Secure Voting System Using Paillier Homomorphic Encryption

GRADUATE PROJECT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University – Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of Master of
Science in Computer Science

By

Cuong Ngo

Fall 2014

Committee Members

Dr. Dulal Kar
Committee Chairperson

Dr. David Thomas
Committee Member
# Table of Contents

LIST OF FIGURES ........................................................................................................ iii

LIST OF TABLES ............................................................................................................ iii

ABSTRACT ...................................................................................................................... 1

1. INTRODUCTION ........................................................................................................ 2

3. BACKGROUND .......................................................................................................... 3

   3.1. E-voting system requirements .............................................................................. 4

   3.2. Motivation for the electronic voting system ......................................................... 5

   3.3. Cryptography ....................................................................................................... 6

       3.3.1. RSA encryption ......................................................................................... 7

       3.3.2 ElGamal encryption .................................................................................. 7

       3.3.3. Paillier cryptosystem ............................................................................... 8

       3.3.4. Homomorphic property .......................................................................... 9

   3.4. Secure Socket Layer Client/Server Protocol .................................................... 14

4. SYSTEM DESIGN AND IMPLEMENTATION .......................................................... 19

   4.1. The client ............................................................................................................ 20

   4.2. The registration server ..................................................................................... 21

   4.3. The voting server ............................................................................................. 23

   4.4. The tally server ................................................................................................. 26

5. TESTING .................................................................................................................. 28

6. FUTURE WORK AND CONCLUSION .................................................................... 31

   6.1. Future work ....................................................................................................... 31

   6.2. Conclusion ....................................................................................................... 31

REFERENCES ............................................................................................................... 32

APPENDIX .................................................................................................................... 1

1. Use keytool to create a server certificate ................................................................. 1

2. Install phpMyAdmin to set up and manage database ........................................... 4

   2.1. Set up MySQL .................................................................................................. 4

   2.2. Set up phpMyAdmin on a Mac ....................................................................... 4
LIST OF FIGURES

Figure 1. Handshake protocol action [14] .................................................................17
Figure 2. Voting system design ..........................................................20
Figure 3. Client login Dashboard ..........................................................21
Figure 4. Register process ..........................................................22
Figure 5. Registration interface ..........................................................22
Figure 6. Voter database ..........................................................23
Figure 7. Login server dashboard ..........................................................24
Figure 8. Login process ..........................................................24
Figure 9. List of candidates in the candidate database .........................25
Figure 10. Tally server dashboard ..........................................................26
Figure 11. Result table for votes ..........................................................27
Figure 12. Tally process ..........................................................27
Figure 13. Homepage of phpMyAdmin setup ........................................5
Figure 14. Homepage of phpMyAdmin database ...................................6

LIST OF TABLES

Table 1. Vote messages to be encrypted [1] ........................................9
Table 2. Encrypted Vote Ci [1] ..........................................................10
Table 3. Voting system survey ..........................................................29
ABSTRACT

A manual voting process is excessively time-consuming and takes a lot of steps to finish the vote. With rapid development of information technology, an electronic voting system (E-voting system) is more convenient to vote for someone from a list of candidates shown on a computer screen. However, security is a big concern for electronic voting. In this project, we build a secure voting system using Paillier homomorphic encryption. Paillier homomorphic encryption is a kind of public key cryptography that has the homomorphic property that can be exploited to calculate the sum of votes without revealing to the system which vote is voted for which candidate. The voting system design has a simple interface that clients can use easily. The system allows a voter to use an ID and PIN to vote for his/her favorable candidate. This online voting system is able to save time for users and it is more reliable than the traditional system.
1. INTRODUCTION

“E-voting” is a term that is described in different information and communication technologies (ICT) platforms: Internet systems, polling booth machines, and telephone voting system [7]. Each platform has both negative and positive features. However, the Internet voting systems are most popular and successful today. Each voter can vote for candidates remotely through the Internet. Voters can easily open websites, software, or applications on their computers or smartphones to vote anytime, anywhere. However, their voting system can be attacked if it does not have any algorithm or protocol to protect it. Attackers can catch packets that voters transfer on the Internet easily; they change information and send to the voting server. In addition, they can interrupt votes which voters send over the Internet. Accordingly, researchers are thinking about methods to defend against such attacks using cryptography algorithms to encrypt data a vote before sending it to the server. This idea was first mentioned in 1981 [3]. Some algorithms are being used in voting systems such as Diffie-Hellman Encryption, RSA Encryption, Elgamal Encryption, and Paillier Encryption. This project uses the homomorphic properties of Paillier Encryption to protect a vote sent over the Internet.

The main goal of this project is to design a secure voting system using the Internet platform to communicate between the voting system and voters. To protect the system, two major security methods are studied: Paillier encryption algorithm and the Secure Socket Layer (SSL) protocol.

This project report includes four sections, introduction, background, system design, and testing and result. The background introduces E-voting system features, challenges, as well as requirements. It also explains cryptography related to Paillier
encryption and SSL protocol to communicate between a client and a server. Next, the section on system design is a main part of the report, which includes our design about a secure voting system. The next section is about testing and showing how the system design secures the voting system. The last section presents the future work of the project.

3. BACKGROUND

Electronic voting systems are basically using machines to vote. Some electronic voting systems require voters to go to their local polling station to vote through machines that have been designed and designated for voting only. It is a kind of convenience when machines can cast votes, save them, and transfer to another machine, which tallies all the votes together [2]. However, voters still have to spend their time to go to a local polling station to vote. With rapid development of technology and the Internet, researchers are developing modern E-voting systems that allow voters vote through their computer from anywhere. In this project, voters should have their email address so that the voting organization can send the voting software and ID to them. They can install the software on their and process their vote accordingly.

Electronic voting systems allow people to vote for candidates anywhere using computers, which are connected to the Internet [8]. This method has multiple advantages in comparison to traditional methods such as paper ballot. Voters do not need to present at ballot box. It provides ease of accessibility and comfort of use. Another advantage of Internet voting is wide usability that can be applied in all forms of municipal, public, state elections, and referendums [8]. Using an electronic voting system, voters also can vote rapidly and exactly because they don’t need to go to Ballot station and vote thought
their computers. However, it is relatively insecure to deploy an electronic voting system because it requires transferring of data and information over the Internet. Many security measures should be taken to protect ballots and the system from possible attacks. Therefore, to design e-voting system, there are some requirements that users must consider. They are discussed in the following section.

3.1. E-voting system requirements

E-voting process consists of several steps, many of which can be found in the traditional voting. It is more accessible and convenient to voters than the traditional voting. The big advantage is that it does not require voters to present at the poll station. Another advantage, voters can vote from anywhere as soon as they meet the requirements to vote. However, to implement a secure voting system is difficult. Here are some requirements which a secure voting system has to adopt.

Authentication: Only authorized voters should be able to vote. Voters can request to change their information if necessary. A voter’s information can be collected from the birth certificate or other similar document [9].

Uniqueness: No voter should be able to vote more than once. This requirement is necessary and same as the traditional voting. This feature prevents coercion or buying votes.

Accuracy: Voting systems should record the votes correctly. After the vote is recorded, the voter should be able to check if his vote was recorded or not. If he tries to re-vote or change the vote, the system should prevent that.

Integrity: A vote is just for one time decision and cannot be changed. No one should be able to determine how the voter voted, so no one can change the vote.
Verifiability: Verifying that the votes are correctly counted in the final tally should be possible. This feature is mandatory. It has to be audited and tested before the system is used.

Auditability: Reliable and demonstrably authentic election records should be generated.

Reliability: Systems should work robustly to prevent electoral frauds or attacks from outside the system. The E-voting system should be very reliable. The result of an election must be correct and shows up to voters after the election ends.

Voter Confirmation: The voting system should send an email to the voter to confirm that his or her vote has been received by the system correctly. At the end, the result of the voting also can be sent to the voters so that they will know which candidate is a winner.

3.2. Motivation for the electronic voting system

E-voting is the most convenient to vote. It is excellent on equality, building a trust in electoral organization, adding reliability to election results, and increasing the overall efficiency of the polling process [11]. However, to build an E-voting system that can work perfectly over the Internet is a big challenge. Two major challenges that can be considered are security and supporting a large number of voters. Most E-voting solutions cannot work with large number of voters. E-voting also faces security issues because voters vote and send their votes over the Internet which is not a controlled environment [11]. In addition, voting under an electronic voting system occurs automatically without any human supervision. Certainly, voters would like to vote by paper votes at a post office or polling station rather than through an E-voting system because they do not trust the E-voting as their ballots are transferred over the Internet [12]. For example, in 2012, the federal government allowed to use the E-voting for Canton of Zurich voters. The
voters had three options to vote: ballot voting, postal voting, and Internet voting. However, only 20 percent of the votes were cast through the Internet voting [13]. To address some of these challenges, in this project, we have developed an E-voting system that can allow a large number of voters. Specifically, Paillier algorithm is used to support a large number of voters and secure a vote when it is counted. To secure a vote on the Internet, the Secure Socket server/client protocol is used to transfer information over the Internet [9]. The security properties of SSL are discussed in the following sections.

3.3. Cryptography

Cryptographic techniques are used for secure storage, communication, and handling of data and information [12]. It is beyond the scope of this report to discuss all these techniques. In this report, we primarily focus on cryptographic voting schemes that are relevant to the project. Cryptographic techniques have been used to voting systems since 1981[3]. These techniques provide a level of assurance of accuracy and secrecy to a voting system. Largely cryptosystems can be divided to two systems: symmetric key cryptosystems and asymmetric key cryptosystem [2]. A symmetric key system uses the same secret key to encrypt and decrypt the message. For example, if A want to send a message m to B, he will use a secret key (K_{A,B}) already shared between A and B to encrypt the message m and send to B. Then, B also uses the same secret key (K_{A,B}) to decrypt the encrypted message and get the original message. On the other hand, asymmetric key cryptosystem uses two different keys to encrypt and decrypt the message. These keys, one of them called the public key and the other the private key, are different. A uses the public key, which can be shared with anyone who wants to send a message to B, to encrypt a message and then send it to B. Next, B uses the private key
that he or she does not share with anyone to decrypt the message. Both schemes have different strengths in terms of security. For instance, asymmetric cryptosystems provide privacy and reliability but it does not work with authentication. However, an asymmetric cryptosystem can provide authentication but it is more complex than a symmetric system.

A number of cryptographic schemes are used to design a voting system. Most of these schemes can count the votes securely. Mostly frequently used algorithms are RSA (Rivest, Shamir, and Adleman) encryption, ElGamal encryption, and Paillier encryption [3] [1] [2].

3.3.1. RSA encryption

RSA is the first concept of the public key scheme, which was proposed by Rivest, Shamir, and Adelman (RSA) in 1978 [3] [2] [17]. The algorithm chooses two large prime numbers $p$ and $q$ randomly to generate a public key $(n, e)$ where $n$ is an integer and $n = pq$, $e$ is an integer, and $\gcd(e, \varphi(n)) = 1$. The $\varphi(n)$ denotes the Euler function and is given by $\varphi(n) = (p - 1)(q - 1)$. The decryption key $d$, a private key, is calculated using the formula: $ed \mod (p-1)(q-1) = 1$. To encrypt a message $m$, the message is divided into many data packets where each data packet $m_i$ is less than $n$ numerically. Each data packet $m_i$ is encrypted using the public key $(n, e)$ as $c_i = m_i^e \mod n$. To get the plaintext $m_i$ back, one uses the private key $d$, and compute $m_i = c_i^d \mod n$. RSA has some homomorphic property which can be used to count votes in a voting system. However, it is computationally very intensive and cannot be used to handle large number of voters.

3.3.2 ElGamal encryption

Similar to the RSA encryption, ElGamal encryption has also some homomorphic property. Therefore, some e-voting system can use this encryption for tallying. The steps
to generate a pair of keys (public and private key) are similar to RSA encryption. If a client wants to send a message to a server using the ElGamal encryption, it can follow the following algorithmic steps. First, to generate a public key, the server selects a large prime number \( p \) and a primitive element \( g \) of the cyclic group \( \mathbb{Z}_p^* \) (\( g \) is a primitive element of \( \mathbb{Z}_p^* \) if every element of \( \mathbb{Z}_p^* \) can be expressed as a power of \( g \)) [3] [1]. Then the server selects \( q = p - 1 \) and a random number \( a \) in \( \mathbb{Z}_p \) and then compute: \( b = g^a \mod p \).

The system’s the public key is \((g, q, b)\) and the private key is \(a\). To encrypt a message \( m \), the client chooses a random number \( k \in \mathbb{Z}_q \), and computes \((c_1, c_2) = (g^k, mb^k) \mod p\).

The client sends this result to the server. The server will decrypt this cipher text by using the formula: \( m = (c_2/c_1^a) \mod p \). This scheme is related to the Diffie-Hellman problem of discrete logarithms [2]. Hence, if the Diffie-Hellman was not an issue anymore, the ElGamal will be broken.

3.3.3. Paillier cryptosystem

Paillier encryption is the most well-known cryptography scheme. Pascal Paillier invented it in 1999 [1] [2]. It is more advantageous than previous schemes. Thus, it is very handy in E-voting applications. Paillier encryption also is an algorithm for public key cryptography similar to other public key cryptography schemes [12]. Here is a detail about how Paillier algorithm works:

**Step 1: Paillier Key Generation**

1. Select two large prime numbers \( p \) and \( q \) randomly and independently of each other such that \( \gcd (pq, (p - 1)(q - 1)) = 1 \). The \( \gcd \) is the general common divisor of two or more non-zero integers which is the largest positive integer that divides the numbers without a remainder.
2. Compute \( n = pq \) and \( \lambda = \text{lcm}(p - 1, q - 1) \) with \( \lambda(n) \) being the Carmichael function. The \( \text{lcm} \) is the least common multiple of two or more non-zero integers which is the smallest integer that is divisible by every member of a set of numbers without a remainder.

3. Choose generator \( g \) where \( g \in \mathbb{Z}_{n^2}^* \). There are two ways to select the \( g \).

   Method 1: Randomly select \( g \) from a set \( \mathbb{Z}_{n^2}^* \) where \( \gcd(g^\lambda \mod n^2 - 1, n) = 1 \).

   Method 2: Select \( a \) and \( b \) randomly from a set \( \mathbb{Z}_n^* \) then calculate \( g = (a \times n + 1) \times b^n \mod n^2 \).

4. Calculate the follow modular multiplicative inverse

   \[
   \mu = \left(L(g^\lambda \mod n^2)\right)^{-1} \mod n \quad \text{where} \quad L(u) = \frac{u - 1}{u}
   \]

   So, the pair of keys generated: the public key is \((n, g)\) and the private key is \((\lambda, \mu)\).

**Step 2: Encryption process**

1. The message \( m \) is a message need to be encrypted where \( m \in \mathbb{Z}_n \)

2. Choose a random number \( r \) with \( r \in \mathbb{Z}_n^* \)

3. Compute ciphertext \( c = g^m \times r^n \mod n^2 \) (use the public key \((n, g)\))

**Step 3: Decryption process**

Ciphertext \( c \) will be decrypted to get message \( m \) as follows by using the private key \((\lambda, \mu)\):

\[
M = L(c^\lambda \mod n^2) \times \mu \mod n
\]

**3.3.4. Homomorphic property**

The encryption algorithm \( E \) is homomorphic if given \( E(m1) \) and \( E(m2) \), one can obtain \( E(m1 +/\times m2) \) without decrypting \( m1 \) and \( m2 \). A straightforward way, the users
want to calculate the product of message m1 and message m2. All they have are the encrypted message E(m1) and E(m2) but they do not want to decrypt E(m1) and E(m2), and then calculate the product (m1 +/× m2). To calculate it, they will calculate T = E(m1) +/× E(m2). Finally, they just decrypt T to get (m1 +/× m2). This property can calculate (m1 +/× m2) but the user will not know what is message m1 and message m2. This is a major feature that is used to tally ballots in an E-voting system. Pallier algorithm is one of algorithms which have this homomorphic property. Addition of the encrypted ballots will be in the encrypted tally [7] [2]. When an administrator decrypts the Paillier result, he will get the final result of a poll but he will not know which voter voted for which candidate.

Here is how to use Paillier encryption’s additive homomorphic property for tallying votes [1] [3]:

Let us call the number of voters is \( N_v \) and the number of candidates is \( N_c \). The base used to encrypt messages, is greater than the number of voters (\( b > N_v \)).

Next, the vote messages for candidates will be seen as: 1\(^{st}\) candidate is \( b^0 \), 2\(^{nd}\) candidate is \( b^1 \), 3\(^{rd}\) candidate is \( b^2 \), etc. \( N_c \)-th candidate is \( b^{(N_c-1)} \).

Then the maximum possible number representing a single vote \( m_{max} \) can be expressed as:

\[
m_{max} = \sum_{i=1}^{N_c} b^{i-1}
\]

The maximum possible tally of all votes can be:

\[
T_{max} = N_v \times m_{max}
\]

Then, there are three major steps that coders have to consider:

**Step 1: Key generation**
Public Key

To be able to encrypt $T_{\text{max}}$, RSA modulus $n$ must hold the following:

$$n \geq T_{\text{max}} + 1 \quad \text{Where } n = pq$$

where $p$ and $q$ are large primes and gcd$(pq, (p - 1)(q - 1)) = 1$.

A random integer $g$ is selected where $g \in Z^*_n$ and gcd$(\frac{g^{n-1}}{n}, n) = 1$

The election authorities should choose the prime numbers $p$ and $q$ considering the number of voters and candidates as described above.

Private Key

$\lambda = \text{lcm}(p-1, q-1)$ with $\lambda(n)$ being the Carmichael function.

Modular multiplicative inverse: $\mu = (L(g^{\cdot n^2}))^{-1} \mod n$

where function $L$ is defined as $L(u) = \frac{u-1}{n}$

Step 2: Encryption

$$E(m_i) = c_i = g^{m_i \times r_i} \mod n^2 \quad \text{where } r \in Z^*_n$$

Tallying

At the end of the election, authorities would have at most $N_v$ of encrypted votes.

Then authorities can calculate the encrypted tally, which is the product of all encrypted votes modulo $n^2$.

$$T = \prod_{i=1}^{N_v} c_i \mod n^2 \quad \text{T-Tally}$$

Step 3: Decryption

As described in homomorphic properties of Paillier encryption:

$$m = L(g^{\cdot n^2}) \times \mu \mod n$$

$$D(T) = \sum_{i=1}^{N_v} m_i \mod n \quad \text{D(T)-Decryption of } T$$
As a result of this decryption function, one gets simple tallying of all votes. To determine how many votes cast for each candidate we can use the “Division remainder” method with number of the voters as base. The following example illustrates the use of Paillier’s cryptosystem in voting.

**Example**

This example demonstrates the use of Paillier algorithm for tallying a small number of votes, similar to the examples in [1] and [2]. Let assume that there are 9 voters A1, A2, A3, . . ., A9 and 6 candidates B1, B2, B3, . . ., B6 so that \( N_v = 9 \) and \( N_c = 6 \). Let us select \( b > N_v \), say \( b = 10 \). Voters are supposed to select only one candidate. The vote messages to be encrypted are shown in the following table [1] [2].

<table>
<thead>
<tr>
<th>Voter Name</th>
<th>B1 (10^0)</th>
<th>B2 (10^1)</th>
<th>B3 (10^2)</th>
<th>B4 (10^3)</th>
<th>B5 (10^4)</th>
<th>B6 (10^5)</th>
<th>Vote messages (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( m = 10^1 = 10 )</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( m = 10^2 = 100 )</td>
</tr>
<tr>
<td>A3</td>
<td></td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td>( m = 10^0 = 1 )</td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td>( m = 10^3 = 1000 )</td>
</tr>
<tr>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td></td>
<td>( m = 10^3 = 1000 )</td>
</tr>
<tr>
<td>A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>( m = 10^4 = 10000 )</td>
</tr>
<tr>
<td>A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( m = 10^5 = 100000 )</td>
</tr>
<tr>
<td>A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>( m = 10^0 = 1 )</td>
</tr>
<tr>
<td>A9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( m = 10^3 = 1000 )</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
The maximum vote message can be: \( m_{\text{max}} = 10^5 = 100000 \)

So the maximum possible tally can be: \( T_{\text{max}} = N_v \times m_{\text{max}} = 9 \times 10000 = 90000 \)

Let us perform the following three steps of the Pallier algorithm.

**Step 1: Key generation**
- Choose two primes randomly \( p \) and \( q > \sqrt{90000} \), \( p = 293 \), \( q = 433 \) that \( \gcd (pq, (p-1) \times (q-10)) = 1 \)
- Calculate \( n = p \times q = 293 \times 433 = 126869 \), \( n^2 = 16095743161 \)
  and \( \lambda = \text{lcm}(p-1,q-1) = \text{lcm}(292, 432) = 73 \times 432 = 31536 \)
- Choose Paillier generator \( g \) randomly where \( g \in Z_n^* \) and \( \gcd \left( \frac{g^{\lambda} \mod (n^2-1)}{n}, n \right) = 1 \), so \( g = 2 \)

**Step 2: Encrypt each message \( m \) as shown in the following table**

\[
E(m_i) = c_i = g^{m_i} \times r_i^n \mod n^2 = 2^{m_i} \times r_i^{126869} \mod 16095743161
\]

<table>
<thead>
<tr>
<th>Voter Name</th>
<th>Vote message to be encrypted</th>
<th>Random ( r_i )</th>
<th>Encrypted Vote ( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>( m=10^1 = 10 )</td>
<td>26181</td>
<td>1476346097</td>
</tr>
<tr>
<td>A2</td>
<td>( m=10^2 = 100 )</td>
<td>11593</td>
<td>2441495758</td>
</tr>
<tr>
<td>A3</td>
<td>( m=10^0 = 1 )</td>
<td>47971</td>
<td>4580939420</td>
</tr>
<tr>
<td>A4</td>
<td>( m=10^3 = 1000 )</td>
<td>15791</td>
<td>10051435966</td>
</tr>
</tbody>
</table>
After tally server received all encrypted messages from voters, the server will multiply all encrypted messages following Paillier algorithm as shown below.

Tally \( (T) = \prod_{i=1}^{N_v} c_i \mod n^2 \) = \((1476346097 \times 2444495758 \times 4580939420 \times \\
10051435966 \times 1698861485 \times 7276056190 \times 8664547807 \times 6400165985 \times \\
11929667045) \mod 1609574361 = 13722328518

Then the tally server decrypts Tally \( T \) to get tally message \( M \).

\[
M = L(c^\lambda \mod n^2) \times \mu \mod n = \\
\left(\frac{13722328518^{31536} \mod 1609574361}{126869}\right) \times 105161 \mod 126869 = 113112
\]

This decrypted tally already has the base 10, so the tally server does not need to convert it. Finally, the result of this voting is decided. Since candidate \( B4 \) has 3 votes, which is the largest number of votes received by a candidate, so he or she is the winner of this election.

### 3.4. Secure Socket Layer Client/Server Protocol

Netscape has developed the SSL protocol for securing communication over the Internet [13]. It is an application layer cryptographic protocol. It provides reliable end-to-end security services. Users can feel confident through encryption of data using symmetric key encryption algorithm. Another service is a non-repudiation of origin.
integrity through digital signatures using asymmetric key algorithms or public key cryptography algorithms. Data integrity through hashing using message digest or hashing algorithms is also an impression service of SSL protocol. There are two important SSL concepts using in SSL process: SSL session and SSL connection [11] [14]. The session links between a client and a server using Handshake Protocol. If the handshaking fails for some reasons, the SSL socket will be closed. Multiple connections can share cryptographic security parameters. Sessions will define them to make a security channel between a client and the server. Besides, the connection provides a logical client/server link to stabilize the type of service. Each connection is a peer-to-peer relationship, which is associated with one session. The handshake protocol is the most important to establish SSL socket client/server connection and Figure 1 shows how it works.

The exchange between a client and a server can be established using 4 rounds [13] [14]. Round one is used to initiate a logical connection. The client sends a client_hello message to server to establish the security connection and waits for the server to respond. The server checks the message. If everything is correct and consistent, the server will send server_hello message to the client to communicate.

During round two, the server sends its certificate to the client if it is necessary to authenticate. The certificate contains one or a chain of X.509 messages, which is an ITU-T standard for a Public Key Infrastructure (PKI) and Privilege Management Infrastructure (PMI). Next, the server may send a server_key_exchange if it is required. The final message that the server sends to the client is the server_done message. The client can verify that the server has provided a valid certificate. Next, the client starts round three and sends one or more messages back to the server. These messages can be a
certificate if it is available or no_certificate alert message instead, a client_key_exchange message, and a certificate_verify message.

Round four finishes a session of a secure connection. The client and the server send a change_cipher_spec message using the Change CipherSpec Protocol and the finished message together to finish the key exchange and authentication processes successfully.
Establish security capabilities including protocol version session ID CipherSuite, Compression method, and initial Random numbers.

Server may send certificate, Key exchange, and request Certificate. Server signals end Of hello message phase

Client send certificate if Requested. Clients sends key Exchange. Client may send Certificate verification.

Chang cipher suite and Finish handshake protocol

Note: Shaded transfers are optional or situation-dependent messages that are not always sent

Figure 1: SSL Handshake protocol [14]
In this project, the SSL socket client/server protocol is implemented using Java code. Java uses Java Secure Socket Extensions (JSSE) to provide an implementation for creating a SSL client socket used by clients and a SSL server socket used by a server. The basic steps for this security connection can be described as follows:

**Steps for creating SSL Client socket**

1. Determine the SSL Server Name and port in which the SSL server will be listening
2. Register the JSSE provider
3. Create an instance of SSLSocketFactory
4. Create an instance of SSLSocket
5. Create an OutputStream object to write to the SSL Server
6. Create an InputStream object to receive messages back from the SSL Server

**Steps for creating SSL Server socket**

1. Register the JSSE provider
2. Set system property for keystore by specifying the keystore which contains the server certificate
3. Set system property for the password of the keystore which contains the server certificate
4. Create an instance of SSLServerSocketFactory
5. Create an instance of SSLServerSocket by specifying the port to which the SSL Server socket needs to bind with
6. Initialize an object of SSLSocket
7. Create InputStream object to read data sent by clients
8. Create an OutputStream object to write data back to clients

In the following, we focus and discuss how these protocols and algorithms can be used to address challenges of designing a secure voting system.

4. SYSTEM DESIGN AND IMPLEMENTATION

Basically, the E-voting system should have a graphical user interface (GUI), which voters can use to communicate with the server system. Three servers are the login server, the registration server, and the tally server (Figure 2). Each server communicates with a database. The registration server connects to the voter database, the login server connects to the candidate database and the voter database, and the tally server connects to the tally database and the candidate database.

The voting server and the tally server are separate and do not share the voter database and the tally database between them. This feature ensures that the tally server cannot find which voter votes for which candidate. The tally server receives both the ID and the vote. Based on the ID, the tally server can check which voter already voted. However, it will not know which voter is voting because the tally server doesn’t have the voter database.

The SSL client/server protocol is the most important protocol to do a secure exchange of a message for the voting system. The client communicates with the voting server so that a voter can authenticate his ID and PIN to process the vote. The registration server allows a user to register his or her information. Next, an administrator who manages the voter database can check this information and consider adding him to the voter database or not. If the information is valid, the prospective voter will receive a
confirmation email. The email includes an ID and PIN and the voting software that the person needs to vote.

![Figure 2: Voting system design](image)

4.1. The client

The client dashboard is shown in Figure 3. All eligible voters will receive a confirmation email that they are eligible to vote for their candidates. In the email, they will have a Voter ID and a PIN that they can use to login to the server. After a voter logs in successfully, he or she will see a list of candidates. If they login three times incorrectly, the software will be closed and they will have to open it again. If they do not remember the Voter ID or the PIN, they will have to read their email again or request to resend the Voter ID and PIN. The “Register” button is used to register a new voter if he or she is not in the voter database. An administrator will check his or her information to
add him or her to the voter database or not. Then, the administrator will send an confirmation email to the voter.

![Client login Dashboard](image)

**Figure 3: Client login Dashboard**

**4.2. The registration server**

Basically, the administrator already has the voter database. If for some reasons, some valid voters’ records are not in the voter database, they can access to the voting website to download the software, install it, and register to vote. The registration information actually is sent by a voter via an email to the server. The administrator who manages this email server will check this information and add to the voter database or not. Then he or she sends an email to confirm that includes ID, PIN, and the voting software if the registration is valid. Figure 4 shows the interface that voters can fill out and send their information to the registration server.
Here is the major steps to finish the registration process.

The administrator manages the voter database through the “phpMyAdmin” website. A technician can set up this website. The administrator is able to add or delete a voter in the database (Figure 6). He or she can click “Insert” to insert a new voter, and “Delete” to delete a voter if it is necessary. For this project, the system can support up to 100,000,000 voters because the system uses the Paillier algorithm that can encrypt and decrypt the votes using Big Integer arithmetic.
4.3. The voting server

Figure 7 shows the voting server (the login server) dashboard that an administrator can use to interact with the system. The login server dashboard shows the voting start date and the voting end date that an administrator can set up to allow voters vote and finish the vote. The administrator presses the “Send Voting Emails to Voters” button to send voting information to voters through email. The email message will include the voter ID, the PIN, the voting start date, the voting end date, and the voting software.
When a voter logs in, the voter’s ID and PIN will be sent to the voting server. The voting server will access the voter database to validate the ID and PIN. If they are valid, the voting server will access the candidate database and send the list of these candidates to the voter. The process is similar to Figure 8. Then the voter will communicate with the tally server to vote for his favorite candidate.

For the candidate database, it is similar with the voter database. An administrator is able to add or delete a candidate. Here is an example of the candidate database (Figure 9).
Figure 9: List of candidates in the candidate database
4.4. The tally server

Figure 10 shows the tally server dashboard that an administrator can interact with the system. The tally server can allow multiple voters vote at the same time. The tally server dashboard has seven options. The “Start” and “Stop” button are used to start and stop the tally server. The “Reset All File” allows the administrator to delete all files in the tally server that will allow the server to be ready for another election. The “Export” button will allow the administrator export the selected results to an excel file. And then, the administrator can send the final results to the voters by pressing the “Send Final Results to Voters” button. The “Results” button shows the vote through a table as in Figure 11.
After a voter already has authorized and obtained the list of candidates, he selects only one favorite candidate. This selection and his voter ID will be sent to the tally server. On the tally server side, it uses the voter ID to check whether the voter has already voted or not. If the voter has already voted, the tally server sends a message back to the voter to let him know that he has voted already and no additional vote will be counted. The voter can vote one time only. If the voter has not voted yet, his vote will be counted and saved to the tally database. It is a file always being updated with new votes. Also, the server will send a message back to the voter to confirm that his vote has already been counted. The tally process is shown in Figure 12.

Figure 11: Result table for votes

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Number of Vote</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>john</td>
<td>terry</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>steven</td>
<td>nguyen</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ngo</td>
<td>hai</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>pham</td>
<td>hoang</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>robbert</td>
<td>daniel</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>tran</td>
<td>ha</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>pham</td>
<td>uyen</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>cuong</td>
<td>ngo</td>
<td>12492</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 12: Tally process
5. TESTING

Basically, this voting system adopts some of the important requirements of a voting system. It can be listed as follows:

Authentication: The voting server warranties only authorized voters to see the list of candidates they should be able to vote for. Only the voters are on the voters’ database will have access to the list of candidates using their voter IDs and PINs. If a voter enters a wrong ID or PIN, he or she will not get the list of the candidates to vote.

Uniqueness: The tally server checks whether a voter has already voted or not. If the voter has already voted, he or she will not be able to vote again. Both the voter ID and the vote will be sent to the tally server. The tally server checks the ID first. If the ID is on the list of IDs that already voted, the voter will get the message: “You already voted and can not vote again. This vote will not be counted. Please wait for the final result. Thank you!” If the ID is not on the list of IDs who already voted, the vote will be counted and the voter will get the message: “You just voted successfully. Thank you for your vote!”

Accuracy: The tally server records all votes to a file correctly. It has been verified that all votes are saved to a file and counted at the end of the voting process. It is assumed that there will not be any hardware, communication, and system failure during the voting process.

Integrity: If a voter has voted, the vote will be saved in a file and cannot be modified without being detected. Basically, votes are coded, encrypted, and saved under Big Integers arithmetic. The system allows the voter to vote one time only and this vote cannot be changed purposefully because the administrator does not know how Big Integers are used for encoding and encrypting votes that hide which voters have voted for
which candidates.

**Verifiability:** Each time a voter votes for his or her candidate, if the vote is submitted successfully, he or she will get a confirmation message such as “You just voted successfully. Thank you for your vote!” So voters verify that their votes are counted or not. It is assumed that there is no network failure and failure in the email system during this step.

**Auditability:** The system can be tested and demonstrated that it works with a number of voters being authorized and voted at the same time. The system was tested with more than 1,000,000 votes. It counted all votes and matched these votes to candidates. [add logging information]

**Voter Confirmation:** The voting system sends an email to the voter to confirm that he or she has already finished voting. The result of the voting is also sent to voters so that they will know which candidate is a winner via an email. The email includes an Excel file that includes candidate information, total votes, and position for each candidate. From the information in the file, a voter can find who is a winner.

The voting system was tested on five voters to receive evaluation feedback on the requirements and how voters feel about this system. Table 3 shows the results on a scale of 0 – 10, where 10 being the highest score:

<table>
<thead>
<tr>
<th>Table 3: Voting system survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voters</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>Average</td>
</tr>
</tbody>
</table>

Basically, this survey shows that the voting system needs to be improved further in future for its use. It still needs to improve GUIs and how to generate IDs and PINs for voter authentication. The integrity feature is not clear because the administrator can read the encoded and encrypted votes. Evidently he or she will be able to change encoded votes and disrupt the accuracy of the voting process. The system was tested many times and the percent for accuracy is 100% for implemented logic and algorithms.

Here is a summary for some of the disadvantages of the voting system:

1. The tally server exports the final results slowly. To calculate 13,000 votes, the system needs 40 milliseconds to finish the result.

2. To manage the database, a technician needs to install “phpMyAdmin” so that users can update the database through a browser. However, the voting system can be supported by an administrator who can build a database through the server dashboard. An administrator can control the voting server independently.

3. The options are not too diverse.

4. The GUIs are simple and easy to use but they are not professional.
6. FUTURE WORK AND CONCLUSION

6.1. Future work

This E-voting system works correctly, but it is still simple. It can implement some secure methods that can convince voters using the E-voting system instead of using traditional voting. However, to have a professional E-voting system, the following challenges should be further investigated in future work.

First, based on this material, a developer can write a plugin that voters can install in their web browsers. They can use web browsers to vote. The plugin has to ensure the use of the Paillier algorithm for tallying and SSL client/server for communication between a client and a server.

Second, applications to vote from smartphones can be built as well. There are two platforms needed to be considered to develop them: Android and iOS. Accordingly, voters will need to download the voting application and install to their smartphones, then they will be able to vote. It is much more convenient because at the present, most people use smartphones. However, cryptography algorithms and the SSL client/server protocol should be considered for such implementation. All methods should be supported to develop a professional E-voting system for such platforms.

6.2. Conclusion

This project, based on SSL client/server and Paillier encryption, implements a secure E-voting system. This combination warrants that the vote can be transferred over the Internet securely and counted correctly. This E-voting system ensures the end-to-end verification of the whole voting process. It is the quickest, cheapest, and the most efficient way to vote. Thus, if an election supports this E-voting system, voters can use it
to vote without worrying about their votes. Moreover, they can save their time by not going to a polling station to vote. All in all, the project meets important secure requirements of an E-voting system. Some other future work as mentioned can be done in further research to complete the professional voting system.

REFERENCES


APPENDIX

This appendix details how to use Keytool to generate two kinds of keys for communication between a client and a server. The next section is about phpMyAdmin to manage databases for the voting system. Using phpMyAdmin, an administrator can manage databases through any web browsers.

1. Use keytool to create a server certificate

Keytool is a tool that can be used to manage keys and certificates. A user can manage his/her public/private key pairs and associated certificates using self-authentication or data integrity and authentication services [4]. A user also can cache the public keys and certificates for the purpose of secure communication.

a. Create private key and keystore:

```
JKents-MacBook-Pro-3:~ test$ keytool -genkey -alias ftpKey -keystore privateKey.store
Enter keystore password: hoilamgi (invisible)
Re-enter new password: hoilamgi (invisible)
What is your first and last name?
[Unknown]: cuong ngo
What is the name of your organizational unit?
[Unknown]: tamucc
What is the name of your organization?
[Unknown]: tamucc
What is the name of your City or Locality?
[Unknown]: corpus christi
What is the name of your State or Province?
[Unknown]: texas
What is the two-letter country code for this unit?
[Unknown]: tx
Is CN=cuong ngo, OU=tamucc, O=tamucc, L=corpus christi, ST=texas, C=tx correct?
[no]: yes

Enter key password for <ftpKey> privatekey (invisible)
(RETURN if same as keystore password):
JKents-MacBook-Pro-3:~ test$
```
This command will generate a private key and store in the file named privateKey.store. An alias for this key is named “ftpKey”. The password for accessing the keystore file is “hoilamgi”. The password for the alias is “privatekey”.

b. Generate a temporary certificate file

    keytool -export -alias ftpKey -file certfile.cer -keystore privateKey.store
    Enter keystore password: hoilamgi (invisible)
    Certificate stored in file <certfile.cer>

c. Import this certificate into a new public keystore

    keytool -import -alias publicCertFromAl -file certfile.cer -keystore publicKey.store
    Enter keystore password: 
    Re-enter new password: 
    They don't match. Try again
    Enter keystore password: 
    Re-enter new password:
    Owner: CN=cuong ngo, OU=tamucc, O=tamucc, L=corpus christi, ST=texas, C=tx
    Issuer: CN=cuong ngo, OU=tamucc, O=tamucc, L=corpus christi, ST=texas, C=tx
    Serial number: fba6d05
    Valid from: Tue Sep 02 09:52:46 CDT 2014 until: Mon Dec 01 08:52:46 CST 2014
    Certificate fingerprints:
        Signature algorithm name: SHA1withDSA
        Version: 3

    Extensions:

    #1: ObjectId: 2.5.29.14 Criticality=false
        SubjectKeyIdentifier [ 
            KeyIdentifier [ 
                0000: 55 72 BB 77 A4 90 3B 24 26 40 E4 D8 4D 08 4B 03 Ur.w..;$@..M.K. 
                0010: 58 A9 D1 A4 X... 
            ] 
        ]

    Trust this certificate? [no]: yes
    Certificate was added to keystore
This command will create a public keystore named “publicKey.store” and the alias is “publicCertFromAl”. The password is the password for keystore named “publicKey.store”. This password should be different from private keystore password.

The following shows how to create a private keystore, a public keystore, and a certificate. For MAC OSX operating system, these files will be stored in Users folder, /Users/test (user test). The following shows how a keystore is created.

```bash
keytool -list -v -keystore privateKey.store
Enter keystore password: hoilamgi (invisible)

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: ftpkey
Creation date: Sep 2, 2014
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=cuong ngo, OU=tamucc, O=tamucc, L=corpus christi, ST=texas, C=tx
Issuer: CN=cuong ngo, OU=tamucc, O=tamucc, L=corpus christi, ST=texas, C=tx
Serial number: fba6d05
Valid from: Tue Sep 02 09:52:46 CDT 2014 until: Mon Dec 01 08:52:46 CST 2014
Certificate fingerprints:
  Signature algorithm name: SHA1withDSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
  SubjectKeyIdentifier [  
  KeyIdentifier [  
    0000: 55 72 BB 77 A4 90 3B 24 26 40 E4 D8 4D 08 4B 03 Ur.w.;$&@..M.K.
    0010: 58 A9 D1 A4 X...
  ] ]
2. Install phpMyAdmin to set up and manage database

2.1. Set up MySQL

For this project, MySQL is installed and set up on a Mac with OSX 10.9.3. Go to dev.mysql.com/downloads/mysql and download version “Mac OS X ver 10.6, DMG Archive”. Next, install all three of the components in the package. They are MySQL software packages that allow MySQL to stop/start and also allow MySQL to start when booted.

An administrator can start MySQL using the following command:

```
sudo /usr/local/mysql/support-files/mysql.server start
```

2.2. Set up phpMyAdmin on a Mac

First of all, fix socket errors with these commands:

```
sudo mkdir /var/mysql
sudo ln -s /tmp/mysql.sock /var/mysql/mysql.sock
```

Next, download phpMyAdmin (http://www.phpmyadmin.net/ home_page/downloads.php), unzip the file, and move it into your Sites folder. Make the config folder

```
mkdir ~/Sites/phpMyAdmin/config
```

Then change the permissions

```
chmod o+w ~/Sites/phpMyAdmin/config
```

Finally, go to the phpMyAdmin localhost URL in the browser and run the phpMyAdmin set up: localhost/~username/phpMyAdmin/setup/
To create a database server, click “New Server”, a new page will load and click on the “Authentication” Tab to set up MySQL root user and password, and save them. Make sure MySQL is running by using the following command:

```
sudo /usr/local/mysql/support-files/mysql.server start
```

Now, an administrator can go to his/her phpMyAdmin url (localhost/~test/phpMyAdmin) on the browser to manage his/her database.
Figure 14: Home page of phpMyAdmin database