RSA Algorithm & Signature In Cloud A Review

Mrs. Dhina Suresh
Research Scholar, Department of Computer Science,St. Joseph's College of Arts and Science for Women, Hosur-635 126
Email: dhinadulcy@gmail.com
Dr. M. Lilly Florence
Professor, Department of Computer Applications,Adhiyamaan College of Engineering, Hosur.
Email: lilly_swamy@yahoo.co.in

--**ABSTRACT**--
The use of cloud computing is rapidly increasing now a days. Cloud is an evolving trend today. It is an internet based service delivery model where user is provided services by CSP (Cloud Service Provider). Cloud provides many internet based services for users. It has many advantages one of which is virtualization. It is the key technique in cloud architecture because it allows greater flexibility and utilization of cloud. Some other benefits are scalability, reliability and efficiency. Security issues are the core problem in cloud computing services. It is required to protect the stored data and applications in the cloud from hackers and intruders.
This article gives an overview on the basics of cloud, cloud security issues and also RSA algorithm is discussed in detail. This paper discusses and gives an overview on homomorphic and digital signature.

Keywords - **cloud security, digital signature, homomorphic, privacy protection and virtualization.**

1.INTRODUCTION

Electronic data transfer is increasing in day today's world. It is very essential to protect these digital data. It is suggested to encrypt the data before storing it on the cloud or before the data is transferred. Cryptosystems are the essential tools that help to assure data storage and accuracy in cloud environment. The process of cryptography is to encrypt the plain text and convert it to a cipher then decrypts the cipher to get the plain text back. One of the commonly used public key cryptosystem is RSA (Ron Rivest, Adi Shamir and Leonard Adleman). In RSA, encryption keys are made public while the decryption keys are kept private, so only the person with the correct decryption key can decipher an encrypted message.

At the beginning of the paper, there is a brief overview of cloud computing models and data security. We discuss the RSA algorithm in detail. We will also have an overview of homomorphic cryptosystems and digital signature.

2.CLOUD MODELS

2.1. SOFTWARE-AS-A-SERVICE

It provides the user complete software or an application running on cloud infrastructure that is hosted by CSP. Users need not to buy or install it. Google Docs is an example.

2.2. PLATFORM-AS-A-SERVICE

The user develops an application through SaaS. The application has to be deployed. PaaS helps to deploy their application on the cloud. The user can control their application but not the infrastructure. PaaS is a delivery of computing platform as a service. Example of PaaS is Google App Engine.

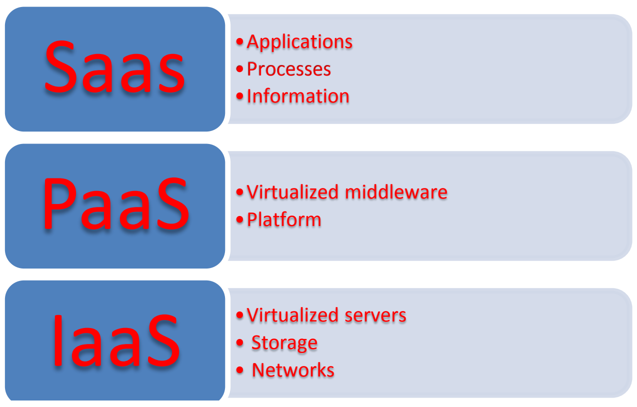

Fig 1: Cloud Models

2.3. INFRASTRUCTURE-AS-A-SERVICE

Through the IaaS the user gets access to resources like storage, server, networks and data center space. It shares the computing resources. User can also deploy and run the application as well the operating system on IaaS. Example of IaaS is Amazon EC2.

3.CLOUD SECURITY ISSUES

The following are the few attacks on cloud as per CSA (Cloud Security Alliance)

- Data breaches which is called as data loss and data leakage.
- Data loss.
- Account or service traffic hijacking.

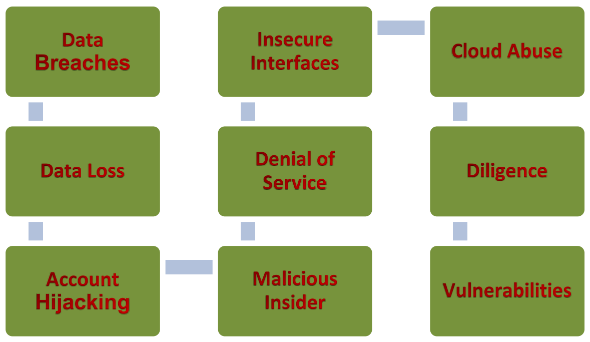

Fig 2: Cloud Security Issues

- Insecure interfaces and APIs
- DoS (Denial of Service)
- Malicious insider
- Cloud corruption or abuse
- Diligence
- Shared vulnerabilities

4.HOMOMORPHIC ALGORITHMS

The concept of homomorphic in Cryptography can be defined as the operations performed on the encrypted data without sharing the secret key. It is implemented in various public key cryptosystems. Data can be secured and stored using various encryption algorithms. To process the data stored in the server, homomorphic encryption is useful. It performs operations on the cipher text.

Normal symmetric encryption such as AES, DES etc. is not homomorphic. Homomorphic can be mainly classified into 2 types.

4.1. Partially Homomorphic

An encryption is said to be partially homomorphic where only one either addition or multiplication operation is performed on the cipher. The examples are as follows

4.1.1. Paillier Cryptosystem (Additive)

The encryption formula of Paillier Cryptosystem is as follows

$C = (g^m * h^n \bmod n^2)$, where C is the cipher text, m is the message.

Consider two cipher texts.

$C1= (g^{m1} * h1^n \bmod n^2)$ and

$C2= (g^{m2} * h2^n \bmod n^2)$

Then $C1.C2= (g^{m1}h1^n \bmod n^2)\ (g^{m2}\, h2^n \bmod n^2)$.

The additive property in Paillier Cryptosystem is

$C1.C2= (g^{m1+m2}\, (h1*h2)^{\,n} \bmod n^2)$

4.1.2. RSA Cryptosystem (Multiplicative)

The encryption formula of RSA Cryptosystem is as follows

$C= M^e \bmod n$, where C is the cipher text, M is the message.

Consider two cipher texts.

$C1= (M1^e \bmod n)$

$C2= (M2^e \bmod n)$

Then $C1.C2= (M1^e \bmod n)\ (M2^e \bmod n)$

The multiplicative property in RSA Cryptosystem is $C1.C2= (M1*M2)^e \bmod n$

4.2 Fully Homomorphic

An encryption is said to be fully homomorphic where different and efficient evaluation are performed on the cipher, example Gentry cryptosystem.

5.RSA ALGORITHM

RSA (Rivest, Shamir and Adleman) uses public key and private key to encrypt and decrypt messages. It was first published in 1978 as one of the first public-key cryptographic systems. A public-key system means the algorithm for encrypting a message is publicly known but the algorithm to decrypt the message is only privately known. RSA uses the modular arithmetic.

5.1. Key Generation

- Select two random numbers p & q. But the condition is (p! = q).
- Compute the value of n where n= (p*q).
- Compute the Euler's totient function phi, where Ø (phi) = (p – 1) (q – 1).
- Select a small odd integer (e) so that gcd (e, Ø) = 1.
- Conditions for e are as follows
  1. 1 < e < Ø
  2. e should be prime
  3. e and (Ø) must be co-prime.

Co-prime means that there is no common factor between e and (Ø) except 1. That is gcd (e, Ø) = 1.

- Receiver will create his own private key. Compute the value of d where (d * e) = (1 mod Ø).
- The value of d is calculated using Extended Euclidean Algorithm table method
- Condition for d is

1. If d is greater than Ø then d must be less than Ø so then d= (d mod Ø).

2. If d is negative then d must be positive so that d= (d+ Ø).

5.2. Encryption

- Encrypt: $C=m^e$ % n(sender encrypts it with senders public key and he sends it to the receiver e and n are public).

5.3. Decryption

- Decrypt: $P=C^d\%n$ receiver end (d is private key of receiver)

6. AN EXPERIMENT FOR RSA ALGORITHM

6.1. Key Generation

- Select two random prime numbers. We take 7 and 11.
- So, p=7 and q=11.
- $n = p * q$, n=7 * 11.
- n=77.
- Compute the value for Ø= (p - 1) (q - 1).
- Ø= (7 - 1) (11 - 1).
- Ø=6*10, Ø=60.
- Choose a random number e, e=13.
    1. 1 < e < Ø, 1<13<60.
    2. e should be prime, 13 is prime.
    3. e and (Ø) must be co-prime, that is gcd(e, Ø)=1.

In our case gcd (13, 60) =1.

- We got the public keys n=77 and e=13
- Receiver will create his private key. Compute the value for d, (d * e) mod Ø = 1
- (d*13) = 1 mod Ø ---------------------------------------Eq(A)
- We can also call d as the multiplicative inverse of (e^{-1} mod Ø) because we can also write d as d= e^{-1} mod Ø.
- We can find d using Extended Euclidean algorithm method.

Extended Euclidean algorithm

- Extended Euclidean algorithm states that ax + by=gcd(a , b)
- In our case a= Ø, b=e, we can re write the equation as Øx + ey=gcd (Ø, e), Substitute the values of Ø and e then we get,
- 60x + 13y=gcd(60,13) ----------------------------------Eq(B)
- The value of y will give us the value of d. We use the table method to get the value of y.

Table method to get the value of y

Row	x	y	d	k
1	1	0	Ø =60	-
2	0	1	e=13	60/13=4
3	1	-4	8 (it must also be the remainder of 60 /13)	13/8=1
4	-1	5	5 (it must also be the remainder of 8 /5)	8/5=1
5	2	-9	3 (it must also be the remainder of 5 /3)	5/3=1
6	-3	14	2 (it must also be the remainder of 3 /2)	3/2=1
7	5	-23	1 (it must also be the remainder of 2 /1)	2/1=2

Set the initial values of x and y as 1, 0 and 0, 1

Row 3:

x3= (x1-x2*k2), 1-0*4=1
y3= (y1-y2*k2), 0-1*4=-4
d3= (d1-d2*k2), 60-13*4=8

Row 4:

x4= (x2-x3*k3), 0-1*1=-1
y4= (y2-y3*k3), 1 - (-4)*1= 5
d4= (d2-d3*k3), 13-8*1=5

Row 5:

x5= (x3-x4*k4), 1-(-1)*1=2
y5= (y3-y4*k4), -4-5*1=-9
d5= (d3-d4*k4), 8-5*1=3

Row 6:

x6= (x4-x5*k5), -1-2*1=-3
y6= (y4-y5*k5), 5-(-9)*1=14
d6= (d4-d5*k5), 5-3*1=2

Row 6:

x7= (x5-x6*k6), 2-(-3)*1=5
y7= (y5-y6*k6), -9-14*1=-23
d7= (d5-d6*k6), 3-2*1=1

We stop the calculation since the final value of d in row 7 is 1.
From the above table we can get the following values.

- x=5, y=-23
- We substitute the above values in Eq(B)
- 60x + 13y=gcd(60,13) ----------------------------------Eq(B)
- 60*5 + 13*-23=gcd(60,13)
- 60*5 + 13*-23=1---L.H.S
- gcd(60,13) is 1---R.H.S
- L.H.S=R.H.S, so the value of x and y are correct
- The value of y gives the value of d.
- The private key of the receiver is d=-23.
- Condition for d is
    1. If d is greater than Ø then d must be less than Ø so then d=d mod Ø
    2. If d is negative then d must be positive so that d=d+ Ø
- In our case d is negative so d=d+ Ø, d=-23+60
- d=37
- We have generated the receivers private key d=37 where the public keys are e=13 and n=77.

6.2. Encryption

- Encrypt: $C=m^e \% n$ where C is the cipher text, m is the message, e=13 and n=77.
- M can be any plain text. In our case we consider m=5. Now 5 has to be encrypted.
- $C=5^{13}$ mod 77
- We can use the modular exponentiation method to encrypt the plain text.
- We implement right to left binary method algorithm to implement modular exponentiation.

Encryption using Modular Exponentiation Algorithm

- Right to left binary method algorithm

```
Declare the variables: base, expo, modulus, result=1
   while (expo>0)
   if current_bit= =1
            result=(result*base) mod modulus
   expo=expo>>1 (shift expo 1 bit right)
   base=(base*base) mod modulus
   return result
```

- Let us implement the Modular Exponentiation algorithm.

In our case $C=5^{13}$ mod 77, base=5(plain text), expo=13(1101), modulus=77.

```
Iteration1:
   base=5, expo=13(1101), result=1, modulus=77
   while (1101>0)
   if current_bit= =1 (condition is true)
            result=1*5 mod 77 (5%77=5)
   expo=0110 (shift 1 bit right)
   base=5*5 mod 77(25%77=25)

Iteration2:
   base=25, expo=0110, result=5, modulus=77
   while (0110>0)
   if current_bit= =1(condition is false)
            result=5
   expo=0011 (shift 1 bit right)
   base=25*25 mod 77(625%77=9)

Iteration3:
   base=9, expo=0011, result=5, modulus=77
   while (0011>0)
   if current_bit= =1(condition is true)
            result=5*9 mod 77 (45%77=45)
   expo=0001 (shift 1 bit right)
   base=9*9 mod 77(81%77=4)

Iteration4:
   base=4, expo=0001, result=45, modulus=77
   while (0001>0)
   if current_bit= =1(condition is true)
            result=45*4 mod 77 (180%77=26)
   expo=0000 (shift 1 bit right)
   base=4*4 mod 77(16%77=16)

Iteration3:
   base=9, expo=0011, result=5, modulus=77
   while (0000>0)
   condition false.
   The encrypted text C=26.
```

6.3. Decryption

- Decrypt: $M=C^d\%n$ receiver end (d is private key of receiver)
- $M=26^{37}$ mod 77
- We can use the same modular exponentiation method to decrypt the cipher text.
- We implement right to left binary method algorithm to implement modular exponentiation.
- If we decrypt we get the plain text 5.

7. DIGITAL SIGNATURE

Digital signature is one of the modern authentication concepts in cryptography. The signature ensures integrity and authenticity. The signature can be defined as a number known only to the signer. Digital signatures may be possible with secret key cryptosystem but true digital signatures are possible only with public key cryptography.

Digital signature algorithm can be classified as follows.

- Digital signature with appendix
- Digital signature with message recovery

7.1. Digital signature with appendix

- Requires the original message as input to the verification algorithm.
- There are two phases:
  - Key generation phase and
  - Signature phase
- Uses hash functions.
- DSA, DSS. ElGamal are examples

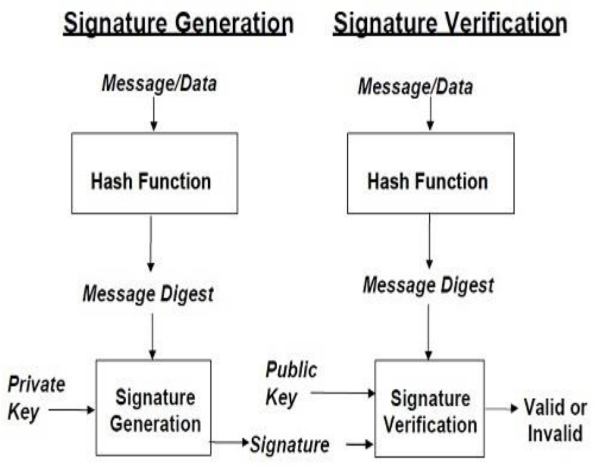

Fig 3: Digital Signature with hash function

7.1.2. Example for Digital signature with appendix

Romeo produces a signature s in signature space S for a message m in the message space M which can be verified later by Juliet.

He uses any hash function and creates the hash of the message space M_h=h (m).

- Key Generation

Romeo selects his private key as R_{Pri} and he defines a signing algorithm S_A.

- Signature Generation

Romeo defines his public key R_{Pub} defining a verification algorithm V_A such that V_A (m*, s) =true if S_A (m*) =s or false otherwise for m* ∈ M_h and s ∈ S where m*=h (m) for m ∈ M.

Compute m*= h (m) and s = S_A (m*)
Send (m, s) to Juliet.

- Signature Verification:

Obtain Romeo's public key R_{Pub}
Compute m* = h (m) and u = V_A (m*, s)
Accept if u is true.

7.2. Digital signature with message recovery

- Do not require the original message as input to the verification algorithm.
- Original message is recovered from signature itself.
- RSA, Rabin are examples

7.2.1. Example for Digital signature with message recovery

M is the message space
M_S is the signing space
S is the signature space

- Key Generation:
  Romeo selects his private key as R_{Pri} and he defines a signing algorithm S_A.
- Signature Generation:
  Romeo defines his public key R_{Pub} defining a verification algorithm V_A such that $V_{A\,*}\,S_{A\text{ is}}$ identity map on Ms

Compute m* = R(m), R is a redundancy function (invertible)
Compute s = S_A(m*)
Obtain authentic public key V_A
Compute m* = V(s)

- Signature Verification:

Verify if m* ∈ M_S (if not, reject the signature) Recover the message m = R-1(m*)

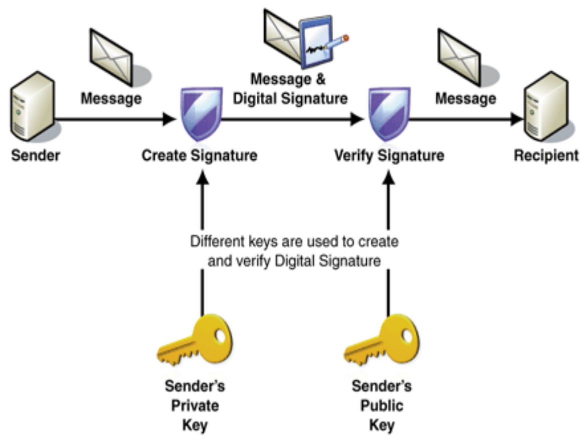

Fig 4: Digital Signature without hash function

8. AN INTRODUCTION TO RSA SIGNATURE

Let us see the digital signature based on RSA algorithm.

8.1. RSA Key Generation

- Select two random numbers p & q.
- Compute the value of n where n=p*q.
- Compute Ø (phi) = (p – 1) (q – 1).
- Select number (e) so that gcd (e, Ø) = 1.
- Compute the value of d where (d * e) = 1 mod Ø.

8.2. Signature Generation and Verification

- Signing message M with the sender's private key.
  Verify 0 < M < n
  Compute S = M^d mod n

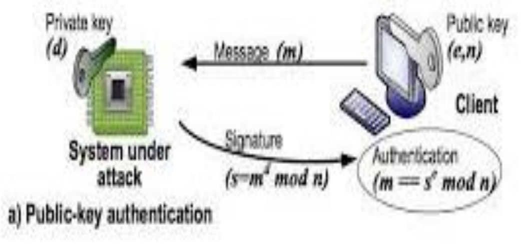
Private key (d)
Message (m)
Public key (e,n)
Client
System under attack
Signature ($s=m^d$ mod n)
Authentication ($m == s^e$ mod n)
a) Public-key authentication

Fig 5: Digital Signature Generation

- Verifying signature S with the sender's public key e and n
  Use public key (e, n)
  Compute M= S^e mod n

9. AN EXPERIMENT FOR RSA SIGNATURE

We consider the same inputs as given in RSA algorithm for easy calculation.

- Select two random numbers p & q.
  p=7, q=11
- Compute the value of n where n=p*q.
  n=77
- Compute Ø (phi) = (p – 1) (q – 1).
  Ø=60
- Select number (e) so that gcd (e, Ø) = 1.
  e=13

- Compute the value of d where (d * e) = 1 mod Ø.
  d=37
  We consider the message m=5
- Signature generation
  Compute s = m^d mod n.
  s=5^{37} mod n, we get the value after modular exponentiation calculation.
  s= 47.
- The message and the signature are sent to the receiver. Send (5, 47).
- Signature verification
  Compute m= s^e mod n.
  The receiver takes the signature and raises it to the e modulo n, then makes sure that this value is equal to the

message that was received, which it is, so the message is valid.

47^{13} mod 77=5 hence the signature is verified

10.RSA MERITS AND DEMERITS

10.1. Advantages

- Primary advantage of RSA is increased security
- Easy to understand and implement
- Encryption and verification are similar
- Signing and decryption are similar
- It is deployed widely
- Brute force attack has been reduced in RSA
- Timing attack is minimized by random delay by multiplying random number with cipher text.
- Since it uses 2048 bits exponents, mathematical attacks are also reduced.

10.2. Disadvantages

- Very slow in generating keys when compared to other symmetric algorithm
- The two are vulnerable to GCD. The keys may have a chance to get attacked if not implemented properly

11.CONCLUSION AND FUTURE WORKS

We saw a brief introduction to homomorphic cryptosystems. RSA can be claimed as a strong encryption algorithm. We have seen two experiments of RSA algorithm and signature. In my future work I would try to increase the security of the data and reducing the attacks by implementing both RSA encryption and signature. I would also like to implement hashing with RSA encryption and signature algorithms

12.ACKNOWLEDGEMENT

I would like to thank THE LORD MY SAVIOR for guiding me and showering HIS blessings upon me throughout my life. I take immense pleasure in thanking my guide Dr. M. Lilly Florence for rendering her valuable knowledge and guidance. I would like to thank my husband for his love and support. I would like to thank my parents and my son for their patience and care.

REFERENCES

[1] W.Diffie and M. Hellman." New Directions in Cryptography". IEEE transactions on Information Theory. IT-22(1978).472-492.

[2] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, volume 21, pages 120-126, February1978.

[3] Dario Catalano, Rosario Gennaro and Shai Halevi, *Computing inverse over a shared secret modulus*, IBM T. J. Watson Research center, NY, USA, 1999.

[4] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik, *New multiparty authentication services and key agreement protocols*, IEEE Journal of Selected Areas in Communication, 18(4), 2000.

[5] Ronald L. Rivest, Adi Shamir, Len Adelman, "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science Technical Memorandum 82 (April 1977).

[6] William Stallings, "Cryptography and Network Security Principles and Practices", 4th edition, Pearson Education Inc, 2006.

[7] An Improved RSA Encryption Algorithm For Cloud Computing Environments : Two Key Generation Encryption (2KGEA) – Ms Shubhra Saggar (Faculty, Guru Nanak Institute Of management, New Delhi), Dr. R.K.Datta (Director, M.E.R.I.T., New Delhi) (Research Paper on IJSWS).

[8] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in Cryptology (EUROCRYPT"99), vol. 1592 of Lecture Notes in Computer Science, pp. 223–238, Springer, New York, NY, USA, 1999.

[9] C. Fontaine , F. Galand, A survey of homomorphic encryption for nonspecialists, EURASIP Journal on Information Security, 2007,p.1-15, January 2007.