Improved RSA: Encryption and acceleration

Letian Xie
letian.xie18@student.xjtlu.edu.cn

Xinzi Li
xinzi.li18@student.xjtlu.edu.cn

Longyang Wang
longyang.wang18@student.xjtlu.edu.cn

Tianlei Shi
tianlei.shi18@student.xjtlu.edu.cn

Xiaoyue Ma
xiaoyue.ma18@student.xjtlu.edu.cn

***Abstract*—One direction in many research efforts is to study and develop many applications of public key cryptography to ensure the security of data as it travels through a system. Among the current mainstream encryption algorithms, RSA is generally regarded as one of the best public key schemes available. In this report, we propose a method to optimize the RSA algorithm for encrypting and decrypting message text. The security of the RSA algorithm is increased by generating large prime numbers and preprocessing the messages by Ascii, and the computational consumption and efficiency of the RSA algorithm is improved by increasing the speed of decryption. Finally, we implement this method in Python and present experimental results against the original RSA algorithm.**

Keywords—RSA, Security, Speed, Large prime generation, Faster decryption, ASCII

I. Introduction, Motivation and Background

Following the development of the Internet and information technology, e-commerce has grown from its rise to popularity. Electronic documents and information management methods are being adopted in more and more occasions, so information security technology with data encryption as the core has been developing rapidly. Encryption algorithms can be divided into three categories: symmetric encryption algorithms, asymmetric encryption algorithms and hash algorithms. Hash algorithms of these are not used for digital signatures as they do not require a key. In symmetric encryption techniques, the same key is used for the process of encrypting and decrypting a message. The advantage of this encryption method is that it simplifies the encryption process and there is no need for both sides to study the encryption algorithm when exchanging messages. In addition, the integrity and confidentiality of the message is guaranteed by not revealing the private key during the exchange phase. AES, 3DES and AES are the representatives of symmetric cryptographic algorithms. [1].

In asymmetric encryption systems two different but related keys are used, the public key and the private key [2]. The public key is used for encryption, while the private key is used for decryption. The encryption process is the conversion of ordinary text into cipher text by the public key. And decryption of cipher text requires the recipient's private key. The private key needs to ensure its confidentiality, but the public key can be disclosed to the public. Asymmetric encryption allows secure communication to be established without exchanging keys. Since the public key does not provide enough information to decrypt the cipher text, message communication can take place in a secure manner.

There are many types of asymmetric encryption algorithms, and the RSA public key cryptosystem is the most representative form of asymmetric encryption [3]. But RSA has some limitations of its own: RSA requires the use of two large prime numbers to generate the key. Due to the limitations of prime number generation technology, users cannot generate a large number of prime numbers in a short period of time. As a result, RSA is inconvenient in situations where the user is required to change the key at any time. In addition, the RSA key length is too large. To ensure security, the value of n in RSA needs to be very large, which makes the calculation very expensive. Compared to symmetric encryption algorithms, RSA is several orders of magnitude slower. And with the development of a large number of decomposition techniques, the length of the key will continue to increase.

In this paper, the RSA algorithm will be described and analyzed later. The Miller Rabin algorithm and large prime number generation will be used to improve the speed of RSA.

II. Literature Survey

As a classic asymmetric encryption algorithm, RSA was proposed by three researchers named Rivest, Shamir, and Adleman. It was invented to address the security risks of symmetric encryption algorithms, because in symmetric encryption model, the process of encrypting and decrypting messages using only one key, and during the messaging process, a third party can easily obtain the key and decrypt the message. However, RSA algorithm adopting the mechanism of public key and private key, and using different keys to encrypt and decrypt information. What's more, a public key can correspond to more than one private key, which makes it easy for people to know the way of divulgement. The mechanisms of encryption and decryption of RSA algorithm are shown as below.

- Setup:
  - $n = p\,q$, with p and q are primes
  - e relatively prime to $\phi(n) = (p-1)\,(q-1)$
  - d inverse of e in $Z_{\phi(n)}$, where $Z_{\phi(n)}$ is the Euler's function

- Keys:
  - Public key: $K_E = (n,\ e)$
  - Private key: $K_D = d$
- Encryption:
  - Plaintext M in Z_n
  - $C = M^e \; mod \; n$
- Decryption:
  - $M = C^d \; mod \; n$

The reliability of a secure communication system depends on the reliability of the cryptographic algorithm. But as people's needs and data needs increase year by year, RSA faces different challenges. The speed and security of the RSA algorithm have been the subject of research in recent years. The security of the RSA algorithm is based on the fact that it is impossible to factor the product of two large prime numbers. And there are two different ways to generate large primes: probable primes and provable primes [4]. People now focus on possible primes, which means that if the integer n passes the prime test, it is likely to be prime (or pseudoprime) with high probability, otherwise, it is composite [5].

Moreover, in order to improve the security of RSA algorithm, a method for encrypting and decrypting message text according to ASCII (American Standard Code for Information Interchange) and RSA algorithm is proposed. The message text is converted into binary representation, and the representation is divided into words. section (80's and 1's), apply a bijection function between these byte groups and ASCII character groups, and then use this mechanism to be compatible with the RSA algorithm [6].

Additionally, it is well known that RSA is a widely used public key cryptosystem. The encryption and decryption of RSA is done through various operations of modular exponentiation. But in general, RSA's modulus is computationally expensive. Therefore, it is not enough to just do the basic operations with RSA [7].

III. Problem Identification

A. *Large prime generation*

A main problem of RSA is how to generate and verify large primes (or probable prime with high probability) efficiently. The traditional solution is to generate integers randomly in a specific interval [n/2, n], in which n is a large positive integer ($n \geq 10^{100}$), and then use primality test method to verify whether the integer is prime.

However, according to the prime number theorem [8], let $\pi(n)$ denote the number of primes less than n, then we have

$$\pi(n) \approx \frac{n}{\ln n}(n \to \infty) \quad (1)$$

and which means that when n increases through the sequence of positive integers, the number of primes in the interval [1, n) will tend to equal to $\frac{n}{\ln n}$ approximately.

The prime number theorem implies that if we generate integers in interval [1, n) randomly, it is only a $\frac{1}{\ln n}$ chance of producing a prime number. Moreover, although RSA can provide adequate security there are limitations. So far, prime number generation is still a problem in prime number checking. The mainstream approach to prime number generation is to generate a random number first and then filter it by a prime number checking algorithm. Furthermore, according to prime number theory, the probability that a random number 'n' belongs to a prime number is $\frac{1}{\ln n}$. The prime number tests can be divided into two categories: probability tests and deterministic tests [9]. Although deterministic tests can guarantee the accuracy of prime numbers, the time cost is exponentially increasing. In contrast, probability tests can save more time, with the Miller-Rabin algorithm being one of the probability testing algorithms. Compared to other prime number testing algorithms, the Miller-Rabin algorithm is the most efficient probabilistic prime number test [9].

Therefore, the probability of generating prime numbers in traditional solution is relatively low, so it must take multiple generation and primality tests to obtain a prime, and this consumes many computing resources and time, and led to an inefficient RSA.

B. *Decryption speed*

In RSA, encryption and decryption are performed by modular exponentiation. If the data contains thousands of bits of information, it takes a long time to encrypt and decrypt. Moreover, both encryption and decryption require modular exponentiation of large integers, so the computational cost is very high [10]. Therefore, accelerating the speed of encryption and decryption has always been an exciting topic. In general, shortening the exponent e to a modular exponentiation is the easiest way to speed up encryption, but not so easy for decryption. Because when the modular exponent is shortened, the corresponding decryption exponent d becomes very large. Therefore, it is very important to speed up the decryption time while shortening the modular exponentiation.

C. *Pre-processed Plaintext*

In order to improve the security of the RSA algorithm and to prevent brute force cracking, both the message passed into the RSA algorithm and the result given by the RSA algorithm are encrypted by this method. In other words, this method actually encrypts the message twice in addition to the RSA algorithm. As mentioned above, converting a message to ascii is a simple matter, but how to pre-process the message to cipher text according to the ascii table and how to convert the result generated by the RSA algorithm to plain text according to the ASCII table should be given significant consideration.

IV. Proposed solution and Novelty

A. *Deal with Large prime generation*

In this paper, we proposed a large primes generation method. Assume r be an integer that be chosen randomly in the interval $1 \leq r \leq 10^{100}$, and based on the prime number theorem, then we can get the following corollaries:

Corollary 1: $\Pr\,(r \text{ is prime}) = \frac{\pi(n)}{n} = \frac{1}{\ln n}$

Corollary 2: Pr (r is prime | r is odd) $= \frac{\pi(n)}{n/2} = \frac{2}{\ln n}$

Corollary 3:

$$\Pr(r \text{ is prime} \mid r \text{ is odd and GCD } (r, 3) = 1) = \frac{\pi(n)}{n/2 \bullet (1 - 1/3)} = \frac{3}{\ln n}$$

According to the Corollary II, we can deduce that if we only generate odd integers, the probability of finding a prime number will become $\frac{2}{\ln n}$, and it is doubled compared to before. What's more, the Corollary III shows that if we generate an odd integer, and it and prime 3 are relatively prime, the probability of finding a prime number will become $\frac{3}{\ln n}$. This corollary implies that using small primes to verify the generated numbers is an effective way to improve the probability, and the probability will increase further as more primes are used to verify the generated numbers, and this operation called trial division. Therefore, the probability of generating a prime can be improved by only produce odd integers and then perform trial division, and through these methods, the probability of generating prime numbers became no lower than $\frac{3}{\ln n}$, an increase of at least 200% compared to previous.

The procedure of large prime generation:

1) randomly generate an integer r in the interval [n/2, n] based on the specified large integer n, and perform r = r × 2 + 1 to make sure r is odd.

2) conduct trial division to check r by using all prime numbers less than 100, and set r = r + 2 and repeat step 2 if r failed in checking.

3) Conduct Miller-Rabin checking (prime number checking).

Pseudocodes of Miller-Rabin:

- Let n-1 = 2c m, c mod 2 = 1 (c is odd number, n is the number for primality test)
- Select a random number r satisfying (2 ≤ r ≤ n-1)
- b ≡ am mod n
- if b ≡ 1 (mod n), then n is prime, return true
- for i =0 to c-1: if b ≡ -1 (mod n): then n is prime number, return true else: b ≡ b2 (mod n)
- n is not prime number, return false.

If r failed in the test, set r = r + 2 and return to step 2, and we produce a prime otherwise.

B. *Faster Decryption*

Through THE CHINESE REMAINDER THEOREM, the problem of speeding up the decryption time can be well solved. But the premise of THE CHINESE REMAINDER THEOREM is that the recipient knows the two prime numbers p and q. When the recipient knows these two prime numbers, the module can be calculated.

- If p < q then defines A, which 0 < A < q-1 and A*p ≡ 1 mod q
- Then calculate:

$$d_p \equiv d \ mod \ (p-1), d_q \equiv d \ mod \ (q-1)$$

$$C_p \equiv C \ mod \ p, C_q \equiv C \ mod \ q$$

$$M_p \equiv C_p^d \ mod \ p, M_q \equiv C_q^d \ mod \ q$$

- To decryption: $\widetilde{M} = \left[\left(\left(M_q + q - M_p\right) * A\right) \ mod \ q\right] * p + M_p$ [11] (which is proved already)

Because the bit size of p and q is only half of n, the decryption speed is 4 times faster than ordinary decryption [10].

C. *Pre-processed Plaintext*

The following will illustrate the use of RSA to encrypt and decrypt messages that will be transmitted from the sender to the receiver via a new method associated with ASCII.

Suppose we have the message text and want to encrypt it according to ASCII by the RSA algorithm, first we convert each character of the message text to the relevant digit according to ASCII and then we will apply the RSA algorithm to render in these digits, but when we want to go back to the original message, there will be a problem because in ASCII some characters are represented as two-digit decimal numbers and others are represented as three-digit numbers. For example, the character "a" is represented as 97, "v" as 118 and "%" as 37, so for example "av%" converts to the number "1189737", but based on the number "1189737", even if we forward the digits that divide it, we cannot discover the information associated with ASCII because we do not know the correct division mechanism!" . , Not only that, but we would also like to represent the text in as small a decimal number as possible, so that the RSA algorithm can be applied directly to the whole text in an efficient way.

Proposed solution:

- Encryption:

1) Step1:
Convert every character of message M to binary representation according to Ascii table (every character must consist of 8 of 0s and 1s) from left to right.

2) Step2:
Convert whole representation(S) of M to decimal representation M1

3) Step3:
Apply RSA encrypting function on the number M1 and get numberM2

- Decryption:

1) Step1:
Apply RSA decrypting function on the number M2 and will get number M1

2) Step2:
Convert M1 to binary representation and ensure the number of digits must divided by 8 (we can add zeros to the left)

3) Step3:
Divide that representation into bytes (8 digits) from left to right

4) Step4:
Convert every bytes into character according to Ascii table.

V. Implementation and Testing

We design an RSA system that can choose different functions. When running the code, the user interface is like this:

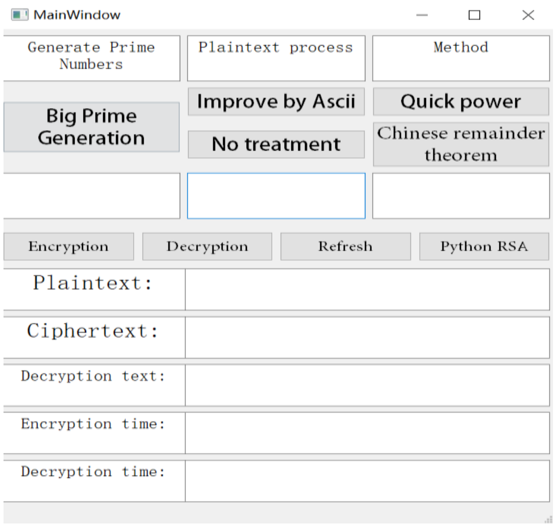

Fig.1. UI of our RSA system

There are five functions to process RSA. "Big Prime Generation" can generate big primes. "Improve by Ascii" can transform the plaintext into Ascii format. "No treatment" means the plaintext is not going to be transformed before encryption. "Quick power" means the RSA uses this function to encrypt and decrypt. "Chinese remainder theorem" means the RSA uses this function to encrypt and decrypt.

Below is the process of using this interface:

1) Input the plaintext and select functions.

2) After choosing the functions and inputting the plaintext, click the "Encryption" button and then the Ciphertext and Encryption time will show up.

3) Next, click "Decryption" button and the Decryption text and Decryption time will show up.

4) Click "Refresh" button to initialize the information.

5) If you click "Python RSA" button, the system will use the python's RSA function.

6) Note: This RSA does not support Chinese plaintext.

As for testing, we select three sizes of generating keys. They are 512, 1024, and 1536. Below is the decryption time of using CRT and quick power. We can see that using CRT to decrypt is faster than using quick power.

We also test the comparison of our RSA and original RSA. Our RSA uses CRT to decrypt and original RSA use normal algorithm to encrypt and decrypt. We can see that our RSA's encryption speed and decryption speed are slightly faster than original RSA in size 512. As the size increases, our RSA's time costs are apparently lower than original RSA's time costs.

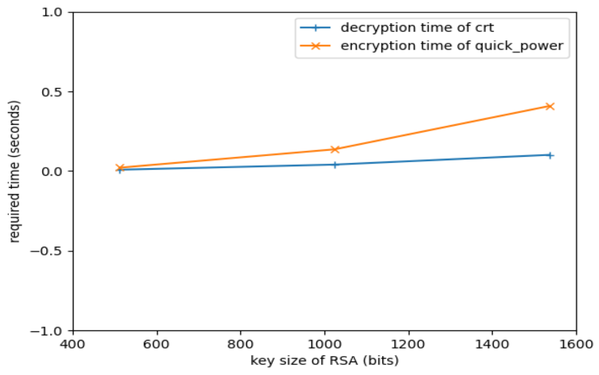

Fig.2. Comparison of CRT and quick power

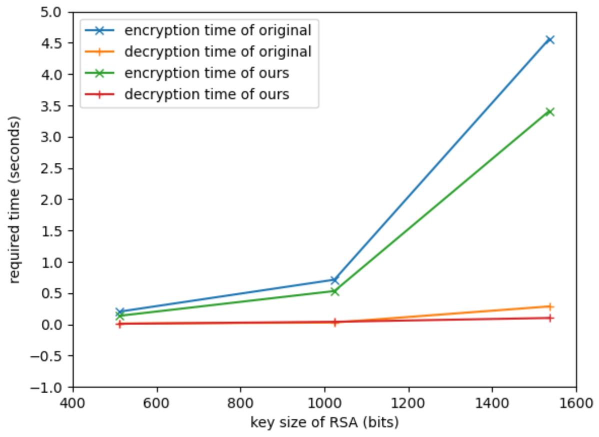

Fig.3. Comparison of running speed

VI. Conclusion

In this report, we analyze recent research on RSA and design an RSA system that can choose different encryption and decryption functions. We also test the encryption speed and decryption speed of 512, 1024, and 1536 key sizes. The result shows that our RSA is faster than the original RSA and using CRT to decrypt is faster than using quick power to decrypt. In addition, as the key sizes increases, the encryption time and decryption time will be larger than before.

In the future, we will add other encryption and decryption functions to the RSA system and finally make comparisons to the python RSA.

MEMBERS CONTRIBUTIONS

In this assignment, each of our group's contributions were weighted at 20% and the following is a breakdown of each individual's specific contribution.

Tianlei Shi (20%): Implement large prime number generation.

Letian Xie (20%): RSA code implementation and algorithms to accelerate modulo operations

Xiaoyue Ma (20%): Conduct controlled variable tests based on what has been implemented and summary the results of the trials.

Zixin Li (20%): Implement the method for accelerating modulo operations.

Longyang Wang (20%): Data pre-processing according to ASCII.

REFERENCES

[1] Kansal S, Mittal M. Performance evaluation of various symmetric encryption algorithms[C] 2014 International Conference on Parallel, Distributed and Grid Computing. IEEE, 2014: 105-109.

[2] A. Chhabra and S. Mathur, "Modified RSA algorithm: a secure approach," IEEE Press, 2011.

[3] ZHOU X, ZHAO F, ZENG D. Research of Asymmetric Encryption Technology [J]. Journal of Sichuan University of Science & Engineering (Natural Science Edition), 2010, 5.

[4] W. Penzhorn, "Fast algorithms for the generation of large primes for the RSA cryptosystem," in Proceedings of the 1992 South African Symposium on Communications and Signal Processing, 1992, pp. 169–172.

[5] J. Gordon, "Strong primes are easy to find," in Workshop on the Theory and Application of of Cryptographic Techniques, 1984, pp. 216–223.

[6] Steef, A., Shamma, M. N., and Alkhatib, A., "RSA algorithm with a new approach encryption and decryption message text by ascii" , arXiv e-prints, 2016.

[7] D. Çalişkan, "An application of RSA in data transfer," 2011 5th International Conference on Application of Information and Communication Technologies (AICT), 2011, pp. 1-4, doi: 10.1109/ICAICT.2011.6110997.

[8] L. J. Goldstein, "A history of the prime number theorem," Am. Math. Mon., vol. 80, no. 6, pp. 599–615, 1973.

[9] C. Duta, L. Gheorghe and N. Tapus, "Framework for Evaluation and Comparison of Primality Testing Algorithms," 2015 20th International Conference on Control Systems and Computer Science, 2015, pp. 483-490, doi: 10.1109/CSCS.2015.153.

[10] W. T. Penzhorn, "Fast decryption algorithms for the RSA cryptosystem," 2004 IEEE Africon. 7th Africon Conference in Africa (IEEE Cat. No.04CH37590), 2004, pp. 361-364 Vol.1, doi: 10.1109/AFRICON.2004.1406693.

[11] JLJ. Quisquater and C. Couwcur, "Fast dcciphment algonlhm for RSA public-key cryptosystcm". Electronic Letters, no. 18, pp. 905-907. 1982