International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 5.22 (SJIF-2017),e-ISSN:2455-2585
International Conference on Smart Cities (ICSC-2019)
Volume-05, Special Issue_March_2019.

A Fast & Efficient Approach for Large Data Encryption & DecryptionUsing RSA Algorithm

Sayyam Jain[1], AlpanaJijja[2]

Ansal University, Gurugram, Haryana - 122003, India [1,2]

Abstract.* With the ever-increasing attacks on data and networks, every organization needs to protect, process, and transmit their data securely through various cryptographic means. One such cryptographic algorithm is RSA (Rivest, Shamir & Adleman). Nowadays, RSA is widely used as it a public – key cryptosystem, which generates 2 keys: Public Key and Private key and requires both to decrypt data. Through this paper, the author has suggested afast and efficient approach of the implementing the algorithm. Furthermore, the performance of both classical and proposed method has been compared using various bit sizes.

Keywords: RSA, AES, Cryptography, Hashmap

1 Introduction

Transmitting data securely is one the biggest challenges today. Data generated by any organization needs to be encrypted to protect it from falling into wrong hands thus preventing its misuse. This can be achieved by using various cryptographic algorithms like RSA & AES. Although being a block cypher, AES offers better security than RSA algorithm, at same bit size. But, even the keys of AES need to be transmitted to decipher the encrypted text (referred as cipher text) into original text (referred as plain text), which are now encrypted using RSA. One of the major advantages of using RSA Algorithm is that it is based on the mathematical problem of factoring large numbers. Exploiting the non - existence of a non – quantum-based algorithm to efficiently factorize large numbers, RSA still remains a vital part of modern cryptography. However, RSA Algorithm are dependent on exponential powers, and is very computationally expensive thus its application on low powered systems to encrypt and decrypt very large messages with larger bit size still remains a challenge [1]. Through this paper, the author has tried to develop a less computationally expensive system, which can be used to encrypt and decrypt very large messages, efficiently through the use of HashMap.

A HashMap works on the principle of hashing. Wherein a Key is stored in a HashMap, thereby generating a Hash for the Key, which is then used to calculate the bucket for the Key, where it would be stored.When a value is to be retrieved from a HashMap, the Hash for Key is calculated again which is again used to calculate the bucket where Key and Value are stored.

Using the same method, the author was able to encrypt 100,000 characters for 2048 bit RSA in less than 7 seconds

2 Literature Review

In 1977, Rivest, Shamir and Adleman (RSA) introduced an important public key crypto-system based on computing modular exponentials. The security of RSA cryptography ultimately lies in the ability of modern devices and computing methods to effectively factorize large integers. [2]

Bahadoriet. al implements a novel approach for secure and fast key generation of the public key cryptographic algorithm of RSA. This method has been implemented on a typicalsmartcard equipped with a crypto-coprocessor and a true random number generator. An efficient method for generating the large random prime numbers is proposed which considerably reduces the total time required for generating a key pair.
There is up to 50% reduction in total generation time compared to the latest reported methods. [3]

H. Ren-Junn, et.al are proposed an efficient decryption method not only based on Chinese remainder theorem (CRT) but also on strong prime RSA criterion. The proposed decryption method only takes 10% computational costs of the traditional decryption method. Effectively reducing 66% computational cost than that of decryption methods based on CRT only. In a word, the speed of our proposed method is almost 2.9 times faster than the decryption method based on CRT. The proposed method enhances the performance of the RSA decryption operation [4].

A. Selby and C. Mitchell proposed two new algorithms that facilitate the implementation of RSA in software. Both algorithms essentially deal with performing modular arithmetic operations on very large numbers, which could be of potential use to applications other than RSA. One algorithm performs modular reduction and the other performs modular multiplication. Both algorithms are based on the use of look-up tables to enable the arithmetic computations to be done on a byte by byte basis. [5]

Shen Guichenget. al did a fast implementation of RSA using Java BigInteger Library. They concluded thatalthough their speed of encryption was very fast, the speed of decryption was slower. The total time we cost in generating two primes, computing N, D, and E, encrypting and decrypting was 7 seconds. [6]

Similarly, another implementation was achieved using the GNU MP Library. Rajorshi Biswas et al. achieved an encryption and decryption time of 0.05 seconds and 0.9 seconds respectively, for 10,000 characters, processing 50 characters at once on 512 bit RSA. [7]Due to the slow nature of RSA Cryptosystem, it is still unsuitable for encryption of large messages. [8-12]

3 Methodology

3.1 Existing System

The RSA Algorithm is based on the mathematical problem of factoring large numbers. [2] This mathematical problem of prime factorisation is limited by non – existence of an efficient algorithm and current processing power, which is exploited by RSA for encrypting and decrypting data as follows [4]:

1. Generate 2 Prime Numbers: p & q. (Eq. 1)
2. Calculate $n = p * q$. (Eq. 2)
3. Calculate $\Phi (n) = (p - 1) * (q - 1)$, where $\Phi (n)$ is kept secret. (Eq. 3)
4. Calculate e such that $1 < e < \Phi (n)$ and $GCD (e, \Phi (n)) = 1$. (Eq. 4)
5. Compute d, modular multiplicative inverse of e (mod Φ (n)) such that it satisfies the relation: $d * e \bmod \Phi (n) = 1$. (Eq. 5)
6. $K_p = (e, n)$ is the public key. (Eq. 6)
7. $K_s = (d, n)$ is the private key. (Eq. 7)
8. Let m is a plaintext, Encrypted Message (c) is : $m^e \bmod n$. (Eq. 8)
9. Letting c be encrypted text, deciphered text is : $c^d \bmod n$. (Eq. 9)

3.2 Proposed Approach

3.2.1Key generation

Keys generation in RSA algorithm is the most essential step in data encryption. For a given 'N' - bit RSA, 2 prime numbers (p & q) are generated of N/2 bits (Eq. 1).
The generated prime numbers p & q are of bit length, half the size of public and private keys. These values are multiplied to be used as modulus for both public and private keys (Eq. 2).

Further, totient is calculated by subtracting 1 from both prime numbers (p & q) and multiplying them (Eq. 3).

To determine the value of coprime, a random number is generated with bit length the size of Key, and is incremented until the greatest common divisor of co-prime and totient is 1 (Eq. 4).

Finally, the modulo Inverse of coprime and totient is calculated (Eq. 5). Hence, public and private keys for the user are generated successfully (Eq. 6 & Eq. 7).

3.2.2Data Encryption

Given the public and private keys, the plaintext is treated as a collection of alphanumeric characters. Each character in the file is then converted to integers by simply mapping it to its ASCII code. Processing the characters and converting them into ASCII form as 'm', encryption is achieved by computing m ^ e mod n (Eq. 8).

Since, text can often be large and there might be recurring characters. In such cases, encrypting same character again and again is expensive in terms of the time required to encrypt. Hence, instead of encrypting the same characters again and again, a HashMap is maintained, which stores characters as key and their encrypted value as Value.This reduces their time complexity to O (1) in case they appear again from ASCII pool which contains 95 printable characters from 32 to 126.When each character is encrypted, the results are combined to form the output.

3.2.3 Data Decryption

Given the public and private keys, in the encrypted file, the function processes each encrypted integer by computing the value of c ^ d mod n (Eq. 9).This time again, instead of decrypting the same integers again and again, we maintain a HashMap which stores integers as key and their decrypted Character Value as Value. This reduces their calculation time to O(1) in case they appear again from ASCII pool which contains 95 printable characters from 32 to 126. The decryption of the encrypted file is complete, once all the integers (cipher text) in the encrypted file are processed and are combined which is then converted to a single stream of alpha-numeric characters and written to a file.This completes the decryption process.

4 Observations

Al the time duration recorded below have been measured on an Intel® Core™2 Duo Processor @ 2.93Ghz, on a GNU/Linux platform (kernel 4.10.0).

4.1 Key Generation Time

Below are the time taken for key generation for different bit sizes. Bit size of key used is twice of generated prime number. The times are averaged over 3 samples and are same for both classical and proposed algorithm

Table 1. Time duration to generate public and private keys

Bit Size (Key)	Bit Size (Prime No.)	Key Generation Time (seconds)
128 Bit	64 Bit	0.023
192 Bit	96 Bit	0.029
256 Bit	128 Bit	0.038
512 Bit	256 Bit	0.059
768 Bit	384 Bit	0.092
1024 Bit	512 Bit	0.122
1280 Bit	640 Bit	0.183
1536 Bit	768 Bit	0.188
1792 Bit	896 Bit	0.380
2048 Bit	1024 Bit	0.541

4.2 Data Encryption/ Decryption Time

Below are the times duration taken for encrypting and decrypting a file with 100,000 characters. The timedurations are recorded for both classical RSA algorithm and the proposed RSA algorithm.

Table 2. Time duration to Encrypt and Decrypt a file with 100,000 characters using Classical RSA Algorithm

Bit Size (Key)	Encryption Time (seconds)	Decryption Time (seconds)
128 Bit	4.423	3.092
192 Bit	12.305	11.409
256 Bit	14.028	13.080
512 Bit	94.418	101.023
768 Bit	190.910	189.342
1024 Bit	418.310	421.657
1280 Bit	741.592	739.675
1536 Bit	1233.232	1231.452
1792 Bit	1946.524	1947.906
2048 Bit	3316.777	3314.633

Table 3. Time taken to Encrypt and decrypt a file with 100,000 characters using Proposed RSA Algorithm

Bit Size (Key)	Encryption Time (seconds)	Decryption Time (seconds)
128 Bit	0.404	0.398
192 Bit	0.484	0.512
256 Bit	0.612	0.656
512 Bit	1.224	1.203
768 Bit	1.765	1.954
1024 Bit	2.715	2.947
1280 Bit	3.591	4.334
1536 Bit	4.230	6.051
1792 Bit	5.418	6.686
2048 Bit	6.632	8.679

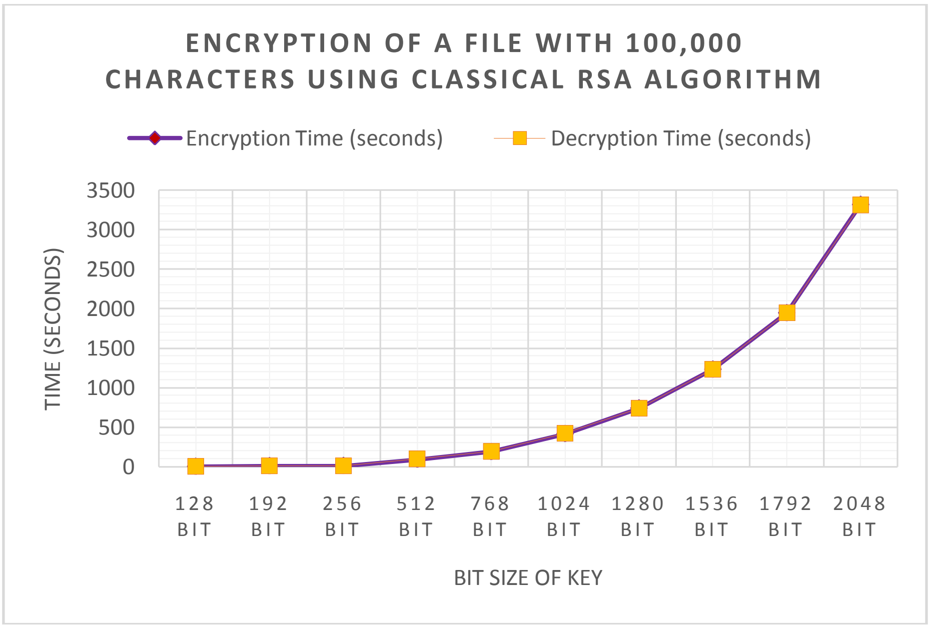

Fig. 1. Time taken to Encrypt a file with 100,000 characters using Classical RSA Algorithm

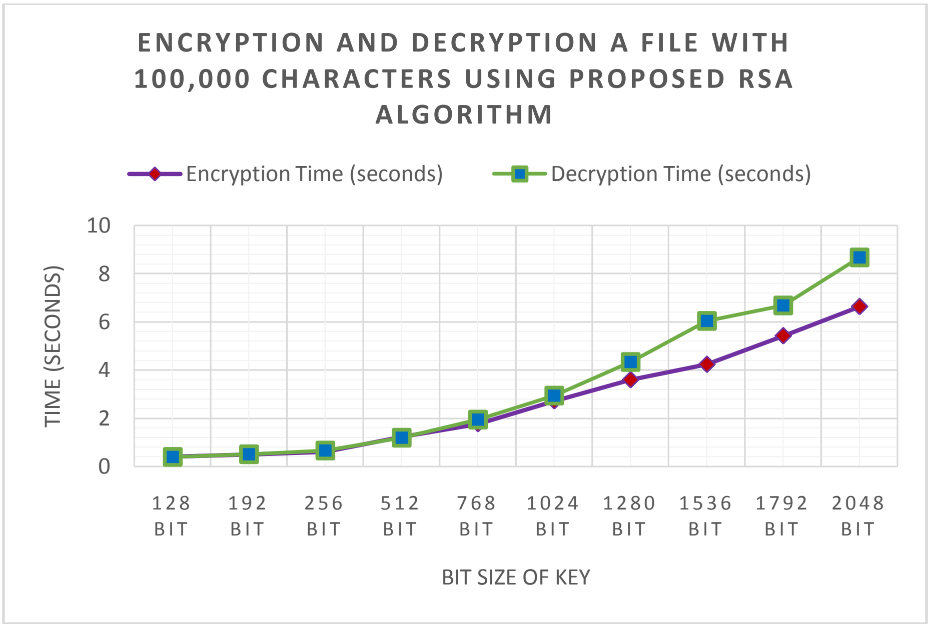

Fig. 2. Time taken to Encrypt and Decrypt a file with 100,000 characters using Proposed RSA Algorithm

5 Results & Analysis

1. From the above results, we infer that, the time taken to encrypt and decrypt the file increased exponentially when classical algorithm was applied (Fig.1) but increased linearly (Fig.2) when proposed RSA algorithm was used.
2. In addition, the files encrypted and decrypted using proposed algorithm were processed more than 450 times faster than classical one (for 2048 bit) (Table 2. & Table 3.)

Hence, using the above approach, large texts can now be encrypted as well as decrypted in a very short amount of time duration.

6 Conclusion & Future Scope

Through this paper, the author has attempted to reduce the time taken to Encrypt and Decrypt a file using Hashmap on RSA Algorithm, which allows larger messages with keys of higher bit size to be encrypted which were not feasible earlier due to time and processing power constraints.

Further, mathematical advancements in future along with the proposed and efficient approach may be applied together to reduce computation times, not only for RSA Algorithm, but also other cryptosystems which rely on Byte by Byte processing.

7 References

1. Kleinjung; et al. (2010-02-18). "Factorization of a 768-bit RSA modulus", International As-sociation for Cryptologic Research.
2. R. L. Rivest, A. Shamir and L. Adleman “A method for obtainingdigital signatures and public-key cryptosystems” Communications ofthe ACM, vol. 21, pp. 120-126, 1978.
3. M. Bahadori, M. R. Mali, O. Sarbishei, M. Atarodi and M. Sharifkhani “A novel approach for secure and fast generation of RSA public and private keys on SmartCard” NEWCAS Conference (NEWCAS), 2010 8th IEEE International, 2010, pp. 265-268
4. H. Ren-Junn, S. Feng-Fu, Y. Yi-Shiung and C. Chia-Yao “An efficient decryption method for RSA cryptosystem” Advanced Information Networking and Applications, 2005 (AINA 2005). 19th International Conference on, 2005, pp. 585-590 vol.1
5. A. Selby and C. Mitchell “Algorithms for software implementations of RSA” Computers and Digital Techniques, IEE Proceedings E, vol. 136, pp. 166-170, 1989.
6. Shen Guicheng, Liu, Bingwu and Zheng, Xuefeng, “Research on Fast Implementation of RSA with Java”, International Symposium on Web Information Systems and Applications (WISA’09), Academy Publisher, Nanchang, China, 2009, pp. 186-189.
7. Rajorshi Biswas, ShibdasBandyopadhyay, Anirban Banerjee, “Fast Implementation of the RSA Algorithm Using the GNU MP library”, in Proc. National workshop on cryptography, (2003), pp. II-30.1 to II-30.15
8. Sami A. Nagar and SaadAlshamma “High Speed Implementation of RSA Algorithm with Modified Keys Exchange”, 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), pp.639-642, March 2012.
9. J. Joshi, et al. “Network Security” Morgan Kaufmann, 2008
10. W. Stallings “Network security Essentials: Applications andStandards” Pearson Education India, 2000.
11. W. Stallings “Cryptography and network security vol. 2” prentice hall, 2003.
12. W. Stallings “Network and internetwork security: principles and practice” Prentice-Hall, Inc., 1995.