ENCRYPTED RECEIPTS FOR VOTER-VERIFIED ELECTIONS USING HOMOMORPHIC ENCRYPTION

Joy Marie Forsythe
MIT

Key words: *encryption, voter verification, homomorphic encryption, Pailler encryption*

VTP WORKING PAPER #41
September 2005

Encrypted Receipts for Voter-Verified Elections Using Homomorphic Encryption

by

Joy Marie Forsythe

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author . . . .
Department of Electrical Engineering and Computer Science
August 16, 2005

Certified by . . . .
Ronald L. Rivest
Viterbi Professor of Computer Science
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Encrypted Receipts for Voter-Verified Elections Using Homomorphic Encryption

by

Joy Marie Forsythe

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Voters are now demanding the ability to verify that their votes are cast and counted as intended. Most existing cryptographic election protocols do not treat the voter as a computationally-limited entity separate from the voting booth, and therefore do not ensure that the voting booth records the correct vote. David Chaum and Andrew Neff have proposed mixnet schemes that do provide this assurance, but little research has been done that combines voter verification with homomorphic encryption. This thesis proposes adding voter verification to an existing multi-candidate election scheme (Baudron et al.) that uses Paillier encryption. A "cut and choose" protocol provides a probabilistic guarantee of correctness. The scheme is straightforward, and could easily be extended to multi-authority elections. The feasibility of the proposed scheme is demonstrated via a simple implementation.

Thesis Supervisor: Ronald L. Rivest
Title: Viterbi Professor of Computer Science

Acknowledgments

I would like to thank the Caltech/MIT Voting Technology Project for there generous assistance during my Master's program. In addition to funding my research during the first semester, VTP provided an interdisciplinary environment that allowed me to make my research more relevant.

I am deeply in debt to Professor Rivest for his guidance over the past year. I would also like to thank Ben Adida and Be Blackburn for their aid.

Most of all, I would like to thank Eric J. Cholankeril for his editing and motivational skills.

Contents

List of Figures

Chapter 1

Introduction

Requirements for an election vary by country and election type, but there are certain properties that are a starting point for all voting systems.

1. Democratic – each eligible voter must be able to vote and may vote at most once.

2. Private – a voter's final ballot must be secret.

3. Uncoercible – a voter cannot prove the contents of her final ballot to anyone.

4. Accurate – the final tally is the sum of the cast votes.

5. Verifiable – an individual can prove to herself that her vote was cast as intended and that it was counted, and anyone can prove that the final tally is accurate.

6. Robust – a small group of people cannot disrupt the election.

7. Fair – Partial totals should not be known early.

It is also important for an election to be convenient and flexible for the voters and officials. Voters will be less likely to vote if the process is complicated and difficult to understand. Officials are unlikely to adopt a system that cannot support voting practices particular to their districts, such as write-in votes and instant runoff elections.

Paper-based voting systems have been the standard since the mid-19th century, when secret votes became the norm. Electronic systems, often called Direct Recording Electronic (DRE) systems, have become more prominent recently. In a society that is increasingly turning to technology to automate and streamline everyday tasks, it is natural to apply technology to an institution as important as elections. Electronic voting systems have the potential to improve accuracy and security of elections as well as alleviate many of the logistical headaches.

One of the major advantages of DRE systems is the potential for consistent implementation of security policies. A machine does only what it is programmed to do, whereas human behavior is situation-dependent and may bias the election system. Despite this potential, most DRE systems still rely exclusively on the integrity of election officials and training of poll workers to ensure the election maintains the proper security and privacy. In order to believe her vote was properly recorded and tallied, the voter must trust election officials in her district, the technicians that set up the machines, the programmers that wrote the software, and the engineers that designed the hardware. She needs to trust that the machines were stored in a way that prevents tampering, and that they have been properly monitored since being removed from storage. She needs to trust that they will be securely delivered to the counting location after the polls close.

Since this issue has come to the forefront in 2000, there has been a push to integrate security into voting systems and thereby eliminate the reliance on trusted third parties. In particular, many have focused on the problem of trusting that the voting machine has recorded the proper vote. Of the two common types of cryptographic voting schemes, only mixnets have proposals for addressing this problem. David Chaum [7] has proposed using visual cryptography to allow the voter to verify that the ballot encrypts the correct choices. Andrew Neff has proposed [19] using receipts with codes corresponding to particular candidates. No such proposals exist for homomorphic encryption voting systems, which have the advantage maintaining greater privacy by never revealing the contents of individual ballots.

The goal of this thesis is to provide a secure and private homomorphic voter-verified election scheme. Chapter 2 of this thesis examines current voting technologies, and Chapter 3 surveys existing cryptographic research in voting. Chapter 4 proposes a new homomorphic scheme with a process for the voter to verify that the machine records and counts her vote properly. The voter is presented with several possible encrypted ballots and asked to choose one among them to use to cast her vote. The remaining ballots are decrypted to reveal whether they were properly formed by the voting machine. This straightforward "cut and choose" protocol provides a probabilistic proof of the voting machine's correctness. Section 4.4 presents a Java implementation that was created to demonstrate the scheme. Finally, Chapter 5 compares the new scheme to those proposed by Chaum and Neff, and discusses areas for future work.

Chapter 2

A Brief Analysis of Current Voting Technologies

This chapter will provide a brief discussion of voting systems used in current elections. This thesis focuses on electronic systems, but it is important to understand the advantages and disadvantages of both paper-based and electronic systems. Section 2.1 analyzes paper-based systems and Section 2.2 focuses on electronic systems and the arguments for and against voter verification.

A more thorough survey of the topic can be found in the Caltech/MIT Voting Technology Project's report [24]. The Election Reform Information Project has a series of briefings [25] on election reform topics such as security.

2.1 Paper-based voting systems

Auditability is the primary argument for paper systems. If ballots are stored safely and securely, there can be as many independent audits as needed. The fact that the audits can be independent is especially important. Each recount involves examining the original ballots, as marked and verified by the voter, rather than relying on a machine's recording of the ballots.

The second major advantage of paper-based ballots, voter verifiability, has become more prominent since the 2000 U.S. presidential election. Many have looked to paper systems to guarantee voters that their ballots were cast as intended because all paper-based systems involve permanently marking a piece of paper. After a voter makes her choice, she can visually inspect the paper to ensure the correct choice is indicated. As long as the voter selects a candidate, the vote indicated cannot be changed without invalidating the race or ballot.

The primary types of paper systems are hand-counted, punch-card, and optical-scan ballots. They differ in the method of marking choices and tabulating the results. The paper systems vary in ease of use and ease of tabulation.

Both hand-counted and optical-scan ballots are marked by hand using a marker or pen. The voter is asked to fill in an oval, put an "x" in a box, or complete a line to indicate her choice. If the ballot is designed well, this is a very intuitive action, and it is readily apparent to the voter which choice she has selected.

Using a punch-card ballot involves punching a hole to indicate a choice. The voter is provided with a device that maps ballot locations to ballot choices, and must punch holes to indicate her choices. This action is complicated, and it is often difficult for users to associate their choices with the holes they punched, especially after the ballot is removed from the polling device. This reduces the voter's ability to verify her choice.

With respect to tabulation, hand-counting is infeasible for elections on the scale of US national elections [24]. It is too slow, expensive, and cumbersome given the complexity of the ballots. However, hand-counting remains a backup method of auditing all paper-based systems. A hand-count of a small statistical sample can trigger a full recount if the distribution of votes differs significantly from that of the electronic or mechanical count.

Optical-scan tabulators are more portable and less expensive than punch-card machines, which makes it easier for them to be placed in polling locations. In-precinct scanning allows an invalid ballot to be rejected immediately, so that the voter has an opportunity to try again. The VTP [24] report shows that precinct-scan setups can reduce the number of uncounted ballots by 50%.

The punch-card machines are more expensive and cumbersome and have been shown to lose more votes than both hand-counted and optical scan systems [24]. This was highlighted during the 2000 presidential election when the Florida recount was bogged down by punch-cards with holes that were not definitively punched [15].

2.2 Electronic voting systems

There are two main groups of supporters for electronic voting systems: voters interested in the convenience and usability of the systems, and election officials interested in a simpler, more flexible, and less costly system. No studies that conclusively demonstrate that electronic voting is more usable exist, mainly because there are so many different systems. This technology is also in its infancy and the cited advantages are not necessarily apparent in current systems.

2.2.1 Usability advantages of DREs

The length of current ballots creates problems for paper-based systems. Elections are rarely a one-race affair and there are typically many more than two candidates for each race. Elections are also used as an opportunity to present referenda on public issues, which are typically written using legal terminology and are difficult to understand. The result ballots that are often double sided and printed in small font sizes. Even with the Federal Election Commission's mandated minimum 6.3 mm character size [9], many elderly and impaired voters are unable to read ballot text.

Electronic systems do not need to display all of the ballot information at once; instead races can be displayed individually. This allows the font size to be increased. For voters with vision impairments, there could be settings with even larger font sizes and greater contrast.

The issues of ballot design go far beyond font size. The ballots should be designed to convey which candidates are running in which races and how to cast a vote for each candidate. However, many current designs fail to do this well, and voters miss key information [24].

An under-vote occurs when a voter does not select a candidate for a race. While it is allowable for a voter to choose not to vote in a race, if the voter casts an under-vote because she did not see the race, it is an error. With large and complicated ballots, these errors are more common.

With DRE machines, races can be presented individually. The voter can be forced to either choose a candidate or acknowledge that she is not voting in the race. This could reduce under-votes because the voter must explicitly choose not to vote in a particular race.

Using a paper-based ballot, a voter can mark multiple selections for a race where only one selection is allowed. This is known as an over-vote. Whether this is due to stray marks or confusion, the result is that the voter's choice is invalidated because election officials are unable to determine the voter's intent. A computer can disallow selecting more than the allowed number of candidates and thereby eliminate over-votes.

Another advantage of DREs is the voter's ability to change her ballot without the intervention of election officials. If a voter marks her ballot, then wishes to change her choice, most paper-based systems would require that she turn in her old ballot. This policy results in a lack of privacy for the voter, who may have only marked one choice incorrectly and is now forced to reveal the rest of her choices. To avoid this, electronic systems allow voters to change their votes without any intervention from election officials. Whether doing so is simple and straightforward depends largely on the user interface.

The extent to which the improvements discussed above are present is dependent on the quality of the user interface. Many current DREs do not achieve these improvements. If electronic systems become widely used, the user interface is sure to improve as more vendors compete to deliver electronic voting systems, and as data about the usability of individual systems becomes available.

Accessibility and DRE systems

The American Association of Disabled People (AADP) is one of the most vocal voting groups supporting the DRE voting machines. The AADP favors the machines because they are more accessible than any current systems – in particular, they allow disabled people to cast secret ballots. With paper-based systems, many disabled people rely on another person to fill out their ballots; this destroys the secrecy of the process and leaves such voters wondering if their votes were cast as intended.

DRE machines can be adapted to accommodate disabilities and allow such voters to vote without assistance. Vision-impaired voters can use headphones and systems that provide verbal feedback. Mobility-impaired voters can use alternative input mechanisms to make their selections. These features provide disabled voters with the ability to vote unassisted, an ability most voters take for granted. In addition to increasing accessibility and improving the secrecy of voting for disabled people, these features are mandated by the Help America Vote Act [22].

2.2.2 Logistical advantages of DREs

For election officials, DREs provide the potential to reduce costs and alleviate many problems in current processes. Using paper-based voting systems, election officials must securely and efficiently print, distribute, transport, and count millions of ballots. Managing this paper is an enormous logistical feat, and even with many years of accumulated wisdom, election officials have not gotten it entirely right. Voting experts regularly describe lapses of security such as ballot boxes that are unaccounted for hours after the polls close, a time period during which they could easily be altered [24].

Each voting district, and possibly each polling location, may have a different ballot due to differences in local races. In primary elections, there must be different ballots for each party. Many voting districts must also provide ballots in different languages for voters who are not native speakers of English. (In some cases, it is not even possible to print ballots, as certain native languages in Alaska and the northwest have no written form.) The result is an enormous number of ballots that must be printed and distributed to specific locations. Electronic voting machines can easily support multiple ballots, and could even support audio ballots for non-written languages.

The differences between ballots prevent voters from voting at an alternate polling location if the assigned location is not conveniently located. Electronic voting may allow voters to easily vote at an unassigned polling location.

Undeniably, maintaining the security of paper ballots after they are cast is extremely difficult, especially if they are counted centrally as they are in many large districts. Additionally, officials must retain and securely store ballots for 22 months after they are cast [9]. With DREs, transferring ballots to a central location is reduced to setting up a secure connection to the polling locations or transporting a small amount of electronic material, rather than boxes of paper. Storing ballots on electronic media requires much less physical space, and the media can be easily destroyed when no longer needed.

Another major problem with paper ballots is the difficulty of counting them. Hand-counting has the advantage of being easily observed by multiple parties, but is largely impractical with voting districts as large as Los Angeles County, which has almost four million registered voters [17]. Automated methods, such as optical scan ballots, speed up the process but vote counts can be inconsistent, as ballots may be marked in ways that are unclear to the machines, resulting in recounts that differ from the original counts even if no fraud exists [12]. DRE machines produce final vote counts instantaneously and consistently. Consistency and speed are appealing to voting officials because the voting public appreciates immediate and definitive results.

DRE voting machines have a high initial cost, but the repeat cost of individual elections is lower than with paper-based systems. Printing ballots is very expensive and must be done for each election. The VTP report [24] calculated that DRE machine with a life span of 15 years is more expensive than an optical scan machine over the same time period. However, if the machine lasts 20 years, the cost is the same.

2.2.3 Disadvantages of DREs

The major criticism of the DRE voting systems is that they give voters no confidence that the machines are doing the proper thing. After a voter submits her ballot, she has no way of knowing that the machine is recording and counting the vote as entered. To believe this occurred, the voter must trust that the vendors did not intend to misrecord votes, that the software developers performed their job competently, that the software was properly certified, and that the machine is running the certified software. This also assumes that the certification standards are high enough to ensure proper security.

The problem with trusting the vendors' intent is that the companies making these machines may not be unbiased parties. The companies that produce voting machines, as well as the executives that run those companies, have a history of supporting and donating to particular political campaigns[29]. Furthermore, some of the officials responsible for selecting and regulating electronic voting equipment are elected. There is clearly a conflict of interest in these cases.

In other situations where partisan individuals are responsible for critical electoral processes, efforts are made to disclose their actions as much as possible and to allow members of any political group to participate. One example is the presence of party observers at poll closings. Poll workers, themselves of varying political beliefs, are watched by representatives of any candidates that choose to provide them. Imposing a similar process on the production of voting machines is not feasible.

Vendors claim that suspicions of bias are unfounded because the software must go through a verification process. However, detecting intentionally faulty software is very difficult. For example, a Rice University professor asked computer science students to introduce bugs into a simple voting system and asked other students to examine the code for bugs [2]. Despite a small code base, only 2,000 lines, some bugs went undetected. Compare this to commercial voting systems with over 50,000 lines of code [2]. Even with professionally trained auditors, malicious bugs could go undetected.

Beyond the issue of vendor intent is the problem of vendor competency. It is extremely difficult to achieve correctness in software, as evidenced by the bugs discovered in commercial software on a daily basis. While some bugs are to be expected, some of those discovered in current election systems provide very little confidence in those writing the software. One of the more publicized such bugs was the hard-coding of keys into the software [30]. This meant that every election district using that software had the same key, and that the key could not be changed without changing the underlying software. These keys were used to encrypt all of the ballots and to set up the memory cards used to authenticate voters. Knowledge of these keys could allow an adversary to cast extra votes, among other things.

One way of reassuring the public of the impartiality and correctness of the voting system is to test the system using predefined standards. Currently, election systems are certified by individual states, based on results from both federal and state tests. These tests generally include auditing the code for errors. The current process is considered inadequate by many, especially because "commercial off the shelf" software is allowed to be included without being audited for errors. Commercial off the shelf software, such as operating systems purchased by vendors from other companies, is used as-is in the voting machines. The current process also treats certification as a one-time process and does not provide an opportunity for citizen involvement or significant public disclosure [12]. The result is that voter confidence is not particularly high. An improved certification process would help improve trust, but examining the code and running tests can never completely ensure correctness, especially if the programmer is malicious.

Another way of improving security and gaining public trust is to require that voting machine software be open source. This solves the problem of transparency by allowing the public to participate in the development process as coders or observers. However, open source based voting machines are not likely to be profitable. A more limited approach would be to make the source code publicly available for evaluation only. In the end, open source can not completely eliminate errors or malicious code, although it may improve public trust.

Even if code can be certified such that the public has complete confidence in it, this will not ensure that the software running on the voting machines is the certified code. Vendors have a history of putting uncertified software onto voting machines without the knowledge of the election districts [29, 31]. This problem can be prevented with rigorous oversight of the installation process and using hash libraries [21] to compare installed software to certified software. These measures may slow down the process of fixing bugs, but they will make the maintenance of the software more transparent to the public.

Other voting systems have similar problems with achieving trust, but manage to avoid the criticism heaped on DREs because it is possible to recover from fraud by recounting ballots. DRE voting machines have no meaningful recount ability. Optical scan machines use software that is susceptible to the same fraud and correctness errors as DREs, but the ballots are not affected by such errors. Optical-scan ballots can be manually recounted if necessary. In contrast, the only copies of the ballot on a DRE machine are the ones the machine chooses to store. Even if large errors such as obvious candidate bias are detected, no recovery is possible.

2.2.4 DREs and voter verification

It is clear that there are many reasons to worry about the DRE machines, even if some problems can be alleviated in the long term. The question of providing assurance that a voter's vote is cast as intended and counted properly remains to be solved. Some form of "voter verification" is necessary. Rebecca Mercuri was apparently the first to suggest that DREs print a paper receipt that the voter cannot take home [18]. The machines would print out a receipt behind a glass window, so that the voter would be prevented from marking or removing it. The voter would then have the opportunity to examine the receipt before choosing to submit her vote. If approved by the voter, the receipt would be put into a sealed ballot box. The receipt serves as a "voter verified paper trail" or "voter verified paper audit trail." This method makes recounts possible, since the paper ballots approved by the voter can be recounted.

While the idea of a voter verified paper trail has gained support from many lawmakers and computer scientists, there are also many who strongly object to them. Using the term "contemporaneous paper trail," they criticize the effectiveness, expense, and feasibility of such a system [28].

When using voter verified paper trails the question is, "What is the official ballot?" While state laws will ultimately determine the answer, the paper ballot, not the electronic ballot, is supposed to provide the final vote count in the event of a challenge. Essentially, the DRE machine has become a device that records the ballot on paper and maintains an unofficial count. The obvious problem is that many of the disadvantages of a paper-based voting system are retained, including some printing costs, the expense of storing the receipts, and the difficulty of managing the paper securely at all times.

Paper trails also reduce the usability of DRE machines for disabled voters. The vision-impaired would not be able to verify the paper receipt, and many object to such systems because resources used for creating and maintaining paper trails could be used instead to improve the overall accessibility of the machines [22]. The practicality of these machines is another serious issue. Many opponents point to the possibility that the printers will fail as a new election day disaster [28].

The goal of the voter verified paper audit trails is worth pursuing. A voter should be able to convince herself that her vote was recorded as she intended and that the vote was included in the final tally. However, counting paper ballots is a technological step backwards. Instead, modern cryptography can offer similar assurances without losing the advantages of modern electronic systems.

Chapter 3

Use of Cryptography in Voting Systems

The main problem with current DRE systems is that they require a large amount of trust from the election officials, who are either elected officials themselves or else appointed by elected officials. However, there has been a significant amount of research on providing cryptographic schemes that reduce this burden of trust. A more detailed survey of the topic can be found in *Secure Electronic Voting*[13].

The problem is that cryptography is often added as an afterthought, rather than than as an integral part of the voting system. An end-to-end scheme allows the voter to verify that her vote was cast as she intended and that the ballot cast was included in the final tally. This should all occur in a secure manner that ensures a fair election while maintaining the privacy of the voters.

There are three general classes of cryptographic voting protocols: those based on blind signatures, those based on mix-nets, and those using homomorphic encryption. Historically, cryptographic research has focused on proving that the tally is the sum of all the ballots, and that the contents of individual ballots remain secret. Cryptographic voting research considered the voter and the polling booth to be one entity. In 2004, two new mix-net based schemes due to Chaum [7] and Neff [20] were proposed that provide true end-to-end verification and enable the voter to verify that the voting machine recorded the correct vote.

Section 3.1 describes homomorphic encryption and its applications to voting. Mix-net voting systems are described in section 3.2, and two voter-verifiable mix-net schemes are discussed in section 3.3.

3.1 Homomorphic Encryption

Homomorphic encryption is naturally suited to election schemes. It allows the votes to be tabulated before decryption, improving privacy. For example, in additive homomorphic encryption, the product of two ciphertexts is a third ciphertext that encrypts the sum of the two original plaintexts.

More generally, let $\perp$ be an operation, m_1, m_2 be two messages, and let $E[m]$ represents the encryption of the message m under an encryption scheme. The scheme is homomorphic for the operation $\perp$ if you can easily find a ciphertext $c = E(m_1 \perp m_2)$ from $E(m_1)$ and $E(m_2)$. That is, the operation $\perp$ can be performed on the underlying messages without revealing them. For election systems, a scheme where $\perp$ is a addition is most useful.

Voting applications may use additive homomorphism to allow tallying to be done before decryption. With other forms of encryption, all the ballots are dissociated from their identifying pieces of information and then decrypted and tallied. If homomorphic encryption is used, the tallying can be done while the votes are still encrypted, and the final total can then be decrypted. This effectively hides the contents of the original ballots while providing an publicly computable tally.

Section 3.1.1 presents a basic two-candidate homomorphic election scheme. Section 3.1.2 describes Paillier encryption, while 3.1.3 proves the security and one-wayness of the scheme. Possible improvements to the basic scheme are discussed in 3.1.4, and an expanded multi-candidate version is described in 3.1.5.

3.1.1 A very basic homomorphic encryption scheme

Before introducing the Paillier encryption scheme, it is necessary to examine exactly how homomorphism can be used in an election protocol.

The most basic type of election is a two-candidate race with v voters where everyone raises their hand for their preferred candidate. To construct an equivalent electronic system, let 0 represent a vote for the first candidate and 1 represent a vote for the second. Everyone posts their vote in some public manner. If the sum of all the votes is less than $v/2$, the first candidate wins. If it is greater, the second candidate wins. If the sum is exactly equal to $v/2$, there is a tie. However, this scheme obviously lacks privacy. If each voter instead posted a homomorphic encryption of her vote, the encrypted ballots could be multiplied and then decrypted to find a plaintext sum of the votes.

There are many issues with this simplistic approach. The first is that there is no proof that the voter submitted a valid vote. Instead of an encryption of 0 or 1, the voter could submit an encryption of a larger number or a negative number and thereby corrupt the sum. Potential for fraud also exists in the decryption operation – it must be done in a verifiable way. Giving any one authority the power to decrypt can also threaten the privacy of individual votes because that authority now has access to the contents of every voter's ballot. Most races contain more than two candidates, so the candidates must be encoded in a way that preserves the summation property. Another major issue for homomorphic election schemes is support for write-in candidates.

3.1.2 Paillier encryption

A public-key encryption scheme frequently used in homomorphic voting systems was designed by Pascal Paillier [23]. It is additively homomorphic and computationally efficient to decrypt. It will be the basis of the scheme proposed in chapter 4.

Paillier encryption is provably secure and one-way based on the Decisional Composite Residuosity Assumption (DCRA) and the Computational Composite Residuosity Assumption (CCRA). We present here an explanation of the scheme drawn from the original paper [23].

Let p and q be two large primes and $n = p*q$. Two functions we will use frequently are Euler's totient function (ϕ) and Carmichael's function (λ). For n, the product of two primes, $\phi(n) = (p-1)(q-1)$ and $\lambda(n) = \mathrm{lcm}(p-1, q-1)$. These functions are used because they have nice properties over the multiplicative group $\mathbf{Z}^*_{n^2}$:

$|\mathbf{Z}^*_{n^2}| = \phi(n^2) = n\phi(n);$

and

$w^{\lambda(n)} = 1 \pmod{n},$

$w^{n\lambda(n)} = 1 \pmod{n^2},$

for any $w \in \mathbf{Z}^*_{n^2}$.

We will also make use of the function $L(u) = (u-1)/n$, $\forall u \in \{u | u = 1 \pmod{n}\}$.

A common term in modular arithmetic is residue, where a is a residue of b modulo n if $a = b \pmod{n}$. A number z is said to be an n-th residue modulo n^2 if there exists a $y \in \mathbf{Z}^*_{n^2}$ such that $z = y^n \pmod{n^2}$. Each n-th residue z modulo n^2 has n such roots y less than n [23]. The set of all n-th residues is a multiplicative subgroup of $\mathbf{Z}^*_{n^2}$. Each n-th residue z has n roots, of which exactly 1 is less than n. In particular, the n-th roots of 1, called the n-th roots of unity, are $(1+n)^x = 1 + xn \pmod{n^2}$ for $x \in \{0, \cdots, n-1\}$.

We can now define the function ε_g, which maps $\mathbf{Z}_n \times \mathbf{Z}^*_n$ to $\mathbf{Z}^*_{n^2}$:

$\varepsilon_g(x, y) = g^x * y^n \pmod{n^2}$

This will be our encryption function, where x is a message encrypted under public key g. To use ε_g as an encryption function, we need to show it is bijective in message for a fixed key. If we choose g such that the order of g is a nonzero multiple of n, ε_g can be inverted.

Lemma 1 (equivalent to Lemma 3 from [23]): *If the order of g is a nonzero multiple of n then ε_g is a bijective map from $\mathbf{Z}_n \times \mathbf{Z}^*_n$ to $\mathbf{Z}^*_{n^2}$.*

Proof: [Proof expanded from Paillier's original paper [23].] We will show ε_g is bijective for g when g has order equal to αn for all $\alpha \in \{1, \cdots \lambda(n)\}$.

Let h be the order of g. To show that ε_g is injective, we will demonstrate that for any $c \in \mathbf{Z}^*_{n^2}$, $x_1, x_2 \in \mathbf{Z}_n$, $y_1, y_2 \in \mathbf{Z}^*_n$, $c = \varepsilon_g(x_1, y_1) = \varepsilon_g(x_2, y_2)$ if and only if $x_1 = x_2$ and $y_1 = y_2$. Let

$g^{x_1} y_1^n = g^{x_2} y_2^n \pmod{n^2},$

which can be simplified to

$g^{x_1-x_2}(y_1y_2^{-1})^n = 1 \pmod{n^2}$.

We can then raise both sides to $\lambda(n)$ and get

$g^{\lambda(n)(x_1-x_2)}(y_1y_2^{-1})^{\lambda(n)n} = 1 \pmod{n^2}$.

When we introduced the function $\lambda(n)$, we also made the following statement: for all $w \in \mathbf{Z}^*_{n^2}$, $w^{\lambda(n)n} = 1 \pmod{n^2}$. This implies

$g^{\lambda(n)(x_1-x_2)} = 1 \pmod{n^2}$.

From this, we know that $\lambda(n)(x_1 - x_2)$ is a multiple of g's order, h. By definition, $\gcd(\lambda(n), n) = 1$, therefore $x_1 - x_2 = 0 \pmod{n}$ and $x_1 = x_2 \pmod{n}$.

If we go back to

$g^{x_1-x_2}(y_1y_2^{-1})^n = 1 \pmod{n^2}$,

we can now determine that $(y_1y_2^{-1})^n = 1 \pmod{n^2}$, which makes $y_1y_2^{-1}$ an n-th root of 1. The roots of 1 take the form $1 + \beta n$ for $\beta \in \{0, \cdots, n-1\}$, therefore $y_1y_2^{-1} = 1 + \beta n \pmod{n^2}$ and $y_1 = y_2 + y_2\beta n \pmod{n^2}$. From this we get $y_1 = y_2 \pmod{n}$. Therefore, ε_g is injective for the chosen g.

Using Euler's totient function, we can show that the two groups $\mathbf{Z}_n \times \mathbf{Z}^*_n$ and $\mathbf{Z}^*_{n^2}$ each have $n\phi(n)$ elements and are therefore the mapping is surjective.

$\square$

In practice, we find g with an order that is a nonzero multiple of n by choosing a random element of $\mathbf{Z}^*_{n^2}$ and testing if $\gcd(L(g^{\lambda(n)} \pmod{n^2}), n) = 1$ as described by Paillier [23].

Lemma 2: *$\gcd(L(g^{\lambda(n)} \pmod{n^2}), n) = 1$ implies the order of g is αn for some $\alpha \in \{1, \cdots, \lambda(n)\}$.*

Proof: Let the order of g modulo n^2 be h. By Carmichael's formula, h divides $\lambda(n^2) = \lambda(n)n$. Therefore, $g^{\lambda(n)} = 1 \pmod{n}$ and $g^{\lambda(n)} = (1+n)^x \pmod{n^2}$ for some $x \in \{0, \cdots, n-1\}$. The order of $(1+n)^x$ modulo n^2 is $n * 1/x \pmod{\phi(n^2)}$ because $(1+n)^n = 1 \pmod{n^2}$. Similarly, the order of $g^{\lambda(n)}$ is $h/\lambda(n)$. We can now see that $h * 1/\lambda(n) = n * 1/x \pmod{\phi(n^2)}$ or $xh = \lambda(n)n \pmod{\phi(n^2)}$.

Given this, we can express $L(g^{\lambda(n)} \pmod{n^2})$ as $L(1 + xn \pmod{n^2}) = x$. We can now simplify $\gcd(L(g^{\lambda(n)} \pmod{n^2}), n) = 1$ to $\gcd(x, n) = 1$. Given this and that $xh = \lambda(n)n \bmod \phi(n^2)$, n must divide h and the order of g is a nonzero multiple of n.

$\square$

We can now define Paillier's encryption scheme, the first described by Paillier [23]. The public key is (n, g), where n is a product of two large primes and g is chosen such that ε_g is bijective. The secret key is $\lambda(n)$. To encrypt the message $m \in \mathbf{Z}_n$ under public key (n, g), we choose a random $r \in \mathbf{Z}_n^*$ and use

$E_g[m, r] = \varepsilon_g(m, r) = g^m r^n \pmod{n^2}$.

To decrypt the ciphertext c with private key $\lambda(n)$ we use

$D_g[c] = \frac{L(c^{\lambda(n)} \pmod{n^2})}{L(g^{\lambda(n)} \pmod{n^2})} \pmod{n}$.

The g will be omitted from E_g and D_g when it is obvious from the context.

Lemma 3:*Given $c = E[m, r]$, $D[c] = m$.* [Equivalent to Lemma 7 of [23]].

Proof: [Proof expanded from Paillier's original paper [23].] We first substitute in $c = g^m r^n \pmod{n^2}$ to get

$D[c] = \frac{L(g^{\lambda(n)m} r^{\lambda(n)n} \pmod{n^2})}{L(g^{\lambda(n)} \pmod{n^2})} \pmod{n}$.

From Carmichael's formula, $r^{\lambda(n)n} = 1 \pmod{n^2}$ and $g^{\lambda(n)}$ is an n-th root of unity and equals $xn + 1 \bmod n^2$ for some $x \in \{0, \cdots, n-1\}$. This gives us

$D[c] = \frac{L((xn+1)^m \pmod{n^2})}{L(xn+1 \pmod{n^2})} \pmod{n}$.

We can simplify $(xn + 1)^m \pmod{n^2}$ to $mnx + 1 \pmod{n^2}$ and end up with

$D[c] = \frac{mx}{x} \pmod{n} = m \pmod{n}$.

$\square$

One of the main advantages of Paillier encryption is that it is additively homomorphic. If we choose some $m_1, m_2 \in \mathbf{Z}_n$ and $r_1, r_2 \in \mathbf{Z}_n^*$, and let

$c_1 = E[m_1, r_1]$ and

$c_2 = E[m_2, r_2]$, we have

$c_3 = c_1 * c_2 \pmod{n^2} = E[(m_1 + m_2 \pmod{n}), r_3]$, for some $r_3 \in \mathbf{Z}_n^*$.

Lemma 4: *Paillier encryption is additively homomorphic.* [Equivalent to Lemma 5 of [23].]

Proof: Let $c_1 = E[m_1, r_1] = g^{m_1} r_1^n \pmod{n^2}$, $c_2 = E[m_2, r_2] = g^{m_2} r_2^n \pmod{n^2}$. We get

$c_3 = c_1 * c_2 = g^{m_1+m_2}(r_1 r_2)^n \pmod{n^2}$.

This will decrypt to $m_1 + m_2$.

$\square$

Security and one-way properties of Paillier Encryption

The problem of distinguishing an n-th residue from a non-n-th residue modulo n^2 is referred to as the Composite Residuosity problem, or $CR[n]$. An important property of $CR[n]$ is that it is randomly self-reducible: a particular instance of the problem can be randomly transformed into a derived instance and a solution to the derived instance can be transformed into a solution to the original instance. That is, if we are given an oracle $\mathcal{O}$ that answers $CR[n]$ for a random $z \in \mathbf{Z}_{n^2}^*$ with probability ρ, with have an ρ chance of using $\mathcal{O}$ to determine $CR[n]$ of a particular $w \in \mathbf{Z}_{n^2}^*$.

Lemma 5: *$CR[n]$ is randomly self-reducible over $w \in \mathbf{Z}_{n^2}^*$.*

Proof: Given $w \in \mathbf{Z}_{n^2}^*$, we let $w' = w * r^n \pmod{n^2}$, where $r \in_R \mathbf{Z}_{n^2}^*$. If $\mathcal{O}$ provides an answer to $CR[n]$ for w', the same answer can be returned for $CR[n]$ for w. Otherwise, fail.

To see that this works, consider the two possible cases. If w is an n-th residue, there will exist a root y such that $w = y^n \bmod n^2$. Therefore, $w' = y^n * r^n \pmod{n^2}$ and will be an n-th residue. If w is not an n-th residue, w' cannot be an n-th because $r^n \bmod n^2$ is an n-th residue and n-th residues modulo n^2 are a multiplicative group. If r is chosen randomly from $\mathbf{Z}_{n^2}^*$, w' will be randomly distributed in $\mathbf{Z}_{n^2}^*$ and the probability of choosing it such that $\mathcal{O}$ will answer $CR[n]$ is ρ. Therefore, $CR[n]$ is randomly self-reducible over all possible $w \in \mathbf{Z}_{n^2}^*$.

$\square$

The assumption that deciding $CR[n]$ is computationally hard is referred to as the Decisional Composite Residuosity Assumption (DCRA). This is dependent only the choice of n due to random self-reducibility.

Theorem 1 (equivalent to Theorem 15 from [23]): *Paillier encryption is semantically secure if and only if DCRA holds.*

Proof: [Proof expanded from Paillier's original paper [23].] We will first show that if Paillier encryption is semantically secure, DCRA holds, by proving the contrapositive. Given $m_0, m_1 \in \mathbf{Z}_n$ and $c \in \mathbf{Z}^*_{n^2}$, where c is the encryption of either m_0 or m_1, we need to determine which message c encrypts. We are also given an oracle $\mathcal{O}_{DCRA}(w)$ which returns whether w is an n-th residue modulo n. If $c = g^{m_0} r_1^n \pmod{n^2}$, $cg^{-m_0} = r_1^n \pmod{n^2}$ and will be an n-th residue. Therefore, if and only if $\mathcal{O}_{DCRA}(cg^{-m_0})$ is true, c is an encryption of m_0.

We will now show that if DCRA holds, Paillier encryption is semantically secure. Given $w \in \mathbf{Z}^*_{n^2}$ and an oracle for Paillier encryption, we can determine whether w is an n-th residue modulo n^2 by constructing $c = g^x w \pmod{n^2}$, $x \in \mathbf{Z}^*_n$, and giving c and x to the oracle. If it accepts c as a valid encryption of x, w is an n-th residue modulo n^2.

Therefore, Paillier encryption is semantically secure based on DCRA.

□

To prove the one-wayness of Paillier encryption, we will introduce the n-th Residuosity Class Problem of base g, or $Class[n, g]$. This is the problem of computing $x \in \mathbf{Z}^*_{n^2}$ given $w = \varepsilon_g(x, y)$. This problem is randomly self-reducible over both w and g.

Lemma 6 (equivalent to Lemma 6 from [23]): *$Class[n, g]$ is random self-reducible over $w \in \mathbf{Z}^*_{n^2}$.*

Proof: [Proof from Paillier's original paper [23].] We are given n, the product of two primes, g, a member of $\mathbf{Z}^*_{n^2}$ with an order that is a nonzero multiple of n, and $w \in \mathbf{Z}^*_{n^2}$. We can create a random instance $w' = wg^\gamma \omega^n \pmod{n^2}$, where $\gamma \in_R \mathbf{Z}_n, \omega \in_R \mathbf{Z}^*_n$. Let x' be the result of the $Class[n, g]$ oracle on w'. We can solve for

$$x = x' - \alpha \pmod{n}.$$

□

Lemma 7 (equivalent to Lemma 7 from [23]): *$Class[n, g]$ is randomly self-reducible over all g with order equal to αn for some $\alpha \in \{1, \cdots, \lambda(n)n\}$. That is, for all g_1, g_2 with orders a nonzero multiple of n, $Class[n, g_1]$ is equivalent to $Class[n, g_2]$.*

Proof: [Proof expanded from Paillier's original paper [23].] Earlier we showed ε_g to be bijective, so we can assume inverses exist for all proper g. For any $w \in \mathbf{Z}^*_{n^2}$, there exists $(x_1, y_1) = \varepsilon^{-1}_{g_1}(w)$ and $(x_2, y_2) = \varepsilon^{-1}_{g_2}(w)$, $x_1, x_2 \in \mathbf{Z}_n$ and $y_1, y_2 \in \mathbf{Z}^*_n$. Let $(z, y_3) = \varepsilon^{-1}_{g_1}(g_2)$. Therefore, $g_2 = \varepsilon_{g_1}(z, y_3) = g_1^z y_3^n \pmod{n^2}$.

We can substitute in for g_2 in $w = \varepsilon_{g_2}(x_2, y_2)$ and get

$w = (\varepsilon_{g_1}(z, y_3))^{x_2} y_2^n \pmod{n^2}$.

We can simplify to get $w = g^{zx_2}(y_2 y_3^{x_2})^n \pmod{n^2}$ and $w = \varepsilon_{g_1}(zx_2, y_2 y_3^{x_2})$.

However, we know that ε is bijective, therefore $x_1 = x_2 z \pmod{n}$. Another way of expressing this is $D_{g_1}[w] = D_{g_2}[w] D_{g_1}[g_2]$. We can also establish that $D_g[g] = 1$ for all proper g by observing $g = \varepsilon_g(x, y) = g^x y^n \pmod{n^2}$ for $x, y = 1$. Using this identity, we get $D_{g_1}[g_1] = D_{g_2}[g_1] D_{g_1}[g_2]$.

If we have an oracle for $Class[n, g_1]$ we can determine $Class[n, g_2]$ of w by asking the oracle for $\varepsilon^{-1}_{g_1}(g_2)$ and $\varepsilon^{-1}_{g_1}(w)$ and solving $\varepsilon^{-1}_{g_2}(w) = \varepsilon^{-1}_{g_1}(w)\varepsilon^{-1}_{g_1}(g_2)$. Therefore, $Class[n, g]$ is random self-reducible over g.

□

$Class[n]$ is $Class[n, g]$ for a specific g, conditioned only on n. Due to random self-reducibility, this is equivalent to $Class[n, g]$. The Computational Composite Residuosity Assumption (CCRA) is the conjecture that $Class[n]$ is intractable. It is obvious that Paillier encryption is one-way if and only if CCRA holds because inverting Paillier is by definition the Composite Residuosity Class Problem.

Paillier's original paper [23] connects the DCRA and CCRA to several problems generally believed to be in intractable. $Class[n]$, the basis of CCRA, is reducible to factoring n and $CR[n]$, the basis of DCRA, is reducible to $Class[n]$.

3.1.3 Improved homomorphic encryption schemes

Since Paillier first proposed his encryption scheme and suggested its relevancy to voting, there have been many different approaches to the problems mentioned at the end of 3.1.1.

One scheme is the multi-candidate, multi-authority scheme proposed by Baudron, Fouque, Pointcheval, Poupard, and Stern [3]. It allows races with multiple candidates and makes use of threshold cryptography to distribute the private decryption key among multiple authorities. The distribution of the key in this manner safe-guards the voters' privacy against malicious authorities. This scheme also adds verification to the encryption and decryption stages of the election scheme.

Another homomorphic voting scheme is the vector-ballot approach proposed by Kiayias and Yung [16]. This scheme is unique in its attempt to support write-in candidates. It makes use of mix-nets to anonymize the write-in ballots but the rest of the system takes advantage of the efficiency of homomorphic encryption.

3.1.4 Multi-candidate homomorphic election systems

Drawing from Baudron, Fogue, Pointcheval, Poupard, and Stern [3], this section describes a multi-candidate election scheme based on Paillier homomorphic encryption.

To set up a k-candidate election for v voters, we choose a m to be an integer greater than v. Note that m can be any integer greater than v and can be chosen to be something computationally convenient such as the next power of 2 larger than v. The public key $n = pq$ must be greater than m^k, g will be chosen in the usual way. The candidates must be assigned an ordering and candidate $i \in \{0, \cdots, k-1\}$ will be uniquely represented as $m_i = m^i$.

To vote for candidate i, the voter must encrypt m_i under the public key (n, g). She will then need to provide a zero-knowledge proof that her ballot is an encryption of a valid vote. To prove that ballot $b = E[m_i, r] = g^{m_i} r^n$, $r \in_R \mathbf{Z}_n$ for some $m_i \in M = \{m_0, \cdots, m_{k-1}\}$ the voter must present a zero-knowledge proof. We use non-interactive zero-knowledge proofs because this information will be posted publicly and we wish to preserve the privacy of the voters while still allowing for universal verification.

To use the proof described by Baudron, Fouque, Pointcheval, Poupard, and Stern [3], the voting machine would compute $k-1$ values, one for each unchosen candidate in the final ballot, based on the ciphertext and the unused messages. The verifier gives the machine a random challenge to which it must respond with the computed values and one final value computed based on the challenge. This prevents cheating by the voting machine because it is not possible to compute the final value if the ciphertext does not encrypt a message from M. This particular proof has length $O(k)$, but an $O(\log(k))$ proof has been described Damgard, Jurik and Nielsen [11].

After the ballots b_i are submitted for all $i \in \{0, \cdots, v-1\}$, we can accumulate and decrypt them. Let $B = \prod_i b_i \pmod{n^2}$ be the accumulated ballots. $T = D[B]$ is the decrypted sum of all the votes. We then need to determine the number of votes for each candidate. If we interpret T as an m-ary number, $T = \{m_{k-1}, \cdots, m_1, m_0\}$, we can determine the final tallies for each candidate.

3.2 Mix-nets

This section describes a generic mix-net voting system, an alternative approach to cryptographic voting systems. This category of voting schemes is interesting because there are proposals that incorporate end-to-end verification into the mix-net protocol.

Mix-nets were introduced by David Chaum [8] as a way to anonymize email. Like ballots in an election, the goal was to dissociate the encrypted message from its sender. To ensure privacy, multiple authorities share key information, and the identity of the sender can only be determined if multiple authorities conspire. In voting, the authorities are usually opposing organizations such as political parties.

There are two types of mix-nets: decryption and reencryption. In a decryption mix-net, the messages are encrypted under all of the authorities' public keys and each authority partially decrypts the message. For a reencryption mix-net, the message is encrypted under a shared public key and reencrypted under each authority's private key. The following description focuses on decryption mix-nets.

If an election has v voters and a authorities, a basic mix-net election can be run in the following manner. Let PK_i, SK_i be the public/private key pair of authority i, for all $i = 0, \ldots, a-1$ for encryption function $E_{PK_i}[x]$ and decryption function $D_{SK_i}[x]$.

Let PK be a key that combines PK_i for all i. In the case of El Gamal encryption, which is described later in this section, PK is simply the product of all PK_i. Each voter encrypts her ballot b_j, for all $j = 0, \ldots, v-1$, with PK to get $c_{j,0}$ and posts it publicly.

When all v votes are posted, the first authority chooses a random permutation of the set $\{0, \cdots, v-1\}$, π_0, to reorder the elements and partially decrypts all of the ciphertexts with SK_0. It outputs $c_{\pi_0(j),1} = D_{SK_0}[c_{j,0}]$ for all $j = 0, \ldots, v-1$ to the public board. This is considered one stage of the mixing. All of the following stages are performed in a similar manner.

After reordering and partially decrypting, an authority then must prove that all the ballots that entered this stage also left it. A simple probabilistic method is to challenge the authority by randomly choosing half of the inputs and asking for proof that they correspond to outputs [14]. This could potentially lead to some privacy loss if the correspondence can be traced from the input ballot and voter to the final decrypted ballot. Instead, each authority can be responsible for two stages of mixing [7]. Again, half of the inputs are randomly chosen as challenges. Any output of the first stage that was not part of the challenge, becomes a challenge for the second stage. This prevents one ballot from being part of more than two successive challenges. These proofs can also be replaced with zero-knowledge proofs.

After the final mixing stage, the ballots are in plaintext form. They can then be tallied in a normal fashion. This tally is easily verified because the ballots are public. This doesn't violate the privacy of the voters because the votes have been anonymized.

El Gamal encryption is commonly used for mix-nets. Let p be a large prime and g be a generator modulus p. The private key is $x \in_R \mathbf{Z}_p^*$ and the public key is $y = g^x \pmod{p}$. The encryption function is $E_y[m] = (a, b) = (g^r \pmod{p}, my^r \pmod{p})$,

$r \in_R \mathbf{Z}_p^*$ and the decryption function is $D_x[c] = a^{-x}b \pmod p$. To generate PK from PK_i for $0 \leq i < a$, simply take the product of all the public keys. That is, $PK = \prod_{0 \leq i < a} PK_i$. To partially decrypt ciphertext $c = (a, b)$ with $SK_i = x$, let $c' = (a, a^{-x}b)$.

3.3 Cryptographic Approaches to Voter Verification

This section describes the existing proposals for cryptographic voter verification. These schemes take existing mix-net protocols and add voter verification to produce schemes with end-to-end verification. The work of David Chaum and Andrew Neff are discussed in sections 3.3.1 and 3.3.2.

As discussed above, mix-net voting schemes all achieve some assurance that votes are not tampered with once they are encrypted. The last step is to achieve verification that the encrypted ballot contains the voter's intended choices without reverting to relying on paper for a final vote count and while maintaining secrecy and anonymity. This is extremely difficult because voters cannot be given a traditional receipt, which would violate secrecy, but must be provided with some physical assurance of their votes being counted. Two schemes that handle this issue have been proposed. David Chaum proposed a scheme that uses visual cryptography to provide an encrypted receipt of the ballot [7] while Andrew Neff's idea uses a codebook of encrypted responses for each voter[20]. They have each outlined a process by which an entire election can be run, but this section will focus on how the receipts are generated and why they provide voter verification.

3.3.1 Chaum and Visual Cryptography

In Chaum's proposed scheme [7], the user enters her vote into the DRE machine as usual. After the ballot is filled out, but before it is submitted, a two layer transparency is printed. The layers are stuck together and the printing is done on the outer surfaces of the layers. Each pixel is represented by the two corresponding pattern squares on these transparencies such that if the patterns are the same, light can shine through, and if not, the square is opaque. The result is a visual method of "xoring" the two layers. The layers are produced from the ballot image determined by the voter through the ballot selection process, a pseudorandom number generation process based on the serial number of the ballot and the public keys of the trustees who will be responsible for decrypting the ballots. When the two layers lay on top of each other, the original ballot image is visible. If they are separated, both layers are encrypted.

At this point, the encrypted receipts have not been cutoff from the printer and the machine will be waiting for a confirmation that the ballot is correct. If the voter approves the ballot, she will then be asked to randomly choose a layer, top or bottom. A final signature will be printed on the layer chosen and the voter will remove the resulting ballot. To complete the process, the voter must shred the layer she did not choose. She should retain the chosen layer, which can later be used to verify after the election is completed that her encrypted ballot was among those counted.

Immediately after leaving the voting booth, the voter can verify that her ballot was encrypted properly by checking the signature on the layer and that the pseudorandomness revealed by that layer was generated properly. These computational tasks can be done through public algorithms. The voter can implement her own version of the verification software or rely on a trusted third party to provide the software that performs the verification.

At the end of the election, all of the layers retained by the voters are also published on a designated public website. These are the official encrypted ballots. A voter can check her receipt against what is posted and provide physical evidence of her vote if it does not appear or if it is different than posted.

If either of these verification steps fail, the voter can bring her ballot to the election authorities to prove that it was not properly formed.

A mix-net is then used to decrypt the ballots while dissociating the encryptions with the final plaintext. Multiple parties with differing interests can be included in this process to ensure the correctness and secrecy of this process.

Security of Chaum's scheme

There are three sources of fraud that Chaum's scheme protects against: the machine, the voter, and the trustee. Protection against machine fraud is achieved by forcing the machine to print the layers before it knows which layer the voter will choose. If the machine attempts to cheat by encrypting a different ballot image while making the transparencies combine to form the correct image, it will have to alter some of the pseudorandomness from what it would correctly use. If the user is equally likely to pick either layer, the machine has a 50% chance of choosing to cheat on the wrong layer. If even a small percentage of the receipts produced are checked for correctly formed randomness, there will still be an overwhelming likelihood that widespread fraud will be caught. Unlike any plaintext receipts, the encrypted ballot gives no information about the original ballot receipt. This prevents voters from selling their votes by proving who they voted for. Against trustees, anonymity and secrecy is achieved by using a mix-net to allow a series of servers to each remove one layer of encryption on each ballot and permute the results. Forcing each server to demonstrate half of the correspondences protects against cheating, and the use of mutually adversarial parties as trustees ensures anonymity as long as one honest trustee exists.

The obvious problems with this scheme are ensuring the destruction of the transparency not chosen, printing the receipts, and voter confusion. If any copy of the other transparency is retained, secrecy is destroyed. This will require vigilance at polls and careful consideration of how the information is stored on the machines. The printing is a huge problem, one that has come up with voter verified paper trails and is made more complex here with two-layer double-sided transparencies. Finally, voters will most likely find choosing between the two layers confusing because it will not be obvious to the average member of the public what purpose this serves. These are a few of the challenges of the scheme, some of which are unavoidable.

3.3.2 Neff and Votehere.net

In one of Neff's schemes, which is marketed commercially by votehere.net [20, 19], a voter receives a random ballot number from poll workers. After providing this number to the voting machine, a preloaded "codebook" is printed out and detached. This specifies each candidate or response for each question and a corresponding code. The voter then fills out her ballot as usual. Before the ballot is confirmed, the codes for the selected choices are printed out on a receipt. The voter can check the codes on her receipt against the codebook printed earlier to ensure her vote was recorded accurately. If the voter is satisfied, she accepts the ballot and takes the receipt and a printed signature. If not, she must request another ballot number and start over. Before leaving any polls, the voter must surrender and destroy the codebooks she received.

For this scheme, observers take an active role. Spot checks of the machines are produced throughout the voting process by having an observer enter a voting booth with an unused ballot number, print the codebook, and cancel the ballot. This codebook is checked against an independently stored record of the precomputed codebook values. If it differs, the machine is either malfunctioning or cheating.

At the end of the election, each ballot id is associated with the encryptions of the selections encoded on the ballot receipt. These ciphertexts are posted in a public location as the inputs to the mix-net which will anonymously decrypt the ballots. In addition to decrypting the ballots and tabulating the results, the trustees post the verification codes of the ballot choices, which are based on the encryptions. The voter can check that her ballot is listed and the codes listed on her receipt are those posted.

Observers protect against malicious voting machines. If a machine wishes to cheat, it must print an incorrect codebook to convince the voter that her choices are recorded properly. However, if there is a significant chance that an observer will be the receiver of the false codebook, the machine is unlikely to get away with widespread fraud. Giving the voter a receipt with encrypted choices prevents her from revealing her vote conclusively to a third party, unless they can steal the codebook, as only the voter saw the correspondence.

The remaining possibility of cheating lies with the trustees, which is prevented by choosing mutually adversarial parties and using mix-nets and threshold encryption for determining the verification codes.

The obvious problem with this scheme is managing the codebooks. Voters must not keep their codebooks. Additionally, the observers must be carefully administered to prevent them from casting extra votes or violating anonymity using their access to the codebooks. Trustees must not have access to the final codebooks, just their portions of them, otherwise they can directly associate the code with choices and violate anonymity. This process seems more physically feasible than the Chaum scheme, but places more trust in election officials.

Chapter 4

A Homomorphic Voter-Verifiable Election Scheme

The challenge in making an election scheme voter-verifiable lies in the fact that a voter, being human, has limited computational abilities. The most one can ask of a voter inside a voting booth is to compare two things and determine if they are different. All other computation must be deferred to a later point. This makes "cut and choose" a natural choice for a protocol with human verification.

Cut and choose was formalized by Brassard, Chaum, and Crepeau [4], though the idea first appears in the protocol described by M. Rabin [26]. The analogy used was the problem of sharing a cake between two mutually distrustful parties. Each party wants as large a slice as possible. To ensure that the slices are as equitable as possible, one party slices and the other chooses which slice to take. If the first party slices the cake unevenly, she will receive the smaller slice. Therefore, she is motivated to divide the cake fairly.

In this case, the voter and the voting booth can be thought of as the mutually distrustful parties. Since the voting booth is the one with the computational power, it does the slicing and the voter chooses a slice.

The scheme proposed in this thesis adds a layer of voter verification on top of existing homomorphic encryption techniques. Section 4.1 presents the scheme and describes the voter experience. Section 4.2 describes the mathematical details of the verification process, and section 4.3 proves the security of the scheme. Section 4.4 describes the implementation of the scheme and demonstrates how a typical election using the scheme might work.

4.1 Overview of the scheme

The scheme extends the voting protocol described in section 3.1.4. First, several possible ballots are prepared for a particular race. The order of candidates is randomly permuted on each ballot. After committing to a set of ballots, the voting machine presents them to the voter, who must select one ballot with which to cast her vote. The voting machine then provides proofs that the unchosen ballots match their commitments, which can be verified by the voter. This prevents the voting machine from falsifying any particular ballot without risking that the falsified ballot is not chosen, in which case the machine would not be able to prove the commitment.

This voter-verification protocol breaks down into three phases: inside the voting booth and receipt verification.

These are the steps inside the voting booth:

1. **Voting Machine:** Generate d ballots and print a commitment for each ballot to the receipt. Display a grid of the candidates committed to on the screen.

2. **Voter:** Choose a candidate from those printed on the screen.

3. **Voting Machine:** Print the row and column of the voter's selection, the the contents of the unchosen ballots and proofs that those commitments were correctly formed.

4. **Voter:** Verify that the ballot selected is the one identified on the receipt and that the commitment proofs correspond to what is displayed on the screen.

5. **Voter:** If satisfied that the voting machine behaved properly, approve the ballot. Otherwise, cancel the ballot and start again or contact an election official.

6. **Voter:** Remove the final receipt.

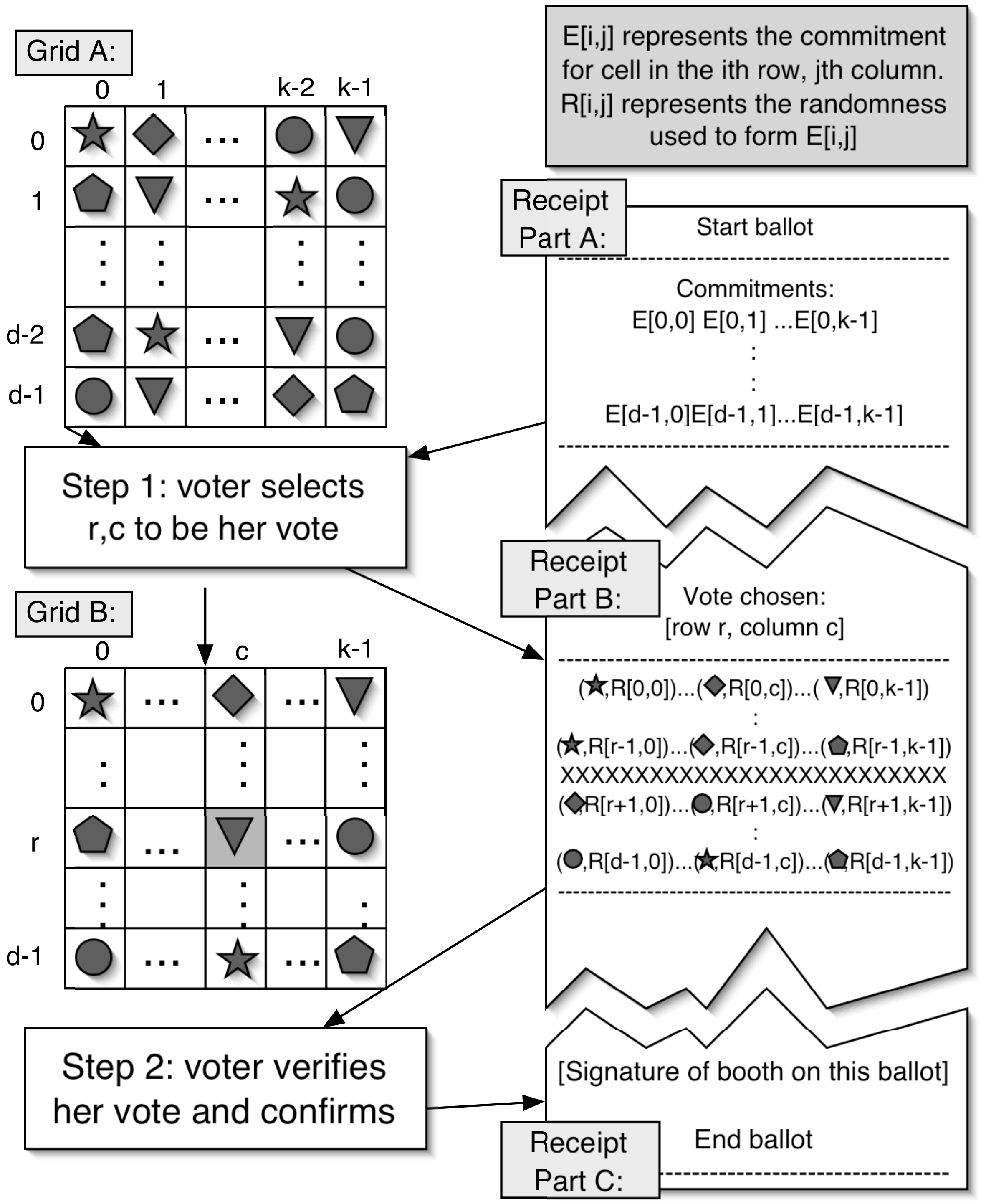
After leaving the voting booth, the voter can verify her receipt was correctly formed.

1. For each candidate in each unchosen ballot, check that the candidate revealed and the proof provided verify the commitment.
2. Verify that the receipt is among those posted to the official website and the posted receipt and paper receipt are identical.

Figure 4-1 shows the steps the voter must take and the inputs to each step. If the voter is asked to choose one of k candidates for a race, she is presented with a d by k grid on the screen as in Grid A of Figure 4-1, where d is an small integer security constant, such as 2 or 3. Each of the d rows contains a random permutation of the k candidates. When the grid is presented, a cryptographic commitment to the grid is printed. Receipt Part A in Figure 4-1 is this commitment. At this point, the voter must select the candidate she wishes to vote for and a row to vote in.

Let $r \in \{0, \cdots, k-1\}$ and $c \in \{0, \cdots, d-1\}$ be the voter's row and column choice. In Figure 4-1, this corresponds to a vote for the triangle candidate. This information is printed on the receipt tape, along with the grid displayed to the voter for rows 0 through $r-1$ and $r+1$ through $d-1$ and reveal information that proves the commitments were properly formed. Grid B and Receipt Part B pictured in 4-1 show this stage of the process.

The voter must then either confirm her vote or start with a new ballot. Before confirming, she should check that the row and column printed on the receipt matches the box chosen on the screen, and that the candidate information printed on the receipt for all rows except r matches the information on the screen. In Figure 4-1, this corresponds to matching the highlighted box in Grid B to the cell specified at the top of Receipt Part B and matching the rest of Grid B to the grid printed on Receipt Part B. If there are any discrepancies, the voter should discard the ballot and either start a new ballot or alert the poll officials. After she approves the ballot, the voting machine prints a digital signature to the receipt, and the voter removes it and leaves. The signature corresponds to Receipt Part C in 4-1.



Grid A:
0
1
k-2
k-1
0
1
d-2
d-1
E[i,j] represents the commitment for cell in the ith row, jth column. R[i,j] represents the randomness used to form E[i,j]
Receipt Part A:
Start ballot
Commitments:
E[0,0] E[0,1] ...E[0,k-1]
E[d-1,0]E[d-1,1]...E[d-1,k-1]
Step 1: voter selects r,c to be her vote
Grid B:
0
c
k-1
0
r
d-1
Receipt Part B:
Vote chosen:
[row r, column c]
(★,R[0,0])...(◆,R[0,c])...(▼,R[0,k-1])
(★,R[r-1,0])...(◆,R[r-1,c])...(⬟,R[r-1,k-1])
XXXXXXXXXXXXXXXXXXXXXXXXX
(◆,R[r+1,0])...(●,R[r+1,c])...(▼,R[r+1,k-1])
(●,R[d-1,0])...(★,R[d-1,c])...(⬟,R[d-1,k-1])
Step 2: voter verifies her vote and confirms
[Signature of booth on this ballot]
End ballot
Receipt Part C:

Figure 4-1: An overview of the proposed voter-verification scheme. The left hand side shows the grids that will be displayed on the screen, while the right hand side shows the receipt that will be printed. Note that there are three contiguous parts to the receipt which are divided to show the points at which the voter interacts with the voting machine.

After the election is concluded, all the ballot receipts are posted electronically for the public to review. A voter should check that the receipt she received in the voting booth was posted correctly. A voter should also verify that her ballot is correctly formed – i.e., that the reveal information matches the commitments. As in Chaum's scheme, trusted third parties such as the ACLU could provide software for this verification, or the voter could implement her own.

Because the votes for each race are encrypted homomorphically with Paillier's scheme, the encrypted tally is formed by multiplying all the votes for a particular race. The encrypted tally can be verified by anyone, using the publicly posted ballots. It can also be decrypted in a verifiable manner.

4.2 Mathematical details

A standard k-candidate election for v voters is set up using Paillier encryption. Let $n = pq$ and $g \in \mathbf{Z}^*_{n^2}$ with order αn for some nonzero α.

After the grid is generated, each cell contains the name of one of the candidates. For the cell in the xth column and yth row containing candidate i, let $c_{x,y} = g^{m_i} \rho^n_{x,y} \bmod n^2$, where $m_i \in M, \rho_{x,y} \in_R \mathbf{Z}^*_n$, be the commitment generated using Paillier encryption. The commitments for all cells are printed to the receipt. The commitments can be condensed by hashing, reducing the length of the receipt.

Let r, c be the row and column the voter chose as her vote. After printing the voter's selection, the machine opens the commitments for all rows except r. That is, $\rho_{x,y}$ for all y and $x \neq r$ is printed to the receipt along with the candidate that the corresponding cell contained. At this point the voter removes the receipt.

If, after leaving the voting booth, the voter chooses to verify her vote, it can be done by checking that all of the commitments in rows outside of r were properly formed. To check that a particular cell x, y is properly formed, construct $c'_{x,y} = g^{m_i} \rho_{x,y} \bmod n^2$ where m_i is the candidate printed on the receipt for that cell. If $c'_{x,y}$ is equal to the $c_{x,y}$ found on the receipt, the commitment was properly formed.

4.3 Security of the scheme

It can now be proven that the voting machine has a uniformly small chance of defrauding the voter. If the machine prints commitments to votes that do not match up to the grid on the screen, there are two possible outcomes.

If the false commitments are in the row the voter chooses, the fraud will not be discovered. The machine will not reveal the contents of this row because that would destroy the voter's privacy. However, if the voter's choice is sufficiently random and unbiased, the machine has only a $1/d$ chance of predicting the row the voter will select.

Otherwise, the machine will have to provide reveal information for the false commitment. The machine could change the screen to match the false commitment and risk the voter noticing. For example, in a race between candidates X, Y, and Z, a voting machine might wish to switch votes for X and Y. Looking at the grids below, the machine could commit to Grid A (on the left) but display Grid B (on the right) on the screen.

Z	Y	X
X	Z	Y
Z	X	Y

Z	X	Y
Y	Z	X
Z	Y	X

Suppose the voter selects Z for the first row, first column of Grid B, which is displayed on the screen. The voting machine can print Grid C, below, to the receipt, and change the screen to display Grid A instead of Grid B.

-	-	-
X	Z	Y
Z	X	Y

If the voter does not notice the reversal of X and Y in the second and third rows of the grid on the receipt, the machine has successfully cheated. The feasibility of such a reversal is discussed later, as it is more an issue of usability and human interface.

The second way for the machine to cheat is to provide the correct reveal information for the false commitment along with the candidate choice the voter expects. For example, say that the machine has committed to candidate Y for row i and column j, but displays candidate Z in that cell on the screen. Let $c_{i,j}$ be the commitment for the cell, which is an encryption of m_Y with some randomness $r_Y \in \mathbf{Z}_n$. If the voter chooses a row other than i for her ballot, the machine prints m_Z, r_Y for the reveal information even though this will not verify $c_{i,j}$ as described in section 4.2. The falsification will be detected if the voter chooses to cryptographically verify their receipt at a later time, but will not be noticed in the voting booth. The feasibility of this attack is dependent on the likelihood that the voter checks their ballot after leaving the voting booth.

The final possibility for the cheating voting machine is to generate a proof for the candidate choice displayed on the screen that corresponds to the false commitment printed to the receipt. More explicitly, given commitment $c = E[m_i, \rho_i]$ to candidate i, find ρ_j such that $c = E[m_j, \rho_j]$. Earlier it was shown that the Paillier encryption scheme is one-way based on CCRA. Therefore, it is not computationally feasible for the cheating voting machine to provide a ρ_i for the false commitment.

Therefore, the voting machine has a $1/d$ chance of undetectably cheating for each vote. The question of whether this is sufficient depends on many factors, not the least of which is d. If it is assumed that the voting machine needs to alter .5% of the votes to significantly affect the outcome of an election (this estimate may be too low, considering the winner's margin during the 2000 presidential election was less than .5% in four states [24]), the machine would need to change 10 votes to change the outcome of a 2,000 person election. If $d = 2$, the machine would have a 1 in 1,024 chance to succeed at altering .5% of the 2,000 votes.

To add more privacy and security, most of the techniques discussed in section 3.1.4 can be added onto the scheme with no modification. It is important to use zero-knowledge proofs to verify the correctness of the ballots, and threshold encryption to maintain the privacy of voters even from election officials.

4.3.1 Human factors and their effect on security

The chance that cheating is detected is actually much lower than described above due to human factors. According to the U.S. Census, only about 60% of eligible citizens registered and voted in the 2000 presidential election [1]. This apathy carries over to election day tasks, but the scheme requires that the voter perform several tasks in addition to actually selecting their candidate. A recent study [27] by Ted Selker found that less than 10% of voters noticed errors on the receipt in a simulated election with a voter verified paper audit trail.

Assume that only 1 out of every t voters actually follows through on all of the verification procedures. Without taking voter apathy into account, there was a $(d-1)/d$ chance that a machine's fraud would be detected by each voter that was cheated. This has been reduced to $(d-1)/(d*t)$. For his voter-verifiable scheme, Chaum assumes a t of 20 [7]. If this value is assumed in the case presented above where the machine altered 10 votes, and $d = 2$, there is around a 75% chance of successful fraud. However, successful cheating would involve far more votes in most elections. Altering just 200 votes reduces the chances of remaining undetected to .5%. Given that the 2000 presidential election had a turnout of 111 million, the chance of undetectably affecting the election seems negligible.

4.4 Implementation

An implementation of the scheme proposed in this chapter is now described. Section 4.4.1 describes our implementation of the Paillier scheme, and section 4.4.2 discusses the structure of the races and ballots. The tabulation process and actual voting experience are presented in sections 4.4.3 and 4.4.4, respectively.

To demonstrate the scheme, a basic implementation was done in Java 1.4.2. The implementation focused on the new aspects of the scheme and did not include some features, such as multiple authorities and verifiable decryption, that would be necessary for a real-world election. Also left out of this implementation were proofs that the final vote is actually from the set of acceptable votes. This is not necessarily an oversight, as the cut-and-choose protocol also provides probabilistic assurance of ballot correctness. All the references to printers refer to printing to the command line.

4.4.1 Paillier implementation

Despite being a common encryption scheme in cryptographic literature, there was no easily available implementation of Paillier encryption for Java. Therefore, a Pallier package was implemented for use in the voting system.

The Paillier keys are generated in the expected fashion by the class `PaillierKeyGenerator`. Two primes of a specified size are found, n and $\lambda(n)$ are calculated, and a g is found. The interfaces `PaillierKey`, `PaillierPrivateKey`, and `PaillierPublicKey` follow the conventions set up in the java.security package.

The other class in this package is `PaillierAlgorithm`, which implements the encryption and decryption function as well as providing methods useful for `PaillierKeyGenerator`. The client is given the option of providing a source of randomness for both the key generation and the encryption options.

4.4.2 Races and ballots

It is important to make the ballot implementation flexible, as a race can be a 100-candidate race or a 500-word proposition. The instances of the `Race` class are generated from a `RaceTemplate` which stores the name of the race or question, the candidates, and the integers that represent each candidate. Similarly, `Ballot`s are constructed from `BallotTemplate`s. Each such `BallotTemplate` contains the name of the election, `RaceTemplate`s, and the public key for the election.

When a new ballot is created, in addition to the `BallotTemplate`, the client must provide a unique ballot ID and d, the number of rows the grid for each race will have. This will be used to generate the `Race`s from the `RaceTemplate`s. The only time `Race` is altered is when the vote is selected by the `selectVote` method, which can only be called successfully once.

After a race is voted on, an instance of `RaceTally` is formed from the commitments and the reveal information of the unvoted rows. A separate class is used here because accessing individual cells within the commitment and reveal grids is not allowed during the actual voting process but is important for the tabulation and verification process. During the voting process, the grids can only be read in their entirety. `RaceTally`s are collected into a `BallotTally` in preparation for tabulation.

4.4.3 Tabulation

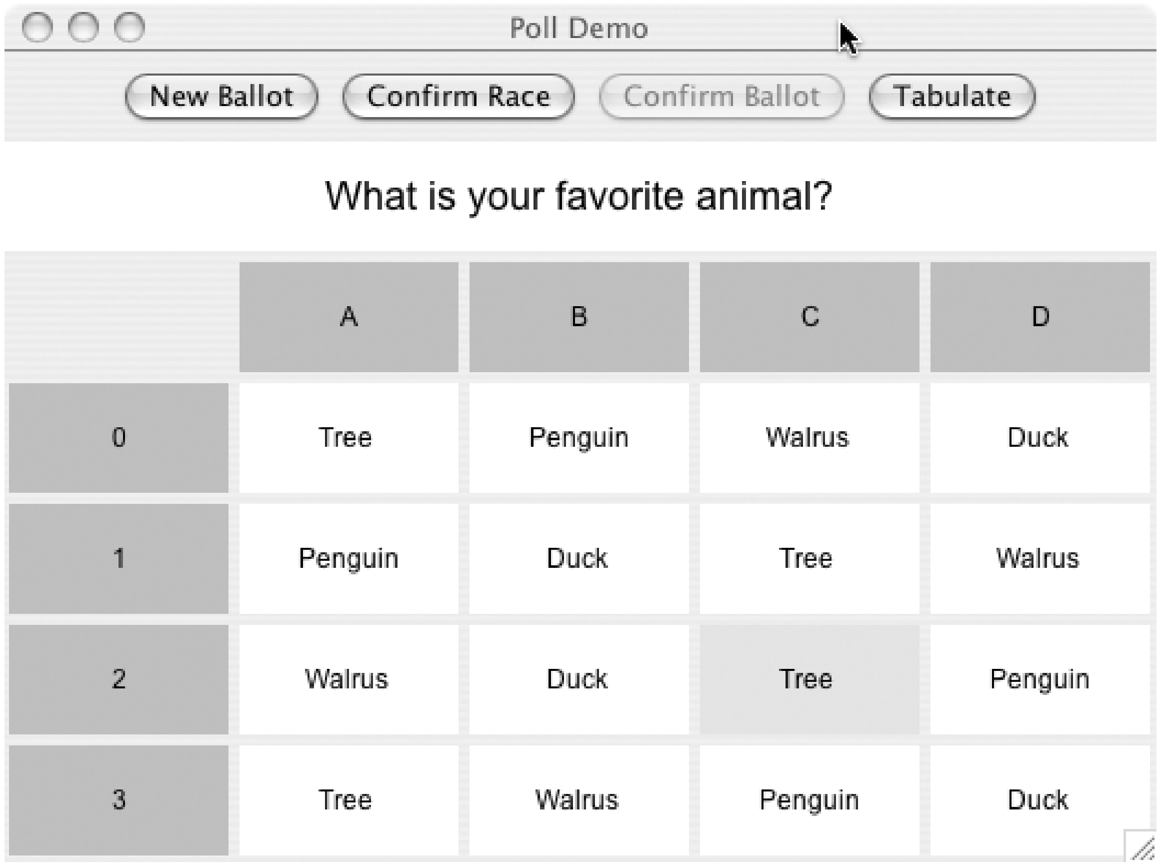
The `Tabulator` class has two tasks: accumulating the ballots, and decrypting and calculating the final tally. To accumulate ballots, it takes in a list of `BallotTally` objects and extracts out the commitments for the actual vote for each race, accumulating the running products. After the accumulation is completed, the `Tabulator` can either return the encrypted tally or determine the actual tallies. The `getTally` method of `Tabulator` requires the client to provide the private key. The decryption and tabulation happens much the way it was previously described.

4.4.4 Virtual voting booth

The "voting booth" is a window with four buttons, a question and a grid of possible answers as in Figure 4-2. Note that the rows and columns are labeled with numbers and letters. The buttons give the user the option of starting a new ballot at all times. The other buttons are greyed out when not available. For this particular example, the voter is asked "What is your favorite animal?" and presented with the following choices: Duck, Penguin, Walrus, Tree [1]. The election pictured in 4-2 has a security factor of $d = 4$, i.e. there are 4 rows.

When a new ballot is started, "Start Receipt" is printed, followed by the name of the ballot and the ballot ID number and then a dashed line. The commitments are printed next, in hashed form. Figure 4-3 shows a voter's receipt for the election from Figure 4-2. Section A of the receipt in the figure shows the "Start Receipt" and

[1] Note that one of these things is not like the others.



What is your favorite animal?

	A	B	C	D
0	Tree	Penguin	Walrus	Duck
1	Penguin	Duck	Tree	Walrus
2	Walrus	Duck	Tree	Penguin
3	Tree	Walrus	Penguin	Duck

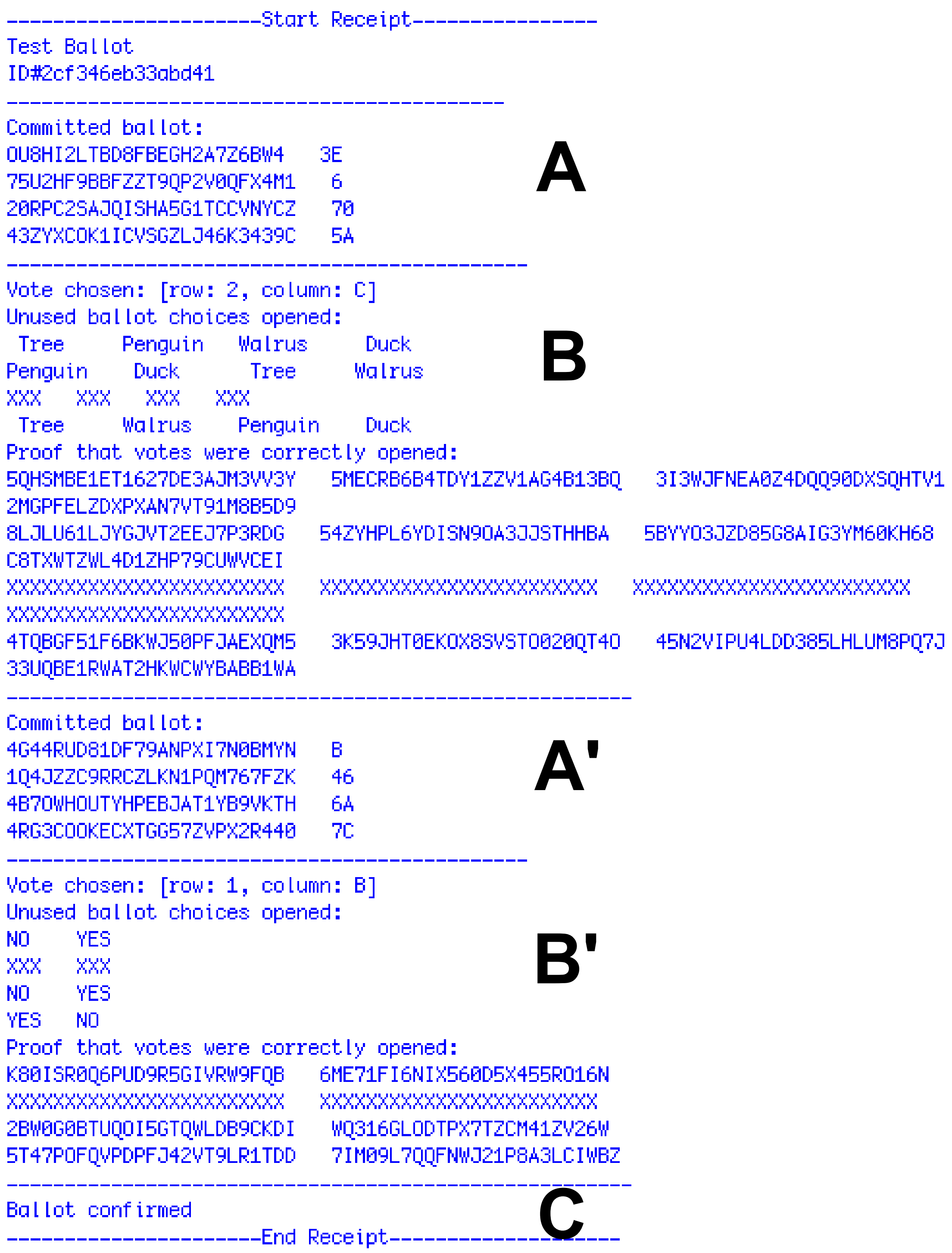
Figure 4-2: A view of the poll booth window, after a vote is selected.

commitments, corresponding to Receipt Part A in Figure 4-1. With 1024 bit keys, each unhashed commitment was 100 hexadecimal characters long. With hashing, each row of commitments could be represented in around 25 characters, plus check digits to provide error-detection when scanning the receipts.

At this point, the voter chooses to vote for the candidate in row r, column c. On the screen, the color of the element in the grid that is selected changes to teal. In Figure 4-2, this is row 2, column C which represents a vote for "Tree." At the same time, the printer records the voter's choice and prints a grid of the candidates, matching the grid still displayed on the screen, with "XXXXX" in place of the elements in row r. In Figure 4-3 this is section B, corresponding to Receipt Part B in Figure 4-1. The reveal information is quite large; for a 1024-bit key, each proof is 100 characters long, and cannot be hashed since it is to be used for verification. The election that produced the receipt in Figure 4-3 used a 64-bit key.

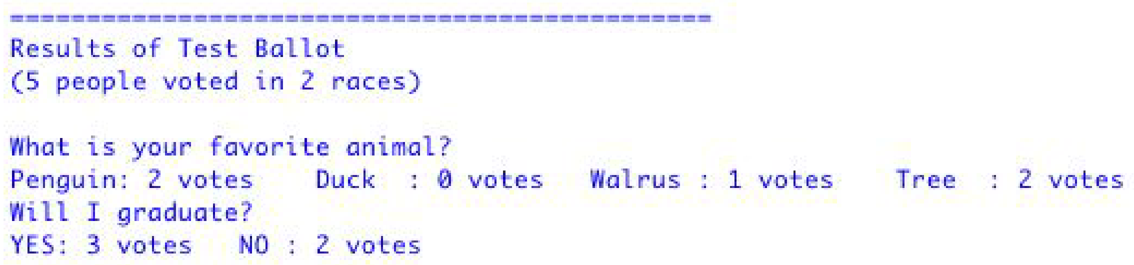
After the vote is entered, the program waits for the voter to either confirm the ballot or restart. The voter should check that the grid on the screen and the grid printed out are the same, and she should then select the Confirm Race button. This will save the entered vote and move on to the next race. In Figure 4-3, section A' shows the commitments for the next race and section B' shows the reveal information.

After the voter has selected and confirmed a candidate for each race, she can click Confirm Ballot and "Ballot confirmed" will be printed, followed by a line that indicates the end of the receipt. This is section C in Figure 4-3, corresponding to Receipt Part C in Figure 4-1. The machine then restarts the whole process with a fresh ballot. Once one ballot has been confirmed, the Tabulate button becomes available. If the Tabulate button is selected, a summary of the results of the election will be printed to the command line. Figure 4-4 shows the result of the election described in 4-2.



```
----------------------Start Receipt----------------
Test Ballot
ID#2cf346eb33abd41
-------------------------------------------
Committed ballot:
OU8HI2LTBD8FBEGH2A7Z6BW4   3E
75U2HF9BBFZZT9QP2V0QFX4M1   6
20RPC2SAJQISHA5G1TCCVNYCZ   70
43ZYXCOK1ICVSGZLJ46K3439C   5A
---------------------------------------------
Vote chosen: [row: 2, column: C]
Unused ballot choices opened:
 Tree     Penguin   Walrus     Duck
Penguin    Duck      Tree     Walrus
XXX   XXX   XXX   XXX
 Tree     Walrus    Penguin    Duck
Proof that votes were correctly opened:
5QHSMBE1ET1627DE3AJM3VV3Y   5MECRB6B4TDY1ZZV1AG4B13BQ   3I3WJFNEA0Z4DQQ90DXSQHTV1
2MGPFELZDXPXAN7VT91M8B5D9
8LJLU61LJYGJVT2EEJ7P3RDG   54ZYHPL6YDISN9OA3JJSTHHBA   5BYYO3JZD85G8AIG3YM60KH68
C8TXWTZWL4D1ZHP79CUWVCEI
XXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
4TQBGF51F6BKWJ50PFJAEXQM5   3K59JHT0EKOX8SVSTO020QT4O   45N2VIPU4LDD385LHLUM8PQ7J
33UQBE1RWAT2HKWCWYBABB1WA
------------------------------------------------------
Committed ballot:
4G44RUD81DF79ANPXI7N0BMYN   B
1Q4JZZC9RRCZLKN1PQM767FZK   46
4B7OWHOUTYHPEBJAT1YB9VKTH   6A
4RG3COOKECXTGG57ZVPX2R440   7C
---------------------------------------------
Vote chosen: [row: 1, column: B]
Unused ballot choices opened:
NO    YES
XXX   XXX
NO    YES
YES   NO
Proof that votes were correctly opened:
K80ISR0Q6PUD9R5GIVRW9FQB   6ME71FI6NIX560D5X455RO16N
XXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXX
2BW0G0BTUQOI5GTQWLDB9CKDI   WQ316GLODTPX7TZCM41ZV26W
5T47POFQVPDPFJ42VT9LR1TDD   7IM09L7QQFNWJ21P8A3LCIWBZ
------------------------------------------------------
Ballot confirmed
----------------------End Receipt--------------------
```

Figure 4-3: The receipt presented to the voter after completing a two-race election.



```
===================================================
Results of Test Ballot
(5 people voted in 2 races)

What is your favorite animal?
Penguin: 2 votes    Duck   : 0 votes    Walrus : 1 votes    Tree   : 2 votes
Will I graduate?
YES: 3 votes    NO : 2 votes
```

Figure 4-4: The receipt presented after tabulating a two race election.

Chapter 5

Conclusions

The previous chapter proposed a new voting scheme which adds voter verification to existing homomorphic voting systems, and presented an initial partial implementation of the scheme. The scheme achieves cryptographic voter verification, despite human limitations, for homomorphic voting systems. The proposal could be further adapted to take advantage of other extensions of the traditional homomorphic voting scheme. The implementation provides an opportunity to evaluate the feasibility and usability of the proposal.

The implementation served to uncover the drawbacks of adding voter verification to a homomorphic election system. The biggest problem is the tradeoff between usability and security. Voters will be confused if they are presented with multiple ways of doing the same task, and the randomness of a candidate's location in the grid does not lessen this confusion. Adding more rows to the grid reduces the possibility that the voting machine will cheat undetectably, but reduces the user-friendliness of the ballot. It also makes the receipt longer and more difficult to verify.

Another issue arises because all proposed cryptographic systems that provide voter verification are designed so that the receipts can be used to prove whether a voting machine cheats, yet the receipts themselves are difficult to authenticate. In the proposed scheme, the last step is for the voting machine to sign the receipt. But inside the voting booth, the voter has no way of knowing if the signature is properly formed. The voter must wait until she exits the booth to verify the correctness of the signature. If she determines that the signature is not valid, she can approach an election official to complain. However, this creates a new problem for the election official: how is it possible to determine whether an improperly formed receipt was created by a voting machine? Possible solutions include using special paper or watermarks, which would raise the costs of the system, or retiring machines that receive complaints for the remainder of the election.

A third issue that needs to be addressed in this scheme is where the randomness comes from. All of the cryptography requires a random seed. In practice, a random seed is chosen, and then a pseudorandom generator produces the randomness used for permutations and encryptions. If the voting machine uses a seed provided by an outside party, the outside party could potentially break some of the privacy of the voters that use that machine. Asking the voter to provide the randomness would allow the voter to sell their vote.

The same issues can be found in the schemes proposed by Chaum and Neff, including the tradeoff between usability and security. This issue can be addressed. For sufficiently large elections, using only two rows should be sufficient. Improving the overall user-friendliness of the design can also alleviate the problem. Hashing the commitments instead of printing them in their entirety helps reduce the length of the receipts. While the reveal information cannot be hashed, it could be printed as a barcode since it can only be verified by computers.

The scheme presented here achieves the goal of a secure and private homomorphic voter-verified election scheme. While it has flaws, they are the same problems that appear in all voter-verified systems.

5.1 Future work

There are several potential areas for improvement in the design and the implementation of this scheme. One major issue is the graphical user interface. Currently, it is very basic. A voting machine should allow voters to skip races or return to previous questions and alter a selection; the implementation does not currently do so. Finally, instead of printing to paper, the implementation presented in this thesis prints to the command line. This is not an accurate simulation, as it allows words to wrap in the terminal window and thereby hides the true size of the receipt. Improving the interface would allow us to determine whether this system is truly feasible.

Another issue is the problem of matching the printed grid with the grid on the screen. If a machine was malicious, it could display a grid different from the one it commits to cryptographically on the receipt. After the voter selects her cell in the grid, the machine could alter the grid on the screen to display the values that correspond to the printed commitments. If the voter does not pay careful attention to the grid on the screen, she might miss this deception. Even if she does notice, her only option is to start over with a new ballot.

A potential solution to this problem would be to color-code the grid, so that a change would be more noticeable. If each candidate assigned a color, altering the arrangement of candidates would alter the pattern of colors in the grid. Such changes are more noticeable to the human eye. Additionally, each voting machine could be assigned a fixed color pattern and the colors randomly assigned to candidates for each voter. A reference color pattern could be kept in the voting booth and would prevent the machine from altering the pattern.

One major deficiency of the proposed voting system is its inability to handle write-in votes. Mixnet's ability to handle such elections is a clear advantage. Kiayias and Yung designed a system that supplemented homomorphic encryption of the ballots with a mixnet for write-in votes [16]. Applying voter verification to that scheme would be tremendously useful.

Bibliography

[1] Hyon B. Shin Amie Jamieson and Jennifer Day. Voting and registration in the election of November 2000, February 2002. `http://www.census.gov/population/www/socdemo/voting.html`.

[2] J. Bannet, D. Price, A. Rudys, J. Singer, and D. Wallach. Hack-a-vote: Security issues with electronic voting systems. *IEEE Security and Privacy*, 2(1):32–37, January 2004. `http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/sp/&toc=comp/mags/sp/2004/01/j1toc.xml`.

[3] O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system, 2001. `http://citeseer.ist.psu.edu/baudron01practical.html`.

[4] Gilles Brassard, David Chaum, and Claude Crepeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988. `http://dx.doi.org/10.1016/0022-0000(88)90005-0`.

[5] Jeremy Bryans and Peter Ryan. A dependability analysis of the Chaum digital voting scheme. Technical report, University of Newcastle upon Tyne, July 2003. `http://eprints.ncl.ac.uk/deposit_details.php?deposit_id=208`.

[6] Jeremy Bryans and Peter Ryan. A simplified version of the Chaum voting scheme. Technical report, University of Newcastle upon Tyne, May 2004. `http://www.cs.ncl.ac.uk/research/pubs/trs/abstract.php?number=843`.

[7] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January 2004. http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/sp/&toc=comp/mags/sp/2004/01/j1toc.xml.

[8] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981. http://doi.acm.org/10.1145/358549.358563.

[9] Election Assistance Commision. Voluntary voting system guidelines, 2004. http://www.glynn.com/eac_vvsg.

[10] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997. http://citeseer.csail.mit.edu/cramer97secure.html.

[11] I. Damgard, M. Jurik, and J. Nielsen. A generalization of Paillier's public-key system with applications to electronic voting, 2003. http://citeseer.ist.psu.edu/damgard03generalization.html.

[12] California Ad Hoc Touch Screen Task Force. Secretary of State's Ad hoc Touch Screen Task Force Report, July 2003. http://www.ss.ca.gov/elections/taskforce_report.htm.

[13] Dimitris Grizalis, editor. *Advances in Information Security: Secure Electronic Voting*. Springer, December 2002.

[14] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking, 2002. http://theory.lcs.mit.edu/~rivest/voting/papers/JakobssonJuelsRivest-MakingMixNetsRobustForElectronicVotingByRandomizedPartialChecking.pdf.

[15] Douglas W. Jones. Strengths and weaknesses of voting systems, March 2004. http://www.cs.uiowa.edu/~jones/voting/panama.html.

[16] A. Kiayias and M. Yung. The vector-ballot e-voting approach, 2004. http://citeseer.csail.mit.edu/kiayias04vectorballot.html.

[17] L.A. County Statistical Profile, 2004. http://lacounty.info/statistical_information.htm.

[18] Rebecca Mercuri. A better ballot box. *IEEE Spectrum*, October 2002. http://www.spectrum.ieee.org/WEBONLY/publicfeature/oct02/evote.html.

[19] C. Andrew Neff. Practical high certainty intent verification for encrypted votes, October 2004. http://votehere.net/vhti/documentation/vsv-2.0.3638.pdf.

[20] C. Andrew Neff. Verifiable mixing (shuffling) of ElGamal pairs, April 2004. http://votehere.net/vhti/documentation/egshuf-2.0.3638.pdf.

[21] NIST. National Software Resource Library: Voting software reference data set, May 2005. http://www.nsrl.nist.gov/votedata.html.

[22] American Association of People with Disabilities. Aapd's executive summary on dre's. Distributed at Electronic Voting in Massachusetts conference February 2004, 2004. http://www.evote-mass.org/.

[23] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes, 1999. http://www.gemplus.com/smart/rd/publications/ps/Pai99pai.ps.

[24] Caltech/MIT Voting Technology Project. What is, what could be. Technical report, California Institute of Technology, July 2001. http://www.vote.caltech.edu/reports/2001report.

[25] Election Reform Information Project. Securing the vote, April 2004. http://electionline.org/Portals/1/Publications/Securing_the_Vote.pdf.

[26] M. O. Rabin. Digitalized signatures. In *Foundations of Secure Computation*, pages 155–166, New York, NY, 1977. Academic Press.

[27] Ted Selker. Testimony on voter verification, June 2005. http://www.vote.caltech.edu/media/documents/wps/vtp_wp31.pdf.

[28] Michael Ian Shamos. Paper v. electronic voting records – an assessment, 2004. http://euro.ecom.cmu.edu/people/faculty/mshamos/paper.htm.

[29] Michael Shnayerson. Hack the vote. *Vanity Fair*, April 2004.

[30] A. Rubin T. Kohno, A. Stubblefield and D. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy 2004*. IEEE Computer Society Press, 2004. http://avirubin.com/vote/analysis/index.html.

[31] Kim Zetter. Suspect code used in state votes. *Wired Magazine*, November 2003. http://www.wired.com/news/evote/0,2645,61092,00.html.