INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES

www.scienceandnature.org

THRESHOLD GENETIC SENSOR SYSTEM TO E-MANUFACTURING SYSTEM

E.MadhuSudhan [1], G. Krishnaiah[2]
[1]Research Scholar, [2]Professor Department of Mechanical Engineering
S.V.University college of Engineering, S.V.University, Tirupati

ABSTRACT
Today's manufacturing companies only see how professionally their company can compete globally. Today's customers provide peak priority for money, superior quality and less risk. With a fast transform in technology especially in the manufacturing sector there is a need to change the manufacturing strategies, which can result in improved performance thereby meeting the customer demands. This paper generates a Threshold Genetic Sensor System with new applications of Jordon-Totient function and applied them to RSA public key cryptosystem with one public key and one private key which can be used for development of protocols to provide secured communication between different E-Manufacturing authorities and customers and developed code using java and shown the graphical performance analysis on test results for key generation time, encryption time and decryption time respectively.

Key Words: Threshold Genetic Sensor System, Jordon-Totient function, E-Manufacturing system and Manufacturing strategies.

I. INTRODUCTION
In this paper 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 extend variant 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.

2. JORDAN – TOTIENT FUNCTION
2.1 Definition: A generalization of the famous Euler's Totient function is the Jordan's Totient function [1] defined by

$$J_k(n) = n^k \prod_{p|n}\left(1 - p^{-k}\right), \text{ Where } k, n \in Z^+$$

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

2.2 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 $n \geq 3$

3) If p is a prime number then

$$J_k(p) = p^k\left(1 - p^{-k}\right) = \left(p^k - 1\right)$$

$$J_k\left(p^\alpha\right) = p^{(\alpha-1)k}\left(p^k - 1\right)$$

4) If n = $p_1^{\alpha_1}.p_2^{\alpha_2}...............p_r^{\alpha_r}$ Then

$$J_k(n) = p_1^{(\alpha_1-1)k}.p_2^{(\alpha_2-1)k}..........p_r^{(\alpha_r-1)k}.(p_1^k-1)(p_2^k-1)........(p_r^k-1)$$

5) $J_1(n) = \phi(n)$

3. 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 ≡ $e^{-1} \pmod{\phi(n)}$ 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.

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

4. MJ_2 - RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY
The main role of RSA cryptosystem is the usage of Euler's 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 E^{-1} mod $J_2(n)$ i.e., $ed \equiv 1(\mod J_k(n))$.

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.

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

5. ALGORITHM FOR NEW VARIANT 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 3: [compute $J_2(n)$] $J_2(n)$ $p_1^2-1) * (p_2^2-1)$

Step 4: [Compute E using gcd method] gcd(E, $J_2(n)$) ıpu

Step 5: [Compute D] D gc $J_2(n)$)

Step 6: [compute public key] Public key

5: [C

Step 7 : [Compute private key] Private Key (D,n)

Step 8: [Read the Plain Text M] read M

Step 9: [Compute Plain Text to Cipher Text using Public Key] C the ^{E} (mod n)

Step 10: [Compute Cipher Text to Plain Text using Private Key] M ^{D} (mod n)

Step 11: Stop

6. IMPLEMENTATION OF MJ$_2$ -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 */
        public MJ2RSA() {
                r = new Random();
            // get two big primes
        p1 = BigInteger.probable Prime(bitlength, r);
        p2 = BigInteger.probablePrime(bitlength, r);
            N = p1.multiply(p2);
phi =
p1.pow(2).subtract(BigInteger.ONE).multiply(p2.pow(2).subtract(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();
            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");
        // decrypt
        long startDecyTime = System.currentTimeMillis();
        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("\nDecrypted String: " + new String(decrypted));
        long endDecyTime = System.currentTimeMillis();
            System.out.println(" Decrypted Time :"+ (endDecyTime-startDecyTime) + "millSecond");
    }
    /**Converts a byte array into its String representations */
```

```
private
static String bytesToString(byte[] encrypted) {
        String test = "";
        for (byte b : encrypted) {
           test += Byte.toString(b);
        }
        return test;
    }
    /** * encrypt byte array  */
    public byte[] encrypt(byte[] message) {
        return     (new     BigInteger(message)).modPow(e,
N).toByteArray();
        }
        /*** decrypt byte array*/
        public byte[] decrypt(byte[] message) {
           return     (new     BigInteger(message)).modPow(d,
N).toByteArray();
        }
    }
```

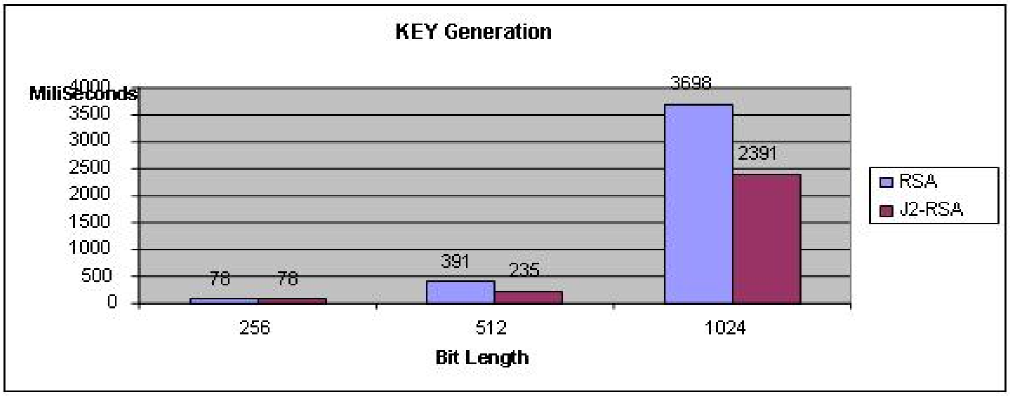
7. TEST RESULTS OF MJ_2 -RSA JAVA PROGRAM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

```
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: 1149711810532105115329732103111111100329811121
Encrypted String in Bytes: 0-1246717-4925-7060-80526-5811041-1153386-1109730418
-68-72116-666381-17107-756-6812510545100230-687-11398-112-117-60-1268210-23102-1
17-4063-1381-38-9034152711222-24-5650-89-11119-3-802-91-9-25-43913066118-39-7334
126-718111911-85-118127-1512210021124-23-115-69-47-589852-97-123-5576-122115-119
-7511-1264560-26-120-2-56-60-501-1993-10-284-2856-128-42-46-60-9146-1047-398-103
50-50-4078-26-62-1006844-3930118-52112-106-4811212-36-13-117-59-113117-21-99-105
81664722-51-94-4340-105-12582-117-37120-15-49-1210-701143-120-3124-18125-101-15-
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 :219millSecond

Decrypted String: pavi is guided by prof.M.Padmavathmma
 Decrypted Time :437millSecond

C:\Documents and Settings\Madhusudhana\Desktop\PHDAshok5\source code>
```

8. GRAPHICAL PERFORMANCE ANALYSIS BETWEEN RSA AND MJ_2 –RSA WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

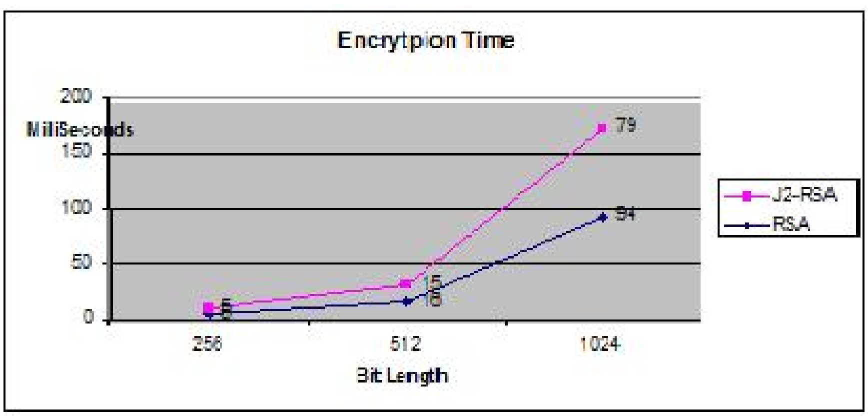
Key Generation Time Performance

Bits	RSA	MJ_2 -RSA
256	78	78
512	391	235
1024	3698	2391



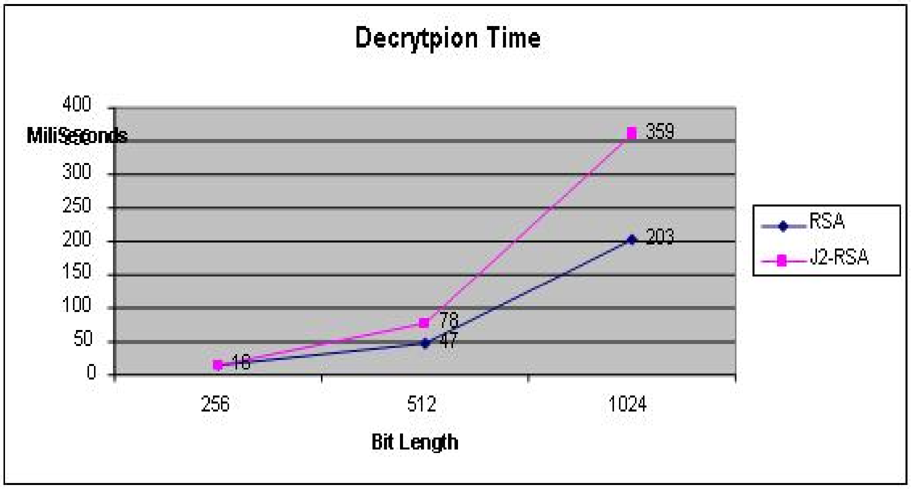
Encryption Time Performance

Bits	RSA	MJ_2 -RSA
256	5	5
512	16	15
1024	94	79



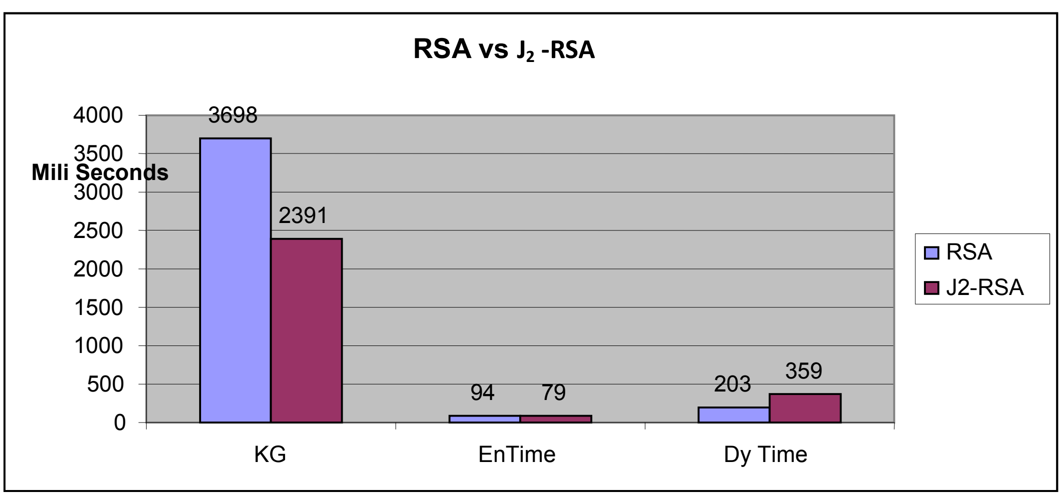
Decryption Time

Bits	RSA	MJ_2 -RSA
256	16	16
512	47	78
1024	203	359



Comparison between RSA and MJ_2 -RSA

1024 bits	RSA	MJ_2 -RSA
KG	3698	2391
En Time	94	79
De Time	203	359



9. CONCLUSION

In this paper we presented design and development of Threshold Genetic Sensor System with new applications of Jordon-Totient function and applied them to RSA public key cryptosystem with one public key and one private key which can be used for development of protocols to provide secured communication between different E-Manufacturing authorities and customers and developed MJ2-RSA cryptosystem with one public key and one private key in Java and analyzed the performance of our program 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.

10. REFERENCES

Apostal T.M, introduction to analytic number theory, Springer International Students Edition 1980.

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.

Manufacturing Engineering Handbook – by Hwaiyu Geng, McGraw Hill Professional 1 edition, March 1, 2004,

Koc M, Ni J, Lee J. Introduction of e-manufacturing. Proceeding of the International Conference on Frontiers on Design and Manufacturing, Dalian, China, July 2002.

E – Manufacturing Review – Jay Lee – Robotics and Computer Integrated Manufacturing Journal., May 23 – 2003

Lee J, Ahad A, Ko@ M. E-manufacturing—its elements and impact. Proceedings of the Annual Institute of Industrial Engineering (IIE) Conference, Advances in Production Session, Dallas, TX, USA, May 21–23, 2001.

Mao M: Modern Cryptology : Theory and Practice, Prentice Hall, 2004.

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

Proceedings of ACM CCS's 93, pages 62-93, ACM, 1993.

Quisquater J.J and Couvreur C, Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. Electronic Lectures, Vol-18, 905-907, 1982.

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.

Thajoddin. S & Vangipuram S; A Note on Jordan's Totient function Indian J.Pure apple. Math. 19(12): 1156-1161, December, 1988.

Lee J, Ni J. Infotronics agent for tether-free prognostics. Proceeding of the AAAI Spring Symposium on Information Refinement and Revision for Decision Making: Modeling for Diagnostics, Prognostics, and Prediction, Stanford University, Palo Alto, CA, March 25–27, 2002.

Galine Setlak, Slawomir Pieczonka "Intelligent Manufacturing Systems" Intelligent information and engineering Systems No: 13, pp 142-149.