Basic Concept of AES and RSA

Hansen-13517146
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517146@std.itb.ac.id

Abstract. This article proposes a little knowledge about cryptography, RSA (Rivest-Shamir-Adleman), and the advanced encryption standard known as AES. The two are encryption and decryption method using different algorithm and standards. Other than definitions and implementations, this article also talk about the 'attacks' and the differences between both method.

AES, RSA, Encryption, Decryption, Cryptography.

I. Introduction to Cryptography

Cryptography is the foundation of all security technologies in now days, becoming an indispensable tool for protecting information in any system (mostly computer). It has been used since the Egyptians era known as *hieroglyph*. The code was secret, only the scholars knew about it, one who used to transmit messages on behalf of the kings. Later, after the alphabetic era, they use Cipher's idea to encrypt messages. The idea is to replace the word with other alphabet using some secret rule. This secret rule will become the key to decrypt the informations.

This article will mainly talk about RSA and AES (Advanced Encryption Standard) thoroughly, from each of their definition, history, operation, version, implementation, weakness, and how to generate those 'key'.

II. CRYPTOGRAPHY PRIMITIVES

This table contains cryptography primitives :

Table 1.1 Cryptography Primitives

Primitive Service	Encryption	Hash Function
Confidentiality	Yes	No
Integrity	No	Sometimes
Authentication	No	No
Non Reputation	No	No

(source : tutorialspoint.com)

Table 1.2 Cryptography Primitives

Primitive Service	Message Authentication Codes (MAC)	Digital Signature
Confidentiality	No	No
Integrity	Yes	Yes
Authentication	Yes	Yes
Non Reputation	Sometime	Yes

(source : tutorialspoint.com)

III. Advanced Encryption Standard

A. History and General Definition

The Advanced Encryption Standard, known as AES is a symmetric encryption algorithm, one of the most secure and used by The United States Government to protect many information and software products. AES development has been started by The National Institute of Standards and Technology (NIST) since 1997. They said that the world need a successor algorithm for the Data Encryption Standard (56 bits key) which was published in 1977 because many people tried to decrypt this encryption type by brute force attacks. The algorithm was chosen among 5 submissions, Rijndael, submitted by Joan Daemen and Vincent Rijmen, MARS, submitted by IBM Research, Serpent, submitted by Ross Anderson, Eli Biham, and Lars Knudsen, RC6, submitted by RSA Security, Twofish, submitted by Counterpane Internet Security including Bruce Schneier. All of the submission were tested in C, Java, and ANSI languages for speed and reliability purpose. The Rijndael cipher was selected as the proposed algorithm for AES and published later. The method basically uses a block cipher, which encrypts data bit by bit. In now days, AES is widely used and supported in many hardware and software.

AES itself is a subset of the Rijndael's block cipher method. In the United States, AES was announced by NIST as the U.S. FIPS PUB 197, following a five year standardization process. AES became effective as a government standard starting from May 26, 2002 and included in IEC standard.

AES is based on a design known as substitution and permutation network, therefor, it is efficient in both software and hardware. AES is no longer using Feistel network like its predecessor (DES). AES's block size is fixed to 128 bits and three key size of 128 bits, 192 bits, and 256 bits. With this, AES only operates on a 4x4 matrix of bytes. If there are 16 bytes, x_0, $x_1, x_2, \ldots, x_{15}$, these bytes are represented as this two dimensional array :

$$\begin{bmatrix} x0 & x4 & x8 & x12 \\ x1 & x5 & x9 & x13 \\ x2 & x6 & x10 & x14 \\ x3 & x7 & x11 & x15 \end{bmatrix}$$

Key size used for this method specifies the number of transformation rounds that convert the input called the plaintext into the final output called the ciphertext.

B. Version and Operation

AES has some version of itself, such as AES-128, AES-192, and AES-256. It depends on the key we choose to encrypt and

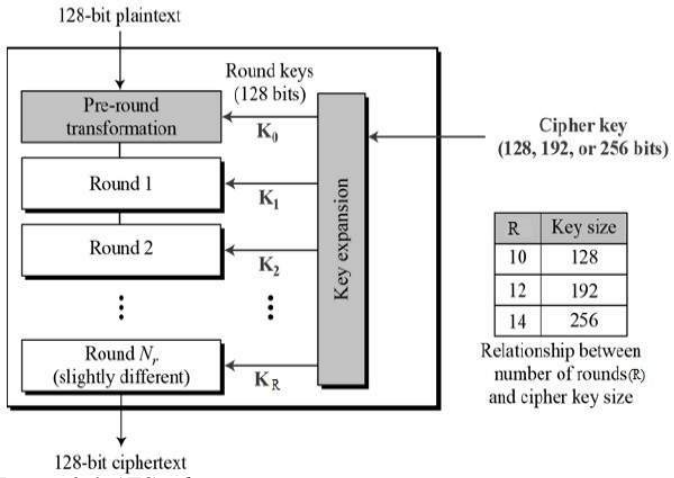

Figure 2.1 AES schematic structure
(source : tutorialspoint.com)

decrypt, 128 bit blocks, 192 bit blocks, etc. Each bit key has different rounds, for example, 128 bit key has 10 rounds of cipher process, 192 bit key has 12 rounds, and 256 bit key has 14 rounds. To protect classified information such as "Secret" level and "Top Secret" level requiring either 192 bit key or 256 bit key lengths. AES is an iterative cipher, based on substitution and permutation network. The interesting thing is that AES operations are on bytes rather on bits. As an example, 128 bit is treated as 16 bytes of plain block text and arranged to four columns and four rows and processed as a matrix. The first step of this algorithm is to define a number of transformations to be performed on data stored in an array. After that, first transformation is substitute the data between table, then shift data rows, and mixes column. Shift data rows will work on the second row, shifted one byte to the left, and on the third row is shifted two bytes to the left, and so on until the last row is shifted (n-1) bytes resulting a new matrix. Mix columns has its own special mathematic function which takes input from every bytes and resulting a whole new matrix, but this method is not used on the last round of the encryption. The last transformation is an exclusive or operation performed on each column. This AES system is still said to be six times faster than triple DES.

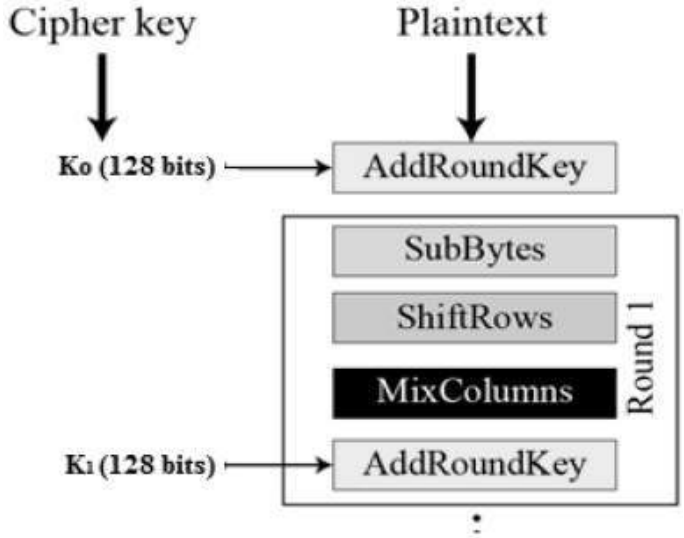

Figure 2.2 Encryption Process
(source : tutorialspoint.com)

Decryption process is similar to encryption process, but in the reverse order.

C. SubBytes Step

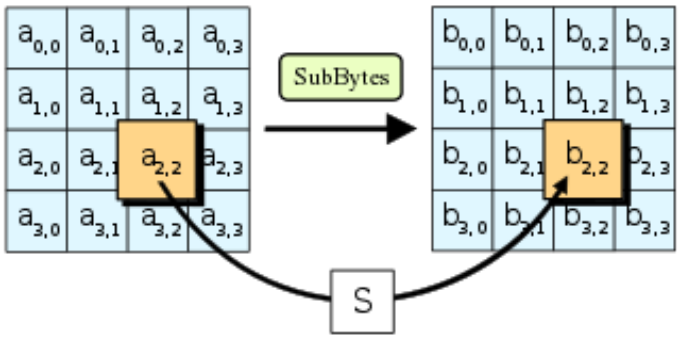

Figure 2.3 SubBytes step
(source : Matt Crypto)

In this step, each byte $(a_{i,j})$ is replaced with a SubByte $(b_{i,j})$ using 8 bit state. This makes the non-linear cipher. The SubBox is constructed by combining the inverse function with an invertible transformation. It is also chosen to avoid fixed points a derangement, and opposite fixed points. From figure 2.3 we know that each byte in the state is replaced with its entry in a fixed 8 bit state.

D. ShiftRows Step

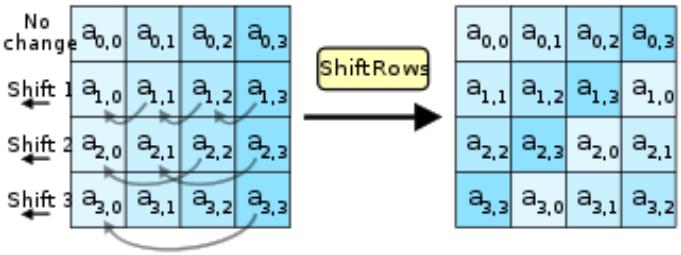

Figure 2.4 ShiftRows step
(source : Matt Crypto)

As mentioned in the name, this step only works on rows of the table. It shifts the bytes in each row by a certain offset. For AES, the first row is not changed while each byte of the second row is shifted one to the left. Each byte of the third row is shifted two to the left and so on as mentioned before. With this method, each column of the output table from this step is composed of bytes from each column of the input table. It is important to avoid the column being encrypted independently, as AES degenerates into four independence block ciphers. From figure 2.4 we could see how the shifting process is.

E. MixColumns Step

In this step, four bytes of each column of the table are combined. The combination process is using an invertible linear transformation, 4 bytes input and 4 bytes output. Each input byte affecting all four output bytes. Together with the second step, MixColumns can generate diffusion in the cipher.

In the operation, each column is transformed using a fixed matrix :

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2.j} \\ a_{3,j} \end{bmatrix} \quad ; \; 0 \le j \le 3$$

The matrix multiplication is composed of multiplication and addition of the entries. Those bits are treated as coefficients of polynomial of order x^7. Addition is simply XOR while multiplication is modulo irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. After processed and shifting, a special conditional XOR with B_{16} should be performed if the shifted value is overflow (larger than FF_{16}). This is a special case of the usual multiplication.

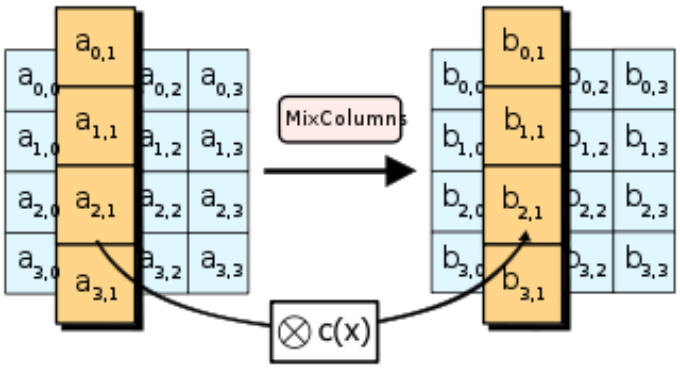

Figure 2.5 MixColumns step
(source : Matt Crypto)

From figure 2.5, we could see that each column of the table is multiplied with a fixed polynomial $c(x)$.

F. AddRoundKey step

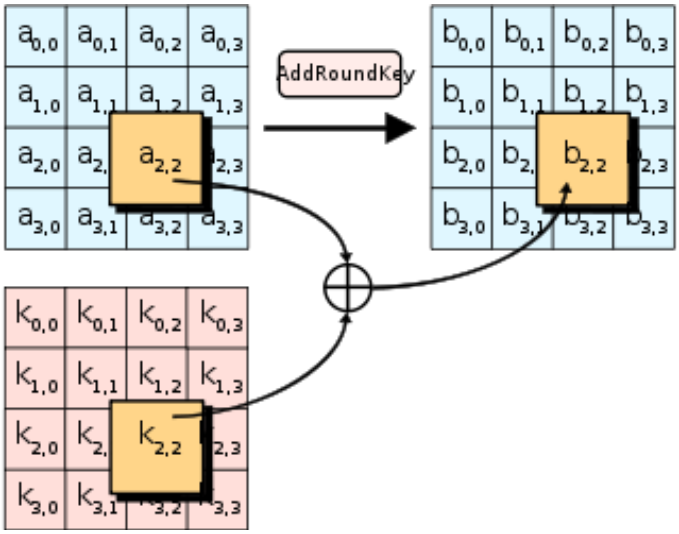

Figure 2.6 AddRoundKey Step
(source : Matt Crypto)

This method is combining the subkey with the state. For each round, a subkey will be created from the main key using Rijndael's key schedule. Each subkey is the same size as the state, added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

G. Cipher Optimization

Cipher execution is possible by combining SubBytes and ShiftRows steps with MixColumns step. It could be done by transforming them into a sequence of table. This method required four 256 entry 32 bit tables. A round can be performed with 16 table lookup operations, followed by four 32 bit XOR operations, or it can perform with a single 256 entry 32 bit table by circular rotation operations.

H. Security

As of May 2009, the only successful published attacks against AES were side-channel attacks on some specific implementations. With this, the National Security Agency stated that AES is secure enough for government non-classified data. For information, [3]by 2006, the best known attacks were on 7 rounds for 128 bit keys, 8 rounds for 192 bit keys, and 9 rounds for 256 bit keys.

I. Weakness

The disadvantage of this key is because it is a symmetric key, which means that you have to share the key you used to encrypt the information to the one who need to access it. Furthermore, if you do not have a secure way to share the key, other people might be able to access your encrypted information. As example, in 2005, a paper named "Cache timing attacks on AES" is published by a cryptographer called Daniel J. Bernstein. He demonstrated a timing attack on AES and capable of achieving a complete AES key recovery. Other research paper published in 2011 named "Biclique Cryptanalysis of the Full AES" by Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. They demonstrated a technique called biclique attack that could recover AES keys, and it is even faster than a brute-force attacks. However, this technique is very complex and not easy to use.

IV. RSA (RIVEST-SHAMIR-ADLEMAN)

A. History and General Definition

RSA is an early public-key cryptosystem. It has two keys, one made for public (encryption key) and the other is private (decryption key). RSA's asymmetry is based on practical factorization of two large prime number, known as the factoring problem. The name itself is made of the initials of the founder sure name, Ron Rivest, Adi Shamir, and Leonard Adleman. The original idea of this asymmetry public and private key itself came from Whitfield Diffie and Martin Hellman.

A RSA user can create then publish a public key based on two large prime numbers. The prime numbers must be kept secret. Anyone can use this public key to encrypt the information. RSA is a slow algorithm, and it makes this algorithm is not commonly used to encrypt user data.

B. Public Key Cryptography

This concept is relative new. Symmetric cryptography was well suited for organizations from governments up to big financial corporations.

These are some properties of public key, first, different key. Different keys are used for encryption and decryption. This is the one that makes this scheme different than the symmetric one.

Each person have a unique decryption key, known as his private key. User need to publish an encryption key as their public key.

Public key encryption scheme are divided into three types, RSA cryptosystem, ElGamal Cryptosystem, and Elliptic Curve Cryptography (ECC). RSA cryptosystem is one with the most employed cryptosystem used, invented by Rivest, Shamir, and Adleman.

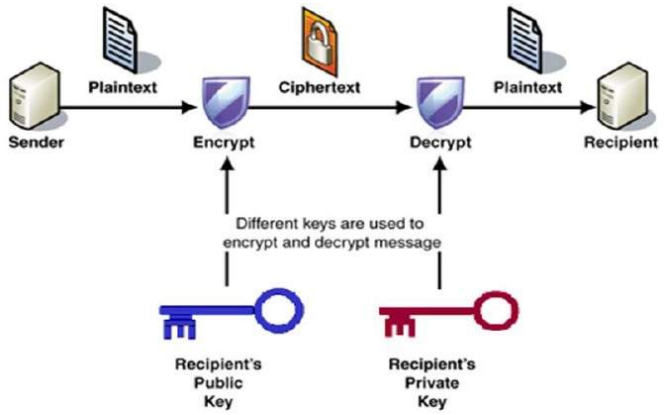

Figure 3.1 Encryption and Decryption process (Source : tutorialspoint.com)

If you want to use public and private key, first you need to generate the key by selecting two large primes, let's call it p and q. n should be p * q.

$$n = p * q$$

The recommendation for n is 512 bits minimum. Now we need to find a derived number. It must be greater than one, and less than (p – 1)(q – 1). Also, there should be no common factor for the derived number and (p – 1)(q – 1) except 1. This is called the two numbers are coprime. The pair of numbers (n, derived number) will be generated as the public key. Private key (p) is calculated from p, q, and the derived number, there will be a unique number (u). (u) is the inverse of derived number mod (p – 1)(q – 1). Below is an example on how to generate RSA key pair cited from tutorialspoint.com (with some modification) :

[1]An example of generating RSA Key pair is given below. (For understanding purpose, the two primes p & q taken here are small values. Practically, these values are very high).

C. Generating RSA Key Pair

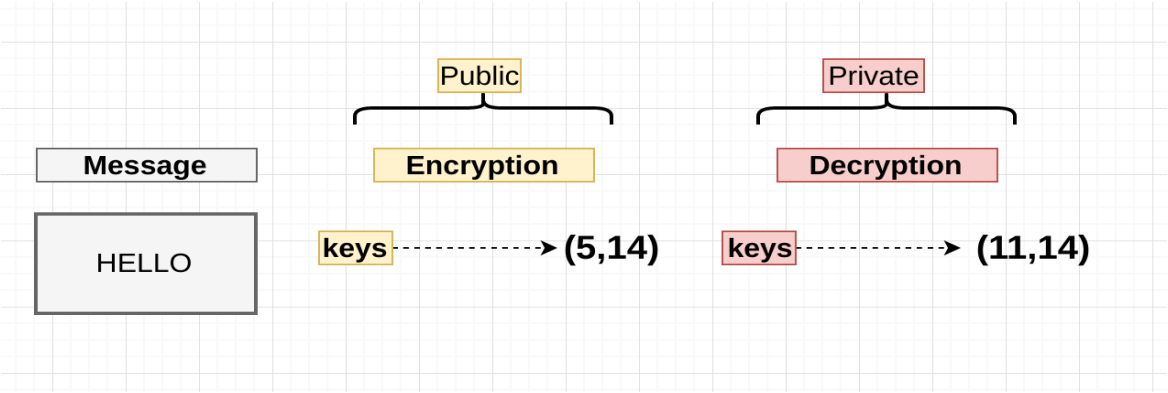

Figure 3.2 Essential RSA in general (Source : hackernoon.com)

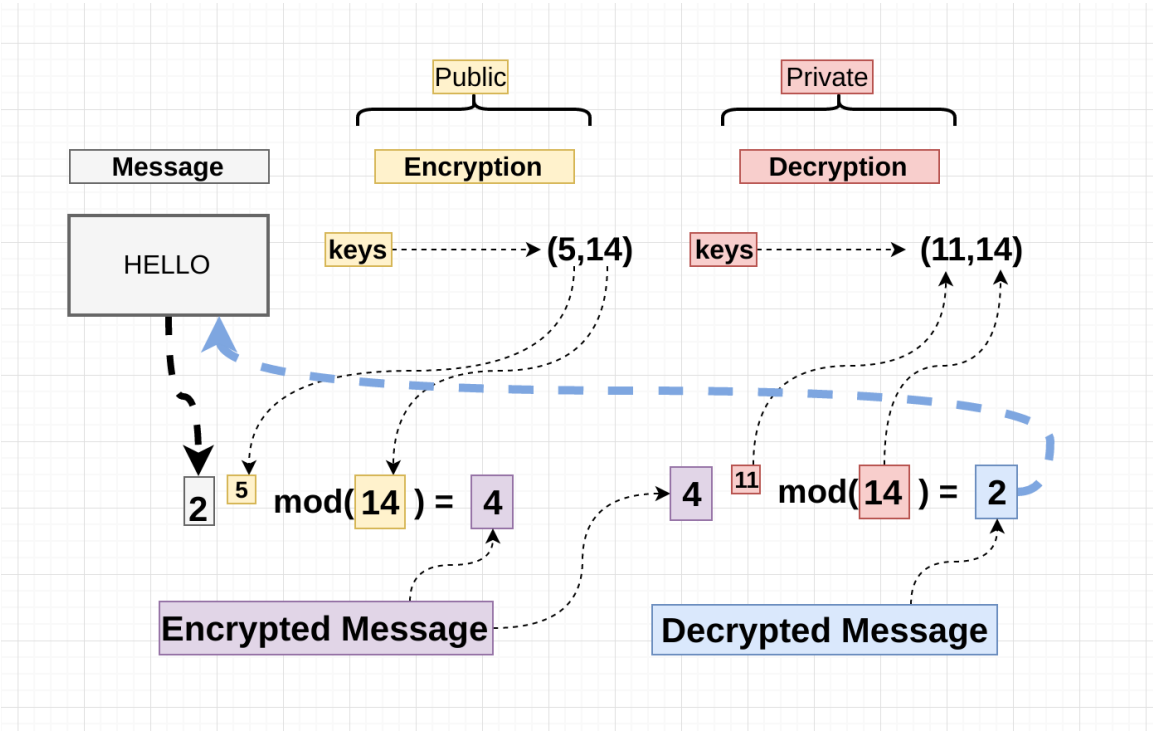

Figure 3.3 RSA in process (source : hackernoon.com)

Let p = 7 and q = 13. Thus, modulus n = p*q = 7 x 13 = 91.

Select derived number = 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.

The pair of numbers (n, derived number) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.

Input p = 7, q = 13, and derived number = 5 to the Extended Euclidean Algorithm. The output will be d = 29. Now, let's take a look at another example from hackernoon.com. Let's take p = 2 and q = 7.

$$p = 2$$
$$q = 7$$

So we will get n equal 14.

$$n = 2 * 7$$
$$n = 14$$

After that, we need to make a list starting from 1 to 14 and we need to remove the common factors.

1,2,3,4,5,6,7,8,9,10,11,12,13,14

By removing 14's common factor, we get a new list :

1,3,5,9,11,13

From the list, we know that the length of the list is six. There is an equation we can use if the list is not only contain 14 numbers but 512 bits of numbers

$$(p-1)(q-1) = L$$

From figure 3.3, we know that (5,14), 14 is the modulus. To generate the key, the rule is it has to be between 1 and L

2,3,4,5

Now we need to coprime L(6) and the modulus (14) and the result is 5. To find the decryptor D (11) from Figure 3.3 we can use this equation below (shown in figure).

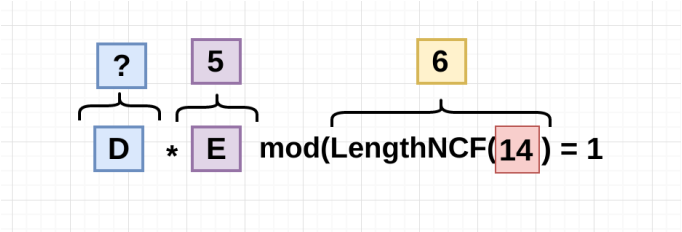

Figure 3.4 Decryption
(source : hackernoon.com)

D. RSA Encryption

If someone has a public key (n, derived number), the plain text should be presented as a series of numbers less than n. To encrypt, simply use this equation :

$$C = p^e \ mod \ n$$

E. Weakness

Low encryption exponent and small values make the cipher text to be easily decrypted. This encryption is also easily decrypted if someone share the file to some recipients in an encrypted way, making every receiver get the same exponent number, and with this people can easily use Chinese Remainder Theorem. RSA is a deterministic encryption algorithm, so if an attacker can launch a chosen plain text attack against the cryptosystem by encrypting like plain text under public key, and it is equal to the cipher text, they can decrypt the information as well.

F. RSA Problems

As mentioned above, RSA cryptosystem is based on large numbers, it means that there is possibility of key generation failure. Other than that, an attacker could decrypt the encrypted information by factoring the modulus n. The time taken to factor 128 bits and 256 bits n on a dual core Intel i7 – 4500U 1.80GHz only requiring less than 2 seconds for 128 bits n up to 35 minutes for 256 bits n.

The largest factored RSA number in 2010 was 768 bits long. The factorization itself took around fifteen hundred CPU years (about 2 years on many hundreds of computers). RSA key are typically 1024 to 4096 bits long, but some experts recommend to use at least 2048 bits long.

G. Faulty key generation

Large prime numbers are usually found by testing some random numbers of the right size with probabilistic. The two numbers (p and q) should not be too close. If $p - q < 2n^{1/4}$, solving them is trivial. Also, if p-1 and q-1 has only small prime factor, n can be factored quickly.

Private key has to be large enough, and should not between q and 2q or private exponent smaller than $(n^{1/4})/3$. If condition like described above is met, it would be easy to compute private exponent efficiently.

H. Importance of Strong Random Number Generation

We have to use a strong random number generator to generate p and q because if the prime numbers are not strong enough, it would be easy to decrypt them by using Euclid's algorithm. A weakness in integer factorization is found if

$$n = pq$$

and $n' = p'q'$ is another, then if by any chance that p = p', then a simple computation of

$$\gcd(n, n') = p$$

factors both n and n'. We also recommend you to use a deterministic function to generate q or p instead of choosing them personally.

Nadia Heninger was part of a group that did a similar experiment. They used an idea of Daniel J. Bernstein to compute the GCD of each RSA key n against the product of all the other keys n' they had found.

She write that [2]bad keys occurred almost entirely in embedded applications, including "firewalls, routers, VPN devices, remote server administration devices, printers, projectors, and VOIP phones" from over 30 manufacturers. She explains that [2]the one-shared-prime problem uncovered by the two groups results from situations where the pseudorandom number generator is poorly seeded initially and then reseeded between the generation of the first and second primes. Using seeds of sufficiently high entropy obtained from key stroke timings or electronic diode noise or atmospheric noise from a radio receiver tuned between stations should solve the problem.


Figure 3.5 Nadia Heninger
(source : cis.upenn.edu)

I. Timing Attacks

An American cryptographer and consultant, Paul Kocher described this theory in 1995. “If the attacker Eve knows Alice’s hardware in sufficient detail and is able to measure the decryption times for several known ciphertexts, she can deduce the decryption key (d) quickly”, he said. This method can also be applied to RSA signature scheme. A solution to solve this attack is to ensure that the decryption operation takes a constant amount of time for every cipher text. The negative side is this approach will significantly reduce performance. Most RSA implementations use an alternative technique known as cryptographic blinding.

J. Adaptive Chosen Ciphertext Attacks

First described in 1998, this method is using PKCS #1 padding scheme (a scheme to randomize and add structure to an RSA encrypted message, so it is possible to determine whether a decrypted message is valid or not). As a result of this method, cryptographers recommend the use of provably secure padding schemes such as Optimal Asymmetric Encryption Padding.

K. Side-Channel Analysis Attacks

This method uses BPA (Branch Prediction Analysis) from computer processors. Many processors use a branch predictor to determine whether a conditional branch in the instruction flow of a program is likely to be taken or not. These processors also implement SMT (simultaneous multithreading). This method use a spy process to find out this private key when processed with these processors.

L. Example of RSA Implementation

These are some cryptography libraries that provide support for RSA : Botan, Bouncy Castle, cryptlib, Crypto++, Libgcrypt, Nettle, OpenSSL, and wolfCrypt.

Botan is a BSD-licensed cryptographic library written in C++, providing a wide variety of cryptographic algorithm, formats, and protocols. Bouncy Castle is a collection of APIs used in cryptography, including APIs for Java and C# programming languages. Crypto++ is also known as CryptoPP, libcrypto++, and libcryptopp. It is a free C++ library of cryptographic algorithms, widely used in academia and student projects. Libgcrypt is a library developed for GnuPG.

OpenSSL is a software library for applications that secure communications over networks, widely used in internet servers. It contains free implementation of SSL and TLS protocols and written in C language.

V. CONCLUSION

AES is a symmetric key algorithm, same key is used for both encryption and decryption and RSA is an asymmetric key algorithm, so it uses two different key, public and private. AES is a 128 bits block cipher algorithm, meaning the data is divided to fixed length of data, processed in AES where each round is dependent on output of its predecessor. RSA is a stream cipher algorithm, meaning that the entire data is encrypted at once (will take more computational power), so it is slower, mainly used for exchanging little information only. AES could permutate it’s key using Rijndael finite field method, while RSA’s strength and weakness lies in the factoring large integers.

VII. ACKNOWLEDGMENT

Thanks to Dra. Harlili M.Sc for teaching basic of cryptography, Thanks to Dr. Ir. Rinaldi Munir, MT. for publishing the slides and some information needed to create this article. Thanks to many papers, writers, and websites which provides the extra information about cryptography, AES, and RSA.

REFERENCES

[1] https://www.tutorialspoint.com/cryptography/public_key_encryption.htm . (accessed on Dec 6, 2018)
[2] https://freedom-to-tinker.com/2012/02/15/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs/. (accessed on Dec 7, 2018)
[3] John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, Improved Cryptanalysis of Rijndael, Fast Software Encryption, 2000 pp213–230. (accessed on Dec 8, 2018)
[4] Kocher, Paul C (1996). "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". Annual International Cryptology Conference. (accessed on Dec 7, 2018)
[5] https://www.cybrary.it/course/cryptography/. (accessed on Dec 1, 2018)
[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/Makalah/Makalah16.pdf. (accessed on Dec 1, 2018)
[7] https://www.toptenreviews.com/software/articles/secure-encryption-methods/. (accessed on Dec 1, 2018)
[8] https://www.quora.com/How-does-RSA-and-AES-differ. (accessed on Dec 1, 2018)
[9] https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard. (accessed on Dec 1, 2018)
[10] https://hackernoon.com/how-does-rsa-work-f44918df914b. (accessed on Dec 1, 2018)
[11] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Algoritma-RSA-(2018).pdf. (accessed on Dec 1, 2018)
[12] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Kriptografi-Kunci-Publik-(2018).pdf. (accessed on Dec 1, 2018)
[13] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Advanced-Encryption-Standard-(AES)-(2018).pdf. (accessed on Dec 1, 2018)
[14] https://www.tutorialspoint.com/cryptography/origin_of_cryptography.htm. (accessed on Dec 6, 2018)

[15] https://searchsecurity.techtarget.com/answer/The-difference-between-AES-encryption-and-DES-encryption. (accessed on Dec 6, 2018)
[16] https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm. (accessed on Dec 6, 2018)
[17] https://courses.cs.washington.edu/courses/cse484/14au/reading/RSA.rivest-slides.pdf. (accessed on Dec 6, 2018)
[18] https://www.securityweek.com/understanding-public-key-cryptography-and-history-rsa. (accessed on Dec 6, 2018)
[19] http://www.math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Calderbank.pdf. (accessed on Dec 6, 2018). (accessed on Dec 7, 2018)
[20] https://www.di.ens.fr/~fouque/ens-rennes/coppersmith.pdf. (accessed on Dec 7, 2018)
[21] https://www.cis.upenn.edu/~nadiah/. (accessed on Dec 7, 2018)
[22] https://botan.randombit.net/index.html. (accessed on Dec 7, 2018)
[23] bouncycastle.org. (accessed on Dec 7, 2018)
[24] https://www.cryptopp.com. (accessed on Dec 7, 2018)
[25] https://gnupg.org/software/libgcrypt/index.html. (accessed on Dec 7, 2018)
[26] https://www.openssl.org. (accessed on Dec 7, 2018)
[27] http://cr.yp.to/papers/pqrsa-20170419.pdf. (accessed on Dec 7, 2018)
[28] https://www.schneier.com/academic/paperfiles/paper-twofish-final.pdf. (accessed on Dec 7, 2018)
[29] https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1. (accessed on Dec 7, 2018)
[30] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf. (accessed on Dec 7, 2018)
[31] https://link.springer.com/chapter/10.1007%2F3-540-36400-5_13?LI=true. (accessed on Dec 8, 2018)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.
Saduran penuh (1 paragraf) dari sumber-sumber tertentu yang menyatakan pernyataan-pernyataan pihak tertentu telah ditandai dan telah dicantumkan sumber jelasnya di dalam bab Referensi, sedangkan saduran pernyataan pihak tertentu yang terdiri atas 1 (satu) atau beberapa kalimat yang tidak memakan 1 (satu) paragraf penuh telah disiratkan sumbernya di dalam kalimat tersebut dan dicantumkan di dalam bab Referensi.

Bandung, 10 Desember 2017

Hansen 13517146