Indian Journal of Science and Technology, *Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/87977, October 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Novice's Perception of Partial Homomorphic Encryption Schemes

B. Selva Rani*

School of Information Technology and Engineering, VIT University, Vellore - 632014, Tamil Nadu, India;
bselvarani@vit.ac.in

Abstract

Objectives: Homomorphic Encryption is a way of performing computations on encrypted data. In this paper, we explore two famous Homomorphic Encryption schemes (RSA and ElGamal) with illustrations and related security issues. **Methods/ Analysis:** Both RSA and ElGamal encryption techniques possess multiplicative homomorphic property. These algorithms support only one operation (multiplication) on encrypted data and so they are termed as partial homomorphic encryption schemes. By using RSA and ElGamal, the encrypted data could be multiplied together without performing decryption. If needed, the results after such computations could be returned in decrypted form. This scheme reduces computation time and increases security and privacy of data being processed. **Findings:** It has been described how RSA and ElGamal algorithms perform key generation, encryption and decryption with simple illustrations. It has also been proved that RSA and ElGamal algorithms support multiplication operation on encrypted data with examples. **Improvement/Applications:** Many organizations rely on third party to outsource their large amount of electronic data for storage. It may be needed to perform some computations on the encrypted data on the server side provided by untrusted third party. Homomorphic Encryption will be much useful in such applications.

Keywords: Cryptography, El Gamal, Homomorphic Encryption, Partial Homomorphic Encryption, RSA

1. Introduction

With drastic usage of electronic data, most of the organizations outsource to store their data in reliable and secure manner, which in turn raises the necessity for techniques to assure the same. Such external data-centers provide storage space and let the organizations to encrypt and keep confidential information securely. Hence, only authorized owner of the secret data can have access to the stored data by using the secret key. Recently, there emerges a need for performing statistical, mathematical computations, predictive analysis etc. on the encrypted data. But conventional data-centers lack in providing computation techniques on the same, since encrypted data must be decrypted as original data for such processing. At this step, data are highly exposed to threats and also the computation time would be increased by searching and decrypting the data.

Homomorphic Encryption (HE) techniques enable us to accomplish operations on encrypted information without decrypting it. Such computation techniques are also termed as 'Privacy-Preserving' computation techniques. In general, each Homomorphic Encryption is a technique that permits computation on encrypted data, generates the result of computation in encrypted form. In section 2, the basic of cryptography is recalled and the idea behind the symmetric encryption and asymmetric encryption schemes is provided. In section 3 the idea about homomorphic encryption is given. In section 4, the history of homomorphic encryption schemes is presented. Section 5 delivers the investigation on two famous partial homomorphic encryption schemes and their features with illustrations followed by simple conclusion in Section 6.

*Author for correspondence

2. Cryptography

Cryptography is an art of preserving the confidentiality of secret information. It relies on two major processes, encryption and decryption along with a secret key. Such schemes were designed to keep the data securely and depend on the strength of the algorithm involved and the secrecy of the key used. There are two types of encryption methods: symmetric encryption scheme and asymmetric encryption scheme.

2.1 Symmetric Encryption

A symmetric encryption scheme involves an encryption algorithm, a secret key and decryption algorithm which can be depicted as below:

$$c = Enc_{k_s}(p) \quad (1)$$

$$p = Dec_{k_s}(c) \quad (2)$$

where, p is the plain text/secret information,
c is the cipher text/encoded information,
Enc is the algorithm used for encryption,
Dec is the algorithm used for decryption and
k_s is the secret key used for both encryption and decryption.

Since the unique key k_s is for both encryption and decryption, this scheme is termed as Symmetric Encryption scheme. Initially, both sender and receiver agree on a shared key/secret key. The sender encrypts the confidential information to be shared using any symmetric encryption algorithm and k_s, the secret key and sends the encoded data to the receiver. The receiver applies the corresponding decryption algorithm and the same secret key k_s on the encoded data and recovers the confidential information on his side. These symmetric encryption schemes can be distinguished as stream ciphers and block ciphers[1] with faster performance. The following are two problems which have to be addressed in the symmetric encryption techniques:

- The secret key should be shared among the sender and the receiver in a very secure manner.
- To communicate with more than one, an user must share distinct secret keys

DES, 3DES, AES and Blowfish are some famous symmetric encryption algorithms.

2.2 Asymmetric Encryption

In contrast, an Asymmetric Encryption scheme involves an encryption algorithm, decryption algorithm, two distinct keys for encryption and decryption separately. This may be depicted as below:

$$c = Enc_{k_U}(p) \quad (3)$$

$$p = Dec_{k_R}(c) \quad (4)$$

where, p is the plain text/secret information,
c is the cipher text/encoded information,
Enc is the algorithm used for encryption,
Dec is the algorithm used for decryption,
k_U is the public key, used for encryption, shared with others and
k_R is the private key, used for decryption, kept confidentially.

Here, each user generates a pair of keys (k_R, k_U) known as Private Key and Public Key. The Private Key is kept confidentially by the user and the Public Key is shared among others. The encrypted information using the public key of a user can only be decrypted with the help of the corresponding private key. In this scheme of secret sharing, there is no need to agree upon anything between the sender and the receiver. When compared with symmetric schemes, asymmetric encryption schemes are slower in performance due to mathematical computations involved. RC4, RSA and ElGamal are some familiar asymmetric encryption algorithms.

As symmetric encryption schemes are much faster than the asymmetric encryption schemes, it is worthy of using those while encrypting much longer data. The secret key may be shared using asymmetric encryption schemes since there is no need for any agreement for keys before sending the information. So, to achieve better performance, the functionality of asymmetric encryption scheme can be integrated with the effectiveness of the symmetric encryption[1].

Due to vast usage of electronic data in the fields like governance, marketing, health care, defense, the data encrypted using the above discussed encryption schemes are started outsourced in clouds with the help of the third parties. We face issues in the performance and secrecy of encryption schemes when there raises any need to perform mathematical computations or analysis on these encrypted data. The only way is the data owner has to decrypt the entire data storage, search the data needed and then to perform any computations on the decrypted data. This consumes a lot of time to get one computational result. There emerges Homomorphic Encryption scheme to perform mathematical computations and analysis on encrypted data without performing decryption.

3. Homomorphic Encryption

Any computation function f is known as homomorphic if it satisfies the following:

$$f(x \ (\ y) = f(x) \ o \ f(y) \quad (5)$$

where, '+' and '*o*' are two arithmetic operations. In the field of Cryptography, '*Enc*', an encryption algorithm is said to be a Homomorphic Encryption Algorithm, if for any two encrypted information, $Enc(m_1)$ and $Enc(m_2)$, we can compute $Enc(m_1 + m_2)$ without performing decryption operation, i.e.,

$$Enc\left(m_1 + m_2\right) = Enc\left(m_1\right) o \, Enc\left(m_2\right) \quad (6)$$

Similarly, '*Dec*', a decryption algorithm is said to be a Homomorphic Decryption Algorithm, if for $Enc(m_1)$ and $Enc(m_2)$,

$$Dec\left(Enc\left(m_1\right) o \, Enc\left(m_2\right)\right) = Dec\left(Enc\left(m_1 + m_2\right)\right) = m_1 + m_2 \quad (7)$$

The main goal of homomorphic encryption scheme is to guarantee secrecy of the confidential information during communication, on storage as well as while processing. Even though it is similar to conventional cryptographic encryption schemes, it provides additional provision for performing computations over the encoded data, searching for the required data and so on. In this way, homomorphic encryption scheme assures security during computation of data even in public cloud.

4. History of Homomorphic Encryption Scheme

In 1976, Diffie and Hellman proposed the first asymmetric algorithm, also known as Diffie-Hellman Key Exchange algorithm. They introduced this technique to interchange the secret keys between the users in a secure manner. Also, this system needs both (sender and receiver) to participate in computing the secret key. But this algorithm was proved vulnerable to brute-force and man-in-the-middle attack. The well-known RSA algorithm which is a public-key cryptosystem ses two distinct keys for encryption and decryption[2]. In 1984, El Gamal[3] described a public-key cryptosystem by simplifying the Diffie-Hellman key-exchange algorithm. He introduced a random exponent to replace the secret exponent of the receiver, which might be useful even though the receiver is not active. RSA and ElGamal support multiplication operations on encrypted data. Goldwasser-Micali, Cohen-Fischer and Pascal Paillier introduced homomorphic encryption techniques which support either x-or or multiplication or addition operation but not more than one on encrypted data[4-6]. In[8], Gentry introduced homomorphic encryption based on lattices, which permits arbitrary number of both addition and multiplication computations on encoded data. Subsequently, other homomorphic encryption schemes have been developed by allowing arbitrary computations successfully on cipher text[9-13]. Homomorphic Encryption scheme is said to be

- Partial, if it supports either addition or multiplication operation but not both on the encrypted data (PHE).
- Somewhat, if it permits a limited number of additions and at least one multiplication operation on the encrypted data (SwHE).
- Fully, if it allows arbitrary number of addition and multiplication operations on the encrypted data (FHE).

5. Partial Homomorphic Encryption Schemes

5.1 RSA Cryptosystem

5.1.1 RSA Algorithm with Illustration

During 1978, Rivest, Shamir and Adleman suggested the secure processing of data by using hardware, which did cost a high to implement. In the same year, they proposed the popular RSA algorithm, which was actually the first practically implemented public key encryption algorithm. RSA algorithm uses two distinct keys viz., Public Key and Private Key for encryption and decryption separately and hence known as Asymmetric Encryption Algorithm. Table 1 shows 1. The RSA key generation algorithm, which generates a pair of keys, public key and private key of a user, 2. RSA encryption algorithm and 3. RSA decryption algorithm with simple illustrations.

5.1.2 RSA Homomorphic Encryption

RSA algorithm supports multiplication operation on the encrypted information and it can be easily proved that

Table 1. RSA algorithm: Key generation, encryption and decryption

Key Generation	Example
i. Pick any two large prime numbers (x and y) ii. Compute $n = xy$ iii. Compute $\varphi(n) = (x-1)(y-1)$ iv. Choose e, so that, $\gcd(e, \varphi(n)) = 1$ and $1 < e < \varphi(n)$ v. Compute d*e≡ 1 (mod φ(n)) vi The keys are public key = {e, n} private key = {d, n}	Let x = 11 and y = 3 n = 11 X 3 = 33 $\varphi(n)$ = (11 – 1) X (3 – 1) = 10 X 2 = 20 Let e = 7 3*7 ≡1 mod φ(n) public key = {7, 33} private key = {3, 33}
Encryption	
Let '*m*' be the secret message and *m* < *n*. Then, encryption of '*m*' can be done as, $c = Enc(m) = m^e \bmod n$	Let m = 2 Then c = Enc(2) = 2^7 mod 33 = 29
Decryption	
Decryption is carried out as, $m = Dec(c) = c^d \bmod n$ where '*c*' is the cipher information	Now c = 29 m = Dec(29) = 29^3 mod 33 = 2

RSA supports Multiplicative Homomorphic property as given below:

Let mi and m_j be two secret messages and *ci* and c_j be the corresponding cipher messages in the space $(\mathbb{Z}, \cdot)$ such that

$$c_i = m_i{}^e \bmod n \text{ and } c_j = m_j{}^e \bmod n$$

$$c_i \cdot c_j = \left(m_i^e \cdot m_j^e\right) \bmod n = \left(m_i \cdot m_j\right)^e \bmod n$$

This can be verified with the worked out example. Let m_i = 2 and m_j = 3 be the two messages. Let us take the public and private key pair from the table. The corresponding cipher messages are obtained as

$$c_i = m_i^e \bmod n = 2^7 \bmod 33 = 29$$

$$c_j = m_j^e \bmod n = \mathbf{3^7} \bmod 33 = \mathbf{9}$$

Now,

$$c_i \times c_j = m_i^e \bmod n \times m_j^e \bmod n$$

$$= 29 \times 9 \bmod \mathbf{33}$$

$$= 261 \bmod 33 = \mathbf{30}$$

$$c_i \times c_j = \mathbf{30}$$

and,

$$Dec(c_i \times c_j) = Dec(30)$$

$$= c^d \bmod n$$

$$= \mathbf{30^3} \bmod \mathbf{33}$$

$$= 27000 \bmod 33 = 6 = m_i \times m_j$$

Hence it is proved that RSA supports multiplicative homomorphic property.

5.1.3 RSA Security

The strength of RSA security is subject to the integer factorization problem. RSA is vulnerable to 1. Brute-force attack: trying to crack the cipher text without knowing the private key. The encryption algorithm is the one-way function also known as trap-door function, where the trap door is the secret key (private key). It would be very

Table 2. ElGamal algorithm: Key generation, encryption and decryption

Key Generation	Example
i. Choose a large prime number '*n*' ii. Choose a primitive value '*α*' in modulo n iii. Choose '*x*' randomly such that $2 \leq x \leq n-2$ iv. Compute $\beta = \alpha^x \bmod n$ v. The keys are public key = {α, β, n} private key = {x, n}	Let n = 11 Let α = 2 Let x = 5 β = 2^5 mod 11 = 10 public key = {2, 10, 11} private key = {5, 11}
Encryption	
Let '*m*' be the secret message and *m* < *n*. Then, encryption of '*m*' can be done as below: i. Choose '*r*', a random integer and keep it as secret ii. Compute $s = \alpha^r \bmod n$ iii. Compute $t = \beta^r \times m \bmod n$ iv. Dispose '*r*', send the cipher information $c = (s, t)$	Let m = 7 Let r = 6 s = 2^6 mod 11 = 9 t = 10^6X 7 mod 11 = 7 c = (9, 7)
Decryption	
Decryption is carried out as $m = Dec(c) = t \times s^{-x} \bmod n$	m = 7 X 9^{-5} mod 11 = 7

difficult to reverse this encryption function without the knowledge of private key.

2. Computing the private key from public key: trying to calculate the value of private key from public key which is equivalent to the integer factorization problem. Multiplication of two prime numbers is said as one-way function. By choosing two larger prime numbers, computing the private key from public key by factoring 'n', would be very difficult.

5.2 ElGamal Cryptosystem

5.2.1 ElGamal Algorithm with Illustration

In 1985, El Gamal proposed a cryptosystem which falls under the kind of asymmetric crypto systems. ElGamal cryptosystem is the origin for numerous famous cryptographic primitives. Table 2 shows 1. The ElGamal key generation algorithm, which generates a pair of keys, public key and private key, 2. ElGamal encryption algorithm and 3. El Gamal decryption algorithm with simple illustrations.

5.2.2 El Gamal Homomorphic Encryption

El Gamal cryptosystem also supports multiplicative homomorphic property on encrypted information. Let m_i = 2 and m_j= 3 be two secret messages and *ci* and *ci* be the corresponding cipher messages in the space $(\mathbb{Z}, \cdot)$. Then c_i and c_j will be obtained as below:

$$s_i = \alpha^{r_i} \bmod n$$
$$= 2^{13} \bmod 11 = 8$$
$$t_i = \beta^{r_i} \times m_i \bmod n$$
$$= 10^{13} \times 2 \bmod 11 = 9$$
$$c_i = (s_i, t_i) = (8, 9)$$
$$s_j = \alpha^{r_j} \bmod n$$
$$= 2^9 \bmod 11 = 6$$
$$t_j = \beta^{r_j} \times m_j \bmod n$$
$$= 10^9 \times 3 \bmod 11 = 8$$
$$c_j = (s_j, t_j) = (6, 8)$$

Now,

$$c_i \times c_j = (s_i, t_i) \times (s_j, t_j)$$

$$= (8, 9) \times (6, 8) \; mod \; 11$$
$$= ((8 \times 6), (6 \times 8)) \; mod \; 11$$
$$= (48, 72) \; mod \; 11 = (4, 6)$$

(ci X cj) = (4, 6)

and,

$$Dec(c_i \times c_j) = Dec(4, 6)$$
$$= 6 \times 4^{-5} \; mod \; 11$$
$$= 6 \times 3^5 \; mod \; 11 = 6 = m_i \times m_j$$

Hence it is proved that ElGamal cryptosystem supports multiplicative homomorphic property.

5.2.3 ElGamal Security

The strength of ElGamal security is subject to the discrete logarithm problem. As long as the chosen prime numbers and the random number are large, it will be harder to break this cryptosystem using chosen cipher text attack. Message expansion while transferring it into cipher text by a factor because of different random numbers chosen per block is yet another shortcoming. Forged-signatures play a major role because of the possible Man-in-the-Middle attack in this system.

6. Conclusion

Until recently Homomorphic Encryption technique remains a major research domain. In this paper, it has been discussed about the homomorphic property of two algorithms, RSA and ElGamal with simple illustrations and also the security challenges against both the algorithms. Further, I would like to extend this study on more Homomorphic Encryption algorithms with pros and cons and the usage Homomorphic cryptosystems in most sensitive fields like storage and processing of medical records and images.

7. References

1. Caroline FCF, Fabien GFG. A survey of homomorphic encryption for nonspecialists. Journal of Information Security. 2009; 1:41-50.
2. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1983; 26(19):96-9.
3. Gamal TE. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory. 1985; 31(4):469-72.
4. Shafi GG, Silvio MSM. Probabilistic encryption and how to play mental poker keeping secret all partial information. Proceedings of the 14th Annual ACM Symposium on Theory of Computing; USA. 1982; p. 365-77.
5. Josh D, Cohen, Michael J, Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). IEEE Proceedings of 26th Annual Symposium on Foundations of Computer Science; Connecticut. 1985. p. 372-82.
6. Pascal PP. Public-key cryptosystems based on composite degree residuosity classes. Advances in Cryptology-EUROCRYPT, LNCS. 1999; 1592:223-38.
7. Dan Boneh DB, Eu Jin Goh EG, Kobbi Nissim KN. Evaluating 2-DNF formulas on cipher texts. Proceedings of Theory of Cryptography, LNCS 3378; 2005. p. 325-41.
8. Craig Gentry CG. Fully homomorphic encryption using ideal lattices. Proceedings of the 41st Annual ACM Symposium on Theory of Computing; BUSA. 2009. p. 169-78.
9. Dijk M, Craig Gentry CG, Shai Halevi SH, Vinod VVV. Fully homomorphic encryption over the integers. Advances in Cryptology-EUROCRYPT, LNCS. 2010; 6110:24-43.
10. Nigel P, Frederik VFV. Fully homomorphic encryption with relatively small key and ciphertext sizes. Public Key Cryptography, LNCS. 2010; 6056:420-43.
11. Damien Stehle DS, Ron Steinfeld RS. Faster fully homomorphic encryption. Advances in Cryptology-ASIACRYPT, LNCS. 2010; 6477:377-94.
12. ZvikaBrakerski ZB, Vinod VVV. Efficient fully homomorphic encryption from (standard) LWE. Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science, LWE, FOCS; 2011. p. 97-106.
13. Zvika Brakerski ZB, Vinod Vaikuntanathan VV. Fully homomorphic encryption from ring-LWE and security for key dependent messages. Advances in Cryptology-CRYPTO, LNCS. 2011; 6841:505-24.