ISSN 22773061

Enhancing RSA algorithm using Mersenne Primes with reduced size of encrypted file

Shilpa M Pund, Chitra G Desai
Ph.D Student,
Department of Computer Science, University of Pune.
Maharashtra, India.
shilpa_pund@yahoo.com
Professor and Head, MCA Department, MIT Aurangabad, India.
chitrag_desai@yahoo.com,chitragdesai@gmail.com.

ABSTRACT

Message passing from source to destination is one of the important aspects of communication. However, it is required many times that this message gets transmitted secretly, so that no unauthorized person gets knowledge of the contents of that message. To retain the confidentiality of the message transmitted is a challenging task as it needs to be guaranteed that the message arrive in the right hands exactly as it was transmitted. Another challenge is of transmitting the message over a public, insecure channel. In this paper, RSA algorithm is implemented using Mersenne Primes which guarantees the primality. This is an enhanced algorithm which increases the strength of RSA by generating large prime numbers and also reduces the size of encrypted file.

General Terms

Security, Algorithm.

Indexing terms

RSA, encryption, decryption, mersenne.

Academic Discipline And Sub-Disciplines

Computer Science.

SUBJECT CLASSIFICATION

E.g. Data Security using Cryptography and Steganography.

TYPE (METHOD/APPROACH)

Experiment, Literary Analysis.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 7, No 1
editor@cirworld.com
www.cirworld.com, member.cirworld.com

INTRODUCTION

Message passing from source to destination is one of the important aspects of communication. However, it is required many times that this message be transmitted secretly, so that no unauthorized person gets knowledge of the contents of them message. To retain the confidentiality of the message transmitted is a challenging task as it needs to be guaranteed that the message arrive in the right hands exactly as it was transmitted. Another challenge is of transmitting the message over a public, insecure channel.

With the growing age of information technology more and more data regarding finance, health, legal, etc are routinely exchanged between computer, mobiles and similar communication facilities. Here society demands for reliable and secure communication and cryptography ensures you for this.

Cryptographic techniques, such as encipherments, digital signatures, key agreement and secrete sharing schemes, are important building blocks in the implementation of any security service. Cryptography is the study of “mathematical” systems involving two kinds of security problems: privacy and authentication [3]. A privacy system prevents the extraction information by unauthorized parties from message transmitted over a public channel, thus assuming the sender of a message that it is being read only by the intended recipient. An authentication system prevents the unauthorized injection of message into a public channel, assuring the receiver of a message of the authenticity of its sender.

Encryption is the standard means of rendering a private communication. The sender enciphers each message before transmitting it to the receiver. The receiver (but no unauthorized person) knows the appropriate deciphering function to apply to the received message to obtain the plain text. The large volume of personal and sensitive information currently held in computerized data banks and transmitted over telephone lines makes encryption increasingly important. In recognition of the fact that efficient, high-quality encryption techniques are very much needed . A public-key cryptosystem needs no private couriers, the key can be distributed over the insecure communication channel [3]. RSA is a well known public key cryptography algorithm. It is the first algorithm known to be suitable for signaling as well as encryption, and was one of the first great advancement in public key cryptography. The security of the RSA cryptosystem is based on two mathematical problems: the problem of factoring large numbers known as mathematical attack and the problem of trying all possible private keys known as brute force attack[1].

RSA ALGORITHM USING MERSENNE PRIMES

Mersenne number *:* Numbers of the form $M_n = 2^n-1$, ($n\geq1$) are called Mersenne numbers after Father Marin Mersenne (1588 - 1648), a French monk who studied which of these numbers are actually prime. If M_n is prime then we call it a Mersenne prime. Mersenne primes have a strong connection with perfect numbers. The currently known Mersenne primes correspond to n = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917 , 20996011 , 24036583 . The 42 nd Mersenne number which corresponds to n=25964951 and which has more than 7.8 million digits and the 43 rd Mersenne prime for n=30402457 is 9,152,052 digits long[6]. The latest 48th mersenne prime having 17 million digits in the number 2 multiplied by itself 57,885,161 times minus 1 - the Mersenne prime discovered in four years on 5th Feb,2013[7]. It is a difficult task to generate large prime numbers and test it for primality. This requires overheads on first generating a large number which should be probably prime and second, testing its primality. This is because the generated large prime number using different algorithms may not necessarily pass the primality test. In order to generate an assured large mersenne prime number we can make use of large primes[2]. This paper implements RSA cryptography algorithm using mersenne prime with reduced encrypted file size.

This algorithm which uses large primes as a input parameter as follows:

Declarations:

a) Ptext.txt : Plain text (source) file to be encrypted .

b) Ctext.txt : Cipher text file to be decrypted.

c) Array L : Used to store the contents of Ptext.txt for encryption.

d) Array S : Used to store the contents of Ctext.txt for decryption.

Algorithm :

1. Choose two large prime numbers p , q.

2. Generate two very large mersenne prime numbers as: $m = 2^p -1$ and $n = 2^q -1$.

3. Calculate $c= m* n$.

4. Calculate the value of Φ using the formula: $\Phi(c) = (m-1) * (n-1)$.

5. Generate the public key 'e' such that it is coprime with Φ(c).

6. Find the value of private key 'd' such that $(d *e) \equiv 1 \mod \Phi(c)$

7. Read plaintext in the form of binary data from the file **Ptext.txt**, store it in an array(**L)** .

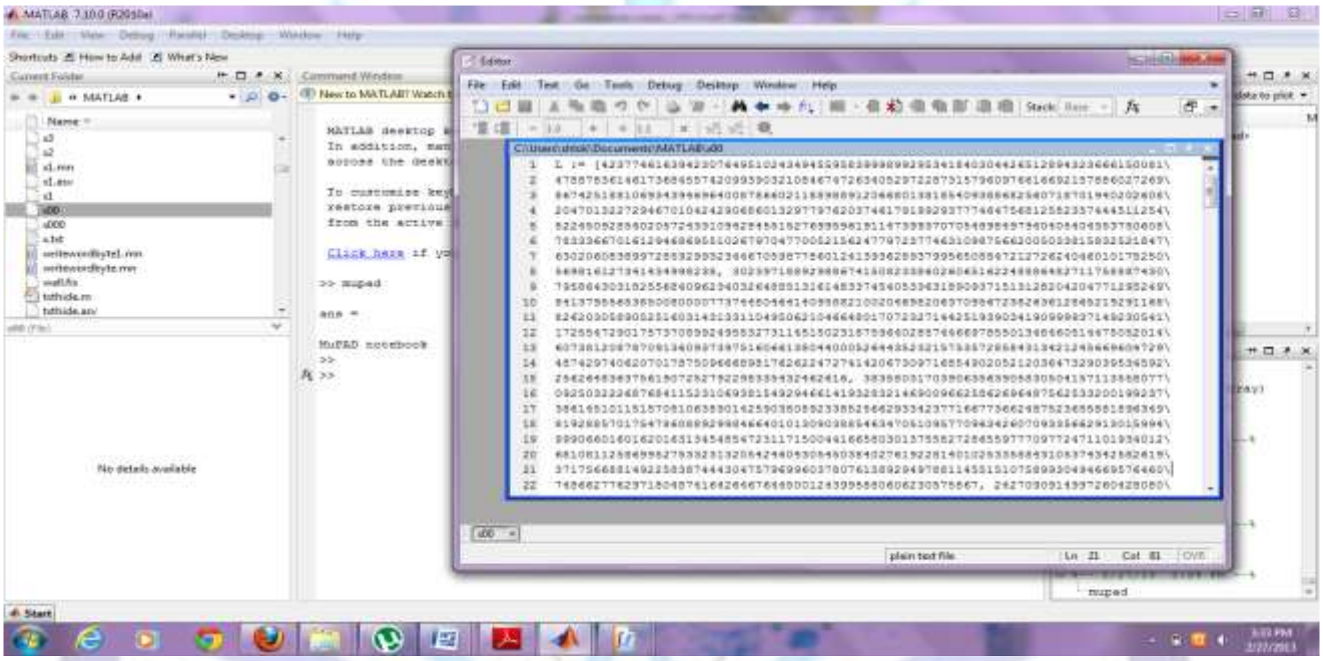
8. Perform L^e mod c (on each element of an array L) to get cipher text and store it in the file **Ctext.txt**.

9. For decryption, read the file **Ctext.txt** , store it in an array (S) .

10. Perform S^d mod c (on each element of an array S) to get a plain text .

The above algorithm is developed in Mupad environment with a symbolic math toolbox provided by Matlab. MuPAD can compute big numbers efficiently. The length of a number that can be computed is limited only by the available main storage. Hence Mupad Environment is preferable for the execution of this algorithm[6]. Here we are accepting two primes

p & q which are used to generate two very large mersenne primes. A prime can generate a large mersenne prime by using the formula m= 2^p-1 which guarantees to be a prime number. Now m and n so generated are very large primes. The product of these two is c = m*n is obviously a very large number which is very difficult to factories. Now read the plaintext character-by-character in the integer format and store it in an array L. Perform L^e mod c on each array element. The value of e generates randomly such that it is coprime with Φ(c) , L^e generate a very large value. So each character of an array of the plaintext is replace by a very large number (probably the size may be 100 to 1000 digits) depending on the size of p & q . Hence after encryption of the plaintext, the size of encrypted file blows up which occupies large memory space. This is the main limitation of the system. The size of private key d generated is also very large. However, the system becomes insecure if the secret exponent d so obtained is small relative to c [4].



```
MATLAB desktop ...
In addition, ...
across the desk...

To customize key...
restore previous...
from the active ...

Click here if yo...

>> mupad

ans =

MuPAD notebook
>>
>>
```

```
L := [[illegible]\
[illegible]
```

Fig1: Encrypted file

Table 1. Comparison of the original , encrypted and decrypted file sizes with respect to the value of p and q.

Value of p	Value of q	Private Key Size(In digits)	File name	Original file size	Encrypted File Size	Decrypted File size	Execution Time	Memory Used
521	607	340	textstego.txt	14.3KB	4.96 MB	14.3KB	60 sec	11MB
127	607	221	textstego.txt	14.3KB	3.23MB	14.3KB	20 sec	11MB
521	2203	821	textstego.txt	14.3KB	11.9 MB	14.3KB	675 sec	14MB
607	1279	568	conversion.txt	3.14KB	1.81MB	3.14KB	54 sec	8 MB
127	2203	689	modrsa.txt	2.27KB	1.62 MB	2.27KB	69 sec	8 MB

3217	4253	2249	rsa.txt	1.01KB	2.30MB	1.01KB	823 sec	7MB
4253	4423	2562	rsa.txt	1.01KB	2.68 MB	1.01KB	1220 sec	8MB
9689	9941	5910	T1.txt	27 Bytes	161 KB	27 Bytes	262 sec	7MB
9689	9941	5909	x25.txt	172 Bytes	1 MB	172 Bytes	1656 sec	7MB
11213	19937	9378	T1.txt	27 Bytes	256 KB	27 Bytes	879 sec	6MB

The real challenge in case of RSA is the selection of public key and generation of private key. Here public key is generated randomly. The use of mersenne prime has been made to generate large primes to enhance the security. Here the file conversion.txt has the original file size 3.14KB and the encrypted file of conversion.txt is of size 1105.92KB which is too large and hence required to reduce the encrypted file size.

ENHANCEMENT

We perform enhancement in the above algorithm to reduce the size of encrypted file. The reduction logic is as follows:

For Encryption

1. Each element of array L is reduced modulo 'e' and the quotients are stored is an array(QUE) and store the remainders in array (REM).

2. Now array(REM) is a cipher text , store it in the file ctext.txt.

For decryption,

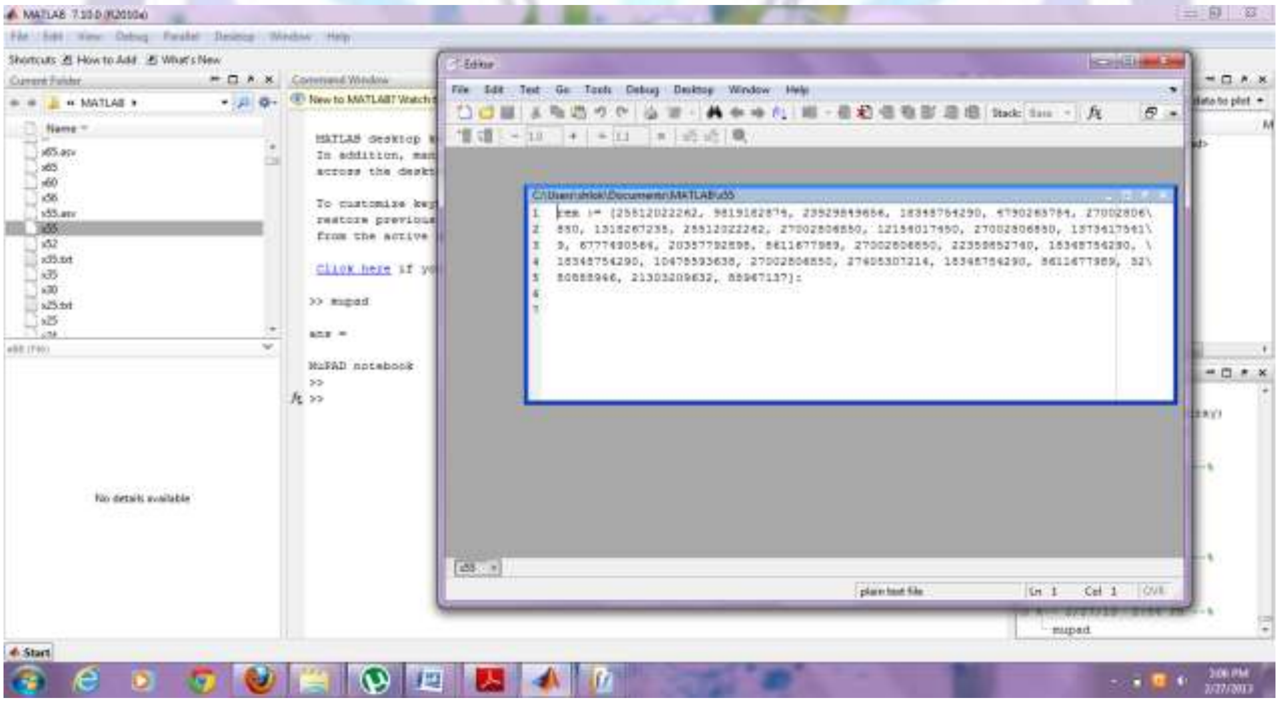
1. Read the file ctext.txt , store it in an array (S) .

2. Retrieve an array(QUE) and multiply each element of array (QUE) by e.

3. Now add each element of array (QUE) into array (S) respectively.

4. Perform S^d mod c (on each element of an array S after step 14) to get a plain text .

In this enhancement, the public key e is used again for reduction . Dividing each array element of the ciphertext again to get quotient and take mod e of each array element of the ciphertext to get the remainder. The array of remainder is now reduced with great extent . Hence the size of each character(in digits) to be encoded get reduced which results the reduced file size. But at the time of decryption , the values of quotients in array(QUE) are needed to retrieve to get the ciphertext again.



```
rem := [25512022262, 9819182874, 23529849656, 18348754290, 4790283784, 27002806\
850, 1318267235, 25512022262, 27002806850, 12158017450, 27002806850, 1373417561\
9, 6777480586, 20357792898, 8611677989, 27002806850, 22358852740, 18348754290, \
18348754290, 10478593638, 27002806850, 27405307214, 18348754290, 8611677989, 32\
80888946, 21303209632, 88967137];
```

```
MATLAB desktop w
In addition, man
across the deskt

To customize key
restore previous
from the active

Click here if yo

>> mupad

ans =

MuPAD notebook
>>
>>
```

Fig1: Encrypted file after applying the reduction logic.

RESULTS

Value of p	Value of q	Private Key Size(In digits)	File name	Original file size	Encrypted File Size	Decrypted File size	Execution Time	Memory Used
521	607	340	textstego.txt	14.3KB	200KB	14.3KB	58sec	16MB
127	607	221	textstego.txt	14.3KB	188KB	14.3KB	18sec	14MB
521	2203	821	textstego.txt	14.3KB	206KB	14.3KB	671sec	21MB
521	607	340	conversion.txt	3.14KB	44.0 KB	3.14KB	20 sec	8MB
127	521	196	conversion.txt	3.14KB	45.5 KB	3.14KB	5 sec	8MB
127	2203	701	modrsa.txt	2.27KB	31.5 KB	2.27KB	110 sec	9 MB
607	1279	568	rsa.txt	2KB	15KB	2KB	27 sec	8 MB
3217	4253	2249	rsa.txt	1.01KB	14.4 KB	1.01KB	825 sec	9MB
4253	4423	2562	rsa.txt	1.01KB	14.5 KB	1.01KB	1397sec	9MB
9689	9941	5910	T1.txt	27 Bytes	399 Bytes	27 Bytes	262 sec	6MB

DISCUSSION

As per the above comparison table, it is observed that the size of the encrypted file is reduced multiple times of the encrypted file before enhancement depending on the values of p and q respectively . But the execution time is an average 5% and the memory requirement is an average 33% more depending on the values of p and q respectively. We have executed this program for the primes numbers greater than 9689 on Intel(R) Core i5 -2410 M CPU with 6144 MB RAMS, it can generates the private and public keys within few second but as the size of the keys so obtained is very large it takes lot of time for encryption and decryption. This algorithm is very useful for small files up to 5KB and for the primes greater than 17. If the file size is too large and the primes are beyond 9000 then MuPAD does not respond because of the main storage limitation.

ACKNOWLEDGEMENT

First author would like to thank Dr. Nivedita H Mahajan for her valuable guidance in Mathematical concepts.

REFERENCES

[1] Ravi Shankar Dhakar Prashant Sharma, Amitkumar Gupta , 2012 Second International Conference on Advanced Computing & Communication Technologies, "Modified RSA Encryption Algorithm (MREA)" .

[2] Shilpa Pund , Chitra Desai, International Journal on Networking and Parallel Computing , " Implementation of RSA Algorithm using Mersenne Primes". ISSN: 2319-452.

[3] L. Adleman, R. Rivest and, A. Shamir "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" Comm. ACM 21, 1978, pp120-126.

[4] Wang Rui , Chen Ju , Duan Guangwen, Department of Computer Science and Engineering, Yunnan University, Kunming 650091, China. 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN) ," A k-RSA Algorithm" .

[5] Symbolic Math Toolbox™ 5 MuPAD® Tutorial by The Mathworks.

[6] http://planetmath.org/MersenneNumbers.html

[7] http://www.foxnews.com/science/2013/02/05/worlds-largest-prime-number-discovered

Author' biography with Photo

Shilpa Pund is basically a science graduate and completes her Masters of Computer Application from Amravati University, Amravati , Maharashtra in 2000. She has total experience of 12 years. Presently she is working as Assistant Professor , Dept of Comp. Science at Modern College of Arts, Science and Commerce, Shivajinagar, Pune-5. Maharashtra. Her research area includes network security , Cryptography, Steganography.

Dr. Chitra G Desai is basically a science graduate and completes her M.C.A from Government Engineering College Aurangabad in 2000. She qualified SET examination in 2004. She has been awarded with Ph.D (2006) in the subject of Computer science with study area in Software Reliability. She has 22 international and 10 national publications in various journals and proceedings of conferences. She has also completed a Minor Research Project funded by University of Pune under the area GIS urbanization. She is a recognized guide from Pune University and seven students are pursuing Ph. D under her guidance. She has a total experience of 12 years. Presently working as Head of the Department of MCA at Marathwada Institute of Technology (Engineering), Aurangabad and completed M. Tech by research (Network Security) from Dr. Babasaheb Ambedkar Marathawada University, Aurangabad