Analysis and Performance Overview of RSA Algorithm

Dr. Himanshu Arora¹, Gaurav Kumar soni² and Deepti Arora³

¹Department. of Computer Science & Engineering, Arya Group of Colleges, Jaipur, Rajasthan, India
²Department of Electronics & Communication, Arya College of Engineering & Research Centre, Jaipur, Rajasthan, India.
³Department of Mathematics, Arya Engineering College, Jaipur, Rajasthan, Indian

Abstract—RSA is one of most popular public key cryptographic algorithm. It is an asymmetrical cryptosystem as it uses a pair of keys - private key and public key. Together, they are referred as key pair. Encryption and verification of signature is done by public key whereas decryption and signing is performed using private key. The keys are generated by a common process, but they cannot be feasibly generated from each other. In RSA if one can factor modulus (n) into it’s prime numbers then he can get the private key too. The security of the RSA system is based on the assumption that factoring of a large number is difficult. But the advancement in the technology (both in hardware as well as in software) puts the advent of new, highly secure, efficient, platform-independent, and scalable solutions. In this paper described the performance analysis of RSA algorithm which is based on the various parameters.

Keywords— RSA, Message Digest, Random Number, Factorization, Encryption, Decryption

I. INTRODUCTION

Public-key cryptography is also called asymmetric. It requires the use of a private key (a key that only its owner knows) and a public key (a key that both know). Public key cryptography is a fundamental technology and widely used throughout the world. It is the approach used by many cryptographic algorithms and commonly used for the distribution of software, financial transactions and in other critical security areas where it is important to protect against counterfeit and alterations.

RSA (named for Rivets, Shamir and Adelman, who first described it publicly) is the first known algorithm for signing and encryption, and was one of the first major advances in public-key cryptography. It uses a pair of keys, one of which is used to create the signature in such a way that it can only be verified with the other key of the pair [5].

The keys are generated through a common process, but they cannot be generated in a viable way among them. The security of RSA depends solely on finding the prime factors that are used in the process of creating and verifying signatures and is based on the assumption that factoring a large number is difficult. “Multiplying two large prime numbers is a unidirectional function It is easy to multiply the numbers to obtain a product, but it is extremely difficult to factor the product and retrieve the two large prime numbers that have been multiplied previously Factoring problem.

II. MATHEMATICAL BACKGROUND OF RSA

RSA Encryption is a public and digital key encryption that offers signatures. The RSA algorithm is based on mathematical fact that it is easy to find and multiply two large primary numbers together, but it is extremely difficult to factor a product and restore two large primes that have been multiplied in the past. Reproduction can be calculated at a polynomial time, since the factoring time can grow significantly to the size of the proportional figure. There are two types of keys: private key and public key. Together, it is known as the key pair. The encryption and signature verification is done by public key, while the decryption and signature are done with a private key. The use of RSA serves as the public key of the unit n where n is the product of two large prime numbers p and q, and the relativamente relays are the necessary codes for P1 (P1). For two numbers to be relatively relative to each other, they are not necessarily other common factors of one. This can be efficiently in large integers calculated using the Euclid algorithm [1 and 3] for calculating GCD [e, r (1) x (p -1)] = 1. The special key includes, what are the decoding factors, The inverse multiplier is any of the unit and (P1) o (P1). On decoding the index can be using a calculated Euclidean extended algorithm, extending GCD [H, Y (1) x (P 1)]. The sender takes numerical representation of the letters in the message, such ASCII and then divided into blocks where the numbers are smaller than the n unit.
The blocks are raised to zero. To decrypt, the message received appears to be decrypted. The public has its own codes, but to obtain the decode, the unit n must be taken into account in p and q, which is used for the obtainer (P1) o (P1). Eleven (P1) x Uncounted (P1), and decoding of the indices can be formulated.

In the algorithm, text messages are divided into a series of characters in this block of fixed-size characters and this has an internal representation of Unicode. Most messages are not divided into blocks of fixed integers, and usually a short block at the end is still present. Encryption requires blocks of static integers. Filling is the way to solve a problem where the last block is filled with the normal pattern, r. Zeros, ones, alternates and zeros - to make it a complete block. To remove padding after decoding, you must add the number of padding bytes as the last byte of the last block. For example, suppose that the block size is 64 bits and the last block consists of 3 bytes (24 bits). Five bytes of padding is required to make a last 64-bit block. Add 4 bytes of zeros and the last bytes with the number 5. After decryption, delete the last five bytes of the last decryption block. In order for this method to work properly, the last block of each message must be partially or fully filled [3].

The RSA algorithm works the following way. You find the first two initial numbers and generate a pair of keys using two prime numbers of those. Then, the signature creation is done and the signature is verified using the key pair.

- p and q are different cousins
- \( N = p \times q \)
- Find e, d such that \( e \times d = 1 \mod (p-1) \) (q-1)
- Public key: \( = (n, e) \)
- Private key: \( = (n, d) \)
- Creation of the signature: \( S(m) = md \mod n = S \)
- Verification of signature: \( V(S) = \text{Mod n} = m \)
- So, for RSA to work, it must have the property:
- \( (md) e = m \mod n \ (3.4) \)
- It must be shown that the above equation is true. If the above equation is shown, then it can be said that the RSA algorithm really works.

### A. Computational complexity of the RSA algorithm

The computational complexity of the RSA encryption and decryption of a single block of n bits is approximately \( O(n^3) \), with n indicating both the length of the block and the length of the key (exponent and module). This is due to the complexity of the multiplication that is \( O(n^2) \) and the complexity of the exponentiation as \( O(n) \) when the square and multiplied algorithm is used. Although multiplication and exponentiation algorithms exist that have less asymptotic complexity, they have a limited technical interest when \( n < 1000 \) is assumed. If the length of message m is sufficiently greater than the length of block n, the number of steps required to process a single message bit is of complexity \( O(n^2) \).

### III. PERFORMANCE ANALYSIS OF RSA ALGORITHM

The RSA algorithm has some important parameters that affect its level of security. Here it is shown that increasing the length of the prime number plays an important role in increasing the complexity of breaking it down into its factors. When the duration of the message is changed, the length of the encrypted message changes proportionally, therefore, larger fragments are selected to obtain a larger encrypted message to increase the security of the data in use.

Any practical implementation of the RSA cryptosystem would involve working with large integers (1024 bits or more in size). There are several libraries that deal with large integers such as the Big Integer library (Java) [15], the GNU MP Arbitrary Precision library (C / C ++) [17] and the Open SSL encryption library (C / C + +) [17]. As the application is developed in JAVA, the Big Integer library is used. The Big Integer library provides operations for modular arithmetic, GCD calculation, primarily tests, master generation, bit manipulation and some other miscellaneous operations.

The evaluation time is a machine-dependent task that must be implemented in a particular system. Once the system configuration is changed, the evaluation time will also change accordingly, however, in this work, the following system configuration is used.

- Operating System : Windows XP Professional (5.1, Build 2600) Service Pack 2
- Processor : Intel Pentium Dual CPU E2200 @ 2.20GHz (2 CPUs) Memory : 1024MB RAM.

#### A. Changing the prime number \((p & q)\) Size:

The value of prime numbers p and q will affect the other parameters as shown in Table 3.2.(taking the size of chunk 128 bit and size of public key 128 bits).
TABLE I
Effect of changing the size of Prime Number on Key Generation, Encryption time and decryption time, taking size of chunk 128 bits, size of Public key 128 bits.

<table>
<thead>
<tr>
<th>Prime number (p &amp; q) size (Bit)</th>
<th>Key Generation Time (ms)</th>
<th>Encryption Time (ms)</th>
<th>Decryption Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>109</td>
<td>78</td>
<td>156</td>
</tr>
<tr>
<td>256</td>
<td>124</td>
<td>266</td>
<td>719</td>
</tr>
<tr>
<td>512</td>
<td>407</td>
<td>875</td>
<td>4812</td>
</tr>
<tr>
<td>1024</td>
<td>1218</td>
<td>3235</td>
<td>35531</td>
</tr>
</tbody>
</table>

As the security of the RSA algorithm [5] depends on the prime factors p and q, increasing the size of the prime number provides more security. It is shown in Table 3.2, as the size of the prime numbers increases, the key generation time, the encryption time and the decryption time also increases. Therefore, increasing the size of the prime number increases security but decreases the speed of the process. The following figures show the effects of the size of the prime number on the key generation time, the encryption time and the decryption time.

Figure 1: Prime Number v/s Key generation time, Encryption time and Decryption time, taking size of chunk 128 bits, size of Public key 128 bits.

B. Changing the public key length
Changing the maximum length of the public key length (e-bit) mainly affects the key generation time and encryption time. As shown in Table 2, the size of the public key does not affect the decryption time. The length of this public key (e) increases proportionally with the increase of the bit.

TABLE II
Effect of changing the Public key size on Key generation time, Encryption time and Decryption time, taking size of Prime number 1024 bits and size of Chunk 128 bit.

<table>
<thead>
<tr>
<th>Public Key (e) size (bit)</th>
<th>Key Generation Time (ms)</th>
<th>Encryption Time (ms)</th>
<th>Decryption Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1218</td>
<td>3235</td>
<td>35531</td>
</tr>
<tr>
<td>256</td>
<td>1234</td>
<td>5343</td>
<td>35469</td>
</tr>
<tr>
<td>512</td>
<td>1906</td>
<td>9782</td>
<td>35672</td>
</tr>
<tr>
<td>1024</td>
<td>9859</td>
<td>18703</td>
<td>35578</td>
</tr>
</tbody>
</table>

Figure 2: Public Key Size v/s Encryption Time & Decryption Time, taking Size of Prime Number 1024 bits and Size of Chunk 128 bits.
TABLE III

<table>
<thead>
<tr>
<th>Chunk Size (Bit)</th>
<th>Encryption Time (ms)</th>
<th>Decryption Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>18703</td>
<td>35578</td>
</tr>
<tr>
<td>256</td>
<td>9187</td>
<td>17937</td>
</tr>
</tbody>
</table>

C. Changing the length of the message to be processed (chunk size):

The Size of Chunk is the number of characters to process at the same time, in encryption or described. Here the massage is divided in sub blocks each of length equal to chunk size. To As Sufficiently the importance of this parameters, the massage will be taken as sufficiently possible and the size may vary in the encryption/decryption process. The effect of changing the size of chunk gives the result as shown in TABLE 3. Obviously, the value of first number s (P & Q), module size (N), Public Key (E) and Private key (D) they are independent of portion size.

The simulation Result of table 3 demonstrate that as the increase of the reference size is increased, the encryption time and the decrease of the time of decryption and the size of the encryption massage is increased, also increase the security.

IV. CONCLUSION

The security of the RSA algorithm lies in factorization of large prime numbers. factorization problem of mathematics that indicates that given a very large number, it is impossible in the current aspect to find two prime numbers whose product is the given number. As the number increases, the possibility of factorizing the number decreases. The security of the RSA algorithm lies in the fact that there is no good way of factoring large prime numbers, but the advance in technology (both hardware and software) uncovers a new, highly secure and efficient platform Independent and scalable solutions. This work explores RSA cryptographic algorithms in details for their drawbacks.

References


