Securing E-Voting Integrity Using Homomorphic Encryption and Chaum-Pederson Zero Knowledge Proof

Nazhif Hilmi Kistijantoro - 13525115
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: hilmikistijantoro@gmail.com , 13525115@std.stei.itb.ac.id

***Abstract*—Voting systems are the most used method in making a decision in a democratic context. It is versatile, robust, and arguably fair. The most promising method of implementing these systems is by holding the voting entirely electronic and online (e-voting). But, ensuring the integrity and security of an e-voting system is a difficult problem that makes these systems still not widely used for major practices. This paper demonstrates how e-voting systems can be implemented by using homomorphic encryption and the Chaum-Pedersen protocol. This combination can guarantee the data integrity for both the voters and conveners while still ensuring voter anonymity.**

Keywords—component; e-voting; chaum-pedersen; zero knowledge proof; homomorphic encryption;

I. Introduction

Voting systems are the most used method in making a decision in a democratic context. It is versatile, robust, and arguably fair. These systems have been implemented throughout history in many various ways, such as by live voice or paper ballots. But, as voting participants increase and voting complexity grows, such methods are infeasible. With the increasing development of technology, a new proposed method emerged. That is, by making the whole voting system electronic and online. However, ensuring that the system meets security requirements is proven to still be a challenge that makes electronic voting (e-voting) not widely used for large scale elections [1].

In a traditional voting scheme using paper ballots, voters can observe the whole voting processes and therefore they can be convinced that the processes are executed honestly. For example, at first the voter is given a blank ballot, makes his choice on the ballot, then puts it into the ballot box [1]. In this step the voter himself can verify his privacy, since he makes his choice in a private space, alone, and his identity is not written in the ballot. After the voting period ends, the votes are counted by authorities in front of the public [1]. Here again, the voters can be convinced that the authorities are behaving honestly because the counting process can be observed publicly.

But, in an electronic voting scheme, gaining the trust of all parties is not as easy as using paper ballots. At any steps in the e-voting process, results can be manipulated if there is a lack of security and cryptography [1]. Therefore, the challenge for e-voting systems is to make sure that the system meets all the necessary security requirements and convince all parties that the system does indeed satisfy them.

The security requirements to be expected for any voting system is as follows [1]:

1. Voter Privacy: A vote must not be able to be traced back to the identity of the voter. The anonymity of a particular vote must be preserved even after the voting process is finished [2].
2. Eligibility: Every voter must be authorized and eligible [3].
3. Uniqueness: Each voter can only have one vote [4].
4. Fairness: No parties, both the voters and conveners, are able to know the result completely or partially before the counting process [5].
5. Uncoercibility: Any party must not be able to coerce the votes.
6. Receipt-freeness: Voters must not be able to absolutely prove the content of their vote [6]. This requirement is to prevent any coercion and bribery.
7. Accuracy: Voting result must be accurate that the result cannot be manipulated, added, or deleted by invalid action.
8. Universal Verifiability: Any observers can verify that their vote and the total vote is counted honestly.

To meet these security requirements, researchers have made many techniques in cryptography. One of them is by using a combination of homomorphic encryption and the Chaum-Pedersen protocol. The former is used to encrypt the individual ballot, while the latter is used to convince both the voters and the conveners that the voting is done fairly and honestly.

II. THEORY

A. Homomorphic Encryption

In the past, there was a major limitation of processing on encrypted data. That is, that the data must be decrypted first before being operated on. This limitation creates significant privacy vulnerabilities since there must be a time period where the data can be seen fully.

It was not until 1978 that a paper proposing the concept and mathematical properties of homomorphic encryption was released [7]. The paper defined homomorphic encryption as an encryption that permits a computer to operate on data without decrypting it first, or more formally [7]:

$$E(f(a,\ b))\ =\ f'(E(a),\ E(b))$$

where a and b are the plaintext data, E is the encryption function, f is the mathematical operation function (for example multiplication or addition), and f' is the function to compute f on the encrypted data.

It is obvious that such encryption schemes have inherent limitations, which is that comparison operations cannot be used or else guessing the decrypted value can be done using binary search [7]. However, modern cryptography has created fully homomorphic encryption (FHE) that supports comparison by making the result remain encrypted, preventing an evaluator from gaining any information when performing binary search. Regardless of which type of homomorphic encryption it is, this scheme of encryption provides a powerful and secure way of processing data.

1) Partially Homomorphic Encryption (PHE)

   The earliest implementations of homomorphic encryption are partially homomorphic encryption, which is an encryption system that only supports one type of mathematical operation, either multiplication or addition. This system, specifically the exponential ElGamal encryption, will be the foundation of the e-voting system that will be laid out in this paper.

2) Fully Homomorphic Encryption (FHE)

   For many years, homomorphic encryptions can only support one type of operation. In 2009, Craig Gentry solved this by making fully homomorphic encryption that supports both multiplication and addition [8].

B. Exponential ElGamal Encryption

The exponential ElGamal encryption is a variation on a more general ElGamal encryption. Therefore, an understanding of the traditional ElGamal encryption is first needed, hence the early part of this subchapter is dedicated to the traditional method.

The ElGamal encryption is a scheme that laid upon the preexisting Diffie-Hellman scheme [9]. Suppose that a person A has a private secret key x_A and a person B has a private secret key x_B. Let p be a large prime number and g is a primitive element of mod p. Both p and g are publicly known. Then, they can compute the public key:

$$y_A\ =\ g^{x_A}\ mod\ p$$

$$y_B\ =\ g^{x_B}\ mod\ p$$

Here is where the ElGamal scheme differs from Diffie-Hellman. Suppose A wants to send a message m to B, then A will generate a random k such that $0 \leq\ k\ \leq\ p\ -\ 1$. Using k, computes the ElGamal "key":

$$K\ =\ y_B^{\ k}\ mod\ p$$

Using K, computes the tuple (C_1, C_2), which is the encrypted message that can be posted publicly:

$$C_1\ =\ g^k\ mod\ p$$

$$C_2\ =\ (K\ .\ m)\ mod\ p$$

With this scheme, it will be hard for people who don't know the value of x_B to solve the tuple for m, but for B it will be very easy:

$$K\ =\ (g^{x_B})^k\ \ mod\ p\ =\ (g^k)^{x_B}\ mod\ p\ =\ C_1^{\ x_B}\ mod\ p$$

since K is known, finding m is just a matter of solving the equation C_2 for m, which can be done by multiplying both sides with the inverse of K.

The ElGamal encryption is a multiplicative partially homomorphic encryption and doesn't provide the needed additive homomorphic property for counting votes. Hence, a modification is done by changing m in the equation into $g^{m_{new}}$ where m_{new} is the secret message, or specifically the number of votes. This modification is called the exponential ElGamal encryption. Its additive homomorphic property is shown here (for simplicity purposes, m_{new} is written just as m_i):

Suppose we have voters: A, B, and C. Each of their vote is encrypted as $(C_{1A},\ C_{2A})$, $(C_{1B},\ C_{2B})$, and $(C_{1C},\ C_{2C})$ respectively, y is a public key that is derived from a distributed secret key x. Then,

$$(C_{1Total},\ C_{2Total})\ =\ (C_{1A}\ .\ C_{1B}\ .\ C_{1C},\ C_{2A}\ .\ C_{2B}\ .\ C_{2C})$$

$$=\ (g^{k_1+k_2+k_3}\ mod\ p,\ y^{k_1+k_2+k_3}\ .\ g^{m_1+m_2+m_3}\ mod\ p)$$

or, more generally,

$$(C_{1Total},\ C_{2Total})\ =\ (g^{\Sigma k_i}\ mod\ p,\ y^{\Sigma k_i}\ .\ g^{\Sigma m_i}\ mod\ p)$$

Using x, it is possible to derive $g^{m_{total}}$. As long as the total count vote is not significantly large, m_{total} can be feasibly computed.

C. Chaum-Pedersen Protocol

Chaum-Pedersen protocol is a protocol in which you can prove that you know a secret key x such that,

$$y = g^x \bmod p \text{ and } z = h^x \bmod p$$

without revealing the value of x itself. Hence, this protocol is a type of zero knowledge proof, which is an intensely researched field in cryptography.

As you may guess, the structure of A and B is similar to the tuple (C_1, C_2) in the exponential ElGamal encryption (note that the variables mentioned in this subchapter are independent of the previous subchapters). This observation provides a hint that this protocol can be used along the exponential ElGamal encryption. A more detailed explanation will be laid out on Section III.

A zero knowledge proof is needed for a voting system so that the conveners can know whether a voter's vote is valid while not revealing the vote's content. Furthermore, this protocol is also needed to prove that the tallies are counted correctly and that no foul play is being done by the conveners.

The initial state of the protocol is as follows: [10] Let q and p be a prime number in which q divides $p - 1$. G_q is then defined as a group of order q in which all the elements are from the multiplicative group of integers modulo p (more concisely, G_q is a subgroup of $\mathbb{Z}^*_p$). The number $g \in G_q$ is a generator for the group G_q, and the number h is a member of G_q. The variables that are publicly known are h, g, p, q. This restriction of all the public variables are required to ensure the security of the proof.

Now, the protocol is as follows:

1) The prover chooses a random number $0 \leq s \leq q - 1$ and sends the pair $(a, b) = (g^s, h^s)$.
2) The verifier sends a challenge $c \in \mathbb{Z}_q$ to the prover.
3) The prover computes $r = s + c.x$.
4) The verifier check the equality

$$g^r = ay^c \text{ and } h^r = bz^c$$

the verifier accepts the proof if the equality holds.

D. Disjunctive Chaum-Pedersen Protocol

In the e-voting system, a vote will be represented as the encryption of g^0 for negative vote and g^1 for a positive vote (g is the base used in the ElGamal encryption ballot). Therefore, the conveners must be able to verify whether the voter's vote is either one without revealing the real content of the vote.

The standard Chaum-Pedersen protocol only allows for one specific value of x such that $y = g^x \bmod q$ and $z = h^x \bmod q$ equality holds. This makes the standard protocol not viable for this e-voting system. A modification is needed.

To modify the protocol, we can use a paper released in 1994 that shows how to make an honest verifier zero knowledge (HVZK) such as Chaum-Pedersen protocol to be able to prove an OR statement [11]. A couple of new definitions are needed for this conversion.

A simulator is a theoretical algorithm that can produce fake conversations of the protocol that are indistinguishable from a real one. An instance is a problem from a set of potential problems that will be proved. For example, a problem S_1 is an instance from a set of potential problems $S_1 \vee S_2$.

The initial state of the new disjunctive Chaum-Pedersen protocol will be stated. The tuple of the form (g, h, u, v) is a problem that satisfies $u = g^x \bmod p$ and $v = h^x \bmod p$, where x is the secret key. Let there be two instances of a problem (g_0, h_0, u_0, v_0) and (g_1, h_1, u_1, v_1). The prover is considered truthful if they know x such that one of the two problems is satisfied. For simplicity purposes, assume that the prover know x_0, which is the solution for the instance (g_0, h_0, u_0, v_0). The protocol is then executed as follows:

1) For the instance that the prover knows the solution of, he will choose a random s and computes

$$(a_0, b_0) = (g_0^s \bmod p, h_0^s)$$

for the other instance, he will make a simulator by choosing a challenge c_1 and a random response r_1 ahead of time. He will then compute

$$(a_1, b_1) = (g_1^{r_1}. u_1^{-c_1} \bmod p, h_1^{r_1}. v_1^{-c_1})$$

The prover sends (a_0, b_0) and (a_1, b_1) to the verifier.

2) The verifier sends the overarching challenge c completely at random.
3) The prover creates a separate challenges that satisfies

$$c_0 + c_1 = c \bmod q$$

where q is the order of the group G_q explained on the last subchapter. Now, using the real secret x_0, the prover computes the real response r_0

$$r_0 = s + c_0. x_0 \bmod q$$

The prover sends (r_0, r_1, c_0, c_1).

4) The verifier first verifies if the challenges are made honestly by checking

$$c = c_0 + c_1 \bmod q$$

and then checks the validity for the instance 0:

$$g_0^{r_0} = a_0 \cdot u_0^{c_0} \bmod p$$

$$h_0^{r_0} = b_0 \cdot v_0^{c_0} \bmod p$$

and then checks the validity for the instance 1:

$$g_1^{r_1} = a_1 \cdot u_1^{c_1} \bmod p$$

$$h_1^{r_1} = b_1 \cdot v_1^{c_1} \bmod p$$

The verifier accepts the proof if the equations are satisfied for both instances.

III. Implementation

In this section, the proposed e-voting scheme will be explained in detail. We will explore how the system guarantees voter anonymity and data integrity in every operational phase using cryptographic techniques such as homomorphic encryption and Chaum-Pedersen protocol. Note that the variables defined in this section are independent from their definitions in the previous section.

E. Initial Conditions

Before the e-voting starts, the system must establish a couple cryptographic paramaters. First, the system selects a large prime number p and a generator g for a subgroup of $\mathbb{Z}_p^*$ of order q. p, g, and q are publicly known both for the conveners and the voters.

A secret master key x must also be generated. To prevent a single corrupt trustee from rigging the election, the secret x must be distributed such that:

1) A set of n independent trustees generate their own private shares x_i independently. The trustees selected are preferably a competitor to each other.
2) They combined their private keys to create public key h. h is known both for the conveners and voters. This public key is the equivalent of $g^{\Sigma x_i}$.
3) To decrypt anything, a minimum t out of n trustees must cooperate.
4) The master key x (or more accurately $\sum x_i$) never exists in its entirety on any single machine on any phase of the e-voting scheme.

This secret master key x can be generated using a distributed key generation (DKG) protocol and threshold cryptography. The protocol is outside the scope of this paper. For more detail, please refer to [12] and [13].

F. Ballot Construction, Encryption, and Validation

For simplicity purposes, we will first assume a "Yes/No" election. A "Yes" is encoded as $v = 1$, while "No" is encoded as $v = 0$. We will later show how to make this scheme support a multiple candidate election later in this subchapter.

Now, using the Exponential ElGamal Encryption, a voter will choose a random r and construct the encrypted ballot by computing

$$B = (C_1, C_2) = (g^r \bmod p,\ g^v \cdot h^r \bmod p)$$

A malicious voter might be able to rig the election by creating a ballot using an inflated v value (e.g. $v = 100$). To prevent such attacks, the conveners must be able to verify that a vote is valid without knowing the content of the vote itself. This need can be achieved using the disjunctive Chaum-Pedersen protocol.

In this scenario, the problem instance where $v = 0$ is,

$$C_1 = g^r \bmod p \text{ and } C_2 = g^0 \cdot h^r \bmod p = h^r \bmod p$$

while the problem instance where $v = 1$ is,

$$C_1 = g^r \bmod p \text{ and } C_2 = g \cdot h^r \bmod p$$

or,

$$C_1 = g^r \bmod p \text{ and } C_2 \cdot g^{-1} = h^r \bmod p$$

where g^{-1} is the modular multiplicative inverse of g modulo p

The process then will have the same steps as the steps described in Section IID, where the voter acts as the prover and the convener acts as the verifier. This protocol can actually be made non-interactive using Fiat-Shamir heuristic, which replaces the interactive verifier with a cryptographic hash function [14].

Now, suppose we have m candidates in the election. A ballot B can be constructed as follows:

$$B = (B_1, B_2, \ldots, B_m)$$

where $B_j = (C_{j,1}, C_{j,2})$ for each candidate $j \in \{1, 2, \ldots, m\}$. The validation of the vote will be done by 1) validating each B_j using the disjunctive Chaum-Pedersen protocol, 2) multiply all B_j (which corresponds to addition of their plaintexts) and validate that the total v is equal to one using the standard Chaum-Pedersen protocol.

G. Counting Phase

Since the exponential ElGamal encryption used in the ballot is a partial homomorphic encryption, the process for counting is rather straightforward, that is:

$$B_{j,total} = \left(\prod_{l=1}^{k} C_{l,j,1},\ \prod_{1}^{k} C_{l,j,2}\right)$$

where k is the total number of voters. Because of this operation, $B_{j,total}$ will have the form $(g^{\Sigma r}, g^{\Sigma v} \cdot h^{\Sigma r})$. Notice how $g^{\Sigma v}$ can be derived if the master key x is known.

H. Decryption and Final Verification

Since the master key x is distributed using a distributed key generation (DKG) protocol and threshold cryptography, the ballot will be decrypted using the partial threshold decryption:

1) Using their own x_i, t trustees will compute $S_i = C_{j,1}^{x_i}$
2) To ensure that the trustee truly use their real x_i, the trustee will post their public key $h_i = g^{x_i}$. To check the integrity of the trustee, we first multiply all h_i and check if it is equal to the master public key h. After that, the Chaum-Pedersen protocol is used to prove that the trustee uses the real x_i such that $h_i = g^{x_i}$ and $S_i = C_{j,1}^{x_i}$.
3) Once t valid partial shares are collected, the full blinding factor can be constructed by using Lagrange interpolation coefficient (λ_i):

$$\prod_{i=1}^{t} (S_i)^{\lambda_i} = (C_{total,j,1})^{\Sigma \lambda_i x_i} = (C_{total,j,1})^{x} = h^{\Sigma r}$$

4) Now that $h^{\Sigma r}$ has been computed, $g^{\Sigma v}$ can be derived by,

$$C_{total,j,2} \div h^{\Sigma r} = g^{\Sigma v}$$

The reason why the Chaum-Pedersen protocol is only used to prove that the trustees uses their real x_i is because the tallying process can be done by everyone. The vulnerabilities on the system is only if the trustees uses a fake x_i to manipulate or break the final tally.

IV. Analysis

The e-voting scheme proposed in this paper uses the exponential ElGamal encryption and the Chaum-Pedersen protocol zero knowledge proof to ensure the integrity for both the conveners and the voters. This section provides a detailed analysis of how effectively this architecture satisfies the fundamental security requirements mentioned on Section I:

1) Voter privacy: Satisfied. As long as r is kept secret by the voter, anyone else who doesn't know r will not be able to trace the vote's content. Even the conveners or the system who verifies the validity of the vote knows nothing of the content itself because of the zero knowledge properties of the Chaum-Pedersen protocol.
2) Eligibility: Out of scope. The e-voting framework mentioned in this paper doesn't account for the eligibility of the voter. It is already assumed that everyone who constructs a ballot is already authenticated. An authentication layer that is built on top of this scheme is needed to satisfy this requirement.
3) Uniqueness: Satisfied. Every vote has a weight of exactly one. A malicious voter cannot create a ballot with a weight of more than one because of the verification process that used the standard and disjunctive Chaum-Pedersen protocol.
4) Fairness: Satisfied under the assumption the threshold of honest trustees is maintained. If t number of trustees decided to collude and decrypt the tally count before the counting phases finished, this requirement will be broken.
5) Uncoercibility and receipt-freeness: Unsatisfied. A voter can prove the content of their vote by showing their value of r. The coercer then can check whether $g^1 . h^r = C_2$. Because the receipt-freeness is unsatisfied, the uncoercibility requirement is then by consequence unsatisfied.
6) Accuracy: Satisfied. The homomorphic property of the exponential ElGamal encryption is mathematically proven ensuring its accuracy.
7) Universal verifiability: Satisfied. Because all the ballots posted can be seen publicly, anyone can check whether the votes are counted correctly. The voters can also verify if the total count is decrypted honestly by using the Chaum-Pedersen protocol.

Even though this scheme is very viable to be implemented on a small scale, this scheme carries an inherent problem from an operational point of view. That is, the decrypted form of the total count is $g^{\Sigma v}$, and not $\sum v$. Therefore, to get the total number of votes, the system must solve a discrete logarithm problem that is computationally hard to solve. Of course, this would be okay if the total votes are still in the single-digit millions, but this will become a problem if the total votes can reach into a hundred million.

V. Conclusion

The e-voting scheme using homomorphic encryption and Chaum-Pedersen protocol has actually been implemented in real world settings [15]. This shows that this scheme is computationally feasible and applicable. Our analysis has also shown that this scheme has satisfied many of the core elections requirements. However, there are many limitations that prevent this scheme unviable for large scale elections.

Specifically, the fact that the system must solve a discrete logarithm problem to find the total votes, hence will be a bad option with a significant number of votes. And more importantly, is that this scheme is still not able to satisfy the receipt-freeness and uncoercibility requirements.

Future research should focus on integrating this scheme with an authentication layer and an anti-coercion mechanism, and to explore other e-voting schemes that can satisfy all the requirements while still being computationally feasible.

VI. Acknowledgement

The author would like to thank first and foremost to God for all His blessings and guidance throughout the process of writing this paper. Second, to my family who have supported me through all my learning journey. Also, I am grateful to Prof. Dr. Ir. Rinaldi, M.T who has shared his knowledge that makes the writing of this paper possible. Lastly, I would like to thank all my friends who shared the same journey with me in the entire semester.

References

[1] Çetinkaya, O. (2009, March). Cryptography in electronic voting systems. In International Conference on eGovernment and eGovernance (pp. 297-310)..

[2] Cranor, L. and R. Cytron (1997), “Sensus: A security-conscious electronic polling system for the Internet”, Hawaii International Conference on System Sciences, Hawaii.

[3] Burmester, M. and E. Magkos (2003), “Towards secure and practical e-elections in the new era”, Chapter in Information Security - Secure Electronic Voting, Kluwer Academic Publishers, pp. 63-76.

[4] Forsgren, O., U. Tucholke, S. Levy and S. Brunessaux (2001), “Report on electronic democracy projects, legal issues of Internet voting and users (i.e. voters and authorities representatives) Requirements Analysis”, European Commission CYBERVOTE Project, D4 vol. 3.

[5] Aditya, R., B. Lee, C. Boyd and E. Dawson (2004), “Implementation issues in secure e-voting schemes”, The 5th Asia-Pacific Industrial Engineering and Management Systems Conference, Goldcoast, Australia.

[6] Benaloh, J. and D. Tuinstra (1994), “Receipt-free secret-ballot elections”, In Proceedings of the 26th ACM Symposium on the Theory of Computing, pp. 544-553, 1994.

[7] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. Foundations of Secure Computation, 4(11), 169–180.

[8] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, 169–178. https://doi.org/10.1145/1536414.1536440.

[9] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4), 469–472.

[10] Chaum, D., & Pedersen, T. P. (1993). Wallet databases with observers. In E. F. Brickell (Ed.), Advances in Cryptology — CRYPTO ’92 (pp. 89–105). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-40064-8_6

[11] Cramer, R., Damgård, I., & Schoenmakers, B. (1994). Proofs of partial knowledge and simplified design of witness hiding protocols. In Advances in Cryptology—CRYPTO ’94 (pp. 174–187). Springer.

[12] Pedersen, T. P. (1991). Non-interactive and information-theoretic secure verifiable secret sharing. In Advances in Cryptology — CRYPTO ’91.

[13] Pedersen, T. P. (1991). "A threshold cryptosystem without a trusted party." *Advances in Cryptology — EUROCRYPT ’91*.

[14] Fiat, A., & Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology — CRYPTO ’86 (pp. 186–194). Springer, Berlin, Heidelberg.

[15] Adida, B. (2008). Helios: Web-based Open-Audit Voting. In Proceedings of the 17th USENIX Security Symposium (pp. 335–348).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2025

Nazhif Hilmi Kistijantoro, 13525115