Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

An End-to-End Verifiable Secure Online Voting System

Author:
Sander Sebastian
Henschien Coates

Supervisor:
Prof. William J. Knottenbelt
Co-Supervisor:
Clarissa Agnew

Summer 2020

Abstract

Online voting systems have recently received remarkable attention, with the incentives for improved remote voting ranging from increased flexibility and participation, to administrative cost savings, to robustness against pandemics like the most recent COVID-19 pandemic. However, the systems proposed so far have not earned sufficient public confidence. In particular, expectations of end-to-end verifiability, balanced with requirements of voter anonymity and usability, make for challenges not yet overcome.

We propose a new proof of concept, end-to-end verifiable, secure online voting protocol based on code voting that sets itself apart from existing systems by a new distribution of election authority trust. Compared to the inspirational work of Ryan and Teague [1] and Chondros et al. [2], we offer improved resilience to attacks on election authority integrity and a more voter-friendly voting process.

Through security analysis, the protocol is found to offer strong security against client side malware, and an intermediary has no opportunity to infer, alter or drop votes without the voter noticing. Similarly, misbehaving election authority agents are reported by others. To mitigate the risk of a compromised election authority agent, we suggest the implementation of distributed election authority agents by use of distributed key generation and threshold cryptography.

A demonstration of the protocol in the shape of online voting system simulation gives insight about protocol performance. Using conventional cryptographic primitives, several hundred votes are processed by a single processor every second. This rate of vote processing is constant across population sizes, and does not suffer considerably from strengthened symmetric encryption, nor the number of candidates or presence of adversaries.

The protocol does well with respect to all requirements for online voting systems but coercion resistance, which is arguably a particular intricate issue for remote systems. End-to-end verifiability and software independence is assured under the assumptions of the threat model. So are secret ballots and receipt freedom. We argue that the protocol is superior in terms of usability.

Together, these achievements form the foundation for the further development and deployment of a new online voting system to be used in democratic elections. We suggest several extensions to the protocol and outline areas of future work.

Acknowledgements

I would like to express my sincere gratitude towards my supervisor, Professor William J Knottenbelt, and co-supervisor, Clarissa Agnew, for their guidance and support during the work with this project. I would also like to extend my thanks to Dr Dalal Alrajeh and Dr Liliana Pasquale who have both shared generously their time and knowledge to guide me in the project. Finally, I would not have had the pleasure of studying at Imperial College if it had not been for Aker Scholarship and I am forever grateful for the support the scholarship has provided.

Contents

List of Tables

List of Figures

List of Acronyms

`bbpk` bulletin board public key

`bbsk` bulletin board private key

`cpk` collector public key

`csk` collector private key

`inpk` initiator public key

`insk` initiator private key

`isk` inner vote code key

`osk` outer vote code key

`rsk` receipt code key

`tpk` tallier public key

`tsk` tallier private key

AES Advanced Encryption Standard

ASCII American Standard Code for Information Interchange

CBC Cipher Blocker Chaining

DKG distributed key generation

DoS denial of service

E2E end-to-end

IV initialisation vector

LSEPI legal, social, ethical and professional issues

MITM man-in-the-middle

PGD Pretty Good Democracy

RSA Rivest-Shamir-Adleman

SHA Secure Hashing Algorithm

UML Unified Modelling Language

ZKP zero-knowledge proof

Chapter 1

Introduction

1.1 Motivation

Democracy is a more than two millenniums old concept, dating back to the Greek transformation from *rule by few* to *rule by many* [7]. Since then, the democratic and election processes have changed considerably. In the last century, electronic devices were introduced to aid the registering and counting of votes, but sparked debate about the security of such systems compared to traditional manual systems after several controversies [8]. This has led to the development of a vast collection of electronic voting systems, each with its improved security features and limitations [9–11].

Figure 1.1: News headline by Schneier in The Guardian [3].

Figure 1.2: News headline published by Corse in The Wall Street Journal [4].

Electronic voting systems may be categorised as *pollsite* or *remote* [9, 12]. As their names suggest, pollsite systems entail vote casting in specified locations whereas

remote voting is done remotely. This categorisation is useful in defining the focus of this project on remote voting. The incentives for improved remote voting are many [11]. One motivating factor is that voters may be unable to attend polling locations, for instance due to remote residency or disabilities. Another issue with pollsite voting is the economic cost of administering polling stations. The most recent COVID-19 pandemic, with severe restrictions on social interaction, too highlights the value a remote alternative to elections could serve. However, remote voting presents some challenges that are not as easily dealt with as with pollsite voting, such as avoiding coercion. Figures 1.1 to 1.4 show a sample of recent news headlines that underline the continued public distrust in remote voting systems and need for better solutions.

Figure 1.3: News headline by Newman in Wired [5].

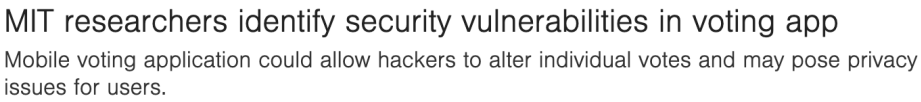


Figure 1.4: News headline published by Abazorius on MIT News [6].

1.2 Objectives

Based on the recent surge in interest for remote voting and the persisting search for a sufficiently secure system, this project is focused on remote voting. More specifically, we address *online voting*, to make a distinction from the predominant alternative remote voting that is postal mail voting. Readers interested in pollsite voting or remote postal mail voting are recommended to read the reviews by Bernhard et al. [12] and Ali and Murray [9], as well as individual system descriptions [13–22].

The objectives of this project are:

1. Design a new proof of concept secure online voting protocol that is practically feasible in terms of processing time and usability
2. Analyse the security of the protocol and propose measures to mitigate any potential threat

3. Demonstrate the protocol as implemented in a voting system and evaluate its practical feasibility both qualitatively and quantitatively

1.3 Contributions

The contribution of this project is two fold. First, a new proof of concept secure online voting protocol is designed and analysed with regards to security. Second, a demonstration of the protocol in the shape of a online voting system simulation is presented and evaluated. Together, the two contributions form the foundation for the further development and deployment of a new online voting system to be used in democratic elections.

1.3.1 A new end-to-end verifiable secure online protocol

The protocol we propose makes use of *code voting* and is inspired by the work of Ryan and Teague [1] and Chondros et al. [2]. In code voting schemes, the voter communicates her choice to the election authority by submitting a cryptographic code, rather than the plaintext choice. Compared to the protocol of Ryan and Teague, we offer improved resilience to attacks on election authority integrity by an original distribution of election authority responsibility and trust. We simplify the voting process for the voter, compared to the work of Chondros et al., by simpler ballots.

In the proposed protocol, voters receive unique, cryptographic ballots prior to an election and vote by submitting two codes: a credential and a vote code. They may verify that their vote is registered as cast by comparing a returned receipt code with one printed on their ballot. After polls close, any voter may verify that votes are counted correctly via a public record of all anonymous votes, thereby verifying votes are counted as registered.

We conduct a security analysis of the proposed protocol and suggest mitigating actions for potential threats. The integrity of the protocol relies on a certain degree of trust in the election authority. We suggest measures to reduce this reliance, should it be required.

1.3.2 Demonstration of an end-to-end verifiable secure online system

The protocol is implemented as part of a voting system simulator and its usability and performance is evaluated. Using conventional key lengths, as of the time of writing, several hundred votes are processed by a single processor every second. This rate of vote processing is found to be constant across population sizes, and furthermore does not depend significantly on the security parameters for symmetric encryption, nor the number of candidates or presence of adversaries.

We assess the trade-offs that exist between usability and security and evaluate the system's satisfaction of the requirements typically set for online voting systems. The protocol does well with respect to all requirements but coercion resistance, which is arguably an particularly intricate matter for remote voting. To this we suggest a rather simple technical modification. In terms of usability, we argue that the protocol is superior and only requires the voter to handle three 10 familiar characters long codes for vote submission and verification. We have also argued that the protocol may likely be extended to allow accommodation to any accessibility needs.

1.4 Report outline

This report is structured into seven chapters.

Chapter 2 introduces the technical preliminaries and discusses the requirements of online voting systems before related work and common threats to online voting systems are considered.

Chapter 3 describes the design of the proposed protocol. The threat model assumptions are also introduced to situate the protocol in its environment.

Chapter 4 builds on the threat model assumptions from Chapter 3 and analyses the security of the protocol. Appended to the security analysis are suggestions for mitigating actions to address the threats identified.

Chapter 5 discusses the implementation of a online voting system simulator using the protocol.

Chapter 6 evaluates the performance of the protocol based on simulations as described in Chapter 5.

Chapter 7 ultimately presents concluding remarks and suggests further work.

1.5 Legal, social, ethical and professional issues

Before proceeding, it is worthwhile to discuss the legal, social, ethical and professional issues (LSEPI) related to this project. There is no involvement of humans or animals, personal data, environmentally harmful materials, nor direct potential for dual or misuse of this project, and so most checkpoints of the LSEPI checklist appended in Appendix A have "No" ticked. Future extensions based on this project may, however, have substantial legal, social, ethical and professional implications. First and foremost, a full scale implementation of the protocol for use in democratic elections will inevitably involve the handling of most private human personal data and the processing of votes whose tally may determine the outcome of an election. This introduces the risk of misuse by nation state adversaries and criminal and terrorist organisations. Lastly, such an extended implementation will find itself in a complex legal landscape where the fundamental requirements of democracy must be satisfied. Although this project has an exclusive civilian application focus without direct contact with LSEPI, the proposed protocol has been designed with potential future implications in mind. The protocol is described in full to encourage public openness and scrutiny, ultimately to earn trust as a secure online voting system.

Chapter 2

Background

In this chapter, we first introduce technical preliminaries before discussing the requirements for online voting systems, related work and finally common threats to online voting systems with a note on quantum computing.

2.1 Technical preliminaries

This section is intended to provide readers that are not too familiar with the technologies related to computer security and online voting that are discussed in subsequent sections and chapters.

Vulnerabilities, exploits and attacks A *vulnerability* is an information system weakness that could could be *exploited* by a threat. An *attack* is any malicious attempt to interfere with an information system [23].

Threat modelling In threat modelling, the threats and mitigating actions to protect a system situated in a specified environment are identified and analysed[1]. A threat model typically includes a description of the system of interest, a presentation of assumption about the system and its environment, a discussion of threats and mitigating actions and a method for validating the model.

Malware *Malware* is any malicious hard-, firm- or software [23].

[1] `https://owasp.org/www-community/Application_Threat_Modeling` [cited 2020 Aug 10]

Pseudo-randomness Pseudo-randomness is the theory of generating seemingly random objects (e.g. numbers) using little or no randomness [24]. Pseudo-random generators are algorithms that output longer sequence of seemingly random bits based on a shorter perfectly random *seed*, a sequence of bits.

Cryptographic primitives The basic cryptographic tools, such as hashing, encryption and digital signature schemes (discussed below), used to provide security are called *cryptographic primitives* [25].

Hashing and SHA A *hash function* takes as input (or *message*) a string of bits and returns as output (or *digest*) a string of bits of fixed length. It is further required that it be computationally infeasible to (i) find for a specific digest the message that hashed to that digest and (ii) find for some specified message a different message that hashes to the same digest [23]. The *secure hashing algorithms* (SHA) is a family of standard hash functions, such as SHA-2.

Symmetric encryption and AES By symmetric encryption, a secret key is used for both encrypting plaintext to ciphertext and decrypting ciphertext to plaintext. The Advanced Encryption Standard (AES) is a symmetric block cipher which encrypts or decrypts blocks of binary data [23].

Asymmetric encryption and RSA In contrast to symmetric encryption, asymmetric encryption makes use of a key pair, one key for encryption and one key for decryption. This allows publicly sharing a public key while holding a private key secret[2]. RSA [26] is a common asymmetric encryption scheme based on the difficulty of factoring the product of two large primes.

Digital signatures A digital signature allows an agent to *sign* a message using a secret private key. The signature may later be verified for the specific message using the signer's public key [25].

Blockchain A blockchain is a ledger of cryptographically signed transactions, implemented in a distributed fashion [23]. Blockchains received considerable attention

[2] `https://support.microsoft.com/en-us/help/246071/description-of-symmetric-and-asymmetric-encryption` [cited 2020 Jun 4]

after Nakamoto described *Bitcoin*[3] [27].

Homomorphic encryption Homomorphic encryption allows computation on encrypted input such that the decrypted output matches the results of the same computation applied to decrypted inputs [9].

Threshold encryption In threshold encryption, the encryption/decryption key is shared among several agents such that a message can only be encrypted/decrypted once a threshold share of the agents cooperate [28, 29].

Mix networks A mix network hides the association between items in its inputs and items in its output [30]. It consists of sequential mixes, resulting in cascading mixing [9].

Base64 Base64 is a binary to text encoding scheme that translates binary data to an ASCII string of characters, each character selected from a collection of 64 characters. Each character represents six bits of information. The encoding scheme is typically used to transfer binary data in textual form[4].

2.2 Requirements for online voting systems

In the following, the requirements of a secure online voting system are discussed. The discussion is based on a survey of common requirements presented in literature [9, 11, 12]. Most requirements follow directly from the requirements for traditional voting systems, such as *secret ballots* and *receipt freedom*, while some are extensions, such as *end to end verifiability*. Note that some requirements are inevitably in conflict, such as *verifiability* (allowing a voter to verify her vote was counted in the final tally) and *receipt freedom* (not providing the voter with any means of proving her vote) [9, 12, 31, 32]. The resulting trade offs call for elegant technical solutions [12].

A list of all requirements is given below.

- E2E verifiability
- Voter authentication and authorisation

[3] `https://bitcoin.org/en/` [cited 2020 Aug 10]

[4] `https://developer.mozilla.org/en-US/docs/Glossary/Base64` [cited 2020 Aug 10]

- Receipt freedom
- Secret ballot
- Usability and accessibility
- Software independence
- Transparency
- Coercion resistance

End-to-end (E2E) verifiability The notion of E2E verifiability has received notable attention in recent publications [9, 11, 12, 33, 34]. An E2E verifiable voting system allows verification of three critical steps: that votes are *cast as intended*, *recorded as cast* and *tallied as recorded*. The three step verification ensures votes are not manipulated over their lifetime as what is ultimately tallied reflects the voter's intention. *Cast as intended* and *recorded as cast* are typically verified by the voter herself, whereas *tallied as recorded* may also be verified universally by a third party [9, 12, 34].

With E2E verifiability comes the risk of voters claiming that their votes were not tallied correctly. In such a case, it is desirable that the system has some mechanism for dispute resolution [11, 12].

It is worth contrasting the requirement of E2E verifiability for online voting systems with the lack of such a requirement for traditional pollsite voting. Whether posted by mail or submitted in person, most voters have no way of verifying that their votes are not manipulated on their way to the final tally (or even dropped all together). E2E verifiability is of elevated importance to online voting systems due to the additional threats faced. Such threats include malware in the voter's system, compromised election authority systems and *man in the middle* (MITM) attacks manipulating or dropping votes. A discussion of threats is provided in Section 2.4. Bernhard et al. require, in addition, that the collection of voters that cast votes should be verifiable [12]. This ties to the next requirement of voter authorisation and authentication.

Voter authorisation and authentication Only voters that are eligible to vote can be authorised to vote [11, 35, 36]. This implies that valid ballots should only be issued to authorised voters. Only votes from authenticated users can count towards the election outcome. Related to voter authorisation and authentication is

the requirement that any voter may at most have one vote counted. Several techniques have been proposed for voter authorisation and authentication, including postal mailed codes and bio-metrics such as iris scans [35].

Receipt freedom *Receipt freedom* requires that the voter obtains no proof of how she voted, an is critical to voter anonymity. This is fundamental to traditional voting as it prevents, to certain extent, coercion and vote trading [11, 12]. The requirement of receipt freedom does not prevent the issuing of proof that the voter did vote, but cannot prove how she voted. Receipt freedom must be asserted even if the voter colludes with a coercer in attempt to produce proof of a particular vote [12]. Ali and Murray discuss how early research systems had an explicit voter-vote linkage, in contradiction with receipt freedom [9].

Secret ballot [9, 12, 33, 36] all emphasise the importance of secret ballots. Like receipt freedom, secret ballots is critical to voter anonymity and prevents third parties, or even the election authority, from observing votes and influencing voting via coercion or vote purchase.

Usability and accessibility Usability and accessibility is paramount to any voting system, including online voting systems [11, 12, 33]. Any voting system must allow all eligible voters to vote. In addition to setting expectations for the support of hardware and software available to all voters, usability also affects the tolerated complexity of the voting process, including verification [9, 11, 36]. There is a risk of false trust by which system security relies on voters verifying their votes, thus reducing the effective security when they do not do so [9, 12]. Usability also sets certain limitations on the computational and spacial complexity allowed for a voting system. What is meant here is that the time and space needed to process votes must not be of such magnitude that it prevents usability. Long computation time may especially be an issue for verifiability [35].

Software independence A voting system is *software independent* if a undetected software change or error cannot cause an undetectable election outcome change or error [37]. Rivest and Bernhard et al. require that any voting system must be software independent, given the difficulty of proving software correctness and evolving

threats [12, 37]. Bernhard et al. argue that an E2E verifiable system achieves software independence [12].

In addition to the features of *software independence*, *strong software independence* also includes the ability to correct a detected change or error without needing to rerun the election [37]. It is desired that votes are not lost or compromised in the event of system malfunction or attack [12, 36].

Transparency Central to an online voting system's success in competing with traditional systems is trust, as perceived by voters, election authorities and legislators [11, 33]. To obtain that trust, observers must be given evidence that the system functions as intended and details about how it achieves that [11]. Several commercial systems are criticised heavily on the basis of lacking transparency [33, 38].

Advocates and critics of online voting have raised interest in open source voting systems, and some go further to demand it [6, 11]. Although open source code invites researchers and hackers to find vulnerabilities, there is no guarantee that identified vulnerabilities are reported and fixed before they may be exploited. As such, the requirement of transparent source code as an extension of a transparent protocol is an intricate issue.

Coercion resistance In contrast to the often supervised setting of pollsite voting, unsupervised online voting opens new opportunities for coercion [11]. This is concern after ensuring receipt freedom and secret ballots too [9], for instance by the coercers forcing the voter to rank candidates in a specific order and identifying that same order in the anonymous tally. Measures should be in place to mitigate such risks. Clarkson et al. [34] present the notion of *coercion resistance* by which the voter cannot prove how she voted, even if she interacts with the coercer. This is similar to the discussion of Ali and Murray [9] and Bernhard et al. [12], where a system is deemed coercion resistant if a voter can cast a legitimate vote while appearing to a coercer to collaborate or the coercer cannot tell whether she followed orders or not. It is common to assume that the voter has some time when she can vote without being observed [12, 32].

2.3 Related work

In 1981, Chaum published the first paper mentioning the use of cryptography for secure remote voting [9]. After proposing a solution to the traffic analysis problem for secure, anonymous communication over an insecure medium between agents, Chaum suggested the protocol could be used for elections where ballots are signed using pseudonyms and mailed anonymously [30]. The following 20 years, most research did not produce any real-world systems. This changed with the entrance of this millennium with several voting systems presented. A selection of the most cited and recent online voting systems, both academic and commercial, are discussed in the following. Systems that intentionally ignore important requirements of voting systems and explicitly state an intention of not being used for nation scale democratic elections are generally left out. The same applies to systems heavily reliant on specific hardware, which can be argued to limit usability. Note that the details available about commercial systems are typically scarcer than for systems presented through academia.

2.3.1 RIES

The Rijnland Internet Election System (RIES) [39], published in 2005 and later used for expatriate voters in Dutch national elections [11], is allegedly the first voting system directly addressing E2E verifiability [35].

The system is comprised of a client side web application and server side tallying software. Prior to election, voters are sent unique authentication codes via postal mail. The authentication code is used together with hashed candidate ids before being sent to the election authority's server using encrypted communication. A table of pre-computed hashes corresponding to all possible voter-candidate votes are published prior to election, such that the voter can verify her vote was cast as valid. At voting close, the election authority checks for every received vote whether a hash of the vote coincides with any pre-computed hash and rejects votes for which there is no match. Validated votes are posted to a public table for post election verification by voters. RIES also allows for combination with postal votes by computing similar hashes of postal votes.

Dzieduszycka-Suinat et al. [11] find that RIES provides stronger verifiability than traditional postal voting, but that the verification procedure is complicated, thus not

fully accommodating to the requirements of a usable, E2E verifiable voting system. Both the system authors themselves and external critics [9, 11] find several weaknesses in terms of lacking security and usability. Ali and Murray claim that the public table of hashed votes and a voter disclosing her authentication code could prove her vote to a third party [9]. In the original paper, Hubbers et al. discuss this issue but offer no remedy [39]. This violates receipt freedom and secret ballots, and leads to significant risk of coercion and vote buying. Another limitation of the system is that if client side malware tampers with a voter's vote before sending it to the election authority server, even if the voter detects the tampered vote, he has no means of proving the vote was tampered with [39]. This constitutes a lack of dispute resolution. With regards to usability, the authentication codes posted in advance of elections could be lost, effectively preventing voters from voting [11].

Plans to use RIES in the 2008 Dutch parliamentary election were discontinued [11] and there has, to the best of our knowledge, not been any concrete plans of Dutch online voting.

2.3.2 Civitas

Civitas [34], published in 2008, is the first systems to deal with coercion in an elaborate way [9, 34]. The system is an extension of the Juels, Catalano and Jacobsson (JCJ) scheme, amongst other improvements offering more secure voter authorisation than its ancestor [9].

Civitas distributes the responsibility for authorisation and tallying processes by threshold encryption. Registered voters obtain their credentials from a combination of all registration tellers' shares. Combining the voter's credentials with an encrypted candidate choice, the voter submits her vote to one or several distributed, redundant ballot boxes. The key to coercion resistance emerges from the voters' ability to generate fake credentials, indistinguishable to an adversary, in lieu of their true credentials and use these to fool the adversary. Under the assumption that the adversary cannot observe the voter over the full duration of the election, the voter can at a different time without the adversary submit a legitimate vote. Upon retrieving all votes from the ballot boxes, the tabulation tellers discard duplicate votes according to some policy and ensure all remaining votes are well formed, still allowing fake credentials. Next, a mix network is used to anonymise a list of real credentials and the votes before removing all fake credentials. At last, the remaining votes are de-

crypted, ignoring the credentials, and a final tally is produced. Proofs are used for the tabulation tellers to make the tally universally verifiable. For individual verification, a voter can look for her vote in the table of votes produced by the tabulation tellers.

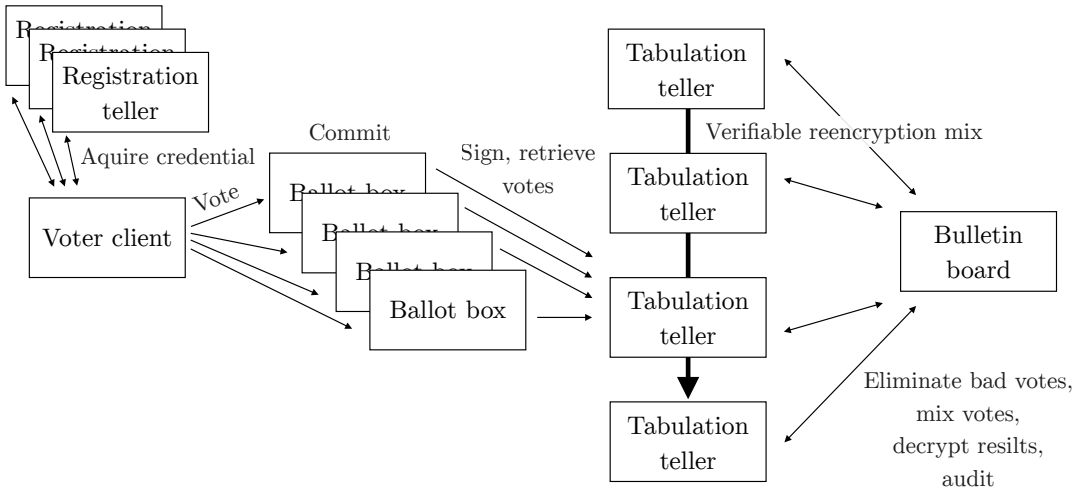


Figure 2.1: Civitas system architecture as illustrated by Clarkson et al. [34].

The main limitations of Civitas are related to the computational complexity of tabulation and removing invalid votes [9, 34]. The authors, Clarkson et al., set a requirement for tabulation to be completed in less than five hours and make the claim that this is achievable with the system. In addition to computational complexity, the system has not resolved the issue malware on the voter's system nor ensured it is sufficiently resilient to DoS attacks.

To the best of our knowledge, Civitas has not been tested in real-world elections.

2.3.3 Pretty Good Democracy

Ryan and Teague presented *Pretty Good Democracy (PGD)*, a code voting scheme, in 2009 [1]. By *code voting*, the voter uses codes from a code sheet to make a candidate choice. With the code sheet distributed prior to voting and relations between candidates and codes hard to guess, the idea is that any intermediary, including the device used to vote, cannot infer how the voter votes. To give verification that a vote code was received unmodified by a legitimate receiver, acknowledgement codes are typically printed on the code sheets to allow comparison to acknowledgement codes

returned to the voter after voting. The PGD ballot is illustrated in Figure 2.2. Critical to all code voting schemes is the distribution of code sheets without any information leak. Postal mail is often assumed a secure channel for distribution. In pursuit of stronger E2E verification, Ryan and Teague suggest the use of threshold encryption to produce legitimate acknowledgement codes to ensure the voter that a vote was not only cast as intended, but also recorded as cast. The authors also discuss measures to guarantee votes are tallied as recorded.

Candidate	Vote code
Asterix	3772
Idefix	4909
Obelix	9521
Panoramix	7387
Ack Code: 8243	
Ballot Id: 3884092844	

Figure 2.2: Pretty Good Democracy ballot as illustrated by Ryan and Teague [1].

Among the limitations of PGD is the fact that acknowledgement codes unique to each code sheet are posted on a public bulletin board. This opens the possibility of forced abstention attacks whereby an adversary with knowledge of a voter's code sheet acknowledgement code can see whether or not the voter has voted and ensure the voter does not vote. Another concern is that voters could simply sell their code sheet, thereby allowing a buyer to vote as he likes. Lastly, should voters wish to undermine the integrity of the voting system, they may submit invalid vote codes which could be misunderstood as the system trying to guess alternative vote codes. The authors ultimately argue that although PGD does not provide full E2E verifiability and is vulnerable to some common threats, it goes far in improving usability and that the scheme is appropriate for non-binding, non-political elections.

2.3.4 Selene

Ryan et al. presented *Selene* in 2016, a voting scheme using a different approach to simplify E2E verifiability [32]. By providing each voter with a private tracking number rather than an encrypted ballot receipt, voters may easily look for their tracking number and plain text choice on a public bulletin. To prevent coercion, tracking numbers are only given to voters after the bulletin board is published. Should a

voter be coerced, she may choose any tracking number from the bulletin board and claim to the coercer that this is hers and that she collaborated. The authors also discuss measures, in Selene II, to mitigate the risk of a voter's reported tracking number coinciding with the coercer's tracking number.

The scheme utilises mix-nets to anonymise votes as well as to ensure unique tracking numbers and accommodates to most requirements of a secure online voting system. With regards to dispute resolution, however, the authors state this is difficult with any tracking number approach and offer no immediate solution.

2.3.5 IVXV (Estonian I-Voting)

An overview of the *IVXV* system is given in [40]. The first Estonian I-voting[5] election was held in 2005.

The IVXV system is not concerned with voter registration and candidate tabulation, which are both handled by other Estonian systems. Voters download software from an official website and authentication is done by an ID card, mobile ID or a digital identity document. To enforce secret ballots and protect the integrity of votes, votes are encrypted with a public key issued by the election authority and signed digitally before submitting to the election authority. Votes are then validated and decrypted with the election authority's private key. The IVXV envelope scheme is displayed in Figure 2.3. Provably correct re-encryption mixing is performed to maintain anonymity. Upon receiving several votes from the same voter, only the last vote is counted. This is intended to mitigate coercion, by enabling the voter to vote again at a later stage if influenced by a coercer at an earlier time in the election. After voting, the voter receives a unique vote ID used to verify the vote was counted. To mitigate the risk of voter malware, verification is done via a separate device, such as a smartphone or tablet. The efficiency of the Estonian system relies heavily on the voters' trust in the election authority and accompanying Estonian digital identity documents.

Springall et al. find severe limitations of the IVXV system that pose a serious risk to the integrity of Estonian elections [31]. The authors observed the 2013 elections and report on numerous procedural vulnerabilities, such insecure software download,

[5] *I-voting* is used by the Estonian voting system to emphasise that it makes use of the internet and distinguish it from other forms of *e-voting*, or electronic voting.

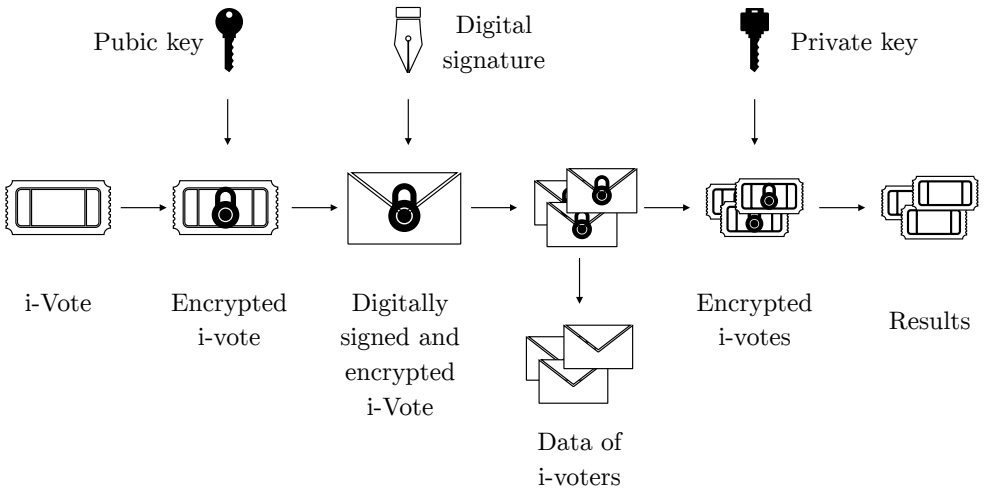


Figure 2.3: Estonian envelope scheme as illustrated by the State Electoral Office of Estonia [40].

and source code issues, for example allowing shell injection or denial of service attacks. Springall et al. recommend abandoning the Estonian i-voting system.

An allegedly up to date open source code repository reflecting the IVXV system is available for public scrutiny[6].

2.3.6 Votem

The commercial Votem[7] system is described in the white paper *PROOF OF VOTE - An end-to-end verifiable digital voting protocol using distributed ledger technology (blockchain)* [10]. Votem Corp. was founded in 2015 with an ambition of 1 billion voters by 2025.

The Votem protocol leverages a blockchain architecture for distributed vote records and key generation. Certain nodes in the blockchain network are given special privileges to administer the process. A re-encryption mixnet is used for voter anonymity, in a similar manner to [34]. For voter authentication and authorization, a multi-signature scheme is used.

One vulnerability identified by the authors is that an authoriser can learn from the timing and IP address of blind signature obtainment, the identity of a voter. The au-

[6] `https://github.com/vvk-ehk/ivxv` [cited 2020 Jun 4]
[7] `https://www.votem.com/` [cited 2020 Jun 4]

thors leave voter registration to election authorities, claiming the system can support common interfaces. In response to a DoS attacks, Votem relies on the blockchain architecture itself to service continuation of an election. The main limitation of the Votem system lies in its commercial secrecy, reducing the ease of trust.

2.3.7 Voatz

Voatz[8] does not offer great detail about its online voting system. The white paper titled *Under The Hood - The West Virginia Mobile Voting Pilot* [41] describes some of the features and procedures, but only at a high level and in light of the 2018 West Virginia pilot. Only 160 users completed the application download for the pilot [41].

Voters are authenticated using bio-metrics. In particular, they take a picture of their ID card along with pictures of their faces from multiple angles using a mobile app. Their ID is then tied to the device, supposedly ensuring that each voter can only use one device and each device can only be used by one voter. Anonymous IDs are generated for each voter upon completing the ballot and committed along with the vote to a blockchain. The blockchain is redundantly distributed across 32 servers provided by two cloud services (Microsoft Azure and Amazon Web Services). Should a voter cast a second ballot, her first vote is discarded. To decrypt votes and tally votes, two encrypted portable drives are inserted into a laptop to assemble the blockchain votes into a PDF report. The report is used with existing tabulating software to arrange aggregated data.

Specter et al. conduct a security analysis of Voatz and present several issues [33]. The authors critize the lack of public disclosure of software implementation details and reverse engineer the application to model its security. An attacker with root access to a voter's device is found capable of reading and modifying votes. A network intermediary can infer voting information and a server-side attacker can likely alter the outcome of an election. West Virginia discontinued the use of Voatz following the critique from Specter et al.[9]

[8] `https://voatz.com/` [cited 2020 Jun 4]

[9] `https://www.forbes.com/sites/jasonbrett/2020/05/19/does-trump-know-blockchain-is-nominating-him-for-president-in-2020/` [cited 2020 Jun 4]

2.3.8 D-DEMOS

Chondros et al. presented D-DEMOS, a collection of two distributed E2E verifiable online voting systems using code voting, in 2019 [2]. One system is asynchronous and one is synchronous. Besides the setup phase, for which the authors assume a secure channel for the communication of secret code sheets, D-DEMOS has no single point of failure. The systems involve the election authority, voters, vote collection sub-systems, a distributed bulletin board and trustee sub-systems. The trustees share responsibility for correct tallying. Chondros et al. claim that their system does not rely on voter device trust, since relationships between choices and vote codes is only communicated to the voter via the assumed secure channel. This allegedly also makes the systems immune to phishing attacks. End to end verifiability is supposedly ensured by vote receipts and zero-knowledge proofs. Proofs of correctness are also applied to initialisation procedures performed by the election authority.

serial-no		
	Part A	
vote-code_1	option_1	vote-receipt_1
...	...	...
vote-code_m	option_m	vote-receipt_m
	Part B	
vote-code_1	option_1	vote-receipt_1
...	...	...
vote-code_m	option_m	vote-receipt_m

Figure 2.4: D-DEMOS ballot as illustrated by the Chondros et al. [2]. The voter makes a candidate choice and chooses arbitrarily one of the two vote codes supplied for any candidate. The part from which no vote code was chosen is used for auditing as the bulletin board discloses these codes as proof of receipt of the other codes.

Both the synchronous and asynchronous system are evaluated through simulating elections. The authors find that both systems scale well in terms of both voter population size and the number of candidates, but note that the synchronous system performs better, though at the cost of robustness offered by the asynchronous system. D-DEMOS has also been used in smaller Greek university elections and is planned to be used in a trade union election. The D-DEMOS systems only address elections where voters vote for one out of many candidates in a single competition and not

less trivial voting schemes. Chondros et al. do, however, suggests some ideas for extension to more complex schemes.

2.3.9 Main limitations of current online systems

Table 2.1: Overview of existing system's satisfaction of the requirements for online voting systems. A legend of the system authors is provided below.

Requirement \ System	[39]	[34]	[1]	[32]	[40]*	[10]	[41]	[2]
E2E verifiability	✓	✓	✓	✓	✓	✓	-	✓
Voter authorisation and authentication	✓	✓	-	✓	✓	✓	✓	✓
Receipt freedom	-	✓	-	-	✓	✓	✓	✓
Secret ballot	-	✓	✓	✓	✓	✓	-	✓
Usability and accessibility	-	-	✓	✓	✓	✓	✓	✓
Software independence	✓	✓	✓	✓	✓	✓	-	✓
Transparency	✓	✓	✓	✓	✓	-	-	✓
Coercion resistance	-	✓	-	✓	✓	✓	✓	-

*Although doing seemingly good with regards to the requirements, Springall et al. [31] find several vulnerabilities.

[39] Hubbers et al. [1] Ryan and Teague [40] Valimised [41] Moore and Sawhney
[34] Clarkson et al. [32] Ryan et al. [10] Becker et al. [2] Chondros et al.

In light of the discussion of online voting system requirements in Section 2.2, review of current systems above and supported by the discussions of [9, 11, 12], it is worthwhile to summarise the main limitations of current online voting systems. Table 2.1 shows the existing system's satisfaction of the requirements for online voting systems. The requirements of secret ballots and receipt freedom, particularly to mitigate coercion, seem to be a top challenge for existing systems. These requirements are especially demanding when balanced with E2E verifiability. Complicated usability and vulnerabilities in client side hardware and software are other challenge that several systems have not mastered. If the voter interface cannot be secured, than neither can the system as a whole.

2.4 Threats to online voting systems

Through the review of existing online voting systems, several common threats have been identified. Although the threat analysis of any system, included that of this project, depends on the system's particular design, it is worthwhile to discuss the commonly faced threats when dealing with an online voting system. Here, following a discussion of potential attacker capabilities, threats are discussed and categorised using the *STRIDE*[10] framework. STRIDE is commonly used for finding threats to a system [42].

An online voting system attacker may be in control of the voter (client-side), the election authority or the infrastructure connecting the voter with the election authority (server-side) [33]. All these attacker capabilities are relevant to online voting systems, especially considering the potential immense interest and resource devotion of adversaries with an interest in election outcomes (e.g. nation states) [11, 31, 33].

A summary of attacks related to online voting systems are presented in Table 2.2. The list is not exhaustive but rather reflects the most common attacks discussed in the literature and further motivates the requirements discussed in Section 2.2.

Table 2.2: Summary of attacks related to online voting systems.

Client-Side	Server-Side
Over-the-shoulder coercion	Eavesdropper information disclosure
Italian attack (coercion)	MITM vote tampering
Voter spoofing	Vote disclosure
Election authority phishing	Vote and/or count tampering
Client-side tampering	Repudiation attack
Client-side information disclosure	Denial of service

Even if none of the attacks are successful in actually manipulating the outcome of an election, they may challenge the trust of a system's integrity if not dealt with evidently [9, 31].

[10] ***S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege*

2.4.1 Client-side attacks

The voter may (e.g. physical force) or may not (e.g. browser malware) know that he is controlled by an adversary [34]. Examples of coercion attacks where the voter knows he is under attack are *over-the-shoulder* or *Italian attacks*. In an over-the-shoulder or *rubber hose* attack, the adversary commands and observes the voter obeying by physical presence [11]. An Italian attack consists of the adversary telling the voter to rank candidates in a specified, distinguishable order and observing the specified, anonymous vote on a public bulletin [12]. An adversary may also control the voter by impersonating the election authority, for example via an illegitimate website, and promote incorrect information about the election [11].

To attack the client-side, an adversary may make use of existing malware on the client device or inject new malware. In their survey of the Estonian i-voting system, Springall et al. discuss the availability of existing botnets that may offer an adversary control over voter malware from a target geographic region [31]. Springall et al. also shine light on the vulnerabilities in third-party software and software that is downloaded by the voter. For example, the binaries comprising a voting application may be compromised themselves, or maliciously modified while downloading. Independent of how the malware is injected, several attacks are possible.

An adversary may monitor the authentication procedure and, if voting several times is permitted, may impersonate the voter on a separate device and cast a new vote to effectively alter the choice of the voter [31] (*spoofing*). The adversary may also impersonate the election authority in a *phishing* attack and obtain the voter's credentials, vote or other sensitive information (*spoofing*).

Even more efficient may be an attack that discards or changes a voter's choice without her noticing (*tampering*). To convince the voter that nothing is wrong, illegitimate feedback may be provided in the form of a web-page or receipt [43]. An adversary posing as the election authority may also alter the ballot presented to the voter.

While controlling the client-side, the adversary is also free to observe voters' choices, thereby compromising *secret ballots* [43] (*information disclosure*).

2.4.2 Server-side attacks

In similar ways to client-side malware, infrastructure and servers may be infected via malware. Other attack vectors include insiders and physical intrusion [11].

Depending on the information transmitted with a vote, an intermediary *eavesdropper* may be able to infer the identities and choices of voters [43] (*information disclosure*). Such revealing information may also be inferred from the packet size of votes. Specter et al. found that votes for longer candidate names were transmitted via significantly larger packet sizes than those for shorter candidate names [33]. A more powerful MITM intermediary may go further and alter or drop the vote and confuse the voter in a similar fashion to a client-side attacker.

If association between choices and voters are maintained, a server side attack may disclose voter choices, thus failing the requirement of secret ballots [31] (*information disclosure*). A server-side attacker can also, in the absence of sufficient E2E verifiability, manipulate votes or the vote counting. Without sufficient intrusion detection and/or immutable server logs, an adversary can destroy any evidence of misconduct (*repudiation*).

DoS attacks is an unavoidable threat to online systems. Bernhard et al. argue that *selective* or *targeted DoS*, by which certain portions of a heterogeneous voting population may be disrupted, thus potentially altering an election outcome [12]. Vote flooding is a concrete example of a DoS attack by which an adversary submits an excessive amount of illegitimate votes to the server, thereby delaying the processing of legitimate votes [9]. Although the cost of maintaining service availability comparable to the large tech companies like Google or Facebook may be in the order of millions of dollars and may not be achievable, an online voting system is expected to have some measure to mitigate the effect of DoS attacks [11] (*denial of service*).

2.5 A note on quantum computing

As the current online voting systems rely heavily on cryptography, it is worth discussing the risk of quantum computers breaking conventional cryptography. Becker et al. [10] split quantum computing's impact on cryptographic algorithms into three groups. The first group are algorithms that are completely susceptible to breaking by a quantum computer via Shor's algorithm (e.g. RSA). Shor's algorithm is a

polynomial-time quantum algorithm for the factoring problem, on which conventional security relies, traditionally consider to be a hard problem [44]. Such algorithms are made insecure by the emergence of quantum computers. Second are algorithms that are partially threatened by quantum computers via Grover's algorithm (e.g. AES and SHA). Becker et al. claims these algorithms can remain secure by increasing the key length used. Grover's algorithm does not reduce time complexity as much as Shor's algorithm, but gives a quadratic speed up [45]. Third and last are algorithms that are unaffected by any quantum algorithm.

As several of the systems discussed in Section 2.3 make use of algorithms that are susceptible to breaking by quantum computers, online voting systems of the future will need to consider the potential impact of quantum computation. Bernhard et al. discuss the *everlasting privacy* property of voting systems for which privacy is independent of assumptions about cryptographic problem hardness [12].

Chapter 3

Protocol Design

Here, we describe the protocol design. First, a brief overview of a voting system model using the protocol is provided as basis for the discussion of the threat model assumptions. Following this, the system is described in greater detail, taking into consideration the threat model assumptions. In Chapter 4, the continuation of the threat model follows and the protocol security is evaluated.

The protocol design was inspired by the designs of Ryan and Teague [1] and Chondros et al. [2], with original improvements to the ballot design and distribution of trust and responsibilities within the election authority.

For the remainder of this report, the following nomenclature is used. In an *election*, a *voter* makes a *choice* of *candidate* from a *ballot* and submits a *vote* to the *election authority* that is to be included in the final *tally*.

3.1 System overview

To distribute trust between several separate entities, the election authority's responsibilities are divided among four election authority agents: the *initiator*, the *collector*, the public *bulletin board* and the *tallier*. This is comparable to how traditional poll-site election schemes make use of separate groups of people for different tasks, such as voter authentication, vote collection and tallying. The election authority communicates with a group of *voters*, as well as *adversaries* who may also be voters. Figure 3.1 illustrates the agent composition.

The initiator is responsible for generating ballots and cryptographic keys and for dis-

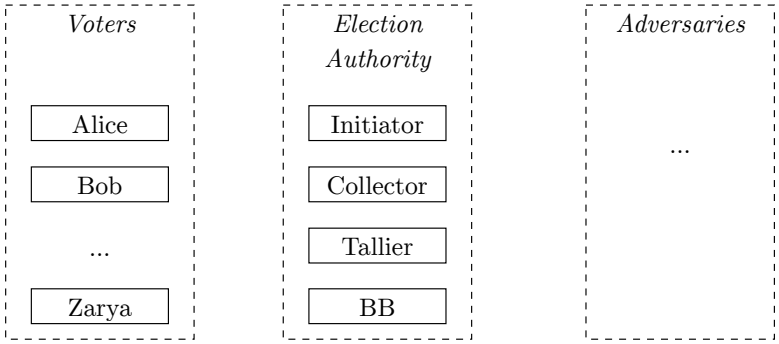


Figure 3.1: System agents. The election authority consists of an initiator, vote collector, public bulletin board (BB) and tallier.

tributing these to the voters and appropriate election authority agents respectively. Like other code voting schemes, ballots contain candidate names and associated vote codes. For this system in particular, ballots are made up of a unique pseudo-random credential, a unique pseudo-random error receipt code, the bulletin board's public key and a list of candidate names along with associated encrypted vote codes and receipt codes in a similar style to Ryan and Teague [1]. The ballot layout is illustrated in Figure 3.2.

The collector receives votes in the shape of a credential and vote code from voters and forwards an anonymous decryption of the submitted vote code to the public bulletin board after checking that the credential and vote code is valid. This includes ensuring that the voter has not already voted. The bulletin board is responsible for presenting an append-only sequence of authenticated messages, as well as returning receipt codes to voters via the collector that assure the voters that their votes are registered. The tallier is responsible for making the final tally based on a bulletin board read after collection terminates and polls close. The votes are encrypted in such a way that the collector cannot infer any submitted vote's underlying candidate choice, thereby protecting secret ballots and voter anonymity. This is discussed in greater detail in the next few paragraphs.

The voting protocol consists of three phases: *initiation*, *collection* and *tallying*. It is illustrated Figure 3.4.

During initiation, the initiator generates and distributes the private keys of all four election authority agents respectively and publishes the associated public keys to

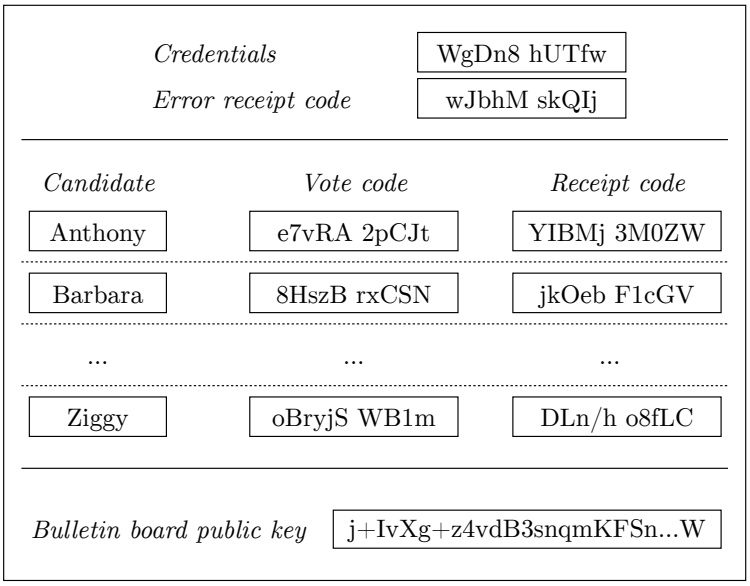


Figure 3.2: Example ballot, comprised of credential, list of candidate, vote code and receipt code triples and the bulletin board public key.

allow election authority agents to present digitally signed messages and have them verified by the other agents and any third party. The initiator also generates a ballot for every voter and distributes it to the respective voter.

Upon completing initiation, collection proceeds by voters submitting their votes to the collector. The composition of vote codes is illustrated in Figure 3.3. Each candidate is mapped to some binary string candidate identifier which is encrypted using an inner vote code encryption key. The resulting inner vote code is again encrypted using an outer vote code encryption key. It is the resulting outer vote code that is represented on the ballot and potentially submitted to the collector, who then decrypts it and posts the corresponding inner vote code on the bulletin board. The collector does not know how to decrypt the inner vote code, and can thus not infer the candidate choice submitted for any credential. To compute the receipt code that is expected in return by the voter, the bulletin board encrypts the inner vote code using a receipt key. The receipt code is sent back to the voter via the collector, and the voter may verify that the vote submitted has reached the bulletin board by comparing the received receipt code with that printed on her ballot.

After vote collection terminates and polls close, the initiator publishes the decryption

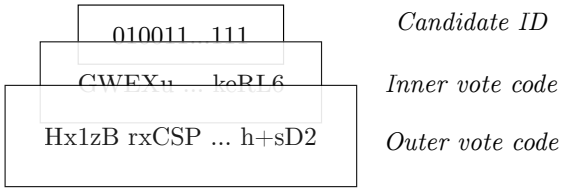


Figure 3.3: Vote code layers. One key is needed to decrypt the outer vote code to the inner vote code, and a different key is needed to decrypt the inner vote code to the candidate ID.

for the inner layer vote to the bulletin board. This allows anyone to decrypt the anonymous inner layer votes on the bulletin board to candidate choices and make up the final tally. While it requires no special authorisation and could be done by any third party, the tallier takes on this responsibility and publishes the tally to ensure that this is done.

3.2 Threat model assumptions

First and foremost, it is assumed that voters trust in the initiator to keep private keys confidential, generate fair ballots with candidates represented evenly, provide each eligible voter with a ballot and keep secret which ballot was sent to who. The latter point is essential to ensure secret ballots. Actions to reduce dependence on this strong assumption are discussed in Chapter 4. We further assume that secure channels exist between the initiator and the other election authority agents. This is comparable to the assumption under traditional pollsite elections that lists of eligible voters may be transferred securely from a central authority to pollsites. A secure channel is assumed between the initiator and the voter for the distribution of ballots. Such a secure channel may be provided by allowing voters to collect ballots from a physical location or by postal mail and is commonly assumed [1, 2, 39].

We assume that the collector does not disclose which inner vote code was decrypted from the outer vote code submitted by a particular credential. This is critical to receipt freedom and ways of mitigating the need for this strong assumption are discussed in Chapter 4.

An append-only public bulletin board is assumed. Such a bulletin board presents to

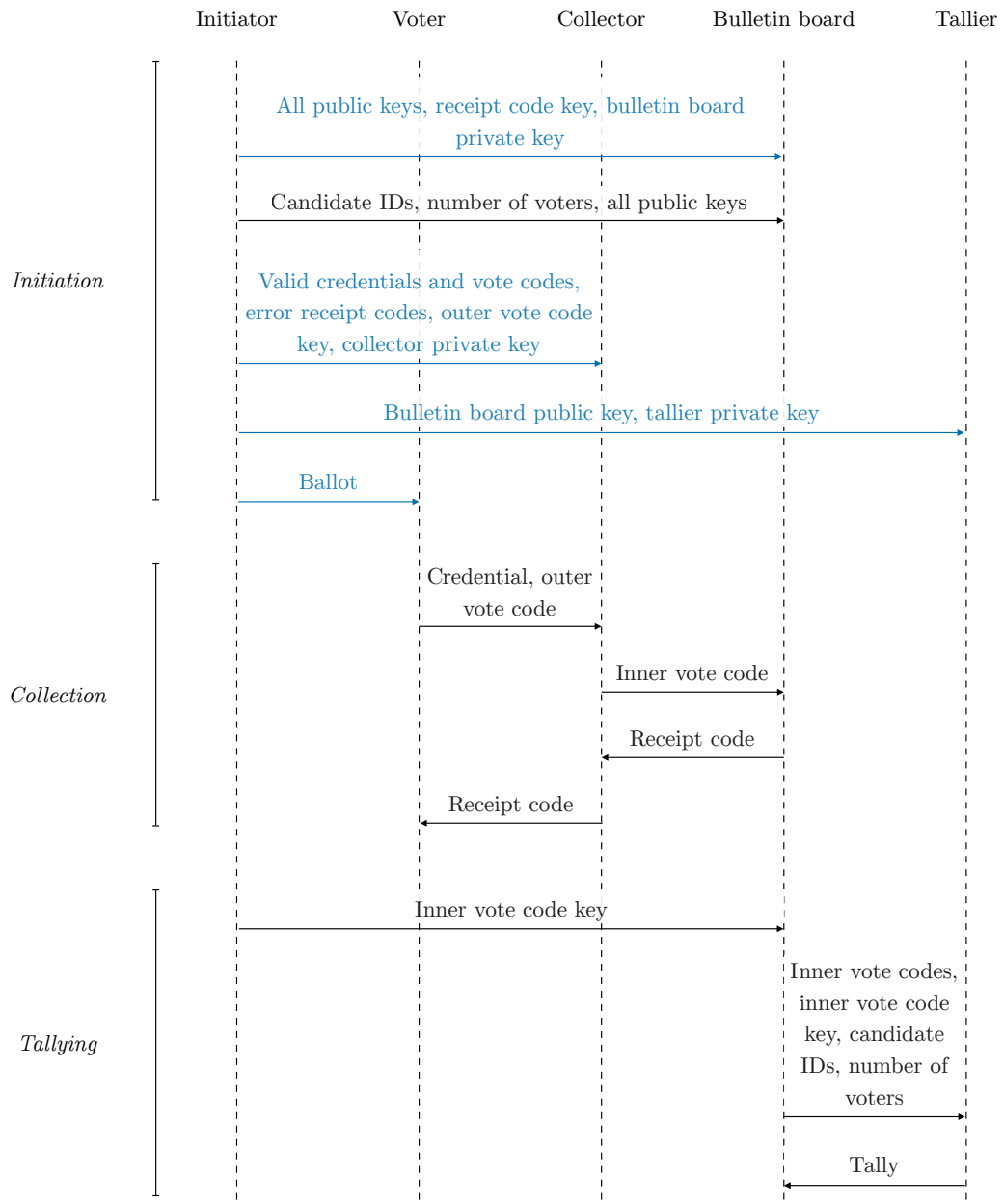


Figure 3.4: Protocol sequence diagram. Communication coloured in blue is assumed carried out via a secure channel. All other communication can be public.

any reader an append-only history of authenticated messages. Heather and Lundin

[46] have proposed a suiting template and the specific design used in this project is discussed in Section 3.3. Since the bulletin board knows how to issue valid receipts for inner vote codes, there is a potential vulnerability in a colluding collector and bulletin board. The collector could simply share with the bulletin board both an inner vote code for which to return a receipt to avoid voter suspicion and an alternative inner vote code for posting to the bulletin board. It is therefore assumed that the election authority agents are separate agents that operate independently.

We assume that an adversary, including a misbehaving election authority agent, may perform any polynomial time computation. The security of underlying cryptographic primitives under this assumption is also assumed. In particular, it is assumed that a symmetric cryptographic system with appropriately set security parameters may be implemented that is practically intractable to break and reveal the underlying secrets, such as AES-128 (128 bits key) at the time of writing. A similarly secure asymmetric encryption schemes is also assumed. To date, RSA-2048 (2048 bits key) is an example of such a scheme.

An adversary may be in control of the voter, either by means of coercion or through malware. A coercer does, however, not maintain control over the voter for the duration of open polls. Following the assumption of a secure channel for ballot distribution, the coercer is unable to interfere with a voter receiving her ballot.

The infrastructure connecting voters and the election authority may be compromised, and an adversary may eavesdrop on and tamper with communication. However, we assume that voters are able to communicate securely with the election authority by alternative means (e.g. via physical meetings) if necessary.

3.3 System details

In this section, we build upon the introductory overview presented in Section 3.1 and discuss the details of the system agents and protocol. This is done in the order of protocol phases before we discuss the design of the append-only bulletin board. An illustrative overview of the protocol cryptography is given in Figure 3.5. We also highlight the protocol differences from those of Ryan and Teague [1] and Chondros et al. [2].

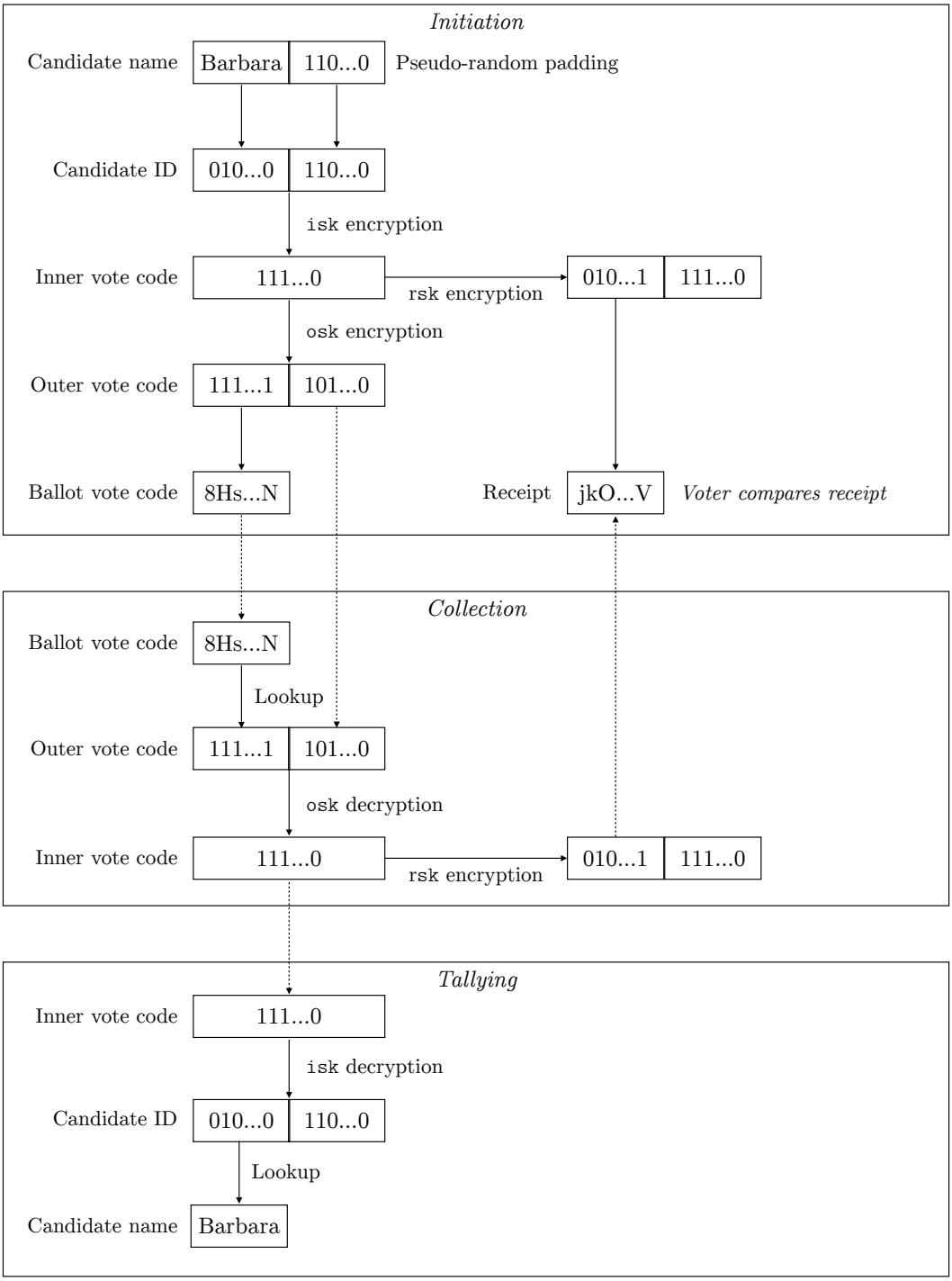


Figure 3.5: Illustrative overview of the protocol cryptography.

3.3.1 Initiation

In the initiation phase, the initiator first generates asymmetric encryption key pairs for each election authority agent, including itself. The public key pairs are denoted (insk, inpk), (csk, cpk), (bbsk, bbpk) and (tsk, tpk) for the initiator, collector, bulletin board and tallier respectively, where keys ending with "sk" are the private keys and "pk" are the public keys. The initiator then generates the symmetric keys for outer vote codes, inner vote codes and receipt codes. These are denoted osk, isk and rsk respectively. The choice of encryption schemes and impact of key lengths is discussed in Chapters 5 and 6.

Next, all public keys are transmitted to the bulletin board, along with the bulletin board private key and the receipt code key. All public keys are also published on the bulletin board, such that they are readily available to any third-party reader.

The candidate IDs are generated by assigning each candidate a unique number and representing this by a string of bits long enough to represent any candidate ID. The *initiator* publishes the candidate name to candidate ID mapping on the bulletin board along with the total number of eligible voters.

What remains is the generation of ballots. To generate a particular candidate's vote code for any ballot, the initiator first pads the candidate's ID code with a pseudo-random pad which makes the padded candidate ID different from those of all other ballots. The initiator then encrypts the padded candidate IDs using the inner vote code key, isk, and a pseudo-random initialisation vector (IV). The IV is used to introduce further randomness in the encryption, thereby enhancing security when using the same keys for repeated encryption [23]. It does, however, not need to be kept secret and is thus appended to the encrypted ciphertext that it is easily extracted and available for decryption.

To generate the receipt code to be printed on the ballot for the specific candidate, the inner vote code is encrypted using the receipt code key, rsk. Finally, the inner vote code is encrypted using the outer vote code key, osk, to make the *outer vote code*.

In the interest of usability, only a certain number of base64 encoded bits of the outer vote code are printed on the ballot. The credential and error receipt codes printed on the ballot are both pseudo-random strings of bits that are, like the printed vote and receipt codes, base64 encoded of human manageable length and unique across

ballots. We argue that a string of 10 characters is easily manageable for most voters, as it is of comparable and smaller length than frequently used credit card numbers and personal identification numbers. The shorter codes do, however, come at the cost of weakened security [1, 2]. This trade-off between usability and security of the vote codes is discussed in Chapter 4.

The outer vote code key, `osk`, the error receipt codes and a shuffled list of all valid outer vote codes for each voter credential is securely transferred to the election authority collector. Note that the collector knows how to compare submitted abbreviated vote codes from the voter to the full outer vote codes.

3.3.2 Collection

Using her ballot received from the election authority, the voter opens a web browser and accesses a specified website. She makes a candidate choice and enters into a web form her anonymous credential and the vote code corresponding to her candidate choice. After submitting the form to the collector, she awaits a receipt from the website.

The collector first ensures the credential is legitimate and has not already been used to vote via lookup in the set of valid credentials transferred from the initiator and a set of spent credentials. Provided that the credential is legitimate, the collector assesses whether the submitted vote code is also legitimate. This is done by matching the submitted ballot vote code with appropriate parts of the valid outer vote codes for those credentials. If the submitted vote code corresponds to a legitimate outer vote code, the collector decrypts the outer vote code and publishes the inner vote code to the bulletin board. The bulletin board returns the corresponding receipt code obtained by encrypting the inner vote code with `rsk` to the collector, who forwards the receipt code to the voter.

Ryan and Teague [1] in their protocol found that voters could undermine the collector's integrity by submitting invalid vote codes that would then be forwarded by the collector. By providing the collector with a shuffled list of valid vote codes for each credential, we avoid this threat to collector integrity, at minimal security sacrifice. First, since the collector is only presented a shuffled list of vote codes, it cannot infer which candidate choice the credential has made. Furthermore, the collector does not know the identity of the voter who used the anonymous credential. Sec-

ond, should the collector attempt to change the vote code submitted by the voter with another valid vote code, it will not be able to generate a corresponding receipt code, as publishing the correct inner vote code on the bulletin board is needed to get the correct receipt code. The same applies to collector attempts at discarding votes. Should, however, the collector and bulletin board collude, they could together pose to the voter as behaving honestly by returning correct receipt codes, while registering altered inner vote codes on the bulletin board. This is discussed further in Chapter 4.

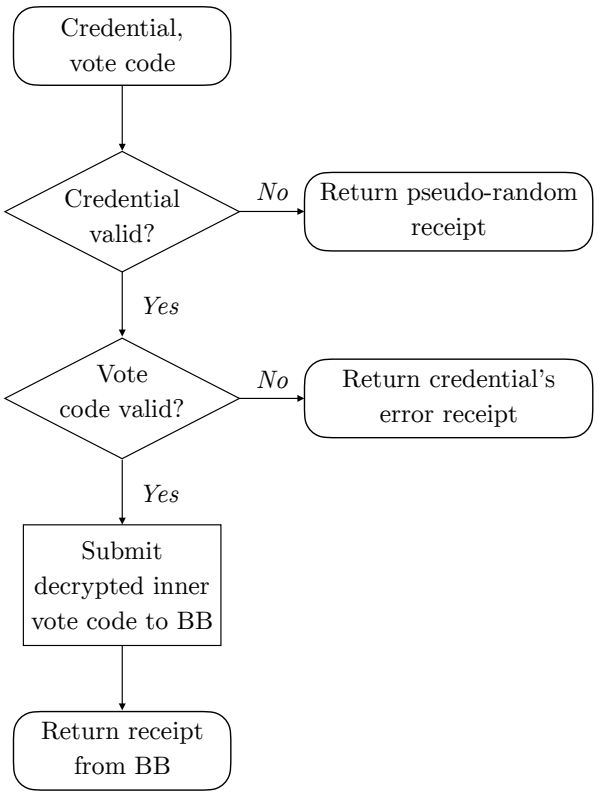


Figure 3.6: Collector logic flow.

Should the credentials or vote code be illegitimate, an error receipt code is returned according to the logic flow illustrated in Figure 3.6. If the credentials were legitimate, the collector returns the error receipt code supplied by the initiator for the credential, to allow the voter to understand that the legitimate collector did not approve of the vote code. This could, for instance, be due to a mistake when entering the vote code and can be retried. Otherwise, given illegitimate credentials, a pseudo-random error code of the appropriate form is returned. This is done to not leak any information to an adversary unaware of the voters expected error receipt code.

The voter receives the receipt code returned by the collector and compares it to the receipt code printed next to the vote code she submitted. If it matches, she can rest assured her vote has been received by the election authority and posted on the bulletin board. Otherwise, she can either retry submitting the vote code, or contact the election authority via a separate secure channel to submit her vote by other means. One alternative to resolve voting by an alternative channel is to allow physically authenticated voters to check with the collector whether any of the vote codes listed on their ballots have been decrypted and registered on the bulletin board. Should that be the case for a voter, she can be allowed to submit an alternative vote code to the collector, who indicates on the bulletin board that the newly decrypted inner vote code should be tallied rather than the identified other inner vote code. If no vote has been registered on her behalf, she can simply submit the vote code she wishes to the collector directly, without needing to rely on her potentially compromised network channel. Such an overwriting of a voter's vote may also be used to mitigate coercion, since a coerced voter may resubmit a new vote after the coercer is not present anymore.

Chondros et al. [2] propose a two part ballot with each part containing a distinct set of vote codes. The voter makes a candidate choice and chooses arbitrarily one of the two vote codes supplied for any candidate. The part from which no vote code was chosen is used for auditing as the bulletin board discloses these codes as proof of receipt of the other codes. In our protocol, we avoid the need for a two part ballot by having the bulletin board issue the receipt codes returned to the voter who can, provided the receipt code matches that printed on the her ballot, rest assured her vote has been received by the bulletin board.

3.3.3 Tallying

After vote collection ends, the initiator publishes on the bulletin board the inner vote code key, `isk`, such that all inner vote codes posted on the bulletin board may be decrypted to padded candidate IDs. Interpreting the candidate IDs and names based on the padded IDs is a trivial exercise. Should a considerable number of inner vote codes not decrypt to candidate codes, the tallier will report that there is evidence that the collector has been dishonest or erroneous and the appropriate measures may be taken.

3.3.4 Bulletin board

The need for an append-only bulletin board often appears in online voting and several solutions have been proposed [46, 47]. In this project, the approach of Heather and Lundin [46] is taken.

The bulletin board is required to have the following three properties: *unalterable history*, *certified publishing* and *timely publication* [46]. The unalterable history property is ensured by hashing bulletin board entries in a chain-like manner[1] and having the authenticated writer and bulletin board commit to the history via digital signatures. This yields the append-only characteristic. Certified publishing is achieved by having message writers sign their messages so that they may later be verified as authentic. Timely publishing is achieved by timestamping messages and prohibiting messages that claim to be written earlier than the latest message on the bulletin board. Any reader of the bulletin board may verify its integrity and writers may verify that their messages are published [46].

Only the initiator, collector and tallier are expected to publish on the bulletin board and only they are permitted by the bulletin board to do so. Should the bulletin board misbehave and attempt to modify the history of recorded votes, any reader, including the tallier, will be able to detect the inconsistency.

[1]For bulletin board entry i consisting of message m, timestamp t and writer ID w, the hash, H_i is expressed as $H_i = \mathcal{H}(m, t, w, H_{i-1})$ where $\mathcal{H}$ is a hash function and H_{i-1} the previous entry's hash.

Chapter 4

Security Analysis

In this chapter, we analyse the security of the voting protocol in light of the discussion about common threats to online voting systems from Chapter 2 and the threat model assumptions made in Chapter 3. After individual consideration of the threats and associated mitigating actions, we summarise our findings in Section 4.8.

4.1 Initiator compromise

In Chapter 3, we discussed the superior assumption of complete voter trust in the election authority initiator. It is indeed a strong assumption and violation of it would make for a tremendous vulnerability. We thus discuss the ramifications should it not hold and suggest some actions to mitigate the reliance on the assumption.

The most serious threat to a code voting system may be argued to be leakage of vote codes [1]. Leaks of this form can potentially enable an eavesdropper to interpret votes, violating secret ballots, or a MITM adversary to tamper with votes. An adversary with knowledge of a voter's credential and vote codes can also simply impersonate the voter and vote on her behalf.

Closely related to the vulnerability of code leakage is the similarly serious vulnerability of key leakage. Were the initiator to be compromised, the impersonation of any election authority is fair game as what gives these agents their privilege are the secret keys they hold. An adversary would not necessarily disclose that the initiator was compromised and instead decrypt votes completely to infer voter choices or manipulate the election entirely.

To address the potential vulnerability of a compromised initiator, we suggest three mitigating actions. First, the initiator responsibilities of key, credential and code generation could potentially be shared among several distributed agents. Gennaro et al. [48] propose an interesting protocol showing promising suitability. By distributed key generation, no single entity is trusted with the generation of the keys, thus making compromise harder as it would require compromising several entities.

Second, there is a risk of the initiator making use of its knowledge of which voter received which ballot and thus credential and threaten secret ballots. To mitigate this risk, one suggestion could be to distribute ballots after passing through a mix network by which the destination addresses and ballots are shuffled so that no one has knowledge of just which ballot went where. However, there may be benefits of knowing which ballot was intended for which voter, for instance in the case of lost ballots. Rather, it would be interesting to look into the possibility of distributed distribution of ballots. That is, rather than the voter receiving one ballot from an initiator, he receives several partial ballots from different initiators that collectively make up a valid ballot.

Third, the voter is also assumed to trust that ballots are fair, in the sense that all candidates are represented evenly. To eliminate the need for such an assumption, zero-knowledge proofs could potentially be employed to provide the initiator with means to prove that a ballot is indeed fair. Closely related to this is the approach of homomorphic encryption, by which a voter could perform some computation on the encrypted ballots and get assurance all candidates are represented. Chondros et al. [2] utilise this for guaranteeing fair ballots.

4.2 Collector compromise

Another fairly strong assumption is that of the collector not disclosing associations between credentials and bulletin board published inner vote codes, or similarly the outer vote code key. Violation of this assumption would directly threaten receipt freedom as voters, in possession of the credential printed on their ballot, could prove which inner vote code they caused to be published on the bulletin board and later decrypted to a particular candidate choice.

Threshold cryptography was first introduced by Shamir [28] in 1979 and has since been used extensively to distribute trust in encryption and decryption [1, 29]. Rather

than the collector being a single entity, it could be divided into several entities, each possessing a share of the inner vote code key. Successful decryption of outer vote codes would then require a threshold number of collectors to cooperate. Influencing the process would consequentially require the compromise of a threshold number of entities.

Should the collector attempt to tamper with the outer vote codes submitted by a voter, it will, as discussed in Chapter 3, not be able to generate the receipt code expected in return by the voter. The same argument applies to dropping votes. Thus, such attempts will be detected by the observant voter. We argue that since receipt code verification from the voter's point of view only consists of comparing two 10 characters long strings, the gained confidence in successful voting from such verification outweighs the burden to the voter of performing the verification exercise. In Chapter 6, we demonstrate that the waiting time between vote submission and the return of a receipt code is expected to be inconsiderable and thus facilitate rapid and easy voter verification. Unexpected collector behaviour could be reported by the voter to a trusted election authority agent.

4.3 Corrupt bulletin board

As introduced in Chapter 3, a corrupt bulletin board inclined to collude with the collector could easily issue receipt codes in response to the inner vote code decrypted from the vote code submitted by the voter, while appending to its history a different vote code. To reduce the dependence on the assumption that the collector and bulletin board do not collude, the bulletin board could be distributed to more closely resemble an immutable distributed ledger, or blockchain. Such a measure would also enhance the bulletin board's resilience to denial of service attacks, as read and write requests could be services by any of the distributed entities. Culnane and Schneider [47] build on the work of Heather and Lundin [46] used in this project and propose a distributed bulletin board.

4.4 Secret ballot breach

Apart from the above discussed threats to the initiator, collector and bulletin board, we highlight a potential threat to ballot secrecy and voter anonymity. With the pro-

tocol described as in Chapter 3, there is a vulnerability in that an adversary with a good overview of the network traffic may be able to learn the inner vote code associated with the voter's vote from the timing of a voter's vote submission, bulletin board addition and voter's receipt reception. Should for instance an adversary observe from network traffic that Alice the voter submits a vote at time T_v and receives a receipt at time T_r, and that the bulletin board appends an inner vote code between times T_v and T_r, the adversary may be able to infer that the Alice submitted what became that inner vote code, which is after polls close open to decryption to a candidate choice.

The current bulletin board design, due to Heather and Lundin [46], does not support delaying bulletin board entries from being written or shuffling the order of these, yet such measures could help mitigate the risk of inferring votes from vote submission and bulletin board writing times.

4.5 Code guessing

In Chapter 3 we mentioned the trade-off between usability and security when choosing the length of codes printed on the ballots. For the suggested 10 base64-encoded characters per code, there are more than a quadrillion quadrillion[1] possible credential and vote code pairs. However, should an adversary be able to eavesdrop on the voter's vote submission and read her credential, the number of possible guesses the adversary can make is reduced to just more than a quadrillion. To guess the receipt code that is expected by the voter for any submitted credential and vote code is easier by a similar argument since it only involves guessing one code rather than a pair of codes. Furthermore, if the adversary is allowed to repeatedly guess credentials and vote codes, the risk of successfully submitting an illegitimate vote increases.

To mitigate any chance of successfully guessing vote or receipt codes for a credential, we suggest the easily implementable mitigating action of allowing at most a certain number of vote submission attempts for any credential. This resembles a password policy commonly used in online web applications. Should a voter fail to submit a vote in the number of attempts given, he can contact the election authority via a separate secure channel and resolve the matter by alternative means, as discussed in Chapter 3. Similarly, a voter that receives an unexpected receipt code (not including

[1] A more than 31 digits number.

the error receipt code) several times should be instructed to contact the election authority via a secure channel, rather than retrying vote submission.

Guessing the longer cryptographic vote codes is assumed intractable, as is the forging of signatures and breaking of hashes. However, in anticipation of quantum computers playing a bigger role in the foreseeable future, the system is vulnerable to the underlying cryptography breaking. This is particularly true for the asymmetric cryptography, susceptible to breaking by a quantum computer, upon which the bulletin board security relies. Post-quantum cryptography is under development[2], and should be considered for the protocol once available.

4.6 Denial of service

Denial of service is, as discussed in Chapter 2, an unavoidable threat to online voting systems. In addition to the general web application measures that should be taken to prevent DoS attacks[3], there are some particular threats to be aware of and that may be mitigated.

We have already discussed the distribution of both collector and bulletin board, which could both help to alleviate DoS attacks. Another threat is that of targeted DoS, for instance targeting geographic areas with a known political bias. To mitigate this, beyond the efforts of distributed collectors, we suggest a backup alternative way of voting. In Chapter 3 we suggested a method by which a physically present and authenticated voter could be allowed to submit a vote using her credential and vote code of choice. We elaborate on this alternative route in a few paragraphs.

A closely related threat to targeted DoS is that of the collector denying the submission of votes based on the voters IP address. To mitigate this, we could suggest the use of a mix network prior to votes arriving at the collector, but argue that the risk is still there that a MITM may drop votes from specific IP ranges before reaching the collector. We suggest again the backup physical route of voting.

[2] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography` [cited 2020 Aug 13]
[3] `https://www.cloudflare.com/learning/ddos/ddos-mitigation/` [cited 2020 Aug 13]

4.7 Coercion

At last we consider the threat of coercion. It is hard to argue that coercion may be avoided completely without a physically controlled environment, such as a pollsite, if even it can be avoided there. To the best of our knowledge, there exists no provably secure way of ensuring a remote client is not under coercion.

Rather than attempting to detect coercion directly, we suggest like many other have done before us (e.g. [34, 40]) the possibility to overwrite votes at a later stage. In contrast to previous approaches, we suggest that such overwriting is done by contacting the election authority and casting a vote in a safe environment like a pollsite or similar. To elaborate on this, a voter could authenticate himself by physically meeting at a specified location and present a valid form of identification document. Upon successful authentication, the election authority agent with similar privilege as the collector could ask the outer vote codes of the voter's ballot, without necessarily learning which code is associated with which candidate choice. Then the collector could decrypt the vote codes to inner vote codes to find whether one of them is posted on the bulletin board. If one was found, the election authority agent could then ask of the physically authenticated voter which vote code to decrypt and overwrite the already published inner vote code. As the bulletin board is append-only, such overwriting would take the shape of a standard message indicating which inner vote code to overwrite, signed by the election authority agent's signature. Upon receiving the receipt code from the bulletin board, the voter could verify that the election authority agent overwrote the previous vote using the vote code intended. The tallier would be instructed to ignore overwritten vote codes.

4.8 Summary of findings

Based on the protocol description in Chapter 3 and discussion of threats and mitigating actions above, we argue that the proposed protocol comes far in providing sufficient security for a voting system. There is no reliance on client side integrity and client malware is avoided as a potential vulnerability by use of vote codes. The anonymous vote codes together with receipt codes support anonymous voting. Furthermore, since only the initiator or other distributor of ballots will know the linking of credentials to voter identities, the unlikely successful attempt at associating candidate choices with credentials will not unveil voter identities and compromise

anonymity. An appropriately set ballot code length provides a high level of security against guessing and tampering attacks while maintaining usability from the voter's point of view. Distributing the responsibility and trust in the election authority among several agents, no one agent but the trusted initiator has sufficient power to alter the outcome of an election without detection.

On the other hand, some of the assumptions introduced in Chapter 3 prove hard to justify in full and suggest further work is needed on the proposed protocol before it is ready to face the expected requirements of a full-scale democratic election. Most noteworthy, the unconditional trust in the election authority initiator may not hold in a voting population. We do, however, believe that appropriate extensions to the protocol can be made for it to sufficiently comfort both voters and regulators. The predominant mitigating actions are those of distributed entities and threshold encryption.

Chapter 5

Implementation

To demonstrate the practical feasibility of the protocol proposed in Chapter 3, a simulator of a voting system using the protocol was implemented. In this chapter, we describe the implementation of the simulator and demonstrate the protocol's behaviour in a variety of election scenarios. First, we give a high-level overview of the software architecture. Next, we elaborate on the implementation details before simulator usage is demonstrated.

5.1 Software architecture

An illustrative overview of the software architecture is shown in Figure 5.1. The architecture closely follows the design described in Chapter 3. For all cryptographic primitives, including encryption, hashing and signatures, the open source cryptography library *OpenSSL* [49] was used. All agents are modelled as separate objects with their own methods for interacting.

The simulator takes as input the number of voters, proportion of voters that are dishonest, number of adversaries, number of candidates, length of AES keys, length of RSA keys, ballot code length, length of underlying cryptographic codes, the behaviour of the collector and behaviour of the bulletin board. Based on these inputs, the initiator initiates the election by setting up the other election authority agents according to input. Next the simulator haves both voters and any adversaries submitting their votes sequentially. Should a voter be configured as dishonest, she tries to resubmit her vote in an attempt to vote twice. Adversaries vote by pseudo-randomly

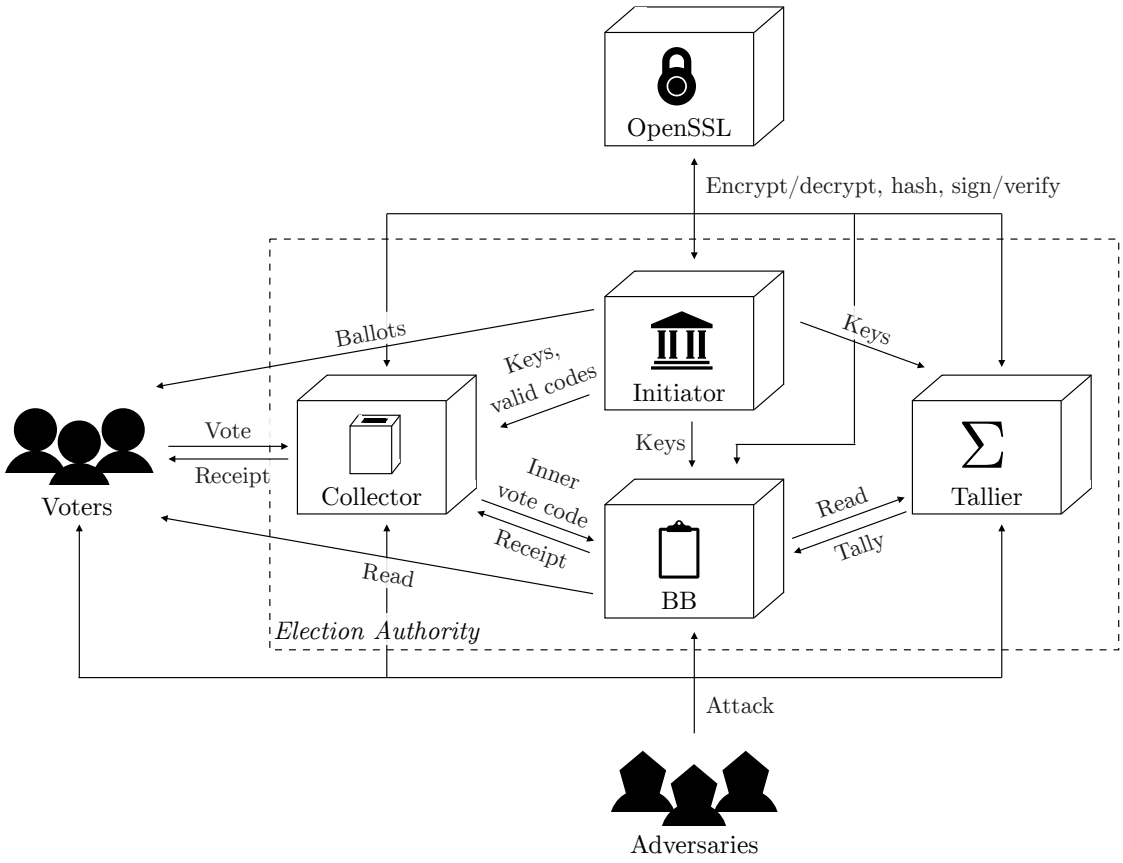


Figure 5.1: Software architecture.

guessing a pair of valid credential and vote code, and do not consider the receipt code returned. The collector reports any illegitimate credential or vote code submitted to it. It also report whenever it fails to publish inner vote codes on the bulletin board. The collector may, however, be configured to behave dishonestly, either by dropping votes altogether or attempting to tamper with them. Upon receiving a receipt code from the collector, a voter compares it to the receipt codes on her ballot. Like the initiator, voter and collector, the bulletin board too reports any unexpected behaviour, including failure of writers to produce valid signatures, hashes or timestamps for the messages they attempt to publish. The bulletin board may, like the collector, be configured to behave dishonestly. In the case of the bulletin board, this is done by attempts to modify the history of bulletin board entries.

After all voters and adversaries have voted, the initiator publishes the inner vote code on the bulletin board. Then the tallier reads the entire bulletin board history, verifies its consistency with regards to hashes, timestamps and signatures, and extracts the

inner vote key. This is used to sequentially decrypt all posted inner vote codes and report the sum of votes for each candidate. Should the tallier find an inconsistency or a mismatch between the number of voters and the number of votes collected, the error is reported.

5.2 Implementation details

All simulator source code is provided in the code repository accompanying this report. Although we refer readers with a particular interest in the finest details of the implementation to the source code documentation[1], we discuss the most noteworthy features here.

The simulator was implemented in the object oriented paradigm using the low-level, high performance language of C++. C++ was chosen for its well acknowledged performance and programmer control. As mentioned in Section 5.1, OpenSSL was used for handling cryptographic primitives. The cryptographic primitives employed in the simulator are AES-128[2], RSA-2048 and SHA-256 for symmetric encryption, asymmetric encryption and hashing respectively. These primitives are, at the time of writing, considered sufficiently secure and used to a large extent [50]. The effect of altering the key lengths for AES and RSA is investigated in Chapter 6.

The implementation consists of six main classes: `Initiator`, `Collector`, `BulletinBoard`, `Tallier`, `Voter` and `Adversary`. The first four classes represent the election authority and the latter two are derived classes of an abstract `Client` class. A simplified overview of these classes' key data members and functions is given in Figure 5.2, while the complete Unified Modelling Language (UML) diagram of the simulator is provided in Figure 5.3.

Interaction between the simulator agents is carried out by use of pointers and public member functions. Interfacing with the OpenSSL library is done using of a collection of helper functions, available to all election authority agents. The shortest helper function, used for hashing, is shown in Listing 5.1 as an example of interfacing with the OpenSSL library.

Feedback from the simulator, including reports from the voters, adversaries and elec-

[1] A documentation PDF is found in */docs/docs.pdf*.
[2] With Cipher Blocker Chaining (CBC).

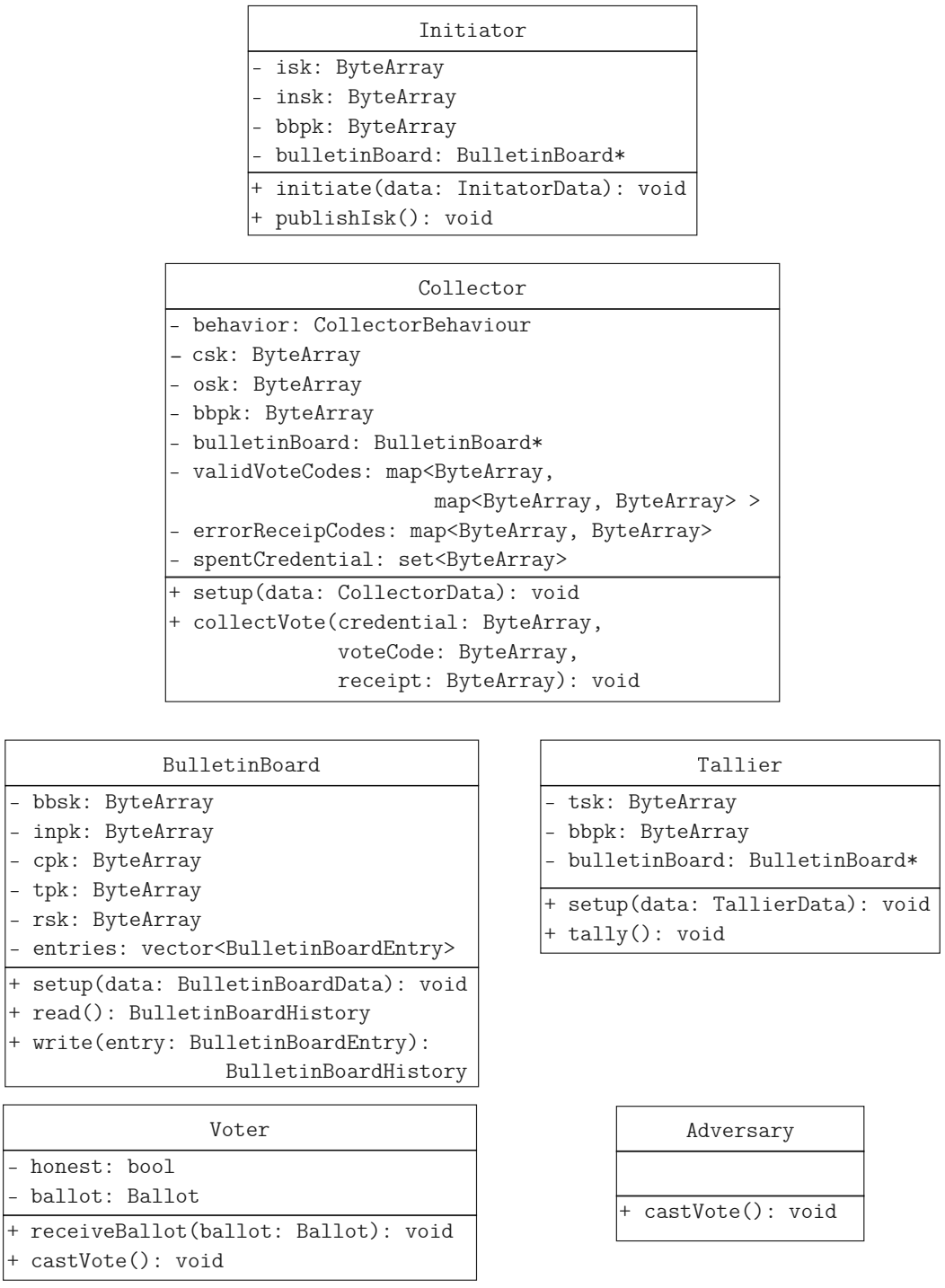


Figure 5.2: Simplified overview of main simulator classes' key data members and functions. We refer to Figure 5.3 for a complete UML diagram.

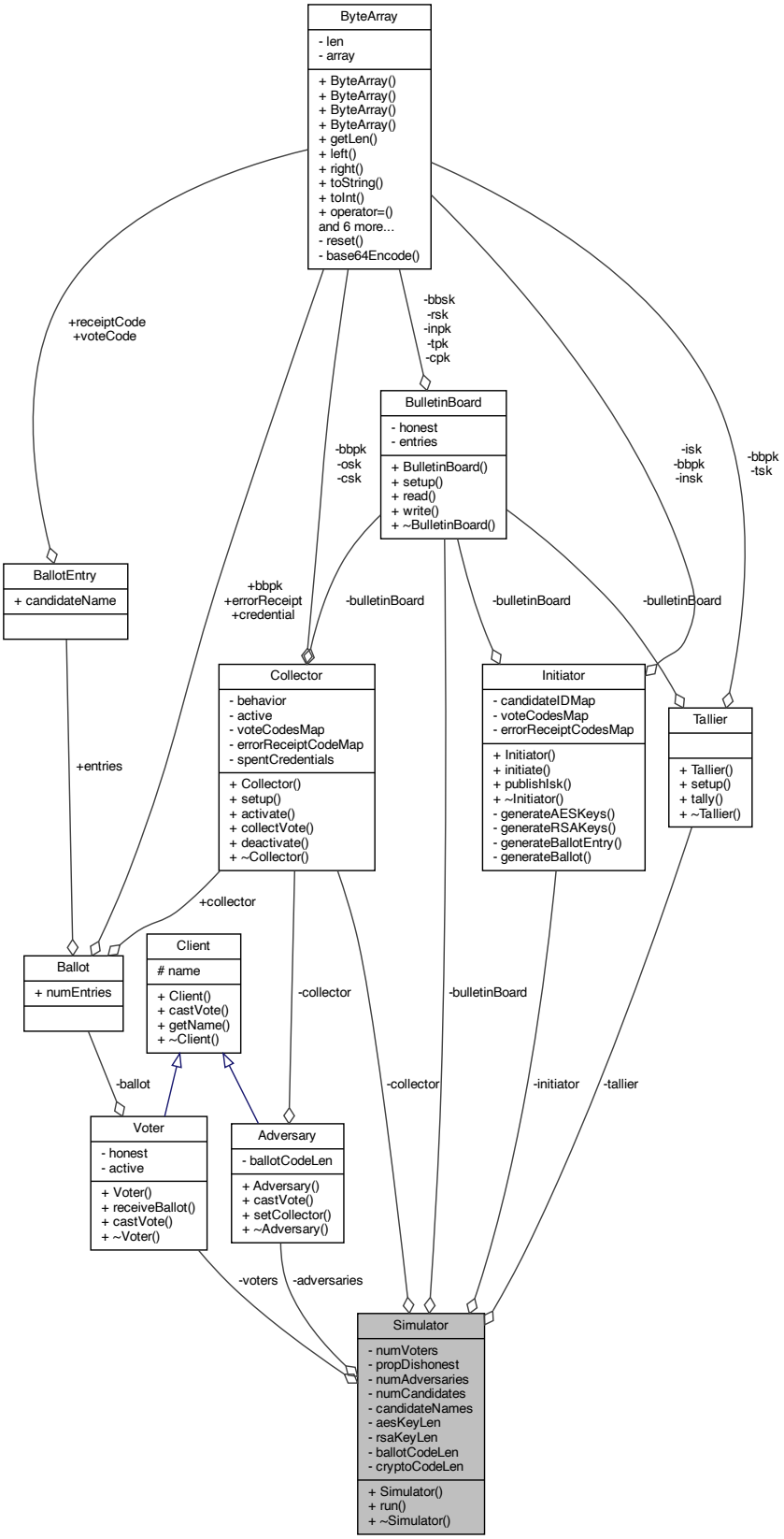


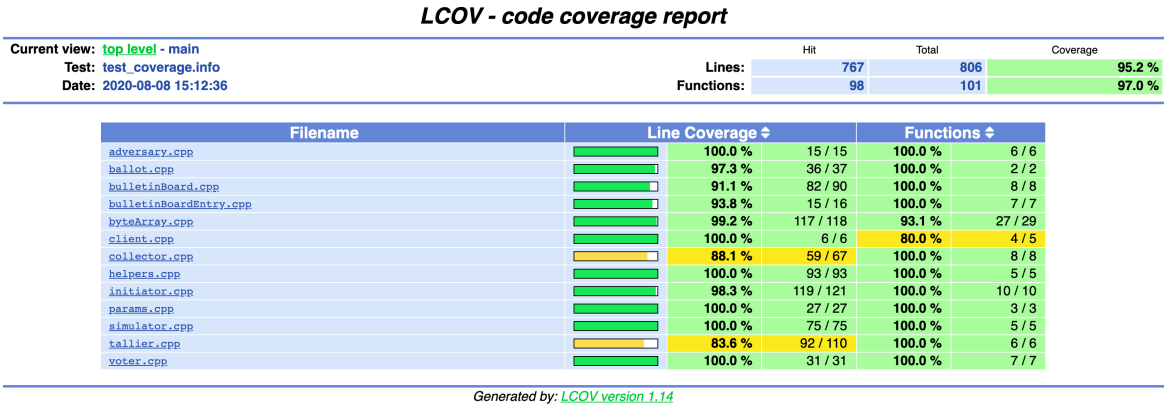
Figure 5.3: UML diagram of protocol simulator implementation.

Listing 5.1: Example helper function for OpenSSL interfacing

```
ByteArray hashMsg(const ByteArray& msg) {
  EVP_MD_CTX* ctx = EVP_MD_CTX_new();
  assert(EVP_DigestInit_ex(ctx, EVP_sha256(), NULL));
  assert(EVP_DigestUpdate(ctx, msg, msg.getLen()));
  ByteArray digest = ByteArray(SHA256_DIGEST_LENGTH);
  unsigned int len = digest.getLen();
  assert(EVP_DigestFinal_ex(ctx, digest, &len));
  EVP_MD_CTX_free(ctx);
  return digest;
}
```

tion authority agents, is produced in the shape of written logs. The logging is configured to output messages in a hierarchy, with reports of misbehaviour outputted as warnings or errors, and bulletin board publications and ballot generations as debug messages.

To aid software development and provide some insight into the functionality and robustness of the simulator, a suite of automated unit tests was developed. The details of the tests as well as the code coverage report may be found in the source code repository[3]. A summarising table is provided in Figure 5.4.



LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level - main	Lines:	767	806	95.2 %
Test:	test_coverage.info	Functions:	98	101	97.0 %
Date:	2020-08-08 15:12:36				

Filename	Line Coverage		Functions	
adversary.cpp	100.0 %	15 / 15	100.0 %	6 / 6
ballot.cpp	97.3 %	36 / 37	100.0 %	2 / 2
bulletinBoard.cpp	91.1 %	82 / 90	100.0 %	8 / 8
bulletinBoardEntry.cpp	93.8 %	15 / 16	100.0 %	7 / 7
byteArray.cpp	99.2 %	117 / 118	93.1 %	27 / 29
client.cpp	100.0 %	6 / 6	80.0 %	4 / 5
collector.cpp	88.1 %	59 / 67	100.0 %	8 / 8
helpers.cpp	100.0 %	93 / 93	100.0 %	5 / 5
initiator.cpp	98.3 %	119 / 121	100.0 %	10 / 10
params.cpp	100.0 %	27 / 27	100.0 %	3 / 3
simulator.cpp	100.0 %	75 / 75	100.0 %	5 / 5
tallier.cpp	83.6 %	92 / 110	100.0 %	6 / 6
voter.cpp	100.0 %	31 / 31	100.0 %	7 / 7

Generated by: LCOV version 1.14

Figure 5.4: Test code coverage. Overall line and function coverage is above 95% with the lowest individual class coverage at 80%.

[3]The complete code coverage report is found in */coverage/index.html*.

5.3 Usage

In the following, we demonstrate the usage of the simulator using a miniature scale population of two voters and two candidates. These unnaturally small scales are used in the interest of short logs and to be able to view the variety of feedback from the simulator. Only for the ideal scenario is the full log shown. For other scenarios, the areas of interest are extracted. The full logs are appended in Appendix B for reference. In Chapter 6, population sizes more in line with what is expected for democratic elections are considered and so are different security parameters. As stated again in Chapter 6, full logs for realistic population sizes are available online[4].

Figure 5.5 shows the behaviour of the simulator with honest behaviour from all election authority agents and voters and complete absence of adversaries. From the logs one can read how the initiator publishes on the bulletin board the public keys of all election authority agents, the mapping of candidate IDs to candidate names and the total number of voters. It is also clear how the initiator creates and distributes ballots to the voters. Upon completing initialisation, the logs show how inner vote codes are published on the bulletin board upon decryption by the collector. At last, the initiator marks the end of vote collection and beginning of tallying by publishing the inner vote code key on the bulletin board. This invites the tallier to decrypt the inner vote codes, which it does, before publishing the count of votes for each candidate.

[4]All log files are available to people associated with Imperial College at `https://imperiallondon-my.sharepoint.com/:f:/g/personal/ssc19_ic_ac_uk/EtlKdpkWNzVOi5eX_t7hpWoBB9b7uk8Ry8HGgH9cBBhN_w?e=QJUAca`

```
18:25:45.506 [info] Simulator setup begun
18:25:45.508 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 0, "numAdversaries": 0, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 0,
"bulletinBoardHonesty": 1, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597163145}
18:25:45.508 [trace] Created Candidate 0
18:25:45.509 [trace] Created Candidate 1
18:25:45.509 [trace] Created Voter 0 (honest)
18:25:45.509 [trace] Created Voter 1 (honest)
18:25:45.509 [info] Simulator setup completed
18:25:45.509 [info] Initiation begun
18:25:45.510 [trace] Generated AES encryption keys
18:25:45.875 [trace] Generated RSA key pairs
18:25:45.879 [trace] Setup bulletin board
18:25:45.884 [debug] Added BB entry: <bbpk, LS0t..., icYy..., Initiator, YWXb..., QJdm..., icYy..., l5Pm...>
18:25:45.891 [debug] Added BB entry: <inpk, LS0t..., icYy..., Initiator, eE5H..., ZE6H..., icYy..., oFFU...>
18:25:45.908 [debug] Added BB entry: <Candidate 0, AA..., icYy..., Initiator, Jevv..., h64u..., icYy..., yNMe...>
18:25:45.916 [debug] Added BB entry: <Candidate 1, AQ..., icYy..., Initiator, JJgI..., VgpY..., icYy..., oeal...>
18:25:45.918 [debug] Made ballot with credential 2Wm+rxZr1w
------------------------------------------------
| Credential         | 2Wm+rxZr1w                |
| Error receipt code | YPb8SNNAfQ                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | kV6Y29mikA | z0w6eAPtQw   |
|        Candidate 1 | obSePSP/iQ | SG8UDVEkqQ   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:25:45.918 [debug] Made ballot with credential REA52xAMwQ
------------------------------------------------
| Credential         | REA52xAMwQ                |
| Error receipt code | u6258fi9MQ                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | 47goT/8aOw | y8GZKXGf3w   |
|        Candidate 1 | ePNIMT1gtw | i49HCpn95w   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:25:45.924 [debug] Added BB entry: <#Voters, AgAA..., icYy..., Initiator, dJnB..., a6PP..., icYy..., bI99...>
18:25:45.927 [trace] Setup collector
18:25:45.927 [trace] Setup tallier
18:25:45.933 [debug] Added BB entry: <cpk, LS0t..., icYy..., Initiator, J3Bf..., BFiR..., icYy..., C1TU...>
18:25:45.941 [debug] Added BB entry: <tpk, LS0t..., icYy..., Initiator, X1KV..., n6NT..., icYy..., feMI...>
18:25:45.944 [trace] Voter 0 (honest) received ballot with credentials: 2Wm+rxZr1w
18:25:45.944 [trace] Voter 1 (honest) received ballot with credentials: REA52xAMwQ
18:25:45.944 [trace] Distributed all ballots to voters
18:25:45.944 [info] Initiation completed
18:25:45.944 [info] Collection begun
18:25:45.944 [trace] Collector activated
18:25:45.944 [trace] Voter 0 (honest) chose vote code: kV6Y29mikA
18:25:45.944 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:25:45.950 [debug] Added BB entry: <Inner vote code, H5b5..., icYy..., Collector, ei72..., n3dY..., icYy..., C5r0...>
18:25:45.952 [trace] Voter 0 (honest) received expected receipt code: z0w6eAPtQw
18:25:45.952 [trace] Voter 1 (honest) chose vote code: 47goT/8aOw
18:25:45.952 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:25:45.958 [debug] Added BB entry: <Inner vote code, 58nx..., icYy..., Collector, zeQl..., U1Sy..., icYy..., PRg/...>
18:25:45.960 [trace] Voter 1 (honest) received expected receipt code: y8GZKXGf3w
18:25:45.960 [info] Collection completed
18:25:45.960 [info] Tallying begun
18:25:45.965 [debug] Added BB entry: <isk, MOBT..., icYy..., Initiator, /2g+..., bKSb..., icYy..., lwCa...>
18:25:45.967 [trace] Initiator published isk
18:25:45.971 [trace] Tallier read bulletin board history
18:25:45.979 [debug] Added BB entry: <Candidate ID (padded), AO48..., icYy..., Tallier, csm2..., TUTR..., icYy..., vrIH...>
18:25:45.987 [debug] Added BB entry: <Candidate ID (padded), AKXw..., icYy..., Tallier, MlKB..., gXaW..., icYy..., aWnY...>
18:25:45.997 [debug] Added BB entry: <Candidate 0: 2, AA..., icYy..., Tallier, d/V/..., snyE..., icYy..., DX0i...>
18:25:46.007 [debug] Added BB entry: <Candidate 1: 0, AQ..., isYy..., Tallier, iFYm..., l5kh..., isYy..., fYHb...>
18:25:46.011 [info] Tallying completed
```

Figure 5.5: Simulator logs for 2 honest voters, 2 candidates, no adversaries and correctly behaving election authority agents. Full logs provided in Appendix B.

As shown in Figure 5.6, a dishonest voter ("Voter 0") is allowed to submit one vote, but is denied in its attempt to submit a second vote. This is detected and reported by the collector.

```
18:33:10.404 [trace] Voter 0 (dishonest) chose vote code: XPnyHnnECg
18:33:10.404 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:33:10.410 [debug] Added BB entry: <Inner vote code, Ub7m..., Rsgy..., Collector, 9Y3o..., F4Ag..., Rsgy..., Gw4J...>
18:33:10.411 [trace] Voter 0 (dishonest) received expected receipt code: EY9HI6mStQ
18:33:10.411 [warning] Credential 3QDnEpOOEQ already voted
18:33:10.411 [warning] Voter 0 (dishonest) received inexpected receipt code: hRblAyXLrg
```

Figure 5.6: Extract from simulator logs with a dishonest voter. Full logs provided in Appendix B.

Somewhat similar to the case of a dishonest vote, the attempt of an adversary to vote is detected by the collector and reported, as shown in Figure 5.7.

```
18:35:48.828 [warning] Collector did not find voter credential: GzTvyVJgog
18:35:48.828 [trace] Adversary 0 attempted to guess a vote and received receipt code: RDiwZCXF6A
```

Figure 5.7: Extract from simulator logs with an adversary. Full logs provided in Appendix B.

Figure 5.8 shows the logs with an dishonest collector. As seen from Figure 5.8a, a voter will detect and report an unexpected receipt received from a collector who has tampered with the vote submitted.

```
18:39:17.658 [trace] Voter 1 (honest) chose vote code: s41GOUoDMg
18:39:17.658 [warning] Dishonest collector tampered with vote from credential: sZSH17IaKA
18:39:17.664 [debug] Added BB entry: <Inner vote code, QSrG..., tcky..., Collector, VlgS..., fTz4..., tcky..., fWBy...>
18:39:17.666 [warning] Voter 1 (honest) received inexpected receipt code: ROXaK81sXQ
```

(a) Vote tampering collector

```
18:37:23.322 [warning] Dishonest collector dropped vote from credential: hTEe19wz8g
18:37:23.322 [warning] Voter 1 (honest) received inexpected receipt code: AAAAAAAAAA
18:37:23.322 [info] Collection completed
18:37:23.322 [info] Tallying begun
18:37:23.328 [debug] Added BB entry: <isk, xWxE..., Q8ky..., Initiator, X6z8..., Rx/3..., Q8ky..., G7jU...>
18:37:23.331 [trace] Initiator published isk
18:37:23.333 [trace] Tallier read bulletin board history
18:37:23.342 [error] Tallier counted unexpected number of votes
```

(b) Vote dropping collector

Figure 5.8: Extract from simulator logs with a dishonest collector. Full logs provided in Appendix B.

Should the collector drop a vote, the tallier will also detect the misbehaviour of a vote dropping collector. The voter will still detect the misbehaviour by an unexpected

```
18:40:59.315 [trace] Tallier read bulletin board history
18:40:59.316 [error] Tallier read inconsistent history (hash)
18:40:59.316 [info] Tallying completed
```

Figure 5.9: Extract from simulator logs with a dishonest bulletin board. Full logss provided in Appendix B.

receipt code, while the tallier finds a mismatch in the expected number of votes and the inner vote codes published on the bulletin board. This is shown in Figure 5.8b.

Lastly, a dishonest bulletin board, for example misbehaving by modifying the history of entries, will be detected as shown in Figure 5.9. In this concrete example, the tallier finds an inconsistent hash in the chain of messages. Other inconsistencies could, resulting from the bulletin board design (see Chapter 3), be invalid signatures or timestamps.

Chapter 6

Evaluation

Following the description of the implemented simulator (Chapter 5) to demonstrate the protocol (Chapter 3), we here evaluate the practical feasibility of the protocol. First, we analyse the quantitative results obtained from the simulator before relating these to the requirements of online voting systems introduced in Chapter 2 and making a qualitative assessment.

6.1 Simulation results

To evaluate the practical feasibility of the proposed protocol in terms of processing time, important to security by voter verification and usability [35], the simulator described in Chapter 5 was timed over a variety of parameter combinations. Not only was the total time of interest, but also the distribution of computational effort and frequency of processing votes. The distribution of time spent may provide insight to time sensitive parts of the protocol, whereas the frequency of vote processing is important to usability, as discussed in Chapter 2.

In total, 68 simulations with different simulator configurations were run on a 2.6 GHz Dual-Core Intel Core i5 MacBook Pro[1] running macOS Catalina[2]. Based on a base case configuration as described in Table 6.1, sets of parameter values were evaluated, one parameter at a time. The set of parameter values tested are listed in Table 6.2. Run time analyses with misbehaving collector and bulletin board were

[1] Mid 2014
[2] Version 10.15.4

Table 6.1: Simulator base case parameter values.

Parameter	Value
Number of voters*	{100, 1000, 10000, 100000}
Dishonest proportion of voters	0
Number of adversaries	0
Number of candidates	10
AES key length	128 bits
RSA key length	2048 bits
Cryptographic code length	128 bits

*The set of number of voters was always tested to investigate scalability.

not conducted as such misbehaviour only leads to an inconclusive election outcome and has an uninteresting impact on run time. Should the collector misbehave, the simulated voter will simply receive an unexpected receipt code and experience no delay. Should the bulletin board misbehave, the tallier will give up tallying the moment the bulletin board history inconsistency is detected, thus reducing run time but with an inconclusive outcome. The ballot code length was neither investigated as it does not affect computations other than the comparison to full outer vote codes performed by the collector.

Table 6.2: Sets of parameter values tested in simulations.

Parameter	Set of values
Number of voters	{100, 1000, 10000, 100000}
Dishonest proportion of voters	{0, 10, 20, 30} (%)
Number of adversaries	{0, 25, 50, 75, 100} (%)
Number of candidates	{5, 10, 15, 20}
AES key length	{128, 256} (bits)
RSA key length	{512, 1024, 2048, 4096} (bits)
Cryptographic code length	{128, 256, 512} (bits)

In the following, we discuss the main findings from the simulation results. The results of all simulator runs may be accessed via the complete log files available

online[3].

In Figure 6.1, the distribution of time across different numbers of voters is shown. As expected, the total time seems to increase linearly with the number of voters. This is expected based on the fact that each voter adds to the protocol an approximately constant marginal cost in terms of processing time. Lookup operations carried out by the collector for each voter do increase slightly, but only by a logarithmic factor. The initiation phase is found to take an order of magnitude less time than the collection and tallying phases, which take similar time. In our discussion of the RSA key length, we come back to this particular result.

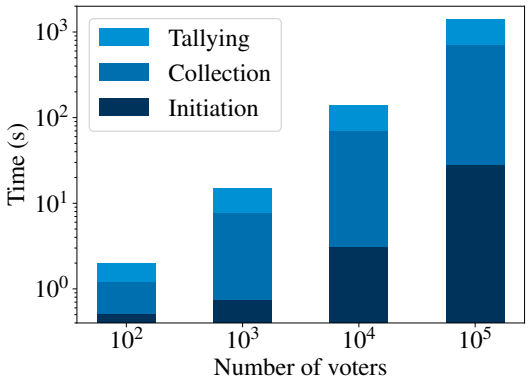


Figure 6.1: Distribution of time for different voter population sizes. Note that both axes are logarithmic.

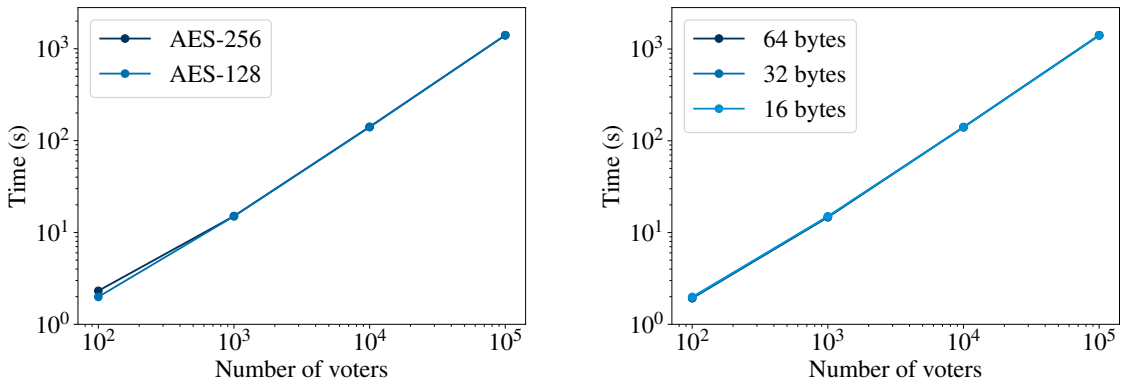
One may wonder whether the highest numbers of voters tested, 100,000, is sufficient to comment on the feasibility for real world elections where the population of voters may easily count several hundred millions. We argue that the analysis of different orders of magnitudes for both number of voters and candidates, as well as dishonest voters and adversaries, gives sound basis to make predictions about the performance on larger populations. It is also worth stressing that the results presented here are based on simulations on a single-threaded program. Should the system be deployed to larger populations, a first step to enhance efficiency would be to implement multi-threaded vote collection to handle concurrent vote submissions from voters.

The impact on total time of varying the AES key length and cryptographic code

[3]All log files are available to people associated with Imperial College at `https://imperiallondon-my.sharepoint.com/:f:/g/personal/ssc19_ic_ac_uk/EtlKdpkWNzVOi5eX_t7hpWoBB9b7uk8Ry8HGgH9cBBhN_w?e=QJUAca`. Should you be unable to access the files, consider the tabulated metrics provided in Appendix C or contact us at `ssc19@ic.ac.uk`.

length is shown in Figure 6.2. As could be expected from the fast and secure Advanced Encryption Standard, increasing either the key length or the plaintext length to be encrypted makes no significant difference on the time spent. We reason that the AES operations are relatively inexpensive in terms of computation time compared to other more costly operations like RSA encryption.

One could based on these results consider making both the AES key length and cryptographic code length longer, as they would both increase the difficulty of breaking the cryptography of full outer vote codes, inner vote codes and padded candidate IDs. However, such longer codes and keys can only do so much as the security of the protocol ultimately also depends on the length of the codes printed on ballots. Regardless of the cryptographic code length and AES key length, an adversary could attack the system by guessing the codes printed on a ballot. We thus argue that an AES key length of 256 bits does make sense, but that stretching the cryptographic code length too far will not make significant difference to security but add the memory consumption of the system.



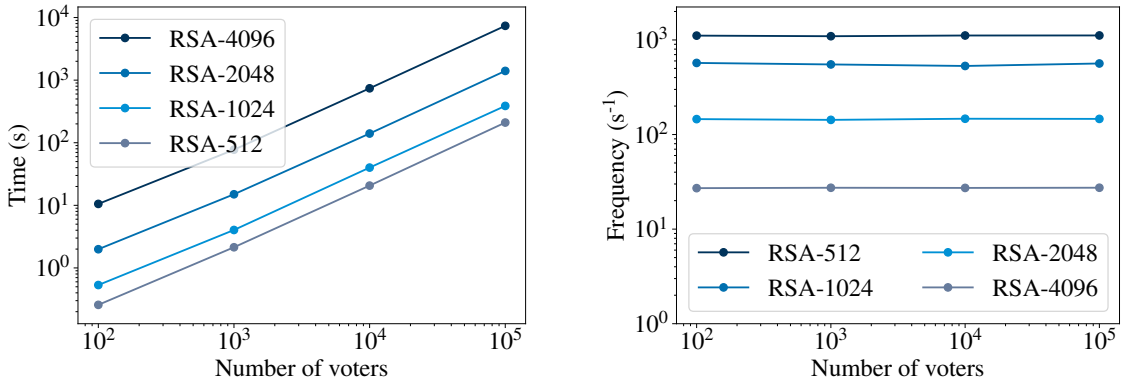
(a) AES key length

(b) Cryptographic code length

Figure 6.2: Impact of AES key length and cryptographic code length on total time. Lines that coincide will only have one of the coinciding lines visible. Note that both axes are logarithmic.

Following the above hypothesis that the RSA key length has a bigger impact on total time, the effect may be analysed by considering Figure 6.3.

From considering both the total time and vote processing frequency, it becomes evident that the RSA key length has a substantial impact on the protocol's performance. This is not unexpected since asymmetric encryption schemes, and in this case the



(a) Total time

(b) Vote processing frequency

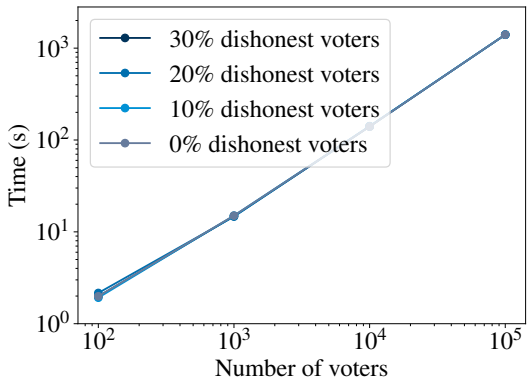
Figure 6.3: Impact of RSA key length on total time and vote processing frequency. The vote processing frequency is computed as the fraction of total time in the vote collection phase over the number of votes submitted. Note that all axes are logarithmic.

RSA algorithm, are known to be more tedious than their symmetric colleagues. Furthermore, RSA encryption is involved with every bulletin board write and so is used extensively. Nevertheless, with the base case RSA key length of 2048 bits, close to 150 votes are processed every second. Should an even more secure key length of 4096 bits be required, nearly 30 votes can still be processed every second by a single processor.

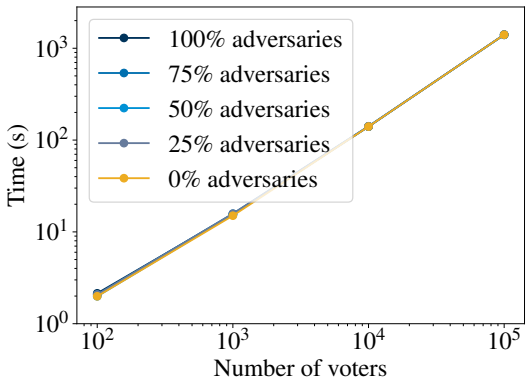
The impact of RSA encryption on total time also helps explain the relatively quick initiation phase compared to the collection and tallying phases. Asymmetric encryption is used with every bulletin board publication, of which there are few during initiation, but several (one for every vote decrypted) during collection and similarly so for tallying.

As can be seen in Figure 6.4, the impact of dishonest voters, adversaries and number of candidates on total time is negligible. We believe the explanation for this to be that all three parameters primarily affect inexpensive lookup operations. Dishonest voters are turned away by the collector once the collector finds that they have already voted. Adversaries that are unsuccessful in guessing valid credential and vote code pairs are likewise denied after lookup operations, without needing to carry out more expensive cryptographic operations.

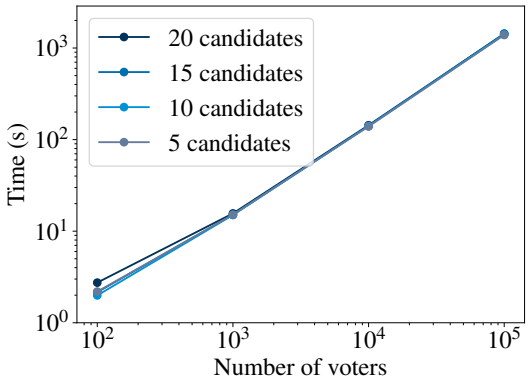
Regardless of the cause, it is encouraging to find that the protocol, with respect to performance, seems to scale without notable issues with dishonest voters, adver-



(a) Proportion of dishonest voters



(b) Number of adversaries relative to voters



(c) Number of candidates

Figure 6.4: Impact of proportion of dishonest voters, number of adversaries and number of candidates on total time. Percentage figures shown are with respect to the number of voters (e.g. 100% adversaries means there are as many adversaries as there are voters). Lines that coincide will only have one of the coinciding lines visible. Note that both axes are logarithmic.

saries or large numbers of candidates.

Before ending our discussion of the simulator results, it is worth noting that we have not investigated the impact of network delays on the protocol's performance. We do, however, reason that the required network traffic needed to carry out the protocol communication is relatively lightweight. Communication is primarily comprised of credentials, vote codes, cryptographic codes, hashes and signatures thereof that are of controllable size, especially noting that security is inevitably limited by the length of credentials, vote codes and receipt codes printed on ballots. This is the foremost trade-off between usability and security.

6.2 Satisfaction of requirements

Recall from Table 2.1 in Chapter 2 that no existing online voting system, except the Estonian system [40] which has been criticised for other vulnerabilities, satisfied the entire collection of requirements for online voting system. In Table 6.3, we have presented our assessment of the proposed protocol's satisfaction of those same requirements.

Table 6.3: Satisfaction of the requirements for online voting systems by the proposed protocol.

Requirement	Satisfaction
E2E verifiability	✓
Voter authorisation and authentication	✓
Receipt freedom	✓
Secret ballot	✓
Usability and accessibility	✓
Software independence	✓
Coercion resistance	-

Adopting the threat model assumptions of Chapter 3, we claim that the protocol we propose is E2E verifiable in that a voter may verify that her vote is published on the bulletin board, and anyone can verify that all votes published on the bulletin board are included in the final tally.

Though we have not explicitly addressed authentication and authorisation, the protocol does require voters to authenticate themselves by secure credentials and only then authorises them to cast a vote. Assuming the election authority has some secure way of authenticating voters, for instance making use of the method for traditional pollsite voter authentication, we thereby avoid both the need for biometric authentication or the like and the need to handle such sensitive privacy data. All that is needed for a voter to be authenticated is the secure distribution of ballots to eligible voters and the voter submitting the credential printed on her ballot.

Receipt freedom is assured as long as the collector does not unveil associations between credentials and inner vote codes. Similarly, as long as the assumed honest initiator keeps keys secret, ballots are secret too. Like Bernhard et al. [12], we argue that software independence is achieved through E2E verifiability, as any attempt to

manipulate the election will be detectable by either the voter or someone else, under the assumptions stated. This was demonstrated in Chapter 5 with several different software agents misbehaving.

We have proposed measures to mitigate coercion, but have not gone far enough in describing the details and considering all potential implications for the protocol. In Chapter 4, we have, however, outlined an alternative route for vote submission that could allow coerced voters to overwrite their votes.

Usability while providing E2E verifiability is arguably the requirement fulfilled to the greatest extent. Voters need only input two 10 familiar characters long codes to submit a vote, and can promptly verify their vote has been received by inspecting a receipt code of equal length. The protocol does not require any trust in the client device, and could as such be implemented as a simple web form, a smartphone app or desktop application.

We note that we have not addressed accessibility explicitly in this report, but trust instead that the use of short credential and vote codes is simple enough from the voter's point of view that effective solutions for interfacing with accessibility systems are within reach.

Chapter 7

Conclusion

In this project we have considered the intricate issue of secure online voting. Since the 1980s, the interest in electronic voting has grown rapidly. More recently, remote online voting systems have received special attention with the incentives for improved remote voting ranging from increased flexibility and participation, to administrative cost savings, to robustness against pandemics like the most recent COVID-19 pandemic. However, the systems proposed so far have not provided sufficient comfort in their security. In particular, expectations of end-to-end verifiability balanced with requirement of voter anonymity is a challenge not yet overcome. Furthermore, usability, of utmost importance for a voting system to be practically viable, is often sacrificed.

In this chapter, we summarise our achievements and outline some areas of future work.

7.1 Summary of achievements

The contribution of this project is two fold. First, we have designed a new proof of concept secure online voting protocol and shown by security analysis that it can withstand attacks typically faced by voting systems. Second, we have demonstrated the protocol in the shape of a voting system simulator and shown that its performance is beyond satisfactory with today's security parameters. In the following, we summarise our achievements.

- We have proposed a *code voting* protocol, inspired by the work of Ryan and

Teague [1] and Chondros et al. [2] with original improvements to enhance usability and strengthen system integrity in the face of adversary attacks. In the proposed protocol, voters receive unique, cryptographic ballots prior to an election and vote by submitting two codes, a credential and a vote code. They may verify that their vote is registered as cast by comparing a returned receipt code with one printed on their ballot. After polls close, any voter may verify that votes are counted correctly via a public record of all anonymous votes, thereby verifying votes are counted as registered.

The protocol sets itself apart from existing systems by a new distribution of election authority responsibilities and trust. Similar to the protocol of Chondros et al. [2], the responsibility of generating ballots and setting up cryptographic keys is separated from the collection of collection and processing of votes. In our protocol, this is handled by an *initiator* and causes the voter, as viewed by all other election authority agents, to be anonymous, authenticated only with a pseudo-random credential. What distinguishes the protocol is that only the vote *collector* learns the vote code submitted by any credential, and is only able to peal off an outer layer of encryption and forward an inner layer encryption onto a public append-only *bulletin board*. The bulletin board is needed for issuing of the receipt, expected by the voter for verification of successful vote submission, and does not learn which credential submitted the vote corresponding to the inner layer encrypted codes posted on the bulletin board. Eventually, inner layer encrypted codes posted on the bulletin board are decrypted by a fourth election authority, the *tallier*. This final exercise is open to universal verifiability since the decryption key used to infer candidate choices from inner layer encrypted vote codes is published on the bulletin board after vote collection terminates and tallying begins. This elegant division of trust and responsibility, with election authority agents reporting on any potential misbehaviour of others, gives the voter-friendly usable and secure protocol.

- Through a security analysis of the proposed protocol, we have addressed threats common to online voting systems in general, and the proposed system in particular, and reasoned about their likelihood and potential impact. To those threats not fully protected against, we have suggested mitigating actions, most of which require modest modifications to the protocol.

The proposed protocol offers strong security against client side malware, as there is no underlying assumption about the integrity of the voter's device. Vote codes only known to the voter via encrypted ballots ensure the voter's device has no opportunity to infer, alter or drop votes without the voter noticing.

Under the assumption of complete trust in the election authority initiator who generates and distributed ballots, voter anonymity is guaranteed, since no entity other than the initiator and the voter knows which voter credential, printed on the ballot, is linked to which voter.

The integrity of the protocol does nevertheless rely on a certain degree of trust in the election authority. We have showed that a compromised initiator or colluding collector and bulletin board can destroy the security offered by division of trust and responsibilities among election authority agents.

Should the risk of initiator compromise or collector and bulletin board collusion be deemed likely, we suggest the implementation of distributed election authority agents. Rather than one entity generating cryptographic keys and ballots and distributing, the task can, via distributed key generation and threshold cryptography, be divided among several entities, of which a threshold number of honest behaving entities is required to successfully carry out the task.

- We have demonstrated the protocol by implementing it as part of a voting system simulator and its performance has been evaluated. The simulator provides a high degree of flexibility to investigate the effect of various election and security parameters. To name a few, the voter population size and composition, number of candidates and cryptographic key lengths are all examples of variable parameters.

  To investigate the effect of different parameter combinations, we have run a total of 68 simulations, yielding a rich set of results. Using conventional key lengths, as of the time of writing, several hundred votes are found to be processed by a single processor every second. This rate of vote processing is found to be constant across population sizes, and furthermore does not depend significantly on the security parameters for symmetric encryption, nor the number of candidates or presence of adversaries. What seems to be the limiting factor in the protocol, is the slower asymmetric encryption used in signing and verifying

bulletin board entries.

Total processing time scales linearly in the number of voters, and although modest population sizes are tested, we argue that the findings provide a sound basis for expecting similar performance for larger populations.

- Lastly, we have evaluated the protocol in light of a set of common requirements for online voting systems. Apart from coercion resistance, which is arguably an particularly intricate matter for remote voting, the protocol does well with respect to all requirements. For better coercion resistance, we propose an alternative route of voting, using the same protocol but with the ability to overwrite votes upon physical authentication of voters.

  The protocol provides end-to-end verifiability under the assumptions of the threat model. Voters can individually verify that their votes are cast as intended and recorded as cast, and anyone can universally verify that the votes that are recorded, are tallied as recorded. As such, software independence is also achieved.

  By the use of code voting and distribution of knowledge amongst election authority agents, secret ballots and receipt freedom has been achieved. Voters are easily authenticated using anonymous credentials, avoiding the need for processing of privacy data.

  In terms of usability, we argue that the protocol is superior and only requires the voter to handle three 10 familiar characters long codes for vote submission and verification. We have also argued that the protocol may likely be extended to allow accommodation to any accessibility needs.

Together, these achievements form the foundation for the further development and deployment of a new online voting system to be used in democratic elections.

7.2 Future work

In the following we propose and discuss several interesting areas of future work.

- As is clearly pointed out in the threat modelling, the protocol is still reliant on rather strong assumptions on the security of the election authority and voters' trust thereof. To reduce the protocol's dependency on these assumptions,

several interesting routes present themselves for future work. First, the distribution of key and code generation performed by the election authority could do much to reduce the need for trust in a single entity. Similarly, the protocol would likely benefit from the incorporation of threshold encryption to distribute the encryption and decryption of sensitive vote information.

- A formal investigation into the security of the system by model checking has not yet been carried out. Public confidence in the voting system is critical to deployment success, and formal verification similar to that used for elevator systems and space ship software could do much to convince voters of the protocol's security.
- To offer better accessibility, it would be interesting to look into how code voting may be made accessible to voters with special needs. Similar high security domains like banking will likely provide some starting points for how to address the presentation of cryptographic codes in an accessible way.
- Related to accessibility is usability, and several decisions about how to present the protocol interface to the user need to be dealt with delicately. The choices about how the vote codes are processed and potential guidelines or aids to interpret receipt codes are expected to have a significant effect on the public perception of the protocol and its security.
- Lastly, the forward looking question of how a digital voting system can be kept secure in the face of quantum computing remains unanswered. As mentioned, efforts to replace vulnerable cryptographic primitives with post-quantum secure algorithms are underway, but have, to the best of our knowledge, not yet been considered explicitly for online voting.

Though we have come far in proposing a new, better protocol for secure, end-to-end verifiable online voting, we hope that our work is simply the beginning of a greater effort to make voting more accessible and flexible without intolerable security sacrifice.

Bibliography

[1] Peter YA Ryan and Vanessa Teague. Pretty good democracy. In *International Workshop on Security Protocols*, pages 111–130. Springer, 2009.

[2] Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. Distributed, end-to-end verifiable, and privacy-preserving internet voting systems. *Computers & Security*, 83:268–299, 2019.

[3] Bruce Schneier. American elections are too easy to hack. We must take action now. *The Guardian [Internet]*. 2018 Apr 18. Available from: `https://g.co/kgs/mwCgZJ` [cited 2020 Aug 13].

[4] Alexa Corse. Democrats' Iowa Caucus Voting App Stirs Security Concerns. *The Wall Street Journal [Internet]*. 2020 Jan 26. Available from: `https://www.wsj.com/articles/dems-iowa-caucus-voting-app-stirs-security-concerns-11580063221` [cited 2020 Aug 13].

[5] Lily Hay Newman. The Iowa Caucus Tech Meltdown Is a Warning. *The Wall Street Journal [Internet]*. 2020 Feb 4. Available from: `https://www.wired.com/story/iowa-democratic-caucus-app-tech-meltdown-warning/` [cited 2020 Aug 13].

[6] Abby Abazorius. MIT researchers identify security vulnerabilities in voting app. *MIT News [Internet]*. 2020 February 13. Available from: `http://news.mit.edu/2020/voting-voatz-app-hack-issues-0213` [cited 2020 Mar 30].

[7] Robert Alan Dahl. *Democracy and its Critics*, pages 1–9. Yale University Press, 1989.

[8] Ronnie Dugger. Annals of democracy: Counting votes. *New Yorker*, 64(38): 40–108, 1988.

[9] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems. In Feng Hao and Peter YA Ryan, editors, *Real-World Electronic Voting: Design, Analysis and Deployment*, chapter 7, pages 175–218. CRC Press, Florida, 2016.

[10] Matt Becker, Lauren Chandler, Patrick Hayes, Wesley Hedrick, Kurtis Jensen, Srini Kandikattu, Pete Martin, Scott Meier, Leopoldo Peña, Kun Peng, Aleck Silva-Pinto, Jeffrey Stern, Liv Stromme, Lakshman Tavag, Dave Wallick, and Noah Zweben. Proof of vote - an end-to-end verifiable digital voting protocol using distributed ledger technology (blockchain). Technical report, Votem Corp., Oct 2018. Available from: `https://github.com/votem/proof-of-vote` [cited 2020 May 27].

[11] Susan Dzieduszycka-Suinat, Judy Murray, Joseph R. Kiniry, Daniel M. Zimmerman, Daniel Wagner, Philip Robinson, Adam Foltzer, and Shpatar Morina. The Future of Voting. End-to-end verifiable internet voting. Specification and feasibility assessment study. Technical report, U.S. Vote Foundation, Jul 2015. Available from: `https://www.usvotefoundation.org/E2E-VIV/` [cited 2020 May 21].

[12] Matthew Bernhard, Josh Benaloh, J Alex Halderman, Ronald L Rivest, Peter YA Ryan, Philip B Stark, Vanessa Teague, Poorvi L Vora, and Dan S Wallach. Public evidence from secret ballots. In *International Joint Conference on Electronic Voting*, pages 84–109. Springer, 2017.

[13] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

[14] Peter YA Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *IEEE transactions on information forensics and security*, 4(4):662–673, 2009.

[15] Kevin Fisher, Richard Carback, and Alan T Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *Proceedings of Workshop on Trustworthy Elections*, pages 19–29, 2006.

[16] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3):40–46, 2008.

[17] Ben Adida and Ronald L Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40, 2006.

[18] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *International Conference on E-Voting and Identity*, pages 111–124. Springer, 2007.

[19] Daniel Sandler, Kyle Derr, and Dan S Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium*, volume 4, page 87, 2008.

[20] Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A new implementation of a dual (paper and cryptographic) voting system. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[21] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, et al. Star-vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[22] Feng Hao, Matthew N Kreeger, Brian Randell, Dylan Clarke, Siamak F Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. In *2014 Electronic Voting Technology Workshop/-Workshop on Trustworthy Elections (EVT/WOTE 14)*, 2014.

[23] Computer Security Resource Center. Glossary. *National Institute of Standards and Technology [Internet]*, 2020. Available from: `https://csrc.nist.gov/Glossary` [cited 2020 Jun 4].

[24] Salil P Vadhan. Pseudorandomness. In *Foundations and Trends in Theoretical Computer Science*, volume 7, chapter 1, pages 2–9. now publishers, Dec 2012.

[25] Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*, chapter 1. CRC press, 1996.

[26] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, bitcoin.org, Oct 2008. Available from: `https://bitcoin.org/bitcoin.pdf` [cited 2020 Jun 10].

[28] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[29] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[30] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[31] 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*, pages 703–715, 2014.

[32] Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *International Conference on Financial Cryptography and Data Security*, pages 176–192. Springer, 2016.

[33] Michael A Specter, James Koppel, and Daniel Weitzner. The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in US Federal Elections. *MIT [Preprint]*. MIT; 2020 Feb. Available from: `https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf` [cited 2020 Mar 30].

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

[35] Mahender Kumar, Satish Chand, and CP Katti. A secure end-to-end verifiable internet-voting system using identity-based blind signature. *IEEE Systems Journal*, 2020.

[36] Robert Kofler, Robert Krimmer, and Alexander Prosser. Electronic voting: algorithmic and implementation issues. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 7–pp. IEEE, 2003.

[37] Ronald L Rivest. On the notion of 'software independence'in voting systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3759–3767, 2008.

[38] Matthew Rosenberg, Nick Corasaniti, Sheera Frenkel, and Nicole Perlroth. Faulty Iowa App Was Part of Push to Restore Democrats' Digital Edge. *The New York Times [Internet]*. 2020 Feb 4. Available from: `https://nyti.ms/2uaeZRh` [cited 2020 Mar 27].

[39] Engelbert Hubbers, Bart Jacobs, and Wolter Pieters. RIES - Internet Voting in Action. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, volume 1, pages 417–424. IEEE, 2005.

[40] Valimised. General Framework of Electronic Voting and Implementation thereof at National Elections in Estonia. Technical report, State Electoral Office of Estonia, Tallinn, Jun 2017. Available from: `https://www.valimised.ee/en/internet-voting/documents-about-internet-voting` [cited 2020 Jun 4].

[41] Larry Moore and Nimit Sawhney. UNDER THE HOOD - The West Virginia Mobile Voting Pilot. Technical report, Voatz, Inc., Feb 2019.

[42] Adam Shostack. *Threat modeling: Designing for security*, chapter 3. John Wiley & Sons, 2014.

[43] Michael A Specter and Alex J Halderman. Security Analysis of the Democracy Live Online Voting System. Technical report, MIT, Jun 2020. Available from: `https://internetpolicy.mit.edu/wp-content/uploads/2020/06/OmniBallot.pdf` [cited 2020 Jun 11].

[44] IBM Quantum Experience. Shor's algorithm. *IBM [Internet]*, 2020. Available from: `https://quantum-computing.ibm.com/docs/guide/q-algos/shor-s-algorithm` [cited 2020 Jun 4].

[45] IBM Quantum Experience. Grover's algorithm. *IBM [Internet]*, 2020. Available from: `https://quantum-computing.ibm.com/docs/guide/q-algos/grover-s-algorithm` [cited 2020 Jun 4].

[46] James Heather and David Lundin. The append-only web bulletin board. In *International Workshop on Formal Aspects in Security and Trust*, pages 242–256. Springer, 2008.

[47] Chris Culnane and Steve Schneider. A peered bulletin board for robust use in verifiable voting systems. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 169–183. IEEE, 2014.

[48] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 295–310. Springer, 1999.

[49] OpenSSL Software Foundation. OpenSSL. Version 1.1.1g. Available from: `https://www.openssl.org/` [cited 2020 May 12].

[50] Elaine Barker. Recommendations for Key Management: Part 1 - General. Technical report, National Institute of Standards and Technology, May 2020. Available from: `https://doi.org/10.6028/NIST.SP.800-57pt1r5` [cited 2020 Aug 11].

Appendix A

LSEPI Checklist

Table A.1: Legal, social, ethical and professional issues checklist, as discussed in Chapter 1.

	Yes	No
Section 1: HUMAN EMBRYOS/FETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human fetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localisation data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓

Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?	✓	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓

Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatisation, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
Section 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?	✓	

Appendix B

Simulator Logs

Logs on the next pages.

```
18:33:09.995 [info] Simulator setup begun
18:33:09.998 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 1, "numAdversaries": 0, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 0,
"bulletinBoardHonesty": 1, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597163589}
18:33:09.999 [trace] Created Candidate 0
18:33:09.999 [trace] Created Candidate 1
18:33:09.999 [trace] Created Voter 0 (dishonest)
18:33:09.999 [trace] Created Voter 1 (honest)
18:33:09.999 [info] Simulator setup completed
18:33:09.999 [info] Initiation begun
18:33:10.000 [trace] Generated AES encryption keys
18:33:10.337 [trace] Generated RSA key pairs
18:33:10.340 [trace] Setup bulletin board
18:33:10.345 [debug] Added BB entry: <bbpk, LS0t..., Rsgy..., Initiator, wNvF..., ighd..., Rsgy..., FoGd...>
18:33:10.352 [debug] Added BB entry: <inpk, LS0t..., Rsgy..., Initiator, DysX..., Af4h..., Rsgy..., IBuz...>
18:33:10.370 [debug] Added BB entry: <Candidate 0, AA..., Rsgy..., Initiator, NQp/..., ksy6..., Rsgy..., WkD+...>
18:33:10.379 [debug] Added BB entry: <Candidate 1, AQ..., Rsgy..., Initiator, KXgv..., IUja..., Rsgy..., v5Df...>
18:33:10.381 [debug] Made ballot with credential 3QDnEpOOEQ
------------------------------------------------
| Credential         | 3QDnEpOOEQ                |
| Error receipt code | hRblAyXLrg                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | XPnyHnnECg | EY9HI6mStQ   |
|        Candidate 1 | 55+ZEEHFaw | OUy6mTfK4Q   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:33:10.381 [debug] Made ballot with credential GJwpHyFspQ
------------------------------------------------
| Credential         | GJwpHyFspQ                |
| Error receipt code | mXgc5464EA                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | mwqrnwoFHA | K4j0oDUJlg   |
|        Candidate 1 | yCFBLMsppA | MDaW7bjE3Q   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:33:10.386 [debug] Added BB entry: <#Voters, AgAA..., Rsgy..., Initiator, 4Fse..., Zhwh..., Rsgy..., FRH6...>
18:33:10.389 [trace] Setup collector
18:33:10.389 [trace] Setup tallier
18:33:10.395 [debug] Added BB entry: <cpk, LS0t..., Rsgy..., Initiator, JzeH..., NTeb..., Rsgy..., KFlA...>
18:33:10.401 [debug] Added BB entry: <tpk, LS0t..., Rsgy..., Initiator, duC4..., pWyR..., Rsgy..., abkx...>
18:33:10.404 [trace] Voter 0 (dishonest) received ballot with credentials: 3QDnEpOOEQ
18:33:10.404 [trace] Voter 1 (honest) received ballot with credentials: GJwpHyFspQ
18:33:10.404 [trace] Distributed all ballots to voters
18:33:10.404 [info] Initiation completed
18:33:10.404 [info] Collection begun
18:33:10.404 [trace] Collector activated
18:33:10.404 [trace] Voter 0 (dishonest) chose vote code: XPnyHnnECg
18:33:10.404 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:33:10.410 [debug] Added BB entry: <Inner vote code, Ub7m..., Rsgy..., Collector, 9Y3o..., F4Ag..., Rsgy..., Gw4J...>
18:33:10.411 [trace] Voter 0 (dishonest) received expected receipt code: EY9HI6mStQ
18:33:10.411 [warning] Credential 3QDnEpOOEQ already voted
18:33:10.411 [warning] Voter 0 (dishonest) received inexpected receipt code: hRblAyXLrg
18:33:10.411 [trace] Voter 1 (honest) chose vote code: mwqrnwoFHA
18:33:10.411 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:33:10.418 [debug] Added BB entry: <Inner vote code, Q7e8..., Rsgy..., Collector, WF/k..., Np3K..., Rsgy..., nvd8...>
18:33:10.420 [trace] Voter 1 (honest) received expected receipt code: K4j0oDUJlg
18:33:10.420 [info] Collection completed
18:33:10.420 [info] Tallying begun
18:33:10.427 [debug] Added BB entry: <isk, cShd..., Rsgy..., Initiator, aJGy..., qVOg..., Rsgy..., n6Gr...>
18:33:10.429 [trace] Initiator published isk
18:33:10.432 [trace] Tallier read bulletin board history
18:33:10.440 [debug] Added BB entry: <Candidate ID (padded), AKkB..., Rsgy..., Tallier, hH/m..., s4+a..., Rsgy..., ddrv...>
18:33:10.450 [debug] Added BB entry: <Candidate ID (padded), ANPW..., Rsgy..., Tallier, 7eeo..., F+1X..., Rsgy..., WVyI...>
18:33:10.458 [debug] Added BB entry: <Candidate 0: 2, AA..., Rsgy..., Tallier, 84Ly..., EnxR..., Rsgy..., aNQC...>
18:33:10.469 [debug] Added BB entry: <Candidate 1: 0, AQ..., Rsgy..., Tallier, OPcA..., fN92..., Rsgy..., VVo7...>
18:33:10.472 [info] Tallying completed
```

Figure B.1: Simulator logs with a dishonest voter.

```
18:35:48.362 [info] Simulator setup begun
18:35:48.362 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 0, "numAdversaries": 1, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 0,
"bulletinBoardHonesty": 1, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597163748}
18:35:48.363 [trace] Created Candidate 0
18:35:48.363 [trace] Created Candidate 1
18:35:48.363 [trace] Created Voter 0 (honest)
18:35:48.363 [trace] Created Voter 1 (honest)
18:35:48.363 [info] Simulator setup completed
18:35:48.363 [info] Initiation begun
18:35:48.364 [trace] Generated AES encryption keys
18:35:48.752 [trace] Generated RSA key pairs
18:35:48.756 [trace] Setup bulletin board
18:35:48.761 [debug] Added BB entry: <bbpk, LS0t..., 5Mgy..., Initiator, 654C..., V+Cp..., 5Mgy..., Ml86...>
18:35:48.768 [debug] Added BB entry: <inpk, LS0t..., 5Mgy..., Initiator, SKEC..., UgVg..., 5Mgy..., J+yv...>
18:35:48.775 [debug] Added BB entry: <Candidate 0, AA..., 5Mgy..., Initiator, 1LBJ..., Zh73..., 5Mgy..., qFfh...>
18:35:48.781 [debug] Added BB entry: <Candidate 1, AQ..., 5Mgy..., Initiator, V/UC..., c0P5..., 5Mgy..., mvYR...>
18:35:48.783 [debug] Made ballot with credential ogrgEk2hHw
------------------------------------------------
| Credential         | ogrgEk2hHw                |
| Error receipt code | HDXTkl+acA                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | jS0LkoBpaQ | odeAFOyu3w   |
|        Candidate 1 | v7DTW+WMqg | O/tXLBcn6g   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:35:48.783 [debug] Made ballot with credential ZV30J8Wnqw
------------------------------------------------
| Credential         | ZV30J8Wnqw                |
| Error receipt code | PLJAzZ19LQ                |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | ViZNnM9+AQ | V1ubfBe+6A   |
|        Candidate 1 | SIgm8C4Ykw | Fju1BFTnuw   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:35:48.789 [debug] Added BB entry: <#Voters, AgAA..., 5Mgy..., Initiator, ebcS..., pGW/..., 5Mgy..., aHl9...>
18:35:48.791 [trace] Setup collector
18:35:48.791 [trace] Setup tallier
18:35:48.802 [debug] Added BB entry: <cpk, LS0t..., 5Mgy..., Initiator, JA95..., aqfZ..., 5Mgy..., mwBa...>
18:35:48.810 [debug] Added BB entry: <tpk, LS0t..., 5Mgy..., Initiator, CzEJ..., jK4z..., 5Mgy..., HMtE...>
18:35:48.811 [trace] Voter 0 (honest) received ballot with credentials: ogrgEk2hHw
18:35:48.811 [trace] Voter 1 (honest) received ballot with credentials: ZV30J8Wnqw
18:35:48.811 [trace] Distributed all ballots to voters
18:35:48.811 [info] Initiation completed
18:35:48.811 [info] Collection begun
18:35:48.811 [trace] Collector activated
18:35:48.812 [trace] Voter 0 (honest) chose vote code: v7DTW+WMqg
18:35:48.812 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:35:48.818 [debug] Added BB entry: <Inner vote code, KDz6..., 5Mgy..., Collector, oQIZ..., rcj7..., 5Mgy..., BNJ9...>
18:35:48.820 [trace] Voter 0 (honest) received expected receipt code: O/tXLBcn6g
18:35:48.820 [trace] Voter 1 (honest) chose vote code: ViZNnM9+AQ
18:35:48.820 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:35:48.826 [debug] Added BB entry: <Inner vote code, 2qml..., 5Mgy..., Collector, pLv8..., ph52..., 5Mgy..., e7Y7...>
18:35:48.828 [trace] Voter 1 (honest) received expected receipt code: V1ubfBe+6A
18:35:48.828 [warning] Collector did not find voter credential: GzTvyVJgog
18:35:48.828 [trace] Adversary 0 attempted to guess a vote and received receipt code: RDiwZCXF6A
18:35:48.828 [info] Collection completed
18:35:48.828 [info] Tallying begun
18:35:48.834 [debug] Added BB entry: <isk, l6sD..., 5Mgy..., Initiator, sBtZ..., mo4g..., 5Mgy..., jmfO...>
18:35:48.836 [trace] Initiator published isk
18:35:48.837 [trace] Tallier read bulletin board history
18:35:48.854 [debug] Added BB entry: <Candidate ID (padded), AW7E..., 5Mgy..., Tallier, oEcn..., VG9B..., 5Mgy..., ZgDW...>
18:35:48.862 [debug] Added BB entry: <Candidate ID (padded), ANvL..., 5Mgy..., Tallier, 4pgg..., eXsR..., 5Mgy..., iVNF...>
18:35:48.871 [debug] Added BB entry: <Candidate 0: 1, AA..., 5Mgy..., Tallier, pJYv..., VP2H..., 5Mgy..., tbdL...>
18:35:48.878 [debug] Added BB entry: <Candidate 1: 1, AQ..., 5Mgy..., Tallier, 46bi..., K357..., 5Mgy..., n3+c...>
18:35:48.880 [info] Tallying completed
```

Figure B.2: Simulator logs with an adversary.

```
18:37:22.859 [info] Simulator setup begun
18:37:22.860 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 0, "numAdversaries": 0, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 1,
"bulletinBoardHonesty": 1, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597163842}
18:37:22.860 [trace] Created Candidate 0
18:37:22.861 [trace] Created Candidate 1
18:37:22.861 [trace] Created Voter 0 (honest)
18:37:22.861 [trace] Created Voter 1 (honest)
18:37:22.861 [info] Simulator setup completed
18:37:22.861 [info] Initiation begun
18:37:22.863 [trace] Generated AES encryption keys
18:37:23.265 [trace] Generated RSA key pairs
18:37:23.269 [trace] Setup bulletin board
18:37:23.274 [debug] Added BB entry: <bbpk, LS0t..., Q8ky..., Initiator, 6aqL..., ftiQ..., Q8ky..., keEY...>
18:37:23.281 [debug] Added BB entry: <inpk, LS0t..., Q8ky..., Initiator, 0IKf..., gX4c..., Q8ky..., JLQq...>
18:37:23.287 [debug] Added BB entry: <Candidate 0, AA..., Q8ky..., Initiator, t4T0..., QrW+..., Q8ky..., DU7m...>
18:37:23.294 [debug] Added BB entry: <Candidate 1, AQ..., Q8ky..., Initiator, 1hZ6..., H737..., Q8ky..., RYwT...>
18:37:23.296 [debug] Made ballot with credential 806PkPdDcw
-----------------------------------------------------
| Credential         | 806PkPdDcw                   |
| Error receipt code | xIwZ9Yfqbw                   |
-----------------------------------------------------
| Candidate          | Vote code  | Receipt code |
-----------------------------------------------------
|        Candidate 0 | 5pDlUBly2w | Z5U64tXM0g   |
|        Candidate 1 | mPHK/PUVGw | iDuldj9H3g   |
-----------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
-----------------------------------------------------
18:37:23.296 [debug] Made ballot with credential hTEe19wz8g
-----------------------------------------------------
| Credential         | hTEe19wz8g                   |
| Error receipt code | Cpp8tSR8uQ                   |
-----------------------------------------------------
| Candidate          | Vote code  | Receipt code |
-----------------------------------------------------
|        Candidate 0 | PS4M0xWWVQ | oN3KBMoUkA   |
|        Candidate 1 | u6he1R43WA | pd5YMLGSdg   |
-----------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
-----------------------------------------------------
18:37:23.302 [debug] Added BB entry: <#Voters, AgAA..., Q8ky..., Initiator, YQjk..., ax57..., Q8ky..., Ym+O...>
18:37:23.304 [trace] Setup collector
18:37:23.304 [trace] Setup tallier
18:37:23.310 [debug] Added BB entry: <cpk, LS0t..., Q8ky..., Initiator, 8qQe..., bTqo..., Q8ky..., KgGs...>
18:37:23.320 [debug] Added BB entry: <tpk, LS0t..., Q8ky..., Initiator, YzWG..., e3Ba..., Q8ky..., ZxVJ...>
18:37:23.322 [trace] Voter 0 (honest) received ballot with credentials: 806PkPdDcw
18:37:23.322 [trace] Voter 1 (honest) received ballot with credentials: hTEe19wz8g
18:37:23.322 [trace] Distributed all ballots to voters
18:37:23.322 [info] Initiation completed
18:37:23.322 [info] Collection begun
18:37:23.322 [trace] Collector activated
18:37:23.322 [trace] Voter 0 (honest) chose vote code: mPHK/PUVGw
18:37:23.322 [warning] Dishonest collector dropped vote from credential: 806PkPdDcw
18:37:23.322 [warning] Voter 0 (honest) received inexpected receipt code: AAAAAAAAAA
18:37:23.322 [trace] Voter 1 (honest) chose vote code: u6he1R43WA
18:37:23.322 [warning] Dishonest collector dropped vote from credential: hTEe19wz8g
18:37:23.322 [warning] Voter 1 (honest) received inexpected receipt code: AAAAAAAAAA
18:37:23.322 [info] Collection completed
18:37:23.322 [info] Tallying begun
18:37:23.328 [debug] Added BB entry: <isk, xWxE..., Q8ky..., Initiator, X6z8..., Rx/3..., Q8ky..., G7jU...>
18:37:23.331 [trace] Initiator published isk
18:37:23.333 [trace] Tallier read bulletin board history
18:37:23.342 [error] Tallier counted unexpected number of votes
18:37:23.342 [info] Tallying completed
```

Figure B.3: Simulator logs with a vote dropping collector.

```
18:39:17.277 [info] Simulator setup begun
18:39:17.279 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 0, "numAdversaries": 0, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 2,
"bulletinBoardHonesty": 1, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597163957}
18:39:17.279 [trace] Created Candidate 0
18:39:17.279 [trace] Created Candidate 1
18:39:17.279 [trace] Created Voter 0 (honest)
18:39:17.279 [trace] Created Voter 1 (honest)
18:39:17.279 [info] Simulator setup completed
18:39:17.279 [info] Initiation begun
18:39:17.283 [trace] Generated AES encryption keys
18:39:17.591 [trace] Generated RSA key pairs
18:39:17.594 [trace] Setup bulletin board
18:39:17.600 [debug] Added BB entry: <bbpk, LS0t..., tcky..., Initiator, Fvh/..., bL05..., tcky..., zLTW...>
18:39:17.606 [debug] Added BB entry: <inpk, LS0t..., tcky..., Initiator, P8eJ..., Vjok..., tcky..., Fwbj...>
18:39:17.613 [debug] Added BB entry: <Candidate 0, AA..., tcky..., Initiator, Ac7p..., WLEw..., tcky..., FLOT...>
18:39:17.620 [debug] Added BB entry: <Candidate 1, AQ..., tcky..., Initiator, WMz8..., keZ0..., tcky..., MRPy...>
18:39:17.622 [debug] Made ballot with credential b/tfWir7ug
-----------------------------------------------------
| Credential         | b/tfWir7ug                   |
| Error receipt code | jEMsK5udrw                   |
-----------------------------------------------------
| Candidate          | Vote code  | Receipt code |
-----------------------------------------------------
|        Candidate 0 | LyKYnYumPw | gd+RpXGt8g   |
|        Candidate 1 | HssFhjVS1Q | zCGv7J3Fhg   |
-----------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
-----------------------------------------------------
18:39:17.622 [debug] Made ballot with credential sZSH17IaKA
-----------------------------------------------------
| Credential         | sZSH17IaKA                   |
| Error receipt code | i1mX5h204Q                   |
-----------------------------------------------------
| Candidate          | Vote code  | Receipt code |
-----------------------------------------------------
|        Candidate 0 | eDar/HF5WA | ROXaK81sXQ   |
|        Candidate 1 | s41GOUoDMg | JzPZ0dB0/g   |
-----------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
-----------------------------------------------------
18:39:17.628 [debug] Added BB entry: <#Voters, AgAA..., tcky..., Initiator, Fd5Y..., Wjfq..., tcky..., V1kz...>
18:39:17.631 [trace] Setup collector
18:39:17.631 [trace] Setup tallier
18:39:17.640 [debug] Added BB entry: <cpk, LS0t..., tcky..., Initiator, Bujo..., T7M3..., tcky..., DGwN...>
18:39:17.649 [debug] Added BB entry: <tpk, LS0t..., tcky..., Initiator, prP0..., I4Wr..., tcky..., zfT1...>
18:39:17.651 [trace] Voter 0 (honest) received ballot with credentials: b/tfWir7ug
18:39:17.651 [trace] Voter 1 (honest) received ballot with credentials: sZSH17IaKA
18:39:17.651 [trace] Distributed all ballots to voters
18:39:17.651 [info] Initiation completed
18:39:17.651 [info] Collection begun
18:39:17.651 [trace] Collector activated
18:39:17.651 [trace] Voter 0 (honest) chose vote code: LyKYnYumPw
18:39:17.651 [warning] Dishonest collector tampered with vote from credential: b/tfWir7ug
18:39:17.656 [debug] Added BB entry: <Inner vote code, nsDy..., tcky..., Collector, Bgfp..., rEbf..., tcky..., taJ/...>
18:39:17.658 [warning] Voter 0 (honest) received inexpected receipt code: zCGv7J3Fhg
18:39:17.658 [trace] Voter 1 (honest) chose vote code: s41GOUoDMg
18:39:17.658 [warning] Dishonest collector tampered with vote from credential: sZSH17IaKA
18:39:17.664 [debug] Added BB entry: <Inner vote code, QSrG..., tcky..., Collector, VlgS..., fTz4..., tcky..., fWBy...>
18:39:17.666 [warning] Voter 1 (honest) received inexpected receipt code: ROXaK81sXQ
18:39:17.666 [info] Collection completed
18:39:17.666 [info] Tallying begun
18:39:17.672 [debug] Added BB entry: <isk, 6pCZ..., tcky..., Initiator, KO4K..., BVBR..., tcky..., zEUL...>
18:39:17.673 [trace] Initiator published isk
18:39:17.683 [trace] Tallier read bulletin board history
18:39:17.690 [debug] Added BB entry: <Candidate ID (padded), AV5I..., tcky..., Tallier, UXqP..., C1rW..., tcky..., ZpbZ...>
18:39:17.700 [debug] Added BB entry: <Candidate ID (padded), AN+t..., tcky..., Tallier, bbgn..., Vofh..., tcky..., da3k...>
18:39:17.707 [debug] Added BB entry: <Candidate 0: 1, AA..., tcky..., Tallier, JYyN..., nfIY..., tcky..., m7bv...>
18:39:17.715 [debug] Added BB entry: <Candidate 1: 1, AQ..., tcky..., Tallier, 4Akn..., E1Qk..., tcky..., Z9SD...>
18:39:17.717 [info] Tallying completed
```

Figure B.4: Simulator logs with vote tampering collector.

```
18:40:58.865 [info] Simulator setup begun
18:40:58.867 [info] Simulator parameters: {"numVoters": 2, "propDishonest": 0, "numAdversaries": 0, "numCandidates": 2,
"aesKeyLen": 16, "rsaKeyLen": 2048, "ballotCodeLen": 7, "cryptoCodeLen": 16, "collectorBehavior": 0,
"bulletinBoardHonesty": 0, "logLevel": 0, "logFormat": "%H:%M:%S.%e %^[%l]%$ %v", "timestamp": 1597164058}
18:40:58.867 [trace] Created Candidate 0
18:40:58.867 [trace] Created Candidate 1
18:40:58.867 [trace] Created Voter 0 (honest)
18:40:58.867 [trace] Created Voter 1 (honest)
18:40:58.867 [info] Simulator setup completed
18:40:58.867 [info] Initiation begun
18:40:58.871 [trace] Generated AES encryption keys
18:40:59.220 [trace] Generated RSA key pairs
18:40:59.224 [trace] Setup bulletin board
18:40:59.229 [debug] Added BB entry: <bbpk, LS0t..., G8oy..., Initiator, LPn7..., hnFp..., G8oy..., WDkm...>
18:40:59.236 [debug] Added BB entry: <inpk, LS0t..., G8oy..., Initiator, AM0s..., YoKb..., G8oy..., bJFC...>
18:40:59.242 [debug] Added BB entry: <Candidate 0, AA..., G8oy..., Initiator, MKAZ..., by9u..., G8oy..., nx1z...>
18:40:59.249 [debug] Added BB entry: <Candidate 1, AQ..., G8oy..., Initiator, kaMa..., YlPN..., G8oy..., Mmrf...>
18:40:59.251 [debug] Made ballot with credential Mha5V1YCqw
------------------------------------------------
| Credential         | Mha5V1YCqw               |
| Error receipt code | /ERh8DlOOQ               |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | OlT6BZnaFQ | Q4WaPq8+Yg   |
|        Candidate 1 | 6drAxSAMnw | R6/YGb8F0w   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:40:59.251 [debug] Made ballot with credential DzQo4BuVIQ
------------------------------------------------
| Credential         | DzQo4BuVIQ               |
| Error receipt code | bPUrpVl3qQ               |
------------------------------------------------
| Candidate          | Vote code  | Receipt code |
------------------------------------------------
|        Candidate 0 | 8oSL8CRzaA | FuPGHHhtZg   |
|        Candidate 1 | iw7OUQ0+dQ | c02TokbYyA   |
------------------------------------------------
| Bulletin board public key | LS0tLQ...          |
------------------------------------------------
18:40:59.257 [debug] Added BB entry: <#Voters, AgAA..., G8oy..., Initiator, kflW..., dndk..., G8oy..., SLz6...>
18:40:59.260 [trace] Setup collector
18:40:59.260 [trace] Setup tallier
18:40:59.271 [debug] Added BB entry: <cpk, LS0t..., G8oy..., Initiator, AEPW..., XF+c..., G8oy..., LRRP...>
18:40:59.280 [debug] Added BB entry: <tpk, LS0t..., G8oy..., Initiator, B9Bp..., zHI2..., G8oy..., pwrX...>
18:40:59.283 [trace] Voter 0 (honest) received ballot with credentials: Mha5V1YCqw
18:40:59.283 [trace] Voter 1 (honest) received ballot with credentials: DzQo4BuVIQ
18:40:59.283 [trace] Distributed all ballots to voters
18:40:59.283 [info] Initiation completed
18:40:59.283 [info] Collection begun
18:40:59.283 [trace] Collector activated
18:40:59.283 [trace] Voter 0 (honest) chose vote code: 6drAxSAMnw
18:40:59.283 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:40:59.289 [debug] Added BB entry: <Inner vote code, aPyF..., G8oy..., Collector, c1ld..., ag3c..., G8oy..., kbVc...>
18:40:59.290 [trace] Voter 0 (honest) received expected receipt code: R6/YGb8F0w
18:40:59.290 [trace] Voter 1 (honest) chose vote code: iw7OUQ0+dQ
18:40:59.290 [trace] Collector found valid vote code, forwarded inner vote code to BB
18:40:59.296 [debug] Added BB entry: <Inner vote code, q4HC..., G8oy..., Collector, 8Ic0..., EzVp..., G8oy..., dAFV...>
18:40:59.298 [trace] Voter 1 (honest) received expected receipt code: c02TokbYyA
18:40:59.298 [info] Collection completed
18:40:59.298 [info] Tallying begun
18:40:59.303 [debug] Added BB entry: <isk, lk8Y..., G8oy..., Initiator, J4zO..., LKI8..., G8oy..., kHaA...>
18:40:59.305 [trace] Initiator published isk
18:40:59.315 [trace] Tallier read bulletin board history
18:40:59.316 [error] Tallier read inconsistent history (hash)
18:40:59.316 [info] Tallying completed
```

Figure B.5: Simulator logs with a dishonest bulletin board.

Appendix C

Simulation Results

Table C.1: Distribution of time (s) for different voter population sizes.

Phase \ # of voters	100	1000	10000	100000
Initiation	5.180e-01	7.520e-01	3.146e+00	2.801e+01
Collection	6.870e-01	6.997e+00	6.807e+01	6.833e+02
Tallying	7.840e-01	7.265e+00	6.904e+01	6.920e+02
Total	1.989e+00	1.501e+01	1.403e+02	1.403e+03

Table C.2: Impact of AES key length (in bytes) on total time (s).

Key length \ # of voters	100	1000	10000	100000
16	1.989e+00	1.501e+01	1.403e+02	1.403e+03
32	2.315e+00	1.507e+01	1.412e+02	1.403e+03

Table C.3: Impact of AES key length (in bytes) on vote processing frequency (s^{-1}).

Key length \ # of voters	100	1000	10000	100000
16	1.456e+02	1.429e+02	1.469e+02	1.463e+02
32	1.370e+02	1.410e+02	1.468e+02	1.464e+02

Table C.4: Impact of cryptographic code length (in bytes) on total time (s).

Code length \ # of voters	100	1000	10000	100000
16	1.989e+00	1.501e+01	1.403e+02	1.403e+03
32	1.930e+00	1.495e+01	1.410e+02	1.411e+03
64	1.928e+00	1.468e+01	1.408e+02	1.415e+03

Table C.5: Impact of cryptographic code length (in bytes) on vote processing frequency (s^{-1}).

Code length \ # of voters	100	1000	10000	100000
16	1.456e+02	1.429e+02	1.469e+02	1.463e+02
32	1.451e+02	1.413e+02	1.460e+02	1.454e+02
64	1.453e+02	1.418e+02	1.460e+02	1.441e+02

Table C.6: Impact of number of adversaries (in percent of voters) on total time (s).

Adversaries \ # of voters	100	1000	10000	100000
0%	1.989e+00	1.501e+01	1.403e+02	1.403e+03
25%	2.075e+00	1.578e+01	1.413e+02	1.403e+03
50%	1.990e+00	1.537e+01	1.413e+02	1.399e+03
75%	2.083e+00	1.578e+01	1.403e+02	1.403e+03
100%	2.142e+00	1.548e+01	1.401e+02	1.408e+03

Table C.7: Impact of number of adversaries (in percent of voters) on vote processing frequency (s^{-1}).

Adversaries \ # of voters	100	1000	10000	100000
0%	1.456e+02	1.429e+02	1.469e+02	1.463e+02
25%	1.445e+02	1.348e+02	1.445e+02	1.467e+02
50%	1.376e+02	1.376e+02	1.447e+02	1.466e+02
75%	1.377e+02	1.302e+02	1.469e+02	1.464e+02
100%	1.221e+02	1.309e+02	1.469e+02	1.463e+02

Table C.8: Impact of number of candidates on total time (s).

Candidates \ # of voters	100	1000	10000	100000
5	2.190e+00	1.510e+01	1.393e+02	1.388e+03
10	1.989e+00	1.501e+01	1.403e+02	1.403e+03
15	2.154e+00	1.516e+01	1.413e+02	1.437e+03
20	2.743e+00	1.562e+01	1.435e+02	1.427e+03

Table C.9: Impact of number of candidates on vote processing frequency (s^{-1}).

Candidates \ # of voters	100	1000	10000	100000
5	1.439e+02	1.397e+02	1.469e+02	1.472e+02
10	1.456e+02	1.429e+02	1.469e+02	1.463e+02
15	1.435e+02	1.443e+02	1.471e+02	1.453e+02
20	1.350e+02	1.377e+02	1.457e+02	1.464e+02

Table C.10: Impact of dishonest proportion of voters (in percent of voters) on total time (s).

Proportion \ # of voters	100	1000	10000	100000
0%	1.989e+00	1.501e+01	1.403e+02	1.403e+03
10%	1.920e+00	1.501e+01	1.410e+02	1.403e+03
20%	2.160e+00	1.467e+01	1.413e+02	1.401e+03
30%	1.982e+00	1.480e+01	1.400e+02	1.400e+03

Table C.11: Impact of dishonest proportion of voters (in percent of voters) on vote processing frequency (s^{-1}).

Proportion \ # of voters	100	1000	10000	100000
0%	1.456e+02	1.429e+02	1.469e+02	1.463e+02
10%	1.439e+02	1.375e+02	1.471e+02	1.466e+02
20%	1.414e+02	1.455e+02	1.459e+02	1.471e+02
30%	1.447e+02	1.451e+02	1.472e+02	1.468e+02

Table C.12: Impact of RSA key length (in bytes) on total time (s).

Key length \ # of voters	100	1000	10000	100000
512	2.570e-01	2.138e+00	2.066e+01	2.100e+02
1024	5.330e-01	4.037e+00	4.007e+01	3.873e+02
2048	1.989e+00	1.501e+01	1.403e+02	1.403e+03
4096	1.054e+01	7.734e+01	7.401e+02	7.372e+03

Table C.13: Impact of RSA key length (in bytes) on vote processing frequency (s^{-1}).

Key length \ # of voters	100	1000	10000	100000
512	1.111e+03	1.095e+03	1.114e+03	1.116e+03
1024	5.714e+02	5.510e+02	5.308e+02	5.639e+02
2048	1.456e+02	1.429e+02	1.469e+02	1.463e+02
4096	2.711e+01	2.737e+01	2.727e+01	2.739e+01