

INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES © 2004-2010 Society for Science and Nature (SFSN) All rights reserved

www.scienceandnature.org

# THRESHOLD MJ<sub>2</sub>–RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

\*E. Madhusudhana Reddy & M.Padmavathamma Department of Computer Science, S.V.University, Tirupati

#### ABSTRACT

Security protocols are a must for communication between parties. We studied new applications of Jordon Totient function and applied them to RSA public key cryptosystem with one public key and one private key, and developed protocols for communication between two parties using java and shown the graphical performance analysis on test results for key generation time, encryption time and decryption time respectively.

KEY WORDS: Cryptosystem, private Key, threshold, protocols etc.

#### INTRODUCTION

In this article we develop a new Public Key Cryptosystems which was extension of the work of Cesar Alison Monteiro Paixao [1] new variant of the RSA Cryptosystem. We extended four variants analyzed in [1] using the properties of Jordan Totient function [2]. We briefly discuss the possibility and validity of combining new variant with algorithm, java code, test result and graphical performance analysis to obtain a new efficient and general Cryptosystem.

#### JORDAN-TOTIENT FUNCTION

**Definition:** A generalization of the famous Euler's Totient function is the Jordan's Totient function [1] defined by

$$J_{k}(n) = n^{k} \prod_{p|n} (1 - p^{-k})$$
, Where k,  $n \in \mathbb{Z}^{+}$ 

#### **RSA CRYPTOSYSTEM**

RSA Public Key Cryptosystem was accomplished by Rivest, Shamir and Adleman in 1978. This cryptosystem works in  $Z_n$ , where n is the product of large primes p and q. *Key Generation:* 1. Generate two primes p and q and compute their product n =pq

2. Pick e such that gcd (e,  $\phi(n)$ ) = 1, where  $\phi(n) = (p-1)(q-1)$ 

3. Compute d such that  $d \equiv e^{-1} \pmod{\phi(n)}$  i.e  $ed \equiv 1 \pmod{\phi(n)}$ 

> Public Key = (e, n)Private Key = (d, n)

*Encryption:* Given a plaintext M and the Public Key = (e, n), compute the ciphertext C by using the formula.

$$C \equiv M^e \pmod{n}$$

*Decryption:* Given a ciphertext C and the Private Key = (d,n), compute the plaintext M by using the formula.

$$M \equiv C^d \pmod{n}$$

MJ<sub>2</sub> - RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

The main role of RSA cryptosystem is the usage of Euler's

We define the conjugate of this function as  $\overline{J_k}(n) = n^k \prod_{p|n} (1 + p^{-k})$ 

**Properties** 

1) 
$$J_k(1) = 1, J_k(2) = 2^k - 1 \equiv 1 \pmod{2}$$

2)  $J_k(n)$  is even if and only if  $n \ge 3$ 

3) If p is a prime number then

$$J_{k}(p) = p^{k} (1 - p^{-k}) = (p^{k} - 1)$$
$$J_{k}(p^{\alpha}) = p^{(\alpha - 1)k} (p^{k} - 1)$$
$$4) \text{ If } n = p_{1}^{\alpha_{1}} . p_{2}^{\alpha_{2}} .... p_{r}^{\alpha_{r}} \text{ Then}$$

$$J_{k}(n) = p_{1}^{(\alpha_{1}-1)k} \cdot p_{2}^{(\alpha_{2}-1)k} \dots p_{r}^{(\alpha_{r}-1)k} \cdot (p_{1}^{k}-1)(p_{2}^{k}-1) \dots (p_{r}^{k}-1)$$
  
5) 
$$J_{1}(n) = \phi(n)$$

Totient function  $\phi(n)$  and Euler's theorem. Now we replace  $\phi(n)$  by Jordan-totient function  $J_2(n)$  with the same property.

*Key Generation:* 1. Generate two primes p and q and compute their product n = pq.

2. Pick e such that gcd (E,  $J_2(n)$ ) = 1 Where  $J_2(n) = J_2(pq) = (p^2-1)(q^2-1)$ .

3. Compute D such that D  $^{-1} \mod J_2(n)$  i.e.,  $ed \equiv 1 \pmod{J_k(n)}$ 

Public Key = (2, E, n)

Private Key = (2, D, n)

*Encryption:* Given a plaintext M and the Public Key = (2, E, n) compute the ciphertext C by using the formula.

$$C \equiv M^e \pmod{n}$$

*Decryption:* Given a ciphertext C and the Private Key = (2, D, n), compute the plaintext M by using the formula.

 $M \equiv C^d \pmod{n}$ 

# ALGORITHM FOR NEW VARIANT $\rm MJ_2$ -RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

Step 1: Start

Step 2: [Generate multiple prime's number]  $p_1, p_2$ Step 2: [Compute  $J_2(n)$ ]  $J_2(n)$   $2p_1^2-1$ ) \*  $(p_2^2-1)$ Step 4: [Compute E using gcd method] gcd(E,  $J_2(n)$ ) Step 5: [Compute D] D Step 6: [compute public key] Public key Step 7: [Compute private key] Private Key', II)

### ITERATIVE METHOD FOR MJ<sub>2</sub> –RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

Compute N=p1\*p2=33  $J_2(N) = (p_1^2-1) (p_2^2-1)=960$ Choose E=7 Check gcd(E,  $p_1^2-1$ )=gcd (7,120) Check gcd (E,  $p_2^2-1$ ) = gcd(7,8) = 1

Select two primes p1=11,p2=3

Check gcd (E,  $p_2^{2}$ -1) = gcd(7,8) = 1 Check gcd (E,  $(p_1^{2}$ -1)  $(p_2^{2}$ -1)) = gcd (7, 960)=1

Compute D such that ED  $_{2}(N)$ i.e., compute D  $\__{1}^{-1}$  (mod J<sub>2</sub>(N))  $D^{1} \__{1}^{-1}$  (mod 960) i.e, find a unique value d such that 960 divides 7d - 1 Simple testing with d=1, 2, 3, ... J<sub>2</sub>(N) 1d a unique value d such that 960 d Public key = (E, N) = (7, 33) Secret key = (D, N) = (823, 33)

Now if we want to encrypt the message M=8 we have  $C = \frac{E( \mod N)}{C} (\mod 33)$  C = 2

To decrypt the cipher text we have  $M \xrightarrow{D} (mod N) \xrightarrow{323} (mod 33)$  (Hard) M

### IMPLEMENTATION OF MJ<sub>2</sub> -RSA WITH ONE PUBLIC KEY AND ONE PRIVATE KEY FOR 1024 BIT LENGTH

import java.io.\*; import java.math.BigInteger; import java.util.Random;

public class MJ2RSA {
 int bitlength = 1024;
 int blocksize = 256; //blocksize in byte
 private BigInteger p1;
 private BigInteger p2;
 private BigInteger N;
 private BigInteger phi;
 private BigInteger e;
 private BigInteger d;
 private Random r;
 /\*\* Init public and private keys \*/

Step 8: [Read the Plain Text M] read M Step 9: [Compute Plain Text to Cipher Text using Public Key]  $C \stackrel{\mathbf{MP}}{\vdash} \mathbf{ME} \pmod{n}$ Step 10: [Compute Cipher Text to Plain Text using Private Key] M (mod n) Step 11: Stop public MJ2RSA() { r = new Random();// get two big primes p1 = BigInteger.probablePrime(bitlength, r); p2 =BigInteger.probablePrime(bitlength, r); N = p1.multiply(p2);phi = p1.pow(2).subtract(BigInteger.ONE).multiply(p2.pow(2).s ubtract(BigInteger.ONE)); // compute the exponent necessary for encryption (private key) e = BigInteger.probablePrime(bitlength/2, r);while (phi.gcd(e).compareTo(BigInteger.ONE) > 0&& e.compareTo(phi) < 0 ) { e.add(BigInteger.ONE); // compute public key d = e.modInverse(phi); public MJ2RSA(BigInteger e, BigInteger d, BigInteger N) { this.e = e; this.d = d: this.N = N; public static void main (String[] args) { long startTime = System.currentTimeMillis(); MJ2RSA rsa = new MJ2RSA(); "+ System.out.println("The bitlength rsa.bitlength); long endTime = System.currentTimeMillis(); System.out.println(" Key Generation Time :"+ (endTime-startTime)); String teststring=new String(); try{ BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); System.out.println("Enter the test string"); teststring = br.readLine(); //teststring = "SVUNIVERSITY"; System.out.println("Encrypting String: +teststring); System.out.println("String in Bytes: bytesToString(teststring.getBytes())); }catch(Exception ex){} // encrypt long startEncyTime = System.currentTimeMillis(); byte[] encrypted rsa.encrypt(teststring.getBytes()); System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted)); long endEncyTime = System.currentTimeMillis(); System.out.println(" Encryption Time :"+ (endEncyTime-startEncyTime) + "millSecond");

```
I.J.E.M.S., VOL. 1(1) 2010: 12-17
                                                                                                                                                                                           ISSN 2229-600X
                                                                                                                                 test += Byte.toString(b);
           // decrypt
            long startDecyTime = System.currentTimeMillis();
                                                                                                                             3
            byte[] decrypted = rsa.decrypt(encrypted);
                                                                                                                            return test:
            System.out.println("\nDecrypted String: " + new
String(decrypted));
                                                                                                                        /** encrypt byte array */
            long endDecyTime = System.currentTimeMillis();
                                                                                                                        public byte[] encrypt(byte[] message) {
                   System.out.println(" Decrypted Time :"+
                                                                                                                                           (new
                                                                                                                                                            BigInteger(message)).modPow(e,
                                                                                                                            return
(endDecyTime-startDecyTime) + "millSecond");
                                                                                                                N).toByteArray();
                                                                                                                         /** * decrypt byte array */
          /**Converts a byte array into its String
representations */
                                                                                                                        public byte[] decrypt(byte[] message) {
        private static String bytesToString(byte[] encrypted)
                                                                                                                            return
                                                                                                                                           (new
                                                                                                                                                          BigInteger(message)).modPow(d,
                                                                                                                 N).toByteArray();
ł
            String test = "";
                                                                                                                         }
            for (byte b : encrypted) {
                                                                                                                   }
       TEST RESULTS OF MJ2 -RSA JAVA PROGRAM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY
                                                                                                                                                                                                  - 8 ×
     C:\WINDOWS\system32\cmd.exe
    Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
    C:\Documents and Settings\Madhusudhana>CD C:\Documents and Settings\Madhusudhana
\Desktop\PHDAshok5\source code
    C:\Documents and Settings\Madhusudhana\Desktop\PHDAshok5\source code>javac MJ2RS
    A.java
     C:\Documents and Settings\Madhusudhana\Desktop\PHDAshok5\source code>java MJ2RSA
   The bitlength 1024

Key Generation Time :3234

Enter the test string

ravi is a good boy

Encrypting String: ravi is a good boy

String in Bytes: 1497118105321051153297321031111111003298111121

Encrypted String in Bytes: 0-1246717-4925-7060-80526-5811041-1153386-11097304186

-68-72116-666381-17107-756-6812510545100230-687-11398-112-117-60-1268210-23102-1

17-4063-1381-38-9034152711222-24-5650-89-1119-3-802-91-9-25-43913066118-39-7334

126-718111911-85-118127-1512210021124-23-115-69-47-589852-97-123-5576-122115-119

-75511-1264560-26-120-2-56-60-501-1993-10-284-2856-128-42-46-60-9146-1047-398-103

50-50-4078-26-62-1006844-3930118-52112-106-4811212-36-13-117-59-113117-21-99-105

58-122278-84-445993-481124110743-116-91171071912361138211069121-38-437448-59-119

95-107-9-5270-4864-2544731209-7810765-3366-48467120-2-120-70-798-81-5634-33-106

Encryption Time :94millSecond
    Decrypted String: ravi is a good boy
Decrypted Time :406millSecond
     C:\Documents and Settings\Madhusudhana\Desktop\PHDAshok5\source code>java MJ2RSA
   The bitlength 1024

Key Generation Time :2235

Enter the test string

pavi is guided by prof.M.Padmavathmma

Encrypting String: pavi is guided by prof.M.Padmavathmma

String in Bytes: 112971181053210511532103117105100101100329812132112114111102467

746809710010997118971161041091099732

Encrypted String in Bytes: 42-43-70120-44-12230-55-65-98112-4411084-8455913397-1

411735112-105-34-5462-7393748-23-49-7362-63-40536735112-115-40-47-1059616-704712

6122-47-40-7496-113-54-104109114126-441676-111-72-4448-8810885-68-47103-61116-59

-59-59127-120-2041-114104-109100-6680-99-1209310176-12711920-124-5472-1112175903

5-22-94-10498-82366399677-28-8110898-7-121-169-92102-9864-38-5389103126-921161-3

5-37-5112111100119-68-393061116-3-4794-12147-125-10511857-16-96-51-120-3295-1171

24-84-11855-127-121959872-54-1052440-717610421-20-11920-114-59-26106-24653108-49

69-3820144-609279-119-85-96831277649-74119194117-1715117-4-67-3014110-16-1134601

087-88-113476622-108-14-855327-63-447440-784-2297108109-756533-6445-35

Encryption Time :219mill8econd
     Decrypted String: pavi is guided by prof.M.Padmavathmma
Decrypted Time :437millSecond
     C:\Documents and Settings\Madhusudhana\Desktop\PHDAshok5\source code>
```

GRAPHICAL PERFORMANCE ANALYSIS BETWEEN RSA AND MJ<sub>2</sub> –RSA WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

# Key Generation Time Performance

| Bits | RSA  | MJ <sub>2</sub> -RSA |
|------|------|----------------------|
| 256  | 78   | 78                   |
| 512  | 391  | 235                  |
| 1024 | 3698 | 2391                 |



**Encryption Time Performance** 

| Bits |      | RSA | MJ <sub>2</sub> -RSA |
|------|------|-----|----------------------|
|      | 256  | 5   | 5                    |
|      | 512  | 16  | 15                   |
|      | 1024 | 94  | 79                   |



| Decryption Time |     |                      |  |  |
|-----------------|-----|----------------------|--|--|
| Bits            | RSA | MJ <sub>2</sub> -RSA |  |  |
| 256             | 16  | 16                   |  |  |
| 512             | 47  | 78                   |  |  |
| 1024            | 203 | 359                  |  |  |



#### Comparison between RSA and MJ<sub>2</sub> -RSA

| 1024 bits | RSA  | MJ <sub>2</sub> -RSA |
|-----------|------|----------------------|
| KG        | 3698 | 2391                 |
| En Time   | 94   | 79                   |
| De Time   | 203  | 359                  |



## CONCLUSION

In this article we presented design and development of Multi prime Jordan-Totient- RSA viz. MJ2-RSA cryptosystem with one public key and two private keys in Java and we analyzed the performance of our programs with the existing RSA cryptosystem and compared the performance of two systems key generation time, the performance of encryption time and decryption time respectively.

This result helps in enhancement of the block size for plaintext and enhances the range of public / private keys. The increase in the size of private key avoids the attacks on private key. This concludes that MJ2-RSA provides more security with low cost.

#### REFERENCES

- [1]. Apostal T.M, introduction to analytic number theory, Springer International Students Edition 1980.
- [2]. Beak J, Lee B, and Kim K: Provable Secure Length Saving Public – Key Encryption Scheme under the Computational Diffie-Hellman Assumption, Electronics and Telecommunications Research Institute (E TRI) Journal, Vol.22,No.4, pages 25-31, 2000.
- [3]. Bellare M: Practice-Oriented Provable Security, Lectures on Data Security (Modern Cryptology in theory and practice), LNCS 1561, pages 1 – 15, Springer Verlag, 1999.

- [4]. Bellare M, Desai A, Pointcheval D and Rogaway P: Relations Among Notations of Security for Public-Key Encryption Scheme, Advances in Cryptology, Proceedings of CYPTO 98, LNCS 1462, Pages 26-45, Springer verlag, 1998.
- [5]. Bellare M and Rogaway P: Random Oracles are practical : A paradigm for Designing Efficient Protocols, ACM conference on computer and communications security.
- [6]. Bellary M and Rogaway P: Exact Security of Digital Signatures-How to sign with RSA and Rabin Schemes, Advances in Cryptology Proceedings of EUROCRYPT'96, LNCS 1070
- [7]. Boneh D and Shacham H: Fast variants of RSA. RSA laboratories 2002.
- [8]. Collins T, Hopkins D, Langford S and Sabin M, Public-Key Cryptographic Apparatus and method. U.S.Patent #5, 848, 159, January – 1997.
- [9]. Cramer R and Shoup V: A Practical Public-Key Cryptosystem provably Secure against Adoptive Chosen Ciphertext Attack, Advances in Cryptology – Proceedings of CRYPTO'98, LNCS 1462 pages 13-25, Springer Verlag, 1998.

- [10]. Diffie W and Hellman M: New Directions in Cryptography, IEEE Transactions on Information Theory. Vol-10, pages 74-84, IEEE, 1977.
- [11]. Mao M: Modern Cryptology : Theory and Practice, Prentice Hall, 2004
- [12]. Pointcheval D: Chosen-Ciphertext security for any One-way Cryptosystems, Public-Key Cryptography-Proceedings of PKC 2000, LNCS 1751, pages 129-146, Springer Verlag, 2000.
- [13]. Proceedings of ACM CCS's 93, pages 62-93, ACM, 1993.
- [14]. Quisquater J.J and Couvreur C, Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. Electronic Lectures, Vol-18, 905-907, 1982.
- [15]. Rivest R: Shamir A and Adleman L: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 21 (2), pages 120-126,1978.
- [16]. Thajoddin. S & Vangipuram S; A Note on Jordan's Totient function Indian J.Pure apple. Math. 19(12): 1156-1161, December, 1988.