Public Key Cryptography: RSA

Manav Chirania
Roll No- 180123026

Kaushal Chhalani
Roll No- 180123023

May 20, 2020

Abstract

The RSA works behind the logic that even though the encryption key is revealed , the decryption key does not get revealed because of it. The decryption key remains safe because of the fact that it easy to generate primes but it is hard to factorise a number.We use two keys,the public key(e,n) and the private key(d,n). Here n is product of two primes p and q. Also $ed \equiv 1(mod(p-1)*(q-1))$. The encryption is done on message M by raising it to the power e and then taking the remainder by dividing it by n and similarly the decryption is done. We study about the various algorithms such as Miller-Rabin, Euclid's Extended Theorem used in RSA and also discuss the security of the system.

Keywords: *signature,cryptography,asymmetric key, public key cryptography,prime number,factorization,privacy.*

1 Introduction

The RSA public-key cryptosystem is one of the most widely used cryptosystem for data encryption. The RSA cryptosystem depends on the difference between the ease of finding large prime numbers and the difficulty of factoring the product of two large prime numbers.

In a public-key cryptosystem , a user has a pair of keys: a public key, which as its name suggests, is known to the public and a private key which is kept secret. In such a system, any person can encrypt a message using the recipient's public key but the message can only be decrypted by the recipient's private key. A public-key cryptosystem which implement signatures also holds the property that decrypting first and then encrypting the message M results in M. The security of this system depends on the condition that revealing the public key does not compromise the security of the private key. Generation of such keys depends on the mathematical problem of producing one-way functions. Let E_a, D_a, E_b, D_b be the encryption and decryption keys of Alice and Bob respectively. So if Alice has to send a private message M to Bob, she uses Bob's public key E_b to encrypt M and sends the encrypted message $E_b(M)$ to Bob. Bob deciphers the message received using his private key by calculating $D_b(E_b(M)) = M$. Since revealing the public key does not compromises the private key, Alice can be ensured that only Bob can compute the original message M.

Signing in public cryptography system is an important concept as the recipient of the signed message has proof that the the message originated from the sender. Here, the electronic message is message dependent as well as signer-dependent so that the recipient could not alter the message before they show the message-signature pair to a "third party/judge". Bob signs his message first with his private key which is done by $S = D_B(M)$. Now he encrypts his signature using Alice's public key and thus sends Alice $E_A(S)$. Again Alice uses her private key to decrypt,$D_A(E_A(S))$) and gets S and uses Bob's public key, $E_B(S)$, to retrieve M. In this way Alice receives a pair of (M, S). In this way Bob cannot later deny that he didn't send the message as no one could have generated S. Thus Alice can prove that the message,$E_B(S) = M$ was signed by Bob. Also Alice can't tamper the message because if she did it to M' then she would also have to generate the signature S' such that $S' = D_B(M')$.

The RSA cryptosystem, by Ron Rivest, Adi Shamir and Leonard Adleman [2], used exponentiation modulo a product of two large prime numbers for encryption and decryption of messages as well as for using using digital signatures.

2 Arithmetic of RSA

RSA works using a public key of the form (e, n) (where $n, e > 0$) and a private key (d, n) using the following encryption and decryption algorithms, E and D respectively:
$C = E(M) \equiv M^e (\mathrm{mod}\, n)$, for message M
$D(C) \equiv C^d (\mathrm{mod}\, n)$, for ciphertext C
The next question is to generate the public key (e, n) and private key d.

The first step is to select two distinct large prime numbers p and q , and compute n as:

$$n = p * q$$

The next step is to select e such that $1 < e < n - 1$ and gcd$(e, (p - 1) * (q - 1)) = 1$.
Finally, d is computed as the multiplicative inverse of e, modulo $(p - 1) * (q - 1)$ so that

$$e * d \equiv 1 (\mathrm{mod}\, (p - 1) * (q - 1)) \tag{1}$$

The public key is then published as (e, n) and the private key as (d, e).

Proof of correctness:
The proof of the correctness of RSA [3] is based on Fermat's little theorem, stating that $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$ for any integer a and prime p, not dividing a. Also, we have $\phi(n) = \phi(p) * \phi(q) = (p - 1) * (q - 1)$, where $\phi(n)$ is the Euler totient function giving the number of positive integers less than n which are relatively prime to n.
After encryption and decryption of message M, we have

$$D(E(M)) = M^{ed} (\mathrm{mod}\, n)$$

Consider $M^{ed} (\mathrm{mod}\, p)$ and as $e * d = 1 + k(p - 1) * (q - 1)$, we get

$$M^{ed} \equiv M^{1+k(p-1)*(q-1)} (\mathrm{mod}\, p)$$

Since, $M^{p-1} \equiv 1 (\mathrm{mod}\, p)$ for M not divisible by p, this results to

$$M^{ed} \equiv M((M^{p-1} \mathrm{mod}\, p)^{p-1})^{k(q-1)} (\mathrm{mod}\, p)$$

$$\equiv M(1)^{k(q-1)} (\mathrm{mod}\, p)$$

$$\equiv M (\mathrm{mod}\, p)$$

Also, $M^{ed} \equiv M (\mathrm{mod}\, p)$ for M divisible by p. Therefore, $M^{ed} \equiv M (\mathrm{mod}\, p)$ for all M.
Similarly, $M^{ed} \equiv M (\mathrm{mod}\, q)$ for all M. Combining these two using Chinese Remainder Theorem,

$$M^{ed} \equiv M (\mathrm{mod}\, n)$$

for all M.

3 Algorithms

The RSA algorithm can be divided into three sections:

1. The generation of prime numbers using Miller-Rabin randomized primality test:
Miller-Rabin primality test[3] is probabilistic test in which if the output is given as composite, then it is surely composite but if the give output as prime, then it is probably prime. We would like to state some points before studying the algorithm.
1. If $a^{n-1} \not\equiv 1$ (mod n) for some a$\not\equiv 0$ (mod n), then a is composite.
2. If n is prime, then $a^{n-1} \equiv 1 (modn)$ for all $a \not\equiv 0 (modn)$ but the converse is not necessarily true.
3.If n is odd prime, we know that an integer can have at most two square roots, mod n. In particular, the only square roots of 1 (mod n) are ± 1. As n is an odd prime, so if $a \not\equiv 0 (modn)$, then $a^{(n-1)/2} \equiv \pm 1 (modn)$

4. If $a^{(n-1)/2} \not\equiv \pm 1(modn)$,then n is composite.
Using above properties we can get rid of Fermat's psuedoprimes and Euler's psuedoprime. Now, in Miller-Rabin test, we write $n-1 = 2^s.m$ where m is odd and s≥1.Using above, we would like to draw some observations on the sequence of $< a^m(modn), a^{m.2}(modn), .., a^{m.2^s}(modn) >$.
1. Since it is continuous squaring in the sequence, so once 1 appears in the sequence the rest of the terms will also appear as 1.
2. So, if the last entry of the sequence which is non-1 is not equal to -1 ,then the number is surely composite. But if the last entry which is non-1 is equal to -1, then the number is probably prime(except if the last term of the sequence is equal to -1 in which case it is composite).
Using the above points,the CHECK function is given below.

1. CHECK(a,n)
2. Let n-1=$2^s.m$
3. Set X_0 to $a^m(modn)$(using exponentiation by repeated squaring and multiplication as given in encrypting)
4. Repeat step 5,6 for i=1,2,..,s
5. Set X_i to $X_{i-1}^2(modn)$
6. If $X_i = 1$ and $X_{i-1} \neq 1$ and $X_{i-1} \neq n-1$, return TRUE (indicating it is composite).
7. If$X_s \neq 1$ return TRUE else return FALSE (indicating it is probably prime)

Using CHECK function, the Miller-Rabin test is given below.
1. Miller-Rabin(n,t)
2. Repeat Steps 3,4 for i=1,2,..,t
3. Set variable a to any random value in between 1 to n-1.
4. If CHECK(a,n), then return the number is surely composite.
5. Return the number is probably prime.
Now, two questions arises out here is that how many times do we to have run this test to get a prime and also regarding the error rate of this test.
1. Let $\pi(n)$ be the prime distribution function which tells us the number of primes which are less than or equal to n.
Theorem : $\lim_{n\to\infty} \pi(n)/(ln(n)/n) = 1$. [3]
This approximation gives reasonably accurate estimates for even small value of n. Thus a randomly chosen number is prime has a probability of $1/ln(n)$. So,we need to examine around ln(n) integers chosen randomly near n in order to find a prime that is of the same length as n. For example, for a 512 bit number around $ln2^{512} \approx 355$ numbers will be chosen for primality.
2.For error rate of this test
Theorem : If n is an odd composite number, then the number of different values of a for which it will show the number to be composite is at least $(n-1)/2$ [3].
So, for a wrong result to be given by Miller-Robin Test, in every loop it should return false by CHECK function and probability of returning composite(if the number is composite) in every loop is half. So the probability of giving a wrong output is 2^{-t}. So if t is even greater than 50, the probability of giving a wrong output is so less that it is almost accurate.

2. Choosing e and d using Extended Euclid's Theorem [3]: It is very easy to choose a number e which is relatively prime to $\phi(n)$. e should be greater than $max(p,q)$ so that cryptanalyst cannot break it directly. Now to compute d,since $ed \equiv 1(mod\phi(n))$and GCD($e, \phi(n)$) = 1, so $ed + an = 1$where a is a constant.Thus d can be calculated using *Euclid's Extended Algorithm* [3].
3. Encrypting and Decrypting properly [2]: We need to compute $M^e(modn)$. Now e can be represented as $e_k e_{k-1}...e_0$ in the binary representation. Since $M^e = M^{e_k 2^{b_k} + e_{k-1} 2^{b_{k-1}} + ... + e_0 2^{b_0}}$, where $b_k, .., b_0$ represents the bitwise position. The algorithm can be done by:
1: Represent e in binary form as mentioned above
2: Set the variable C to 1.
3: Repeat the step 3a and 3b for i=k,k-1,...,1,0
3a: Set C to the remainder when C^2 is divided by n.

3b: If $e_i = 1$, set C to the remainder of C*M when divided by n.

The final C obtained is the encrypted form of M. This procedure is called is exponentiation by repeated squaring and multiplication which will have a time complexity of $2 * log_2 e$. Similarly we can decrypt the text and obtain the original message M .

4 Security Of RSA public-key cryptosystem

The security of the RSA cryptosystem is based on two problems : the RSA problem and the problem of factoring large integers. The RSA problem is defined as the task of computing M given an RSA public key *(n,e)* and a ciphertext $C = M^e \pmod n$. Rivest, Shamir, and Adleman, in section IX/D of their paper [2], noted that they had not found a proof that the RSA problem is equally as hard as integer factorization. Still, currently the most likely approach to solve the RSA problem is to factor n. No polynomial-time method for factoring large integers on a classical computer has yet been found. Clearly, the problem of integer factorization is in class NP but it has not been proved to be or not be NP-complete.

Most of the factoring methods use the same basic technique, which is the 'difference of squares' method [1]. The method depends on the observation that if we find two integers x and y, such that $x \neq y \pmod n$ and

$$x^2 \equiv y^2 (\mathrm{mod}\, n)$$

then gcd$(x + y, n)$ and gcd$(x - y, n)$ can give us non-trivial factors of n. In cases where n is the product of distinct primes p and q, such as in RSA, a non-trivial factor is extracted in 2/3 of the cases.

The Quadratic Sieve (QS) factoring algorithm of Carl Pomerance [1] is the second fastest algorithm for integer factorization and is based on the above method. It is still the fastest algorithm for integers under 100. QS uses the concepts of factor bases and smoothness for producing squares. A nonempty set F of positive prime integers is called a *factor base*. An integer k is said to be *smooth* over the factor base F if all primes occurring in the unique factorization of k into primes are members of F(insert ref).

QS begins by fixing a factor base $F = \{p_1, p_2, ..., p_m\}$ and then proceeds to compute a set M of random integers r_i with the property that $f(r_i) = r_i^2 - n$ is smooth over F. When more than m such integers are found, a subset U of the integers in the sequence can be found such that if

$$x = \prod_{r_i \in U} r_i \quad \text{and} \quad y = p_1^{e_1} p_2^{e_2} .... p_m^{e_m}$$

We get

$$x^2 = \prod_{r_i \in U} r_i^2 \equiv \prod_{r_i \in U} (r_i^2 - n) \equiv \prod_{r_i \in U} f(r_i) \equiv y^2 (\mathrm{mod}\, n) \tag{2}$$

For finding U, we associate a vector $v_i \in \mathbb{F}_2^m$, where $\mathbb{F}_2^m$ is the m-dimensional vector space over $\mathbb{Z}/2\mathbb{Z}$, to each r_i. The j^{th} co-ordinate of v_i is set to 1 if p_j divides $f(r_i)$ odd number of times and is set to 0 otherwise .It is a theorem of linear algebra that with more vectors than each vector has elements, a linear dependency always exist. Since the only scalars are 0 and 1, a linear dependence relation is simply a subset sum equaling the 0-vector. This means the product of $f(r_i)$ values corresponding to the v_i's in the dependency will be a perfect square which leads to (2).

Our next objective is to find enough r_i's, in other words, more than m integers, to obtain the linear dependency mentioned above. To do this, we first fix a r_i and a prime $p \in F$ such that $f(r_i) = r_i^2 - n \equiv 0 (\mathrm{mod}\, p)$. Now, due to the quadratic nature of f, if $f(r_i)$ is divisible by p, $f(r_i + kp)$ will also be divisible by p for any integer k. A bound $-u < r_i < u$ is selected where it is expected that more than m r_i's can be found. An array is initialized to all the $f(r_i)$ values possible in the range. For each prime $p \in F$, the congruence $r_i^2 \equiv n (\mathrm{mod}\, p)$ is solved and for every $k \in \mathbb{Z}$ such that $-u < r_i + kp < u$, the corresponding $f(r_i + kp)$ is divided by p. After this is performed for every $p \in F$, the values of $f(r_i)$ which factor completely over F will be equal to 1 in the array. The r_i's corresponding to the $f(r_i) = 1$ are the values we need for the set M.

This method is similar to the Sieve of Eratosthenes. In the Sieve of Eratosthenes, for every prime all the numbers divisible by that prime is crossed off. In QS, instead of crossing off, the numbers are divided by the prime numbers of the factor base F. So, at the end of sieve, any number which is F-smooth will change to 1. QS has a running time of $e^{(1+o(1))\sqrt{\ln n \ln\ln n}}$.

The General Number Field Sieve(GNFS) has a running time of $e^{(\sqrt[3]{64/9}+(o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3})}$ and is the best factorization algorithm for integers larger than 100 digits. GNFS, is based on applying the notions of factor bases,smoothness and dependency-finding on algebraic number fields.

In practice, RSA keys are generally 1024 bits to 4096 bits long and the largest RSA number which was factored is 795 bits long. So, it is presumed that RSA is generally secure when n is large enough.

5 Observations

Two basic observations that were made while working on this project:
1. While calculating public key(e,n) and private key(d,n), since e and d must satisfy $ed \equiv 1(mod(\phi(n)))$, we can also use $ed \equiv 1(mod(lcm(p-1, q-1)))$ for calculating e and d.
2. If e or d is set to be one of the primes amongst p, q *(say e or d is set to p)*. then the cipher texts of all the messages ,which are the multiples of the other prime *(here the other prime is q)* which is not set as e or d from p or q , will be same as the plain texts. For example, n=11*13, public key=(37,143) and private key=(13,143) so the cipher text of 11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121, 132 is same as 11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 132 respectively.

Proof: The above statement can be written mathematically as p,q are two primes we need to prove that $(x.q)^p \equiv x.q(\mathrm{mod}\,(p*q))$ for all x such that $0 < x < p$.
Now since $0 < x < p$ and q is prime so GCD$(x.q, p) = 1$.
Using Euler's Theorem, $(x.q)^{p-1} - 1 = k_1 p$, where k_1 is a constant.
So, $x.q((x.q)^{p-1} - 1) = k_1.x.pq$
Therefore, $(x.q)^p \equiv x.q(\mathrm{mod}\,(p*q))$

6 Conclusion

RSA cryptosystem is one of the most widely used public-key cryptosystem. We discussed about the underlying mathematics of RSA, and various algorithms used in the implementation of RSA. In brief, we discussed about the Miller-Rabin algorithm for generation of primes, Extended Euclid's for calculating e,d and exponentiation by repeated squaring and multiplication which has been implemented for the execution of RSA. Further studying about the security of RSA, we see that the security depends on the hardness of the problem of integer factorization. We see how modern integer factorization algorithms have developed from the 'difference of squares' method to the Quadratic Sieve and then to the General Number Field Sieve.

References

[1] Briggs, Matthew Edward. *An introduction to the general number field sieve*. Diss. Virginia Tech, 1998,pp.3-6

[2] *A method for obtaining digital signatures and public-key cryptosystems*. R[onald] L. Rivest, A[di] Shamir, and L[eonard M.] Adleman. CACM 21,2 (1978) pp. 120–126

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). *Introduction to algorithms*. MIT press,pp.962-975