VOL 1 (2017) NO 1

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Advanced Homomorphic Encryption for Cloud Data Security

D.Chandravathi [#], Dr.P.V.Lakshmi [*]

[#] *GVP College for Degree and PG courses Rushikonda, Viakhapatnam-45*
E-mail: chandravathi.d@gmail.com
[*] GITAM University, Rushikonda, Viakhapatnam-45

Abstract— **This paper aims to provide security of data in the Cloud using Multiplicative Homomorphic Approach. Encryption process is done with RSA algorithm. In this RSA algorithm, Shor's algorithm is used for generating Public key Component, which enhances the security. Shor's algorithm plays as important role in generating public key. Plain Text Message is encrypted with Public Key to generate Cipher Text and for decryption Chinese Remainder Theorem (CRT) is used to speed up the computations. By doing so, it shows how the CRT representation of numbers in Zn can be used to perform modular exponentiation about much more efficiently using three extra values pre-computed from the prime factors of n. Hence, security is enhanced in the cloud provider.**

Keywords— **Multiplicative homomorphic, encryption, RSA algorithm, Shor's algorithm, CRT, public key, modular exponentiation**

I. Introduction

Security of data is the most important aspect that has to be considered in the cloud, which gives an importance and value of exchanged data over the Internet or other media types. This paper tries to increase a fair performance of the most commonly used RSA(Rivest-Shamir-Adleman) algorithm in the data encryption field[9]. Cryptography is usually referred to as "the study of secret", while now a day is most attached to the definition of encryption. Encryption is the process of converting plain text to a hidden form or scrambled manner to secure it against attacks[1]. This process has another part where cryptic text needs to be decrypted on the other end to be understood. The process of encryption is carried using RSA algorithm and the cipher is generated which is stored in the cloud. Further,in the cloud homomorphic multiplicative operation is performed for authentication . If authentication is successful then we using Chinese reminder theorem to decrypt the cipher text to generate plain text[1][2].

RSA Cryptosystem

The RSA system is an asymmetric public key cryptosystem .This means that there are many number of pairs of algorithms (E, D) both defined on the same set of values. E is the public encryption algorithm and D is the private decryption algorithm [4][9]. These satisfy:

- Encryption process followed by decryption process : If c=E(m) is the chipper text generated from some plaintext m, then m=D(c) i.e. m=D(E(m))
- Encrypt efficiently: For any message m, there is an efficient algorithm to calculate E(m).
- Decrypt efficiently: or any message or cipher text x, there is an efficient algorithm to calculate D(x).
- Public and private keys intact: From knowledge of E, there is no efficient way to discover D.
- Signing phase: The set of messages m are the same as the set of cipher texts c=E(m), for all m, so that the decryption algorithm can be applied to a message, resulting in signature.
- Verification: If s=D (m) is the signature corresponding to some plaintext m, then m=E(s).

Shor's Algorithm

This algorithm is used in the generation of the public key used in RSA encryption. It helps in the generation of two GCD values and from which we can decide on one as the public key. We first compute gcd of the number by using Euclidean algorithm.Then we check whether the gcd of the number is not equal to 1.If it so then it is the gcd otherwise we use to find period subroutine $a^{r/2} \not\equiv -1$ mod B,which generates two gcd values ie, gcd ($a^{r/2}$ + 1, N) and gcd ($a^{r/2}$ - 1, N) .The below is the Shor's algorithm :

Pick a random number $a < N$.

1. Compute gcd(a, N). This can be done using the Euclidean algorithm.
2. If gcd(a, N) $\neq 1$, then there is a nontrivial factor of N, so we are done.
3. Otherwise, use the period-finding subroutine (below) to find r.
4. If r is odd, go back to step 1.
5. If $a^{r/2} \not\equiv -1 \pmod{N}$, go to step 1.
6. gcd ($a^{r/2} + 1$, N) and gcd ($a^{r/2} - 1$, N) are both nontrivial factors of N. We are done.

Homomorphic Encryption

Homomorphic Encryption systems are used to perform operations on encrypted data.This is done without knowing the private key (without decryption).The client is the only holder of the secret key[4][5]. When we decrypt the result of any operation, it is the same as if we had carried out the calculation on the raw data. For example let us consider two ciphers as 3 and 2.Then we need to calculate the multiplicative operation for homomorphic encryption as

$3 * 2 = ?$

Encrypt 3 to get =10

Encrypt 2 to get =20

$10 * 20 = 200$

Decrypt 200 with private key to get original value 6.

Chinese Reminder Theorem:

The major importance of CRT is that it speeds up the calculations for RSA algorithm. CRT representation of numbers in $\mathbf{Z}_n$ is used to perform modular exponentiation which is about four times more efficient. By using three extra values pre-computed from the prime factors of n, and Garner's formula for the calculations make it more efficient.To decrypt ciphertext ‘ c ‘ or to generate a signature using RSA with private key (n, d), calculation of the modular exponentiation $m = c^d \bmod n$ is needed. The private exponent ‘d’ is not as convenient as the public exponent .Hence we can choose a value with as few '1' bits as possible. For a modulus n of k bits, the private exponent ‘d’ will be of similar length, with approximately half being '1'. CRT is used to compute $m = c^d \bmod n$ more efficiently which is given below.

CRT is used for the process of decryption of the data stored in the cloud.

- Compute the following values given P, Q with $P > Q$

  $dP = (1/e) \bmod (p-1)$

  $dQ = (1/e) \bmod (q-1)$

  $QInv = (1/Q) \bmod p$

  the (1/e) notation means the modular inverse. The expression $x = (1/e) \bmod N$ is also written as $x = e^{-1} \bmod N$, and x is any integer that satisfies $x.e \equiv 1 \pmod{N}$. In our case, where $N = n = pq$, we use the unique value x in $\mathbf{Z}_n$, $=\{0, 1, 2, ..., n-1\}$.
- To compute the Plain message m given c do

  $m_1 = c^{dP} \bmod P$

  $m_2 = c^{dQ} \bmod Q$

  $h = QInv.(m_1 - m_2) \bmod P$

  $m = m_2 + h*Q$

  Store the private key as the quintuple (P, Q, dP, dQ, $QInv$).

Chinese Remainder Theorem (CRT) and Euler's Theorem (also called the Euler-Fermat Theorem) are used from number theory.

II. MATERIALS AND METHOD

Proposed Method

There are many traditional algorithms for encryption and decryption such as RSA,DES,IDEA etc. But these are not sufficient to provide security in cloud, since much of calculations are done for using private key. The proposed scheme uses Shor’s algorithm for generating two GCD values which enhances more security and the computed cipher is stored in cloud. It provides confidentiality to the data because at no stage data is exposed in plain text and also uses Chinese remainder theorem(CRT) to decrypt the cipher which would be much more faster than standard RSA.

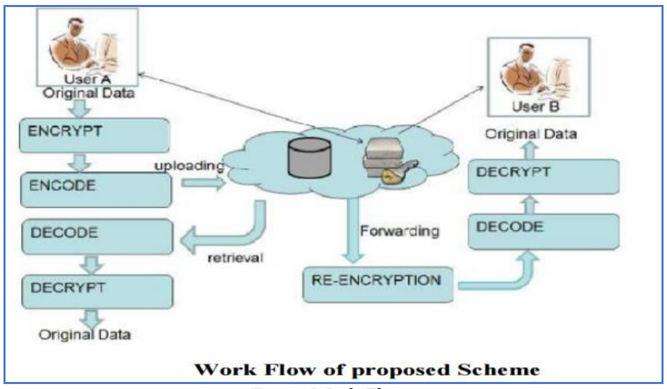


Fig. 1 Work Flow

In standard RSA, for encryption we need a public key. For finding it, we calculate ‘e’ value which is difficult for the user to choose such value. To overcome this problem shor’s algorithm is used to find two GCD values. It reduces the effort of user for finding public key.

Algorithm

- Step 1. Choose random large prime integers P and Q which are roughly the same size and which are not too close together.
- Step 2. Calculate the product N= P * Q.
- Step 3. Calculate Ø(N)=(P-1)*(Q-1)
- Step 4. Calculate ‘e’ such that 1 < e < phi(n)

  (For Calculating ‘e’ using Shor’s Algorithm, randomly choose

  ‘x’ between 1 and Ø(N), then find r = x mod Ø(N) and

  y = x^(r/2) mod Ø(N) .Then find GCD(y+1, Ø(N) =1 and

  GCD(y-1, Ø(N) =1)
- Step 5. Calculate ‘d’ such that e * d mod Ø(N) = 1
- Step 6. Encryption: C = M^e % Ø(N).

Homomorphic Encryption to provide Security in Cloud

- Step 7. Compute (C1 * C2*….. * Cn) and Store the Mulitple value in cloud.

- Step 8. Decrypt the Cipher Multiple Data with private key to get some value which is equal to the original multiples of the Plain Text.

  (if authentication is Failed then Stop the Process otherwise go to Step 9 for decryption)
- Step 9:Precompute the following values given P, *Q* with P > *Q*

  dP = (1/e) mod (P-1)

  dQ = (1/e) mod (Q-1)

  qInv = (1/Q) mod P

  where the (1/e) notation means the modular inverse. The expression

  x = (1/e) mod N is also written as x = e^{-1} mod N, and *x* is any integer that satisfies x.e ≡ 1 (mod N). In our case, where N = n = P*Q, we use the unique value *x* in $\mathbf{Z}_n$, the set of numbers {0, 1, 2, ..., n-1}.
- Step 10. To compute the Plain message *m* given *c* do

  $m_1 = c^{dP}$ mod P

  $m_2 = c^{dQ}$ mod Q

  h = QInv.(m_1 - m_2) mod P

  m = m_2 + h*Q

Store the private key as quintuple (P,*Q*, *dP*, *dQ*, *QInv*).

Example:

RSA encryption Process using Shor's algorithm for gcd generation:

- Let P = 137, Q= 131, N = 137*131 = 17947,
- Calculate phi(n)=136 * 130 =17680
- 1 < e < 17680 (using shor's Algorithm)

  choose random value x = 30

  r = 30 mod 17680 = 30

  and y = 30^15 mod 17680= 480
- GCD(481,17680) ≠ 1 GCD(479,17680) = 1
- Hence e = 479
- e * d % Ø(N) =1
- Therefore, d= 2879
- M1 = 513 (Plain Text) M2 = 171
- c1= 513^{479} mod 17947 = 5337 and c2 = 171^{479} mod 17947 = 13822
- c1 * c2 = 5337 * 13822 = 73768014 (Store this value in cloud for authentication)
- Decrypt c1 * c2 with the private key 'd'
- 73768014^{2879} mod 17947 = 15935(which is equal to the multiples of plain text message)
- 513 * 171 = 36423 mod 17947 = 15935 (Authentication Successful)

Standard RSA Process for Decryption:

- To decrypt *c* we could compute c^d mod n directly
- **m1 = 5337^{2879} mod 17947 = 513. m2 = 13822 ^{2879} mod 17947= 171**
- Pretty difficult to do on your pocket calculator. Now let's use the CRT method - notice how the exponent and modulus values are much smaller and manageable. This simple (but obviously insecure) example should demonstrate how much easier it is to break down the RSA calculation into smaller ones.

Proposed Process:

- dP = e^{-1} mod (P-1) = 479^{-1} mod 136 = 23
- dQ = e^{-1} mod (Q-1) = 479^{-1} mod 130 = 19
- QInv = Q^{-1} mod P = 131^{-1} mod 137 = 114
- $m_1 = c^{dP}$ mod P = 5337^{23} mod 137 = 102
- $m_2 = c^{dQ}$ mod Q = 5337^{19} mod 131 = 120
- h = QInv*(m_1 - m_2) mod P = 114.(102-120+137) mod 137 = 3

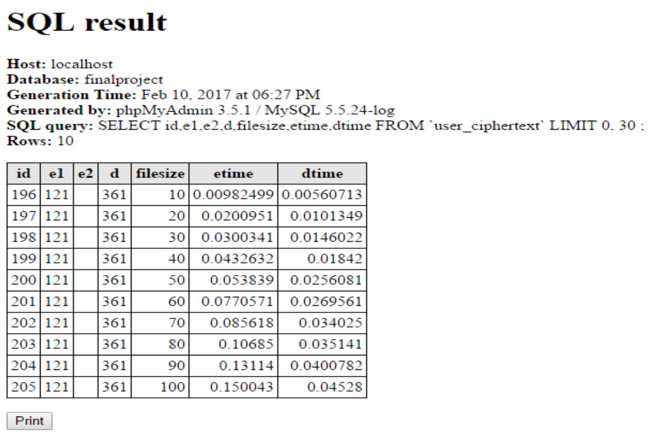
  [*we add in an extra p here to keep the sum positive*]
- m = m_2 + h*Q = 120 + 3.131 = 513.
- $m_1 = c^{dP}$ mod P = 13822 ^{23} mod 137 = 34
- $m_2 = c^{dQ}$ mod Q = 13822 ^{19} mod 131 = 40
- h = QInv*(m_1 - m_2) mod P = 114.(34-40+137) mod 137 = 1

  [*we add in an extra p here to keep the sum positive*]

  m = m_2 + h.Q = 40 + 1.131 = 171

Results:

The implementation is carried using python and the database as MySQL.Here the exection is done taking different file sizes with respect to the keyids .'e1' and 'e2' are public keys that are generated using shor's algorithm and d is the private key.Below is shown the analysis of encryption and decryption time. The average execution time for encryption and decryption is generated using MYSQL query which is shown below. It is clear that the process of decryption is faster than encryption process. Hence homomorphic encryption takes less time and it is faster and secure.



SQL result

Host: localhost
Database: finalproject
Generation Time: Feb 10, 2017 at 06:27 PM
Generated by: phpMyAdmin 3.5.1 / MySQL 5.5.24-log
SQL query: SELECT id,e1,e2,d,filesize,etime,dtime FROM `user_ciphertext` LIMIT 0, 30 ;
Rows: 10

id	e1	e2	d	filesize	etime	dtime
196	121		361	10	0.00982499	0.00560713
197	121		361	20	0.0200951	0.0101349
198	121		361	30	0.0300341	0.0146022
199	121		361	40	0.0432632	0.01842
200	121		361	50	0.053839	0.0256081
201	121		361	60	0.0770571	0.0269561
202	121		361	70	0.085618	0.034025
203	121		361	80	0.10685	0.035141
204	121		361	90	0.13114	0.0400782
205	121		361	100	0.150043	0.04528

Fig. 2 SQL Result

SELECT AVG(etime) FROM `user_ciphertext`;

avg(etime)

0.07077646255493164

SELECT AVG(dtime) FROM `user_ciphertext`;

avg(dtime)

0.025585246086120606

Encryption Analysis

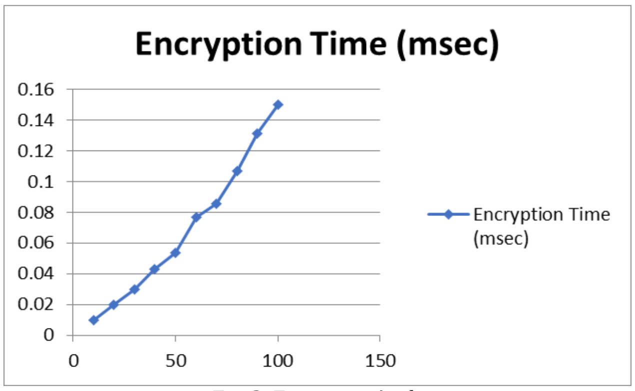


Fig. 3 Encryption Analysis

Decryption Analysis

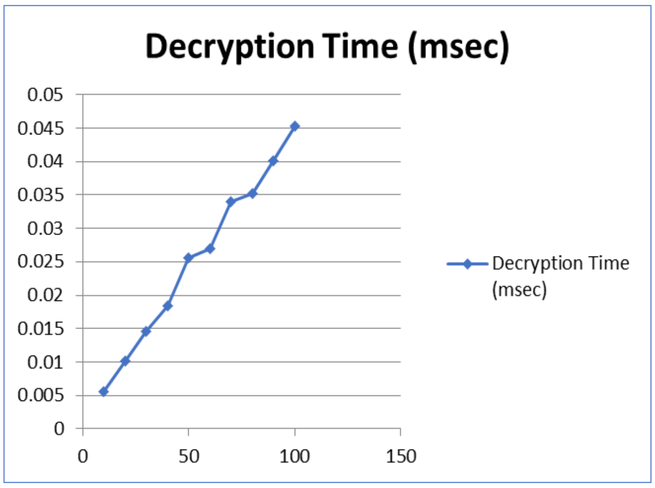


Fig. 4 Decryption Analysis

III. Conclusions

The Security of Data is a major concern in Cloud Computing. Homomorphic Encryption is a new concept of security which enables 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. From the analysis it is clear that the decryption process is decreased than encryption process. Even though computing of two exponentiations instead of one and there are additional steps involved in doing CRT, overall, the decryption would be much faster.

References

1. Faster RSA Algorithm for Decryption Using Chinese Remainder Theorem G.N. Shinde and H.S. Fadewar , ICCES, vol.5, no.4, pp.255-261.
2. C. Lamprecht "Investigating the efficiency of Cryptographic algorithm in Online Transaction"ISN 1473-804X online, 1473-8031 I.J. of Simulation Vol.7, No.2.
3. AlexanderMay, "Cryptanalysis of Unbalanced RSA with Small CRT-Exponent", CRYPTO 2002, LNCS 2442, pp 242-256, 2002.
4. JohannnesBlomer,Martin Otto, "a newCRT-RSA Algorithm Secure Against Bellcore", CC'03, October 27-30,Washington, DC, USA.
5. Craig Gentry, A Fully Homomorphic Encryption Scheme, 2009.http://crypto.stanford.edu/craig/craig-thesis.pdf.
6. Understanding Homomorphic Encryption http://en.wikipedia.org/wiki/Homomorphic_encryption.
7. Homomorphic Encryption Applied to the Cloud Computing Security Maha TEBAA, Saïd EL HAJJI, Abdellatif EL GHAZI.
8. Vic (J.R.) Winkler, "Securing the Cloud, Cloud Computer Security, Techniques and Tactics", Elsevier.
9. 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.
10. 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.*