International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 15 (2014), pp. 1519-1530
© International Research Publications House
http://www. irphouse.com

# Homomorphic Encryption Method Applied to Cloud Computing

**Iram Ahmad [1] and Archana Khandekar [2]**

*PG Student, Maharashtra Institute Of Technology(MIT), Pune*
*Maharashtra Institute Of Technology(MIT), Pune*

## Abstract

"Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertext and obtain an encrpted result which when decrypted matches the result of operations performed on the plaintext". For example, a person can add two encrypted numbers and then the second person can decrypt the result, without being able to find the value of the individual numbers. When the data is transferred to the cloud we use standard encryption methods to secure this data, but when we want to do the calculations on data located on a remote server, it is necessary that the cloud provider has access to the raw data, and then it will decrypt them. As we all know, the demand for privacy of data and algorithms to handle the information of enterprise has increased tremendously over the last decades. To achieve this, technology such as data encryption methods with the use of tamper-resistant hardware is used. However, a critical problem arises when there is a requirement of computing on such encrypted data (publicly) where privacy is established. Hence, the homomorphic cryptosystems can be applied in this case. It is a method that enables us to perform computations on encrypted data without decryption. In this paper we propose the application of a method to perform the operation on encrypted data without decrypting and provide the same result as well that the calculations were carried out on raw data and I use proxy re-encryption technique that prevents ciphertext from chosen cipher text attack.

**Keywords:** Cloud Computing, Homomorphic Encryption, Paillier, RSA, Security.

## Introduction

Homomorphic encryption has been used for supporting simple aggregations, numeric calculations on encrypted data as well as for private information retrieval. Recently, theoretical breakthroughs on homomorphic encryption resulted in fully homomorphic encryption, which is able to compute arbitrary functions on encrypted data. As a result, homomorphic encryption is generally believed to be the Holy Grail for solving database queries on encrypted data. The demand for privacy of digital data and of algorithms for handling more complex structures have increased exponentially over the last decade. This goes in parallel with the growth in communication networks and their devices and their increasing capabilities. At the same time, these devices and networks are subject to a great variety of attacks involving manipulation and destruction of data and theft of sensitive information. For storing and accessing data securely, current technology provides several methods of guaranteeing privacy such as data encryption and usage of tamper- resistant hardware. However, the critical problem arises when there is a requirement for computing (publicly) with private data or to modify functions or algorithms in such a way that they are still executable while their privacy is ensured. This is where homomorphic cryptosystems can be used since these systems enable computations with encrypted data.

Our basic concept was to encrypt the data before sending to the service provider. But there is a problem still faced by the client. Because the service provider needs to perform the calculations on data to respond the request from the client so he must provide the key to the server to decrypt the data before execute the calculations required, which might affect the confidentiality of data stored in the cloud. A method enable to perform the operations on encrypted data without decrypting them is homomorphic encryption. In our research we try to deal the problem of security of data hosted in a Cloud Computing provider. We all know that the cloud or on-demand computing brings a lot of advantage to the computer science of today and tomorrow. But the adoption of Cloud passage applies only if the security is ensured. How to ensure better data security and how a client can keep their private information confidential? There are two major questions that present a challenge for providers of Cloud Computing. In this work we focus the application of Homomorphic Encryption of the security of Cloud Computing, particularly the possibility to execute the calculations of confidential data encrypted without decrypted them. In Section B, we introduce the concept of Cloud Computing and the necessity to adopt Homomorphic Encryption to secure the calculation of data hosted by the Cloud provider. In section C, we define Homomorphic Encryption and we illustrate some examples of existing Homomorphic cryptosystems. In section D, we present some of the problems in the existing system. In section E we have defined our proposed system. The conclusion and perspectives are mention in section F.

## Cloud Computing

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. There are many problem related with cloud computing traffic, security and resource management. We can provide security in cloud by many ways like on data, network and storage. Homomorphic encryption method provides more security on data because provider is not involving in key management. I have use proxy re-encryption technique that prevents ciphertext from chosen cipher text attack. This system is more secure than existing system

**Definition:**
By Cloud Computing we mean: The Information Technology (IT) model for computing, which is composed of all the IT components (hardware, software, networking, and services) that are necessary to enable development and delivery of cloud services via the Internet or a private network. This definition has no notion of security for data in the cloud computing even if it's a very new. Cloud providers like: IBM, Google and Amazon use the virtualization in their Cloud platform, and in the same machine can coexist the storage space and treatment virtualized which belong to the concurrent enterprises.

**Cloud Computing Architecture:**
Cloud computing system is divided into two sections: the front end and the back end. Front end through which user can interact with the server and backend is the server which provides data to the client. Between server and client network is working as middleware.

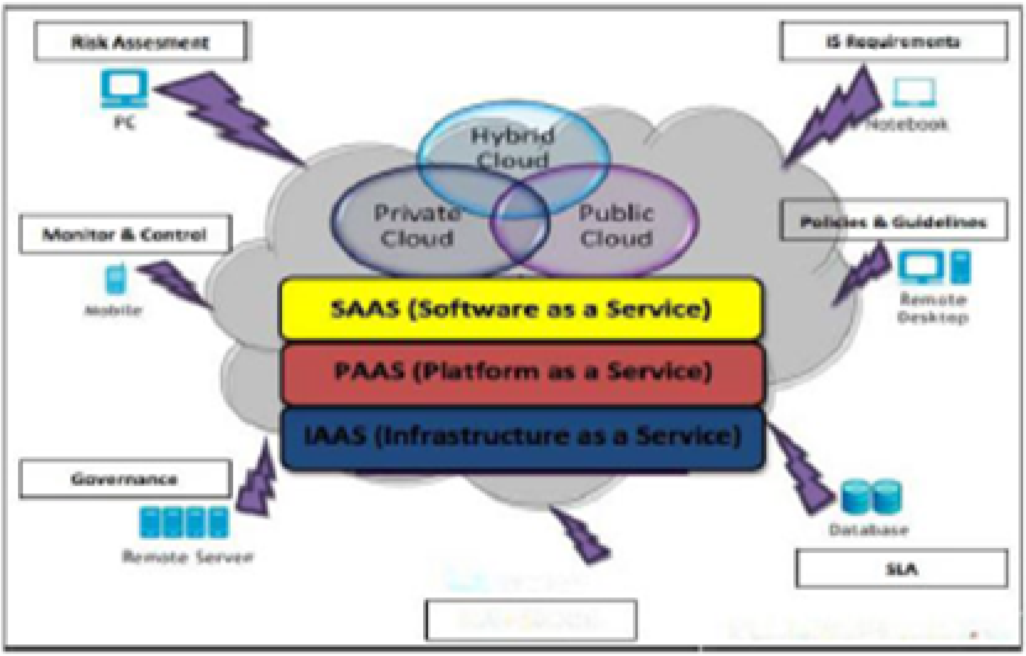


**Fig.1 Cloud Architecture**

**Homomorphic Encryption**

Homomorphic encryption alludes to encryption where plain texts and cipher texts both are treated with an equivalent algebraic function. Homomorphic Encryption allows server to do operation on encrypted data without knowing the original plaintext.

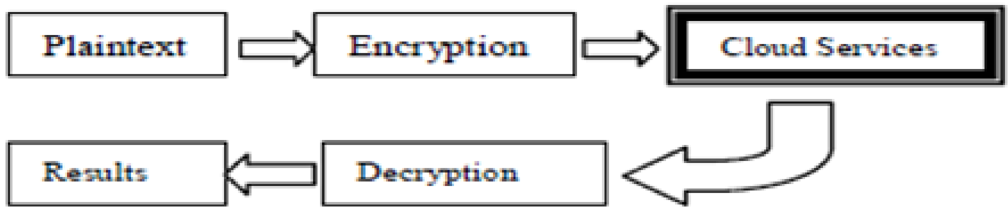


**Fig. 2 A general framework for data protection over cloud**

Homomorphic encryption allows complex mathematical operations to be performed on encrypted data without using the original data. For plaintexts X1 and X2 and corresponding ciphertext Y1 and Y2, a Homomorphic encryption scheme permits the computation of X1 Θ X2 from Y1 and Y2 without using P1 Θ P2.The cryptosystem is multiplicative or additive Homomorphic depending upon the operation Θ which can be multiplication or addition.

**Existing System**

A Homomorphic encryption has different Homomorphic schemes according to its properties:

**Table 1. Homomorphic Encryption Schemes**

| Scheme | Homomorphic properties | Algorithm |
|---|---|---|
| RSA | Multiplicative | Asymmetric |
| Elgaml | Multiplicative | Asymmetric |
| Goldwasser Micali | XOR | Asymmetric |
| Benalh | Additive | Symmetric |
| Paillier | Additive | Asymmetric |
| Okamoto uchiyama | Additive | Asymmetric |

**Additive homomorphic encryption**

A Homomorphic encryption is additive, if:

$$Enc(x \oplus y) = Enc(x) \otimes Enc(y)$$

$$Enc\left(\sum_{i=1}^{1} m_i\right) = \prod_{i=1}^{1} Enc(m_i)$$

**Table 2: Paillier Cryptosystem (1999):**

| **Key Generation: KeyGen(p,q)** | **Encryption: Enc(m, pk)** | **Decryption: Dec(c, sk)** |
|---|---|---|
| Input: p , q € P | Input: m € $Z_n$ | Input: C € $Z_n$ |
| Compute: n=p*q, and λ=lcm(p-1)(q-1)<br>Choose g € $Z_n$ such that Gcd(L(g^λ mod $n^2$), n)=1<br>With L(u)=(u-1)/n | Choose r € $Z_n$<br>Compute: c= $g^m * r^n$ mod $n^2$ | Compute:<br>m= mod n [L(c^λ mod $n^2$)/L(g^λ mod $n^2$)] |
| Output: (pk , sk)<br>Public key: pk=(n, g)<br>Secret key: sk=(p,q) | | Output: m € $Z_n$ |

Suppose we have two ciphers $C_1$ and $C_2$ such that:
$C_1=g^{m1}. r_1^n \bmod n^2$
$C_2=g^{m2}.r_2^n \bmod n^2$
$C_1. C_2= g^{m1}.r_1^n.g^{m2}.r_2^n \bmod n^2= g^{m1+m2}(r_1r_2)^n \bmod n^2$

So, Paillier cryptosystem realizes the property of additive Homomorphic encryption. An application of an additive Homomorphic encryption is electronic voting: Each vote is encrypted but only the "sum" is decrypted.

**Multiplicative Homomorphic Encryption**
A Homomorphic encryption is multiplicative, if:
$Enc(x \otimes y)= Enc(x) \otimes Enc(y)$
$Enc(\prod_{i=1}^{1} m_i)= \prod_{i=1}^{1} Enc(m_i)$

**Table 3: RSA Cryptosystem (1978)**

| **Key generation KeyGen(p,q)** | **Encyption: Enc(m,pk)** | **Decryption: Dec(c, sk)** |
|---|---|---|
| Input: p,q € P | Input: m€ Zn | Input: c € Zn |
| Compute: n=p*q, and ∅(n)=(p-1)(q-1) | Compute: c =$m^e$ mod n | Compute: m=$c^d$ mod n |
| Choose e such that Gcd(e,∅(n))=1<br>Determine d such that e * d≊1mod ∅(n) | Output: c € Zn | Output: m € Zn |
| Output: (pk ,sk)<br>Public key: pk=(e,n)<br>Secret key: sk=(d) | | |

Suppose we have two ciphers $C_1$ and $C_2$ such that:
$C_1 = m_1^e \bmod n$
$C_2 = m_2^e \bmod n$
$C_1 . C_2 = m_1^e m_2^e \bmod n = (m_1 m_2)^e \bmod n$

So, RSA cryptosystem find the properties of the multiplicative Homomorphic encryption, but does not satisfied good notions of security, Because if we assume two ciphers $C_1$, $C_2$ corresponding to the messages $m_1$, $m_2$, respectively, so :
$C_1 = m_1^e \bmod n$
$C_2 = m_2^e \bmod n$

The transmitter sends the pair ($C_1$, $C_2$) to the Cloud server, the server will perform the calculations requested by the client and sends the encrypted result ($C_1 X C_2$) to the customer. If the attacker intercepts two ciphers $C_1$ and $C_2$, which are encrypted with the same key, it will be able to decrypt all messages exchanged between the two communications because the Homomorphic encryption is multiplicative, i.e. the product of the ciphers equal to the cipher of the product.

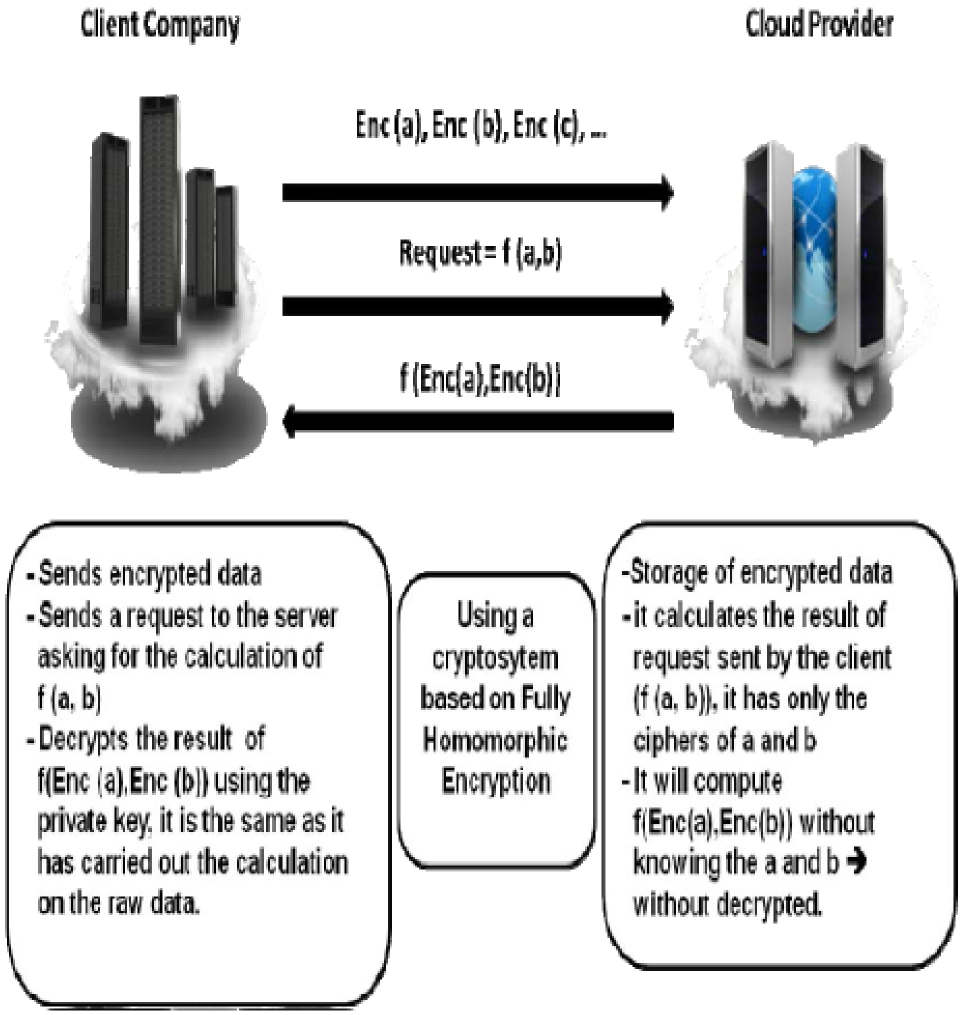


**Fig. 3 A general framework for cloud service with data protection**

**Problem In Existing System**

Suppose we have two ciphers C1 et C2 such that:

C1 = m1e mod n
C2 = m2e mod n
C1.C2 = m1em2e mod n = (m1m2)e mod n

RSA cryptosystem is working with property of multiplicative Homomorphic encryption, but it has a lake of security, because if we have two ciphers C1, C2 corresponding respectively to the messages m1,m2 so:

C1 = m1e mod n
C2 = m2e mod n

The client sends the pair (C1, C2) to the Cloud server and server performs the calculations requested by the client and sends the encrypted result (C1 × C2) to the client. If the attacker intercepts two ciphers C1 et C2, which are encrypted with the same private key, so they are able to decrypt all messages exchanged between the server and the client. Because the Homomorphic encryption is multiplicative, i.e. the product of the ciphers equals the cipher of the product.

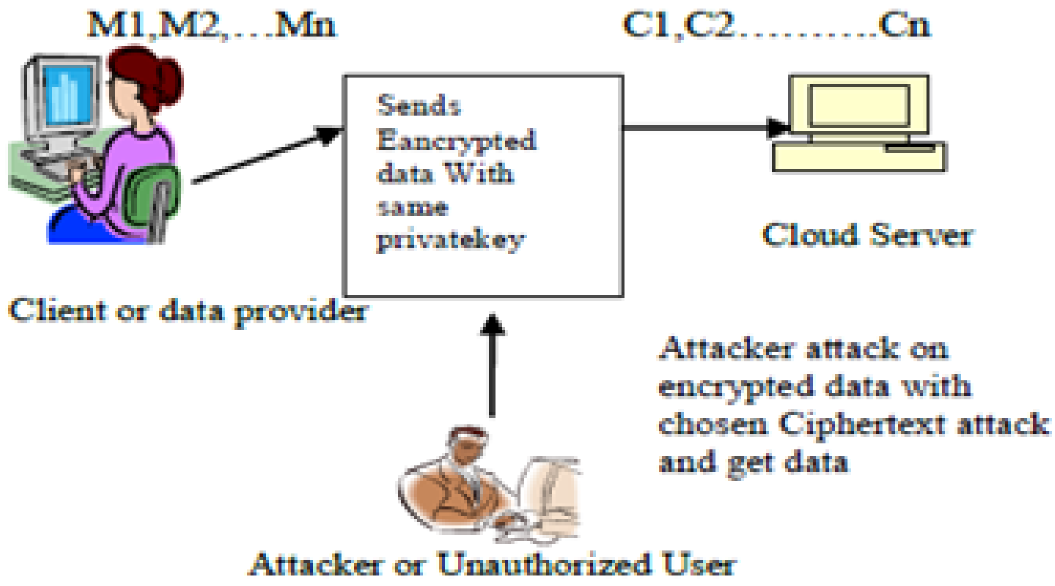


**Fig. 4 Existing System Model**

The basic RSA algorithm and Paillier Cryptosystem is vulnerable to chosen ciphertext attack (CCA).CCA is defined as an attack in which adversary chooses a number of ciphertext and is given the corresponding plaintext, decrypted with the target's private key. Thus the adversary could select a plaintext, encrypt it with the target's public key and then be able to get plaintext back by having it decrypted by private key. So attacker will know the entire data in-between client and cloud server.

**Proposed System**

To prevent cipher data from CCA (chosen ciphertext attack) I propose Proxy Re-Encryption algorithm with paillier and RSA Cryptosystem. In Homomorphic encryption scheme data was encrypted by the private key and public key was kept with client only. We again pass that data in proxy re-encryption algorithm and get every time random key generated cipher data. If attacker gets that key ones then they need to decrypt that data twice with two different keys. If once attacker gets the plaintext than he is not able to get every plaintext between client and server. So this system provides more security than existing system.

**Table 4: Proxy Re-encryption Algorithm**

| **Key Generation-keygen(p,q)** | **Encryption- Enc(c, Rpk)** | **Decryption: Dec(rc, Rsk)** |
|---|---|---|
| Choose two prime numbers p and q | Let m be the message to be encrypted where m $\in Z_n$ | Ciphertext c $\in Z_n$ |
| Compute n=p.q, and ∅(n)=(p-1)(q-1)<br>Choose e such that gcd(e, ∅(n))=1 | Compute ciphertext as:<br>Rc= m.e mod n | Compute message as:<br>M=c.d mod n |
| Determine d such that e.d=1 mod ∅(n) | | |
| Proxy public key is generated Rpk= (e,n)<br>Proxy private key is generated<br>Rsk= d | | |

**Table 5: Proposed Proxy Re-Encryption Based Paillier Algorithm**

| **Key Generation** | **Encryption: Enc(m,pk)** | **Proxy Re-Encryption(c)** | **Decryption: Dec(c, sk)** |
|---|---|---|---|
| Choose two large prime numbers p and q randomly and independently of each other such that gcd (pq ,(p-1)(q-1))=1. | Let m be the message to be encrypted where m $\in Z_n$ | Compute Private and Public key.(Rsk,Rpk). | Ciphertext c Є Zn2*. |
| Compute n=pq and λ=lcm (p-1, q-1). | Select random where r Є Zn*. | Re Encrypt Ciphertext generated by Paillier algorithm and send Public key (Rpk) to cloud server. | Compute message: m=L(c λ mod n2)/ L (g λ mod n2). Mod n |
| Select random integer g where g Є Z*n2 | Compute ciphertext as: c=gm . rn mod n2. | | |
| . Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: | | | |

| | | | |
|---|---|---|---|
| μ=(L(a λ mod n2))-1 mod n, where function is defined as L(u)=u-1/n. | | | |
| The public (encryption) key is. (n,g)<br>The private (decryption) key is (λ, μ) | | | |

**Table 6: Proposed proxy Re-Encryption based RSA algorithm:**

| **Key Generation - keygen(p,q)** | **Encryption: Enc (m,pk)** | **Proxy Re-Encryption(c)** | **Decryption: Dec(c,sk)** |
|---|---|---|---|
| Take two prime number p and q. | Let m be a message to be encrypted where m Є Zn. | Compute Private and Public key.(Rsk,Rpk) | Ciphertext c Є Zn. |
| Compute n=p.q, Φ (n)=(p-1)(q-1) and choose e such that gcd(e, Φ(n)))=1. | Compute ciphertext as: c=me mod n | Re Encrypt Ciphertext generated by Paillier algorithm and send Public key (Rpk) to cloud server. | Compute message m=cd mod n. |
| Determine d such that e.d=1 mod Φ(n) | | | |
| The public key (pk) is (e,n) is generated | | | |
| The Secret key (sk) is (d) is generated | | | |

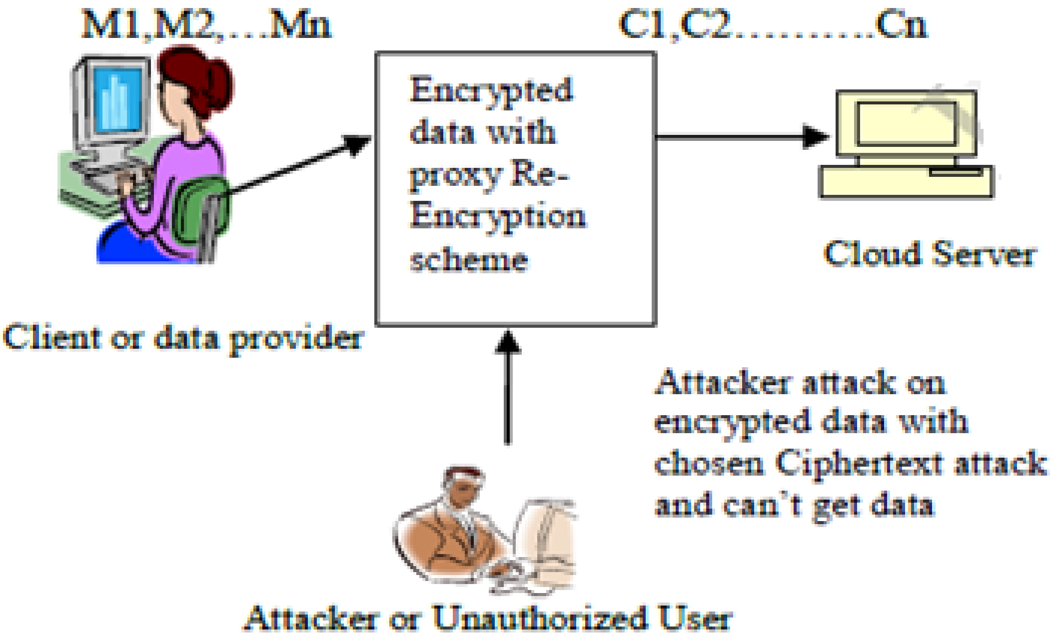


**Fig..5 Proposed System Model**

**Conclusion**

In this paper I have use Homomorphic encryption technique to provide security on cloud. Homomorphic encryption is a new concept of security which enables providing results of calculations on encrypted data without knowing the raw data on which the calculation was carried out, with respect of the data confidentiality. In this paper I have proposed RSA and Paillier algorithm for homomorphic encryption using proxy-Re-encryption algorithm that prevents cipher data from Chosen Cipher text Attack (CCA).So This system is more secure than existing system. In future we can work with efficiency of the system by reducing size of the key and we can also check proxy Re-Encryption method for other Homomorphic Encryption Scheme. Security of cloud computing based on fully Homomorphic encryption is a new concept of security which is enable to provide the results of calculations on encrypted data without knowing the raw entries on which the calculation was carried out respecting the confidentiality of data. Our work is based on the application of fully Homomorphic encryption to the security of Cloud Computing:

**REFERENCES**

[1] John Harauz, Lorti M. Kaufinan, Bruce Potter, "Data Security in the World of Cloud Computing", IEEE Security & Privacy Co- published by the IEEE Computer and Reliability Societies, July/August 2009.

[2] Alecsandru Patrascu, Diana Maimu, Emil Simion, "New Directions in Cloud Computing: A Security Perspective", IEEE 2012.

[3] Vinod Vaikuntanathan, Rajeev Shukla "Computing Blindfolded: New Developments in Fully Homomorphic Encryption", International Journal of Computer Science Research & Technology(IJCSRT), June- 2011.
[4] Jing-Li Han, Ming Yang, Cai-Ling Wang, Shan-Shan Xu, "The Implementation and Application of Fully Homomorphic Encryption Scheme", IEEE 2012.
[5] Nitin Jain, Saibal K. Pal & Dhananjay K. Upadhyay, "Implementation and Analysis of Homomorphic Encryption Schemes", International Journal on Cryptography and Information Security(IJCIS), Vol.2, No.2, June 2012.
[6] Ms. Parin V. Patel, Mr. Hitesh D. Patel, Prof. Pinal J. Patel, "A Secure Cloud using Homomorphic Encryption Scheme", International Journal of Computer Science Research & Technology (IJCSRT) Vol. 1 Issue 1, June-2013.
[7] Guangli Xiang, Benzhi Yu, Ping Zhu, "An Algorithm of Fully Homomorphic Encryption", IEEE 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery.
[8] Bhabendu Kumar Mohanta, Debasis Gountia,"Fully Homomorphic Encryption Equating to Cloud Security: An approach", IOSR Journal of Computer Engineering (IOSR-JCE) Volume 9, Issue 2 Feb 2013.
[9] Sunanda Ravindram, Parsi Kalpana, "Data Storage Security Using Partially Homomorphic Encryption in a Cloud", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 4, April 2013.