ISSN No. 0976-5697

Volume 6, No. 6, July - August 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Novel Approach to Factorize the RSA Public Key Encryption

Ms. Mrinalini shringirishi[1]
Assistant professor Bundelkhand Institute of Engineering Technology & P.h.d Scholor M.tech (C.S.) Jhansi (U.P.)

Dr. Manoj Gupta[2]
Director Bundelkhand Institute of Engineering Technology Jhansi (U.P.)

Dr. Anil Kumar Solanki[3]
Head of Deptt. Information Technology
Bundelkhand Inst. of Engineering Technology Jhansi (U.P.)

Dr. Yashpal singh[4]
Head of Deptt. Computer science & Technology
Bundelkhand Inst. of Engineering Technology Jhansi (U.P.)

Ms. Pritee Gupta[5]
Assist. Professor Department of computer science & engineering
I.T.S Engineering College Noida

Abstract The security of public key encryption such as RSA algorithm relied on the integer factoring problem of N. The security of RSA algorithm based on positive integer N, which is the product of two large prime numbers, the factorization of N is very complex. In this paper a factorization method is proposed, which is used to obtain the factor of positive integer N. Prime numbers play a very important role in the complexity and security of the public key crypto system. RSA Algorithm is one type of public key algorithm and its security depends on the complexity of factoring value of N. Any encryption algorithm depends on the length of the key and the computational effort required breaking the key. This paper introduces an efficient algorithm to attack on the RSA algorithm. Obtaining the private key of the RSA algorithm is the target of the suggested algorithm by factoring the modulus based on the public key (N, e) of the RSA algorithm. The suggested algorithm is very fast due to its treatments for the factorizing problem. It will limit the search for the two prime no's p & q values. The suggested algorithm is more efficient than most existed algorithms of attack since it will interrupt the search process and takes less running time.

Keywords-Public Key Cryptography, RSA Algorithm, Factorization Problem, Attacks, Public Key Cryptography.

I. INTRODUCTION

Public key cryptography is one of the applications that are valuable in sending information via insecure channel. RSA algorithm is a public key encryption algorithm. RSA has become most popular cryptosystem in the world because of its simplicity. According to number theory, it is easy to finds two big prime number, but the factorization of the product of two big prime numbers is very difficult task. The difficulty of computing the roots N, where N is the product of two large unknown primes, it is widely believed to be secure for large enough N. Since RSA can also be broken by factoring N, the security of RSA is often based on the integer factorization problem [1]. The integer factorization problem is a well-known topic of research within both academia and industry. It consists of finding the prime factors for any given large modulus. Currently, the best factoring algorithm is the general number field sieve or GNFS for short. On December 12, 2009 a small group of scientists used the previously mentioned approach to factor a RSA-768 bit modulus, that is, a composite number with 232 decimal digits. Their achievement required more than two years of collaborative work and used many hundreds of computing machines. Hence, factoring large primes is a laborious and complex task [2]. A method for factoring algorithm (specially designed) for semi primes based on new mathematical ideas. Since this method is relatively simple and scalable, it can be suitable for parallel processing [2]. But the main condition of this algorithm is that to break RSA modulus firstly we should have public key and modulus N. On the basis of this public key (e, N) proposed algorithm disclose the private key [3]. An attacker has access to the public key e and N and the attacker wants the private key d. To get d, N needs to be factored (which will yield p and q, which can then be used to calculate d).

Factoring n is the best known attack against RSA to date. (Attacking RSA by trying to deduce (p-1) (q-1) is no easier than factoring N, and executing an exhaustive search for values of d is harder than factoring N.) Some of the algorithms used for factoring are as follows Trial division oldest and least efficient Exponential running time. Try all the prime numbers less than sqrt (N). Quadratic Sieve (QS): The fastest algorithm for numbers smaller than 110 digits. Multiple Polynomial Quadratic Sieve (MPQS): Faster version of QS. Double Large Prime Variation of the MPQS Faster still. Number Field Sieve (NFS) Currently the fastest algorithm known for numbers larger than 110 digits. These algorithms represent the state of the art in warfare against large composite numbers against RSA. We can identify many approaches to attacking RSA mathematically, factor N into two prime factors.

II. RSA ALGORITHM

In 1978, RSA developed a public key cryptosystem that is based on the difficulty of integer factoring. The RSA public key encryption scheme is the first example of a provable secure public key encryption scheme against chosen message chosen attacks [5].

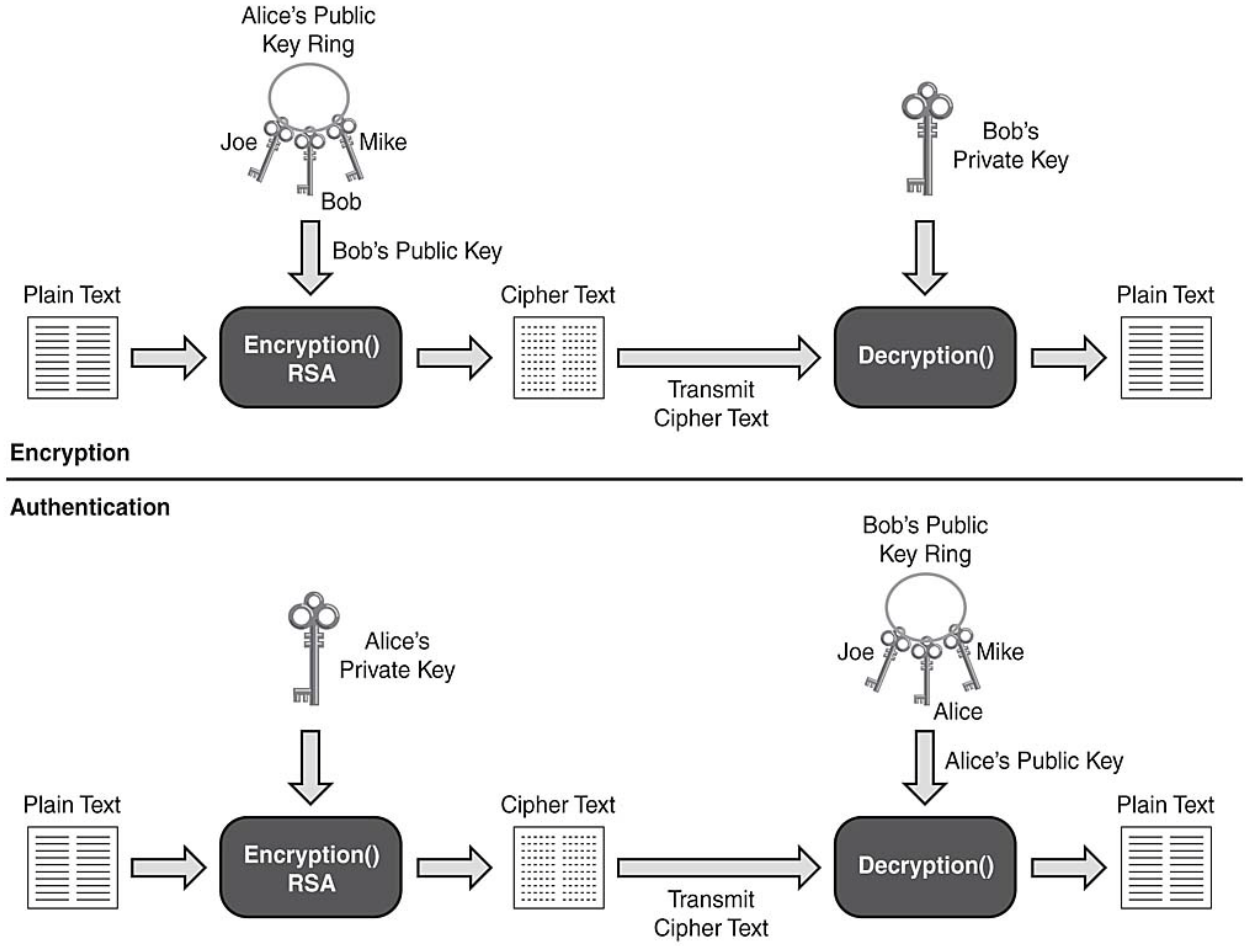

Figure: 1

A. The RSA Algorithm is as follows [6]:

Key generation algorithm, to generate the keys entity A must do the following:

a. Randomly and secretly select two large prime numbers p and q both are prime but p! =q.
b. Compute the N = p*q.
c. Compute Euler Totient function phi (N) = (p-1) (q-1).
d. Select random integer e, 1<e<N, where gcd (e, (n))=1
e. Compute the secret exponent d, 1 < d < phi, such that ed =1(mod (phi)).
f. The public key is (N, e) and the private key is (N, d). Keep all the values d, p, q and phi secret. N is known as the modulus.

e is known as the public exponent or encryption exponent or just the exponent.
d is known as the secret exponent or decryption exponent

B. Public key encryption algorithm:

Entity A encrypt a message M for entity B which entity decrypt. Entity A should do the following operations:
Obtain entity A's public key (N, e)
Represent the message M as an integer in the interval [0….N-1].
Compute C= (M) power mod N.
Send the encrypted message C to entity B.

C. Decryption:

To recover the message m from the cipher text C. Entity B performs the following operations:
Obtain the cipher text C from entity A.
Recover the message M = (C) power d mod n.

D. Literature Survey

The RSA Factoring Challenge was started in March 1991 by RSA Data Security to keep abreast of the state of the art in factoring. Since its inception, well over a thousand numbers have been factored, with the factories returning valuable information on the methods they used to complete the factorizations. The Factoring Challenge provides one of the largest test-beds for factoring implementations and provides one of the largest collections of factoring results from many different experts worldwide. In short, this vast pool of information gives us an excellent opportunity to compare the effectiveness of different factoring techniques as they are implemented and used in practice. Since the security of the RSA public-key cryptosystem relies on the inability to factor large numbers of a special type, the cryptographic significance of these results is self-evident. Some factorization methods are explored in following paragraphs: J *[N, (k. N) + Δ]* and gcd *[N, (k. N) – Δ]* result in nontrivial factors of *N* for different values of Δ where N is the reverse of *N* and k is a positive integer ranging from one to infinity. Sattar J Aboud [3] was introducing a method that breaking the RSA Algorithm based on the knowing

public key *(e, N).* This method will work efficiently if the decryption key d is small.

Aboud and AL-Fayoumi [2] tried analysis the algorithm reversely. By other way, find (d) value firstly, the Euler, and then prime numbers (p, q) by converting the number type of n value to binary and assuming some equations to define the key.

E. Attacks:

The RSA function is an example of a one-way trapdoor function (definition here [8]). It is not known to be easily invertible without the trapdoor, d. General attacks on RSA involve trying to compute d, or some other way to get to M. We will even discuss a way to forge a signature if Alice is not careful. Marvin may know only <C, e, N>, or <C,M,e,N> (known plaintext/ciphertext). RSA will be shown to hold up to all known attacks.

F. Brute Force, Factoring N, computing eth root modulo N:

Given <C, e, N>, Marvin can do a brute force search to find M. But factoring N involves a much smaller search space and thus, more efficient to do. Factoring is a well researched problem. Pomerance has an interesting introduction into what clever methods can be done. For example, how to factor 8051? Naïve approach is trying all primes up to square root of 8051, or almost 90. Clever trick is to find a square of primes that subtract to the number, example 90^2 – 7^2 = 8051. Thus (90 – 7) * (90 + 7) = 8051; 83 & 97 are the factors. However, this trick is only easy if factors are close to square root of the number. There are only a very small set of numbers where this is true. But much more sophisticated methods are known. Currently the general number field sieve is the best known running time with, reported by Boneh [6], a time on n-bit integers of Exp((c + o(1))n^1/3 log^2/3 n) for some c < 2.

To be complete, Peter Shor [10] has shown a quantum computer has a polynomial time algorithm for factoring integers. But engineering a quantum computer still has a way to go, so RSA is safe for now.

RSA security depends on the computation to be unreasonable for 1) the factoring of N, and 2) computing the eth roots modulo N. The first is fairly well researched and complexity is understood. However, the second is an open question. It has not yet been proven that taking the eth roots modulo N is at least as hard as factoring N. This means RSA security has not been proven to be at least as hard as factoring. But it has withstood time and a large amount of research, which is encouraging that it will stay secure well into the future.

G. Elementary misuse of RSA:

Fixing N. One blatant misuse of RSA would be using the same modulus N for all users. So instead, have a central authority issue out unique e,d pairs to each user. This sort of works, in the fact that C1 = M^e1 mod N can only be decrypted with corresponding pair d1. So a different user with e2, d2 doesn't appear able to decrypt. But from the property that knowing <e,d,N> you can factor N, then the game is over. Any user can use their ei,di pair to factor N. Then, given any other user's ej and the factors of N, dj can be computed. RSA key pairs should never use the same N twice.

Signature forgery can be done with a technique called blinding. If Alice ever "blindly" signs M, then Marvin might be able to forge Alice's signature. If Marvin sends evil plaintext M for Alice to sign, it's easy to assume Alice would reject and not sign it. But what if Marvin sends a random, harmless looking M? Depending on the implementation, Alice might be fooled and sign it. Marvin then can take advantageous of this is by generating M' = r^e M mod N, for some random r. If Alice provides a signature, S' on M', then Marvin can compute Alice's signature S for M by S = S'/ r mod N. Proof:

S = S' / r mod N
= M'^d / r mod N
= (M r^e)^d / r mod N
= M^d r^ed / r mod N
= M^d r/r mod N
= M^d mod N

Alice should never freely sign a random M.

H. Low private exponent:

Having a low private exponent d will reduce decryption and signature computing costs. However, too low of a d is insecure. Boneh [6] describes approximations using fractions can allow Marvin to solve for a small d that is d < 1/3 N^1/4. If N is 1024 bits, then d should be at least 256 bits long.

I. Low public exponent:

Having a low public exponent will reduce encryption and signature validation computing costs. However, too low of an e is also insecure. Today's standard e is set at 2^16 + 1. This is a large enough value to avoid attacks and needs only 17 mod multiplications for M^e mod N using repeated squares.

But if a very small e is used instead, it can be subject to attacks such as Hastad's Broadcast Attack. If the same M is encrypted with many users <e,N> keys and broadcasted out, Marvin can collect each and compute M. If all users have the same e, then Marvin needs to collect at least e messages. Restating Boneh [6], take example if e=3:
C1 = M^3 mod N1, C2 = M^3 mod N2, C3 = M^3 mod N3
M < {N1, N2, N3} thus M^3 < N1N2N3
Using CRT C1C2C3 = M^3 mod N1N2N3, thus taking cube root of C1C2C3 gives M.

Stronger attacks are also known on a small e. If you pad M in the above scenario to make it unique for each message, then the broadcast attack fails. But Hastad shows if the padding scheme is a public, fixed polynomial function it doesn't defend from the attack. Franklin-Reiter found an attack on two related messages encrypted with same modulus in time quadratic to e. And Coppersmith took it farther to show an attack on same messages that used a short, random pad (1/9th the size of M). So using a small e is not wise.

To defend against all above low public exponent attacks, large e such as the standard 2^16 + 1 should be used. A good randomized pad also helps make random M's to remove relationship amongst messages.

J. The Proposed Method:

As we mentioned n = *p* * *q*, so if we discover the two prime numbers, this will lead us to gain the key *d*. When we discover any one of q & p, it is so easy to get the other

number. It is known that, (*n)* and (*e)* are representing the public key. From this point we start to construct the mathematical equation as in the proposed Algorithm.

Now, we present the proposed algorithm, and then we show an example to view how the proposed algorithm is applied.

The following algorithm:

a. Find the square root of (n).
b. Identify prime numbers from square root of (n) to 1 by using formula 1.
c. Pick up first prime numbers that its length is the same length of square root of (n).
d. Fulfill $k = w$ (L∈ N) statement.
e. Fulfill (n modular $k = 0$) equation.
f. Apply the equation GCD (value, n) = 0.
g. Find the prime number p or q.
h. Find p or q by this equation = n/q or n/p.
i. Find *p* value
j. Calculate *q* value

Equation

ax2+bx+c

3.234x2 -9x +5 (1)

6.065x2 -90x +500 (2)

8.719x2 -900x +50000 (3)

11.26x2 -9000x+ 5000000 (4)

The value of a can be calculated from the equation 0.072x2 +3.035x +0.274

Now, take an example for each step that shown in Algorithm:

a) Compute the square root of (*n)*. Suppose we have the value of (n) which is equal to 1457. Now, the square root of (n) is almost= 38 and this value will be called sqt *of (n).*
b) Find all the prime numbers, which falls between [1 and square root of (*n*)], and put them in array called (*v)*. This interval will be: *v* = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37].
c) Find the first number that has length exactly as the length of square root number in step 1 from *v* array (suppose it is *w*). As it mentioned at step 3 of the algorithm, then *w* = 2.
d) From (*v)* array, collect all numbers that start with (*w)* (length = 2) till last number of (*v)* (suppose it is *G*). On the safe side, collect the remaining numbers that except *G* to make feed back if there is no true answer of later step 6 (suppose it as *G'*) in the Algorithm. Following – up the example, the array will be: *G* = [11, 13, 17, 19, 23, 29, 31, 37].
e) Find the numbers that testify the following statement: $k = G$(L∈ N) , Where: L = last digit of the prime numbers in *G* array elements, N = set of numbers that if multiplied together, the first number of these results will be the same number that at last digit of *n* value. So, we can classify these sets into two collections as the following:

N1 = [1, 3, 7] when the last digit of *n* value will be 1, 3 or 7.

N2 = [1, 3, 5, 7] when the last digit of *n* value will be 5.

For the prime numbers that held 9 in the last digit of them will go to *G'*. In other words, we can describe k = (L∈ N) statement as collection in *k* - as the residual numbers of probabilities database elements - all numbers that held the last digit they included into N, where N had been classified. Note the results of each step and regard how much we are decreasing the space of search area to reach to desired factors with lowest range of probabilities for these factors.

As it displayed of the proposed algorithm, the variables of (L) of the example will be defined as the following:

L = [2, 3, 5, 7, 1, 3, 7, 9, 3, 9, 1, 7] and N = [1, 7], Then, k =[11, 17, 31, 37].

In this work we tried a method that aims at reducing the unwanted prime numbers as possible as to increase the possibility of a successful attack.

f) Find the indexes of special numbers that only investigated with the following equation: indexes = (n modular k = 0). The meaning is to save the locations for any prime number when the remainder with n values = 0 in array (k). So, the index value of the example is 3, (k=31).
g) The result is the smallest prime number (p or q) constructing n. this number is unique and corresponding to one index location. As mentioned before, any value of (n) is constructed by multiplication of a pair of prime numbers p & q. Since any two prime numbers have no common factor, the result will be finding one part of n. Following – up the example, the value that hold index = 3 as in k array is 31.
h) Finally, since number 31 is define the p, immediately the q =pn, q = 47.

III. EXPERIMENTAL WORK

By using the suggested Algorithm to factorize the value of (n), we apply a collection of variant lengths of presuming (n) numbers on the proposed method. From this examination, we got the results as follows, Table (1).

It is a fact that length of p & q is getting longer and longer and this will make this type of attack harder and harder, due to the limitation of the machines and the processing time. So, we make an elapsed time equation that was evaluated and derived from the results that we got in Table I. The equation was:

This equation used to simulate the next lengths of (n) values to testify how the result of the proposed algorithm is if length of (n) was more than 100 digits. As we know, the current operating system does not configure to receive and generate a big process like an input of (n) could have length 150 digit or more (Window 32 bit or 64 bit). To solve this situation, and using the Table (1) results, we examine the equation to see how much it matched between the assuming and real result, and we got a closed similarity of results. As shown in Figure (2). Although these numbers were not large as desired, where the max length of digit of n we have reached is 19, because the limitations of hardware and software of the computer are not suitable to manipulate more large numbers of (n), although that, we were able to reach to a very encourage results by applying the proposed method as it shown in Table (1).

The results that we obtained proved that the proposed algorithm is reasonable and could be used for large numbers of (n). We used a computer with limited specification such as the hard disk size (40 GB), CPU speed (3.0 GHz) and the RAM capacity is 1.5 GB. Also, there are many programs and applications that could be considered affections on the operating systems speed. We have applied the proposed Time equation of the proposed algorithm to estimate the

factorization time on a different large prime numbers (n), we got a very good result as shown in Table III.

IV. CONCLUSION

RSA has withstood the test of time and much research as being secure. Its security depends on factoring large integer N and taking the eth root modulo N as not being computable in any reasonable amount of time. So far, this has been a safe and secure bet.

On the other hand, poor or naïve implementations of RSA have been shown to be insecure. Engineers that implement RSA must understand this knowledge to avoid making these mistakes. RSA implemented correctly has been a very successful cryptosystem.

V. REFERENCES

[1]. AL-Hamami AL-Ani, Technology of information security and protection systems, ISBN 978-9957-11-697-2, pp.173 - 223, Dar Wael , Jordan. 2007

[2]. Aboud, AL-Fayoumi; "Efficient Method for Breaking RSA Algorithm"; Ubiquitous Computing and Communication Journal, vol. 4,no.2 , p:1520, 2008.

[3]. Coppersmith, D. "Attack on the Cryptographic Scheme", Advances in Cryptology–CRYPTO '94, Springer-Verlag, LNCS 839, pp.294-307, 1994.

[4]. Hastad, J. "On Using RSA with low exponent in a public key Network", Advances in Cryptology –CRYPTO '85, Springer-Velag LNCS 218, pp. 403-408, 1986.

[5]. Nguyen, H. Number Theory and the RSA Public Key Cryptosystem. http://cdn.bitbucket.org/mvngu/numtheorycrypto/downloads/numtheory-crypto.pdf.

[6]. D. Boneh. Twenty Years of Attacks on the RSA. Notices of the American Mathematical Society, vol 46(2):203–213, 1999.

[7]. 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.

[8]. Alfred J. Menezes, Paul van Oorschot, and Scott Vanstone. Handbook of Applied Cryptography. CRC Press, 1997

[9]. C. Pomerance. A tale of two sieves. Notices Amer. Math. Soc., 43:1473-1485, 1996.

[10]. P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SICOMP, Volume 26 Issue 5 pages 1434–1402, 1997.