

**《形式语言与计算理论》**

**课程报告**

**题目：时间复杂度应用：投票系统 .**

**姓名：NARANJO VILLOTA JOSE LUIS**

**学号： B20200692 .**

**专业： 计算机科学与技术 .**

**学院： 计算机与通信工程 .**

**成绩：**

**任课教师： 段世红 .**

**2020年11月30日**

**Time Complexity Application: A voting System**

**Overview**

The improvement of the computing capabilities of everyday devices suggests that soon there will be among us machines based on quantum computing, capable of executing complex tasks in a matter of seconds. This idea triggers the alarm of security as a big challenging problem for some of the applications of daily use, therefor in this document is introduced an alternative solution to the existing electronic voting systems.

The important aspect of this report is to analyze according to time complexity , the asymptotical behavior of the proposed scheme which incorporates an advanced encryption system which has homomorphic properties that allow performing arithmetic calculations on encrypted text, the system’s security (BGV) relies on a difficult mathematical problem of lattices theory called Learning with Errors (LWE), which was proved to be safe against quantum computer-based attack.

This system is a different solution to traditional paper-based systems that are very expensive, slow and inefficient, yet provides privacy, integrity, verifiability, and correctness which guarantee transparent processes. Here in we will show how do we achieve an acceptable asymptotically.

**Introduction**

To provide a voting system solution that bases its security on Lattice Theory (LWE, and RLWE) is the main goal of this report. Up to now the security of different schemes is based on mathematical problems which are considered hard to be solved. Unfortunately, those schemes might become unsafe if the quantum computation becomes commercially available. In 2016 the National Institute of Standards and Technology (NIST) proclaims the starting of the development of post-quantum cryptography standards. They stated that cryptosystems such as Diffie-Hellman key exchange algorithm based on the Discrete Logarithm Problem or Elliptic Curves are vulnerable. So, it is urgent to find a way to substitute them as well.

This solution enhances the security and provide the confidence for the user to trust the system and therefore accept the final results without any protest.

Furthermore, the system was designed to provide various security properties such as [9]:

**Privacy and Confidentiality**: The identity of a voter cannot be linked with his vote

**Authentication and Authorization:** Check the credentials of those attempting to cast a vote and allow only those have registered before. Voters only vote once.

**Integrity:** The result of the election cannot be altered in any way.

**Robustness:** Provide a strong system that even with some authorities misbehaving the system truthfulness cannot be compromised.

**Verifiability:** Voters can verify the integrity of their casted vote and that it has been tallied.

**Availability:** the system must be accessible and usable upon demand even while suffering from network attacks.

The system uses the Plurality Voting Method (PVM) [10] for determining the election winner.

**The BGV Cryptosystem**

In this section, some basic ideas related to the BGV cryptosystem are introduced. The (levelled) Fully Homomorphic Encryption Scheme based on GLWE (General-LWE) without Bootstrapping provided use a parameter *L* indicating the number of levels of arithmetic circuits than can handle, a security parameter , and a bit b to determine if the setting parameters corresponds to the LWE-based scheme or a RLWE-based scheme.

The Schema is made up by the following functions:

A. FHE.Setup ()

B. FHE.KeyGen ({})

C. FHE.Enc (params, pk, m)

D. FHE.Dec (params, sk, c)

E. FHE.Add ()

F. FHE.Mult ()

G. FHE.Refresh ()

It is crucial to determine the parameters L in order to set up the system according to the degree of the homomorphism that will be needed in this application, for example, the most complex function to be computed is the calculation of the Hamming weight, which at the same time will depend on the number of voters that take part in the election process.

The noise hardly increases when addition operations take place so there is no need to refresh the ciphertext constantly.

The data will be publicly computed. The parameters of each district are selected according to the number of eligible voters and other considerations analyzed in the following sections.

**The Proposed Scheme**

In this section is described with detail the full process of the system as well as the entities and the roles of each one.

The system guarantees the transparency of the process by having information publicly know all the time, however the published information cannot be modified under any circumstance by anyone.

The proposed scheme will be structured as:



Fig. 1 Block Diagram of the System

These are the entities involved in the system and acronyms used in the flow diagrams:

1) *Voter (VOT):* The authorized person to cast a vote during the elections.

2) *Authority (AUT):* A trustable person that is in charge to supervise, verify and control that the election process takes part in an integral way.

3) *Public Bulletin Board (PBB):* The entity which is in charge of publishing in real time the information, including voters that have been successfully authenticated by the authorities, and had cast a vote.

4) *Digital Signature Algorithm (DSA):* Is the software that provides several services to guarantee the authenticity of the involved entities.

**A. System Configuration**

This is the first step, the people who have been selected as authorities decides the parameters that HELib uses to generate the public and private keys.

When the election is held in various regions, is needed a central server. In that case, the local servers need to be authenticated by the central server and generate a public-private key pair to sign and send to it one by one the valid ballots of the local voters encrypted with the central server public key.



Fig. 2 Flow Diagram of the System Configuration Block

**B. Users Registration**

Once the System is configured, the voters can be register in it. Each voter presents to the authority his ID card or use any biometric method to prove his identity, after a successful authentication the authority provides him with the media to generate a key pair for the digital signature; This guarantees that later anyone can verify the authenticity of each casted vote, an unsuccessfully authenticated voter cannot generate a public-private key pair. In that case, even if an intruder gets access to the system and submits a vote, it is not tallied.

**C. Ballot Processing**

This block is the most complex process in the system, it has different parts executed successively since the moment a voter selects a candidate until it is tallied to be computed in the final stage.

To make it simpler it is explained separately:

1) *Ballot Casting:* The voter can select the candidate of their preference and conceal the vote, if the voter doesn’t like any of the candidates, he can choose to avoid giving his vote to any of them.



Fig. 3 Signed Encrypted Ballot E(vote)

Fig 3, shows sent data after the vote has been cast, as can be seen, it is a list of encrypted values and the digital signature of the voter as a whole packet of data.

2) *Ballot Encryption:* After the voter has selected any of these choices, the voter uses the common public key of the system to encrypt the vote and then sign it using his private key. The information is published by the PBB to allow anyone interested in verifying (authorities, the voter or even the candidates).

3)  *Ballot Verification:* The system verifies the authenticity of the voter by using the signature that comes with the submitted ballot if the vote is authentic is then taken to the classification stage if not is immediately discarded.

4) *Ballot Classification:* A verified vote is classified as a valid, blank or null, for that purpose is calculated the Hamming Weight (HW) to determine to which class the vote belongs to:

a) Valid Vote: A vote with only one choice, when the HW must be one.

b) Blanc Vote: If the voter does not select any option. The HW is zero.

c) Null vote: If the voter selects more than one choice. The HW is greater than one.

Each candidate has their own list of encrypted votes, and if is a valid ballot, it will be added to the list of each one, one encrypted value, there are two extra lists, one for the null votes and one for the white votes, at the end of the election the length of all the list will be the same.



Fig. 4 Flow Diagram of the Ballot Classification process

**D. Results Computation**

In this stage is calculated the function that calculates the HW of the list of encrypts values of each candidate. The value obtained by adding all the encrypted values of the casted ballots that comes in the first position, will correspond to the number of valid votes obtained by the candidate located at the beginning of the ballot.

This stage will be performed for the central server in the case that multiple servers took part in the election process.



Fig. 5 Flow Diagram of the Results Computation

**The Proposed Scheme’s Algorithms**

In this section are presented some of the algorithms implemented in the proposed solution; and some features of the system are explained.

Code Segment 1 shows the implementation of the classification of the ballots after being checked the authenticity of each, this process will be executed in the server that provided the encryption keys in the case that are more than one server.

In this section is the first time that is needed the homomorphic capabilities of the system. Thus, because to classify is necessary to calculate the total sum of the votes contained in each ballot, the classification was implemented according to the following code:

**Code Segment 1:** Ballot Classification

**Input:** List of Candidates

**Output:** Classification of the Votes

1. **QList**<**Ctxt**> lstData;
2. **vector**<**Ctxt**> vc(numCandidates,c0); //vector of ciphertext
3. **int** k = 0 ;
4. **foreach**( Candidate c, ballot.lstCandidatos){
5. **ZZX** plainvote = to\_ZZX((c.vote));
6. publicKey.Encrypt(vc[k], plainvote); // ballot encryption
7. k++; }
8. c0 = calculateHammingWeight(vc);
9. **int** sum = DecryptCipheredValue (c0, ea, secretKey );
10. **if** (sum == 1){
11. **for**(**int** i=0; i< numCandidates ;i++){
12. **Candidate** a = lstCandidates.takeFirst();
13. a.LstVotes.append(vc[i]);
14. lstCandidates.append(a); }
15. whiteCandidate.LstVotes.append(v0);
16. nuleCandidate.LstVotes.append(v0);
17. }**else** **if** (sum == 0){
18. **for**(**int** i=0; i< numCandidates ;i++){
19. **Candidate** a = lstCandidates.takeFirst();
20. a.LstVotes.append(vc[i]);
21. lstCandidates.append(a); }
22. whiteCandidate.LstVotes.append(v1);
23. nuleCandidate.LstVotes.append(v0);
24. }**else** **if** (sum >1) {
25. **for**(int i=0; i< numCandidates ;i++){
26. **Candidate** a = lstCandidates.takeFirst();
27. a.LstVotes.append(v0);
28. lstCandidates.append(a); }
29. whiteCandidate.LstVotes.append(v0);
30. nuleCandidate.LstVotes.append(v1); }

Code Segment 2, presents the Computation of the HW; its practical when the number of elements is few because the noise associated with the sum can be dismissed, yet, the algorithm segment 1, show another way to perform the operation without increasing the homomorphic level.

**Code Segment 3:** Computing Hamming Weight

**Input:** QList (List of Ciphered Values)

**Output:** Integer HW (Hamming Weight Value)

1. **Ctxt** calculateHammingWeight (**QList**<**Ctxt**> values ){
2. **Ctxt** ctSum = values.takeFirst();
3. **foreach**( **Ctxt** value , values){
4. ctSum += value; }
5. **return** ctSum;}

In the Algorithm Segment 1, is calculated the sum of the encrypted values, notice that at the end of line four is added an encryption of cero to make an even number of values in the vector, the sum up of this will be managed for the encryption function, the BGV cryptosystem will call the refresh function first and apply it to the encryption of the zero added as many times as the two entrance are encrypted with the same level subkey.

**Algorithm Segment 1:** Addition

**Input:** V (Vector of Ciphered Values)

**Output:** Integer HW (Hamming Weight Value)

1. R [] =0
2. L= V.length
3. **if** L odd **do**
4. V.addEnd(E (0))
5. **else**
6. **for** i=1 **to** L/2 **do**
7. R[i] = .Add ()
8. **end end**
9. **Return** R[] **end**

At the beginning of an election process, the Authorized Authorities determine the parameters needed in the Setup function of the leveled FHE scheme of BGV cryptosystem and give the parameters to the PBB to be generated the public common keys.

**Security and Performance Analysis**

Here are shown the analysis of both: the security and the asymptotical performance, under determined circumstances.

First are mention some security aspects for the system and later an analysis of the security of the relying problem on which the BGV cryptosystem is based:

1. Eligibility of Voter: The system will require to sign the ballot with the voters’ private key:
2. This information is published immediately by the PBB, and can be verified the cast of a vote illegally.
3. Multiple Voting Detecting: The submission of more than one vote is detected easily because the PBB publishes cast votes on line.
4. Voter Verifiable: Since the PBB publishes everything, everyone can verify the authenticity of any encrypted data, and the final results, can be computed publicly.

**Security of the Encryption**

Because the submitted ballot contains the voters’ preferences in the election process, to keep its privacy is necessary to remain encrypted during the whole process, the BGV cryptosystem can handle the computation of the HW weight over encrypted data.

The problem is to distinguish between:

1. Samples uniformly from , and
2. Draws uniformly, then samples by sampling:

, (gauss distribution),

Then setting

Thus, the problem is infeasible.

Is suspected that GLWE is hard for any, recall is an integer dimension and is a power of two such that [12], where B is a bound (with overwhelming probability) on the length of elements output by. For fixed, perhaps GLWE gradually becomes harder as increases, whereas increasing is probably often preferable for efficiency.

There for the security in contrast with the system proposed by Yang et. al. has been improved dramatically, because that system was based on the exponential ElGamal cryptosystem.

**Performance of the Prototype**

The asymptotical performance is analyzed according to:

Processor: Intel® Core™ i3-7100T CPU @ 3.40GHz

Memory (RAM): 2.00GB

System type: 64-bit OS, x64-based processor

Operating System: Ubuntu Desktop 18.04.1

To determine the HW computation, and other critical points the following test were run:

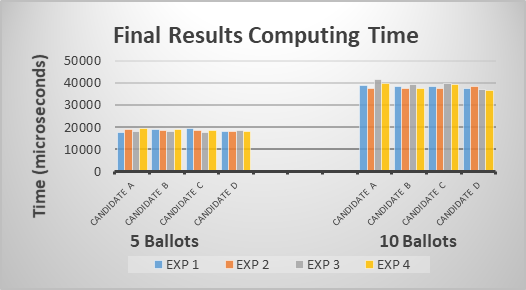


Fig.6 Computing time for a test of 5 and 10 voters using ballots containing 4 candidates

From Fig.6 can be seen how in the first encrypting the time reaches (with 5ballots), a maximum peak of approximately 13.5 ms later gets stable around the value of 8 ms. The second critical part is the calculation of the final results, there were run several tests with 5 and 10 ballots, the results show that the time required to calculate also is duplicated, but remains constant while computing final results.

Fig.7 shows the computing time needed to calculate the final results which does not depends on the value that is encrypting the ciphertext used.

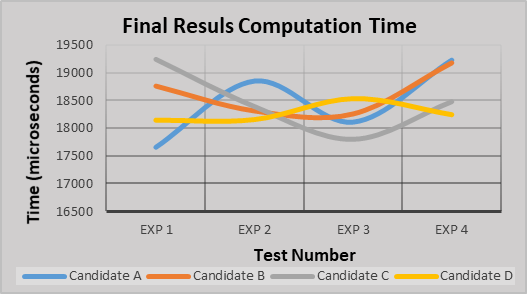


Fig. 7 Final results computing time (5 ballots of 4 candidates)

In the table 1, is shown the size of the public and private key, as well as the signature, produced by the signing system used by the DSA, the parameter, is the security parameter used.

Table 1: Properties of the Lattice-Based Signature Scheme [7]

|  |  |  |  |
| --- | --- | --- | --- |
| **Scheme** | **Sign size** | **Pk size** | **Sk size** |
| DS\_LWE | O(*n log n*) | O(*n log n*) | O(*n log n*) |

If the system is instantiated with the RLWE it can evaluate L-lev circuits with per-gate computation O(

Finally, is shown a comparison between similar systems:

Table 2: Comparison between similar scheme (10 voters)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **System Scheme** | **Underlying problem** | **Submitted Ballot Size** | **Encrypting Time (s)** | **Level of security** |
| Our | RLWE | Variable | 0.008 | Variable |
| Yang | El Gamal | 120kB | 0.9 | Fixed |
| Helios | El Gamal | 1.024kB | 0.3 | N/E |
| Shifa | Paillier | 0.152kB | N/E | N/E |

**Conclusions**

From Fig.6 can be easily inferred the asymptotical behavior of the implemented algorithms are linear, as the time required to calculate the 5000 ballots is also doubled when is raised to 10000 ballots. There for the general asymptotical performance will depend on the modules that are forming the module, in this case the signature will have an asymptotically behavior dominant which determines the system and it is the one shown in the table 1.

The presented solution incorporates an optimization of the recent work here security has been based in the Learning with Errors Problem throw the use of the BGV cryptosystem, and the Digital Signature for a full solution based in Lattices. Part of the security relies on the PBB that safeguards the decryption key, is not needed honest authority to safeguard the integrity of the whole system; because all the data is published constantly by the PBB it makes hard of an attacker to try to modify the data that in any case is digitally signed.

A prototype of the proposed system has been implemented for research purposes to determine the performance of the system, and a practical version is been developed, both projects can be download from the GitHub repository:

*https://github.com/joseluisnaranjo/HEVotingSystem*

As future direction might be interesting to determine the possibility of using secret sharing password to generate the public common key for the whole process, as a significant improvement for the system.

The scheme was implemented using library HELib.

**References**

* Chris Peikert. Lattice Cryptography for the Internet, Springer Publishing, Cham, 2014，pp 197-219.
* Stepanka Bilova. Eduard Fuchs, ed. Lattice theory its birth and life. Prometheus. pp. 250–257. 2001.
* Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In FOCS, pages 97–106. 2011.
* V. Lyubashevsky, et al. On Ideal Lattices and Learning with Errors over Rings, Eurocrypt 2010.
* V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for R-LWE cryptogrp. In EUROCRYPT, pag 35–54. 2013.
* P. Deshpande et al. Experimental study of Diffie-Hellman key exchange algorithm on embedded devices. IEEE10.1109/.8389808, Chennai, India, 2017.
* J. Zhang, L. Chen. An Improved Algorithm for Discrete Logarithm Problem, IEEE 10.1109/ESIAT.2009.457, Wuhan, China 2009.
* S. Som, R. Majumder   S. Dutta. Elliptic curve cryptography: A dynamic paradigm, IEEE [10.1109/ICTUS.2017.8286045](https://doi.org/10.1109/ICTUS.2017.8286045), Dubai, United Arab Emirates, 2017.
* A. Azougaghe, et al. An Electronic voting system based on HE and prime numbers. ENSIAS Rabat, Morocco, Conference on Security (IAS), 2015.
* B.Zwattendorfer, C.Hillebold, P. Teufl, “Security and Privacy-Preserving Voting System,” IEEE 10.1109. 2013.
* X. Yang, X Yi, S. Nepal, A Kelarev, and F Han, A secure Verifiable Ranked Choice Online Voting System Based on Homomorphic Encryption, OI 10.1109/ ACCES.1817518, IEEE Access. 2018.
* Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. Innovations in Theoretical Computer Science Conference, pages 309-325. ACM, 2012.
* A. Berzati, C. Dumas, and L. Goubin, “A survey of differential fault analysis against classical RSA implementations”. In Fault Analysis in Cryptography (pp. 111-124). Springer Berlin Heidelberg. 2012.
* S. Manaruliesya et al. Advanced e-voting system using Paillier, IEEE 10.1109/IAC.2016.7905741, Mataram, Indonesia, 2016.
* B. Adida, “Helios: Web-based open-audit voting,” in USENIX security symposium, pp. 335–348, 2008.
* M. Mikhail, Y. Abouelseoud, and G. Elkobrosy. Extension and Application of El-Gamal, IEEE 10.1109/WCCAIS.2014.69, Hammamet, Tunisia, 2014.
* G. Tsoukalas, K. Papadimitriou, P. Louridas, and P. Tsanakas, ‘‘From Helios to Zeus,’’ USENIX J. Election Technol. Syst., vol. 1, pp. 1–17, Aug. 2013.