EFFICIENT SECURE E-VOTING AND ITS APPLICATION IN CYBERSECURITY EDUCATION

by

Nathan Swearingen

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Science

Indianapolis, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Xukai Zou, Chair

Department of Computer and Information Science

Dr. Feng Li

Department of Computer and Information Technology

Dr. Qin Hu

Department of Computer and Information Science

Approved by:

Dr. Shiaofen Fang

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Xukai Zou, for his assistance and support through my research, my graduate studies, and even during some of my time as an undergraduate. In addition to teaching me many of the fundamental concepts of cryptography that helped to develop my knowledge and understanding, Dr. Zou has provided me with many research opportunities that I would not have had otherwise.

I would also like to thank the other members of my committee, Dr. Feng Li and Dr. Qin Hu. I appreciate their time in reviewing my research and helping me earn my degree.

Finally, I would like to thank my parents for their support throughout my time at IUPUI. It is because of their help that I have been able to focus on my academic studies and succeed in my coursework.

This work is funded by NSF DGE-2011117 and NSF OAC/CICI-1839746.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

M	the number of candidates in the election
N	the number of voters in the election
L	the length of a voting vector ($L = MN$)
$\mathcal{C}_1$	collector 1
$\mathcal{C}_2$	collector 2
$\boldsymbol{\pi}_1$	$\mathcal{C}_1$'s random permutation
$\boldsymbol{\pi}_2$	$\mathcal{C}_2$'s random permutation
l_{i}	voter i's 0-indexed location
c_{i}	voter i's vote
$v_{\mathrm{i}}, v'_{\mathrm{i}}$	voter i's vectors
$p_{\mathrm{i}}, p'_{\mathrm{i}}$	voter i's ballots
$x_{\mathrm{j,i}}, x'_{\mathrm{j,i}}$	$\mathcal{C}_{\mathrm{j}}$'s shares for voter i
$\sigma_{\mathrm{j}}, \sigma'_{\mathrm{j}}$	sum of $\mathcal{C}_{\mathrm{j}}$'s corresponding shares
A	a large safe prime
g	a generator of $G_{(A-1)/2}$
h	another generator of $G_{(A-1)/2}$
n	the modulus of a Paillier public key
X	the agreed share bound

ABSTRACT

As the need for large elections increases and computer networking becomes more widely used, e-voting has become a major topic of interest in the field of cryptography. However, lack of cryptography knowledge among the general public is one obstacle to widespread deployment. In this paper, we present an e-voting scheme based on an existing scheme. Our scheme features an efficient location anonymization technique built on homomorphic encryption. This technique does not require any participation from the voter other than receiving and summing location shares. Moreover, our scheme is simplified and offers more protection against misbehaving parties. We also give an in-depth security analysis, present performance results, compare our scheme with existing schemes, and describe how our research can be used to enhance cybersecurity education.

1. INTRODUCTION

Voting is a valuable technique in making decisions that affect multiple people. Not only is it used in many organizations to allow its members to be involved in the decision-making process, it is also used in democratic governments for functions such as electing leaders or passing laws.

In theory, voting provides a fair way for each participant to have a say in the result of an election. In practice, the effectiveness of voting is inhibited by many difficulties that can impact the election. Voters should be able to ensure that their vote is included in the final result, but should also be assured that no one, not even an election official, is able to learn their vote. Anyone should be able to verify the result of the election themselves, rather than blindly trust a published result. In some cases, the election must be resistant to coercion, meaning that a voter cannot prove how they voted.

Paper elections can partly achieve some of the desired properties of elections. For example, an election that is conducted openly and monitored closely may be difficult to tamper with. However, such processes can be expensive, and ultimately the process is less effective for large-scale elections. Alternatively, electronic voting can be much more scalable, and many security properties can be achieved using cryptography rather than physical monitoring. As a result, e-voting has the potential to allow for secure elections that are much larger and performed much faster than traditional paper elections.

While the field of cryptography provides many tools that can be used to develop a secure voting scheme, the properties of secure voting are particularly difficult to achieve because they tend to be contradictory [20]. The privacy of individual votes must be protected, but the aggregation of the votes must be public to everyone. A voter must be assured that their vote is counted correctly, but they should not be able to prove to others that their vote is counted a certain way. For this reason, e-voting schemes must be designed carefully to resist various attacks, and often must make tradeoffs that favor certain properties over others.

In this work, we present a simplified and more secure derivative of an existing scheme [20]. Both schemes use a voting vector to allow voters to view their votes as plaintext. Ours improves upon the existing scheme in multiple ways. First, it features an efficient location

anonymization technique based on homomorphic encryption that requires very little participation from voters. Second, it improves and elaborates on the secret sharing mechanism that protects the privacy of votes. Third, it provides more protection against malicious behavior by a collector. We also present a detailed security analysis of the new system, performance test results, and a comparison with other schemes from the literature. Finally, we describe how our scheme can be applied to cybersecurity education to improve students' understanding of cryptography concepts.

2. RELATED WORK

Many e-voting schemes currently exist in the literature. Different schemes are often built upon different techniques and try to achieve particular properties. For example, some attempt to achieve receipt-freeness, which means that there is no way for a voter to get a receipt that proves how they voted [5]. Benaloh and Tuinstra propose such a scheme that makes use of a private voting booth to prevent coercion [2]. It also makes use of homomorphic encryption. Aditya et al. give a receipt-free scheme that uses mixnets [1]. Mixnets are also used by Clarkson et al., who present yet another receipt-free scheme [4]. Further research into coercion-resistant e-voting includes work by Spycher et al. [18] as well as by Howlader et al. [8].

Some schemes use polynomial-based secret sharing to allow for vote tallying. Nair et al. [12] propose a scheme that involves converting a vote to a bitwise representation and then applying secret sharing to the resulting value. Liu and Zhao [11] place vote information on polynomial coefficients.

Liu and Wang describe an e-voting scheme that uses blind signature and blockchain [10]. Blind signatures are used to authorize eligible voters, and the blockchain ensures that everyone has a consistent view of the election.

One more technique that we will consider is zero-knowledge proofs. These are used in DRE-i, proposed by Hao et al. [7], as well as in DRE-ip, proposed by Shahandashti and Hao [17]. A variant of zero-knowledge proofs is also used in Koinonia, a system proposed by Ge et al. [6].

Our scheme is based on the non-interactive scheme described by Zou et al. in [20] in 2017, a later version of the scheme proposed in [19] in 2014. Compared to the scheme in [20], this scheme is simpler and more efficient, and provides more protection against election tampering.

3. PRELIMINARIES

Before we discuss our scheme, we explain some important concepts that will be used, namely, the Pedersen commitment [15], the Paillier cryptosystem [13], secure two-party multiplication [16], and secret sharing.

3.1 Pedersen Commitment

The Pedersen commitment [15] provides a means by which one party can prove various properties of secret values without revealing any other information about the values. The technique uses a large prime p and another large prime q that divides $p-1$. It also requires two generators, g and h, of subgroup G_q of $\mathbb{Z}_p^*$. It is important that no party knows $\log_g h$.

A commitment of integer $0 \leq s < q$ can computed as $g^s h^t \pmod{p}$, where t is a random integer $0 \leq t < q$. When s is revealed, t must also be revealed for the commitment to be verified. In our scheme, $p = A$ is a safe prime and $q = (A-1)/2$.

Our scheme uses Pedersen commitments so that collectors can prove that the sums of their shares are a particular value. Such a proof is slightly simplified by also generating the t values such that they sum to zero $\pmod{(A-1)/2}$. This allows anyone with access to the published commitments to compute their product and verify the result. s and t for an individual commitment are only revealed to the voter for which s, the share, is intended.

3.2 Paillier Cryptosystem

The Paillier cryptosystem [13] [14] is a homomorphic cryptosystem that our scheme uses for secure two-party multiplication (discussed below) and location anonymization. We will not discuss the details here, but we describe the basics, as well as the notation we use, below.

Key generation requires two primes, p and q, whose product $n = pq$ is made public. Plaintexts are integers from 0 to $n-1$ (inclusive). Encryption of x will be notated as $E(x)$, and decryption of x will be notated as $D(x)$. Note that since one plaintext can encrypt to multiple ciphertexts, $E(x)$ may sometimes be used to indicate any possible encryption of x.

However, if the scheme calls for $E(x)$ to be produced by one of the parties involved in the election, it must be generated randomly.

Given plaintexts x and y, $E(x)E(y) \pmod{n^2}$ produces a valid encryption of $x+y \pmod{n}$. This property allows parties to perform operations on encrypted values, which is useful in distributing trust between two collectors.

3.3 Secure two-party Multiplication

Secure two-party multiplication, or STPM, allows two parties to multiply secrets without revealing their secrets to each other [16] [20]. Specifically, if A has x_1 and B has x_2, A and B can use the Paillier cryptosystem to compute s_1 and s_2 respectively such that $s_1 + s_2 = x_1 x_2 \pmod{n}$.

Assume A initializes a Paillier cryptosystem and gives B the public key. First, A computes $E(x_1)$ and sends the result to B. B computes $E(x_1)^{x_2} E(s_2)^{-1}$ (or $E(x_1)^{x_2} E(n - s_2)$) for random integer $0 \leq s_2 < n$ and sends the result to A. Finally, A decrypts the received value to obtain $s_1 = D(E(x_1)^{x_2} E(s_2)^{-1}) = x_1 x_2 - s_2 \pmod{n}$. A ends up with s_1 and B ends up with s_2 such that $s_1 + s_2 = x_1 x_2 \pmod{n}$, but just as before, A does not know x_2, and B does not know x_1.

3.4 Secret Sharing

Secret sharing allows one value to be split into multiple "shares" such that no share on its own reveals a significant amount of information on the original value. One common way this is achieved is with modular arithmetic. For example, if we have some integer $0 \leq x < n$ for some upper bound $n \in \mathbb{Z}$, we could split x into three shares by randomly generating integers $0 \leq x_1, x_2 < n$ and then computing $x_3 = x - x_1 - x_2 \pmod{n}$. Note that x_1, x_2, and x_3 appear random on their own and reveal no information on which of the n possible values x is, but that $x_1 + x_2 + x_3 = x \pmod{n}$.

While the method above is simple and results in no information leakage, using it for our scheme, as well as for the scheme ours is based on [19] [20], results in a security flaw, which is discussed in more detail later. For this reason, we choose a variation on the above

method that does not use modular arithmetic. Shares are taken from a uniform distribution, but when they are used by voters, the voters do not apply any modular operations. As we discuss the details of our e-voting scheme, we will describe more precisely how these shares are generated and used.

4. LOCATION ANONYMIZATION

Before discussing the voting process, it is important to understand the topic of location anonymization. One feature that this voting scheme provides, as well as the system this scheme is based on [19] [20], is the use of a voting vector. This vector is more effective than aggregate tallies on their own because in addition to providing enough information to determine the final tally, it allows voters to check their location on the vector and ensure that their vote is correct.

However, with this advantage in verifiability comes the necessity to ensure that locations cannot be associated with individual voters, by other voters, outsiders, or even election officials. This conflict necessitates a protocol to distribute unique locations among voters such that only a voter knows their location. Other concepts in cryptography can be adapted for this purpose, such as shared secret shuffle [3]. The technique presented below is based on STPM, and a similar technique is also described in [3], although it is not the focus of that paper. For the e-voting scheme in particular, the two collectors should be the ones who execute the protocol.

4.1 Basic Technique

Initially, just as with STPM, one collector needs a Paillier key pair, and the other collector needs to know the corresponding public key. Later when the collectors need to perform STPM, they can use the same key pair. We will refer to the modulus of the Paillier cryptosystem (the product of the two primes) as n. For simplicity, we will assume that $\mathcal{C}_1$ has the private key to the cryptosystem and $\mathcal{C}_2$ only has the public key.

The basic idea behind the technique described here is as follows:

1. $\mathcal{C}_1$ shuffles a list of locations, encrypts them, and passes them to $\mathcal{C}_2$
2. $\mathcal{C}_2$ re-shuffles the locations, applies secret sharing, and then passes encrypted shares back to $\mathcal{C}_1$
3. $\mathcal{C}_1$ decrypts the received shares

4. Both collectors pass their shares to the voters, who sum received shares to obtain their location

4.2 Details

We denote $\mathcal{C}_1$'s permutation as $\boldsymbol{\pi}_1$ and $\mathcal{C}_2$'s as $\boldsymbol{\pi}_2$, such that $l_{\mathrm{i}} = \boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i}-1))$. Below we detail the process that is used to allow voters to compute their l_{i} values.

To perform location anonymization for N voters, $\mathcal{C}_1$ randomly permutes a location vector $\langle 0, 1, \ldots, N-1\rangle$ into $\langle \boldsymbol{\pi}_1(0), \boldsymbol{\pi}_1(1), \ldots, \boldsymbol{\pi}_1(N-1)\rangle$. They encrypt the permuted vector with their Paillier key. We denote the encrypted and permuted vector as $\langle y_0, y_1, \ldots, y_{N-1}\rangle$ where $y_{\mathrm{i}} = E(\boldsymbol{\pi}_1(\mathrm{i}))$ for $0 \leq \mathrm{i} < N$. This vector is sent to $\mathcal{C}_2$.

As stated above, $\mathcal{C}_2$ re-shuffles the values received. $\mathcal{C}_2$ can perform secret sharing on the encrypted values by taking advantage of the homomorphic property of the Paillier cryptosystem. First, they generate N random integers $0 \leq r_{2,\mathrm{i}} < n$ for $0 \leq \mathrm{i} < N$. $\mathcal{C}_2$ then computes $\langle z_0, z_1, \ldots, z_{N-1}\rangle$ where $z_{\mathrm{i}} = y_{\boldsymbol{\pi}_2(\mathrm{i})} E(n - r_{2,\mathrm{i}}) \pmod{n^2}$ for $0 \leq \mathrm{i} < N$, which they send back to $\mathcal{C}_1$.

Once $\mathcal{C}_1$ receives these values, they decrypt them into $\langle r_{1,0}, r_{1,1}, \ldots, r_{1,N-1}\rangle$. Now note that voter i's location can be computed as $l_{\mathrm{i}} = r_{1,\mathrm{i}-1} + r_{2,\mathrm{i}-1} \mod n$. However, an additional simplification is possible.

The collectors compute the following ($1 \leq \mathrm{i} \leq N$, $\mathrm{j} = 1, 2$):

$$\begin{cases} l_{\mathrm{j,i}} = r_{\mathrm{j,i}-1} \mod N & r_{\mathrm{j,i}-1} < n/2 \\ l_{\mathrm{j,i}} = r_{\mathrm{j,i}-1} - n \mod N & r_{\mathrm{j,i}-1} \geq n/2 \end{cases}$$

These values are given to the voters, who compute their locations as $l_{\mathrm{i}} = l_{1,\mathrm{i}} + l_{2,\mathrm{i}} \mod N$ for $1 \leq \mathrm{i} \leq N$.

The simplification above can be omitted, but it allows voters to compute their location without knowing n and without large $r_{\mathrm{j,i}}$ values being sent to voters.

4.3 Correctness

We will first prove that $r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n = \boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i}))$ for $0 \leq \mathrm{i} < N$. After collector 2 repermutes and masks the received values, we have $z_\mathrm{i} = y_{\boldsymbol{\pi}_2(\mathrm{i})} E(n - r_{2,\mathrm{i}}) = E(\boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})))E(n - r_{2,\mathrm{i}}) = E(\boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})) - r_{2,\mathrm{i}} \mod n)$ due to the homomorphic property of the Paillier cryptosystem. This leads to the following:

$$
\begin{aligned}
& r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n \\
&= D(z_\mathrm{i}) + r_{2,\mathrm{i}} \mod n \\
&= D(E(\boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})) - r_{2,\mathrm{i}} \mod n)) + r_{2,\mathrm{i}} \mod n \\
&= (\boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})) - r_{2,\mathrm{i}} \mod n) + r_{2,\mathrm{i}} \mod n \\
&= \boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})) - r_{2,\mathrm{i}} + r_{2,\mathrm{i}} \mod n \\
&= \boldsymbol{\pi}_1(\boldsymbol{\pi}_2(\mathrm{i})) \\
&= l_{\mathrm{i}+1}
\end{aligned}
$$

To show the correctness of the simplification, we can show that $l_{1,\mathrm{i}} + l_{2,\mathrm{i}} \mod N$ is equal to the correct location l_i. Note that both $r_{1,\mathrm{i}}$ and $r_{2,\mathrm{i}}$ are uniform random in $\{0, 1, \ldots, n-1\}$ for $0 \leq \mathrm{i} < N$ (although not independently). Therefore, we can assume that $r_{1,\mathrm{i}} \notin \{0, 1, \ldots, N-1\} \cup \{(n+1)/2, (n+1)/2+1, \ldots, (n+1)/2+N-1\}$ (note that n is odd), as the probability that this is true is $1 - 2N/n \approx 1$ as long as $N << n$. If $r_{1,\mathrm{i}}, r_{2,\mathrm{i}} < n/2$, then $r_{1,\mathrm{i}} + r_{2,\mathrm{i}} < n$, so $l_{\mathrm{i}+1} = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \geq N + r_{2,\mathrm{i}} \geq N$, which is impossible because $l_{\mathrm{i}+1} < N$. If $r_{1,\mathrm{i}}, r_{2,\mathrm{i}} \geq n/2$, then $n \leq r_{1,\mathrm{i}} + r_{2,\mathrm{i}} < 2n$, so $l_{\mathrm{i}+1} = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} - n = (r_{1,\mathrm{i}} - n/2) + (r_{2,\mathrm{i}} - n/2) \geq N + (r_{2,\mathrm{i}} - n/2) \geq N$, which is again impossible. So one of $r_{1,\mathrm{i}}, r_{2,\mathrm{i}}$ is less than $n/2$ and the other is greater than or equal to $n/2$, meaning that $l_{1,\mathrm{i}} + l_{2,\mathrm{i}} = r_{1,\mathrm{i}-1} + r_{2,\mathrm{i}-1} - n \pmod N$.

If $r_{1,\mathrm{i}} + r_{2,\mathrm{i}} < n$, then $l_{\mathrm{i}+1} = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \geq n/2 > N$ since one of them is greater than or equal to $n/2$. This is impossible since $l_{\mathrm{i}+1} < N$, so $n \leq r_{1,\mathrm{i}} + r_{2,\mathrm{i}} < 2n$. Therefore, $l_{\mathrm{i}+1} = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} \mod n = r_{1,\mathrm{i}} + r_{2,\mathrm{i}} - n$. Finally, we have:

$$l_{1,\mathrm{i}} + l_{2,\mathrm{i}} \mod N$$

$$= r_{1,\mathrm{i}-1} + r_{2,\mathrm{i}-1} - n \mod N$$

$$= l_{\mathrm{i}} \mod N$$

$$= l_{\mathrm{i}}$$

Hence, the location anonymization procedure above allows voters to compute their correct locations with very high probability.

5. OUR SCHEME

We now present the remaining details of our e-voting scheme.

5.1 Election Structure

An election consists of N voters ($N \geq 3$) and M candidates ($M \geq 2$). $L = MN$ is the length of a voting vector. For simplicity, we will assume that each voter votes for exactly one candidate, although other rules are possible.

The election also requires two mutually restraining collectors, $\mathcal{C}_1$ and $\mathcal{C}_2$. The two collectors would need to collude to perform certain attacks. They should be chosen such that they are trusted not to collude. One possible choice in the United States would be the two political parties [19] [20].

All voters and collectors must agree on large safe prime A, generators g and h of subgroup $G_{(A-1)/2}$ of $\mathbb{Z}_A^*$ with unknown $\log_g h$, and share bound X. X should be at least 2^L. In fact, it should be much larger (say, 2^{64} times larger) to allow shares to protect votes. A should be at least $2NX$. The Paillier system used by the collectors should be such that $n \geq 18X^2$. Keep in mind that the bounds we give for our scheme are chosen to be simple and are not always as tight as possible.

Our scheme is depicted in Figure 5.1. The colors distinguish who generates or computes each value. Anyone can compute the final voting vector.

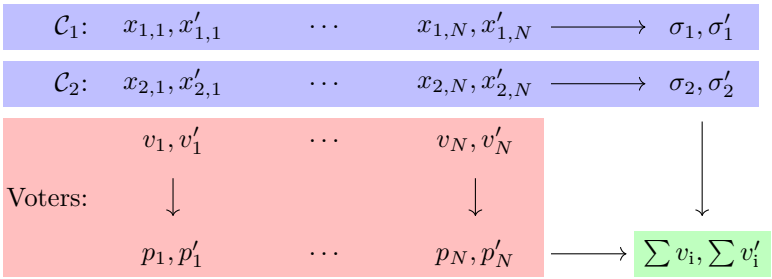

Figure 5.1. Visualization of our scheme.

5.2 Initialization

Before voting, voters must obtain unique and anonymous locations. The location anonymization process described in the previous chapter is performed by the two collectors. They then distribute location information to the voters so that each voter can obtain their location. A voter does not need to be given location data until they are ready to vote.

5.3 Generating Shares

In order for voters to create their ballots, the collectors need to generate shares for the voters to protect the confidentiality of their vote. Each collector $\mathcal{C}_\mathrm{j}$ needs to generate N independent uniform random integers $0 \leq x_{\mathrm{j,i}} < X$ $(1 \leq \mathrm{i} \leq N)$. The collector also publishes $\sigma_\mathrm{j} = \sum_{\mathrm{i}=1}^{N} x_{\mathrm{j,i}}$ so that the final voting vector can be revealed.

Each collector also needs to generate N more shares, namely $x'_{\mathrm{j,i}}$ for $1 \leq \mathrm{i} \leq N$, and publish the corresponding σ'_j. This is done using the same procedure.

Each collector distributes their generated shares to the voters. It is important that they do not allow anyone to access shares other than the voter for which they are intended. Each half of a voter's ballot will effectively be "secret shared" among the corresponding value published by the voter as well as the corresponding shares they receive from each collector.

When voters receive shares from collectors, they must verify that the shares are in the range $[0, X)$ and reject shares that are not in this range. Furthermore, anyone checking the result of the election should verify that σ_1, σ'_1, σ_2, and σ'_2 are in the range $[0, (A-1)/2)$. These requirements ensure that collectors cannot take advantage of the modular nature of Pedersen commitments such that the shares are inconsistent with σ_j and σ'_j.

To prove that $\sum_{\mathrm{i}=1}^{N} x_{\mathrm{j,i}} = \sigma_\mathrm{j}$, $\mathcal{C}_\mathrm{j}$ should publish Pedersen commitments for their $x_{\mathrm{j,i}}$ values. These commitments should be generated so that their product yields g^{σ_j}, proving that the sum of $x_{\mathrm{j,i}}$ is σ_j. When a collector gives a voter $x_{\mathrm{j,i}}$, they should also give the voter the information needed to verify that the shares match the published Pedersen commitments. Normally, this technique would only prove that the sum of the shares is σ_j modulo $(A-1)/2$. However, since voters verify that shares are within $[0, X)$, and since $A \geq 2NX$, $0 \leq \sum_{\mathrm{i}=1}^{N} x_{\mathrm{j,i}} \leq NX - 1 < (A-1)/2$. Therefore, if the product of the Pedersen commitments

is g^{σ_j} and $0 \leq \sigma_j < (A-1)/2$, then the corresponding shares sum to σ_j without a modulus. This is important in our system since ballots are not summed with a modulus.

The same technique as discussed above is used to prove that $\sum_{i=1}^{N} x'_{j,i} = \sigma'_j$.

5.4 Voting

Voter i must choose exactly one candidate to vote for. Let c_i be the index corresponding to the candidate voter i chooses, such that $1 \leq c_i \leq M$. Once voter i has their selection (c_i), location (l_i), and shares ($x_{1,i}, x_{2,i}, x'_{1,i}, x'_{2,i}$), they can compute their ballot.

The voting vector consists of L bits and is divided by location. Therefore, the bit they need to set to 1 has index $b_i = l_i M + c_i$. The voter computes $v_i = 2^{L-b_i}$ and $v'_i = 2^{b_i-1}$, and then can compute their ballot as $p_i = v_i + x_{1,i} + x_{2,i}$ and $p'_i = v'_i + x'_{1,i} + x'_{2,i}$ (note that the ballot consists of both p_i and p'_i). p_i and p'_i should be published, allowing others to verify the result of the election. The collectors verify that $0 \leq p_i, p'_i < 3X$.

5.5 Verification

Note that $v_i v'_i = 2^{L-b_i} \cdot 2^{b_i-1} = 2^{L-1}$. In fact, this is only possible if the voter votes in exactly one location. The collectors can use this property to verify that the voter has voted properly.

To perform verification, the collectors need to compute $v_i v'_i$. However, neither collector can share any value with the other that would leak significant information about the voter's vote. Since $v_i v'_i = (p_i - x_{1,i} - x_{2,i})(p'_i - x'_{1,i} - x'_{2,i}) = p_i p'_i - p_i x'_{1,i} - p_i x'_{2,i} - p'_i x_{1,i} + x_{1,i} x'_{1,i} + x_{1,i} x'_{2,i} - p'_i x_{2,i} + x'_{1,i} x_{2,i} + x_{2,i} x'_{2,i}$, we can treat each of the nine terms separately.

Since two of the terms, $x_{1,i} x'_{2,i}$ and $x'_{1,i} x_{2,i}$, require secrets from both collectors, STPM needs to be used twice. One application yields $s_1 + s'_2 = x_{1,i} x'_{2,i} \pmod{n}$, and another application yields $s'_1 + s_2 = x'_{1,i} x_{2,i} \pmod{n}$.

Next, each collector creates a sum of secret terms. They include the results from the two applications of STPM. So, collector 1 computes $S_1 = -p_i x'_{1,i} - p'_i x_{1,i} + x_{1,i} x'_{1,i} + s_1 + s'_1 \pmod{n}$ and collector 2 computes $S_2 = -p_i x'_{2,i} - p'_i x_{2,i} + x_{2,i} x'_{2,i} + s_2 + s'_2 \pmod{n}$. Each collector commits to S_j (a Pedersen commitment is not necessary). Finally, each collector

gives its sum (only after receiving the other's commitment) to the other and verifies that $p_i p'_i + S_1 + S_2 = 2^{L-1} \pmod{n}$. Collectors also share the information necessary to check the commitments, and perform these checks with the S_j values received from each other. The purpose of the commitments is discussed later.

5.6 Computing Final Voting Vector

Individual voting vectors are protected using shares provided by the collectors. However, if all p_i values or all p'_i values are summed, the final voting vector can be revealed easily:

$$
\begin{aligned}
& -\sigma_1 - \sigma_2 + \sum_{i=1}^{N} p_i \\
&= -\sigma_1 - \sigma_2 + \sum_{i=1}^{N} (v_i + x_{1,i} + x_{2,i}) \\
&= -\sigma_1 - \sigma_2 + \left(\sum_{i=1}^{N} v_i\right) + \left(\sum_{i=1}^{N} x_{1,i}\right) + \left(\sum_{i=1}^{N} x_{2,i}\right) \\
&= \sum_{i=1}^{N} v_i
\end{aligned}
$$

and similarly, $-\sigma'_1 - \sigma'_2 + \sum_{i=1}^{N} p'_i = \sum_{i=1}^{N} v'_i$.

As long as voters vote in the correct location with valid voting vectors, none of their 1-bits will overlap, and the final voting vector will anonymously reveal each voter's vote. The vectors can be derived from the sums by converting to the binary representation.

5.7 Voting Vector Validation

Once the final voting vector is available, further checks can be performed to gain confidence in its correctness. Note that summing the p'_i values and subtracting σ'_1 and σ'_2 should yield a vector that is the reverse of the vector computed by summing the p_i values and subtracting σ_1 and σ_2. Anyone can check that these two vectors are indeed reverse one another. The voting vector should have exactly one bit set in each location (contiguous sequence of

M bits). Each voter can check their location on the voting vector and see that their vote is correct.

5.8 Tallying

Once the final voting vector is checked using the checks above, the election result can be determined by summing the number of votes on the voting vector for each candidate. The final tallies for each candidate should be published, but all voters and anyone else whom the election result concerns should have access to all ballots so that they can sum them, check the resulting vector, and perform their own tally. Such transparency and reproducibility helps voters gain trust in the result of the election.

6. SECURITY ANALYSIS

Our system holds many important security properties that make it useful for real-world elections. Below we analyze the security of our scheme and its susceptibility to attacks.

6.1 Security of Location Anonymization

The security of the location anonymization used by our system depends on the security of the Paillier cryptosystem. In the first step of location anonymization, $\mathcal{C}_1$ applies a random permutation to a vector of locations, encrypts each location, and sends the resulting vector to $\mathcal{C}_2$. Because each value is encrypted and only $\mathcal{C}_1$ has the private key, $\mathcal{C}_2$ cannot learn any meaningful information from the vector. Also note that the Paillier cryptosystem is not deterministic–$\mathcal{C}_2$ cannot simply encrypt location indices and compare them to the values in the vector to learn π_1.

After $\mathcal{C}_2$ applies secret sharing, neither share of a location reveals any information about the location on its own. The vector that is sent back to $\mathcal{C}_1$ only contains random encryptions of $\mathcal{C}_1$'s shares, which reveal no information without $\mathcal{C}_2$'s shares. Nothing else is sent between $\mathcal{C}_1$ and $\mathcal{C}_2$, so neither collector gains any meaningful information about any voter's share unless that voter shares information it receives with one of the collectors.

It is still worth discussing what kinds of attacks are possible if one of the collectors does not obey the protocol. $\mathcal{C}_2$ cannot use any such attack to learn information on $\mathcal{C}_1$'s permutation, because by the time $\mathcal{C}_2$ sends any data to $\mathcal{C}_1$, it is too late to leak information of π_1 since $\mathcal{C}_1$ will not send any more data to $\mathcal{C}_2$.

There is more opportunity for an attack the other way around, where $\mathcal{C}_1$ tries to learn information about π_2 by intentionally following the protocol incorrectly. $\mathcal{C}_2$ uses π_2 only once, when re-permuting the values received from $\mathcal{C}_1$, so the attack must be performed before this. One somewhat trivial example of a possible attack by $\mathcal{C}_1$ involves sending 0 for some values on the vector. These values will remain 0 after $\mathcal{C}_2$ performs its operations. $\mathcal{C}_2$ can mitigate this particular attack by ensuring that the values received are reasonable (e.g. check that they are in $\mathbb{Z}_{n^2}^*$), and its usefulness would likely be limited anyway since $\mathcal{C}_1$ would not be able to decrypt those values after $\mathcal{C}_2$'s operations. However, $\mathcal{C}_1$ may be able to come up with

a key that violates the properties required by the Paillier cryptosystem in a way that cannot be easily detected by $\mathcal{C}_2$. More research is needed to determine whether such an attack would be possible or useful, and whether it could be prevented.

Lastly, we should consider in what ways a collector can tamper with locations. There are multiple ways this can occur, such as $\mathcal{C}_1$ initially creating invalid locations, $\mathcal{C}_2$ taking advantage of homomorphic encryption to multiply locations with constants, or either collector modifying its shares before giving them to the voters. It would likely be difficult to do much more than invalidate the election using such techniques, but more detailed analysis and defense is one possible area of future work.

Although $\mathcal{C}_2$ could send invalid values back to $\mathcal{C}_1$, this could be detected by checking that the received values are relatively prime to n. In fact, values should be validated in our scheme whenever possible to prevent these types of attacks.

6.2 Privacy of Individual Votes

To prove the privacy of individual votes, we assume that at least one of the two collectors is benign and is not compromised. In an election, it is practically impossible to determine which voters voted in which bits from their ballots. Assume the adversary is a collector, say, $\mathcal{C}_2$, so that they even know the $x_{2,\mathrm{i}}$ values. Then, the adversary can compute $Q_\mathrm{i} = p_\mathrm{i} - x_{2,\mathrm{i}} = v_\mathrm{i} + x_{1,\mathrm{i}}$ for each voter. The adversary wants to distinguish between two election possibilities. For each possibility, they can compute $x_{1,\mathrm{i}} = Q_\mathrm{i} - v_\mathrm{i}$ for each voter. Then they must determine which combination of random values was generated. But since these random values were taken from a uniform distribution in $[0, X)$ where X is much larger than 2^L, it is very likely that both combinations will have the same probability, and the adversary will have no way of knowing which is correct. This proof can be adapted to prove that the p'_i values also cannot be used to determine voters' votes.

The Pedersen commitments reveal no information about the shares other than that their sums agree with the σ and σ' values. This basically follows from the security of Pedersen commitments.

6.3 Security of Verification

We first consider attacks on the ability of verification to detect invalid ballots. We then consider attacks that use verification to leak vote information.

Recall that a ballot is created from both v_i and v'_i. For the ballot to be valid, both must have exactly one bit set, and they must be in reverse positions of each other. Of course, if either has zero bits set, it is equal to zero and verification will yield $v_\mathrm{i}v'_\mathrm{i} = 0 \neq 2^{L-1}$. If either has more than one bit set, it has a prime factor other than two. The product $v_\mathrm{i}v'_\mathrm{i}$ will have this prime factor as well and hence cannot be equal to 2^{L-1}. Finally, if the bits in v_i and v'_i are not positioned opposite each other, the product will be $v_\mathrm{i}v'_\mathrm{i} = 2^k$ for some $k \neq L-1$. Although the result of verification is modulo n, since collectors check that $0 \leq p_\mathrm{i}, p'_\mathrm{i} < 3X$ for all ballots and since $v_\mathrm{i} = p_\mathrm{i} - x_{1,\mathrm{i}} - x_{2,\mathrm{i}}$ (and similarly for v'_i), voters cannot submit ballots for voting vectors that exceed the range $-3X < v_\mathrm{i}, v'_\mathrm{i} < 3X$. Therefore, $-n/2 \leq -9X^2 < v_\mathrm{i}v'_\mathrm{i} < 9X^2 \leq n/2$, so verification will never exceed the modulus n as long as the collectors behave correctly. Note that if we used modular secret sharing for our scheme, we could not simply use modular verification because checking that $v_\mathrm{i}v'_\mathrm{i} = 2^{L-1} \pmod K$ for some K would not necessarily ensure that v_i and v'_i are correctly formatted.

Verification cannot detect a voter who simply votes in the wrong location. This attack is discussed more later.

It is also possible for a voter to collude with a collector to pass verification. One technique involves the collector using different shares during verification than during vote casting. Another technique involves adding $2^{L-1} - v_\mathrm{i}v'_\mathrm{i}$ to S_j so that to the other collector, the product appears to be 2^{L-1} instead of the invalid $v_\mathrm{i}v'_\mathrm{i}$.

Next, we formally prove that neither $\mathcal{C}_1$ nor $\mathcal{C}_2$ can determine any individual vote. In the initial 2014 paper [19], sub-protocol 1 would allow either collector to figure out a voter's vote from the information received from the other by trying each $v_\mathrm{i} = 2^\mathrm{j}$ for $\mathrm{j} = 0, 1, \ldots, L-1$. Sub-protocol 1 was revised in the 2017 paper [20] which also formally proves that the revised sub-protocol 1 is secure and prevents any collector from figuring out any voter's vote.

A similar idea can be used here to show the security of our verification protocol. If $\mathcal{C}_1$ did not send S_1 as described but instead sent $s_1 + s'_1$ as well as other necessary values separately to

$\mathcal{C}_2$, then $\mathcal{C}_2$ could try different valid v_i and v'_i until $(p_\mathrm{i}-v_\mathrm{i}-x_{2,\mathrm{i}})x'_{2,\mathrm{i}}+(p'_\mathrm{i}-v'_\mathrm{i}-x'_{2,\mathrm{i}})x_{2,\mathrm{i}} = x_{1,\mathrm{i}}x'_{2,\mathrm{i}}+x'_{1,\mathrm{i}}x_{2,\mathrm{i}} = s_1+s'_1+s_2+s'_2$ holds true. However, since $\mathcal{C}_1$ only gives $\mathcal{C}_2$ S_1 which contains random value $s_1 + s'_1$, $\mathcal{C}_2$ may obtain a non-random variable by computing $S_1 + s_2 + s'_2$ or $S_1 + S_2$. However, regardless of v_i and v'_i, we have $S_1 + s_2 + s'_2 = 2^{L-1} - p_\mathrm{i}p'_\mathrm{i} + p_\mathrm{i}x'_{2,\mathrm{i}} + p'_\mathrm{i}x_{2,\mathrm{i}} - x_{2,\mathrm{i}}x'_{2,\mathrm{i}}$ and $S_1 + S_2 = 2^{L-1} - p_\mathrm{i}p'_\mathrm{i}$. Therefore, $\mathcal{C}_2$ cannot brute force the voter's vote.

Alternatively, we can show that $\mathcal{C}_2$ could have already computed S_1 assuming the voter voted correctly. If this is the case, we can assume that $p_\mathrm{i}p'_\mathrm{i} + S_1 + S_2 = 2^{L-1}$. For this reason, $\mathcal{C}_2$ could have computed S_1 as $S_1 = 2^{L-1} - p_\mathrm{i}p'_\mathrm{i} - S_2$ anyway. The same applies for S_2 and $\mathcal{C}_1$.

We should also consider attacks involving intentional incorrect behavior by one of the collectors. One possible attack of interest on STPM involves using a share value other than the one shared with the voter. Ultimately, this could allow one of the collectors to change, in effect, the ballot submitted for verification, even if the voter submitted a correct ballot.

Assume that a voter correctly votes $v_\mathrm{i} = 2^{L-b_\mathrm{i}}$ and $v'_\mathrm{i} = 2^{b_\mathrm{i}-1}$, so that $v_\mathrm{i}v'_\mathrm{i} = 2^{L-1}$. A collector who uses a different $x_{\mathrm{j},\mathrm{i}}$ and/or $x'_{\mathrm{j},\mathrm{i}}$ for verification can force the process to yield $(v_\mathrm{i} + t)(v'_\mathrm{i} + t') = 2^{L-1} + tv'_\mathrm{i} + t'v_\mathrm{i} + tt'$ for any t and t' of their choosing. Of course, this has the potential to leak information. For example, the collector can simply choose $t = 0$ and $t' = 1$, yielding $2^{L-1} + v_\mathrm{i}$, from which they simply need to subtract 2^{L-1} to obtain the voter's vote. Furthermore, they can do this without detection if they are sent the other S_j before sending or committing to their own. They first perform the attack above to learn the voter's vote. Then, before sending their S_j to the other collector, they subtract from it $tv'_\mathrm{i} + t'v_\mathrm{i} + tt'$. That collector will compute the product as 2^{L-1}, so the evidence of the attack has effectively been removed.

By using commitments during verification, we can ensure that if such an attack is performed, it is at least detected, unless the collector is able to guess the voter's selection and location. Before the malicious collector receives S_j from the benign collector, they cannot learn the voter's vote, and hence do not know how to adjust their S_j to remove the evidence of the attack. Requiring them to commit to S_j before receiving $S_{3-\mathrm{j}}$ from the other collector ensures that the attack will likely result in either failed verification or an incorrect commitment from the other collector's point of view. Preventing the attack in the first place, or

distinguishing between an invalid ballot and an attack from the other collector, are possible directions for future work.

Just as with location anonymization, data received should be validated, and attacks involving invalid Paillier cryptosystems remain an area of future work.

As shown in this section, there are still some shortcomings with verification. Nevertheless, the transparency enabled by the voting vector can allow participants to detect attacks that were not detected during verification. This is discussed in the next section.

6.4 Security against Tampering

A collector may attempt to tamper with the final voting vector by publishing incorrect σ_j or σ'_j. This attack can be detected by anyone because this collector will be unable to produce Pedersen commitments for these shares whose product is g^{σ_j} or $g^{\sigma'_j}$, or either σ_j or σ'_j will not be in $[0, (A-1)/2)$.

As discussed above, some attacks performed by voters or collectors may get through verification and end up included in the final voting vector. One attack, mentioned earlier, involves voting in the wrong location. Most likely, this attack would result in two bits being set in one location, which could be detected by anyone who can see the final voting vector. It is also possible that the bit collides with another voter's bit and carries. This will decrease the number of 1-bits on the final voting vector, which can also be detected by anyone with the vector.

Even if attacks are performed such that the number of 1-bits on the final voting vector is correct, a voter can check whether their vote is included correctly. This way, incorrect results can be detected as long as voters have unique locations. From our analysis of the location anonymization technique used, it would likely be difficult to perform a meaningful attack on this process, let alone one that causes collisions in a very particular way to manipulate the results of the election.

7. PERFORMANCE

In order to evaluate the performance characteristics of our e-voting scheme, we implemented some of the key components in Java. We present the results of our tests below.

Note that in our Paillier implementation, we use various simplifications given in [9]. Further optimizations may be possible to obtain better performance.

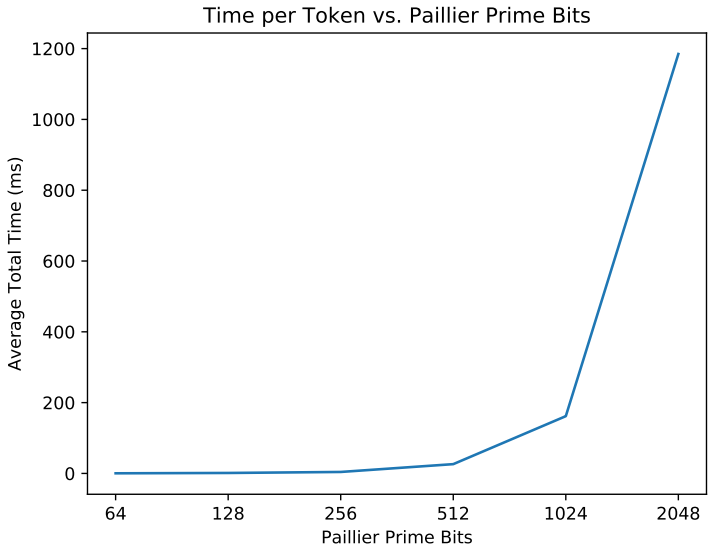

Figure 7.1. The total time taken by both collectors to perform location anonymization on a single location token with different size keys.

Figure 7.1 shows the results of our location anonymization test. Paillier keys were generated of different sizes by varying the bit lengths of the primes generated. At each bit length, the figure shows the average total time taken by both collectors to encrypt a location token (chosen randomly from 0 to 31), perform secret sharing, decrypt the result, and then perform the simplification (assuming 32 voters). The sample size is 10 for each bit length.

This test shows that despite the approximately cubic time complexity, primes as large as 2048 bits can be used while still maintaining reasonable performance.

Figure 7.2 shows the results of our commitment test. Pedersen commitments are created with different primes, shown in Table 7.1. These are the first safe primes at various bit lengths. For each safe prime, we use $g = 4$ and $h = 9$. These parameter choices are

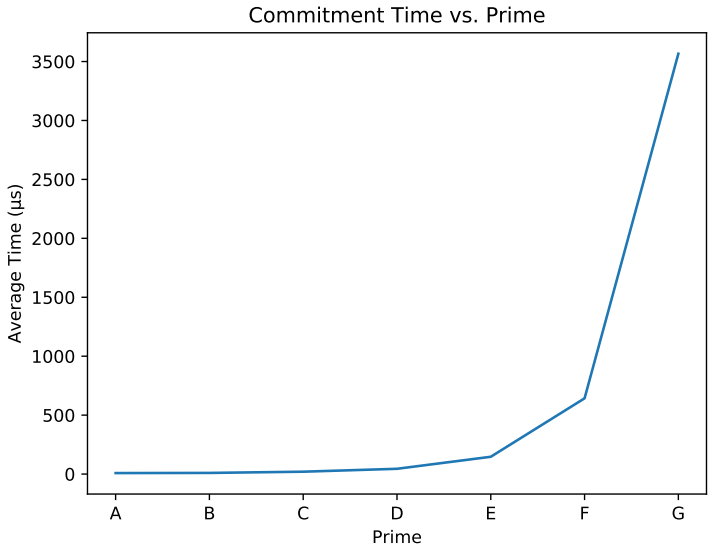

Figure 7.2. The time to create a Pedersen commitment with different primes.

only intended for performance evaluation–we do not perform any security analysis on these particular values. s and t are chosen randomly for each commitment. The sample size is 10000 for each prime.

Table 7.1. The primes used for our commitment performance test.

Symbol	A
A	$2^{16} + 7$
B	$2^{32} + 91$
C	$2^{64} + 3103$
D	$2^{128} + 12451$
E	$2^{256} + 230191$
F	$2^{512} + 286867$
G	$2^{1024} + 1657867$

Again we have roughly cubic time complexity. However, commitment creation took less than a hundredth of a second on average, even for $A \approx 2^{1024}$. Keep in mind that four commitments are necessary for each voter, two per collector.

Figure 7.3 shows the results of our verification test. Just as with Figure 7.1, we measure performance at various key lengths. We use random values for p_{i}, p'_{i}, $x_{1,\mathrm{i}}$, $x_{2,\mathrm{i}}$, $x'_{1,\mathrm{i}}$, and $x'_{2,\mathrm{i}}$.

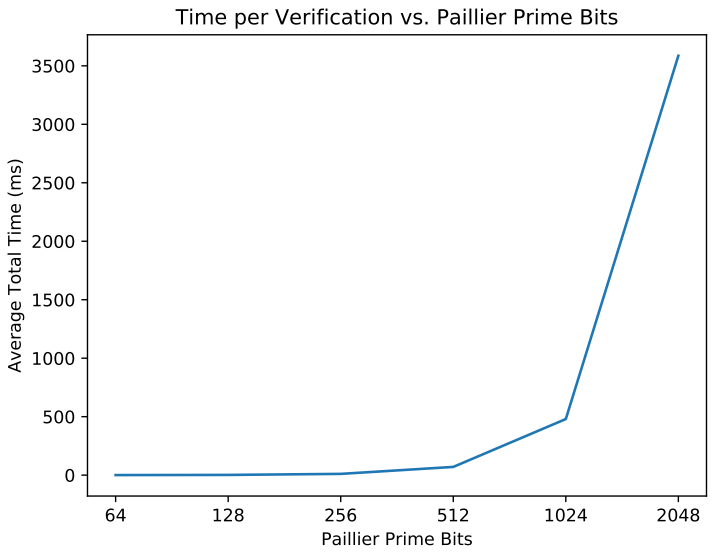

Figure 7.3. The total time taken by both collectors to verify a single voter's ballot with different size keys.

The test involves performing STPM twice as well as performing the final summations. It does not include checking the value of the final summation (the inputs are random anyway). The sample size is 10 for each bit length.

This procedure also exhibits cubic time complexity. But even with 2048-bit primes, this step seems to take only a few seconds per voter.

To better understand the implications of our performance tests, assume we want to conduct an election with 300 voters and 3 candidates. Then, $L = MN = 900$. X can be set to 2^{1000}, which is much larger than $2^L = 2^{900}$. A can be set to $2^{1024} + 1657867$, which is at least $2NX = 2 \cdot 300 \cdot 2^{1000}$. 1024-bit primes can be used for the Paillier cryptosystem, which will yield $n = pq \geq 2^{1023} \cdot 2^{1023} = 2^{2026} = 2^{26}X^2 \geq 18X^2$. Based on our performance tests, the collectors in total may take on the order of seconds of computation time per voter. Furthermore, parallelism can be used to reduce the actual time spent performing the various operations.

Our performance tests show that while the scheme on its own does not scale well to very large elections, it should be able to exhibit reasonable performance for smaller and medium-sized elections, for example with hundreds of voters. For large elections, a hierarchical

structure can be employed as suggested in the 2014 paper [19] as well as the 2017 paper [20]. This would likely be acceptable for most large elections since it is usually not important to preserve the privacy of votes when aggregated in hundreds. In fact, sometimes partial results are desirable.

8. COMPARISON WITH OTHER SCHEMES

Although our scheme does not achieve all security properties that may be desired for an election, it offers many advantages compared with other schemes that could make it a good choice for certain scenarios. In this chapter we discuss how our scheme compares with some other schemes in the literature that we mentioned earlier.

The scheme proposed by Benaloh and Tuinstra [2] is able to prevent coercion. Furthermore, the authors discuss how multiple tallying authorities can be employed to prevent a single authority from learning voters' votes. However, unlike our scheme, their scheme requires a voting booth, so it would not be an option for elections that are entirely online. Furthermore, the procedure may be confusing for a voter, particularly one who does not understand the underlying theory. On the other hand, our scheme can in most part be handled by an implementation, and although this means trusting that the computer is performing the protocol properly, voters are free to choose an implementation they trust.

DRE-i [7] and DRE-ip [17] involve voting devices. As a result, these schemes are also not suitable for online elections, and voters have to trust the system to keep their vote private. However, these systems have auditing mechanisms that prevent voter coercion. Koinonia [6] is an online system, but does not prevent voter coercion. None of these three systems allows for non-technical voters to verify their plain votes directly as our scheme does.

One scheme that allows for plain vote verification is the blockchain-based scheme by Liu and Wang [10]. Once a voter obtains the needed blind signatures on their vote string, they create and cast their ballot. To cast the ballot, the voter puts it on a blockchain. While any voter can see their ballot on the blockchain, the authors do not provide a strong mechanism for submitting to the blockchain anonymously. Our system does not require ballots to be submitted anonymously because they do not reveal votes on their own.

Next, we consider the schemes by Nair et al. [12] and Liu and Zhao [11], both of which use some form of polynomial-based secret sharing. One potential concern with the system by Nair et al. is the lack of individual verifiability. While Liu and Zhao's scheme has a verifiability mechanism, both schemes use physical devices for vote casting.

Finally, we consider the non-interactive scheme by Zou et al. [20]. Because our scheme is based on this one, the two share many security properties. However, there are some advantages to the scheme presented here. The scheme in [20] uses a multi-round location anonymization technique that requires active participation from voters, possibly through multiple rounds. With our system, voters would only need to log in once they are ready to vote. At this point, they receive location shares from the collectors and perform a single modular addition to compute their location. Our scheme also does not require collectors to generate as many shares (order of N vs. order of N^2). Finally, our scheme uses commitments to prevent certain types of attacks by collectors.

Table 8.1 summarizes and adds to these remarks.

Table 8.1. Comparison of our scheme with other schemes.

Scheme	**Advantages vs. ours**	**Disadvantages vs. ours**
Benaloh and Tuinstra [2]	Prevents coercion, strong cryptographic verification	Requires voting booth, may be confusing to voters, no plain vote verification
DRE-i [7] and DRE-ip [17]	Strong cryptographic verifiability, prevent coercion	Use voting devices, no plain vote verification
Koinonia [6]	Strong cryptographic verifiability	No plain vote verification
Liu and Wang [10]	Plain vote verification with random string	No strong anonymity mechanism
Nair et al. [12]	Efficient	Uses voting devices, no individual verification
Liu and Zhao [11]	Efficient	Uses voting devices, no plain vote verification
Zou et al. (non-int.) [20]	Could save time by not generating commitments	Inefficient location anonymization, more shares needed, less secure against attacks by collectors

9. APPLICATION TO CYBERSECURITY EDUCATION

When learning the fundamentals of cybersecurity and cryptography, it may be more difficult to understand and remember information if students do not understand the motivation behind the theoretical concepts that are being taught. For this reason, it is important to provide real-world examples of how these fundamentals can be applied. E-voting is one particularly useful tool in this regard. Most students already have an understanding of some of the requirements and difficulties often associated with conducting elections. These requirements tend to correspond directly to cryptographic concepts. For example, when a voter submits their ballot, their vote must be kept confidential. Therefore, students could apply new cryptography concepts, such as encryption, and try to think of ways in which this requirement can be achieved.

Our scheme in particular uses many different concepts from the field of cryptography. Hence, one way students can benefit from our scheme is by implementing various portions of it. Students can enhance their understanding of secret sharing by implementing the secret sharing used by our system, and by comparing it to other types of secret sharing. Our system provides one possible application of Pedersen commitments, and again students can write an implementation to further enhance their understanding. Concepts that can be taught using our system are summarized in Table 9.1. Some apply to e-voting in general, whereas some apply to our scheme in particular (and possibly other existing schemes).

Table 9.1. Concepts from cryptography related to our e-voting system.

Concept	**Application**
Secret Sharing	Ballot Secret Sharing
Homomorphic Encryption	Paillier Cryptosystem
Secure Computation	STPM
Commitment	Share Commitments
Anonymity	Anonymous Voting
Secure Communication	Communication between Parties
Authentication	Authentication of Parties

The instructor can further motivate students by incorporating the e-voting system in the class structure. Lessons can be made interactive by using the e-voting system to ask and answer questions, and results from these "elections" can be used to pace the class. Students can also use the system to provide feedback at certain points during the term. In addition to providing a more adaptive learning environment for students, use of the e-voting system will potentially increase student interest in the concepts that it uses.

10. FUTURE WORK

While the transparency of the system we presented enables protection against a wide variety of attacks, elections performed on a large scale also require efficient correction of such attacks. Due to the confidentiality of submitted ballots, it may be difficult to trace back to the source of an attack by a voter. We also saw that there is some opportunity for collectors to perform attacks that are difficult to trace. Allowing for efficient recovery from attacks is one possible direction for future research.

While voters can use the final voting vector to verify that their vote is correct, they still may not be confident that they have a unique location. To address this issue, it may be possible to allow the voter to submit random information along with their vote and then verify that the information they submit appears in the correct location, similar to the random string in [10].

One more limitation with our work is the lack of receipt-freeness. Voters can provide others with data received from collectors such as location data or share data to demonstrate that they voted for a particular candidate. Receipt-freeness is not always a requirement, but for elections where this is necessary, it may be useful to have a variant of this scheme that holds this property. Such is one more possible direction for future work.

11. SUMMARY

Voting is essential both to many private organizations as well as to democratic governments. As the need for larger elections increases, it is crucial that schemes are developed that are efficient for elections with many voters and provide security properties that voters expect. Here we presented a simple, efficient e-voting scheme that allows voters to visually verify that their vote is included in the final tally.

Our scheme uses homomorphic secret sharing to ensure that individual votes are confidential but that the final tally is visible to everyone. The presence of multiple collectors distributes trust among multiple parties. Commitments are used to prevent collectors from tampering with election results. Secure two-party multiplication ensures that voters are not able to vote in multiple locations. The use of a voting vector allows non-technical voters to verify that their vote is counted without understanding the underlying cryptography.

We also discussed how the scheme can be used in cybersecurity education to improve students' understanding of security concepts. Because the scheme uses many different cryptographic concepts, students can use and study the system and even implement components of it to see how cryptography can be used to solve practical problems.

Our secure e-voting scheme is practical for many different types of elections and can even be used to teach cybersecurity. Moreover, it can help establish trust in both technical and non-technical voters. Consequently, our scheme is a major step in the direction of widespread adoption of e-voting.

REFERENCES

[1] Riza Aditya et al. "An efficient mixnet-based voting scheme providing receipt-freeness". In: *International Conference on Trust, Privacy and Security in Digital Business*. Springer. 2004, pp. 152–161.

[2] Josh Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 544–553.

[3] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. "Secret-shared shuffle". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 342–372.

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

[5] Laure Fouard, Mathilde Duclos, and Pascal Lafourcade. "Survey on electronic voting schemes". In: *supported by the ANR project AVOTÉ* (2007).

[6] Huangyi Ge et al. "Koinonia: verifiable e-voting with long-term privacy". In: *Proceedings of the 35th Annual Computer Security Applications Conference*. 2019, pp. 270–285.

[7] Feng Hao et al. "Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting". In: *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*. 2014.

[8] Jaydeep Howlader et al. "Uncoercibility in e-voting and eauctioning mechanisms using deniable encryption". In: *International Journal of Network Security and Its Applications* 3.2 (2011), pp. 97–109.

[9] Jonathan Kaltz and Yehuda Lindell. "Introduction to modern cryptography: principles and protocols". In: *Chapman and Hall* (2008).

[10] Yi Liu and Qi Wang. "An e-voting protocol based on blockchain". In: *Cryptology ePrint Archive* (2017).

[11] Yining Liu and Quanyu Zhao. "E-voting scheme using secret sharing and K-anonymity". In: *World Wide Web* 22.4 (2019), pp. 1657–1667.

[12] Divya G Nair, VP Binu, and G Santhosh Kumar. "An improved e-voting scheme using secret sharing based secure multi-party computation". In: *arXiv preprint arXiv:1502.07469* (2015).

[13] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 223–238.

[14] Pascal Paillier and David Pointcheval. "Efficient public-key cryptosystems provably secure against active adversaries". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 1999, pp. 165–179.

[15] Torben Pryds Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing". In: *Annual international cryptology conference*. Springer. 1991, pp. 129–140.

[16] Saeed Samet and Ali Miri. "Privacy preserving ID3 using Gini index over horizontally partitioned data". In: *2008 IEEE/ACS International Conference on Computer Systems and Applications*. IEEE. 2008, pp. 645–651.

[17] Siamak F Shahandashti and Feng Hao. "DRE-ip: a verifiable e-voting scheme without tallying authorities". In: *European Symposium on Research in Computer Security*. Springer. 2016, pp. 223–240.

[18] Oliver Spycher et al. "A new approach towards coercion-resistant remote e-voting in linear time". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2011, pp. 182–189.

[19] Xukai Zou et al. "Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties". In: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE. 2014, pp. 136–144.

[20] Xukai Zou et al. "Transparent, auditable, and stepwise verifiable online e-voting enabling an open and fair election". In: *Cryptography* 1.2 (2017), p. 13.