Secure Message Transmission using Lagrange Polynomial Interpolation and Huffman Coding

R. Siva Ranjani
Research Scholar, Dept.of CS&SE
Andhra University,
Visakhapatnam, Andhra Pradesh, India.

D. Lalitha Bhaskari, PhD.
Associate Professor, Dept.of CS&SE
Andhra University,
Visakhapatnam, Andhra Pradesh, India,

P. S. Avadhani, PhD.
Professor, Dept.of CS&SE
Andhra University,
Visakhapatnam,
Andhra Pradesh, India,

ABSTRACT

In this paper, an algorithm for secure transmission of message is proposed based on Lagrange's interpolation. The message is viewed as a polynomial of degree n where n is kept secret and transmitted to the receiver's side using the concept of a digital envelope. As any nth degree polynomial is uniquely determined by n+1 points, n +1 points are communicated to the other side, where the polynomial and hence the message is reconstructed. Padding of length m is added to the message to overcome the message length issue. Although any coding scheme may be used, in this paper Huffman coding is used for converting the plaintext into binary form. Finally, the proposed algorithm is compared with the performance of RSA algorithm and found to be efficient.

Keywords: *Lagrange interpolation, Hamming code, Padding, polynomial, digital envelope, RSA*

1. INTRODUCTION

RSA [2,9] is one of the oldest and most widely used public key cryptographic systems. It is using modular exponentiation function for encryption as well as for decryption. A practical concern in this algorithm is computation cost [3] of P^e mod n and C^d mod n. When P, C, d and e are large numbers RSA algorithm takes more computation time to evaluate P^e , C^d and then reduce modulo n. In this paper, a new algorithm based on Lagrange Polynomial Interpolation (LPI)[1] is proposed. This new algorithm used for passing on messages securely between the sender and receiver. Here the message is represented as polynomial coefficients. It uses the basic operations like addition/ subtraction and multiplication for polynomial coefficients calculation, which reduces the computation time.

The rest of the paper is organized as follows. In Section 2, a description of the proposed algorithm s given. Section 3 gives the experimental results and compared with RSA. Finally, in Section 4, we conclude the paper.

2. CRYPTO SYSTEM BASED ON LAGRANGE INTERPOLATION

In this work, the developed technique for encryption and decryption is based on mathematical idea of Lagrange Interpolation. A random number N is assumed by the sender and the respective N value securely communicated to receiver through the concept of digital enveloping [8]. Security of message is mainly depends on this randomly generated N. Hence, the sharing of N value between sender and receiver kept secret.

The sharing of N value, encryption and decryption of the message transfer are presented here. Sender assumes the N number, securely communicated to receiver through digital envelope.

Encryption: The sender is converting the actual message into points like (x_i,y_i) by passing the original message using the algorithm given in Fig 1. In Step 4, M is constructed by using the formula

$$M= \sum_{i=0}^{N-1} a_i$$

In this algorithm, f(x) is calculated by polynomial function f(x) using $[a_0, a_1,—a_{N-1}----a_{N+m-1}]$. where

$$f(x)= \sum_{i=0}^{N+m-1} ai*x^i.$$

Step 1: Take the plain text message (P)

Step 2: Construct the Huffman dictionary for the plain text(P). Convert the plain text into binary equivalent by using Huffman Dictionary.

Step 3: Take n bits find the decimal equivalent, named as M.

Step 4: Divide M into N parts by using random number generator function, i.e
M=[a_0,a_1,------, a_{N-1}], (The N is securely communicated to Receiver, the M is constructed by adding all the N parts)

Step 5: Append some m random numbers (m<N) to M i.e.,M=[a_0, a_1,—a_{N-1}---- a_{N+m-1}]

Step 6: Construct the polynomial function f(x) using [a_0, a_1,—a_{N-1}----a_{N+m-1}].

Step 7: Choose n+m+1 different values of x and then find their corresponding f(x) values named as y. $y_i=f(x_i)$

Step 8: Transmit x and y values in the form of points (x_i,y_i) onto the receiver side.

Fig 1: Encryption Process

Decryption: Receiver collects all the interpolation points, retrieves the original message using the algorithm in fig2. Step 2 is used for constructing polynomial function is achieved by using following formula

$$\mathbf{f(x)=}\sum_{i=0}^{N+m+1}\prod_{j=0,j\neq i}^{n+m+1} fi*\frac{(x-xj)}{(xi-xj)}$$

Step 1: Collect all the n+m+1 (xi,yi) points coming from sender.

Step 2: Construct the polynomial function f(x) by using Lagrange interpolation method

Step 3: Extract the coefficients from f(x), those are nothing but [$a_0, a_1, --a_{N-1}, ---, a_{N+m-1}$]

Neglect the coefficients from a_N-- a_{N+m-1}

Step 4: Calculate M = $a_0+a_1+a_2+----------+a_{N-1}$

Step 5: Find the binary equivalent of M

Step 6: Convert binary stream into text by using Huffman decoder. The extracted text is the plain text (P) transmitted by the sender

Fig 2: Decryption Process

SYSTEM MODEL FOR MESSAGE ENCRYPTION**:** In this section we review the message encryption process, as illustrated by the message encryption model in Fig 3, plain text file is given to the Huffman coding block. Huffman coding [6] is assigning short code words to the frequently occurring symbols of the text messages and rarely occurring symbols are assigned long code words. Huffman coding generates the binary stream file to the given text file by substituting each character in the text file with their corresponding codeword. In the figure text file contains the string 'hello'. Huffman coding block calculates the probability (P) of individual characters (A) and then assigns the code word(C).

A = The Alphabet set ={ h,e,l,o}

P = Set of Probabilities of Symbols in

Alphabet A= { ¼,1/4,2/4,1/4}

C = The Codewords={ 000,001,1,01}

The binary coded text is given to the decimal block; it converts the binary stream into decimal number. The binary stream input to the decimal block is 0000011101, decimal equivalent is 29. Decimal number 29 is given to the random function block. Random function [7] block will divide the input decimal number into N parts. In the figure the taken N is 5, random set generated is a={4,4,6,7,8}. The actual input is retrieved by using the following formula. M= $\sum_{i=0}^{N-1} ai$ = $\sum_{i=0}^{N5-1} ai$= 4+4+6+7+8=29. The generated random number set is concatenated with m cheating numbers, in figure assumed m=3 and cheating numbers are {9,3,5}. Resulting set is delivered to polynomial function block. Polynomial functional block receives n+m points as the input, constructs a polynomial by using n+m points as the coefficients in that polynomial. Constructed polynomial f(x) = $4+4x+6x^2+7x^3+8x^4+9x^5+3x^6+5x^7$. Now, the polynomial function assumes a set of values to x and finds their corresponding f(x) values. Assumed x values set is {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.Finally, (xi,f(xi)) are transmitted as cipher text onto receiver side.

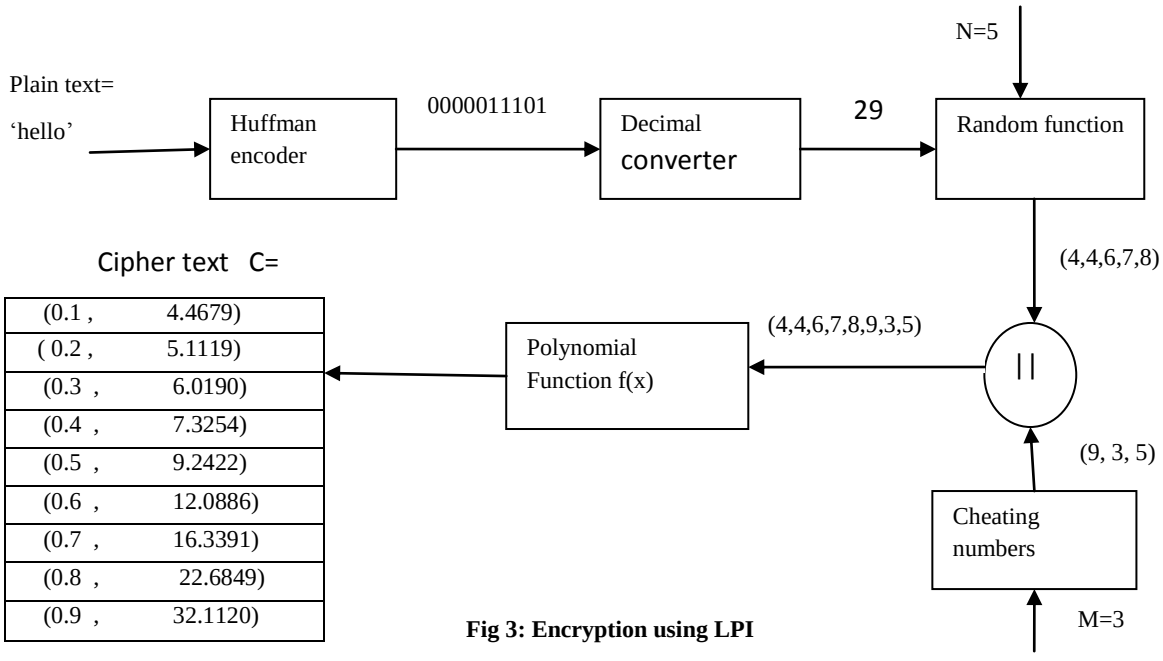


Fig 3: Encryption using LPI

SYSTEM MODEL FOR MESSAGE DECRYPTION**:** Message decryption process using Lagrange interpolation is shown in Fig 4. The receiver collects the cipher text in terms of coordinate points (xi,f(xi)) and then given to the Lagrange interpolation block. Lagrange polynomial block gives nth degree polynomial that passes through n+ 1 point. The Lagrange interpolation formula is given as

$f(x)= \sum_{k=0}^{n} f(xk) * Ln,k(x)$

$= f(x_0)L_{n,0}(x)+ f(x_1)L_{n,1}(x)+-------+ f(x_n)L_{n,n}(x)$

where

$L_{n,k}(x)= \frac{\prod_{i\neq k}^{n}(x-xi)}{\prod_{i\neq k}^{n}(xk-xi))}$

$=\frac{(x-x0)(x-x1)(x-x2)----(x-xk-1)(x-xk+1)--(x-xn)}{(xk-x0)(xk-x1)---(xk-xk-1)(xk-xk+1)---(xk-xn)}$

In the figure Lagrange interpolation [8] block constructs the polynomial f(x). Data extractor block receives the polynomial f(x), extracts first n coefficients and neglects m coefficients. The extracted n coefficients are delivered to decimal adder. Decimal adder adds all the n numbers and given to binary converter. Binary converter block converts the decimal number into binary equivalent. Resulting binary code is given to Huffman decoder. Huffman decoder block converts the binary code into text format by using Huffman dictionary.

3. EXPERIMENTAL ANALYSIS

We selected 1000 samples from The Hindu paper of various string lengths to conduct the experiment. The samples are encoded and decoded by using the proposed algorithm. For each sample we varied the N value. Table 1 shows the average computational time of different stings by varying N values. for k=0,1,2--n

From this table the following observations were made Computational time is directly proportional to string length.

- With increasing of N value computation time also increases.

Encryption and Decryption Analysis of RSA and LPI: Both RSA and LPI algorithms are used for encrypting the plaintext and decrypting the cipher text. RSA is using the exponent multiplication for encryption[5] and decryption[4].

LPI is based on addition, subtraction and multiplication operations. Computation time for encryption and decryption operations in RSA and LPI algorithms is shown in table 2 and table 3. Computational times for various message sizes (128bit, 256 bits, 512 bits, 1KB, 2KB and 5KB) is shown in fig 5. RSA algorithm is taking more time than the LPI algorithm.

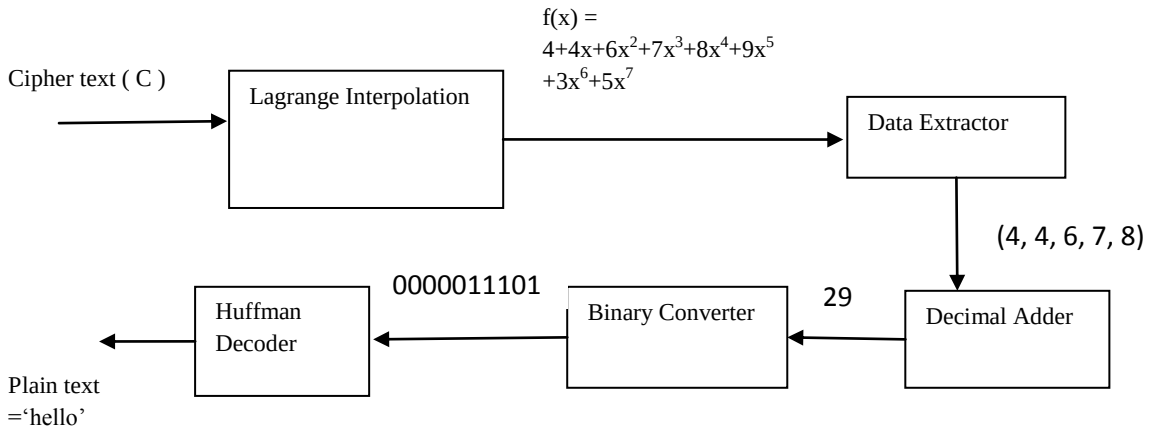


Fig 4: Decryption using LPI

Table 1: Computational time for various string lengths and N * L=String Length

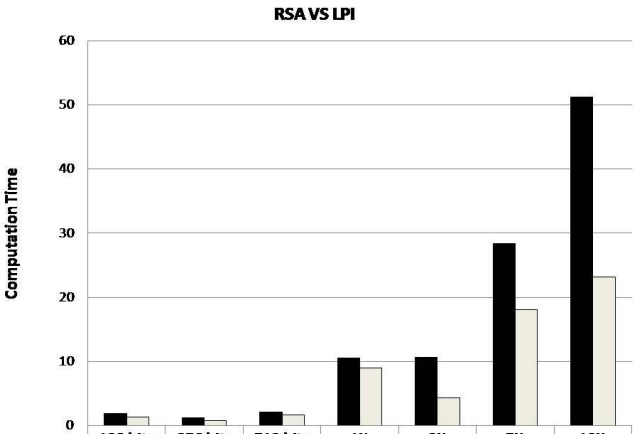
N Value	Computational Time				
	L=20	L=50	L=100	L=1000	L=5000
10	1.064484	1.740442	1.495543	4.325383	53.690359
20	1.363794	1.524745	1.769478	9.128155	45.752999
30	1.385233	1.875590	2.326145	10.327036	80.093865
50	1.812982	2.384235	2.847852	15.437109	212.134117
75	1.656579	3.096193	4.145984	21.088577	308.252668
100	1.766307	3.511301	4.324928	27.216914	274.386321
200	2.125207	5.890670	7.203134	49.691209	263.504649
300	3.673964	7.534049	9.384951	73.789074	300.691209
500	3.678756	7.933754	13.899052	191.427691	391.427691
1000	5.707674	13.977571	27.448549	255.360123	491.427691

Table 2: RSA and LPI Encryption Computation time (in Seconds)

Text Size	RSA	LPI
128 bits	0.107836	0.062900
256 bits	0.172001	0.022131
512 bits	0.250444	0.223889
1KB	4.891582	0.265534
2KB	9.409527	0.449042
5KB	27.034133	1.135417

Table 3: RSA and LPI Decryption Computation time (in Seconds)

Text Size	RSA	LPI
128 bits	0.052309	0.033177
256 bits	0.132727	0.027290
512 bits	0.041105	0.032280
1KB	5.192805	0.319827
2KB	8.478156	0.603114
5KB	25.863768	2.175781



	128 bits	256 bits	512 bits	1K	2K	5K	10K
RSA	1.848191	1.23347	2.156114	10.51523	10.62749	28.39546	51.18531
Lagrange	1.334908	0.7547	1.628188	8.958173	4.299975	18.08227	23.22732

Fig 5:RSA Vs LPI computation time

4. CONCLUSION

The problem in secured message transmission is studied and then, a new algorithm is proposed for secured message exchange based on mathematical techniques. Security to the message is providing selecting N value and by using random number generator. Confidentiality and integrity is achieved with this algorithm. Strong evidence has been supplied for practical implementation of message transfer. The advantage of this scheme is 1) using a simple polynomial interpolation calculations are used for message exchange, 2) can transfer massive messages between the communicating persons, 3) more secure because of usage of random function and 4) reducing the length of the code using Huffman technique.

5. REFERENCES

[1] Th. Sauer and Y. Xu. On multivariate Lagrange interpolation. Mathematics of computation Volume 64, pages 1147-1170, jul 1995.

[2] R.L.Rivest, A.Shamir, L.Adleman "A method for obtaining digital signatures and Public-Key Cryptosystems", Communications of the ACM Volume 21,pp:120-126, 1978.

[3] A.Nadeem, "A performance comparison of data encryption algorithms", IEEE information and communication technologies, pp.84-89, 2006.

[4] Hwang, R. J., Su, F. F., Yeh, Y. S. and Chen, C. Y. 2005. An Efficient Decryption Method for RSA Cryptosystem. *AINA'05* **Pp:**585- 590, *2005*.

[5] Davis, T. *RSA Encryption* [Online]. Available from World Wide Web: http://www.geometer.org /mathcircles, March 2006.

[6] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., pp 1098–1102, September 1952.

[7] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP "Chapter 7. Random Numbers". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8, 2007.

[8] R. Lidl and H. Niederreiter, *Finite fields*, 2nd edition, Cambridge University Press, Cambridge, 1997.

[9] Chia-Long Wu*, Der-Chyuan Lou, and Te-Jen Chang , "Computational Complexity Analyses of Modular Arithmetic for RSA Cryptosystem",23rd Workshop on Combinatorial Mathematics and Computation Theory (CMCT'06) , Taichung, Taiwan, pp. 215-224, April 28-29, 2006.

[10] Saurabh Singh, Gaurav Agarwal, "Use of Chinese Remainder Theorem to generate random numbers for cryptography",IJAER, pp.168-174 Nov 2010.

[11] Eugéne C.Ezin,"Modeling Data transmission through a Channel Based on Huffman coding and Encryption methods", IJCSIS, Vol 8, pp:195-199, No.9 December 2010.

[12] Publicly Verifiable Secret Sharing Schemes Using Bilinear Pairings, International Journal of Network Security, Vol.14, No.3, PP. 142:148, May 2012.