International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.5, No.5, May 2016 **DOI:10.15693/ijaist.2016.v5.i5.65-70**

Comparison of Timing Attack on AES and RSA Algorithms for Data Security on Cloud Storage

ANJU P.R
Department of CSE
Adi Shankara Institute of Engineering and Technology Kalady/MG University, Kerala, India

DEEPIKA M.P
Assistant Professor/Department of IT
Adi Shankara Institute of Engineering and Technology Kalady/MG University, Kerala

***Abstract*— Data security on cloud storage is one of the challenging problem in nowadays. Multi tenants can store their on data on cloud and share it with others with owner permission. One of the way to give security on data is through encryption methods. We need an efficient cryptographic method that can tolerate any attacks from malicious users on data which is stored on cloud. Symmetric as well as Asymmetric algorithms are considering for encrypting the data for security. In this paper, we implement a virtual environment for executing AES and RSA algorithm and comparing these two algorithms with Timing Attack. The result shows that performance of AES algorithm give better tolerance against timing attack.**

***Index terms* - AES, Asymmetric algorithm, RSA, Symmetric algorithm, Timing attack.**

I. INTRODUCTION

A cloud storage provider is a third-party company that offers organizations and end users the ability to save data to an off-site storage system. Instead of storing data to a local hard drive or other local storage device, such as tape backup or flash drives, the data is stored on a system in a remote data center. Customers can access their files from anywhere with an Internet connection. Both individual and corporate customers can store unlimited data capacity on the provider's servers at a low price point. The key to a cloud storage model is flexibility, so the user has the advantage of not worrying about storage limitations. Typically, they only pay for what they use, much like a utility. The key to storing data in the cloud is availability, and the level of availability can impact the price. Companies can use cloud storage providers as an offsite vault for backups, thus eliminating the need for tape backups, or they can use cloud storage as the primary destination for backups.

It can also be used for primary storage. Cloud encryption is a service offered by cloud storage providers whereby data, or text, is transformed using encryption algorithms and is then placed on a storage cloud. Cloud encryption is the transformation of a cloud service customer's data into ciphertext. Cloud encryption is almost identical to in-house encryption with one important difference -- the cloud customer must take time to learn about the provider's policies and procedures for encryption and encryption key management. The cloud encryption capabilities of the service provider need to match the level of sensitivity of the data being hosted. Because encryption consumes more processor overhead, many cloud providers will only offer basic encryption on a few database fields, such as passwords and account numbers. At this point in time, having the provider encrypt a customer's entire database can become so expensive that it may make more sense to store the data in-house or encrypt the data before sending it to the cloud. To keep costs low, some cloud providers have been offering alternatives to encryption that don't require as much processing power. These techniques include redacting or obfuscating data that needs to remain confidential or the use of proprietary encryption algorithms created by the vendor.

The cryptographic algorithms used to encrypt the data before storing into cloud. This paper implements a comparative study between cryptographic algorithm such as AES and RSA based on the parameters such as speed and encryption time on data also the timing attacks which are later uploaded into cloud storage. The timing attack of both algorithms is analyzed by taking the encryption time as parameter. Timing attack can be defined as the size of the file is proportional to the encryption time of that file. In this paper the algorithms are executing in a virtual environment.

II. RELATED WORK

A. Methods and Materials

2.1 RSA Algorithm

RSA was first described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology. Public-key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys, one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method of assuring the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage. [1]

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- factoring -- is considered infeasible due to the time it would take even using today's super computers.

The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are generated using the Rabin-Miller primality test algorithm. A modulus n is calculated by

multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus n, and a public exponent, e, which is normally set at 65537, as it's a prime number that is not too large. The e figure doesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient of n.[1][2]

2.1.1 Generation of RSA Key Pair:

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below

1) Generate the RSA modulus (n)

- Select two large primes, p and q.
- Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

2) Find Derived Number (e)

- Number e must be greater than 1 and less than (p 1)(q 1).

There must be no common factor for e and (p 1)(q 1) except for 1. In other words two numbers e and (p 1)(q 1) are coprime.

3) Form the public key

- The pair of numbers (n, e) form the RSA public key and is made public.
- Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

4) Generate the private key

- Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
- Number d is the inverse of e modulo (p -1)(q 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo(p - 1)(q - 1).
- This relationship is written mathematically as follows as e^d = 1 mod (p 1)(q 1)

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

2.1.2 RSA Algorithm Equation [3]

1. ProductOfPrime1Prime2 = Prime1 * Prime2
2. Totient = (Prime1 – 1) * (Prime2 -1)
3. (Totient * AnyInteger) + 1 = 1 mod Totient
4. EncryptPrime * DecryptPrime = 1 mod Totient
5. EncryptPrime * DecryptPrime = (Totient * AnyInteger) + 1 where (Totient * AnyInteger) + 1 has exactly prime factors
6. CipherText = $PlainText^{EncryptPrime}$ mod ProductOfPrime1Prime2
7. PlainText = $Ciphertext^{DecryptPrime}$ mod ProductOfPrime1Prime2
8. PlainText = $Ciphertext^{DecryptPrime}$ mod ProductOfPrime1Prime2 = $(PlainText^{EncryptPrime})^{DecryptPrime}$ mod ProductOfPrime1Prime2 = $(PlainText^{DecryptPrime})^{EncryptPrime}$ mod

2.1.3 Example: An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).

1. Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
2. Select e = 5, which is a valid choice since there is no number that is common factor of 5 and (p 1)(q 1) = 6 12 = 72, except for 1.
3. The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
4. Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm.
5. The output will be d =29.Check that the d calculated is correct by computing de = 29 5 = 145 = 1 mod 72Hence, public key is (91, 5) and private keys is (91, 29).

2.1.4 Encryption and Decryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy. Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent the plaintext as a series of numbers less than n.[3]

- *RSA Encryption*

[1] Suppose the sender wish to send some text message to someone whose public key is (n, e).

[2] The sender then represents the plaintext as a series of numbers less than n.

[3] To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as − C = P^emod n

In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n. Returning to our Key Generation example with plaintext P = 10, we get ciphertext C ,C = 105 mod 91

- *RSA Decryption*

The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C.[4] Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.Plaintext = C^ d mod n. Returning again to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 −Plaintext = 8229 mod 91 = 10

2.2 AES Algorithm

AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. However, AES is quite different from DES in a number of ways. The algorithm Rijndael allows for a variety of block and key sizes

and not just the 64 and 56 bits of DES' block and key size. AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys -128,192,256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES-256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure.[5] Recall that in a feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. In this case the entire data block is processed in parallel during each round using substitutions and permutations. A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particular implementation.[6] ;

Encryption Time = time taken to encrypt the data
= end time (end time of execution of algorithm on virtual environment) – start time (start of execution of algorithm on virtual environment)

The input is a single 128 bit block both for decryption and encryption and is known as the in matrix. This block is copied into a state array which is modified at each stage of the algorithm and then copied to an output matrix . Both the plaintext and key are depicted as a 128 bit square matrix of bytes. This key is then expanded into an array of key schedule words (the w matrix). It must be noted that the ordering of bytes within the in matrix is by column. The same applies to the w matrix. [7]
The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm [8][9]. The four stages are as follows:

- Substitute bytes
- Shift rows
- Mix Columns
- Add Round Key

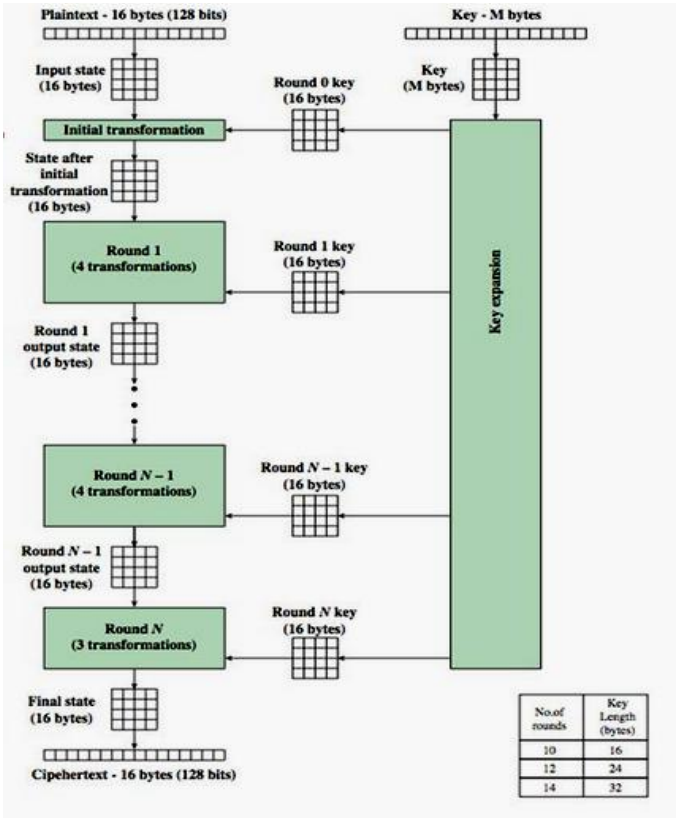

Figure 1 AES Algorithm internal structure

The tenth round simply leaves out the Mix Columns stage.
The first nine rounds of the decryption algorithm consist of the following:

- Inverse Shift rows
- Inverse Substitute bytes
- Inverse Add Round Key
- Inverse Mix Columns

1. Byte Substitution (SubBytes)}

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.[10][11]

2. Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.

The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

3. MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

4. Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

2.2.1 AES Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

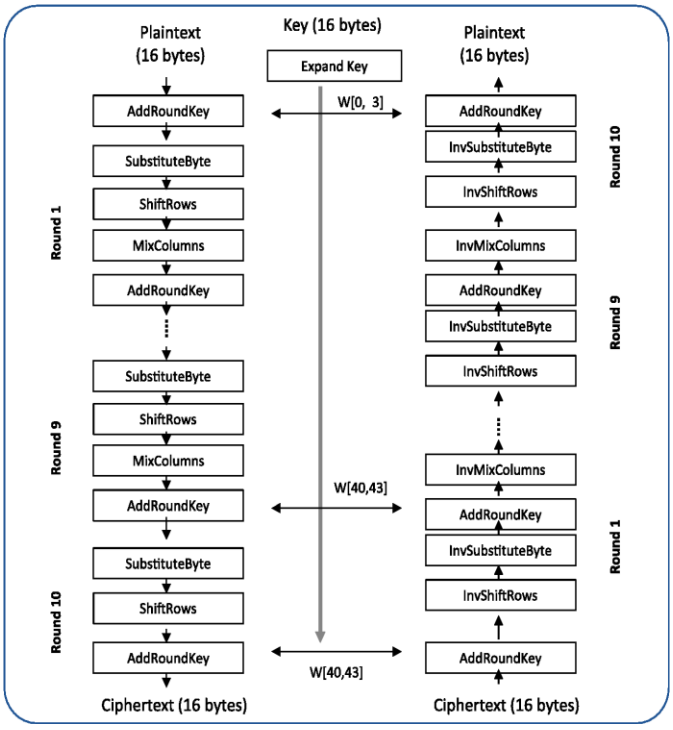

Figure 2 AES Overview with Operations

III. Objectives & Overview Of The Proposed Mechanism

A. Objectives

This paper implements the comparative study among the cryptographic algorithm such as AES and RSA. The comparison parameter is considered to be time to encrypt the different size input file. AES takes different size input file with same key for encryption.

IV. Performance Evaluation

A. Simulation Model and Parameters

We use Virtual Machine to simulate our algorithms. In our simulation, In this simulation the different size text file is inputed to AES and RSA algorithm. We assume each algorithms are simulated under the same execution environment.

Our simulation settings and parameters are summarized in table 1.

Virtual machine specification	Features
OS	Ubuntu(32bits)
Storage Memory	2048 Mb

B. Results

The execution environment for Algorithms is following properties such as Ubuntu 32 bit Operating System.

EXP NO	10KB	20KB	30KB	40KB	50KB	60KB	70KB	80KB
1	0.009145	0.005941	0.004892	0.011230	0.061244	0.043417	0.337378	0.018755
2	0.003993	0.003133	0.006561	0.009273	0.011530	0.012721	0.023599	0.008750
3	0.006247	0.000092	0.007816	0.010442	0.009653	0.018040	0.017347	0.011837
4	0.008340	0.004833	0.009626	0.009647	0.013263	0.009363	0.010537	0.020437
5	0.010052	0.000175	0.010419	0.008892	0.010152	0.007139	0.010815	0.019529
6	0.007294	0.008415	0.011771	0.011895	0.008873	0.013250	0.014084	0.010655
7	0.005180	0.009112	0.006071	0.010774	0.011481	0.012895	0.013219	0.017239
8	0.008305	0.010800	0.007652	0.007052	0.011663	0.015238	0.012449	0.011091
9	0.000110	0.008413	0.011210	0.011593	0.011466	0.011204	0.019095	0.016010
10	0.003871	0.011809	0.010139	0.010906	0.014909	0.011232	0.014247	0.0127737
AVG	0.0062537	0.0062723	0.0086157	0.0090930	0.0164234	0.0154499	0.0472770	0.0147040

Table 2 Encryption time for AES algorithm in different file size with same key.

EXP NO	10KB	20KB	30KB	40KB	50KB	60KB	70KB	80KB
1	0.150575	0.421563	0.331769	0.397825	0.960479	0.582362	0.071831	0.958212
2	0.221773	0.610865	0.613496	0.747840	0.324772	0.511911	0.845416	0.388992
3	0.302370	0.642307	0.769482	0.431926	0.080737	0.936259	0.652985	0.045310
4	0.333956	0.269814	0.548035	0.018758	0.938688	0.441213	0.043597	0.585959
5	0.229141	0.441293	0.390785	0.626797	0.237791	0.519500	0.591381	0.333069
6	0.131224	0.361750	0.322429	0.515207	0.706499	0.230264	0.758899	0.921634
7	0.315876	0.512446	0.610864	0.810121	0.770216	0.630911	0.808178	0.266627
8	0.242674	0.646144	0.053661	0.051416	0.405706	0.583091	0.707114	0.858619
9	0.157089	0.533889	0.493522	0.711984	0.547068	0.587836	0.587843	0.432403
10	0.248523	0.266094	0.323795	0.568758	0.509316	0.947856	0.683716	0.633117
AVG	0.2333201	0.4306165	0.4457838	0.4880632	0.5481272	0.5971003	0.5990960	0.5993944

Table 3 Encryption time for RSA algorithm in different file size with same prime number keys.

Table 2 for AES and Table 3 for RSA algorithm shows that the times taken to encrypt the different size file with same key. Using the virtual machine environment execute algorithm 10 times and the time noted. The average of time is calculated. The corresponding graph is plotted.

In the graph X axis is consider as time taken to encrypt the files and Y axis is consider as size of file in KB. First Graph shows the performance of AES algorithm and second graph shows performance of RSA algorithm.

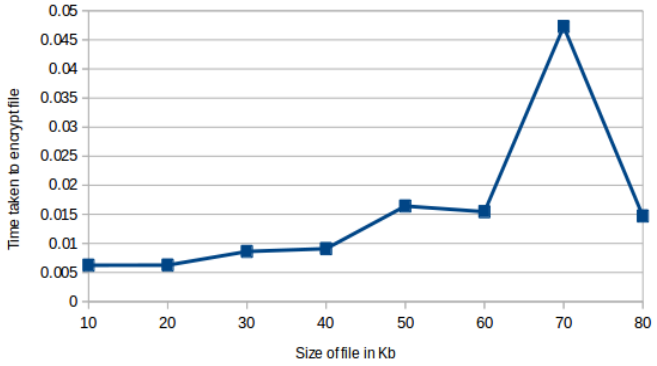

Figure 3 Graph shows the timing attack performance of AES algorithm (For Table1)

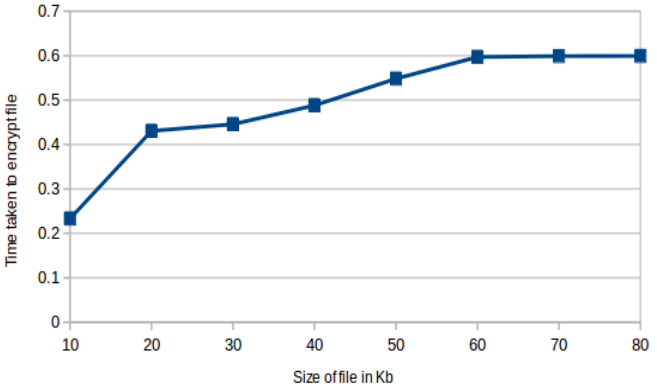

Figure 4 Graph shows the timing attack performance of RSA algorithm (For Table2)

The result shows that the encryption time for AES algorithm is following an irregular pattern and can not predict the size of input file compared with encryption time. For certain range it will decreasing and some range it will increasing. The RSA algorithm following increasing pattern when the size is increasing. So we can predict the size of the files compared to encryption time. Thus analysis gives AES is better against timing attack compared to RSA algorithm.

V. CONCLUSION

Cryptographic algorithms have great value when it club with security related problems in any areas. Always we need a efficient algorithm that couldn't break easily by malicious users. In order to find the efficiency we can check it against many attacks. AES algorithm is proved to be more secure and efficient algorithm nowadays where security is important concern. In this paper it is proven that AES is better one against timing attack compared to RSA algorithm. Cloud storage stores multiple tenant's data with encrypted format. For that we need a efficient cryptographic algorithm such as AES. In future we can check it other attacks such as network attacks compared with other algorithms.

REFERENCES

[1]. Nicolas courtois, Josef pieprzyk, *"Cryptanalysis of Block Ciphers with Overdefined Systems of Equations". PP267–287,* Asiacrypt 2002.

[2]. Cormen, Thomas H.; Leiserson, Charles E.;Rivest, Ronald L.; Stein, Clifford (2001).*Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill. pp. 881–887.ISBN 0-262-03293-7.*

[3]. A. Menezes, P. Van oorschot, and S. Vanstone, handbook of applied cryptography, CRC press, new york, 1997, p. 81-83.

[4]. Rivest, R., A. Shamir, L. Adleman (1978)."A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Communications of the ACM* **21** (2): 120–126. doi:10.1145/359340.359342.

[5]. Christof Paar, Jan Pelzl, *"The Advanced Encryption Standard", Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". (companion web site contains online lectures on AES),Springer, 2009.*

[6]. Joan Daemen, Vincent Rijmen, *"The Design of Rijndael: AES – The Advanced Encryption Standard."* Springer, 2002. ISBN 3-540-42580-2

[7]. Menezes, Alfred, Van oorschot, Paul c., Vanstone, Scott a. *(OCTOBER 1996).Handbook of Applied Cryptography . CRC press. ISBN 0-8493-8523-7 .*

[8]. *J.* Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999

[9]. J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

[10]. A. Lee, Nist special publication 800-21,guideline for implementing cryptography in the federal government, national institute of standards and technology, november 1999.

[11]. J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2.

Authors Profile

Anju P R is currently doing her master's degree in Technology, specializing in Computer Science and Engineering at Adi Shankara Institute of Engineering and Technology, Kalady. Her areas of interest include network security, Cryptography and Steganography

Deepika M P is currently working at Adi Shankara institute of Engineering and Technology as Assistant Professor in Information Technology Department. Received her M.Tech degree in Software engineering from CUSAT. Her area of interest is in Visual Cryptography.