Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 10; October, 2014 pp. 48-63

http://www.krishisanskriti.org/jbaer.html

Enhancing the Securing RSA Algorithm from Attack

Swati Srivastava[1], Meenu[2]

[1]M.Tech Student, Department of CSE, Madan Mohan Malaviya University of Technology, Gorakhpur, U.P.
[2]Department of CSE, Madan MohanMalaviya University of Technology, Gorakhpur, U.P.

Abstract: **The RSA public key and signature scheme is often used in modern communications technologies; it is one of the firstly defined public key cryptosystem that enable secure communicating over public unsecure communication channels. In praxis many protocols and security standards use the RSA, thus the security of the RSA is critical because any weaknesses in the RSA crypto system may lead the whole system to become vulnerable against attacks. This paper introduce a security enhancement on the RSA cryptosystem, it suggests the use of randomized parameters in the encryption process to make RSA many attacks described in literature, this enhancement will make the RSA semantically secure, this means that that an attacker cannot distinguish two encryptions from each other even if the attacker knows (or has chosen) the corresponding plaintexts A comparison introduced in this paper between the basic RSA and the modified RSA version shows that the enhancement can easily be implemented. This paper also briefly discuss some other attacks on the RSA and the suitable choice of RSA parameter to avoid attacks, also an important issue for the RSA implementation is how to speed up the RSA encryption and decryption process.**

Keywords: **RSA cryptosystem, RSA signature, RSA Problem, Public Key Cryptosystems, Private Key Cryptography, Crypto Analysis, Finite Fields, Quantum Computers.**

1. INTRODUCTION

The computer and communication technology's today are very important parts for a strong economy, thus it is important to have suitable security standards systems and technologies to meet that security needs. Many security systems and protocols have been developed that are based on standards, such standards comes mostly from well known standard organizations (e.g. Internet Architecture Board (IAB), Internet Engineering Task Force (IETF), etc.) that specify a huge set of security protocols, algorithms and applications which provide security services and meets the needs for data privacy and secure communication. A powerful tool for protection is the use of Cryptography. Cryptography underlies many of the security mechanisms and builds the science of data encryption and decryption. Cryptography [1] enables us to securely store sensitive data or transmit across insecure networks such that it cannot be read by anyone except the intended recipient. By using a powerful tool such as encryption we gain privacy, authenticity, integrity, and limited access to data. In Cryptography we differentiate between private key cryptographic systems (also known as conventional cryptography systems) and public key cryptographic systems. Private Key Cryptography, also known as secret-key or symmetric-key encryption, has an old history, and is based on using one shared secret key for encryption and decryption. The development of fast computers and communication technologies did allow us to define many modern private key cryptographic systems, e.g. in 1960's Feistel cipher [2], Data Encryption Standard (DES), Triple Data Encryption standards (3DES), Advanced Encryption Standard (AES), The International Data Encryption Algorithm (IDEA), Blowfish, RC5, CAST, etc. The problem with private key cryptography was the key management, a system of n communicating parties would require to manage $((n-1)*n)/2$ this means that to allow 1000 users to communicate securely, the system must manage 499500 different shared secret key, thus it is not scalable for a large set of users. A new concept in cryptography was introduced in 1976 by Diffie and Hellman [2] this new concept was called public-key cryptography and is based on using two keys (Public and Private key). The use of public key cryptography solved many weaknesses and problems in private key cryptography, many public key cryptographic systems were specified (e.g. RSA [3], ElGamal [4], Diffie-Hellman key exchange [2], elliptic curves [5], etc.). The security of such Public key cryptosystems is often based on apparent difficulties of some mathematical number theory problems (also called "one way functions") like the discrete logarithm problem over finite fields, the discrete logarithm problem on elliptic curves, the integer factorization problem or the Diffie-Hellman Problem, etc. [1].

One of the firstly defined and often used public key cryptosystems is the RSA. The RSA cryptosystem is known as the ―de-facto‖ standard for Public-key encryption and signature worldwide and it has been patented in the U.S. and Canada. Several standards organizations have written standards that use of the RSA cryptosystem for encryption, and digital signatures [6], in praxis RSA is used in many internet security protocol and applications e.g. securing emails, securing e-payment and in related certification solutions.

The RSA cryptosystem was named after his inventors R. Rivest, A. Shamir, and L. Adleman and is one of the mostly used public-key cryptosystem, the patent (4, 405, 829) was registered in the 14 of December 1977 (and did expired on September 21, 2000), it was assigned to the Massachusetts Institute of Technology, and it covers the RSA public-key encryption and the digital signature method.

Many well known standard organizations specified security standards which define the implementation and the use of RSA in security systems [7] [8].

Due to the wide use of the RSA cryptosystem, is it critical to ensure a high level of security for the RSA, in this paper I introduce a new enhancement to the security of the RSA cryptosystem, this is achieved by using randomized parameter, this will make the encrypted message more difficult for an adversary to break, thus making the RSA more secure.

2. PROBLEM FORMULATION

The security of the RSA cryptosystem is based on the intractability of the RSA problem. This means that if in the future the RSA problem is generally solved then the RSA cryptosystem will no longer be secure.

The following algorithms describe the RSA key generation, and the RSA cryptosystem (basic version)

Algorithm 2.1: Key generation for the RSA public-key encryption

Each user A creates an RSA public key and the corresponding private key.
User A should do the following:

1. Generate two large random (and distinct) primes *pA*and *qA*, each roughly the same size.
2. Compute $n= p*q$ and $\phi(n) = (p-1)(q-1)$.
3. Select a random integer e, $1 < e < \phi(n)$, such that $gcd(e, \phi(n))=1$.
4. Use the Euclidean algorithm to compute the unique integer d, $1 < d < \phi(n)$, such that
   $e*d \equiv 1 \pmod{\phi(n)}$.
5. User A public key *is (n, e)* and A's private key is *d*

Definition The integer's *e* and *d* in RSA key generation are called the encryption exponent and the decryption exponent, respectively, while *n* is called the modulus.

Algorithm 2.2: The RSA public-key encryption and decryption (Basic version)
User B encrypts a message *m* for user A, which A decrypts.

1. **Encryption**. User B should do the following:
   (a) Obtain user A authentic public key *(n, e).*
   (b) Represent the message as an integer *m* in the interval *[0, n-1]*
   (c) Compute *c = m^e mod n*
   (d) Send the encrypted text message *c* to user A.
2. **Decryption**. To recover plaintext *m* from c, user A should do the following:
   (a) Use the private key *d* to recover *m = (m^e)^d mod n*

The original RSA encryption, decryption does not contain any randomized parameter making the RSA cryptosystem deterministic, which means that an attacker can distinguish between two encryptions, based on this many of the attacks listed below can be performed on the RSA basic version.

3. **Enhancing the security of the RSA cryptosystem**

The key generation remain unchanged as in the original RSA, see above. The following algorithms describe the enhanced RSA cryptosystem.

Algorithm 3.1: The enhanced RSA public-key encryption and decryption (Modified version)
User B encrypts a message *m* for user A, which A decrypts.

1. Encryption. User B should do the following:
(a) Obtain user A authentic public key *(n, e).*
(b) Represent the message as an integer *m* in the interval *[0, n-1]*
(c) Select a random integer k, $1 < k < n$, such that $gcd(k, n))=1$
(d) Compute *c1 = k^e mod n*
(e) Compute *c2 = m^e k mod n*
(f) Send the encrypted text message *(c1, c2)* to user A

2. Decryption. To recover plaintext *m* from *c2*, user A should do the following:
(a) Use own private key *d* and compute: *c1*
d = k mod n
(b) Use the Euclidean algorithm and calculate the unique integer s, $1< s < n$, such that $s*k \equiv 1 \pmod{n}$.
(c) Compute *c2s = (m e k) s = (me) k s = m e mod n*
(d) Recover *m*, use the private key *d* and compute: *(m^e)*
d = m mod n

The following example illustrates the use of modified RSA cryptosystem.
Example: (RSA Encryption/Decryption)
Key Generation: Assume $p= 2350$, $q= 2551$, $n = p*q = 6012707$

1. **Encryption**. User B should do the following:
   (a) User A authentic public key *e = 3674911*
   (b) Message *m = 31*
   (c) Random *k = 525*
   (d) Compute: *5253674911 = 20639 mod 6012707*
   (e) Compute: *313674911 525 = 2314247 mod 6012707*
   (f) Send (20639, 2314247) to user A
2. **Decryption**. To recover plaintext *m* from *c*, user A should do the following:
   (a) User A private key *d* = *422191*, compute: *20639422191 = 525 mod 6012707*
   (b) Extended *GCD(525, 6012707)* *s = 3516002*

(c) Compute: *2314247 * 3516002 = 2913413 mod 6012707*
(d) Recover m: *2913413422191 = 31 mod 6012707*

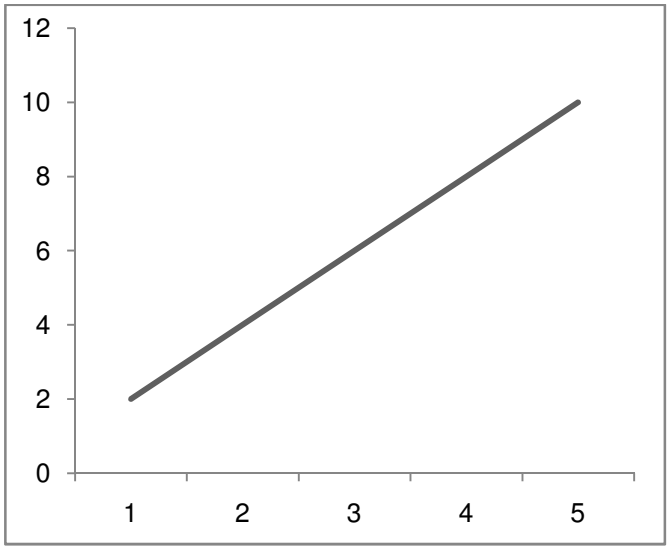
The RSA encryption/decryption is much slower than commonly used symmetric-key encryption algorithms such as the well know algorithm DES and this is the reason why in practice RSA encryption is commonly used to encrypt symmetrical keys or to encrypt small amount of data, there are many software solutions or hardware implementations to speeding up the RSA encryption/ decryption process. For more information about speeding up RSA software implementations see [6].

Because the basic version of the RSA cryptosystem has no randomization component an attacker can successfully launch many kinds of attacks, now we discuss some of these attacks.
1. Known plain-text attack; a known-plaintext attack is one where the adversary has a quantity of plaintext and corresponding cipher-text [6].

Given such a sorted set *S = {{p1, c1}, {p2, c2}, ..., {pr, cr}}* (where *pi* □*P* plaintext set, *ci* □*C* ciphertext set, *r <ø (n)* is the order of *Zn*)* anadversary can determine the plaintext *px*if the corresponding *cx* is in *S*.

The following example shows the relationship between the length of *S* and the probability of finding a searched element*px*in *S*.

Example: assume *p = 10^8*, the function *f = x/p*, where *x x ∈ Zn**

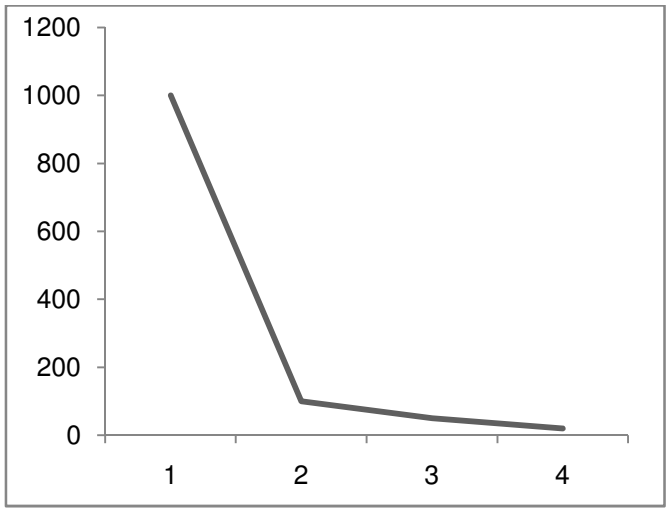


The modified version of the RSA described above use *k* as randomizing parameter; this can protect the encrypted text against known plain text attacks, because even if we know *px*, in the equation:

px = kxmx
*kx*and *mx* will still remain unknown.

Next we investigate the probability *p* that two encrypted blocks of the same message *m* have equal random integer *ki,* , this probability is equal:
Pk1=k2(m) = 2/n,

Where *n* is the RSA modulus, the following sketch show that for approx. 1024 bit *n*, the probability is very close to zero.



2. **Chosen Cipher Text Attack:** RSA has the property that the product of two cipher texts is equal to the encryption of the product of the respective plaintexts. That is *m1e m2e = (m1m2) e mod n.* Because of this multiplicative propertya chosen-cipher text attack is possible. the algorithm can be described as follows: Let *c = me mod m*, the attacker chooses a random number r where *1 < r < n*, such that *gcd(r, n))=1* then compute *x = r^e mod n, c'= x * c mod n, z = r-1 mod n,* and send *c'*to victim. The victim compute *m'= (c')d mod n*, then send *m'*to the attacker, the attacker recovers original message *m = z * m'mod n.* This attack is based on the theoretical assumption that the attacker has access to a decryption device that returns the complete decryption for a chosen cipher text. The run time estimation is *O(m' * n3),* which is polynomial thus impractical.

3. **Exhaustive Search Attack**: It involves systematically checking all possible keys until the correct key is found. In the worst case, this would involve traversing the entire search space, thus *O(n)* elements to check. To avoid such attack is it important for RSA security that the size of the modulus *n* which depends on the size of the prime's *p* and *q*, where *p* and *q* should be so selected that factoring is computationally infeasible.

4. **Johan Håstad and Don Coppersmith Attack**: If the same clear text message is sent to more recipients in an encrypted way, and the receivers share the same exponent e, but different p, q, and n, then it is easy to decrypt the original clear text message via the Chinese remainder theorem [6]. Johan Håstad [9] described this attack and Don Coppersmith [10] improved it.

5. **Common Modulus Attack:** If also same message *m* is encrypted twice using the same modulus *n*, then one can recover the message *m* as follows: Let *c1=me1 mod n*, and *c2=me2 mod n* be the cipher texts corresponding to message *m*, where gcd(e1, e2)=1, then attacker recovers original message m1=c1a * c2b mod n for e1*a+e2*b=1. Using the extended great common divisor (GCD) one can determine *a* and *b* then calculate *m* without knowing private key *d*, this is known in the literature as the Common Modulus Attack that requires *O((log k)2),* where *k* is maximum size of *a* or *b*.

6. **Timing Attack:** One attack on the RSA implementation is the Timing Attack; Kocher [11] demonstrated that an attack can determine a private key by keeping track on how long a computer takes to decrypt a message.

7. **Small Public/Private exponent e/d Attack**: To reduce decryption time, one may wish to use a small value of private exponent *d* or reduce the encryption time using a small public exponent *e*, but this can result in a total break of the RSA cryptosystem as Coppersmith [12] and M.Wiener [13] showed.

8. **Adaptive chosen cipher text attacks:** In 1998, Daniel Bleichenbacher [14] described the first practical adaptive chosen ciphertext attack, against RSA-encrypted messages using the PKCS #1 v1 [15] padding scheme (a padding scheme randomizes and adds structure to an RSA-encrypted message, so it is possible to determine whether a decrypted message is valid.) Bleichenbacher was able to mount a practical attack against RSA implementations of the Secure Socket Layer protocol (SSL) [16], and to recover session keys, here it is important to mention that such protocol is still often used in internet to secure emails and e-payment via internet. As a result of this work, cryptographers now recommend the use of provably secure padding schemes such as Optimal Asymmetric Encryption Padding, and RSA Laboratories has released new versions of PKCS #1 that are not vulnerable to these attacks.

9. **Attacks on the factorization problem:** Some powerful attacks on the RSA cryptosystem are the attacks on the factorization problem; the factoring algorithms to solve the factorization problem come in two parts: special purpose and general purpose algorithms. The efficiency of special purpose depends on the unknown factors, whereas the efficiency of the latter depends on the number to be factored. Special purpose algorithms are best for factoring numbers with small factors, but the numbers used for the modulus in the RSA do not have any small factors. Therefore, general purpose factoring algorithms are the more important ones in the context of cryptographic systems and their security. A major requirement to avoid factorization attacks on the RSA cryptosystem is that *p* and *q* should be about the same bits length and sufficiently large. For a moderate security level *p* and *q* should be at least 1024 bits length, this will result in a 2048 bit length for modulus *n*. furthermore *p* and *q* should be random prime number and not of some special case binary bit structure. The following table summarizes the running time for some of the well known integer factoring algorithms where *p* denotes the smallest prime factor of *n*, and *e=2.718* is the Euler's constant.

Table: Factorization algorithms Algorithm Runtime estimation

1. Pollard's Rho*O(p)*
2. Pollard's*p-1 O(p*)* where *p** is the largest prime factor of p-1.
3. William's*p+1* O(*p**) where *p** is the largest prime factor of *p+1*.
4. Elliptic Curve Method *O(e(1+o(1)) (2ln p lnln p)1/2)*
5. Quadratic Sieve *O(e(1+o(1)) (ln N lnln N)1/2)*
6. Number Filed Sieve *O(e(1.92+o(1))(ln N)1/3(lnln N)2/3)*

In 2010, the largest number factored by a general-purpose factoring algorithm was 768 bits long [23] using distributed implementation thus some experts believe that 1024-bit keys may become breakable in the near future so it is currently recommended to use 2048 for midterm security and a 4096-bit keys for long term security. Now, the described RSA security enhancement in this paper can protect us against the following attacks:

Table: RSA enhancement is immune against the following attacks

Attack	**Justification**
1. Known plain-text attack	Is not possible as described above
2. Small public e	Is not possible due to the exponent use of random integer k
3. Johan Hasted and Don Coppersmith	Is not possible because every msg. have attack unique *k*i
4. Common Modulus Attack	Is not possible because every msghave unique *k*i
5. Timing Attack	Using *k* in encrypt. & decrpt. process will make it difficult

		To distinguish between time for*k* and the time for public e or private key e
6.	Adaptive chosen cipher textattacks	integer *k* instead of secure padding.

This will make the RSA cryptosystem more secure compared with the basic version of the RSA cryptosystem. The enhancement makes the RSA cryptosystem semantically secure this means that an attacker cannot distinguish two encryptions from each other even if the attacker knows (or has chosen) the corresponding plaintexts For more detailed information about attacks on RSA see [6] [24].

3. CONCLUSIONS

In this paper I briefly discussed enhancing the security of the RSA public-key cryptosystem, this enhancement use randomized parameter to change every encrypted message block such that even if the same message is sent more than once the encrypted message block will look different. The major advantage gained in the security enhancement described above is making RSA system immune against many well known attacks on basic RSA cryptosystem, thus making the RSA encryption more secure, this is essential because RSA is implemented in many security standards and protocols and a weak RSA may result in a whole compromised system. One solution that is used in praxis to overcome this problem is the use of padding bits in the encryption process, but this may not always work well if we have a long message where many blocks are without padding or if the adversary knows the padding bits. Although the security enhancement make RSA more secure nevertheless it should be noted that the RSA modulus *n* bit length should be at least *2048* to ensure a moderate security and to avoid powerful attacks on the discrete logarithm and factorization problem. This security consideration and other mentioned in literature should be used to define an improved version of the RSA.

Many public key cryptographic system such as the RSA build their security on the intractability of the factorization and the discrete logarithm problem, if such problem are solved in future due to new mathematic insights or new computer technologies [25] this may result in huge set of compromised public key crypto and security systems.

REFERENCES

[1] D. Kahn, The Code breakers: The comprehensive History of Secret Communication from Ancient to the Internet, Published 1967

[2] W. Diffie and M. Hellman, ―New directions in cryptography‖, IEEE Transactions on Information Theory, 22 (1976) 644-654.

[3] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Commun.of the ACM, 21:120-126, 1978.

[4] T. ElGamal, ―A public-key cryptosystem and a signature scheme based on discrete logarithms‖, IEEE Transactions on Information Theory, volume 31, pages 469-472, 1985.

[5] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48:203-209, 1987

[6] A. Menezes, P. van Oorscot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, ISBN: 0-8493-8523-7, 1999

[7] ANSI press specified ANSI X9.44 is a draft standard for key transport based on the RSA algorithm

[8] IEEE press, 2000. Specified the IEEE P1363 working group is developing standards for public-key cryptography based on RSA and Diffie-Hellman algorithm families and on elliptic curve systems

[9] J. Håstad and M. Näslund, The Security of all RSA and Discrete Log Bits, Journal of the ACM, Vol 51:2, 2004, pp 187-230.

[10] Don Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities", Journal of Cryptology, v. 10, n. 4, Dec. 1997

[11] Kocher, P., 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. Advances in Cryptology, 1109: 104-113.

[12] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, Michael K. Reiter: Low-Exponent RSA with Related Messages. EUROCRYPT 1996: 1-9

[13] M.Wiener. Cryptanalysis of short RSA secret exponents.IEEE Transactions on Information Theory, 36:553- 558, 1990.

[14] Daniel Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. Advances in Cryptology — CRYPTO '98 Lecture Notes in Computer Science, 1998, Volume 1462.

[15] PKCS #1: RSA Cryptography Standard, website: http://www.rsa.com/rsalabs/node.asp?id=2125

[16] The Transport Layer Security (TLS) Protocol, version 1.2, website: http://tools.ietf.org/html/rfc5246

[17] Pollard, J. M. (1975), "A Monte Carlo method for factorization", BIT Numerical Mathematics 15 (3): 331–334

[18] Pollard, J. M. (1974), "Theorems of Factorization and Primality Testing", Proceedings of the Cambridge Philosophical Society 76 (3): 521–528

[19] Williams, H. C. (1982), "A p+1 method of factoring", Mathematics of Computation 39 (159): 225–234,

[20] B. Dixon, A.K. Lenstra, Massively parallel elliptic curve factoring, 183–193.

[21] C. Pomerance, The quadratic sieve factoring algorithm, 169–182.

[22] J. Buchmann, J. Loho, J. Zayer, An implementation of the general number field sieve, 159–165

[23] RSA Laboratories, the RSA Factoring Challenge http://www.rsa.com/rsalabs/node.asp?id=2092

[24] Boneh, D., 1999. Twenty years of attacks on the RSA Cryptosystem. Notices of the AMS, 46: 2003-2013.