ISSN:0975-9646

# Providing Arithmetic Operations on RSA Using Homomorphic Technique

Shiwali[1], Namita kakkar[2]

[1, 2] *Computer Science & engineering department*
*Rayat & Bahra Institute of Engineering & Biotechnology, Mohali*

**Abstract- Cloud computing is a sound area for the field of research. In the current scenario of advanced technology the client and server architecture is been shifting from distributed or cluster to cloud architecture. The main part of this research relies on a robust architecture which deals with Cloud Storage as a Service (SAAS) and comparative security measures of improvement and modifications. Cloud computing is the internet based technology which providing the computing resources in the form of services over the internet. But security is the main issue occurred in the cloud computing because of that growth of cloud computing is less.**
**By using cryptography algorithms, we improve the data security in cloud computing. The cryptography algorithms are used for purpose of secure transmission of private or secret message. There are several number of cryptography algorithms but here we integrate the new technique with RSA algorithm using Homomorphic Encryption and performing the arithmetic process on RSA which improving the security level without compromising the security of existing technique.In this paper our main work to ensure the security of data, so we proposed a method by implementing RSA algorithm using Homomorphic Encryption.**

**Keywords: Cloud computing, cryptography, RSA, Homomorphic Encryption**

## 1. INTRODUCTION

There are many applications supported by cloud computing in many areas. In cloud computing security is the biggest challenge, so we mainly focus on security which is the main concern to provide the security to end user to protect the data from unauthorized access user [1].
In this research, we are securing the cloud by using the concept of Homomorphic technique with cryptography algorithm. That technique is based on the Homomorphic Encryption is the conversion of data into cipher text that can be analyzed and worked with as if it were still in its original form. Homomorphic Encryptions allow complex mathematical operations to be performed on encrypted data without compromising the encryption.
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 is an emerging cryptographic technique to permit computation on encrypted data directly in the cloud without the need to bring the data back to the computational node. However, this technique comes with significant performance and storage related challenges. Due to use of this technique of Homomorphic Encryption, less time will be consumed, more throughput and fast data transfer over network [5].
This paper is organized as follow: section 2 gives the brief introduction about related work; section 3 explains about the methodology followed and section 4 the experimental results of the study. Final conclusion presented in section 5.

## 2. RELATED WORK

Cryptography is most popular technique which is helpful for providing the security, integrity on data transmission through various different network channels. Here are some papers studied related to various techniques as following:
**Kumar et al.** explains about RSA algorithm which is the most popular and asymmetric key cryptographic algorithm. It may used to provide both secrecy and digital signature. It uses the prime no. to generate the public and private key based on mathematical fact and multiplying large numbers together. It uses the block size data in which plaintext and cipher text are integers between 0and n1 for some n values. Size of n is considered 1024bits or 309 decimal digits. In this two different keys are used for encryption and decryption purpose. As sender knows encryption key and receiver knows decryption key.
**Thambiraja et al.** focuses mainly on the different kinds of encryption techniques that are existing with an experimental study of implementations of various available encryption techniques and also focuses on image encryption techniques, information encryption techniques. This study extends to the performance parameters used in encryption processes and analyzing on their security issues. To sum up, all the techniques are useful for real-time encryption.
**P.saveetha et al.** studies on improvement in RSA algorithm and its implementation. The network security means to protect data during their transmission over channel of networks similarly internet security also to protect data during their transmission over a collection of interconnected networks in all over the world. Cryptography is the way of hiding the information during transmission over the channel. There are lots of cryptographic algorithms available to protect our data from intruders. RSA is also one of effective public key algorithm which needs the time and memory. The performance of RSA algorithm will be improved by reducing the modulus and private exponents.
**Brenner et al.** presented a method to perform the execution of encrypted programs operating on encrypted data and described a simple Homomorphic encryption scheme as a

reference model and developed a method to represents the circuits by means of Homomorphically encrypted integer arithmetic's and provided basic performance figures that help to establish a rough runtime estimation for encrypted program. Also presented a method to compute a secret program on an untrusted resource using fully Homomorphic encrypted circuits and sketch an algebraic homomorphism as a cryptographic foundation and define a simple architecture for which we provide a software implementation.

**Tebaa et al.** describes the 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 calculations was carried out respecting the confidentially of data. Basically, work is based upon the application of fully Homomorphic encryption to the security of cloud computing: a) analyze and improve the existing cryptosystems to allow servers to perform various operations requested by the client, b) improve the complexity of the Homomorphic encryption algorithm and study the response time to requests according to the length of the public key.

**Vishwagupta et al.** explains about Advance cryptography algorithm for improving data security. Information security is the process of protecting information. It protects its availability, privacy and integrity. Access to stored information on computer databases has increased greatly. More companies store business and individual information on computer than ever before. Much of the information stored is highly confidential and not for public viewing. In this paper mainly developed a new cryptography algorithm which is based on block cipher concept and used logical operation like XOR and shifting operation that proposed algorithm is very efficient and secured. The proposed algorithm has the batter speed compared with the comparing encryption algorithm. Nevertheless, the proposed algorithm improves encryption security by inserting the symmetric layer. The proposed algorithm will be useful to the applications which require the same procedure of encryption and decryption.

**Yashpal Mote et al.** in their paper says about data encryption algorithm, this algorithm play important role in encrypting and decrypting the data there are present various types of the algorithm that are AES, DES, Triple DES, RSA, and each of these algorithm have their specific role in day to day life . The two main characteristics that identify and differentiate one encryption algorithm from another its ability to secure the protected data against attacks by hacker and its speed and efficiency. This paper provides a performance comparison between five of the most common encryption algorithm. The comparison has been conducted by running several encryption settings to process different sizes of data blocks to evaluate the algorithm encryption and decryption algorithm. This paper concerns about the performance of these algorithms under different conditions, the presented comparison takes into consideration the behavior and performance of the algorithm when different data loads are used. This paper also presents a highly efficient implementation of RSA.

## 3. METHODOLOGY

We have introduced modified RSA algorithm using the concept of Homomorphic Encryption that provide the more security than simple RSA algorithm as well as improved the efficiency of the data. Basically we compared the new invented modified RSA algorithm to existing cryptography algorithm such that simple RSA algorithm. General steps of methodology are given below:

**Phase 1:-** In this phase, we will design the user interface using java swings.

**Phase 2:-** This phase includes the design of admin interface for the admin panel which can manage user list.

**Phase 3:-** This phase includes implement RSA algorithm for encrypt the content of file.

**Phase 4:-** In this phase we apply the Homomorphic Encryption using gates.

**Phase 5:-** Final results will be validated and will be compared with other algorithm.

### 3.1 System Flow Design:

In this design(Fig 1) we mentioned the entire steps we are going to perform in our entire work like uploading the content and then encrypting the data using RSA algorithm with its cipher text formation and public and private key is enabled by RSA and then the user can download the content and after that decryption is performed.

There are two types of cryptography algorithms used which have explained as follow:

### 3.2 RSA ALGORITHM

The algorithm was published in year 1977 and given by three MIT's namely Rivets, Shamir & Adelman [9]. It is an asymmetric key algorithm based on two keys such as public and private key. In this algorithm, a message encryption cryptosystem is used in which two prime numbers are chosen initially and then the product of these two numbers are calculated to generate a public and a private key. These keys are further used during encryption and decryption.

**The RSA algorithm involves three phases:**

- Key generation,
- Encryption, and
- Decryption.

**Key generation**

RSA involves a **public key** and a **private key.** The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated in the following way:

**Step 1:** Firstly two prime number p and q are chosen. For security purposes, the integers

*p* and *q* should be chosen at random.

**Step 2:** Then value of n is computed by using n= p x q. *n* is used as the modulus for both

the public and private keys. Its length, usually expressed in bits, is the key length.

**Step 3:** Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1)$, where $\varphi$ is Euler's totient function.

**Step 4:** Choose an integer $e$ such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e. $e$ and $\varphi(n)$ are co prime. $e$ is released as the public key exponent.

**Step 5:** Determine $d$ as $d{-}1 \equiv e \pmod{\varphi(n)}$, i.e., $d$ is the multiplicative inverse of $e$ (modulo $\varphi(n)$). This is more clearly stated as solve for $d$ given $d\cdot e \equiv 1 \pmod{\varphi(n)}$. $d$ is kept as the private key exponent.

**Step 6:** By construction, $d\cdot e \equiv 1 \pmod{\varphi(n)}$. The public key consists of the modulus $n$ and the
public (or encryption) exponent $e$. The private key consists of the modulus $n$ and the private (or decryption) exponent $d$, which must be kept secret. $p$, $q$, and $\varphi(n)$ must also be kept secret because they can be used to calculate $d$.

**Encryption**

Receiver transmits his public key ($n$, $e$) to Sender and keeps the private key secret. Sender then wishes to send message $M$ to Receiver. He then computes the ciphertext $c$ corresponding to

$c \equiv m^e \pmod{n}$ Sender then transmits $c$ to Receiver.

**Decryption**

Receiver can recover $m$ from $c$ by using his private key exponent $d$ via computing
$m \equiv c^d \pmod{n}$.
Given $m$, he can recover the original message $M$ by reversing the padding scheme.
The RSA algorithm is based on factorization. It becomes hard for the eavesdropper to detect the prime numbers from the factorization. Hence it is hard to break this algorithm.

**3.2.2 Disadvantage of RSA**

Main disadvantage of RSA is to consume large amount of time at the execution of encryption and decryption process resulting low throughput. Other drawbacks of RSA algorithm are speed and it is not suitable for wireless networks.

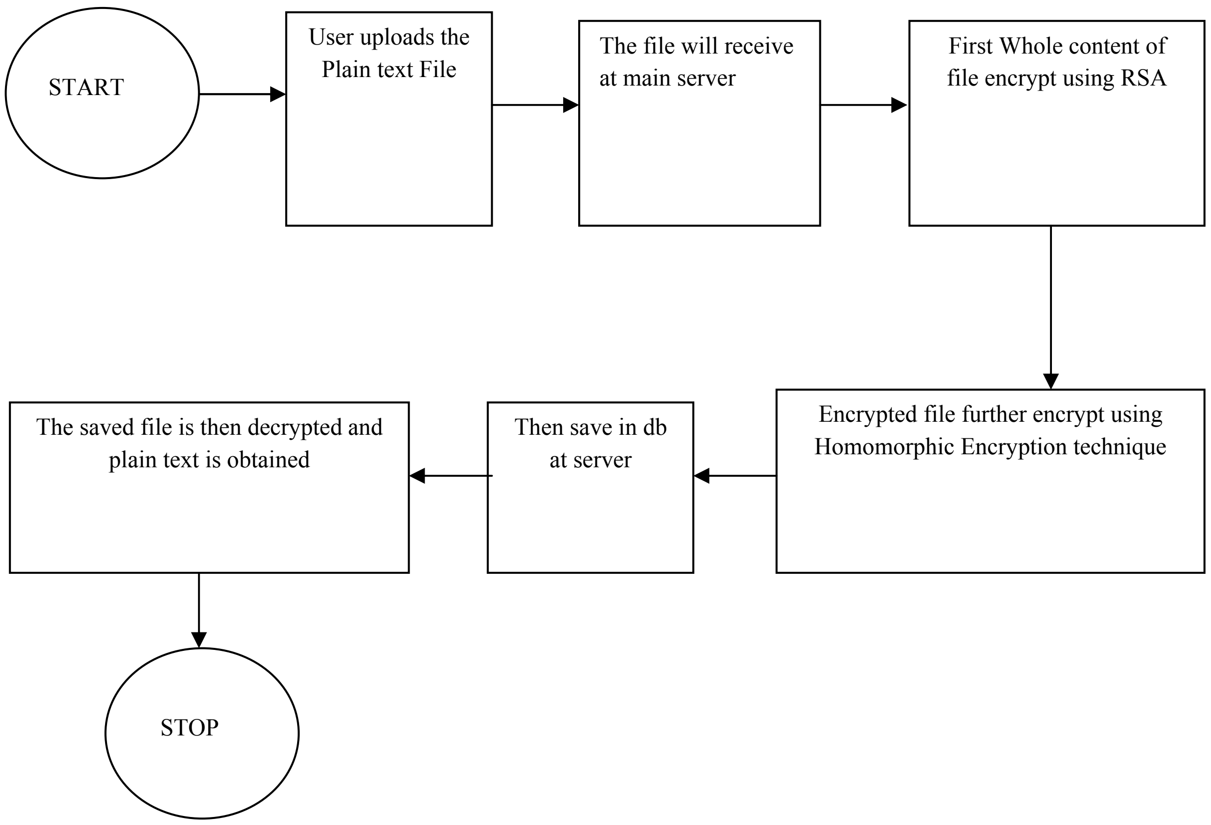
START
User uploads the Plain text File
The file will receive at main server
First Whole content of file encrypt using RSA
Encrypted file further encrypt using Homomorphic Encryption technique
Then save in db at server
The saved file is then decrypted and plain text is obtained
STOP

Figure 1: system flow design

**3.3 MODIFIED RSA ALGORITHM USING HOMOMORPHIC ENCRYPTION**

This modified RSA algorithm is similar with RSA algorithm with some modifications. Modified RSA algorithm is cryptography algorithm which we have extremely large number that has two prime factors same as like in RSA. A modification increases the security of cryptosystem. Modified RSA algorithm is given below:

**Step 1:** Choose two very large random prime integers p and q.

**Step 2:** Compute n and $\varphi(n)$: $n = pq$ and $\varphi(n) = (p-1)(q-1)$.

**Step3:** Choose an integer e, $1<e<\varphi(n)$ such that: $\gcd(e,\varphi(n)) = 1$.
(Where, gcd means greatest common denominator).

**Step4:** Compute d, $1<d<\varphi$ such that: $ed = 1(\bmod \varphi(n))$.
(Where, the public key is (n,e) and private key is (n,d), the values of p,q and $\varphi(n)$ are private, e is the public and encryption exponent, d is the private and decryption exponent).

**Step4:** After that program reads the encrypted RSA file and then reads each character one by one and first converts that each character in ASCII standard value.

**Step5:** Then convert each ASCII value in binary value using java byte class and after that we apply first XOR gate operation and get the length of bits, then divide it by number of index.

**Step6:** So that we can encrypt the block size and copy new array list.
**Step7:** After that applying the shift left operation on all bits and storing it further in a new array list.
**Step8:** Then converts bits in ASCII standard value first and then convert in characters and then write in cipher text file.
**Step 9:** The cipher text file is then received by the server and decryption process starts (reverse encryption) and final plain text is then sent to the user.

### 3.3.1 The modified RSA using Homomorphic Encryption algorithm involves two phases:

- **Encryption Process of RSA using Homomorphic Encryption**

This algorithm is based on the idea of maintaining a balance between security and speed of an algorithm. In this algorithm, two stacks are taken. First stack holds the information about sequence counter and other stack maintains record for the data on which encryption has to be performed.

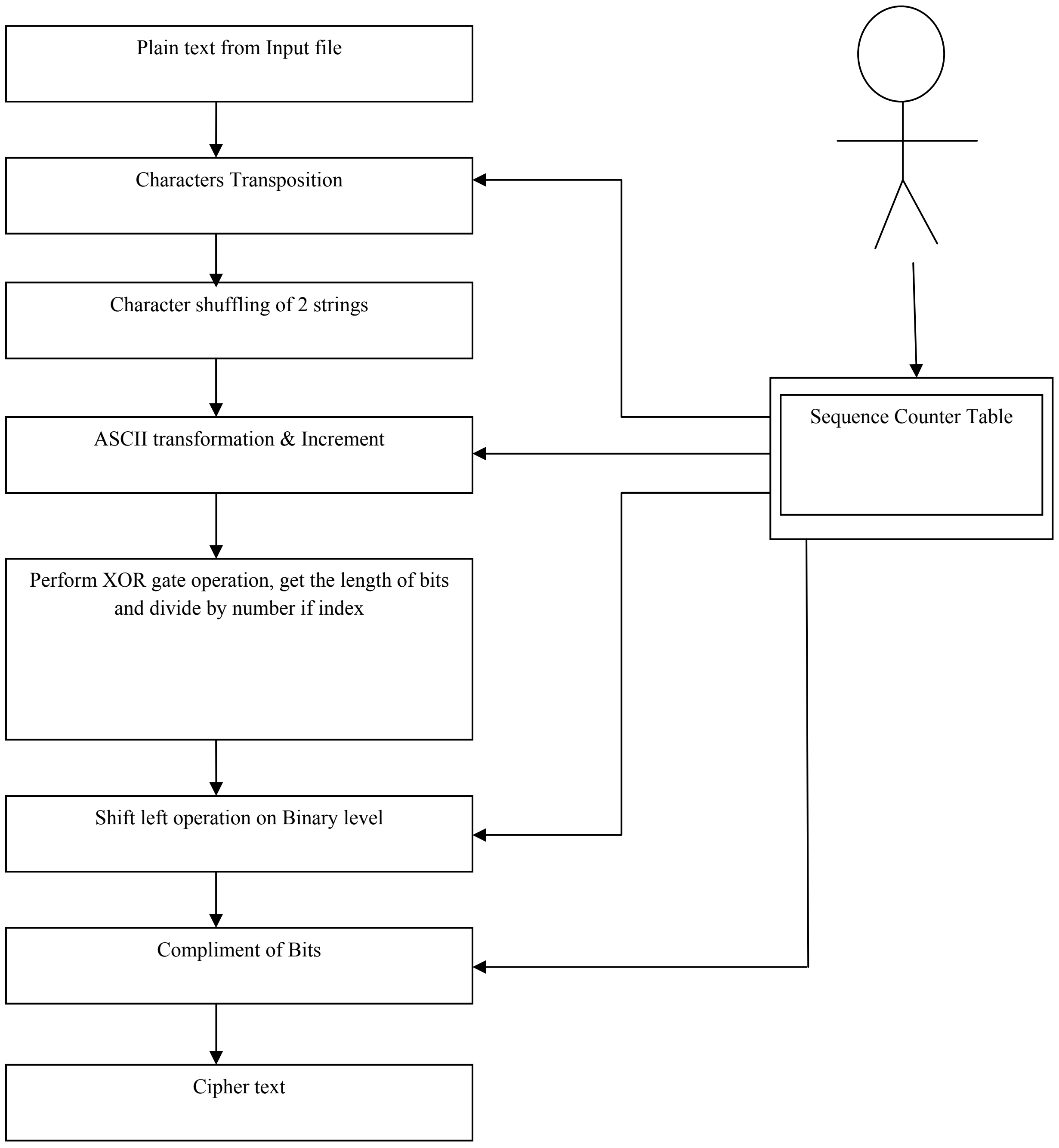
Plain text from Input file
Characters Transposition
Character shuffling of 2 strings
ASCII transformation & Increment
Perform XOR gate operation, get the length of bits and divide by number if index
Shift left operation on Binary level
Compliment of Bits
Cipher text
Sequence Counter Table

Figure 2 Modified RSA Encryption Procedure

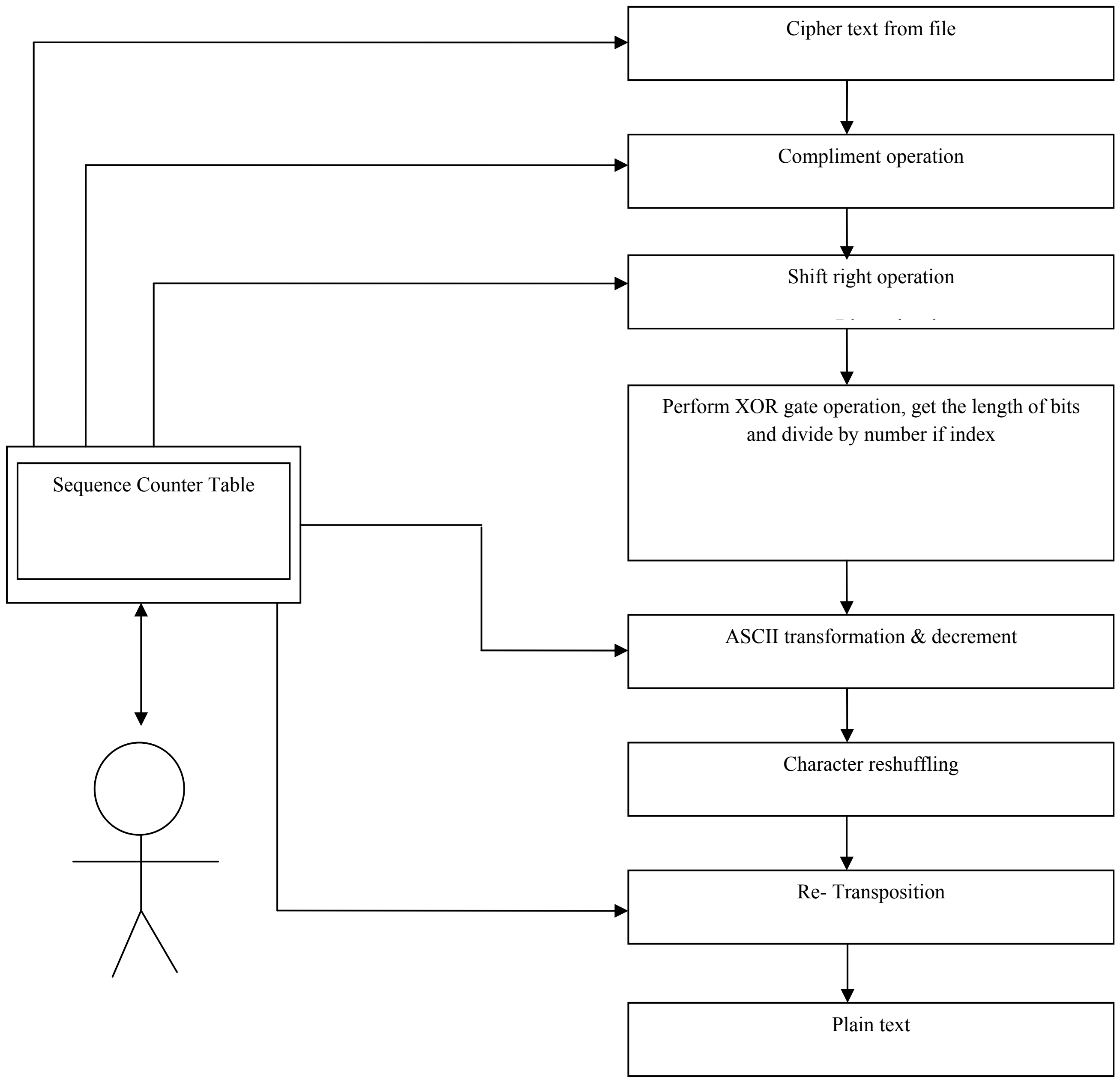

Figure 3 Modified RSA Decryption Procedure

- **Decryption Process of using Homomorphic Encryption**

Decryption process is exactly the reverse of encryption process. The proper Sequence counters are identified from the sequence counter table for decryption procedure.

### 3.3.1 Significance of Modified RSA

Main advantage of the modified algorithm is no key shared over the network while maintaining the data encrypted over the network. Total security of the data no matter which network you use to access the data. No data integrity lost. Data cannot be decrypted by any technique for no information will be made available about the key. This modified RSA algorithm gives more security as compare to RSA algorithm by performing arithmetic operations on encrypted data. By adding the feature of Homomorphic Encryption, increases the speed, security and throughput.

## 4. Experimental Results

The Table 1 represents the five different sizes of files and corresponding encryption execution time taken by simple RSA algorithm and RSA using Homomorphic encryption in seconds. By analyzing the Table 1, we conclude that the encryption time taken by RSA using Homomorphic encryption is very small as compare to RSA. The encryption time taken by simple RSA, RSA using Homomorphic encryption and five different size input files are also shown in figure 4.

Table 1
**Encryption Execution Time**

| **Input File Size (KB)** | **Encryption Execution Time** | |
|---|---|---|
| | **RSA(ms)** | **RSA using HE(ms)** |
| 118 | 1000 | 950 |
| 153 | 7300 | 1358 |
| 196 | 8500 | 1138 |
| 312 | 7800 | 2050 |
| 868 | 8200 | 3950 |

The Table 2 represents the five different sizes of files and corresponding decryption execution time taken by simple RSA algorithm and RSA using Homomorphic encryption in seconds. By analyzing the Table 2, we conclude that the decryption time taken by RSA using Homomorphic encryption is very small as compare to RSA. The decryption time taken by simple RSA and RSA using Homomorphic encryption and five different size input files are also shown in figure 5.

Table 2
**Decryption Execution Time**

| **Input File Size (KB)** | **Decryption Execution Time** | |
|---|---|---|
| | **RSA(s)** | **RSA using HE(ms)** |
| 118 | 5000 | 1823 |
| 153 | 4900 | 2057 |
| 196 | 5900 | 2115 |
| 312 | 5100 | 3679 |
| 868 | 5100 | 3814 |

The Table 3 represents the five different sizes of files and corresponding throughput execution time taken by simple RSA algorithms and RSA using Homomorphic encryption in KB/Seconds. By analyzing the Table 3, we conclude that the throughput time taken by RSA using Homomorphic encryption is large as compare to RSA. The throughput time taken by simple RSA and RSA using Homomorphic encryption and five different size input files are also shown in figure 6.

Table 3
**Throughput Execution Time**

| **Input File Size (KB)** | **Throughput Execution Time(KB/Seconds)** | |
|---|---|---|
| | **RSA(s)** | **RSA using HE(ms)** |
| 118 | 94.400 | 99.368 |
| 153 | 167.671 | 901.325 |
| 196 | 184.71 | 1377.856 |
| 312 | 320.00 | 1217.561 |
| 868 | 846.829 | 1761.096 |

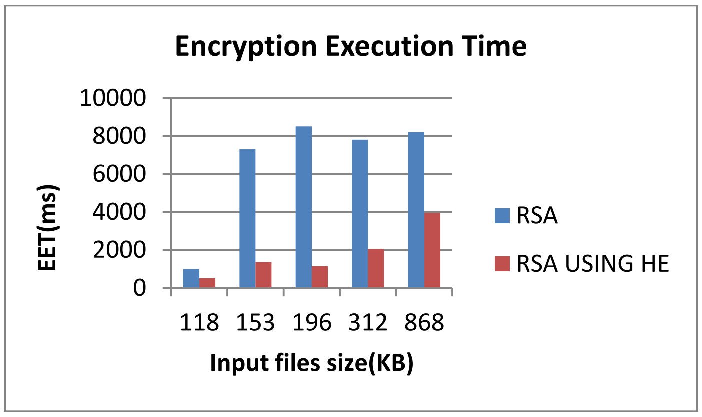

Figure 4 EET among RSA and modified RSA using Homomorphic Encryption

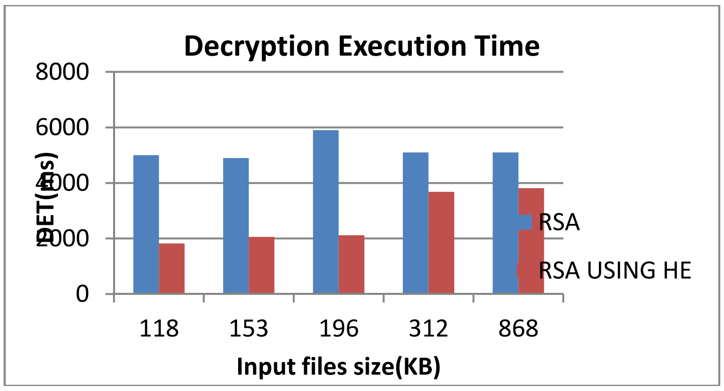

Figure 5 DET among RSA and modified RSA using Homomorphic Encryption

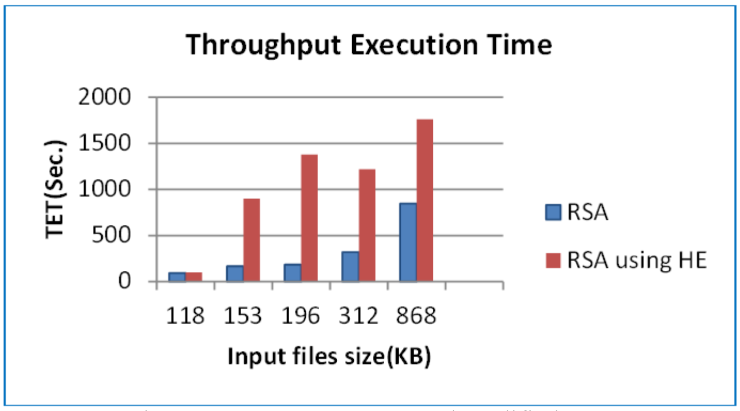

Figure 6 TET among RSA and modified RSA using Homomorphic Encryption

## 5. Conculsion

In this research paper, good results of proposed system that is RSA using Homomorphic Encryption have been achieved which increases efficiency and prevents overhead on server. Hence it will help us in improving the security in cloud computing. We have seen from results that the encryption and decryption time of data with RSA using Homomorphic Encryption was taken less as compared to other system that is RSA. So, with decrease in time consumption there will be increase in throughput.

Moreover, the security issue has been resolved. With the help of Homomorphic technique user can easily upload the data over server and no hacker can corrupt the encrypted data. The sequence counter added in the Homomorphic algorithm is only known to the user itself. So, this will be very helpful in protecting the data from unauthorized users. During this algorithm, it was the data which was uploaded in form of .txt extension which must not lose its originality and its contents were retained intact.

## REFERENCES:

[1] Bhatt, P.; Maheta, A. (2010), *" Security In Cloud Computing Using File Encryption,* IJERT, ISSN 2278-0181,Vol. 1 issue 9, November 2012.

[2] Kota, C.M. and Aissi, C. (2002), *"Implementation of the RSA algorithm and its cryptanalysis",* in proceedings of the 2002 ASEE Gulf-Southwest Annual Conference, March 20 – 22, 2002

[3] Brenner, M.; Wiebelitz, J.; Vonvoigt, G and Smith, M. (2011), *"Secret Program Execution in the Cloud Applying Homomorphic Encryption",* IEEE 5th International Conference on Digital Ecosystems and technologies (IEEE DEST), June 2011, pp 114-119.

[4] Kumar, A.; Jakhar, S and Makkar, S. (2012), *"Comparative Analysis between DES and RSA Algorithm's",* IJARCSSE, Vol.2, July 2012, pp 386-390.

[5] Tebaa, M and Hajji, S.E. (2012), *"Homomorphic Encryption Method Applied to Cloud Computing",* IEEE, June 2012, pp 86-89.

[6] Saveetha, P and Arumugam, S. (2015), *"Study on Improvement in RSA Algorithm and its Implementation",* IJCCT, Vol. 3, Issue 6, 7 august 2012, pp 61-65.

[7] Thambiraja, E.; Ramesh, G and Umarani, R. (2012), *"A Survey on Various Most Common Encryption Techniques",* IJARCSSE, Vol. 2, July 2012, pp 226-233.

[8] Gupta, V.; Singh, G and Gupta, R. (2012), *"Advance Cryptography Algorithm for Improving Data Security",* International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 1, January 2012.

[9] Mote, Y and Gaikwad, S. (2012), *"Superior Security Data Encryption Algorithm",* Internatinal Journal of Engineering Science, july 2012, Vol. 6.