**RSA cryptosystem with large key length**

**1 Introduction**

**1.1 Description of problem**

RSA is a practical public-key cryptosystem which is widely used in cyber security. It is usually involved in such a question that what key length is appropriate, 200 bits, 1000 bits, 6000 bits or more? 200 bits was first considered as a secure length of public-key according to Rivest, Shamir and Adleman [6]. It took more than one billion years to break RSA with 200 bits public-key length in 1978. However, with the development of computer science technique and engineering, by using proper integer factorization algorithm, it only took less than 30 minutes to break 200 bits long RSA for the computer with Intel Dual-Core i7-4500U 1.80GHz. So far, a 768-bit key has been broken by Shor’s algorithm with quantum computers.

Typically, the RSA key with length from 1024 to 4096 bits is considered to be reasonable recently. As we already know, the longer the key length is, the more difficult it would be cracked. What is the reason that stopping us to pick the longest public-key. The reason is that RSA is a relatively slow algorithm, as the length of key increase, although it makes itself difficult to be cracked and improves the security of data, it also takes more time to encrypt and decrypt data.

In this paper, I will implement a RSA cryptosystem with several large length key from 200 bit to 4096 bits or more. The time complexities of best case, worse case and average case for each algorithm I use would be obtained from these experimentations. Combined with the theoretical time complexity of integer factorization algorithm, I will prove that the key length within certain range will be practice feasible.

**1.2 Description of algorithms**

The RSA algorithm is based on Diffie-Hellman algorithm [1] which is a specific method to exchange cryptographic keys through public network channel securely. For instance, first Alice and Bob agree publicly on a prime modulus and a generator like 23 and 5. Then Alice chooses a secret random number 13, calculates 5 to the power 13, mod 23, and sends the result publicly to Bob. Meanwhile, Bob also chooses a secret random number 17, calculates 5 to the power 17, mode 23, and sends the result publicly to Alice. Once Alice got Bob’s result, she raises the result to the power of her secret number 13, mod 23 to obtain the shared secret. The same as Bob. After this exchange is done, both Alice and Bob will have the same shared secret key.

In this RSA cryptosystem, a pair of keys will be generated to encrypt and decrypt data. It involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption. In this paper, I will focus on key generation, data encryption and decryption.

Several modern cryptographic algorithms will be used to generate large size key pairs, including Euclidean algorithm, Chinese remainder theorem [2], modular exponentiation and primality test like Miller-Rabin primality test [3]. The procedure of key generation is as follows. First of all, it will generate two big random numbers p and q which are differ in the length of several bits to make them more difficult to guess. Then it implemented Rabin-Miller primality test algorithm with certain number of witnesses like 64 to determine whether those two big number are prime or not. The Rabin-Miller algorithm is a prime probability test algorithm, and it would be only 1/(2^128) chance of not being a prime number with 64 witnesses. After that, it randomly generates e and uses Euclidean algorithm to check until the greatest common divisor with (p-1)(q-1) is 1. For the data encryption and decryption, modular exponentiation will be used to calculate a type of exponentiation performed over a modulus.

**1.3 Specific Aims**

**Specific Aim 1** I will implement the RSA cryptosystem by using Euclidean algorithm, Chinese remainder theorem, modular exponentiation and primality test.

**Specific Aim 2** I will add a time complexity monitor for each algorithm to obtain the experimentation results. For the test cases, several different length of public keys will be used to test this RSA cryptosystem, and the stages will cover key pair generation, data encryption and data decryption.

**Specific Aim 3** In order to measure the efficiency of RSA algorithm. I will analysis the statistic data gathered from the experimentation. Estimate the time complexities for each algorithm including best case, worse case and average case. Meanwhile, the time complexity for encryption and decryption will also be covered in this statistic analysis.

**2 Background**

**2.1 RSA cryptosystem**

RSA is a widely used public-key cryptosystem for secure data transmission. It was first described in 1977. For the key generation, it will choose two random prime number p and q. compute pq as n which is used as the modulus for both the public and private keys. Then, compute the Euler’s totient function with (p-1)(q-1). After that, choose an integer e such that e is between 1 and the totient and their greatest common divisor is 1. In other hand, e and the totient are coprime. At the end, compute d given de = 1(mod(totient)). As a result, (e, n) is the public key and (d, n) is the private key. After key generation, it will compute ciphertext c by raise plaintext m to the power of e, and mod n. For the decryption, simply raise ciphertext c to the power of d, and mod n.

**2.2 Primality test and probabilistic test**

The primality test is an algorithm to determine whether the number is prime or not. The probabilistic test is one of the popular primality test, including Fermat primality test, Miller-Rabin primality test, Forbenius primality test and so on. These test will not guarantee the random number is a real prime number. However, the probability of error can be reduced to be very small.

**2.3 Euclidean algorithm**

The Euclidean algorithm is an efficient way to calculate the greatest common divisor of two numbers. This algorithm can be used in the RSA cryptosystem where e has a requirement that the greatest common divisor with totient is 1. The basic principle of Euclidean is that the greatest common divisor of two number will not change if the larger number is replaced with the difference of these two number.

**2.4 Chinese remainder theorem and Modular exponentiation**

The Chinese remainder theorem is that if knows the remainders of the division of an integer n by several integers, it can determine the remainder of division of n by the product of these integers if the divisors are pairwise coprime. Shinde discussed using Chinese Remainder Theorem in [2]. Modular exponentiation is useful in RSA cryptosystem. It is a type of exponentiation performed over a modulus. It calculates the remainder when an integer m raises to the power of d (the exponent), and divided by n (the modulus).

**2.5 Factoring algorithm**

Factoring algorithm is the decomposition of a composite number into product of small numbers which is not part of RSA algorithm. However, it is much important since it can determine whether the RSA cryptosystem is secure or not. The RSA cryptosystem is based on the difficulty of factoring large composite integer. For instance, in 2009, it took two years with hundreds of machines to factor RSA-768.

**3 Experimentation Results**

The following tables (Table 1, 2) showing time complexity analysis for RSA cryptosystem. The data in these tables will be replaced with real results from the experimentation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key length | Number of attempts / Time cost (sec)  (Primality test) | Time cost (sec)  (Euclidean algorithm) | Time cost (sec)  (Modular exponentiation) | Time cost (sec)  (Chinese remainder) |
| 200 | 155/0.01 | 0.01 | 0.01 | 0.01 |
| 1000 | 155/0.2 | 0.2 | 0.2 | 0.2 |
| 3000 | 155/0.45 | 0.45 | 0.45 | 0.45 |
| … | … | … | … | … |
| 6000 | 155/0.9 | 0.9 | 0.9 | 0.9 |

**Table 1:** Results obtained from experimentation

|  |  |  |  |
| --- | --- | --- | --- |
|  | **B(n)** | **W(n)** | **A(n)** |
| Primality test |  |  |  |
| Euclidean algorithm |  |  |  |
| Modular exponentiation |  |  |  |
| Chinese remainder |  |  |  |
| Factoring algorithm |  |  |  |

**Table 2:** Time complexity for each algorithm

**4 Work Plan**

This section describes the work plan for achieving the three specific aims.

**4.1 Aim 1: Implement the RSA cryptosystem**

I will develop the RSA cryptosystem using Euclidean algorithm, Chinese remainder theorem, modular exponentiation and primality test. There are five stages in this period.

4.1.1 I will implement the primality test algorithm. The main task of this stage is generate two large distinct random prime numbers p and q. These integers can be efficiently found by using primality test.

4.1.2 I will implement Euclidean algorithm which will be used to compute the greatest common divisor. Although this algorithm is much simple than primality test algorithm, it is an efficient method for computing the greatest common divisor of two numbers.

4.1.3 The Chinese Remainder Theorem is a common practice during decryption. I will implement this algorithm as an option to replace modular exponentiation.

4.1.4 The modular exponentiation will be used in both encryption and decryption. In this stage, I will implement an efficient modular exponentiation.

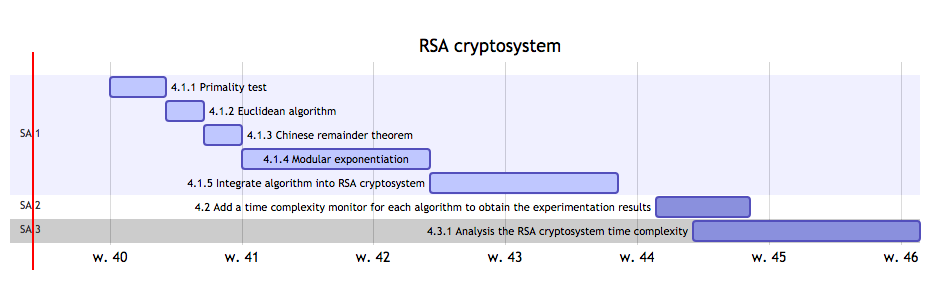
4.1.5 After implemented all the algorithms, the last stage is to integrate these modules into RSA cryptosystem.

**4.2 Aim 2: Add a time complexity monitor for each algorithm to obtain the experimentation results**

After completed RSA cryptosystem, I will add time complexity monitor into each module. In this period, I will create different test cases as the input data of the cryptosystem, and obtain the experimentation results.

**4.3 Aim 3: Analysis the RSA cryptosystem time complexity**

The time complexity analysis of RSA cryptosystem is the most important part. By using the results obtained from the previous experimentation, and combined with the theoretical time complexity of integer factorization algorithm, I will prove that the key length within certain range will be practice feasible.

****

**Figure 1:** Gantt chart showing the schedule of the work plan.

**5 Broader Impacts**

RSA cryptosystem was first introduced in 1977, and it has been decades of years. However, the length of RSA key is still an open question because of the rapid development of computer science. This project is an attempt to analysis the RSA cryptosystem to find the key within certain range that will be practice feasible which will help to improve the current RSA algorithm.

**References:**

[1] Diffie, W., and Hellman, M. “New directions in cryptography”, IEEE Trans. Inform. Theory IT-22, pp. 644-654, 1976

[2] G.N. Shinde, and H.S. Fadewar, “Faster RSA Algorithm for Decryption Using Chinese Remainder Theorem”, ICCES, vol.5, no.4, pp.255-26, 2008

[3] H. W. Lenstra, Jr., “Miller’s primality test”, Inform. Process. Lett. 8:2, pp. 86- 88, 1979.

[4] L. Zhong, “Modular exponentiation algorithm analysis for energy consumption and performance,” tech. rep., Citeseer,2000.

[5] Nentawe Y. Goshwe, “Data Encryption and Decryption Using RSA Algorithm in a Network Environment”, International Journal of Computer Science and Network Security, Vol.13 No.7, pp. 9-13, 2013.

[6] R.L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, Volume21 Issue 2, pp. 120-126, 1978.

[7] Rene Schoof, “Four primality testing algorithms”, Algorithmic Number Theory, Volume 44, 2008.