Secure Online Elections

Roland Wen
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia
rolandw@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-0436
October 2004

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY • AUSTRALIA

Abstract

Advances in cryptography have contributed significantly to the Internet revolution, and one of the next major applications is secure online elections. In the wake of the debacle of the US presidential election in 2000, governments have rapidly begun to adopt online voting. However, this is somewhat premature given the current state of research. No existing election scheme is able to satisfy all the democratic principles of traditional elections whilst also being suitable for large scale voting.

This report is intended to be a self-contained introduction to the field of online elections. It develops the necessary background knowledge, firstly by examining the traditional and online models, as well as the requirements and properties for election protocols. Then it presents an overview of the relevant mathematical and cryptographic preliminaries.

The crux of this report is the survey of the most important election schemes found in the literature. These are categorised according to the approach on which they are based: homomorphic encryption, mix nets and blind signatures. A description and evaluation of each scheme is provided, followed by a comparison of all the protocols and a discussion of the limitations and outstanding concerns.

Acknowledgements

This report was originally submitted in November 2002 as a thesis for the degree of Bachelor of Engineering in Computer Engineering, at the School of Computer Science and Engineering, The University of New South Wales.

I would like to thank Richard Buckland (my supervisor) and Ken Robinson (my co-supervisor) for their support, encouragement and feedback throughout my thesis. In particular, Richard's guidance and the countless discussions we had bouncing ideas around, on some occasions even after the witching hour, helped make the thesis such a fruitful and enjoyable experience.

Finally, this work is dedicated to the memory of my grandfather, Quentin Hsu (1912–2002).

Indeed, you won the elections, but I won the count.

Anastasio Somoza Debayle, Nicaraguan dictator.

Contents

1 Introduction **1**

2 Traditional Elections **4**
2.1 Democratic Model 4
2.2 Voting Systems 6
2.2.1 Majority voting 6
2.2.2 Plurality voting 7
2.2.3 Proportional representation 7
2.2.4 Other voting systems 8
2.3 Vote Casting and Counting Systems 8
2.3.1 Paper ballot 9
2.3.2 Mechanical lever 9
2.3.3 Punch card 9
2.3.4 Optical mark recognition 9
2.3.5 Computer terminal 9
2.3.6 Electronic systems 10
2.4 Typical Election Process 10
2.5 Problems 11
2.5.1 Trust 11
2.5.2 Fraud 11
2.5.3 Cost and logistics 12
2.5.4 Accuracy and efficiency 13
2.5.5 Voter turnout 13

3 Online Elections **14**
3.1 Model 14
3.1.1 Participants 15
3.1.2 Communication 15
3.1.3 Votes 16
3.1.4 Ballots 16
3.1.5 Electoral rolls 17
3.2 Typical Election Process 17
3.3 Security Requirements 17
3.4 Usability Properties 19

3.5 Classification of Schemes . . . 20
3.6 Advantages . . . 21
3.6.1 Convenience . . . 21
3.6.2 Trust . . . 21
3.6.3 Fraud . . . 22
3.6.4 Cost and logistics . . . 22
3.6.5 Accuracy and efficiency . . . 22
3.7 Problems . . . 23
3.7.1 Trust . . . 23
3.7.2 Fraud . . . 23
3.7.3 Disruptions . . . 24
3.7.4 Auditability . . . 24
3.7.5 Social considerations . . . 24

4 Mathematical Primitives **26**
4.1 Higher Algebra . . . 26
4.1.1 Groups . . . 27
4.1.2 Rings . . . 28
4.1.3 Fields . . . 29
4.1.4 Homomorphisms . . . 30
4.2 Number Theory . . . 31
4.2.1 Integers and division . . . 31
4.2.2 The integers modulo n . . . 32
4.2.3 Congruence . . . 33
4.2.4 Linear congruences . . . 34
4.2.5 The Legendre symbol . . . 34
4.2.6 The Jacobi symbol . . . 35
4.3 Complexity Theory . . . 36
4.3.1 Complexity . . . 37
4.3.2 Computational complexity classes . . . 38
4.3.3 Intractability . . . 39
4.3.4 One-way functions . . . 40
4.3.5 Trapdoor one-way functions . . . 40
4.4 Number Theoretic Problems . . . 41
4.4.1 Factoring integers . . . 41
4.4.2 Discrete logarithms . . . 42
4.4.3 Quadratic residues . . . 43

5 Cryptographic Primitives **44**
5.1 Basic Concepts . . . 44
5.1.1 Security . . . 44
5.1.2 Randomness . . . 45
5.1.3 Hash functions . . . 45
5.1.4 Encryption functions . . . 45
5.2 Public Key Cryptosystems . . . 46
5.2.1 RSA cryptosystem . . . 46

5.2.2 Goldwasser-Micali cryptosystem . . . 48
5.2.3 ElGamal cryptosystem . . . 49
5.2.4 Threshold ElGamal cryptosystem . . . 50
5.3 Protocols . . . 51
5.3.1 Digital signatures . . . 51
5.3.2 Blind signatures . . . 52
5.3.3 Bit commitment . . . 53
5.3.4 Trapdoor commitment . . . 54
5.3.5 Chameleon commitment . . . 55
5.3.6 Secret sharing . . . 56
5.3.7 Threshold secret sharing . . . 57
5.3.8 Verifiable secret sharing . . . 58
5.3.9 Mix nets . . . 59
5.3.10 Interactive proofs . . . 60
5.3.11 Zero knowledge proofs . . . 61
5.3.12 The Fiat-Shamir heuristic . . . 62

6 Schemes Based on Homomorphic Encryption 63
6.1 Cohen (Benaloh) and Fischer [CF85] . . . 64
6.1.1 Cryptographic tools . . . 64
6.1.1.1 Encryption . . . 64
6.1.1.2 Decryption . . . 65
6.1.2 Election protocol . . . 65
6.1.2.1 Stage 1: Initialising . . . 65
6.1.2.2 Stage 2: Ballot Preparation . . . 65
6.1.2.3 Stage 3: Voting . . . 65
6.1.2.4 Stage 4: Counting . . . 66
6.1.3 Interactive proofs . . . 66
6.1.3.1 Proving the public key is valid . . . 66
6.1.3.2 Proving the ballot is valid . . . 67
6.1.3.3 Proving the tally is valid . . . 68
6.1.4 Evaluation . . . 69
6.1.4.1 Security . . . 69
6.1.4.2 Usability . . . 69
6.1.4.3 Extensions . . . 70
6.2 Benaloh [Ben87b] . . . 70
6.2.1 Cryptographic tools . . . 70
6.2.1.1 Encryption . . . 70
6.2.1.2 Verifiable secret sharing . . . 70
6.2.2 Election protocol . . . 71
6.2.2.1 Stage 1: Initialising . . . 71
6.2.2.2 Stage 3: Ballot Preparation . . . 71
6.2.2.3 Stage 4: Voting . . . 71
6.2.2.4 Stage 5: Counting . . . 72
6.2.3 Interactive proofs . . . 72
6.2.3.1 Proving the public key is valid . . . 73

6.2.3.2 Proving the ballot is valid . . . 73
6.2.4 Evaluation . . . 73
6.2.4.1 Security . . . 73
6.2.4.2 Usability . . . 74
6.2.4.3 Extensions . . . 74
6.3 Benaloh and Tuinstra [BT94] . . . 75
6.3.1 Cryptographic tools . . . 75
6.3.2 Election protocol . . . 75
6.3.2.1 Stage 1: Initialising . . . 75
6.3.2.2 Stage 2: Ballot Preparation . . . 76
6.3.2.3 Stage 3: Voting . . . 76
6.3.2.4 Stage 4: Counting . . . 77
6.3.3 Interactive proofs . . . 78
6.3.3.1 Proving the masking factor is valid . . . 78
6.3.3.2 Proving the ballot is valid . . . 79
6.3.4 Evaluation . . . 80
6.3.4.1 Security . . . 80
6.3.4.2 Usability . . . 81
6.4 Cramer, Franklin, Schoenmakers and Yung [CFSY96] . . . 81
6.4.1 Cryptographic tools . . . 82
6.4.2 Election protocol . . . 82
6.4.2.1 Stage 1: Initialising . . . 82
6.4.2.2 Stage 2: Ballot Preparation . . . 82
6.4.2.3 Stage 3: Voting . . . 83
6.4.2.4 Stage 4: Counting . . . 83
6.4.3 Interactive proofs . . . 84
6.4.3.1 Proving the ballot is valid . . . 85
6.4.4 Evaluation . . . 86
6.4.4.1 Security . . . 86
6.4.4.2 Usability . . . 87
6.4.4.3 Extensions . . . 87
6.5 Cramer, Gennaro and Schoenmakers [CGS97] . . . 88
6.5.1 Cryptographic tools . . . 88
6.5.1.1 Encryption . . . 88
6.5.1.2 Decryption . . . 88
6.5.2 Election protocol . . . 89
6.5.2.1 Stage 1: Initialising . . . 89
6.5.2.2 Stage 2: Voting . . . 90
6.5.2.3 Stage 3: Counting . . . 90
6.5.3 Interactive proofs . . . 90
6.5.3.1 Proof of discrete logarithm equality . . . 90
6.5.3.2 Proving the ballot is valid . . . 91
6.5.4 Evaluation . . . 93
6.5.4.1 Security . . . 93
6.5.4.2 Usability . . . 94
6.5.4.3 Extensions . . . 94

7 Schemes Based on Mix Nets **95**
7.1 Chaum [Cha81] . . . 95
7.1.1 Cryptographic tools . . . 95
7.1.1.1 Encryption . . . 95
7.1.1.2 Digital signatures . . . 95
7.1.1.3 Mix nets . . . 96
7.1.2 Election protocol . . . 96
7.1.2.1 Stage 1: Initialising . . . 96
7.1.2.2 Stage 2: Authorising . . . 96
7.1.2.3 Stage 3: Voting . . . 97
7.1.2.4 Stage 4: Counting . . . 97
7.1.3 Evaluation . . . 97
7.1.3.1 Security . . . 97
7.1.3.2 Usability . . . 98
7.2 Sako and Kilian [SK95] . . . 98
7.2.1 Cryptographic protocols . . . 99
7.2.1.1 Encryption . . . 99
7.2.1.2 Mix nets . . . 99
7.2.1.3 Chameleon commitment . . . 100
7.2.2 Election protocol . . . 100
7.2.2.1 Stage 1: Initialising . . . 100
7.2.2.2 Stage 2: Ballot Preparation . . . 101
7.2.2.3 Stage 3: Voting . . . 101
7.2.2.4 Stage 4: Counting . . . 101
7.2.3 Evaluation . . . 101
7.2.3.1 Security . . . 101
7.2.3.2 Usability . . . 102
7.3 Hirt and Sako [HS00] . . . 102
7.3.1 Cryptographic tools . . . 102
7.3.2 Election protocol . . . 103
7.3.2.1 Stage 1: Initialising . . . 103
7.3.2.2 Stage 2: Ballot Preparation . . . 103
7.3.2.3 Stage 3: Voting . . . 104
7.3.2.4 Stage 4: Counting . . . 104
7.3.3 Evaluation . . . 104
7.3.3.1 Security . . . 104
7.3.3.2 Usability . . . 105

8 Schemes Based on Blind Signatures **106**
8.1 Fujioka, Okamoto and Ohta [FOO93] . . . 106
8.1.1 Cryptographic tools . . . 106
8.1.1.1 Bit commitment . . . 106
8.1.1.2 Digital signatures . . . 106
8.1.1.3 Blind signatures . . . 107
8.1.2 Election protocol . . . 107
8.1.2.1 Stage 1: Initialising . . . 107

8.1.2.2 Stage 2: Ballot Preparation . . . 107
8.1.2.3 Stage 3: Authorising . . . 107
8.1.2.4 Stage 4: Voting . . . 108
8.1.2.5 Stage 5: Opening . . . 108
8.1.2.6 Stage 6: Counting . . . 108
8.1.3 Evaluation . . . 108
8.1.3.1 Security . . . 108
8.1.3.2 Usability . . . 109
8.2 Okamoto [Oka97] . . . 109
8.2.1 Cryptographic tools . . . 109
8.2.1.1 Encryption . . . 109
8.2.1.2 Trapdoor commitment . . . 110
8.2.1.3 Digital signatures . . . 110
8.2.1.4 Blind signatures . . . 110
8.2.2 Election protocol . . . 110
8.2.2.1 Stage 1: Initialising . . . 110
8.2.2.2 Stage 2: Ballot Preparation . . . 111
8.2.2.3 Stage 3: Authorising . . . 111
8.2.2.4 Stage 4: Voting . . . 111
8.2.2.5 Stage 5: Claiming . . . 111
8.2.2.6 Stage 6: Counting . . . 112
8.2.3 Evaluation . . . 112
8.2.3.1 Security . . . 112
8.2.3.2 Usability . . . 113

9 Comparison of Existing Schemes **114**
9.1 Overview . . . 114
9.2 Security Requirements . . . 118
9.2.1 Privacy . . . 118
9.2.2 Incoercibility . . . 118
9.2.3 Robustness . . . 118
9.2.4 Verifiability . . . 118
9.2.5 Correctness . . . 119
9.2.6 Fairness . . . 119
9.2.7 Vote independence . . . 119
9.3 Usability Properties . . . 119
9.3.1 Computational complexity . . . 119
9.3.2 Communication complexity . . . 120
9.3.3 Round complexity . . . 120
9.4 Summary . . . 120

10 Conclusion **122**

A Notation **124**

Bibliography **126**

Chapter 1

Introduction

The concept of voting is an integral part of contemporary society. We use it every day to exercise our free will in making group choices, from voting for a favourite flavour of ice cream to voting in a jury. The process of voting is most closely associated with elections to choose leaders or decide on important issues, and naturally the pinnacle is the democratic election of government. This is a complex procedure as it is essential to guarantee an individual's freedom of choice in electing government, which is universally recognised as one of the basic and fundamental human rights. Such elections are very formal with strict rules specified by law, and these are enforced and closely scrutinised by officials.

From the time of the ancient Athenians, who were the first to hold democratic elections, many efforts have been made to improve the election process. These have been focused on ensuring that the result accurately reflects the desired choices of the voters, and on making the process more convenient and efficient for both voters and officials. A wide range of technological advances have been incorporated into the process, and with the proliferation of the Internet the next logical progression is online elections.

We are currently in the midst of an information revolution, with computers and the Internet permeating almost every aspect of our lives to make day to day activities more convenient. New and fascinating applications are continually being discovered, and this has been facilitated by the protection of information afforded by cryptography. This field has made unprecedented advances ever since the advent of public key cryptography in 1976, and the focus of cryptography has rapidly shifted from military purposes to commercial ones. As a result, techniques such as digital signatures and digital cash are starting to achieve the same legal status as their conventional counterparts.

Over the last 20 years, a substantial amount of research has been carried out in the area of online elections, which is seen as one of the most significant applications of modern cryptography. Protection of the privacy of voters, prevention of electoral fraud and verification of the election results can all be achieved by cryptographic means. Online election schemes have been proposed for a wide variety of elections, and some have even been implemented and used

for conducting elections in several jurisdictions.

However, no completely satisfactory solution has yet been found. It is very difficult to create a protocol that satisfies all the required properties, is scalable for large elections and is efficient for voters. Furthermore, constructing election schemes composed from cryptographic protocols known to be secure does not ensure that the election protocol is also secure. In fact, many proposed schemes have been found to be flawed.

An additional problem is that the Internet is a double-edged sword and many of the properties giving rise to its advantages are also the sources of its pitfalls. Remote applications dramatically improve convenience and the quality of life for honest users, but also expose users to countless new and serious threats. These can compromise the integrity of online elections.

Voting is a unique process that affects society as a whole, not just individuals and organisations, because it essentially governs the way we all live. As a consequence, elections must be protected very carefully and the question of whether online elections are currently suitable remains a contentious issue.

Organisation of the report

This report surveys the development of schemes suitable for conducting public elections online. It provides a thorough overview of the current state of research by classifying and summarising existing schemes. Minimal assumed knowledge is necessary, as this is a complete reference that also covers the cryptographic and mathematical concepts behind online elections. The chapters are ordered in a way that progressively builds the required knowledge.

Understanding how democratic elections work is essential, and the traditional model is described and analysed in Chapter 2. This contains an outline of the most common systems in use and some of their problems. It provides the framework from which the online election model is derived.

Chapter 3 formalises the election model in an online context. The results from the previous chapter are applied to develop requirements to ensure that the democratic process is followed. There are also additional criteria developed to measure how practical the scheme is for voting in realistic situations. These properties are used to compare different online election schemes and to evaluate how successful they are in achieving security and usability. The method used in this report for classifying election schemes is described, and the chapter concludes with a discussion of the advantages of online voting over conventional voting, as well as some of the main problems.

Chapter 4 provides an overview of the necessary mathematical principles. Results in the fields of higher algebra, number theory and complexity theory are presented. These concepts are relevant to intractable problems, which form the basis of many cryptographic assumptions. Most of the suitable problems are found in computational number theory, and three of the main ones are described. Those with a strong mathematical background may wish to pass over this chapter, or refer back to it only if necessary.

Many different cryptographic tools may be employed to achieve the desired properties for online election schemes. Chapter 5 explains the most important cryptographic protocols, and provides examples actually used in the election schemes covered in subsequent chapters.

Detailed descriptions of the most significant online election schemes in the literature are found in Chapters 6–8. Chapter 6 covers schemes based on homomorphic encryption. Chapter 7 covers schemes based on mix nets. Chapter 8 covers schemes based on blind signatures. In each chapter, the schemes are presented in chronological order to illustrate how election protocols have evolved to fulfil the necessary requirements and to improve on efficiency. For each scheme, the cryptographic tools are briefly explained, followed by the actual protocol, the interactive proofs used and then an evaluation of the scheme.

Chapter 9 then compares the advantages and disadvantages of the different election schemes, based on the criteria laid out in Chapter 3. It also explains which of the existing schemes are most suitable for conducting online elections.

Finally, Chapter 10 gives some concluding remarks.

Appendix A summarises the notation used throughout this report.

This work has been drawn from a large number of sources, and I am indebted to the authors of these. The material is expressed using my own wording and phrasing, and I accept full responsibility for any inaccuracies.

Chapter 2

Traditional Elections

For thousands of years, elections have been used to achieve democracy, which literally means "rule by the people". From a conceptual perspective, an election is a basic problem of casting votes and then combining these to make a group decision. However, in practice this is very complicated, and a large number of solutions have been proposed in attempts to realise an ideal election process.

This chapter examines the conventional model for electing government in a democracy. It describes the more common methods for determining the outcome of an election from individual preferences, and covers some of the developments made to improve the process of casting and counting votes. The flaws with the current election process are also discussed.

2.1 Democratic Model

Voting is a procedure allowing a social group to form a collective choice on an issue, based on the sentiments of individuals. It has become synonymous with the election process, where a body of eligible voters, known as the electorate, vote either to select a person for a position of leadership or to decide on a formal proposition. Elections are conducted in accordance with a set of rules, the formality of which is dependent on the purpose of the election and the size of the electorate. These rules specify aspects including the eligibility of voters, the frequency of elections and the voting system for calculating the result. Officials are appointed to administer the election and enforce the integrity of the process. Many organisations hold elections to make important decisions, for example public companies, where the voters are shareholders, and unions, where the voters are members. The most interesting scenario is the election of political government, and this is also the most formal and rigorous election process.

Elections are primarily related to two different forms of democracy. The first is a direct democracy, which was introduced by the ancient Greeks in Athens and is the original type of democracy. Formal decisions relating to government are made directly by the electorate. Voters may decide whether to accept or reject

a proposal, or they may select a proposal from several possible options. This is usually feasible only for small electorates, but may occasionally be used for large electorates in the form of a referendum to decide on particularly important issues.

The second type is a representative democracy, which was introduced by the ancient Romans. The electorate is divided into constituencies, generally based on population and geographical location. Voters in each constituency elect a representative from several possible candidates. These representatives are responsible for making formal decisions relating to government, on behalf of the voters. This is the most common and practical system of government.

In order for the election of government to be truly democratic, it must follow certain principles. The aim of these is to ensure that voters are able to freely express their choices and that the outcome can only be influenced by voting correctly.

1. Any individual satisfying minimum requirements is eligible to vote. These minimum requirements must not be too restrictive, and they are usually based on age and citizenship. Each constituency has an electoral roll, which is a certified list of eligible voters in that constituency.

2. All eligible voters must be given the opportunity to participate in the election, and the voting process must not be a burden on the voter. There are designated polling stations in each constituency. Other measures should made available for voters unable to attend polling stations.

3. Only eligible voters are permitted to cast votes. Almost all elections require a vote to be cast be the actual voter. However, some elections allow a proxy, whereby a voter may nominate another party to cast their vote.

4. All voters are considered to be equal. Each voter can only cast a single vote, and all votes are of equal value. A vote is formal if it is filled out correctly according to the rules for voting. Otherwise, a vote is informal. Voters may also be given the option to abstain, that is to deliberately indicate that they refrain from voting.

5. Voters must cast their votes secretly. This ensures freedom of choice, so that voters can make decisions independent of influences such as fear of retribution and monetary incentive.

6. Votes must not be modified after they have been cast.

7. The votes must be counted fairly and accurately. All formal votes, and only formal votes, contribute to the result of the election.

8. The election process must be verifiable and auditable.

9. The result of the election should reflect the will of the majority, whilst also taking minorities into account.

For the remainder of this chapter, the context used is the election of political government in a representative democracy, as it is the most well defined and familiar setting. In spite of this, all of the concepts are equally applicable to most other elections.

2.2 Voting Systems

The simplest form of voting requires voters to decide between two alternatives, such as selecting a candidate in a two-party political system. This is referred to as YES/NO voting. The more general case involves deciding on a single choice from more than two options, and is referred to as multiway voting. This is more common as political systems tend to have many different groups.

A voting system, also known as an electoral system or electoral formula, defines how the result is calculated from the votes. It is a set of rules designed to translate votes into an outcome that satisfies certain reasonable criteria. In the case of simple YES/NO voting, it is evident that the procedure is straightforward, and the result unequivocally expresses the desired choice of the electorate. What is not as obvious is that for multiway voting, there is no method for producing a result fulfilling the criteria for an ideal outcome. This remarkable result is known as Arrow's impossibility theorem or Arrow's paradox, and was proven by Kenneth Arrow [Arr51], a Nobel Laureate in Economics. Arrow defines a set of reasonable criteria and proves that for certain situations, in order to satisfy one criteria it is necessary to violate another. As a very simple example, it is possible to elect the candidate who is the least preferred by the majority of voters. Some have argued that the term "reasonable" is subjective and the criteria given by Arrow are too restrictive. Nevertheless, it remains an important theoretical result and is widely accepted in the field of voting theory.

Consequently, many electoral systems exist, each with different properties to determine the effectiveness of capturing voter preferences. The suitability of an electoral formula is dependent on many factors, including the purpose of the election, the number of candidates, the size of the electorate and the literacy level of the voters. A sample of the more common systems is given below. Majority and plurality systems are typically used to elect a single candidate, although many can be extended to elect multiple candidates.

2.2.1 Majority voting

As the name suggests, the candidate receiving a majority (more than half) of the votes is elected. When there are more than two candidates, runoffs may be required to eliminate weaker candidates until one receives a majority.

Second ballot Voters may vote for a single candidate. If no candidate receives a majority of the vote, then a second election is held between the two candidates receiving the greatest numbers of votes. This is also known as two-round runoff voting.

Alternate vote This is a preferential system, as voters are required to rank the candidates in order of preference. In some cases, the voter is required to rank all the candidates. In others, a minimum number of preferences must be entered. The counting process takes place over several rounds if necessary. A candidate is immediately elected if a majority of first preference votes is received. Otherwise, during each round the candidate receiving the least number of votes is eliminated and their votes are distributed to the other candidates according to the preferences indicated by the voters. This occurs until a candidate receives a majority of votes. This is a runoff system that avoids the need for further elections, and is thus also known as instant runoff voting.

2.2.2 Plurality voting

These are the most basic and familiar electoral systems. The candidate receiving the greatest number of votes is elected, regardless of whether a majority is reached.

First past the post Voters may vote for a single candidate. Although this is a popular system, it is particularly vulnerable as it is possible for the winner to be the least preferred choice of the majority of voters. This occurs most frequently when there are two strong candidates and other weak candidates. The votes for the weaker candidates are discounted, even though they may be substantial and possibly greater than the number votes for the eventual winner. Hence the preferences of these voters are ignored, and a different candidate may be elected if these are taken into account.

Approval voting Voters may vote for several candidates, essentially casting a YES/NO vote for each candidate. In some cases, the voter may be permitted to vote for as many candidates as desired. In others, there is a maximum limit, or an exact number of candidates must be chosen.

2.2.3 Proportional representation

Multiple candidates are usually elected using proportional representation systems. These systems are much more complicated, because the aim is to elect candidates in proportion to the votes cast. A quota is often used to specify the minimum number of votes required to be elected.

Single transferable vote Voters are required to rank the candidates in order of preference. A candidate is immediately elected if the number of votes received equals or exceeds a quota. For all the elected candidates, the surplus votes in excess of the quota are proportionally distributed to the remaining candidates according to the preferences of the voters. For example, if a remaining candidate receives 10% of the second preferences in votes cast for elected candidates, then they receive 10% of the surplus

votes. This process continues until all the vacancies have been filled. If all the surplus votes are distributed and vacancies still exist, then the candidate receiving the least number of votes is eliminated and their votes are distributed.

2.2.4 Other voting systems

Many mixed systems also exist, combining aspects of several voting systems. Some systems even allow adjustments so that the number of candidates elected may be greater than the original number of vacancies.

Weighted voting systems are a class of election where the votes cast by certain voters have a greater value than that of others. Another variation is the power of veto, where certain voters are given the power to reject a proposal. Neither of these are truly democratic and they are not considered here.

2.3 Vote Casting and Counting Systems

The most elementary form of voting involves a gathering of eligible voters. The election is conducted by simple methods such as a show of hands or acclamation. In this situation the number of voters must be relatively small. It is most suitable when debate is required, for example in parliament, and especially when a consensus must be reached, as is often necessary for a jury.

The most common form of voting is by secret ballot, and this more formal scenario is of primary interest. A ballot is a form used for casting a vote. In secret ballots, the identity of the voter is concealed by submitting the ballot anonymously, thus guaranteeing freedom of choice. Early secret ballots were held by the ancient Greeks, who voted by casting coloured stones into an urn. White stones were used to vote YES and black stones were used to vote NO. Write-in ballots were later introduced, where voters wrote their choice, such as the name of the desired candidate, on tablets.

The first suitable form of secret ballot was the Australian ballot, introduced in 1856. This is a standard ballot paper printed by officials in charge of conducting the election. The ballot lists all the candidates and a vote is indicated by making a mark or writing a number next to the name of the desired candidate. Almost all modern ballots are based on the Australian ballot. They are designed with the aim of enabling voters to unambiguously enter their choices based on the electoral system used, and must be simple enough for the voter to easily comprehend.

There are many mechanisms for counting votes in secret ballot elections. The trend has been towards automating the counting process, to improve efficiency and accuracy. Although the methods for casting votes have changed to reflect this, the majority of systems still use some form of physical ballot, which provides a paper audit trail. The suitability of a system for casting and counting votes depends largely on the voting system used, for example a punch

card system is not compatible with preferential voting. The main systems in use are explained here.

2.3.1 Paper ballot

The ballot is a variety of the Australian ballot. Write-in ballots are occasionally used, but more often they are part of the ballot in other systems, to provide an alternate method of indicating the vote. The votes are usually counted manually, under strict conditions, although optical character recognition systems are sometimes employed. This is still the most common type of ballot.

2.3.2 Mechanical lever

There is no physical ballot, as voters use machines with levers beside the names of the candidates, so there is only a mechanical audit trail. A vote is indicated by pulling the lever corresponding to the desired choice. Each machine records the votes cast, and the tallies of the machines are manually combined. This system is antiquated and is being replaced, but it is still used by a large population in some elections.

2.3.3 Punch card

The ballot is a punch card for old computers. It lists the candidates or numbers corresponding to candidates. A vote is indicated by punching a hole in the appropriate position of the ballot. The votes are counted by machines reading the punch cards. This is another outdated system, and it tends to have a large percentage of informal votes, partly due to the unreliability of the machines and partly because holes are often not punched correctly by voters. Even in manual recounts it can be difficult to determine the vote. Nevertheless, it is still used widely in some elections.

2.3.4 Optical mark recognition

The ballot is a paper listing the candidates and there is a symbol, for example a circle, next to each. A vote is indicated by colouring in the corresponding circle. The votes are counted by optical scanning machines reading the ballots. This system is fairly popular and is replacing older methods.

2.3.5 Computer terminal

There is no physical ballot, as voters use computer terminals with candidates listed on the screen, so there is only an electronic audit trail. This is also known as direct recording. A vote is indicated by using a touch screen, push buttons, a keyboard or some other input device. Each terminal records the votes cast, and the tallies of the terminals are downloaded and combined electronically. This system is gaining in popularity and is particularly favoured by the elderly and disabled.

2.3.6 Electronic systems

At this point, it should be noted that the term "electronic election" is quite ambiguous. It is sometimes used to refer only to online voting through the Internet, but it often pertains to any election using electronic methods. To clarify matters, the following terminology is used in this report to distinguish between systems:

Electronic vote recording Any system using a voting machine to record votes, such as punch cards, optical mark recognition and computer terminals. The results from the machines may be combined either manually or electronically.

Electronic vote counting Any system using computers to count the votes and apply the electoral formula. The votes may be cast using machines or ballot papers, and the data is then entered into the system.

Electronic election Any system using electronic vote recording and counting.

Online election An electronic system by which votes are cast through an online system and computers are used to calculate the result of the election. Although any online system may be used, including interactive digital televisions, touch-tone telephones and mobile telephones, the context for this report is the Internet.

2.4 Typical Election Process

This section describes how a basic election takes place. It is assumed that all the specific rules have been established and officials have been appointed.

Prior to the election, the voters must register their details and be placed on the electoral roll. In addition to this, the date and time of the polling period and the locations of the polling stations must be announced well in advance of the election.

At any time during the prescribed polling period, voters may attend a designated polling station to cast their votes. The election procedure is as follows:

Authorising Officials identify a voter by checking against the electoral roll. If the voter is eligible and has not voted previously, then a blank ballot is issued and the voter is marked on the electoral roll.

Voting The voter enters a private voting booth and fills out the ballot, which is then submitted anonymously into a ballot box.

Counting After the close of the polling period, the ballot boxes are collected. The officials count the votes and calculate the result of the election.

Parallel elections are often used to hold multiple elections simultaneously. A separate ballot is issued for each election. For example in Australia, elections for both the Upper House (Senate) and the Lower House (House of Representatives) of Parliament are held at the same time.

2.5 Problems

Despite all the efforts to improve the election process, there remain many deficiencies. Even disregarding the voting system, the procedure is still flawed and is subject to errors by humans and machines, as well as disruptive actions by sinister parties. This section covers some of the main areas of concern.

2.5.1 Trust

Voters must be confident that the integrity of the election process is guaranteed, otherwise the outcome may not be accepted. Due to the anonymous nature of the election process, this means that a great deal of trust is placed in independent officials to follow the correct procedures to calculate the result. Elections are very different from other types of activities such as banking, as there is no way for a voter to confirm that their vote has been received and will be counted. Also, there is no way for a voter to dispute any discrepancies in counting their vote, since voters have no record like an account statement for financial transactions.

In regions where the political environment is extremely unstable, it may not be possible to trust government officials to protect the democratic principles of an election. A recent example is the 2002 presidential election in Zimbabwe, which was surrounded by government sponsored violence and intimidation aimed at disrupting the democratic process. Most international observers indicated that the election was not legitimate. Throughout history, many instances of rigged elections have occurred, especially in dictatorships.

2.5.2 Fraud

Most forms of widespread fraud require the cooperation of corrupt officials. Nevertheless, even when officials are trustworthy, dishonest parties may still cheat. Election fraud can be difficult to detect, and almost impossible to resolve once it has occurred.

The most common problem is ballot stuffing, where additional ballots are submitted. A corrupt voter may illegally obtain extra blank ballots, and submit them along with the legal ballot. This can be detected if the number of voters who attended the election is known, because there are more votes cast than there should be. Even so, due to the anonymous nature of secret ballot elections, once fraudulent votes have been cast it is almost impossible to discount them.

Impersonation of voters is another type of cheating. In Australia, a voter is simply identified by providing their name and address. It is relatively easy for a dishonest party to physically take the place of another voter and illegally cast a vote. More detailed identification is only required if a "declaration vote" is cast, where the voter must sign an envelope with a declaration stating their entitlement to vote, and the completed secret ballot is placed inside the envelope. This is only necessary under special circumstances, such as when there are discrepancies in the voter's details on the electoral roll or when the electoral roll indicates the voter has already attended the election.

This form of fraud is detected if a dispute is raised when the real voter casts a declaration vote. The authentic vote will be counted, but again there is no way to tell which ballots are illegal.

Other methods may be used for identification, for example signatures, but these can still be circumvented without too much effort. Electoral roll fraud is another major concern, with false identities created and added to the roll. Only biometric identification, for example fingerprints and retina scans, can be considered to be sufficiently reliable to solve these problems. Although biometric methods are already in use, the cost of implementing such a system makes it impractical in most settings. Furthermore, it raises other issues because recording this information on the entire population is seen by many as an invasion of privacy. Electoral rolls are publicly available and commonly used by many organisations to verify the identity of individuals. Hence biometric information may be used for malevolent purposes.

2.5.3 Cost and logistics

Large scale elections are expensive and complicated operations, with strict procedures that must be followed and thoroughly regulated. Ballots need to be printed and equipment must be purchased or hired. Many officials are required at polling stations, to transport the ballots, to count the ballots and to supervise every aspect of the election process. All of these people must receive thorough training prior to the election. Scrutineers may be appointed by every candidate to observe polling, counting and verification of ballots. Recounts and audits may be necessary, and these consume further resources.

Arrangements must be made so that all members of the electorate are given the opportunity to vote in a convenient manner. Special voting provisions must be made for those unable to attend the election, in particular disadvantaged groups such as the elderly, those with disabilities and those living in isolated areas. Mobile polling stations may be provided for those unable to attend the election, for example in hospitals. Absent votes may be cast at polling stations outside of a voter's constituency. When this is not possible, for example if a voter is out of the country at the time of the election, postal votes may be used. Prepoll voting may also be used in special circumstances, to cast a vote prior to the election.

As an indication of the cost and complexity, some approximate figures are given for the 2001 Federal Election in Australia. According to the Australian Electoral Commission, the cost was over $67 million, with $10.5 million spent on advertising and the rest mainly on administration of the election. The voting population was 12.7 million, with a voter turnout of 95%. Of these, 85% of the votes were cast at polling stations, and the remainder were special provision votes. Only paper ballots were used and the counting was done manually. No machinery was used, except for computers to combine the results from each constituency.

2.5.4 Accuracy and efficiency

Even in stable democratic environments, the result of the election may still be controversial when the reliability of the system is questionable. The most obvious example is the 2000 presidential election in the USA. The number of informal votes was greater than the number of votes separating the two main candidates, due to the inaccuracy of the punch card system used by many voters in Florida. This was compounded by the use of the first past the post voting system, as the weakness of this method was clearly exposed.

Manual counting is subject to human error and ballots may be open to interpretation. This occurs if the ballot is not filled out in an entirely correct manner, but the intent of the voter seems to be clear. All these inaccuracies may seem insignificant, and indeed they often are. Nevertheless, there are many cases where they can have a major effect on the result, most notably in closely contested elections involving two strong political parties.

The counting and scrutinising process can be very time consuming. Also, recounts and voting by special provisions, in particular postal votes, may cause a delay in calculating the final result of the election, and this can be in the order of several weeks.

2.5.5 Voter turnout

In compulsory elections, all eligible voters are required to participate in elections as part of their civic duty. The voter turnout is close to 100%, so the result of the election reflects the sentiment of the electorate. The more common situation is that the election is voluntary. In order for an election to be valid, a minimum voter turnout is required, known as a quorum, and this is usually a majority. However, the turnout in voluntary elections tends to be substantially lower than 100%, and in many cases is even less than a majority. This is a significant issue as the result of the election is not necessarily a true reflection of the preference of the entire electorate. Many attempts have been made to rectify this, mainly by making elections more convenient for the voter.

Chapter 3

Online Elections

An online election system is a means for casting votes through the Internet. From a user's perspective, the actual method for casting a vote is essentially the same as for existing computer terminal systems. However, what separates online elections from traditional systems is that a large proportion of the electorate is given the ability to vote remotely, rather than visiting a polling station. Online elections also have the potential to solve many problems with traditional methods, and many of the administration aspects are greatly simplified.

In this chapter the online model is presented, as derived from traditional elections. The properties required by an online scheme to satisfy democratic principles and to be suitable for large scale elections are explained. There is a description of how schemes are classified in this report according to the different techniques used to construct the election protocol. There is also a discussion of both the advantages and the outstanding concerns about online elections.

3.1 Model

At the highest level, an online election scheme is a distributed system consisting of a set of protocols for votes to be cast and counted. The participants in the scheme are communicating, independent processes modelled as probabilistic Turing machines, which are finite state machines where transitions may be randomly chosen from a finite set of alternative next states. Each process executes a protocol and can perform computations in probabilistic polynomial time. The computational model is discussed more thoroughly in Section 4.3.

Online elections are often considered as a subclass of secure multiparty computation protocols, and many papers describing generic multiparty protocols have suggested that elections are a suitable application. These elections are conducted entirely by the voters, and hence a large amount of computation and communication is required. Such schemes are suited to private elections, for example boardroom ballots. However, this approach does not scale well and is extremely impractical for large elections. Consequently, this model is

not considered here. Nevertheless, secure multiparty computation has led to some important and fascinating theoretical results, and the interested reader is referred to [GMW87, BOGW88, CCD88, Bea92].

To avoid interaction between voters and to minimise effort on the part of voters, a more realistic approach is taken. This is based on the client-server paradigm, where servers respond to requests made by clients. It is the most common and effective architecture for providing distributed services on large networks. The remainder of this section explores the main aspects of the online election model. It describes the participants and how they communicate, as well as how votes, ballots and electoral rolls are used.

3.1.1 Participants

There are three types of participants in an online election:

Voter Any party eligible to vote in the election. A voter is a client who initiates the protocol for casting a vote. It is assumed that in general, voters have limited resources with respect to the capacity for computation, communication and storage.

Authority An official conducting the election and assigned by the government, candidates and/or independent parties. An authority is a server responsible for recording votes and then counting them. It is assumed that authorities have very large resources.

Observer Any external party who passively observes part of the election process, for example verifying the count. Strictly speaking, observers are not really participants as they do not actively participate and they are not always involved in the election. They are considered as another type of client.

3.1.2 Communication

The participants communicate by sending messages through communication channels. Various types of channels may be used, and these have different properties. In the ideal models described below, messages always reach their destination without being modified. Also, physical properties such as delays and errors are not considered. In practice, most of these may be implemented by applying cryptographic techniques to public channels like the Internet.

Public channel A channel that can be read by any party. Authentication of the sender may or may not be provided.

Private channel A channel that guarantees messages are perfectly secret from all parties except for the intended recipient. Authentication of the sender is provided. Eavesdropping is not possible, so this is also known as an untappable channel.

Anonymous channel A public channel that guarantees the identity of the sender remains secret from all parties. Messages cannot be traced to the source.

Bulletin board A publicly readable broadcast channel with memory. Authentication of the sender is usually provided, so that a party can exclusively post messages to a designated section by appending to the previous message. No party may modify or delete posted messages.

3.1.3 Votes

A vote is the expression of the desired option of a voter. The structure of a vote is related to the voting system used, and votes are represented numerically. Write-in responses, consisting of arbitrary strings such as the name of a candidate, are also possible. However, these are not considered here because they are not very practical or necessary for most elections. The following types of votes cover essentially all suitable voting systems:

yes/no voting A voter chooses one of two options, indicated either by YES or NO.

Multiway voting A voter chooses a single option from L options, where $L > 2$. The vote is a number between 1 and L. This is also known as $1-L$ voting, and may be thought of as L parallel elections for YES/NO voting, with the restriction that a single YES vote is cast.

***K*-*L* voting** A voter chooses K options from L possible options, where L is constant and K may be variable for each voter. The vote is a list of K distinct numbers, each between 1 and L. When the order of the K options is irrelevant, this is approval voting. It may be thought of as L parallel elections for YES/NO voting, where K is the number of YES votes cast by a voter. When ordering matters, this is preferential voting, which is more complex. $K-L$ voting may be considered as multiway voting, where all L possible combinations of votes are represented and the voter selects one of these. However, this is inefficient as L would be very large.

3.1.4 Ballots

In online elections, or indeed in any system where votes are recorded directly on computers, the ballot is much more abstract than in conventional systems. It is simply a visual representation on the screen, listing all the candidates. Clearly, it is not usually necessary to send all of this information to cast a vote, which can be a single number.

For online election schemes, the term "ballot" does not refer to what is displayed to the voter, and it is a rather vague concept. A ballot is basically considered to be some object, with a specific structure, which is used in the process of casting a vote. In some cases it is created by the authorities, but in

others it is created by the voter. It may contain all the possible options, only the desired vote or something else relating a submitted value to the desired vote. Ballots are often encrypted and most schemes employ interactive proofs to show that the structure of the ballot is valid. Some schemes do not even have ballots, as only the vote needs to be sent.

3.1.5 Electoral rolls

An electronic form of the electoral roll is needed to authenticate eligible voters. This may simply be in the form of unique numbers to identify each voter, which may be associated with passwords. Alternatively, the electoral roll may consist of keys associated with each voter, for authenticating messages using digital signature schemes.

3.2 Typical Election Process

From the perspective of online elections, the most relevant aspect of a traditional election is the vote casting and counting process. Online schemes are only concerned with counting the votes to produce a tally for each possible option. It is then a simple matter to calculate the overall result of the election by applying the appropriate electoral formula to these tallies.

The basic structure of an online election scheme closely resembles that of the traditional model. The protocol is divided into several stages, each of which may consist of several substages. Most schemes have many stages, but at minimum they contain the following:

Initialising The authorities generate and publish all required system parameters. This may include public keys and security parameters.

Voting The voters cast their votes.

Counting The authorities count the votes to produce the necessary tallies.

Protocols often have an authorising stage prior to voting. However, authentication of voters may be provided implicitly by other methods, such as employing bulletin boards for all communication.

There is a prescribed polling period during which the voters may cast their votes. Many schemes have a sequence of steps, each of which involves all participants and must be completed before the next step can commence. In such cases, the steps can be considered as sessions. Usually, each session also has a prescribed time period for participants to complete the necessary actions. For some schemes, each stage is a separate session.

3.3 Security Requirements

For an online election scheme to be secure, it must meet certain requirements. The aim is to be at least as secure as traditional elections, and so these are

more stringent than for other secure online applications, including banking, which essentially focus on the objectives of privacy, authentication, integrity and verification. This is due to the fact that the election process is defined by democratic principles (see Section 2.1). Consequently, the security requirements are derived from these principles, which must be enforced. The following properties are used to evaluate the security of election schemes:

Privacy The vote cast by a voter must be kept secret, in order to ensure that the voter remains anonymous. No collusion of parties can determine any information on the vote cast by a voter without the cooperation of that voter.

Incoercibility Privacy must always maintained regardless of whether the voter cooperates. A voter must not be able to prove the vote that was cast to any party. This property in fact subsumes privacy, and is necessary to prevent coercion and vote buying. It is provided implicitly by traditional elections, as the act of voting occurs within a private voting booth. Outside of this booth, there is no way to prove the vote that was cast. It is particularly difficult to satisfy incoercibility in write-in type elections, where a coercer may force a voter to include a specified string.

This property is also known as receipt-freeness. Note that in certain contexts there is a subtle distinction between receipt-freeness and incoercibility. The latter is defined as a weaker property, which only allows a voter to forge a proof for their vote, for example by creating a fake receipt. However, this does not necessarily ensure that a voter is unable to prove the vote cast if desired. Hence coercion is prevented, but vote buying may still occur. Here the stronger definition of incoercibility is used, and it is considered to be the same as receipt-freeness.

Robustness The election must not be disrupted by any collusion of participants. Hence the system must be capable of recovering from any faulty parties, such as authorities failing or voters not completing the voting process.

Verifiability The tally of the election can be proved to be valid. In other words, the published tally must be equivalent to the actual tally of the election. There are two types of verifiability:

- Public: Any party, including passive observers, can verify that the final tally is consistent with the votes cast correctly. This is also known as universal verifiability.
- Private: Each voter can verify that their individual vote was counted correctly and contributes to the final tally. This is also known as individual or atomic verifiability.

Public verifiability is highly desirable as it allows the election tally to be audited more easily, and it does not rely on all voters to be vigilant in

verifying their individual votes. However, private verifiability is desirable as an additional property for those voters wishing to minimise trust in the authorities.

Correctness The election must be carried out according to certain rules:

- Eligibility: Only eligible voters are permitted to cast votes.
- Uniqueness: Each voter can only cast a single vote.
- Integrity: Votes must not be modified after they have been cast.
- Completeness: All formal votes, and only formal votes, are counted.

Fairness Counting of the votes must not influence the voting. Hence intermediate tallies cannot be revealed by any participant, even if the election is disrupted.

Vote independence The votes must be functionally independent, so that a voter cannot be influenced by any votes previously cast. For example a voter must not be able to cast a vote that is a copy of an existing vote or the opposite of an existing vote (to cancel it out), even without knowing the actual vote.

These requirements are realised by employing various cryptographic techniques, which are discussed in Chapter 5. Online elections rely on cryptographic assumptions, so only computational security can be achieved rather than unconditional security (see Section 5.1.1). However, since it is assumed that processes, including any adversary, have probabilistic polynomial computational power, this is sufficient for all intents and purposes. This is discussed further in Section 4.3.3.

Even within this model, there are limitations on security. In most cases, the security requirements, in particular privacy and robustness, are only resilient to a reasonably sized coalition of dishonest participants, where dishonest is used to refer to both malicious and benign corruption. The definition of a "reasonable size" depends on whether authorities or voters are involved. With respect to authorities, a reasonable size is the number of authorities not exceeding a certain prescribed threshold. With respect to voters, a reasonable size is the total number of voters.

3.4 Usability Properties

In democratic elections, one of the main considerations is that the procedure should require minimum effort on the part of the voter. The election protocol must be simple, efficient and convenient for the voter, who has limited resources. It must also be reasonably efficient for the authorities, even though they are assumed to have very large resources. The efficiency of a protocol is determined by the cryptographic tools chosen and how they interact.

The following properties are used to compare the usability of election schemes:

Computational complexity A measure of the time required to perform calculations in a protocol. This describes how the amount of computation grows with respect to the input size. The worst case complexity is considered and it is expressed in terms of the number of N bit modular multiplications performed, where N is a security parameter. It is assumed that modular exponentiation can be done in linear time.

Communication complexity A measure of data transmitted over communication channels in a protocol. This describes how the amount of communication grows with respect to the input size. The worst case complexity is considered and it is expressed in terms of the number of bits.

Round complexity A measure of the number of rounds of interaction in a protocol. This is expressed in terms of the minimum number of separate sessions a voter must participate in. Note that in some other contexts, this is the number of messages sent.

For computational and communication complexity, the input size usually relates to three possible variables: a security parameter, the number of voters and the number of authorities. Sometimes it can also relate to the number of possible options to vote for. The input size may consist of any combination of these variables, according to which ones the complexity is dependent on. Complexity in general is covered more thoroughly in Section 4.3.1.

Ideally, the round complexity is 1, that is the voter is only required to participate in a single session. This is convenient because voters can act independently and simply "vote and go" at any time during the polling period, regardless of the actions of others. However, when the round complexity is greater, voters must be involved in several sessions, so the voting process may be quite time consuming. This is because for each session, the voter must wait until either the session expires or the necessary tasks have been completed by all other participants.

3.5 Classification of Schemes

At the heart of an election protocol is the need to conceal the correspondence between voters and their votes. This forms the basis for the approach taken to designing election schemes. There are essentially three different approaches, and these are used in this report to classify online election schemes.

Homomorphic encryption The first approach is based on number theoretic techniques used to conceal objects such as votes and ballots. The homomorphic property (see Definition 4.1.4) of certain encryption functions is exploited. Votes are submitted in encrypted form, and the homomorphic property allows these to be combined to form an encrypted tally. Thus rather than decrypting each individual vote, only the tally needs to be decrypted. This has the potential to ensure privacy and make calculating

the tally relatively efficient. These schemes are usually suitable only for YES/NO voting options and they are described in Chapter 6.

Mix nets This approach is based on anonymous channels to conceal the identities of the voters. Such channels are implemented by using mix nets (see Section 5.3.9), which take a set of input messages and produce a set of randomly permuted output messages. Privacy is guaranteed by submitting ballots anonymously. Mix nets with various properties, such as robustness and verifiability, are employed to fulfil the security requirements. These schemes are usually suitable for arbitrary voting options and they are described in Chapter 7.

Blind signatures This is similar to the mix net approach, in that anonymous channels are utilised to conceal the identities of the voters. However, rather than employing mix nets directly, it is simply assumed that an anonymous channel exists and any implementation of such a channel is suitable. The idea is that a voter is authenticated by the authorities and given an anonymous certificate using a blind signature scheme (see Section 5.3.2). This certificate is the actual signature for an object, such as a vote or ballot, so neither the certificate nor the object can be linked to the voter. The voter can then act anonymously by communicating through anonymous channels and using the certificate. These schemes are usually suitable for arbitrary voting options and they are described in Chapter 8.

A comparison of these methods and the advantages of different schemes is provided in Chapter 9.

3.6 Advantages

There are many benefits with online elections, which have the potential to avoid the shortcomings of traditional elections. The most important advantages are discussed in this section.

3.6.1 Convenience

The most obvious advantage is the convenience of voting remotely, and this is of particular benefit to groups such as the elderly, those with disabilities and those in isolated areas or away from home. Naturally, the convenience of voting from the comfort of home can also greatly increase voter turnout. However, it is most significant in situations where groups deliberately disrupt the election process through violent protests and other public acts of intimidation. Online elections give voters the opportunity to participate in the election in a safe environment.

3.6.2 Trust

One of the primary aims of online elections is to minimise the trust required by the voters. This is achieved largely by distributing the power so that rather than

designating a single independent organisation to conduct the election, it can be run by multiple independent parties. Hence groups distrusting the government may be actively involved in the process instead of merely observing to ensure that the correct procedures are followed.

The whole process can be made much more transparent than traditional elections, and the results are verifiable, thus further reducing the necessary trust. A voter can confirm that the authorities have received their vote. A voter is also able to check that the recorded vote matches the vote cast, so it will be counted correctly.

3.6.3 Fraud

Assuming that the electoral roll is correct, most types of cheating are essentially eliminated, as it can be guaranteed that each eligible voter may only cast a single vote. Also, once a vote has been cast, there can be no tampering.

3.6.4 Cost and logistics

The amount of resources and the ensuing cost of conducting elections can be greatly reduced by online schemes, since a large part of the process becomes automated. Rather than achieving this through specialised equipment, online elections utilise the existing infrastructure of the Internet. Although powerful computers are required to conduct the election, most government departments are already in possession of the necessary hardware, and it may be possible to briefly divert these resources for the purposes of an election. Even if such equipment must be purchased, governments can easily afford this as a large amount is already spent on elections. Also, these costs can be amortised because the generic nature of the hardware allows it to be reused for other purposes.

Further savings are made because this system can replace many of the special voting provisions, for example postal ballots. Overall, running an online election requires much less organisational effort.

3.6.5 Accuracy and efficiency

Any system where votes are directly entered on computer terminals has perfect accuracy in counting. Not only does this always produce a correct tally, but it also avoids the need for recounts. In addition to this, these systems can guarantee that all votes cast are formal, and that the choice of a voter is not open to interpretation.

The counting of the votes is much more efficient, and is essentially instantaneous. There is no delay waiting for postal ballots to arrive, or even for ballot boxes to be collected. Hence the tally can be determined as soon as the polling period ends.

3.7 Problems

In spite of all its advantages, online voting poses a new set of problems, similar to that of any online system. Many critics believe that the risks created exceed the benefits. Most of the concerns apply to all computerised methods for casting and counting votes, and the issues are related not to the election schemes themselves, but to how they are implemented and how they operate in the insecure setting of the Internet. The adage "a chain is only as strong as its weakest link" is particularly applicable to the security of a system. Consequently, even though this report focuses on cryptographic election protocols, other aspects must be taken into account.

3.7.1 Trust

Much of the trust required by the voter is shifted from the officials conducting the election to the company developing the election system. Bugs in software are accepted as inevitable, even when good software engineering practices are followed. Most security failures are caused by implementation flaws rather than problems with protocols. However, it is often very difficult for developers, let alone voters and officials, to detect problems with the system. Most election systems are proprietary, and the details are closely guarded by trade secret agreements. Even when the algorithms are revealed, the implementation is still concealed to ensure security of the system. This notion of achieving security through obscurity is recognised as being significantly flawed and open source solutions seem to be a much more sensible approach. Allowing all of the details to be publicly evaluated can greatly diminish the chance that substantial errors remain undetected, especially those which may be exploited by sinister parties. Nevertheless, the average voter must still rely on other qualified individuals to examine the system thoroughly.

Furthermore, trust must still be placed in the officials managing the system. This is another point of weakness in secure online systems, as the overwhelming majority of problems are caused by administrators. Many incidents in ecommerce applications, which are the most widespread, have illustrated this. Fraud can be committed by corrupt officials or by external parties gaining sensitive information from naive officials through "social engineering" methods. Inadvertent errors are also commonplace.

3.7.2 Fraud

The methods for identification in remote voting make large scale fraud easier in some respects. An adversary may compromise systems to obtain private keys for digital signatures. Alternatively, the keys may be obtained directly from voters through coercion and vote buying. This sinister party can then cast any desired votes. Most remote voting systems suffer from this problem, and an analogous situation can be illustrated with postal votes. An adversary may obtain blank

ballots and envelopes with signed declarations, which can then be used to cast any desired votes.

Some schemes employ special hardware, like smart cards, to provide more stringent authentication. However, the use of such hardware makes this technique unsuitable for most large scale elections.

3.7.3 Disruptions

Elections must be completed in a limited time, usually within a day, and one of the main problems is the potential for disruption. These may be as simple and innocent as power blackouts and faults in the communication network. More serious concerns are based on the vulnerability of the Internet, which exposes the voting process to a wide variety of potential threats that are very difficult to prevent. Groups in any part of the world may interrupt the process with less risk and greater anonymity than conventional methods like violent intimidation and stealing physical ballots.

The average user of a home computer is not security conscious, as illustrated by instances of large scale infections by viruses and worms. Such methods can be used by adversarial groups to undermine the election process, especially with logic bombs triggered by the election date.

All online systems are susceptible to communication attacks, most notably denial of service. Malicious parties can target authorities to render them useless. Advanced methods such as delay and replay attacks are also possible. Even more serious disruptions can occur, in the form of hacking to compromise authorities or authenticity attacks to assume the identities of authorities.

3.7.4 Auditability

The lack of paper ballots makes it almost impossible for a voter to verify that the ballot displayed on the screen corresponds to the actual vote cast. It also makes external auditing of the election difficult, as it is not possible to perform a recount.

Suggestions have been made that computerised systems should print a paper ballot, which is then submitted in case a recount is necessary. However, this is only suitable for voting at polling stations. Some online election schemes deliberately give voters an electronic receipt, but this contradicts the incoercibility requirement and such a sacrifice is unacceptable for important elections.

3.7.5 Social considerations

Online voting can be a disadvantage for those unfamiliar with the technology, and these tend to be groups such as the elderly and those with lower levels of education. It is also a problem for those without Internet access, and this mainly consists of people who cannot afford such luxuries. Hence these factors can lead to social inequality, especially in voluntary elections. Although the voter turnout may be increased, this in fact serves to increase the proportion

of representation for those who are more well off. Clearly, this goes against the spirit of democracy.

Chapter 4

Mathematical Primitives

Mathematical principles are the foundation of modern cryptography. In particular, many cryptographic primitives are derived from number theoretic problems. This chapter provides a brief overview of the important concepts. Some key results from the areas of algebra, number theory and complexity theory are presented. These are then applied to formulate hard problems fundamental to cryptography. The concepts and terminology are important as they are commonly used in the literature on cryptography and election schemes.

The material in this chapter comes from several texts. Most of the higher algebra comes from [Chi95], which provides a excellent reference for abstract algebra, and also from a section in [MvOV97]. A thorough introduction to number theory can be found in [IR90] or [Kob94]. [CLR90] has a very good chapter on complexity theory, and [MvOV97] also has a section on this topic. The interested reader is directed to these for examples, proofs of theorems and comprehensive explanations.

4.1 Higher Algebra

Algebraic concepts provide the framework for number theory. Properties of groups and fields are of primary interest, and rings are also covered for the sake of completeness.

Definition 4.1.0.1. A *binary operation* $\circ$ on a nonempty set S is a mapping

$$\circ : S \times S \to S.$$

This has the property of closure. For all $a, b \in S$,

$$a \circ b \in S.$$

Definition 4.1.0.2. Let a^n denote a composed with itself n times: $a^n = \overbrace{a \circ a \circ \cdots \circ a}^{n}$. This operation is known as *exponentiation*.

4.1.1 Groups

Definition 4.1.1.1. A *group* is a set G with a binary operation, denoted arbitrarily by $\circ$ (commonly referred to as multiplication), satisfying the following group axioms:

Axiom 1. Associativity. For all $a, b, c \in G$,

$$(a \circ b) \circ c = a \circ (b \circ c).$$

Axiom 2. Identity. There exists a unique identity element $e \in G$ such that for all $a \in G$,

$$a \circ e = e \circ a = a.$$

Axiom 3. Inverses. For all $a \in G$, there exists an inverse $a' \in G$ such that

$$a \circ a' = a' \circ a = e.$$

A group is denoted by $(G, \circ)$, or more simply G if the context is understood.

Definition 4.1.1.2. A group G is *additive* or a *group under addition* if the group operation $\circ$ is addition, denoted by $+$. A group G is *multiplicative* or a *group under multiplication* if the group operation $\circ$ is multiplication, denoted by $\times$.

Definition 4.1.1.3. A group G is *abelian* if it has the property of commutativity. For all $a, b \in G$,

$$a \circ b = b \circ a.$$

Definition 4.1.1.4. The *order* of a group G is the cardinality, denoted by $|G|$. If $|G|$ is finite, then G is a *finite group*. A finite group of order n is denoted by G_n.

Theorem 4.1.1.5. Abstract Fermat Theorem. Let G be a finite abelian group of order n. Then $a^n = e$ for all $a \in G$.

This is an abstraction of Euler's Theorem (Theorem 4.2.2.7), which is in turn an abstraction of Fermat's Theorem (Theorem 4.2.2.8).

Definition 4.1.1.6. Let G be a group. The *order* of an element $a \in G$ is the smallest positive integer i such that $a^i = e$. If i does not exist then the order of a is defined as ∞.

Theorem 4.1.1.7. Let G be a finite group of order n. Then for all $a \in G$, the order m of a is a positive integer such that $m \leq n$.

Definition 4.1.1.8. Let $(G, \circ)$ be a group and let $H \subseteq G$, $H \neq \emptyset$. Then H is a *subgroup* of G if $(H, \circ)$ is also a group. If $H \subset G$ then H is a *proper subgroup* of G. When the group operation $\circ$ is understood, the notation $H \subseteq G$ is used to indicate that H is a subgroup of G.

Definition 4.1.1.9. A group G is *cyclic* if and only if there exists an element $g \in G$ such that for every $a \in G$, there is an integer i where $g^i = a$. In other words, all the elements of G are produced by powers of a single element g, which is known as a *generator* or a *primitive element*. The cyclic group G is said to be generated by g and is often denoted by $\langle g \rangle$. All cyclic groups are abelian.

Theorem 4.1.1.10. A finite group G of order n is cyclic if and only if there exists an element $g \in G$ such that g is of order n. Note that from Theorem 4.1.1.5, $g^n = e$. Since $g^0 = g^n = e$, there is a repeating sequence with n distinct elements, so

$$G = \left\{ x \mid x = g^i, 0 \leq i \leq n-1 \right\}.$$

Theorem 4.1.1.11. Let G be a finite group of order n and let $a \in G$ be of order m. Then a subgroup $\langle a \rangle$ of G generated by a is a cyclic group of order m.

Theorem 4.1.1.12. Let G be a finite cyclic group of order n. Then all subgroups of G are also cyclic. For each positive integer d such that $d \mid n$, G contains exactly one subgroup of order d.

Theorem 4.1.1.13. Let G be a finite group of order p. If p is prime, then G is cyclic.

4.1.2 Rings

Definition 4.1.2.1. A *ring* is a set R with two binary operations, denoted arbitrarily by $+$ and $\times$ (commonly referred to as addition and multiplication, respectively), satisfying the following ring axioms:

Axiom 1. Additive associativity. For all $a, b, c \in R$,

$$(a + b) + c = a + (b + c).$$

Axiom 2. Additive commutativity. For all $a, b \in R$,

$$a + b = b + a.$$

Axiom 3. Additive identity. There exists a unique additive identity element $0 \in R$, called the *zero element*, such that for all $a \in R$,

$$a + 0 = 0 + a = a.$$

Axiom 4. Additive inverses. For all $a \in R$, there exists an additive inverse $-a \in R$ such that

$$a + (-a) = (-a) + a = 0.$$

Axiom 5. Multiplicative associativity. For all $a, b, c \in R$,

$$(a \times b) \times c = a \times (b \times c).$$

Axiom 6. Distributivity of $\times$ over $+$. For all $a, b, c \in R$,

$$\begin{aligned} a \times (b + c) &= (a \times b) + (a \times c) \quad \text{(Additive)}, \\ (a + b) \times c &= (a \times c) + (b \times c) \quad \text{(Multiplicative)}. \end{aligned}$$

A ring is denoted by $(R, +, \times)$, or more simply R if the context is understood. A ring is in fact an abelian group under addition, with the identity element denoted by 0.

Definition 4.1.2.2. A ring R is *commutative* if it has the property of multiplicative commutativity. For all $a, b \in R$,

$$a \times b = b \times a.$$

Definition 4.1.2.3. A ring R is a *ring with identity* or *unit ring* if it has the property of a multiplicative identity. There exists a unique multiplicative identity element $1 \in R$ where $1 \neq 0$, called the *unit element*, such that for all $a \in R$,

$$a \times 1 = 1 \times a = a.$$

Definition 4.1.2.4. Let R be a ring with identity. An element $a \in R$ is a *unit* or *invertible element* if there exists a multiplicative inverse $a^{-1} \in R$ such that

$$a \times a^{-1} = a^{-1} \times a = 1.$$

Definition 4.1.2.5. Let R be a ring with identity. Then R is a *division ring* if a is a unit for all $a \in R$, $a \neq 0$. In other words, all nonzero elements have multiplicative inverses.

Theorem 4.1.2.6. Let R be a division ring. Then $R - \{0\}$ is a group under multiplication, denoted by R^*.

4.1.3 Fields

Definition 4.1.3.1. A *field* is a set F with two binary operations, denoted arbitrarily by $+$ and $\times$ (commonly referred to as addition and multiplication, respectively), satisfying the following field axioms:

Axiom 1. Additive associativity. For all $a, b, c \in F$,

$$(a + b) + c = a + (b + c).$$

Axiom 2. Additive commutativity. For all $a, b \in F$,

$$a + b = b + a.$$

Axiom 3. Additive identity. There exists a unique additive identity element $0 \in F$, called the *zero element*, such that for all $a \in F$,

$$a + 0 = 0 + a = a.$$

Axiom 4. Additive inverses. For all $a \in F$, there exists an additive inverse $-a \in F$ such that

$$a + (-a) = (-a) + a = 0.$$

Axiom 5. Multiplicative associativity. For all $a, b, c \in F$,

$$(a \times b) \times c = a \times (b \times c).$$

Axiom 6. Multiplicative commutativity. For all $a, b \in F$,

$$a \times b = b \times a.$$

Axiom 7. Multiplicative identity. There exists a unique multiplicative identity element $1 \in F$ where $1 \neq 0$, called the *unit element*, such that for all $a \in F$,

$$a \times 1 = 1 \times a = a.$$

Axiom 8. Multiplicative inverses. For all $a \in F$, $a \neq 0$, there exists a multiplicative inverse $a^{-1} \in F$ such that

$$a \times a^{-1} = a^{-1} \times a = 1.$$

Axiom 9. Distributivity of $\times$ over $+$. For all $a, b, c \in F$,

$$\begin{array}{rcll} a \times (b + c) & = & (a \times b) + (a \times c) & \text{(Additive)}, \\ (a + b) \times c & = & (a \times c) + (b \times c) & \text{(Multiplicative)}. \end{array}$$

A field is in fact a commutative division ring.

Definition 4.1.3.2. The *order* of a field F is the cardinality, denoted by $|F|$. If $|F|$ is finite then F is a *finite field*. A finite field of order n is denoted by $\mathbb{F}_n$. It is also known as a *Galois field*, so it is commonly denoted by $GF(n)$.

Theorem 4.1.3.3. Let F be a field of order n. Then F is finite if and only if n is prime or the power of a prime. Hence for clarity, a finite field of order n is often denoted by $\mathbb{F}_{p^m}$ or $GF(p^m)$, where $p^m = n$ for some prime p and a positive integer m.

Definition 4.1.3.4. Let $\mathbb{F}_n$ be a finite field. $\mathbb{F}_n - \{0\}$ is the *multiplicative group* of $\mathbb{F}_n$, denoted by $\mathbb{F}_n^*$.

Theorem 4.1.3.5. The multiplicative group $\mathbb{F}_n^*$ is a cyclic group of order $n-1$.

4.1.4 Homomorphisms

A homomorphism is a transformation of one set to another that preserves operations on members of the original set.

Definition 4.1.4.1. Let $(G, \circ)$ and $(H, \cdot)$ be groups with identity elements e_G and e_H, respectively. A *group homomorphism* is a function $f : G \to H$ such that

$$f(a \circ b) = f(a) \cdot f(b) \qquad \forall a, b \in G$$

and

$$f(e_G) = e_H.$$

The function f is said to have a $(\circ, \cdot)$-homomorphism property or is $(\circ, \cdot)$-homomorphic.

Definition 4.1.4.2. A *partial function* is a function that is undefined for some elements of its domain. That is, for a partial function $f : G \to H$, not all elements in G are mapped to elements in H.

Definition 4.1.4.3. A *partial homomorphism* is a partial function f that is a group homomorphism, except for the fact that f is a partial function.

This more relaxed notion of homomorphism is extremely useful because in order to attain the desired homomorphic property, it may be necessary to use a partial function. Usually, only the identity elements are possibly undefined in the domain. Thus for the remainder of this report, this definition will be used, so the term "homomorphism" will not distinguish between proper group homomorphisms and partial homomorphisms.

Homomorphism is an important concept that is often exploited by cryptographic protocols. The most important here are homomorphic encryption functions, where the encryption of the compositions equals the composition of the encryptions.

4.2 Number Theory

4.2.1 Integers and division

Theorem 4.2.1.1. The set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ forms a ring, denoted by $\mathbb{Z}$. It is also a cyclic group under addition.

Definition 4.2.1.2. Let a and b be integers where $a \neq 0$. Then a *divides* b if and only if there exists an integer m such that $b = am$. This is denoted by $a \mid b$. Also, a does not divide b is denoted by $a \nmid b$.

When $a \mid b$, it is said that

- b is *divisible* by a,
- b is a *multiple* of a,
- a is a *factor* or *divisor* of b.

Definition 4.2.1.3. Let p be an integer greater than 1. Then p is *prime* if and only if its only positive integer divisors are 1 and itself. Otherwise, p is *composite*. The integer 1 is neither prime nor composite.

Theorem 4.2.1.4. The Fundamental Theorem of Arithmetic. Any positive integer greater than 1 can be represented uniquely as a product of primes. This is equivalent to prime factorisation, and hence this is commonly known as the **Unique Factorisation Theorem**.

Definition 4.2.1.5. From Theorem 4.2.1.4, any integer $n > 1$ can be expressed in the form

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

where k is a positive integer, $p_1, p_2, \ldots, p_k$ are distinct primes and $e_1, e_2, \ldots, e_k$ are positive integers. This is a unique representation of n, disregarding different permutations of $p_1, p_2, \ldots, p_k$. If $p_1 < p_2 < \cdots < p_k$, then this is called the *standard factored form* of n.

Definition 4.2.1.6. Let a and b be integers such that $a \neq 0$ or $b \neq 0$. An integer d is a *common divisor* of a and b if $d \mid a$ and $d \mid b$.

Definition 4.2.1.7. Let a and b be integers such that $a \neq 0$ or $b \neq 0$. The *greatest common divisor* of a and b, denoted by $\gcd(a, b)$, is the largest integer d such that $d \mid a$ and $d \mid b$. Note that $\gcd(0, 0)$ is either undefined or defined to be 0.

Definition 4.2.1.8. The integers a and b are *relatively prime* or *coprime* if $\gcd(a, b) = 1$. In other words, a and b have no common factors apart from 1.

Definition 4.2.1.9. The integers $a_1, a_2, \ldots, a_n$ are *pairwise coprime* if $\gcd(a_i, a_j) = 1$ for $1 \leq i < j \leq n$.

Definition 4.2.1.10. The *Euler Totient Function* $\phi(n)$ is the number of integers a, $1 \leq a < n$, that are relatively prime to n. It has several properties:

1. If p is prime, then $\phi(p) = p - 1$.
2. The function ϕ is multiplicative. If a and b are relatively prime, then $\phi(ab) = \phi(a)\,\phi(b)$.

4.2.2 The integers modulo n

Definition 4.2.2.1. The set of integers modulo n is $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. More accurately, $\mathbb{Z}_n$ is the set of equivalence classes of the integers $0, 1, \ldots, n-1$.

Theorem 4.2.2.2. The set $\mathbb{Z}_n$ is a commutative ring with identity under addition and multiplication modulo n, for $n \geq 2$. Hence all operations are performed modulo n.

Theorem 4.2.2.3. $\mathbb{Z}_n$ is a field if and only if n is prime. Henceforth $\mathbb{Z}_p$ is used to denote a field of integers for some prime p.

Definition 4.2.2.4. The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$. In other words, $\mathbb{Z}_n^*$ is the set of integers in $\mathbb{Z}_n$ relatively prime to n.

Corollary 4.2.2.5. The order of $\mathbb{Z}_n^*$ is $\phi(n)$. This follows from the definition of $\mathbb{Z}_n^*$ and $\phi(n)$ (see Definition 4.2.1.10).

Corollary 4.2.2.6. Let $\mathbb{Z}_p^*$ denote the multiplicative group of the field $\mathbb{Z}_p$, where p is prime. Then $\mathbb{Z}_p^*$ is a cyclic group of order $\phi(p) = p - 1$.

Theorem 4.2.2.7. Euler's Theorem. If a and n are relatively prime, then $a^{\phi(n)} \equiv 1 \pmod{n}$. This theorem provides an extremely simple method for checking if $a \in \mathbb{Z}_n^*$.

Theorem 4.2.2.8. Fermat's Theorem. Let p be prime. If a and p are relatively prime, then $a^{p-1} \equiv 1 \pmod{p}$. This is a specific case of Euler's Theorem.

4.2.3 Congruence

Definition 4.2.3.1. Let a and b be integers and let n be a positive integer. Then a is *congruent* to b *modulo* n if and only if $n \mid a - b$. This is written as

$$a \equiv b \pmod{n}.$$

The integer n is known as the *modulus*. For a fixed modulus, congruency is an equivalence relation because it is a relation that is reflexive, symmetric and transitive:

1. Reflexivity. $a \equiv a \pmod{n}$.
2. Symmetry. If $a \equiv b \pmod{n}$, then $b \equiv a \pmod{n}$.
3. Transitivity. If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.

Theorem 4.2.3.2. If $a \equiv b \pmod{m}$ and $d \mid m$, then $a \equiv b \pmod{d}$.

Definition 4.2.3.3. Let $a \equiv b \pmod{n}$. Then b is known as the *residue* or *remainder* of a modulo n. If $0 < b < n$, then b is called the *common residue* or *least residue* of a modulo n.

Definition 4.2.3.4. Let a and n be coprime. If $x^2 \equiv a \pmod{n}$ has a solution for some integer x, then a is a *quadratic residue* modulo n, also known as a *square* modulo n. Otherwise, a is a *quadratic nonresidue* modulo n.

The general case is known as *higher residues* or r-th *residues*. If $x^r \equiv a \pmod{n}$ has a solution for some integer x, then a is an r-th residue modulo n. Otherwise, a is an r-th nonresidue modulo n.

Definition 4.2.3.5. The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic nonresidues modulo n is denoted by $\overline{Q}_n$.

4.2.4 Linear congruences

Definition 4.2.4.1. A *linear congruence* is a congruence of the form $ax \equiv b \pmod{n}$, where x is an integer variable. All the integers x that satisfy this congruence are solutions.

Theorem 4.2.4.2. Let $d = \gcd(a, n)$. Then $ax \equiv b \pmod{n}$ has solutions if and only if $d \mid b$. If $d \mid b$, then there are d solutions.

Corollary 4.2.4.3. If a and n are coprime, then $ax \equiv b \pmod{n}$ has a unique solution.

Theorem 4.2.4.4. Chinese Remainder Theorem. Let $n_1, n_2, \ldots, n_k$ be pairwise relatively prime and let $a_1, a_2, \ldots, a_k$ be integers. The system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

has a unique solution modulo n, where $n = n_1 n_2 \cdots n_k$.

4.2.5 The Legendre symbol

The Legendre symbol is used to determine whether an integer a is a quadratic residue modulo an odd prime p.

Definition 4.2.5.1. The *Legendre symbol* $\left(\frac{a}{p}\right)$ is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \mid p, \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is a quadratic nonresidue modulo } p. \end{cases}$$

Some of the most important properties are:

1. If $a \equiv b \pmod{p}$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
2. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.
3. $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$.
4. $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$.

Corollary 4.2.5.2. From the definition, it immediately follows that for a fixed modulus:

1. The product of two quadratic residues is a quadratic residue.
2. The product of two quadratic nonresidues is a quadratic residue.
3. The product of a quadratic residue and a quadratic nonresidue is a quadratic nonresidue.

Theorem 4.2.5.3. Euler's Criterion. Let p be an odd prime. If a and p are coprime, then

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

This provides an efficient method for computing the Legendre symbol.

Corollary 4.2.5.4. For any odd prime p, there are $(p-1)/2$ quadratic residues and $(p-1)/2$ quadratic nonresidues modulo p, that is $|Q_p| = \left|\overline{Q}_p\right| = (p-1)/2$. This is because $a^{(p-1)/2} \equiv 1 \pmod{p}$ and $a^{(p-1)/2} \equiv -1 \pmod{p}$ each have $(p-1)/2$ solutions.

Theorem 4.2.5.5. Gauss's Law of Quadratic Reciprocity. Let p and q be distinct odd primes. Then the congruences

$$\begin{aligned} x^2 &\equiv q \pmod{p}, \\ x^2 &\equiv p \pmod{q} \end{aligned}$$

are either both solvable or both unsolvable unless $p \equiv q \equiv 3 \pmod{4}$. Using the Legendre symbol, this can be expressed as

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}.$$

This is one of the most famous results in number theory, and it was originally called the *Theorema Aureum*, the "golden theorem".

4.2.6 The Jacobi symbol

The Jacobi symbol is a generalisation of the Legendre symbol where the modulus can be any odd number, not only an odd prime.

Definition 4.2.6.1. Let n be a positive odd number with prime factorisation $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. The *Jacobi symbol* $\left(\frac{a}{n}\right)$ is defined in terms of the Legendre symbol as

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

Since the Jacobi symbol is the product of Legendre symbols, then $\left(\frac{a}{n}\right) = 0, 1$ or -1. The Jacobi symbol has many similar properties to the Legendre symbol, and some of the most important are:

1. If $a \equiv b \pmod{n}$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.

2. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
3. $\left(\frac{a}{bc}\right) = \left(\frac{a}{b}\right)\left(\frac{a}{c}\right)$.
4. If $a = 2^k b$ where b is odd, then $\left(\frac{a}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{b}{n}\right)$.
5. $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$.
6. $\left(\frac{2}{n}\right) = (-1)^{\left(n^2-1\right)/8}$.
7. If a and b are coprime and odd, then

$$\left(\frac{a}{b}\right)\left(\frac{b}{a}\right) = (-1)^{(a-1)(b-1)/4}.$$

   This is a generalisation of Gauss's Law of Quadratic Reciprocity (Theorem 4.2.5.5).

The Jacobi symbol can be calculated efficiently without factoring n, by repeatedly applying these identities, in a similar fashion to Euclid's algorithm.

Theorem 4.2.6.2. Let n be a composite odd number with prime factorisation $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Then an integer a is a quadratic residue modulo n if and only if a is a quadratic residue modulo each of the prime factors of n. This can be proven using the Chinese Remainder Theorem (Theorem 4.2.4.4).

Corollary 4.2.6.3. If a is a quadratic residue modulo n, then $\left(\frac{a}{n}\right) = 1$ since the Legendre symbol equals 1 for each prime factor of n. However, it is important to note that the converse is not true, that is $\left(\frac{a}{n}\right) = 1$ does not mean a is a quadratic residue modulo n.

Definition 4.2.6.4. The set of integers with a Jacobi symbol equal to 1 is $J_n = \left\{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\right\}$.

Definition 4.2.6.5. An integer a is a *pseudosquare* modulo n if it has a Jacobi symbol $\left(\frac{a}{n}\right) = 1$ and it is also a quadratic nonresidue. The set of pseudosquares modulo n is $\tilde{Q}_n = J_n \cap \overline{Q}_n = J_n - Q_n$.

Theorem 4.2.6.6. Let $n = pq$ where p and q are distinct odd primes. Then $|Q_n| = \left|\tilde{Q}_n\right| = (p-1)(q-1)/4$, so half of the elements in J_n are quadratic residues and the other half are pseudosquares.

4.3 Complexity Theory

The security of modern cryptography is largely based on mathematical problems that are intractable, that is they are computationally infeasible to solve. Consequently, computational complexity theory is one of the most important fields of mathematics with respect to cryptography. This section explains how the difficulty of problems is measured by the computational resources required, and the types of problems that can be considered suitably hard.

4.3.1 Complexity

Complexity is a measure of the amount of resources required to execute an algorithm in terms of a variable input size. Typically, if the input is a single numerical value, then the input size is the number of bits needed to represent the input value. Otherwise, if the input is a list of values, then the input size is the number of elements in the list.

Computational complexity is usually described in relation to the time (number of operations) and space (amount of memory) needed. In this report, computational complexity will be used to refer to time complexity, which is the main concern, and space complexity is not considered. Complexity is also used to measure the communication (number of bits sent) required by an algorithm

It is often very difficult to ascertain the exact complexity of an algorithm, so of most interest is in the worst case complexity. Thus big-O notation is used to describe the asymptotic upper bound on the growth of a function or algorithm, that is the maximum rate at which a function grows with respect to the input size. The following is a more formal definition.

Definition 4.3.1.1. Let f and g be functions and let n be the input size. Then $f(n) = O(g(n))$ if there exist positive constants c and k such that

$$0 \leq f(n) \leq cg(n) \text{ for all } n \geq k.$$

This means that the function f grows no faster asymptotically than $g(n)$, to within a constant factor. The notation $f(n) = O(g(n))$ is read as "f of n is big-O g of n" or "f of n is of order g of n".

Definition 4.3.1.2. Let n be the input size and let c be a positive constant. The complexity of an algorithm is *polynomial* if the worst case complexity is $O(n^c)$. If the complexity is greater than this, then the algorithm is *superpolynomial*.

Definition 4.3.1.3. Let n be the input size and let c be a positive constant. The complexity of an algorithm is *exponential* if the worst case complexity is $O(c^n)$. If the complexity is lower than this, then the algorithm is *subexponential*.

Algorithms with polynomial complexity are generally considered to be efficient, and algorithms with exponential complexity are inefficient. In reality though, polynomial algorithms are only efficient for large input sizes when the polynomial exponent is small, say $O(n^3)$ or better for computational complexity. Also, big-O notation hides the constant coefficient of the polynomial. Thus when comparing two algorithms with the same complexity, there may be a significant difference in practice if the hidden constant for one algorithm is much larger.

In general, these principles relate to both computational and communication complexity. For the remainder of this chapter, the focus is on computational complexity.

4.3.2 Computational complexity classes

Computational complexity classes provide a formal method for classifying various problems depending on the amount of time required to find the solution. The following definitions are provided for the relevant complexity classes, although many more classes exist.

Definition 4.3.2.1. A *decision problem* is a computational problem where the solution is either YES or NO.

Definition 4.3.2.2. A *Turing machine* is a finite state machine with deterministic state transitions. There may be only one possible next state for any given input and current state. This is the traditional computational model.

Definition 4.3.2.3. A *nondeterministic Turing machine* is a Turing machine with nondeterministic state transitions. There may be more than one possible next state for any given input and current state, and the machine simultaneously selects all possible next states. It can be considered as a set of parallel Turing machines that are not able to communicate.

Definition 4.3.2.4. A *probabilistic Turing machine* is a nondeterministic Turing machine where only a single transition is selected from the possible next states. Each transition is randomly chosen based on some probability. This is a more realistic computational model, especially in an adversarial setting.

Definition 4.3.2.5. The complexity class **P** is the set of decision problems that can be solved in *polynomial time*. In other words, there exists a polynomial time algorithm to solve the problem on a Turing machine.

Definition 4.3.2.6. The complexity class **NP** is the set of decision problems that can be solved in *nondeterministic polynomial time*. In other words, there exists a polynomial time algorithm to solve the problem on a nondeterministic Turing machine. An alternative, and equivalent, definition of **NP** is the set of decision problems for which solutions can be verified by a polynomial time algorithm, given some special information.

Definition 4.3.2.7. The complexity class **BPP** is the set of decision problems that can be solved in *bounded error probabilistic polynomial time*. In other words, there exists an expected polynomial time algorithm to solve the problem, with a bounded error probability, on a probabilistic Turing machine. The algorithm is randomised and may produce an incorrect solution, and is known as a *Monte Carlo* algorithm. The probability of success is usually specified to be at least $\frac{2}{3}$ or $\frac{3}{4}$, but it may be any value between $\frac{1}{2}$ and 1. For convenience, bounded error probabilistic polynomial time will simply be referred to as *probabilistic polynomial time*. However, more precise definitions may differentiate between the two.

Definition 4.3.2.8. The complexity class **NP-hard** is the set of decision problems for which finding a solution in polynomial time means that it is possible for

all problems in **NP** to be solved in polynomial time. That is, the problem is at least as hard to solve as all other problems in **NP**. Note that not all **NP-hard** problems are in **NP**.

Definition 4.3.2.9. The complexity class **NP-complete** is the set of decision problems that are in both **NP** and **NP-hard**. These are the most difficult problems to solve in **NP**.

Definition 4.3.2.10. Let A and B be two distinct decision problems. If there exists a polynomial time function for transforming an algorithm for solving B into an algorithm for solving A, then A is *polynomial time reducible* to B, denoted by $A \leq_P B$. This means that B is at least as difficult to solve as A.

Corollary 4.3.2.11. Let A and B be two distinct decision problems.

1. If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$.
2. If $A \leq_P B$, $A \in$ **NP-complete** and $B \in \mathbf{NP}$, then $B \in$ **NP-complete**.
3. If $A \leq_P B$ and $B \leq_P A$, then A and B are computationally equivalent.

From these definitions, clearly $\mathbf{P} \subseteq \mathbf{NP}$, **NP-complete** $\subseteq \mathbf{NP}$ and $\mathbf{P} \subseteq \mathbf{BPP}$. The most famous open problem in theoretical computer science is whether $\mathbf{P} = \mathbf{NP}$. If a polynomial time solution for any **NP-complete** problem can be found, then $\mathbf{P} = \mathbf{NP}$. However, a large number of problems have been proven to be **NP-complete**, and it is highly unlikely that all of these problems can be solved in polynomial time. Thus it is widely believed that $\mathbf{P} \neq \mathbf{NP}$, in which case **P** and **NP-complete** are disjoint sets.

Another open problem is the relationship between **BPP** and **NP**. If $\mathbf{NP} \subseteq \mathbf{BPP}$, then all **NP-complete** problems can be solved in polynomial time with randomised algorithms. This is also highly unlikely, and it is conjectured that the relationship may be $\mathbf{BPP} \subset \mathbf{NP}$.

4.3.3 Intractability

Intractable problems are "computationally infeasible", "hard" or "difficult". Tractable problems are "computationally feasible" or "easy". A more precise definition of intractability is difficult to find, and many different models may be used. The following definition is based on complexity classes and it relies on the reasonable assumptions that $\mathbf{P} \neq \mathbf{NP}$ and $\mathbf{NP} \not\subseteq \mathbf{BPP}$. Although it is quite simple, this is sufficient for all intents and purposes.

Definition 4.3.3.1. A problem is *tractable* if it is in **BPP**. Otherwise, a problem is *intractable*.

The more traditional definition uses the class **P** rather than **BPP**. The reason that **BPP** is used here is because probabilistic algorithms are often more efficient than deterministic ones, so **BPP** is a more realistic computational model for an adversary.

The class **BPP** is very broad, even just **P** is a very large class, and only a reasonably small proportion of problems in this class can be solved efficiently in practice. This makes it safe to assume that any problem not in **BPP** is intractable, as long as the parameters are chosen correctly. In particular, if a problem can be proven to be **NP-complete**, then there is very strong evidence to suggest that it is intractable.

In cryptography, the intractable problems of most interest are in **NP**, largely because the solutions can be verified efficiently (in polynomial time). Algorithms associated with problems harder than those in **NP** are much too inefficient for cryptographic purposes. Thus only problems in **NP** will be considered.

4.3.4 One-way functions

One-way functions form the basis of most cryptographic primitives. They are functions that are easy to compute but are infeasible to invert.

Definition 4.3.4.1. A function $f : X \to Y$ is a *one-way function* if

1. There exists a probabilistic polynomial time algorithm that given any input $x \in X$, produces the output $f(x) \in Y$. In other words, computing f is tractable.

2. There is no probabilistic polynomial time algorithm that given a randomly selected input $y \in \mathrm{Im}(f)$, produces the output $x \in X$ such that $f(x) = y$. In other words, computing f^{-1} is intractable, except for possibly a negligible number of values.

Although it is widely believed that one-way functions exist, this has not been proven.

4.3.5 Trapdoor one-way functions

Trapdoor one-way functions are a mathematical concept fundamental to public key cryptography. These have the additional property that given some extra secret information (the trapdoor), it is possible to efficiently calculate the inverse of the function.

Definition 4.3.5.1. A function $f : X \to Y$ is a *trapdoor one-way function*, or more simply a *trapdoor function*, if

1. f is a one-way function.

2. There exists a probabilistic polynomial time algorithm that given some additional information and any input $y \in \mathrm{Im}(f)$, produces the output $x \in X$ such that $f(x) = y$. In other words, if the trapdoor is known, then computing f^{-1} is tractable.

As the name suggests, these are derived from certain one-way functions for which a secret trapdoor can be developed.

4.4 Number Theoretic Problems

Most of the candidates for intractable problems come from computational number theory. In order to be suitable for cryptographic purposes, they must be one-way functions, so that instances of the problem can be constructed efficiently. The most relevant families of problems are:

1. The integer factorisation problem.
2. The discrete logarithm problem.
3. The quadratic residuosity problem.
4. The subset sum problem. Although this is also commonly referred to as the knapsack problem, the usual definition of the knapsack problem is somewhat different, and the subset sum problem is actually a specific instance of the knapsack problem.

These problems have been studied intensively over a period of many years. No probabilistic polynomial time algorithm is known for solving any of these and they are widely believed to be intractable if the parameters are chosen correctly. The subset sum problem has been proven to be **NP-complete**. The others have been proven to be in **NP** and are conjectured to be somewhere between **BPP** and **NP-complete**. The first three are of primary interest, and they are described later in this section.

Note that these are usually more abstract problems, and they must be rephrased as decision problems, if necessary, for complexity theory to be applied. For example:

1. The integer factorisation problem: Given a positive integer n, are there positive integers p and q such that $n = pq$?
2. The discrete logarithm problem: Let G_n be a finite cyclic group of order n. Given G_n, a generator $g \in G_n$ and an element $y \in G_n$, is there an integer x such that $x = \log_g y$?

The abstract versions of the problems are believed to be more difficult than the corresponding decision problems.

4.4.1 Factoring integers

Definition 4.4.1.1. The Integer Factorisation Problem. Given a positive integer n, find its prime factorisation. That is, find

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

where k is a positive integer, $p_1, p_2, \ldots, p_k$ are distinct primes and $e_1, e_2, \ldots, e_k$ are positive integers.

If n is prime, then the problem essentially becomes deciding whether n is prime or composite, known as primality testing. This is generally an easier problem, which is in **BPP** but not in **P**, and there are many efficient algorithms for finding a solution. However, there is no subexponential time algorithm known for factoring composite integers.

This is one of the most important computational problems because many others are conjectured to be computationally equivalent. Hence if the integer factorisation problem can be proven to be intractable, then there would be strong evidence to suggest that these other problems are also intractable.

4.4.2 Discrete logarithms

Definition 4.4.2.1. Let G_n be a finite group of order n and let $g, y \in G_n$. The *discrete logarithm* of y to base g is the unique integer x, $0 \leq x \leq n-1$, such that $y = g^x$. This is denoted by $x = \log_g y$. The term "discrete" refers to the context being in finite groups, rather than the continuous case.

Definition 4.4.2.2. The Discrete Logarithm Problem. Let G_n be a finite cyclic group of order n. Given G_n, a generator $g \in G_n$ and an element $y \in G_n$, find the unique integer $x = \log_g y$.

If G_n is of infinite order, then the problem problem can be solved in polynomial time. However, for certain families of finite cyclic groups, in particular the multiplicative groups of prime fields and the group of points on an elliptic curve in a finite field, no efficient algorithm is known for finding discrete logarithms.

It is believed that the discrete logarithm problem is generally more difficult than the integer factorisation problem. The most powerful algorithm for computing discrete logarithms is the index-calculus algorithm, which is subexponential (though not polynomial) but it only applies to certain groups.

In the general case of the problem, g is required to be an element of G_n but not a generator, and G_n is required to be finite but not cyclic. However, if G_n is a finite cyclic group and g is a generator, then clearly a solution exists for all $y \in G_n$.

The most relevant setting is in certain subgroups of the multiplicative group of a prime field $\mathbb{Z}_p^*$, where p is a large prime. An appropriate subgroup is chosen by selecting a large prime q such that $q \mid p-1$. Then G_q is the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$. The advantage of using such subgroups is that q can be substantial smaller than p, so that constructing instances of the problem and verifying solutions is more efficient within the smaller group G_q than in $\mathbb{Z}_p^*$. However, the index-calculus algorithms for solving the discrete logarithm problem must still be applied directly in $\mathbb{Z}_p^*$ rather than in G_q.

The discrete logarithm problem in G_q becomes: Given the primes p and q, a generator $g \in G_q$ and an element $y \in G_q$, find the unique integer x, $0 \leq x \leq q-1$ such that $g^x \equiv y \pmod{p}$.

This family of groups is used in this report when describing cryptographic constructs. However, in almost all cases it may be substituted by other families

of finite cyclic groups where computing discrete logarithms is considered to be intractable.

4.4.3 Quadratic residues

Definition 4.4.3.1. The Quadratic Residuosity Problem. Given an odd composite integer n and an integer a with a Jacobi symbol $\left(\frac{a}{n}\right) = 1$, decide if a is a quadratic residue modulo n.

If n is prime, then the Legendre symbol can be efficiently calculated to determine whether a is a quadratic residue. Otherwise if n is composite, then there is no efficient algorithm known for deciding quadratic residues unless the prime factorisation of n is known.

If the prime factorisation of n is known, then according to Theorem 4.2.6.2, the solution can be found efficiently by calculating the Legendre symbol for each factor. Hence the quadratic residuosity problem reduces in polynomial time to the integer factorisation problem. It is believed to be as hard as the integer factorisation problem, but this has not been proven.

In most cases, $n = pq$ is used, where p and q are distinct odd primes. By Theorem 4.2.6.6, the probability of guessing correctly is $\frac{1}{2}$. Also, if p and q are known, then it can easily be determined whether a is a quadratic residue modulo n. This is because

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right) = 1 \Rightarrow \left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1 \text{ or } -1.$$

Thus it is only necessary to compute the Legendre symbol $\left(\frac{a}{p}\right)$ or $\left(\frac{a}{q}\right)$, which can be done efficiently by using Euler's Criterion (Theorem 4.2.5.3).

Chapter 5

Cryptographic Primitives

A wide variety of cryptographic protocols are used in constructing election schemes. Hence in order to understand the schemes, it is essential to be familiar with the underlying concepts. Some of these protocols are well known, whilst others are quite esoteric. This chapter provides descriptions of the most relevant cryptographic tools. Almost all of the examples given are used in the election schemes discussed in the following chapters.

Most of the concepts described here are dealt with more thoroughly in the excellent book by Schneier [Sch96], which is easy to understand and does not delve too much into mathematical details. [MvOV97] provides a more structured and mathematically oriented approach, with many algorithms and analyses of their complexities. [Sti95, Sal96] are also very good references. Much of the material in this chapter can be found in these texts. Where applicable, further references are also included for the interested reader.

5.1 Basic Concepts

5.1.1 Security

A cryptographic protocol is *unconditionally secure* or *perfectly secure* if it is mathematically impossible to gain any partial information on the secret, apart from perhaps the length. Thus the secret cannot be compromised even by an adversary with infinite computational resources. This concept of security is the most strict and is based on Shannon's classical information theory.

More commonly, cryptographic protocols are *computationally secure*, which means that the best methods currently known for breaking the system require unreasonably large computational resources. The amount of resources required must far exceed the resources of any potential adversary. This model of security relies heavily on complexity theory (see Section 4.3) and is based on assumptions such as the existence of intractable problems and one-way functions. Almost all the protocols covered in this report are computationally secure, and this is considered to be a sufficient security model.

Security parameters are employed by computationally secure protocols to determine the level of security achieved. The desired level of security depends on the purpose of the protocol, which can range from protecting government secrets to hiding personal information from younger siblings. If small security parameters are used, it may be possible to break the protocol with existing techniques. Whereas if very large security parameters are used, it may be difficult to break the protocol even with improved computational power in the foreseeable future.

Security parameters may determine the size (the length in bits) of the system parameters, which affect factors such as how difficult it is to solve a particular instance of an intractable problem. Alternatively, they may determine the probability of cheating, which affects the confidence in an interactive proof. In all the protocols presented here, it is assumed that security parameters exist, even if this is not explicitly stated.

5.1.2 Randomness

Most cryptographic protocols need certain numbers, such as system parameters, to be randomly generated. It is important that these numbers are uniformly distributed and independent to ensure that they cannot be predicted. The random numbers are typically required to satisfy some properties, for example they must be elements of a group or large primes of a certain size. Some protocols assume the existence of a *beacon*, which is a publicly readable, trusted source of unbiased random bits. This is commonly employed in interactive proofs. There are many well-known algorithms for generating (pseudo)random numbers suitable for all of these purposes, and they are described in the texts cited at the beginning of this chapter.

5.1.3 Hash functions

Hash functions are one of the basic building blocks for many protocols.

Definition 5.1.3.1. A hash function $h : \{0,1\}^* \to \{0,1\}^k$ is a function that maps a binary string of arbitrary length, known as a pre-image, to a binary string of a fixed length k, known as a hash value, a hash code or simply a hash.

For cryptographic purposes, one-way hash functions are used so that given a pre-image, it is easy to compute the hash. However, the inverse of this process is hard. Also, hash functions are chosen to minimise collisions, that is it should be difficult to find two distinct pre-images producing the same hash. For convenience, the term "hash function" will be used to refer only to such functions suitable for cryptography.

5.1.4 Encryption functions

Encryption functions transform a plaintext message into a disguised ciphertext message. There are three basic categories of encryptions functions:

1. One-way encryption. The encryption algorithm is a one-way function, so there is no way decrypt the ciphertext to obtain the plaintext. Such encryption functions are very efficient and secure. Hash functions are one type of one-way encryption function, as are many commitment schemes (Section 5.3.3).

2. Symmetric key encryption. The plaintext is encrypted using a secret key and the ciphertext is decrypted using the same key, or a key that can easily be derived from the encryption key. Symmetric cryptosystems are best for encrypting data as they are less susceptible to certain attacks than asymmetric cryptosystems. They are also more efficient, because encryption and decryption algorithms using a shared key are less complicated. However, secure key distribution poses a major problem, as the key must be known by both the sender and receiver. If the key is compromised during this process, or at any other point, all messages can be decrypted. Symmetric cryptosystems are not covered here.

3. Public key encryption. This is also known as asymmetric key encryption, as it uses a pair of distinct keys. The plaintext is encrypted using a public key and the ciphertext is decrypted using a private key. It is computationally infeasible to derive the private key from the public key. Public key cryptosystems are suited to key management and digital signatures, and they are also the most useful in constructing other cryptographic protocols. Public key cryptosystems are covered in the next section.

5.2 Public Key Cryptosystems

The seminal paper by Diffie and Hellman [DH76] revolutionised cryptography. The authors introduced the notion of a public key cryptosystem, which is in essence a set of trapdoor one-way functions. Encryption with the public key is straightforward, but decryption without the private key (the trapdoor) is difficult. The encryption algorithms are based on the intractable problems discussed in Section 4.4. These intractable problems are one-way functions, and they are used to formulate trapdoor one-way functions. This section describes the algorithms for key generation, encryption and decryption in the most relevant cryptosystems.

5.2.1 RSA cryptosystem

Rivest, Shamir and Adleman [RSA78] invented the RSA cryptosystem, which was the first public key cryptosystem suitable for providing both encryption and digital signatures. It remains the easiest to implement and is the most widely used. The algorithm is based on the integer factorisation problem (Section 4.4.1), or more precisely the RSA problem, which is considered to be computationally equivalent.

Key generation Each party creates a public and private key pair as follows:

1. Randomly generate two large and distinct primes p and q. For maximum security, p and q should be of equal length.
2. Compute the modulus $n = pq$ and the Euler totient function $\phi(n) = (p-1)(q-1)$. The primes p and q are discarded, but they must not be revealed.
3. Randomly select the encryption key e, $1 < e < \phi(n)$, such that e and $\phi(n)$ are relatively prime.
4. Compute the decryption key d, $1 < d < \phi(n)$, such that

$$d = e^{-1} \bmod \phi(n).$$

The public key is (n, e) and the private key is d.

Encryption A message $m \in \mathbb{Z}_n$ is encrypted using the public key (n, e) by computing

$$c = m^e \bmod n.$$

For large messages where $m \geq n$, m is divided into numerical blocks such that each is less than n. Then the ciphertext for each block is computed.

Decryption The plaintext m is recovered from a ciphertext c by computing

$$m = c^d \bmod n.$$

Decryption works because of the following: Since

$$d = e^{-1} \bmod \phi(n),$$

then

$$ed \equiv 1 \pmod{\phi(n)}$$

so there is a positive integer k such that

$$\begin{aligned} ed &= k\phi(n) + 1 \\ &= k(p-1)(q-1) + 1. \end{aligned}$$

Therefore,

$$\begin{aligned} m^{ed} &\equiv m^{k(p-1)(q-1)+1} \pmod{p} \\ &\equiv \left(m^{p-1}\right)^{k(q-1)} m \pmod{p}. \end{aligned}$$

By Fermat's Theorem (Theorem 4.2.2.8), $m^{p-1} \equiv 1 \pmod{p}$ so

$$\begin{aligned} m^{ed} &\equiv 1^{k(q-1)} m \pmod{p} \\ &\equiv m \pmod{p}. \end{aligned}$$

Similarly,

$$m^{ed} \equiv m \pmod{q}.$$

Since $n = pq$, where p and q are distinct primes, then by the Chinese Remainder Theorem (Theorem 4.2.4.4)

$$m^{ed} \equiv m \pmod{n}.$$

Finally,

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \pmod{n}.$$

From this it can be seen that equivalently, a message m may be encrypted with d and decrypted with e.

5.2.2 Goldwasser-Micali cryptosystem

Deterministic public key encryption algorithms, such as RSA, always encrypt a plaintext m as the unique ciphertext c, for a given public key. This leaks partial information which may be used to develop an attack. For example, in a chosen plaintext attack, an adversary may determine m by encrypting a chosen plaintext m' as the ciphertext c', and then checking whether $c = c'$.

In order to eliminate this, Goldwasser and Micali [GM82] invented probabilistic encryption, and this is the most secure type of cryptosystem known. As the name suggests, the encryption algorithm is probabilistic rather than deterministic, so that a given plaintext may be encrypted in a large number of ways, with the ciphertext being selected randomly. However, the decryption of a given ciphertext corresponds to a unique plaintext. A consequence of this one-to-many mapping is message expansion, that is the ciphertext is longer than the corresponding plaintext.

The original cryptosystem is based on the quadratic residuosity problem (Section 4.4.3) and is described here. It exploits the fact that the product of a quadratic residue and a quadratic nonresidue is a quadratic nonresidue.

Key generation Each party creates a public and private key pair as follows:

1. Randomly generate two large and distinct primes p and q. For maximum security, p and q should be of equal length.
2. Compute the modulus $n = pq$.
3. Select an integer $y \in \tilde{Q}_n$, so y is a pseudosquare modulo n. In other words, y is a quadratic nonresidue modulo n and the Jacobi symbol $\left(\frac{y}{n}\right) = 1$.

The public key is (n, y) and the private key is (p, q).

Encryption A message m is encrypted using the public key (n, y) as follows:

1. The message m is represented as a binary string of length l, $m = m_1, m_2, \ldots, m_l$.

2. For each bit m_i, $1 \leq i \leq l$, randomly select $r_i \in \mathbb{Z}_n^*$.
3. If $m_i = 1$, then compute a pseudosquare modulo n

$$c_i = yr_i^2 \bmod n.$$

Otherwise, compute quadratic residue modulo n

$$c_i = r_i^2 \bmod n.$$

The ciphertext is the l-tuple $c = (c_1, c_2, \ldots, c_l)$. There is message expansion by a factor of $\log_2 n$ bits.

Decryption The plaintext m is recovered from a ciphertext c as follows:

1. For each c_i, $1 \leq i \leq l$, compute the Legendre symbol $\left(\frac{c_i}{p}\right)$.
2. If $\left(\frac{c_i}{p}\right) = 1$, then c_i is a quadratic residue modulo n, so $m_i = 0$. Otherwise, c_i is a pseudosquare modulo n, so $m_i = 1$.

The recovered message is $m = m_1, m_2, \ldots, m_l$.

5.2.3 ElGamal cryptosystem

ElGamal [ElG85] developed a probabilistic public key cryptosystem based on the discrete logarithm problem (Section 4.4.2). It works for any family of groups for which the discrete logarithm problem is considered intractable. The system described here uses G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$.

Key generation Each party creates a public and private key pair as follows:

1. Randomly generate two large primes p and q such that $q \mid p - 1$.
2. Randomly select a generator $g \in G_q$.
3. Randomly select an integer $x \in \mathbb{Z}_q^*$ and compute

$$h = g^x \bmod p.$$

The public key is (p, g, h) and the private key is x. The system parameters are p, q and g, which can be shared by a group of entities.

Encryption A message $m \in G_q$ is encrypted using the public key (p, g, h) as follows:

1. Randomly select an integer $r \in \mathbb{Z}_q$.
2. Compute

$$\begin{aligned} a &= g^r \bmod p, \\ b &= h^r m \bmod p. \end{aligned}$$

The ciphertext is $c = (a, b)$. There is message expansion by a factor of 2.

For large messages where $m \geq n$, m is divided into numerical blocks such that each is less than n. Then the ciphertext for each block is computed.

Decryption The plaintext m is recovered from a ciphertext c by computing

$$m = a^{-x} b \bmod p.$$

Decryption works because

$$a^{-x} b \equiv (g^r)^{-x} h^r m \equiv g^{-xr} (g^x)^r m \equiv m \pmod{p}.$$

5.2.4 Threshold ElGamal cryptosystem

A threshold cryptosystem enables the private key to be shared amongst a group, so that a subset of this group must collaborate to decrypt a message. It relies on the concept of threshold secret sharing as described in Section 5.3.7.

For a threshold ElGamal cryptosystem, the public key is (p, g, h) and the private key is x as in the the normal ElGamal cryptosystem. The encryption algorithm also remains the same. However, in this case x is distributed amongst n group members such that any threshold of t or more members can decrypt messages. Shares of x may be created and distributed by a trusted party using Shamir's scheme, as outlined in Section 5.3.7. A method avoiding the trusted party is discussed in [Ped92b].

The decryption algorithm is modified to prevent the private key x from being reconstructed when decrypting messages. Let there be n shares of the private key, x_j, $1 \leq j \leq n$. Let a message m be encrypted as the ciphertext $c = (a, b)$, where

$$\begin{aligned} a &= g^r \bmod p, \\ b &= h^r m \bmod p. \end{aligned}$$

The ciphertext is decrypted as follows:

1. Each group member A_j, $1 \leq j \leq n$, reveals

   $$y_j = a^{x_j} \bmod p$$

   and proves that

   $$\log_g h_j = \log_x y_j.$$

   A possible proof is described in Section 6.5.3.1 as part of an election scheme.

2. Let S be a set of any t group members collaborating to reconstruct the secret. Assuming that all these members have given valid proofs, the message is decrypted as

   $$m = \left(b / \prod_{j \in S} y_j^{P_j(0)} \right) \bmod p$$

where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k-j}.$$

5.3 Protocols

5.3.1 Digital signatures

A digital pseudonym is a unique identifier for an individual, created for a particular purpose and used to authenticate the holder. This can be realised by using digital signatures to sign a message. A signature must satisfy the following properties:

1. Authenticity. A signature convinces recipients that a message was genuinely signed by the signer. It cannot be forged.
2. Uniqueness. A signature applies only to a particular message. It cannot be reused for another message.
3. Integrity. A message may not be modified after it is signed.
4. Verifiability. A signature provides proof that a message was signed. It cannot be repudiated by the signer.

There are two classes of digital signature schemes.

1. Digital signature schemes with appendix. The original message is required as an input for verification. These schemes usually accept messages of arbitrary length and sign a hash value of the message.
2. Digital signature schemes with message recovery. The original message is not required for verification, as the message is recovered from the signature. These schemes usually accept messages of a fixed length and are only practical for short messages.

A digital signature scheme consists of algorithms for signature generation and signature verification. The signature depends on a secret known only to the signer, and so schemes can be based on symmetric cryptosystems or public key cryptosystems. The latter is of most interest here, and in this case a message is signed with the private key. The public key is the pseudonym, which is used to verify the signature of the holder. Both the RSA and ElGamal cryptosystems can be used to provide digital signatures, and the RSA scheme is described here.

Suppose the signer has a public key (n, e) and a private key d. As explained in Section 5.2.1, a message may in fact be encrypted by d and decrypted by e. This property is exploited to create a signature scheme with message recovery.

Signature generation The signature s is created for a message $m \in \mathbb{Z}_n$ by using the private key d to compute

$$s = m^d \bmod n.$$

Signature verification The message m is recovered from a signature s by using the public key (n, e) to compute

$$m = s^e \bmod n.$$

5.3.2 Blind signatures

Chaum [Cha83] developed blind signatures to allow a requester R to obtain a digital signature for a message m without revealing any information about m to a signer S. Since m is signed blindly, there is no correlation between the signature and the operations performed during the signing process. Thus it is not possible for S or any other party to derive the exact correspondence between m and R, and this affords the property of unlinkability. This guarantee of anonymity is essential in protecting privacy, and it has particular importance in relation to the notion of "Big Brother" from George Orwell's fictional novel *1984*. Hence blind signatures have significant value, and extensive research has been undertaken in this area, especially in the field of electronic commerce for untraceable payments such as digital cash.

The principle behind blind signatures is to conceal m by applying a randomly chosen blinding factor x. The simple scheme presented here uses the RSA signature.

Suppose S has a public key (n, e) and a private key d. A blind signature s is obtained for m in the following manner:

Blinding R randomly generates $x \in \mathbb{Z}_n$ and blinds m by computing a blinded message

$$b = mx^e \bmod n.$$

Signature generation S signs the blinded message b as

$$s' \equiv (mx^e)^d \pmod{n}.$$

Unblinding To retrieve s, R unblinds s' by computing

$$s = s'x^{-1} \bmod n$$

to give

$$s = m^d \bmod n,$$

which is a standard RSA signature.

The unblinding works because of the following: The signature for the blinded message is

$$s' \equiv (mx^e)^d \equiv m^d x^{ed} \pmod{n}.$$

As shown in Section 5.2.1,

$$x^{ed} \equiv x \pmod{n}$$

so

$$s' \equiv m^d x \pmod{n}.$$

Thus

$$s \equiv s' x^{-1} \equiv m^d \pmod{n}.$$

There exist many variations and more complicated schemes. One of the most interesting is the method from [CFN89], which is to prevent double spending in untraceable digital cash. The signature is an unreusable certificate generated using a unique identifier. When using this certificate, a proof must be provided that it is valid. If the certificate is used only once, then anonymity is maintained as the identifier cannot be retrieved. However, if the certificate is used multiple times, then the identifier is revealed and provides proof that this has occurred.

5.3.3 Bit commitment

Bit commitment is a protocol initially proposed by Blum [Blu82] to solve the problem of coin flipping over a telephone. However, it has a wide range of applications and is used by many protocols, in particular for zero knowledge proofs. It can essentially be considered as analogous to delivering a message in a locked box, which enables the message to be concealed from the receiver whilst also guaranteeing the sender is committed to this message.

Practical bit commitment schemes are able to commit to messages consisting of multiple bits, and are hence more simply known as commitment schemes. The general scheme is an interactive protocol between a sender S and a receiver R, and consists of initialisation, commitment and opening phases.

Initialisation Set up the required parameters, which may be used to make multiple commitments.

Commitment S commits to a message m by constructing a commitment blob b, which is generated from m in such a way that R may not gain any information about m from b.

Opening S reveals m to R and provides some proof for R to verify that b can be opened as m.

Commitment schemes may be made non-interactive by employing public parameters generated either by a trusted third party or randomly by the participants. Hence the commitment and opening phases each require only a single message to be sent from S to R.

The following non-interactive scheme is suggested by Pedersen [Ped92a] and is based on the discrete logarithm problem.

Initialisation The scheme uses G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$.

1. Randomly generate two large primes p and q, such that $q \mid p-1$.

2. Randomly select elements $g, h \in G_q$ such that $\log_g h$ is not known by any party.
3. The public parameters are p, q, g and h.

Commitment S commits to a message $m \in \mathbb{Z}_q$ by randomly generating $r \in \mathbb{Z}_q$ to construct the blob

$$b = g^m h^r \bmod p.$$

S sends b to R. If $b \in G_q$, then it is a valid commitment. R verifies this by checking that $b \in \mathbb{Z}_p^*$ and $b^q \equiv 1 \pmod{p}$.

Opening S reveals m and r. R verifies this is a valid opening of b by checking that $m, r \in \mathbb{Z}_q$ and $b = g^m h^r \bmod p$.

5.3.4 Trapdoor commitment

Trapdoor commitment schemes allow the commitment blob to be opened ambiguously given the knowledge of some special trapdoor information. The following example is a non-interactive scheme based on the discrete logarithm problem and comes from [JSI96, BCC88, Fis01], all of which contain several other constructions with different properties.

Initialisation The scheme uses G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$. The parameters are the same as those for the ElGamal cryptosystem (Section 5.2.3).

1. Randomly generate two large primes p and q such that $q \mid p - 1$.
2. Randomly select a generator $g \in G_q$.
3. Randomly select an integer $x \in \mathbb{Z}_q^*$ and compute

$$h = g^x \bmod p.$$

4. The public parameters are p, q, g and h. The trapdoor is x, which is the private key in the ElGamal cryptosystem.

Commitment S commits to a message $m \in \mathbb{Z}_q$ by randomly generating $r \in \mathbb{Z}_q$ to construct the blob

$$b = g^m h^r \bmod p.$$

S sends b to R. If $b \in G_q$, then it is a valid commitment. R verifies this by checking that $b \in \mathbb{Z}_p^*$ and $b^q \equiv 1 \pmod{p}$.

Opening S reveals m and r. R verifies this is a valid opening of b by checking that $m, r \in \mathbb{Z}_q$ and $b = g^m h^r \bmod p$.

The trapdoor x can be used to open b as any $m', r' \in \mathbb{Z}_q$ where

$$m + xr \equiv m' + xr' \pmod{q}.$$

To open b as $m' \in \mathbb{Z}_q$, compute

$$r' = \left(r - (m' - m)\, x^{-1}\right) \bmod q.$$

Then

$$\begin{aligned} b &\equiv g^m h^r \pmod{p} \\ &\equiv g^m h^{r' + (m' - m)x^{-1}} \pmod{p} \\ &\equiv g^m h^{r'} \left(g^x\right)^{(m' - m)x^{-1}} \pmod{p} \\ &\equiv g^m h^{r'} g^{m' - m} \pmod{p} \\ &\equiv g^{m'} h^{r'} \pmod{p}. \end{aligned}$$

5.3.5 Chameleon commitment

Another type of commitment scheme is chameleon commitment, which allows the verifier to lie about the opening of the blob, although the blob can only be opened correctly in a unique way. The scheme presented here comes from [BKK90] and requires that a separate commitment is made for each individual bit.

Initialisation The scheme uses G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$.

1. Randomly generate two large primes p and q such that $q \mid p - 1$.
2. Randomly select a generator $g \in G_q$.
3. Both R and S know q, which remains secret.
4. R randomly selects an element $s \in \mathbb{Z}_q$ and reveals $x = g^s \bmod p$ to S.
5. The public parameters are p and g.

Commitment S commits to a bit $m \in \{0, 1\}$ by randomly generating $r \in \mathbb{Z}_q$ to construct the blob

$$b = x^m g^r \bmod p.$$

Thus if $m = 0$, then

$$b = g^r \bmod p$$

and if $m = 1$ then

$$b = xg^r \bmod p.$$

S sends b to R. If $b \in G_q$, then it is a valid commitment. R verifies this by checking that $b \in \mathbb{Z}_p^*$ and $b^q \equiv 1 \pmod{p}$.

Opening S reveals r. R computes $g^r \bmod p$ and $xg^r \bmod p$ to find $m \in \{0, 1\}$.

R may lie about the opening by claiming that r' was revealed. This uses the fact that $x = g^s \bmod p$. If $m = 0$, then

$$r' \equiv r + s \pmod{q}$$

so

$$b \equiv g^{r'} \equiv g^{r+s} \equiv g^r g^s \equiv xg^r \bmod p.$$

If $m = 1$, then

$$r' \equiv r - s \pmod{q}$$

so

$$b \equiv xg^{r'} \equiv xg^{r-s} \equiv xg^r g^{-s} \equiv xg^r x^{-1} \equiv g^r \bmod p.$$

5.3.6 Secret sharing

This notion was independently introduced by Blakley [Bla79] and Shamir [Sha79] and it is a protocol for sharing a secret message m between n parties so that a subset of the group must collaborate to reconstruct m. The basic approach is simply to divide m into n pieces, called shadows or shares, and to distribute these to members of the group. A secret sharing scheme is perfectly secure if any unauthorised subset of shares has no advantage in discovering the secret over an external party with no shares. Shamir's scheme is perfect, and Blakley's scheme can be modified to be perfect.

The original motivation was secure key management, and secret sharing was designed to safeguard against the loss of cryptographic keys without duplicating them. However, the distribution of partial knowledge has the advantages of increasing both reliability (fault tolerance) and security, and these desirable properties make secret sharing extremely valuable in many contexts.

The most elementary and efficient secret sharing scheme uses modular addition.

Distributing shares Let i be a large positive integer. A dealer D shares a secret message $m \in \mathbb{Z}_i$, amongst n parties.

Create n shares $s_j \in \mathbb{Z}_i$, $1 \leq j \leq n$, such that

$$m = \left(\sum_{j=1}^{n} s_j \right) \bmod i$$

and send these secretly to the members of the group.

Reconstructing the secret The secret is reconstructed by summing the shares modulo i.

5.3.7 Threshold secret sharing

Threshold secret sharing is the most common method of specifying the minimum subset of the group required to reveal the shared secret. The message m is divided into n pieces such that it can be recovered by a threshold of t shares or more, where $t \leq n$. This is known as a (t, n)-threshold scheme (or t-out-of-n scheme), and the choices for t and n are a compromise between reliability and security.

Both of the original methods by Blakley and Shamir are in fact threshold schemes. Shamir's scheme is based on polynomial interpolation in finite fields, and the method described here is for the finite field $\mathbb{Z}_p$, where p is prime.

Distributing shares A dealer D shares a secret message $m \in \mathbb{Z}_p$ amongst n parties, where $n < p$. Any $t \leq n$ shares can be used to reconstruct m.

1. Construct a polynomial P of degree $t-1$ by randomly selecting $t-1$ independent coefficients $c_i \in \mathbb{Z}_p$, $1 \leq i \leq t-1$, and setting m as the constant term. The polynomial is

   $$P(x) = m + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}$$

   and the secret is $m = P(0)$.

2. Create n shares s_j, $1 \leq j \leq n$, such that

   $$s_j = P(j) \bmod p$$

   and send these secretly to the members of the group. Each share s_j describes a point (j, s_j) on P.

Reconstructing the secret A set of t or more shares forms a system of linear equations which are solved to reveal the secret m.

Given t distinct points (x_j, y_j), $1 \leq j \leq t$, on an unknown polynomial P of degree less than t, the coefficients of P can be found by the Lagrange interpolation formula

$$P(x) = \sum_{j=1}^{t} P_j(x)\, y_j$$

where

$$P_j(x) = \prod_{k=1, k \neq j}^{t} \frac{x - x_k}{x_j - x_k}.$$

Let S be a set of any t group members collaborating to reconstruct the secret. Since $m = P(0)$ and $(j, s_j) = (x_j, y_j)$, the Lagrange interpolation formula simplifies to

$$m = P(0) = \sum_{j \in S} P_j(0)\, s_j$$

where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k-j}$$

are constants which can be precomputed and made public. Hence m is a linear combination of the t shares s_j.

5.3.8 Verifiable secret sharing

The basic secret sharing scheme requires a trusted dealer to distribute the shares. To minimise trust in the dealer, verifiable secret sharing is an extension that allows each party receiving a share of the secret to verify that their share is valid, without collaborating with others to reveal the secret. Pedersen [Ped92a] presents an efficient non-interactive scheme, where only the dealer is required to send messages. It combines Shamir's (t, n)-threshold scheme with the basic commitment scheme described in Section 5.3.3.

Initialisation The scheme uses G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$.

1. Randomly generate two large primes p and q, such that $q \mid p-1$.
2. Randomly select elements $g, h \in G_q$ such that $\log_g h$ is not known by any party.
3. The public parameters are p, q, g and h.

Distributing shares A dealer D shares a secret message $m \in \mathbb{Z}_q$ amongst n parties, where $n < q$. Any $t \leq n$ shares can be used to reconstruct m.

1. Commit to m by randomly generating $r \in \mathbb{Z}_q$ to construct the blob

   $$b = g^m h^r \bmod p$$

   and then publishing b.
2. Construct a polynomial P of degree $t-1$ by randomly selecting $t-1$ independent coefficients $c_i \in \mathbb{Z}_q$, $1 \leq i \leq t-1$, and setting m as the constant term, so

   $$P(x) = m + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}.$$

3. Construct a polynomial Q of degree $t-1$ by randomly selecting $t-1$ independent coefficients $b_i \in \mathbb{Z}_q$, $1 \leq i \leq t-1$, and setting r as the constant term, so

   $$Q(x) = r + d_1 x + d_2 x^2 + \cdots + d_{t-1} x^{t-1}.$$

4. Commit to the coefficients of P and Q by constructing the blobs

   $$b_i = g^{c_i} h^{d_i} \bmod p$$

   and publishing these values.

5. Create n shares $s_j = (m_j, r_j)$, $1 \leq j \leq n$, such that

$$\begin{aligned} m_j &= P(j) \bmod q, \\ r_j &= Q(j) \bmod q \end{aligned}$$

and send these secretly to the members of the group.

Verifying shares Each group member verifies that their share $s_j = (m_j, r_j)$ is valid by using all the published blobs to check that

$$g^{m_j} h^{r_j} \equiv b \prod_{i=1}^{t-1} b_i^{j^i} \pmod{p}.$$

The principle behind this is that if all the shares are consistent, then there exist unique polynomials P and Q of degree $t-1$ such that

$$\begin{aligned} P(j) &\equiv m_j \pmod{q}, \\ Q(j) &\equiv r_j \pmod{q} \end{aligned}$$

for all j, $1 \leq j \leq n$. This is illustrated explicitly by expanding the right hand side of the equation:

$$\begin{aligned} b \prod_{i=1}^{t-1} b_i^{j^i} &\equiv g^m h^r \prod_{i=1}^{t-1} \left(g^{c_i} h^{d_i}\right)^{j^i} \pmod{p} \\ &\equiv g^m h^r \prod_{i=1}^{t-1} g^{c_i j^i} h^{d_i j^i} \pmod{p} \\ &\equiv \left(g^m \prod_{i=1}^{t-1} g^{c_i j^i}\right) \left(h^r \prod_{i=1}^{t-1} h^{d_i j^i}\right) \pmod{p} \\ &\equiv \left(g^{m + \sum_{i=1}^{t-1} c_i j^i}\right) \left(h^{r + \sum_{i=1}^{t-1} d_i j^i}\right) \pmod{p} \\ &\equiv g^{P(j)} h^{Q(j)} \pmod{p} \\ &\equiv g^{m_j} h^{r_j} \pmod{p}. \end{aligned}$$

Note that the exponents are all taken to be modulo q.

Reconstructing the secret Since m and r are both shared secrets, they can each be reconstructed as in Shamir's scheme. The message m can be verified by checking that the blob b is valid.

5.3.9 Mix nets

Mix nets were created by Chaum [Cha81] as a computationally secure implementation of an anonymous channel, which ensures that the sender of a message is untraceable. This has many uses, including anonymous email and hiding traffic patterns.

A series of mixes or mix servers are used to conceal the correspondence between inputs and outputs. Each mix processes a group of encrypted messages by decrypting and permuting them. The original protocol is very basic and uses a public key cryptosystem such as RSA.

Let $M_1, M_2, \ldots, M_n$ be a cascade of n mixes. Let M_j denote an arbitrary mix and let E_j denote encryption with the public key of M_j. The symbol $\|$ is used to denote concatenation.

1. In order to send a message m through the mix net, a sender first randomly generates n binary strings $r_1, r_2, \ldots, r_n$. Then m, which has presumably already been encrypted with the public key of the recipient, is encrypted with the public keys of the mixes to give $E_1(r_1 \| E_2(r_2 \| \cdots E_n(r_n \| m)))$. The randomly selected binary strings are used to eliminate the chosen plaintext attack, which verifies a guess that $x = y$ by checking if $E(x) = E(y)$. This encrypted message is sent to M_1.

2. Each mix M_j receives a sequence of messages, then removes the corresponding layer of encryption to produce a set of messages of the form $E_{j+1}(r_{j+1} \| E_{j+2}(r_{j+2} \| \cdots E_n(r_n \| m)))$. This batch is shuffled either by randomly permuting the order or sorting the items lexicographically. The batch is then forwarded to M_{j+1}.

3. The final mix M_n sends the messages to their destinations.

As long as at least one mix is honest and keeps their private key and the permutation used secret, the correspondence between the messages and the senders remains hidden as the final output is a set of untraceable, randomly ordered messages.

5.3.10 Interactive proofs

Interactive proofs were introduced by Goldwasser, Micali and Rackoff [GMR85] as protocols for a prover P to convince a verifier V of the validity of a claimed assertion, which may be any **NP** predicate. The assertion is usually based on a secret s, and it is desirable to prove the assertion without revealing the secret. It is assumed that P has essentially infinite computational power (exponential time) and V has limited computational power (probabilistic polynomial time).

A proof is valid if it satisfies the following properties:

1. Completeness. If both P and V are honest, then V accepts the proof if the assertion is valid.

2. Soundness. If V is honest, then V rejects the proof if the assertion is invalid.

As opposed to mathematical proofs, where correctness is absolute, in interactive proofs these properties hold with a certain probability. Hence interactive proofs are often called proofs by protocol. The probability is determined by the number

of rounds in the proof, as specified by a security parameter N. Each round is a challenge-response protocol, where V asks a question, P answers the question and then V verifies the answer. It consists of two steps:

Challenge V randomly selects a number as the challenge c and sends this to P.

Response P creates a response r, which proves some relationship between c and the secret s that can only be known if s is known. P then sends r to V. If r is a valid response for c, then the round is successful.

If all the rounds are successful, then V accepts the proof. Otherwise, V rejects the proof.

It is possible for P to cheat, but the probability of this being undetected can be made negligible by choosing N to be arbitrarily large, so V can be convinced with overwhelming probability that the proof is valid. A consequence of this is that interactive proofs are generally inefficient, as many repetitions are required to achieve the desired level of confidence.

5.3.11 Zero knowledge proofs

A zero knowledge proof is an interactive proof that does not reveal any partial information about the secret. In other words, the verifier, whether honest or dishonest, gains no additional information by participating in the protocol, hence the term "zero knowledge". Usually, zero knowledge protocols are computationally secure rather than perfectly secure, because they rely on one-way functions. Only computationally secure zero knowledge proofs are considered in this report.

Zero knowledge proofs combine challenge-response protocols with cut-and-choose protocols. A cut-and-choose protocol is used to divide something fairly, so that one party cuts an object into equal parts and the other party selects one portion. This is used by the verifier V to randomly select a challenge from several possible alternatives. The prover P must be able to correctly answer all of these challenges. In the most common case, there are two possible challenges and a beacon may be used to randomly generate challenge bits.

The protocol follows the structure of interactive proofs and consists of a series of rounds. Each round typically has three steps:

Witness P randomly selects a commitment m and uses this to construct a witness w. P then sends w to V. A witness is commonly created by using a hash function, in which case w is a hash of m, or a commitment scheme (Section 5.3.3), in which case w is a commitment blob for m. Witnesses should not be reused, otherwise the secret may be revealed.

Challenge V selects a challenge c, which is assumed in this case to be a randomly generated bit, and sends this to P.

Response P creates a response r and sends it to V. If $c = 1$, then r is created such that it proves some relationship between w and the secret s that can only be known if s is known. This proves that s is known if w is a valid witness. If $c = 0$, then r is created such that it explicitly reveals the committed value m, which can be used by V to reconstruct w. This prevents cheating and gives high confidence that w is a valid witness in at least one of the rounds where $c = 1$. If r is a valid response for c, then the round is successful.

If all the rounds are successful, then V accepts the proof. Otherwise, V rejects the proof. In each round, the probability of P cheating without being detected is $1/2$. Thus over N rounds, the probability of P cheating without being detected is $1/2^N$, so the probability that the proof is valid is $1 - 1/2^N$.

5.3.12 The Fiat-Shamir heuristic

Fiat and Shamir [FS87] developed a technique for making interactive proofs non-interactive. In most interactive proofs, the challenges are bits randomly generated by the verifier or a beacon. The Fiat-Shamir heuristic replaces the verifier or beacon by employing some well-known hash function to generate the challenge bits.

For a security parameter N, the prover P constructs N witnesses. This set of witnesses is typically used as the input to the hash function, which outputs an unpredictable binary string. The first N bits of this output are used as the challenge bits, and P constructs the corresponding N responses. P publicly reveals the witnesses, the challenge bits and the responses. Remarkably, any party may verify this proof.

Cheating may occur if P can predict the output of the hash function by predicting the input. Hence the input must be carefully specified, so that this is infeasible. Also, cheating may occur if P repeatedly generates different inputs for the hash function until the output consists of challenges for which valid responses can be guessed correctly. In order to prevent this, the security parameter N must be much larger than in the interactive case, so that this is infeasible.

Chapter 6

Schemes Based on Homomorphic Encryption

Certain encryption functions E are $(+, \times)$-homomorphic, so that

$$E(a+b) = E(a)\,E(b).$$

This is most suitable for YES/NO voting options, and schemes represent a YES vote by $v = 1$ and a NO vote either by $v = 0$ or $v = -1$. The tally of the election is thus simply the sum of all votes. The advantage of using a $(+, \times)$-homomorphic encryption function for YES/NO voting is that individual votes are not decrypted. Encrypted votes are combined to produce an encrypted tally, and only this is decrypted. This has the potential to improve both privacy and efficiency. The schemes presented in this chapter are formulated on this principle.

Shamir's (t, n)-threshold secret sharing scheme is $(+, +)$-homomorphic. The shares of sums of values are equal to sums of the shares of values, that is

$$\text{share}(a+b) = \text{share}(a) + \text{share}(b).$$

This property is exploited by schemes to distribute shares of votes to the authorities. It enables direct computations on shares, as the authorities can each create a subtally, which is the sum of the vote shares. These subtallies are shares of the overall tally, which is the sum of all votes. Since the tally is the shared secret, any threshold of t or more subtallies may be combined to reconstruct it. This often used to achieve privacy and robustness.

All of the schemes in this chapter use a bulletin board for communication, which requires a digital signature scheme for authentication. Although it is not explicitly stated in any of the protocols, all messages are signed when they are posted. An electoral roll containing pseudonyms for verifying signed messages is assumed to be available. This implicitly provides most of the properties of correctness, since the eligibility of voters, uniqueness of votes and integrity of votes are ensured. Completeness is satisfied by the use of interactive proofs.

6.1 Cohen (Benaloh) and Fischer [CF85]

In 1985, Cohen and Fischer [CF85] proposed the first detailed protocol for public elections, and it is suitable for YES/NO voting. Their paper introduces several important concepts, which are extended by later election protocols and are also relevant to other areas of cryptography.

All communication in this scheme is public, with messages being posted on a bulletin board. A beacon is employed to randomly generate bits for interactive proofs, and these bits are posted on the bulletin board.

6.1.1 Cryptographic tools

6.1.1.1 Encryption

Cohen and Fischer develop a $(+, \times)$-homomorphic probabilistic public key cryptosystem based on higher degree residues. It is essentially a generalised form of the Goldwasser-Micali cryptosystem (Section 5.2.2), which is based on quadratic residues. The algorithm uses a weak version of the r-th residue problem, where r remains fixed but is larger than the number of voters I. Also, the messages to be encrypted are restricted.

A public and private key pair is created as follows:

1. Let N be a security parameter. Let r be a system parameter such that $r > I$.
2. Randomly generate two large and distinct primes p and q of length N such that $r \mid p-1$, $r^2 \nmid p-1$ and $r \nmid q-1$.
3. Compute the modulus $n = pq$.
4. Randomly select $y \in \mathbb{Z}_n^*$ such that y is not an r-th residue modulo n.

The public key is (n, y) and the private key is (p, q). Let E denote probabilistic encryption using (n, y).

A vote v is encrypted as the ciphertext

$$E(v) = y^v x^r \bmod n$$

where $x \in \mathbb{Z}_n^*$ is randomly chosen.

The scheme represents a YES vote by $v = 1$ and a NO vote by $v = 0$. Hence only two possible values can be encrypted:

1. If $v = 0$, then $E(0) = x^r \bmod n$, so $E(0)$ is an r-th residue modulo n.
2. If $v = 1$, then $E(1) = yx^r \bmod n$, so $E(1)$ is an r-th nonresidue modulo n.

6.1.1.2 Decryption

The tally τ of the election is simply the sum of all the votes. Since E is $(+, \times)$-homomorphic, then τ can be calculated by forming the product of all the encrypted votes and then decrypting this product. Note that since r is greater than the number of voters and each vote is either 0 or 1, then $0 \leq \tau < r$. Thus $E(\tau)$ can be decrypted by an exhaustive search of all possible values modulo r. This is feasible if r is not too large.

Let $E(\tau) = y^{\tau} x^{r} \bmod n$ for some positive integer x and let $\phi(n) = (p-1)(q-1)$. Then by Euler's Theorem (Theorem 4.2.2.7), the correct value of τ is found when

$$(E(\tau)/y^{\tau})^{\phi(n)/r} \equiv 1 \pmod{n}.$$

6.1.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and a single authority A.

6.1.2.1 Stage 1: Initialising

The authority A does the following:

1. Post a prime $r > I$ and a security parameter N.
2. Create a public key (n, y) and the private key (p, q), where p and q are of length $N/2$.
3. Post (n, y) and prove that it is valid (Section 6.1.3.1).

6.1.2.2 Stage 2: Ballot Preparation

1. Each voter V_i posts a master ballot b_i. A valid ballot is a randomly ordered pair of encrypted 0 and 1 votes constructed as follows:
   (a) Randomly select $f_i, g_i \in \mathbb{Z}_n^*$.
   (b) Compute

$$\begin{aligned} E(0) &= f_i^r \bmod n, \\ E(1) &= y g_i^r \bmod n. \end{aligned}$$

   (c) Randomly set $b_i = (E(0), E(1))$ or $b_i = (E(1), E(0))$.
2. V_i posts a proof that b_i is valid (see Section 6.1.3.2).

6.1.2.3 Stage 3: Voting

1. Each voter V_i casts a vote by selecting the element of their master ballot b_i corresponding to the desired vote v_i.
2. V_i posts the encrypted vote $E(v_i) \in b_i$.

6.1.2.4 Stage 4: Counting

1. The authority A verifies the following for each voter V_i:
   (a) The master ballot is of the form $b_i = (u_i, u_i')$ such that $u_i, u_i' \in \mathbb{Z}_n^*$.
   (b) The proof for the validity of b_i is correct.
   (c) The encrypted vote $E(v_i) \in b_i$.
2. A calculates the tally of the election from the valid votes. Assuming all votes are valid, A computes the product

$$\pi \equiv \prod_{i=1}^{I} E(v_i) \pmod{n}.$$

   Due to the $(+, \times)$-homomorphic property of E, this is the encrypted form of the tally τ, so

$$E(\tau) \equiv \pi \pmod{n}.$$

   This is decrypted by finding τ, $0 \leq \tau < r$, and x such that

$$E(\tau) \equiv y^{\tau} x^{r} \pmod{n}$$

   as described in Section 6.1.1.2.
3. The tally τ is published and the proof in Section 6.1.3.3 is conducted to show it is valid without revealing x.

6.1.3 Interactive proofs

Cohen and Fischer propose a method for conducting interactive proofs in a zero knowledge manner, as discussed in more detail in [Ben87a]. They introduce the notion of a cryptographic capsule, which is defined as a randomly ordered collection of objects. It can be used in conjunction with a beacon to allow a party to conduct a proof for passive verifiers.

6.1.3.1 Proving the public key is valid

The public key (n, y) and the private key (p, q) is a valid key pair if $y \in \mathbb{Z}_n^*$ such that y is not an r-th residue modulo n, and $n = pq$. Also, it is required that $r \mid p - 1$, $r^2 \nmid p - 1$ and $r \nmid q - 1$ for a fixed r. If the key pair is not of this form, it may be possible for the authority to decrypt an encrypted message as any desired message. The validity of key pair is to be proven by the authority without revealing (p, q). The proof may be verified by any party.

1. Let the security parameter be N.
2. The authority constructs a list of N randomly chosen candidate key pairs and posts all the public keys.

3. The beacon is used to randomly generate a challenge c, $1 \leq c \leq N$.

4. The actual key pair to be used is the c-th pair.

5. To show that all the other key pairs are valid, all the corresponding private keys are posted. The verifier checks that the revealed key pairs are of the required form.

The probability that this proof is valid is $1 - 1/N$.

6.1.3.2 Proving the ballot is valid

A ballot b is valid if it is a randomly ordered pair of encrypted 0 and 1 votes, so it is of the form

$$b = \{f^r \bmod n, yg^r \bmod n\}$$

as a set (ordering is unimportant). This is to be proven by the voter without revealing the ordering of the ballot. The proof may be verified by any party.

The premise of this proof is that since E is $(+, \times)$-homomorphic, then $E(1)/E(1) = E(0)$ and $E(0)/E(0) = E(0)$ (note that E here is probabilistic). Thus if two ballots are each of the form $\{E(0), E(1)\}$, then it is possible to combine them to produce the zero ballot $\{E(0), E(0)\}$.

1. Let the security parameter be N.

2. In addition to the master ballot $b = \{f^r \bmod n, yg^r \bmod n\}$, the voter posts N randomly created auxiliary ballots $w_1, w_2, \ldots, w_N$ as witnesses. Each witness w_k, $1 \leq k \leq N$, must be a valid ballot, so it is of the form $w_k = \{f_k^r \bmod n, yg_k^r \bmod n\}$, where f_k and g_k can be considered as the committed values.

3. The beacon is used to randomly generate N challenge bits $c_1, c, \ldots, c_N$.

4. For each bit $c_k = 1$, the voter posts the committed values f_k and g_k. The verifier constructs

$$\begin{aligned} E(0) &= f_k^r \bmod n, \\ E(1) &= yg_k^r \bmod n. \end{aligned}$$

   Let the auxiliary ballot $w_k = (a_k, a'_k)$. If

$$\{E(0), E(1)\} = \{a_k, a'_k\}$$

   as sets (ordering is unimportant), then w_k is a valid witness and gives high confidence that at least one of the remaining witnesses is also valid.

5. For each bit $c_k = 0$, the voter posts $f_k/f \bmod n$ and $g_k/g \bmod n$. The verifier constructs

$$\begin{aligned} u &= (f_k/f)^r \bmod n, \\ u' &= (g_k/g)^r \bmod n. \end{aligned}$$

Since u and u' are both r-th residues modulo n, they are both equivalent to $E(0)$. Let the master ballot $b = (\alpha, \alpha')$ and the auxiliary ballot $w_k = (a_k, a'_k)$. If either

$$\{u, u'\} = \{a_k/\alpha, a'_k/\alpha'\}$$

or

$$\{u, u'\} = \{a_k/\alpha', a'_k/\alpha\}$$

as sets, then b and w_k are both of the same type as they are encryptions of the same votes, although the ordering may differ. Thus if w_k is a valid witness of the form $\{E(0), E(1)\}$, then b is also valid.

The probability that this proof is valid is $1 - 1/2^N$.

6.1.3.3 Proving the tally is valid

The tally τ is a valid decryption of $E(\tau)$ if $E(\tau)/y^\tau$ is an r-th residue modulo n. The authority proves that this is the case by showing that

$$E(\tau)/y^\tau \equiv x^r \pmod{n}$$

for some positive integer x, without revealing x.

1. Let the security parameter be N.
2. The authority randomly selects N commitments $m_1, m_2, \ldots, m_N$ such that $\gcd(m_k, n) = 1$ for all m_k, $1 \leq k \leq N$. The witnesses $w_k \equiv m_k^r \pmod{n}$, $1 \leq k \leq N$, are posted.
3. The beacon is used to randomly generate N challenge bits $c_1, c, \ldots, c_N$.
4. For each bit $c_k = 1$, the authority posts the commitment m_k. The verifier checks that

   $$w_k \equiv m_k^r \pmod{n}.$$

   This shows that w_k is an r-th residue modulo n and gives high confidence that at least one of the remaining w_k is also an r-th residue modulo n.
5. For each bit $c_k = 0$, the authority posts $m_k x$. The verifier checks that

   $$(m_k x)^r \equiv w_k E(\tau)/y^\tau \pmod{n}.$$

   This shows that $w_k E(\tau)/y^\tau$ is an r-th residue modulo n. Thus if w_k is an r-th residue modulo n, then $E(\tau)/y^\tau$ is also an r-th residue modulo n.

The probability that this proof is valid is $1 - 1/2^N$.

6.1.4 Evaluation

6.1.4.1 Security

1. Privacy with respect to voters is satisfied by encrypting votes. However, privacy can be violated if the authority is dishonest and reveals decrypted votes. Cohen and Fischer acknowledge this as a problem.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to encrypt their vote.

3. Robustness with respect to voters is satisfied by the use of proofs. However, if the authority is dishonest, then the election can be disrupted.

4. Public verifiability is satisfied since any party, including passive observers, may verify the proofs.

5. Correctness is satisfied by the use of the bulletin board for all communication and by providing proofs for all actions.

6. Fairness is not satisfied. Although counting occurs after all votes have been cast, the authority may still leak intermediate tallies during the voting stage.

7. Vote independence is satisfied by encrypting ballots using a probabilistic algorithm. In order to prove that a ballot is valid, the ballot must be known.

6.1.4.2 Usability

1. The computational complexity is $O\left(N^2\right)$ for each voter and $O\left(N^2 I\right)$ for the authority. These are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ modular multiplications are done by every voter, which is $O\left(N^2\right)$. Verifying this proof takes $O\left(N^2\right)$ time, and the authority must do this for I voters, which is $O\left(N^2 I\right)$.

2. The communication complexity is $O\left(N^2\right)$ for each voter and $O\left(N^2 I\right)$ for the authority. Again, these are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ bits are transmitted by every voter, which is $O\left(N^2\right)$. Verifying this proof takes $O\left(N^2\right)$ bits, and the authority must do this for I voters, which is $O\left(N^2 I\right)$.

3. The round complexity is 2. Every voter must participate in the Ballot Preparation and Voting stages. Each stage must be completed before the next stage commences, so stages are sessions with specified deadlines. However, the stages are not required to occur in immediate succession, and ballot preparation may take place well in advance of voting.

6.1.4.3 Extensions

The authors propose an extension for multiway voting, but the cost is impractical because decrypting the tallies is exponential in the number of options L. The decryption of the tally in the standard scheme does not increase the computational complexity. However, for L options, computing the tallies would become the dominant term in the complexity for the authority. The other complexities would be increased by a factor of L.

6.2 Benaloh [Ben87b]

Benaloh [Ben87b] extends the last scheme by distributing the functionality of the single authority using secret sharing, thus improving privacy and robustness with respect to authorities. Voters distribute shares of their votes to multiple authorities. It is based on a very similar scheme by Benaloh and Yung [BY86], which employs the simple secret sharing scheme described in Section 5.3.6. In this scheme, Shamir's (t, n)-threshold secret sharing scheme (Section 5.3.7) is used instead, to provide robustness.

All communication is public, with messages being posted on a bulletin board.

6.2.1 Cryptographic tools

6.2.1.1 Encryption

The scheme uses the same encryption algorithm based on r-th residues as in the previous scheme. Again, r remains fixed but is larger than the number of voters I. In this case, each authority A_j creates a public key (n_j, y_j) and private key (p_j, q_j) of the form described in Section 6.1.1.1.

6.2.1.2 Verifiable secret sharing

Shamir's scheme is used to split a vote v into J shares $v_j \in \mathbb{Z}_r$, $1 \leq j \leq J$. Each authority receives a share, and a prescribed threshold of any t out of the J authorities may reconstruct the vote.

To encrypt v, each share v_j is encrypted with E_j, resulting in a vector

$$E(v) \Leftrightarrow (E_1(v_1), E_2(v_2), \ldots, E_J(v_J)).$$

Benaloh suggests that the shares can be verified by proving that the polynomial P used for creating the shares is of degree t or less. This is done by randomly creating a set of polynomials of degree t and either revealing the polynomial to show that it is of degree t or showing that the sum of the polynomial and P is of degree t. The full details of this interactive proof can be found in [Ben87b]. It is reasonably inefficient, and a non-interactive verifiable secret sharing scheme, such as that of Pedersen (see Section 5.3.8), would be preferable.

The scheme also represents a YES vote by $v = 1$ and a NO vote by $v = 0$. Using the $(+, \times)$-homomorphic property of E_j, each authority computes a subtally of vote shares in a similar fashion to the original single authority protocol. From the $(+, +)$-homomorphic property of the secret sharing scheme, these subtallies are shares of the overall tally. The tally is revealed by combining the subtallies of any t authorities.

6.2.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$.

6.2.2.1 Stage 1: Initialising

1. The authorities post a prime $r > I$ and a security parameter N.
2. Each authority A_j creates a public key (n_j, y_j) and private key (p_j, q_j), where n_j is of length N.
3. A_j posts (n_j, y_j) and proves that it is valid (Section 6.2.3.1).

6.2.2.2 Stage 3: Ballot Preparation

1. Each voter V_i posts a master ballot b_i. A valid ballot is a randomly ordered pair of encrypted 0 and 1 votes constructed as follows:
   (a) Split the 0 vote into J shares $s_{i,j} \in \mathbb{Z}_r$, $1 \leq j \leq J$.
   (b) Split the 1 vote into J shares $s'_{i,j} \in \mathbb{Z}_r$, $1 \leq j \leq J$.
   (c) Compute

$$\begin{aligned} E(0) &\Leftrightarrow \left(E_1(s_{i,1}), E_2(s_{i,2}), \ldots, E_J(s_{i,J})\right), \\ E(1) &\Leftrightarrow \left(E_1\left(s'_{i,1}\right), E_2\left(s'_{i,2}\right), \ldots, E_J\left(s'_{i,J}\right)\right). \end{aligned}$$

   (d) Randomly set $b_i = (E(0), E(1))$ or $b_i = (E(1), E(0))$.
2. V_i posts a proof that b_i is valid (see Section 6.2.3.2).

6.2.2.3 Stage 4: Voting

1. Each voter V_i casts a vote by selecting the element of their master ballot b_i corresponding to the desired vote v_i.
2. V_i posts encrypted vote $E(v_i) \in b_i$. Note that this is a vector of encrypted shares, $E(v_i) \Leftrightarrow (E_1(v_{i,1}), E_2(v_{i,2}), \ldots, E_J(v_{i,J}))$.

6.2.2.4 Stage 5: Counting

1. Each authority A_j verifies the following for each voter V_i:

   (a) The shares for b_i are valid.

   (b) The proof for the validity of b_i is correct.

   (c) The encrypted vote $E(v_i) \in b_i$.

2. A_j calculates the subtally of its shares of the valid votes. Assuming all votes are valid, A_j computes the product

$$\pi_j \equiv \prod_{i=1}^{I} E_j(v_{i,j}) \pmod{n_j}.$$

   Due to the $(+, \times)$-homomorphic property of E_j, this is the encrypted form of the subtally τ_j, so

$$E_j(\tau_j) \equiv \pi_j \pmod{n_j}.$$

   This is decrypted by finding τ_j, $0 \leq \tau_j < r$, and x_j such that

$$\pi_j \equiv y_j^{\tau_j} x_j^r \pmod{n_j}$$

   as described in Section 6.1.1.2 for the previous scheme.

3. The subtally τ_j is posted and the proof from the previous scheme (see Section 6.1.3.3) is conducted to show it is valid without revealing x_j.

4. Due to the $(+, +)$-homomorphic property of the secret sharing scheme, the tally τ of the election is the shared secret and is reconstructed as per Shamir's scheme. Let S be the set of any t subtallies used to reconstruct τ. Simply compute

$$\tau = \sum_{j \in S} P_j(0)\, \tau_j$$

   where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k-j}.$$

   This can be done by any party.

6.2.3 Interactive proofs

The same method of zero knowledge proofs using cryptographic capsules is employed. A beacon may be used, although Benaloh describes how it can be replaced by the distributed authorities.

6.2.3.1 Proving the public key is valid

The public key (n, y) and the private key (p, q) is a valid key pair if $y \in \mathbb{Z}_n^*$ such that y is not an r-th residue modulo n, and $n = pq$. Also, it is required that $r \mid p-1$, $r^2 \nmid p-1$ and $r \nmid q-1$ for a fixed r. If the key pair is not of this form, it may be possible for the authority to decrypt an encrypted message as any desired message. The validity of key pair is to be proven by the authority without revealing (p, q). The interactive proof from the previous scheme (see Section 6.1.3.1) may be used, but Benaloh suggests a method with a higher confidence for a given security parameter N. The probability that this proof is valid is $1 - 1/2^N$ rather than $1 - 1/N$.

The proof requires a set of verifiers, which may consist of any voters or authorities. To reduce the amount of work required by the voters, it is assumed that inspectors are the set of J authorities. Each authority conducts a separate instance of the proof for every other authority, so $J - 1$ proofs are conducted by each authority.

The essence of the proof relies on the fact that if an invalid key pair is formed so that a message can be decrypted in multiple ways, then if the verifier provides encrypted messages it is difficult for the prover to correctly guess the original messages. The details are too complicated to be presented here, but they can be found in [Ben87b].

6.2.3.2 Proving the ballot is valid

A ballot b is valid if it is a randomly ordered pair of encrypted 0 and 1 votes, so it is of the form $b = \{E(0), E(1)\}$. This is to be proven by the voter without revealing the ordering of the ballot.

The proof is similar to that from the previous scheme (see Section 6.1.3.2). The same principle is used to show that an auxiliary ballot and the master ballot are encryptions of the same votes. However, it becomes much more elaborate since $E(0)$ and $E(1)$ are each vectors of encrypted shares. The details are too complicated to be presented here, but they can be found in [Ben87b].

6.2.4 Evaluation

6.2.4.1 Security

1. Privacy with respect to voters is satisfied by encrypting shares of votes. Privacy with respect to authorities is only violated if t or more authorities collude to reveal decrypted votes.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to encrypt their vote.

3. Robustness with respect to voters is satisfied by providing proofs for all actions and using verifiable secret sharing. Robustness with respect to authorities is only violated if t or more authorities collude to disrupt the election.

4. Public verifiability is satisfied since any party, including passive observers, may verify the proofs. Also, any party may compute the tally from any t subtallies.

5. Correctness is satisfied by the use of the bulletin board for all communication and by providing proofs for all actions.

6. Fairness is satisfied because during the voting stage, each authority can only reveal subtallies of vote shares. Since counting occurs after all votes have been cast and requires a threshold of authorities to collaborate to decrypt the votes, intermediate tallies cannot be leaked unless t or more authorities are dishonest.

7. Vote independence is satisfied by encrypting ballots using a probabilistic algorithm. In order to prove that a ballot is valid, the ballot must be known.

6.2.4.2 Usability

1. The computational complexity is $O\left(N^2J\right)$ for each voter and $O\left(N^2IJ\right)$ for each authority. These are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ modular multiplications are done by every voter for J ballot shares, which is $O\left(N^2J\right)$. Verifying this entire proof takes $O\left(N^2J\right)$ time, and every authority must do this for I voters, which is $O\left(N^2IJ\right)$.

2. The communication complexity is $O\left(N^2J\right)$ for each voter and $O\left(N^2IJ\right)$ for each authority. Again, these are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ bits are transmitted by every voter for J ballot shares, which is $O\left(N^2J\right)$. Verifying this entire proof takes $O\left(N^2J\right)$ bits, and every authority must do this for I voters, which is $O\left(N^2IJ\right)$.

3. The round complexity is 2. Every voter must participate in the Ballot Preparation and Voting stages. Each stage must be completed before the next stage commences, so stages are sessions with specified deadlines. However, the stages are not required to occur in immediate succession, and ballot preparation may take place well in advance of voting.

6.2.4.3 Extensions

Benaloh proposes an extension for multiway voting, but the cost is impractical because decrypting the tallies is exponential in the number of options L. The decryption of the tally in the standard scheme does not increase the computational complexity. However, for L options, computing the tallies would become the dominant term in the complexity for the authorities. The other complexities would be increased by a factor of L.

6.3 Benaloh and Tuinstra [BT94]

Benaloh and Tuinstra present the first election scheme incorporating incoercibility, a property which was first suggested as an open problem by Iversen [Ive92]. It is based on the earlier multiple authority scheme by Benaloh, and achieves incoercibility by providing masking factors to mask vote shares. These masking factors are generated by the authorities and privately revealed to voters. This additional private information prevents a receipt for the vote from being constructed, because a voter may lie about their vote by creating false transcripts consistent with all public information.

The scheme assumes the existence of a voting booth for masking factors to be distributed to voters. Whilst in the voting booth, the voter is isolated from all external processes to prevent monitoring of any actions and information received. The voter may not write to any channels but may read from any public or private channels. This voting booth is unconditionally secure, and allows unconditionally private communication with an authority.

All other communication in this scheme is public, with messages being posted on a bulletin board. A beacon is employed to randomly generate bits for interactive proofs, and these bits are posted on the bulletin board.

6.3.1 Cryptographic tools

Encryption

The scheme uses the same encryption algorithm based on r-th residues as in the previous schemes. Again, r remains fixed but is larger than the number of voters I. In this case, each authority A_j creates a public key (n_j, y_j) and private key (p_j, q_j) of the form described in Section 6.1.1.1. Here, encryption is essentially used as a commitment scheme by the authorities to commit to masking factors for each voter.

Secret sharing

Using Shamir's scheme, a vote v is split into J shares $v_j \in \mathbb{Z}_r$, $1 \leq j \leq J$. Each share is associated with a masking factor $m_j \in \mathbb{Z}_r$, so the masked share

$$(v_j + m_j) \bmod r$$

is distributed to the authority A_j.

6.3.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$.

6.3.2.1 Stage 1: Initialising

1. The authorities post a prime $r > I$ and a security parameter N.

2. Each authority A_j creates a public key (n_j, y_j) and private key (p_j, q_j), where n_j is of length N.

3. A_j posts (n_j, y_j) and proves that it is valid. This may be done as in Section 6.1.3.1 or Section 6.2.3.1.

6.3.2.2 Stage 2: Ballot Preparation

1. Each authority A_j randomly generates a masking factor $m_{i,j} \in \mathbb{Z}_r$ for each voter V_i.

2. A_j posts all the encrypted masking factors $E_j(m_{i,j})$, $1 \leq i \leq I$.

3. A voter V_i enters the voting booth.

4. Each A_j privately reveals $m_{i,j}$ to V_i.

5. Each A_j privately proves to V_i that $E_j(m_{i,j})$ is valid without revealing a certificate that can be used to prove it is the encryption of $m_{i,j}$ (See Section 6.3.3.1).

6. V_i exits the voting booth.

7. Each voter V_i creates a master ballot b_i. A valid ballot is a randomly ordered pair of 0 and 1 votes, so $b_i = (0, 1)$ or $b_i = (1, 0)$.

8. V_i creates shares for each half of b_i as follows:

   (a) Let $b_i = (s_i, s'_i)$

   (b) Split s_i into J shares $s_{i,j} \in \mathbb{Z}_r$, $1 \leq j \leq J$.

   (c) Split s'_i into J shares $s'_{i,j} \in \mathbb{Z}_r$, $1 \leq j \leq J$.

9. V_i posts the masked shares $(s_{i,j} + m_{i,j}) \bmod r$ and $\left(s'_{i,j} + m_{i,j}\right) \bmod r$, $1 \leq j \leq J$. Note that b_i is not revealed.

10. V_i posts a proof that the masked shares are valid (see Section 6.3.3.2).

6.3.2.3 Stage 3: Voting

1. Each voter V_i casts a vote by selecting the element of their master ballot b_i corresponding to the desired vote v_i.

2. V_i posts the masked shares for v_i, that is $(v_{i,j} + m_{i,j}) \bmod r$, $1 \leq j \leq J$.

6.3.2.4 Stage 4: Counting

1. Each authority A_j verifies the following for each voter V_i:

   (a) The proof for the validity of the masked shares $(s_{i,j} + m_{i,j}) \bmod r$ and $\left(s'_{i,j} + m_{i,j}\right) \bmod r$ are valid.

   (b) Each masked vote share $(v_{i,j} + m_{i,j}) \bmod r$, $1 \leq j \leq J$, corresponds to either $(s_{i,j} + m_{i,j}) \bmod r$ or $\left(s'_{i,j} + m_{i,j}\right) \bmod r$.

2. A_j calculates the subtally of its shares of the valid votes. Assuming all votes are valid, A_j computes the sum

$$\begin{aligned} \sigma_j &\equiv \left(\sum_{i=1}^{I} (v_{i,j} + m_{i,j})\right) \pmod{r} \\ &\equiv \left(\sum_{i=1}^{I} v_{i,j} + \sum_{i=1}^{I} m_{i,j}\right) \pmod{r}. \end{aligned}$$

3. A_j uses the corresponding encrypted masking factors to form the product

$$\pi_j \equiv \prod_{i=1}^{I} E_j\left(m_{i,j}\right) \pmod{n_j}.$$

   Due to the $(+, \times)$-homomorphic property of E_j, this is the encrypted form of the sum of the masking factors

$$m_j \equiv \left(\sum_{i=1}^{I} m_{i,j}\right) \pmod{r}.$$

   Since A_j knows all the masking factors $m_{i,j}$, this can easily be computed.

4. A_j posts m_j and proves that it is valid by revealing a certificate x_j such that

$$E_j\left(m_j\right) \equiv y_j^{m_j} x_j^r \pmod{n_j}.$$

5. Any party may then remove the masking factors to form the subtally

$$\begin{aligned} \tau_j &\equiv \left(\sigma_j - \sum_{i=1}^{I} m_{i,j}\right) \pmod{r} \\ &\equiv \left(\sum_{i=1}^{I} v_{i,j}\right) \pmod{r}. \end{aligned}$$

6. Due to the $(+, +)$-homomorphic property of the secret sharing scheme, the tally τ of the election is the shared secret and is reconstructed as per

Shamir's scheme. Let S be the set of any t subtallies used to reconstruct τ. Simply compute

$$\tau = \sum_{j \in S} P_j(0)\, \tau_j$$

where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k-j}.$$

This can be done by any party.

6.3.3 Interactive proofs

6.3.3.1 Proving the masking factor is valid

A masking factor m_j is valid if the posted value is

$$E_j(m_j) \equiv y_j^{m_j} x^r \pmod{n_j}$$

for some positive integer x. This is to be proven by the authority without revealing x, which can be used as a certificate by the voter to prove that $E_j(m_j)$ is the encryption of m_j. The proof may only be verified by the voter and is conducted privately within the voting booth.

The premise of this proof is that since E_j is $(+, \times)$-homomorphic, then $E_j(a)/E_j(b) = E_j(a-b)$.

1. Let the security parameter be N.
2. Each authority A_j randomly selects N auxiliary masking factors $m_{j,1}, m_{j,2}, \ldots, m_{j,N}$ and encrypts them as $E_j(m_{j,k})$, $1 \leq k \leq N$. Then $E_j(m_{j,k})$, $1 \leq k \leq N$, are posted and $m_{j,k}$, $1 \leq k \leq N$, are revealed privately to the voter.
3. The beacon is used to randomly generate N challenge bits $c_1, c, \ldots, c_N$.
4. For each bit $c_k = 1$, A_j reveals $m_{j,k}$ and some x_k such that

   $$E_j(m_{j,k}) \equiv y_j^{m_{j,k}} x_k^r \pmod{n_j}.$$

   This shows that $m_{j,k}$ is a valid masking factor and gives high confidence that at least one of the remaining $m_{j,k}$ is valid.
5. For each bit $c_k = 0$, A_j reveals $m_{j,k} - m_j$ and some x_k such that

   $$E_j(m_{j,k} - m_j) \equiv y_j^{m_{j,k} - m_j} x_k^r \pmod{n_j}.$$

   This shows that $m_{j,k} - m_j$ is a valid masking factor. Since $E_j(m_{j,k} - m_j) = E_j(m_{j,k})/E_j(m_j)$, if $m_{j,k}$ is a valid masking factor, then m_j is also a valid masking factor.

The probability that this proof is valid is $1 - 1/2^N$.

6.3.3.2 Proving the ballot is valid

A ballot b is valid if it is a randomly ordered pair of 0 and 1 votes, so it is of the form $b = \{0, 1\}$. The corresponding masked shares are $(s_j + m_j) \bmod r$ and $\left(s'_j + m_j\right) \bmod r$, $1 \leq j \leq J$, and the ballot is valid if the shared secret of one is 0 and the other is 1. This is to be proven by the voter without revealing the shares, the masking factors or the order of the ballot. The proof may be verified by any party.

The premise of this proof relies on the secret sharing scheme being $(+, +)$-homomorphic. If two ballots are each of the form $\{0, 1\}$, then subtracting the corresponding masked shares of the 1 vote of one ballot from the other results in masked shares of a 0 vote. The same applies for shares of the 0 vote. For each of these masked shares of a 0 vote, the combined masking factor can be revealed so that the 0 vote can be reconstructed to verify that the two ballots are valid.

1. Let the security parameter be N.

2. Each authority A_j in fact randomly generates an additional N auxiliary masking factors $m_{j,k} \in \mathbb{Z}_r$, $1 \leq k \leq N$, for each voter. A_j posts the encrypted auxiliary masking factors $E_j\left(m_{j,k}\right)$, $1 \leq k \leq N$.

3. While the voter is in the voting both, each A_j privately reveals $m_{j,k}$, $1 \leq k \leq N$.

4. In addition to the master ballot b, the voter randomly creates N auxiliary ballots $b_1, b_2, \ldots, b_N$ as commitment values. Each ballot b_k, $1 \leq k \leq N$, must be valid, so it is of the form $b_k = \{0, 1\}$. The corresponding masked shares $w_{j,k} = (s_{j,k} + m_{j,k}) \bmod r$ and $w'_{j,k} = \left(s'_{j,k} + m_{j,k}\right) \bmod r$, $1 \leq j \leq J$, $1 \leq k \leq N$, are posted as witnesses.

5. The beacon is used to randomly generate N challenge bits $c_1, c, \ldots, c_N$.

6. For each bit $c_k = 1$, each A_j posts $m_{j,k}$ and a certificate $x_{j,k}$ such that

$$E_j\left(m_{j,k}\right) \equiv y_j^{m_{j,k}} x_{j,k}^r \pmod{n_j}.$$

   The verifier can then remove the masking factors from the witnesses $w_{j,k}$ and $w'_{j,k}$, $1 \leq j \leq J$, to obtain the corresponding shares $s_{j,k}$ and $s'_{j,k}$. These shares are used to reconstruct the shared secrets as in Shamir's scheme, to check that $b_k = \{0, 1\}$. Thus if $w_{j,k}$ and $w'_{j,k}$, $1 \leq j \leq J$, is a valid set of witnesses, this gives high confidence that at least one of the remaining sets of witnesses is also valid.

7. For each bit $c_k = 0$, the voter designates which half of the master ballot b corresponds to which half of the auxiliary ballot b_k. Each A_j posts the combined masking factor $(m_{j,k} - m_j) \bmod r$ along with a proof this is valid without revealing $m_{j,k}$ or m_j. Since $E_j\left(m_{j,k}\right)$ and $E_j\left(m_j\right)$ have

been previously posted and E_j is $(+, \times)$-homomorphic, this is done simply by revealing a certificate x such that

$$E_j\left(m_{j,k}\right) / E_j\left(m_j\right) \equiv y_j^{m_{j,k}-m_j} x^r \pmod{n_j}.$$

Let the master ballot $b = \{s, s'\}$ and the auxiliary ballot $b_k = \{s_k, s'_k\}$. Say that the voter designated $s = s_k$ and $s' = s'_k$. Then both $s - s_k$ and $s' - s'_k$ should be equal to a 0 vote. For s and s_k, the verifier can check that the corresponding shares are correct by computing

$$\begin{aligned} & \left(s_j + m_j\right) \bmod r - \left(s_{j,k} + m_{j,k}\right) \bmod r + \left(m_{j,k} - m_j\right) \bmod r \\ \equiv \quad & \left(s_j - s_{j,k}\right) \pmod{r}. \end{aligned}$$

Then $\left(s_j - s_{j,k}\right) \pmod{r}$, $1 \leq j \leq J$, are shares of a secret, which can be reconstructed as in Shamir's scheme. If this secret is a 0 vote, then s and s_k, are votes of the same value. The same is done for s' and s'_k. Thus if $w_{j,k} = \left(s_{j,k} + m_{j,k}\right) \bmod r$ and $w'_{j,k} = \left(s'_{j,k} + m_{j,k}\right) \bmod r$, $1 \leq j \leq J$ are a valid set of witnesses, then $\left(s_j + m_j\right) \bmod r$ and $\left(s'_j + m_j\right) \bmod r$, $1 \leq j \leq J$, are a valid set of masked shares. In other words, if $b_k = \{0, 1\}$ then $b = \{0, 1\}$.

The probability that this proof is valid is $1 - 1/2^N$.

6.3.4 Evaluation

6.3.4.1 Security

1. Privacy with respect to voters is satisfied by masking shares of ballots. Privacy with respect to authorities is only violated if all authorities collude to reveal the masking factors.

2. Incoercibility is not satisfied. Despite the efforts of Benaloh and Tuinstra, the protocol is flawed. Hirt and Sako [HS00] demonstrate that it is possible for the voter to construct a receipt. During the proof of validity of the ballot, the randomness is determined by the voter, who may in fact commit to the ordering of the master ballot and the auxiliary ballots. A coercer may then verify that the proof is consistent with this commitment, which acts as a receipt.

3. Robustness with respect to voters is satisfied by providing proofs for all actions and using secret sharing. Robustness with respect to authorities is only violated if t or more authorities collude to disrupt the election.

4. Public verifiability is satisfied since any party, including passive observers, may verify the proofs. Also, any party may unmask the subtallies and then compute the tally.

5. Correctness is satisfied by the use of the bulletin board for all communication and by providing proofs for all actions.

6. Fairness is satisfied because during the voting stage, each authority can only reveal subtallies of vote shares. Since counting occurs after all votes have been cast and requires a threshold of authorities to collaborate to decrypt the votes, intermediate tallies cannot be leaked unless all authorities are dishonest.

7. Vote independence is satisfied by using unique masking factors for each voter. In order to prove that a ballot is valid, the masking factors must be known.

6.3.4.2 Usability

1. The computational complexity is $O\left(N^2J\right)$ for each voter and $O\left(N^2IJ\right)$ for each authority. These are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ modular multiplications are done by J authorities, which is $O\left(N^2J\right)$. Every voter must verify these proofs, which takes $O\left(N^2J\right)$ time. Every authority must conduct a separate proof for I voters, which takes $O\left(N^2IJ\right)$ time.

2. The communication complexity is $O\left(N^2J\right)$ for each voter and $O\left(N^2IJ\right)$ for each authority. Again, these are dominated by the proof of validity for the ballot. There are N rounds, and in each round $O\left(N\right)$ bits are transmitted by every voter for J ballot shares, which is $O\left(N^2J\right)$. This assumes that $\log_2 r$ is insignificant. Every authority must also transmit $O\left(N^2J\right)$ bits, and this is done for I voters, which is $O\left(N^2IJ\right)$

3. The round complexity is 2. Every voter must participate in the Ballot Preparation and Voting stages. Each stage must be completed before the next stage commences, so stages are sessions with specified deadlines. However, the stages are not required to occur in immediate succession, and ballot preparation may take place well in advance of voting.

6.4 Cramer, Franklin, Schoenmakers and Yung [CFSY96]

Cramer, Franklin, Schoenmakers and Yung [CFSY96] propose a scheme with considerable improvement in performance by using a more general family of homomorphic encryption functions based on discrete logarithms rather than r-th residues. To prove the validity of a ballot, a very simple witness indistinguishable proof is used instead of zero knowledge proofs. Also, the Fiat-Shamir heuristic (see Section 5.3.12) is employed to make this proof non-interactive.

Private communication channels are required for voters to send shares of their votes to the authorities. All other communication is public, with messages being posted on a bulletin board. The private channels may be implemented in this scheme by using public key cryptography and posting encrypted values on the bulletin board.

6.4.1 Cryptographic tools

Verifiable secret sharing

The essence of the election protocol is Pedersen's non-interactive verifiable secret sharing scheme, described in Section 5.3.8. It augments Shamir's (t, n)-threshold secret sharing scheme with a $(+, \times)$-homomorphic commitment scheme. Shares of the votes are distributed to the authorities, who can verify that the shares received are valid.

The scheme represents a YES vote by $v = 1$ and a NO vote by $v = -1$. The tally of the election is thus simply the sum of all the votes. Each authority computes a subtally of vote shares. From the $(+, +)$-homomorphic property of the secret sharing scheme, these subtallies are shares of the overall tally. The tally is revealed by combining the subtallies of any t authorities.

Voters commit to vote shares using the commitment scheme. Since this is $(+, \times)$-homomorphic, each authority forms a product of the commitment blobs, and this is the commitment blob for the subtally. This can then be used to verify that the subtallies of the authorities are valid. Although the authorities refer to the commitment scheme as an encryption function, decryption is not required at any point of the protocol. In fact, decryption is infeasible because "encryption" is a one-way function in this case.

6.4.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$. Each voter V_i has a unique binary string ID_i for identification. This is used to ensure that the non-interactive proofs are unique.

6.4.2.1 Stage 1: Initialising

The authorities collaborate to construct to do the following:

1. Post a security parameter N.
2. Randomly generate a large prime p of length N.
3. Randomly generate a large prime q such that $q \mid p - 1$.
4. Randomly select elements $g, h \in G_q$ such that $\log_g h$ is not known by any party.
5. Post the system parameters p, q, g and h.

6.4.2.2 Stage 2: Ballot Preparation

Each voter V_i prepares a masked vote m_i to be shared using the verifiable secret sharing scheme as follows:

1. Randomly select a masked vote $m_i \in \{1, -1\}$ and randomly generate $r_i \in \mathbb{Z}_q$ to construct the ballot

$$b_i = g^{m_i} h^{r_i} \bmod p.$$

This ballot is the commitment blob for m_i.

2. Construct a polynomial P_i of degree $t-1$ by randomly selecting $t-1$ independent coefficients $c_{i,k} \in \mathbb{Z}_q$, $1 \leq k \leq t-1$, and setting m_i as the constant term, so

$$P_i(x) = m_i + c_{i,1}x + c_{i,2}x^2 + \cdots + c_{i,t-1}x^{t-1}.$$

3. Construct a polynomial Q_i of degree $t-1$ by randomly selecting $t-1$ independent coefficients $b_{i,k} \in \mathbb{Z}_q$, $1 \leq k \leq t-1$, and setting r_i as the constant term, so

$$Q_i(x) = r_i + d_{i,1}x + d_{i,2}x^2 + \cdots + d_{i,t-1}x^{t-1}.$$

4. Commit to the coefficients of P and Q by constructing the blobs

$$b_{i,k} = g^{c_{i,k}} h^{d_{i,k}} \bmod p$$

for each k, $1 \leq k \leq t-1$.

5. Post the commitment blobs b_i and $b_{i,k}$, $1 \leq k \leq t-1$

6. Post a unique proof for b_i that is specific to V_i by using ID_i (see Section 6.4.3.1).

7. Create J shares $s_{i,j} = (m_{i,j}, r_{i,j})$, $1 \leq j \leq J$, such that

$$\begin{aligned} m_{i,j} &= P_i(j) \bmod q, \\ r_{i,j} &= Q_i(j) \bmod q \end{aligned}$$

and send each $s_{i,j}$ to A_j through a private channel.

6.4.2.3 Stage 3: Voting

1. V_i casts a vote v_i by computing $w_i \in \{1, -1\}$ such that $v_i = m_i w_i$.

2. V_i then posts w_i.

6.4.2.4 Stage 4: Counting

1. Each authority A_j verifies the following for each voter V_i:

   (a) The proof for b_i is valid.

(b) The share $s_{i,j} = (m_{i,j}, r_{i,j})$ is valid, which is verified by checking that

$$g^{m_j} h^{r_j} \equiv b_i \prod_{k=1}^{t-1} b_{i,k}^{j^i} \pmod{p}.$$

See Section 5.3.8 for an explanation of how this works.

2. Assuming all votes are valid, A_j computes the subtally

$$\tau_j = \left(\sum_{i=1}^{I} m_{i,j} w_i \right) \bmod q$$

and the corresponding commitment key

$$\sigma_j = \left(\sum_{i=1}^{I} r_{i,j} w_i \right) \bmod q$$

and then posts these values.

3. Due to the $(+, \times)$-homomorphic property of the commitment scheme, any party may verify the share (τ_j, σ_j) posted by A_j is valid by checking that the commitment blob is

$$g^{\tau_j} h^{\sigma_j} \equiv \prod_{i=1}^{I} \left(b_i \prod_{k=1}^{t-1} b_{i,k}^{j^k} \right)^{w_i} \pmod{p}$$

where the exponents are all taken to be modulo q.

4. Due to the $(+, +)$-homomorphic property of the secret sharing scheme, the tally τ of the election is the shared secret and is reconstructed as per Shamir's scheme. Let S be the set of any t subtallies used to reconstruct τ. Simply compute

$$\tau = \sum_{j \in S} P_j(0) \, \tau_j$$

where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k - j}.$$

This can be done by any party.

6.4.3 Interactive proofs

Since verifiable secret sharing is used, the only proof necessary is for the validity of the ballot. It is a witness indistinguishable proof.

6.4.3.1 Proving the ballot is valid

A ballot b is valid if it is a commitment blob for a vote $v = 1$ or $v = -1$, so it is of the form

$$b = g^v h^r \bmod p$$

for some $r \in \mathbb{Z}_q$, and r is known by the voter. This is to be proven by the voter without revealing v. The proof may be verified by any party.

The premise of this proof is based on the fact that a commitment blob can only be opened in a single way. The voter proves that either

$$r = \log_h (b/g)$$

or

$$r = \log_h (bg)$$

is known, which corresponds to either $v = 1$ or $v = -1$, respectively. It is assumed that b, p and q are publicly known. The proof is described in interactive form.

If $v = 1$, then $b = gh^r \bmod p$. The following is done to prove that $r = \log_h (b/g)$ is known:

1. The voter randomly selects the commitment $m \in \mathbb{Z}_q$ and additional values $x, y \in \mathbb{Z}_q$.

2. Construct the witnesses

$$\begin{aligned} w &= h^x (bg)^{-y} \bmod p, \\ w' &= h^m \bmod p \end{aligned}$$

   and send these to the verifier.

3. The verifier randomly selects a challenge $c \in \mathbb{Z}_q$ and sends this to the voter.

4. The voter constructs

$$\begin{aligned} x' &= (m + ry') \bmod q, \\ y' &= (c - y) \bmod q \end{aligned}$$

   and sends x, x', y and y' to the verifier.

If $v = -1$, then $b = h^r/g \bmod p$. The following is done to prove that $r = \log_h (bg)$ is known:

1. The voter randomly selects the commitment $m \in \mathbb{Z}_q$ and additional values $x', y' \in \mathbb{Z}_q$.

2. Construct the witnesses

$$\begin{aligned} w &= h^m \bmod p, \\ w' &= h^{x'} (b/g)^{-y'} \bmod p \end{aligned}$$

and send these to the verifier.

3. The verifier randomly selects a challenge $c \in \mathbb{Z}_q$ and sends this to the voter.

4. The voter constructs

$$\begin{aligned} x &= (m + ry) \bmod q, \\ y &= (c - y') \bmod q \end{aligned}$$

and sends x, x', y and y' to the verifier.

For either case, the verifier checks that

$$\begin{aligned} c &\equiv y + y' \pmod q, \\ w &\equiv h^x (bg)^{-y} \pmod p, \\ w' &\equiv h^{x'} (b/g)^{-y'} \pmod p. \end{aligned}$$

This is a very clever proof, because the verifier cannot distinguish between whether $v = 1$ or $v = -1$, hence the term "witness indistinguishable". Cheating can only occur if the discrete logarithm problem can be solved.

The verifier may be replaced by using the Fiat-Shamir heuristic to randomly generate the challenge c as the hash of

$$b, w, w', ID$$

where ID is the binary string unique to each voter used to prevent vote duplication. The proof may then be verified by any party.

6.4.4 Evaluation

6.4.4.1 Security

1. Privacy with respect to voters is satisfied by posting only commitment blobs for ballots and ballot shares. Privacy with respect to authorities is only violated if t or more authorities collude to reveal the vote shares.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to commit their ballot.

3. Robustness with respect to voters is satisfied by providing proofs for all actions and using verifiable secret sharing. Robustness with respect to authorities is only violated if t or more authorities collude to disrupt the election.

4. Public verifiability is satisfied since any party, including passive observers, may verify the proofs and the commitment blobs for the subtallies. Also, any party may compute the tally from any t subtallies.

5. Correctness is satisfied by the use of the bulletin board for all communication, except for distributing shares to authorities, and by providing proofs for all actions.

6. Fairness is satisfied because during the voting stage, each authority can only reveal subtallies of vote shares. Since counting occurs after all votes have been cast and requires a threshold of authorities to collaborate to decrypt the votes, intermediate tallies cannot be leaked unless t or more authorities are dishonest.

7. Vote independence is satisfied by encrypting ballots using a probabilistic algorithm. In order to prove that a ballot is valid, the ballot must be known, as the proofs are unique to each voter. This is why the ID is used in the hash function for the Fiat-Shamir heuristic.

6.4.4.2 Usability

1. The computational complexity is $O(NJ)$ for each voter and $O(NI)$ for each authority. These are dominated by the ballot preparation. To prepare the ballot takes $O(N)$ modular multiplications for each of J ballot shares, which is $O(NJ)$ for every voter. To verify each ballot share takes $O(N)$ modular multiplications, and every authority must do this for I voters, which is $O(NI)$.

2. The communication complexity is $O(NJ)$ for each voter and $O(NI)$ for each authority. Again, these are dominated by the ballot preparation. A ballot consists of J shares, each being $O(N)$ bits, so every voter must transmit $O(NJ)$ bits. Every authority receives ballot shares from I voters, each being $O(N)$ bits, which is $O(NI)$ bits.

3. The round complexity is 1. A voter may simply transmit all the values constructed during the Ballot Preparation and Voting stages in a single session.

6.4.4.3 Extensions

The authors propose extensions for parallel elections and multiway voting, which are efficient as all the complexities are only increased by a factor linear in the number of options L. Arbitrary votes are also possible, but the proofs become much more complicated.

6.5 Cramer, Gennaro and Schoenmakers [CGS97]

Cramer, Gennaro and Schoenmakers [CGS97] modify the previous scheme to create an extremely efficient protocol by employing a threshold cryptosystem, instead of verifiable secret sharing, to ensure privacy and robustness. Each voter submits a single vote encrypted by the public key of the authorities, along with a proof of validity. The private key is shared amongst the authorities, who collaborate to decrypt the tally without revealing the private key.

All communication in this scheme is public, with messages being posted on a bulletin board.

6.5.1 Cryptographic tools

6.5.1.1 Encryption

The threshold ElGamal cryptosystem from Section 5.3.7 is used in G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$, where p is prime. The public key is (p, g, h) and the private key is s. Each authority A_j, $1 \leq j \leq J$, has a share s_j of the private key.

ElGamal encryption is $(\times, \times)$-homomorphic, but a $(+, \times)$-homomorphic encryption is desired. This can be achieved by making some slight modifications. Using a fixed generator f a message $m \in \mathbb{Z}_q$ is encrypted as

$$c = (x, y)$$

where

$$\begin{aligned} x &= g^r \bmod p, \\ y &= h^r f^m \bmod p \end{aligned}$$

where $r \in \mathbb{Z}_q$ is randomly chosen. This is simply the ElGamal encryption of f^m. Let E denote probabilistic encryption using (p, f, g, h).

The scheme represents a YES vote by $v = 1$ and a NO vote by $v = -1$. The tally of the election is thus simply the sum of all the votes. Since E is $(+, \times)$-homomorphic, the tally can be calculated by forming the product of all the encrypted votes and then decrypting this product.

6.5.1.2 Decryption

The encrypted tally $E(\tau) = (x, y)$ is decrypted without revealing the private key s.

1. Each authority A_j, $1 \leq j \leq J$, reveals

   $$u_j = x^{s_j} \bmod p$$

   and proves that

   $$\log_g h_j = \log_x u_j.$$

   This proof is described in Section 6.5.3.1.

2. Let S be a set of any t authorities collaborating to reconstruct the secret. Assuming that all these members have given valid proofs, the message is decrypted as

$$f^{\tau} = \left(y / \prod_{j \in S} u_j^{P_j(0)} \right) \bmod p$$

where

$$P_j(0) = \prod_{k \in S, k \neq j} \frac{k}{k-j}.$$

3. Since each vote is either 1 or -1, then $-I \leq \tau \leq I$. So f^{τ} is decrypted by an exhaustive search of all possible values modulo τ. This may be done by any party.

6.5.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$. Each voter V_i has a unique binary string ID_i for identification. This is used to ensure that the non-interactive proofs are unique.

6.5.2.1 Stage 1: Initialising

The authorities collaborate to do the following:

1. Post a security parameter N.
2. Randomly generate a large prime p of length N.
3. Randomly generate a large prime q such that $q \mid p-1$.
4. Randomly select distinct generators $f, g \in G_q$.
5. Randomly select an integer $s \in \mathbb{Z}_q^*$ and compute

$$h = g^s \bmod p.$$

   Note that s must not be known by any of the authorities, and may be generated by a trusted third party.

6. Split s into J shares s_j, $1 \leq j \leq J$, and distribute s_j to the authority A_j, $1 \leq j \leq J$.
7. Post the system parameters p, q, f, g and h.

6.5.2.2 Stage 2: Voting

1. Each voter V_i randomly selects $r_i \in \mathbb{Z}_q$ to encrypt the desired vote v_i as the ballot
$$b_i = E(v_i) = (x_i, y_i)$$
where
$$\begin{aligned} x_i &= g^{r_i} \bmod p, \\ y_i &= h^{r_i} f^{v_i} \bmod p. \end{aligned}$$

2. V_i posts b_i along with a unique proof of validity using ID_i (see Section 6.5.3.2).

6.5.2.3 Stage 3: Counting

1. The authorities verify the proofs of validity for each b_i.

2. The authorities form the products
$$\begin{aligned} x &\equiv \prod_{i=1}^{I} x_i \pmod p, \\ y &\equiv \prod_{i=1}^{I} y_i \pmod p. \end{aligned}$$

3. Thus $E(\tau) = (x, y)$ is the encrypted form of the tally τ. This is decrypted by any t authorities as explained in Section 6.5.1.2.

6.5.3 Interactive proofs

The proofs used are very similar in structure to those of the previous scheme by [CFSY96].

6.5.3.1 Proof of discrete logarithm equality

Each authority is required to prove that
$$\log_g h_j = \log_x u_j$$
without revealing either $\log_g h_j$ or $\log_x u_j$. This is done by showing that
$$\begin{aligned} h_j &= g^{s_j} \bmod p, \\ u_j &= x^{s_j} \bmod p \end{aligned}$$
without revealing s_j. It is assumed that h_j, u_j, p and q are publicly known. Although this proof is not zero knowledge or witness indistinguishable, it is sufficient for all intents and purposes. The proof may be verified by any party. It is described in interactive form.

1. The authority randomly selects the commitment $m \in \mathbb{Z}_q$.

2. Construct the witnesses

$$\begin{aligned} w &= g^m \bmod p, \\ w' &= x^m \bmod p \end{aligned}$$

   and send these to the verifier.

3. The verifier randomly selects a challenge $c \in \mathbb{Z}_q$ and sends this to the authority.

4. The authority constructs a response

$$r = (m + s_j c) \bmod q$$

   and sends this to the verifier.

5. The verifier checks that

$$\begin{aligned} g^r &\equiv w h_j^c \pmod q, \\ x^r &\equiv w' u_j^c \pmod q. \end{aligned}$$

The verifier may be replaced by using the Fiat-Shamir heuristic to randomly generate the challenge c as the hash of

$$h_j, u_j, w, w'.$$

The proof may then be verified by any party.

6.5.3.2 Proving the ballot is valid

A ballot b is valid if it is the encryption of a vote $v = 1$ or $v = -1$, so it is of the form

$$b = (x, y)$$

where

$$\begin{aligned} x &= g^r \bmod p, \\ y &= h^r f^v \bmod p \end{aligned}$$

for some $r \in \mathbb{Z}_q$, and r is known by the voter. This is to be proven by the voter without revealing v. The proof may be verified by any party.

The voter proves that

$$\log_g x = \log_h (y/f^v)$$

without revealing whether $v = 1$ or $v = -1$. It is assumed that x, y, p and q are publicly known. The proof is described in interactive form.

If $v = 1$, then $y = h^r f \bmod p$. The following is done to prove that $\log_g x = \log_h (y/f)$.

1. The voter randomly selects the commitment $m \in \mathbb{Z}_q$ and additional values $d, e \in \mathbb{Z}_q$.

2. Construct the witnesses

$$\begin{aligned} w_1 &= g^e x^d \bmod p, \\ w_2 &= h^e (yf)^d \bmod p, \\ w_1' &= g^m \bmod p, \\ w_2' &= h^m \bmod p. \end{aligned}$$

   and send these to the verifier.

3. The verifier randomly selects a challenge $c \in \mathbb{Z}_q$ and sends this to the voter.

4. The voter constructs

$$\begin{aligned} d' &= (c - d) \bmod q \\ e' &= (m - rd') \bmod q, \end{aligned}$$

   and sends d, d', e and e' to the verifier.

If $v = -1$, then $y = h^r / f \bmod p$. The following is done to prove that $\log_g x = \log_h (yf)$.

1. The voter randomly selects the commitment $m \in \mathbb{Z}_q$ and additional values $d', e' \in \mathbb{Z}_q$.

2. Construct the witnesses

$$\begin{aligned} w_1 &= g^m \bmod p, \\ w_2 &= h^m \bmod p, \\ w_1' &= g^{e'} x^{d'} \bmod p, \\ w_2' &= h^{e'} (y/f)^{d'} \bmod p. \end{aligned}$$

   and send these to the verifier.

3. The verifier randomly selects a challenge $c \in \mathbb{Z}_q$ and sends this to the voter.

4. The voter constructs

$$\begin{aligned} d &= (c - d') \bmod q \\ e &= (m - rd) \bmod q, \end{aligned}$$

   and sends d, d', e and e' to the verifier.

For either case, the verifier checks that

$$\begin{aligned} c &\equiv d + d' \pmod{q}, \\ w_1 &\equiv g^e x^d \pmod{p}, \\ w_2 &\equiv h^e (yf)^d \pmod{p}, \\ w_1' &\equiv g^{e'} x^{d'} \pmod{p}, \\ w_2' &\equiv h^{e'} (y/f)^{d'} \pmod{p}. \end{aligned}$$

This is essentially the same as the proof for the validity of the ballot in the previous scheme (Section 6.4.3.1). Cheating can only occur if the discrete logarithm problem can be solved.

The verifier may be replaced by using the Fiat-Shamir heuristic to randomly generate the challenge c as the hash of

$$x, y, w_1, w_2, w_1', w_2' ID$$

where ID is the binary string unique to each voter used to prevent vote duplication. The proof may then be verified by any party.

6.5.4 Evaluation

6.5.4.1 Security

1. Privacy with respect to voters is satisfied by encrypting ballots. Privacy with respect to authorities is only violated if t or more authorities collude to decrypt the ballots.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to encrypt their ballot.

3. Robustness with respect to voters is satisfied by providing proofs for all actions. Robustness with respect to authorities is only violated if t or more authorities collude to disrupt the election.

4. Public verifiability is satisfied since any party, including passive observers, may verify the proofs.

5. Correctness is satisfied by the use of the bulletin board for all communication and by providing proofs for all actions.

6. Fairness is satisfied. Since counting occurs after all votes have been cast and requires a threshold of authorities to collaborate to decrypt the votes, intermediate tallies cannot be leaked unless t or more authorities are dishonest.

7. Vote independence is satisfied by encrypting ballots using a probabilistic algorithm. In order to prove that a ballot is valid, the ballot must be known, as the proofs are unique to each voter. This is why the ID is used in the hash function for the Fiat-Shamir heuristic.

6.5.4.2 Usability

1. The computational complexity is $O(N)$ for each voter and $O(NI)$ for each authority. To prepare the ballot and to construct the proof takes $O(N)$ modular multiplications for every voter. To verify a ballot takes $O(N)$ modular multiplications, and every authority must do this for I voters, which is $O(NI)$.

2. The communication complexity is $O(N)$ for each voter and $O(NI)$ for each authority. The ballot and proof of validity consists of $O(N)$ bits, and this is all that is transmitted by every voter. During decryption, every authority obtains the ballot and proof for I voters, which is $O(NI)$ bits.

3. The round complexity is 1. A voter is only required to post a single message consisting of the ballot and a proof of validity.

6.5.4.3 Extensions

The authors propose an extension for multiway voting, but the cost is impractical because decrypting the tallies is exponential in the number of options L. The decryption of the tally in the standard scheme does not increase the computational complexity. However, for L options, computing the tallies would become the dominant term in the complexity for the authorities. The other complexities would be increased by a factor of L.

Chapter 7

Schemes Based on Mix Nets

The schemes in this chapter rely on anonymous channels to provide privacy for the voters. These channels are implemented by using mix nets as described in Section 5.3.9). Here, the authorities form a cascade of mix servers to randomly permute batches of messages from voters, thus concealing their identities. Bulletin boards are used but usually only for authorities to post messages. Thus the majority of security requirements for online elections must be achieved by employing mix nets with certain properties, for example publicly verifiable mix nets or robust mix nets.

7.1 Chaum [Cha81]

The very first election scheme was proposed by Chaum [Cha81] as an application of mix net anonymous channels. It is suitable for arbitrary voting options. The scheme uses mix nets with messages being posted on a bulletin board by the final mix in the cascade.

7.1.1 Cryptographic tools

7.1.1.1 Encryption

The RSA cryptosystem (Section 5.2.1) is used for encrypting messages. Each authority A_j has a public key (n_j, e_j) and private key d_j.

7.1.1.2 Digital signatures

Digital signatures are used to sign votes, and an anonymous electoral roll of digital pseudonyms is formed from the corresponding public keys needed to verify the signatures. The RSA digital signature scheme (Section 5.3.1) is used, which employs the same parameters as those in the encryption function described above. Each voter V_i has a public key (n_i, e_i) and private key d_i.

7.1.1.3 Mix nets

The scheme employs the very basic mix net outlined in Section 5.3.9. Each authority A_j, $1 \leq j \leq J$, is a mix. Let E_j denote encryption with the public key of A_j. The mix net is a cascade of J authorities. In order to send a message m through the mix net, the following is done:

1. The sender randomly generates J binary strings $r_1, r_2, \ldots, r_J$. Then m is encrypted with the public keys of the mixes to give $E_1(r_1 \| E_2(r_2 \| \cdots E_J(r_J \| m)))$. The randomly selected binary strings are used to eliminate the chosen plaintext attack. This encrypted message is sent to A_1.

2. Each authority A_j receives a sequence of messages, then removes the corresponding layer of encryption to produce a set of messages of the form $E_{j+1}(r_{j+1} \| E_{j+2}(r_{j+2} \| \cdots E_J(r_J \| m)))$. This batch is shuffled either by randomly permuting the order or sorting the items lexicographically. The batch is then forwarded to A_{j+1}.

3. The final authority A_J posts the messages.

All messages sent by voters are processed as a single batch by the mix net.

7.1.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$.

7.1.2.1 Stage 1: Initialising

1. The authorities post a security parameter N.

2. Each authority A_j creates a public key (n_j, e_j) and private key d_j, where n_j is of length N. The public keys are posted. Let E_j denote encryption using (n_j, e_j).

3. Each voter V_i creates a public key (n_i, e_i) and private key d_i, where n_i is of length N. Let S_i denote signing a message using d_i.

7.1.2.2 Stage 2: Authorising

An anonymous roster of eligible voters is created by using the public keys of voters as pseudonyms. Each voter V_i has a unique binary string ID_i used for identification.

1. Each voter V_i randomly generates J binary string $r_1, r_2, \ldots, r_J$ and encrypts the public key (n_i, e_i) as $m_i = E_1(r_1 \| E_2(r_2 \| \cdots E_J(r_J \| (n_i, e_i))))$.

2. V_i then sends the message $ID_i \| m_i$ through the mix net, after encrypting it with $E_1, E_2, \ldots, E_J$ along with the necessary random binary string prefixes. The messages $ID_i \| m_i$, $1 \leq i \leq I$, are posted by A_J in lexicographical order.

3. The authorities verify that ID_i is valid.

4. For all valid ID_i, $m_i = E_1(r_1 \| E_2(r_2 \| \cdots E_J(r_J \| (n_i, e_i))))$ is sent to A_1. This is processed through the mix net and the public keys (n_i, e_i), $1 \leq i \leq I$, are posted by A_J in lexicographical order. This list of untraceable public keys forms the anonymous roster.

5. Each V_i checks that (n_i, e_i) is posted, and complains if this is not the case.

7.1.2.3 Stage 3: Voting

1. Each voter V_i selects a vote v_i and signs it as $S_i(c \| v_i)$, where c is a constant.

2. V_i sends the message $(n_j, e_j) \| S_i(c \| v_i)$ through the mix net, after encrypting it with $E_1, E_2, \ldots, E_J$ along with the necessary random binary string prefixes.. The messages $(n_j, e_j) \| S_i(c \| v_i)$, $1 \leq i \leq I$, are posted by A_J in lexicographical order.

7.1.2.4 Stage 4: Counting

1. Each V_i checks that $(n_j, e_j) \| S_i(c \| v_i)$ is posted, and complains if this is not the case.

2. Any party may count the votes and use the roster of public keys to verify the votes.

7.1.3 Evaluation

7.1.3.1 Security

1. Privacy with respect to voters is satisfied by the mix net. Privacy with respect to authorities can only be violated if all the authorities are dishonest and reveal the permutations used.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to encrypt their vote.

3. Robustness with respect to voters is satisfied by using the roster of pseudonyms to verify votes. However, if any authority is dishonest, then the election can be disrupted.

4. Private verifiability is satisfied since each voter may check that their vote is counted.

5. Correctness is satisfied by the use of the bulletin board for posting votes and the roster of pseudonyms for the authentication of voters and the verification of votes.

6. Fairness is satisfied. All votes must be cast before any are processed by the mix net, so intermediate tallies cannot be leaked during the voting stage.

7. Vote independence is satisfied. All votes must be cast before any are processed by the mix net, so no voter has any knowledge of the other votes cast.

7.1.3.2 Usability

1. The computational complexity is $O\left(NJ^2\right)$ for each voter and $O\left(NIJ\right)$ for each authority. Encryption of a message takes $O\left(N\right)$ modular multiplications. Due to the message expansion from prefixing each message with randomly generated bits, this is $O\left(NJ\right)$ in the worst case for J message blocks to be encrypted. Every voter must do this for J authorities, which is $O\left(NJ^2\right)$. Decryption of a message takes $O\left(N\right)$ modular multiplications. Again, due to the message expansion by prefixing each message with randomly generated bits, this is $O\left(NJ\right)$ in the worst case for J message blocks to be decrypted. Every authority must do this for I voters, which is $O\left(NIJ\right)$.

2. The communication complexity is $O\left(NJ\right)$ for each voter and $O\left(NIJ\right)$ for the authority. An encrypted message takes $O\left(N\right)$. Due to the message expansion from prefixing each message with randomly generated bits, this is $O\left(NJ\right)$ in the worst case for J message blocks to be encrypted. Every voter transmits a constant number of messages, which is $O\left(NJ\right)$. Every authority transmits a constant number of messages for I voters, which is $O\left(NIJ\right)$.

3. The round complexity is 2. Every voter must participate in the Authorising and Voting stages. Each stage must be completed before the next stage commences, because messages are not processed by the mix net until all messages have been received by the first authority. Hence stages are sessions with specified deadlines. However, the stages are not required to occur in immediate succession, and authorising may take place well in advance of voting.

7.2 Sako and Kilian [SK95]

Sako and Kilian [SK95] improve the previous scheme by modifying the mix net to achieve public verifiability. Incoercibility is also achieved by employing chameleon blobs to allow verifiers to forge proofs. The basis of this scheme is that the randomness for encrypting the ballot is determined by the authorities, who prove the value of the ballot to the voter. The voter is able to cast the desired encrypted vote but cannot prove the value of this vote.

The authorities post messages on a bulletin board, and it is assumed that unconditionally private channels exist between voters and mixes for revealing commitments. This scheme is suitable for YES/NO voting options.

7.2.1 Cryptographic protocols

7.2.1.1 Encryption

The ElGamal cryptosystem from Section 5.2.3 is used in G_q, the unique cyclic subgroup of order q in $\mathbb{Z}_p^*$, where p is prime. The system parameters are p, q and g, which are shared. Each authority A_j, $1 \leq j \leq J$, has a private key s_j and a public key $h_j = g^{s_j} \bmod p$ for the mix net described below. Each voter V_i, $1 \leq i \leq I$, has a private key u_i and a public key $w_i = g^{u_i} \bmod p$ for the chameleon commitment scheme described below.

7.2.1.2 Mix nets

An improved mix net is developed which does not expand messages and also allows public verifiability. The mixes each have an ElGamal key pair, and the $(\times, \times)$-homomorphic property of the encryption algorithm allows a sender to efficiently encrypt a message $m \in G_q$ with multiple public keys, so that it can be sent through the mix net.

1. Randomly select an integer $r_1 \in \mathbb{Z}_q$.
2. Compute
$$\begin{aligned} x_1 &= g^{r_1} \bmod p, \\ y_1 &= z_1^{r_1} m \bmod p \end{aligned}$$
where $z_1 = h_1 h_2 \cdots h_J$ are the public keys of the authorities.
3. The ciphertext $c_1 = (x_1, y_1)$ is sent to the first mix A_1.

The notation E_n is adopted to denote probabilistic encryption using the public keys of the authorities A_j, $n \leq j \leq J$.

Each mix A_j receives a ciphertext $c_j = (x_j, y_j)$ and uses the private key s_j to remove the j-th layer of encryption, thus producing the message encrypted with the public keys of the remaining mixes. In order to conceal the relationship between the received message and the output message, a new re-encrypted message is formed from the remaining public keys. The decryption and re-encryption can be done in one step by the following:

1. Randomly select an integer $r \in \mathbb{Z}_q$.
2. Compute
$$x_{j+1} \equiv x_j g^r \equiv g^{r_j} g^r \equiv g^{r_{j+1}} \pmod p$$
and
$$y_{j+1} \equiv y_j z_{j+1}^r / x_j^{s_j} \equiv z_{j+1}^{r_j} z_{j+1}^r m \equiv z_{j+1}^{r_{j+1}} m \pmod p$$
where $r_{j+1} = r + r_j$ and $z_{j+1} = h_{j+1} h_{j+2} \cdots h_J$.

Each mix must provide a zero knowledge proof that the correct procedures are followed. This is not covered here, but can be found in the original paper [SK95].

Note that all the messages sent are permuted as a single batch by the mix net.

7.2.1.3 Chameleon commitment

The chameleon commitment scheme in Section 5.3.5 is used to provide incoercibility. The scheme represents a YES vote by $v = 1$ and a NO vote by $v = 0$. A ballot is a randomly ordered pair of encrypted 0 and 1 votes, so there are two possible permutations. Each authority shuffles the ballot for a voter, and commits to the ordering using the chameleon commitment scheme.

An authority commits to an ordering $m \in \{0, 1\}$ by randomly generating $r \in \mathbb{Z}_q$ to construct the blob

$$b = w_i^m g^r \bmod p$$

where w_i is the public key of the voter V_i.

To open b, the authority reveals r to V_i through a private channel. Thus V_i can lie about whether $m = 1$ or $m = 0$.

7.2.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$.

7.2.2.1 Stage 1: Initialising

The authorities collaborate to construct to do the following:

1. Post a security parameter N.
2. Randomly generate a large prime p of length N.
3. Randomly generate a large prime q such that $q \mid p - 1$.
4. Post the system parameters p, q, g.
5. Randomly select $v^1, v^0 \in G_q$ and post these as the standard 1 and 0 votes, respectively.

Then the following is done using the system parameters:

1. Each authority A_j creates a public key h_j and private key s_j.
2. Each authority V_i creates a public key w_i and private key u_i.

7.2.2.2 Stage 2: Ballot Preparation

1. The final authority in the cascade A_J posts a unique ballot $b_i = \left\{E_1\left(v^1\right), E_1\left(v^0\right)\right\}$ for each eligible voter V_i. Note that each element is encrypted with the public keys of all the authorities. A zero knowledge proof is provided to show that the ballot is valid, without revealing the ordering. The details of this proof can be found in [SK95].

2. A_J commits to the ordering of b_i and reveals this to V_i through a private channel.

3. The ballots b_i, $1 \leq i \leq I$, are shuffled in reverse order through the mix net, that is in the order $A_J, A_{J-1}, \ldots, A_1$. Each A_j commits to how b_i is shuffled and reveals this to V_i through a private channel.

4. The first authority in the cascade A_1 then sends the final unencrypted permutations of $\left\{v^1, v^0\right\}$ to each voter.

7.2.2.3 Stage 3: Voting

1. Each voter V_i knows the ordering of b_i by keeping track of the permutations made by each authority.

2. V_i casts the vote $v_i \in \left\{v^1, v^0\right\}$ by sending $E_1\left(v_i\right) \in b_i$ through the mix net, this time in forward order and without commitments being required by each authority.

3. The first authority in the cascade A_1 verifies that V_i has not submitted a vote previously and that $E_1\left(v_i\right) \in b_i$.

7.2.2.4 Stage 4: Counting

1. The final authority in the cascade A_J posts the list of votes v_i, $1 \leq i \leq I$.

2. Any party may compute the tally from these posted votes.

7.2.3 Evaluation

7.2.3.1 Security

1. Privacy with respect to voters is satisfied by the mix net. Privacy with respect to authorities can only be violated if all the authorities are dishonest and reveal the permutations used.

2. Incoercibility is satisfied. A voter cannot prove the ordering of the ballot, due to the chameleon commitment scheme.

3. Robustness with respect to voters is satisfied since voters can only cast votes from the ballot created by the authorities. However, if any authority is dishonest, then the election can be disrupted.

4. Public verifiability is satisfied since the authorities provide public proofs for all actions.

5. Correctness is satisfied since a unique ballot is created by the authorities for each eligible voter.

6. Fairness is satisfied. All votes must be cast before any are processed by the mix net, so intermediate tallies cannot be leaked during the voting stage.

7. Vote independence is satisfied. Voters can only cast votes from the designated ballots created by the authorities.

7.2.3.2 Usability

1. The computational complexity is $O(NJ)$ for each voter and $O(NI)$ for each authority. A commitment takes $O(N)$ modular multiplications to verify, and every voter must do this for J authorities, which is $O(NJ)$. Encryption and decryption of a message takes $O(N)$ modular multiplications, and every authority must do this for I voters, which is $O(NI)$.

2. The communication complexity is $O(NJ)$ for each voter and $O(NI)$ for the authority. A commitment takes $O(N)$ bits, and every voter receives commitments from J authorities, which is $O(NJ)$. An encrypted message takes $O(N)$ bits, and every authority transmits a constant number of messages for I voters, which is $O(NI)$.

3. The round complexity is 1. The Ballot Preparation and Voting stages may be conducted in a single session.

7.3 Hirt and Sako [HS00]

Hirt and Sako [HS00] modify the mix net in the scheme by Sako and Kilian to create a generic protocol. This can be applied to make any homomorphic scheme incoercible, as long as the encryption function satisfies certain properties. A specific example is given in the paper using [CGS97]. Robustness is also achieved by allowing voters to reject permutations made by dishonest authorities.

Both authorities and voters post messages on a bulletin board, and it is assumed that digital signatures are employed for authentication of voters. It is also assumed that unconditionally private channels exist between voters and mixes for revealing permutations. The scheme outlined here is suitable for multiway voting, but the voting options naturally depend on the specific homomorphic scheme used.

7.3.1 Cryptographic tools

Any election scheme based on homomorphic encryption can be used if the following are satisfied:

1. The encryption algorithm E is $(+, \times)$-homomorphic and probabilistic.
2. The decryption is verifiable, so that it can be proven that $E(m)$ is an encryption of a message m, without revealing the randomness used.
3. A message m can be re-encrypted multiple times as $E(E(\cdots E(m)))$.
4. There exists is a $1-L$ re-encryption proof to show that a set of encrypted messages corresponds to a re-encrypted set of these messages. The ordering of the re-encrypted set is different, and the permutation used must not be revealed.
5. There exists a designated verifier re-encryption proof to show a specific verifier, and no other party, that $E(E(m))$ is a re-encryption of $E(m)$.

7.3.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and J authorities A_j, $1 \leq j \leq J$.

7.3.2.1 Stage 1: Initialising

The authorities collaborate to do the following:

1. Generate the necessary system parameters, including the public key.
2. Let E denote probabilistic encryption using the public key.
3. Let $v_1, v_2, \ldots, v_L$ be the set of all valid votes for L options.
4. Post the standard ballot $b_0 = E(v_1), E(v_2), \ldots, E(v_L)$. This is a list of standard encrypted valid votes, where the randomness for E is set to a known constant, such as the all-0 string.

7.3.2.2 Stage 2: Ballot Preparation

Each authority A_j does the following for each voter V_i:

1. Randomly select a permutation $\pi_{i,j}$ for shuffling the ballot $b_{i,j-1}$ to produce the ballot $b_{i,j}$. The process of shuffling involves re-encrypting each element and then reordering the values. Note that the first authority in the cascade A_1 takes the standard ballot b_0 to be shuffled.
2. A $1-L$ re-encryption proof is publicly conducted to show that the shuffling is valid.
3. The permutation $\pi_{i,j}$ along with a designated verifier proof for each re-encrypted element of the ballot is sent to V_i through a private channel.

4. V_i may publicly complain about this proof, in which case the shuffling of the authority is ignored. That is, the next authority A_{j+1} shuffles the ballot $b_{i,j-1}$ instead of $b_{i,j}$. A voter may make complaints against at most $J - t$ authorities.

5. The final authority in the cascade A_J posts the ballot $b_{i,J}$.

7.3.2.3 Stage 3: Voting

1. Each voter V_i knows the ordering of $b_{i,J}$ by keeping track of the permutations made by each authority.

2. V_i casts a vote by posting w_i, $1 \leq w_i \leq L$, which is the position of the desired vote in $b_{i,J}$.

7.3.2.4 Stage 4: Counting

1. Using the pairs w_i and $b_{i,J}$, $1 \leq i \leq I$, the desired encrypted votes of each voter are combined to produce the encrypted tally (or tallies).

2. Due to the $(+, \times)$-homomorphic property of E, this is decrypted to give the tally, and a proof is given to verify the decryption.

7.3.3 Evaluation

7.3.3.1 Security

1. Privacy with respect to voters is satisfied by the mix net. Privacy with respect to authorities can only be violated if all the authorities are dishonest and reveal the permutations used.

2. Incoercibility is satisfied. A voter cannot prove the ordering of the ballot, due to the designated verifier proofs.

3. Robustness with respect to voters is satisfied since voters can only cast votes from the ballot created by the authorities. Robustness with respect to authorities is only violated if t or more authorities collude to disrupt the election.

4. Public verifiability is satisfied since the authorities provide public proofs for all actions.

5. Correctness is satisfied by the use of the bulletin board for posting votes and ballots and by providing proofs for all actions.

6. Fairness is satisfied. All votes must be cast before any are processed by the mix net, so intermediate tallies cannot be leaked during the voting stage.

7. Vote independence is satisfied. Each ballot is unique for each voter.

7.3.3.2 Usability

1. The computational and communication complexities depend on the specific homomorphic scheme used.

2. The round complexity is 1. The Ballot Preparation and Voting stages may be conducted in a single session.

Chapter 8

Schemes Based on Blind Signatures

Blind signatures (see Section 5.3.2) are commonly used to obtain anonymous certificates which can later be used by parties to provide authentication without revealing their identities. The schemes covered in this chapter employ this technique to provide privacy. Voters may communicate anonymously with the authorities after being granted a certificate of eligibility. It is assumed that anonymous channels are available, and these may be implemented by mix nets or other means. These schemes are suitable for arbitrary voting options.

8.1 Fujioka, Okamoto and Ohta [FOO93]

Fujioka, Okamoto and Ohta [FOO93] present an election scheme that employs digital signatures and blind signatures based on RSA signatures, and a simple commitment scheme. It is suitable for arbitrary voting options. Bulletin boards are used by authorities to post results.

8.1.1 Cryptographic tools

8.1.1.1 Bit commitment

A bit commitment scheme is employed to ensure fairness by concealing the vote until the counting stage. Any scheme is suitable, such as that of Pedersen (Section 5.3.3). The notation $BC(v, r)$ is adopted here to denote the commitment blob of a vote v using a randomly chosen r. The blob is opened by revealing v and r.

8.1.1.2 Digital signatures

Digital signatures, such as the RSA digital signature scheme (Section 5.3.1), are used by voters to sign commitment blobs. The notation $S_i(m)$ is adopted here

to denote signing a message m by a voter V_i

8.1.1.3 Blind signatures

Eligible voters obtain a blind signature for the commitment blob of their vote. This is used as the anonymous certificate, so that votes can be cast anonymously. The RSA based scheme in Section 5.3.2 is employed. The notation $S_A(m)$ is adopted here to denote signing a message m by the administrator A.

8.1.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and two different authorities: an administrator A for authorising voters, and a tallier T for counting the votes. Each voter V_i has a unique binary string ID_i for identification. This is used for A to authorise the voter.

8.1.2.1 Stage 1: Initialising

The administrator A does the following:

1. Post a security parameter N.
2. Generate and post the parameters for the blind signature scheme and the commitment scheme, using the security parameter N.

Each voter V_i creates a public and private key pair for the digital signature scheme, using the security parameter N.

8.1.2.2 Stage 2: Ballot Preparation

1. Each voter V_i selects a vote v_i and randomly generates a commitment key r_i to construct a ballot
$$b_i = BC(v_i, r_i).$$
2. V_i randomly generates a blinding factor to blind b_i as b_i'.
3. V_i signs b_i' as $S_i(b_i')$ and sends $(ID_i, b_i', S_i(b_i'))$ to A.

8.1.2.3 Stage 3: Authorising

1. The authority A uses ID_i to verify that V_i is eligible to vote and has not previously applied for a certificate.
2. A verifies that $S_i(b_i')$ is the signature for b_i'.
3. If all the checks are successful, then A signs b_i' as $S_A(b_i')$, which is the blinded certificate. A sends this to V_i.
4. V_i unblinds $S_A(b_i')$ to retrieve the actual signature c_i for b_i, which is used as the certificate.

5. V_i verifies that c_i is the signature for b_i.

6. If c_i is invalid, then V_i complains by revealing (b_i, c_i) and may obtain a new pair by repeating this process.

7. At the end of this stage, A posts the list C of all $(ID_i, b'_i, S_i(b'_i))$ for which a certificate was granted.

8.1.2.4 Stage 4: Voting

1. The voter V_i casts (b_i, c_i) to the tallier T through an anonymous channel.

2. The tallier T verifies that c_i is the signature for b_i. If this check succeeds, then (l_i, b_i, c_i) is added to the list of valid ballots as the l_i-th element.

3. At the end of this stage, T posts the list B of valid ballots.

8.1.2.5 Stage 5: Opening

1. The voter V_i checks that (b_i, c_i) is in B. If this check fails, then V_i claims this by revealing (b_i, c_i). The vote is ignored or the election must be restarted. If this vote is added to the list, then privacy is violated.

2. V_i sends v_i, r_i and l_i to T through an anonymous channel.

8.1.2.6 Stage 6: Counting

1. The tallier T uses v_i and r_i to open the ballot b_i, which corresponds to the l_i-th element ballot on the list B.

2. T counts the valid votes and posts the result.

8.1.3 Evaluation

8.1.3.1 Security

1. Privacy with respect to voters is satisfied by posting only commitment blobs for votes. Privacy with respect to authorities is satisfied by using blind signatures and anonymous channels.

2. Incoercibility is not satisfied. A voter may construct a receipt by simply revealing all the values used to commit their vote.

3. Robustness with respect to voters is satisfied by the use of commitment blobs for votes and obtaining blind signatures for these. Robustness with respect to authorities is not satisfied because if either the administrator or the tallier is dishonest, then the election can be disrupted.

4. Private verifiability is satisfied since each voter may check that their ballot is in B, the list of valid ballots.

5. Correctness is satisfied by authorising each voter and also by posting the list of valid ballots.

6. Fairness is satisfied since opening and counting occur after all votes have been cast, so intermediate tallies cannot be leaked.

7. Vote independence is satisfied by posting only commitment blobs for votes. In order to open the blob, the vote and commitment key must be known.

8.1.3.2 Usability

1. The computational complexity is $O(N)$ for each voter and $O(NI)$ for each authority. Signing or verifying messages takes $O(N)$ modular multiplications, as does constructing or verifying commitment blobs. Every voter does this for a constant number of messages, which is $O(N)$. Each authority does this for I voters, which is $O(NI)$.

2. The communication complexity is $O(N)$ for each voter and $O(NI)$ for each authority. All messages are $O(N)$ bits each. Every voter transmits a constant number of messages, which is $O(N)$. Each authority transmits a constant number of messages for I voters, which is $O(NI)$.

3. The round complexity is 2. Initialising, ballot preparation, authorising and voting may all be done in one session. However, to ensure fairness, opening and counting must be done after all voters have completed voting, so another session is required.

8.2 Okamoto [Oka97]

Okamoto [Oka97] improves the previous scheme to achieve incoercibility and public verifiability. This protocol is in fact a slight modification of [Oka96] and fixes a minor flaw preventing incoercibility. A bulletin board is used for posting certain messages, but authentication is not provided. It is assumed that anonymous, unconditionally private channels exist between voters and the tallier T.

For simplicity, the scheme outlined here has a single tallier, but the paper describes how a simple secret sharing scheme (see Section 5.3.6) can be used to share the vote amongst multiple talliers. A separate certificate must be obtained for each share.

8.2.1 Cryptographic tools

8.2.1.1 Encryption

Messages sent to the administrator A are encrypted using the RSA cryptosystem (Section 5.2.1). The notation $E_A(m)$ is adopted here to denote encrypting a message m with the public key of the administrator A.

8.2.1.2 Trapdoor commitment

As in the previous scheme, ballots are commitment blobs for votes. However, by using the trapdoor commitment scheme in Section 5.3.4, the blob can be opened as any vote. The notation $BC(v, r)$ is adopted here to denote the commitment blob of a vote v using a randomly chosen r. The blob is opened by revealing any appropriate pair v' and r'.

8.2.1.3 Digital signatures

Digital signatures, such as the RSA digital signature scheme (Section 5.3.1), are used by voters to sign commitment blobs. The notation $S_i(m)$ is adopted here to denote signing a message m by a voter V_i

8.2.1.4 Blind signatures

Eligible voters obtain a blind signature for the commitment blob of their vote. This is used as the anonymous certificate, so that votes can be cast anonymously. The RSA based scheme in Section 5.3.2 is employed. The notation $S_A(m)$ is adopted here to denote signing a message m by the administrator A.

8.2.2 Election protocol

The participants are I voters V_i, $1 \leq i \leq I$, and two different authorities: an administrator A for authorising voters, and a tallier T for counting the votes. Each voter V_i has a unique binary string ID_i for identification. This is used for A to authorise the voter.

8.2.2.1 Stage 1: Initialising

The administrator A does the following:

1. Post a security parameter N.
2. Generate and post the parameters for the encryption scheme and the blind signature scheme using the security parameter N.
3. Randomly generate a large prime p of length N.
4. Randomly generate a large prime q such that $q \mid p - 1$.
5. Randomly select generators $g, h \in G_q$ such that $\log_g h$ is not known by any party.
6. Post the system parameters p, q, g and h.

Each voter V_i creates a public and private key pair for the digital signature scheme, using the security parameter N.

8.2.2.2 Stage 2: Ballot Preparation

1. Each voter V_i randomly generates a trapdoor $t_i \in \mathbb{Z}_q^*$ and computes $f_i = g^{t_i} \bmod p$.
2. V_i selects a vote v_i and randomly generates a commitment key $r_i \in \mathbb{Z}_q$ to construct the ballot

$$b_i = BC(v_i, r_i) = g^{v_i} f_i^{r_i} \bmod p.$$

3. V_i constructs a hash $x_i = H(b_i \| f_i)$ and randomly generates a blinding factor to blind x_i as x_i'.
4. V_i signs x_i' as $S_i(x_i')$ and sends $E_A(S_i(x_i') \| x_i' \| ID_i)$ to A.

8.2.2.3 Stage 3: Authorising

1. The authority A decrypts $E_A(S_i(x_i') \| x_i' \| ID_i)$ and uses ID_i to verify that V_i is eligible to vote and has not previously applied for a certificate.
2. A verifies that $S_i(x_i')$ is the signature for x_i'.
3. If all the checks are successful, then A signs x_i' as $S_A(x_i')$, which is the blinded certificate. A sends this to V_i.
4. V_i unblinds $S_A(x_i')$, to retrieve the actual signature c_i for $x_i = H(b_i \| f_i)$, which is used as the certificate.

8.2.2.4 Stage 4: Voting

1. The voter V_i sends $(b_i \| f_i, c_i)$ to the bulletin board through anonymous channels.
2. V_i sends (v_i, r_i, b_i) to the tallier T through anonymous channels.
3. V_i proves to T that the trapdoor t_i for the commitment blob b_i is known. This is necessary to ensure that V_i can open b_i as any desired vote and a coercer has not provided V_i with b_i, without the trapdoor. The proof is conducted through anonymous private channels and requires proving knowledge of discrete logarithms, which in this case is the trapdoor t_i. A suitable proof is described in the paper.

8.2.2.5 Stage 5: Claiming

1. The voter V_i verifies their ballot is posted on the bulletin board. If this is not the case, $(b_i \| f_i, c_i)$ is used to claim this.
2. The tallier T verifies that v_i, r_i is a correct opening of b_i.

8.2.2.6 Stage 6: Counting

1. The tallier T takes all the valid (v_i, r_i, b_i) and partitions them into n disjoint groups of votes, where each group covers all different votes if possible.

2. For each group k, $1 \leq k \leq n$, T generates a list V'_k, consisting of the randomly permuted votes, and a list B'_k consisting of the randomly permuted commitment blobs.

3. T posts (V'_k, B'_k), $1 \leq k \leq n$, and provides a publicly verifiable proof that the votes in V'_k correspond to the commitment blobs in B'_k. This is done without revealing the permutation used or any of the commitment keys r_i, and the details are covered in the original paper.

4. Any party may then compute the tallies from the posted votes.

8.2.3 Evaluation

8.2.3.1 Security

1. Privacy with respect to voters is satisfied by posting only commitment blobs for votes. Privacy with respect to authorities is satisfied by using blind signatures and anonymous channels.

2. Incoercibility is satisfied. Since a trapdoor commitment scheme is used and the desired opening is known only to the tallier, a voter may not construct a receipt for their vote.

3. Robustness with respect to voters is satisfied by the use of commitment blobs for votes and obtaining blind signatures for these. Robustness with respect to authorities is not satisfied because if the administrator is dishonest, then the election can be disrupted.

4. Public verifiability is satisfied since the tallier provides proofs that the votes are valid. Private verifiability is satisfied since each voter may check that their ballot is posted on the bulletin board.

5. Correctness is satisfied by authorising each voter and also by posting all ballots.

6. Fairness is satisfied in the multiple tallier version since counting occurs after all votes have been cast, so intermediate tallies cannot be leaked.

7. Vote independence is satisfied by posting only commitment blobs for votes. In order to open the blob, the vote and commitment key must be known.

8.2.3.2 Usability

1. The computational complexity is $O(NJ)$ for each voter and $O(NI)$ for each authority. Signing or verifying messages takes $O(N)$ modular multiplications, as does constructing or verifying commitment blobs. Every voter does this J talliers, which is $O(NJ)$. Each authority does this for I voters, which is $O(NI)$.

2. The communication complexity is $O(NJ)$ for each voter and $O(NI)$ for each authority. All messages are $O(N)$ bits each. Every voter transmits a messages to J talliers, which is $O(NJ)$. Each authority transmits a constant number of messages for I voters, which is $O(NI)$.

3. The round complexity is 1. Voters may vote in a single session in the multiple tallier version.

Chapter 9

Comparison of Existing Schemes

The last three chapters have covered a wide variety of schemes for online elections, each with different properties. This chapter compares the schemes and discusses which ones are suitable for conducting elections in practice. Firstly, a brief overview of the achieved requirements and properties are presented in tabular form. Then there is a more detailed discussion of the security requirements as outlined in Section 3.3 and the usability properties as covered in Section 3.4. Finally, a summary is given to explain which schemes are the best for carrying out online elections.

9.1 Overview

In the following tables, N is a security parameter, I is the number of voters and J is the number of authorities. These apply to the computational and communication complexities.

	[CF85]	[Ben87b]	[BT94]	[CFSY96]	[CGS97]
Privacy (voters)	yes	yes	yes	yes	yes
Privacy (authorities)	no	yes	yes	yes	yes
Incoercibility	no	no	no	no	no
Robustness (voters)	yes	yes	yes	yes	yes
Robustness (authorities)	no	yes	yes	yes	yes
Verifiability	public	public	public	public	public
Correctness	yes	yes	yes	yes	yes
Fairness	no	yes	yes	yes	yes
Independence	yes	yes	yes	yes	yes
Computation (voters)	$O\left(N^2\right)$	$O\left(N^2J\right)$	$O\left(N^2J\right)$	$O\left(NJ\right)$	$O\left(N\right)$
Computation (authorities)	$O\left(N^2I\right)$	$O\left(N^2IJ\right)$	$O\left(N^2IJ\right)$	$O\left(NI\right)$	$O\left(NI\right)$
Communication (voters)	$O\left(N^2\right)$	$O\left(N^2J\right)$	$O\left(N^2J\right)$	$O\left(NJ\right)$	$O\left(N\right)$
Communication (authorities)	$O\left(N^2I\right)$	$O\left(N^2IJ\right)$	$O\left(N^2IJ\right)$	$O\left(NI\right)$	$O\left(NI\right)$
Rounds	2	2	2	1	1
Voting options	YES/NO	YES/NO	YES/NO	YES/NO	YES/NO

Table 9.1: Comparison of homomorphic schemes

	[Cha81]	[SK95]	[HS00]
Privacy (voters)	yes	yes	yes
Privacy (authorities)	yes	yes	yes
Incoercibility	no	yes	yes
Robustness (voters)	yes	yes	yes
Robustness (authorities)	no	no	yes
Verifiability	private	public	public
Correctness	yes	yes	yes
Fairness	yes	yes	yes
Independence	yes	yes	yes
Computation (voters)	$O\left(NJ^2\right)$	$O\left(NJ\right)$	-
Computation (authorities)	$O\left(NIJ\right)$	$O\left(NI\right)$	-
Communication (voters)	$O\left(NJ\right)$	$O\left(NJ\right)$	-
Communication (authorities)	$O\left(NIJ\right)$	$O\left(NI\right)$	-
Rounds	2	1	1
Voting options	Arbitrary	YES/NO	Multiway

Table 9.2: Comparison of mix net schemes

	[FOO93]	[Oka97]
Privacy (voters)	yes	yes
Privacy (authorities)	yes	yes
Incoercibility	no	yes
Robustness (voters)	yes	yes
Robustness (authorities)	no	no
Verifiability	private	public and private
Correctness	yes	yes
Fairness	yes	yes
Independence	yes	yes
Computation (voters)	$O(N)$	$O(NJ)$
Computation (authorities)	$O(NI)$	$O(NI)$
Communication (voters)	$O(N)$	$O(NJ)$
Communication (authorities)	$O(NI)$	$O(NI)$
Rounds	2	1
Voting options	Arbitrary	Arbitrary

Table 9.3: Comparison of blind signature schemes

9.2 Security Requirements

These requirements determine how effective election schemes are at satisfying principles for democratic elections.

9.2.1 Privacy

Privacy with respect to voters is obviously compulsory, to protect the secrecy of the votes from any coalition of voters. All the schemes achieve this. Privacy with respect to authorities is also important, as authorities might not be trusted. With the exception of [CF85], all schemes achieve this. Privacy can only be violated in homomorphic schemes if a threshold of t or more authorities are dishonest and reveal the encrypted values. In mix net schemes, all authorities must be dishonest and reveal the permutations used. In blind signature schemes, privacy is always guaranteed since messages are signed blindly, so even a dishonest authority cannot reveal the original message.

9.2.2 Incoercibility

Incoercibility is the most difficult requirement to satisfy. Only the schemes in [SK95], [HS00] and [Oka97] manage to achieve incoercibility. However, these all require the authorities to reveal some secret information to the voters, who must no be able to prove this secret to any party. Thus they rely on the existence of unconditionally private communication channels. This is a strong assumption as such channels are very difficult to implement in practice. However, it seems necessary for incoercibility.

9.2.3 Robustness

Robustness with respect to voters is essential, to ensure that the election cannot be disrupted by any coalition of voters. All the schemes achieve this. Robustness with respect to authorities is also important, as authorities may be dishonest. With the exception of [CF85], all homomorphic schemes achieve this as long as a threshold of t or more authorities are honest. The only mix net scheme to achieve this is [HS00], and neither of the blind signature schemes satisfy robustness with respect to authorities.

9.2.4 Verifiability

All of the schemes are verifiable. Public verifiability is preferred, so that any party may verify the result. Private verifiability relies on all voters to be vigilant in verifying their individual votes. All homomorphic schemes are publicly verifiable. [Cha81] is privately verifiable, and the other mix net schemes are publicly verifiable. Blind signature schemes are privately verifiable, however [Oka97] is also publicly verifiable, and is the only election scheme to achieve both forms of verifiability, which is ideal.

9.2.5 Correctness

All of the schemes satisfy the correctness properties of eligibility, uniqueness, integrity and completeness. That is only eligible voters can cast votes, each voter may only cast a single vote, votes must not be modified after they are cast, and all valid votes are counted.

9.2.6 Fairness

With the exception of [CF85], all schemes achieve fairness as intermediate tallies cannot be revealed.

9.2.7 Vote independence

All schemes achieve vote independence, as voters cannot copy the votes of others or deliberately cast the opposite vote to others.

9.3 Usability Properties

These properties determine how much effort is required by the participants of an election scheme. This must be as little as possible for the voters, and reasonable for the authorities. Being generous, the voting process should take at most 10 minutes for the voter and computing the result should take at most 5 hours for the authorities.

The computational and communication complexities are given in terms of a security parameter N, the number of voters I and the number of authorities J. Some rough figures are given for practical values of these variables. For most elections, $N = 1024$ bits or possibly more for longer term security. For smaller scale elections, such as university elections, $I = 10^4$ and $J = 10$. For large scale elections, $I = 10^7$ and $J = 1000$.

9.3.1 Computational complexity

The complexities are given in terms of N bit modular multiplications performed in the worst case. With current technology, it is assumed that the computational power of a voter is in the order of 1 GHz and the computational power of an authority is in the order of 100 GHz.

The most inefficient protocols are the earlier homomorphic schemes [CF85], [Ben87b] and [BT94]. The complexity for the voter is quadratic in N and linear in J, which is acceptable for smaller scale elections and barely feasible for large scale elections. However, the complexity for the authority is quadratic in N and linear in I and J, and this is too large except for smaller elections.

The schemes in [CFSY96], [SK95] and [Oka97] are substantial improvements, with the complexity for the voter being linear in N and J, which is good for large scale elections. The complexity for the authority is linear in N and I, and this is much more feasible for large elections. [Cha81] is also reasonable,

with the complexity for the voter being linear in N and quadratic in J. The complexity for the authority is linear in N, I and J, which is still feasible for large elections.

[CGS97] and [FOO93] are extremely efficient schemes for the voter. The complexity is optimal as it is linear in N. The complexity for the authority is linear in N and I, which is very reasonable.

The generic mix net scheme of [HS00] can be very efficient, when combined with the homomorphic scheme in [CGS97].

9.3.2 Communication complexity

The complexities are given in terms of the number of bits transmitted in the worst case. With current technology, it is assumed that the voter is limited to dial-up modems, with a transfer rate of 56 Kbits per second. An authority is assumed to have a transfer rate of 100 Mbits per second, which is essentially unlimited in this context. Thus only the complexity for the voter is considered, as this is the limiting factor.

Again, the most inefficient protocols are the earlier homomorphic schemes [CF85], [Ben87b] and [BT94]. The complexity for the voter is quadratic in N and linear in J, which is barely acceptable for smaller scale elections but clearly not for large scale elections.

The schemes in [CFSY96], [Cha81], [SK95] and [Oka97] are substantial improvements, with the complexity for the voter being linear in N and J, which is acceptable for large scale elections.

[CGS97] and [FOO93] are extremely efficient schemes for the voter. The complexity is optimal as it is linear in N.

As per the computational complexity, the generic mix net scheme of [HS00] can be very efficient, when combined with the homomorphic scheme in [CGS97].

It is worth mentioning that apart from [CF85] and [FOO93], all of the schemes have distributed authorities. With the notable exception of [CGS97], which requires only a single message to be sent, the others all require the voter to communicate with each authority separately. Hence many connections must be established, and this can be time consuming.

9.3.3 Round complexity

A practical election must have a round complexity of 1, so voters can cast their votes in a single session. Otherwise, the voting process becomes too inconvenient. The schemes of [CFSY96], [CGS97], [SK95], [HS00] and [Oka97] all allow voting in a single session.

9.4 Summary

The only scheme that satisfies all the security requirements is [HS00], and it also satisfies the usability properties. However, the scheme is limited to mul-

tiway voting options. [Oka97] is the only scheme suitable for arbitrary voting options that satisfies most of the security requirements. It can be augmented by threshold schemes for blind signatures, secret sharing and zero knowledge proofs to remedy the lack of robustness with respect to authorities. This scheme is also unique in providing both public and private verifiability, which is highly desirable.

All of the homomorphic schemes are limited to YES/NO voting options and do not satisfy incoercibility. However, for elections where this is acceptable, the clear choice is the very elegant protocol by [CGS97].

Chapter 10

Conclusion

This report has covered all the necessary background for understanding the principles behind online elections and the different approaches taken to formulate schemes. Only the most important election schemes have been described here, and many more exist. These satisfy the election requirements with varying degrees of success. In particular, the requirement of incoercibility, which is essential for democratic elections, is achieved by very few schemes. All the protocols that do satisfy this requirement make the impractical assumption that unconditionally private communication channels are available. It remains to be shown whether incoercibility can be satisfied without such channels, but it is widely believed that this is not possible.

We have seen that constructing practical schemes is a difficult task, and much of the research has been focused on homomorphic schemes for simple YES/NO voting options. Extending these to more complicated voting options, as required by most real elections, is expensive with regards to efficiency.

Currently no scheme exists that satisfies all the security requirements and is also suitable for arbitrary voting options. As such, online elections are a very active area of research. The interest in this field has been further triggered by the 2000 presidential election in the USA, where problems with the traditional process were clearly exposed. However, it is crucial to note that online elections cannot replace most large scale traditional elections in the foreseeable future, as they are intended to provide an alternative means of voting.

Despite the flaws in election protocols, several have already been implemented. Many of these are based on [FOO93], with modifications to solve the problems with robustness, and also on [CGS97]. Most of these are proprietary systems created by private companies who can be hired to conduct online elections. There are also many open source projects at research institutions. Unfortunately, none of these schemes are incoercible.

Several large scale trials of online elections have been held, the most notable of which was in fact the 2000 presidential election in the USA. Military personnel were given the opportunity to vote online. In order to ensure security, voters were provided with a CD-ROM to verify the operating system was safe and to

install a custom browser. Obviously, such extreme measures are impractical, and they detract from the convenience that online voting offers. This demonstrates that in terms of security, the Internet has not yet reached the stage of maturity to be a suitable medium for elections.

There remain many open problems with online elections, and the search for suitable schemes continues. However, in the rush to embrace online voting, the most important issue to keep in mind is that we must not sacrifice democratic principles for convenience. Hence we must proceed with caution when introducing online voting, and the consequences should be carefully considered. Nevertheless, with the major developments in cryptography and the Internet that have had such a profound effect on our lives in recent times, it is hopeful that the exciting possibility of viable secure online elections will soon become a reality.

Appendix A

Notation

The following notation is used throughout this report. Wherever necessary, the terminology is explained when it is first defined.

$a \mid b$	An integer a is a factor or divisor of an integer b.
$A - B$	The set difference of A and B, $\{x \mid x \in A \text{ and } x \notin B\}$. This is sometimes written as $A \setminus B$.
$\gcd(a, b)$	The greatest common divisor of the integers a and b.
ϕ	The Euler totient function.
$\mathbb{Z}$	The set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
$\mathbb{Z}_n$	The set of integers modulo n, $\{0, 1, \ldots, n-1\}$.
$\mathbb{Z}_n^*$	The multiplicative group of $\mathbb{Z}_n$, $\{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$.
$\mathbb{Z}_p$	The prime field $\{0, 1, \ldots, p-1\}$, where p is a prime.
$\mathbb{Z}_p^*$	The multiplicative group of the prime field $\mathbb{Z}_p$, $\{a \in \mathbb{Z}_p \mid \gcd(a, n) = 1\}$.
G_n	A finite group of order n.
G_q	The unique cyclic subgroup of order q in $\mathbb{Z}_p^*$, where p and q are primes such that $q \mid p - 1$.
$\left(\frac{a}{p}\right)$	The Legendre symbol, where a is an integer and p is an odd prime.
$\left(\frac{a}{n}\right)$	The Jacobi symbol, where a is an integer and n is an odd number.
J_n	The set of integers with a Jacobi symbol equal to 1, $\left\{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\right\}$.
Q_n	The set of all quadratic residues modulo n.

$\overline{Q}_n$	The set of all quadratic nonresidues modulo n.
$\tilde{Q}_n$	The set of pseudosquares modulo n, $\tilde{Q}_n = J_n \cap \overline{Q}_n = J_n - Q_n$.
$\{0,1\}^k$	An arbitrary binary string of length k.
$\{0,1\}^*$	An arbitrary binary string of arbitrary length.
$a \| b$	The concatenation of a and b, which are usually binary strings.

Bibliography

[Arr51] Kenneth J. Arrow. *Social Choice and Individual Values*. Number 12 in Cowles Commission for Research in Economics Monographs. John Wiley and Sons, Inc., New York, 1951.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.

[Bea92] Donald Beaver. Foundations of Secure Interactive Computing. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391, Berlin, 1992. Springer-Verlag.

[Ben87a] Josh Cohen Benaloh. Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 213–222, Berlin, 1987. Springer-Verlag.

[Ben87b] Josh Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, September 1987. Technical Report number YALE/DCS/TR561.

[BKK90] Joan F. Boyar, Mark W. Krentel, and Stuart A. Kurtz. A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[Bla79] G. R. Blakley. Safeguarding Cryptographic Keys. In Richard E. Merwin, Jacqueline T. Zanca, and Merlin. Smith, editors, *1979 National Computer Conference*, volume 48 of *AFIPS Conference Proceedings*, pages 313–317, Montvale, NJ, USA, 1979. AFIPS Press.

[Blu82] Manuel Blum. Coin Flipping By Telephone. In Allen Gersho, editor, *Advances in Cryptography - CRYPTO '81*, pages 11–15, Santa Barbara, California, USA, 1982. University of California, Santa Barbara.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Noncryptographic Fault-tolerant Distributed Computations. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 1–10, New York, 1988. ACM Press.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-Free Secret-Ballot Elections (Extended Abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC '94)*, pages 544–553, New York, 1994. ACM Press.

[BY86] Josh Cohen Benaloh and Moti Yung. Distributing the Power of a Government to Enhance the Privacy of Voters (Extended Abstract). In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC '86)*, pages 52–62, New York, 1986. ACM Press.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty Unconditionally Secure Protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 11–19, New York, 1988. ACM Press.

[CF85] Josh Cohen and Michael Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme (Extended Abstract). In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372–382. IEEE Computer Society, 1985. Yale University, Department of Computer Science, Technical Report number YALE/DCS/TR416.

[CFN89] David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '89*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Berlin, 1989. Springer-Verlag.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Authority Secret-Ballot Elections with Linear Work. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83, Berlin, 1996. Springer-Verlag.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Berlin, 1997. Springer-Verlag.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[Cha83] David Chaum. Blind Signatures for Untraceable Payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology - CRYPTO '82*, pages 199–204, New York, USA, 1983. Plenum Publishing.

[Chi95] Lindsay N. Childs. *A Concrete Introduction to Higher Algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1995.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts, 1990.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Berlin, 1985. Springer-Verlag.

[Fis01] Marc Fischlin. *Trapdoor Commitment Schemes and Their Applications*. PhD thesis, Fachbereich Mathematik, Johann Wolfgang Goethe-University, Frankfurt am Main, December 2001.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251, Berlin, 1993. Springer-Verlag.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, pages 186–194, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science Volume 263.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC '82)*, pages 365–377. ACM Press, 1982.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC '85)*, pages 291–304. ACM Press, 1985.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play Any Mental Game — A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*, pages 218–229, New York, 1987. ACM Press.

[HS00] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556, Berlin, 2000. Springer-Verlag.

[IR90] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory*. Number 84 in Graduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1990.

[Ive92] Kenneth R. Iversen. A Cryptographic Scheme for Computerized General Elections. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 405–419, Berlin, 1992. Springer-Verlag.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154, Berlin, 1996. Springer-Verlag.

[Kob94] Neal Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1994.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, USA, 1997.

[Oka96] Tatsuaki Okamoto. An Electronic Voting Scheme. In Nobuyoshi Terashima and Edward Altman, editors, *Advanced IT Tools - Proceedings of the IFIP World Conference on IT Tools 1996*, pages 21–30, London, 1996. IFIP, Chapman & Hall.

[Oka97] Tatsuaki Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35, Berlin, 1997. Springer-Verlag.

[Ped92a] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Berlin, 1992. Springer-Verlag.

[Ped92b] Torben P. Pedersen. A Threshold Cryptosystem Without a Trusted Party (Extended Abstract). In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1992. Springer-Verlag.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sal96] Arto Salomaa. *Public-Key Cryptography.* Texts in Theoretical Computer Science. Springer-Verlag, Berlin, second, enlarged edition, 1996.

[Sch96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley and Sons, Inc., New York, second edition, 1996.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SK95] K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A Practical Solution to the Implementation of a Voting Booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403, Berlin, 1995. Springer-Verlag.

[Sti95] Douglas R. Stinson. *Cryptography: Theory and Practice.* Discrete Mathematics and its Applications. CRC Press, Boca Raton, 1995.