Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Performance Analysis of Countermeasures against Timing Attack in RSA Algorithm

Prof. Hemani Shah
Computer Engineering Dept., SVIT, Gujarat Technological University, India

Prof. Vibhavari Patel
Computer Engineering Dept., MBICT, Gujarat Technological University, India

Prof. Nisha Shah
Computer Engineering Dept., SVIT, Gujarat Technological University, India

Abstract— Public key cryptography is based on two keys, in which decryption key is private key. Among the different cryptanalytic attacks, timing attack is one of the possible attacks on RSA that determines bits of private key. This is done by determining time for each iteration in computing modular exponentiation. There are different mechanisms to counter such attack. Among them, blinding method and random delay are considered as proper mechanisms against timing attack. Kocher has also suggested masking method as a mechanism against timing attack. In this paper, we have shown modification of modular exponentiation algorithm as a proposed way to insert random delay. This paper compares blinding method and masking method with proposed method with respect to decryption time.

Keywords— Security, Cryptanalysis, RSA algorithm, Timing Attack

I. Introduction

In last few decades there is a revolution in the field of Telecommunication and Information Technology[1]. The World is becoming smaller day by day. Computations are becoming faster and computing devices became smaller. The 21st century which is only a decade old has given many inventions like robotics, nanotechnology, genetic engineering and many more that made our life better. In this era the world got universally electronic connectivity. But on the dark side viruses and hackers, eavesdropping and electronic fraud become active. And it makes the security essential for every connected system and communication. This has led to heightened awareness of the need to protect data and resources from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks.

In last several decades many changes affected security. The security of information in organization felt to be valuable was provided primarily by physical and administrative means. After introducing computers, the need arises to protect the information in shared system. Computer security is the collection of tools designed to protect data and to thwart hackers. With the introduction of distributed system, networks and communication facilities for carrying data the term network security and internet security is used[1]. To transmit the data securely private and public key cryptography is used as per the need. In our paper we focus on RSA public key cryptosystem and its attacks. Paper is organized as follows: Section II discusses about various cryptanalysis methods. Section III is about RSA algorithm and attacks on it. Section IV mainly focus on cryptanalytic attack - timing attack and different countermeasures against it. Section V explains proposed method to deal with timing attack. Section VI explains results of implementation of different methods and section VII concludes from the results of different methods.

II. Cryptanalysis

Most important automated tool for network and communications security is encryption. Before starting, we define some basic terms related to network security. The original message is known as the plaintext. Encoded message is called ciphertext. The process of converting plaintext to ciphertext is known as enciphering or encryption. Restoring the plaintext from ciphertext is deciphering or decryption. The area of studying the various schemes for enciphering is known as cryptography and such schemes are called cryptographic system or a cipher. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. The area of cryptography and cryptanalysis together are called cryptology[1]. The word Cryptanalysis came from the Greek word kryptos- "hidden" and analyein- "to loosen" or "to untie"[2]. Various types of cryptanalytic attacks, based on the amount of information known to the cryptanalyst are summarized in table 1. These attacks are performed on encrypted message only[1].

TABLE I: Cryptanalysis Attack

Type of attack	**Known to cryptanalyst**
Ciphertext only	Encryption algorithm Ciphertext to be decoded
Known	Encryption algorithm

plaintext	Ciphertext to be decoded Pairs of plaintext-ciphertext formed with the key
Chosen plaintext	Encryption algorithm Ciphertext to be decoded Chosen plaintext with corresponding ciphertext
Chosen ciphertext	Encryption algorithm Ciphertext to be decoded Chosen ciphertext with corresponding plaintext
Chosentext	Encryption algorithm Ciphertext to be decoded Chosen plaintext with corresponding ciphertext Chosen ciphertext with corresponding plaintext

Ciphertext only method is the most difficult problem when all that is available. It is the easiest to defend. Known plaintext is referred as a probable word attack. Example of chosen plaintext strategy is differential cryptanalysis. Chosen ciphertext and chosen text attacks are less commonly employed.

III. RSA ALGORITHM AND ATTACKS

All the cryptosystems before RSA were all based on the fact that both the decoding and encoding parties had to know the method of encryption and the key to decrypting the cipher. Key distribution is the main problem with all previous cryptosystems[6].

The problem was solved by Whitfield Diffie, working in collaboration with Martin Hellman. Diffie had discovered a revolutionary type of cipher with an asymmetric key. There are two distinct keys: the public and private keys. The private key is the decryption key and the public key is the encryption key. If sender A, wants to send a message to another person, B, all he has to do is use B’s public key to encrypt the message. As only B knows his private key no one other than B can decrypt the message. Diffie and Hellman could discover one-way function but could not discover such cipher. The solution was given by Rivest, Shamir and Adleman. And the derived solution is named as RSA from the last names of its three inventors: Rivest, Shamir and Adleman.

RSA is a worldwide de facto standard and can be used for encryption, digital signatures and key exchange. It has become the most popular asymmetric algorithm, beyond the wildest expectations of its inventors[6].

RSA is a block cipher in which the plaintext and ciphertext are integers between 0 to n-1. A typical size of n is 1024 bits. Steps of RSA algorithm are given below. Encryption and decryption are shown in table 2[1].

1. Select two large prime numbers p and q.
2. Calculate N=p*q; p and q are of the same size in terms of bits in binary representation and N is called RSA modulus.
3. Find Euler totient function, $\Phi(N)=(p-1)*(q-1)$ is the number of primes in the interval of [1..N-1]
4. Two integers e and d are chosen such that $e*d=1 \bmod \Phi(N)$; e and d are multiplicative inverse of each other. The private key consists of {d, N} and the public key consists of {e, N}.

TABLE II
ENCRYPTION AND DECRYPTION

Encryption	**Decryption**
Plaintext: $M<N$	Ciphertext: C
Ciphertext: $C=M^e$ (mod N)	Plaintext: $M=C^d$ (mod N)

Encryption and decryption of RSA algorithm involve raising an integer to an integer power; mod n. steps of an algorithm for computing a^b mod n are given below. Variable c is added for explanatory purpose. Final value of c is the value of exponent.

```
1.  c=0;d=1;
2.  for i=k to 0
3.  c=2*c;
4.  d=(d*d) mod n
5.  if bi=1 then c=c+1, d=(d*a) mod n
6.  return d
```

Main issue with key generation is to select large prime numbers. At present no useful technique is available to test for primality. Although many probabilistic tests are available, efficient and popular algorithm is Miller-Rabin algorithm given in [1].

Three possible approaches to attacking the RSA algorithm

- **Brute force attack:** These involve trying all private keys.
- **Mathematical attack:** There are several approaches that involve factoring the product of two prime numbers.
- **Timing attack:** These depend on the running time of decryption algorithm.

A. Brute-force attack

RSA uses large key space. Thus large number of bits in e and d makes the brute-force attack impractical. However, it will slow down the system.

B. Mathematical attack

There are three approaches to attack on RSA mathematically.

- Factor n into its two prime factors. This enables calculation of $\Phi(n)=(p-1)*(q-1)$, which in turn enables determination of $d=e^{-1}(\text{mod } \Phi(n))$.
- Determine $\Phi(n)$ directly that enables determination of $d=(\text{mod } \Phi(n))$.
- Determine d directly.

IV. Timing Attack And Its Countermeasures

This idea was first presented by Kocher, who laid the foundations of timing attack. One problem of the attack presented by Kocher is that the attacker needs a very detailed knowledge of the implementation of the system he is attacking, as he has to be able to compute the partial timings due to the known part of the key[3]. Timing attacks are a form of side channel attack, where an attacker gains information from the implementation of a cryptosystem rather than from any inherent weakness in the mathematical properties of the system. Timing attacks exploit the timing variations in cryptographic operations. Because of performance optimizations, computations performed by a cryptographic algorithm often take different amounts of time depending on the input and the value of the secret parameter[8]. If RSA private key operations can be timed plausibly accurately, in some cases statistical analysis can be done to recover the secret key used in the computations.

```
c = 0; f = 1
for i = k down to 0
      do c = 2 * c
          f = (f * f) mod n
       if b_i=1 then
            c = c + 1
            f = (f * a) mod n
return f
```

Note: The integer b is expressed as a Binary number $b_k b_{k-1} \ldots b_0$

Fig. 1 Algorithm for computing a^b mod n

Here the targeted computation is modular exponentiation operation which is mostly square and multiply algorithm mentioned in figure 4.1. Attack proceeds in this way. Suppose the target system uses a modular multiplication function that is very fast in almost all cases but in a few cases takes much more time than an entire average modular exponentiation. The attack proceeds bit-by-bit starting with the leftmost bit, b_k. Suppose that the first j bits are known (to obtain the entire exponent, start with $j = 0$ and repeat the attack until the entire exponent is known). For a given ciphertext, the attacker can complete the first j iterations of the **for** loop. The operation of further step depends on the unknown exponent bit. If the bit is set, $f = (f \times a) \bmod n$ will be executed. For a few values of a and d, the modular multiplication will be very slow, and the attacker knows which these values are. Therefore, if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1. If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0[1].

Different countermeasures against timing attack are:

A. Constant exponentiation time: Make sure that all exponentiations take the same amount of time before returning a result. This is a simple method to fix but degrades performance.

B. Random delay: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.

C. Blinding: Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

The private-key operation $M = C^d$ mod n is implemented as in [1] is as follows.

1. Generate a secret random number r between 0 and n - 1.
2. Compute $C = C(r^e)$ mod n, where e is the public exponent.
3. Compute $M' = (C)^d$ *mod n*
4. Compute $M= M'$ r^{-1} mod n. In this equation, r^{-1} is the multiplicative inverse of r mod n.

D. Masking: similar to blinding but use only private key with random number to mask modular exponentiation operation.

1. select random X.
2. compute M'= $(M*X)^d$ mod n
3. compute X'=$(X^{-1})^d$ mod n
4. multiply the two. i.e M=(M' * X') mod n

V. Proposed Method Using Random Delay

Random delay is a way through which modular exponentiation algorithm is modified to insert delay of random duration in each iteration. Enough noise has to be added, otherwise attackers could still succeed by collecting additional measurements to compensate for the random delays. In this paper, in the modular exponentiation algorithm mentioned in fig. 1 is modified. In this algorithm, if bit is 0, a random number is generated and is multiplied with a and modular multiplication is done. Below are the steps performed:

1. If bit is 0, generate random number r.
2. Perform modular multiplication of r with a i.e r=(r*a)mod n.

Every time new random number is generated and used in multiplication, so time duration of each iteration will vary. Even the bit is 0, computation time can be more than with bit 1. So, guessing of bit of exponent is difficult and timing attack cannot be applied feasibly.

VI. Results And Discussions

In this paper, three countermeasures are implemented and their decryption time are computed and compared with general RSA decryption.

TABLE III.
Decryption Time for Blinding, Masking and Proposed Random Delay Methods

RSA (ms)	**RSA with Blinding (ms)**	**RSA with masking (ms)**	**RSA with proposed method using random delay (ms)**
1024-bit key size			
60	90	275	97
65	85	290	91
60	90	288	94
65	87	297	89
67	87	287	93
2048-bit key size			
465	680	2140	616
470	634	2211	631
465	625	2228	623
462	622	2201	625
469	627	2228	630
3072-bit key size			
1541	2107	7140	2112
1520	2100	6970	2027
1490	2005	7040	2116
1515	2000	7185	2028
1505	2025	7085	2031

In implementation of all these methods, public key of RSA is taken as fixed value i.e 65537 and based on that private key is generated. So, in each iterations, private key generated is sometimes bigger. First column in table 3 represents decryption time in milliseconds for general RSA algorithm. Second column represents time duration for blinding method and third column represents masking method. Fourth column represents decryption time for proposed method. Results are taken for three different sizes of n.

In the implementation, for most cases private key can be larger than public key and so, results of masking column is much more than other columns. Blinding method and proposed method with random delay, both gives same results.

VII. CONCLUSION

In this paper, three different methods to prevent timing attack are implemented and compared. Blinding and proposed method with random delay gives similar results and also better than masking method. So, both the methods can be preferred as a countermeasure of timing attack on RSA. Some performance penalty is there and that is approximately 30%. But time computation and bit determination in modular exponentiation algorithm will be difficult with these methods and so, some penalty in performance can be accepted.

REFERENCES

[1] William Stallings "Cryptography and Network Security – Principles and Practice", 5th ed. Prentice Hall, pp. 277-291.
[2] http://en.wikipedia.org/wiki/Cryptanalysis
[3] J F. Dhem, F. Koeune, P. A. Leroux, P. Mestre, J. J. Quisquater, J. L. Willems, "A Practical Implementation of the timing attack", UCL Crypto group Technical Report Series, CG-1998/1.
[4] Chris Davis, "Timing cryptanalysis of Public key cryptosystems", Fall 2001, ECE 646, George Mason University, USA.
[5] Ronan Killeen, "Possible attacks on RSA", (RSA: Hacking and Cracking)
[6] Michael Calderbank, "The RSA cryptosystem: History, Algorithms, Primes", Aug. 20, 2007.
[7] Xiao Lei Cui, "Attacks on the RSA cryptosystem", A Report.
[8] Chia Long Wu, Der Chyuan Lou, Te-Jen Chang, "An efficient Montgomery exponentiation algorithm for cryptographic applications, INFORMATICA, 2005, Vol. 16, No. 3, 449-468.
[9] Lawrence Brown, "Techniques for implementing the RSA public key cryptosystem", A paper.
[10] http://members.tripod.com/irish_ronan/rsa/attacks.html