TELKOMNIKA, Vol.16, No.2, April 2018, pp. 862~867
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v16i2.8420

Secure E-voting System by Utilizing Homomorphic Properties of the Encryption Algorithm

Rifki Suwandi*, Surya Michrandi Nasution, Fairuz Azmi
Electrical Engineering Faculty, Telkom University, Bandung, Indonesia
*Corresponding author, e-mail: rifki.s@hotmail.com[1], michrandi@telkomuniversity.ac.id[2], worldliner@telkomuniversity.ac.id[3]

Abstract

The use of cryptography in the e-voting system to secure data is a must to ensure the authenticity of the data. In contrast to common encryption algorithms, homomorphic encryption algorithms had unique properties that can perform mathematical operations against ciphertext. This paper proposed the use of the Paillier and Okamoto-Uchiyama algorithms as two homomorphic encryption algorithms that have the additional properties so that it can calculate the results of voting data that has been encrypted without having to be decrypted first. The main purpose is to avoid manipulation and data falsification during vote tallying process by comparing the advantages and disadvantages of each algorithm.

Keywords*: cryptography, encryption, homomorphic, Paillier, Okamoto-Uchiyama*

1. Introduction

There are numerous issues with the electronic vote casting system, together with system errors, network safety, information security, and so on. One of the primary issues is cheating committed by either insider or outsider, especially for some important role holders with authority that can access the system itself. To overcome these issues, cryptography can be used as a safety approach. Cryptography technique is used to encrypt the information such as voter's data, voter's choice, and the voting result before that data stored on the server. Therefore the system can ensure authenticity and security of voting information [1]. The different cryptographic algorithm can be implemented in this system, a few of them are Paillier and Okamoto-Uchiyama algorithm.

The paillier algorithm, as well as the Okamoto-Uchiyama algorithm, are asymmetric cryptographic algorithms that utilized the public key to encrypt messages, whereas the other key is called private key to decrypt the messages [2]-[3]. As shown by Rivest, Adleman, and Dertouzos in an encryption system, a data system can simply store and recover encrypted information for users. Decryption is needed for further operation. The information is not secure anymore after it is decrypted [4]. “Privacy homomorphism”, the new concept that allows direct calculation of encrypted information without decryption was proposed. With homomorphic encryption, the operations on the encrypted information including additions and multiplications may be carried out with the aid of using the public key algorithm. This algorithm also has homomorphic property, that could carry out mathematical calculations using messages which are still encrypted, also know as ciphertext.

This research centered on the effectiveness of Paillier and Okamoto-Uchiyama algorithms in terms of encrypting and decrypting the data while carrying out in e-voting system. as well as the accuracy of the calculation final result of the ballots using the homomorphic property of both algorithms, which is a unique property that not all kinds of asymmetric algorithms have it [5].

2. Research Method

As already described before, there are two keys needed in both Paillier and Okamoto-Uchiyama algorithms to perform encryption and decryption. To achieve that, there are several numbers that need to be generated before with following procedure.

Received December 23, 2017; Revised February 9, 2018; Accepted February 27, 2018

2.1. Key Generation

In the Paillier algorithm, to generate the private key and public key, we need to select two different big prime numbers p and q with gcd $(pq,(p-1)(q-1)=1$. Then calculate RSA modulus $n = pq$ and calculate Carmichael's function that can be computed with $\lambda=(p-1)(q-1)$ gcd $(p-1,q-1)$. After that, choose random g as generator where $g \in \mathbb{Z}_{n}{}^{2}$ with gcd $((g^{\lambda}\ mod\ n^{2}-1)/n,n)=1$. Later, find modular multiplicative inverse with $\mu=(L(g^{\lambda}\ mod\ n^{2}))^{-1} mod\ n$, when function L is defined as $L\ (u)=(u-1)\ /\ n$ [6]. After following those steps before, we could get the public-key for encryption (n,g) and the private-key for decryption (λ,μ).

As for Okamoto-Uchiyama algorithm, choose two big prime numbers p and q randomly, then calculate $n = p^2q$. Then select generator g, when a randomly selects $g \in (\mathbb{Z}/n\mathbb{Z})^*$, where $g^p \neq 1\ mod\ p^2$. After that calculate $h = g^n\ mod\ n$ [7]. After following those steps before, we acquire the public key tuple $< n, g, h >$ and the private key tuple $< p, q >$.

2.2. Encryption Process

Encryption is a process to scramble the actual data into something that can not be understood or at least does not reveal the actual meaning of the information. For Paillier algorithm, the message m that needs to be encrypted must be integer where $m \in \mathbb{Z}n$. Then select a random number r with $r \in \mathbb{Z}_n^*$. After that compute ciphertext $c=g^m r^n\ mod\ n^2$ [6].

As for Okamoto-Uchiyama algorithm, the message m must be integer where $m \in \mathbb{Z}/n\mathbb{Z}$. Then choose random number of r with $r \in \mathbb{Z}_n^*$. After that, compute to acquire ciphertext $C = g^m h^r\ mod\ n$ [7].

2.3. Decryption Process

Decrypting the previously disguised messages are needed to obtain the actual information. For Paillier algorithm, the ciphertext to be decrypted is c, where $c \in \mathbb{Z}_{n2}$. we can obtain the plaintext message using $m=L\ (c^{\lambda}\ mod\ n^{2})$ * μ mod n [6].

As for Okamoto-Uchiyama algorithm, we need to first define an auxiliary function $L(x) = \frac{x-1}{p}$. With the assist of this function, the decryption method to acquire message m is $m = \frac{L(C^{p-1}\ mod\ p^2)}{L(g^{p-1}\ mod\ p^2)}$ mod p [7].

2.4. Homomorphic Process

To perform the calculation using ciphertext, we can not simply calculate like the usual integer. The Paillier algorithm provides homomorphic addition properties that will be used to tally the data. With public key (n,g) and private-key (λ,μ), The ciphertext can be decrypted with $D(E(m_1,r_1)\ *\ E(m_2,r_2)\ mod\ n^2)=m_1 + m_2\ mod\ n$. The ciphertext with a plaintext raising g can be decrypted with $D(E(m_1,r_1)\ *\ g^{m^2}\ mod\ n^2)=m_1 + m_2\ mod\ n$ [6].

As for Okamoto-Uchiyama algorithm, using the public key $< n, g, h >$ and private key $< p, q >$ that we obtained before, given $C_0 = g^{m_0}h^{r_0}$ and $C_1 = g^{m_1}h^{r_1}$ as valid encryptions of m_0 and m_1 respectively, $C_0C_1 = g^{m_0+m_1}h^{r_0+r_1}\ mod\ n$ [8].

2.5. Implementation

For the whole election process, there are at least three important parts representing the e-voting system. That is from the side of the selector, from the server side, and from the administrator side. Figure 1 represents the system design that used in this research.

Whole design referred above will be applied in e-voting application using MySQL for its database and with Java programming language. Figure 2 shows the GUI mockup for a ballot paper in a software application. Figure 3 shows the preview of MySQL databases that will be used to stored election data in this e-voting system.

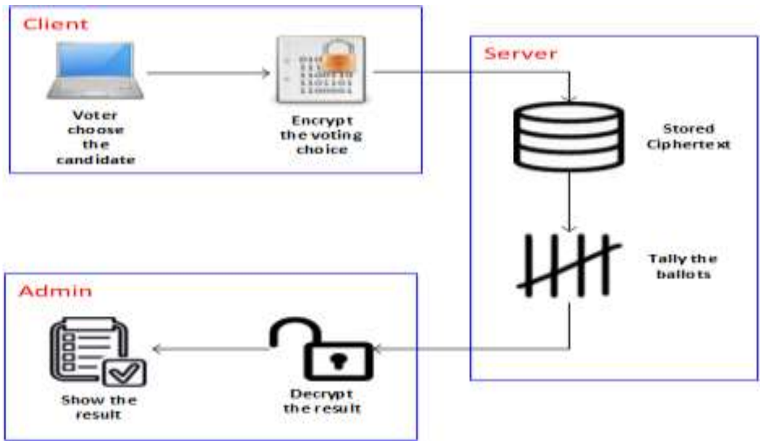
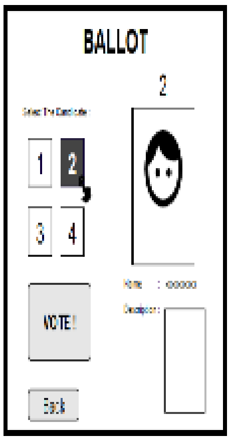
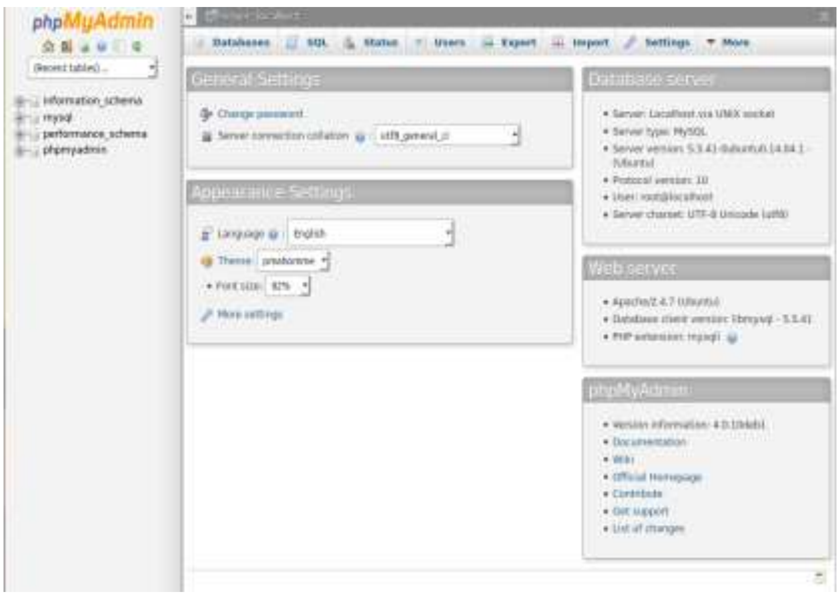


Figure 1. System design





phpMyAdmin

(Recent tables) ...

- information_schema
- mysql
- performance_schema
- phpmyadmin

Databases | SQL | Status | Users | Export | Import | Settings | More

General Settings

- Change password
- Server connection collation: utf8_general_ci

Appearance Settings

- Language: English
- Theme: pmahomme
- Font size: 82%
- More settings

Database server

- Server: Localhost via UNIX socket
- Server type: MySQL
- Server version: 5.5.41-0ubuntu0.14.04.1 - (Ubuntu)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.7 (Ubuntu)
- Database client version: libmysql - 5.5.41
- PHP extension: mysqli

phpMyAdmin

- Version information: 4.0.10deb1
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

Figure 2. Ballot's preview

Figure 3. Preview of databases

The voter selected the candidate using a computer that is connected to the internet. Then the ballot will be encrypted automatically and stored on the server in the ciphertext form. Every voter could have a unique ciphertext even though they selected the same candidates. The server will do the tallying process with all the encrypted data using the homomorphic properties owned by each algorithm. To obtain the voting final result, the admin has to do the decryption procedure.

3. Results and Analysis

The sample will contain 20 voters that will choose four different candidates. After every voter finishes deciding on the candidates, their vote's data will be encrypted through the encryption program. Table 1 indicates that even though the voters select same candidates, every ciphertext will be unique.

Table 1. Vote Messages to be Encrypted

Voter's Name	Candidates				Vote Messages (m)
	C1 (10^0)	C2 (10^1)	C3 (10^2)	C4 (10^3)	
V1			v		m=100
V2	v				m=1
V3				v	m=1000
V4				v	m=1000
V5		v			m=10
V6				v	m=1000
V7		v			m=10
V8	v				m=1
V9	v				m=1
V10	v				m=1
V11				v	m=1000
V12				v	m=1000
V13			v		m=100
V14		v			m=10
V15				v	m=1000
V16		v			m=10
V17		v			m=10
V18		v			m=10
V19			v		m=100
V20				v	m=1000
Total	4	6	3	7	

Now we will encrypt every message m as shown in the table above using keys and formulation as defined before. Table 2 shows the ciphertext from each message that being encrypted by both algorithms.

Table 2. Encrypted Vote

Voter's name	Vote message to be encrypted	Paillier		Okamoto-Uchiyama	
		Random r_p	Encrypted vote C_p	Random r_i	Encrypted vote C_i
V1	100	62437	351538799948203897	22343	95553799249047
V2	1	59119	4590993851485312445	52763	144506974646181
V3	1000	31138	2619624597605475591	56930	23028688473571
V4	1000	37055	2493084312669549831	15457	58757985293624
V5	10	33251	8868564164657005235	38622	116413472979662
V6	1000	48639	2610278780605368606	43950	135117603880356
V7	10	10920	7815835086000532337	28197	125447449827855
V8	1	37782	4173633540368342306	44574	123129075359966
V9	1	11337	6166033743055358318	54460	97754308571804
V10	1	37790	2314588247885282943	61282	90997884949109
V11	1000	40454	5956929699649780388	31554	123468550161811
V12	1000	56727	2236617076652008725	53292	131014172338142
V13	100	1867	10666027394708715038	30854	77830549352647
V14	10	16765	939972971689839058	32469	141964459664168
V15	1000	27529	8423042122639404412	41520	66277724374192
V16	10	17867	9705879308879738285	34098	4133661057407
V17	10	16072	7939254862481225551	52778	73845488491901
V18	10	34348	5125409701887114569	31671	72860561589603
V19	100	49819	5091673028311841742	16049	128303389793170
V20	1000	40942	5014329154833019458	30532	94893426438749

For the tallying process with homomorphic properties of Paillier algorithm, the server will sum up all encrypted data and mod by n.

$Tally\ (T_C) = \ (\sum_{i=1}^{N_V} C_p)\ mod\ n$ =106551642554900044578639457146674631799.
Then the admin can decrypt T_C to get message m. Using decryption formula :

$$\mathrm{m} = \mathrm{L}\left(\mathrm{c}^{\lambda} \bmod \mathrm{n}^2\right) * \ \mu \bmod \mathrm{n} = \ 4637$$

As for Okamoto-Uchiyama algorithm. The server will multiply all encrypted messages.

$Tally\ (T_C) = \prod_{i=1}^{N_V} C_i =$
22382287002697674738521592813970232513773189954874615858347778861309453055942547470176654671058047678083339665228177336963905078357936498030161767157374895307923552110472151367783435582826105461016292508642357152199784868731781133887865263766430981416960354953869688529097916416.

Then the admin can decrypt T_C to get message m. Using decryption formula:

$$m = \frac{L({T_C}^{p-1}\ mod\ p^2)}{L(g^{p-1}\ mod\ p^2)}\ mod\ p = 4637$$

Eventually, the final result of this vote casting is decided. From the final result from both algorithms before, m=4637, this means that candidate C1 has four votes, candidate C2 has six votes, candidate C3 has three votes, and candidate C4 has 7 votes. So the candidate C4 is the winner of this election.

If we count number manually the selections in Table 1, four voters select candidate C1, six voters select candidate C2, three voters select candidate C3, and seven voters select candidate C4. The final result will be equal to the final result as calculated earlier.

We can see, the size of the ciphertext is a whole lot larger than the plaintext (original message), due to the fact these algorithms primarily based on big prime numbers of p and q with pretty complex calculation to perform factoring or discrete logarithm, along with calculating values of n=p^{2}q for Okamoto-Uchiyama algorithm, which may be very big compared with Paillier algorithm, which generally has a value of n=pq.

The function of random number r in both algorithms is to make the result of encryption message that has exactly same value will different result of ciphertext. Therefore, these algorithms can guarantee the uniqueness of the data to be stored.

From the experimental results above shows that the ciphertext size of the Okamoto-Uchiyama algorithm is smaller than the Paillier algorithm. While for expansion factor of ciphertext size to prime size p and q, the Okamoto-Uchiyama algorithm has size three times bigger and four times bigger for the Paillier algorithm. As for the runtime of encrypting and decrypting process, Okamoto-Uchiyama algorithm is faster than the Paillier algorithm. But for the tallying process, the Paillier algorithm is much faster because each message is only summed so it does not generate many larger numbers than each initial ciphertext, while in the Okamoto-Uchiyama algorithm it will be several times larger than the initial ciphertext because each message will be multiplied .

The homomorphic properties that we use in this study proved to produce accurate results between calculated result using ciphertext that similar with desirable result that calculated manually after decrypted. So, we can use that property to anticipate cheating by the outsider to manipulate the stored data or trying to interrupt the tally process.

With the usage of homomorphic properties in these algorithms, we can anticipate cheating by either insider or outsider that try interrupt the tallying process or manipulate the stored data. This sample also proved to produce an exactly same result between tallying done manually and decrypted result after tallying using ciphertext by the program.

4. Conclusion

In this research, the use of Paillier and Okamoto-Uchiyama homomorphic encryption algorithms in e-voting system can utilize homomorphic properties of the algorithm to calculate the votes that processed by the system and ensure data confidentiality. These algorithms also produce a unique value for each ciphertext even with the exact same plaintext (original message). The desirable result that calculated manually is similar to the final result after being decrypted that calculated using the homomorphic method. The Okamoto-Uchiyama algorithm has a smaller ciphertext size than the Paillier algorithm but has a longer tallying processing time.

References

[1] Agustina Esti Rahmawati, Agus Kurniati. *Pemanfaatan Kriptografi dalam Mewujudkan Keamanan*

Informasi pada E-voting di Indonesia. Informatics National Seminar 2009 (semnasIF 2009), UPN Veteran, Yogyakarta.

[2] Nathanael Ezra Hizkia. Implementasi Algoritma Kriptografi Kunci Publik Okamoto-Uchiyama. Informatics Engineering, Bandung Institue of Technology. 2013.

[3] Ngo Cuong. *Secure Voting System Using Paillier Homomorphic Encryption*. Faculty of the Department of Computing Sciences Texas A&M University–Corpus Christi, Texas. 2014.

[4] Hyungjick Lee, Jim Alves-Foss, and Scott Harrison. *The use of encrypted fucntions for mobile agent security*. Proceeding of the 37th Annual Hawaii Internasional Conference on, IEEE. 2004: 10.

[5] Rifki S, Surya M N, Fairuz A. *Okamoto-Uchiyama Homomorphic Encryption Algorithm Implementation in E-Voting System*. Internasional Conference on Informatics and Computing (ICIC). IEEE Conferences. 2016: 329-333.

[6] Sansar Choinyambuu. Homomorphic tallying with paillier cryptosystem. HSR Hoschule fur Technik Rapperswil. 2009.

[7] Okamoto Tatsuaki, Shigenori Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. NTT Laboratories, 1-1 Hikarinooka, Yokosuka-shi, 239-0847 Japan.

[8] Henry Kevin, 2008. The Theory and Applications of Homomorphic Cryptography. Computer Science, University of Waterloo, Canada.

[9] Coron Jean-Sebastian, David Naccache, Pascal Paillier. Accelerating Okamoto-Uchiyama's Public-Key Cryptosyste. *Electronic Letters*. 1999; 35(4): 291-291.

[10] Peng Kun, Colin Boyd, Ed Dawson, Byoungcheon Lee, and Riza Aditya. 2004. *Multiplicative Homomorphic E-Voting*. In Canteaut, A & Viswanathan, K (Eds.) Progress in Cryptology–INDOCRYPT. 5th International Conference on Cryptology in India. 2004.

[11] Rahmmadian Tika, Aswin Suharsono, Ahmad Afif Supianto. Desain dan Implementasi Sistem Keamanan E-Voting dengan Jaminan Confidentiality Data. Computer and Informatics Engineering, University of Brawijaya.

[12] Vaghasia Chandni, Kirti Bathwar. Public Key Encryption Algorithms for Wireless Sensor Networks in TinyOS. *International Journal of Innovative Technology and Exploring Engineering (IJITEE).* 2013; 2(4). ISSN: 2278-3075

[13] Zhao Yingming, Yue Pan, Sanchao Wang, and Junxing Zhang. *An Anonymous Voting System Based on Homomorphic Encryption*. Inner Mongolia University. School of Computer Science. Huhhot. China. 2014.