also requires re-voting which is not part of a normal election and can lead to confusion for those using the system.

3.3 Helios

Helios is a protocol designed by Ben Adida in 2008[1] it is an open audit web-based system for low coercion elections. The protocol states that one should "trust no one for integrity, trust Helios for privacy" relating to a compromise between unconditional integrity or unconditional privacy. Helios chooses to favour unconditional integrity. Even if every election administrator is corrupt they cannot fake a tally. The protocol chooses to ensure privacy by making Helios the only trustee, privacy is therefore only guaranteed if one trusts the Helios server.

Protocol Design:

Preparing a Vote: The protocol uses Benaloh's verifiable voting model[2] which states that a ballot can be filled and viewed by anyone at any time, but it is only authenticated at ballot casting time. The vote choices in this model are encrypted with ElGamal public key encryption.

Vote Auditing: A user can audit their vote before casting it, this option displays the cipher and randomness used to encrypt the vote. This allows for verification but also increase the possibility of coercion.

Casting a Vote: To cast a vote the voter will seal the ballot and are then called on to authenticate. If the voter is authenticated the ballot is accepted. The bulletin board displays the voter's ID, name and ballot, in order for voters to confirm that their ballot is present.

Anonymisation: After the election is finished the votes held on the bulletin board are passed to a mix-net for shuffling. Shuffling this way anonymises voters by re-encrypting votes, and placing them into a random order.

Tallying: The votes are decrypted and then counted. Later versions include the ability to use homomorphic encryption for tallying before decryption increasing voter privacy. However, earlier versions of the protocol did not contain this feature.

Benefits and Drawbacks:

The main benefit of Helios is that it is a web-based implementation of a real world voting model, it was the first publicly available online voting system and shows a proof of concept for the idea of web voting. The system draws another strength in its ability to be openly auditable. Any party running in the election or any independent entity can audit the votes and results of the system to ensure trust in the system. The main drawback to the Helios protocol comes in terms of security, incorrect shuffling, ballot corruption, and other cases are a possibility if the Helios server is corrupt. However, breaches such as these would be caught during user auditing if enough users audited, or if the feature was made mandatory. Helios also warns voters that it should only be used in low coercion environments as there is the risk of coercion.

3.4 BeleniosRF

BeleniosRF is a protocol developed by Cortier et al. in 2016[7] that expands upon earlier work on the Belenios protocol[9]. This new protocol offers receipt freeness, in which the voter has no knowledge of the randomness used to form their ballot. The protocol also enables coercion resistance without the need for an extraneous strategy that the voter must undertake as in the case of Civitas.

Protocol Design:

Cryptographic Primitives: BeleniosRF uses cryptographic primitives such as signatures on randomisable ciphertexts as well as ElGamal encryption, signatures, and proofs in order to provide receipt freeness.

Receipt Freeness: In the majority of electronic voting protocols the voter can record the randomness used to encrypt their ballot. With this recorded randomness the user could reverse engineer a ballot to prove how they have voted and produce a receipt. As a consequence a user could be coerced or could sell their ballot. With signatures on randomisable ciphertexts the server will re-randomise the encryption of a ballot and then adapt the corresponding signature and proof.

Verification: Due to the fact the server server adapts the signature and proof when a ballot is re-randomised the voter can still check that their vote is legitimate as the newly randomised ballot comes with a signature that is still valid under the voter's verification key. However, the voter cannot record the randomness used as it has been changed by the server therefore the voter cannot produce a receipt.

Benefits and Drawbacks:

Much like the Civitas protocol the BeleniosRF system allows for coercion resistance, however, its main benefit in this regard is that it imposes no requirement on the voter to avoid coercion. The voter simply cannot provide a receipt to prove how they have vote. BeleniosRF might not be quite as fast a system as Helios, however, it makes up for this in terms of its increased security over the BeleniosRF protocol. Another advantage of Helios over BeleniosRF is that the registrar and voting server of BeleniosRF can collude in order to alter votes. This can be prevented however, by having voters generate their own signing keys and having the registrar simply authorise their vote.

3.5 Comparison

Cramer et al.'s solution Multi-Authority Secret Ballot Elections is a good protocol for its time, though it was proposed before coercion resistance and receipt freeness where fully developed concepts. It is however, a good starting point for many of the future protocols proposed as it allows for privacy, reliability, accessibility, and simplicity in use key features as outlined by Haas[17].

Civitas by Clarkson et al. is the one of the first examples of a protocol that was developed with the key requirement of coercion resistance in mind and it is a good first attempt at this with a robust solution to the problem. However, it does require that the user perform extra, somewhat complex, steps in order to provide coercion resistance and this usability impact does affect its usefulness as a voting protocol where ease of use is a key feature. It does cover more of the requirements as laid out by Haas such as voter free will, and the prevention of fraud and ballot trade.

Ben Adida's Helios does not propose any cryptographic novelty and is instead a practical implementation of Multi-Authority Secret Ballot Elections with some variations. It is for use in a low coercion environments only and as such does not explicitly cover Haas' requirements for voter free will and prevention of ballot trade. It does make up for these with its simplicity and usability as well as its accessibility.

BeleniosRF by Cortier et al. is the most well rounded of the above protocols covering seven out of eight of Haas' requirements: privacy; free will; reliability; prevention of ballot fraud and trade; accessibility; and simplicity. Whilst the final requirement affordability is still a theoretical.

A comparison of the features offered by the above protocols can be found in Table 1 below.

	Multi-Authority...	**Civitas**	**Helios**	**BeleniosRF**
Year Created	1996	2008	2008	2016
Ballot Privacy?	✓	✓	✓	✓
Robust?	✓	✓	✓	✓
Personally Verifiable?	✓	✓	✓	✓
Universally Verifiable?	✓	✓	✓	✓
Coercion Resistant		✓		✓
Receipt Free?				✓
ElGamal Encryption?	✓	✓	✓	✓
Homomorphic Encryption?		✓	✓	✓
Anonymisation Via Mixnet?		✓	✓	✓
Signatures on Randomisable Ciphertexts?				✓

Table 1: Comparison of Security Protocols

4 The Blockchain

The first practical implementation of a blockchain network was theorised and implemented by Satoshi Nakamoto in their white paper for the Bitcoin currency[30]. Though the phrase 'blockchain' was never explicitly stated in Nakamoto's paper the designed system called for a distributed network of encrypted transaction blocks. Since then many different blockchains have been developed, however, they all function on the same three key principles:

- Cryptography
- Distributed Networks
- Consensus Protocols

The signatures and public key encryption methods that make up the cryptography of the blockchain are well defined. Public key cryptography and digital signatures are key elements of blockchain security and integrity. Public keys are used with a hash function to create public wallet addresses that users can send and receive funds from. Digital signatures are used to ensure that a transaction is authentic and that an unauthorised party is not attempting to send funds that do not belong to them. Miners will confirm the signatures on transactions to verify they are valid.

The distributed network is made up of a large number 'nodes' each of which has a full copy of the blockchain's transaction history, and is able to mine new blocks. Each transaction request is broadcast to every 'node' in the network, spreading a copy of each transaction over the whole network, in order to stop wallets from double spending. A transaction on the blockchain is performed as follows:

- Transaction requested
- Transaction broadcast to peer-to-peer network
- Network validates the transaction
- Combined with other valid transactions to form a block
- New block appended to the chain of blocks

Most of the major choices for a blockchain network come in the design of its consensus protocol. The consensus protocol as theorised for the Bitcoin currency was a proof-of-work method, most major blockchains still use this method. Work is being done to design and create a consensus method based around proof-of-stake towards which some languages are intent on migrating.

4.1 Proof-of-Work

The proof-of-work system as designed for the Bitcoin currency involves, "scanning for a value that when hashed...the hash begins with a number of zero bits"[30]. The work can then "be verified by executing a single hash" This means that whilst the work done to validate a transaction can be large, once validated other blocks in the network can very quickly confirm that the transaction is correct.

Once this work has been done a block cannot be changed without the monumental effort of redoing the proof-of-work for this block and every following block, as each block is hashed with the data from their previous block.

The consensus opinion is therefore maintained by the largest chain of blocks which represent the most work done. Satoshi notes that, "If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains."[30] In order to modify a past block the requirement would be to preform the proof-of-work on that block and every following block in order to match and then surpass the work done by honest nodes, this would require that the attacker controls 51% of the networks computing power.

The computation work done to calculate the correct hash value is called mining and the node which correctly mines the next block receives a small quantity of currency as a reward. The difficulty of the proof can be altered in order to keep the average number of blocks mined per hour fixed.

4.2 Proof-of-Stake

Proof-of-stake is the main alternative consensus method to proof-of-work and is theorised as a a lower energy alternative to proof-of-work. An estimated six-hundred trillion hashes occur every second trying to mine the next block on the Bitcoin network. As hashes are not mined in proof-of-stake blocks are consider to be forged.

Rather than requiring a that the prover performs a certain amount of computational work, a proof-of-stake system instead requires that the prover shows ownership of a certain amount of currency. Blocks are then distributed to be forged in a psuedo-random way determined the wealth of each user. The cryptographic problems involved are therefore a lot simpler to solve, you only have to prove that you own a certain portion of all the coins available in a given currency. In such a system someone with five times more currency than you would forge five times the amount of blocks. As you don't mine new currency in a proof-of-stake blockchain forgers are instead gifted with the transaction fee attached to each transaction.

Matthew Czarnek, preformed a cost analysis of the forging of nodes in a proof-of-stake system, Nxt, compared to Bitcoin[11], and found that, "Bitcoin uses approximately 8000 times more energy to power the network" and that, "Bitcoin costs approximately 2200 times more than Nxt to power the network". This illustrates the potential cost benefits of a proof-of-stake system at large scale.

A additional benefit of proof-of-stake is the potential increase in cost to an attacker, the attacker would need to own a near majority of all of the available currency which would severely devalue it.

However, as users do not also have anything at risk, in terms of computational power, they could quite easily work on multiple chains in order to try and double spend. Some argue that because a proof-of-stake network requires little computational effort users could vote for multiple different block histories and thereby prevent a consensus.

4.3 Ethereum

Ethereum[15] is an open-source proof-of-work blockchain that was designed in order to be more than just a digital currency. It allows anyone to produce, deploy, and use decentralised applications that require blockchain technology.

Rather then giving individual users a predefined set of transactions as Bitcoin does users can define their own transactions via smart contracts. These smart contracts run on the Ethereum Virtual Machine(EVM). The EVM is Turing complete which means that it can theoretically solve any computation problem, given enough time and memory. This means that fully functional distributed application development is possible. The EVM compiles high-level smart contracts down into Ethereum bytecode which is then run by the Ethereum network. Each node in the network runs the EVM in order for each node to verify each transaction listed in a block.

In order to facilitate this each computational operation on the Ethereum network is assigned a gas cost, for example the EVM operation code ADD costs 3 gas[16]. Each transaction must include a gas limit and a fee they are willing to pay for the gas used. Miners have the option of including the transaction and collecting the fee or ignoring the transaction if they think the fee is too low.

The blockchain has useful potential for online voting, with inbuilt digital signatures and encryption often found in voting protocols, as well as a strong immutable ledger that allows for tamper proof voting. Ethereum is a particularly strong choice for this with its design allowing for implementing and developing custom smart contract decentralised applications.

5 Methodology

The literature review portion of this project has been undertaken as a continuous literature research cycle. Starting from a simple review on the topic of online voting, finding research papers on the protocols and systems that exist, as well as papers on the usage and implementations of these protocols in the real world. As the project has progressed so has my sphere of research as this is the case my literature review expanded to include cryptographic primitives such as public key encryption, signatures, and proofs. My project then further expanded into a study of the block chain and its potential for use as an immutable ledger of cast votes in electronic voting. To facilitate this more literature was collected to study current examples of this implementation and to gain a deeper understanding of the blockchain.

The software development portion of this project followed on from the initial research and study phase, and took place alongside additional research. The main considerations for the software design and development portion of this system where the choices of programming languages and frameworks for the front-end and back-end. The front-end handles the user interaction portion of the voting framework, whilst the back-end is the code for the logic of the website, and the blockchain.

For the software development the agile design methodology has been implemented. This allows for rapid prototyping based around sprint cycles using project management tools such as Git and Trello. These help maintain a fast development cycle and allow me to produce quick iterations of the product as requirements change, feedback is received, and new literature alters design plans.

An alternative development approach was considered, mainly the Waterfall methodology. I opted not to choose this method as I believe it is too inflexible for this type of project. The Waterfall method has testing take place at the end of the production phase and therefore might not have allowed for fast enough adjustments to changes in requirements and any fundamental issues.

Unit Testing occurred as an automated process in both the Python website back-end and the JavaScript and Solidity blockchain voting protocol. This testing helped to drive the development process and meant that any potential issues were straightened out before they could arise in an integrated environment. Integration testing was performed manually by myself as individual components were finished and able to be assembled together to form the full system. These enabled fine tune adjustments to the higher-level elements of the system such as user interfaces and ease of use.

In order to evaluate the potential of blockchain technology to enhance online voting protocols I chose to examine the costs of its transactions and its potential for scaling. Further discussion of the evaluation will occur in section 8.

6 Design

I have selected key elements from the voting protocols studied in section 3 in order to design the voting protocol that has been developed. It represents online voting with a voting booth and voter registration website front-end whilst the other elements such as election set up and vote verification are handled back-end. It has been created to explore how online voting functions and as an study of blockchain technology in online voting.

6.1 System Entities

The chosen system contains the following entities:

- **Voters:** All registered and eligible voters for this election.
- **Election Administrator:** Runs the election server and determines both voter and ballot criteria.
- **Election Server:** Starts the election, registers and maintains the list of all registered voters and candidates.
- **Ballot Box:** Stores all pending valid ballots.
- **Blockchain:** A publicly viewable list of all verified and cast ballots.
- **Authorities:** The authority or authorities able to view and tally the ballots.

6.2 System Protocol

The voting protocol has five stages: voter registration; election setup; vote casting; vote tallying; and vote verification.

Voter Registration: The voter will choose to register with the election administration server. Once a voter is confirmed and registered they will be added to the registered voter list maintained by the Election Administrator.
Election Setup: The election server's administrator determines when to start an election. The voter roll is confirmed, The ballot box is initialised, the bulletin board is setup.
Vote Casting: To cast a vote a registered voter must follow these steps:

- A registered voter will login to their account with their credentials.
- The voter will navigate to the vote page, select a candidate option and confirm their vote transaction.
- If the election smart contract confirms that a transaction is valid it will be mined and appended to the blockchain.

Vote Tallying: Both during and after the election the vote can be tallied in real time by any authority that wishes to download and monitor the blockchain's transaction history. The results of the election are also displayed in real time via the websites results page. The final tally will be calculated and confirmed by the election administrator after the election has been ended.
Vote Verification: During and after the election a voter can log in and verify their vote by searching the blockchain ledger for their specific account address.

6.3 System Protocol Flow

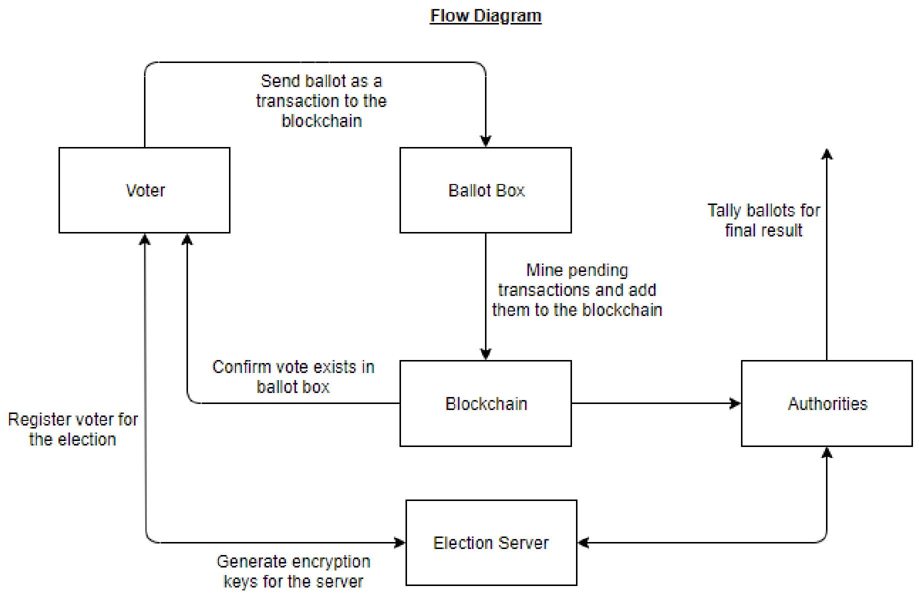

Flow Diagram
Send ballot as a transaction to the blockchain
Voter
Ballot Box
Tally ballots for final result
Mine pending transactions and add them to the blockchain
Confirm vote exists in ballot box
Register voter for the election
Blockchain
Authorities
Election Server
Generate encryption keys for the server

Figure 1: System Flow Diagram

7 Implementation

7.1 System Software

Stated below are the tools, software, and frameworks used to implement the practical portion of this dissertation.

- Languages: Python, HTML, CSS, JavaScript, Solidity, SQL
- Frameworks: Flask, Python, Trufflesuite
- Database: PostgreSQL
- Ethereum Tools: MetaMask, Ganache
- Cloud Deployment: Heroku
- Version Control: GitHub
- Project Management: Trello

The project's code has been stored in a repository on GitHub which was used as version control for the development of the project. The website portion of the code has been deployed to Heroku. However, the voting and results sections of the system have been disabled for the deployed version. They can currently only be run on a local Ethereum development test network due to limitations with the system.

This website's front-end was developed using HTML, CSS - with Bootstrap[39], and JavaScript. These are the current industry standard development languages for developing, styling, and interacting with websites and were therefore the obvious choice.

The main choice came in determining which language to use for the project's back-end logic. Consideration was between the languages, Python and Java, which both have similar strengths and drawbacks outlined in Table 2 below.

	Benefits	**Drawbacks**
Python	Large mathematical and cryptographic libraries Python Flask framework for RESTful architecture Object oriented programming Platform independent Fast development for quick prototyping Easy to learn	Harder to analyse code Comparatively slower
Java	Large mathematical and cryptographic libraries Java Play framework for RESTful architecture Object oriented programming Platform independent Mature and stable language	Can sometimes be verbose Not suited for high performance

Table 2: Comparison of Python and Java for back-end programming

The final choice of language was Python with which I had less hands on experience. Despite this I felt that it was the perfect choice for this project as it allows for rapid prototyping and fast development. Which are both key features of the agile design methodology. Python as an object-oriented language allowed for the easy separation of entities in the voting system.

Python also grants access to the excellent Flask framework[33] for developing websites with RESTful architecture. Initial testing was needed to pick up this particular framework, however, once an initial structure was produced the framework became more familiar and relatively easy to use. An alternative approach is the Python Django framework which offers a similar RESTful approach to web application design. The RESTful Architecture is beneficial as it decreases coupling between the server and client, this means that there is more separation between entities. It is also extremely useful if the server is interacting with a large number of clients that you do not have control over.

The data store used for this project is a relational database using PostgreSQL. This database is responsible for securely storing the voter roll and voter information, such as usernames, passwords, and other details.

The blockchain element for this project could have been developed as a standalone blockchain in either Python or Java. However, a simpler and more efficient method for this project was to use Ethereum smart contracts[15] implemented with their programming language Solidity[35] which allows for the design of custom tokens, in this case as a ballot for electronic voting.

Trufflesuite[41] is a JavaScript framework for writing Ethereum smart contracts it enables: automated smart contract testing; local blockchain networks; built in contract compilation; and many other features. Truffle's Ganache[42] is a graphical user interface for local blockchain testing, and MetaMask[34] is a web extension that allows for blockchain testing in the browser without needing to download the whole Ethereum node. These three tools together allowed for the testing, development, and integration of the Ethereum smart contract.

The hosting for the website was another concern. The website is deployed to a cloud server in this case Heroku[18], which has strong integration with Python as well as with Git and Postgres. Another alternative was the Google App Engine which contains similar functionality, not much seems to separate these two options but Heroku provided relational database support where the Google App Engine did not.

7.2 System Functionality

Figure 2, below, outlines the system's architecture showing some components of the system and their interactions. The components involved in the system are defined below:

- Clients - Voter clients performing interactions via a web browser.
- Server - The server can be deployed locally. An alternative server without blockchain functionality is available on Heroku's cloud platform.
- RESTful Application - The architecture of the system follows the RESTful model. Implemented via the Flask framework.
- Services - The services are the logic for the system, they include among others:
  - Voter Registration - Accesses the Postgres datastore to insert newly registered voters.
  - Voter Login - Accesses the Postgres datastore to read and confirm login details and then allow user access to specific pages.
  - Login Required - Controls who can access which pages and locks certain pages out of reach of users.
  - Cast Vote - Called when a user casts a vote. Sends a vote transaction to the blockchain.
  - View Results - Makes a call to the blockchain's smart contract to read the data and display results.
- Datastores - The user data is stored in a Postgres database. This includes login details such as email addresses and hashed passwords, as well as data on whether the user has voted. The vote and candidate data is stored on the blockchain via the smart contract and updated via transactions.

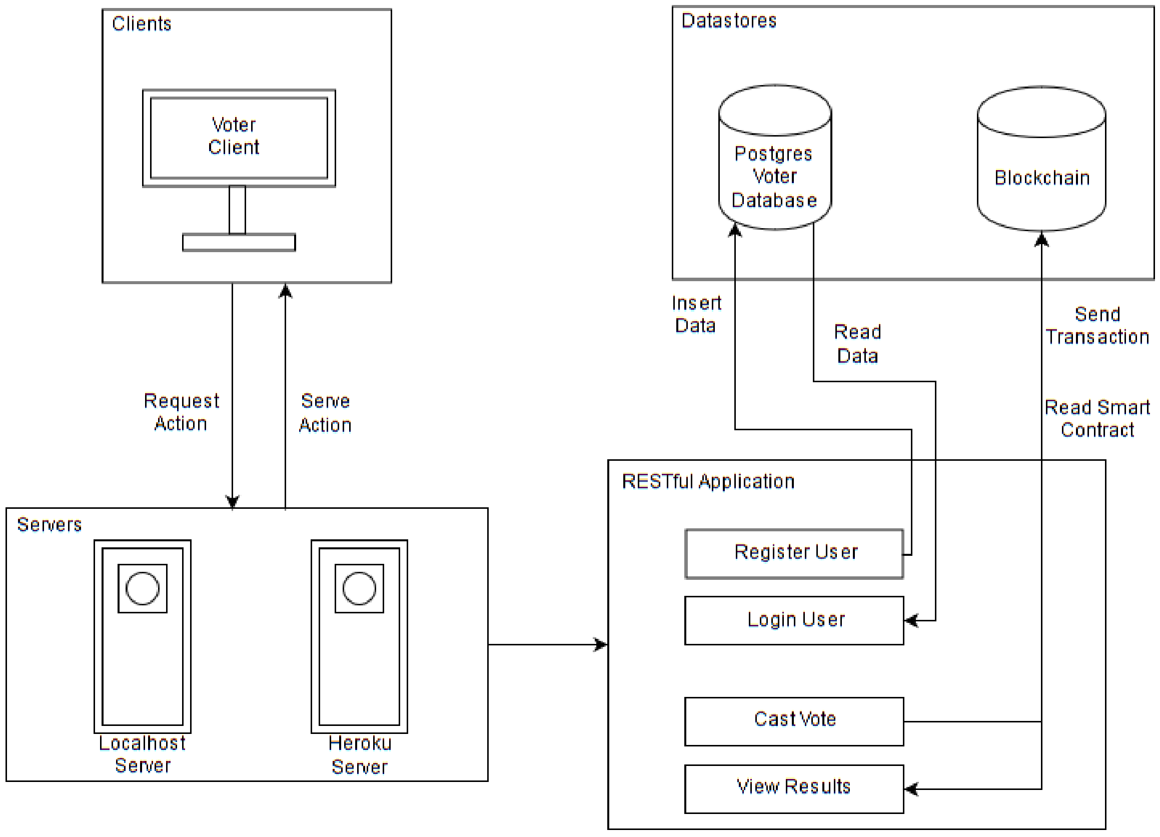


Figure 2: System Architecture Diagram

7.3 Source Code

The key files for the blockchain voting portion of the system are the *app.js* file, the *Election.sol*, and the *Election.json* file which are respectively located in the source code at:

Election-Website/project/static/Election/src/js/app.js
Election-Website/project/static/Election/src/contracts/Election.sol
Election-Website/project/static/Election/build/contracts/Election.json

The *Election.sol* file is a Solidity language file that defines the smart contract to be used on the blockchain, this file contains a struct for each candidate storing the candidate's ID, Name, and Vote Count. This contract handles creating the list of candidates in the election, storing which accounts have sent a transaction along the blockchain, and also keeps track of how many votes each candidate receives.
The *app.js* file is a JavaScript language file which allows web interaction with the blockchain and smart contract, this interaction occurs through a JSON (JavaScript Object Notation) file *Election.json*. JSON is a data interaction format and this file provides a version of the smart contract that can be manipulated via JavaScript code. The *app.js* file allows the reading of information from the blockchain and smart contract, including current candidates and their vote counts, it then renders these to the screen for the vote and results pages of the website. It also contains functions for casting a vote transaction to the

blockchain, and listening to transaction updates. Talking to Ethereum nodes from JavaScript files occurs via the web3.js library which allows for remote procedure calls.

They key files for the website front-end and back-end of the system are the two templates folders, the two *views.py* files, the *forms.py*, the *models.py*, and the *__init__.py* files. These are respectively located in the source code at:

Election-Website/project/account/templates/
Election-Website/project/account/views.py
Election-Website/project/users/templates/
Election-Website/project/users/views.py
Election-Website/project/users/forms.py
Election-Website/project/models.py
Election-Website/project/__init__.py

The *models.py* file is a Python file that defines the model used by the Postgres database, laying out the two tables, Users and Details as well as their relationships. It states how data should be initialised into the two tables, such as the hashing of passwords and the default value of the voted boolean to be false.

The *forms.py* file is a Python file that layouts the requirements and validators the for data being entered into the LoginForm and RegisterForm.

The Users/Templates folder contains two HTML files with CSS styling *register.html* and *login.html*. The *login.html* file renders the login page for the website, including the LoginForm. This form allows for comparison of entered data to the data stored in the Postgres database via the project's Python-Flask back-end. The *register.html* file renders the registration page to the user, including the RegisterForm. This form allows for the data entered to be stored into the Postgres database, assuming all validators are met, this means that new users can be added to the database as they register.

The Account/Templates folder contains three HTML files with CSS styling all of which are locked behind login access. *account.html*, *vote.html*, and *results.html*. The *account.html* file renders the user's specific account page, using Flask's login manager package in order to isolate accounts for a specific logged in user. Reading data from the Postgres database this page renders a user's, name, age, city, email address, and whether or not they have voted. This page also controls access to the voting and results pages depending on whether an election is in process and the user is eligible to vote or view results. The *vote.html* file renders the voting page which connects to the smart contract and allows the user to send a transaction on the blockchain via the *app.js* file. The *results.html* file renders the results page and allows the user to see the vote count for each candidate, this also connects via the *app.js* file.

The *views.py* file, contained in the Users folder, controls the RESTful functionality for the Users templates. This file allows for 'GET' and 'POST' requests for reading and writing data to and from the database. This file controls the registration, login, and log-out methods for user accounts via three routes and implements the functionality of the registration and login forms. The home route functionality confirms if the details entered into the form are present in the database and if so logs the user in and forwards them to the account page. The register route confirms that the entered details are valid and enters them into the database, then logs the user in and forwards the user to the account page. The log-out route removes the user from the logged in list and then returns them to the login page.

The *views.py* file, contained in the Accounts folder, controls the RESTful functionality for the Account templates. This file also allows for 'GET' and 'POST' requests for reading and writing data from and to the database. The account route simply renders the *account.html* page. The voting route allows the user to access the vote page as long as they have not already voted whereas the results route allows the user to view results once they have already voted. All three of these routes require the user to be logged in.

The *__init__.py* file is the main setup file for the app, called from the *run.py* file, this file sets up the database entry, password hashing, login management, and registers both the Users and Account blueprints allowing those routes to be accessed.

8 Evaluation

8.1 Protocol Evaluation

The protocol manages to adhere to the majority of Haas' electronic voting requirements. Some properties are not fulfilled, mostly due to limitations in the system, this is discussed in section 8.2.3.

- Privacy - Ballot privacy is only maintained if the individual user chooses not to reveal their blockchain account address.
- Free Will - Linked to ballot privacy voters are able to cast their votes of their own free will but there is the potential for coercion.
- Reliability - Votes are transmitted and counted as intended.
- Prevention of Ballot Fraud - Ballots are protected from fraud, only registered users are able to vote.
- Prevention of Ballot Trade - This is linked to free will and privacy in which there is the potential for coercion.
- Accessibility - Due to the limited scope of this project no specific policies have been used to increase accessibility, the system is fully accessible to anyone with regular computer access.
- Affordability - Affordability is still mostly a theoretical, some evaluation of cost of Ethereum transactions is covered in section 8.3.
- Simplicity and Usability - The process of casting a vote is simple and easily understood, the user only has to log into the system and select a candidate option and submit their vote on the vote page.

The current protocol manages to accomplish four of the eight requirements laid out by Haas, namely reliability, ballot fraud, simplicity and usability, and accessibility. The accessibility requirement faces some limitations, this is mainly due to the small nature of this project and dedicated accessibility options for less able users could be implemented. The affordability requirement is still a theoretical concept, with potential server hosting and maintenance fees not considered.

The protocol as it is currently implemented faces problems on three of the requirements, privacy, free will, and prevention of ballot trade. They all stem from the same root problem, the fact that a user can choose to disclose their blockchain account number and therefore reveal their ballot. This will be further discussed in the limitations and extensions portion of this chapter.

The protocol is both personally and universally verifiable. Individual users are able to confirm that their account number is in the blockchain and that their correct vote has been cast. Outside entities are able to verify the election by checking the transaction log of the blockchain to confirm that a correct number of transactions have been cast for each candidate when compared to the tally kept by the candidate data structures themselves.

8.2 Implementation Evaluation

8.2.1 System Analysis

The key points of the system are outlined below:

- Heroku Deployment - The front-end code has been deployed to Heroku, allowing online use of the website.
- RESTful Architecture - This lets the protocol to make 'POST' and 'GET' requests, allowing the system to interact with datastores, and navigate through various web pages.
- PostgreSQL Database - Data is stored, read and written securely to a Postgres database.

- Ethereum Blockchain - The smart contract was developed to be deployed on the Ethereum blockchain network.

These features allow for the system to correctly perform its required functionality. This involves multiple user accounts, a fluid website, allowing for the casting and reading of votes on the blockchain.

Due to checks at each route's 'GET' and 'POST' requests users cannot access a page to which they do not have authorization, this means that users are not able to double vote, view results early, or access pages without logging in. If the user attempts to access a page which they are not authorised to view they will be redirected back to a different page.

Due to the local network testing method of the Ethereum smart contract, we are able to execute transactions and mine blocks instantly, this means there is no delay between a transaction being posted and the resulting block appearing on the blockchain network.

One potential area of issue within the system is the user's ability to view the results page as soon as they have cast their vote. This could lead voters towards a potentially biased vote, if via leaked results they think their candidate might not win. A potential avenue to stop this is to not reveal the final tally of results until all votes have been cast and the election is closed. This has not been implemented in this system as it was not deemed necessary for such a small trial system.

8.2.2 Error Testing

The software has been put through testing during and after the development phase to ensure that any functionality offered to users is not error prone. Areas of error testing include, but are not limited to, the below:

- Entering correct and incorrect website login details
- Entering correct and incorrect registration details.
- Accessing locked pages, such as the account page, when not logged in.
- Accessing locked pages, such as the vote and results, page when not authorised.
- The contract correctly initialising.
- Voting on the contract, such as duplicated votes or an incorrect candidate option.

8.2.3 Limitations & Extensions

Outlined below are some of the limitations and possible extensions of the system.

The software lacks https due to its testing on a local machine, the Heroku implementation of the software can be https enable however, there is a monthly monetary cost. It is assumed that this project is unlikely to be the target of man-in-the-middle attacks. Note that this lack of https can cause issues when running the Heroku deployment on Google's Chrome browser.

Voters could register twice for the system as there is currently no method of ensuring voters are specific individuals. This is considered outside the scope of this project due to its small scale nature, for a real world voting system a secure and trusted voter roll and registration system could be developed as an extension.

Privacy is a key concern of any voting system, along with coercion resistance and ballot trade. Currently voter privacy is only ensured when a user chooses not to reveal their blockchain account address. Coercion and ballot trade could both occur when a voter reveals their Ethereum account address to an outside party. One avenue to combat this would be to assign each individual voter an account at random on registration, keeping this account address securely hashed in the Postgres database and not revealing the address to the voter. This would allow the individual voter to vote, but would prevent them revealing their account address and therefore a receipt for their vote. This would in turn impact the individual verifiablity of the election, to combat this a function could be developed to search transaction logs for the user's account address and a confirmation of their vote, after the election has ended.

The main limitation of the system is due to its Ethereum blockchain nature, the system was not deployed to the main Ethereum network due to the monetary cost involved. Deploying the contract would have required an investment in Ether, as well as requiring Ether for any potential voters. As such the blockchain portion of the system is only implemented on a local Ethereum network for local machine testing, this requires a few tools to function properly. Truffle and Ganache as outline in the implementation section, for the deployment and monitoring of local blockchains. The MetaMask web extension is also required in order to inject accounts into the smart contracts web interface. As such setting up the process on a local machine is a lengthy and intricate process and hosting an election requires a large quantity of work with external tools. A major extension to this project would be to abstract all blockchain functionality, including transaction costs and account details, away from the voter. Enabling them to simply log in and vote when an election is running.

8.3 Ethereum Costs

Ethereum converts each computation action into a fixed gas cost, the total gas costs for each of the segments of an election are outlined below:

- Election Setup & Contract Creation:
  - Contract Creation Call - 268,471 Gas
  - Candidate 1 Initialisation - 41,981 Gas
  - Contract Creation Call - 537,857 Gas
  - Candidate 2 Initialisation - 26,981 Gas
- Voting:
  - First Transaction for Candidate 1 - 64,074 Gas
  - First Transaction for Candidate 2 - 64,074 Gas
  - Further Transactions for Candidate 1 - 49,074 Gas
  - Further Transactions for Candidate 1 - 49,074 Gas

Gas values are fixed and relate to the amount of computational work that is required.

The values listed above are the fixed gas costs for the computation work done by each transaction. Every operation that can be executed by a contract on the Ethereum platform costs a certain amount of gas. You cannot actually own gas, gas only exists within the Ethereum Virtual Machine, instead users are charged for the amount of work in Ether, Ethereum's built in currency, resulting in a gas price.

In order to determine the cost of a transaction the total gas costs are multiplied by the current gas price in order to produce the cost in Ether. The current recommended gas price for a block to be mined on the main Ethereum network is 2 gwei or 0.000000002 Ether, as of April 19th[38]. The current costs of each transaction, as of April 19th[38], in both Ether and GBP are outlined below:

- Election Setup & Contract Creation:
  - Contract Creation Call - 0.0005369 Ether - 0.21 GBP
  - Candidate 1 Initialisation - 0.000084 Ether - 0.03 GBP
  - Contract Creation Call - 0.0010757 Ether - 0.42 GBP
  - Candidate 2 Initialisation - 0.000054 Ether - 0.02 GBP
- Voting:

- First Transaction for Candidate 1 - 0.0001281 Ether - 0.05 GBP
- First Transaction for Candidate 2 - 0.0001281 Ether - 0.05 GBP
- Further Transactions for Candidate 1 - 0.0000981 Ether - 0.04 GBP
- Further Transactions for Candidate 1 - 0.0000981 Ether - 0.04 GBP

The election setup and contract creation calls are transactions called from the first account in the list, in practice the election administrator would control this account and assume these costs. In order to ensure that every voter can be able vote, and deal with any extreme fluctuation in gas price, every voter account should have Ether equal to at least three times the first transaction fee in their account.

This does not take into account the electricity costs of running the hardware required to mine blocks in a proof-of-work system. Alex de Vries estimates that as of April 19th[12] Ethereum's current annual electricity consumption is 17.6 terawatt hours or 17,600,000,000 kilowatt hours at an estimated cost of £1,505,110,138. The estimated cost per transaction is 75 kilowatt hours, by comparison an average house in the UK uses 10 kilowatt hours of electricty per day. This helps to illustrate the potential cost implications of a large scale proof-of-work blockchain network.

9 Summary

9.1 Project Management

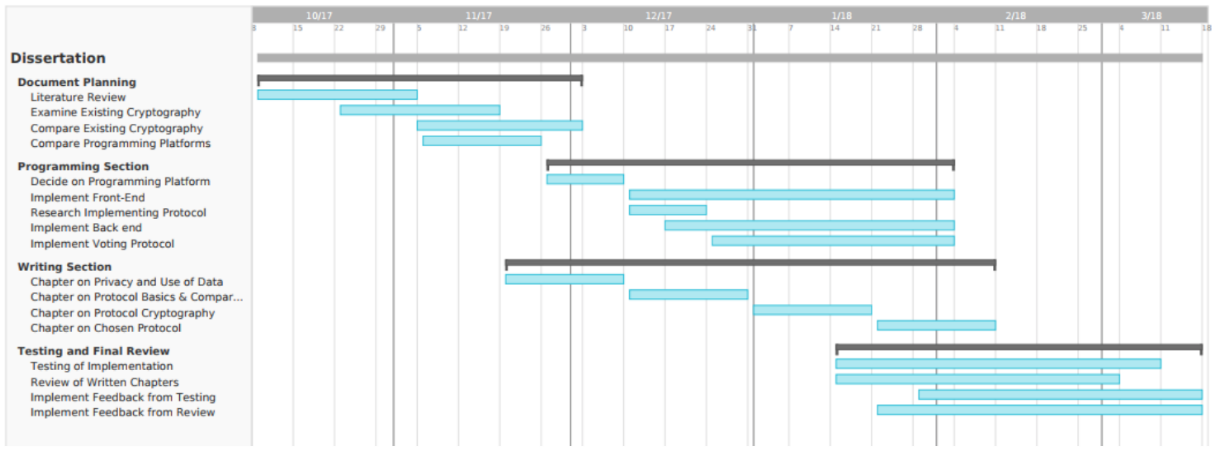


Figure 3: Work Plan

The Gantt chart shown above in Figure 3 is the initial timeline estimated for this dissertation. However, as this project has taken shape during the course of my work its scope has changed and therefore planned elements have been removed and new elements have been added instead. I stuck mostly to the initial timeline with some changes being made as the project continued.

From the document planning section of the Gantt chart above most of the stages have been completed as desired and they have been kept relatively within their timeframe. There was less of a key focus on cryptography as the project unfolded instead most of the document planning leant towards voting protocols and blockchain technology.

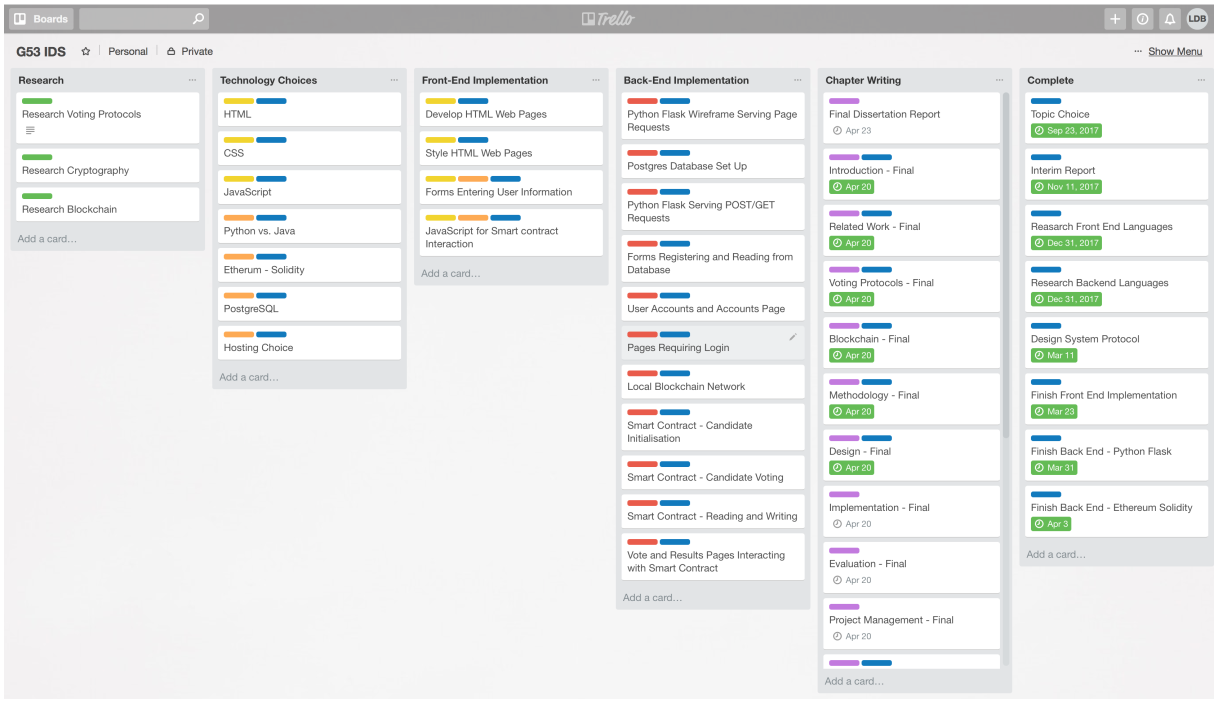
Since this project started I have learnt that literature review is an iterative and cyclical process that is more likely to go on for the entire project as new documents are discovered and read. This initial estimate for that part of the project of four weeks is therefore incorrect. The other sections of the document review including the examination and comparison of the existing protocols for electronic voting have stuck to the time schedule with some slight extension due to the vastness of the topic area.

Compared to the programming section of the Gantt chart above I was slightly delayed in determining which programming platform to use, this was due to the complexity of the project. As can be seen in the implementation section of this document, section 7, work was undertaken to decide on a programming platform though this took slightly longer than two weeks originally intended. Work done exploring an implementation of the protocol in terms of the entities and stages of the protocol also took longer than the expected two weeks. This research into the protocol's implementation ended up being a continuous and iterative process which was aided by the agile nature of the project. Overall the programming and implementation of the system took longer than was originally allotted.

Differing from the original proposal and Gantt chart's writing stage I have removed the chapter on the privacy of electronic voting. Instead adding a chapter on the blockchain covering its design, implementation, and usefulness for an electronic voting protocol. The chapter on cryptography of the various protocols was also removed and a condensed summary of cryptographic primitives was added to the chapter on voting protocols in section 3. The chapter on the chosen protocol was also removed and incorporated into the design section of this document, section 6.

With regard to the testing and final review of the document, testing of development and implementing feedback to this testing occurred alongside the programming and development of the system. Including automated unit testing and some physical integration testing. Unfortunately due to limitations with the system I was unable to perform full scale qualitative testing of the system with human participants, the initial proposal to host a full election cycle had to be dropped. Review and implementation of changes to written chapters has also been a continuously ongoing process throughout the writing of this document occurring alongside the writing of each chapter. As such the original estimates for this section of the process are incorrect.

The main use of the Gantt chart above was for the initial designation of project milestones. Trello[40], a project management system often used for agile development, was my choice for tracking tasks, milestones, and deadlines. This covered both the writing and research of the project as well as the software implementation. Trello and Git Combined to keep me on track with my revised timeline. My Trello board is included below for reference, Figure 4.



G53 IDS · Personal · Private · Show Menu

Research
- Research Voting Protocols
- Research Cryptography
- Research Blockchain
- Add a card…

Technology Choices
- HTML
- CSS
- JavaScript
- Python vs. Java
- Etherum - Solidity
- PostgreSQL
- Hosting Choice
- Add a card…

Front-End Implementation
- Develop HTML Web Pages
- Style HTML Web Pages
- Forms Entering User Information
- JavaScript for Smart contract Interaction
- Add a card…

Back-End Implementation
- Python Flask Wireframe Serving Page Requests
- Postgres Database Set Up
- Python Flask Serving POST/GET Requests
- Forms Registering and Reading from Database
- User Accounts and Accounts Page
- Pages Requiring Login
- Local Blockchain Network
- Smart Contract - Candidate Initialisation
- Smart Contract - Candidate Voting
- Smart Contract - Reading and Writing
- Vote and Results Pages Interacting with Smart Contract
- Add a card…

Chapter Writing
- Final Dissertation Report — Apr 23
- Introduction - Final — Apr 20
- Related Work - Final — Apr 20
- Voting Protocols - Final — Apr 20
- Blockchain - Final — Apr 20
- Methodology - Final — Apr 20
- Design - Final — Apr 20
- Implementation - Final — Apr 20
- Evaluation - Final — Apr 20
- Project Management - Final — Apr 20
- Add a card…

Complete
- Topic Choice — Sep 23, 2017
- Interim Report — Nov 11, 2017
- Reasarch Front End Languages — Dec 31, 2017
- Research Backend Languages — Dec 31, 2017
- Design System Protocol — Mar 11
- Finish Front End Implementation — Mar 23
- Finish Back End - Python Flask — Mar 31
- Finish Back End - Ethereum Solidity — Apr 3
- Add a card…

Figure 4: Trello Board

9.2 Reflections

As with any project, scope and time constraints limit the amount of work achieved. This has been the case for some sections of my work, mainly in the implementation of the blockchain portion of the system. Such a system could easily be adapted to work on the full Ethereum network if users were provided with subsidised Ether wallets. Some future work could be done on the accessibility of the system for those less able. Work could also take place on exploring voter opinions to online voting and methods of educating potential voters to the technology at work.

Some limitations to the system were also due to constraints around how the test network interacts with accounts and therefore the system can be prohibitively hard to deploy for local testing. These all leave plenty of room for future work, perhaps the truest full extension of this project would be a custom defined and built blockchain wherein its only token is designed for voting. As opposed to co-opting the Ethereum smart contract system to build a custom token.

Overall I feel the project ran smoothly, some aspects I might have approached differently if I was starting over. The project area is electronic voting which personally I feel is a very interesting area of study, a nice confluence of computing theory and technology and useful software. As I found the project interesting I was able to meet the aims and objectives of the project whilst remaining engaged, excited, and enthusiastic throughout. The combination of Trello and GitHub helped keep me, mostly, inline with my original project timeline and enabled the easy tracking of project milestones. I feel my chosen programming languages for this project were both reasonable, and perhaps the best choices. Python especially was a very good language choice, I had less of a personal choice for the other languages such as SQL, HTML, CSS, and Solidity as they are the standards for a project like this.

In conclusion I believe that my system has achieved its intended aims and goals as originally proposed. From the practical perspective users can securely register with the system and can also login to an account. Users can also cast a vote in and view the results of an election. I was very excited by the Heroku deployment, in particular being able to navigate to a web address and see my code running properly brings with it a real sense of pride. I was disheartened to note the that smart contract deployment to Heroku was currently out of bounds. In terms of research and theoretical study I believe this project achieved its aims. I undertook an in-depth analysis of cryptography, voting protocols, and the blockchain in order to produce this project. As a result of my research and undertaking this project I believe that electronic voting provides numerous benefits and has a real opportunity to impact people all over the world. I do, however, believe that the technology is not currently in place to achieve this and that my personal implementation falls short of the requirements of a full voting system. I feel that a proof-of-scale blockchain network developed solely for voting purposes is the best choice for online elections that aim to use blockchain technology. I look forward to seeing what further research and experimentation in this space results in.

Bibliography

[1] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[2] Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'09, pages 10–10, Berkeley, CA, USA, 2009. USENIX Association.

[3] bitinfocharts. *Bitcoin, Ethereum Average Transaction Fee.* https://bitinfocharts.com/comparison/transac tionfees-btc-eth.html, Accessed 08/12/17.

[4] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. *Public Key Cryptography – PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy.*, pages 403–422, March 6-9, 2011.

[5] Daniel Bochsler. *Can Internet Voting Increase Political Participation?* Remote Electronic, 2010.

[6] US Government Census. *Table 397. Participation in Elections for President and U.S. Representatives: 1932 to 2010.* Census.gov, https://www.census.gov/prod/2011pubs/12statab/election.pdf, 2011.

[7] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1614–1625, New York, NY, USA, 2016. ACM.

[8] Electoral Commission. *Electoral Pilot Scheme Technical Evaluation.* http://www.electoralcommission.org.uk/__d ata/assets/electoral_commission_pdf_file/0017/16190/SouthBucksFinal_27220-20135__E__N__S__W__.pdf, Accessed 31/10/17.

[9] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In *19th European Symposium on Research in Computer Security - Volume 8713*, ESORICS 2014, pages 327–344, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[10] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 72–83, Berlin, Heidelberg, 1996. Springer-Verlag.

[11] Matthew Czarnek. *Nxt Network: Energy and Cost Efficiency Analysis.* https://www.scribd.com/document/254930279/Nxt-Network-Energy-and-Cost-Efficiency-Analysis, Accessed 20/03/18.

[12] Alex de Vries. *Ethereum Energy Consumption Index.* https://digiconomist.net/ethereum-energy-consumption, Accessed 19/04/18.

[13] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[14] Sarah P. Everett, Kristen K. Greene, Michael D. Byrne, Dan S. Wallach, Kyle Derr, Daniel Sandler, and Ted Torous. Electronic voting machines versus traditional methods: Improved preference, similar performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 883–892, New York, NY, USA, 2008. ACM.

[15] Ethereum Foundation. *Ethereum Project.* https://www.ethereum.org/, Accessed 05/12/17.

[16] Ethereum Foundation. *Ethereum Virtual Machine Operation Costs.* https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs/editgid=0, Accessed 20/03/18.

[17] Bertrand Haas. Engineering better voting systems. In *Proceedings of the 2006 ACM Symposium on Document Engineering*, DocEng '06, pages 56–58, New York, NY, USA, 2006. ACM.

[18] Salesforce Heroku. *Heroku.* https://www.heroku.com/home, Accessed 05/12/17.

[19] Stan Higgins. *Abu Dhabi Stock Exchange Launches Blockchain Voting Service.* Coindesk, https://www.coindesk.com/abu-dhabi-exchange-blockchain-voting/, October 21, 2016, Accessed 21/11/17.

[20] Sarah Hyde and Ruth Abbey. No country for older people? age and the digital divide. *Journal of Information, Communication and Ethics in Society*, 7(4):224–242, 2009.

[21] Simona Ibba, Andrea Pinna, Matteo Seu, and Filippo Eros Pani. Citysense: Blockchain-oriented smart cities. In *Proceedings of the XP2017 Scientific Workshops*, XP '17, pages 12:1–12:5, New York, NY, USA, 2017. ACM.

[22] UK Political Info. *General election turnout 1945 – 2017.* UK Political Info, http://www.ukpolitical.info/Turnout45.htm, Accessed 29/11/17.

[23] National Democratic Institute. *Re-evaluation of the Use of Electronic Voting in the Netherlands.* https://www.ndi.org/e-voting-guide/examples/re-evaluation-of-e-voting-netherlands, 2006, Accessed 31/10/17.

[24] National Democratic Institute. *Evaluation of E-Voting in Norway.* https://www.ndi.org/e-voting-guide/examples/evaluation-of-e-voting-norway, 2011, Accessed 31/10/17.

[25] Jackie B. Koven. *Block The Vote: Could Blockchain Technology Cybersecure Elections?* Forbes, https://www.forbes.com/sites/realspin/2016/08/30/block-the-vote-could-blockchain-technology-cybersecure-elections/3747a5362ab3, August 30, 2016, Accessed 20/11/17.

[26] Dan Lohrmann. *Can Blockchain Technology Secure Your Vote?* Govtech, http://www.govtech.com/blogs/lohrmann-on-cybersecurity/can-blockchain-technology-secure-your-vote.html, April 29, 2017, Accessed 20/11/17.

[27] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. *A Smart Contract for Boardroom Voting with Maximum Voter Privacy.* 2016.

[28] Jonathan Mellon, Tiago Peixoto, and Fredrik M. Sjoberg. Does online voting change the outcome? evidence from a multi-mode public policy referendum. *Electoral Studies*, 47(Supplement C):13 – 24, 2017.

[29] Andrew C. Myers, Michael Clarkson, and Stephen Chong. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368. IEEE, May 2008.

[30] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* https://bitcoin.org/bitcoin.pdf, Accessed 07/12/17.

[31] Parliamentary Office of Science & Technology. *Trends in Political Participation.* Houses of Parliament, 2015.

[32] Ronald L. Rivest, Aad Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[33] Armin Ronacher. *Flask Microframework.* http://flask.pocoo.org, Accessed 02/12/17.

[34] MetaMask Open Source. *MetaMask.* https://metamask.io/, Accessed 05/12/17.

[35] Solidity Open Source. *Solidity Documentation.* https://solidity.readthedocs.io/en/v0.4.22/, Accessed 05/12/17.

[36] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 703–715, New York, NY, USA, 2014. ACM.

[37] Reuters Staff. *France drops electronic voting for citizens abroad over cybersecurity fears.* Thomson Reuters, http://uk.reuters.com/article/uk-france-election-cyber/france-drops-electronic-voting-for-citizens-abroad-over-cybersecurity-fears-idUKKBN16D235, March 6, 2017, Accessed 31/10/17.

[38] Ethereum Gas Station. *Ethereum Gas Station.* https://ethgasstation.info/, Accessed 19/04/18.

[39] Bootstrap Core Team. *Bootstrap Web Framework.* https://getbootstrap.com, Accessed 01/12/17.

[40] Atlassian Trello. *Trello.* https://trello.com/about, Accessed 03/10/17.

[41] Truffle. *Truffle Framework.* http://truffleframework.com/, Accessed 01/11/18.

[42] Truffle. *Ganache.* http://truffleframework.com/ganache/, Accessed 05/12/17.

[43] Laura Vozzella. *Virginia scraps touch-screen voting machines as election for governor looms.* Washington Post, https://www.washingtonpost.com/local/virginia-politics/virginia-scraps-touch-screen-voting-machines-as-election-for-governor-looms/2017/09/08/e266ead6-94fe-11e7-89fa-bb822a46da5b_story.html, September 8, 2017, Accessed 25/10/17.