Secure Electronic Voting with Flexible Ballot Structure

by

Riza Aditya

Bachelor of Information Technology *(QUT)* – 2000
Bachelor of Information Technology (Honours) *(QUT)* – 2001

Thesis submitted in accordance with the regulations for
Degree of Doctor of Philosophy

Information Security Institute
Faculty of Information Technology
Queensland University of Technology

November 2005

Keywords

Secure Electronic Voting, Cryptographic Voting Protocols, Secret-ballot Voting Scheme, Receipt-free Voting, Australian Federal Election, Preferential Systems, Batch Theorems, Batch Zero-knowledge Proofs and Verifications, Efficient Voting Protocols, Homomorphic Encryption, Mix-network, Hybrid Scheme.

Abstract

Voting is a fundamental decision making instrument in any consensus-based society. It is employed in various applications from student body elections, reality television shows, shareholder meetings, to national elections. With the motivation of better efficiency, scalability, speed, and lower cost, voting is currently shifting from paper-based to the use of electronic medium. This is while aiming to achieve better security, such that voting result reflects true opinions of the voters.

Our research focuses on the study of cryptographic voting protocols accommodating a flexible ballot structure as a foundation for building a secure electronic voting system with acceptable voting results. In particular, we search for a solution suitable for the preferential voting system employed in the Australian Federal Election.

The outcomes of the research include: improvements and applications of batch proof and verification theorems and techniques, a proposed alternative homomorphic encryption based voting scheme, a proposed Extended Binary Mixing Gate (EBMG) mix-network scheme, a new threshold randomisation technique to achieve receipt-freeness property in voting, and the application of cryptographic voting protocol for preferential voting.

The threats and corresponding requirements for a secure secret-ballot voting scheme are first discussed. There are significant security concerns about the conduct of electronic voting, and it is essential that the voting results reflect the true opinions of the voters - especially in political elections.

We examine and extend batch processing proofs and verifications theorems and proposed applications of the theorems useful for voting. Many instances of similar operations can be processed in a single instance using a batch technique based on one of the batch theorems. As the proofs and verifications provide formal assurances that the voting process is secure, batch processing offers great

efficiency improvements while retaining the security required in a real-world implementation of the protocol.

The two main approaches in cryptographic voting protocols, homomorphic encryption based voting and mix-network based voting, are both studied in this research. An alternative homomorphic voting scheme using multiplicative homomorphism property, and a number of novel mix-network schemes are proposed. It is shown that compared to the mix-network approach, homomorphic encryption schemes are not scalable for straight-forward adaptation of preferential systems.

One important requirement of secret-ballot voting is receipt-freeness. A randomisation technique to achieve receipt-freeness in voting is examined and applied in an efficient and practical voting scheme employing an optimistic mix-network. A more general technique using threshold randomisation is also proposed.

Combination of the primitives, both the homomorphic encryption and mix-network approach, yields a hybrid approach producing a secure and efficient secret-ballot voting scheme accommodating a flexible ballot structure. The resulting solution offers a promising foundation for secure and practical secret-ballot electronic voting accommodating any type of counting system.

Contents

List of Figures

List of Tables

Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed:..... **Date:**.....

Previously Published Material

The following papers have been published or presented, and contain material based on the content of this thesis.

[1] Riza Aditya, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Secure e-voting for preferential elections. In *Electronic Government: Second International Conference, EGOV 2003*, volume 2739 of *Lecture Notes in Computer Science*, pages 246–249. Springer-Verlag, 2003.

[2] Riza Aditya, Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Batch verification for equality of discrete logarithms and threshold decryptions. In *Applied Cryptography and Network Security: Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 494–508. Springer-Verlag, 2004.

[3] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. A secure and efficient mix-network using extended binary mixing gate. In *Cryptographic Algorithms and their Uses – 2004: International Workshop*, pages 57–71. QUT Publications, 2004.

[4] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In *Trust and Privacy in Digital Business: First International Conference, TrustBus 2004*, volume 3184 of *Lecture Notes in Computer Science*, pages 152–161. Springer-Verlag, 2004.

[5] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. Implementation issues in secure e-voting schemes. In *The Fifth Asia-Pacific Industrial Engineering and Management Systems Conference, APIEMS 2004*, pages 36.6.1–36.6.14. QUT Publications, 2004.

[6] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *Progress in Cryptology – INDOCRYPT 2004: 5th International Conference on Cryptology in India*, volume 3348, pages 61–72. Springer-Verlag, 2004.

[7] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. Two models of efficient mixnet-based receipt-free voting using (threshold) re-encryption. *Computer Systems Science and Engineering*, page to appear, 2005.

Acknowledgements

The Ph.D. journey has certainly been an interesting and a valuable experience in life for me. I would like to thank my supervisors, the co-authors, and the reviewers (anonymous or otherwise). Also, to others that have provided input in some way to this research or my life during an enjoyable time at the Information Security Research Centre (ISRC). I owe this thesis to all of them, as if it was not for them, this research and myself would not have come to where we are now.

Firstly, I would like to express my heartfelt gratitude to Colin Boyd, Ed Dawson, and Byoungcheon Lee, for their supervision, guidance, advice, encouragement, understanding, and patience. Especially to Colin Boyd for accepting me as one of his apprentice "protocol wizards", and always making the time for my supervision. To Ed Dawson for starting me on this research, and helping me on my scholarship. To Byoungcheon Lee, for being a mentor, and always reviewing our papers.

To Kun Peng, who always came up with ideas, answered my questions, and allowed me to be co-author in several of our papers. To Kapali Viswanathan, who was involved in the early stage of this research. To Greg Maitland, for his thorough review and valuable advice on this thesis.

Secondly, to Christine Orme, Mark Looi, Ernest Foo, Andrew Clark, Gary Gaskell, Sam Lor, Linda Burnett, Suzie Hlaing, Ricco Lee, Matt Bradford, Minna Yao, Kevin Chen, Nathan Carey, Sherman Chow, Raymond Choo, Jason Smith, Loo Tang Seet, Mehdi Kianiharchegani, Rupinder Gill, Jason Reid, Juanma Gonzales, Yvonne Hitchcock, Praveen Guaravaram, Jaimee Brown, Roland Chua (especially for completing the eVACS honours project), Gerald Pang, Elizabeth Hansford, Sultan Al-Hinai, Selwyn Russell, Lauren May, Les Smith, Mike Milford, Kenneth Wong, Charles Woo, Robert Dawson, Mark Branagan, Steven Panichprecha, Jared Ring, and other people who were/are in (and my other friends outside) the centre (now institute) for their help, friendship, and interesting dis-

cussions.

Also, to the research centre, faculty, university, and the Australian government for their financial support in this research. These include: a Faculty scholarship; a QUT Postgraduate Research Award scholarship; an ISRC top-up scholarship; an ARC Discovery 2002, Grant No: DP0211390 (e-voting); an ARC Discovery 2003, Grant No: DP0345458 (e-auction); and an ARC Linkage International Fellowship 2003, Grant No: LX0346868 (international visiting academic).

Finally, to my parents and partner, for being with me, supporting me, keeping me going, and chasing me everyday to complete my study. And to God, for everything.

Riza Aditya
Brisbane, 8 August 2005

Chapter 1

Introduction

Voting is fundamental to any consensus-based society. It is a basic mechanism to reveal the opinion of a group on a matter that is under consideration. Based on the promise of greater efficiency, better scalability, faster speed, lower cost, and more convenience, voting is currently shifting from manual paper-based processing to automated electronic-based processing. The term "electronic voting" typically refers to the use of some electronic means in voting.

To date, the wide range of application of voting include its use in reality television shows, student body elections, jury duty, shareholder meetings, and the passing of legislation in parliament. Perhaps the most important, influential, publicised, and widespread use of voting is its use in national elections.

Either for political, financial, or personal gain, there are considerable motives to cheat in voting. Especially in national elections, the threats of cheating or tampering are very seriously considered. In such a case, voting results affect the life of a great number of people. Non-acceptance of a voting result can cause confusion, violence, or even civil war. Hence, it is essential that a voting result reflects true opinions of the voters.

There are two equally important parts in ensuring the acceptance of a voting result. One is to apply an appropriate counting method to interpret the votes cast and produce a voting result that reflects the true opinions of the voters. The other is to ensure security in voting to prevent cheating or tampering of the voting result.

The study of different counting methods and their usage appropriateness in

different voting scenarios are in the area of political or social science. Hence, they are beyond the scope of this thesis.

This thesis details cryptographic advancements in the area of electronic voting security. As well as the efficiency or practicality of such schemes, the research presented in this thesis also considers the ability of different cryptographic voting protocols in accommodating available various counting methods, or the ability to accommodate a flexible ballot structure. In particular, we search for a solution suitable for the preferential voting system employed in the Australian Federal Elections.

1.1 Aims and Objectives

The research is aimed at two major goals. The first is to study security-critical cryptographic primitives supporting available voting protocols in the literature. The second is to develop efficient and secure electronic voting schemes supporting a flexible ballot structure.

To achieve the two major goals, the research was organised into the following four objectives:

1. **to increase the fundamental knowledge in designing practical and secure electronic voting schemes**;
   The study contributed to improving the fundamental knowledge for the design of secure electronic voting in particular, and cryptographic protocols in general. This is a fundamental objective, providing the basics for further advancements in the research.

2. **to identify critical functions related to secure and efficient implementation and deployment of such schemes**;
   Investigation into the efficiency and performance of cryptographic primitives and protocols employed in electronic voting schemes is conducted in the research. This objective is crucial as a reference for designing practical schemes to be developed and deployed in a real-world scenario.

3. **to achieve a practical and flexible design framework for electronic voting schemes suitable for the various counting methods, especially for the Australian Federal Elections**;
   The research adds to the number of currently available electronic voting

schemes by proposing a number of new or improved efficient schemes. They are particularly suited for the national elections in Australia, which use preferential voting systems.

4. **to design specific electronic voting schemes with proven security features**.
   Specific schemes addressing specific requirements are designed with proven security. The designs offer solid and secure schemes for particular situations. Again, they are specifically focused at the Australian elections. However, such focus need not necessarily limit the possibility of applying, improving, or extending our schemes to various other schemes and voting applications.

1.2 Main Contributions

The research is conducted to produce outcomes satisfying the aims and objectives described in the previous section. Below lists main contributions of the research presented in this thesis. Table 1.1 illustrates some of the many areas in secure secret-ballot electronic voting. Areas investigated in the research are highlighted in bold.

Observations are made to an electronic counting and voting system (eVACS) trialled in the 2001 and 2004 Australian Federal Election at the Australian Capital Territory. The conclusion is that cryptographic voting protocol is required as a foundation in designing a secure secret-ballot voting system.

Using a batch technique, many instances of similar operations can be processed in a single instance. Extensions and applications of novel batch theorems to batch proofs and verifications of such operations are provided. The applications offer a significant performance gain when applied in any cryptographic protocols requiring the processing of many instances of similar cryptographic operations. The applications of the theorems and their corresponding batch techniques allow such protocols to be more practical and suitable for a real-world scenario. This is particularly suited for cryptographic voting protocols when naturally there are many similar operations required in a voting process. Such operations include: ballot constructions, voter identity verifications, and the combining of individual votes to produce a voting result.

Homomorphic encryptions are researched as one of the two existing basic approaches used to construct a cryptographic voting protocol. An alternative

Table 1.1: Some of the many areas in secure secret-ballot electronic voting.

Design	Development	Deployment
Cryptographic protocol		
homomorphic encryption	certification	legislation
mix-network	standards	policy, procedures
hybrid	testing	training
	voting systems	education
Cryptographic primitives		background checks
zero-knowledge protocol		physical security
batch theorems and techniques		backup
public-key infrastructure		redundancy
blind signature		network security
secret-sharing		
cryptosystems		

scheme is proposed during this research. Research is also performed on mix-networks as the other basic approach used in constructing a cryptographic voting protocol. An improvement on an existing scheme is made, and a new mix-network scheme is produced. The new primitives developed from each of the two approaches are then combined using an existing framework to form a hybrid scheme. A preferential voting system case study is given in each of the approaches.

A randomisation technique to achieve receipt-freeness in secret-ballot voting is applied to an optimistic mix-network. This is to offer a practical scheme providing a receipt-freeness property. A proposal is made for a new technique allowing threshold randomisation. The new technique is more general and is applicable to either the homomorphic encryption or mix-network approach.

Our research results offer a promising foundation for a secure and practical secret-ballot electronic voting accommodating any type of counting systems or ballot structures.

1.3 Thesis Outline

The main contents of this thesis contain material from our previously published papers. All of the publications are results of joint work with different authors. The previously published materials and their corresponding author names are listed at the beginning of this thesis.

This research concentrated on the study of cryptographic voting protocols in the context of secure electronic voting. The protocols are required to accommodate a flexible ballot structure, especially in the context of preferential systems used in Australia. A number of cryptographic primitives used in constructing voting protocols are examined and improved. These include the application of batch proof and verification, a threshold re-encryption technique, and techniques for verifying correct shuffling in a mix-network. Several voting protocols are surveyed, and the new primitives are combined in an existing framework to accommodate a flexible ballot structure.

The next chapter, Chapter 2, provides background on: types of available counting systems focusing on the Australian preferential systems, a review of an electronic voting system, the importance of security in producing an acceptable voting result, and an overview of cryptographic voting protocols. This chapter contains material that has been previously published in [ALBD04b].

In Chapter 3, existing batch theorems and techniques are extended to batch zero-knowledge proofs and verifications of two common operations in cryptographic voting protocols. They are: to verify valid encryptions or valid re-encryptions, and to verify valid decryptions or valid threshold decryptions. Novel applications of the theorems are provided as batch techniques using both ElGamal and Paillier as the underlying cryptosystems. Applications of these techniques offer great performance increase in schemes employing many instances of the operations. Specifically, the techniques are developed with applications to cryptographic voting protocols in mind. This chapter is an extension to the previously published paper [APB+04].

A more detailed study into the homomorphic encryption approach used in voting schemes is provided in Chapter 4. As most homomorphic encryption voting schemes in the literature use an additive homomorphism property, an alternative scheme exploiting a multiplicative homomorphism property is proposed. Afterward, a case study on homomorphic encryption voting schemes and their suitability for a preferential voting system is presented. While the homomorphic encryption approach allows for efficient tallying, it is shown that this approach is not practical to accommodate a straight-forward adaptation of a preferential system. Material previously published in [PAB+04a, ABDV03] is included in this chapter.

Chapter 5 details the other approach in cryptographic voting protocol: the

mix-network approach. This approach mimics the use of ballot boxes in traditional paper-based voting. An improvement of an existing shuffling scheme is proposed by using our batch theorems from Chapter 3. Combining group shuffling of ciphertexts and batch verification of valid shuffles, an Extended Binary Mixing Gate (EBMG) mix-network is proposed. A case study on mix-network voting schemes and their suitability for a preferential voting system is also presented. As tallying using this approach is not elegant compared to the homomorphic encryption approach, the mix-network approach is suitable for any type of counting system. Material from this chapter has been previously published in [PAB+04b, ALBD04a, ABDV03].

Chapter 6 discusses the requirement of receipt-freeness in secret-ballot voting. An existing randomisation technique is applied to an optimistic mix-network to achieve an efficient and receipt-free voting scheme. A threshold-randomisation technique that can be used in both approaches is proposed. This chapter contains material previously published in [ALBD04a, ALBD05].

Primitives from previous chapters are combined in Chapter 7. The multiplicative homomorphic voting and EBMG mix-network are employed in an existing hybrid framework (named vector-ballot). This allows the application of preferential voting by using a number of set preferences, as well as allowing the choosing of other non-set preferences simultaneously. A case study for the Australian Senate elections is presented using this hybrid approach. The resulting scheme offers a promising foundation for secure and practical secret-ballot electronic voting accommodating a flexible ballot structure.

Chapter 8 provides a thesis summary, along with possible future research directions. Main contributions of the research are again highlighted in the summary. A number of possible extensions or improvements continuing from the research presented in this thesis are listed. A brief final remark is provided at the end.

There are three appendices in this thesis. Appendix A recalls a well-known secret-sharing technique. Appendix B details the cryptosystems of ElGamal and Paillier, and their corresponding threshold versions. Appendix C presents the concept of zero-knowledge proof and verification, including its properties, a number of related standard protocols, and a brief description on how to construct a combination of proofs in one protocol run.

Chapter 2

Voting, Electronic Voting, and Security

Design, development, and deployment of secure electronic voting systems require expertise in areas from politics and technology, to security and cryptography. While there are a large number of issues in each of these areas, this thesis specifically concerns the study of electronic voting security using cryptographic means.

In this chapter, a typical voting procedure is described. Categorisation of the different types of counting systems is provided. The Australian Federal Election using preferential systems is explained. Electronic voting is discussed, and an electronic voting and counting system (eVACS) trialled in the Australian Capital Territory (ACT) is analysed. Security threats in electronic voting are further discussed, and security requirements for electronic voting are listed. An introduction to cryptographic voting protocol and the different approaches is presented.

This chapter provides background information for the thesis. Material from this chapter has been previously published in [ALBD04b].

2.1 A Typical Voting Procedure

The process of voting follows a standard procedure. Illustrated in Figure 2.1, a voting scenario typically consists of four phases as outlined below.

1. Set-up phase.

   During this phase, voting parameters are initialised. The parameters in-

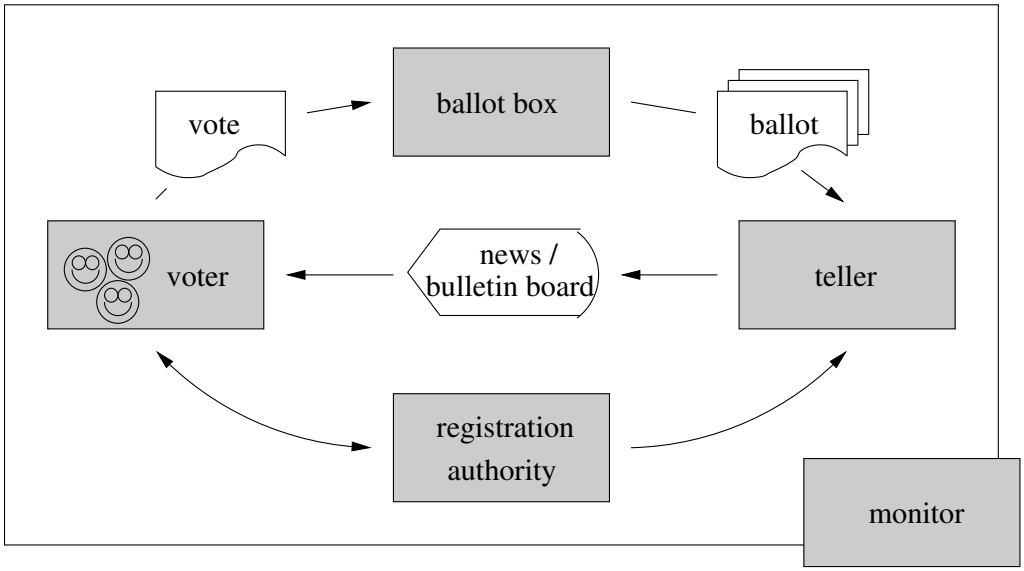

Figure 2.1: Registration, voting, and tally phase in a typical voting scenario.

clude eligibility criteria for candidates, voters, and authority; voting procedures; ballot validity rules; and counting rules. Eligible candidates register themselves to the authority, and the registration and tally authorities are selected. Afterwards, the voting parameters, the candidates, and the authorities are made public, such that they can be known and verified.

2. Registration phase.
   Voters are required to register to the registration authorities during this phase. Their eligibility is determined by the criteria set in the previous phase, and ineligible voters are not allowed to register and participate in voting. The list of authorised voters is published for public verification afterwards.

3. Voting phase.
   In this period, registered voters are allowed to cast their votes as follows:

   (a) Voter authentication: each voter is authenticated according to the list of registered voters from the registration phase, and those not found in the list are not allowed to participate in this stage.

   (b) Voter registration: each of the authenticated voters receives an empty ballot, and registers a vote in the ballot inside a physically private and secure location to avoid coercion/intimidation.

(c) Ballot casting: the ballot is anonymised, such that the "voter-vote" relationship is kept secret; in paper-based voting, this is achieved by using a sealed ballot-box where ballots are anonymised inside the ballot-box.

4. Tally phase.
   In this last phase, all ballots from the voting phase are processed to reveal the voting result as below.

   (a) Ballot collection: the ballots are collected; in paper-based voting, ballots are obtained after the sealed ballot-boxes are opened by tally authorities.

   (b) Ballot verification: each of the ballots is verified to be valid or invalid according to the rules set during the set-up phase, where invalid ballots are not included for counting.

   (c) Vote counting: valid ballots are counted as per the counting rules; the results from each polling location are aggregated, and the voting result is produced and made public.

2.2 Counting Systems

The choice of a counting system employed in a particular voting scenario contributes to the perceived fairness of the voting result. This is an important foundation for the acceptance of election results, or accepting a voting result in any of the voting scenarios.

Also known as "electoral systems", there are various counting systems employed for elections in different countries. Each of these systems offer its own advantages and disadvantages. Some systems might be considered to produce a fairer result than others.

In the "Handbook of Electoral System Design" [RR97], these systems are categorised as below.

- Plurality-majority systems
  In plurality-based systems, a candidate with the highest number of votes wins (no threshold of votes). On the other hand, a candidate can only win with a majority of votes (above a certain threshold) in majority-based

systems. Countries employing this system include Australia, the United Kingdom, Canada, and India.

- Proportional representation systems
  Voting results under this category are proportional to the number of votes received. For example, if a major party wins 40% of the total votes in an election, the party will be allocated approximately 40% of the seats. Proportionally, a minor party with 10% of the total votes will also gain 10% of the parliamentary seats. This counting system reduces the disparity between the total number of votes and the total number of parliamentary seats available. Countries employing this system include South Africa, Finland, and Ireland.

- Semi-proportional systems
  This type of system translates votes cast into seats won in a way that falls somewhere between the proportionality of proportional representation systems, and the majority-based systems of plurality-majority systems. Each voter may have more than one vote allowed for the candidates, e.g. as many votes as there are seats available. Countries employing this system include Japan, Jordan, and Vanuatu.

The choice of using a particular counting system in a particular scenario depends on the culture, policy or other considerations. Further discussions for which counting system is more appropriate for a particular scenario belongs to social study or political science. They are outside the scope of this research.

Accommodating a flexible ballot structure, this research mainly concerns the use of preferential systems in the Australian Federal Election scenario. The preferential system is a majority-based systems of plurality-majority systems. The next section offers more descriptions on preferential systems, specifically those used for the Australian Federal Elections.

2.3 The Australian Federal Elections

The Australian parliament is composed of two houses. One is the lower house or the House of Representatives, and the other is the upper house or the Senate. While further discussions on both houses are outside the scope of this thesis, we provide descriptions of preferential voting systems and their associated counting

mechanisms employed in elections for both houses. This is to illustrate a voting scenario requiring a flexible ballot structure and set the parameters used for cryptographic voting protocols case study in Chapter 4, Chapter 5, and Chapter 7.

Totalling more than 50 registered political parties in Australia, there are three major (have many seats in parliament), five medium (have seat in parliament), and eight minor parties.

2.3.1 Preferential Systems

In preferential voting, a voter is required to provide a preference (or to rank) each participating candidate. Majority is defined to be more than half of the total number of votes. A voter communicates a list containing the candidate names in a ballot according to his/her own preference. The sequence of choices in the ballot is very important in the counting of votes to produce a voting result.

This election system uses a single round of voting, and potentially multiple rounds of counting. If no candidate receives a majority of votes, the candidate with the lowest first preference vote is eliminated. Votes for the eliminated candidate are redistributed to the remaining candidate according to the second preference. Repeatedly, more candidates are eliminated until one candidate reaches a majority.

The preferential system is employed in the Australian Federal Elections since it is regarded as a fairer system compared to the other systems. Valid preferential vote rules differ for each state, e.g. valid by specifying at least the first preference, valid by specifying 90% of of the preferences, or only valid by specifying a complete list (from the first preference to the last preference) of preferences. A voting result produced in the preferential system reflects the will of the majority of voters.

2.3.2 Elections for The House of Representatives

Elections for the House of Representatives are designed such that only the candidate with a majority of votes in an electoral division is elected to represent the division in the House of Representatives. Using preferential voting system, the House of Representatives truly represents (a majority of) the voters.

A counting example is presented using Table 2.1. Let there be 22000 voters in this voting scenario. The counting of first preference votes indicates that candidate A receives the most votes (indicated in the second column). However,

Table 2.1: A counting example in a preferential voting system.

	First preference votes	Distribution of votes for candidate B	Total
Candidate A	10000	500	10500
Candidate B	4000	-	-
Candidate C	8000	3500	11500
Total	22000	4000	22000

since the number of votes received by candidate A is below the majority threshold, or below $\frac{22000}{2} + 1 = 11001$ votes, another round of counting is performed. Since candidate B receives the least number of votes for the first preference, votes for candidate B are redistributed to candidate A and candidate C according to the second preference (shown in the third column) in the second round of counting. The second preference distribution for the 4000 voters choosing candidate B as their first preference is 500 votes for candidate A, and 3500 for candidate C. After the votes are redistributed for the second round of counting, candidate C wins by receiving 11500 votes (shown in the fourth column, as additions of the second column and the third column), over the majority threshold of 11001 votes.

A voting result for the election is produced by using an alternative vote counting method as explained in the previous paragraph. Voters are commonly provided with a "how-to-vote" card indicating how to arrange the ordering of candidates preference according to a specific party.

Using a green ballot paper, there is a maximum of 22 candidates, and 100000 voters recorded in a district. The average number of candidates per district is twelve. A voter cast a vote by either following the "how-to-card" of a particular party, or by specifying his/her own preference.

2.3.3 Senate Elections

Senate elections in Australia also employ preferential voting systems. Using this system, the Senate also represents the voters (in majority) as in the election for the House of Representatives.

However, the counting method used in Senate elections is single-transferable-vote counting with proportional representation. Several candidates with a number of votes equal to or exceeding a required proportion of votes quota are elected

using this particular counting method. The number of elected candidates are pre-determined. The quota is calculated as the total number of votes divided by one more than the number of candidates to be elected, plus one (this is known as the Hare-Clark quota). Candidates receiving votes more than the quota (a proportion of the votes) have the additional votes distributed according to the preferences in those votes. Aside from the calculation of quota, the rest of the vote counting rules follows the counting of votes using alternative vote counting. That is, the candidate with the least number of votes is eliminated and their voters distributed to the other candidates according to the preference in the votes. After checking the quota and distributing surplus votes (if any), this process is repeated until all available positions are filled.

In the case of Australian Senate elections, the pre-determined number of elected candidates is originally determined by the Australian Constitution. As provided by the parliament, currently there are twelve Senators from each of the six states, and two from each of the Northern Territory and the Australian Capital Territory.

Voting on a white ballot paper, there is a maximum of about 70 candidates and about 1000000 to 4000000 voters in a particular state/territory. The average number of candidates per state/territory is about 60. A voter casts a vote by either choosing a party's preference (pre-set preferences), or over-the-line voting; or by providing a rank for each of the candidates him/herself, or below-the-line voting.

A well-known statistic for Senate elections is that over 95% of the voters cast their votes using over-the-line voting (refer to the article "How Senate Voting Works", available online from `http://www.abc.net.au/elections/federal/2004/guide/senatevotingsystem.htm`, last accessed 5 August 2005). The case study using our hybrid scheme in Chapter 7 exploits this statistic.

2.4 Electronic Voting

Compared to its traditional paper-based counterpart, electronic voting is considered to have many greater potential benefits. These benefits include: better accuracy by eliminating the negative factor of human error, better coverage for remote locations, increased speed for tally computation, lower operational cost through automated means, and the convenience of voting from any location (e.g.

using mobile devices).

Whether or not electronic voting is a necessary replacement for the traditional paper-based method, it is irrefutable that the conduct of voting has been shifting to the use of electronic medium. To date, electronic databases are used to record voter information, computers are used to count the votes and produce voting results, mobile devices are used for voting in interactive television shows, and electronic voting machines have been used in some national elections.

Generally, the term "electronic voting" refers to the definition, collection, and dissemination of people's opinions with the help of some machinery that is more or less computer supported. Despite the transition from traditional paper-based systems to electronic medium, the purpose and requirements for voting remain. Voting is a decision making mechanism in a consensus-based society, and security is indeed an essential part of voting.

Based on the locality, electronic voting can be categorised into two types.

- Polling-site electronic voting
  In the polling-site electronic voting scenario, the casting of votes can only be conducted inside a voting booth at a polling site. This is similar to the current paper-based voting systems. Typically, voting booths at the site contain electronic voting terminals. Voters are authenticated and authorised at the site before allowed access to the voting booths. Votes are cast using the terminal inside the voting booths. At the end of the voting period, the votes are to be communicated to a central server for tallying. Examples of this category include eVACS[1] (an Electronic Voting and Counting System) used in Australia, Diebold systems[2] used in the United States (US), and EVM[3] (Electronic Voting Machines) used in India.

- Remote electronic voting
  In this scenario, voters cast their vote from the convenience of their own location through a communication network. This includes the use of a private (closed) network, mobile network, or even the Internet. For authentication, the credential of a voter is arranged prior to the voting period through the use of a password or some type of authentication token, or even only

[1] `http://www.softimp.com.au/index.php?id=evacs`, last accessed on 30 June 2005.
[2] `http://www.diebold.com/dieboldes`, last accessed on 30 June 2005.
[3] `http://www.bel-india.com/Website/StaticAsp/prod_niche4.htm`, last accessed on 30 June 2005.

through the voter's telephone number or an IP (Internet Protocol) address. Examples of this category include telephone polling, voting using mobile text-messages, and Internet voting. Note that the security level of voting in this category is considered to be low. In a remote electronic voting scenario, ensuring the security of voting terminals - or the public network where votes are communicated - is very cumbersome for a large number of voters.

Security is one of the main topics in this thesis, and is also essential for the election of a central government. Thus, we only consider a polling-site electronic voting scenario. It is inefficient and impractical to ensure the physical security of individual voting terminals and communication networks in the scenario of remote electronic voting, or voting from home. In the case of Internet voting, the underlying infrastructure of the communication network is administered by various different entities with their own interests. Because of this, electronic voting over the Internet is inherently insecure. This point is also noted by Jefferson *et al.* in a report [JRSW04] analysing the security of an experimental Internet-based voting system to be used by US citizens to vote from overseas. However, we believe that the results of our research can be extended for a remote electronic voting scenario.

More discussions on security threats and requirements in voting, particularly in electronic voting, are provided in Section 2.6 and Section 2.7 of this chapter.

Where security is not of paramount importance, other voting scenarios may choose to employ a less restrictive environment. Such scenarios include voting in an interactive television program, or polling of a favourite celebrity on a website over the Internet.

Electronic voting has been used in national elections in a number of countries including the United States, India, Brazil, and Venezuela. Trials were conducted in the United Kingdom, Australia, and some European countries. Considerable coverage on electronic voting in the popular media highlights security issues in electronic voting. Note that some of the security problems in electronic voting can also be found in traditional paper-based voting, and cheating has always been a threat in voting.

In national elections, security threats in voting are considered to be serious as voting results affect the entire nation. Chapter 2 of the book by Harris [Har03] offers a compilation of examples on electronic voting anomalies in the United States. It contains many newspaper articles reporting that electronic voting ma-

chines were not operating properly. Examples include voting machines that reported more votes cast than the number of voters, counted less votes than ones cast, cast votes for a different candidate than intended by the voter, and swapped votes cast for the candidates during the counting stage. Whether they are deliberate mistakes or not, the security of electronic voting machines used for national elections must be ensured. They must achieve a much higher standard of security and tolerate less errors.

Furthermore, the paper by Kohno *et al.* [KSRW04] offers a critical review of a specific voting machine (AccuVote-TS by Diebold, Inc) used for elections in some states in the United States. The paper highlights the lack of security mechanisms implemented, and even highlights the lack of high quality on the software developed. Example threats include the possibility of voters to produce their own smartcards (authentication token) to cast multiple votes, view partial results, and terminate the voting period early; non-existent cryptographic mechanisms to protect the communication of the voting machines; and deficiency in the quality of the software developed.

The next section briefly analyse the security of an Electronic Voting And Counting System (eVACS) trialled in the 2001 and 2004 federal elections in the Australian Capital Territory (ACT).

2.5 An Analysis of eVACS

eVACS stands for Electronic Voting and Counting System. It was commercially developed for the federal elections in the Australian Capital Territory (ACT). The source-code is publicly available from the ACT electoral commission `http://www.elections.act.gov.au/Electvote.html`, last accessed on 29 June 2005. However, simulation of the actual voting environment and the actual voting process are not possible since the configuration data was not made public. Using email correspondence, the ACT electoral commission declined to provide the complete source code, documentations, and configuration data. Hence, we only made informal security analysis based on the available source code. The 2004 source code was chosen to be analysed since it is more recent than the 2001 source code.

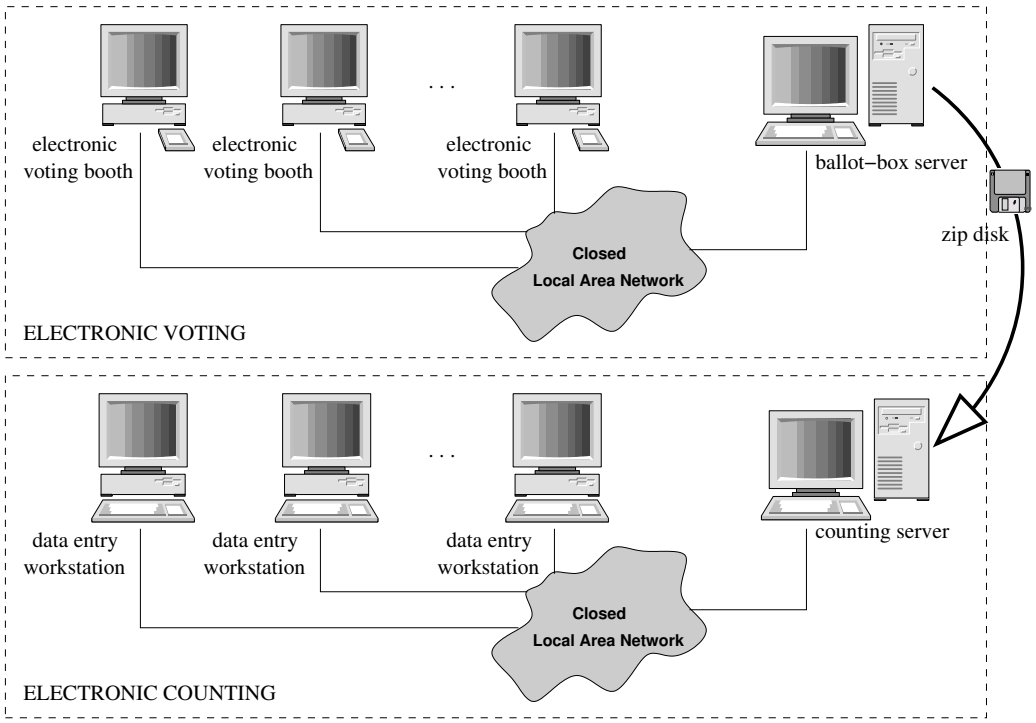

Figure 2.2: A high-level system diagram of eVACS.

2.5.1 System Overview

eVACS is written in C, and is built on a Debian Linux platform. The 2004 source code has 86 .h files with 3812 lines in total, and 171 .c files totalling 33242 lines of code.

Deployed in a polling station, the voting system has four main components consisting of electronic voting booths, a ballot-box server, data entry workstations, and a counting and configuration server. Figure 2.2 illustrates eVACS with its main components based on the information available from the ACT electoral commission website.

The *electronic voting* part of eVACS consists of a number of electronic voting booths connected to a ballot-box server through a closed local area network. The *electronic counting* part of eVACS consists of a number of data entry workstations connected to a counting server.

Voters cast their votes electronically through the electronic voting booths. Votes cast are communicated to the ballot-box server. At the end of the voting period, the entire votes stored on the ballot-box server are transported to the counting server by using zip disks and not by using network connection. Also,

votes cast manually using paper ballots are translated to their electronic votes equivalent using the data entry workstations by the authorities at the end of the voting period. The counting server receives both the electronic votes and the translated votes, and use the appropriate counting method to produce a voting result. The counting server is also used to perform other administrative functions on the system, e.g. generating barcodes for authentication to the electronic voting booths. The barcode is required to gain access to an electronic voting booth, and confirm the vote before being cast to the ballot-box server.

2.5.2 The Voting Process

Prior to the election, all configuration data is set up on the counting (configuration) server. The configuration is then transferred to the ballot-box server. Configuration data include: candidate names, polling station identity, and a list of barcodes.

During the voting period, voters are authenticated as per the traditional paper-based voting, and asked whether they wish to vote electronically or use the traditional paper-based method. A voter choosing to use the traditional paper-based method proceeds by being given a ballot paper, casting the vote on the ballot paper, and placing the ballot paper in a ballot-box.

On the other hand, a barcode is chosen at random and is given to the voter choosing to use eVACS. Voter authorisation on the electronic voting booth computer is by using the barcode. The electronic voting booth computer communicates the barcode to the ballot-box server for validation and to inform that the voting process is initiating. Upon validation of an invalid barcode, the ballot-box server returns an error message to the voting booth computer. Otherwise, the ballot-box server returns the equivalent of a ballot-paper containing the names of candidates to the voting booth computer.

The voter may select the candidates in a particular preference ordering, and restart or complete their selection afterwards. The selection is displayed on the screen for confirmation, and the voter is allowed to change or confirm their selection. The voting booth computer returns a warning given invalid selection or informal vote, however casting invalid or informal vote is allowed. The voter confirms the selection by using the barcode, and both the vote and a log of key sequence pressed are then communicated to the ballot-box server.

The ballot-box server checks that the same barcode is used to initiate the

voting process, checks that the keystroke recorded reflects the vote cast, marks the barcode as being used, and stores the vote and barcode in two different database tables in two different hard drives. Otherwise, the ballot-box server returns an error message to the voting booth computer.

At the end of the voting period, paper-ballots are translated into their equivalent electronic form using the data entry workstations by voting authorities, and these are communicated to the counting server. Votes collected on the ballot-box server are then communicated to the counting server via a zip disk. The counting server counts the votes, and produces a voting result.

The system uses an Apache web server to store and communicate (using HTTP) the configuration data, and uses a PostgreSQL server to store the votes cast.

2.5.3 Observations

Government review of the 2004 system is not available as of the writing of this thesis. The 2001 review is available, but it does not provide much detail regarding testing and auditing of the system. The findings of BMM International, a software auditing firm contracted to audit the software code, were that eVACS code "appeared to neither gain nor lose votes, appeared to faithfully implement the Hare-Clark algorithm or vote counting provided to BMM by the Commission; and was written in a consistent, structured and maintainable style".

Our own observations on the source code reveal that although the code conform to a coding standard, it relies on the physical security of the closed local area network for its communication, relies on a simple vote validity verification based on recorded keystroke, and does not rely on the appropriate cryptographic primitives to generate barcodes.

While the security level might be sufficient for a small-scale deployment scenario, appropriate security mechanisms and cryptographic primitives are required for a secure deployment in a larger scale. This is because it is more difficult to ensure physical security on each of the computer or local area network in a large-scale deployment scenario.

Example threats include gaining unauthorised access to the closed network by using the network cable of a voting booth computer, generation of fraudulent barcodes, and a corrupt authority manipulating the voting result.

By gaining access to the closed network, a malicious entity can observe and

manipulate network traffic, or even gain unauthorised access to the ballot-box server. Since only plain HTTP protocol and available security mechanisms are not properly implemented on the server (e.g. authenticated database connection), the malicious entity can perform the following actions:

- observe barcodes communicated, and produce fraudulent barcodes,
- disrupt the voting process by pretending to be the ballot-box server and returning error messages to the voting booth computers,
- disrupt the voting process by performing a Denial of Service attack against the ballot-box server,
- modify recorded keystrokes and votes communicated from the voting booth computers to the ballot-box server,
- manipulate the configuration data on the ballot-box server, and
- manipulate the barcode or vote database on the ballot-box server.

Further threats also exist should an attacker gain access to either a data entry workstation, the counting server, or the electronic counting network. It is easier to manipulate the voting result having access to the counting machine.

By having a number of fraudulently generated barcodes, a voter can cast the same vote multiple times. This can affect the voting result, benefiting the malicious voter. Also, it is straight-forward that a corrupt authority can easily manipulate the system to his/her advantage.

In conclusion, the security of eVACS is adequate with the assumptions of small-scale deployment, closed network, and physical security. However, it is not scalable for a large-scale deployment since security mechanisms implemented are not sufficient.

Proper implementation of cryptographic primitives offer stronger logical audit trail compared to typical audit trail mechanism using log files. While log files themselves may be corrupted or compromised, it is substantially more difficult to compromise the confidentiality and integrity of data based on a hard mathematical problem (e.g. a discrete log problem). This is discussed further in Section 2.8.

2.6 Security Threats

There are various security threats in any voting scenario. Typical threats include coercion/intimidation of a voter to cast a vote in a particular manner, disruption to the voting process, to tampering of votes cast or tampering of the voting result itself. A number of specific possible threats using a particular electronic voting system (eVACS) are listed in the previous section.

Based on the separation of duties, below is a list of entities and examples of possible security threats in current secret-ballot electronic voting.

- developers/vendors: as investigated in [Har03], developers/vendors have their own interest which may motivate them to deliberately corrupt the voting machine/software. Voting machines may be compromised in such a way as to produce a particular result which benefits the developers/vendors. An example of this type of threat is to modify the counting program to discard a particular type of vote, or to produce a fixed result regardless of votes cast. A number of suggested mechanisms to counter this type of threat include quality assurance, secure programming principles, and independent testing and certification.

- authorities: similar to the above threat, the authorities may also tamper with the voting machine or software for their own benefit. Note that this is also a problem with the paper-based voting. Election authorities may selectively disable voting terminals in a particular district, or corrupt the voting result. The ease of tampering with the result is inversely proportional to the trust level given. An authority with a high level of trust can more easily manipulate the voting result compared to an authority given a low level of trust. The level of trust should be chosen based on reputation and security clearance of an authority. Furthermore, a set of regulations (e.g. codes of conduct, procedures, fines, and punishments) and auditing mechanisms must be enforced to prevent such a security breach.

- voters: vote buying/selling (trading), or even coercion and intimidation are threats to producing a voting result reflecting the true opinion of voters. Private voting was enforced to eliminate this problem. Voting in national elections is now conducted privately inside a polling booth. A dishonest voter may also try to cast more than the one allowed vote (double voting)

to affect the voting result in favour of the dishonest voter, undermining the votes of other honest voters. In traditional paper-based voting, official ballots printed and distributed by the government are enforced to alleviate these problems. This is known as the Australian ballot, or secret-ballot, since it was first used in the states of Victoria and South Australia in 1856 to enforce compulsory secrecy and prevent double voting.

- external: external entities might disrupt or manipulate voting result for their own benefit whether it is personal, financial, or political (e.g. a terrorist organisation). Examples of this type of threat include physically blocking polling sites or coercion/intimidation of voters not to cast their votes during the voting period. Other possible threats also include compromising the voting machines, tally server, or performing a denial of service attack on the system such that voting is somehow disrupted or the voting result is manipulated. Currently, precautions to prevent such threats include placing security guards in polling places to ensure the physical security of the system (and the polling site), and using secure private networks as the communication channel.

- equipment failures/glitches and unforeseen events: accidents or some unforeseen events may occur and disrupt the voting process. Suggestions to minimise problems from this threat include sealing the voting machines, or making them tamper-resistant, and employing some redundancy for robustness.

Compared to current online commercial transactions, a different and higher security level is required for an electronic system since voting results affect a great deal of people's lives. The use of electronic voting also allows the attack sophistication to increase with the use of computer automation. A set of security requirements and proper design and development are required to produce a secure electronic voting system, in which the results are publicly acceptable.

2.7 Security Requirements

Security in voting is one important factor to ensure acceptance of voting result. In order to mitigate the threats discussed in the previous section, there are a number of security requirements that need to be satisfied in a secure secret-ballot voting

scenario. While the complexity of the requirements differ according to the voting application and its corresponding risk assessment, listed below are the important requirements to provide security in secret-ballot voting commonly found in the literature.

- Privacy: An essential requirement for voting, the relationship of voter identity and the corresponding vote cast are to be kept private only to that particular voter, such that voters are able to express their true opinions without being coerced or intimidated. In the case of voting, anonymity also depends on the total number of voters and variations of votes cast (i.e. hiding in a crowd; refer to Definition 1 in Section 2.8).

- Receipt-freeness: A stronger notion of the privacy requirement, voters must not be able to obtain nor construct a receipt which can prove the content of their vote to a third party. This is to prevent vote buying and/or selling, such that voters are not used as proxies to cast votes. This concept is further discussed in Chapter 6.

- Accuracy (Correctness): As a basic property, voting results must be produced from the correct tally of individual votes, i.e. only valid votes are to be counted, and invalid votes are to be discarded.

- Fairness: Each candidate or choice in voting must be given an equal chance. This is achieved by ensuring that no partial tally is to be revealed before the end of the voting period, so as not to advantage or disadvantage a particular candidate or choice.

- Eligibility: As a more specific requirement under fairness, only eligible and rightful voters can cast a vote to prevent fraudulent votes from being counted.

- Non-reusability: Also a more specific requirement under fairness, an eligible voter can only cast his/her vote once to ensure that each voter has equal influence in the voting result.

- Robustness: Voting systems need to be able to tolerate certain faulty conditions and manage some disruptions.

- Verifiability: Correctness of the voting process must at least be "publicly verifiable" by voting participants. A stronger notion of verifiability is "universal verifiability" where everyone (including observers and outside parties - not only those participating in voting) is able to verify that the voting was conducted correctly and that the result is not corrupted.

Especially in the United States, the media has highlighted the fact that current electronic voting machines lack a proper auditing mechanism. Note that the electronic voting machines scrutinised did not use a proper cryptographic voting protocol as a foundation to their electronic voting system. Compared to paper-based voting, it seems to be quite problematic to have an auditing mechanism and satisfy receipt-freeness while maintaining privacy at the same time.

The use of paper ballots provides a straight-forward mechanism for a physical audit-trail in voting results verification by using a recount of the paper ballots. However, it is not as straight-forward to verify the integrity of electronic data should some manipulation of the voting result be suspected in electronic voting.

For this reason, cryptography is required as a foundation in designing electronic voting systems. This is discussed in the next section.

2.8 Cryptographic Voting Protocols

Having highlighted the lack of security provided by a straight-forward implementation of vote collection and counting in software (most commercially available electronic voting products), this research is focusing on providing a solid foundation of secure secret-ballot electronic voting systems. This is by enforcing security mechanisms to achieve the requirements by using cryptographic voting protocols in designing such voting systems.

In electronic voting, cryptography offers a mechanism for a verifiable logical audit trail as compared to the traditional paper-based voting. Cryptography offers verifiability through mathematical proofs that the confidentiality and integrity of the voting result is preserved.

While accuracy is essential, cryptographic voting protocols in the literature are developed based on the privacy requirement. A ballot initially contains an encrypted vote originated from a particular voter. To satisfy the accuracy requirement, voting result must reflect all the votes from each individual ballot. To satisfy the privacy requirement, the relationship of the identity of a voter

Table 2.2: Framework categorisation of cryptographic voting protocols.

Framework	Voter identity	Corresponding vote
Naive	in clear	in clear
Homomorphic encryption	in clear	**hidden**: encrypted
Mix-network	**hidden**: anonymised	in clear
Hybrid	**hidden** (some)	**hidden** (others)

and its corresponding vote ("voter-vote" relationship) must remain private to the particular voter.

Definition 1. *After the end of the voting period, if every vote is only known to lie in the vote space (containing all the possible choices), then we say that a* ***complete vote privacy*** *is achieved. Otherwise, if every vote is only known to be among a large number of published votes - whose number is much larger than the number of all possible choices - we say that* ***strong vote privacy*** *is achieved.*

In cryptographic voting protocols, correct voting results are produced while voter-vote relationships are kept secret. Voting results are produced by decrypting the combination of valid ballots cast; or by tallying (in the usual manner without any cryptographic means) the anonymised individual votes cast in the ballots. In other words, the privacy requirement is satisfied by either maintaining the confidentiality of the individual vote cast, or by maintaining the confidentiality of each voter's identity.

The first method is achieved by using a homomorphic encryption function. Homomorphic encryption and related voting protocols are further discussed in Chapter 4. The second method is achieved by simulating the use of ballot-boxes using mix-networks. Mix-networks and related voting protocols are further discussed in Chapter 5. Cryptographic voting protocols in the literature are typically classified into these two main frameworks. This is summarised in Table 2.2.

Homomorphic Encryption. Pioneered by Benaloh [Ben96], privacy is satisfied by employing a confidentiality service for the individual vote cast. Voting result is obtained by decrypting the combination of the encrypted votes.

Mix-networks. Pioneered by Chaum [Cha81], privacy is provided by anonymising ballots cast. As it receives encrypted votes as inputs, the mix-network

outputs shuffled votes in plaintext format. A mix-network is also known as an anonymous channel.

To satisfy other security requirements, cryptographic voting protocols employ other cryptographic primitives aside from using either homomorphic encryption or mix-networks. Some of the primitives are listed below.

- Public Key Infrastructure: an infrastructure for entity identification, and public-private key certification. This is required to satisfy the eligibility requirement.

- Digital signatures: prove authenticity and integrity of messages. This is required to satisfy the accuracy requirement.

- Blind signatures: this technique is comparable to the use of carbon copy in physical paper-based transaction. The primitive is typically used to authorise the ballot from an authenticated eligible voter (privacy and eligibility requirements). Privacy is achieved using pseudonyms. However, voting protocols employing blind signatures often also employ mix-networks for stronger anonymity.

- Threshold cryptography: distributing trust to a quorum of authorities, threshold cryptography is employed to prevent a number of corrupt authorities below the threshold value being able to somehow tamper with voting results. Based on secret-sharing (Appendix A), techniques include threshold re-encryption (Chapter 6) to provide stronger re-encryption service on ballots and satisfy the receipt-freeness requirements; and threshold decryption to ensure individual ballot remains unopened (privacy and robustness requirements).

- Zero-knowledge proof techniques: an important technique to verify correct operations of other cryptographic primitives, a zero-knowledge proof technique convinces a verifier with a certain (high) probability that the prover indeed holds the knowledge of a particular value without revealing it to the verifier (verifiability and privacy requirement). Appendix C contains more information on zero-knowledge protocols.

The Australian preferential system is used as an example of a flexible ballot structure in voting. The ballot for preferential voting systems is quite complex

as it contains a preference of the available choices. Thus the possibility of all available preferences is of factorial size. This presents an interesting challenge when providing a simple design to enable real-world electronic voting for such a system in a secure and efficient manner.

Chapter 4 and Chapter 5 presents our work in the homomorphic encryption and mix-network area respectively for voting. A case-study using straight-forward adaptation of preferential systems is presented for both approaches. Using a set number of preferences prior to a voting period, both frameworks can be combined. This offers benefits from both frameworks, and is presented in Chapter 7.

Basic primitives such as secret sharing, ElGamal and Paillier cryptosystems, and the concept of zero-knowledge proof techniques are described in Appendix A, Appendix B, and Appendix C respectively. Further details on some other cryptographic primitives used in voting protocols are provided later in this thesis as required.

2.9 Summary

This chapter offers an overview of the many complex issues involved in voting. The issue of voter education, legislation, and politics were not covered as they are outside the scope of the research. Descriptions of voting procedures, various counting systems, types of electronic voting, general observations on a specific voting system (eVACS), and security threats and requirements in electronic voting were provided in this chapter. An overview of cryptographic voting protocols was also provided as a basis for later chapters.

The next chapter contains our work on batch verification. The techniques presented can be implemented in many schemes, especially in electronic voting, for better efficiency and performance. These techniques are essential for designing secure and practical cryptographic voting protocols.

Chapter 3

New Batch Theorems and Their Applications in Zero-Knowledge Protocols

In cryptographic voting protocols, producing a voting result requires the processing of many ballots. Such processing may include encryption, re-encryption, shuffling, decryption, or threshold decryption operations (more details on these operations are provided in later chapters). Correctness of such operations are proven and verified using zero-knowledge (ZK) proof and verification protocols.

Batching is a technique to perform a number of similar operations with a single operation. Such a technique offers great computational cost savings as compared to when performing the operations individually.

In this chapter, we propose five new batch theorems – two for computing equality of discrete logarithms with a common base, two for computing equality of discrete logarithms with a common exponent, and one for computing N^{th} roots. These theorems employ the small exponents (SE) test by Bellare *et al.*

Previous work on batching is reviewed and analysed. Deficiencies and limitations in the currently existing theorems and techniques are avoided. The work by Bellare *et al.* [BGR98] was successfully attacked by Boyd and Pavlovski [BP00]. The original work by Bellare *et al.* is extended in this chapter.

The batch theorems are applied to the ZK protocols. The resulting techniques include batch ZK proof and verification of correct re-encryptions, and batch ZK

proof and verification of correct decryptions. Both the proof and verification operations are batched. Later chapters illustrate the use of these techniques in cryptographic voting protocols. However, the theorems can be applied in many other appropriate schemes.

This chapter provides a fundamental cryptographic primitive for this thesis. Some material from this chapter has been previously published in [APB+04].

3.1 Background

In this thesis, *batching* is defined as a cryptographic technique by which many instances of the same cryptographic operation can be performed in a batch (a single instance of operation achieving the same effect as many instances of the individual operation), such that the overall computational cost can be lowered. The first practical batch scheme was proposed by Fiat [Fia89], where encryption, decryption, digital signature generation and digital signature verification using the RSA cryptosystem are batched.

Batch has a very wide range of applications, especially for combining many zero-knowledge (ZK) proof constructions and verifications. This is because they are frequently applied operations in cryptographic protocols. Also, it is very often that many instances of the same ZK proof and verification function appear simultaneously. More details on ZK proof and verification protocols are provided in Appendix C.

Bellare *et al.* [BGR98] gave the first definition of batch ZK proof and verification, in which three batch techniques for verification of common-base exponentiations are proposed. The objective is to verify $y_i = g^{x_i} \bmod p$, where $i \in \{1, 2, \ldots, n\}$, $x_i \in \mathbb{Z}_q$, $y_i \in \mathbb{Z}_p^*$, p and q are large primes, $p|q-1$, and g is a generator for the group G of order q. The naive solution is to individually calculate g^{x_i} and compare the results with y_i for $i = 1, 2, \ldots, n$.

An intuitive idea to batch verify the n equations is to test $\prod_{i=1}^{n} y_i = g^{\sum_{i=1}^{n} x_i}$. Harn [Har98] used this idea to construct a batch verification protocol. However this method is not sound since it is easy to pass the verification for an input containing a pair (x_i, y_i) where $y_i \neq g^{x_i}$. For example, $y_1 = zg^{x_1}$ and $y_2 = z^{-1}g^{x_2}$, where any random z can pass the verification. In this chapter, this scheme is called *naive batch verification*.

Bellare *et al.* proposed three batch techniques based on discrete logarithms

- **RS test**: Repeat the following atomic test independently L times and accept if and only if all sub-tests accept.
  1. For $i \in \{1, 2, \ldots, n\}$, choose t_i at random from $\{0, 1\}$.
  2. Compute $z_1 = \prod_{i=1}^{n} y_i^{t_i}$ and $z_2 = \sum_{i}^{n} x_i t_i$.
  3. Accept if $z_1 = g^{z_2}$, reject otherwise.
- **SE test**:
  1. For $i \in \{1, 2, \ldots, n\}$, choose small integers t_i with length L at random.
  2. Compute $z_1 = \prod_{i=1}^{n} y_i^{t_i}$ and $z_2 = \sum_{i=1}^{n} x_i t_i$.
  3. Accept if $z_1 = g^{z_2}$, reject otherwise.
- **Bucket test**: Set $m \geq 2$ and $M = 2^m$. Repeat the following atomic test independently $L/(m-1)$ times and accept if and only if all sub-tests accept.
  1. For $i \in \{1, 2, \ldots, n\}$, choose t_i from $\{1, 2, \ldots, M\}$ at random.
  2. For $j = 1, 2, \ldots, M$, let $B_i = \{i \ : \ t_i = j\}$.
  3. For $j = 1, 2, \ldots, M$, compute $z_{j,1} = \prod_{i \in B_j} y_i$ and $z_{j,2} = \sum_{i \in B_j} x_i t_i$.
  4. Run SE test on the instances $(z_{1,1}, z_{1,2}), (z_{2,1}, z_{2,2}), \ldots, (z_{M,1}, z_{M,2})$.

Figure 3.1: Techniques for batch verifying exponentiations with common bases by Bellare *et al.*.

with a common base. They are RS (random subset) test, SE (small exponents) test and bucket test. These three tests are illustrated in Figure 3.1. Unless specified otherwise, multiplication in the figure is computed modulo p.

For $i \in \{1, 2, \ldots, n\}$, Bellare *et al.* proved that if $y_i \in G$, the SE test costs $n + L + nL/2 + ExpCost(\log_2 q)$ modular multiplications, where $ExpCost^n(\log_2 q)$ denotes the number of modular multiplications required to compute n exponentiations in a common base with different exponents of the same bit-length of q. The probability that incorrect inputs can pass the verification in this test is no more than 2^{-L}. Thus, when L is 20 the failure probability is smaller than one in a million. Efficiency can be improved greatly when the bit-length of L is much smaller than the bit-length of q. A similar efficiency improvement can be achieved with the bucket test.

Bellare *et al.* use the SE test or bucket test together with a slightly modified

Digital Signature Standard (DSS) scheme to achieve efficient batch signature verification.

Although the SE test and bucket test are more secure than the naive batch verification through use of the random small exponents t_i for $i \in \{1, 2, \ldots, n\}$, they are sound only under the assumption that $y_i \in G$. Otherwise, an input containing a false pair (x_i, y_i) with $y_i \neq g^{x_i}$ can still be generated to pass the batch verification. Boyd and Pavlovski noticed that Bellare *et al.* seem to overlook the impact of this assumption. Although the security theorem provided for the SE test is correct, its application to DSS verification is inappropriate as there is no efficient method to verify $y_i \in G$ (one exponentiation is required). When $y_i \notin G$, the probability for a false batch to pass the verification can be much greater than 2^{-L} (when $y_i = g'^{(p-1)/2} g^{x_i}$, the probability is at least $\frac{1}{2}$ where g' denotes a generator for $\mathbb{Z}_p^*$). In this application, soundness and high efficiency cannot be achieved simultaneously.

The RS test does not offer as much efficiency improvement as the other two tests. The bucket test is a variation of the SE test and is more efficient when the batch is of greater size. For $i \in \{1, 2, \ldots, n\}$, all the pairs of (x_i, y_i) are divided into buckets, and SE test is performed in each bucket. In this chapter we focus on the SE test. It should be noted that the bucket test can also be applied in all the theorems and applications provided in this chapter.

3.2 Batch Theorems

The fundamental nature of batch is to obtain efficiency improvements. Batch theorems presented in this section replace some of the full-length exponentiations with short-length exponentiations.

The length of an exponent in a full-length exponentiation is greater than the length of an exponent in a short-length exponentiation. This comparison also applies to computational cost in terms of modular exponentiations. The computational cost for a full-length exponentiation is greater than the computational cost for a short-length exponentiation.

When a batch theorem has two versions, with and without group membership test respectively, the version with the test is called a *strict* theorem and the version without the test is called a *loose* theorem. This follows the terminology used by Hoshino *et al.* [HAK01].

Five batch theorems are presented in this section. The theorems employ small exponents test from the work by Bellare *et al.* In Section 3.3, applications of these theorems are based on the ElGamal and Paillier cryptosystem. The first four theorems are based on the ElGamal cryptosystem parameters. The last theorem is based on the Paillier cryptosystem parameters. More details on the cryptosystems are provided in Appendix B.

Background for the first four theorems (Section 3.2.1 and Section 3.2.2) are described as follows. Let q be a large prime, such that $p = 2q+1$ is a strong prime. The group G, of order $ord(G) = q$ and a generator g, is a cyclic multiplicative subgroup in $\mathbb{Z}_p^*$. The value of $y = g^x$ where x is selected at random from $\mathbb{Z}_q^*$. The absolute value of z is defined as $\pm z$ where $+z$ or z denotes that $z \in G$, and $-z(\mathrm{mod}p)$ denotes that $z \in \mathbb{Z}_p^* \setminus G$ (in the group $\mathbb{Z}_p^*$, but outside the group G). As g' denotes a generator for $\mathbb{Z}_p^*$, $-1 = g'^q$. For $i \in \{1, 2, \ldots, n\}$, and $z_i \in \mathbb{Z}_p^*$, the following two equations hold:

$$(\pm z_1)(\pm z_2)\ldots(\pm z_n) = \pm(z_1 z_2 \ldots z_n) \quad (3.1)$$

$$(\pm z)^n = \pm(z^n) \quad (3.2)$$

Parameters for the last theorem are described at the beginning of Section 3.2.3.

3.2.1 Equality of Logarithms with Common Bases

For $i \in \{1, 2, \ldots, n\}$, the two theorems in this subsection are designed to batch instances of $\log_g y_i = \log_c z_i$ into a single equation. The left hand side of the previous equation has g as a common base, while the right hand side has c as the common base.

In some applications, a strict batch theorem is applied if both values of $y_i, z_i \in G$. Otherwise, a loose batch theorem is applied if either values of $y_i \notin G$ or values of $z_i \notin G$.

A Strict Theorem

Unless specified otherwise, any multiplicative computation in this subsection occurs in the cyclic group G, and $i \in \{1, 2, \ldots, n\}$.

The following theorem is designed to batch instances of $\log_g y_i = \log_c z_i$ into a single equation, where both values of $y_i, z_i \in G$. An application of this theorem

To prove and verify: $\log_g y_i = \log_c z_i$, for $i \in \{1, 2, \ldots, n\}$.

Prover			*Verifier*	
τ_i	$\in_R \mathbb{Z}_q$			
$\gamma_{i,1}$	$= g^{\tau_i} \bmod p$			
$\gamma_{i,2}$	$= c^{\tau_i} \bmod p$			
		$\xrightarrow{\gamma_{i,1},\gamma_{i,2}}$		
			t_i	$\in_R \{1, 2, \ldots, 2^L\}$
			u	$\in_R \mathbb{Z}_q$
		$\xleftarrow{t_i,u}$		
w_i	$= \tau_i - u t_i r'_i \bmod q$			
		$\xrightarrow{w_i}$		
			$\prod_{i=1}^n \gamma_{i,1}$	$\stackrel{?}{=} g^{(\sum_{i=1}^n w_i)}$ $(\prod_{i=1}^n y_i^{t_i})^u \bmod p$
			$\prod_{i=1}^n \gamma_{i,2}$	$\stackrel{?}{=} c^{(\sum_{i=1}^n w_i)}$ $(\prod_{i=1}^n z_i^{t_i})^u \bmod p$

Figure 3.2: A batch ZK proof-verification technique for equality of discrete logarithms.

is to batch zero-knowledge proof-verification of equality of logarithms illustrated in Figure 3.2.

Theorem 3.2.1. *G is a cyclic group with q as the smallest factor of $ord(G)$, generators g and c, and a security parameter L, where $2^L < q$. The small exponents t_i are random L-bit strings, and $y_i, z_i \in G$. If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y_k \neq \log_c z_k$, then $\log_g \prod_{i=1}^n y_i^{t_i} \neq \log_c \prod_{i=1}^n z_i^{t_i}$ with a probability (taken over choice of t_i) of no less than $1 - 2^{-L}$.*

To prove Theorem 3.2.1, we first prove the following lemma:

Lemma 3.2.2. *If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y_k \neq log_c z_k$, given a definite set $S = \{t_i | t_i < 2^L \wedge i \in \{1, \ldots, k-1, k+1, \ldots, n\}\}$, then there is only at most one small exponent t_k satisfying $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$.*

Proof (Lemma 3.2.2). If the lemma is incorrect, the following two equations are

satisfied simultaneously where $\log_g y_k \neq \log_c z_k$ and $t_k \neq t'_k$.

$$\begin{aligned} \log_g \prod_{i=1}^{n} y_i^{t_i} &= \log_c \prod_{i=1}^{n} z_i^{t_i} \\ \log_g (\prod_{i=1}^{k-1} y_i^{t_i})(y_k^{t'_k})(\prod_{i=k+1}^{n} y_i^{t_i}) &= \log_c (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^{n} z_i^{t_i}) \end{aligned}$$

Without loss of generality, suppose $t'_k > t_k$. We can combine and simplify the previous two equations to be $\log_g y_k^{t'_k - t_k} = \log_c z_k^{t'_k - t_k}$. Thus, $(t'_k - t_k)\log_g y_k = (t'_k - t_k)\log_c z_k$. Note that $t'_k - t_k \neq 0$ because $1 \leq (t_k, t'_k) < 2^L < ord(G)$ and $t_k \neq t'_k$. Therefore, $\log_g y_k = \log_c z_k$. This is contradictory to the assumption of $\log_g y_k \neq \log_c z_k$. □

Proof (Theorem 3.2.1). Lemma 3.2.2 implies that among the $(2^L)^n$ possible combinations of t_i, at most $(2^L)^{n-1}$ of them can satisfy $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ when $y_i, z_i \in G$ and $\log_g y_k \neq \log_c z_k$. Therefore, given a random small exponent t_i, if $\log_g y_k \neq \log_c z_k$, then $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ is accepted with a probability of no more than 2^{-L}. □

For $i \in \{1, 2, \ldots, n\}$, instances of $\log_g y_i = \log_c z_i$ can be batched using the equation $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ when $y_i, z_i \in G$. This is according to Theorem 3.2.1. The probability that $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ while $\log_g y_k \neq \log_c z_k$ for some $k \in \{1, 2, \ldots, n\}$ is no more than 2^{-L}.

A Loose Theorem

In Theorem 3.2.1, there is a condition that $g, y_i, c, z_i \in G$ for $i \in \{1, 2, \ldots, n\}$. However, in some applications there is an uncertainty of satisfaction on this condition. An additional computation is often required to verify the condition. In reality, this extra computation is too expensive such that in many cases it prevents the applicability of Theorem 3.2.1.

To overcome this problem, Theorem 3.2.3 is proposed. This theorem does not require the pre-condition that the LHS and RHS of the batch equation be in the same cyclic subgroup of $\mathbb{Z}_p^*$.

An application of this theorem is to batch zero-knowledge proof-verification of equality of logarithms for checking valid ElGamal re-encryptions. This is described in Section 3.3.1.

Unless specified otherwise, any multiplicative computation in this subsection occurs in the cyclic group G, and $i \in \{1, 2, \ldots, n\}$.

Theorem 3.2.3. *Let q be a large prime, such that $p = 2q + 1$ is a strong prime. The group G, of order q and generator g, is a cyclic multiplicative subgroup in $\mathbb{Z}_p^*$. For $x \in_R \mathbb{Z}_q^*$, $y_i, z_i \in \mathbb{Z}_p^*$, L is a security parameter satisfying $2^L < q$, and small exponents t_i are random L-bit strings. If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g \pm y_k \neq \log_c \pm z_i$, then $\log_g \prod_{i=1}^n y_i^{t_i} \neq \log_c \prod_{i=1}^n z_i^{t_i}$ with a probability (taken over choice of t_i) of no less than $1 - 2^{-L}$.*

To prove Theorem 3.2.3, we first prove the following lemma:

Lemma 3.2.4. *If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g \pm y_k \neq \log_c \pm z_k$, given a definite set $S = \{t_i | t_i < 2^L \wedge i \in \{1, \ldots, k-1, k+1, \ldots, n\}\}$, then there is only at most one small exponent t_k satisfying $\log_g \prod_{i=1}^n y_i^{t_i} \neq \log_c \prod_{i=1}^n z_i^{t_i}$.*

Proof (Lemma 3.2.4). If this lemma is incorrect, the following two equations are satisfied simultaneously where $\log_g \pm y_k \neq \log_c \pm z_k$ and $t_k \neq t'_k$.

$$\begin{aligned} \log_g \prod_{i=1}^{n} y_i^{t_i} &= \log_c \prod_{i=1}^{n} z_i^{t_i} \\ \log_g (\prod_{i=1}^{k-1} y_i^{t_i})(y_k^{t'_k})(\prod_{i=k+1}^{n} y_i^{t_i}) &= \log_c (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^{n} z_i^{t_i}) \end{aligned}$$

Without loss of generality, suppose $t'_k > t_k$. We can combine and simplify the previous two equations to be $\log_g \pm y_k^{t'_k - t_k} = \log_c \pm z_k^{t'_k - t_k}$. Thus, $(t'_k - t_k) \log_g \pm y_k = (t'_k - t_k) \log_c \pm z_k$. Note that $t'_k - t_k \neq 0$ because $1 \leq (t_k, t'_k) < 2^L < q$ and $t_k \neq t'_k$. Therefore, $\log_g \pm y_k = \log_c \pm z_k$. This is contradictory to the assumption of $\log_g \pm y_k \neq \log_c \pm z_k$. □

Proof (Theorem 3.2.3). Lemma 3.2.4 implies that among the $(2^L)^n$ possible combinations of t_i, at most $(2^L)^{n-1}$ of them can satisfy $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ when $\log_g \pm y_k \neq \log_c \pm z_k$. Therefore, given a random small exponent t_i, if $\log_g \pm y_k \neq \log_c \pm z_k$, then $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ is accepted with a probability of no more than 2^{-L}. □

For $i \in \{1, 2, \ldots, n\}$, instances of $\log_g \pm y_i = \log_c \pm z_i$ can be batched using the equation $\log_g \prod_{i=1}^n y_i^{t_i} = \log_c \prod_{i=1}^n z_i^{t_i}$ when q is a large prime, $p = 2q+1$ is a strong prime, and $g, c \in G$. This is according to Theorem 3.2.3. The probability that

$\log_g \prod_{i=1}^{n} y_i^{t_i} = \log_c \prod_{i=1}^{n} z_i^{t_i}$ while $\log_g \pm y_k \neq \log_c \pm z_k$ for some $k \in \{1, 2, \ldots, n\}$ is no more than 2^{-L}.

Note: Tests of $\left(\frac{y_i}{p}\right) = 1$ and/or $\left(\frac{z_i}{p}\right) = 1$ (using Legendre symbol) can be performed to determine whether $y_i, z_i \in G$ or $-y_i, -z_i \in G$. If the test is accepted, then $y_i, z_i \in G$, otherwise $-y_i, -z_i \in G$.

3.2.2 Equality of Logarithms with Common Exponents

For $i \in \{1, 2, \ldots, n\}$, the two theorems in this subsection are designed to batch instances of $\log_g y = \log_{c_i} z_i$ into a single equation. The value of $y \in G$, and the value of $\log_g y$ is the same for every pair of (c_i, z_i).

In some applications, a strict batch theorem is applied if both values of $c_i, z_i \in G$. Otherwise, a loose batch theorem is applied if either values of $c_i \notin G$ or values of $z_i \notin G$.

A Strict Theorem

Applications of this theorem are to batch zero-knowledge proof - verification of equality of logarithms for checking valid centralised ElGamal decryptions and for checking valid threshold Paillier decryptions. This is described in Section 3.3.2.

As before, any multiplicative computation in this subsection occurs in the cyclic group G, and $i \in \{1, 2, \ldots, n\}$ unless specified otherwise.

The following theorem is designed to batch instances of $\log_g y = \log_{c_i} z_i$ into a single equation, where both values of $c_i, z_i \in G$.

Theorem 3.2.5. *G is a cyclic group with q as the smallest factor of $ord(G)$, generators g and c_i, and a security parameter L, where $2^L < q$. The small exponents t_i are random L-bit strings, and $y, z_i \in G$. If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y \neq \log_{c_k} z_k$, then $\log_g y \neq \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$ with a probability (taken over choice of t_i) of no less than $1 - 2^{-L}$.*

To prove Theorem 3.2.5, we first prove the following lemma:

Lemma 3.2.6. *If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y \neq log_{c_k} z_k$, given a definite set $S = \{t_i | t_i < 2^L \wedge i \in \{1, \ldots, k-1, k+1, \ldots, n\}\}$, then there is only at most one small exponent t_k satisfying $\log_g y = \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$.*

Proof (Lemma 3.2.6). If the lemma is incorrect, the following two equations are satisfied simultaneously where $\log_g y \neq \log_{c_k} z_k$ and $t_k \neq t'_k$.

$$\begin{aligned}
\log_g y &= \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i} \\
\log_g y &= \log_{(\prod_{i=1}^{k-1} c_i^{t_i})(c_k^{t'_k})(\prod_{i=k+1}^{n} c_i^{t_i})} (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^{n} z_i^{t_i})
\end{aligned}$$

Let $y = g^x$, the two previous equations can be re-written as:

$$\begin{aligned}
(\prod_{i=1}^{n} c_i^{t_i})^x &= \prod_{i=1}^{n} z_i^{t_i} \\
((\prod_{i=1}^{k-1} c_i^{t_i})(c_k^{t'_k})(\prod_{i=k+1}^{n} c_i^{t_i}))^x &= (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^{n} z_i^{t_i})
\end{aligned}$$

Without loss of generality, suppose $t'_k > t_k$. We can combine and simplify the previous two equations to be $c_k^{x(t'_k - t_k)} = z_k^{t'_k - t_k}$. Thus, $(c_k^x / z_k)^{t'_k - t_k} = 1$. As $(c_k^x / z_k) \in G$, $t'_k - t_k$ is a factor of $ord(G)$ if $(c_k^x / z_k) \neq 1$. Since $0 < (t'_k - t_k) < ord(G)$, therefore $(c_k^x / z_k) = 1$ or $c_k^x = z_k$. This is contradictory to the assumption of $\log_g y \neq \log_{c_k} z_k$. □

Proof (Theorem 3.2.5). Lemma 3.2.6 means that among the $(2^L)^n$ possible combinations of t_i, at most $(2^L)^{n-1}$ of them can satisfy $\log_g y = \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$ when $c_i, z_i \in G$ and $\log_g y \neq \log_{c_k} z_k$. Therefore, given a random small exponent t_i, if $\log_g y \neq \log_{c_i} z_i$, then $\log_g y = \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$ is accepted with a probability of no more than 2^{-L}. □

For $i \in \{1, 2, \ldots, n\}$, instances of $\log_g y = \log_{c_i} z_i$ can be batched using the equation $\log_g y = \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$ when $c_i, z_i \in G$. This is according to Theorem 3.2.5. The probability that $\log_g y = \log_{\prod_{i=1}^{n} c_i^{t_i}} \prod_{i=1}^{n} z_i^{t_i}$ while $\log_g y \neq \log_{c_k} z_k$ for some $k \in \{1, 2, \ldots, n\}$ is no more than 2^{-L}.

A Loose Theorem

As for Theorem 3.2.1, we also specify a loose version of Theorem 3.2.5. An application of this theorem is to batch zero-knowledge proof-verification of equality of logarithms for checking valid threshold ElGamal decryptions described in Section 3.3.2.

Theorem 3.2.7. *Let q be a large prime, such that $p = 2q + 1$ is a strong prime. The group G, of order q and generator g, is a cyclic multiplicative subgroup in $\mathbb{Z}_p^*$. For $x \in_R \mathbb{Z}_q^*$, $y = g^x \in G$, $z_i \in \mathbb{Z}_p^*$, L is a security parameter satisfying $2^L < q$ and small exponents t_i are random L-bit strings. If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y \neq \log_{c_k} \pm z_k \bmod p$, then $\log_g y \neq \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$ with a probability of no less than $1 - 2^{-L}$.*

To prove Theorem 3.2.7, we first prove the following lemma:

Lemma 3.2.8. *If $\exists k \in \{1, 2, \ldots, n\} \wedge \log_g y \neq \log_{c_k} \pm z_k$, given a definite set $S = \{t_i | t_i < 2^L \wedge i \in \{1, \ldots, k-1, k+1, \ldots, n\}\}$, then there is only at most one small exponent t_k satisfying $\log_g y = \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$.*

Proof (Lemma 3.2.8). If the lemma is incorrect, the following two equations are satisfied simultaneously where $\log_g y \neq \log_{c_k} \pm z_k$ and $t_k \neq t'_k$.

$$\begin{aligned} \log_g y &= \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i} \\ \log_g y &= \log_{(\prod_{i=1}^{k-1} c_i^{t_i})(c_k^{t'_k})(\prod_{i=k+1}^n c_i^{t_i})} (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^n z_i^{t_i}) \end{aligned}$$

Let $y = g^x$, the two previous equations can be re-written as:

$$\begin{aligned} (\prod_{i=1}^n c_i^{t_i})^x &= \prod_{i=1}^n z_i^{t_i} \\ ((\prod_{i=1}^{k-1} c_i^{t_i})(c_k^{t'_k})(\prod_{i=k+1}^n c_i^{t_i}))^x &= (\prod_{i=1}^{k-1} z_i^{t_i})(z_k^{t'_k})(\prod_{i=k+1}^n z_i^{t_i}) \end{aligned}$$

Without loss of generality, suppose $t'_k > t_k$. We can combine and simplify the previous two equations to be $c_k^{x(t'_k - t_k)} = z_k^{t'_k - t_k}$. Thus, $(c_k^x / z_k)^{t'_k - t_k} = 1$. Because $(c_k^x / z_k) \in \mathbb{Z}_p^*$, $t'_k - t_k$ is a factor of $p - 1$ if $(c_k^x / z_k) \neq 1$. As $1 \leq t_k < t'_k$, $0 < (t'_k - t_k) < q$. Hence, if $t'_k - t_k$ is a factor of $p - 1$, $(t'_k - t_k) = 2$. Therefore, $(c_k^x / z_k) = 1 \vee (c_k^x / z_k)^2 = 1$. In short $(c_k^x / z_k) = \pm 1$, or $c_k^x = \pm z_k$. This is contradictory to the assumption of $\log_g y \neq \log_{c_k} \pm z_k$. □

Proof (Theorem 3.2.7). Lemma 3.2.8 means that among the $(2^L)^n$ possible combinations of t_i, at most $(2^L)^{n-1}$ of them can satisfy $\log_g y = \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$

when $y \in G$ and $\log_g y \neq \log_{c_k} \pm z_k$. Therefore, given a random small exponent t_i, if $\log_g y \neq \log_{c_i} \pm z_i$, then $\log_g y = \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$ is accepted with a probability of no more than 2^{-L}. □

For $i \in \{1, 2, \ldots, n\}$, instances of $\log_g y = \log_{c_i} \pm z_i$ can be batched using the equation $\log_g y = \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$ when q is a large prime, $p = 2q + 1$ is a strong prime, and $g, y \in G$ according to Theorem 3.2.7. The probability that $\log_g y = \log_{\prod_{i=1}^n c_i^{t_i}} \prod_{i=1}^n z_i^{t_i}$ while $\log_g y \neq \log_{c_k} \pm z_k$ for some $k \in \{1, 2, \ldots, n\}$ is no more than 2^{-L}.

Note: A test of $\left(\frac{z_i}{p}\right) = 1$ (using Legendre symbol) can be performed to determine whether $z_i \in G$ or $-z_i \in G$. If the test is accepted, then $z_i \in G$, otherwise $-z_i \in G$.

3.2.3 Computations of N^{th} Root

Unless specified otherwise, parameters for the following theorem are described as follows. The values of p' and q' are large primes, such that both $p = 2p' + 1$ and $q = 2q' + 1$ are strong primes. The value of $N = pq$, and $\text{GCD}(N, p'q') = 1$ where GCD denotes a Greatest Common Divisor. The cyclic group G is a set of quadratic residues in $\mathbb{Z}^*_{N^2}$. Any multiplicative computation in this subsection occurs in the cyclic group $\mathbb{Z}^*_{N^2}$ with modulo N^2, and $i \in \{1, 2, \ldots, n\}$.

The following theorem is designed to batch instances of $r_i^{\frac{1}{N}}$ into a single calculation, where the values of $r_i \in \mathbb{Z}_{N^2}$. An application of this theorem is to batch zero-knowledge proof-verification of knowledge of roots for checking valid Paillier re-encryptions in Section 3.3.1.

Theorem 3.2.9. *Let the values of $r_i \in \mathbb{Z}^*_{N^2}$, and L be a security parameter, where $2^L < min(p', q')$. Small exponents t_i are random L-bit strings. If there exists a polynomial-time deterministic algorithm which can calculate $(\prod_{i=1}^n r_i^{t_i})^{\frac{1}{N}}$ with a probability (taken over choice of t_i) bigger than 2^{-L}, then the values of $r_i^{\frac{1}{N}}$ can be calculated in polynomial time.*

To prove Theorem 3.2.9, we first prove the following lemma:

Lemma 3.2.10. *Let $r_i \in \mathbb{Z}^*_{N^2}$, $t_i < 2^L < min(p, q)$, and $k \in \{1, 2, \ldots, n\}$. If more than one possible small exponents t_k can be found such that $(\prod_{i=1}^n r_i^{t_i})^{\frac{1}{N}}$ can be calculated in polynomial time, given a definite set $S = \{t_i | t_i < 2^L \wedge i \in \{1, \ldots, k-1, k+1, \ldots, n\}\}$, then $r_k^{\frac{1}{N}}$ can be calculated in polynomial time.*

Proof (Lemma 3.2.10). Let the two possible small exponents be t_k and t'_k, where $t_k \neq t'_k$. Suppose $\Gamma = (\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ and $\Gamma' = ((\prod_{i=1}^{k-1} r_i^{t_i})(r_k^{t'_k})(\prod_{i=k+1}^{n} r_i^{t_i}))^{\frac{1}{N}}$ can be calculated in polynomial time. Without loss of generality, suppose $t'_k > t_k$. The value of $\omega = (\Gamma/\Gamma')$, or $\omega^N = r_k^{t'_k - t_k}$. According to the Eucledian algorithm, there exist integers a and b, such that $b(t'_k - t_k) = aN + \mathrm{GCD}(N, t'_k - t_k)$. Note that $t'_k - t_k < 2^L < min(p, q)$. Thus, $\mathrm{GCD}(N, t'_k - t_k) = 1$ and $(\omega^b / r_k^a)^N = r_k$. Since ω can be calculated in polynomial time, both values of a and b can also be calculated in polynomial time from N and $t'_k - t_k$ using the algorithm. Therefore, N^{th} root of r_k can be calculated in polynomial time if $r_i^{\frac{1}{N}}$ can also be calculated in polynomial time. □

Proof (Theorem 3.2.9). Assume that there exists a polynomial-time deterministic algorithm which can calculate $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ with a probability (taken over choice of t_i) bigger than 2^{-L}. If the assumption is incorrect, then for every possible combination of small exponents $t_1, t_2, \ldots, t_{k-1}, t_{k+1}, \ldots, t_n$ in $\{0, 2, \ldots, 2^L - 1\}^{n-1}$ there exists at most one small exponent t_k in $\{0, 2, \ldots, 2^L - 1\}$ such that $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ can be calculated in polynomial time. This implies the probability that the calculation of $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ can be computed in polynomial time is no more than 2^{-L} (calculation of $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ in polynomial time is only possible with at most $2^{(n-1)L}$ combinations of t_i out of 2^{nL} possible combinations). This is a contradiction to the assumption.

Hence, we can deduce that for every integer $k \in \{1, 2, \ldots, n\}$, there must exist integers $t_k, t'_k \in \{1, 2, \ldots, 2^L - 1\}^{n+1}$ such that $t_k \neq t'_k$ and the following two equations can be computed in polynomial time.

$$\begin{aligned} \Gamma &= (\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}} \\ \Gamma' &= ((\prod_{i=1}^{k-1} r_i^{t_i})(r_k^{t'_k})(\prod_{i=k+1}^{n} r_i^{t_i}))^{\frac{1}{N}} \end{aligned}$$

Therefore (combining this proof with Lemma 3.2.10) for $k \in \{1, 2, \ldots, n\}$, the calculation of $r_k^{\frac{1}{N}}$ can be computed in polynomial time. □

For $i \in \{1, 2, \ldots, n\}$, instances of $r_i^{\frac{1}{N}}$ can be batched using the equation $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$. This is according to Theorem 3.2.9. If the values of $r_i^{\frac{1}{N}}$ is not known, then the probability of computing $(\prod_{i=1}^{n} r_i^{t_i})^{\frac{1}{N}}$ is negligible.

3.3 Applications in Zero-Knowledge Proof - Verification Protocols

In a zero-knowledge (ZK) proof-verification protocol, a prover demonstrates to a verifier the knowledge of a secret value satisfying a certain relation. The verifier is not to obtain knowledge of the secret. Thus, in this scenario there are two players: the prover P and the verifier V; and two operations: proof and verification. More details on ZK proof-verification protocols are provided in Appendix C.

Especially in cryptographic voting protocols, a large number of inputs (ballots) from voters are required to be processed by voting authorities. This processing includes operations such as: encryption, re-encryption, decryption, or threshold decryption. It is necessary to prove these operations (typically their correctness) for verifiability. This is performed by using ZK proof-verification protocol. When the proofs and their corresponding verifications are batched, efficiency is greatly increased.

The applications provided are to batch the proofs and verifications per voting authority (not per voter). This is because the number of voting authorities are normally much smaller than the number of voters, e.g. in a national election. It should be noted that straight-forward extensions of the theorems to many other applications and schemes are not limited to electronic voting scenario.

Batch theorems provided in the previous section are extended and applied to construct batch ZK proof and verification techniques. In the traditional batch techniques [BGR98, HAK01], there is only one verifier, while no secret information is involved in the verification. Our proposed techniques batch both the ZK proofs and their corresponding verifications.

For simplicity, we describe the batch techniques for one authority. It is straight-forward to apply the techniques for many authorities. The technique is simply repeated by each authority.

In order to make the details as clear as possible, interactive descriptions of the proof verifications are used in this section. In practice, the proofs are usually applied in a non-interactive manner.

Section 3.3.1 details applications of some of the new techniques to prove and verify valid re-encryptions. Another application to prove and verify valid decryption or threshold decryption is provided in Section 3.3.2.

3.3.1 Re-Encryptions

After a party re-encrypts multiple ciphertexts (or encrypts multiple secret messages), it is necessary to prove that each re-encryption (or encryption) is valid. Efficiency is greatly increased when the re-encryption (or encryption) proofs and their corresponding verifications are batched.

For simplicity, only the batch technique for ZK proof-verification of valid re-encryption is described in this subsection. For encryptions, the messages can be regarded as a special ciphertext encrypted using the identity function. As such, it is straight-forward to apply the technique to verification of valid encryption.

ElGamal Cryptosystem

Designing a strict batch ZK proof-verification technique to check correctness of ElGamal re-encryptions requires the application of Theorem 3.2.1. However, application of this theorem requires $2n$ instances of membership test in G for n instances of ZK proof-verification protocols. This test is usually of high cost (of $2n$ full-length exponentiations). Hence it does not provide significant efficiency improvement over individual ZK proof-verification.

We present a batch ZK proof-verification techniques for checking valid ElGamal re-encryptions. It employs a loose verification based on Theorem 3.2.3.

This technique does not provide a strict validity verification, but it is sufficiently strong for applications such as mix-networks (Chapter 5). Unless specified otherwise, all multiplications in this subsection are computed in the group of $\mathbb{Z}_p^*$.

For $i \in \{1, 2, \ldots, n\}$, suppose there are n ciphertexts $c_i = (\alpha_i, \beta_i)$. These ciphertexts are re-encrypted to $c_i' = (\alpha_i', \beta_i')$. According to Theorem 3.2.1, loose verification of correct re-encryptions of ElGamal ciphertexts (from one re-encryption authority - the prover) can be batched using SE test. This is by using the Chaum-Pedersen [CP93] ZK proof of equality of discrete logarithm (Appendix C.3) as:

$$\log_g \prod_{i=1}^{n} \left(\pm \frac{\alpha_i'}{\alpha_i} \right)^{t_i} = \log_y \prod_{i=1}^{n} \left(\pm \frac{\beta_i'}{\beta_i} \right)^{t_i} \tag{3.3}$$

The interactive batch ZK proof-verification protocol for this scenario is shown in Figure 3.3. This protocol can be made non-interactive. This is by using a hash function, employing the well-known Fiat-Shamir heuristic [FS86] using two collision-resistant hash functions H_1 and H_2. The range of H_1 is $\{0,1\}^L$ for

To prove and verify: $\log_g \pm(\alpha'_i/\alpha_i) = \log_y \pm(\beta'_i/\beta_i)$, for $i \in \{1, 2, \ldots, n\}$.

Prover			*Verifier*	
τ_i	$\in_R \mathbb{Z}_q$			
$\gamma_{i,1}$	$= g^{\tau_i} \bmod p$			
$\gamma_{i,2}$	$= y^{\tau_i} \bmod p$			
		$\xrightarrow{\gamma_{i,1},\gamma_{i,2}}$		
			t_i	$\in_R \{1, 2, \ldots, 2^L\}$
			u	$\in_R \mathbb{Z}_q$
		$\xleftarrow{t_i,u}$		
w_i	$= \tau_i - u t_i r'_i \bmod q$			
		$\xrightarrow{w_i}$		
			$\prod_{i=1}^n \gamma_{i,1}$	$\stackrel{?}{=} g^{(\sum_{i=1}^n w_i)}$ $(\prod_{i=1}^n (\alpha'/\alpha)^{t_i})^u \bmod p$
			$\prod_{i=1}^n \gamma_{i,2}$	$\stackrel{?}{=} y^{(\sum_{i=1}^n w_i)}$ $(\prod_{i=1}^n (\beta'/\beta)^{t_i})^u \bmod p$

Figure 3.3: A batch ZK proof-verification technique for verifying valid ElGamal ciphertext re-encryptions.

producing the small exponents t_i, and the range of H_2 is $\mathbb{Z}_q$ for producing the challenge u. The small exponents and challenge are generated as follows:

$$\begin{aligned} t_i &= H_1(\gamma_{i,1}, \gamma_{i,2}, g, y, (\alpha'_i/\alpha_i), (\beta'_i/\beta_i)) \\ u &= H_2(g, y, \{\gamma_{i,1}, \gamma_{i,2}, (\alpha'_i/\alpha_i), (\beta'_i/\beta_i)\}) \end{aligned}$$

According to Theorem 3.2.3, the above batch ZK proof-verification technique guarantees that:

$$\log_g \pm \left(\frac{\alpha'_i}{\alpha_i}\right) = \log_y \pm \left(\frac{\beta'_i}{\beta_i}\right) \tag{3.4}$$

Equivalently $D(c'_i) = D(c_i)$ OR $D(c'_i) = g'^q D(c_i)$, where D denotes an ElGamal decryption function for the corresponding ciphertext.

This loose verification technique does not completely guarantee correct re-encryption. Namely, unless $\pm(\alpha'_i/\alpha_i) \in G \wedge \pm(\beta'_i/\beta_i) \in G$, the batch verification can only be passed with negligible probability. Thus, the batch verification result is not yet satisfactory as the recovered secret message may be incorrect: $\pm s_i \in G$.

To fix this, the decryption requires one extra step, i.e. multiplying s_i with (-1) when $s_i \notin G$. After s_i is recovered through the decryption procedure, we

test if $\left(\frac{s_i}{p}\right) = 1$ (using the Legendre symbol). If it is accepted, $s_i \in G$. Otherwise, $s_i = -s_i \bmod p$. The additional cost is only one exponentiation per ciphertext. This batch technique is sufficient in some applications where there are many re-encryption authorities (provers).

Paillier Cryptosystem

Paillier cryptosystem (Appendix B.2) also allows re-encryption of its ciphertext. For $i \in \{1, 2, \ldots, n\}$, suppose there are n ciphertexts c_i each containing a secret message s_i. The ciphertexts c_i can be re-encrypted using new random values r'_i as $c'_i = c_i r'^N_i \bmod N^2$. Theorem 3.2.9 is developed using the parameters of Paillier cryptosystem. Thus, the theorem is suitable to batch verify re-encryptions of these Paillier ciphertexts.

According to Theorem 3.2.9, verification of correct re-encryptions of Paillier ciphertexts (from one re-encryption authority - the prover) can be batched using SE test using Guillou-Quisquater [GQ88] ZK proof of knowledge of root (Appendix C.2) as:

$$\left(\frac{\prod_{i=1}^{n} c'^{t_i}_i}{\prod_{i=1}^{n} c_i^{t_i}}\right)^{\frac{1}{N}} \bmod N^2 \tag{3.5}$$

The interactive batch ZK proof-verification protocol for this scenario is shown in Figure 3.4. This protocol can be made non-interactive. This is by using a hash function, employing the well-known Fiat-Shamir heuristic [FS86] using two collision-resistant hash functions H_1 and H_2. The range of H_1 is $\{0,1\}^L$ for producing the small exponents t_i, and the range of H_2 is $\mathbb{Z}_N$ for producing the challenge u. The small exponents and challenge are generated as follows:

$$\begin{aligned} t_i &= H_1(\gamma_i, N, (c'_i/c_i)) \\ u &= H_2(N, \{\gamma_i, (c'_i/c_i)\}) \end{aligned}$$

According to Theorem 3.2.9, the above batch ZK proof-verification technique guarantees knowledge of the new random values r'_i by proving the knowledge of $(\prod_{i=1}^{n} c'^{t_i}_i / \prod_{i=1}^{n} c_i^{t_i})^{\frac{1}{N}} \bmod N^2$. This proves correct re-encryptions of Paillier ciphertexts. Consequently, decryptions of the re-encrypted ciphertexts corresponds to decryptions of the original ciphertexts $D(c'_i) = D(c_i)$, where D denotes a Paillier decryption function for the corresponding ciphertext.

To prove and verify proof of knowledge of $(c'_i/c_i)^{\frac{1}{N}} \bmod N^2$, for $i \in \{1, 2, \ldots, n\}$.

Prover			*Verifier*	
τ_i	$\in_R (1, N^2 - 1)$			
γ_i	$= \tau_i^N \bmod N^2$			
		$\xrightarrow{\gamma_i}$		
			t_i	$\in_R \{1, 2, \ldots, 2^L\}$
			u	$\in_R \mathbb{Z}_N^*$
		$\xleftarrow{t_i, u}$		
w_i	$= \tau_i (\prod_{i=1}^n r_i'^{t_i})^{-u} \bmod N^2$			
		$\xrightarrow{w_i}$		
			$\prod_{i=1}^n \gamma_i$	$\stackrel{?}{=} \left(\frac{\prod_{i=1}^n c_i'^{t_i}}{\prod_{i=1}^n c_i^{t_i}}\right)^u (\prod_{i=1}^n w_i)^N \bmod N^2$

Figure 3.4: A batch ZK proof-verification technique for verifying valid Paillier ciphertext re-encryptions.

3.3.2 (Centralised) Decryptions and Threshold Decryptions

After a party (a decryption authority) decrypts multiple ciphertexts on his/her own (we name this *centralised decryptions*), it is necessary to prove that each of the decryptions is valid. The recovered plaintexts must correspond to those contained in the decrypted ciphertexts.

This is also true for multiple parties (decryption authorities) cooperatively decrypting multiple ciphertexts. Each partial decryption must be proven to be valid.

The computational cost of individual ZK proof-verification protocol for each decryption or partial decryption is high. Efficiency is greatly increased when a batch technique is applied for this scenario.

Table 3.1 summarises the applicability of the batch theorems to either centralised or threshold decryption of ElGamal and Paillier cryptosystems. We describe one batch technique for ZK proof-verification of valid ElGamal decryptions (centralised, non-threshold) and describe the rest of the techniques for threshold decryptions based on ElGamal and Paillier cryptosystems.

Table 3.1: Applicability of batch to different types of decryption.

Decryption type	ElGamal Cryptosystem	Paillier Cryptosystem
Centralised	yes	no
Threshold	yes	yes

ElGamal Cryptosystem (Centralised and Threshold)

Correctness proofs and verifications of both centralised and threshold decryption for ElGamal cryptosystem can be batched. Theorem 3.2.5 is applicable in both the centralised and threshold decryptions scenarios, while Theorem 3.2.7 is also applicable in the threshold decryptions scenario. In this subsection, all multiplicative computations are of modulo p.

Centralised Decryptions: Designing a strict batch ZK proof-verification technique to check correctness of centralised decryptions for ElGamal cryptosystem requires the application of Theorem 3.2.5 (strict). However, application of this theorem requires $2n$ instances of membership test in G for n instances of ZK proof-verification protocols. This test is usually of high cost (of $2n$ full-length exponentiations). To implement efficient membership test in G, parameters and algorithm for the ElGamal cryptosystem are modified as follows:

- Key generation:
  The private key x is chosen at random from $\mathbb{Z}_q$. The value of g is randomly chosen as a generator of G. The public parameters $(g, y = g^x)$ are published.

- Encryption:
  A message $s \in \mathbb{Z}_p^*$ is encrypted using a random value $r \in \mathbb{Z}_q$ as $c = (\alpha, \beta) = (g^r, sy^{2r})$.

- Re-encryption:
  A ciphertext is re-encrypted using a new random value $r' \in \mathbb{Z}_q$ as $c' = (\alpha', \beta') = (\alpha g^{r'}, \beta y^{2r'})$.

- Decryption:
  The original message is reconstructed from the ciphertext c as $s = (\beta / s'^{\,2})$, where $s' = \alpha^x$. Reconstruction of the original message from the re-encrypted ciphertext follows accordingly.

For $i \in \{1, 2, \ldots, n\}$, suppose there are n ciphertexts $c_i = (\alpha_i, \beta_i)$, decrypted to $s_i = (\beta_i / {s'_i}^2)$. Theorem 3.2.5 is suitable to batch verify centralised decryptions of these ElGamal ciphertexts as:

1. For this modified version of ElGamal, G is a cyclic subgroup of $\mathbb{Z}_p^*$.
2. The public parameters of $g \in G$ and $y \in G$ are publicly verifiable by testing $\left(\frac{g}{p}\right) = 1$ and $\left(\frac{y}{p}\right) = 1$ (using the Legendre symbol as in [HAK01]). This proves g and y to be generators of G, if $g, y \neq 1$.
3. The values of α_i^2 and ${s'_i}^2$ in the verification equation $\log_g y = \log_{\alpha_i^2} {s'_i}^2$ are explicitly in G.
4. The small exponents t_i can be chosen at random while satisfying $t_i < 2^L < q$.

According to Theorem 3.2.5, verification of correct centralised decryptions of ElGamal ciphertexts (from one decryption authority - the prover) can be batched using SE test using Chaum-Pedersen ZK proof of equality of discrete logarithm (Appendix C.3) as:

$$\log_g y = \log_{(\prod_{i=1}^n \alpha_i^{t_i})^2} (\prod_{i=1}^n {s'_i}^{t_i})^2 \tag{3.6}$$

The interactive batch ZK proof-verification protocol for this scenario is shown in Figure 3.5. This protocol can be made non-interactive. This is by using a hash function, employing the well-known Fiat-Shamir heuristic [FS86], and the challenge u using the collision-resistant hash function H, where $H : (0,1)^* \rightarrow \mathbb{Z}_q$ as follows:

$$u \;=\; H(\gamma_1, \gamma_2, g, y, \{\alpha_i, s'_i, t_i\})$$

Producing the small exponents non-interactively requires a different scenario. The decryption authority (the prover) is required to commit to the small exponents prior to receiving the ciphertexts. This is as follows:

1. Prior to receiving the ciphertexts, the decryption authority (the prover) selects initial small exponents $t'_i \in \{1, 2, \ldots, 2^L\}$ at random. Using a suitable commitment function, the small exponents are committed and published, e.g. using a hash function with a range of $\{0,1\}^L$ as $\{H(t'_i)\}$.

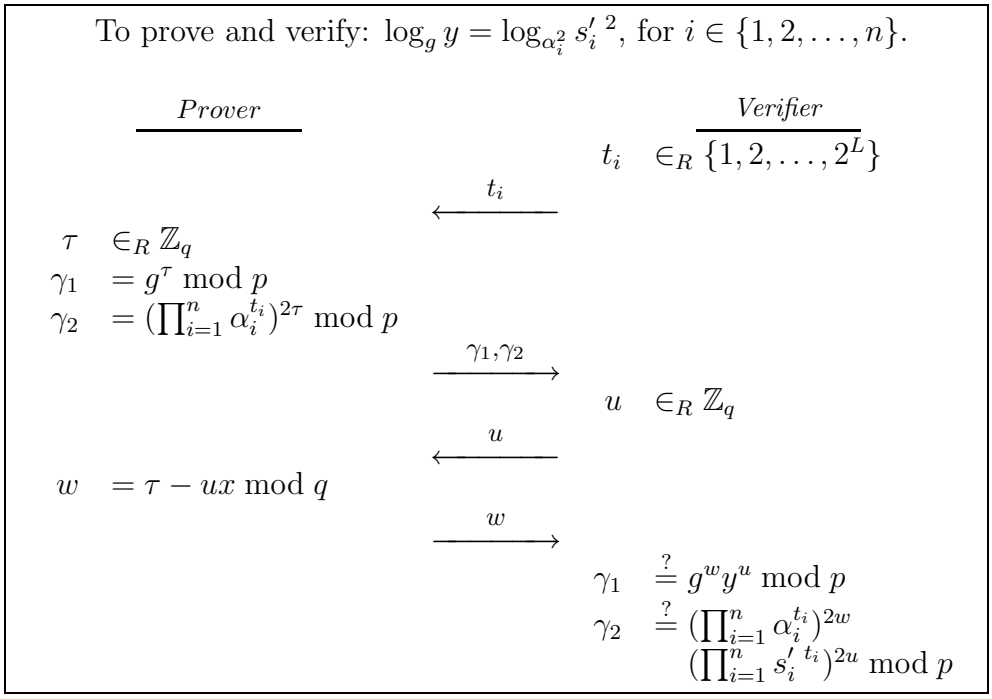

Figure 3.5: A batch ZK proof-verification technique for verifying valid centralised decryptions of ElGamal ciphertexts.

2. The decryption authority then receives and produces their values of s'_i.

3. The small exponents t_i are then calculated using a collision-resistant hash function as $t_i = H(t'_i, s'_i)$.

Note that the use of digital signature on the published values is required to authenticate them. The rest of the non-interactive batch ZK proof-verification protocol follows from the interactive one, with an additional verification for the small exponents.

According to Theorem 3.2.5, the above batch ZK proof-verification technique guarantees that $\log_g y = \log_{\alpha_i^2} {s'_i}^2$. Equivalently $D(c_i)$ are performed using the corresponding private key x, where D denotes an ElGamal decryption function for the corresponding ciphertext.

Threshold Decryptions: Pedersen [Ped92] presented a threshold ElGamal signature scheme. This scheme can be used for threshold decryption. This scheme is recalled in Appendix B.1.2.

Using the modification as in the above centralised decryptions, it is straightforward to apply the centralised decryptions batch technique for a threshold de-

		P_1	P_2		P_j		P_m		
c_1	$\longrightarrow$	$z_{1,1}$	$z_{i,2}$	$\cdots$	$z_{1,j}$	$\cdots$	$z_{1,m}$	$\longrightarrow$	s_1
c_2	$\longrightarrow$	$z_{2,1}$	$z_{2,2}$	$\cdots$	$z_{2,j}$	$\cdots$	$z_{2,m}$	$\longrightarrow$	s_2
$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
c_i	$\longrightarrow$	$z_{i,1}$	$z_{i,2}$	$\cdots$	$z_{i,j}$	$\cdots$	$z_{i,m}$	$\longrightarrow$	s_j
$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
c_n	$\longrightarrow$	$z_{n,1}$	$z_{n,2}$	$\cdots$	$z_{n,j}$	$\cdots$	$z_{n,m}$	$\longrightarrow$	s_m

Figure 3.6: A threshold decryption scenario of m participants $\{P_j\}$, n ciphertexts $\{c_i\}$, nm partial decryptions $\{z_{i,j}\}$, recovering n secret messages $\{s_i\}$.

cryptions scenario. This is based on Theorem 3.2.5 (strict).

However, since threshold decryptions are performed in this scenario, the modification is not necessary. A loose batch ZK proof-verification technique is sufficient with a final check at the end. This is based on Theorem 3.2.7.

For $j \in \{1, 2, \ldots, m\}$, suppose a threshold decryption authority (the prover) has a private key share x_j and the corresponding public verification key v_j. For $i \in \{1, 2, \ldots, n\}$, there are n ciphertexts $c_i = (\alpha_i, \beta_i)$. The threshold decryption authority compute his/her corresponding partial decryptions of $z_{i,j} = \alpha_i^{x_j}$. Figure 3.6 offers an illustration of a threshold decryption scenario. For simplicity, we present a batch technique for one threshold decryption authority (the prover). The index $j = 1$ is omitted from our description, except to distinguish the different private key share x_j, its corresponding public verification key v_j, and the partial decryption of a particular authority $z_{i,j}$.

According to Theorem 3.2.7, verification of correct threshold decryptions of ElGamal ciphertexts (from one decryption authority - the prover) can be batched using SE test using Chaum-Pedersen ZK proof of equality of discrete logarithm (Appendix C.3) as:

$$\log_g(v_j) = \log_{\prod_{i=1}^{n} \alpha_i^{t_i}} (\prod_{i=1}^{n} z_{i,j}^{t_i}) \tag{3.7}$$

The interactive batch ZK proof-verification protocol for this scenario is shown in Figure 3.7. This protocol can be made non-interactive as in the centralised decryption scenario (fixing the small exponents).

Another method to produce the small exponents by cooperation of the threshold decryption authorities is as below (we use the index $j \in \{1, 2, \ldots, m\}$ to distinguish the different authorities in the description below). This is also shown in

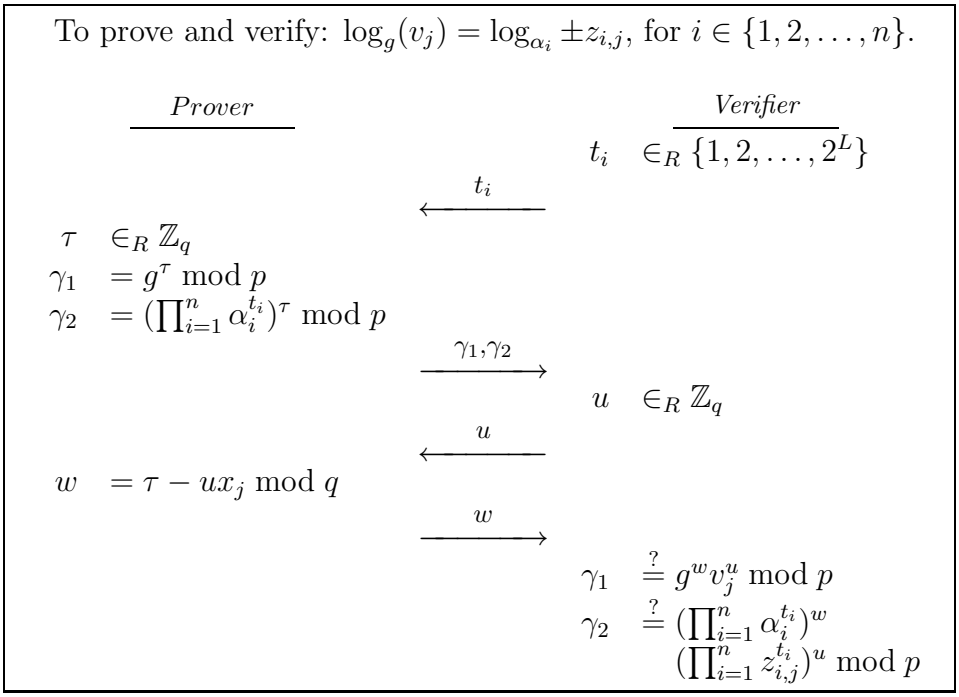

Figure 3.7: A batch ZK proof-verification technique for verifying valid threshold decryptions of ElGamal ciphertexts.

Figure 3.8.

1. For $j \in \{1, 2, \ldots, m\}$, each authority (prover) P_j selects the initial small exponents $t'_j \in \{1, 2, \ldots, 2^L\}$ at random. Using a suitable commitment function, the small exponents are committed and published, e.g. using a hash function with a range of $\{0, 1\}^L$ as $\{H(t'_j)\}$.

2. Each authority P_j then produces and publishes their partial decryptions $z_{i,j} = \alpha_{i,j}^{x_j}$.

3. The initial small exponents published in the first step is then revealed by publishing them.

4. The small exponents t_i are then calculated using a collision-resistant hash function as $t_i = H(\{t'_j, \alpha_{i,j}\}, i)$.

Note that the use of digital signature on the published values is required to authenticate them. Non-interactively each threshold decryption authority uses the same small exponents t_i as opposed to using different small exponents values $t_{i,j}$ provided by the verifier for each authority in the interactive version.

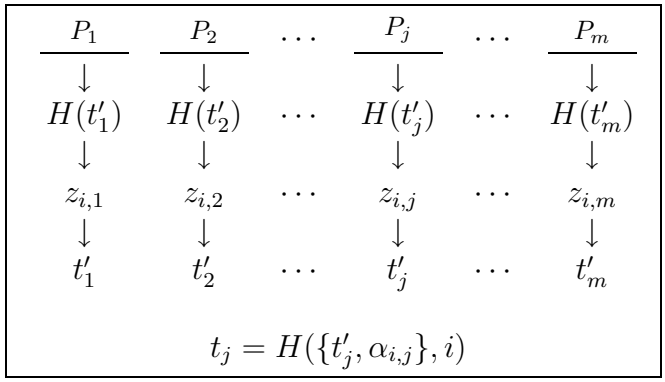

Figure 3.8: A technique to produce small exponents non-interactively.

This technique (loose) does not completely guarantee correct threshold decryptions. Namely, unless $z_{i,j} = \pm\alpha_i^{x_j}$, the batch verification can only be passed with negligible probability. Thus, the batch verification result is not yet satisfactory as the recovered secret message may be incorrect: $\pm s_i \in G$. To fix this, the decryption requires one extra step, i.e. multiplying s_i with (-1) when $s_i \notin G$. After s_i is recovered through the combining (of partial decryptions) procedure, we test if $\left(\frac{s_i}{p}\right) = 1$ (using the Legendre symbol). If it is accepted, $s_i \in G$. Otherwise, $s_i = -s_i \bmod p$. The additional cost is only one exponentiation per ciphertext. This batch technique is sufficient in some applications where there are many threshold decryption authorities (provers).

Paillier Cryptosystem (Threshold)

Damgård and Jurik [DJ00] improved the threshold version of Paillier cryptosystem by Fouque *et al.* [FPS00]. This scheme is recalled in Appendix B.2.2.

For $j \in \{1, 2, \ldots, m\}$, suppose a threshold decryption authority (the prover) has a private key share x_j and the corresponding public verification key v_j. For $i \in \{1, 2, \ldots, n\}$, there are n ciphertexts c_i. The threshold decryption authority compute his/her corresponding partial decryptions of $z_{i,j} = c^{2\Delta x_j}$, where $\Delta = m!$. Theorem 3.2.5 (strict) is suitable to batch verify threshold decryptions for Paillier cryptosystem as:

1. For threshold version of Paillier, G is a set of quadratic residues in $\mathbb{Z}^*_{N^2}$ with order $MN = pqp'q'$, the smallest factor of which is $min(p, q, p', q')$.

2. The value of v is trusted to be a generator of squares in $\mathbb{Z}^*_{N^2}$. As v_j is

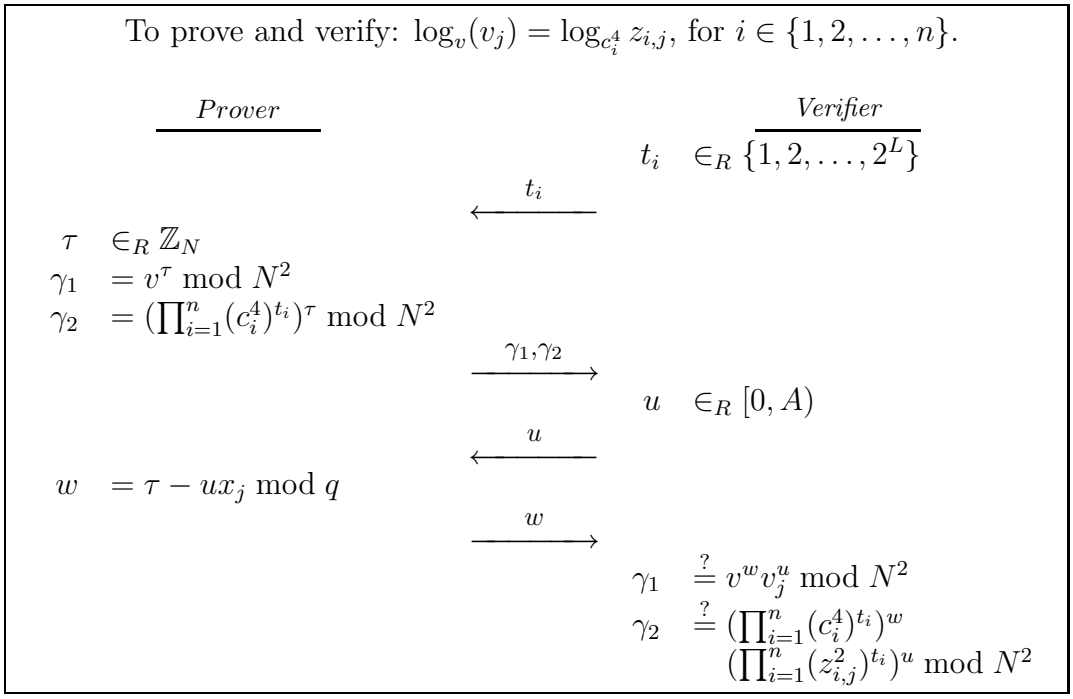

Figure 3.9: A batch ZK proof-verification technique for verifying valid threshold decryptions of Paillier ciphertexts.

produced using v, and $z_{i,j}^2$ are explicitly squared by the verifier, thus $v_j, z_{i,j} \in G$ (a set of quadratic residues in $\mathbb{Z}_{N^2}^*$).

3. The values of c_i^4 is a square, and v is trusted to be squares in $\mathbb{Z}_{N^2}^*$ chosen by the trusted dealer. Therefore, both $c_i^4, v \in G$. Thus, c_i^4 and v are generators of G with a very large probability ($\frac{\text{ord}(MN)}{MN}$).

4. The small exponents t_i can be chosen at random while satisfying $t_i < 2^L < min(p, q, p', q')$.

According to Theorem 3.2.5, verification of correct threshold decryptions of Paillier ciphertexts (from one decryption authority - the prover) can be batched using SE test using Chaum-Pedersen ZK proof of equality of discrete logarithm (Appendix C.3) as:

$$\log_v(v_j) = \log_{\prod_{i=1}^n (c_i^4)^{t_i}} (\prod_{i=1}^n (z_{i,j}^2)^{t_i}) \tag{3.8}$$

The interactive batch ZK proof-verification protocol for this scenario is shown in Figure 3.9. Where $A \times ord(G)$ is much smaller than N, the challenge u must be chosen in $[0, A)$ such that the shared secret key x_i is statistically hidden in the

response w as in [PS99, BFP+01]. Analysis in [PS99] suggests the minimum size of the challenge A to be 80 bits, and 128 bits for more secure applications.

Using a hash function and employing the well-known Fiat-Shamir heuristic, the protocol is made non-interactive. The prover produces the small exponents t_j, and challenge u very similar to the non-interactive protocol explained in the previous (threshold decryptions for ElGamal cryptosystem) subsection.

Unlike in the previous subsection, extra verification to ensure that partial decryptions passing the batch verification are not $-z_{i,j}$ is not necessary. This is because partial decryptions $z_{i,j}$ are explicitly squared in the share combining phase to reconstruct the secret message.

According to Theorem 3.2.5, the above batch ZK proof-verification technique guarantees that $\log_v(v_j) = \log_{c_i^4} z_{i,j}$. Equivalently, partial decryptions $z_{i,j}$ are computed using the corresponding shared private key x_j.

3.4 Analysis

This section offers security and efficiency analysis of the theorems and techniques presented in this thesis.

3.4.1 Security

It is straight-forward that each of the theorems and techniques in this chapter is **complete**. This is because if the batch verification equations in the batch ZK proof-verification protocols are correct, they output positive results.

Batch theorems in this chapter are applied using standard three-move ZK proof-verification protocols, known as Σ-protocol [CD95]. Hence, the techniques are **correct, specially sound, and honest-verifier ZK**. The proof of soundness for the batching operation is proven in the respective theorems. This is straight-forward for batch re-encryptions techniques.

The batch techniques for centralised decryptions and threshold decryptions are very similar. They are based on Chaum-Pedersen's zero-knowledge proof of equality of discrete logarithms protocol. We slightly modify the protocol where the verifier randomly selects the small exponents at the beginning of the protocol run. We discuss the soundness of these particular techniques below. A short discussion on error probability is also provided at the end of this subsection.

Soundness

The proof of soundness for the batch techniques for centralised decryptions and threshold decryptions follows from Chaum-Pedersen's scheme as they are essentially the same. The small exponents t_i are chosen randomly in a very similar manner ($t_i < 2^L$) to choosing the random challenge.

Given the same random small exponents and commitments, no matter which challenge is chosen, the prover reveals no other information than the fact that the discrete logarithms of the verification key to the base of verification base equals the discrete logarithms of the product of the decryption shares to the base of the product of the ciphertexts (Equation 3.6, 3.7, 3.8).

In the interactive version, the probability for a prover to cheat is negligible. It is not feasible to forge the decryption shares where the verification is accepted without the knowledge of the share decryption key. Also, where the prover indeed holds the decryption key share, the probability of producing bad decryption shares where the verification is accepted is also negligible. This is because the small exponents and challenge are chosen randomly by the verifier. For example, in batching the verification of correct ElGamal decryption shares, the probability of a prover guessing a correct random small exponent and challenge, and the verification is accepted is $2^{-L}C^{-1}$ (where C denotes the choice of possible challenge selected - accordingly to each batch technique).

In the non-interactive version, we also follow Chaum-Pedersen's protocol with a slight addition in choosing the random small exponents based on the coin-flipping protocol. On one version of the non-interactive protocol, a hash function can be used with the input (s'_i or $z_{i,j}$, accordingly) chosen by a single prover to compute the small exponents if it is guaranteed that the ciphertexts are not received prior to the commitment phase.

Otherwise, we avoid the use of a hash function with the input (s'_i or $z_{i,j}$, accordingly) chosen by a single prover to compute the small exponents. This is because it might be possible for a dishonest participant to try fixing the corresponding values (s'_i or $z_{i,j}$, accordingly) and produce the small exponents, such that the verification is accepted and the share combining fails. A distributed source of randomness (based on the coin-flipping protocol) is required as the small exponents are only of length L, where L is small.

The probability of a prover forging the corresponding values (s'_i or $z_{i,j}$, accordingly) and fixing the small exponent share is negligible. This is because the

prover is required to commit to the random share first before publishing the corresponding values, and the small exponents are produced by hashing the combined random shares (common reference string) of all the participants. As a collision-resistant hash function is used to produce the small exponents, a prover can only attempt to forge his decryption share if all the participants collude.

The rest of the protocol is a Σ-protocol [CD95], and thus has a special soundness property as proven in [CD95]. The proof of soundness for the batching operation is proven in the respective theorems.

Error probability

In any of the batch techniques presented, the probability that a dishonest authority (prover) is discovered is overwhelmingly large for the following reasons.

- As indicated in the theorems, the probability that the batch verification equation is satisfied given incorrect parameters is 2^{-L}.
- As the prover has to guess the challenge u at random, the probability that the batch verification test is accepted where the batch verification equation is not satisfied is C^{-1}, where C denotes the choice of possible challenge selected - accordingly to each batch technique.
- Therefore, the probability that the batch verification is not accepted given incorrect share decryption is $(1 - 2^{-L})(1 - C^{-1})$.

As C^{-1} is very small, e.g. 2^{-1024}, the probability that a dishonest participant being undetected given incorrect share decryption(s) is approximately 2^{-L}.

3.4.2 Efficiency

Efficiency improvements using the batch theorems are summarised in Table 3.2 based on the number of modular multiplications required. In the table, *CBL* stands for common-base logarithm and *CEL* stands for common-exponent logarithm. The figures in the table show that all the applications of the batch theorems greatly reduce the computational cost required compared to when batch is not applied.

We follow Bellare *et al.* [BGR98] in measuring the cost of our algorithms, where $ExpCost^n(\log_2 q)$ denotes the number of modular multiplications required

Table 3.2: Efficiency improvement for applications using the batch theorems.

	Verification (Theorem)	Computational cost without batch	Computational cost with batch
Prover	Equality of CBL (strict)	$2nExpCost(\log_2 ord(G)) + n$	$2nExpCost(\log_2 ord(G)) + n + 1$
	Equality of CBL (loose)	$2nExpCost(\log_2 q) + n$	$2ExpCost(\log_2 q) + n + 1$
	Equality of CEL (strict)	$2nExpCost(\log_2 ord(G)) + n$	$2nExpCost(\log_2 ord(G)) + ExpCost^n(L) + 1$
	Equality of CEL (loose)	$2nExpCost(\log_2 q) + n$	$2nExpCost(\log_2 q) + ExpCost^n(L) + 1$
	Knowledge of root	$2nExpCost(\log_2 N) + n$	$2ExpCost(\log_2 N) + ExpCost^n(L) + 1$
Verifier	Equality of CBL (strict)	$4nExpCost(\log_2 ord(G)) + 2n$	$4ExpCost(\log_2 ord(G)) + 2ExpCost^n(L) + 2$
	Equality of CBL (loose)	$4nExpCost(\log_2 q) + 2n$	$4ExpCost(\log_2 q) + 2ExpCost^n(L) + 2$
	Equality of CEL (strict)	$4nExpCost(\log_2 ord(G)) + 2n$	$4ExpCost(\log_2 ord(G)) + 2ExpCost^n(L) + 1$
	Equality of CEL (loose)	$4nExpCost(\log_2 q) + 2n$	$4ExpCost(\log_2 q) + 2ExpCost^n(L) + 1$
	Knowledge of root	$2nExpCost(\log_2 N) + n$	$2ExpCost(\log_2 N) + ExpCost^n(L) + 1$

to compute n exponentiations in a common base with different exponents of the same bit-length of q.

From Table 3.2, the batch technique cost one additional modular exponentiation for proving equality of common base logarithms on both the strict and loose versions. However, for the rest of the operations shown in the table, it is clear that the batch technique offers an efficiency improvement. The cost of n modular exponentiations without batch (third column in the table) is replaced by $ExpCost^n(L) + 1$ modular exponentiations by using batch (fourth column in the table).

Batching offers a performance increase in many cryptographic protocols. The increase is proportional to the number of instances batched. The more instances batched, the efficiency improvement increases accordingly.

3.5 Summary

Five new batch theorems are presented in this chapter. They are more complex and versatile than realised in previous works [BGR98, HAK01, SK95]. Issues concerning group memberships in previous works are efficiently addressed in the batch theorems and their corresponding applications.

The theorems are extended and applied into batch techniques for batching zero-knowledge (ZK) proof-verification protocols related to typical operations in ElGamal and Paillier cryptosystems. These operations are encryptions and decryptions.

As these operations are widely applied in cryptographic schemes, the techniques developed are beneficial to improve the efficiency of such schemes. Detailed applications of the batch theorems into the corresponding batch techniques are provided. It should be straight-forward to modify existing schemes to employ our batch techniques and greatly increase their efficiency.

The techniques were developed with specific applications to cryptographic voting schemes. This is introduced at the beginning of this chapter, and is shown in later chapters, especially in Chapter 5. As described in Chapter 2.8, homomorphic-encryption and mix-network are two main frameworks in cryptographic voting protocol. Two mix-network schemes employing the batch techniques are presented in Chapter 5.

The next chapter discusses the use of homomorphic encryption in cryptographic voting protocol.

Chapter 4

Homomorphic Encryption based Voting

In a cryptographic voting protocol, homomorphic encryption is one of the fundamental primitives employed to satisfy the privacy requirement. That is, the voter-vote relationships must be kept private. A particular vote cast must not be linkable to its corresponding vote. The voting result is revealed, while each individual ballot is never opened. A particular voter can only be linked to a particular ballot, where the vote contained in the ballot is kept private to the voter. This is discussed in Chapter 2.8.

This chapter offers research in homomorphic encryption based voting. Background information for homomorphic voting is presented. A generic description of the homomorphism property in a cryptosystem is provided. Two brief examples of homomorphic encryption schemes and their use in voting are recalled.

We present a new voting scheme based on a multiplicative homomorphism property. It is an alternative to the commonly used additive homomorphic voting protocols with similar security and a comparable (or better) efficiency. The multiplicative homomorphic voting scheme has been previously published in [PAB+04a].

A preferential voting system case study using a homomorphic encryption scheme is also discussed in this chapter. The Australian House of Representatives described in Chapter 2.3.1 is used as a reference for the preferential voting system. This case study has been previously published in [ABDV03].

4.1 Background

Homomorphic voting schemes are efficient when the number of candidates or choices is small. However, homomorphic voting has a drawback where each vote must be verified to be valid. With no vote validity check, correctness of the tallying cannot be guaranteed. When the number of candidates or choices is large (e.g. in a preferential voting), computational and communicational cost for the proof and verification of vote validity is so high that homomorphic voting becomes less efficient than mix voting. Hence, it is widely believed that homomorphic voting is only suitable for elections with a small number of candidates or choices (e.g. "YES/NO" voting).

This section contains background information for this chapter. The concept of homomorphic encryption is explained. Two examples of additive homomorphic cryptosystems popularly used in the literature - ElGamal and Paillier - are recalled.

4.1.1 Homomorphic Encryption

Homomorphic encryption is generically defined as below.

Definition 2. *Let E be a public encryption function, s a secret message, and $D()$ the corresponding function. A cryptosystem has a homomorphism property when $E(s_1) \odot E(s_2) = E(s_1 \oplus s_2)$, where $\odot$ and $\oplus$ are some binary operators. This is also known as* ***homomorphic encryption****.*

If we observe the left hand side of the above equation, it is evident that the ciphertexts of individual messages can be combined using a binary operator, $\odot$. The right hand side of the equation suggests that such a combining operation will result in another ciphertext, the decryption of which will result in a combination of the individual votes, $s_1 \oplus s_2$. Note that the binary operators $\odot$ and $\oplus$ may be equal. Thus, it will be possible to compute the combination of the individual messages without having to retrieve the individual messages themselves. Thereby, the individual messages can remain confidential.

For such a conclusion to be valid, the following assumption should be satisfied.

Assumption 1. *If the decryption authority can decrypt the combination ciphertext $E(s_1 \oplus s_2)$, then the decryption authority can also decrypt the individual ciphertexts, namely $E(s_1)$ and $E(s_2)$. If the decryption authority decrypts the*

individual ciphertexts, then the confidentiality of the individual message is compromised. Therefore, the decryption authority must be trusted not to decrypt the individual ciphertexts.

Such an assumption can be satisfied by designing a threshold decryption function (refer to Appendix B), which can be informally defined as follows:

Definition 3. *The ability to compute D is distributed among a set of decryption authorities such that a threshold of the decryption authorities can decrypt any ciphertext by co-operating with each other. Therefore, a threshold of the decryption authorities are trusted not to co-operate in the decryption of the individual ciphertexts.*

In cryptographic voting protocols, the additive homomorphism property is commonly used in the literature. By using a cryptosystem with such a property, a combination of encrypted votes (ballots) yields accumulation of votes. The voting result is then obtained from the accumulation of votes, while no individual ballot is opened and the corresponding individual vote remains a secret.

4.1.2 A Modified ElGamal Cryptosystem

The ElGamal cryptosystem (see Appendix B.1) has a natural multiplicative homomorphism property. For $i \in \{1, 2, \ldots, n\}$, n ciphertexts are combined as the following equation.

$$\prod_{i=1}^{n} c_i = \left(\left(\prod_{i=1}^{n} \alpha_i \right), \left(\prod_{i=1}^{n} \beta_i \right) \right) \tag{4.1}$$

The corresponding decryption yields a combination of the individual messages as the following equation, where x denotes the private decryption key.

$$\prod_{i=1}^{n} s_i = \frac{\prod_{i=1}^{n} \beta_i}{\left(\prod_{i=1}^{n} \alpha_i\right)^x} \tag{4.2}$$

Where D denotes a corresponding decryption function, the above equation can also be written as $\prod_{i=1}^{n} D(c_i) = D(\prod_{i=1}^{n} c_i)$.

For the ElGamal cryptosystem to produce an accumulation of the messages (additive homomorphism) instead, the message structure for encryption is slightly modified. A common generator g_1 for the group G is published. A message s is encrypted using a value $r \in \mathbb{Z}_q$ selected at random, using g_1, as $c = (\alpha, \beta) =$

$(g^r, g_1^s y^r)$, where $y = g^x$ and g is also a public generator of the group G. This property was used in the scheme by Cramer *et al.* [CFSY96].

After the modification, combination of the ciphertexts yields accumulation of the individual messages. The ElGamal cryptosystem is now additive homomorphic. For $i \in \{1, 2, \ldots, n\}$, n ciphertexts are combined as in the original ElGamal cryptosystem (Equation 4.1). The corresponding decryption now yields an accumulation of the individual messages using a discrete logarithm (DL) search as the following equation, where DL denotes a discrete logarithm search function.

$$\begin{aligned} \mathrm{DL}\left(\frac{\prod_{i=1}^{n}\beta_i}{\left(\prod_{i=1}^{n}\alpha_i\right)^x}\right) &= \mathrm{DL}\left(\prod_{i=1}^{n} g_1^{s_i}\right) \\ &= \mathrm{DL}\left(g_1^{\sum_{i=1}^{n} s_i}\right) \\ &= \sum_{i=1}^{n} s_i \end{aligned}$$

Recovering the accumulation of individual messages requires a discrete logarithm (DL) search. The search may be computationally expensive if the number of ciphertexts is large. The cost of such a search may be lowered by lowering the number of ciphertexts to be decrypted. Instead of decrypting a combination of the entire ciphertexts at once, the ciphertexts can be grouped into smaller number for decryptions.

A very simple "YES/NO" voting scenario using this setting (this is the scenario in [CFSY96], although extensions to a 1-out-of-K voting scenario are possible) is as follows. Let 1 denote a "YES" vote, and -1 denote a "NO" vote. Each voter encrypts either a 1 or a -1, and submits the corresponding ballots and a non-interactive zero-knowledge proof of correct ballot construction (i.e. the ballot must only contain either a 1 or a -1) by proving the predicate $\log_g(\alpha) = \log_y(\beta) \vee \log_g(\alpha) = \log_y(-\beta)$ (later we show a more complex protocol in Figure 4.2).

The proofs are verified, and valid ballots are combined. Decryption of the valid ballots (with the DL search) yields either a positive (more 1 than -1 votes) or negative number (more -1 than 1 votes). If the result is a positive, than "YES" wins, otherwise "NO" wins.

Paillier cryptosystem, on the other hand, has a natural additive homomorphism property. A voting scheme based on this cryptosystem is recalled in the

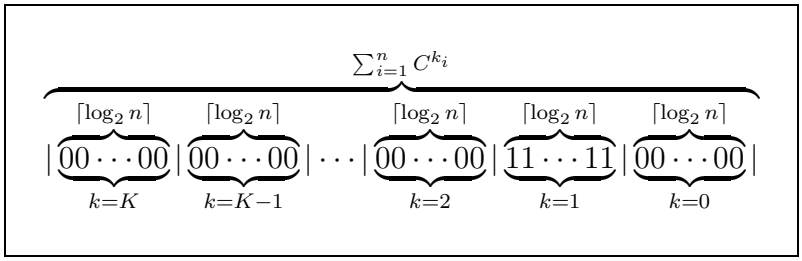

Figure 4.1: A pictorial representation of the accumulation of homomorphic votes, where all voters selects the same vote of C^1, namely $k_i = 1$.

next subsection.

4.1.3 The Scheme by Baudron *et al.*

The Paillier cryptosystem (refer to Appendix B.2) has a natural additive homomorphism property. Decryption of a combination of ciphertexts yields an accumulation of the messages. It is very straight-forward to adapt the cryptosystem for a voting system.

Baudron *et al.* [BFP+01] proposed a novel technique for the design of cryptographic voting protocols. They assumed a 1-out-of-K voting system and presented arguments for the efficiency of their proposal. The message structure is modified as to allow a summation register for each candidate or choice.

Let n be the maximum number of voters in a particular constituency or district. The voting authorities compute a system constant $C = 2^{\lceil \log_2 n \rceil}$. If there are K candidates or choices in voting, the authorities assign a unique number, $k \in \{1, 2, \ldots, K!\}$, to every candidate or choice.

The rest of the protocol follows from the original Paillier cryptosystem (after the modification to the message structure). The vote to be encrypted is of the form C^k, where k represents the candidate or choice selected. The value s' is obtained after the decryption of a combination of ballots. The value s' represents a concatenation of K bit strings of length $\log_2 C = \lceil \log_2 n \rceil$. Each bit string counts the number of voters who voted for a value $k \in \{1, 2, \ldots, K\}$.

For simplicity, let $\log_2 C = \log_2 n$. Figure 4.1 presents a pictorial representation of the accumulation of votes, where all voters choose for the first candidate or choice, namely, $k = 1$. For $i \in \{1, 2, \ldots, n\}$, it is shown that the summation register for $k = 1$ accommodates all the votes, where k_i denotes a candidate or choice selection of the i^{th} voter.

Baudron *et al.* use three zero-knowledge proof of knowledge protocols in their scheme. We provide brief descriptions for these protocols as below. More detail on zero-knowledge protocols is provided in Appendix C.

1. **Proof of knowledge of an encrypted message**: The voter employs this protocol to prove that he knows the vote that is encrypted in a particular vote ciphertext. This proof requires the voter (prover) to compute four exponentiations and the voting authorities (verifier) to compute three exponentiations.

2. **Proof that an encrypted message lies in a given set of messages**: The voter employs this protocol to prove that he has encrypted a valid vote. For a 1-out-of-K voting system, Q is the set of valid votes of size K. This proof requires the voter (prover) and the voting authorities to compute $3K$ exponentiations each.

3. **Proof of equality of plaintexts**: Baudron *et al.* expected the voter to encrypt the same vote to three different sets of officials (local, regional, and national). The voters were expected to prove such a fact.

4.2 Multiplicative Homomorphic Voting

Current homomorphic encryption based voting schemes exploit an additive homomorphism property in the underlying cryptosystem (e.g. Paillier cryptosystem [Pai99]). A single threshold decryption is performed over the combination of encrypted votes (ballots) to recover the accumulation of the votes and reveal the total number of votes for any candidate or choice. Using this method, vote privacy is achieved since no single vote is decrypted (Table 2.2 in Chapter 2.8).

We discover that a multiplicative homomorphism property in the underlying cryptosystem (e.g. ElGamal cryptosystem [ElG85]) can also be exploited to construct an alternative homomorphic encryption tallying with comparable (or better) efficiency without decrypting any individual vote.

A multiplicative homomorphic encryption algorithm (e.g. ElGamal cryptosystem) is employed to produce a ballot and a single decryption is performed over the combination of ballots to obtain the product of the votes. Afterward, this product is factorised and the voting result is revealed.

As in the additive homomorphic voting, no single vote is decrypted in multiplicative homomorphic voting. Thus, vote privacy is also preserved. The most important advantage of multiplicative homomorphic voting is that the scheme is more efficient than additive homomorphic voting and more efficient than other voting schemes when the number of candidates is small. Multiplicative homomorphic voting offers an alternative homomorphic encryption tallying with comparable (or better) efficiency without compromising vote privacy.

A disadvantage of additive homomorphic voting compared to multiplicative homomorphic voting is inefficiency due to the following reasons.

- If the Paillier cryptosystem is employed, the following drawbacks in efficiency exist.
  - Inefficient set-up:
    In voting schemes, the private key of the encryption algorithm must be generated and shared by multiple tally authorities, such that it is not required to trust any single party to achieve vote privacy. As the private key is a factorisation secret in the Paillier cryptosystem, distributed key generation is highly inefficient. In comparison, distributed key generation in the ElGamal cryptosystem (distributed generation of a secret logarithm as the private key) is much more efficient as described in [Fel87, Ped91, GJKR99].
  - Multiple encryption:
    Typically, a voter has to perform an encryption for each candidate or choice, and prove that each of his encryption contains a valid message (candidate or choice).
  - Inefficiency of multiplicative and exponentiation computations:
    In the Paillier cryptosystem, each multiplication is performed modulo N^2, where N is the product of two large primes and its factorization is the private key (see [Pai99] or Appendix B.2). In comparison, each multiplication is performed modulo a large prime p in the original ElGamal cryptosystem. If the same encryption strength is required, N and p should have the same length (e.g. 1024 bits). Although the Chinese Remainder Theorem can be employed to improve the efficiency of multiplicative computation with a composite modulus in Paillier cryptosystem, Paillier admitted that this efficiency improvement is only

available for key generation and decryption when the factorisation of N is known. Paillier indicated that a multiplication in his scheme is more than three times as costly as a multiplication in ElGamal cryptosystem when N and p are of the same length (e.g. 1024 bits). Typically, threshold decryption is employed in voting schemes to minimise trust and strengthen robustness. Hence, the factorisation of N is not known to any single tally authority who performs the decryption. Therefore, we can assume that when the same security strength is required, a multiplication in the Paillier cryptosystem with threshold decryption is at least three times as costly as a multiplication in the ElGamal cryptosystem.

- If the modified ElGamal cryptosystem is employed, the following drawbacks in efficiency exist.
  - Multiple encryption:
    Usually, a voter has to perform an encryption for each candidate and prove that each encryption contains a valid message.
  - Inefficient discrete logarithm (DL) search:
    As previously stated, a search for logarithm is required in the decryption function. Even though the (currently known) most efficient solution for DL in a certain interval - the Pollard Lambda method - is employed, $0.5 \log_2 n$ exponentiations, $\mathcal{O}(n^{0.5})$ multiplications, and $\mathcal{O}(0.5 \log_2 n)$ storage are required for n number of voters. As the number of voters is often large in voting applications (e.g. in a national election scenario), this is of high cost. For a more efficient search, the votes may be divided into multiple groups. A separate tallying is performed in each group. However, this division increases the number of decryptions. A separate decryption is required for every candidate or choice in each group.

In [LK00, LK02], the modified ElGamal encryption and its additive homomorphism are exploited in a very special manner. Only one encryption is required in a vote, which is composed of several sections, each responding to a candidate. Thus, only one decryption is required to decrypt the product of all the ballots. Although the number of encryptions and decryptions are slightly reduced, they are not the main computational burden in a voting

scheme. The main computational cost contributing to the overall computational cost in a voting scheme is from the cost for zero-knowledge proof of correct ballot construction and the cost of the DL search. This cost increases to $\mathcal{O}(Kn^{Km-1})$ multiplications and $\mathcal{O}(n^{K-1})$ full-length (e.g. 1024 bits) storage space, where K denotes the total number of candidates or choices. As the number of voters is often large in a voting applications, the cost for the search is intolerable. Therefore, the special additive homomorphic tallying in [LK00, LK02] may actually deteriorate efficiency although they were supposed to improve efficiency.

Our multiplicative homomorphic voting inherits an efficient distributed key generation, requires only one encryption per vote and requires no brute-force search. At the same time, it achieves vote privacy not weaker than that of additive homomorphic voting.

This section presents a design of a multiplicative homomorphic voting scheme. Security and efficiency analysis of the new scheme are also provided in this section.

4.2.1 The Scheme

A multiplicative homomorphic voting scheme exploits a multiplicative homomorphism property of the underlying cryptosystem. The cryptosystem is used to produce a ballot (an encrypted vote) and to efficiently tally all the ballots without revealing any individual votes. Each voter only needs to encrypt one value as his/her vote using the multiplicative homomorphic scheme.

Generically, a cryptosystem is multiplicative homomorphic if the following equation holds $E(s_1 s_2) = E(s_1)E(s_2)$, where E denotes a corresponding encryption function for the cryptosystem, and s_1 and s_2 are two secret messages. When the combination (product) of the secret messages is not greater than the multiplicative modulus, decryption of the corresponding combination of ciphertexts yields the combination of secret messages.

Using the ElGamal cryptosystem, the above multiplication of two encrypted messages corresponds to: $(\alpha_1, \beta_1)(\alpha_2, \beta_2) = (\alpha_1\alpha_2 \bmod p, \beta_1\beta_2 \bmod p)$.

In voting, the the secret message s_i denotes a vote, $c_i = E(s_i)$ denotes a ballot, and $i \in \{1, 2, \ldots, n\}$ where n denotes the number of voters. Using multiplicative homomorphic voting, decryption of the product of ballots reveals the product of votes. This is if their product is not over the multiplicative modulus. A certain mechanism is used to guarantee this assumption (using groups as detailed below).

Then, the product is factorised to recover the tally of votes. Where K denotes the number of candidates (or set of choices), a 1-out-of-K multiplicative voting protocol is as follows:

1. **Preparation phase**: For $j \in \{1, 2, \ldots, m\}$, each tally (decryption) authority cooperates to generate the parameters of ElGamal cryptosystem, such that each holds a shared secret key x_j, and publish their corresponding verification key $v_j = g^{x_j}$. The public key g and y are published. The encryption modulo is p. Detail on threshold ElGamal cryptosystem is provided in Appendix B.1.2.

   Suppose there are K candidates or choices. A set $Q = \{q_k\}$ is selected, where $k = \{1, 2, \ldots, K\}$. Each of the element in the set Q represents each candidate or choice, and has the following two properties:

   (a) the element must be a co-prime to the modulo p, such that the multiplication of the elements can be factorised; and

   (b) the element must be a quadratic residue, in the group G.

   To achieve the above properties, the set Q is generated as follows:

   (a) The set $Q = \{1\}$ is initialised. An integer k', representing the current size of the set, is initialised as $k' = 1$. The index k is also initialised to be 1.

   (b) The k^{th} smallest prime p_k is tested as follows:
   If $p_k^q = 1 \bmod p$, then

   - p_k is a quadratic residue; and
   - p_k is placed into the set Q and k' is incremented by 1.

   Otherwise, if $p_k^q \neq 1 \bmod p$, then p_k is a quadratic non-residue and discarded.

   (c) If $k' < K$, then the index k is incremented by 1, and we go back to the previous step to choose another small prime p_k.

   (d) Otherwise, if $k' = K$, then the set Q is generated.

   From the properties of an element in the set Q, an ElGamal encryption of such an element is indistinguishable.

2. **Voting phase**: Each voter chooses a vote, represented by an element from the set Q. The voter constructs a ballot, by selecting a value r_i at random from $\mathbb{Z}_q$ and encrypting the vote s_i as $c_i = E(s_i) = (\alpha_i, \beta_i) = (g^{r_i}, s_i y^{r_i})$. Afterward, the voter construct a non-interactive zero-knowledge (ZK) proof of correct ballot construction without revealing the encrypted vote. Both the ballot and its corresponding ZK proof are made public.

   The proof convinces a verifier (public) that the encrypted message s_i in the ballot c_i corresponds to an element in the set Q. This is shown in Figure 4.2. For simplicity, we show the proof construction for a voter ($i = 1$) and remove the subscript i except to distinguish α_i, β_i, s_i, and r_i in the figure.

   The proof is based on the ZK 1-out-of-K proof of knowledge [CDS94], extending the ZK proof of equality of discrete logarithms [CP93], and employing the Fiat-Shamir heuristic [FS86] (non-interactive). Zero-knowledge protocols are described further in Appendix C.

3. **Tally phase**: Tally authorities can verify the validity of ballots by checking the proofs (Figure 4.2). For $k \in \{1, 2, \ldots, K\}$, verification of the above proof can be performed by checking the digital signature $\text{sig}(u)$ and the following equations: $u \stackrel{?}{=} H(g, y, \alpha_i, \beta_i, \{\gamma_{k,1}\}, \{\gamma_{k,2}\})$; $u \stackrel{?}{=} \sum_{k=1}^{K} u_k$; $\gamma_{k,1} \stackrel{?}{=} g^{w_k} \alpha_i^{u_k}$; and $\gamma_{k,2} \stackrel{?}{=} y^{w_k} (\beta_i / q_k)^{u_k}$. If all the tests are accepted, then we are convinced that the vote in the ballot is indeed one of the elements in the set Q with a very large probability.

   Valid ballots c_i (for simplicity, assume all n ballots are valid where $i \in \{1, 2, \ldots, n\}$) are then divided at random into groups of size n', such that $max(Q)^{n'} < p$. The function $max(Q)$ outputs the largest element in the set Q. If $max(Q)^n < p$, then the division is not necessary and all the ballots can be processed in one group.

   In each group, the following multiplicative homomorphic tallying is performed.

   (a) Let $c'_1, c'_2, \ldots, c'_{n'}$ be the ballots in the group.

   (b) The tally authorities cooperate to calculate $s' = D(\prod_{i=1}^{n'} c'_i)$, where D denotes an appropriate threshold decryption function (refer to Appendix B.1.2).

To prove:
$(\log_g \alpha_i = \log_y(\beta_i/q_1)) \vee (\log_g \alpha_i = \log_y(\beta_i/q_2)) \vee \cdots \vee (\log_g \alpha_i = \log_y(\beta_i/q_K))$.

(a) Where $1 \leq k' \leq K$ and $k \in \{1, 2, \ldots, K\} \setminus k'$, the voter chooses the vote $s_i = q'_k$. Values of $\tau_{k'}, \{\tau_k, u_k, w_k\} \in \mathbb{Z}_q$ are selected at random. Both generator g and y are committed to two witnesses as below.

$$\begin{aligned} \gamma_{k',1} &= g^{\tau_{k'}} \\ \gamma_{k',2} &= y^{\tau_{k'}} \end{aligned}$$

The voter then simulates the rest of the witnesses as below.

$$\begin{aligned} \{\gamma_{k,1} &= g^{w_k} \alpha_i^{u_k}\} \\ \{\gamma_{k,2} &= y^{w_k} (\beta_i/q_k)^{u_k}\} \end{aligned}$$

(b) The voter produces a random challenge u, using a collision-resistant hash function H with a range of $\mathbb{Z}_q$, where $k \in \{1, 2, \ldots, K\}$ as below.

$$u = H(g, y, \alpha_i, \beta_i, \{\gamma_{k,1}\}, \{\gamma_{k,2}\} \tag{4.3}$$

The challenge is also digitally signed by the voter as $\text{sig}(u)$ to prove the authenticity of the challenge. The signature can also be used to prevent double voting (non-reusability, refer back to Chapter 2.7).

(c) The voter computes the challenge $u_{k'}$ and the response $w_{k'}$ as below.

$$\begin{aligned} u_{k'} &= u - \sum_{k \in \{1,2,\ldots,K\} \setminus k'} u_k \\ w_{k'} &= \tau_{k'} - u_{k'} r_i \end{aligned}$$

(d) For $k \in \{1, 2, \ldots, K\}$, the values of $\{\gamma_{k,1}, \gamma_{k,2}, u_k, w_k\}$, u, and $\text{sig}(u)$ are published.

Figure 4.2: A non-interactive ZK proof of correct ballot construction.

(c) The value of s' is factorised[1] as $s' = \prod_{k=1}^{K-1} q_j^{\hat{s}_j}$. The number of votes in this group for the k^{th} candidate or choice is $\hat{s}_{k-1}$ for $k = 2, 3, \ldots, K$. The number of votes in this group for the first candidate or choice is $K - \sum_{k=1}^{K-1} \hat{s}_k$.

Tally authorities combine the results in all the groups to obtain a final voting result.

4.2.2 Analysis

This section offers security and efficiency analysis of the multiplicative homomorphic voting scheme.

Security

The following theorem guarantees correctness of our scheme.

Theorem 4.2.1. *The multiplicative homomorphic tallying in each group of ballots $c'_1, c'_2, \ldots, c'_{n'}$ is correct.*

Proof (Theorem 4.2.1). Let D denotes a corresponding decryption function for a given ciphertext. In multiplicative homomorphic tallying of a group containing ballots $c'_1, c'_2, \ldots, c'_{n'}$, the following holds where $\prod_{k=1}^{K-1} q_k^{\hat{s}_k}$ is a factorization of s':

$$\begin{aligned} D\left(\prod_{i=1}^{n'} c'_i\right) &= s' \\ &= \prod_{k=1}^{K-1} q_k^{\hat{s}_k} \end{aligned}$$

As encryption in the ElGamal cryptosystem is multiplicative homomorphic, the following decryptions equality holds:

$$D\left(\prod_{i=1}^{n'} c'_i\right) = \prod_{i=1}^{n'} D(c'_i) \bmod p$$

When the ballots are divided into groups, it is guaranteed that $max(Q)^{n'} < p$.

[1] This factorisation is very efficient as each element in the set Q is very small.

Thus, $\prod_{i=1}^{n'} D(c'_i) < p$. Therefore, the following holds:

$$\begin{aligned}\prod_{i=1}^{n'} D(c'_i) &= D\left(\prod_{i=1}^{n'} c'_i\right) \\ &= \prod_{k=1}^{K-1} q_k^{\hat{s}_k}\end{aligned}$$

For $i \in \{1, 2, \ldots, n'\}$, $D(c'_i)$ are verified to be in the set Q at the beginning of the tally stage, $\prod_{i=1}^{n'} D(c'_i)$ is also a factorization of s'.

As there is a unique factorization for any integer, $\prod_{i=1}^{n'} D(c'_i)$ and $\prod_{k=1}^{K-1} q_k^{\hat{s}_k}$ are the same factorization. Namely, each prime factor in $\prod_{i=1}^{n'} D(c'_i)$ is also a prime factor in $\prod_{k=1}^{K-1} q_k^{\hat{s}_k}$. Each prime factor in $\prod_{k=1}^{K-1} q_k^{\hat{s}_k}$ is also a prime factor in $\prod_{i=1}^{n'} D(c'_i)$.

Therefore, all the non-one votes encrypted in ballots $c'_1, c'_2, \ldots, c'_{n'}$ and only these votes (contained in the ballots) are prime factors in $\prod_{k=1}^{K-1} q_k^{\hat{s}_k}$. Hence, every non-one vote is correctly recovered.

As the number of vote in each group is a constant n', the number of "1" votes is also correctly recovered if there are any. □

The following offers a discussion that *the multiplicative homomorphic tallying does not reveal any individual vote.*

- **Indistinguishability**: The use of ElGamal encryption is semantically secure due to the choice of message space in the set Q. Elements in the set Q are guaranteed to be either all quadratic residues or all non-quadratic residues, where the value $p = 2q + 1$ is a strong prime (a parameter of the ElGamal cryptosystem, see Appendix B.1). Without the private key to decrypt the ballots, it is difficult to obtain any information about any vote.

- **Private key (decryption) security**: As the private key is protected by a threshold key sharing mechanism (Appendix B.1.2), no individual ballot is decrypted if a threshold trust on the tally authorities is assumed.

- **Unlinkability**: The only decryptions performed are decryptions of combinations of ballots. The resulting decryptions only reveal the product of votes (in each group). This offers unlinkability to each vote, since a decryption does not link a particular vote to its corresponding voter. The

revealed information (from the decryptions) says no more than that a particular voter in each group may have cast a vote in that group. Note that this also depends on the size of each group and variations of votes cast (i.e. hiding in a crowd).

- **The group size is sufficiently large for strong vote privacy**: As homomorphic tallying is only applied to elections with a small number of candidates, K and $max(Q)$ are small[2]. As p is large (e.g. with a length of 1024 bits), the size of a group $\lceil \log_{max(Q)} p \rceil$ is large compared to K where $\lceil x \rceil$ denotes the smallest integer no smaller than a real number x. For example, when $K = 2$ and $\log_2 p = 1024$ (p is of 1024 bit size), we obtain: $Q = \{1, 2\}$ (for simplicity, assuming 2 is a quadratic residue), $max(Q) = 2$, and the group size is larger than 1024. When there are only two candidates (small variation of votes) and more than 1024 votes combined in each group (a vote is hidden inside a large group), very strong vote privacy is achieved.

Every operation in the voting scheme is publicly verifiable. Note that public proofs of correct ballot construction and correct decryptions are provided by the voters and tally authorities respectively.

Efficiency

The computational cost of additive homomorphic voting employing Paillier encryption and that of the proposed multiplicative homomorphic voting are listed in Table 4.1. As the DL search in the decryption of the modified ElGamal encryption in [HK04, KY02, LK00, LK02] is too inefficient[3] (both in computation and space requirement), computational cost for the modified ElGamal encryption is omitted from the comparison table.

As only small primes are employed to represent the candidates or choices, the computational cost for the final factorisation in multiplicative homomorphic voting is negligible compared to full-length exponentiation.

[2]The value of $max(Q)$ is no larger than the $(2K-1)^{th}$ smallest prime, which is several times of K when K is small.

[3]Although some computation in the Pollard Lambda method can be pre-computed, pre-computation can be employed in most voting schemes. For example, the exponentiation computation in ballot construction and all the computation in the proof of correct ballot construction (if necessary) can be pre-computed in mix-network voting, Paillier-based additive homomorphic voting, and multiplicative homomorphic voting.

Table 4.1: A computational cost comparison of the two types of homomorphic voting.

Operation	Homomorphic voting	
	Additive	Multiplicative
Distributed key generation	highly inefficient	efficient
Encryption per vote	$6K$	2
Vote validity proof per vote	$12K + 6$	$4K - 2$
Vote validity verification per vote	$12K + 6$	$4K$
Tallying computation per tallier	$9K$ or[4] $9(K-1)$	$3\lceil n \log_p max(Q) \rceil$

To make a precise efficiency comparison of the two types of homomorphic voting, it is supposed that the same strength of encryption security is required in both types of homomorphic voting. That is, N in Paillier cryptosystem for additive homomorphic voting and p in ElGamal cryptosystem for multiplicative homomorphic voting have the same length. Thus, an exponentiation computation in ElGamal cryptosystem is counted as one standard exponentiation, and an exponentiation computation in Paillier cryptosystem is counted as three standard exponentiations.

Standard exponentiations are counted in every operation in Table 4.1. This table shows that multiplicative homomorphic voting is always more efficient than additive homomorphic voting in key generation, vote encryption and ballot validity check.

When the number of voters is not too large, multiplicative homomorphic voting is also more efficient than additive homomorphic voting in tallying. For example, when $K = 2$, $\log_2 p = 1024$ and $n = 1024$, the required number of standard exponentiations for tallying in additive homomorphic voting is 12 or 6, while the required number of standard exponentiations for tallying in multiplicative homomorphic voting is three. Even if multiplicative homomorphic tallying is less efficient than additive homomorphic tallying when the number of voters is large, it has a trivial influence on the total cost of the voting scheme as illustrated in Table 4.1. It is assumed that the additive homomorphic voting (no existing example is referred to in this section) employs a threshold Paillier cryptosystem and performs every necessary operation listed in the table.

[4]It is often assumed that a decryption is necessary for every candidate or choice. However, when n, the total number of voters, is known and each vote has been verified to be valid, $K-1$

Table 4.2: An efficiency comparison of MV, AHV, and MHV.

	Key generation	A voter's computation	A tallier's computation	Communicational cost
MV	efficient	8	$18n$ $= 18000000$	$1024(6n + 18mn)$ $= 98304000000$
AHV[5]	highly inefficient	$18K + 6$ $= 42$	$(12K + 6)n$ $+9(K - 1)$ $= 30000009$	$2048(6m(K - 1)$ $+(10K + 4)n)$ $= 49152061440$
MHV	efficient	$4K$ $= 8$	$4Kn$ $+3\lceil n \log_p max(Q)\rceil$ $= 8003000$	$1024(2n(m + 1)$ $+3\lceil n \log_p max(Q)\rceil)$ $= 6147072000$

A more comprehensive efficiency comparison is presented in Table 4.2. The computational and communicational cost of MV (mix-network based voting), AHV (additive homomorphic voting), and the proposed MHV (multiplicative homomorphic voting) are presented in the table. Mix-network based voting is presented in Chapter 5 in this thesis.

In the table, m denotes the number of tally authorities. For simplicity, voters' signature on the votes are omitted. Therefore, signature generations and verifications are not taken into account. In this comparison, it is supposed that Golle's mix network [GZB+02] (one of the most efficient mix-network schemes) with the tally authorities as mix servers are employed in the mix-network based voting, threshold Paillier cryptosystem is employed for the additive homomorphic voting, and ElGamal cryptosystem is employed for the multiplicative homomorphic voting (our scheme). The computational cost is measured in terms of the number of standard exponentiations required, and the communicational cost is measured in terms of the size of the communicated messages in bits.

An example scenario is given in Table 4.2, where $m = 5, K = 2, \log_2 p = 1024$ and $n = 1000000$. For simplicity, it is assumed that 2 is a quadratic residue modulo p, hence $Q = \{1, 2\}$. In this example, it is shown that even when the number of voters is large, multiplicative homomorphic voting is still more efficient than mix-network based voting and additive homomorphic voting.

decryption is sufficient. The tally authorities choose $K - 1$ candidates or choices at random, and decrypt the accumulation of votes for each of them. The vote of the left candidate is n minus the sum of the votes for the $K - 1$ chosen candidates. We call this economical tallying.

[5]It is assumed economical tallying in Table 4.1 is employed.

The above tables apply to previous suggested implementations of each scheme. However, our suggestion for using relatively small groups of voters can also be applied to additive homomorphic voting using ElGamal cryptosystem. In this case, most of the savings shown for multiplicative homomorphic voting also apply.

4.3 A Preferential Voting Case Study

We propose the following straight-forward adaptation of the scheme by Baudron *et al* [BFP+01] (refer back to Section 4.1.3) to design a cryptographic preferential voting protocol. A preferential voting system is described in Chapter 2.3.1.

The zero-knowledge proof of equality of plaintexts is not applicable in our scheme (as used in the original scheme) as it requires the voters to encrypt the vote only for a set of local authorities (constituency). Note that such a modification will not adversely affect the security of the system because the threshold decryption function of Paillier cryptosystem requires every tally authority to prove correct decryption operations.

In this new system, the voter is expected to vote for a particular sequence of candidates rather than to vote for the candidates themselves, as was proposed in the original scheme. Hence, this scheme can also be called a 1-out-of-$K!$ voting scheme.

1. **Preparation phase**: In this discussion, we assume that the voter must provide a rank for every candidate or choice. The size of the vote in this cryptosystem is $\log_2 C^{K!}$ bits. That is, each sequence is represented by a counter that can count up to C. Figure 4.3 presents a pictorial representation of the preferential vote (C^k), which chooses the first sequence of candidates, namely, $k = 1$.

   The voting authorities choose and publish a public key for a Paillier cryptosystem (Appendix B.2), which can be used by the voters to communicate their votes confidentially to the authorities, for each constituency or district. The modulus, N, must be chosen such that the entire vote can be encrypted in one block, namely $C^{K!} < N^2$. Since the size of N increases exponentially with the increase in the number of candidates, this scheme would be impractical when $K > 5$ for a polling booth with 1000 voters.

2. **Voting phase**: Each voter i performs the following.

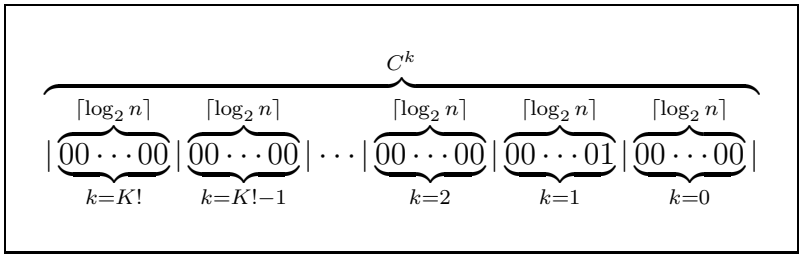

Figure 4.3: A pictorial representation of the homomorphic preferential vote C^k, where $k = 1$

(a) identifies to the vote collecting authority appropriately;

(b) selects a sequence, k_i, to represent his/her preference;

(c) constructs a ballot by encrypting the selection as $c_i = E(C^{k_i})$ using the Paillier cryptosystem, where E is the corresponding encryption function for Paillier cryptosystem (this operation costs 2 exponentiations);

(d) proves knowledge of encrypted message using the aforementioned proof technique (refer back to Section 4.1, this operation costs 4 exponentiations);

(e) proves that the encrypted message lies in the set $\{C, C^2, \ldots, C^{K!}\}$ (this operation costs $3(K!)$ exponentiations).

This phase requires each voter to compute $2 + 4 + 3(K!) = 3(K!) + 6$ exponentiations.

3. **Tally phase**: The vote collecting authority performs the following.

(a) verifies the two proofs generated by every voter;

(b) forwards the validated votes to the tally authorities.

Note that this simple system does not provide privacy for voters who do not cast a proper vote. In order to do so, a dummy vote must be encoded for use in the system. That is, $k \in \{1, 2, \ldots, K!, (K!)+1\}$, where $(K!)+1^{th}$ vote would represent a dummy vote. The vote collecting authority must perform $(3(K!) + 4)$ exponentiations in n parallel runs.

Afterward, the tally authority performs the following.

(a) combine the ballots as $c = \prod_{i=1}^{n} c_i = \prod_{i=1}^{n} E(C^{k_i}) = E(\sum_{i=1}^{n} C^{k_i} \bmod N^2)$ (for simplicity, assume all ballots are valid);

(b) decrypt the combined ciphertext c in a threshold fashion to obtain $s' = \sum_{i=1}^{n} C^{k_i}$.

Note that s' represents a concatenation of $K!$ bit strings of length $\log_2 C = \lceil \log_2 n \rceil$. Each bit string counts the number of voters who voted for a sequence $k \in \{1, 2, \ldots, K!\}$. The number of exponentiations that each tally authority must perform is three. To verify correct decryption operation, four exponentiations per tally authority is required. If the number of tally authorities who took part in the decryption procedure is m, then the verification of this step would require $4m$ exponentiations.

The Australian House of Representatives is used as a reference for the preferential voting system (refer back to Chapter 2.3.1). The average case figures for this system are 100000 voters and 20 candidates ($K = 20$) per constituency. We partition the constituency into polling booths containing 1000 voters ($n = 1000$) each for efficiency reasons.

Table 4.3 summarises the complexity of the above protocol, when $K = 20$ and $n = 1000$, in terms of the number of exponentiations and parallel processes each entity must perform. Two processes are said to be in parallel when the input of each process is not dependent on the output of the other process.

A vote for such a preferential voting system requires a size of at least $\log_2(K!)$ bits, where K denotes the number of candidates or choices. Clearly, the size of the vote for a K-candidate preferential voting strategy will be much greater than the corresponding vote for a 1-out-of-K voting strategy when K increases. Hence, the voting scheme proposed by Baudron *et al.* (or any other homomorphic voting schemes) is highly impractical for a *straight-forward adaptation* of the elections of the Australian House of Representatives. This also applies for an election scenario for the Australian Senate, especially since there is more candidates for senate elections (average case: $K = 60$). It is information theoretically impossible to reduce or compress the ballot size in this (straight-forward) scenario since a preferential system is of factorial order.

However, it is possible to use this approach for a preferential system when a small number of pre-set preferences is allowed, tallying of the pre-set preferences uses the homomorphic encryption approach, while tallying of the non pre-set preferences uses a mix-network approach.

Table 4.3: The computational complexity for the adapted voting system using homomorphic encryption.

Entity	**Parallel processes**	**Exponentiations per process**
Voter	1	$3(K!) + 6$ $= 14597412049059840000$
Vote collecting authority	$n = 1000$	$3(K!) + 4$ $= 14597412049059839998$
Tally authorities	1	3

This is a promising framework inheriting benefits from both approaches, and is further discussed in Chapter 7. We name this a *hybrid* approach.

4.4 Summary

A new homomorphic encryption based voting protocol is presented in this chapter. It uses a multiplicative homomorphism property, as compared to the commonly used additive homomorphism property in voting. The new scheme provides an alternative, with comparable security and efficiency, to other existing homomorphic encryption based voting protocols.

From the preferential voting case study, the size of a vote for a preferential voting system is inherently larger than a 1-out-of-K voting system. In preferential voting, the vote size is at least $\log_2 K!$ bits to accommodate all the available preferences. Thus, the size increases in a factorial order when K increases. On the other hand, the vote size only increases linearly in a 1-out-of-K voting system when K increases.

Also, any voting system that employs the homomorphic encryption approach require the voter to prove that a valid vote was encrypted in the ballot. This is such that the tally obtained from individual vote is correct. The computational complexity of such a proof is $\mathcal{O}(K!)$.

Straight-forward implementation of preferential voting system using this approach is not possible. Unless an efficient proof technique is available to validate the encrypted votes, the homomorphic encryption approach is not practical for a straight-forward adaptation of a preferential voting system.

However, it is possible to combine the homomorphic encryption approach with

a mix-network approach (later described in Chapter 5) into a *hybrid* approach when a small number of pre-set preferences is used. This promising approach is further discussed in Chapter 7.

Voting protocols that employ the mix-network approach do not require a complex proof of correct ballot construction. In fact, the computational complexity of the mix-network is not adversely affected by K. Therefore, mix-network based voting protocols are ideally suited for a straight-forward adaptation of preferential voting systems.

The next chapter provides a more detailed study on mix-networks and their use in cryptographic voting protocols.

Chapter 5

Mix-Network based Voting

Mix-networks are an important tool to implement anonymity. They are widely employed in many cryptographic applications such as anonymous email, electronic auction, and electronic voting. The first mix-network scheme was proposed by Chaum [Cha81] mainly to realise anonymous email. Various types and usage of mix-networks have been proposed subsequently. Although they are of different design, they share some common properties.

In cryptographic voting protocols, voters submit encrypted votes (ballots) as inputs to the mix-network. The mix-network outputs anonymised plaintext votes corresponding to those in the input ballots. The voting result is then obtained from the plaintext votes. Using a mix-network in a voting scenario, voter-vote relationships are kept private to each voter.

This chapter contains study on mix-networks. Background information for a mix-network is presented. A generic description of a mix-network is provided. Two mix-network schemes by Abe [Abe99, AH01] and Golle *et al.* [GZB$^+$02] are recalled.

A new mix-network scheme extending from the work by Abe is presented. As Abe uses binary gates in his mix-network construction, we use extended binary mixing gates (EBMGs) in our mix-network construction. This is made efficient by using batch techniques from Chapter 3. The new scheme has been previously published in [PAB$^+$04b].

A modification to the mix-network scheme of Golle *et al.* is provided. The modification allows a more efficient scheme, while having a trace-back mecha-

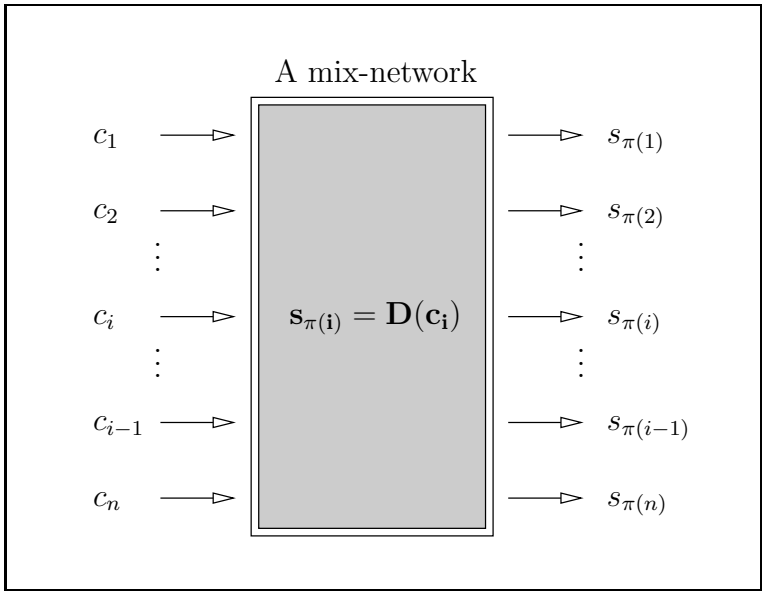

Figure 5.1: An illustration of a mix-network with n inputs.

nism to identify dishonest entities. This work has been previously published in [ALBD04a]. The scheme is later used in Chapter 6.

Batch theorems and techniques from Chapter 3 are used to offer example applications of batching in a mix-network.

A preferential voting system case study using a mix-network scheme is also discussed in this chapter. This case study continues from the one presented in Chapter 4.3. The case study has been previously published in [ABDV03].

5.1 Background

A mix-network is typically composed of a few mix servers, each in charge of a shuffling. Suppose a number of users are to use the mix-network to achieve anonymity. Each of them encrypts his/her input and submits it to the mix network. Each server shuffles the inputs sequentially.

The shuffling operation of each server on its inputs includes two steps. The first step is to process the inputs, which may be through either re-encryption or decryption. Afterward, the inputs are then permuted in the second step. Finally the outputs of the mix-network are published in plaintext.

It is required that outputs of a mix-network is a permutation of the users' inputs, but are anonymous and unlinkable to the users. A typical mix-network is illustrated in Figure 5.1.

A mix-network is generically defined as below.

Definition 4. *Let D be a decryption algorithm (corresponding to its input ciphertext) computable only by a mix-network, and $\pi : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ be a secret permutation function selected at random. For $i \in \{1, 2, \ldots, n\}$, the mixing operation of the mix-network can be described (outputs) as $s_{\pi(i)} = D(c_i)$.*

The left hand side of the equation (see Figure 5.1) is a random sorted set of (output) plaintexts $\{s_{\pi(i)}\}$. These (output) plaintexts correspond to the right hand side, decryptions D of a set of (input) ciphertexts $\{c_i\}$. Thus, a mix-network can be viewed as a confidentiality translation service, translating the confidentiality service from the input ciphertexts to the identity of each ciphertext owner. A mix-network anonymises its outputs plaintexts given a set of input ciphertexts, or hides its input-output relationships.

To achieve a stronger level of anonymity, a mix-network typically consists of a number of mix-servers. Each mix-server shuffles a set of inputs, and produces a permuted set of outputs. When at least one of the mix-server holds its permutation secret, input-output relationships are also kept secret.

According to the processing of inputs performed by each server, mix-networks in the literature are typically classified into those (schemes) employing a decryption chain [Cha81, JJ01, OA00, PIK93] and those (schemes) employing a re-encryption chain [GZB$^+$02, JJR02, BG02, OKST97, SK95, Abe99, AH01, FS01, Nef01, Gro03, PBDV04].

In a **decryption chain** mix-network (the original scheme by Chaum), the shuffling of a mix server is composed of decryption and permutation. Its characterisation is as follows:

- Each input is encrypted with every mix server's public key in sequence.
- Each mix server removes one layer of encryption on all the inputs by performing decryption using the private key of the server. Afterward, the inputs are permuted and forwarded to the next server.
- Outputs of the last mix server are plaintexts of the original inputs of the first server.

In a **re-encryption mix-network**, the shuffling of a server is composed of re-encryption and permutation. Its characterisation is as follows:

- Each input is encrypted only once with a public encryption key while the corresponding private key is shared by several decryption authorities.
- Each mix server re-encrypts, permutes, and forwards all the inputs to the next mix server.
- The decryption authorities (can be the mix servers themselves) cooperate to decrypt the final encrypted outputs.

In the former type (decryption chain), each input is sequentially encrypted for each of the mix server to decrypt. Consequently, failure of any mix server implies that the corresponding plaintexts cannot be recovered if each mix server holds its own private key secret (as required to achieve strong privacy). Therefore [OKST97], decryption chain mix-networks inherently lack robustness. Hence, we only consider re-encryption chain mix-networks employing threshold decryption in this thesis.

5.1.1 Re-Encryption Chain Mix-Networks

Ogata *et al.* [OKST97] introduced a basic structure for re-encryption chain mix-networks illustrated in Figure 5.2. Their scheme was further developed in many later papers.

Suppose an ElGamal cryptosystem with threshold decryption is employed using the scheme by Pedersen [Ped92]. Decryption authorities employ a distributed key generation scheme [Fel87, Ped91, GJKR99] to generate a private key x, which is shared by themselves. The public key is $(g, y = g^x)$.

For $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$, m servers SV_j form a mix-network to mix n ciphertext inputs c_i. Inputs to SV_j are $c_{j-1,i}$, while $c_{0,i} = c_i$. On server SV_j, inputs $c_{j-1,i} = (\alpha_{j-1,i}, \beta_{j-1,i})$ are re-encrypted and permuted (shuffled) to $c_{j,\pi_j(i)} = (\alpha_{j,\pi_j(i)}, \beta_{j,\pi_j(i)}) = (g^{r_{j,i}}\alpha_{j-1,i}, y^{r_{j,i}}\beta_{j-1,i})$, where the values of $r_{j,i}$ are selected at random from $\mathbb{Z}_1$, and π_j is a secret random permutation of $[1, n]$. The outputs of SV_j are $c_{j,i}$, while the final outputs $c'_i = c_{m,i}$.

The shuffling correctness, from n inputs to n outputs on every server, must be verified. This is discussed on the next subsection. Finally, a quorum of the decryption authorities (e.g. the mix servers themselves) cooperate to decrypt c'_i.

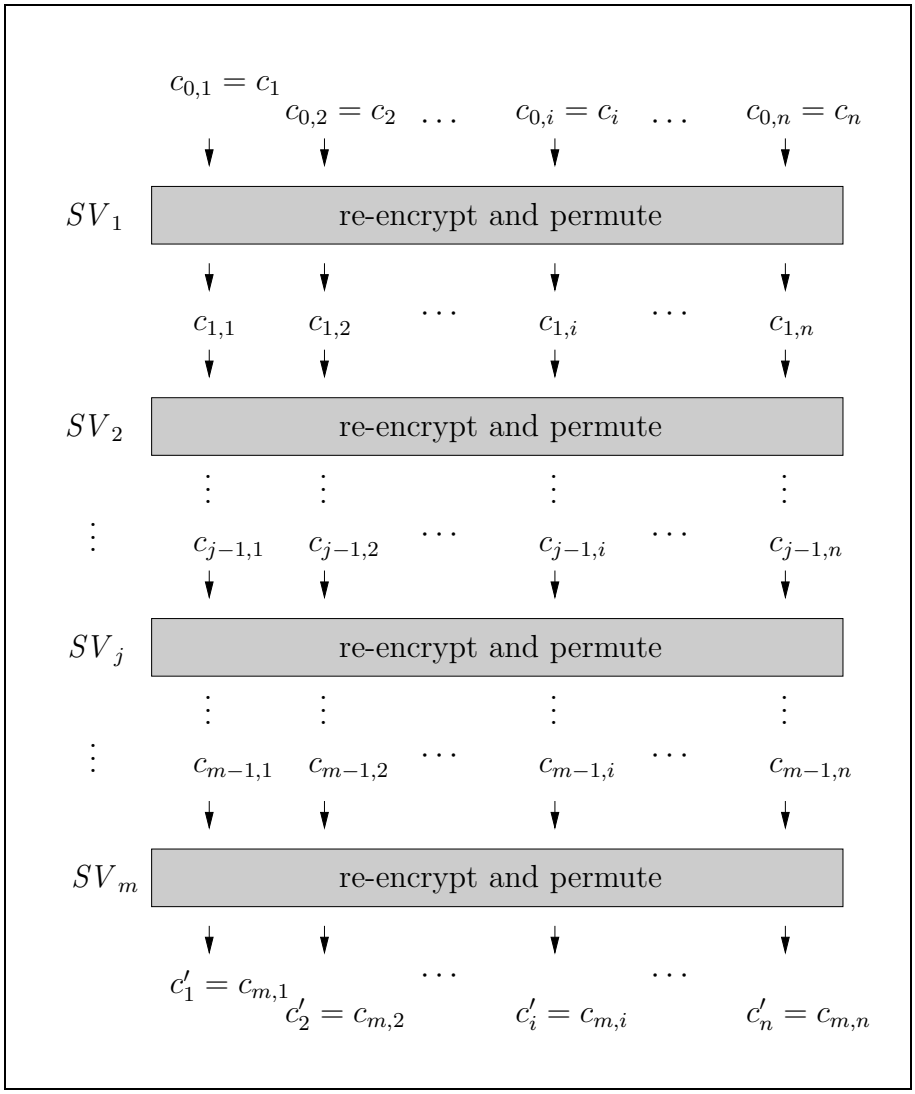

Figure 5.2: A basic structure for re-encryption mix-networks introduced by Ogata *et al.*

5.1.2 Verification of Correct Mixing Operations

Many of the early mix-network schemes offer security and robustness in exchange for efficiency. Proof-verification of correct mixing and decryption operations are often time consuming. Some recent schemes improve on efficiency by sacrificing the verification of correct mixing and decryption operations. According to the different correctness verification mechanisms, mix-networks can be classified into three categories:

1. **No verification**: In this category, correctness is not verified and the mix servers are trusted to perform the shuffling correctly. An example of this

category is the scheme by Ohkubo and Abe [OA00]. Strong trust is necessary in such mix-networks.

2. **Global mix verification**: Mix-networks in this category do not provide a verification of correct shuffling by each mix server separately. Instead, mixing correctness on the entire mix-network is verified after the final plaintext outputs are produced. Schemes in this category include [Cha81, PIK93, VBD00, GZB+02]. Drawbacks to this approach include:

   (a) A cheating mix server cannot be identified instantly.

   (b) If an incorrect shuffling is found, a mix-network in the third category must be employed to perform another mixing.

   (c) Some outputs may be revealed in plaintexts even when the shuffling is incorrect and another mixing is required.

3. **Individual mix server verification**: In this category [SK95, JJ01, Abe99, AH01, FS01, OKST97, JJR02, BG02, Nef01, Gro03, PBDV04], each mix server first verifies the correctness of the previous server's shuffling. Then, it shuffles its inputs, proves the correctness of its own shuffling, and forwards the outputs to the next mix server. Although schemes in the first two categories are more efficient, schemes in this category are still very useful as;

   (a) they overcome the shortcomings of the first two categories.

   (b) they form a necessary sub-function (to handle anomalous situations when cheating in the shuffling is found) in the second category.

   However, there exist various problems with schemes in this category. For example, the scheme by Juels and Jakobsson [JJ01] is not publicly verifiable; some schemes [JJR02, BG02] do not provide sufficient correctness and privacy guarantee for many applications; and other schemes [Abe99, AH01, OKST97, SK95] are inefficient.

   Three recently proposed mix-network schemes [FS01, Nef01, Gro03] offer great improvement in this category. However, these three schemes are still not optimally efficient for large-scale applications (e.g. a national election) as their computational cost is linear to the number of inputs.

In the third category, the scheme by Abe [Abe99] offers an essential efficiency improvement technique over naive mix-network schemes. For $l \in \{1, 2, \ldots, n!\}$, suppose $\pi_{j,l}$ represents all $n!$ possible permutations for π_j. For $i \in \{1, 2, \ldots, n\}$, a naive method to verify correctness of shuffling by SV_j is to test the following assertion:

$$\begin{aligned}\left(\log_g \frac{\alpha_{j,\pi_{j,1}(i)}}{\alpha_{j-1,i}} = \log_y \frac{\beta_{j,\pi_{j,1}(i)}}{\beta_{j-1,i}}\right) \quad &\vee \quad \left(\log_g \frac{\alpha_{j,\pi_{j,2}(i)}}{\alpha_{j-1,i}} = \log_y \frac{\beta_{j,\pi_{j,2}(i)}}{\beta_{j-1,i}}\right) \vee \\ \cdots \qquad\qquad &\vee \quad \left(\log_g \frac{\alpha_{j,\pi_{j,n!}(i)}}{\alpha_{j-1,i}} = \log_y \frac{\beta_{j,\pi_{j,n!}(i)}}{\beta_{j-1,i}}\right)\end{aligned}$$

A proof for the above assertion can be implemented using zero-knowledge proof of n-out-of-$n \times n!$ equality of discrete logarithms. The proof is based on the zero-knowledge proof of partial knowledge by Cramer *et al.* [CDS94] (refer to Appendix C),This type of verification allows proving of correctness without compromising privacy.

However, this test is too inefficient because the computational cost for both the prover and verifier on every mix server is $\mathcal{O}(n \cdot n!)$ exponentiations. Abe made an efficiency improvement on this test by dividing an n-input-to-n-output shuffling into a number of 2-input-to-2-output shuffling. Correctness verification in each of the 2-input-to-2-output shuffling is much more efficient, where the cost of the correctness verification in all the of 2-input-to-2-output shuffling is now $\mathcal{O}(n \log_2 n)$. However, Abe's schemes are still not efficient enough for many applications.

Based on the proof of valid shuffling, an alternative classification to mix-network schemes can be presented as below.

- Permutation-based: this method proves that some valid permutation is used in the mix-network which may be any permutation of $\{1, 2, \ldots, n\}$, where n is the number of inputs to the mix-network. Examples of this method include mix-network scheme by Groth [Gro03], and Golle *et al.* [GZB$^+$02]. The computational cost of the proof and the corresponding verification is high if no batch technique (refer back to Chapter 3) is employed.

- Input-based: this method proves that each input is shuffled to any output. An example of this method is the mix-network by Peng *et al.* [PBDV04]. It is more efficient, but its privacy and soundness are weaker.

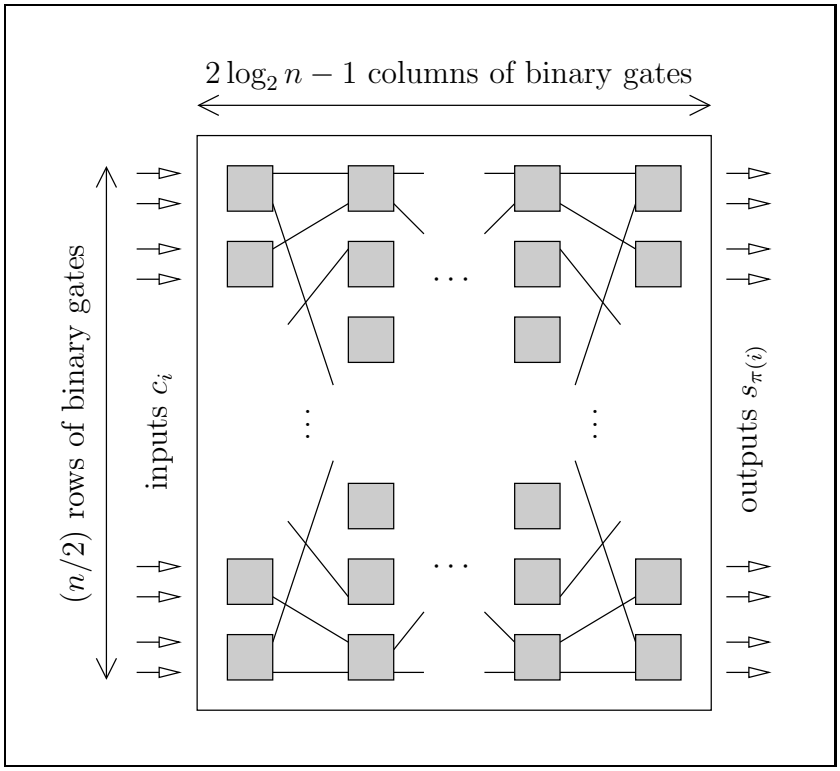

Figure 5.3: An example mix-network by Abe with n inputs.

In this mix-network area, our research is focused on re-encryption chain mix-networks with permutation-based proof of valid shuffling. This is because re-encryption chain mix-networks offer better robustness compared to decryption-chain mix-networks, and permutation-based mix-networks offer stronger privacy and soundness compared to input-based mix-networks.

5.1.3 The Mix-Network Scheme by Abe

Abe [Abe99, AH01] proposed an efficient and robust mix-network scheme. For $i \in \{1, 2, \ldots, n\}$, it uses the ElGamal cryptosystem (see Appendix B.1) in re-encrypting the set of input ciphertexts $\{c_i\}$, to produce a randomly permuted set of output plaintexts $\{s_{\pi(i)}\}$, where π denotes a secret permutation function. Figure 5.3 illustrates the mix-network by Abe constructed from a number of mixing gates.

Gates with two inputs and two outputs construct the mix-network. We call this a *binary gate*. The binary gate provides an efficient zero-knowledge proof-verification of correct shuffling at each gate for proof of correct mixing. The following provides a formal description of the shuffling operation in each gate as

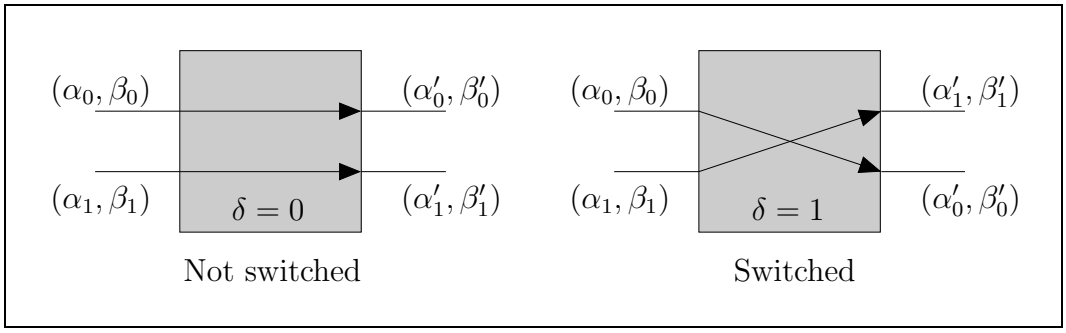

Figure 5.4: A shuffling operation inside a mixing gate.

shown in Figure 5.4. Select randomiser values of $r_0, r_1 \in \mathbb{Z}$ at random, and a switching value of $\delta \in \{0, 1\}$ at random. The secret permutation π is determined by the value of δ, which determines outputs of a gate to either be switched or not switched (see Figure 5.4). Compute the following as outputs of the shuffle.

$$\begin{aligned}(\alpha'_\delta, \beta'_\delta) &= (\alpha_0 g^{r_0}, \beta_0 y^{r_0}) \\ (\alpha'_{\bar{\delta}}, \beta'_{\bar{\delta}}) &= (\alpha_1 g^{r_1}, \beta_1 y^{r_1})\end{aligned}$$

A shuffling requires 4 modular exponentiations (for the re-encryptions). Each gate then proves in zero-knowledge (i.e. without revealing their secret switching value δ), that the shuffling is correct (refer to Appendix C), where:

$$\log_g\left(\frac{\alpha'_0}{\alpha_0}\right) = \log_y\left(\frac{\beta'_0}{\beta_0}\right) \quad \wedge \quad \log_g\left(\frac{\alpha'_1}{\alpha_1}\right) = \log_y\left(\frac{\beta'_1}{\beta_1}\right)$$
$$\text{OR}$$
$$\log_g\left(\frac{\alpha'_0}{\alpha_1}\right) = \log_y\left(\frac{\beta'_0}{\beta_1}\right) \quad \wedge \quad \log_g\left(\frac{\alpha'_1}{\alpha_0}\right) = \log_y\left(\frac{\beta'_1}{\beta_0}\right)$$

This zero-knowledge proof-verification costs 12 modular exponentiations to compute the proof and 16 modular exponentiations for verification. The total cost for shuffling is shown in Table 5.1.

There are two similar schemes proposed by Abe, namely *MiP-1* and *MiP-2*, briefly described as follows:

MiP-1 is a one-phase scheme of randomised decryption. Each gate in the mix-network performs a threshold decryption and mixing operation of the inputs. Proofs of correct threshold decryptions and shuffling operation are produced at each gate. This is to ensure that outputs and inputs may differ only in their ordering, and that the outputs are not modified or replaced in

Table 5.1: Computational cost at each gate for MiP-2.

Type of operation	Modular exponentiations required
Main (re-encryptions)	4
Proof construction	12
Verification	16
Total	32

any way from the inputs. The final outputs are a mixed (permuted) set of plaintext messages.

MiP-2 is a two-phase scheme of permutation and decryption. Each gate in the mix-network randomly permutes the ordering of the two input ciphertexts. A proof of correct shuffling operation is also produced to ensure the integrity of the output ciphertexts. Threshold decryptions are performed after the gates are finished mixing the set of input ciphertexts. This is to produce the corresponding mixed (permuted) set of plaintext outputs.

The robust mix-network of Abe offers efficiency in computation and communication of complexity $\mathcal{O}(tn \log_2 n)$, where t is the number of tolerable corrupt mix servers and m is the number of input ciphertexts. There are m servers in the mix-network, each controlling a number of gates. The mix-network employs a (t, m) threshold decryption scheme, where $t + 1$ denotes the minimum number of servers required for successful decryptions of the ciphertexts out of the m servers in the mix-network. Thus, the mix-network can tolerate up to t servers to be corrupt as a decryption will fail unless at least $t + 1$ servers collaborate in decrypting a ciphertext.

Combined with the batch technique in Chapter 3, the binary gate can be extended to shuffle two groups of inputs to offer better efficiency. We detail this extension in Section 5.2.1.

5.1.4 The Optimistic Mix-Network Scheme by Golle *et al.*

Using a threshold ElGamal cryptosystem, Golle *et al.* [GZB+02] proposed a very efficient re-encryption mix-network scheme using an optimistic approach. Correct mixing is checked by using a *proof of product* (POP, refer to Expression 5.1),

proving that the product of input ciphertexts is preserved in the product of output ciphertexts. The proof of product exploits the (multiplicative) homomorphic property of the underlying ElGamal cryptosystem. However, a checksum is required to verify the integrity of the messages. Also, the inputs are required to be encrypted twice, named *double enveloping*, to support backup mixing.

Double enveloping protects the anonymity of the original ciphertext owner from a relation attack by a dishonest mix server. When an input message is encrypted only once, a dishonest server can modify its output by multiplying two inputs (c_i and $c_{i'}$) and outputs the re-encryptions of the product of the two inputs ($c_i c_{i'}$, where $i \neq i'$) and an encryption of 1. This attack passes the proof of product test.

By observing the attacked (combined) plaintext output after decryption, the related ciphertexts can be identified. Double encryption is used to prevent such attacks, such that when the first mixing for the outer encryption is found to be incorrect, the inner encrypted messages are recovered by the decryption authorities and mixed again using a more robust, heavy-weight verifiable mix-network.

Based on the scheme by Pedersen [Ped91], a threshold version of ElGamal cryptosystem is employed with properly generated parameters. The private key is x, and the public keys are g and $y = g^x$. Several decryption authorities share the private key x as x_j, where $j \in \{1, 2, \ldots, m\}$. A secret message s is encrypted with a random value r as $c = (\alpha, \beta) = (g^r, sy^r)$. A collision-resistant hash function H is used to produce the hash checksum $h = H(\alpha, \beta)$. A double encrypted ciphertext is then produced with different random values r_1 and r_2 as $c_1 = (\alpha_1, \beta_1) = (g^{r_1}, \alpha y^{r_1})$ and $c_2 = (\alpha_2, \beta_2) = (g^{r_2}, \beta y^{r_2})$. An encryption of the hash checksum is produced using a random value r_3 as $c_3 = (\alpha_3, \beta_3) = (g^{r_3}, hy^{r_3})$.

For n messages and $i \in \{1, 2, \ldots, n\}$, inputs to the mix-network is a triplet of the form $(c_{i,1}, c_{i,2}, c_{i,3})$. The mix-network is a basic re-encryption mix-network, where each mix-server re-encrypts its input triplets (each of the element in a triplet) and outputs them in a random order. Afterward, the mix-server proves the preservation of product of messages in the mixing (POP) by proving the following expression.

$$\left(\prod_{i=1}^{n} \alpha_i = \prod_{i=1}^{n} \alpha'_{\pi(i)}\right) \wedge \left(\prod_{i=1}^{n} \beta_i = \prod_{i=1}^{n} \beta'_{\pi(i)}\right) \wedge \left(\prod_{i=1}^{n} h_i = \prod_{i=1}^{n} h'_{\pi(i)}\right) \tag{5.1}$$

Using this technique, the computational complexity for proving correct mixing

operation is independent of the number of inputs n.

After the mixing is completed, each output is decrypted by a quorum of decryption authorities. The final output of the mix-network are triplets of the form $(\alpha'_{\pi(i)}, \beta'_{\pi(i)}, h'_{\pi(i)})$, where $\pi(i)$ denotes the result of total permutation of i. Integrity of each output is verified by checking the corresponding hash checksum.

$$h'_{\pi(i)} = H(\alpha'_{\pi(i)}, \beta'_{\pi(i)})$$

Recent research revealing possible attacks on this mix-network scheme include Abe and Imai [AI03] and Wikström [Wik02]. Abe and Imai show that anonymity can be compromised by cheating the hash checksum. The first mix server can intentionally swap the hash checksum of two inputs, and a malicious user can use the hash checksum of another input. Although the dishonest entity will be identified, anonymity is compromised. This type of attack can be avoided by requiring a proof of knowledge of the hash value. However, efficiency is sacrificed since proving knowledge of a plaintext value in an encryption without revealing it is quite complex.

Wikström shows that an anonymity compromise is possible by exploiting the same key used for the inner and outer encryptions. The second mix session can be used as a decryption oracle to the first mix session. Simply using different keys for the inner and outer encryptions will eliminate this problem.

We propose a more optimistic approach by modifying this scheme. A trace-back protocol is employed to identify a corrupt entity in the mix-network. This is presented in Section 5.3.

5.2 A New Mix-Network using Extended Binary Mixing Gates

Input-output relationships of a mix-network must be kept secret. Because of this requirement, proving correctness of the mixing operation is often made complicated. Currently, there is no acceptably efficient method to check this. Hence, mix-network schemes to date do not accommodate real-world applications requiring a large number of messages to be shuffled, e.g. in a national election. Current research in this area aims at producing a secure and practical scheme for implementation in the real-world.

The proposed scheme in this section offers an alternative to the straightforward implementation of our batch technique discussed in the previous section. The resulting scheme balances security and performance by using extended binary mixing gates (**EBMG**s). This is a new notion, an extension of the use of binary mixing gates originally proposed by Abe [Abe99] (refer back to Section 5.1.3). In our scheme, each gate employs batch techniques to re-encrypt ciphertexts, and to prove and verify correctness of the shuffling operation. The design is based on a re-encryption chain mix-network employing individual mix server verification.

In an EBMG, a new batch re-encryption technique is employed to improve the efficiency of re-encryptions. The permutation is gate-based and simplified, such that correctness proof and verification of the shuffling can be batched.

Our proposed mix-network achieves the essential property of correctness and privacy. Proofs that outputs of the mix-network indeed correspond to its inputs are publicly verifiable. Although our mix-network does not offer all possible permutations, its privacy level is sufficient for many schemes, such as for a secret-ballot voting scheme.

Allowing a flexible input format, the shuffling process requires at least two rounds of mixing to adequately achieve pairwise privacy. The proposed mix-network is robust as a compromised mix server only weakens the privacy of the mix-network, where the switching positions of EBMGs controlled by that mix server is revealed.

Our proposed mix-network is correct, private, robust and publicly verifiable while being one of the most efficient mix-network schemes to date. The proposed batch re-encryption technique can be implemented in other re-encryption chain mix-network schemes to increase their performance. The proposed mix-network offers a higher level of security and efficiency compared to previous schemes.

In this section, a batch re-encryption technique using combinations of shared randomisers is presented. Details of two types of EBMG, one employing ElGamal cryptosystem and the other employing Paillier cryptosystem are provided. Two mix-network protocols based on the EBMGs and their privacy analysis are presented. Security and efficiency analysis of the proposed mix-network is also discussed.

5.2.1 A New Batch Re-Encryption Technique

In a re-encryption chain mix-network, re-encryptions and permutations are performed on every mix server. The computational cost of shuffling for n ciphertexts on a mix server is $\mathcal{O}(n)$. In this section, we present a new technique to batch re-encryption of ciphertexts. The n instances of re-encryption are batched, such that the computational cost on a server is reduced to $\mathcal{O}(\log_2 n)$.

Suppose an encryption function E is:

- semantically secure (see Appendix B or [Mao03, Chapter 14]), in which a message s is encrypted to $c = E(s, r)$, where r is a random integer;
- homomorphic, and there is an identity message I such that for any message $D(E(s)E(I)) = s$ (e.g. I is 1 in ElGamal encryption and 0 in Paillier encryption), where D denotes the corresponding decryption function.

Ciphertexts $c_1, c_2, \ldots, c_n$ encrypted using the encryption function E have to be shuffled. A batch re-encryption and permutation technique for n inputs in a shuffling (inside a mix network) is as follows, where $B_k(i)$ denotes the k^{th} bit of integer i.

1. For $k \in \{1, 2, \ldots, \log_2 n\}$, the server randomly selects $\log_2 n$ secret integers $r_{k,0}$, and $\log_2 n$ secret integers $r_{k,1}$.
2. The server performs $\log_2 n$ different probabilistic encryptions, each to obtain the values for $R_{k,0} = E(I, r_{k,0})$ and to obtain the values for $R_{k,1} = E(I, r_{k,1})$.
3. The server calculates $c'_i = c_{\pi(i)} \prod_{k=1}^{\log_2 n} R_{k,B_k(i)}$, where π is a random permutation function of $\{1, 2, \ldots, n\}$.

The random value for each ciphertext is interdependent according to its position. Thus, the batch re-encryption technique is only secure when the permutation is kept secret. This is such that the dependencies of random values used in the re-encryption are not obvious for an observer.

This technique of batch re-encryption can be applied to any re-encryption chain mix-network. It does not complicate validity verification of correct shuffling, as the original verification function can still be used after the batch re-encryption technique is employed. It does not compromise the privacy of the mix-network as the number of possible permutations and their distribution is unchanged after batch re-encryption technique is employed.

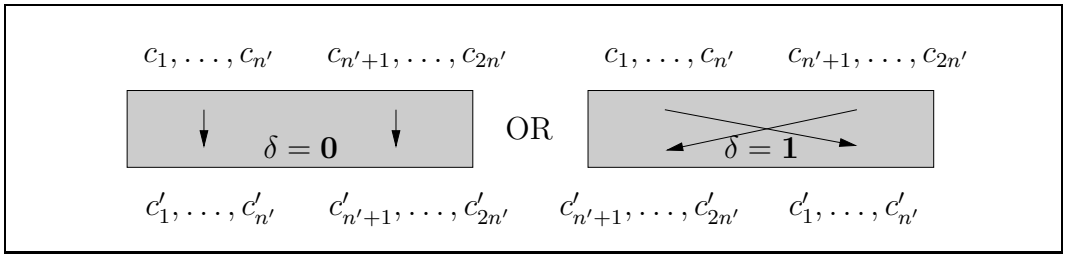

Figure 5.5: Two possible inputs to outputs permutations in an EBMG.

5.2.2 The Extended Binary Mixing Gate

Using the batch re-encryption technique in the previous subsection and the batch theorem from Chapter 3.2.1 and Chapter 3.2.3, we detail two types of Extended Binary Mixing Gate (**EBMG**) in this subsection. One performs ElGamal re-encryptions, and the other performs Paillier re-encryptions. The proposed mix-network is comprised of EBMGs as shown in Figure 5.6. The mix-network protocol is described in the next subsection.

A normal binary mixing gate (as in the scheme by Abe [Abe99]) mixes two inputs to two outputs by re-encrypting the inputs, and randomly permutes the ordering of its output. The mixing is required to be:

- correct: output plaintexts are a permutation of plaintexts contained in the input ciphertexts; and
- private: the permutation must be kept secret.

An EBMG shuffles $2n'$ inputs $\{c_1, c_2, \ldots, c_{2n'}\}$, to $2n'$ outputs $\{c'_1, c'_2, \ldots, c'_{2n'}\}$ by re-encrypting the inputs and partially permuting them. For $i \in \{1, 2, \ldots, 2n'\}$, the shuffling by an EBMG must be as below. This is illustrated in Figure 5.5.

- correct: when D denotes the corresponding decryption function, the shuffling is correct if: $D(c_i) = D(c'_i)$ OR $D(c_i) = D(c'_{i+n' \bmod 2n'})$; and
- private: whether $D(c_i) = D(c'_i)$ OR $D(c_i) = D(c'_{i+n' \bmod 2n'})$ is not revealed.

Either ElGamal or Paillier cryptosystem may be employed to encrypt and re-encrypt the inputs, and decrypt the outputs in a threshold manner. We use these two cryptosystem as examples and detail two EBMG protocols accordingly.

ElGamal Cryptosystem

ElGamal cryptosystem (Appendix B.1) is employed with the public parameters of (p, g, y) and input messages $c_i = (\alpha_i, \beta_i)$, where $i \in \{1, \ldots, 2n'\}$, and $2n'$ is the number of input messages. Suppose $n = 2n'$ for simplicity. The batch re-encryption technique in Section 5.2.1 is employed, where random values for re-encryptions are pre-computed and shared by all the EBMGs. Inside an ElGamal cryptosystem-based EBMG, its inputs are shuffled as follows:

1. Select a bit δ at random from $\{0, 1\}$ as a switching variable (see Figure 5.5).

2. For $k \in \{1, 2, \ldots, \log_2 n\}$, select secret integers $r_{k,0}$ and $r_{k,1}$ from $\mathbb{Z}_q$ at random. Compute the values of $RA_{k,0} = g^{r_{k,0}} \bmod p$; $RA_{k,1} = g^{r_{k,1}} \bmod p$; $RB_{k,0} = y^{r_{k,0}} \bmod p$; and $RB_{k,1} = y^{r_{k,1}} \bmod p$.

3. For $i \in \{1, 2, \ldots, 2n'\}$ and $i' = i + k\delta \bmod 2n'$, the output ciphertexts are constructed as follows:

$$\begin{aligned} c'_{i'} &= (\alpha'_{i'}, \beta'_{i'}) \\ &= \left(\left(\alpha_i \prod_{k=1}^{\log_2 n} RA_{k,B_k(i)} \bmod p \right), \left(\beta_i \prod_{k=1}^{\log_2 n} RB_{k,B_k(i)} \bmod p \right) \right) \end{aligned}$$

4. For $i \in \{1, \ldots, 2n'\}$, construct a non-interactive zero-knowledge (ZK) proof of correct shuffling operation by proving that one of the two following equations hold (similar to ZK proof construction in Chapter 4.2.1):

$$\log_g \pm \left(\frac{\alpha'_i}{\alpha_i} \bmod p \right) = \log_y \pm \left(\frac{\beta'_i}{\beta_i} \bmod p \right) \quad (5.2)$$

$$\log_g \pm \left(\frac{\alpha'_{i+n' \bmod 2n'}}{\alpha_i} \bmod p \right) = \log_y \pm \left(\frac{\beta'_{i+n' \bmod 2n'}}{\beta_i} \bmod p \right) \quad (5.3)$$

   The notation $\pm x$ denotes an absolute value of x. This notation is described in Chapter 3.2.

The computational cost for proving "Equation 5.2 OR Equation 5.3" is $6n'$ full-length exponentiations, and the corresponding verification costs $8n'$ full-length exponentiations. When n' is large, the computational cost increases accordingly.

For $i \in \{1, \ldots, 2n'\}$ and $t_i < 2^L < q$, we use Theorem 3.2.3 from Chapter 3.2.1 to construct a batch ZK proof-verification technique and batch both Equation 5.2

and Equation 5.3 respectively as the following two equations:

$$\log_g \pm \left(\prod_{i=1}^{2n'} \left(\frac{\alpha'_i}{\alpha_i} \right)^{t_i} \bmod p \right) = \log_y \pm \left(\prod_{i=1}^{2n'} \left(\frac{\beta'_i}{\beta_i} \right)^{t_i} \bmod p \right) \quad (5.4)$$

$$\log_g \pm \left(\prod_{i=1}^{2n'} \left(\frac{\alpha'_{i+n' \bmod 2n'}}{\alpha_i} \right)^{t_i} \bmod p \right) = \log_y \pm \left(\prod_{i=1}^{2n'} \left(\frac{\beta'_{i+n' \bmod 2n'}}{\beta_i} \right)^{t_i} \bmod p \right) \quad (5.5)$$

Using the batch technique, computational the cost for proving "Equation 5.4 OR Equation 5.5" is 6 full-length exponentiations. The corresponding verification costs 8 full-length exponentiations. We ignore exponentiation cost using an L-bit exponents (e.g. $\log_2 L = 20$ bits) since the computational cost is much less compared to full-length exponentiations (e.g. $\log_2 q = 1024$ bits), and is typically smaller than 5% of the entire computational cost.

Paillier Cryptosystem

Paillier cryptosystem (Appendix B.2) is employed with the public key of (p, g, y) and input messages c_i, where $i \in \{1, \ldots, 2n'\}$, and $2n'$ is the number of input messages. Suppose $n = 2n'$ for simplicity. The batch re-encryption technique in Section 5.2.1 is employed, where random values for re-encryptions are pre-computed and shared by all the EBMGs. Inside a Paillier cryptosystem-based EBMG, its inputs are shuffled as follows:

1. Select a bit δ at random from $\{0, 1\}$ as a switching variable (see Figure 5.5).

2. For $k \in \{1, 2, \ldots, \log_2 n\}$, select secret integers $r_{k,0}$ and $r_{k,1}$ from $\mathbb{Z}_N^*$ at random. Compute the values of $R_{k,0} = r_{k,0}^N \bmod N^2$; and $R_{k,1} = r_{k,1}^N \bmod N^2$.

3. For $i \in \{1, 2, \ldots, 2n'\}$, output ciphertexts $c'_{i'} = c_i \prod_{k=1}^{\log_2 n} R_{k,B_k(i)}$, where $i' = i + k\delta \bmod 2n'$ and $B_k(i)$ denotes the k^{th} bit of integer i.

4. For $i \in \{1, \ldots, 2n'\}$, construct a non-interactive ZK proof of correct shuffling operation by proving the knowledge (by proving the ability to compute)

of one of the two following calculations:

$$\left(\frac{c'_i}{c_i}\right)^{\frac{1}{N}} \mod N^2 \tag{5.6}$$

$$\left(\frac{c'_{i+n' \bmod 2n'}}{c_i}\right)^{\frac{1}{N}} \mod N^2 \tag{5.7}$$

The computational cost for proving the knowledge of one of the above two calculations is $4n'$ full-length exponentiations, and the corresponding verification costs $4n'$ full-length exponentiations. When n' is large, the computational cost increases accordingly.

For $i \in \{1, \ldots, 2n'\}$ and $t_i < 2^L < min(p, q, p', q')$, we use Theorem 3.2.9 from Chapter 3.2.3 to construct a batch ZK proof-verification technique and batch both of the previous naive calculations respectively as the following two calculations:

$$\left(\prod_{i=1}^{2n'}\left(\frac{c'_i}{c_i}\right)^{t_i}\right)^{\frac{1}{N}} \mod N^2 \tag{5.8}$$

$$\left(\prod_{i=1}^{2n'}\left(\frac{c'_{i+n' \bmod 2n'}}{c_i}\right)^{t_i}\right)^{\frac{1}{N}} \mod N^2 \tag{5.9}$$

Using the batch technique, the computational cost for proving the ability to compute one of the above calculations (from a possible of the above two calculations) is 4 full-length exponentiations. The corresponding verification costs 4 full-length exponentiations. We ignore exponentiation cost using an L-bit exponents (e.g. $\log_2 L = 20$ bits) since the computational cost is much less compared to full-length exponentiations (e.g. $\log_2 N = 1024$ bits).

5.2.3 The Mix-Network Protocol

The mix-network consists of a number of mix servers, each in charge of one level (row in Figure 5.6) of EBMGs. For simplicity, let the number of inputs to the mix-network n to be a power of 2.

The Core

There are a total of $m = \log_2 n$ levels (row in Figure 5.6). In the j^{th} level, there are $2^{-j}n$ EBMGs, where $j \in \{1, 2, \ldots, m\}$. The number of EBMGs required in

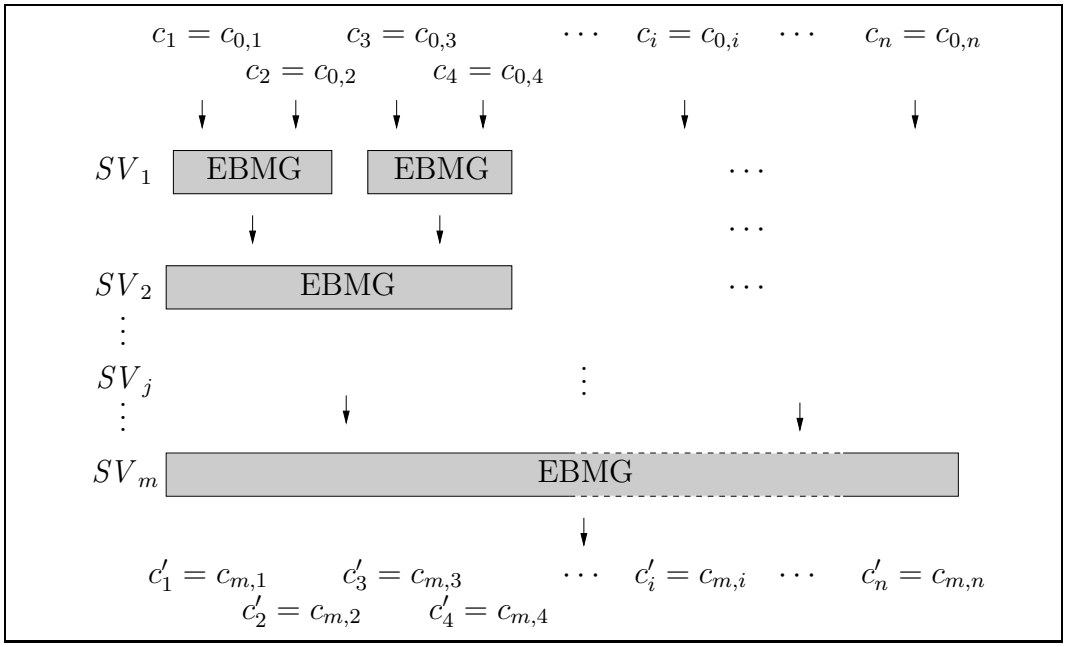

Figure 5.6: A core EBMGs construction in the proposed mix-network.

the mix-network core is $n - 1$. This is illustrated in Figure 5.6.

As the EBMGs can employ either ElGamal or Paillier re-encryption (refer back to Section 5.2.2), the proposed mix-network can either use threshold version of ElGamal [Ped92] or Paillier [DJ00] cryptosystem (refer to Appendix B).

Batch re-encryption technique in Section 5.2.1 is applied at each mix servers (rows in Figure 5.6), and batch zero-knowledge proof-verification technique - in Chapter 3.2.1 for ElGamal cryptosystem, or Chapter 3.2.3 for Paillier cryptosystem - is applied at each EBMG in the core mix-network.

The core mix-network protocol is detailed as below.

1. The inputs $c_{0,1}, c_{0,2}, \ldots, c_{0,n}$ are divided into $\frac{n}{2}$ pairs in the first mix server (first row of EBMGs; refer to SV_1 in Figure 5.6). For $i \in \{1, 2, \ldots, \frac{n}{2}\}$, every two successive inputs of $c_{0,2i-1}$ and $c_{0,2i}$ are shuffled to $c_{1,2i-1}$ and $c_{1,2i}$ by an EBMG. The shuffled outputs $\{c_{1,1}, c_{1,2}, \ldots, c_{1,n}\}$ of the first mix server SV_1 are then sent to the second mix server SV_2 to be its inputs.

2. Inputs of the j^{th} row of EBMGs are ciphertexts $c_{j-1,1}, c_{j-1,2}, \ldots, c_{j-1,n}$. For $i = 1, 2, \ldots, 2^{-j}n$, ciphertexts $c_{j-1,2^j(i-1)+1}, c_{j-1,2^j(i-1)+2}, \ldots, c_{j-1,2^j i}$ (every 2^j successive inputs) are mixed to $c_{j,2^j(i-1)+1}, c_{j,2^j(i-1)+2}, \ldots, c_{j,2^j i}$ by one EBMG. The j^{th} mix server has $\frac{n}{2^j}$ EBMGs. The outputs of this j^{th} mix server $c_{j,2^j(i-1)+1}, c_{j,2^j(i-1)+2}, \ldots, c_{j,2^j i}$ are forwarded as inputs to be shuffled

by the $j + 1^{th}$ mix server.

3. Each output from the last row of mixing is verified to be in the group G. Any output not in the group G is changed to its absolute value (refer back to the loose theorems in Chapter 3.2 and Chapter 3.3.2).

4. The final output ciphertexts are decrypted in a threshold manner by some decryption authorities (e.g. the mix servers themselves).

Ciphertexts Distances

As illustrated in Figure 5.6, the value of the switching variable δ determines the output positions of the ciphertexts processed in an EBMG. Structured as in Figure 5.6, the final output positions of the ciphertexts mixed by the core mix-network are determined by the values of the switching variables δ, each in the EBMGs processing the ciphertexts.

A user with a unique input in c_i can identify his own message output by the core mix-network (in plaintext after c_i' is decrypted), and identify the values of switching variables in every EBMG the ciphertext went through[1]. Using this information, the user can further identify the initial input positions of the output plaintexts (after the final output ciphertexts are decrypted) with a certain probability in relation to the distances between the ciphertexts. This is because after going through the core mixing, although all input ciphertexts are output to different output positions, the distances between those ciphertexts (Definition 5) are constant.

Definition 5. *The function $\Delta(c_{\hat{i}}, c_{\tilde{i}})$ denotes the distance of the target ciphertext $c_{\tilde{i}}$ from the anchor ciphertext $c_{\hat{i}}$. The value of $\Delta(c_{\hat{i}}, c_{\tilde{i}}) = (\log_2 \varepsilon(c_{\hat{i}}, c_{\tilde{i}})) - 1$, where $\varepsilon(c_{\hat{i}}, c_{\tilde{i}})$ denotes the number of elements in the smallest set of $2^{i'}$ successive ciphertexts $\{c_i, c_{i+1}, \ldots, c_{i+2^{i'}-1}\}$ containing $c_{\hat{i}}$ and $c_{\tilde{i}}$, where $2^{i'} | i - 1$.*

For n number of inputs, there are $\log_2 n$ bits of switching variables determining the final output position of each ciphertext. Each of the bit indicates the value of each switching variable in each EBMG that the ciphertext went through. The most significant bit indicates the value of the switching variable in the EBMG in SV_m, and $m = \log_2 n$.

[1]The user can trace back and deduce the value of each switching variable δ in each EBMG that his/her ciphertext went through, from the last EBMG in SV_m back to the first EBMG in SV_1. Thus, switching positions in those EBMGs are revealed.

An attack scenario with 8 input ciphertexts and one malicious user is as follows. Let $n = 8$, and a malicious user submits a unique input as c_1 into the core mix-network. As input c_2 is in the same pair with c_1, their distance is $\Delta(c_1, c_2) = 0$. Thus, the malicious user can successfully identify the user with input c_2. We name c_2 the immediate neighbour of c_1 as they have the minimum possible distance. The value of $\Delta(c_1, \{c_3, c_4\}) = 1$, and $\Delta(c_1, \{c_5, \ldots, c_8\}) = 2$. Thus, the malicious user can identify the senders of c_3 and c_4 as one of two users, and identify the senders of $\{c_5, \ldots, c_8\}$ as one of four users. This is because the malicious user can only guess the switching position δ in EBMG in SV_1 for c_3 and c_4, and need to guess the switching position δ in EBMG in SV_2 and SV_1 for $\{c_5, \ldots, c_8\}$. Note that $\{c_5, \ldots, c_8\}$ have the maximum possible distance to c_1.

For n inputs, and $1 \leq \hat{i}, \check{i} \leq n$, a malicious user with a unique plaintext input in $c_{\hat{i}'}$ can identify the initial input position $\check{i}$ of the final ciphertext output $c_{\check{i}'}$ from $2^{\Delta(c_{\hat{i}'}, c_{\check{i}'})}$ possible number of initial input ciphertexts. This is because $\Delta(c_{\hat{i}}, c_{\check{i}}) = \Delta(c_{\hat{i}'}, c_{\check{i}'})$. The malicious user requires $\Delta(c_{\hat{i}'}, c_{\check{i}'})$ bits of switching information on the final ciphertext output of $c_{\check{i}'}$ to determine the initial input position $\check{i}$ of $c_{\check{i}'}$.

Since the smallest possible distance Δ between two ciphertexts is 0, mixing using the core mix-network only achieves pairwise privacy. Furthermore, the privacy of the core mix-network can be compromised when half of the users collaborate, such that each of the ciphertexts input by the honest user is an immediate neighbour (have the minimum distance of $\Delta = 0$) of one of the ciphertexts input by the malicious users. The best case scenario for an honest user is when the distance between his ciphertext $c_{\hat{i}}$ and the ciphertext of a malicious user $c_{\check{i}}$ is maximum, where $\Delta(c_{\hat{i}}, c_{\check{i}}) = (\log_2 n) - 1$.

When the number of choices for the input messages is very small compared to the number of the input n (e.g. in a "yes/no" voting system), the probability for any input to be unique is small. Thus, the privacy of the mix-network can be achieved in most cases. However, when the number of choices for the input messages is not small enough, the probability for an input to be unique may be significant. Then, the previously described attack scenario is feasible. In this case, a solution is required to overcome this problem. We provide a method to prevent the attack using two rounds of mixing in the next subsection.

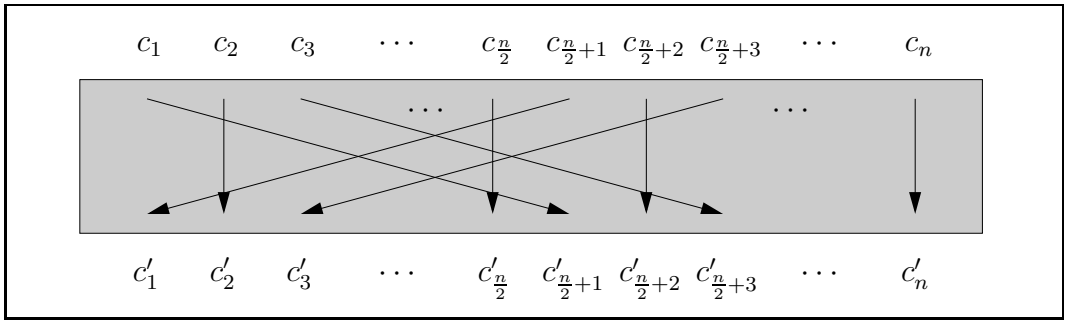

Figure 5.7: An example of a public fixed n-to-n permutation.

Two Rounds of Mixing

Limiting the allowable input format is not acceptable for schemes requiring a more flexible input format, such as in electronic auction schemes with unspecified threshold of biddable price, or in electronic voting schemes using a preferential system. Two rounds of mixing are required to sufficiently alter the relative positions of ciphertexts, such that the ciphertexts are in maximum distances to each other.

We employ two rounds of mixing and a public fixed n-to-n permutation in between the two rounds. The public n-to-n permutation (illustrated in Figure 5.7) permutes the ciphertexts, such that a final output distance in any two consecutive ciphertexts in the pair is of maximum value. The two rounds of mixing protocol are as follows:

1. Input ciphertexts are shuffled as in the core mixing protocol (refer back to Section 5.2.3). The output ciphertexts of the first round of mixing are directly forwarded (no decryption is performed) to the public fixed n-to-n permutation.

2. For n ciphertexts c_i, the public fixed n-to-n permutation outputs $c_{i'} = c_{(i+(n/2)(i \bmod 2)) \bmod n}$, where $i \in \{1, 2, \ldots, n\}$.

3. The output ciphertexts of the public fixed n-to-n permutation are forwarded as inputs to the second round of mixing, shuffled as in the core mixing protocol (refer back to Section 5.2.3).

4. The final output ciphertexts are decrypted in a threshold manner by some decryption authorities (e.g. the mix servers themselves).

The final output ciphertexts are not decrypted at the end of the first round. They are directly forwarded to the public fixed n-to-n permutation before being submitted as inputs to the second round. A malicious user can only guess the initial position of the ciphertext at the second round. Thus, the best case scenario for a malicious user is to guess the initial position of his final ciphertext output pair from the other half of the initial input ciphertexts with a probability of $\frac{2}{n}$.

The probability of a malicious user successfully identifying the initial input ciphertext position of any of the other ciphertexts is $\frac{1}{n-1}$. A collaboration of $n-1$ malicious users is required to successfully compromise the privacy of the mix-network.

5.2.4 Analysis

Security and efficiency of the proposed mix-network is analysed in this subsection. Correctness and privacy level of the mix-network are discussed. A privacy and computational cost comparison of the proposed scheme and other efficient schemes is also provided.

Security

The implementation of EBMG in Section 5.2.2 is correct as the batch technique in Chapter 3 fails with negligible probability. Also, as the batch technique is witness-hiding, the EBMG protocol is private.

Our proposed mix-network is correct as each EBMG constructing the mix-network is also correct. Also, since each EBMG is private, any input to the mix-network may be mixed to any of the n outputs in the mix-network. Moreover, for any input, the n possible shuffling results are equally likely. Thus, diffusion of any single input is optimally achieved.

Since binary gates are used to shuffle the inputs, the core mixing protocol only achieves pairwise privacy. After the core mixing process has concluded, a malicious user with a unique input can identify the message of his pair input shuffled by the same EBMG in SV_1. Furthermore, a collaboration of malicious users can further weaken the privacy of the core mix-network. We refer to Section 5.2.3 for a detailed analysis and two alternatives to alleviate this problem.

The number of possible permutations using the core mix-network is 2^{n-1}, where each permutation is equally likely. Two rounds of mixing achieves $2^{2(n-1)}$ possible permutations.

Table 5.2: A comparison of privacy level in terms of diffusion achieved, where κ indicates the number of honest mix servers in a (t, m) threshold cryptosystem.

Mix-network scheme	One input	All inputs	Uniform
Abe [Abe99]	(if $\kappa > t$) 1 among n	(if $\kappa > t$) $n!$ perms	no
Abe & Hoshino [AH01]	(if $\kappa > t$) 1 among n	(if $\kappa > t$) $n!$ perms	yes
Furukawa & Sako [FS01]	1 among n	$n!$ perms	yes
Neff [Nef01]	1 among n	$n!$ perms	yes
Groth [Gro03]	1 among n	$n!$ perms	yes
Our scheme (the core mix-network)	1 among n	2^{n-1} perms	yes
Our scheme (two rounds)	1 among n	$2^{2(n-1)}$ perms	yes

Although the proposed mix-network does not offer all possible permutations, we consider the privacy level to be strong enough for many applications requiring a large number of messages to be communicated anonymously, i.e. where n is large.

Table 5.2 compares the anonymity level of our proposed scheme with other high-performance mix-network schemes. The degree of privacy is measured in terms of diffusion offered by a mix-network. A mix-network with perfect privacy has $n!$ permutations, each of them equally likely.

In the proposed mix-network, if one mix server is compromised or the mixing for that mix server is revealed, the number of possible outputs for an input and the number of possible permutations in the mix-network is reduced by half (i.e. reveal the values of switching variables as in Section 5.2.3), while the rest of possible shuffling and permutations are still equally likely.

To address this problem and achieve a stronger privacy, the entire mixing process can be repeated a number of times sufficiently (i.e. extending from Section 5.2.3).

Efficiency

The computational cost of one mixing operation (the core mix-network) is as follows:

- Re-encryption: $4(\log_2 n)^2$ full-length exponentiations using ElGamal cryptosystem; and $2(\log_2 n)^2$ full-length exponentiations using Paillier cryptosystem.

Table 5.3: A computational cost comparison for mixing the ciphertexts, in full-length exponentiations.

Mix-network scheme	Mixing	Verification of correct mixing
Abe [Abe99]	$> 16(n \log_2 n - n + 1)$	$> 16(n \log_2 n - n + 1)$
Furukawa & Sako [FS01]	$40n$	$40n$
Groth [Gro03]	$32n + 12n/\kappa' + 12$	$24n + 12n/\kappa' + 24$
Our scheme (one round)	$4(\log_2 n)^2 + 6(n + 1)$	$10(n - 1)$
Our scheme (two rounds)	$8(\log_2 n)^2 + 12(n + 1)$	$20(n - 1)$

- Proof of valid mixing: $6(n - 1)$ full-length exponentiations using ElGamal cryptosystem; and $4(n - 1)$ full-length exponentiations using Paillier cryptosystem.
- Verification of valid mixing: $10(n-1)$ full-length exponentiations using ElGamal cryptosystem; and $4(n-1)$ full-length exponentiations using Paillier cryptosystem.

We compare computational cost of the proposed mix-network against other mix-network schemes currently considered efficient. The comparison figure is summarised in Table 5.3 based on the use of ElGamal cryptosystem. In the table, we assume 4 mix servers are used in the mix-network [FS01, Nef01, Gro03], n is the number of inputs to the mix-network, and κ' is a parameter smaller than n.

In the mix-network scheme by Abe [Abe99], we only provide a lower threshold of its computational cost as the concrete number of gates was not provided. The mix-network by Neff [Nef01] is not included in the table as the protocol is not provided in great detail. However, its performance should be similar to the mix-network scheme by Groth [Gro03], as they employ similar techniques.

Research in mix-network schemes mainly focuses on improving the efficiency of verifying correct mixing operation. Although the performance gain is not significant, the batch re-encryption technique in Section 5.2.1 can also be implemented in other re-encryption chain mix-network schemes (or other appropriate schemes) to improve their overall efficiency.

From Table 5.3, our proposed mix-network scheme performs better than the other schemes. Our proposed mix-network is able to achieve correctness and privacy in a very efficient manner. Note that one round of mixing using our proposed mix-network is sufficient to provide security and privacy for simple

schemes, such as in a "yes/no" voting system.

5.3 A Proposed Optimistic Mix-Network

This section offers a modification of the scheme by Golle *et al.* discussed in Section 5.1.4. This is to be used for a receipt-free voting scheme in Chapter 6.

To provide a private re-encryption for the initial mix-network inputs and to eliminate the attacks to the mix-network scheme, the scheme is modified as follows:

- The hash checksum is removed to invalidate the relation attacks [AI03, Wik02].
- Single encryption is used instead of double encryption to prevent a ciphertext owner from using the inner encryption of the double enveloping as a receipt (refer to Chapter 6).
- We only check that $\prod s_i = \prod s'_i$ in the proof of product, where s_i and s'_i are the plaintext messages before and after the mixing respectively.

For n messages and $i \in \{1, 2, \ldots, n\}$, each ciphertext owner interacts with a re-encryption authority to generate a ciphertext (α_i, β_i) for his/her message s_i. These are input ciphertexts to the mix-network.

Using the appropriate ElGamal cryptosystem parameters (see Appendix B.1), the proposed mix-network protocol is as follows:

1. Re-encrypt and randomly permute the ordering of ciphertexts:
   Each mix server receives n input ciphertexts (α_i, β_i). The ciphertexts are re-encrypted to be (α'_i, β'_i). The mix server then outputs the re-encrypted ciphertexts in a random order $(\alpha'_{\pi(i)}, \beta'_{\pi(i)})$, where $\pi(i)$ is a random permutation of i.

2. Prove preservation of products (individual mix server verification):
   Each mix server proves Equation 5.10 in zero-knowledge.

$$\log_g \frac{\prod_{i=1}^{n} \alpha'_i}{\prod_{i=1}^{n} \alpha_i} = \log_y \frac{\prod_{i=1}^{n} \beta'_i}{\prod_{i=1}^{n} \beta_i} \tag{5.10}$$

   Correctness of the mixing operation is publicly verifiable by anyone using the public values of g, y, (α_i, β_i) and (α'_i, β'_i). Using the Chaum-Pedersen

protocol in [CP93] (refer to Appendix C), this zero-knowledge proof requires 2 exponentiations for a proof construction and 6 exponentiations for the corresponding verification.

For a highly trusted mix-network, a verification variation named *global verification* can be used. This technique takes a more optimistic approach as the preservation of product is verified, not by the mix servers in each shuffling, but by the decryption authorities after the entire mixing (all the shufflings) is completed. Decryption authorities decrypt the product of the first input ballots and the product of the last output ballots of the mix-network, and check the equality of these two values.

Individual mix server verification offers early detection of error in mixing. Thus, mixing can be aborted and done by other mix servers. This verification technique is preferable as it provides a check for correct shuffling on each mix server. Using the global verification, each mix server is not required to produce any proof. Thus, mixing is more efficient, however errors will only be detected when the proof of products is checked.

The scheme employs a *trace-back protocol* to reveal a dishonest entity should an invalid ciphertext be found. The invalid ciphertext is traced backward to each of the entities starting from the mix server, re-encryption authority, to the ciphertext owner. Thus, cheating is discouraged since a dishonest entity will be revealed. The protocol is as follows:

1. The last mix server is required to reveal the i^{th} input corresponding to the $\pi(i)^{th}$ invalid output, and prove the correctness of its re-encryption by revealing the random value used for re-encryption. This process is repeated by all mix servers in a reverse order of shuffling until an invalid shuffling is found.

2. If mixing was found to be correct, the re-encryption authority is required to reveal the corresponding input and output re-encryption, and prove the re-encryption by revealing the random number.

3. If re-encryption by the re-encryption authority was found to be correct, the ciphertext owner is identified.

A dishonest mix server takes two different inputs, and produces two outputs which are re-encryptions of 1 and the product of the two input ciphertexts. As

the product of both the inputs and outputs are still preserved, proof of correct mixing is accepted, but the recovered messages are invalid.

However, the dishonest mix server will be identified using the trace-back protocol and can be sanctioned accordingly. When a trace-back occurs to a specific mix server in the middle of the mix servers, input-output relationship will not be revealed. When an invalid ballot is traced back to the first mix server, this server will know the input-output relationship.

Our optimistic mix-network requires three times less computation compared with the scheme by Golle *et al.* This is because our scheme uses a single encryption while the scheme by Golle *et al.* uses three encryptions for the double enveloping. In terms of the proof of product (POP), our scheme requires three times less computation if we use the individual mix server verification. If we use the global verification, our scheme is much more efficient, since only the initial input product and final output product are decrypted by a quorum of decryption authorities and compared. Attacks [AI03, Wik02] on the mix-network scheme by Golle *et al.* are not applicable in this scheme, while efficiency is improved threefold.

The proposed optimistic mixnet is more light-weight because single encryption is used. One major limitation of the mix-network is the possibility of an invalidation attack by a dishonest mix server. Although, any misbehaviour causing invalidation can be traced back easily. As we believe that the mix-network is very efficient, future work will be focused on improving the security properties while retaining the efficiency.

We employ this proposed scheme to construct an efficient receipt-free voting scheme in Chapter 6.

5.4 Applications of Batching in A Mix-Network

Continuing from Chapter 3, theorems detailed in that chapter are used in this section. Example applications on the use of batch ZK proof and verification techniques for re-encryption and decryption/threshold decryption in mix-network are presented. An improvement analysis on the mix-network using our batch techniques is also provided.

5.4.1 Batching in A Re-Encryption Chain Mix-Network

An important application of batch zero-knowledge proof-verification of correct re-encryptions is batch zero-knowledge proof-verification of correct shuffling. We discuss the application of batching for a mix-network for batch proof and verification techniques without sacrificing either privacy or soundness.

Shuffling using ElGamal Re-Encryption

A single shuffling is seldom used in a mix-network. Normally, several shufflings are employed together to form a mix-network, such that the mixing is private if the permutation of at least one mix server is kept secret. Thus, it is sufficient to achieve correctness, soundness and privacy of the entire mix-network while ignoring correctness, soundness and privacy of any single shuffling in most applications. A modification with an added extra test for group memberships on the final plaintext outputs of the mix-network is required. After this modification, the mix-network is satisfactory for most applications.

A straight-forward application of batching for ElGamal only guarantees that $\log_g \pm(\alpha'_i/\alpha_{\pi(i)}) = \log_y \pm(\beta'_i/\beta_{\pi(i)})$. A final check must be performed after the mix-network has completed the (entire) mixing process. The final output ciphertexts of the mix-network are decrypted. Each output message must be verified to be in the group G. If all the messages are in G, no more processing is required. Otherwise, one or more of the messages are outside the group G. These messages are then corrected to be in G (refer back to Chapter 3.3.1).

Although any single shuffling is not guaranteed to be strictly correct or sound, the final check guarantees that the mix-network as a whole is strictly correct and sound. The mix-network is also publicly verifiable and efficient (with a computational cost linear to the product of the number of inputs and the number of mix servers).

This modification adds n full-length exponentiations (for the final membership check) to the total cost of the mix-network. Where it is not necessary to perform a membership test in every shuffling, the mix-network is still efficient. After the batch technique is adopted, efficiency of the mix-network is improved.

Shuffling using Paillier Re-Encryption

With the support of Theorem 3.2.9, a batch ZK proof-verification technique can be employed to design a correct, publicly verifiable, sound and efficient re-encryption mix-network employing Paillier encryption.

The protocol is correct and publicly verifiable. According to Theorem 3.2.9, the shuffling is sound. It is also efficient, requiring only $\mathcal{O}(n)$ exponentiations. The probability that a dishonest mix server does not perform the shuffling as described, but passes the verification is negligible.

5.4.2 Threshold Decryptions

An important application of batch proof-verification of correct threshold decryptions is checking of valid final output decryptions in mix-networks.

In a *decryption chain* mix-network, centralised decryption is employed. Hence, if ElGamal encryption is employed, batch verification of correct decryption operation is feasible. Each mix server must publicly prove that decryption of all its inputs is correct. If the proof for each decryption is presented and verified separately, the computational cost is high for both the mix server and any verifier.

The mix server can employ the batch verification technique as in Chapter 3.3.2 to efficiently prove and verify correctness of decryptions. If the batch verification fails, the mix server must have performed at least one incorrect decryption.

In a *re-encryption* mix-network, the power to decrypt is typically shared among multiple decryption authorities. After the shuffling is completed by all the mix servers, a quorum of decryption authorities have to cooperate to decrypt the mixed ciphertexts. The authorities have to publicly prove that their partial decryptions are correct. Once again, separate proofs for each partial decryption imply a high computational cost.

After the efficiency of shuffling has been improved (such as in [GZB$^+$02, Gro03, PBDV04]), proof and verification of correct decryption operation becomes an efficiency bottleneck. Thus, batch proof-verification technique for verifying correct threshold decryptions can be applied to improve the efficiency of a re-encryption mix-network and remove this bottleneck.

Batch proof-verification of correct threshold Paillier decryptions in Chapter 3.3.2, or of threshold ElGamal decryptions in Chapter 3.3.2 can be employed to batch correct partial decryption proofs and verifications on the final decryp-

Table 5.4: Efficiency improvement in a mix-network using batching techniques.

Mix network	Soundness	Computational cost
Permutation-based verification of shufflings without batch verification of decryptions [Gro03]	strong	$4n(n!-1)+5mn+2n$
Input-based verification of shufflings without batch verification of decryptions [PBDV04]	moderate	$mn(4n-2)+5mn$
Permutation-based batch verification of shufflings and batch verification of decryptions	strong	$9mn+2m$

tions in a re-encryption mix network.

After batching, decryption verification becomes more efficient as the computational cost for proof and verification of correct partial decryptions for a decryption authority decreases from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ full-length exponentiations. The probability that a dishonest authority performs incorrect decryption but passes the validity check is negligible.

5.4.3 Improvements Analysis

Using theorems in Chapter 3, issues concerning group membership in the algorithms of Bellare *et al.* are efficiently solved. Batch ZK proof-verification techniques have been applied to batch proving and verifying correctness of encryptions, re-encryptions and decryptions based on these theorems.

Efficiency improvement in mix-network is illustrated in Table 5.4, where the number of full-length exponentiations are counted in an ElGamal re-encryption mix-network with n inputs and m mix servers.

As illustrated in the table, application of batching in a mix-network offers a great efficiency improvement. In the table, row 2 and 3 (without batching) is computationally more expensive compared to the last row (with batching).

5.5 A Preferential Voting Case Study

In a mix-network, the size of each input does not directly affect the efficiency of a mix-network. Thus, we propose an efficient generic secret-ballot voting scheme

utilising a robust mix-network. We use the preferential voting system described in Chapter 2.3.1. This section continues from the case study in Chapter 4.3.

The scheme is as follows:

1. **Preparation phase**: Parameters for the cryptosystem are initialised, and integers representing each preference of the candidates or choices are defined. For K number of candidates or choices, there are $K!$ such integers.

2. **Voting phase**: Each voter i performs the following:

   (a) selects a sequence, $k \in \{1, \ldots, K!\}$, to represent his/her preference;

   (b) encrypts the selection, e.g. $E(k_i) = c_i = (\alpha_i, \beta_i) = (g^{r_i}, k_i y^{r_i})$, using a semantically secure encryption algorithm such as the ElGamal cryptosystem.

   (c) communicates the encrypted vote (ballot) c_i to a vote collecting authority, and attach a digital signature of the ballot.

3. **Tally phase**: The vote collecting authority performs the following:

   (a) verifies the identity of the voter by checking the digital signature;

   (b) forwards the ballots from authorised voters to a mix-network to be anonymised.

   Afterward, the tally authority performs the following:

   (a) collects the valid plaintext votes $s'_{\pi(i)}$ output by the mix-network, where π is a secret permutation function of the mix-network;

   (b) processes the plaintext votes electronically by using a program for counting the preferences and calculating the winning candidate or choice.

In contrast to voting protocols based on the homomorphic encryption approach, the size of each input does not affect the complexity of the scheme. This is because the votes are plaintexts after being anonymised by the mix-network. Therefore, any voting system (e.g. preferential voting system) can be implemented using the proposed scheme. Tallying is transparent and can be performed using a counting program for that voting system. Only the number of inputs contribute to the efficiency of the scheme.

Table 5.5: A complexity analysis for the adapted voting system using a robust mix-network.

Entity	Parallel processes	Exponentiations per process
A voter	1	2
The mix-network	$(2t \log_2 n) - 1$ $= 20t - 1$	$32t(2 \log_2 n - 1)$ $= 608t$

In a straight-forward adaptation of electronic preferential voting employing mix-network, the vote need not be in a special form as in the homomorphic encryption approach. In this case, the vote can be of the form an integer ranging from 1 to $K!$, where K denotes the total number of candidates or choices in the preferential voting. Thus, the vote size is $\lceil \log_2(K!) \rceil$.

We employ the mix-network by Abe [Abe99, AH01] described in Section 5.1.3 in this case study. We choose to use MiP-2 since it separates the mixing and decryption, and thus is more straight-forward to understand.

Each voter generates exactly one ballot (mix-network input). Therefore, the number of inputs is the same as the number of voters, n. Analysing MiP-2, the total number of modular exponentiations required for the mixing operation and constructing the corresponding zero-knowledge proof is $32(2 \log_2 n - 1)$ (see Table 5.1). The main factor contributing to the computational cost of the scheme is the construction of a zero-knowledge proof required at each switching gate. The number of mixing gates an input is required to pass through in the mix-network is $2 \log_2 n - 1$ [Abe99], directly proportional to the number of input n.

In the case of an election for the Australian House of Representatives[2], the number of modular exponentiations required for the scheme is presented in Table 5.5 (we count $\log_2 1000$ to be 10 for simplicity). In addition to Table 5.5, the number of threshold decryption operation required is $3 + 4t$ exponentiations. Vote tabulation is performed as in the traditional counting of votes without any cryptographic processing required. This is because outputs of the mix-network are randomly permuted plaintext votes.

When the mix-network by Abe is replaced by our two-rounds EBMG mix-network, the computational cost for the voter remains. The total computational cost for the mix-network is now 12812 modular exponentiations (refer back to

[2]We use parameters as in Chapter 4.3, where the number of candidates $K = 20$, and the number of voters (per polling booth) $n = 1000$.

Section 5.2.4) as compared to $608t(20t - 1) = 12160t^2 - 608t$ (see Table 5.5) using the mix-network by Abe.

The size of the electronic vote for a preferential voting system is inherently larger than for a 1-out-of-K voting system, when the number of candidates, K, increases. In the case of preferential voting system, the size of the vote is at least $\log_2 m!$ bits. Each vote in this scenario is of $\lceil \log_2(20!) \rceil = 62$ bits.

The number of candidates does not affect the computational complexity of the mix-network. Only the number of inputs affect the computational complexity of the mix-network. In the case of an election for the Australian Senate where the number of voters (per polling booth) is also $n = 1000$, the computational complexity previously discussed remains.

As discussed in the Chapter 4.3, any voting system that employs the homomorphic encryption approach will require the voter to prove that a valid vote was encrypted. The computational complexity of such a proof is $\mathcal{O}(K!)$. Also, the vote size is $\mathcal{O}(2^{\log_2 n})$. Unless an efficient proof technique is available to validate the encrypted vote, the homomorphic encryption approach will not be practical for a straight-forward adaptation for preferential voting systems as shown in Chapter 4.3.

Voting systems that employ a mix-network, on the other hand, do not require such a proof from the voter or are restricted by a structured vote requirement. In fact, the computational complexity of a mix-network is not adversely affected by the number of candidates or choices. Therefore, a mix-network based voting system is well suited for a straight-forward adaptation of a preferential voting system. This include the Australian elections for the house of representatives and the Australian senate elections.

However, a combination of both approaches leads to a promising framework (a hybrid approach) where a small number of pre-set of preferences is allowed. This is shown in Chapter 7.

5.6 Summary

Two mix-network schemes and a shuffling scheme have been reviewed and improved. A preferential voting case study using a mix-network has also been presented in this chapter.

A mix-network scheme by Abe [Abe99, AH01] is extended, such that its ef-

ficiency is increased. A further work to further optimise the extended scheme is possible. Mix servers may be arranged in such a way, such that the shuffling process can be done in parallel. This will allow the mix-network to achieve better throughput.

The scheme by Golle *et al.* [GZB+02] is modified to allow better efficiency with a weaker trust assumption. However, a trace-back protocol is provided in the modification to identify a dishonest entity.

Example applications of batch techniques in a mix-network have also been presented.

From the case study, secure secret-ballot voting scheme accommodating write-in votes can only be realised using mix-network. In addition to that, mix-network is more suitable to realise secure secret-ballot voting scheme accommodating a straight-forward adaptation of a preferential system. In Chapter 7, the homomorphic encryption and the mix-network approach are combined to form a hybrid framework with the advantages of both approaches.

The next chapter offers research into receipt-freeness. It is one of the many important requirements for a secure secret-ballot voting scheme.

Chapter 6

Receipt-Free Voting

Receipt-freeness is one important requirement for secret-ballot voting (refer back to Chapter 2.7). It is a stronger notion of the privacy requirement, such that a voting result reflects true opinions of the voters. This is to prevent vote buying/selling, or coercion/intimidation. Without receipt-freeness, a voter may be used as a proxy to cast a vote instead of casting his/her own vote.

Two models of receipt-free voting are presented in this chapter. The first model improves the efficiency of the scheme by Lee *et al.* [LBD+04] by using an optimistic mix-network instead of a verifiable one. An optimistic mix-network from Chapter 5.3 is used for this model. This model has been previously published in [ALBD04a].

Based on the first model, the second model lowers the trust requirement of one re-encryption authority (a randomiser) to a threshold of re-encryption authorities. This is by using two new techniques. One is a threshold re-encryption technique, and the other is a batch verification technique to verify designated verifier re-encryption proofs. This model has been previously published in [ALBD05].

The work presented in this chapter offers a foundation to providing receipt-freeness in an efficient manner. Example models provided in this chapter are based on the mix-network approach.

6.1 Background

The concept of receipt-freeness in voting is described. A receipt-free voting scheme by Lee *et al.* [LBD+04] is recalled, and a designated verifier re-encryption proof (DVRP) protocol is also recalled. This section provides a basis for both of our proposed models.

6.1.1 Receipt-Freeness

Abuses of voting include the use of coercion, intimidation, and selling/buying of votes. These abuses are aimed at corrupting a voting result, such that the result favours a particular candidate or choice. The concept of *secret-ballot* was first designed to prevent such threats from producing an unfair and corrupted result. This is achieved by using official ballot papers, vote casting in a private booth, and public ballot submissions into a ballot box.

The concept of receipt-freeness was first investigated by Benaloh and Tuinstra [BT94]. It is an important property required to provide a fair voting result, to ensure that voters are not used as proxies to cast votes. To achieve this, voters should not be able to provide a receipt for any vote cast in the ballot. Otherwise, the receipt can be used to satisfy a coercing or a vote-buying party.

In a cryptographic voting protocol, a ballot contains an encrypted vote. A vote buyer can provide the random value r used to construct a particular ballot (encrypt a particular vote). Since ballots are public, a buyer can check that a particular ballot contains a particular vote by opening the ballot using the random value r.

For example, in an ElGamal cryptosystem setting, a ballot is constructed as $c = (\alpha, \beta) = (g^r, sy^r)$. With a knowledge of the random value r, the ballot can be opened by simulating the ballot construction. Hence, the vote s is revealed.

A solution to this problem is to provide a private randomisation service to each ballot, such that the random value in a ballot is changed and unknown to the voter.

Both for the homomorphic encryption and mix-network approaches, randomisation can be performed right after a ballot is submitted. For the homomorphic encryption approach, the randomisation is performed before the ballot is made public. For the mix-network approach, the randomisation is performed before the ballot is forwarded as one of the inputs to a mix-network.

However, the randomisation process requires an untappable channel. Used by Okamoto [Oka97], such a channel is required to hide a ballot before it is randomised. No third party can observe (tap) the communication between a sender and receiver using the untappable channel. Such a channel can be realised using a physical security assumption.

6.1.2 The Scheme of Lee *et al.*

Receipt-freeness is typically achieved by randomising the ballot. This is to remove a voter's knowledge of the random value used for ballot construction (encryption of a vote). The randomisation eliminates the voter's ability to prove the content of the ballot to a third party using a knowledge of the random value used for the ballot construction. At the same time, the voter must receive an assurance that the randomised ballot contains the same vote.

The mix-network based receipt-free voting scheme of Lee *et al.* employs a tamper-resistant hardware device named tamper-resistant randomiser (TRR) and an untappable channel. The TRR performs the role of a third party randomiser. Correct randomisation is verifiable through the use of a designated verifier re-encryption proof (DVRP).

Afterward, the ballots are then anonymised by the mix-network, and each output of the mix-network is decrypted by a quorum of tally (decryption) authorities.

The voting phase consists of the following four sub-phases.

1. Each voter prepares a **first ballot** by encrypting his vote. The ballot is then sent to TRR for randomisation.

2. The TRR randomises the first ballot with a re-encryption to produce a **final ballot**.

3. The TRR also produces a **designated verifier re-encryption proof** (DVRP) to prove the correctness of the re-encryption to the voter. The final ballot and the DVRP are then sent to the voter.

4. The voter checks the DVRP, then signs and **submits the final ballot** to a mix-network if the check is accepted.

We note that the scheme employs a verifiable mix-network. Efficiency improvements are possible by alternatively using an optimistic mix-network.

6.1.3 A Designated Verifier Re-Encryption Proof

A designated verifier proof was first introduced by Jakobsson *et al.* [JSI96], and applied for re-encryption in voting by Lee and Kim [LK02]. The fundamental idea is that the proof will only convince a designated verifier, and the proof is not transferable (not convincing) to others (to verify). This is possible as the prover proves knowledge of either the value in question OR the private key belonging to the designated verifier.

The prover constructs the proof by simulating a proof of knowledge of the private key belonging to the designated verifier. On the other hand, the designated verifier is convinced of the proof if the verification is accepted. This is because only the designated verifier holds the knowledge of the private key. Thus, the proof is not transferable (not convincing to others) since a designated verifier can construct a proof of knowledge of the private key and simulate the proof of knowledge for the random value used for the re-encryption.

Correctness proof of a re-encryption by a re-encryption authority is only verifiable to (only convinces) a designated verifier for the proof. A designated verifier re-encryption proof (**DVRP**) only convinces a designated verifier. A re-encryption authority constructs the proof by simulating the zero-knowledge proof of knowledge of the private key corresponding to the public key y_i owned by the designated verifier, and combines this proof with a zero-knowledge proof of equality of discrete logarithms (that the same randomisation value r' is used in both ciphertext parts). The second premise is proved using the following equation.

$$\log_g \frac{\alpha'}{\alpha} = \log_y \frac{\beta'}{\beta}$$

A non-interactive version[1] of the DVRP protocol is recalled as follows:

1. The re-encryption authority selects values of $\tau, u', w' \in \mathbb{Z}_q$ at random.

2. The re-encryption authority computes commitments of $(\gamma_1, \gamma_2) = (g^\tau, y^\tau)$, and simulates the knowledge of the private key of y_i owned by a designated verifier by producing $\gamma_3 = g^{w'} y_i^{u'}$.

3. The re-encryption authority computes the value of a challenge u using a one-way collision-resistant hash function H, with a range of $\mathbb{Z}_q$, as $u = H(\gamma_1, \gamma_2, \gamma_3, \alpha', \beta')$.

[1] In the interactive version of the protocol, the voter selects the challenge u at random.

4. The re-encryption authority calculates the response $w = \tau - r'(u + u')$.

5. The re-encryption authority sends the values of (u, u', w, w') to the designated verifier.

After receiving the proof, the designated verifier can perform the check using the following Equation 6.1.

$$u = H\left(g^w \left(\frac{\alpha'}{\alpha}\right)^{(u+u')}, y^w \left(\frac{\beta'}{\beta}\right)^{(u+u')}, g^{w'} y_i^{u'}, \alpha', \beta'\right) \tag{6.1}$$

If the verification is accepted, a designated verifier is convinced that the re-encryption was indeed correct since the re-encryption authority knows the value of r' used in re-encrypting the ciphertext. On the other hand, a proof transferred to another entity will not be convincing since the designated verifier can always construct a valid proof from his/her possession of the private key corresponding to the public key y_i (construct the proof by both simulating the knowledge of the random value r' AND proves the knowledge of the private key corresponding to the public key y_i).

6.2 Two New Cryptographic Primitives

We present two new cryptographic primitives to construct a receipt-free voting protocol for our model 2 (refer to Section 6.4). A threshold re-encryption technique is proposed, and a batch designated verifier re-encryption proof (DVRP) is presented.

6.2.1 A Threshold Re-Encryption Technique

A new technique is developed to perform a re-encryption by a threshold of re-encryption authorities instead of relying on a single trusted authority. The assumption of a single trusted re-encryption authority can be removed. The technique offers robustness in a re-encryption operation, and lowers the trust for a re-encryption authority.

The idea is based on the secret sharing scheme by Shamir [Sha79] (refer to Appendix A). The scheme is applied to share a message $s = g^{\epsilon}$ into m messages $s_j = g^{\epsilon_j}$, where ϵ is shared using Shamir's (t, m) secret sharing scheme. Each

s_j is then encrypted with the value of $r_j \in \mathbb{Z}_q$ selected at random using the ElGamal cryptosystem as $(\alpha_j, \beta_j) = (g^{r_j}, s_j y^{r_j})$ and forwarded to m authorities for re-encryption.

The j^{th} re-encryption authority performs the re-encryption by selecting a value $r'_j \in \mathbb{Z}_q$ at random, and computes $(\alpha'_j, \beta'_j) = (\alpha_j g^{r'_j}, \beta_j y^{r'_j})$.

From a quorum S of honest authorities, the re-encrypted ciphertext is then constructed as below where $\mu_j = \prod_{j' \neq j} \frac{j'}{j'-j}$.

$$\begin{aligned}(\alpha', \beta') &= \left(\prod_{j \in S} \alpha_j'^{\mu_j}, \prod_{j \in S} \beta_j'^{\mu_j}\right) \\ (\alpha', \beta') &= \left(g^{\sum_{j \in S}(r_j + r'_j)\mu_j}, g^{\sum_{j \in S} \epsilon_j \mu_j} y^{\sum_{j \in S}(r_j + r'_j)\mu_j}\right)\end{aligned}$$

Decryption of (α', β') using the secret key x (where $y = g^x$) is performed to recover the original message $s = \beta'/\alpha'^x$. This is because $\epsilon = \sum_{j \in S} \epsilon_j \mu_j$.

As in a re-encryption of a ciphertext, each re-encryption authority is also required to prove that his/her re-encryption of a partial ciphertext is correct by using a DVRP as in the Section 6.1.3.

Individual verification of each DVRP is inefficient. A batch theorem from Chapter 3 can be used to construct a batch DVRPs verification technique. The batch technique allows efficient verification of the DVRPs. This is detailed in the next subsection.

6.2.2 A Batch DVRP Verification Technique

Based on Theorem 3.2.3 (refer back to Chapter 3.2.1), verifications of each of the DVRP can be batched for better efficiency. This is similar to the loose verification technique in Chapter 3.3.1.

We use the small exponents test to batch verify the designated verifier re-encryption proofs. Small exponents of L-bit length are used to bind each corresponding challenge and response values in batching. A typical value of the security parameter L might be 20. For $j \in \{1, 2, \ldots, m\}$, the batch verification is implemented to check a zero-knowledge proof of knowledge of the following equation.

$$\log_g \prod_{j=1}^{m} \left(\frac{\alpha'_j}{\alpha_j}\right)^{t_j} = \log_y \prod_{j=1}^{m} \left(\frac{\beta'_j}{\beta_j}\right)^{t_j} \tag{6.2}$$

The j^{th} re-encryption authority produces a non-interactive proof to Equation 6.2 as follows:

1. Selects at random the values of $\tau_j, u'_j, w'_j \in \mathbb{Z}_q$.
2. Commits to the generator g and public key y as $(\gamma_{j,1}, \gamma_{j,2}) = (g^{\tau_j} \bmod p, y^{\tau_j} \bmod p)$, and simulates the knowledge of the private key of y_i owned by the designated verifier by computing $\gamma_{j,3} = g^{w'_j t_j} y_i^{u'_j t_j} \bmod p$.
3. Produces a small exponent t_j and a challenge u_j by using a one-way collision-resistant hash function H as $(t_j, u_j) = H(\gamma_{j,1}, \gamma_{j,2}, \gamma_{j,3}, \alpha'_j, \beta'_j)$, where the range of H is $\{0,1\}^L \times \mathbb{Z}_q$.
4. Calculates the response $w_j = \tau_j - t_j r'_j (u_j + u'_j) \bmod q$.
5. Sends the values of $(\gamma_{j,1}, \gamma_{j,2}, \gamma_{j,3}, u'_j, u_j, w_j, w'_j, t_j)$ to the designated verifier.

Verification of the proofs can be batched by checking the following:

$$
\begin{aligned}
(t_j, u_j) &\stackrel{?}{=} H(\gamma_{j,1}, \gamma_{j,2}, \gamma_{j,3}, \alpha'_j, \beta'_j) \\
\prod_{j=1}^{m} \gamma_{j,1} &\stackrel{?}{=} g^{\sum_{j=1}^{m} w_j} \prod_{j=1}^{m} \left(\frac{\alpha'_j}{\alpha_j}\right)^{t_j(u_j+u'_j)} \bmod p \\
\prod_{j=1}^{m} \gamma_{j,2} &\stackrel{?}{=} y^{\sum_{j=1}^{m} w_j} \prod_{j=1}^{m} \left(\frac{\beta'_j}{\beta_j}\right)^{t_j(u_j+u'_j)} \bmod p \\
\prod_{j=1}^{m} \gamma_{j,2} &\stackrel{?}{=} g^{\sum_{j=1}^{m} w'_j} y_i^{\sum_{j=1}^{m} t_j u'_j} \bmod p
\end{aligned}
$$

Cheating is possible if a dishonest authority can guess the value of the small exponents and the value of the challenge. The probability of guessing the challenge is negligible (q^{-1}). Hence, the probability of cheating the batch verification is similar to the probability of guessing the value of the small exponents 2^{-L} (refer back to Chapter 3.1).

Note that when the verification is accepted (Equation 6.2 is satisfied), further checking needs to be performed to verify that both values of α'_j and β'_j are in the group G, the subgroup of $\mathbb{Z}_p^*$ of order q. This can be performed efficiently by

using the Legendre symbol as follows:

$$\left(\frac{\alpha'_j}{p}\right) = 1$$
$$\left(\frac{\beta'_j}{p}\right) = 1$$

If either $\alpha'_j \notin G$ or $\beta'_j \notin G$, then the value that is not in G is simply converted to be in G by multiplying by -1.

Should the batch verification fail, divide and conquer, cut and choose (select a subset of the proofs at random to verify), binary search method [PMPS00], or even individual verification, can be performed to identify the dishonest re-encryption authority and discard his/her partial re-encryption.

6.3 Model 1 - with A Single Administrator

We use the proposed optimistic mix-network in Chapter 5.3 to construct model 1. Efficiency improvement is obtained over the scheme by Lee *et al.* [LBD+04] by employing the optimistic mix-network in Chapter 5.3 and combining the role of the first mix server with the role of a randomiser.

Figure 6.1 offers an overview of our Model 1. Participants of the scheme are voters, a trusted administrator (re-encryption authority), an optimistic mix-network, and tally (decryption) authorities. The scheme employs a two-way untappable communication channel to be used by the voters and the administrator.

6.3.1 The Protocol

Based on the scheme by Pedersen [Ped91], a threshold version of ElGamal cryptosystem is employed (for threshold decryptions). The appropriate parameters p, q, g and $y = g^x$ are made public, while the secret key x is shared among tally authorities for opening the mixed ballots. Each voter has a public-private key pair through an already established key distribution mechanism, such as Public Key Infrastructure (PKI), where $y_i = g^{x_i}$ corresponds to the public key of the i^{th} voter.

Focusing on the interaction between the voter and administrator (**voting phase**, step 1 in Figure 6.1), the protocol is as follows. For simplicity, we provide

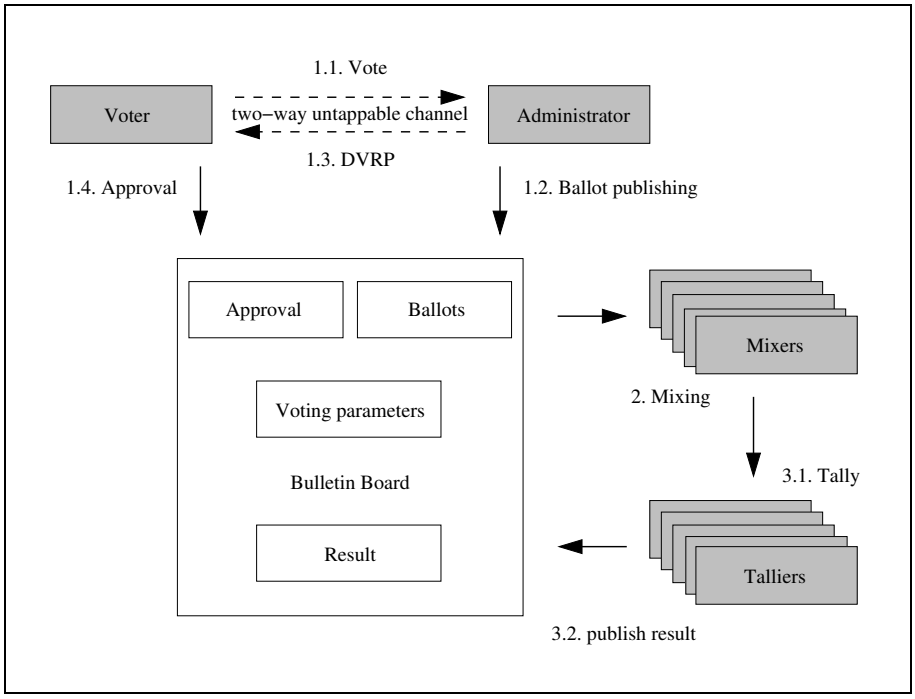

Figure 6.1: An overview of Model 1.

details of the protocol for one voter ($i = 1$) and omit the subscript i (n number of voters, $i \in \{1, 2, \ldots, n\}$).

1. **Vote casting (using two-way untappable channel)**:
   Each voter chooses and encrypts a vote s as a ballot $c = (\alpha, \beta) = (g^r, sy^r)$, where $r \in \mathbb{Z}_q$ is a value selected at random by the voter. The ballot (α, β) is then sent to the administrator with the voter's signature. The administrator checks the eligibility of the voter and the validity of the voter's signature, and accepts or rejects the submitted ballot accordingly.

2. **Ballot publishing**:
   After the voting period has ended, the administrator re-encrypts each ballot using a new random value r' as $(\alpha', \beta') = (\alpha g^{r'}, \beta y^{r'})$, and posts the re-encrypted vote (α', β') in a random order to the bulletin board.

3. **DVRP (using two-way untappable channel)**:
   The administrator provides each voter with a DVRP which proves the correctness of the re-encryption. Using the DVRP, the administrator personally proves to the voter his/her knowledge of either the random value of r' used for re-encryption OR the private key of the voter x_i ($y_i = g^{x_i}$, refer back to Section 6.1.3).

4. **Approval**:
   Each voter checks the validity of the DVRP as in equation 6.1 and posts an approval message with his/her signature on the bulletin board if the DVRP is accepted, or refutes otherwise. The approval message format can be pre-agreed in the system such that it is fresh but does not include any personal information which can be used as a receipt. For example, a voter can sign the hash value of all the published ballots.

Afterward, the re-encrypted ballots are forwarded as inputs to the optimistic mix-network (**mixing phase**, step 2 in Figure 6.1). Either individual or global verification can be employed according to the confidence level (refer back to Chapter 5.3).

Outputs of the mix-network are then decrypted by a quorum of tally authorities. The plaintext votes are published on the public bulletin board along with the voting result (**tally phase**, step 3 in Figure 6.1). Correctness of the threshold decryption process is publicly verifiable as each tally authority publishes a proof of correct partial decryption on the bulletin board, by proving the knowledge of his/her share of the secret key used to produce the partial decryptions.

The scheme employs a *trace-back protocol* (refer back to Chapter 5.3) to reveal a dishonest entity should an invalid vote be found. Cheating is discouraged since a dishonest entity will be identified and sanctioned accordingly. Note that in this scenario, vote content of a ballot remains a secret unless the administrator is also a buyer (i.e. the voter was forced to use a particular random value beforehand). Otherwise, a dishonest administrator can only use the signature of a voter to prove that a particular ballot came from the voter without knowing its vote content (the vote content of the first ballot submitted to the administrator).

6.3.2 Trust and Security Issues

Since the scheme employs a single trusted administrator, a compromised administrator can collude with the voter to buy/sell the vote (or act as an agent for the buyer). A dishonest voter can send the value r to the administrator, along with the ballot (α, β), through the two-way untappable channel during re-encryption. The administrator can then check that a particular vote s is contained in the ballot by repeating the encryption process.

Also, a collusion between an administrator and a voter allows the administrator to submit a ballot on behalf of the voter, where the voter approves the ballot

irrespective of the proof from the administrator.

This particular cheating method cannot be detected since a two-way untappable channel is employed and the ballot is not tampered in any way. Both privacy and receipt-freeness are compromised.

We propose a novel way to alleviate this issue by employing multiple administrators instead of employing a single trusted administrator. Thus, the trust is distributed among a threshold (quorum) number $t + 1$ from all m administrators (t-out-of-m). A voter splits a vote and constructs partial ballots for the administrators. The administrators re-encrypt and produce DVRPs for the voter. Receipt-freeness can still be satisfied up to a collaboration of t administrators.

6.4 Model 2 - with Multiple Administrators

We present our second model in this section. Note that the privacy issues in the first model remain in the second model. The first mix server still needs to be trusted to maintain the secrecy of voter-vote relationships. However, our second model offers more robustness in receipt-freeness by employing a threshold of administrators to perform re-encryption of the initial ballot. A threshold of administrators must collude to provide evidence of a vote and compromise receipt-freeness.

For simplicity, we describe the threshold re-encryption protocol for one voter ($i = 1$), and omit the subscript i.

1. **Vote casting (using two-way untappable channels)**:
   The voter chooses a value $r_j \in \mathbb{Z}_q$ at random, and a vote $s = g^{\epsilon}$ (the list of valid votes can be published prior to the voting phase). For $j \in \{1, 2, \ldots, m\}$, the value ϵ is then shared using (t, m) Shamir's secret-sharing scheme into m values ϵ_j. Each of the partial votes $s_j = g^{\epsilon_j}$ is then encrypted as partial ballots $(\alpha_j, \beta_j) = (g^{r_j}, s_j y^{r_j})$. Each of the partial ballot (α_j, β_j) are then sent to the corresponding administrators along with the signature of the voter. Each administrator checks the eligibility of the voter and the validity of the signature. The partial ballot is either accepted or rejected depending on the verification result.

2. **Ballot publishing**:
   At the end of the voting period, each of the administrators re-encrypts

each partial ballot (α_j, β_j) using a new random value r'_j as $(\alpha'_j, \beta'_j) = (\alpha_j g^{r'_j}, \beta_j y^{r'_j})$, and posts the re-encrypted partial ballot (α'_j, β'_j) to the bulletin board, tagged with the identity of the voter for ballot identification and partial ballots combining, and a signature of the re-encrypted partial ballot. Note that since each partial ballot is tagged for partial ballots combining, shuffling is not provided by the administrators.

3. **DVRP (using two-way untappable channels)**:
   Each administrator provides the voter with a DVRP which proves the correctness of the partial re-encryption. Using the DVRP, each administrator individually proves to the voter his/her knowledge of either the random value of r'_j used for re-encryption or the private key of the voter x_i ($y_i = g^{x_i}$, refer back to Section 6.2.2).

4. **Batch DVRP verification**:
   The voter batch verifies the proofs by performing the checks as in Section 6.2.2.

5. **Approval**:
   A period of time is provided for voters to mark the invalid partial ballot to be excluded in the partial ballots combining step.

6. **Partial ballots combining**:
   Correct partial ballots are combined from a quorum of partial ballots as in Section 6.2.1.

6.5 Analysis

Security and efficiency analysis of our two proposed models are presented in this subsection.

6.5.1 Security

Our proposed models are based on known building blocks whose security properties are already established. This subsection discusses the overall security of our models. We analyse our proposed models based on the security requirements discussed in Chapter 2.7.

- **Privacy**: In the first model, ballots are randomised and mixed first by the administrator and then by the mix servers. If at least one of these entities remains honest, privacy of voters is maintained. A threat in privacy can occur when a specific invalid ballot is traced back to the voter. If the invalid ballot is traced back only to the mix servers, privacy is maintained since we assume that the administrator does not disclose voter-vote relationship. Since an optimistic mix-network is employed, the second model also inherits this problem. The first mix server is assumed to be honest; a verifiable mix-network can be employed if this can not be reasonably assumed.

- **Receipt-freeness**: In the first model, since a voter's ballot is randomised additionally by the administrator, a voter loses his knowledge of the randomness of the encrypted ballot and cannot construct any receipt. Also, the voter cannot transfer the DVRP of the administrator to any third party. This is because it is a personal proof and the voter can construct the proof using his private key.

  Since a two-way untappable channel is used between the voter and the administrator, a buyer cannot observe the communication between the voter and administrator during the voting phase.

  Compared to employing a trusted administrator in the first model, the second model offers a stronger notion of receipt-freeness. This is by employing multiple administrators performing threshold re-encryptions. This model can accommodate up to a threshold number t of dishonest administrators, and can still maintain receipt-freeness.

- **Accuracy**: The plaintext votes output by the mix-network are public. Hence, accuracy of the voting result is straight-forward.

- **Fairness**: Since voting is only allowed during the voting period prior to mixing and tallying, the fairness of voting is guaranteed.

- **Eligibility**: The list of eligible voters is made public and only authenticated voters are allowed to participate.

- **Non reusability:** Voters can vote only once since they participate in voting with their signatures. Any misbehaviour by the administrator, for example, addition of ballots, is prevented, since a voter's approval is required for a ballot to be valid.

- **Robustness**: Using the individual mix server verification, backup mixing is possible when an invalid mixing in the proof of product is detected. Also, ballots opening are robust as a threshold decryption is employed.

- **Verifiability**: In the voting phase, a voter can personally verify the correctness of administrator's randomisation by checking the DVRP. Correct mixing operation is publicly verifiable as anyone can observe and verify the equality of the product of input and output ballots. The tally phase is publicly verifiable.

In both of the proposed models, a corrupt mix server can disrupt the voting by invalidating some ballots during the mixing phase. For example, from two different inputs, a mix server produces two outputs which are respectively the product of the two inputs and a re-encryption of 1. As the product of inputs and outputs is still preserved, the proof of correct mixing is accepted but the recovered messages are invalid. Note that this is an inherent weakness in an optimistic mix-network by using a proof of product to prove a correct mixing operation (refer back to Chapter 5.1.4).

However, the cheating mix server will be identified using the trace-back protocol and can be sanctioned accordingly. When a trace-back occurs to a specific mix server in the middle of the mix-network, the voter-vote relationship will not be revealed. When an invalid ballot is traced back to the first mix server, the administrator knows the voter-vote relationship.

In the first model, we assume that the administrator is a reputable entity and does not disclose his/her knowledge (of voter-vote relationship) when a trace-back occurs. In the second model, a stronger notion of receipt-freeness is achieved by employing multiple administrators.

We assume that the first mix server is honest such that privacy is not compromised. Alternatively, a verifiable mix-network can be employed if such assumption is considered to be too strong.

6.5.2 Efficiency

Compared to a voting scheme based on the mix-network by Golle *et al.*, our first model is more efficient both in computational cost (in terms of the number of modular exponentiations required) and communicational cost (in terms of the message size in bits) as shown in Table 6.1 and Table 6.2 respectively.

Table 6.1: A computational cost comparison of our first model against a voting scheme based on the mix-network of Golle *et al.*, where n denotes the number of voters.

	Entity	Operation	Proposed	Golle *et al.*
Voting phase	Voter	Encrypt	2	8
		Verify (DVRP)	6	N/A
	Administrator	Re-encrypt	2	N/A
		Prove (DVRP)	4	N/A
Mixing phase	Mix server	Re-encrypt	$2n$	$6n$
		Prove	2	6
	Public	Verify	6	18

Efficiency is improved mainly because our optimistic mix-network scheme uses a single encryption, while the scheme by Golle *et al.* uses three encryptions for the double enveloping.

In the voting phase, our first model requires each voter to encrypt the vote once (2 modular exponentiations), to submit it to the administrator, and later to verify a DVRP from the administrator (6 modular exponentiations). The scheme by Golle *et al.* [GZB+02] requires each voter to perform a double encryption (8 modular exponentiations). We do not compare the cost for digital signature, since it is an essential operation and requires the same cost. Our scheme is thus a little more computationally expensive for the voter.

In the mixing phase, our optimistic mix-network requires three times less computational cost compared with the scheme by Golle *et al.*, since our scheme uses a single encryption while the scheme by Golle *et al.* uses three encryptions for the double enveloping process. In terms of proof of product (POP), our scheme requires three times less computational cost, if we use the individual mix server verification. If we use the global verification (refer back to Chapter 5.3), our scheme is much more efficient, since only the initial input product and final output product are decrypted by a quorum of decryption authorities and compared.

In the tally phase, our scheme only requires one threshold decryption for each ballot, where the scheme by Golle *et al.* requires four threshold decryptions.

The size of a ballot in our scheme is $2\log_2 p$ bits as we use a single ElGamal encryption, and the DVRP by the administrator is $4\log_2 q$ bits in length. The ballot size in the scheme by Golle *et al.* is $6\log_2 p$ bits as they use double encryption. In the mixing phase, our scheme requires three times less bandwidth as

Table 6.2: A communicational cost comparison of our first model against a voting scheme based on the mix-network of Golle *et al.*, where n denotes the number of voters.

	Entity	Operation	Proposed	Golle *et al.*
Voting phase	Voter	Encrypt	$2\log_2 p$	$6\log_2 p$
	Administrator	Proof (DVRP)	$4\log_2 q$	N/A
Mixing phase	Mix server	Re-encryption	$2n\log_2 p$	$6n\log_2 p$
		Proof	$2\log_2 p + \log_2 q$	$6\log_2 p + 3\log_2 q$

compared to the scheme by Golle *et al.*. However, in the voting phase our scheme requires interactive communication between voters and the administrator since voters have to cast their ballots first and approve them later.

For our proposed model 2, the increase in computational cost (especially for each voter) is linear in the number of administrators due to partial ballot constructions (each costs 2 modular exponentiations). The increase in communicational cost is also linear in the number of administrators.

6.6 Summary

Two models of efficient and receipt-free mix-network based voting schemes have been presented. We successfully combined two mix-network based voting schemes by Lee *et al.* [LBD+04] and Golle *et al.* [GZB+02] to provide both efficient mixing and receipt-freeness at the same time. In our first model, the administrator provides both randomisation service and mixing service in the voting phase.

Although an optimistic mix-network is employed, and an invalidation attack by a mix server is possible, the public trace-back procedure discourages any misbehaviour by the administrator or the mix server. Because of its efficiency, the proposed voting scheme can be preferred in practical real world election applications. An example is in political elections, in which the administrator is considered to be a reputable entity and a timely tally is required. Moreover a mix-network based voting scheme offers more flexibility on the ballot structure, such as preferential voting.

In addition, our proposed model 2 offers a stronger notion of receipt-freeness and robustness. It employs threshold re-encryptions and batch verification techniques. The first technique is to re-encrypt partial ballots, and the second is to

batch verify the designated verifier re-encryption proofs.

Both models presented in this chapter offer a framework for providing receipt-freeness efficiently. Future work is possible by employing different mix-network schemes to obtain different security and efficiency levels. Based on the work from this chapter, it is also possible to produce a receipt-free homomorphic encryption based voting scheme. A recent mechanism providing receipt-freeness by Chaum [Cha04] can also be further examined.

The next chapter presents a hybrid framework to realise a cryptographic voting protocol accommodating a flexible ballot structure. It combines the use of homomorphic encryption and mix-network approaches. The homomorphic encryption approach is more efficient in the tally stage, where a mix-network allows write-in votes. The hybrid framework inherits these two unique properties from each approach.

Chapter 7

Voting using A Hybrid Approach

This chapter presents a hybrid cryptographic voting protocol. The protocol combines both approaches of homomorphic encryption (see Chapter 4), and mix-network (see Chapter 5). Both the multiplicative homomorphic voting scheme in Chapter 4.2 and the EBMG mix-network scheme in Chapter 5.2 are employed to form the hybrid scheme.

In this chapter, primitives in previous chapters are placed into a recent framework presented by Kiayias and Yung [KY04] to form the hybrid scheme.

The hybrid approach offers the benefit of efficient tallying from the use of the homomorphic encryption approach, while allowing write-in votes from the use of the mix-network approach. Thus, the hybrid approach allows the conduct of a secure secret-ballot election with a flexible ballot structure. An Australian Senate preferential system election is presented as a case study for the hybrid scheme.

7.1 Background

Cryptographic voting protocols in the literature are normally based on either homomorphic encryption or mix-network based approach. A recent paper by Kiayias and Yung [KY04] offers a framework for combining both of the approaches to accommodate a more flexible ballot structure. The combination is most suitable for voting systems allowing write-in votes, or voting systems with complex ballot rules.

This is suited for the Australian preferential system as described in Chap-

Table 7.1: The vector-ballot framework inherits two essential properties from homomorphic encryption and mix-network based approaches.

Approach	Efficient tallying	Allow write-in votes
Homomorphic Encryption	yes	no
Mix-networks	no	yes
Vector-ballot	**yes**	**yes**

ter 2.3.1. The use of both homomorphic encryption and mix-network approaches allows for combining the benefits of efficient tallying and allowing write-in votes. This approach is applicable to our research goals. The main benefits are illustrated in Table 7.1.

A straight-forward approach, to allow efficient tallying and also allow write-in votes, is to use both homomorphic encryption and mix-network approaches at the same time. Voters can either submit a vote using the homomorphic encryption approach, or using the mix-network approach. However, this approach requires grouping of voters into two groups. One uses the homomorphic encryption approach, and the other uses the mix-network approach. This may leak some voter-vote relationship information (refer back to Table 2.2).

For example, public verification of the submitted ballots allows anyone to decide which voters made a ballot using the homomorphic or mix-network approach. More information on a specific voter-vote relationship is revealed when a particular group of voters (using either the homomorphic or mix-network approach) is much larger than the other. In the homomorphic encryption group, a vote is selected from a small pre-determined set of choices. In the mix-network group, the anonymised individual plaintext votes are made public.

In this scenario, a better approach to offer better protection to hide voter-vote relationships is by using the vector-ballot framework by Kiayias and Yung [KY04]. This framework allows more protection of voter-vote relationships in the smaller-sized group category in such a scenario.

Based on the modified additive ElGamal cryptosystem (refer back to Chapter 4.1.2), the vector-ballot framework contains three main ideas. They are: provably consistent vector-ballot encoding, shrink-and-mix network, and punch-hole-vector-ballot. Two schemes are provided in their paper. One is realised by using the first two main ideas. The second is an extension of the first scheme us-

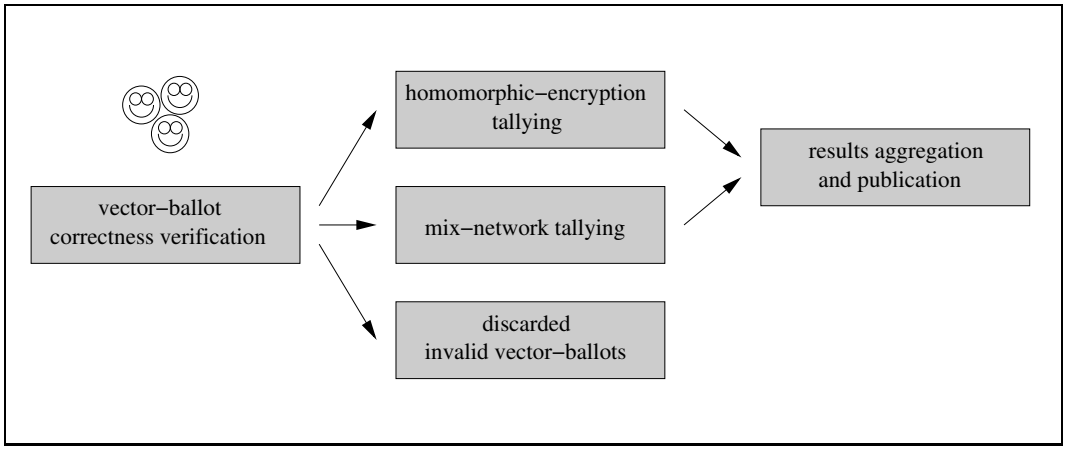

Figure 7.1: Combining both the homomorphic-encryption and mix-network approaches in a vector-ballot framework.

ing the punch-hole-vector-ballots idea. As there are n vector-ballots, we describe details for one vector-ballot and omit the vector-ballot index for simplicity.

In *provably consistent vector-ballot encoding*, the vector-ballot consists of three parts containing ciphertexts $\langle c_1, c_2, c_3 \rangle$. The first ciphertext c_1 contains a pre-determined voting choice s_1 in the pre-determined set Q, where $s_1 \in Q$. The second ciphertext c_2 contains a flag s_2 indicating whether the ballot contains a pre-determined vote ($s_2 = 0$) or contains a write-in vote ($s_2 = 1$). The third ciphertext c_3 contains the write-in vote s_3.

The first ciphertext c_1 is to be processed using homomorphic encryption. The second ciphertext c_2 indicates whether the vector-ballot is to be forwarded to the homomorphic encryption tallying, or to be forwarded to a mix-network. The third ciphertext c_3 is to be forwarded to a mix-network if required (according to the flag in c_2).

Correct ballot construction is proved by constructing a zero-knowledge (ZK) proof for the following premise (see ZK proof construction in Appendix C):

$$(s_1 \in Q \wedge s_2 = 0 \wedge s_3 = 0) \vee (s_1 = 0 \wedge s_2 = 1) \tag{7.1}$$

All first ciphertexts c_1 are forwarded for homomorphic encryption tallying after the proofs are verified.

In *shrink-and-mix networks*, the second ciphertext c_2 of the vector-ballots are homomorphically combined and decrypted in small groups. A decryption result of 0 indicates that there are no write-in votes in that small group of vector-ballots.

Thus, the mix-network part of the vector ballots c_3 for that group are discarded. Otherwise, the decryption result indicates the number of write-in votes in that small group of vector-ballots. Thus, the mix-network part of the vector-ballots c_3 are forwarded to the mix-network.

This approach "shrinks" the original number of vector-ballots to a smaller number of inputs to the mix-network. Hence, mixing of vector-ballots by the mix-network is more efficient when a majority of the vector-ballots contains predetermined voting choices (in the first ciphertext c_1). This is because computational cost in a mix-network is proportional to the number of its inputs.

In *punch-hole-vector-ballots*, each candidate has his/her own summation register encrypted separately (punched) for the homomorphic encryption part. Each of the three parts in the vector-ballot now contains: ciphertexts each containing a vote for a candidate (encoded using "1" or "0"), a ciphertext containing the homomorphic or mix-network flag, and a ciphertext containing the write-in vote.

This approach is suited towards voting with a large number of candidates or choices. The argument is that the use of separate registers relaxes the burden of voters by allowing them to construct smaller ciphertexts. It was noted that the computational complexity for proving a correct punch-hole-vector-ballot encoding is higher than the normal vector-ballot encoding. However, the efficiency gain argument is that tallying will be more efficient as a brute-force search (using the modified additive ElGamal cryptosystem) is more efficient over smaller register size.

We note that an alternative to the punch-hole-vector-ballots method is to tally the homomorphic votes using small-sized groups, and accumulate the result of the small-sized groups to obtain the voting result. This approach would offer lower computational cost for the voters. Also, as more detail on the punch-hole-vector-ballots approach was not provided, we only focus on the approach using the provably consistent vector-ballot encoding and the shrink-and-mix-network. Figure 7.1 offers a high-level view of a cryptographic voting protocol using the vector-ballot approach.

7.2 The Hybrid Scheme

A more detailed system diagram of the vector-ballot framework using our primitives is shown in Figure 7.2.

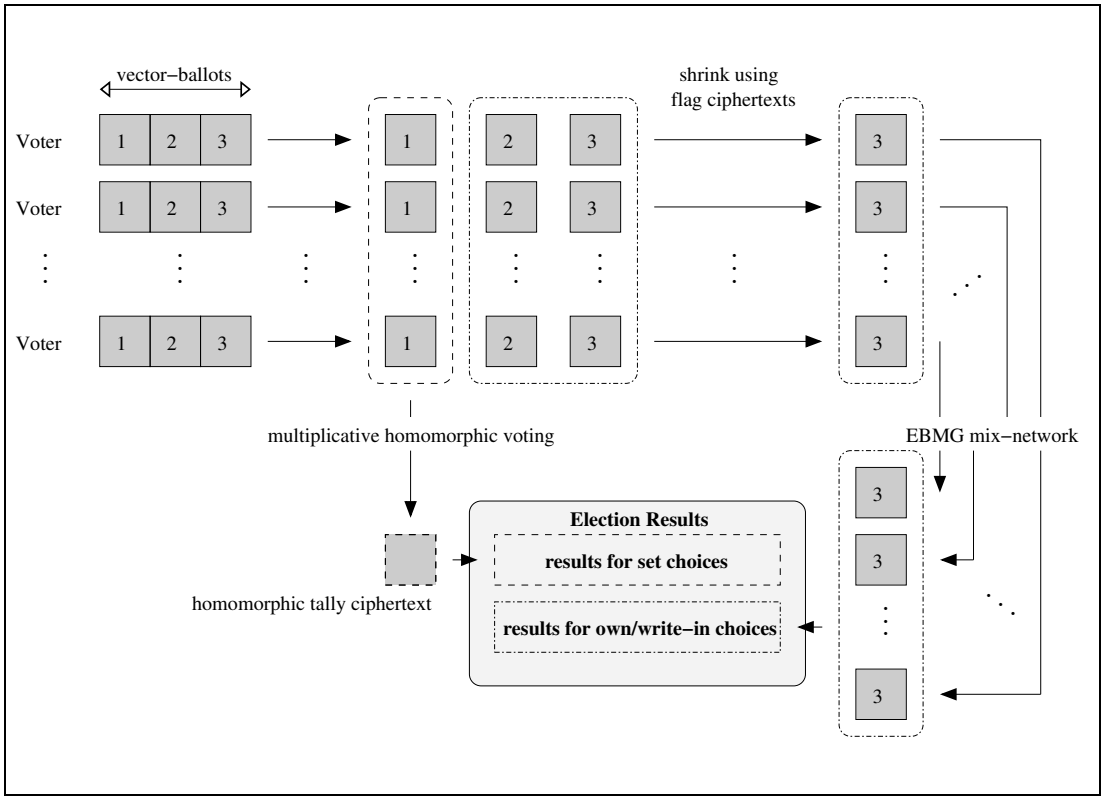

Figure 7.2: A high-level diagram of vector-ballots processing using our primitives.

Note that by allowing a write-in vote, it is possible for a voter to include identifying information in the write-in vote (e.g. using a digital signature). This information reveals his/her particular voter-vote relationship, and compromises the receipt-freeness requirement. Receipt-freeness is an inherent problem in write-in votes. As such a vote can be invalidated, there is still an inherent problem of buying/selling invalid votes.

Both the multiplicative homomorphic voting scheme from Chapter 4.2 and EBMG mix-network from Chapter 5.2 are employed in the framework by Kiayias and Yung to realise the hybrid scheme. The ElGamal cryptosystem used is as in Chapter 4.2 (i.e. not modified for additive homomorphic).

The flag s_2 is represented using 0 or 1 in the original scheme by Kiayias and Yung. This is because the modified ElGamal cryptosystem (additive homomorphism, refer back to Chapter 4.1.2) is employed. Since the non-modified ElGamal cryptosystem (multiplicative homomorphism, refer back to Chapter 4.2) is employed in our scheme, the flag s_2 is now represented using q_0 or q_1 to accommodate the multiplicative homomorphism property.

Below are descriptions for the phases in our protocol. More details on both

primitives used in this approach have been provided in their respective sections in this thesis.

1. **Preparation phase**:
   Parameters for a threshold ElGamal cryptosystem and the set of $K+1$ suitable elements $\{q_0, q_1, q_2, \ldots, q_K\}$ are generated for the multiplicative homomorphic voting and EBMG mix-network as per Chapter 4.2 and Chapter 5.2 respectively. The pre-determined voting choices is in the set $Q = \{q_1, q_2, \ldots, q_K\}$. The public key g and $y = g^x$ are made public, and the secret key x is shared among a number of tally authorities in a threshold manner.

2. **Voting phase**:
   A ciphertext c hides a secret message s, and is constructed using a random value r as $c = (\alpha, \beta) = (g^r, sy^r)$. The vector-ballot consists of ciphertext triplets $\langle c_1, c_2, c_3 \rangle$.

   - If casting a pre-determined voting choice, a voter constructs a vector-ballot $\langle c_1, c_2, c_3 \rangle$, where $s_1 \in Q$, $s_2 = q_0$, and $s_3 = q_0$. The voter then constructs the corresponding zero-knowledge proof. The proof construction cost $12K + 2$ exponentiations.
   - Otherwise, casting a write-in vote, a voter constructs the vector ballot $\langle c_1, c_2, c_3 \rangle$, where $s_1 = q_0$, $s_2 = q_1$, and s_3 containing the write-in vote. The voter then constructs the corresponding zero-knowledge proof as in the equation above. The proof construction costs $12K + 4$ exponentiations.

   Expanding the proof from Equation 7.1, correct vector-ballot construction is proved by constructing a zero-knowledge proof using Expression 7.2.

   The proof can be constructed by combining a proof of equality of discrete logarithms (using AND logic) inside a proof of 1-out-of-$K+1$ equality of discrete logarithms (using OR*logic*). Refer to Appendix C.4 on how to construct such a proof.

$$
\begin{array}{rcl}
(\log_g \alpha_1 = \log_y(\beta_1/q_1) \ \wedge \log_g \alpha_2 & = & \log_y(\beta_2/q_0) \wedge \log_g \alpha_3 = \log_y(\beta_3/q_0)) \\
& \vee & \\
(\log_g \alpha_1 = \log_y(\beta_1/q_2) \ \wedge \log_g \alpha_2 & = & \log_y(\beta_2/q_0) \wedge \log_g \alpha_3 = \log_y(\beta_3/q_0)) \\
& \vee & \\
& \vdots & \\
& \vee & \\
(\log_g \alpha_1 = \log_y(\beta_1/q_K) \wedge \log_g \alpha_2 & = & \log_y(\beta_2/q_0) \wedge \log_g \alpha_3 = \log_y(\beta_3/q_0)) \\
& \vee & \\
(\log_g \alpha_1 = \log_y(\beta_1/q_0) \ \wedge \log_g \alpha_2 & = & \log_y(\beta_2/q_1))
\end{array}
\tag{7.2}
$$

3. **Tally phase**:

   - **Homomorphic encryption** processing: Tally authorities check the published zero-knowledge proofs, and accept or reject each vector-ballot accordingly. Each correct vector-ballot proof verification costs $12K + 8$ exponentiations.

     For n valid vector-ballots, all of the first ciphertext part c_1 of the vector-ballots are combined for homomorphic decryption if $max(Q)^n < p$ where $max(Q)$ denotes the largest element in Q. Otherwise, the vector-ballots are combined in smaller-sized groups for this requirement to hold.

     The resulting combination is then decrypted in a threshold manner by a quorum of tally authorities. The decryption result is then factorised as in Chapter 4.2 to obtain the homomorphic encryption voting result. The decryption for each tally authority costs one exponentiation. Its corresponding proof costs two exponentiations. The homomorphic tallying costs $3\lceil \log_p max(Q) \rceil$ exponentiations.

   - **Shrink-and-mix network** processing: The second ciphertext part of the vector-ballots c_2 are homomorphically combined (multiplicative) and decrypted in smaller groups of size n'. The value of n' denotes the number of ("shrunken") inputs to the mix-network, where shrinking was performed in groups of size $\hat{n}^1$. There are a total of $(n/\hat{n})$ groups.

The processing of the flag (c_2) requires each ($n/\hat{n}$) homomorphic processing and ($n/\hat{n}$) decryptions. Each homomorphic processing requires a proof construction with a cost of $3\lceil \log_p max(q_0, q_1) \rceil$ exponentiations. Each group decryptions and their corresponding proofs cost $\hat{n}$ and $\hat{n}+2$ exponentiations (using batch verification) respectively.

If the group decryption indicates that there is at least one vector-ballot containing a write-in vote, then all the third ciphertext part of the vector-ballots c_3 in the group are forwarded to the EBMG mix-network and are processed as in Chapter 5.2. Otherwise, they are discarded.

For each mix server (two-rounds EBMG mixing), shuffling of the ciphertexts (shrunken) costs $8(\log_2 n')^2 + 12(n' + 1)$ exponentiations. Decryptions and their corresponding proofs cost n' and $n'+2$ exponentiations (using batch verification) respectively. Validity of the write-in votes output by the EBMG mix-network are then verified without any cryptographic processing. Only valid write-in votes are included in the mix-network tally to obtain the mix-network voting result.

Mix-network processing of vector-ballots may be performed after tallying of all the homomorphic part is complete. Using this scenario, the voting result may be revealed by just using the homomorphic tally if the winning margin for the candidate is larger than the number of write-in votes identified.

Verification of the correct mixing operation (two-rounds EBMG mixing) costs $20(n'-1)$ exponentiations. Verification of a correct decryption operation for each the homomorphic and mix-network processing requires $3n+4$ and $3n'+4$ exponentiations respectively. The value of n' denotes the total number of vector-ballots forwarded to the mix-network (accumulation of $\hat{n}$).

Afterward, the official total tally is obtained by combining both the homomorphic and mix-network tally. All the homomorphic and write-in tally results are then made publicly available.

[1]We recommend that $\hat{n}|n$, such that all groups have equal size. Otherwise, the level of privacy for ballots in the last few groups may be less than the other groups.

Table 7.2: A computational cost comparison for each voter in terms of the number of modular exponentiations required.

Operation	Homomorphic Enc. (multiplicative)	Mix-network (2-rounds EBMG)	Hybrid
ballot construction	2	2	6
proof construction	$4K-2$	2	$12K+4$
proof verification	$4K$	4	$12K+8$

7.3 Analysis

Security of the hybrid voting protocol is inherited from both the security of homomorphic encryption and the security of the mix-network scheme employed. Both the security of the multiplicative homomorphic encryption voting scheme and the security of the EBMG mix-network scheme have been discussed in Chapter 4.2.2 and Chapter 5.2.4 respectively.

The zero-knowledge proof of correct vector-ballot construction follows a standard Σ-protocol [CD95], and thus has a special soundness property as proven by Cramer and Damgård [CD95]. The security of the batching operation has been discussed in Chapter 3.

Table 7.2 compares the computational cost required by a voter in the three different approaches. Table 7.3 compares the computational cost required by a tally authority in three different approaches. In those tables, the value of n denotes the total number of vector-ballots, $\hat{n}$ denotes the group size for the "shrink-and-mix" process, n' denotes the shrunken total number of vector-ballots after the "shrink-and-mix" process and m denotes the number of mix servers or tally authorities.

Note that the computational cost for proof and verification for each tally authority employs batch technique in Chapter 3. As the hybrid approach bears the computational cost from both the homomorphic encryption and mix-network approaches, a lower cost can be obtained by using more efficient homomorphic encryption and mix-network schemes. The computational cost required to process the ciphertexts containing the flag (for homomorphic or mix-network tallying) is also considered in both tables.

For the homomorphic encryption approach (second column in both tables), the computational cost is mainly contributed to proof constructions and verifica-

Table 7.3: A computational cost comparison for each tally authority in terms of the number of modular exponentiations required.

Operation	Homomorph. Enc. (multiplicative)	Mix-network (2-rounds EBMG)	Hybrid
shuffle (re-encrypt)	not applicable	pre-computed	pre-computed
shuffle proof	not applicable	$8(\log_2 n)^2$ $+12(n+1)$	$8(\log_2 n')^2$ $+12(n'+1)$
shuffle verification	not applicable	$20(n-1)$	$20(n'-1)$
decryption	1	n	$n' + (n/\hat{n}) + 1$
dec. proof	2	$m+2$	$m+4$
dec. verification	$3m+4$	$3m+4$	$6m+8$
homomorph. proc.	$3\lceil n \log_p max(Q) \rceil$	not applicable	$3\lceil n \log_p max(Q) \rceil$ $+3(n/\hat{n})$ $\lceil \hat{n} \log_p max(q_0, q_1) \rceil$

tions of correct vector-ballot. The cost is proportional to the number of allowed choices K (e.g. candidates). Afterward, the tallying is very efficient as only the combination of the ballots are required to be decrypted to reveal the voting result.

For the mix-network approach (third column in both tables), the computational cost is mainly contributed to the shuffling and decryption proofs and verifications of mixed vector-ballots. The cost is typically proportional to the number of input ciphertexts n. However, ballot construction is very efficient as it does not require such complicated proofs and verifications of correct ballot construction compared to the homomorphic encryption approach.

7.4 A Preferential Voting Case Study

As illustrated in Table 7.1, the hybrid approach combines two essential properties. The first is efficient tallying from the use of the homomorphic encryption approach. The second one is accommodation of write-in votes from the use of the mix-network approach.

The hybrid approach can be applied to the preferential system for Australian Senate election (refer back to Chapter 2.3.3). Votes from a pre-determined set of party preferences are tallied using the homomorphic encryption approach. At the same time, votes not in the pre-determined set (write-in votes) are tallied using the mix-network approach.

Since a pre-determined set of preferences is not used in the official ballot papers in the elections for the House of Representatives, the hybrid approach offers no additional advantage from the use of a mix-network approach. Hence, the Australian elections for the House of Representatives are not discussed in this case study.

The case study presented in this section is specific to the Australian Senate Election using our hybrid protocol. A comparison is provided to the use of only the mix-network approach for the election. The use of only the homomorphic encryption approach is not included in the comparison since it is not possible to accommodate all the preferences for the election as discussed in Chapter 4.3.

We use the following parameters for the case study.

- The value of $K = 20$ denotes the pre-determined set of party preferences, as there are 20 political parties. These pre-determined set of party preferences are for the first ciphertext c_1 in the vector-ballot. They are later tallied using the homomorphic encryption approach.

- There are 60 candidates, indicating a total of possible preferences of 60!. Each of the possible preferences can be represented by a single integer. In this case, the size of the integer is $\log_2 60! = 273$ bits. A voter is to select his/her own preference from the space of all possible preferences when his/her vote is not in the pre-determined set of party preference. These non pre-determined preferences are for the third ciphertext c_3 in the vector-ballot. They are later tallied using the mix-network approach.

- We assume an average case scenario, where the value of $n = 2500000$ denotes both the number of voters and the number of valid vector-ballots. For simplicity, all the vector-ballots are assumed to be valid.

- A well-known statistic for the Australian Senate elections (refer back to Chapter 2.3.3) indicates that 95% of voters select a vote from the pre-determined set of party preferences. This means that the group-size for voters casting his/her own preference (write-in vote) is much smaller than the group-size for voters casting a party preference. From the statistic, the probability that a voter selects his/her own preference (inside c_3, the write-in vote part of the ballot) is 20^{-1}. In a worst-case scenario, one out of every 20 ballots contains a write-in vote (for 2500000 ballots).

Table 7.4: A computational cost comparison for each voter in an Australian Senate election scenario.

Operation	Mix-network (2-rounds EBMG)	Hybrid
ballot construction	2	6
proof construction	1	244
proof verification	2	248

For efficiency reasons, we define the size of the groups for "shrinking" the mix-network inputs to be 10. Where each group contains exactly one write-in vote, the value of $n' = 1250000$ denotes the number of vector-ballots forwarded to the mix-network.

In the best case scenario, this grouping hides every one ballot containing a write-in vote in a group of $\hat{n} = 10$ ballots. This also means that all the 125000 (5%) ballots each containing a write-in vote is hidden among a total of $n' = 1250000$ ballots. Note that there is a trade-off between the size of the group $\hat{n}$ and the privacy of a ballot from the smaller-sized group category.

- Let the value of $m = 10$ denotes both the number of tally authorities and the number of mix servers. For simplicity, the number of tally authorities and the number of mix servers are assumed to be the same.

- Also for simplicity, it is assumed that the value of $\lceil \log_2 2500000 \rceil = 22$, the value of $\lceil \log_2 1250000 \rceil = 21$, the value of $\log_p max(Q) = 0.006$, and the value of $\log_p max(q_0, q_1) = 0.001$.

Table 7.4 offers a computational cost comparison for each voter in an Australian Senate election scenario. The second column shows the computational cost required using only the 2-rounds EBMG mix-network. Total computational cost using this approach is 3 modular exponentiations for each voter, and 2 modular exponentiations for a tally authority (verifier). The third column shows the computational cost required using the hybrid scheme. Total computational cost using this approach is 250 modular exponentiations for each voter, and 248 modular exponentiations for an administrator (verifier).

Table 7.5 offers a computational cost comparison for each tally authority in

Table 7.5: A computational cost comparison for each tally authority in an Australian Senate election scenario.

Operation	Mix-network (2-rounds EBMG)	Hybrid
shuffle (re-encrypt)	pre-computed	pre-computed
shuffle proof	30003884	15003540
shuffle verification	49999980	24999980
decryption	2500000	1500001
dec. proof	12	14
dec. verification	34	68
homomorphic proc.	not applicable	795000

an Australian Senate election scenario. Total computational cost using the 2-rounds EBMG mix-network is 32503896 modular exponentiations for each mix server, and 50000014 modular exponentiations for a verifier. Total computational cost using the hybrid approach is 17298555 modular exponentiations for a tally authority, and 25000048 modular exponentiations for a verifier. Using this figure, the hybrid approach is about 50% more efficient than the straight-forward mix-network approach. Moreover, an initial count of 95% of votes in the homomorphic group can be released after only 4.8% of the total computational effort.

Assume a worst case scenario for an average computer today to take one millisecond to compute one modular exponentiation. A voter requires 0.25 second to construct a vector-ballot and its corresponding proof in a hybrid scenario. Accordingly, a tally authority requires 17298.555 seconds, or 4.81 hour, to process the entire vector-ballots.

Note that the most computationally intensive task of a tally authority is to produce the shuffle proof. Also note that the computational cost for producing a shuffle proof is proportional to the number of input ciphertexts. Hence, a performance increase can be obtained by parallellising the shuffle operation using a smaller number of input ciphertexts, i.e. in smaller-sized groups. Further performance increase can also be obtained by parallellising output decryptions of the mix-network.

Dividing the mix-network inputs into four equally smaller groups of 312500 inputs, the computational cost of producing the shuffle proof becomes 3752900 modular exponentiations. The entire tally process for one tally authority now

requires less than two hours (about 100 minutes) to complete.

Better performance for the tally authorities can also be achieved by using a more powerful computer to process the vector-ballots. Also, as technology advances and more powerful computers are available, performance of the tallying process will also increase accordingly.

7.5 Summary

The homomorphic encryption and mix-network schemes presented in the thesis have been combined as a hybrid approach using the vector-ballot framework of Kiayias and Yung [KY04]. Both the multiplicative homomorphic encryption scheme from Chapter 4.2 and the EBMG mix-network scheme from Chapter 5.2 are employed.

This is to realise a hybrid approach suitable for the preferential system in Australia by allowing both a pre-determined party's preference in the homomorphic part, while also allowing voters to choose their own preference in the mix-network part. This hybrid approach offers the benefit of efficient tallying while allowing write-in votes.

The hybrid scheme is suited for the scenario of senate elections in the Australian federal elections as there are a large number of candidates to be given a preference, or choose from a much smaller number of pre-determined party preferences.

We observe that more efficiency gain may be obtained by using batch proof and verification on the correct vector-ballot proofs. Our application of batching techniques has been based on "AND logic" zero-knowledge proof construction. Future study can be performed on batching techniques for "OR logic" zero-knowledge proof construction. This can be applied to constructing and verifying proofs of correct homomorphic ballot construction. This offers more efficiency for voters.

Chapter 8

Summary and Research Directions

This thesis contains new work in the area of secure secret-ballot electronic voting allowing a flexible ballot structure, specifically in accommodating the Australian federal elections. Areas related to cryptographic voting protocols have been studied, a case study on preferential voting system has been performed, and novel techniques and improvements in some areas have been proposed.

The next section provides a summary of our research and an outline of the main contributions. A brief description of each contribution is recalled.

Afterward, discussions on potential future works are presented. A short conclusion is also provided.

8.1 Summary of Research

An electronic voting system named eVACS has been analysed. The conclusion is that the use of cryptographic primitives should be one of the fundamental mechanisms implemented to provide security. It complements physical security by providing logical security. The analysis has been presented in Chapter 2.

This research is concentrated in cryptographic voting protocols. A cryptographic primitive (batch), and details of cryptographic voting protocols (homomorphic encryption, mix-network, and hybrid) have been studied. Table 8.1 presents a summary of main contributions in this research.

Table 8.1: Summary of main contributions.

Topic	Sub-topic	Contribution
Electronic voting system	evaluation	review of eVACS
Cryptographic primitives	batch theorems	new batch theorems
		theorems applications to zero-knowledge protocols
Electronic voting framework	homomorphic encryption	multiplicative homomorphic voting
		preferential system case study
	mix-networks	a scheme using batch techniques
		a scheme using EBMG
		applications of batching
		preferential system case study
	receipt-freeness (requirement)	receipt-free and efficient mix-network based scheme
	hybrid	combine new schemes in this research
		preferential system case study

Voting involves processing of individual ballots. Since all individual requirements are similar, it is possible to perform a single processing accommodating all the ballots. This is called batching. Generically, individual operations with a specific similarity can be combined into a single batch. Batch theorems have been presented, and new techniques using the theorems have been developed. The use of these techniques in appropriate schemes increases the efficiency and practicality of such schemes. These theorems and techniques have been presented in Chapter 3.

Homomorphic encryption is the basis of one framework for cryptographic voting protocols. It maintains the encryption of a vote inside a ballot. Individual votes are kept secret, while decryption of a ballot combination reveals the voting result. Two main contributions have been made in this area. First, an alternative scheme has been presented. It exploits a multiplicative property of homomorphism instead of the commonly used additive one. Efficiency of the new scheme is comparable to existing schemes. Second, a preferential voting case study concluded that this framework provides efficiency in vote tallying. However, it is only suited for voting using a structured vote. These results have been presented in Chapter 4.

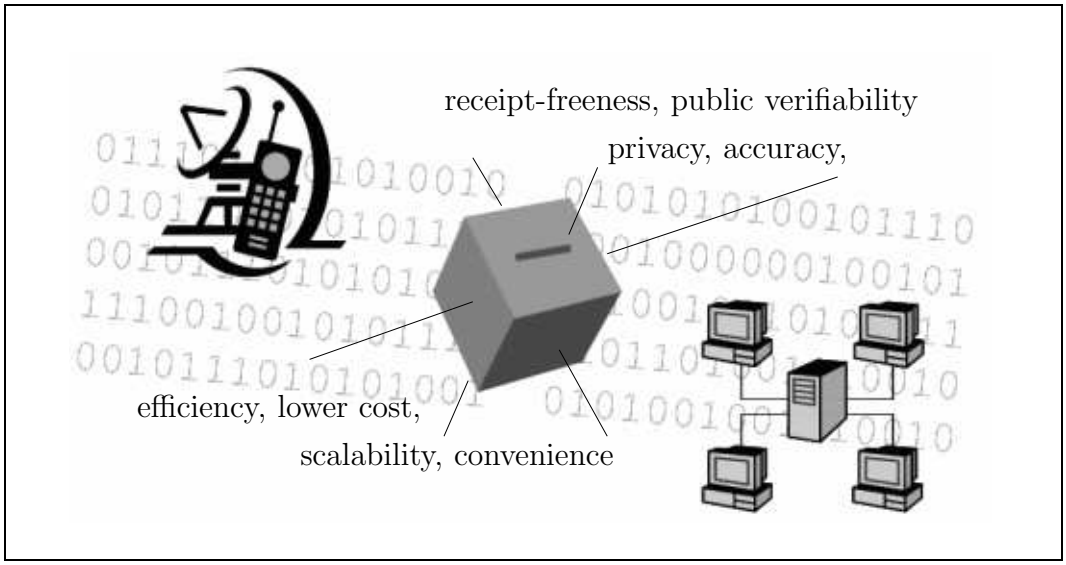

Figure 8.1: Secure electronic voting with a flexible ballot structure.

Mix-network is the basis for another framework for cryptographic voting protocols. Similar to the use of ballot-boxes, ballots are anonymised inside a mix-network. The resulting output is a set of anonymised plaintext votes. It is essential that the output votes corresponds to votes inside the input ballots. Two new primitives have been presented in achieving this requirement. One offers efficiency by extending on the use of grouping and batch (EBMG), and the other offers to maintain security while improving efficiency of existing schemes by using batch techniques. A preferential voting case study concluded that this framework allows voting using write-in (unstructured/free-form) votes. However, vote tallying is not as efficient as the homomorphic encryption one. These results have been presented in Chapter 5.

Receipt-freeness is an important requirement for secret-ballot voting. It prevents vote selling/buying or coercion/intimidation. Emphasising on this requirement, an efficient mix-network based scheme have been presented. The scheme employs a particular re-encryption technique to achieve this property. The re-encryption is combined with an optimistic mix-network for efficiency. A new threshold re-encryption technique, and a batch designated verifier re-encryption proof technique have also been presented. These techniques allow the distribution of trust from a single trusted re-encryption authority to a (threshold) number of re-encryption authorities. The scheme and techniques have been presented in Chapter 6.

The hybrid approach combines both the homomorphic encryption and mix-network approach. Both the multiplicative homomorphic voting scheme and the EBMG mix-network scheme have been combined into a hybrid approach. A preferential voting case study concluded that this approach is suitable for preferential voting allowing a small pre-set of preferences using the homomorphic encryption, and other available preferences using the mix-network. This hybrid approach has been presented in Chapter 7.

Figure 8.1 offers a high-level abstract description of our research. The resulting contributions offer a promising foundation for a secure and practical secret-ballot electronic voting accommodating any type of counting systems. The next section list a number of possible future work directions for this area.

8.2 Possible Research Directions

This section offers a number of possible research directions continuing from the work presented in this thesis.

Optimum efficiency for more practical schemes: Batch theorems presented in Chapter 3 can be developed into many other batch techniques. Applications of such techniques in appropriate cryptographic schemes would improve efficiency, performance, and practicality of such schemes. This seems to be a straight-forward extension of this research.

Specifically, techniques developed from the batch theorems in this research have been on the use of "AND logic". It should be possible to devise a technique for the use of "OR logic". This offers a more efficient construction of zero-knowledge proof of correct homomorphic ballot. Furthermore, it might be possible to combine verification of valid homomorphic ballot and correct homomorphic tally decryption.

Using a combination of batch techniques, it might also be possible to merge all the required proof verifications in a re-encryption mix-network scheme. Zero-knowledge proof verifications of knowledge of plaintext, re-encryption shuffle, and (threshold) decryption may be combined into a single verification. This allows for a much more efficient verifiable mix-network.

More research into homomorphic encryption voting schemes: Our research has been concentrating on the novel use of the mix-network approach. This is due to the focus on allowing a flexible ballot structure. Outside of this

focus, we believe there are avenues for improvements or alternative techniques available for better use of homomorphic encryption in cryptographic voting protocols.

A possible extension would be to distribute the ballots during casting instead of distributing the power of ballot decryption (threshold). The scheme of Schoenmaker [Sch00] may be extended/improved.

Alternative mix-network schemes: Our research has mainly been concentrated on mix-network based voting protocols. However, we believe that more work is possible in this area. Our contributions have been in the use of a shuffling proof to formally show a correct mixing operation. Further research is possible on the use of input-based verifications.

Most re-encryption chain based mix-network schemes employ decryption at the end. A combination with threshold decryption in the shuffling may be possible by extending the scheme of Abe [Abe99]. This promises improved efficiency. Further work is also possible in the area of decryption chain based mix-network.

Use of other cryptosystems: ElGamal and Paillier are the two popular cryptosystems used as the foundation of protocols in this research. It might be possible to use other cryptosystems optimised for voting purposes. The use of an identity-based cryptosystem in voting protocols deserves attention. It is an alternative use of public-key based cryptosystems. It may eliminate the need of a Public Key Infrastructure, which will be beneficial to a voting protocol considering the large number of voters and their corresponding certificates verification overhead.

Develop a prototype system: We acknowledge that further study and testing are required before deployment of a live secure secret-ballot electronic voting system is possible. However, a prototype system can be built as a proof of concept that outcomes of this research are practical and usable. Such a prototype will also show that it is possible to further develop the system to be used in a real-world.

Among other issues, scalability need to be considered and tested thoroughly. This is because there are many factors affecting the running of an actual nation-wide election with a large number of voters.

Investigations into preventative measures for network security attacks: Aside from authentication and verification using public key cryptosystems and zero-knowledge protocols respectively, other cryptographic primitives may be

integrated into a voting protocol to help minimise network security attacks. Distributed Denial of Service (DDoS) is an example of a network security attack. It only requires a low level of skill to perform (using available tools on the Internet), but may cause serious consequences (denies voters their right to vote, affects a voting result, etc).

The use of cryptographic measures such as client puzzles [JB99] or cryptographic salt [PKBD01] can be included in the protocol to minimise the impact of this attack. Otherwise, servers must simply provide a high bandwidth connections to accommodate the attack bandwidth.

Other network security attacks must also be considered. Each respective possible protective measures must be implemented.

Research outside the cryptographic voting protocol area: Actual deployment of a secure electronic voting system in the real-world clearly requires collaboration from other areas of discipline as well. From selecting a suitable counting system to creating a new law on electronic voting, inter-disciplinary research collaboration is required to produce a mature, secure, and publicly acceptable electronic voting system.

In conclusion, electronic voting offer many potential benefits compared to its traditional counterpart. A study on cryptographic voting protocols allowing a flexible ballot structure has been presented in this thesis. Novel primitives and new schemes have been developed. Further extensions and improvements on our research are possible.

We acknowledge that further work is required in other areas such as policies, procedures, and standards before a secure electronic voting system is to actually be used in a real world environment. However, we believe that our research offers a promising start towards the goal of having such a system.

Appendix A

Shamir's Secret-Sharing Scheme

This appendix recalls the well-known "Shamir's secret sharing scheme" [Sha79], also known as "(t, m) threshold scheme" where $t < m$. Such a scheme is useful for robustness, or distributing trust from a single entity to a threshold/quorum of entities.

A secret d is divided into m shares d_j with a threshold of t using a polynomial function $f(x)$ of degree t. The secret d can only be reconstructed using at least $t+1$ shares of d_j using Lagrange interpolation. The secret d can not be reconstructed using t (the threshold value) or fewer shares.

The (t, m) secret sharing scheme is as follows:

Secret sharing: Let the polynomial function be $f(x) = \sum_{r=0}^{t} a_r x^r$. The value of a_0 is set as the secret d, and the remaining values for a_r are set at random. For $j \in \{1, 2, \ldots, m\}$, the secret share d_j is computed as $d_j = f(j)$.

Secret shares combining: The secret d is reconstructed using Lagrange interpolation by using the set S containing at least $t + 1$ of the secret shares d_j as $d = \prod_{j \in S} d_j \mu_j$, where $\mu_j = \prod_{j' \in S \setminus j} \frac{j'}{j' - j}$.

An example application of the scheme is in the sharing of a secret decryption key to m participants, where each participant is given one share d_j of the secret d. The plaintext contained in a known ciphertext can only be revealed using the secret decryption key d. If an attacker successfully compromises t or fewer participants (their shares d_j are known to the attacker), their compromised shares can not be used to reconstruct the decryption key d and the plaintext contained

in the ciphertext remains secret to the attacker. On the other hand, the message can be revealed by at least $t + 1$ uncompromised/honest participants. This is by reconstructing the secret decryption key d using Lagrange interpolation with their shares d_j on the polynomial $f(x)$, and decrypting the ciphertext with the secret decryption key. This illustrates robustness and trust distribution among m number of participants.

Appendix B

ElGamal and Paillier Cryptosystems

ElGamal and Paillier are two popularly used public-key cryptosystems in cryptographic voting protocols in the literature. Both cryptosystems are used as foundations to the cryptographic voting protocols in the thesis.

Both cryptosystems are recalled in this appendix. We also recall the application of Shamir's secret-sharing technique (Appendix A) in the threshold version of the cryptosystems. Both versions allow decryption by a quorum of decryption authorities instead of decryption by a single decryption authority. This offers robustness and distribution of trust to the authorities.

B.1 ElGamal Cryptosystem

The security of ElGamal cryptosystem [ElG85] is based on the hard problem of finding discrete logarithms over finite fields. Exponentiations are easy, but computing logarithms is not easy. Currently there is no known algorithm to efficiently solve such a problem.

B.1.1 Basic Cryptosystem

The ElGamal cryptosystem used in the thesis is as below.

1. Key generation:

Randomly select a large prime q, such that $p = 2q + 1$ is a strong prime. G is a cyclic subgroup in $\mathbb{Z}_p^*$ of order q with a generator g. The private decryption key is $x \in \mathbb{Z}_q$, while g and $y = g^x \bmod p$ is the public encryption key. The parameters p, q, g, and y are made public, while x is kept secret.

2. Encryption:
   Select a random $r \in \mathbb{Z}_q$ and encrypt a secret message (plaintext) $s \in \mathbb{Z}_p^*$ as $c = (\alpha, \beta)$, where $\alpha = g^r \bmod p$ and $\beta = sy^r \bmod p$.

3. Decryption:
   The original message s is reconstructed by using the decryption key x and computing $s = \frac{\beta}{\alpha^x} \bmod p$.

Two different ciphertexts can correspond to the same plaintext using a different value of r. This allows the ElGamal cryptosystem to possess the ciphertext indistinguishability property, or equivalently to satisfy semantic security [GM84].

This cryptosystem also allows **re-encryption** of ciphertexts. Without knowing the secret message, a ciphertext can be re-encrypted by updating the random value. Let a new re-encryption random value be r'. The re-encrypted ciphertext c' is constructed as $c' = (\alpha', \beta') = (\alpha g^{r'} \bmod p, \beta y^{r'} \bmod p)$. The original random value r is now updated (after re-encryption) to be $r + r'$. Decryption of the ciphertext will yield the original plaintext. Specifically, $D(c) = D(c')$ where D denotes an ElGamal decryption function for the corresponding ciphertext.

Without revealing the plaintext, one can produce a *zero-knowledge proof of ciphertext construction*. This is by proving the knowledge of the random value r used in α (Appendix C.1) as $\log_g \alpha$.

For a *correct re-encryption proof*, one proves the knowledge of the new random value r' used to produce both α' and β' from α and β using Chaum-Pedersen [CP93] zero-knowledge proof of equality of discrete logarithms (Appendix C.3) as $\log_g \frac{\alpha'}{\alpha} = \log_y \frac{\beta'}{\beta}$.

B.1.2 Threshold Version

Pedersen [Ped92] presented a threshold ElGamal signature scheme based on Shamir's (t, m) secret-sharing scheme (Appendix A). It is straight-forward to adjust the scheme into a threshold decryption protocol. The threshold decryption version is obtained by sharing the private decryption key d to m number of

decryption authorities. For $j \in \{1, 2, \ldots, m\}$, each of the m authorities has a share x_j of the private decryption key x. To decrypt a ciphertext, each authority computes a partial decryption using his/her share. A quorum of the partial decryptions are then combined to reconstruct the secret message s.

The protocol is as below.

1. Key generation and sharing:
   Randomly select a large prime q, such that $p = 2q + 1$ is a strong prime. G is a cyclic subgroup in $\mathbb{Z}_p^*$ of order q with a generator g. The private decryption key is $x \in \mathbb{Z}_q$, while g and $y = g^x \bmod p$ is the public encryption key. Using Shamir's (t, m) secret sharing scheme, let $f(x) = \sum_{r=0}^{t} a_r x^r$, where $a_0 = x$, and the rest of a_r are random values. For $j \in \{1, 2, \ldots, m\}$, distribute the secret share $x_j = f(j)$ to m decryption authorities, and each authority computes the verification key $v_j = g^{x_j}$. The parameters p, q, g, y, and v_j are made public, while x and x_j are kept secret.

2. Encryption:
   Select a random $r \in \mathbb{Z}_q$ and encrypt a secret message $s \in \mathbb{Z}_p^*$ as $c = (\alpha, \beta)$, where $\alpha = g^r \bmod p$ and $\beta = sy^r \bmod p$.

3. Partial decryption:
   Each of at least $t + 1$ authorities compute correct partial decryptions $z_j = \alpha^{x_j}$, and proves the knowledge of the secret share x_j using a zero-knowledge proof of equality of discrete logarithms (Appendix C.3) of: $\log_g(v_j) = \log_\alpha(z_j)$. Since q is public, g and α can be publicly verified to be generators of G.

4. Decryption:
   Correctness of each partial decryption z_j is verified using the zero-knowledge protocol shown in the previous step. S is the set containing at least $t + 1$ correct partial decryptions. The original plaintext is reconstructed by computing $s = \frac{\beta}{\prod_{j \in S} z_j^{\mu_j}}$, where $\mu_j = \prod_{j' \in S \setminus j} \frac{j'}{j' - j}$.

B.2 Paillier Cryptosystem

The Paillier cryptosystem [Pai99] is based on the hard problem of determining composite degree residuosity. Currently there is no known algorithm to efficiently solve such a problem.

B.2.1 Basic Cryptosystem

There are three closely related cryptosystems presented in the original paper by Paillier [Pai99]. We recall the first one, as it is the most well-known of the three. The Paillier cryptosystem used in this thesis is as below.

1. Key generation:
   Randomly select primes p' and q', such that both $p = 2p'+1$ and $q = 2q'+1$ are strong primes, and $GCD(N, \phi(N)) = 1$, where $N = pq$ and $M = p'q'$. Select x and e, such that $x = 0 \bmod M$ and $x = e^{-1} \bmod N$. Choose a random value of $b \in \mathbb{Z}_N^*$ and set the value of g to be $g = (1+N)^x b^N \bmod N^2$. The parameters N and g are made public, while M, p, q, p', q', x, e and $\lambda(N)$ are kept secret, where $\lambda(N) = \text{lcm}((p-1)(q-1))$.

2. Encryption:
   Select a random value $r \in \mathbb{Z}_N^*$, and encrypt a secret message (plaintext) s as $c = g^s r^N \bmod N^2$.

3. Decryption:
   The original message s is reconstructed as $s = \frac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$, where input of the L-function is an element from the set $\{u < N^2 | u = 1 \bmod N\}$, and $L(u) = \frac{u-1}{N}$.

Two different ciphertexts can correspond to the same plaintext using a different value of r. This is known as the ciphertext indistinguishability property, or having semantic security [GM84].

This cryptosystem also allows **re-encryption** of ciphertexts. Without knowing the secret message, a ciphertext can be re-encrypted by updating the random value. Let a new re-encryption random value be r'. The re-encrypted ciphertext c' is constructed as $c' = cr'^N \bmod N^2$. The original random value r is now updated (after re-encryption) to be rr'. Decryption of the ciphertext will yield the original plaintext.

Without revealing the plaintext, one can produce a *zero-knowledge proof of ciphertext construction* by proving the knowledge of the secret message s in the ciphertext c (Appendix C.2) similar to using zero-knowledge proof of knowledge of root.

For a *correct re-encryption proof*, one proves the knowledge of the new random value r' by using zero-knowledge proof of knowledge of root (Appendix C.2) as $(\frac{c_i'}{c_i})^{\frac{1}{N}}$.

B.2.2 Threshold Version

Based on the threshold version of RSA signature by Shoup [Sho00], Fouque *et al.* [FPS00] proposed a threshold version of Paillier cryptosystem in the context of voting or lotteries. The scheme was later improved by Damgård and Jurik [DJ00] oriented toward a homomorphic electronic voting scheme. We recall the protocol as follows:

1. Key generation:
   Randomly select primes p' and q', such that both $p = 2p' + 1$ and $q = 2q' + 1$ are strong primes, and $GCD(N, \phi(N)) = 1$, where $N = pq$ and $M = p'q'$. Select x and e, such that $x = 0 \bmod M$ and $x = e^{-1} \bmod N$. Choose a random value of $b \in \mathbb{Z}_N^*$ and set the value of g to be $g = (1 + N)^e b^N \bmod N^2$. Using Shamir's (t, m) secret sharing scheme, let $f(x) = \sum_{r=0}^{t} a_r x^r \bmod MN$, where $a_0 = x$ and random values for the rest of $a_r \in \{1, 2, \ldots, N * M - 1\}$. For $j \in \{1, 2, \ldots, m\}$, distribute the secret share $x_j = f(j)$ to m decryption authorities. Select a random value of v, a square that generates the cyclic group of squares in $\mathbb{Z}_{N^2}^*$, to be the verification base. Each decryption authority then computes their corresponding verification key $v_j = v^{\Delta x_j} \bmod N^2$ and $\Delta = m!$. The parameters N, g, v and v_j are made public, while M, p, q, p', q', x, e and x_j are kept secret, where $j \in \{1, 2, \ldots, m\}$.

2. Encryption:
   Select a random value $r \in \mathbb{Z}_N^*$, and encrypt a secret message (plaintext) s as $c = g^s r^N \bmod N^2$.

3. Partial decryption:
   Each of at least $t + 1$ authorities compute correct partial decryptions $z_j = c^{2\Delta x_j}$ and proves the knowledge of the secret share x_j using a zero-knowledge proof of equality of discrete logarithms (Appendix C.2) of: $\log_v(v_j) = \log_{c^4}(z_j^2)$.

4. Decryption:
   Correctness of each partial decryption z_j is verified using the zero-knowledge protocol shown in the previous step. S is the set containing at least $t + 1$ correct partial decryptions. The original plaintext s is reconstructed by

computing $s = \frac{L(c' \bmod N^2)}{4\Delta^2 \bmod N^2)} \bmod N$, where $c' = \prod_{j \in S} z_j^{2\mu_{0,j}^S} \bmod N^2$, and $\mu_{0,j}^S = \Delta \prod_{j' \in S \setminus j} \frac{-j}{j-j'} \in \mathbb{Z}$.

As in the original scheme by Damgård and Jurik, v is chosen to be a generator of the group in $\mathbb{Z}_{N^2}^*$ by the trusted dealer. Therefore, v and v_j are squares in the group of $\mathbb{Z}_{N^2}^*$.

Appendix C

Zero-Knowledge Proof and Verification Protocols

Zero-knowledge (ZK) protocols allows a *prover* to demonstrate knowledge of a secret to a *verifier* while revealing nothing extra of use to the verifier. Verification of a ZK proof only convinces the verifier that the prover actually does (or does not) know the secret.

More generally, the protocol allows proving and verifying the truth of an assertion while revealing no extra information about the assertion itself other than its actual truth.

Three important property of the protocol are recalled as below.

- **completeness**: given an honest prover and an honest verifier, the protocol succeeds with an overwhelming probability.
- **soundness**: a dishonest prover can successfully execute protocol runs only by either knowing the secret, or by having the ability of extracting the secret in polynomial time.
- **zero-knowledge**: a polynomial-time simulator can produce an indistinguishable transcript compared to a transcript from a real protocol run.

A standard three-move ZK protocol, known as Σ-protocol [CD95], has a special soundness property as proven in [CD95]. A general structure of such a protocol is as follows:

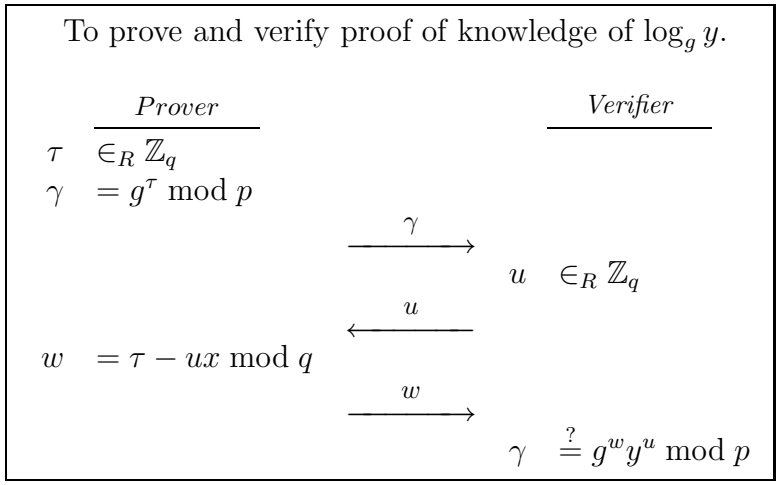

Figure C.1: An interactive ZK proof-verification protocol for verifying knowledge of a discrete logarithm.

1. *Prover* $\rightarrow$ *Verifier*: witness (commitment)
2. *Prover* $\leftarrow$ *Verifier*: challenge
3. *Prover* $\rightarrow$ *Verifier*: response

In an *interactive* version of the protocol, the verifier selects the challenge at random. In a *non-interactive* version of the protocol, a well-known Fiat-Shamir heuristic technique [FS86] is typically used. The technique uses a collision-resistant hash function for a prover to generate the challenge.

This appendix recalls a number of standard ZK protocols.

C.1 Knowledge of A Discrete Logarithm

Based on the scheme by Schnorr [Sch91], Figure C.1 illustrates a standard interactive ZK proof of knowledge of a discrete logarithm in an ElGamal cryptosystem setting, where $y = g^x$.

This proof can be made non-interactive by using the well-known Fiat-Shamir's heuristic by employing a collision-resistant hash function H with a range of $\mathbb{Z}_q$. The challenge is generated by the prover as $u = H(g, y, \gamma)$. Given the transcript values of $\{\gamma, u, w\}$, a verifier can check that $u \stackrel{?}{=} H(g, y, (g^w y^u \bmod p))$.

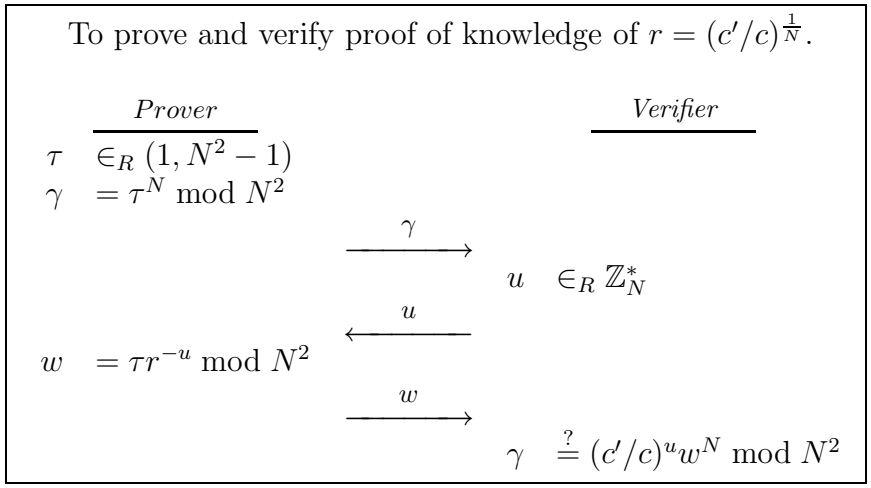

Figure C.2: An interactive ZK proof-verification protocol for verifying knowledge of a root.

C.2 Knowledge of a Root

Based on the scheme by Guillou-Quisquater [GQ88], Figure C.2 illustrates a standard interactive ZK proof of knowledge of a root in a Paillier cryptosystem setting, where $c' = cr^N \bmod N^2$.

This proof can be made non-interactive by using the well-known Fiat-Shamir's heuristic by employing a collision-resistant hash function H with a range of $\mathbb{Z}_N$. The challenge is generated by the prover as $u = H(c', c, \gamma)$. Given the transcript values of $\{\gamma, u, w\}$, a verifier can check that $u \stackrel{?}{=} H(c', c, ((c'/c)^{\frac{1}{N}} w^N \bmod N^2))$.

C.3 Equality of Discrete Logarithms

Based on the scheme by Chaum-Pedersen [CP93], Figure C.3 illustrates a standard interactive ZK proof of equality of discrete logarithms in an ElGamal cryptosystem setting, where $y = g^x$ and $z = c^x$.

This proof can be made non-interactive by using the well-known Fiat-Shamir's heuristic by employing a collision-resistant hash function H with a range of $\mathbb{Z}_q$. The challenge is generated by the prover as $u = H(g, y, c, z, \gamma_1, \gamma_2)$. Given the transcript values of $\{\gamma_1, \gamma_2, u, w\}$, a verifier can check that the challenge $u \stackrel{?}{=} H(g, y, c, z, (g^w y^u \bmod p), (c^w z^u \bmod p))$.

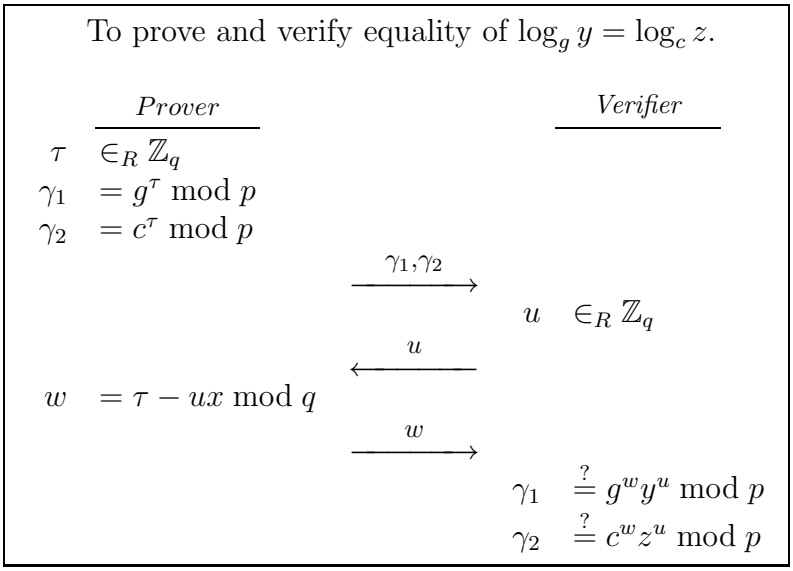

Figure C.3: An interactive ZK proof-verification protocol for verifying equality of discrete logarithms.

C.4 Proof Construction

This section offers two examples of combining similar ZK proofs by using AND logic, and OR logic respectively.

The first is by using the same challenge to produce the different responses from the corresponding witnesses.

The second is by combining the challenges to produce the different responses from the corresponding witnesses.

C.4.1 AND Logic

Based on the scheme by Chaum-Pedersen [CP93], Figure C.4 illustrates a standard interactive ZK proof of equality of n discrete logarithms in an ElGamal cryptosystem setting, where $\alpha_i = g^{x_i}$ and $\beta_i = y^{x_i}$.

Note that in this proof construction, each of the equations has its own witness and response, but all of them share the same challenge.

This proof can be made non-interactive by using the well-known Fiat-Shamir's heuristic by employing a collision-resistant hash function H with a range of $\mathbb{Z}_q$. The challenge is generated by the prover as $u = H(g, y, \{\alpha_i, \beta_i, \gamma_{i,1}, \gamma_{i,2}\})$. Given the transcript values of $\{\gamma_{i,1}, \gamma_{i,2}, u, w_i\}$, a verifier can check that $u \stackrel{?}{=} H(g, y, \{\alpha_i, \beta_i, (g^{w_i}\alpha_i^u \bmod p), (y^{w_i}\beta_i^u \bmod p)\})$.

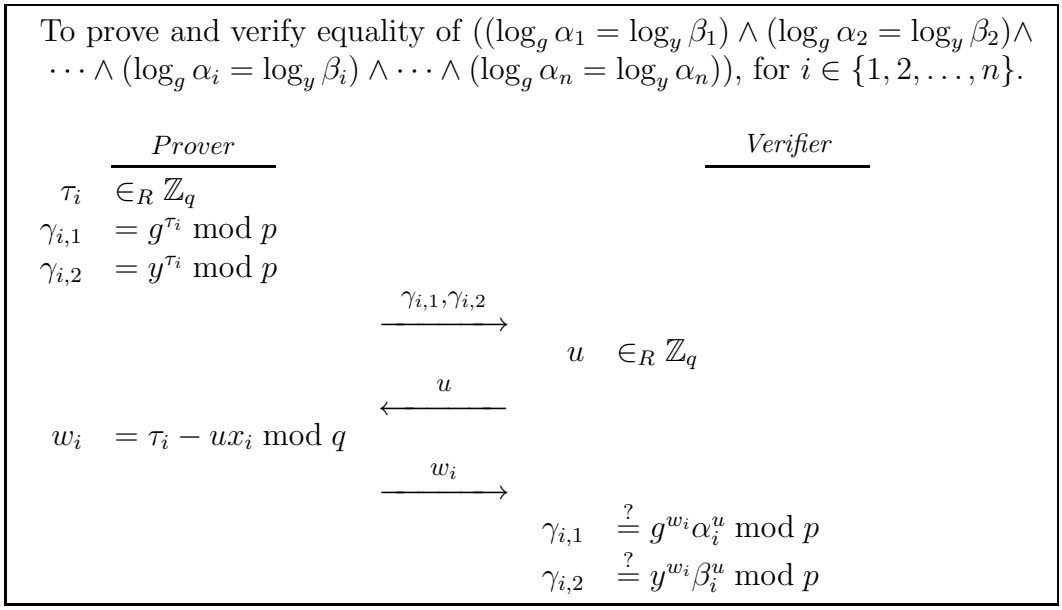

Figure C.4: An interactive ZK proof-verification protocol for verifying n equality of discrete logarithms.

C.4.2 OR Logic

Based on the scheme by Chaum-Pedersen [CP93], Figure C.4 illustrates a standard interactive ZK proof of 1-out-of-n equality of discrete logarithms in an ElGamal cryptosystem setting, where $\alpha_i = g^{x_i}$ and $\beta_i = y^{x_i}$. The prover only knows one of the discrete logarithms, namely where $i \in \{1, 2, \ldots, n\}$, $i' \in \{1, 2, \ldots, n\} \backslash k$ and $1 \leq k \leq n$.

Note that in this proof construction, each of the equations has its own witness and response, and the combination of challenges is specified by the verifier.

This proof can be made non-interactive by using the well-known Fiat-Shamir's heuristic by employing a collision-resistant hash function H with a range of $\mathbb{Z}_q$. The challenge is generated by the prover as $u = H(g, y, \{\alpha_i, \beta_i, \gamma_{i,1}, \gamma_{i,2}\})$. Given the transcript values of $\{\gamma_{i,1}, \gamma_{i,2}, u, u_i, w_i\}$, a verifier can check that the combined challenge $u \stackrel{?}{=} H(g, y, \{\alpha_i, \beta_i, (g^{w_i} \alpha_i^{u_i} \bmod p), (y^{w_i} \beta_i^{u_i} \bmod p)\})$.

To prove and verify equality of $((\log_g \alpha_1 = \log_y \beta_1) \vee (\log_g \alpha_2 = \log_y \beta_2) \vee \cdots \vee (\log_g \alpha_i = \log_y \beta_i) \vee \cdots \vee (\log_g \alpha_n = \log_y \alpha_n))$, for $i \in \{1, 2, \ldots, n\}$.

Prover			*Verifier*	
$1 \leq k \leq n$,	and			
i'	$\in \{1, 2, \ldots, n\} \setminus k$			
$\tau_k, \{\tau_{i'}, u_{i'}, w_{i'}\}$	$\in_R \mathbb{Z}_q$			
$\gamma_{k,1}$	$= g^{\tau_k} \bmod p$			
$\gamma_{k,2}$	$= y^{\tau_k} \bmod p$			
$\gamma_{i',1}$	$= g^{w_{i'}} \alpha_{i'}^{u_{i'}} \bmod p$			
$\gamma_{i',2}$	$= y^{w_{i'}} \beta_{i'}^{u_{i'}} \bmod p$			
		$\xrightarrow{\gamma_{i,1}, \gamma_{i,2}}$		
			u	$\in_R \mathbb{Z}_q$
		$\xleftarrow{u}$		
u_k	$= u - \sum u_{i'} \bmod q$			
w_k	$= \tau_k - u_k x_k \bmod q$			
		$\xrightarrow{w_i}$		
			$\gamma_{i,1}$	$\stackrel{?}{=} g^{w_i} \alpha_i^{u_i} \bmod p$
			$\gamma_{i,2}$	$\stackrel{?}{=} y^{w_i} \beta_i^{u_i} \bmod p$

Figure C.5: An interactive ZK proof-verification protocol for verifying 1-out-of-n equality of discrete logarithms.

Bibliography

[ABDV03] Riza Aditya, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Secure e-voting for preferential elections. In *Electronic Government: Second International Conference, EGOV 2003*, volume 2739 of *Lecture Notes in Computer Science*, pages 246–249. Springer-Verlag, 2003.

[Abe99] Masayuki Abe. Mix-networks on permutations networks. In *Advances in Cryptology – ASIACRYPT '99: 5th International Conference on the Theory and Application of Cryptology and Information Security*, volume 1716 of *Lecture Notes in Computer Science*, pages 258–273. Springer-Verlag, 1999.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 317–324. Springer-Verlag, 2001.

[AI03] Masayuki Abe and Hideki Imai. Flaws in some robust optimistic mix-nets. In *Information Security and Privacy : 8th Australasian Conference, ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 39–50. Springer-Verlag, 2003.

[ALBD04a] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In *Trust and Privacy in Digital Business: First International Conference, TrustBus 2004*, volume 3184 of *Lecture Notes in Computer Science*, pages 152–161. Springer-Verlag, 2004.

[ALBD04b] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. Implementation issues in secure e-voting schemes. In *The Fifth Asia-Pacific Industrial Engineering and Management Systems Conference, APIEMS 2004*, pages 36.6.1–36.6.14. QUT Publications, 2004.

[ALBD05] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. Two models of efficient mixnet-based receipt-free voting using (threshold) re-encryption. *Computer Systems Science and Engineering*, page to appear, 2005.

[APB+04] Riza Aditya, Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Batch verification for equality of discrete logarithms and threshold decryptions. In *Applied Cryptography and Network Security: Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 494–508. Springer-Verlag, 2004.

[Ben96] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Faculty of Graduate School, Yale University, 1996.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 01*, pages 274–283. ACM Press, 2001.

[BG02] Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *9th ACM Conference on Computer and Communications Security, CCS '02*, pages 68–77. ACM Press, 2002.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology – EUROCRYPT '98: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, 1998.

[BP00] Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In *Advances in Cryptology – ASIACRYPT 2000:*

6th International Conference on the Theory and Application of Cryptology and Information Security, volume 1976 of *Lecture Notes in Computer Science*, pages 58–71. Springer-Verlag, 2000.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC '94*, pages 544–553. ACM Press, 1994.

[CD95] Ronald Cramer and Ivan Damgård. Secure signature schemes based on interactive protocols. In *Advances in Cryptology – CRYPTO '95: 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science*, pages 297–310. Springer-Verlag, 1995.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO '94: 14th Annual International Cryptology Conference*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology – EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83. Springer-Verlag, 1996.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January/February 2004.

[CP93] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO '92: 12th Annual International Cryptology Conference*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.

[DJ00] Ivan Damgård and Mads Jurik. Efficient protocols based on probabilistic encryption using composite degree residue classes. Cryptology ePrint Archive, Report 2000/008, 2000. `http://eprint.iacr.org/`.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual IEEE Symposium on Foundations of Computer Science*, pages 427–437. IEEE Computer Society, 1987.

[Fia89] Amos Fiat. Batch RSA. In *Advances in Cryptology – CRYPTO '89: Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 175–185. Springer-Verlag, 1989.

[FPS00] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography: 4th International Conference, FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer-Verlag, 2000.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86: Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1986.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology – CRYPTO 2001: 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer-Verlag, 2001.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology – EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310. Springer-Verlag, 1999.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences (JCSS)*, 28(2):270–299, April 1984.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology – EUROCRYPT '88: Workshop on the Theory and Application of Cryptographic Techniques*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 1988.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography – PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160. Springer-Verlag, 2003.

[GZB+02] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2501 of *Lecture Notes in Computer Science*, pages 451–465. Springer-Verlag, 2002.

[HAK01] Fumitaka Hoshino, Masayuki Abe, and Tetsutaro Kobayashi. Lenient/strict batch verification in several groups. In *Information Security: 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 81–94. Springer-Verlag, 2001.

[Har98] Lein Harn. Batch verifying multiple DSA-type digital signatures. *IEEE Electronic Letters*, 34(9):870–871, April 1998.

[Har03] Beverly Harris, editor. *Black Box Voting: Ballot Tampering in the 21st Century*. Plan Nine Publishing, 2003.

[HK04] Alejandro Hevia and Marcos Kiwi. Electronic jury voting protocols. *Theoretical Computer Science*, 321(1):73–94, June 2004.

[JB99] Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Network and*

Distributed System Security Symposium, NDSS '99, pages 151–165, 1999.

[JJ01] Markus Jakobsson and Ari Juels. An optimally robust hybrid mix network. In *Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 01*, pages 284–292. ACM Press, 2001.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium*, pages 339–353, 2002.

[JRSW04] David Jefferson, Aviel Rubin, Barbara Simons, and David Wagner. A security analysis of the secure electronic registration and voting experiment (SERVE). Online, 2004. Available from `http://www.servesecurityreport.org`, last accessed 26 May 2005.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology – EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.

[KSRW04] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *2004 IEEE Symposium on Security and Privacy*, pages 27–40. IEEE Computer Society, 2004.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 141–158. Springer-Verlag, 2002.

[KY04] Aggelos Kiayias and Moti Yung. The vector-ballot e-voting approach. In *Financial Cryptography: 8th International Conference, FC 2004*, volume 3110 of *Lecture Notes in Computer Science*, pages 72–89. Springer-Verlag, 2004.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Information Security and Cryptology – ICISC 2003: 6th International Conference*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer-Verlag, 2004.

[LK00] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *JW-ISC 2000*, pages 101–108, 2000.

[LK02] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Information Security and Cryptology – ICISC 2002: 5th International Conference*, volume 2587 of *Lecture Notes in Computer Science*, pages 389–406. Springer-Verlag, 2002.

[Mao03] Wenbo Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference, 2003.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security, CCS '01*, pages 116–125. ACM Press, 2001.

[OA00] Miyako Ohkubo and Masayuki Abe. A length-invariant hybrid mix. In *Advances in Cryptology – ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security*, volume 1976 of *Lecture Notes in Computer Science*, pages 178–191. Springer-Verlag, 2000.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols: 5th International Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer-Verlag, 1997.

[OKST97] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In *Information and Communication Security: First International Conference, ICICS '97*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444. Springer-Verlag, 1997.

[PAB+04a] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *Progress in Cryptology – INDOCRYPT 2004: 5th International Conference on Cryptology in India*, volume 3348, pages 61–72. Springer-Verlag, 2004.

[PAB+04b] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. A secure and efficient mix-network using extended binary mixing gate. In *Cryptographic Algorithms and their Uses – 2004: International Workshop*, pages 57–71. QUT Publications, 2004.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[PBDV04] Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. A correct, private, and efficient mix network. In *Public Key Cryptography – PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 439–454. Springer-Verlag, 2004.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology – EUROCRYPT '91: Workshop on the Theory and Application of Cryptographic Techniques*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer-Verlag, 1991.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '89: Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology – EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *Lecture Notes in Computer Science*, pages 248–259. Springer-Verlag, 1993.

[PKBD01] DongGook Park, JungJoon Kim, Colin Boyd, and Ed Dawson. Cryptographic salt: A countermeasure against denial-of-service attacks. In *Information Security and Privacy : 6th Australasian Conference, ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 334–343. Springer-Verlag, 2001.

[PMPS00] Jaroslaw Pastuszak, Dariusz Michatek, Josef Pieprzyk, and Jennifer Seberry. Identification of bad signatures in batches. In *Public Key Cryptography: Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 28–45. Springer-Verlag, 2000.

[PS99] Guillaume Poupard and Jacques Stern. On the fly signatures based on factoring. In *6th ACM Conference on Computer and Communications Security, CCS '99*, pages 37–45. ACM Press, 1999.

[RR97] Andrew Reynolds and Ben Reilly, editors. *The International IDEA Handbook of Electoral System Design.* International Institute for Democracy and Electoral Assistance, 2 edition, 1997.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, May 1991.

[Sch00] Berry Schoenmakers. Fully auditable electronic secret-ballot elections. *XOOTIC Magazine*, July 2000.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sho00] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology – EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer-Verlag, 2000.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *Advances in Cryptology – EUROCRYPT '95: International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.

[VBD00] Kapali Viswanathan, Colin Boyd, and Ed Dawson. A three phased schema for sealed bid auction system design. In *Information Security and Privacy : 5th Australasian Conference, ACISP 2000*, volume 1841 of *Lecture Notes in Computer Science*, pages 412–426. Springer-Verlag, 2000.

[Wik02] Douglas Wikström. How to break, fix and optimize "optimistic mix for exit-polls". Technical report, Swedish Institute of Computer Science, 2002. Available from `http://www.sics.se/libindex.htlm`, last accessed 08 October 2003.