

Secure And Verifiable Election Systems

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Gavin Weld White

May 2002

Approved for the Division
(Mathematics)

James D. Fix

Acknowledgments

I appreciate the support and encouragement of Karin Edwards, without whose enduring kindness and patience this work would not have been possible. I also appreciate the tremendous help I have had in writing this thesis, and in particular the guidance given to me by professors Jim Fix, Joe Buhler, David Perkinson, and Albyn Jones. The final editing by my good friend David Clark was invaluable. My family, to whom this work is cryptically dedicated, deserve my thanks as well. I appreciate and acknowledge the help I have received from other people too numerous to mention. Last, but certainly not least, is Betsy Ladd, whose friendliness helped me to stay at Reed in my Freshman year, and without whose friendship I might not have returned after my leave of absence — thank you.

Preface

This thesis was inspired by a chain of events transpiring late in the year 2000. In early November, there was a widely known crisis involving the U.S. Presidential elections. Due to irregularities in voting, vote counting, ballot construction, and numerous other factors, the selection of the President of the United States of America fell to the Supreme Court.

In late November, while all this was happening, the cryptography class at Reed College was covering secure electronic voting schemes. I knew my thesis topic when I found myself pondering possible extensions of material covered in class — at all hours of the day and in the middle of traffic.

Also around that time, I read an article published in the Oregonian about the theses of some recent Reed College graduates; how helpful, relevant, and applied they were; and how this in some way typified Reed theses [3]. I remember wondering, at the time, how the thesis topic I had in mind counted as relevant, helpful, or applied. My conclusion at this time is that I still do not know — I can only hope that this thesis contributes to the creation of a better world.

The protocol presented here could become the basis for the voting systems used at Reed College, and, eventually, on a global scale. While I recognize that there are major differences between these environments, and many steps along that path, I have designed these protocols with flexibility and extensibility in mind.

Table of Contents

- Acknowledgments **i**
- Preface **iii**
- List of Figures **vii**
- Dedication **xi**
- 1 Introduction 1**
 - 1.1 Election 2000 1
 - 1.2 Strategic Voting and IIA 2
 - 1.2.1 Borda Count 2
 - 1.2.2 Instant Runoff 3
 - 1.3 This Thesis 4
- 2 Privacy versus Verifiability 7**
 - 2.1 Voting 7
 - 2.2 Privacy 8
 - 2.2.1 Homomorphic Encryption 9
 - 2.2.2 A Homomorphic Encryption Scheme 9
 - 2.3 Verifiability 10
- 3 Cryptographic Election Scheme 13**
 - 3.1 Overview 13
 - 3.2 Another Homomorphic Encryption Scheme 13
 - 3.3 Extending the Cryptosystem to a Pairwise Vote 14
 - 3.3.1 Pairwise Encryption 14
 - 3.3.2 Voting 15
 - 3.4 Verification 15
 - 3.4.1 Proof: Valid Cryptosystem 15
 - 3.4.2 Proof: Valid Ballot 16
 - 3.5 Tallying 18
- 4 Extensions 21**
 - 4.1 Poly-Candidate Elections 21
 - 4.1.1 Plurality Voting 21

4.1.2	Borda Count	22
4.2	Instant Runoff	22
4.3	Receipt-Free Voting:	
	A Backwards Ballot Transfer and a Public Vote	24
4.3.1	Receipt-Free Instant Runoff Elections	25
5	Extending the Scheme for Use with Multiple Authorities	27
5.1	Blinding Factors	27
5.2	Voting Revisited	28
5.2.1	Secret Bits — Randomness from the Authorities	28
5.2.2	Ballot Validity	29
5.2.3	Multi-Authority Tallying	29
6	Discussion	31
6.1	Voter Authentication	31
6.2	Security	32
6.3	Implementation	32
6.3.1	No Write-Ins	33
6.3.2	Ease of Use	33
6.4	Conclusion	34
	Bibliography	35

List of Figures

2.1	IP that p knows x to be a square.	11
2.2	IP that p knows a square root of x modulo n	12
3.1	IP of Valid Cryptosystem.	16
3.2	ZKP that the submitted ballot is of a valid form	18
5.1	IP of the value of the blinding factor sent by A	28

Abstract

This thesis examines the possibility of implementing a secure and verifiable election scheme using homomorphic encryption, based partly on the work of Benaloh and Tuinstra. After a brief overview of voting and the tradeoffs between privacy and verifiability, this thesis explores some cryptographic enhancements that would address these problems. The problems of strategic voting and voting paradoxes are also briefly looked at, and the instant runoff Borda Count is suggested as a remedy to these. The result is the description of a robust system that is implementable at Reed and scalable for use elsewhere in the world.

Dedication

To H, K, M, N, and P, with love, from G.

Chapter 1

Introduction

1.1 Election 2000

On November 7, 2000, citizens in the United States engaged in an historic election. While this election will be remembered for years to come as having been problematic, the problems that were displayed have always been present in the election system that was used.

The problems observed in our current election system are of two types: strategic voting and verifiability. In defense of the voting system that is used in the United States, it can be said that it addresses another problem that is a major concern, that of privacy. This thesis addresses all three of these concerns.

In discussions of the election of 2000, people often unknowingly raise the specter of strategic voting. What seems to have happened is that people who would otherwise have supported Gore voted for Nader. There may also have been a number of people who, supporting Nader, still voted for Gore because they feared, reasonably, that Bush would win if they voted for Nader. That is, some voters voted based on the predicted election outcome, rather than voting their preferences. This is called *strategic voting*, and was probably done because of a justified belief by these voters that if they did otherwise, the election result would more likely run contrary to their actual preferences.

The other problem that is frequently raised with the 2000 election is the counting of the ballots. This, at its heart, is a problem of *verifiability*: in the election system used, there is no way for anyone, including the voters, to know that only valid ballots were submitted; there is no way to know that the ballots submitted were the ones that were counted; and there is no way to check that the tally announced by the voting authorities is the sum of all the valid votes.

In Florida these problems were particularly apparent. There have been allegations of stolen or misdirected ballots, and many votes were invalidated due to poor ballot design. Furthermore, the Supreme Court ordered that recounts be stopped, which meant that some of the votes that were valid may not have been accurately counted in some parts of Florida.

Voters in Florida submitting ballots with hanging, dimpled, or pregnant chads could have double-checked their own ballots, but even so there is no way for them

to prove to anyone else that they submitted a valid vote.

The election system used by the United States does, however, address issues of privacy. Because voters submit their votes on ballots that are not associated with them in particular, voters have no receipt of their votes. Furthermore, no voter can prove to anyone else how they voted, and thus votes cannot reliably be bought or coerced. Obviously, this concern about privacy, or security, is important to any unbiased election.

1.2 Strategic Voting and IIA

In considering secure elections, I first wanted to create a system that could replace the one currently in use by the Reed student body. For a while I talked with people about what would make a good voting system, and finally I looked for some good books on the subject. What I found was amazing.

According to Arrow's Theorem, as described in [15], there are no voting systems that are free of paradoxes. The main kind of paradox that shows up is referred to as an Independence of Irrelevant Alternatives (IIA) paradox, in which the introduction of a new candidate to an election changes the rankings of the other candidates relative to each other.

One such paradox probably occurred in 2000, when the election outcome was Bush > Gore > Nader. Exit polls and other data suggest that, had Nader been less well liked, Gore would have beaten Bush. Although it is possible to use strategic voting to counteract this kind of paradox, the election of 2000 shows that people are not reliable enough for strategic voting to work in this way on a large scale. We need some other approach to reduce the occurrence of IIA paradoxes.

Thus, it seems wise to seek a voting system that would eliminate the troubling IIA paradoxes and reduce the difference between strategic and honest voting. Ideally, this could be implemented in an election system that would be easy to use and provide both privacy and verifiability.

1.2.1 Borda Count

Fortunately, some voting systems are more equal than others. As an example, we will consider the following scenario, paraphrased from [15], using three different voting systems. Three brothers, N, G, and H, have been nominated for a prestigious award. The awards committee, made up of five people, has had long, elaborate discussions about the merits of the candidates. As they leave their final meeting, three of the committee members prefer $N > G > H$, while the other two prefer $H > N > G$.

Shortly before voting, the committee learns that G attended a small liberal arts college favored by the first group of committee members. This is enough, in each of their minds, to make G a better candidate for the award than N. As such, those three committee members change their preference ranking to $G > N > H$. Now the votes are cast.

We first assume that the votes are tallied as they are in U.S. Presidential elections, using the plurality method. This method gives each candidate one point for every voter that ranked them first. When the votes are tallied in this way, a strange thing happens. Note that the relative ranking of $N > H$ remains unchanged among the group of three committee members — the ranking of G should be irrelevant to the relative rankings of N and H . Nevertheless, the election result is now $G > H > N$.

To further complicate matters, consider the quandary at the awards ceremony when candidate G , in a fit of brotherly love, declines the award — is it passed to his brother H , or to his other brother, N ? The announced result would have the award passed to H , but if G had not been a candidate the award would have gone to N .

Another frequently used tallying method is the anti-plurality vote. This is the method that is being suggested when someone says, “let’s just eliminate a candidate.” Formally, we give to each candidate one point for every voter that did not rank them last. Here the rankings are the same before and after the three committee members changed their votes — $N > H = G$. Notably, there no difference between the two sets of votes.

Now consider the same situation, but instead use the following method to tally the votes.

Definition 1.1 (Borda Count). *A Borda Count election is one in which each candidate is given one point for every candidate above whom they are ranked by each voter.*

Here we see that, although the point spread changes, the relative rankings of N and H remain unchanged. Before the committee members changed their minds, N had 8 points, H had 4, and G had 3. After the switch, G has 6 points, N has 5, and H has 4.

According to several theorems by Saari, the Borda Count stands alone as the least paradoxical voting system[15].

1.2.2 Instant Runoff

Regardless of the voting system used, so long as there is only one round of voting and tallying, it is easy to apply strategic voting principles to unduly influence the outcome of an election while being dishonest about one’s own actual preferences. That is, a well informed, intelligent voter in a non-runoff system will not necessarily indicate their true preferences with their vote.

Ideally, the voting system would be designed such that the best strategy coincides with the simplest one. This avoids ethical dilemmas around favoring those that lie about their preferences.

For example, consider a close three-way race. Voter V ’s actual preference ranking is $H > N > G$; V wants H to win, and failing that, wants G to lose. V has been paying attention to the rankings, and, while the election is going to be close, it seems clear that H will take about 30% of the vote, while N and G will split the other 70%. Thus V votes for $N > H > G$, hoping to prevent a win for G . While this

seems natural and obvious to many people, it means that the election results have an even lower chance of reflecting the actual wishes of the voters.

As is discussed in [15], strategic voting works best in close elections, and only when the strategic voters are a very small minority. When there are too many strategic voters, the power of the strategic vote is diluted and the outcomes of strategic votes are much harder to predict.

As an alternative, consider an election using an Instant Runoff.

Definition 1.2 (Instant Runoff). *An Instant Runoff election is one in which the loser of the first tally is removed from the preference rankings of all the voters, and the tally is recomputed based on the new preference rankings.*

In an Instant Runoff election, V can freely vote $H > N > G$, knowing that the first tally will reflect V 's preference for H , and that if or when H is eliminated from the election, V will be counted as ranking N over G . This can be extended such that, for m candidates, there are $m - 2$ eliminations and $m - 1$ tallies.

Instant runoff addresses a subset of the IIA paradoxes that is introduced by the voters themselves, rather than the voting system. V prefers H to N , unless G is involved, in which case V prefers N to H . An instant runoff solves this paradox by requiring a retallying of votes, which may reorder the candidates, each time a candidate is eliminated. This should make voters more comfortable with expressing their true preferences. While the elimination could take place elsewhere than the bottom of the list (loser of the most recent tally), it is not clear how such an alternative elimination would be chosen. Thus, the canonical elimination is the loser of each successive tally.

In an instant runoff election, there can still be strategy, but the strategy is different. A strategic voter will now rank lowest the candidates that they want to be eliminated first. While this is not necessarily the same as the voter's least favored candidates, it coincides with the candidates the voter thinks will produce the worst societal outcomes and yet have the best chance at winning.

1.3 This Thesis

I propose the design of a secure election system using the Borda Count and instant runoff methods.

Ideally, such a system would have the following properties:

Authorized Voters. Only those who are authorized can vote.

Authorized Votes. No voter can vote more than once.

Receipt-Free. No one can prove how they voted.

Secure (against duplication). No one can know that they have cast the same vote as another person without knowing how the other person voted.

Secure (against tampering). No one can change anyone else's vote without being discovered.

Verifiable. Anyone can verify that every vote is included in the tally.

Multiple Authorities. No single authority has the ability to reveal any vote or set of votes.

The first two properties are difficult to provide, though electronic systems using the Kerberos protocol may provide a usable solution. Generally, though, user authentication remains an open problem, and is pursued by many organizations. This issue is largely set aside in these election systems, allowing voter authentication outside of Kerberos systems to continue as it has, using voting booths. In this regard, this system is at least as secure as current voting methods.

Receipt-freeness was first explored by Benaloh and Tuinstra in [1], though their multiple-authority scheme was shown insecure in [7], which also proposed an alternative scheme. The receipt-free scheme proposed in [1] for single authorities also provides verifiability, as well as security against both tampering and duplication.

In this thesis, I will provide an overview of several election schemes proposed by Benaloh and Tuinstra, fix the multiple-authority scheme proposed by them, and extend their work to include the Borda Count and Instant Runoff. Finally, I will consider the design of the resultant election system with regard to all of the above properties, ease of use, and extensibility.

Chapter 2

Privacy versus Verifiability

2.1 Voting

We say an election consists of voters, candidates, an authority, ballots, votes, and a tally.

Voters — The set, V , whose members are eligible to influence the election outcome (e.g.: citizens, board members, shareholders). Say $|V| = s$.

Candidates — An ordered set, C , of election outcomes, the choices from which the voters choose. (e.g.: {Bush, Gore, Nader}, {no, yes}). Say $|C| = m$.

Authority — The set, A , of entities responsible for running the election, generating the tally, and, often, implementing the election outcome. A may be a singleton set, depending on the election system. Say $|A| = \ell$.

Ballots — The set, B , of media for the expression of a vote, whose members are not in and of themselves indicative of the voters' preferences.

Votes — The set, v , of indications of voter preferences. (e.g.: [N > G > B], [yes > no]).

Tally — For each candidate, the sum of the points awarded to them by the voting system based on the votes of the voters.

Let us consider a simple election, held in a public place, with candidates G and N, voters a, b, and c, and authority g, using the hands of the voters as ballots. Let a and c vote [N > G], and let b vote [G > N]. Then it is trivial to count the votes and determine that there are more votes for N than for G. Furthermore, anyone can verify that each voter voted only once, that only valid voters voted, that all the votes were of a valid form, and that the tally reported by g is the correct one. What is missing is privacy.

As a solution to the problem of privacy, let us consider the case in which the ballots are three pieces of paper that are stuffed into a ballot box and submitted to g before revealing what is on them. In this case, g can release the ballots to the

public to be counted, and anyone can verify the outcome reported by g . Suppose g pulls the ballots out of the box, and releases the following votes: two voters voted [$G > N$], while a third voter voted [$N > G$]. Now G wins. The problem here is that there is no guarantee that the ballots revealed by g are the same as those submitted by a , b , and c . While we have gained privacy, we have lost verifiability.

Although elections can be held such that they are verifiable, or such that they are private, there is no way to have both in a traditional election scheme. In order to solve this problem of privacy versus verifiability, we will introduce encryption and zero knowledge proofs.

2.2 Privacy

Encryption, in general, is the process of hiding some information such that it is difficult for unauthorized entities to obtain. Usually, this difficulty does not rely on the obscurity of the hiding process, but on the secrecy of some key that allows decryption. Decryption is the inverse process — revealing encrypted secrets.

We say the encryption function E with secret key x takes messages m to ciphertext c , as $E_x(m) = c$, and the decryption function D takes ciphertext to messages, as $D_x(c) = m$. In contexts where x is not relevant, it may be omitted. It is notable that some encryption and decryption systems, or cryptosystems, do not require x for decryption, but rely instead on a different secret.

The simplest and most secure form of encryption is the one-time pad. In this method, the sender and receiver must each have a copy of the secret key, which must be at least as long as the message to be sent. The bits of the secret key are XORed¹ with the message bits, and then sent to the receiver. The receiver XORs the bits of the secret key with the bits of the ciphertext, and reads the message.

The security of this method relies on the fact that, given a particular message and any secret key, any ciphertext can be generated. That is, given a particular ciphertext, for all messages of equal or lesser length there is a one-time pad that would generate the ciphertext. This is called information-theoretic security, because it is impossible to gain any information about the message from the ciphertext without having information about the key.

While this method is absolutely secure from anyone not possessing the secret key, it is usually infeasible. The main problem is key distribution, because anyone having the key can read the message. Furthermore, the receiver must be sure that the key they have corresponds exactly to the one with which the message was encoded. For these reasons, one-time pads are usually used only in situations where a secure key-exchange can be made in anticipation of the need for later secure communication over long distances.

Instead, most cryptosystems rely on what is called computational security. Computational security is based on what are known as standard hardness assumptions,

¹For $a \in \{0, 1\}$, $a \text{ XOR } 1 = 1 - a$, $a \text{ XOR } 0 = a$, and thus $a \text{ XOR } b = b \text{ XOR } a$ and $a \text{ XOR } b \text{ XOR } b = a$.

the belief that, given current resources, certain mathematical operations are difficult to compute in a reasonable amount of time. What this means is that the best known algorithms for solving these problems take an amount of time that increases roughly exponentially with the size of the numbers involved.

The most common argument for computationally secure cryptosystems is that gaining any information about a message encrypted with the given cryptosystem is equivalent to solving a hard problem. For example, the security of an encryption scheme we examine later relies on the difficulty of factoring large numbers. Since this is believed to be hard to do, it is believed to be hard to decrypt such messages. The form of the argument is thus, “If cows dance, then pigs fly, therefore cows do not dance.” Although pigs may fly some day, and cows may also dance, they have not been known to do so yet, and significant technological advances will have to be made before they do so with ease!

Such computationally secure encryption schemes are often referred to as one-way trapdoor functions. Here, informally, one-way means that the function is hard to invert and easy to compute. It is not known whether such functions exist, though there are functions that seem to fit this description. Trap-door means that, for someone who knows the secret key, the function is easy to invert, taking polynomial time or less.

One frequently used model for cryptosystems is known as public-key encryption. In this model, one entity, say A, instantiates the cryptosystem, publishing a public key to be used by others. Anyone having this public key can then encrypt messages for A using their own secret keys. A can decrypt these messages because of some extra knowledge about the public key, but based on some hardness assumption, it is believed that no-one else can gain this knowledge of the public key.

2.2.1 Homomorphic Encryption

For the purposes of this election system, we will need an encryption and decryption scheme where some easy operation on the encrypted secrets is homomorphic to tallying the votes. That is, we need a homomorphic encryption scheme.

We will take votes to be integers, tallying to be addition, and, for convenience, call the homomorphic operation on the encrypted votes multiplication. The homomorphism we need is described by the following properties:

$$E_{x_1}(e_1) \cdot E_{x_2}(e_2) = E_{x_1 x_2}(e_1 + e_2) \quad (2.1)$$

$$E_{x^{-1}}(-e) = E_x(e)^{-1} \quad (2.2)$$

We will use these properties to reliably distinguish between encryptions of all possible valid tallies. While we could use any secure homomorphic encryption scheme with these properties, we introduce here one such scheme.

2.2.2 A Homomorphic Encryption Scheme

As an example of a homomorphic encryption scheme, consider the following, designed by Goldwasser and Micali[10]. We will use this scheme to prove that two

parties have agreed to a yes/no decision without revealing the decision.

The Cryptosystem

This cryptosystem will rely on the difficulty of determining quadratic residuosity modulo n when the factorization of n is unknown. We say that a number y is a quadratic residue modulo n if, for some $a \in \mathbb{Z}_n^*$, $a^2 \equiv y \pmod{n}$. Otherwise, y is a quadratic non-residue. We say that the set of quadratic residues modulo n is the set QR_n , and the set of quadratic non-residues is QNR_n .

To instantiate the cryptosystem, choose two large primes, p and q , let $n = pq$, and choose $\gamma_1 \in \text{QNR}_p$ and $\gamma_2 \in \text{QNR}_q$. By the Chinese Remainder Theorem, there is a unique $y \in \mathbb{Z}_n^*$ such that $y \equiv \gamma_1 \pmod{p}$ and $y \equiv \gamma_2 \pmod{q}$. Note that y is in QNR_n . The public key is the pair (n, y) .

The encryption and decryption functions are

$$E_x(b) = y^b x^2 \pmod{n}, D(z) = \begin{cases} 1 & \text{if } z \in \text{QNR}_n \\ 0 & \text{if } z \in \text{QR}_n \end{cases}$$

The decryption function relies on knowledge of $n = pq$ and an algorithm for determining quadratic residuosity, described in [10].

This scheme is homomorphic in the required way because

$$\begin{aligned} E_{x_1}(b_1) \cdot E_{x_2}(b_2) &= y^{b_1} x_1^2 \cdot y^{b_2} x_2^2 \pmod{n} \\ &\equiv y^{b_1+b_2} (x_1 x_2)^2 \pmod{n} \\ &= E_{x_1 x_2}(b_1 + b_2) \end{aligned}$$

Although there is no known proof, it is believed that the determining quadratic residuosity is equivalent to factoring integers[10].

2.3 Verifiability

One of the concerns with our current voting systems is that there is no guarantee that only valid votes are counted, or that all the valid votes are counted. While traditional election systems do not allow for both privacy and verifiability, we present here the notion of a zero knowledge proof, which will be used in the design of a system with both privacy and verifiability.

In order to define a zero knowledge proof, it is useful first to define, informally, an interactive proof.

Definition 2.1 (Interactive Proof). *An Interactive Proof (IP) of w is a finite set of messages between a verifier, v , and a prover, p , at the end of which v accepts w as true.*

As an example of an IP, consider the following trivial example.

Interactive Proof 2.1. *p knows that x is a square number.*

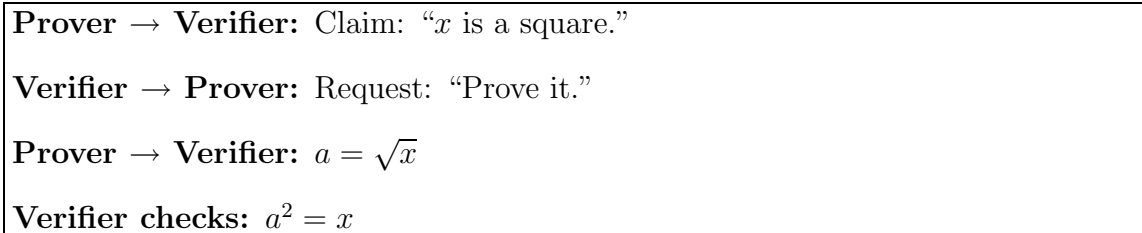


Figure 2.1: The steps of IP 2.1

Figure 2.1 illustrates the IP described below.

The prover claims to know that a number, x , is a square. The verifier responds with a request for proof of this knowledge, and the prover sends to the verifier $a = \sqrt{x}$. The verifier checks that $a^2 = x$. If this holds, then v believes that p knows that x is a square. While this example is simplistic, it contains all the essential elements of an interactive proof.

There are cases, however, where the prover will want to prove knowledge of something without revealing anything but the existence of that knowledge. In such situations, it is more appropriate to use a zero knowledge proof.

Definition 2.2 (Zero Knowledge Proof). *An interactive proof of w is a Zero Knowledge Proof (ZKP) of w if all interpretations of the messages are true with equal probability and the probability that the prover could get away with cheating in any single round is less than or equal to $\frac{1}{2}$.*

Typically, a ZKP of w can be repeated N times, making the probability that v accepts a false proof less than or equal to $\frac{1}{2^N}$ while maintaining zero knowledge.

Consider the following example of a zero knowledge proof (ZKP), found in [17].

Zero Knowledge Proof 2.2. *P knows a square root of x modulo n , where n is composed of two large primes.*

Figure 2.2 shows the steps in one round of the ZKP described below.

Let n be composed of two or more large primes, and let $x = a^2 \pmod n$. The prover claims to know a square root of x , but is unwilling to reveal it. After telling the verifier of this knowledge, the prover chooses a random number, $\omega \in_{\mathbb{R}} \mathbb{Z}_n^*$, and sends $\rho = \omega^2 \pmod n$ to the verifier. The verifier replies with $b \in_{\mathbb{R}} \{0, 1\}$. The prover checks that $b \in \{0, 1\}$, and returns $z = a^b \omega \pmod n$. Finally, the verifier checks that $z^2 = x^b \rho$.

Without knowing a square root of x , a cheating prover has at best a 50% chance of success in any given round. This is because the prover sends to the verifier another square, ρ , which will either be revealed or combined with a . If $b = 0$, the prover reveals $\sqrt{\rho}$, and if $b = 1$ the prover reveals $\sqrt{\rho x}$. Thus, half the time the prover will only have to prove that ρ is a square, but the other half of the time the prover will have to show that if ρ is a square then so is x . Since the decision to reveal ρ is made by the verifier, the prover cannot successfully cheat. So long as both checks are passed N times without any failures, the verifier can be assured that,

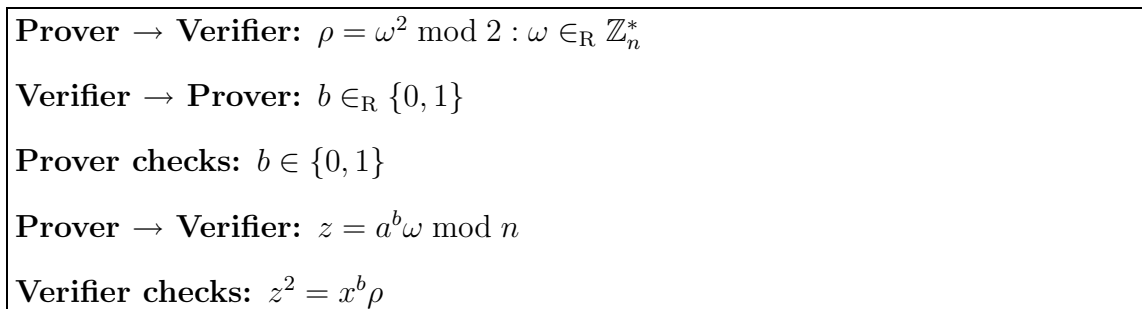


Figure 2.2: This figure shows the steps in one round of ZKP 2.2, in which the Prover displays knowledge of a square root of $x = a^2 \bmod n$

with probability $1 - \frac{1}{2^N}$, the prover does in fact know a square root of x modulo n . Thus, this is a ZKP that the prover knows a square root of x modulo n . \square

Chapter 3

Cryptographic Election Scheme

3.1 Overview

In this chapter, we will look at another homomorphic encryption scheme fulfilling the properties described in Section 2.2.1. This scheme will be extended for use with a pairwise vote, and tallying will be described. We will save poly-candidate, receipt-free, and multi-authority elections for later, along with the design of an instant runoff system.

In this election scheme, the authority publishes the public key for a homomorphic cryptosystem, anyone can check that the cryptosystem is valid, the voters submit encrypted votes as their ballots — meanwhile proving that they are valid — and the authority publishes a tally of the votes.

First, we will need an encryption scheme.

3.2 Another Homomorphic Encryption Scheme

We say a number y is an r^{th} -residue modulo n if, for some $a \in \mathbb{Z}_n^*$, $a^r \equiv y \pmod{n}$. Notably, the r^{th} -residuosity problem is a generalization of the quadratic residuosity problem — while there is no known proof of its equivalence to factorization, factorization is sufficient and, it is assumed, necessary for determining r^{th} -residuosity.

In general, we can define a scheme that relies, at worst, on the r^{th} -residuosity hardness assumption. For this scheme, take a prime number r greater than the largest message to be encrypted — here, the number of voters times the number of candidates, so $r > sm$. Recall that A is the set of authorities. For now, take $A \in A$ to be the sole member of that set, and, when $|A| > 1$, understand the following to refer to the actions of each element of A .

A chooses prime numbers p and q . In order for encryption and decryption to take place, p and q need to be such that r divides the order of \mathbb{Z}_p^* , but does not divide the order of \mathbb{Z}_q^* . Because, for all prime p , $|\mathbb{Z}_p^*| = p - 1$, we need:

$$r \mid p - 1, \quad r \nmid \frac{p - 1}{r}, \quad r \nmid q - 1.$$

Let $n = pq$. This makes 1 in r integers non- r^{th} -residues modulo n .

A chooses a number y in the multiplicative group of integers modulo n such that y is not an r^{th} -residue, modulo n . A publishes n, r , and y , keeping p and q secret.

Now, to encrypt any number $e < r$, we randomly choose x from the multiplicative group of n . This x is the *private key*. The encryption and decryption functions are as follows:

$$E_x(e) = y^e x^r \pmod{n}, D(z) = \log_y(z) \pmod{n}$$

Decryption involves taking the discrete log, with base y , modulo n . This requires x, p , or q , and, given one of these, takes linear time in $e \pmod{r}$.

Finding x is equivalent to solving the discrete logarithm problem in \mathbb{Z}_n^* , while finding p or q requires prime factorization of n . Since finding the discrete logarithm modulo n is equivalent to factoring n , it is sufficient to consider factoring. The best known algorithm for factoring numbers composed of two or more large primes takes $O(e^{\sqrt{\ln n \ln \ln n}})$ time, which, while it is sub-exponential, is still considered computationally infeasible for sufficiently large n .

If it is known that $e \in \{0, 1\}$, finding e is equivalent to the r^{th} -residue problem, which is also sub-exponential, but, like factoring, still computationally infeasible. Therefore, this cryptosystem is computationally secure.

As was suggested earlier, if any of these open problems are solved this encryption scheme can be replaced with a similarly homomorphic scheme.

For A, and anyone else knowing x, p , or q , decryption is as follows. By Euler's extension of Fermat's Little Theorem, since $r \mid p - 1$,

$$\begin{aligned} z^{\frac{p-1}{r}} &\equiv y^{e(\frac{p-1}{r})} x^{p-1} \pmod{n} \\ &\equiv y^{e(\frac{p-1}{r})} \pmod{p} \end{aligned}$$

An entity with q can easily find $p = \frac{n}{q}$. Entities with p can easily find $\frac{p-1}{r}$, and thus $y^{\frac{p-1}{r}}$. By multiplying this number by itself, modulo p , as many as r times, such entities can find $e \pmod{r}$.

An entity having x can, instead, find $x^r \pmod{n}$, and multiply this by y modulo n , as many as r times to find $e \pmod{n}$.

Since r is not significantly more than the number of voters times the number of candidates, neither of these is much harder than counting votes normally.

3.3 Extending the Cryptosystem to a Pairwise Vote

In order to encrypt a vote, it will be helpful to extend the notion of encryption for individual values to encryption of pairs.

3.3.1 Pairwise Encryption

Let E be an encryption function as described in Section 2.2.1. Define

$$E_{(x_1, x_2)}(a, b) = (E_{x_1}(a), E_{x_2}(b)).$$

The decryption function is now

$$D_{(x_1, x_2)}(\alpha, \beta) = (D_{x_1}(\alpha), D_{x_2}(\beta)).$$

As mentioned in Section 2.2, we may write these functions without reference to the secret keys when the context does not require them.

This encryption function can be used for a simple pairwise voting system.

3.3.2 Voting

In a full election between two candidates, each voter submits a ballot. In a simple pairwise voting system, a valid vote is of the form (e, e') , where $e \in \{0, 1\}$ and $e' = e - 1$. We say that e and e' are vote-elements. To form a ballot, we associate two candidates, say G and N, with an encrypted vote, $E(e, e')$.

The canonical ordering of candidates is alphabetical. Thus the vote $E(1, 0)$ indicates a vote of 1 for G and 0 for N. The vote $E(0, 1)$ would indicate the reverse: 0 for G and 1 for N.

3.4 Verification

It remains to be shown that all the votes are valid, that the cryptosystem used by A is valid, and that the tally published by A is valid. This is equivalent to requiring proofs that all ballots are of the proper form, that A can distinguish between $E_x(1)$ and $E_x(0)$, and that the product of all the ballots is an encryption of the published tally.

While the following proofs can be used with any cryptosystem having the properties outlined in Section 2.2.1, they will be described in terms of the scheme introduced in Section 3.2.

In this encryption scheme, distinguishing between $E_x(0)$ and $E_x(1)$ is equivalent to distinguishing between r^{th} -residues and non- r^{th} -residues modulo n .

3.4.1 Proof: Valid Cryptosystem

It is easy to check that r is correct, that $r \nmid n - 1$, and, probabilistically, that n is not prime. Because A can decrypt ballots, it is unclear why a malicious A would make a bad n such that votes were still countable, rather than just giving p and q to any accomplices. Thus, since we must trust A not to reveal p and q , we will also trust A not to reveal them by choosing cryptographically weak p or q . Therefore, it remains for A to prove to all verifiers that there exist non- r^{th} -residues mod n , and in particular that y is one such non- r^{th} -residue mod n .

To check that A can distinguish between $E_x(1)$ and $E_x(0)$ with probability $1 - \frac{1}{2^N}$, a verifier can engage in the following IP before voting begins. This IP can be used for any encryption scheme meeting the requirements described in Section 2.2.1. In this IP, anyone can be the verifier, and the prover is the authority, A . Anyone wishing to verify the published cryptosystem must do so before voting begins because, though

<p>Verifier chooses: $x \in_{\mathbb{R}} \mathbb{Z}_n^*, e \in_{\mathbb{R}} \{0, 1\}$</p> <p>Verifier \rightarrow Prover: $w_v = E_x(e)$</p> <p>Prover \rightarrow Verifier: $w' = E_x(1 - e)$</p> <p>Verifier checks: $w' = E_x(1 - e)$</p>
--

Figure 3.1: This figure shows the steps in one round of IP 3.1, in which the Prover displays the ability to determine r^{th} -residuosity modulo n .

it reveals no additional information about the factorization of n , it could be used to reveal a person's vote.

Interactive Proof 3.1. *A can count votes.*

Figure 3.1 illustrates the IP described below.

For each round of this protocol, the verifier chooses $x \in_{\mathbb{R}} \mathbb{Z}_n^*$, and generates $w_v = E_x(e), w'_v = E_x(e')$, where $e \in_{\mathbb{R}} \{0, 1\}$. The verifier then sends w_v to A . A , receiving this challenge, decrypts it and returns $w' = E_x(e')$, where x and e' are the same as were used by the verifier. Upon receiving this w' , the verifier checks that $w' = w'_v$. If not, A 's scheme is invalid. Otherwise, the verifier considers A to have passed this round.

For the encryption scheme described in Section 3.2, if y is an r^{th} -residue, then $y \equiv a^r \pmod{n}$. Thus $E_x(1) = E_{xy}(0)$, so A must guess which was meant by the verifier. A must send back $w' \in \{w_v y, \frac{w_v}{y}\}$ with no better than a 50% chance of guessing correctly.

For an observer of the messages between A and the verifier, there is nothing to see but a number going one way, and that number being returned either multiplied by y or divided by y , seemingly at random.

This IP is repeated N times, each with different x, e , and e' . Each repetition has an approximately 50% chance of exposing any faulty scheme. As noted before, the verifier can conclude that, with probability $1 - \frac{1}{2^N}$, the scheme published by A is correct.

Note, however, that a crafty verifier could use the publisher of this scheme, via this protocol, as an oracle for determining r^{th} -residuosity modulo n .

Therefore, by Definitions 2.1 and 2.2, this is an IP, but not a ZKP, that y is a non- r^{th} -residue modulo n . \square

3.4.2 Proof: Valid Ballot

The proof that a ballot, and thus the vote expressed through it, is valid will involve generating extra ballots, and randomly determining whether to reveal them or to compare them with the original ballot. When the prover reveals one of the extra ballots, the verifier can check that it is valid. When one of the extra ballots is compared with the original, the verifier can check that if one is valid then both are valid.

In order to explore the proof that a ballot is valid, it will be helpful to look at some frequently used notation.

First, we will examine the operation \times . In general, π is a permutation — in this example, it is a permutation on two elements, but we will assume the natural extension to m elements.

$$(a, b) \times \pi = \begin{cases} (a, b) & \text{if } \pi = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ (b, a) & \text{if } \pi = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{cases} \quad (3.1)$$

Now we define the operation \otimes_π , for which, in the absence of π , we assume the identity.

$$(a, b) \otimes (c, d) = (ac, bd) \quad (3.2)$$

$$(a, b) \otimes_\pi (c, d) = ((a, b) \times \pi) \otimes (c, d) \quad (3.3)$$

The multiplicative inverse of a ballot is as required by the homomorphism properties described in Section 2.2.1, and is only defined on encrypted ballots, as follows:

$$(a, b)^{-1} = (a^{-1}, b^{-1}) \quad (3.4)$$

Zero Knowledge Proof 3.2. *The ballot submitted is of a valid form.*

Figure 3.2 illustrates the ZKP described below. Note that all N rounds are condensed into one through the use of a public beacon.

To prove that a ballot is valid with probability $1 - \frac{1}{2^N}$, the publisher of the ballot, here called p , will generate N extra ballots. Half of these ballots will be chosen at random to be revealed, and the other half will be related to the real ballot so as to prove that if one is valid then both are valid. If all the revealed ballots are valid, since the ballots to be revealed were chosen at random, there is a $1 - \frac{1}{2^N}$ probability that the real ballot is also valid.

The entity generating the ballot, p , publishes an additional N ballots, $B_1 \dots B_N$, along with the main ballot B_0 . We assume for the moment a public bit beacon, a generator of random bits that supplies the bits on demand. The entity proving the validity of its ballots requests N bits, $b_1 \dots b_N$.

In each case where $b_i = 0$, p publishes the private keys, $(x_i, x'_i) = \eta_i$, used to encrypt the ballot $B_i = E_{\eta_i}(e, e')$. Using these private keys, anyone can decrypt all such B_i and verify that they are encryptions of valid votes.

In each case where $b_i = 1$, p publishes a permutation with which to reorder B_0 for comparison with B_i , π_i such that

$$D(B_0) \times \pi_i = D(B_i),$$

and the “zero certificate”,

$$\eta_0 \otimes_{\pi_i} \eta_i^{-1} = \xi_i.$$

<p>Prover → public: $B_0 \dots B_N$, such that $D(B_0) \times \pi_i = D(B_i)$, where π_i is a permutation.</p>
<p>Beacon → public: $b_1 \dots b_N$</p>
<p>Prover → public: $\forall i \in 1 \dots N : \begin{cases} \eta_i & \text{if } b_i = 0 \\ \xi_i = \eta_0 \otimes \eta_i^{-1}, B'_i = B_0 & \text{if } b_i = 1 \otimes_{\pi_i} B_i^{-1} \end{cases}$</p>
<p>Verifier checks: $\forall i \in 1 \dots N : \begin{cases} B_i = E_{\eta_i}(e, 1 - e) \text{ with } e \in \{0, 1\} & \text{if } b_i = 0 \\ B'_i = E_{\xi_i}(0, 0) & \text{if } b_i = 1 \end{cases}$</p>

Figure 3.2: This figure shows the steps in ZKP 3.2, in which it is proven that, with probability $1 - \frac{1}{2^N}$, B_0 is a valid ballot. Note that this ZKP does not have separate rounds.

For example, if $D(B_0) = D(B_i)$, π_i is the identity permutation and

$$\xi_i = (\chi_i, \chi'_i) = (x_0 x_i^{-1}, x'_0 x_i'^{-1}).$$

Thus, by equations (2.2), (2.1), (3.3), and (3.4), $B_0 \otimes_{\pi_i} B_i^{-1} = E_{\xi_i}(0, 0)$. This shows that all such B_i are of the same form as B_0 .

Because approximately half of the N extra ballots were chosen at random to be revealed, and the other half were shown to be formally equivalent to the actual ballot, p has shown that B_0 is a valid ballot with probability $1 - \frac{1}{2^N}$. Therefore, this is a zero knowledge proof that B_0 is a valid ballot. \square

3.5 Tallying

Now the votes for each candidate are tallied. Again, let multiplication refer to the operation on the group of encrypted vote-elements that is homomorphic to addition on the group of vote-elements mod r . Each candidate-specific tally is then the “product” of all the encrypted vote-elements that are associated with that candidate. Because the cryptosystem is homomorphic — $E_{x_1 x_2}(e_1 + e_2) = E_{x_1}(e_1) \cdot E_{x_2}(e_2)$ — the votes for a particular candidate are tallied to an encrypted sum.

In our two-candidate example, the tally computed by A , along with its certificate, is as follows:

$$\mathbb{W} = \bigotimes_{i=1}^s \text{ballot}_i \tag{3.5}$$

$$\mathbb{T} = D(\mathbb{W}) \tag{3.6}$$

$$\mathbb{X} = \bigotimes_{i=1}^s \eta_i \tag{3.7}$$

where η_i is the pair of secret keys, (x_1, x_2) , of the i^{th} voter, and $\text{ballot}_i = E_{\eta_i}(e_i, e'_i)$ is the i^{th} voter’s encrypted vote. Then \mathbb{T} is a vector (a_1, a_2) , the tallies for candidates 1 and 2, respectively.

A publishes these \mathbb{W} , \mathbb{T} , and \mathbb{X} . Since anyone can verify equation (3.5) and that $\mathbb{W} = E_{\mathbb{X}}(\mathbb{T})$, the public can verify that, given that the cryptosystem is correct and that all ballots are correct, the published tally is correct. These other proofs are in Sections 3.4.1 and 3.4.2. If A is not implementing Instant Runoff, that's all there is — the one with the most votes wins.

Chapter 4

Extensions

There still remain several problems with this voting scheme. Voters can still be coerced to vote in particular ways, and the authority has the power to reveal votes. Furthermore, strategic voting and IIA paradoxes remain a problem for elections among three or more candidates so long as we do not implement Instant Runoff and the Borda Count. In order to address all of these, we will extend the election scheme described in Chapter 3. We will begin by examining these extensions independent of each other.

4.1 Poly-Candidate Elections

In order to extend this election scheme for use with m candidates, we can simply encode the votes as m -dimensional vectors,

$$(e_1, \dots, e_m).$$

4.1.1 Plurality Voting

In a plurality voting system, a valid vote for the i^{th} candidate would be of the form described above, where

$$e_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

These ballots can be proven valid, and tallied, in a way similar to that described in ZKP 3.2 and Section 3.5. In order to accomplish these goals, we need be able to “add” encrypted votes, and so we extend the operation \otimes :

$$(a_1, \dots, a_m) \otimes (b_1, \dots, b_m) = (a_1 b_1, \dots, a_m b_m).$$

Also, we need to be able to reorder B_0 to demonstrate that it is of the same form as a sample ballot B_i , and so we extend the operation \otimes_π :

$$(a_1, \dots, a_m) \otimes_\pi (b_1, \dots, b_m) = ((a_1, \dots, a_m) \times \pi) \otimes (b_1, \dots, b_m)$$

The final tally is generated in the same way as in Section 3.5, but with these new definitions of votes, ballots, and the operation \otimes . The result is similarly extended: \mathbb{T} is a vector containing the sum of the votes for all the candidates.

4.1.2 Borda Count

In implementing the Borda Count for m candidates, the values of the vote elements are slightly more complicated. For each voter, let π_0 be the permutation taking the canonical ordering of the candidates to the voter's preference ranking of the candidates. A valid vote expressing the preference ranking generated by π_0 is then of the form:

$$(m - 1, m - 2, \dots, 0) \times \pi_0.$$

Tallying is then as described in Section 4.1.1.

4.2 Instant Runoff

The voting system that is most discouraging to strategic voting, and least likely to bring about an IIA paradox, is to remove one candidate at a time from the election. The obvious choice is the lowest ranked candidate

Each time a candidate is removed, all preference rankings must be collapsed. This means that, for each vote, all candidates ranked below the candidate to be removed have their rank increased by one. This will happen in all tallies where the bottom-ranked candidate is not unanimously rejected.

To find a second winner, or a runner-up, we do not take the last eliminated candidate, who lost the final vote to the first-place winner. Instead we remove the first-place winner from the election and start the entire tallying process again. This ensures that candidates who lost early to the first-place winner are not eliminated for that reason alone.

The challenge here lies in removing candidates from the election. When using the permutations described in Sections 4.1.1 and 4.1.2, it is not possible to remove a candidate and verifiably collapse the voters' preference rankings without revealing how they voted.

Instead, we must encode the voters' preference rankings as a vector of $\binom{m}{2}$ pairwise votes, each of which represents the voter's preference between a pair of candidates that are not both associated with any other single vote in the vector. That is, if the i^{th} pairwise vote in the vector is associated with candidates G and N, then no other pairwise vote in the vector is associated with both G and N.

Conveniently, the Borda tally for each candidate is equivalent to a sum of all pairwise votes for that candidate. Here, we will only consider instant runoff using the Borda Count.¹

In order to understand and tally votes, we define an algorithm to take the previously described poly-candidate votes to this set of pairwise votes. While there is a canonical ordering to the pairwise votes, and there is a method to compute the values of the pairwise votes, it can be difficult to grasp. Thus, we will develop some tools and terminology with which to examine an alternative method.

Each pairwise vote is associated with one pair of candidates, and the valid values are the same as for a two-candidate election. A pairwise vote is said to be decided

¹While it is possible to implement Instant Runoff for other voting systems, it is less intuitive.

when its values are set, and it is otherwise undecided. A candidate is said to have undecided votes, and the candidate is called undecided, if any of the pairwise votes associated with that candidate are undecided. A pairwise vote is said to be decided in favor of the candidate whose vote-element is 1, and against the candidate whose vote-element is 0. Finally, we say that the highest-ranked undecided candidate is next.

We decide all pairwise votes associated with the first-ranked candidate in that candidate's favor. Since this candidate is associated with $m - 1$ pairwise votes, this will decide one vote for every undecided candidate, giving the first-ranked candidate as many points as there are lower-ranked candidates.

Now, suppose that the i^{th} -ranked candidate is next. Because all higher ranked candidates are decided, this candidate has $m - i$ undecided pairwise votes. We decide these votes in this candidate's favor. Now this candidate has as many points as there are lower-ranked candidates, and each lower-ranked candidate has one less undecided vote.

By induction on i , we decide all pairwise votes such that each candidate has as many votes decided in their favor as there are lower-ranked candidates.

It is still necessary to define the canonical association of candidates with pairwise votes. We say that the first candidate in the canonical ordering of candidates is associated with the first element of the first $m - 1$ pairwise votes. Each of those votes is associated, in order, with the remaining candidates. Now the next $m - 2$ pairwise votes similarly have their first elements associated with the second candidate, and so on.

For example, consider the three-candidate case, with candidates C_1, C_2, C_3 . The vote $[C_3 > C_1 > C_2]$ would be expressed as follows:

$$\begin{pmatrix} C_1 & (1, 0) & C_2 \\ C_1 & (0, 1) & C_3 \\ C_2 & (0, 1) & C_3 \end{pmatrix}$$

For $(a_1, \dots, a_{\binom{m}{2}})$, pairwise votes in an instant runoff ballot B , let the notation $B \times \pi$ mean:

- Associate the pairwise votes $a_1, \dots, a_{\binom{m}{2}}$ with the candidates $(C_1, \dots, C_m) \times \pi$ as if this reordering were the canonical ordering.
- Rearrange the ballot B such that it has the “canonical” associations with respect to this new “canonical” ordering of (C_1, \dots, C_m) . That is, if $(C_1, \dots, C_m) \times \pi = (C_{i_1}, C_{i_2}, \dots, C_{i_m})$ then the first $m - 1$ pairwise votes are associated with C_{i_1} and C_{i_2}, \dots, C_{i_m} — similarly for the remaining pairwise votes.

In order to verify and tally the ballots, we now extend the \otimes operation once again. For vectors of encrypted pairwise votes $(a_1, \dots, a_{\binom{m}{2}})$ and $(b_1, \dots, b_{\binom{m}{2}})$,

$$(a_1, \dots, a_{\binom{m}{2}}) \otimes (b_1, \dots, b_{\binom{m}{2}}) = (a_1 \otimes b_1, \dots, a_{\binom{m}{2}} \otimes b_{\binom{m}{2}})$$

where the \otimes function on the pairs of pairwise votes, a_i, b_i , is as described in equation (3.2), and

$$(a_1, \dots, a_{\binom{m}{2}}) \otimes_{\pi} (b_1, \dots, b_{\binom{m}{2}}) = ((a_1, \dots, a_{\binom{m}{2}}) \times \pi) \otimes (b_1, \dots, b_{\binom{m}{2}}).$$

Now ballot and tally validation are as described for pairwise votes, in ZKP 3.2 and Section 3.5, except that \mathbb{W}, \mathbb{T} , and \mathbb{X} are now vectors of pairwise values. Once these numbers have been verified, a tally can be generated for each candidate by adding together all of the vote-elements associated with that candidate. Alternatively, this can be done prior to decryption using the homomorphic operation, multiplication, on the encrypted vote-elements.

4.3 Receipt-Free Voting: A Backwards Ballot Transfer and a Public Vote

One problem with the pairwise election schemes described up to this point is that a dishonest party could require a voter to submit a particular vote. They could do this by requiring the voter to reveal their set of private keys, η_0 , allowing the dishonest party to decrypt the voter's ballot. Thus, it is desirable not only to allow privacy, but to require it. [1]

In order to require privacy, we change the voting protocol such that the voter does not have the secret keys for B_0 , nor for any ballot whose relationship to B_0 has been established. To do this, the authority generates a B_0 for each voter and proves that it is valid, and then the voter submits a permutation that is used to interpret that B_0 .

Let us call the j^{th} voter V_j , and call this voter's ballot $B_{0,j}$. Rather than having V_j publish $B_{0,j}$, A randomly chooses a $\pi_{0,j}$ with which to permute the canonical ballot — in which the candidates are ranked in alphabetical order — such that $D(B_{0,j}) = D(B_{\text{canonical}}) \times \pi_{0,j}$.

The authority generating the ballot transmits $D(B_0)$ to V_j via a private channel. Then A engages in ZKP 3.2 as the Prover, with $\pi_{i,j}$ chosen at random. Before the third round of the proof, A sends to the voter, again through a private channel, the permutations $\pi_{i,j}$. Given this proof and these permutations, the voter can verify with probability $1 - \frac{1}{2^N}$ that the ballot $B_{0,j}$ is as claimed by A .

Now, since V_j knows what $B_{0,j}$ represents, V_j can publish a vote, Π_j , that indicates a permutation of the candidates from their canonical ordering, to be used in interpreting $B_{0,j}$. In essence, $B_{0,j}$ represents a way of assigning votes to indices, and Π_j assigns indices to candidates.

Tallying is now as follows:

$$\mathbb{W} = \bigotimes_{j=1}^s B_{0,j} \times \Pi_j \quad (4.1)$$

$$\mathbb{T} = D(\mathbb{W}) \quad (4.2)$$

$$\mathbb{X} = \bigotimes_{j=1}^s \eta_{0,j} \times \Pi_j \quad (4.3)$$

4.3.1 Receipt-Free Instant Runoff Elections

When combining instant runoff and receipt-free elections, the tallying and verifying is somewhat more complicated. The ballots consist of a set of encrypted pairwise votes, while the voters' votes are permutations on the ballots. It is important that these permutations be interpreted correctly.

The voter's vote redefines the associations between pairwise votes and candidates. Instead of having the associations described in Section 4.2, the pairwise votes will be associated with the numbers $1, \dots, m$, in the canonical way, as though these numbers were the candidates. The voter's vote is then a permutation on these numbers, assigning numbers to candidates.

Let the operation \times be redefined to take an instant runoff ballot and a permutation and return another instant runoff ballot having the same pairwise votes, possibly rearranged. If the pairwise votes are rearranged, this is done as follows:

- Associate the pairwise votes with the numbers $(1, \dots, m)$, as though they were candidates.
- Associate the numbers $((1, \dots, m) \times \pi)$ with the candidates.
- Associate the pairwise votes with the candidates with which their numbers are associated.
- Rearrange the pairwise votes so that they are associated with candidates in the canonical way.

Now tallying is as described in Section 4.3.

Chapter 5

Extending the Scheme for Use with Multiple Authorities

Perhaps the greatest concern with the system described thus far is that it places all of the secrets in the hands of a single authority. If, for some reason, this authority is untrustworthy, it could easily abuse its power by revealing the decryption function. In the encryption scheme described in Section 3.2, this corresponds to releasing p and q . A could also reveal any individual vote by releasing the set of secret keys, η , with which that vote was encrypted. Alternatively, A could decide to review the voting record, and punish those who voted in a way that A did not like.

If such an Authority has the ability to enforce negative or positive consequences on any voter, that voter's vote may be compromised. In order to prevent this behavior, it is desirable to introduce multiple authorities.

Here we use the same cryptosystem as before, but we now assume that $|A| = \ell \geq 2$. We assume that at least t members of A are trusted. Several different entities are members of the Authority, and information is sent to the voter from these multiple sources. In this scheme, the ballot will be divided among the authorities in such a way that t of them are required to tally and decrypt the ballot. The divided ballot will be concealed through the use of secret *blinding factors*.

The scheme presented here is similar to that of [1], but it includes modifications of the author's design, resolving the problems pointed out in [7].

5.1 Blinding Factors

Consider a single voter. Each voting authority, A_j , sends to the voter the encryption, z_j , of a random blinding factor, χ_j , and uses a private interactive proof to prove the value of that blinding factor to the voter [1].

Interactive Proof 5.1. *The blinding factor χ_j is the number claimed by A_j .*

Figure 5.1 illustrates this proof.

While this is clearly an interactive proof, it is, importantly, also private. The voter can use the transcript of this proof to generate a false proof for any other $\chi \in \mathbb{Z}_r^*$.

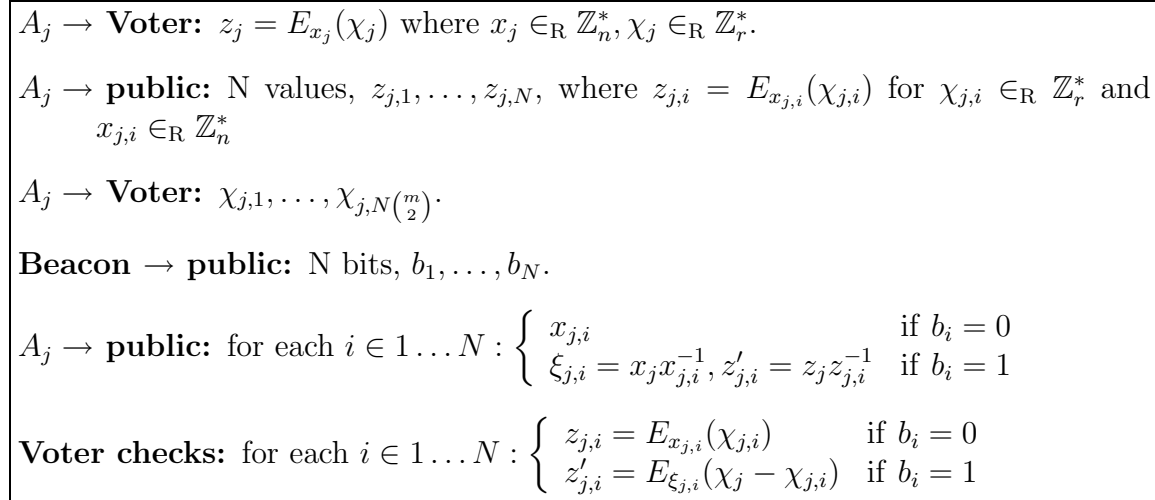


Figure 5.1: This figure shows the steps in IP 5.1

Because the voter got $\chi_j, \chi_{j,1}, \dots, \chi_{j,N}$ before the bits were generated by the beacon, the voter can be assured, with probability $1 - \frac{1}{2^N}$, that the authority actually sent χ_j . □

5.2 Voting Revisited

In order to share a voter's vote as a secret among the authorities, we define a blinding and sharing function, Γ , such that

$$\Gamma_j(e) = P_e(j) - \chi_j,$$

where

$$P_e(j) = \sum_{i=0}^{t-1} a_i j^i$$

for $a_0 = e$ and $a_1, \dots, a_{t-1} \in_{\mathbb{R}} \mathbb{Z}_n^*$. Recursively extend Γ such that

$$\Gamma_j((a_1, \dots, a_k)) = (\Gamma_j(a_1), \dots, \Gamma_j(a_k))$$

Each voter will now get $N + 1$ blinding factors from each authority, and use these to blind and share $N + 1$ ballots among all authorities A_j . The secret-sharing scheme used here is as described in [1], except that the randomness of the ballot permutations, rather than being determined by the voter or some vote-buyer, is determined by secret permutations from the authorities.

5.2.1 Secret Bits — Randomness from the Authorities

A proof similar to IP 5.1 and ZKP 3.2 can be performed to transmit secret permutations to the voter from each authority. The voter thus acquires N secret permutations, $\pi_{j,1}, \dots, \pi_{j,N}$, from each authority, and applies them in sequence to get N random permutations, π_1, \dots, π_N , to be used in the proof of ballot validity.

5.2.2 Ballot Validity

The voter generates $N + 1$ unencrypted ballots, B_0, \dots, B_N , such that $B_i = B_0 \times \pi_i$. The voter then sends $\Gamma_j(B_0, \dots, B_N)$ to the public for all $j \in [1, l]$.

Now, N bits are requested from the bit-beacon to determine, for each $i \in [1, N]$, whether to reveal all $x_{j,i}$ and $\chi_{j,i}$ or, after having the authorities engage in an interactive proof of all $\pi_{j,i}$, to reveal all $\chi_j - \chi_{j,i}$. The first of these options reveals the blinding factors for B_i , while the second combines B_i with B_0 in such a way that they are shown to be of the same form.

5.2.3 Multi-Authority Tallying

Because t points uniquely determine a polynomial of degree $t - 1$, up to $\ell - t$ members of A can misbehave without disrupting the election. To tally the votes, the A_j each tally all of their shares of the votes, and then reveal the sum of their blinding factors $\chi_{j,0}$ and the sum of their vote-shares, $\Gamma_j(B_0)$. The result is a tally in the form of an unencrypted ballot whose elements are shared as polynomials crossing the y-axis at the sum of the votes. It is simple then to find the winner of the election using the methods described in earlier sections.

Chapter 6

Discussion

6.1 Voter Authentication

One of the issues facing this, and any, election scheme is voter authentication. How can the public in general, or any verifier in particular, be certain that a voter V is who they claim to be? There are three main issues here: uniqueness, validity, and “singleness”. These mean that a verifier is concerned that V be the only such V , that V really be V , as claimed, and that V not be one among several voting identities used by this voter.

In real world terms, the voter V who claims at the voting booth to be Dana must be able to prove that V is only voting once as Dana, that V really is Dana, and that V isn’t also voting as Sam in another town or voting area. The first two are generally addressed by the voting booth model. The third is very difficult to prove — anyone solving this problem should contact the author.

In some situations, it is feasible to employ some other method of voter authentication besides voting booths. Particularly in environments with ubiquitous or nearly ubiquitous computing, a Kerberos-like authentication system seems like a good alternative to voting booths. One benefit of using Kerberos-like authentication is that it provides secure channels for communications between the voters and the authorities, which are assumed for all receipt-free schemes.

Concerns about replay attacks in Kerberos-like environments can be addressed by reducing the lifetimes of authentication for voting purposes. Due to the insecurity of DES, it is also recommended that anyone implementing these protocols use an authentication system based on something else, e.g.: AES.

In environments without ubiquitous computing, it seems inappropriate to allow voting from places other than public voting booths. If voting booths were disallowed entirely, this would disenfranchise the large portion of the population that lacks computing facilities, whereas allowing both voting booths and remote computer voting would make it easier for the wealthy to vote. As such, voting booths equipped with the simple machines necessary to this scheme, and possibly a Kerberos-like authentication system, are the recommended facility for voting. Local procedures for verification of voters should probably remain in place, largely to ensure voter authentication properties 1 and 2.

6.2 Security

The discrete log problem in \mathbb{Z}_n^* is the basis of the security of the encryption scheme described in Section 2.2.1, but it is not the basis of the security of this election scheme. If the discrete log problem is cracked, another encryption system can be used, and the zero knowledge proofs used for validation and verification will remain secure.

The discrete log problem in \mathbb{Z}_n^* is equivalent to factorization of n , which is currently computationally infeasible, and assumed to be hard. The best publicly known algorithm for factorization of n takes, at best, $O(e^{\sqrt{\ln n \ln \ln n}})$ time[10].

In the case of the zero knowledge proofs, breaking the encryption scheme and revealing a vote is equivalent to solving the discrete log problem, and thus to factorization. If another one-way trapdoor function were to be used for the homomorphic encryption scheme, ballot-cracking would be equivalent to solving the problem on whose hardness the encryption scheme depends.

There is a private channel assumed between the voter and the authority for a small portion of this protocol. This channel should rely on strong encryption and or totally secure communication. Given the infeasibility of the latter, the former is sufficient. This leaves us with man in the middle, known ciphertext, and possibly known plaintext attacks. The cryptosystem used for these communications should be robust against these.

For convenience, given that Kerberos is recommended for user authentication, it is also recommended that Kerberos secure messaging be used for private communications between the voter and the authority or authorities. The same caveat applies here as before: it is better to use a modern cryptosystem, such as AES, than to use one that has been demonstrated to be insecure, such as DES.

To some extent, current voting systems require voters to trust the government. In our current systems, the government is trusted to count our votes, not to burn them. They are trusted not to record (or allow the recording of) our voting process (by videotape or whatever other means), or to fingerprint our votes. Apparently, we trust our government to count our votes properly, and to abide by the result, though it is apparent from the U.S. Presidential election of 2000 that this trust may be misplaced. Using this protocol, we do not have to trust that the government will properly count our votes, only that they will not reveal how we have voted. With the extension to multiple authorities described in Chapter 5, we only need to trust that at least t of the authorities will not reveal how we voted.

6.3 Implementation

One of the possibilities raised in the proposal for this thesis was the implementation of whatever usable system resulted. While this was not done, it seems entirely feasible. Algorithms exist to make the necessary computations, and numerous students at Reed College seem excited about the prospect of having a thesis as their voting system.

The main drawback that most students at Reed seem to see with this system is the admitted improbability of allowing write-in candidates.

6.3.1 No Write-Ins

One of the consequences of using the Borda Count is that write-in candidates cannot be included in the election. If they were, a strategic voter could displace the candidates they dislike using write-ins that are unlikely to win, in some cases generating huge tallies for their favored candidate. In order to emulate the behavior of voting systems that allow write-ins, a nominations round is suggested, during which any voter can suggest any number of candidates for the ballot. Only those candidates nominated by a threshold percentage of the voters would appear on the ballot. Alternatively, the m most popular candidates could appear on the ballot, though this seems troublesome — it seems that the choice of m would be harder to make than the choice of a threshold. Given that nominations could be submitted and processed electronically, they could be taken almost up until voting begins. If electronic nominations are not desired, local governments can keep track of which valid voters nominated whom, so as to eliminate multiple nominations of the same candidate by a single voter.

It may also be possible to allow write-in candidates when the write-in is top- or bottom-ranked. This would be significantly more difficult in a single-authority, receipt-free election, in which the voter would have to indicate in some way the need for an extra candidate. If write-ins were to be allowed, this would also raise the issue of how to encrypt write-in candidates' names, how to resolve their names when they differ only slightly from that of another candidate, and how and when to reveal the names.

In general, allowing write-ins seems to be problematic, and allowing a nomination round seems fairly easy in any situation where this system would be feasible.

6.3.2 Ease of Use

Were a system such as this to be implemented, it could be very easy to use. A voter in this system could enter the voting booth, supply a username and password to the vote-computer, and indicate, for each pair of candidates, which was preferred.

A person wishing to verify that the voting authority behaves properly could, before the start of voting, verify the authority's ability to count votes with a few simple computations, using IP 3.1. After the close of the polls, this person could watch for the published results and verify them as described in Section 3.5.

To verify the validity of every ballot would take somewhat longer, but it would not be impossible. A verifier with this goal could scan the published proofs of ballot validity for any mis-cast ballots. Alternatively, a group of verifiers could divide the set of ballots, and each examine a portion of them.

6.4 Conclusion

We have seen that it is possible to design a secure and verifiable election system that uses the instant runoff and Borda Count voting methods, and is extensible to any other positional voting method.

If this design falls short, it is in that it does not include voter authentication. The difficulty of this problem is similar to a difficulty faced, it seems, by all attempts at security: in order to accomplish anything, it is necessary at some point in time to trust someone other than oneself.

Bibliography

- [1] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th Symposium on Theory of Computing (STOC '94)*, pages 544–553, New York, 1994.
- [2] Josh Cohen Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters. *PODC*, pages 52–62, 1986.
- [3] Steven Carter. Global positioning at 22. *Oregonian*, May 2000.
- [4] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372–382, Portland, OR, October 1985.
- [5] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Theory and Application of Cryptographic Techniques*, pages 103–118, 1997.
- [6] Ronald Cramer, Ivan Damgård, and Stefan Dziembowski. On the complexity of verifiable secret sharing and multiparty computation. In *ACM Symposium on Theory of Computing (STOC '00)*, pages 325–334, Portland, OR, 2000.
- [7] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Theory and Application of Cryptographic Techniques*, pages 539–556, 2000.
- [8] RSA Laboratories. Rsa labs cryptography faq: What are the best discrete logarithm methods in use today? <http://www.rsasecurity.com/rsalabs/faq/2-3-8.html>, 2002.
- [9] Emmanouil Magkos, Mike Burmester, and Vassilis Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *I3E*, pages 683–694, 2001.
- [10] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, <http://www.cacr.math.uwaterloo.ca/hac/>, October 1996.
- [11] Sun Microsystems. Crypto-politics: Decoding the new encryption standard. <http://research.sun.com/features/encryption/>, 2001.

-
- [12] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, December 1978.
- [13] NIST. Commerce department announces winner of global information security competition. http://www.nist.gov/public_affairs/releases/g00-176.htm, October 2000.
- [14] A. M. Odlyzko. On the complexity of computing discrete logarithms and factoring integers. Bell Laboratories, Murray Hill, New Jersey.
- [15] Donald G. Saari. *Chaotic Elections! : A mathematician looks at voting*. AMS, 2001.
- [16] J. Steiner, C. Neuman, and J. Schiller. An authentication service for open network systems, 1988.
- [17] Martin A. Tompa. Zero knowledge interactive proofs of knowledge. In Moshe Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, number 2 in Theoretical Aspects of Reasoning About Knowledge, pages 1–12, Monterey, California, March 1988. IBM Research Division, Morgan Kaufmann.
- [18] Dave Wreski. Linuxsecurity.com speaks with aes winner. http://www.linuxsecurity.com/feature_stories/interview-aes-1.html, October 2000.