International Journal of Security and Its Applications
Vol. 9, No. 5 (2015), pp. 319-324
http://dx.doi.org/10.14257/ijsia.2015.9.5.31

Homomorphic Encryption for Cluster in Cloud

N. Vamshinath[1], K. Ruth Ramya[1], Sai Krishna[1], P. Gopi Bhaskar[1], Geofrey L. Mwaseba[1] and Tai-hoon Kim[2]

[1]*Department of Computer Science and Engineering,*
KL University,
Vaddeswaram, AP, 522502, India
ramya_cse@kluniversity.in
[2]*Department of Convergence Security, Sungshin Women's University,*
249-1, Dongseon-dong 3-ga, Seoul, 136-742, Korea
taihoonn@daum.net
(Corresponding Author)

Abstract

Data storage and its security have been a distress, since the development of the several computing capabilities. Any potential data mishandling can escalate to leakage or breaching, resultant of this can be decline or impinging of trust, privacy or economical stance of the related cloud delegates. Cloud computing is a new technological trend that enables outsourcing of data into the cloud aimed towards elimination of sneakernet as there is no need for rudimentary storage of data as previously in confined physical storages. During decryption data will be vulnerable for some instant of time, as the plain text can be expose, contrary to homomorphic encryption which promotes privacy of the secured data by allowing some operation to be performed on the encrypted data, The homomorphic encryption is presently available for traditional system, the same procedure is applied to the cloud data and in transit.

Keywords: *Homomorphic encryption, Cloud computing, Data security, Cluster*

1. Introduction

Since the prevalence of cloud computing as predicted by Gartner's Hype cycle [1] for the emerging technology. Cloud storage have become the predominant methodology for storing the data by utilizing the network interconnection. Although cloud storage have been a major game changer for our data storage, still there is a need to secure sensitive data (dynamic and static) such as tax records, social security number and other personal data related contents so as to ensure privacy. The cloud service as a mean for outsourcing data for storage purpose and manipulation of the data which is achieved by having high computational resources depending on the need of the cloud service based on self-demand, have been in scrutiny and it have arisen the fear of satisfactory privacy provision, as data will not be effectively secured enough to quench the qualms of the service utilizers.

This problem has awaken many viable solutions to curtail the existing problem. Encryptions of data before sending it to the cloud for storage have been a good technique adopted for several users of cloud service. Depending on cryptographic algorithms there are two types of algorithms based on key distribution namely symmetric key and asymmetric key cryptography. In the symmetric/convectional – key cryptography the sender and the recipient share a key which is known to the parties involved in the given cryptosystem, both the sender and the receiver. They are further subdivided into stream and block ciphers [16]. A stream cipher deals with bit by bit encryption while a block

ISSN: 1738-9984 IJSIA

cipher deals with group or specific fixed length bits of the data. Usually this kind of cryptography algorithms is relatively fast because there is a less burden for the system to evaluate the encrypted and decrypted content. Despite of the advantages it is still not preferable to utilize them as most of the cryptanalysis has opted for use of asymmetric key. Examples of such algorithms include RC4, 3DES, DES, AES, blowfish and so forth. While asymmetric/Public-key cryptography involves the use of separate keys, which are not the same for both sender and recipient, thus for this kind of key cryptography different keys will be used for decryption and encryption contrary to the symmetric key algorithm. Thus there arise a need to specify more keys in which a public key will be universally known to all the parties confined in the crypto-system and a private/secret key will be known only to a single user. This in turn helps to preserve confidentiality provided that the cryptosystem used is less vulnerable to attacks. For asymmetric/public-key cryptography a sender can encrypt with public key and the receiver can decrypt with private key or a sender can encrypt with private key and receiver can decrypt with public key. Several encryption techniques and methodology have been adopted but we will give in a nutshell preview of the encryption techniques used. Proxy based re-encryption [2] which was presented by Mambo and Okamoto [3] aimed at improving the traditional cryptosystem. This allowed the proxies to transfer the decryption rights from source to destination target, without giving out the private keys of the concerned parties. It can be utilized for file system security and email for the purpose of spam filtering. Identity Based Encryption, IBE [4] was a notion brought forward by Adi Shamir in 1984 and further developed in 2001 arose the first operational IBE scheme [5]. IBE is a public key based encryption in which a feasible public key can be whichever string; this was developed to help in managing the certificates in electronic mailing system.

There are two profound types [15] of encryption based on the characteristics of homomorphism, which are partially and fully homomorphic encryption. Partial homomorphic encryption enables the user to perform only a limited single operation on cipher text such as additive and multiplicative homomorphic operations at a time. And the real life application of it is electronic voting [14]. Examples of partial homomorphic schemes are El-Gamal and RSA which is unpadded. While full homomorphic encryption enables several operations to be performed on encrypted data without knowing the secret key, example of such schemes is DGHV scheme, Gentry full homomorphic scheme and BGH [15] scheme.

Below we will give overview for homomorphic encryption in further details. Thereafter we will describe DGHV [6] by van Dijk M., Gentry C., *et al.*, and GEN10 [7-9] by Craig Gentry. Imagine the possibility of being able for a third party to work on data and perform some simple computation on the data which is in encrypted format without disclosing the data, this promotes privacy in turn, and that is in short homomorphic encryption.

In homomorphic cryptosystem, both sets of all possible plaintexts and cipher texts are groups such that for any $\mathcal{K} \in \mathcal{K}$ and any two cipher texts $c_1 = e\mathcal{K}(m_1)$, $c_2 = e\mathcal{K}(m_2)$, the following condition holds:

$$d\mathcal{K}(c_1 \bullet c_2) = m_1 \bullet m_2 \quad (1.1)$$

$$d\mathcal{K}(c_1 + c_2) = m_1 + m_2 \quad (1.2)$$

In which • and + symbolizes the group operations in the given cipher texts, C and messages, M respectively for multiplication and additional.

Homomorphic cryptosystem can be regarded as an arbitrary box in which an operation can be done over two cipher texts and the result will be as follows:

$$d\mathcal{K}(e\mathcal{K}((m_1) \otimes (e\mathcal{K}(m_2)) = m_1 \bullet m_2 \quad (1.3)$$

$$d\mathcal{K}(e\mathcal{K}(m_1) \oplus (e\mathcal{K}(m_2)) = m_1 + m_2 \quad (1.4)$$

Where $\otimes$ and $\oplus$ denotes operations on cipher texts to obtain encrypted product or sum respectively for the given two messages in the plaintext.

2. Existing System

2.1. DGHV Scheme

In this scheme the message, м is a composition of bit such that м ∈ {0, 1} and the shared key, k is an integer which is positive and odd. p and q are random integers selected in such a manner that in the given interval hold this relation, | 2r |<k/2.

The plaintext messages during encryption can be attained by computational of the following parameters

$$c = м + kq + 2r \quad (2.1.1)$$

The crypto message during decryption can be obtained by evaluation of the following

$$(c \bmod p) \bmod 2 = м \quad (2.1.2)$$

Observe the following properties for homomorphic operation possible in the DGHV scheme¬ [6, 9]

Consider $c_i = м_i + q_{ik} + 2r_i$, for i=1 and 2 then

$$c_1 = м_1 + q1k + 2r_1 \quad (2.1.3)$$

$$c_2 = м_2 + q2k + 2r_2 \quad (2.1..4)$$

The possible homomorphic operations are addition and product that is c1 + c2 and c1• c2 respectively.

Addition

$$c_1 = м_1 + q1k + 2r_1 \quad (2.1.5)$$

$$c_2 = м_2 + q2k + 2r_2 \quad (2.1.6)$$

Thus $c_1 + c_2 \equiv м' + kq' + 2r'$

Where $м' = м_1 + м_2$, $q' = q_1 + q_2$ and $r' = r_1 + r_2$

Multiplication

$$c1 = м_1 + q1k + 2r_1 \quad (2.1.7)$$

$$c2 = м_2 + q2k + 2r_2 \quad (2.1.8)$$

Thus $c_1 \bullet c_2 \equiv м'' + kq'' + 2r''$

Where $м'' = м_1 \bullet м_2$, $q'' = м_1q_2 + м_2q_1 + kq_1q_2 + 2q_1r_2 + 2r_1q_2$ and, $r'' = м_1r_2 + м_2r_1 + 4r_1r_2$

The problem with this scheme, DGHV [11] is that after several manipulations for both addition and multiplication operations are carried out, it causes escalation of noises through which it will be hard to obtain the plaintext as the cipher text will tremendously grow in size.

2.2. Gentry Scheme (GEN 10)

Craig Gentry in 2010 proposed this scheme, hence termed as GEN10 [9]. In this scheme message, м is a composition of N-bits such м ∈ {0, 1}.

The ciphertext carries the following general format, c= kq + м where; c depicts a ciphertext, k assumes a shared key, q is a random integer with odd bits and m represent plaintext message.

The decryption for the given ciphertext can be obtained by evaluation of c mod q

Upon assuming that $c_i = м_i + kq_i$ for i=1 and 2 this follows that,

$$c_1 = м_1 + kq_1 \quad (2.2.1)$$

$$c_2 = м_2 + kq_2 \quad (2.2.2)$$

This scheme, GEN10 [7-10] have the following possible operations addition, subtraction and multiplication which are $c_1 + c_2$, $c_1 - c_2$ and $c_1 \cdot c_2$ respectively.

Addition

$$c_1 = м_1 + kq_1 \quad (2.2.3)$$

$$c_2 = м_2 + kq_2 \quad (2.2.4)$$

Thus $c_1 + c_2 \equiv м' + kq'$

Where $м' = м_1 + м_2$ and, $q' = q_1 + q_2$

Subtraction

$$c_1 = \mathit{м}_1 + kq_1 \quad (2.2.5)$$

$$c_2 = \mathit{м}_2 + kq_2 \quad (2.2.6)$$

Thus $c_1 - c_2 \equiv \mathit{м}' + kq'$

Where $\mathit{м}' = \mathit{м}_1 - \mathit{м}_2$ and, $q' = q_1 - q_2$

Multiplication

$$c_1 = \mathit{м}_1 + kq_1 \quad (2.2.7)$$

$$c_2 = \mathit{м}_2 + kq_2 \quad (2.2.8)$$

Thus $c1 \cdot c2 = \mathit{м}' + kq''$

Where $\mathit{м}' = \mathit{м}_1 \cdot \mathit{м}_2$ and, $q'' = \mathit{м}_1 q_2 + \mathit{м}_2 q_1$

For the GEN10 [11] scheme to be utilized the noises should be negligible enough such that the following relation should hold

$$|\mathit{м}_1 \triangle \mathit{м}_2| < k/2. \quad (2.2.9)$$

Where $\triangle$ assumes algebraic operations which are addition (+), subtraction (-) and multiplication (•).

2.3. SDC Scheme

This scheme, SDC [11] was derived by taking into consideration of the Gentry's cryptosystem [6-10]. It helps in recovery of cipher text by modification of the cipher text recovery algorithm. This enables to prevent exposing of the plaintext hence it is more secure.

Key-generation (k): The key is a random k-bit odd integer k.

Encryption (k, m): The cipher text carries the following general format

$$C = m + k + r * k * q, \quad (2.3.1)$$

Where r is a random number, q is a constant big integer, m is the plaintext message, and c is the cipher text.

Consider the possible homomorphic operation for addition and multiplication that is $c_1 + c_2$ and $c_1 \cdot c_2$ respectively.

Upon taking $c_i = m_i + k + r_i * k * q$ for i=1 and i=2 thus

$$c_1 = m_1 + k + r_1 * k * q \quad (2.3.2)$$

$$c_2 = m_2 + k + r_2 * k * q \quad (2.3.3)$$

Addition

$$c_1 + c_2 \equiv \mathit{м}' + 2k + r'*k*q, \quad (2.3.4)$$

where $\mathit{м}' = m_1 + m_2$ and, $r' = r_1 + r_2$

Multiplication

$$c_1 \cdot c_2 = \mathit{м}' + k'' + z''*k*q, \quad (2.3.5)$$

Where $\mathit{м}' = m_1 \cdot m_2$, $k'' = (m_1 + m_2 + k)k$

$$z'' = r_1(k + m_2 + r_2) + r_2(k + m_1) \quad (2.3.6)$$

Decryption: c mod k

Retrieval: C mod q where C is equal to the difference between c_i and c index.

In order for a client to retrieve the message contents of a specific index that is m index, then he delivers c index to the required server by using the encryption formula described

above and upon submitting the cipher text to the server, it computes C mod q if it is equivalent to zero. Then retrieval for the given cipher text is successful attained.

3. Proposed System

As referred to the below Figure every single node, d_i in a cluster, C have its own public key and private key. For all the available nodes d_1, d_2, ….. d_i must have a session key which is utilized for a particular instant of accessing the nodes in the cluster. When the user uploads the file (plaintext, m1) into the cloud it will be encrypted using RSA which makes use of the public key appended with session key to get the cipher text, c_1, Furthermore the plaintext m1 is encrypted by changing the parameters in public key and appending it again with session key, in turn we will get another cipher text, c_{11}. Thus arithmetic operations based on homomorphic properties can be applied to both the cipher texts c_1 and c_{11} to obtain a new cipher text, say, Ç. If a particular user want to download the data which is encrypted he must have a private key through which will be used for decryption.

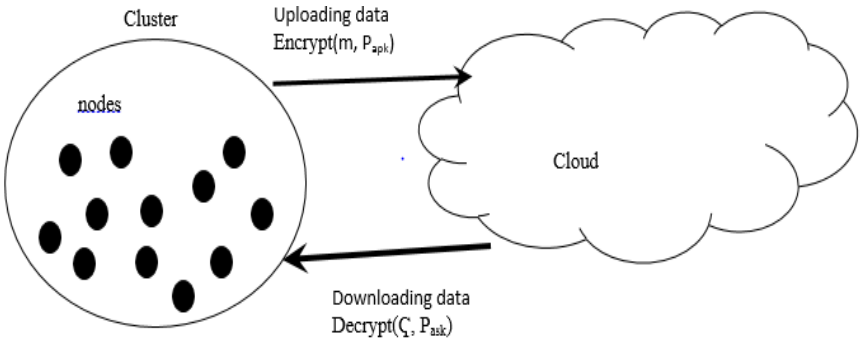

Figure 1. A Cluster which Composes of Nodes Uploading and Downloading of Data

Key Generation Step:

1. Generation of two random large prime numbers such that, the large prime numbers p and q which are not be equal.
2. n = p * q
3. ϕ(n)=(r-1)(k-1)
4. Choosing of an integer, ę such that it will be within the following range 1< ę <ϕ(n) and its GCD(ϕ(n), ę) = 1
5. Calculation of a secret d, for which (d * ę) mod ϕ(n) = 1, Choosing of an integer, ę such that it will be within the following range 1< ę <ϕ(n)
6. Appended Public key with session key, P_{apk} and appended Private key with session key, P_{ask} can be obtained by { ę,n} and {d,n} respectively.

Encryption:

encrypt (m, P_{apk})

The general format for the cipher text is $C= m^{ę} \bmod n$

For both cipher texts c_i and c_{ii} , $c_i = m_i^{ę} \bmod n$ and $c_{ii} = m_{ii}^{ę} \bmod n$ respectively

Decryption:

decrypt (Ç, P_{ask})

Evaluate $m = c^d \bmod n$

Evaluate:

Upon taking into account that RSA allows multiplicative homomorphic operation then the following can be considered;

If we have two cipher texts c_i and c_{ii} for $c_{ii} = m_i^{e} \bmod n$ and $c_{ii} = m_{ii}^{e} \bmod n$

$c_i \cdot c_{ii} = Ç = (m_i \cdot m_{ii})^{e} \bmod n$ which is multiplicative homomorphism thus upon decrypting we will get the corresponding plaintext which is equivalent to multiplication of the plaintexts.

4. Conclusion

Although there have been success in implementation of simple homomorphic encryption full homomorphic encryption remains a great challenge to deploy and implement it in the cloud. Provided that, fully homomorphic encryption will be given much emphasis and then effectively utilized, cloud data will be much more secured from encroachment as during the traditional methodology data was to be decrypted before doing operations on it this made the data to be vulnerable for particular time instant before decrypting it again. For, there is a need to adopt and implement a new security approach to enhance security for cloud service. With regards to the properties of the homomorphic encryption the possible operations that can be performed on the cipher texts can help to guarantee security for the data submitted to cloud.

References

[1] D. Chin, In Top 10 IT trends that will shape, vol. 2014, **(2013)**.
[2] A. Giuseppe, K. Fu, M. Green and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage", ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, **(2006)**, pp. 1-30.
[3] M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt cipher texts", IEICE transactions on fundamentals of electronics, Communications and computer sciences, vol. 80, no. 1, **(1997)**, pp. 54-63.
[4] A. Boldyreva, V. Goyal and V. Kumar, "Identity-based encryption with efficient revocation", Proceedings of the 15th ACM conference on Computer and communications security, pp. 417-426. ACM, **(2008)**.
[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", SIAM Journal on Computing, vol. 32, no. 3, **(2003)**, pp. 586-615.
[6] V. D. Marten, C. Gentry, S. Halevi and V. Vaikuntanathan, "Fully homomorphic encryption over the integers", Advances in cryptology–EUROCRYPT, Springer Berlin Heidelberg, **(2010)**, pp. 24-43.
[7] C. Gentry, "Fully homomorphic encryption using ideal lattices", Proceedings of the 41st annual ACM symposium on Theory of computing, ser. STOC '09. New York, NY, USA: ACM, **(2009)**, pp. 169-178.
[8] C. Gentry, "A fully homomorphic encryption scheme", PhD diss., Stanford University, **(2009)**.
[9] C. Gentry, "Computing arbitrary functions of encrypted data", Communications of the ACM, vol. 53, no. 3, pp. 97-105.
[10] J.-S. Coron, A. Mandal, D. Naccache and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys", Advances in Cryptology–CRYPTO, Springer Berlin Heidelberg, **(2011)**, pp. 487-504.
[11] J. Li, S. Danjie, S. Chen and X. Lu, "A simple fully homomorphic encryption scheme available in cloud computing", IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), vol. 1, **(2012)**, pp. 214-217.
[12] J.-S. Coron, D. Naccache and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers", Advances in Cryptology–EUROCRYPT 2012, Springer Berlin Heidelberg, pp. 446-464.
[13] Somee.com© https://somee.com/VirtualServer.aspx.
[14] A. Huszti, "A homomorphic encryption-based secure electronic voting scheme", Publ. Math. Debrecen vol. 79, **(2011)**, pp. 3-4.
[15] J. M. Shah and H. kothadiya, "A Survey on Homomorphic Encryption Techniques in Cloud Computing", International Journal of Advanced Engineering and Research Development, vol. 2, Issue 2, **(2015)** February, pp. 234-242.
[16] P. Garg and J. S. Dilawari, "A Review Paper on Cryptography and Significance of Key Length", International Journal of Computer Science and Communication Engineering IJCSCE Special issue on "Emerging Trends in Engineering" ICETIE, **(2012)**, pp. 88-91.

