A Novel Homomorphic Encryption Technique for Generation of Keys Using Cluster Classification for Cloud Security

D.Chandravathi, Dr.P.V.Lakshmi

Abstract— In the past decade, Cloud Security has been the most promising innovation in the computing world. Related with critical data, its usage is still hindered with the concern of security. To bridge the security gap, the most widely used technique is encryption of remotely stored data. The security on cloud is still at big risk due to vast usage of data storage. Computing environments change. This change presents fundamental challenge of outsourcing computation, which motivates the computing power of asymmetry available. In recent computing scenarios clients are trusted (but weak), while computationally strong servers are untrusted. Homomorphic Encryption is a good idea which enhances the security measures of un-trusted systems or applications that stores and manipulates sensitive data proposed on cloud. It is the conversion of data into cipher text that can be analyzed and worked with as if it were still in its original form. It allows complex mathematical operations to be performed on encrypted data without compromising the encryption. This paper presents an effective solution to enhance the security in cloud by using RSA encryption scheme along with clustering of prime numbers for the generation of keys for encryption and decryption. The weakness of RSA lies in the generation of Prime numbers. This is achieved by a new classification technique. By classifying the keys, elimination of redundant messages is done on the same values of the product of two prime numbers. Hence, Security is enhanced.

Index Terms— Cloud Security , RSA Algorithm, Homomorphic Encryption, Clustering .

IJSER

———————————— ◆ ————————————

1 Introduction

The information technology model for computing is composed of all the IT components like hardware, software, networking, and services. It is necessary to enable the development and delivery of cloud services via the Internet or a private network. Cloud Provider and Cloud User are the prominent actors in Cloud Computing. Cloud Provider is the enterprise that provides cloud services [2]. A Cloud Users are organizations, educational institutes, individuals utilizing the cloud services. Hence, there is a necessity for security, confidentiality and visibility with respect to the cloud providers [3]. The main aspect is to protect the data from hacking.

At present both in Public Cloud and Private Cloud, security should be provided to encrypt data that is stored and also to provide secure transmission from a local machine to a cloud data store. The stored data is encrypted and the channel of data transmission is well secured with key exchanges. But actually performing computations on the data stored in the cloud requires decrypting it first, which makes critical data available to the cloud provider. Data Mining and other Data Analysis onto the Encrypted Database is a far distant thing to achieve by using available encryption standards[3]. The proposal here is to encrypt data before sending to the cloud providers. Thereby performing computations on clients' data at their request, such as analyzing sales patterns, without exposing the original data. To achieve this it is also necessary to hold the cryptosystems based on Homomorphic Encryption either a Fully Homomorphic Encryption (FHE) or Somewhat Homomorphic Encryption (SHE)[3][4].

2 Background

In 1978, the notion of encryption schemes that permits nontrivial computation on encrypted data was first proposed by Rivest, Adleman and Dertouzos. Rivest proposed the exponentiation function and the RSA function as additive and multiplicative privacy homomorphisms, respectively [1]. Note, however, that neither of these functions by themselves provides even chosen plaintext security.

There are several partially homomorphic cryptosystems, and also a number of fully homomorphic cryptosystems. Although a cryptosystem which is unintentionally malleable can be subject to attacks on this basis, if treated carefully homomorphism can also be used to perform computations securely [2].

Partially homomorphic cryptosystems

a. Unpadded RSA
  - ElGamal
  - Goldwasser–Micali
  - Benaloh
  - Paillier
  - Other partially homomorphic cryptosystems

b. Fully homomorphic encryption

Early homomorphic cryptosystems
  - Gentry's cryptosystem
  - Cryptosystem over the integers

The 2nd generation of homomorphic cryptosystems□

3 Homomorphic Encryption

Encryption has traditionally been viewed as a mechanism that enables secure communication. In particular, Public-key Encryption provides a way for Alice to encrypt a message into a ciphertext using Bob's public key, and for Bob to decrypt the ciphertext to obtain the message using his secret key[3]. In this view of encryption schemes, access to encrypted data is all or nothing – having the secret decryption key enables one to learn the entire message, but without the decryption key, the ciphertext is completely useless.

Can we do arbitrary computations on data while it remains encrypted, without ever decrypting it? This state of affairs raises an intriguing question, first posed by Rivest, Adleman and Dertouzos in 1978, which promoted the idea of performing computations on encrypted data without being able to "see" the data. Such ability also gives rise to a number of useful applications including the ability to privately outsource arbitrary computations to the "cloud" and the ability to store all data encrypted and perform computations on encrypted data, decrypting only when necessary[7].

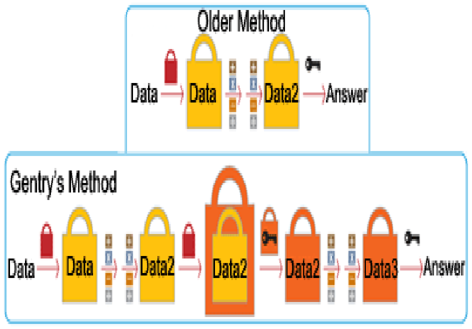

Fig 1.1

Fully Homomorphic encryption is a special type of encryption system that permits arbitrarily complex computation on encrypted data. Homomorphic encryption is the conversion of data into ciphertext that can be analyzed and worked with as if it were still in its original form. Homomorphic encryption is expected to play an important part in cloud computing, allowing companies to store encrypted data in a public cloud and take advantage of the cloud provider's analytic services.

Homomorphic encryption permits computing on encrypted data. That is, the client can encrypt his data *x* and send the encryption **Enc(*x*)** to the server. The server can then take the ciphertext **Enc(*x*)** and evaluate a function *f* on the underlying *x* obtaining the ***encrypted*** result **Enc(*f*(*x*))[2][5].** The client can decrypt this result achieving the wanted functionality, but the server learns nothing about the data that he computed on. Homomorphic encryption is **functional encryption**, where our goal is to reveal the result of the computation to the server, but protect all other information about our encrypted input [3].

For example, a user sends a request to add the numbers 1 and 2, which are encrypted to become the numbers 33 and 54, respectively. The server in the cloud processes the sum as 87, which is downloaded from the cloud and decrypted to the final answer, 3.A normal symmetric cipher -- DES, AES is not homomorphic[2]. The RSA algorithm is homomorphic but only with respect to multiplication.

Homomorphic encryption schemes are malleable by design. Malleability is a property of somecryptographic algorithms[2]. An encryption algorithm is malleable if it is possible for an adversary to transform a ciphertext into another ciphertext which decrypts to a related plaintext. That is, given an encryption of a plaintext m, it is possible to generate another ciphertext which decrypts to $f(m)$, for a known function f, without necessarily knowing or learning m[4].

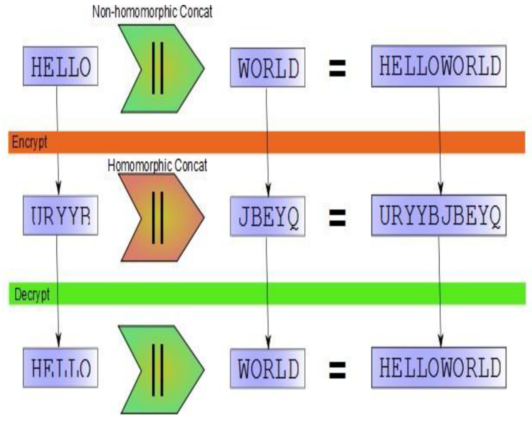

Fig 1.2

4 Implementation on cloud

In 2009 Craig Gentry of IBM has proposed the first encryption system "Fully Homomorphic" that evaluates an arbitrary number of additions and multiplications and thus calculate any type of function on encrypted data[1][3]. The internal working of this adds another layer of encryption every few steps and uses an encrypted key to unlock the inner layer of scrambling. This decryption "refreshes" the data without exposing it, allowing an infinite number of computations on the same.

The application of fully Homomorphic encryption is an important brick in Cloud Computing Security; more generally, outsourcing of the calculations on confidential data to the Cloud server is possible, keeping the secret key that can decrypt the result of calculation[3]. In our implementation, we analyze the performance of existing homomorphic encryption cryptosystems, and are working on a virtual platform as a Cloud server, a VPN network that links the Cloud with the customer, and then simulating different scenarios[3][4].

5 Methodology

This method follows RSA implementation. The encryption process is carried out with tow prime numbers p,q. This papers gives a clear cut approach to the generation to prime numbers ,which are clustered and solves the problem of redundant messages. The clustering simplifies the selection of prime numbers which are nearest to each other. By doing the most common attacks which are vulnerable is reduced [16].So a situation in which the cipher text is the same as the plaintext in some values of n which is the product of two prime numbers p and q are resolved and messages which are redundant is eliminated.

The proposed method aims at classification of prime numbers for key generation. In this method, an agreement is done for communication with the parties with a secure set of alternative prime numbers (PR). From this set they can choose alternative values of prime number for p or q or both. In addition, this set of prime numbers is divided into different classes. Each class contains a specified numbers of primes. We generate the prime numbers by taking odd numbers within the range N. This is a process of filtration. We eliminate all even numbers since they are not prime. This clustering is done by taking into the account of the number of cluster or classes. Then each of the prime numbers starting from the beginning are taken one at a time and grouped one by one, cluster by cluster ie., 2 in c1, 3 in c2, 5 in c3, 7 in c4 and so on .

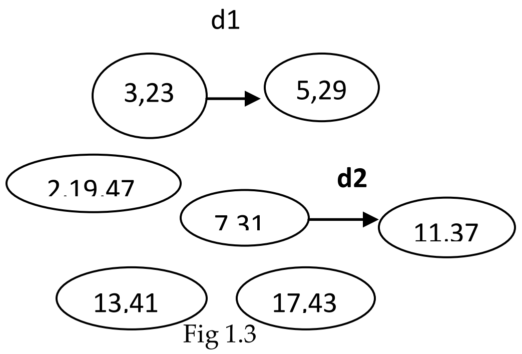

Fig 1.3

The number of clusters must be less than the half of the range of N and prime numbers within the range and can be limited. To select a certain neighbor of one of the classes in that set it is dependent on a secure distance (d1). This distance will be used to choose one prime number or both. By assigning another secure distance (d2) inside the selected class we generate the keys for encryption [16]. The purpose of the distance (d1) is to use an agreement secure parameter to choose one of the classes inside the set of all classes and this distance must be changed periodically to remove the redundant messages and to enhance more security for the RSA algorithm. Ciphers are generated for the corresponding keys due to a specific value of n and we can generate a new secure value of n to overcome these redundant values of messages.

The purpose of the distance (d1) is to fine the distance which is used as an agreement secure parameter of the classes inside the set of all classes. By doing so the distance is changed periodically to remove the redundant messages which enhance more security for the RSA algorithm [8]. Another secure parameter is the distance (d2), which is the distance to select the nearest neighbour of primes inside the selected class by distance (d1).

Selection of the prime number 'p' is done from the clusters say p'. Then we compute 'n'ie n'. An agreement secure parameters is generated. In order to acknowledge the receiver by changing 'n', the sent ciphertex must be appended by a secure agreement parameter, denoted by f, inside the ciphetext[9][10] . This suggested parameter is used to prevent sending the value of alternative value as a public key, so we get a more secure procedure for RSA algorithm by reducing the public key into one parameter that is the public key of the user only because in the traditional method of RSA, the public key consists of two parameter; the value of n and the public of the user (e)[11].

6 Algorithm

The Algorithm has three phases:

Clustering Algorithm

1. Let C be the cluster where c1,c2,c3…cn be subsets of C.
2. Enter C value. Ex C=5.
3. Let N be the number of prime numbers starting from 2.
4. Input N. Say N=50.
5. Eliminate all even numbers with in N value.
6. Let it be N1.
7. Then select all the prime numbers from N1.
8. Depending on C, Place the numbers one by one in each cluster as shown in fig 1.3.
9. Now choose the one prime number from one of the cluster.
10. Select the next prime number and find the nearest from the first by Euclidean distance.

Key Generation:

Choose two prime numbers from PR
n= p*q
Ø (n)=(p-1)*(q-1)
Let e be the public key
Let d be the private key
c= m^e mod n.
if c=m then

Sender operation:

1: Choose d1 of the one of subsets Ci in S for the secure class
2: Choose d2 inside Ci to pick one alternative prime p'
3: Compute n'=p'*q
4: Compute Ø (n')=(p'-1)*(q-1)
5: Choose alternative public key , lets e'
6: Generate the corresponding private key d'
7: Compute the ciphertext C'=$m^{e'}$ mod n'
8: Combine the agreement factor f with the new ciphetext and send C'' as:
C''=[C',f]

Multiplicative Homomorphic encryption:

*Generate two ciphers and s*uppose we have two ciphers C1 and C2 such that:
C1 = $m1^e$ mod n
C2 = $m2^e$ mod n
C1.C2 = $m1^e m2^e$ mod n = $(m1m2)^e$ mod n

PLAIN TEXT MESSAGE :

P1=36489656439701145972457523098132091447225668989528382573694967128594581341696354706942814918028461206078760419576264857826027092284914144780780178579009996542341516588962738284849075319698448391792163349917312048380078921817475348243532804152596828268924244844990873836766877568082509832988033782159782 41

P2=49518741130087182465637648778236384775182912156967721238994295457070053629048845086663658191635002703350352059016504245111847283095045661395721838825182381783961806338360633127086800719999410156920479487110084831319772491986370

7351780704405393391994884673314421978369697599
22869057504050347880295257008984226

CIPHERTEXT

C1=4309806803004551571243529055103100691323093
0156162662286651969290649839951713817432670747
5111906742497348974548364196871115298821065461
2746688048470072863780344978020290306540677349
1256356811091040601916581354421393941333651882
9979873903310013049936429396863299883766901028
99270227169764349991246927004094271111

C2=4113058132856990303207440829066756506737095
9359548059460155417705292755907344103405740522
5591834709667660745524543881188996216300282348
3264243856971990446848940289537617738142124130
9469296967727137872569988498570962177744307331
5399832348842073472460444033155410993038278439
4912755453980703456543143775430976 7 9

Output:

7726485922140255511988936390222013964175744997
2836056316936741506427346711690917848086879214
7276276186521095992120791621422052510115493454
4727877834671636615096126321046003694189150294
8773547117970857756607928191458691923278297087
4624469518732614669041990897709591965751322073
1740475003375822929396048385876533991863703011

IJSER

1808129272918289813391501678516780951705471176
0756024701190106965445598519391118937364897198
4551451032570077632163732770686132777056321017
7127221171609831566617631127320748428036941983
6013861473726182630676291593397264858179769773
3737904109746852294911665593356342273834459675
39464928379475833 7

Results:

Fig 1.4

	K B	K B	K B	KB	KB	KB	KB
file size	26	45	50	100	250	512	800
executio n time	19	37	43	102 3	246 2	283 2	313 6

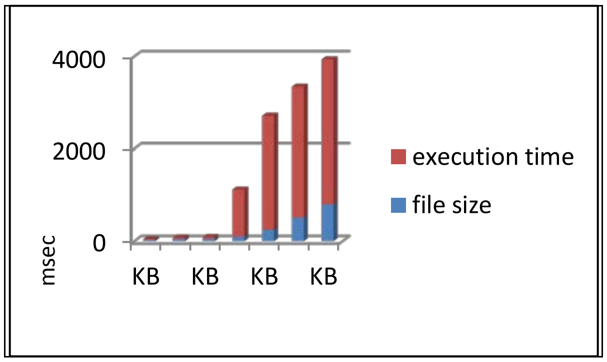

Fig: 1.5

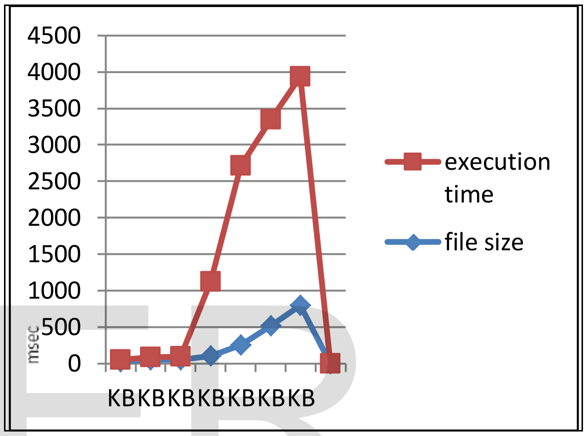

Fig: 1.6

Conclusion

The cloud security is based on Homomorphic encryption, is a new concept of security which enables us to provide results of calculations on encrypted data without knowing the raw data on which the calculation was carried out, with respect of the data confidentiality. Our work is based on the application of partially Homomorphic encryption to the Cloud Computing security . It also analyzes and the improvement of the existing cryptosystems to allow servers to perform various operations requested by the client. The improvement of the complexity of the Homomorphic encryption algorithms and compare the response time of the requests to the length of the public key is to be considered. Also this method reduces the redundant messages occurred in RSA method. We see that for some values of n, there is a major problem in which the message and its corresponding ciphetext are the same. At the presence of recent active attacks, this problem can be exploited by many attackers. For this reason, this method presents an active solution by changing the value of n.

References

[1] Craig Gentry, A Fully Homomorphic Encryption Scheme, 2009.http://crypto.stanford.edu/craig/craig-thesis.pdf.
[2] Understanding Homomorphic Encryption http://en.wikipedia.org/wiki/Homomorphic_encryption.
[3] Computing Blindfolded: New Developments in Fully Homomorphic Encryption Vinod Vaikuntanathan.
[4] A Fully Homomorphic Encryption Implementation on Cloud Computing Shashank Bajpai and Padmija Srivastava Cloud Computing Research Team, Center for Development of Advanced Computing [C-DAC], Hyderabad.
[5] Homomorphic Encryption Applied to the Cloud Computing Security Maha TEBAA, Saïd EL HAJJI, Abdellatif EL GHAZI.
[6] Vic (J.R.) Winkler, "Securing the Cloud, Cloud Computer Security, Techniques and Tactics", Elsevier.
[7] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 21(2):120-126, 1978.
[8] R. L. Rivest, A. Shamir and L. Adleman "A method for obtaining digital signatures and public – key cryptosystems" Communications of the ACM, vol. 21,pp. 120 - 126, 1978.
[9] W. Stallings "Network and internetwork security: principles and practice" Prentice - Hall, Inc., 1995.
[10] W.Stallings "Network security Essentials: Applications and Standards" Pearson Education India, 2000.
[11] J. Joshi, et al. "Network Security" Morgan Kaufmann, 2008.
[12] W. Stallings "Cryptography and network security vol. 2" prentice hall, 2003. K. Ming Leung , k-Nearest Neighbor Algorithm for Classification , POLYTECHNIC UNIVERSITY Department of Computer Science / Finance and Risk Engineering , 2007.
[13] C .Aayush ,and M . Srushti ," Modified RSA Algorithm: A Secure Approach ", CSDL HomeCCICN2011Computational Intelligence and Communication Networks, International Conference on 2011.
[14] V.Kuldeep, Kr.Rajesh,and C .Ritika , "Modified Prime Number Factorization Algorithm (MPFA) For RSA Public Key Encryption", International Journal of Soft Computing & Engineering 2012 .
[15] R.S. Dhakar, and P. Sharma, " Modified RSA Encryption Algorithm (MREA)" , Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on , Page(s): 426 – 429, 7-8 Jan. 2012 .
[16] A Modified RSA Algorithm for Security Enhancement and Redundant Messages Elimination Using K-Nearest Neighbor Algorithm , Dr. Abdulameer K. Hussain ,Computer Science Department, Jerash University,Jerash, 00962-02, Jordan, IJISET , Vol. 2 Issue 1, January 2015.

IJSER