I.J.E.M.S., VOL. 1(1) 2010: 12-17 ISSN 2229-600X

INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES

www.scienceandnature.org

THRESHOLD MJ_2–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

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

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

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

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

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

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} $(\bmod\ \phi(n))$ i.e $ed \equiv 1(\bmod\ \phi(n))$

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.

$$C \equiv M^e (\bmod\ n)$$

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

$$M \equiv C^d (\bmod\ n)$$

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 $^{-1}$ mod $J_2(n)$ i.e., $ed \equiv 1(\bmod\ J_k(n))$.

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.

$$C \equiv M^e \quad (\bmod\ n)$$

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

$$M \equiv C^d \quad (\bmod\ n)$$

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

ITERATIVE METHOD FOR MJ_2 –RSA CRYPTOSYSTEM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY

Select two primes p1=11,p2=3
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 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 ____ $^{-1}$ (mod $J_2(N)$) D $^{-1}$ (mod 960) i.e, find a unique value d such that 960 divides 7d – 1 Simple testing with d=1, 2, 3, … $J_2(N)$ nd 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 ^{E} (mod N) ^{7} (mod 33) C = 2
To decrypt the cipher text we have M ^{D} (mod N) ^{823} (mod 33) (Hard) M

IMPLEMENTATION OF MJ_2 -RSA WITH ONE PUBLIC KEY AND ONE PRIVATE KEY FOR 1024 BIT LENGTH

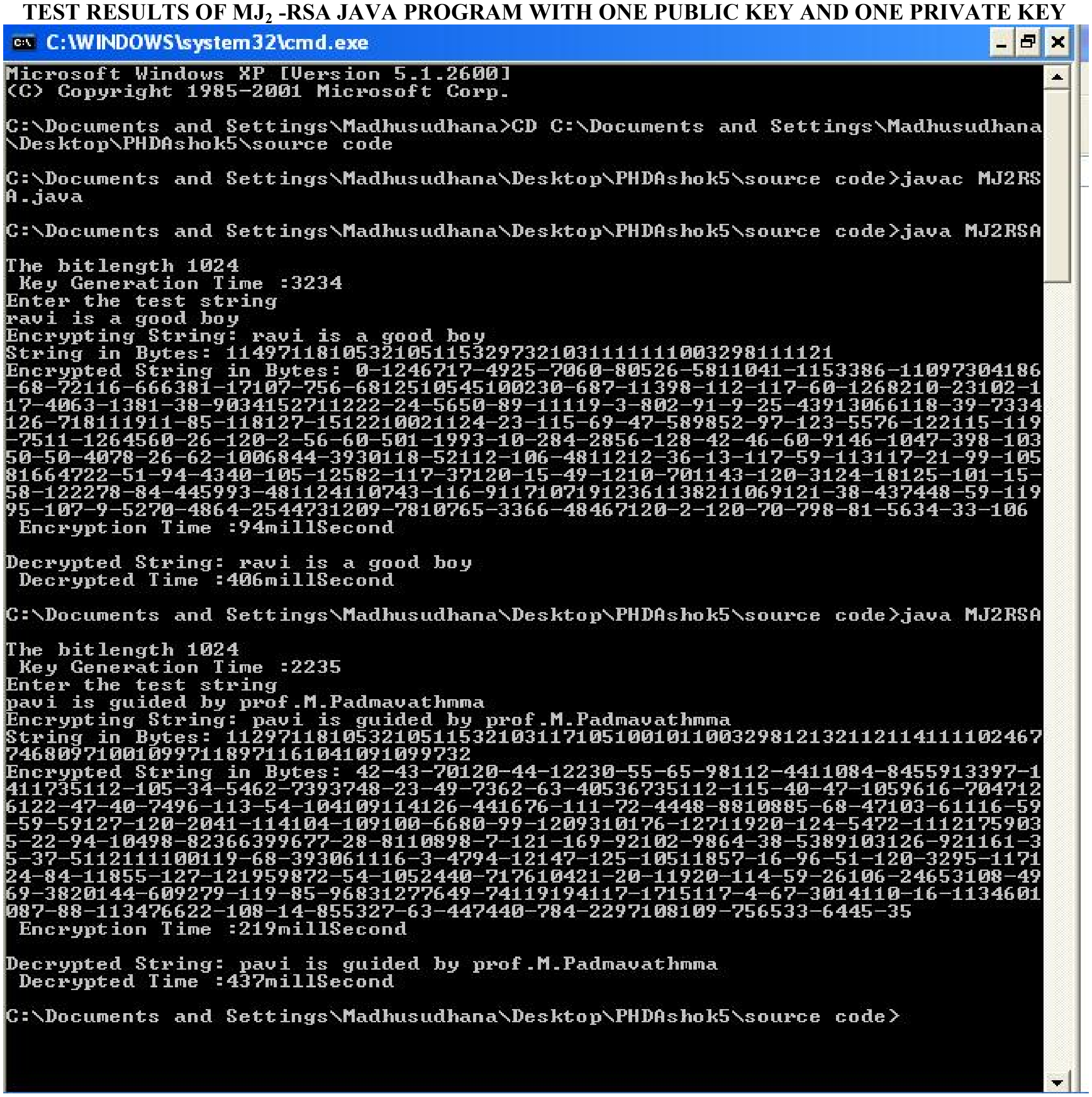
```
import java.io.*;
import java.math.BigInteger;
import java.util.Random;

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 */
```

Step 8: [Read the Plain Text M] read M
Step 9: [Compute Plain Text to Cipher Text using Public Key] C ^{E} (mod n)
Step 10: [Compute Cipher Text to Plain Text using Private Key] M ^{D} (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");
```

```
        // 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();
    }
 }
```

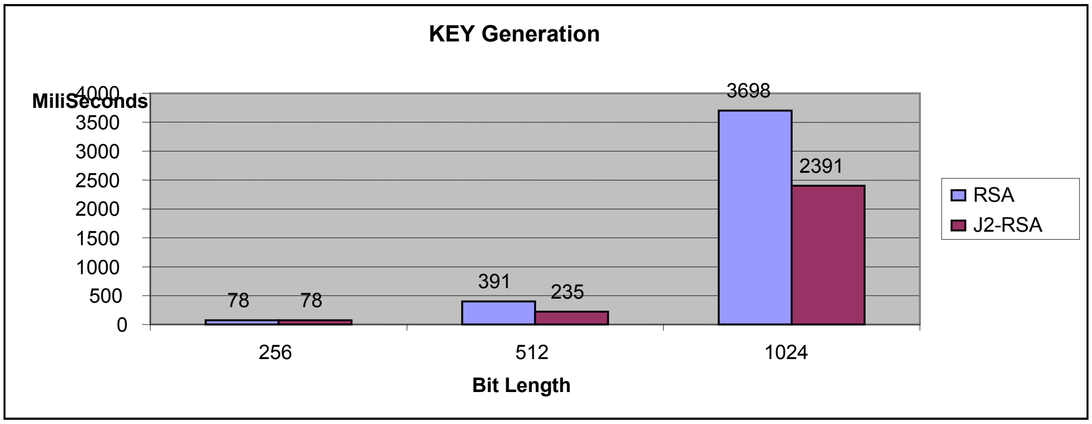
TEST RESULTS OF MJ_2 -RSA JAVA PROGRAM WITH ONE PUBLIC KEY AND ONE PRIVATE KEY



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

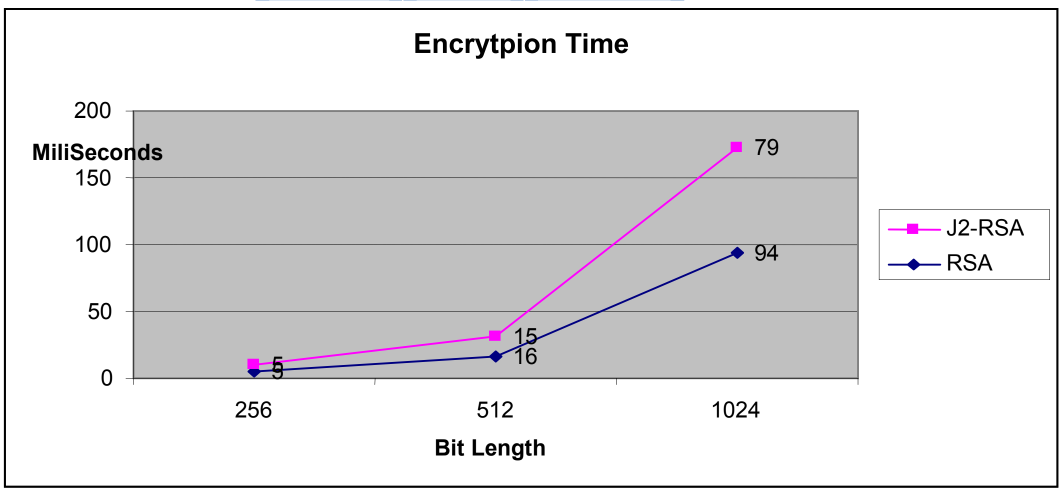
Key Generation Time Performance

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



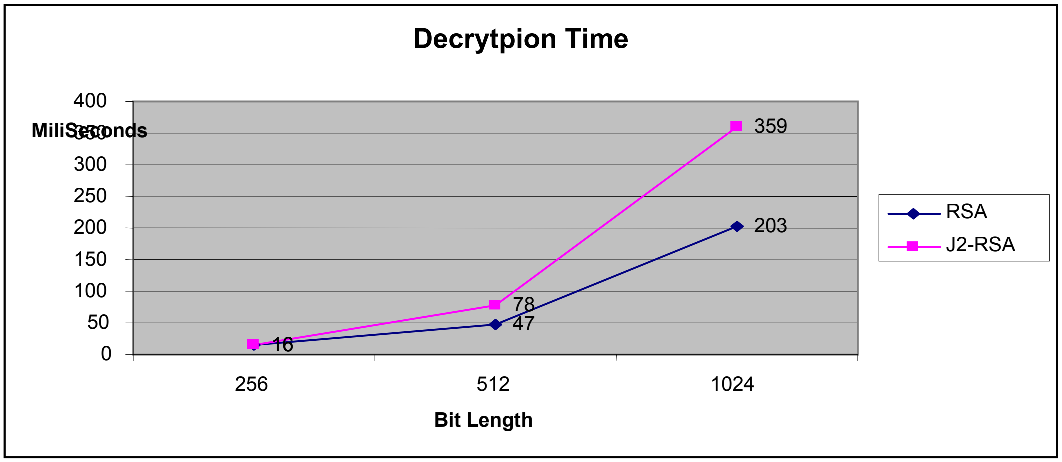
Encryption Time Performance

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



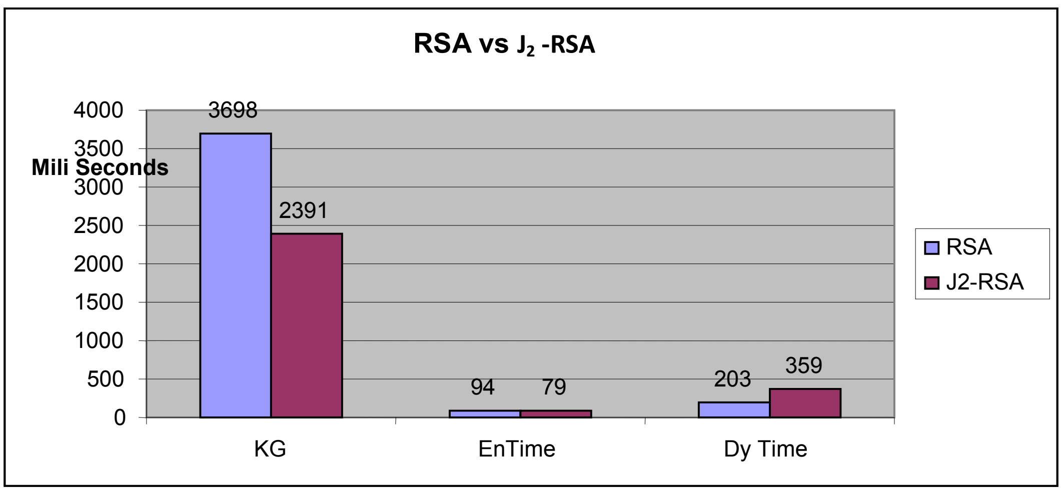
Decryption Time

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



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.