Contemporary RSA- 1024 Cryptosystem: A Comprehensive Review Article

Sanjeev Karmakar
Bhilai Institute of Technology, Durg, Bhilai House, 491001, Chhattisgarh, India

Siddhartha Choubey
Shri Shankaracharya Technical Campus / Shri Shankaracharya group of Institutions, Bhilai, Durg, India

ABSTRACT
Security strength of RSA Cryptography is an enormous mathematical integer factorization problem. Deducing the private key ‘d’ from its equation e • d $\equiv$ (1 mod $\psi$) where $\psi = (p-1)$, (q-1), and $n \in I^{+}$, such that $n = p \cdot q$; is a world wide effort. This paper introduced very significant integer factoring algorithms such as trial division, p- method, ECM, and NFS and effort to factor RSA-150 composite number ‘n’ of 512 bits by using NFS. It is found that the 512 bit RSA number may be believed to safe from the intruder. However, this system is slow for large volume of data. The computation of c $\equiv$ me mod n required O (size (e) (size n)) and space O(size c + size n). Similarly, decryption process also has required O (size (d) (size n)) and space O (size d + size n). Java ‘BigInteger’ class is introduced to overcome this shortcoming and successfully applied is presented through this paper.

Keywords
RSA, RMI, Cryptography, Encryption, Decryption, Network, Security, RSA-1024, NFS, ECM

1. INTRODUCTION

Cryptography (from the Greek kryptós lógos, meaning “hidden word”) is the discipline of cryptography and cryptanalysis combined. To most people, cryptography is concerned with keeping communications private. Indeed, the protection of sensitive communications has been the stress of cryptography throughout much of its history [1]. Cryptography has two phases’ encryption and decryption.

Encryption is the transformation of data into a form that is as close to impossible as possible to read without the suitable knowledge (a key). Its reason is to ensure privacy by keeping information hidden from anyone for whom it is not proposed, even those who have access to the encrypted data. Decryption is the reverse of encryption; it is the transformation of encrypted data back into an original form. Encryption and decryption generally require the use of some secret information, referred to as a key. For some encryption mechanisms, the same key is used for both encryption and; for other method, the keys used for encryption and decryption is different. There are two types of cryptosystems: secret-key and public-key cryptography. In secret-key cryptography, also referred to as symmetric cryptography, the same key is used for both encryption and decryption. The most popular secret-key cryptosystem in use today is the Data Encryption Standard (DES). In public-key cryptography, each user has a public key and a private key. The public key is made public while the private key remains secret. Encryption is performed with the public key while decryption is done with the private key. The RSA public-key cryptosystem is the most popular form of public-key cryptography. RSA stands for Rivest, Shamir, and Adleman, the inventors of the RSA cryptosystem.

The Digital Signature Algorithm (DSA) is also a popular public-key technique although it can only be used for signatures, not encryption. Elliptic curve cryptosystems (ECCs) are cryptosystems based on mathematical objects known as elliptic curves. Elliptic curve cryptography has been gaining in popularity recently. Lastly, the Diffie-Hellman key agreement protocol is a popular public-key technique for establishing secret keys over an insecure channel [2].

Surveys by Rivest [3] and Brassard [4, 5] form an excellent introduction to modern cryptography. Some textbook treatments are provided by Stinson [6] and Stallings [7], while Simmons provides an in-depth coverage of the technical aspects of cryptography [8]. A comprehensive review of modern cryptography can also be found in Applied Cryptography [9]; Ford [10] provided detailed coverage of issues such as cryptography standards and secure communication.

This paper explained one of the most important public key cryptography called RSA cryptography and its practical execution by using Java in the following sections intended to provide more practical observation of this cryptosystem for the reader. Theoretical concept of RSA cryptography, example of RSA-1024 cryptography, its implementation RSA-1024 cryptography using java, security strength, shortcoming, research scope, and finally conclusions have been discussed.

1.1 RSA Cryptography

The RSA cryptosystem is a public-key cryptosystem that offers both encryption and digital signatures (authentication). Ronald Rivest, Adi Shamir, and Leonard Adleman developed the RSA system in 1977 [11]; RSA stands for the first letter in each of its inventors’ last names. Its security is based on the intractability of the integer factorization on the problem [1, 2] will be discussed in this chapter afterward sections. In RSA algorithm encryption and decryption process is depicted in the following Fig.1 (a,b, & c) and its processes is explained in the following section.

To generate public key encryption RSA algorithm selects two large prime numbers p, q such that n=p • q and $\psi = (p – 1)$, (q – 1), and $\exists e \in I^{*}$, $1<e<\psi$, such that gcd (e, $\psi$) = 1. After that algorithm used Euclidean algorithm to compute the unique integer d, $1<d<\psi$, such that $e \cdot d \equiv (1 \text{ mod } \psi)$. Through that e public key (n, e) and private key is d. these e and d is RSA key generation are called the encryption exponent and the
decryption exponent respectively, while \( n \) is called the modulus.

To send message \( (m) \), where \( m \in \mathbb{Z}^+ \), \( m \in \mathbb{Z} [0, n - 1] \) interval, compute cipher text \( c \equiv m^e \mod n \) at sender end this task is known as encryption. At the receiver and again compute \( c^d \mod n = m \) to get the original message \( (m) \), this task is known as decryption. To proof decryption task, since \( e \cdot d \equiv 1 \mod \psi \), \( \psi \equiv \phi(n) \), where \( \phi(n) = (p - 1)(q - 1) \). Now if \( \text{gcd}(m, p) = 1 \) then by Fermat’s theorem \( m^{p-1} \equiv 1 \mod p \).

On the other hand, if \( \text{gcd}(m, p) = p \), then this last congruence is again valid since each side congruence to 0 modulo \( p \).

Hence in all cases
\[
m^{ed} \equiv m \mod p
\]
By same argument \( m^{ed} \equiv m \mod q \).

Finally, since \( q \) and \( p \) are distinct primes, it follows that \( (m^e)^d \equiv m \mod n \); and hence
\[
ed \equiv (m^e)^d \equiv m \mod n
\]

shows how server will send unique public key for each connected clients and maintain unique private key for each client as well. At this juncture, three clients are connected with the server. Server will generate three set of public and private key for each connected clients. It is cleared that, each client have their own pair of key for the secure communication. This feature produces security potency of the system as well. In the subsequent sections of the chapter, detail design and Java code of RSA-1024 cryptography system over a multi-client server network is discussed. Readers of this book may use this code and exhibit the system in their machine or LAN. In the next section, an example of RSA-1024 cryptography is explained.

1.2 Paradigm of RSA -1024 Cryptography

RSA-1024 has 1024 bits (309 decimal digits) have following steps in the server as well as client side.

Step-I: Server randomly generates two large prime number say \( p \) and \( q \) like
\[
(p) = 197235498938199875492256271561672

(q) = 9435422515733970967431625949240939

\]

Step-II: Server computes \( n = p \cdot q \) (called RSA Modulus \( n \) is a 1024 bits number has 309 decimal digits)
\[
(n) = 6191235984721236967128806246260138

Step-III: Server computes \( \psi = (p-1)(q-1) \)
\[
(\psi) = 17192757119189123047484712102553171

Step-IV: Server computes \( \psi = (p-1) \cdot (q-1) \)
\[
(\psi) = 6191235984721236967128806246260138

Step-V: Server computes \( d \) by relation \( d = 1 \mod \psi \)
\[
d = 6191235984721236967128806246260138
Step IV: Sets public key (e, n) like (3, n) and private key (d, n)
Step V: Sends (e, n) to the client for encryption and maintains (d, n) for decryption.

Now consider following plaintext is to be sent from the client

> Dear Aditya

RSA cryptosystem is based on Integer factorization problem thus produced high security however slow for large data.

Then message (m) can represent as big integer form

\[
m = 19758115125724582576198000160674597125830169667617460323700570937224305196374821769819490568588874154753462907279950510449042856821769393224914094779898878520658882083203243848960352963245530601865612407734034727694349660868721362455283422285429118942947515730286936015560470540526195414944963119855824109045264180220720925192068676942008466666872796524.
\]

Encryption - Client computes cipher text\((C = m^e \mod n)\) and sends to the server:

\[
8551146655827907853971989413138402658275762711728119244056939508344326296821353505391879999991853526444256819661609282584274642638671442627409493562235722430512750635171576727275290769373765840822052239103000167725187910286440768639641798582591897729695509999305653990347778404072093912855418839373258758465111412362631740342476297464736175447314274005658082468579870830840092041813685219727678442818385702518102099707553553827777408363223162044122005339249049204062233870198908544236959758780210470582787642204649922083818888064564459
\]

Decryption - Server computes original message \((m)\) by using \((m = C^d \mod n)\) as

\[
(44322696821353505391879999991853526444256819661609282584274642638671442627409493562235722430512750635171576727275290769373765840822052239103000167725187910286440768639641798582591897729695509999305653990347778404072093912855418839373258758465111412362631740342476297464736175447314274005658082468579870830840092041813685219727678442818385702518102099707553553827777408363223162044122005339249049204062233870198908544236959758780210470582787642204649922083818888064564459
\]

2. IMPLEMENTATION OF RSA-1024 BY USING JAVA

Java BigInteger provided analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations. Modular arithmetic operations are provided to compute residues, perform exponentiation, and compute multiplicative inverses. It has been found the implementation of RSA algorithm can be easily obtained from following code segment. Detail of implementation procedure as well as code is given in the subsequent sections of this chapter.
3. SECURITY STRENGTH OF RSA CRYPTOGRAPHY

Security strength of RSA is enormous mathematical integer factorization problem. Deducing the private key ‘d’ from its equation \( e \cdot d \equiv 1 \pmod{\phi(n)} \) where \( \phi(n) = (p-1) \cdot (q-1) \), \( p \cdot q \in \mathbb{I}^+ \), such that \( n = p \cdot q \) (p and q is large distinct prime number) is a world wide effort. It required more processing power along with storage. This paper introduced significant integer factoring algorithms number field sieve (NFS) and its effort to factor while The RSA-150 (512 bit) composite number ‘n’ have been considered. Other significant mathematical methods/algorithm also has been discussed for assessment through survey of Mathematical Review 94-2008A60. It has been found that the 512 bit RSA number may be believed to safe from the intruder.

4. POSSIBLE ATTACK ON RSA

If the advisory is able to factor the public modulus ‘n’ of same entity A, then the adversary can compute \( \psi \) and then, using Extended Euclidean algorithm deduce the private key d from \( \psi \) and the public exponent ‘e’ by solving \( ed \equiv 1 \pmod{\psi} \). This constitutes a total break of the system. To guard against this, A must set p and q so that factoring ‘n’ is a computationally infeasible task [12].

4.1 Methodology

One of the importation applications of integer factorization is RSA public key cryptosystem. The security of cryptosystem depends on the intractability of factoring integers [13]. The integer factorization is one of the problems that have been long considered in the world of the number theory [12]. In the last few decades, together with the rapid progress of computer technology, methods for factoring integers efficiently were studied, and as a result some algorithms were invented through review. The major methodology or algorithm through which ‘d’ can deduce from \( e^d \equiv 1 \pmod{\psi} \), if we capable to factor n. In fact it is strength of RSA security; nobody can say which algorithm is best for factor ‘n’ on the word of computational complexity.

Dixon’s method based on finding a congruence of squares. Format’s factorization algorithm finds such congruence by selecting random or pseudo-random x value and hoping, one satisfied the congruence. \( x^2 \equiv y^2 \pmod{n} \); \( X = \pm Y \); values will take an impractically long time to find a congruence of squares. In Pollard’s \((p-1)\) algorithm meaning that it is only suitable for integers with specific types of factors. The algorithm is based on the insight that number of the form \((a^n-1)\) tends to be highly composite when b is itself composite. Since it is computationally simple to evaluate number of this form in modular arithmetic, the algorithm allows one to quickly check many potential factors with greatest efficiency. In particular the method will find a factor \( \rho \) if \( b \) is divisible by \( \rho - 1 \). When \( \rho - 1 \) is smooth (the product of only small integer) then this algorithm is well suited to discovering the factors.

The trial division is the simplest and easiest to understand of the integer factorization algorithm. Given a composite integer \( n \), trial division consists of trial dividing n by every prime number less than or equal to \( \sqrt{n} \). If a number is found which divides evenly into n, a factor of \( n \) has been found. Trial division is guaranteed, to find a factor of \( n \), since it checks all possible prime factors of ‘n’. Then if the algorithm is fail, it is proof. That \( n \) is prime. In worst case, trial division is very
inefficient algorithm. It starts from 2 and work up to the \( \sqrt{n} \), the algorithm requires \( (\sqrt{n})^2 \) trial divisions, where \( n \) is the number of primes less than \( n \).

NFS (Number field sieve) algorithm uses four main steps; polynomial selection, sieving, linear algebra and square root. In polynomial section step two irreducible polynomial \( f_1(x) \) and \( f_2(x) \) which are commonly root m (mod N) selected having as many as practically possible smooth values over given factor base. In the sieving steps, which is by far the most time consuming steps of NFS pair \((a,b)\) are found with gcd \((a,b)\) = 1 such that both

\[
 b^{\deg(f_1)} f_1(a/b) \quad \text{and} \quad b^{\deg(f_2)} f_2(a/b)
\]

are smooth over given factor base i.e., factor completely over the factor bases. Such a pair \((a,b)\) called relation. The purpose of this step is to collect so many relations that several subsets \( S \) of them can be found with the property that a product taken \( S \) yield on expression of the form \( X^2 = Y^2 \mod N \) for approximately half of these subsets, computing gcd \((X-Y, N)\) yields a non-trial factor of \( N \) (if \( N \) has exactly to distinct factors). We only detailed sieving steps in NFS here [14-16].

ECM (Elliptic curve method) by Lenstra, and Takayuki [17,18] makes use of property of Groups \((G)\) of points on the elliptic curve to find factor of composite number \( n \). ECM can find relatively small factors \((< 50 \text{ digits})\) of so large integers that NFS cannot treat, because ECM does not have a limitations with respect to \( n \). For this reason ECM is still an important technique for factorization at the present time [19,20]. Lenstra, [21], and since than various improvements has been worked out. The time complexity of some algorithm is dependent on the given composite number \( n \), and that other is dependent on the smallest prime factor \( p \) of \( n \). As per algorithms discussed above, the average (or worst time) complexity is shown below [21, 22].

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFS</td>
<td>( L_{\text{nf}}(1/3,1.901) )</td>
</tr>
<tr>
<td>Trial Division</td>
<td>( O(\rho) )</td>
</tr>
<tr>
<td>( \rho ) method</td>
<td>( O(\sqrt{n}) )</td>
</tr>
<tr>
<td>( \rho ) -1 method</td>
<td>( O(\rho^2) )</td>
</tr>
<tr>
<td>ECM</td>
<td>( L_{\text{ECM}}(1/2,1.414) )</td>
</tr>
</tbody>
</table>

Here \( \rho' \) is the largest prime division of \( \rho -1 \). Here the function \( L_{\rho}(1,c) \) defined as follows:

\[
 L_{\rho}(1,c) = \exp(c + O(1)(\log x)^{1/(\log \log x + 1}))
\]

And an algorithm that has the time complexity \( L_{\rho}(1,c) \) for some \( c \) and \( x \) (where, \( n \) is the input number) is called a sub exponential time algorithm. Note that \( L_{\rho}(0,C) = O(\log n)^c \) (polynomial time) and \( L_{\rho}(1,c) = O(n^c) \) (exponential time with respect to the input length \( \log n \)) [21]. It is concluded that the best-known algorithm for factoring integer is NFS, asymptotically and practically for very large composite number (over 150 digits). The sieving phase that search fixed set of prime number for candidate that have a particular algebraic relationship, modulo the number to be factor. The sieving phase can be done in distributed fashion, on large number of processors simultaneously. The matrix-solving phase require large amount of storage space [23-25]. The RSA-150 (512 bit) composite number ‘\( n \)’ have been considered for factor by assuming NFS algorithm wherein \( n = p \times q \) while for the sieving steps 200000 parallel Pentium-4 microprocessors (2.53 GHz), FSB 53 MHz, Intel Desktop Motherboard D850EV2, 1850 chip set, 1024 MB RAM, PC800, Free BSD were used [25].

After factor through NFS obtained

\[
 p = 34800986710228369548397045104759342
 48310128173503854568995956375482781
 0717
 q = 44564774490364074153324112578708617
 6005442562977661534934197245324602
 96199
\]

The estimated sieving time to factor above RSA-150 (512 bit) composite number ‘\( n \)’ was 20597260 second, approximately of 239 days. This work supported by the CRIPTREC project is promoted by Telecommunication Advancement Organization of Japan. (mailto: macro@ntt.co.jp) [25]. Zimmermann, P., is a French computational mathematician, working at INRIA (The National Institute for Research in Computer Science and Control (French: Institut national de recherche en informatique et en automatique,INRIA)). His interests include asymptotically-fast arithmetic — he wrote a book [26] on algorithms for computer arithmetic with Richard Brent. He has developed some of the fastest available code for manipulating polynomials over \( \mathbb{F}(2) \), and for calculating hyper geometric constants to billions of decimal places. He is presently associated with the CARAMEL project (http://caramel.loria.fr/members.en.html ) to develop efficient arithmetic, in a general context and in particular in the context of algebraic curves of small genus; arithmetic on polynomials of very large degree turns out to be useful in algorithms for point-counting on such curves. He factored RSA -704 on July 2, 2012 [26]. The CARAMEL project-team has three main research themes [27]:

1. The number field sieve algorithm and its siblings, for integer factorization and discrete logarithm in finite fields,
2. Algebraic curves for cryptography, in particular genus 2 curves and pairings,
3. Arithmetic in general, from integers to floating-point numbers, in software and hardware.

Thus, it is concluded that, even ‘\( n \)’ is obtainable by the intruder however, deducing the private key ‘\( d \)’ from its equation \( e \cdot d \equiv (1 \mod \psi) \) where \( \psi = (p-1) \cdot (q-1), \) \( f \in \mathbb{F}^* \), such that \( n = p \cdot q \) (\( p \) and \( q \) is large distinct prime number) is almost complicated. Currently RSA modulus ‘\( n \)’ of size 1024 bits is being used for all protocols/applications [28, 29].

5. SHORTCOMING AND OVERCOME BY JAVA BIGINTEGER.

It is observed that this system is quite slow for large volume of data. Foundation of this shortcoming is, in RSA public key ‘\( e \)’ is an integer and message in \( C \in \mathbb{Z} \) \( \{0,1,2,3,4, \ldots, n-1\} \) to be encrypted therefore the computation of ‘\( m^e \mod n \)’ required processing time \( O(\text{size of } e \times \text{size of } n^2) \) and space \( O(\text{size of } e + \text{size of } n) \). Similarly, cause of slow for large volume of data is again complexity occurred during the decryption process to compute original message from cipher text \( c \). The complexity in the decryption process is \( O((\text{size of } d) \times \text{size of } n) \). Means the computational complexity is directly proportional to the
multiplication of (size e) and (size n)^2 while 'm' is a large integer to be send. In other hand complexity is directly proportional to the multiplication of (size d) and (size n)^2 during the decryption process while 'c' is a large integer. It may be 1024 bits or more. The time and space complexity of both the operations is given in the Tab. 1. This problem can be solve by fast exponentiation algorithm [1]. Many researchers are used this algorithm to do so. However, it is found that, Java BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods as discussed in the section 1.3. The encryption and decryption is discussed as :

```java
return m.modPow(e, n);
return c.modPow (d, n);
```

This feature of Java is applied instead of fast exponentiation algorithm to overcome the shortcoming.

### Table 1. Complexity in RSA cryptography

<table>
<thead>
<tr>
<th>Process</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>(O((\text{size e})(\text{size n})^2))</td>
<td>(O(\text{size e} + \text{size n}))</td>
</tr>
<tr>
<td>Decryption</td>
<td>(O((\text{size d})(\text{size n})^2))</td>
<td>(O(\text{size d} + \text{size n}))</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

RSA cryptosystem is presently used in a wide variety of products (TCP/IP, MIME, WAN, TELNET etc), platform (Apple, Sun, Novel, and Microsoft) around the world computer network for safely communication and transformation. This paper discussed the comprehensive view of the RSA cryptosystem, its straight, limitations, and various methods such as trial division, p- method, ECM, and NFS for breaking RSA system. It is found that RSA-1024 is absolutely secure from the intruder, however has introduced more computational complexity while increasing public key (e, n) and private key (d, n). As far as concern to solve this problem one can solve by fast exponentiation algorithms. However many researchers are often used these algorithm. It is found that, Java BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods. This feature of Java is successfully applied on the algorithm instead of fast exponentiation algorithm to overcome this shortcoming.

7. REFERENCES


[16] Aoki,K., Ueda, H., Kida, Y., 2004: NFS Factoring Statistics, NTT Labs, Rikuko University, Japan


