ISSN No. 0976-5697

Volume 7, No. 2, March-April 2016

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Enhancement of Specific Encryption Scheme

Sattar J. Aboud
Department of Computer Science and Technology
University of Bedfordshire
Luton, UK

Abstract: Homomorphic cryptography scheme give the approach to outsource calculation to the cloud while preserving the secrecy of the information. To manage and increased the information that are being treated these days, good encryption result is the significant step for realism of homomorphic cryptography technique. In this paper, we consider the cryptography result of the Paillier scheme, the partly homomorphic scheme that lets to present sums on encrypted information without decrypt it. With the combination of both new and known schemes, we can enhance the scheme result by orders of magnitude compared with the simple execution. The new scheme decrease the sound computation by using sound calculation to produce fresh sound in the much quicker manner than by using standard schemes.

***Keywords**:* homomorphic cryptography technique, Paillier scheme, integer factoring,

I. Introduction

Homomorphic cryptography scheme is now bring many researchers attention since it permits protecting secrecy of sensitive information in cloud computing. Though, homomorphic cryptography scheme allows both multiplication and addition operations in encrypted information, is still useless in the practical preparations. But, study effort is going to dissimilar classes of homomorphic cryptography, one of which is partly homomorphic cryptography.

These schemes carrying out only one certain operation in encrypted information, either addition or multiplication. Therefore, we discuss the familiar partly homomorphic that is the Paillier scheme [1]. It permits carrying out sums on encrypted information, which is vital in numerous cases like encrypted database [2] e-voting [3], and machine learning on encrypted information [4]. In its operation, the Paillier scheme has poor cryptography results. There are couple of standard approaches, some of which already discussed in Paillier paper, that significantly enhance the cryptography implementation. In this paper, we adjust the presented approach and combine it with a new approach. By achieving this, we make the cryptography result sufficient for high use situations. On the good machine, the result is between $40,000-90,000$ operations per second, based on the security level. Sections 4 and 5 give more details on the results.

From the theoretical view, cryptography with the Paillier scheme contains the determined source modular exponentiation with the message and the generation of the sound employed to mask the message. For determined basis exponentiation, it is familiar that the result can be improved by calculating powers of the determined basis. By modification this and other common schemes, we decrease the complexity of this step greatly. For the generation of the sound, we apply the new algorithm that using calculated sound to produce new sound. This decreases the sound calculation of the modular multiplications. Jointly, these methods accomplish a great result in cryptography. We describe the scheme in section 3.

There are other works constructed on the Paillier scheme which enhance its result. Catalano *et al.* presented another decryption method, expansion the scheme to let the random exponent e in place of N. The weakness is that the scheme misplaces its additive homomorphic characteristic [5]. Damgard *et al.* suggested the generalization of the system by which an extending factor is decreased and performances of both schemes are evaluated without lacking a homomorphic characteristic [6]. Their scheme reaches the rapidity of 0.262 sec/bit for the Paillier scheme, as good as $3{,}816.79$ bits/sec. This result was achieved by the bright alternative of basis and using standard calculation methods for determined basis exponentiation. Though, the cryptography result can be increased more by using the method considered in this paper. We achieve the rapidity of $9{,}197{,}824$ to $48{,}810{,}496$ bits/sec by using the security parameter. The schemes discussed in this paper become particularly efficient when the message size is short compared with the key size, and thus an encrypted message. This has the drawback of the larger encrypted message expansion. To decrease an encrypted message expansion, some shorter messages can be packed into the larger one, as presented in [7]. Though, using this technique needs to use the additional complex operation than uncomplicated multiplication to add the related messages.

The remaining of this paper is organized as follow, we begin by briefly illustrating the Paillier scheme in Section 2 in order to determine the notations used in this paper. We also remark on practical matters with key generation and keys selection that are not entirely delivered in the paper. We carry on by considering the scheme we employ for result enhancement in Section 3. Finally, we epitomize the result in Section 4 and illustrate the conclusion in Section 5.

.

II. The Paillier Scheme

In 1999, Paillier presents the homomorphic scheme [1], The scheme is quicker in decryption. A security of the Paillier scheme is under $n-th$ residues $\in Z^*_{n^2}$ and difficult of integer factoring. The established of the Paillier scheme is a multiplicative group $Z^*_{n^2}$ for $n = pq$. Note that $Z^*_{n^2}$ has $|Z^*_{n^2}| = \theta(n)^2 = n$, and $\theta(n) = (p-1)(q-1)n$. The Carmichael theorem on n, $\lambda(n)$, is short-handed to λ. The steps of the scheme are as follows:

A. *Algorithm for Key Generation*

To create the Paillier public-key and the related private-key. Every entity A must do the following:

1. Generate two large arbitrary primes p and q equally likely.
2. Find the modulus $n = pq$
3. Compute $\lambda = lcm(p-1)(q-1)$
4. Choose an arbitrary integer a where a is the divisor of λ
5. Choose an arbitrary integer $g \in Z^*_{n^2}$, such that the order of g is an.
6. Entity A public key is (n, g) ; where entity A private key is (p, q, a).

B. *Algorithm for Public-Key Encryption*

Entity B encrypts the message m for entity A, which entity A decrypts.

1. Encryption. Entity B must do the following:
   a. Get entity A valid public-key (n, g).
   b. Choose an arbitrary message $m < n$.
   c. Represent the message as an integer m in the range $(0, n-1)$.
   d. Choose a random integer $r < n$ where $r \in Z^*_n$.
   e. Calculate $c = g^m (g^n)^r \bmod n^2$.
   f. Post the cipher-text c to entity A.
2. Decryption. To decrypts the message m from c, entity A must do the following:
   a. Check that cipher-text $c < n^2$
   b. Use the private-key a to decrypt the massage $m = \dfrac{L(c^a \bmod n^2)}{L(g^a \bmod n^2)} \bmod n$ where $L(u) = \dfrac{u-1}{n}$, and $u = 1 \bmod n$.

Since the Paillier scheme is using integer factoring, the same conditions must keep for n as for the modulus length in the RSA scheme. If selecting the integer g, it requires to be verified if the order of the selected g is the multiple of n. Corresponding to Paillier scheme formula (4), this can verified by testing if $\gcd(L(g^\lambda \bmod n^2), n) = 1$.

Also, consistent with Paillier g must be small for implementation. It means $g = 2$. In the execution illustrated in this section we select to work with the subgroup $\langle g \rangle$ produced by the integer g of order an. This permits decryption by doing an exponentiation with exponent a in place of λ, which rapidity up decryption based on the length of a. where $1 \le a \le \lambda$. As there is an integer g of order an, it follows from Carmichael function that a has the divisor of λ, and $r < n$. However, it be sufficient to have $r < a$ and an order of g is a. The element a requires to be sufficiently large in order to keep away from Shank algorithm attacks on a.

The problem to find a from the public keys g and n is by solving $(g^n)^a = 1 \bmod n^2$. This problem is close to the discrete logarithm problem, but not the same. To be more accurate, instead of finding the discrete logarithm in the subgroup of known order, compute the order of the known element. Also, algorithms such as index calculus and Pohlig-Hellman cannot used for such problem. Also, Shank algorithm can be used to compute a. But, using such attack requires $O(\sqrt{a})$ multiplications. The Shank algorithm can be used to compute the length of a as for key length of ECC schemes. Also, it must be difficult to determine a for the security level and must be as high as the security level for n. According to $NIST$ 2048 - 3072 bit security and $256 - 383$ bit security for ECC schemes [7].

The only new scientific issue with key generation for Paillier scheme is to create g with the right order of an. This can be accomplished by using the DSA key generation twice g_1 of order $a_1 \in Z^*_p$ and g_2 of order $a_2 \in Z^*_q$. Then, we deduce g_1 and $g_2 \in Z^*_{p^2}$ and $Z^*_{q^2}$. It is easy to verify if an order of $g_1 \in Z^*_{p^2}$ is $a_1 p$ and an order of $g_2 \in Z^*_{q^2}$ is $a_2 q$. One can compute g of order $a_1 a_2 pq = an \in Z^*_{p^2q^2} = Z^*_{n^2}$ using the Chinese Remainder Theorem. It can be simply verified that a is the divisor of λ. Table 1. shows the key lengths in bits using DSA key generation twice.

Table I. Key Sizes in Bits for Paillier Scheme

p and q	a_1 and a_2	n	a
512	160	1024	320
1024	160	2048	320
1500	160	3072	320
2048	256	4096	512

C. *Security of the Paillier Scheme*

The Paillier scheme is not remark on how to produce the arbitrary $n-th$ powers g^{nr}. It is sufficient to select r smaller than a since the order of g is na. But, it can be possible to select r from the smaller set. To determine the requirements of this smaller set, we study two attacks on r with g^{nr} and how to avoid them.

The first attack on the encrypted message $c = g^m g^{nr}$ is by selecting r. To evade this type of attack, a set of r is arbitrarily selected. Note that, by selecting r, only the single encrypted message can be decrypted, thus selecting r is not break the entire scheme. Supposing that $r's$ are selected arbitrarily and no other relations between them, then the length of an arbitrary space can determine how difficult to attack the single encrypted message. For example, in section 3.3, we select r arbitrarily from 2^{70} values. Thus, we use $70-bit$ security for the single encrypted message.

The second type of attack is to use relations between the arbitrary values used in different encrypted messages. For example, if the relation is $r_3 = r_2 + r_1$. Then the hacker can calculate $\frac{c_3}{c_1 c_2} = \frac{g^{m_3} g^{nr_3}}{g^{m_1} g^{nr_1} g^{m_2} g^{nr_2}} = g^{m_3 - m_1 - m_2}$ .Finding $m_3 - m_1 - m_2$ is the discrete logarithm problem. To prevent this type of attack, must not present the selection of (r^n or g^{nr}). The method to produce sound in section 3.3 is to resist the two attacks.

III. IMPLEMENTATION

In this section, we describe both the known and the new methods to enhance the encryption result of the Paillier scheme. Suppose that the message size is small compared to the key length. As the key length is 2048 bits and the messages are integers, this appears to be the practical assumption for actual use cases. There is, an option way will use the method illustrated in [8], which decreases encrypted message.

A. *Reduced Moduli*

When the private keys are known to an entity, then he can reduced moduli by dividing modulo n^2 in two elements p^2 and q^2. The moduli p^2 and q^2 can be more reduced down to $\bmod p$, and $\bmod q$. The calculations are then done on the elements.

B. *Computing the Message*

To calculate the message $g^m \bmod n^2$, the classical methods can be used for determined basis modular exponentiation. The calculation of powers $g \bmod n^2$ decrease the number of modular operations required to calculate $g^m \bmod n^2$. This is work when the message m is short. There is the compromise between calculation space and time; and encryption rate. We supposed that the messages are $32-bit$ numbers. Then, we calculated the large number of powers of g, that is $g^{(2^{16})^i j}$, and $j = 0..2^{16}-1$. To calculate g^m, we divide m into two $16-bit$ integers j_i, with $j_i = \lfloor m/(2^{16})^i \rfloor \bmod 2^{16}$. Then g^m can be calculated in one modular multiplication $g^m \bmod n^2 = g^{2^{16} j_1} g^{j_0}$.

C. *Computing the Sound*

To find the sound of the encrypted message, it is necessary to use comparable calculation methods for the message part. In this case only arbitrary power of g^n is required to calculate. The concept is to calculate the large number of arbitrary powers of g^n and to use them to make new arbitrary powers. To find $(g^n)^r$ select r while $(g^n)^r$ is calculated as a product of calculated arbitrary powers of g^n. There is an exchange between storage of calculated powers and the arbitrariness of the power of g^n. In the execution of section 2.3, we need to make the possibility of guessing r to be at most 2^{-70}. We find the table of 2^{16} arbitrary $(g^n)^r$ and multiplying 5 of them as one in encryption. The $(g^n)^r$ selected may repeat themselves, thus the number of probabilities is the 5-combination with repetitions, it means $\left(\binom{2^{16}}{5}\right) = \binom{2^{16}+5-1}{5} \approx 2^{73}$.

We discuss other values for the length of the table 2. It describes the calculation time (sec) and encryption rapidity (sec) for dissimilar key sizes (bits) n, and the probability P of presuming r for the single encrypted message. Up to now, we have illustrated the method that an arbitrary sound is of the type $(g^n)^r$. Though, the same method can be used for the arbitrary sound where calculated sound can be employed to make new sound. This method can be used for arbitrary sound of the type h^r such that h is determined and r is arbitrary. It can be used for sound of the type r^n for arbitrary r and determined n. For example, the Paillier scheme. The reason is that the new sound can be produced by multiplying calculated sound factors of the same type.

Table II Encryption Speed based on Size of the Key n, with Fixed Computed Value 2^{16}.

key	*sound* $= 5 \rightarrow P = 2^{-75}$		*sound* $= 8 \rightarrow P = 2^{-113}$	
size of key length n	*computing*	*encryption*	*computing*	*encryption*
1024	2.554725	190666	2.563800	116974
2048	5.661963	89483	5.848805	32480
4096	25.156933	35236	25.510914	26553
key	*sound* $= 10 \rightarrow P = 2^{-138}$		*sound* $= 16 \rightarrow P = 2^{-112}$	
size of key length n	*computing*	*encryption*	*computing*	*encryption*
1024	2676721	108682	2.674735	70687
2048	5.733593	56558	5.973831	34955

4096	24.990774	23759	25.657435	17659

IV. RESULTS

The proposed enhancements increase the encryption results of the Paillier scheme considerably. The execution of Paillier scheme is quite slow. The speed about 500 encryptions per second for 2048 bit size of n. The execution of the proposed enhancements, split of $32-$ bit number into smaller units, reduced moduli. The pace increase. For 2048 key size, we accomplished the encryption pace of 89,483 encryptions (sec). Table 3 shows the pace of encryption (sec) for dissimilar key sizes. Encryption per second with dissimilar variations of Paillier, based on the size of the key n. For the proposed scheme execution, we use $73-$ bit security for guessing r.

The laptop used has an Intel $i7-4600$ CPU at $2.10GHz$ with 4 cores. Note that we are interested in the number of encryptions per second rather than the number of bits encrypted per second.

Table III Encryptions per second with different variations

Size of key	Paillier Scheme	The proposed Scheme
1024	1898	190666
2048	522	89483
3072	269	35929

V. CONCLUSIONS

The implementations of the Paillier scheme have poor encryption result, the result can be enhanced greatly by using the combination of known and new methods. The known methods are calculations of fixed-basis powers, and calculating with reduced moduli. The new methods illustrated in this paper are the method to increase the pace of generating sound, and the method to manage changeable information speed. With this enhancement, the Paillier scheme becomes practical even for use cases that want high encryption or have vacillate information speed.

VI. ACKNOWLEDGMENT

The author wishes to extend his thanks to the University of Bedfordshire, Department of Computer Science and Technology for their helpful suggestions and support.

REFERENCES

[1] Pascal Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes", Advances in Cryptology - EUROCRYPT'99, vol. 1592 of Lecture Notes in Computer Science, pp. 223-238, 1999.

[2] Popa R., Redfield C., Zeldovich N., and Balakrishnan H., "CryptDB: Protecting Confidentiality with Encrypted Query Processing", Proceedings of the 23rd ACM Symposium on Operating Systems Principles, pp. 85-100, 2011.

[3] Grofig P., Harterich M., Hang I., Kerschbaum F., Kohler M., Schaad A., Schropfer A., and Tighzert W., "Experiences and observations on the industrial implementation of a system to search over outsourced encrypted data", Sicherheit 2014: Informatik e.V. (GI), pp. 115-125, 2014.

[4] Bost R., Popa R., Tu S., and Goldwasser S., "Machine Learning Classification over Encrypted Data", NDSS Symposium, 2015.

[5] Catalano D., Gennaro R., Howgrave-Graham, and Nguyen P., "Paillier cryptosystem revisited", Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 206-214, 2001.

[6] Damgard I., Jurik M., and Nielsen J., "A generalization of Paillier public-key system with applications to electronic voting", International Journal of Information Security, no. 6, pp. 371-385, 2010.

[7] Sattar J. Aboud, "Cryptanalysis of Certificateless Aggregate Signature Scheme", International Journal of Computer Networks and Security, Volume 23, Issue 1, pp. 1109-1112, 2013.

[8] NIST. SP 800-57: Recommendation for Key Management - Part 1: General (Revision 3), 2012.