A Thesis Report on

Homomorphic Encryption on Cloud Data

by

KANITHI SAUJANYA
(Registration No. 1571077011)

Submitted to the

Department of Computer Science & Engineering

in partial fulfillment of the requirements
for the degree of

Dual Degree Engineering in Computer Science and Engineering

at the

Institute of Technical Education and Research
S'O'A University, Bhubaneswar, Odisha, India

Under the supervision of

Dr. ROURAB PAUL
Department of CSE

July 2020

©Department of CSE, ITER 2020. All rights reserved.

CERTIFICATE

This is to certify that the project work entitled **"Homomorphic Encryption on Cloud Data"** is a bonafied work done by **KANITHI SAUJANYA** bearing registration number **1571077011** of 10th semester, **Department of Computer Science & Engineering**, towards the completion and award of DDE(CSE) from Information Technology of **Institute of Technical Education and Research**, under **Siksha 'O' Anusandhan University, Bhubaneswar, Odisha**.

Dr. Rourab Paul	Dr. Kaberi Das	Prof.(Dr). Debahuti Mishra
(Supervisor)	(Project Coordinator)	(Head of Dept.)

DECLARATION

I Kanithi Saujanya do hereby declare that the work embodied in this thesis titled "Homomorphic Encryption on Cloud Data" submitted to Siksha 'O' Anusandhan Deemed to be University in partial fulfilment of the requirement for the award of M.Tech in Computer Science and Engineering is a record of original and independent research work carried out by me under the supervision and guidance of Dr. Rourab Paul, Department of Computer Science & Engineering, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar. This is original and has not been submitted for the award of any other degree else where.

Kanithi Saujanya

Kanithi Saujanya

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratefulness and appreciation to my supervisor Dr. Rourab Paul, Department of Computer Science and Engineering, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar for his inspiration and guidance at all stagesof my work. His timely observations, valuable suggestions and comments helped me to establish the overall direction of my work. I wish to thank all my friends at SOA Deemed to be University, Bhubaneswar for their continuous support and inspiration. Lastly, I would like to thank God for being with me every single moment and blessing me with good health.

Kanithi Saujanya

Kanithi Saujanya

Contents

1 Introduction **3**

2 Literature Review **8**

3 Challenges/Issues Identified **12**

4 Problem Statement **12**

5 Schematic Layout **13**

6 Data sets collected **14**

7 Methodologies Adopted **15**

8 Experimentation and Result Analysis **18**

9 Action Plan **26**

10 Summary **27**

11 References **28**

List of Figures

Figure 1: Storing of data 13
Figure 2: Computation on data 13
Figure 3: rsa homomorphic encryption algorithm 15
Figure 4: paillier's encryption algorithm 17

Abstract

With the growth of technology everything is controlled by our fingertips. Each and every important and confidential data is shared over cloud. Cloud computing is where our data is kept, computed and handled and they are provided by different service providers like Amazon, Google, etc known as CSP(Cloud Service Provider). Security is a bigger area of concern here as we cannot ignore any risk regarding our confidential sensitive data. So we proposed a model of Hybrid Homomorphic Encryption which focuses on building a model in which the operations are carried on the encrypted data and getting the desired result without any need to decrypt the data. The model uses a combination of two encryption algorithms that can perform some basic mathematical operations, they are:

- RSA algorithm is used to achieve partial multiplicative homomorphism.
  - By simple power function
  - By successive modulus
  - By Modulus Exponentiation
- Paillier's algorithm is used to achieve partial additive homomorphism.

Keywords

Cloud Computing, Cryptography, Homomorphic Encryption, CSP, RSA algorithm, Paillier's algorithm.

1 Introduction

Technology and science are evolving largely day by day and they are interdependent. It has nowadays became a part of our daily life making it easier and reliable. With its advancement has touched every field from running an application on mobile to holding the country's security. Technology enhancement has its own pros and cons. Cloud computing came into fashion and every important sensitive data from personal to professional is shared on it. Clouds is a metaphor used for Cloud Computing. Cloud keeps us connected throughout the world. We can share or access any information over the internet without using or installing any extra software or hardware on the system be it on desktop or a wireless mobile. The cloud service excludes the use where we need to store the data on local system memory which is limited and costs highly for an extra addition. Here cloud service providers(CSP) in today's trending which operate huge cloud service and numerous clients are AMAZON, GOOGLE, FACEBOOK that promise highly reliable service and confidentiality of data is their first priority.

Cloud enhances the distribution of resources and high reliability maintaining the data integrity and security. There is another concept called "Intercloud" where switching between the different service providers in the same Intercloud is possible, whereas switching from one cloud service provider to another can be highly difficult and your data security can be at risk. As cloud is a internet-based application and share highly confidential and sensible data, leaking of any data can be of high risk as these data is of high value in black market and attackers everywhere try to steal your data and sell for higher price which can greatly harm you.

So data security is always the first priority for the cloud service providers so they store data in an encrypted form and access authentication. If someone wants to make changes to their existing data which is in an encrypted form, then the simple way is to decrypt that particular data first and make changes as per the user's request and again encrypt it.

But here the biggest question is:

- **"Is our data secure?"**

And by security we mean to:

- **data loss or any misuse of our data.**

We have been aware of terms like cyber attacks, cyber crimes, cyber warfare, cyber bullying, etc and we also know that the data shared over cloud is actually in form of encryption but still these things take place. This is because there have been many security issues such as data availability, legal issues and privacy, third-party control, etc. They are mainly focused on maintaining data integrity, data security, confidentiality, authentication, etc.

Our solution can be:

- **What if the data shared over cloud doesn't need to be decrypted just to perform some operations on it?**

What if an attacker stole the data in the process where data is decrypted. We have been aware of terms like cyber attacks, cyber crimes, cyber warfare, cyber bullying, etc and we also know that the data shared over cloud is actually in form of encryption but still these things take place. This is because there have been many security issues such as data availability, legal issues and privacy, third-party control, etc. They are mainly focused on maintaining data integrity, data security, confidentiality, authentication, etc.

So this still is a question for data security. What if we perform necessary changes or can say required operations on the encrypted data without decrypting it? Sounds cool right. This will surely decrease the rate of risk. There is an encryption algorithm which makes this possible. Before going into the details we have first know what is Encryption and its types.

Encryption or Cryptography is a way where readable data known as plaintext is converted into some random data known as ciphertext which is impossible to understand but it follows a logic behind it, only the sender and the authenticated receiver will be able to understand it as it converts a plaintext to ciphertext with the help of a key. The one who has it can only decrypt the data else its hard for the third party to even understand. In simple words imagine the courier system, where you store some things inside a package and seal it and courier it to your who only can open it to see the contents inside it. Intermediate people won't be able to see it because they do not have the

authentication to open it. Encryption characteristics include confidentiality, integrity, non-repudiation and authentication. Cryptography is where security meets mathematics. We have different kinds of encryption/decryption algorithms in which a set of instructions are performed on the data to enhance its security and helps in maintaining its integrity.

Types of Cryptography systems:

- **Symmetric:** It is also known as secret-key cryptography and here for encryption and decryption only single key is used. That means the sender uses the key for converting plaintext into ciphertext and the receiver uses the same key to convert the ciphertext to plaintext. Here the key is known to both the sender and the receiver and the key distribution is its biggest difficulty.
- **Asymmetric:** It is also known as public-key cryptography in which two keys are used for encryption and decryption respectively and they are related mathematically. Here the security is maintained between two users even over a non-secure connection as there is no need to share the secret key.
- **Hash function:** It is also known as message digests or one-way encryption in which there is no use of any kind of key instead it uses a fixed length hash value based upon the plaintext. Basically used for password encryption and maintaining the file's integrity.

So our proposed model follows the Homomorphic Encryption that comes under Asymmetric Cryptography system where the keys come in pair. The public key can be shared publically but the private key is only there with the owner. Asymmetric Cryptography is always much more secure than the Symmetric Cryptography as it uses two keys and it helps maintain data confidentiality and integrity.

Why Homomorphic Encryption?

- Confidentiality Problems.
- Ability to compare over ciphertext instead of plaintext.

- One cloud use information without knowing the content of that information.
- Privacy guranteed.
- Data integrity and reliabilty is maintained.

Homomorphic encryption schemes were first proposed by Rivest, Adleman, Dertouzos and later Craig Gentry proposed fully homomorphic encryption. Basically HE can be viewed as an extension of symmetric or asymmetric cryptosystem that directly perform arithematical operations on the encrypted data or say ciphertext. The encrypted result obtained by performing operations on encrypted data which upon decryption is same as the result we get by performing the same operations on decrypted data. Homomorphic encryption includes the following steps such as Key generation, Encryption/Decryption, Database Storage, Request/Response, Evaluation. Homomorphic Encryption works best with numeric values where there we can perform arithematic calculations like add or multiply. Homomorphic Encryption in simple words can be explained as a glass box with objects which is locked and only the owner has the key to access the data but the employees can operate some functions or modify the objects inside the with the help of a glove attached to that box. It means at the end the desired result or model is obtained without the employees knowing its inner details. Some of its real world applications are securing cloud data, election transparency, etc.

Some types of Homomorphic Encryptions:

- Partial homomorphic encryption Here only one mathematical operation is used any no.of times.
- Somewhat homomorphic encryption Here addition can be used any no. of times but multiplication can be used only limited no. of times.
- Fully homomorphic encryption Here arbitrary no.of operations are performed on the ciphertext.

Here we propose a model with the combination of two encryption algorithms that are RSA(Rivest Shamir Adleman) algorithm and Paillier's algorithm. Fully Homomorphic encryption(FHE) though performs unlimited operations over large set of data or say ciphertext, still its not practical. Our proposed

cannot compete with FHE but yes it can satisfy some operations carried by FHE and here we are using small size data.

2 Literature Review

1. Abdelali El Bouchti et al proposed "Encryption as a service for data healthcare cloud security" in which they highlighted about the issues regarding security of the healthcare information shared over cloud that are at a high risk because it contains the medical records which contains confidential data that can even let access to hack credit cards. Here two encryption algorithms are used which are Homomorphic Encryption scheme and RSA Cryptosystem to avail protection to those data. The author suggested a hybrid model based on CaaS. They claim their proposed model is provides more security than the existing model such as EHRs(Electronic Health Record).

2. Priyanka Ora et al proposed "Data Security and integrity in cloud computing based on RSA partial homomorphic and MD5 crptography" says with the advancement of the technology, cloud computing came into fashion which made security a bigger area of concern. As the data is shared over public cloud, they are more liable to risks so here the author proposed a solution to maintain data security and integrity. The proposed model is a combination of MD5 hashing algorithm and RSA PHE. RSA PHE first encrypts the data and it is uploaded to cloud and then the hash value of the uploaded data is calculated to specify the access to the following required data. Therefore this approach includes steps like Encryption/Decryption, Uploading data on cloud server, Hash value generation, Access permission and Verification.

3. Xidan Song et al proposed "Homomorphic cloud computing scheme based on hybrid homomorphic encryption" which defines a software application called "Encryption Decryption Machine(EDM)" runs on private cloud used for encryption and decryption. It is developed because in present situation the fully homomorphic encryption method has many disadvantages and is not practical and Partial Homomorphic Encryption provide limited number of operations. So they proposed a model of SHE over cloud which satisfies addition and multiplication by using Paillier's algorithm for addition or substraction known as additively homomorphic and RSA algorithm for multiplication or division known as multiplicatively homomorphic. Overally the paper contains all the tests and results to prove that their scheme is efficient and practical.

4. Mostapha Derfouf et al proposed "Cloud Secured Protocol based on Partial Homomorphic Encryptions" in which their proposed scheme is compared with the FHE and discusses its advantages over FHE. The proposed scheme based on the combination of the partial additive cryptography and partial multiplicative cryptography as in to choose the best implementation concerning security and performance. First the protocol starts with an interface where the choice of operation (addition or multiplication) is requested and depending on it the protocol chooses which partial encryption to be carried out. The protocol sends back the encrypted data to the cloud and this is carried on till no more operations exist. Basically each field is encrypted twice in the database, one for addition and another for multiplication to enhance the security in the cloud database.

5. Baocang Wang et al in their paper "Cyptanalysis of a Symmetric Fully Homomorphic Encryption Scheme" recovers some of the disadvantages of the Li et al 's proposed model of homomorphic encryption. They pointed out different remarks such as the model actually follows somewhat homomorphic encryption, it contradicts its own claimed equations. They have basically done analysis on Li et al 's proposed model and put forward an attack algorithm on it.

6. Yi Dou et al in their paper "A Distributed Trust Evaluation Protocol with Privacy Protection for Intercloud" states about the problems faced by cloud user to shift from one cloud service provider to another cloud service provider. This is the case where "Intercloud" concept came in. Intercloud composes of the root ,exchanges and clouds. It includes atleast 2 clouds. It basically is helpful when the cloud data is managed and there will be good resource utilization and used mainly during disaster and different region delivery purpose. But here there is a trust bond between different cloud service providers. Trust means a lot to them as it states the reliability, quality and efficiency of a cloud service provider. And this trust evaluation is conducted on the existing users of that cloud to give their feedback regarding their cloud service provider which will be helpful for other users in the future to decide on. But through this the users will only give positive comments as they wont be wanting to risk their relationship with their cloud service provider as they handle very sensitive data and future transactions. So to overcome this here the authors proposed a model which evaluates the trust based on the user's feedback by being flexible and effective keeping in mind the privacy protection

for intercloud. So here homomorphic encryption is used for protecting the feedback privacy allowing verified sharing and customised processing. Apart from the protocols which only support static config, this proposed protocol supports dynamic config.

7. Zainab Hikmat Mahmood et al in their paper "New Fully Homomorphic Encryption Scheme based on Multistage Partial Homomorphic Encryption applied in Cloud Computing" propose a hybrid structure of Homomorphic Encryption as it is impossible to build a fully homomorphic encryption practically as it also has drawbacks. Here they use two famous algorithms namely RSA and Goldwasser-Micali for achieving partially homomorphic multiplication and patially homomorphic addition respectively. They also explained the basic knowledge behind this proposed model. Their compare their proposed model with EHES and show very promising results as here the main motive is to enhance the performance and efficiency i.e. low computation cost and high reliability. Through this the data will be more secured and its integrity is maintained.

8. Florencia Irena et al in their paper "CryptoBlaze: A Partially Homomorphic Processor with Multiple Instructions and Non-Dterministic Encryption Support" entitles the possibilities of homomorphic encryption not only limited to software but also hardware as here they claim they have created an additional hardware for homomorphic encryption that is called as "Cryptoblaze" which allows an additional eight specialized instructions. Here they claim their proposed model hardware run and tested on Field Programmable Gate Arrays (FPGA) and got benchmark as 3. They say they are inspired by the existing models HEROIC(Homomorphically EncRypted One Instruction Computer) and FURISC(Fully Homomorphic Ultimate RISC) structures. HEROIC supports non-deterministic variant of Paillier Cryptosystem whereas FURISC supports or works on deterministic variant of Paillier Cryptosystem. Both work on single instruction machine(SUBLEQ) and here they also mentioned about the third processor which works on OpenRISC ,supports multiple instruction and uses deterministic encryption. Cyrptoblaze uses non-deterministic encryption. So after such comparisions they claim Cryptoblaze is faster than HEROIC and FURISC and communication here is done on-chip and hence data security is also maintained.

9. Lim Tsu Chean et al in their paper "Authentication Scheme using Unique Identification method with Homomorphic Encryption in Mobile Cloud Computing" discusses about MCC(Mobile Cloud Computing) which is an extension of Cloud Computing and they also discuss how RSA encryption can also be attacked using Shor's algorithm by Quantum Computer. So here they explained different frameworks and their uses and proposed a framework that works as a "middleman" based on Identity Management(IDM) that replaces the use of password which is more susceptible to risk with the client details like name, email, phone number, etc and the identity parameters keeps on changing everytime there is an attempt authentication . The work of IDM is also explained clearly. There is a use of Homomorphic Encryption mostly on every level to make it more secure. Here they also explain why a framework should be proposed rather than a scheme with enhanced encryption. Their utmost motivation is the increased use of mobiles now-a-days than the computer users which questions the safety of the MCC as it is believed to be susceptile to the attackers as long as they are connected to the internet. Still the authors say their proposed model should be made changes frequently to maintain security as attackers can always find a way to attack.

3 Challenges/Issues Identified

The code proposed has to face some hurdles like:

- **Efficiency**
  Its always important that the higher the efficiency of the code the better performance. Overally Partial Homomorphic Encryption algorithms are efficient in performing only one dedicated operation but when it comes to different operations at a go, then Fully Homomorphic Encryption is efficient but it still not practical. And Somewhat Homomorphic Encryption is also limited in number of operations.

- **Robustness**
  It depends on the encryption key size like for example the key size should be greater than 1024 bits in case of RSA HE. The larger the key, the slower the system.

- **Delay**
  Maintaining the robustness can slow down the system's performance as larger key size can effect processing, encryption and decryption time.

4 Problem Statement

The overall motive is to develop an efficient upgraded code through hybrid homomorphic encryption.

5 Schematic Layout

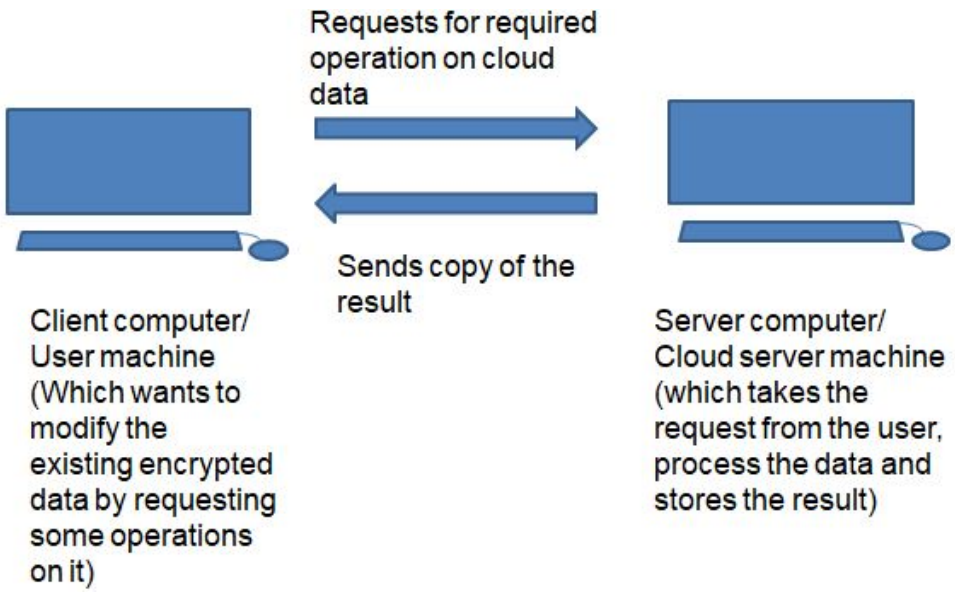

Figure 1: Storing of data

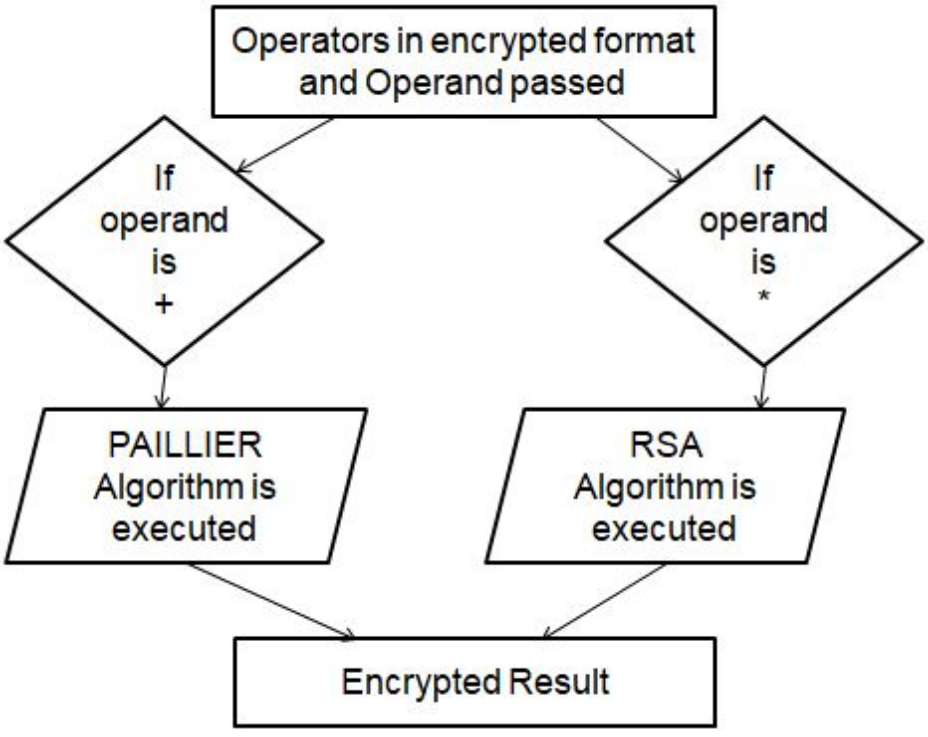

Figure 2: Computation on data

6 Data sets collected

Here we used some manual data else any data that is encrypted and on which the requested operations can be performed.

7 Methodologies Adopted

RSA Algorithm
By Ron Rivest, Adi Shamir and Len Adleman in 1977. Used to achieve multiplicative homomorphism.

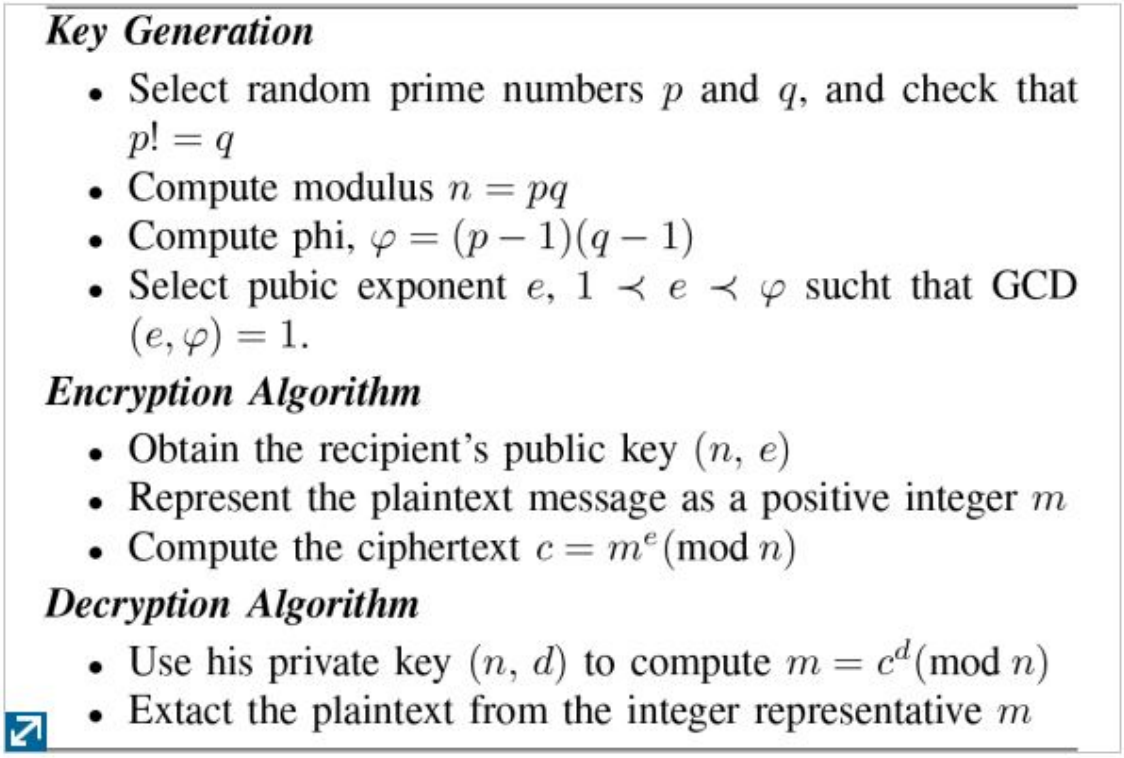

Figure 3: rsa homomorphic encryption algorithm

Binary Method of Modulus Exponentiation:

Input: M, e, n.
Output: C := Mê mod n.
Step 1. if e(k-1) = 1 then C := M else C := 1
Step 2. for i = k - 2 down to 0
2a. C := C * C (mod n)
2b. if e(i) = 1 then C := C * M (mod n)
Step 3. return C

Successive Moduling Method:

Initialize result=1
loop i:0 to e
result = (result*m) mod n
loop end
return result

Here the main thing in this whole RSA algorithm is to compute the Mê mod n in a lesser time. We first used simple power function to implement the RSA Algorithm but it doesn't work for large keys. We then implemented the code using successive modulus function which showed better results but here the loop iterates for e times, so for a larger value of 'e' the code will take longer time to run and probably fails to show the desired output. So for a better approach we then shifted to 'Modulus Exponentation' which iterates the loop on the bit-length of binary value of 'e' which will help execute the program in a lesser time and thus provides the desired output even for a larger key and here we used 8-bit length key and 18-19 bit-length inputs. There are some cases where it still fails to produce the desired result and we look further to work on that.

PAILLIERS' Algorithm

By Pascal Paillier in 1999. Used to achieve additive homomorphism.

Key generation [edit]

1. Choose two large prime numbers p and q randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$. This property is assured if both primes are of equal length.[1]
2. Compute $n = pq$ and $\lambda = \operatorname{lcm}(p-1, q-1)$.
3. Select random integer g where $g \in \mathbb{Z}^*_{n^2}$
4. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$,

where function L is defined as $L(x) = \dfrac{x-1}{n}$.

Note that the notation $\dfrac{a}{b}$ does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of a divided by b, i.e., the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.

- The public (encryption) key is (n, g).
- The private (decryption) key is (λ, μ).

If using p,q of equivalent length, a simpler variant of the above key generation steps would be to set $g = n + 1, \lambda = \varphi(n)$, and $\mu = \varphi(n)^{-1} \bmod n$, where $\varphi(n) = (p-1)(q-1)$.[1]

Encryption [edit]

1. Let m be a message to be encrypted where $0 \leq m < n$
2. Select random r where $0 \leq r < n$
3. Compute ciphertext as: $c = g^m \cdot r^n \bmod n^2$

Decryption [edit]

1. Let c be the ciphertext to decrypt, where $c \in \mathbb{Z}^*_{n^2}$
2. Compute the plaintext message as: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

Figure 4: paillier's encryption algorithm

8 Experimentation and Result Analysis

RSA using simple power function:

```
import java.util.*;
import java.lang.*;
public class RSA_imp
public static void main(String[] args)

int n,p,q,a=0,b=0,phi,i,k=0;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the value for p:");
p=sc.nextInt();
System.out.print("Enter the value for q:");
q=sc.nextInt();
n=p*q;
phi=(p-1)*(q-1);
for(i=2;i < phi;i++)

if(gcd(phi,i)==1)

b=i;
break;


System.out.println(b);
/*do

a++;
//a=(k*phi)+1;
if((a*b)%phi == 1)

//a=a/b;
break;

while(a < 10);*/
//a=(1/b)%phi;
System.out.print("\nEnter number1 to be encrypted :");
```

```
int x1=sc.nextInt();
int y1=(int)Math.pow(x1,b)%n;
//x1=(int)Math.pow(y1,a)%n;
System.out.println("Encrypted number1 :"+y1);
System.out.print("\nEnter number2 to be encrypted :");
int x2=sc.nextInt();
int y2=(int)Math.pow(x2,b)%n;
//x2=(int)Math.pow(y2,a)%n;
System.out.println("Encrypted number2 :"+y2);
//if(x1¡0)y1=-1*y1;
//if(x2¡0)y2=-1*y2;
int y3=(y1*y2)%n;
System.out.println("\nTheir encrypted multiplication :"+y3);
int x3=(int)Math.pow((x1*x2),b)%n;
System.out.println("Their actual multiplication in ecryption:"+x3);

static int gcd(int a,int b)

if(b==0)
return a; else
return gcd(b,a%b);
```

RSA using successive modulus:

```
import java.util.*;
import java.lang.*;
public class RSA_Ex
public static void main(String[] args)

int n,p,q,a=0,b=0,phi,i,k=0;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the value for p:");
p=sc.nextInt();
System.out.print("Enter the value for q:");
q=sc.nextInt();
```

```
n=p*q;
phi=(p-1)*(q-1);
for(i=2;i < phi;i++)

if(gcd(phi,i)==1)

b=i;
break;


System.out.println(b);

System.out.print("\nEnter number1 to be encrypted :");
int x1=sc.nextInt();
int y1=power(x1,b,n);

System.out.println("Encrypted number1 :"+y1);
System.out.print("\nEnter number2 to be encrypted :");
int x2=sc.nextInt();
int y2=power(x2,b,n);

System.out.println("Encrypted number2 :"+y2);

int y3=(y1*y2)%n;
System.out.println("\nTheir encrypted multiplication :"+y3);
int x3=power(x1*x2,b,n);
System.out.println("Their actual multiplication in ecryption:"+x3);

static int gcd(int a,int b)

if(b==0)
return a;
else
return gcd(b,a%b);

static int power(int m, int e, int n)

int a=1;
```

```
for(int i=0;i < e;i++)

a=(a*m)%n;

return a;
```

RSA using modulus exponentiation:

```
import java.util.*;
import java.lang.*;
public class RSA_ME {
public static void main(String[] args)
{
int n,p,q,a=0,b=0,phi,i,k=0;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the value for p:");
p=sc.nextInt();
System.out.print("Enter the value for q:");
q=sc.nextInt();
n=p*q;
phi=(p-1)*(q-1);
for(i=2;i < phi;i++)
{
if(gcd(phi,i)==1)
{
b=i;
break;
}
}
System.out.println(b);
```

```
System.out.print("\n Enter number1 to be encrypted :");
int x1=sc.nextInt();
```

```
int y1=power(x1,b,n);


    System.out.println("Encrypted number1 :"+y1);
System.out.print("\nEnter number2 to be encrypted :");
int x2=sc.nextInt();
int y2=power(x2,b,n);


    System.out.println("Encrypted number2 :"+y2);


    int y3=(y1*y2)%n;
System.out.println("\nTheir encrypted multiplication :"+y3);
int x3=power(x1*x2,b,n);
System.out.println("\Their actual multiplication in ecryption:"+x3);
}
static int gcd(int a,int b)
{
if(b==0)
return a;
else
return gcd(b,a%b);
}
static int power(int x, int y, int p)
{
// Initialize result
int res = 1;


    // Update x if it is more
// than or equal to p
x = x % p;


    if (x == 0) return 0; // In case x is divisible by p;
```

```
    while (y > 0)
{
// If y is odd, multiply x
// with result
if((y & 1)==1)
res = (res * x) % p;


    // y must be even now
// y = y / 2
y = y >> 1;
x = (x * x) % p;
}
return res;
}
}
```

Analysis

Here we roughly took two different values for each p and q and checked for different numeric values n1 and n2.

for p=11,q=17
n1=7,n2=2

By simple power function	By successive modulus	By modulus exponentiation
successful	successful	successful
9secs	9secs	9secs

n1=999,n2=999

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	successful	successful
--	8secs	9secs

for p=61,q=53
n1=12,n2=7

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	successful	successful
--	9secs	12secs

n1=844,n2=84

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	successful	successful
--	5secs	4secs

n1=7236,n2=4598

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	unsuccessful	successful
--	--	14secs

n1=99999,n2=9999

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	unsuccessful	successful
--	--	11secs

n1=88887,n2=88887

By simple power function	By successive modulus	By modulus exponentiation
unsuccessful	unsuccessful	unsuccessful
--	--	--

Here we used Modulus Exponentiation in implementing the RSA algorithm to achieve multiplicative homomorphism because we compared it with simple power function and successive modulus function but we got modulus exponentiation more efficient that has lesser time complexity and works on large integer as here the number of iterations depend upon the number of bits of a number, but still to improve its efficiency we have to work on that.

9 Action Plan

- Topic chosed.
- Information gathered from both online and offline sources.
- Got some reference from the research papers.
- Drafted the required data.
- Proposed a model.
- Implementation of our proposed model.
- Testing and analysis.
- Conclusion

10 Summary

The main focus of this paper is performing different mathematical operations on the set of encrypted data without any need for decryption .The user sends requests for specific operation to be performed on the encrypted data and the system processes the request and sends back the encrypted result which matches the actual or say desired result. This will avoid any data loss and if ever exposed to hackers still they cannot able to get it as its in encrypted form. And through this data security and integrity is maintained.

11 References

1. Priyanka Ora ,P. R. Pal, "Data Security and integrity in cloud computing based on RSA partial homomorphic and MD5 crptography", International Conference on Computer, Communication and Control (IC4) ,2015

2. Abdelali El Bouchti , Samir Bahsani, Tarik Nahhal, "Encryption as a service for data healthcare cloud security" , Fifth International Conference on Future Generation Communication Technologies (FGCT), 2016

3. Xidan Song , Yuling Wang, "Homomorphic cloud computing scheme based on hybrid homomorphic encryption", 3rd IEEE International Conference on Computer and Communications (ICCC), 2017

4. Kamile Nur Sevis , Ensar Sekar, "Cyber warfare:terms,issues,laws and controversies", International Conference On Cyber Security And Protection Of Digital Services (Cyber Security), 2016

5. Wikipedia.com

6. Mostapha Derfouf, Moshine Eleuldj, "Cloud Secured Protocol based on Partial Homomorphic Encryptions", 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech), 2018

7. Baocang Wang, Yu Zhan, Zhili Zhang, "Cryptanalysis of a Symmetric Fully Homomorphic Scheme", IEEE Transactions on Information Forensics and Security(Vol:13, Issue:6), 2018

8. Jung Hee Cheon, Jinsu Kim, "A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption", IEEE Treansactions on Information Forensics and Security(Vol:10, Issue:5), 2015

9. https://baffle.io/blog/why-is-homomorphic-encryption-not-ready-for-primetime/

10. Khalid El Makkaoui, Abdellah Ezzati, Abderrahim Beni Hssane, "Challenges of using Homomorphic Encryption to secure Cloud Computing", International Conference on Cloud Technologies and Applications (CloudTech)

11. https://www.sciencedirect.com/topics/computer-science/homomorphic-encryption

12. https://www.thesslstore.com/blog/what-is-homomorphic-encryption/

13. https://www.researchgate.net/publication/272863199_Applicability_of_Homomorphic_Encryption_and_CryptDB_in_Social_and_Business_Applications_Securing_Data_Stored_on_the_Third_Party_Servers_while_Processing_through_Applications

14. https://globaljournals.org/GJCST_Volume13/4-A-Study-of-Encryption-Algorithms.pdf

15. RSA Implementation by Cetin Kaya Koc ,Oregon State University

16. Yi Dou, Henry C. B. Chan, Man Ho Au, "A Distributed Trust Evaluation Protocol with Privacy Protection for Intercloud", IEEE Transactions on Parallel and Distributed Systems (Volume: 30 , Issue: 6), June 1 2019

17. Zainab Hikmat Mahmood, Mahmood Khalel Ibrahem, "New Fully Homomorphic Encryption Scheme Based on Multistage Partial Homomorphic Encryption Applied in Cloud Computing", 1st Annual International Conference on Information and Sciences (AiCIS), 2018

18. Lim Tsu Chean, Vasaki Ponnusamy, Suliman Mohamed Fati, "Authentication Scheme using Unique Identification method with Homomorphic Encryption in Mobile Cloud Computing",IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2018

19. Florencia Irena, Daniel Murphy, Sri Parameswaran, "CryptoBlaze: A Partially Homomorphic Processor with Multiple Instructions and Non-Dterministic Encryption Support", 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018