Proposal for Scrambled Method based on NTRU

Ahmed Tariq Sadiq
Computer Science Department
University of Technology
Baghdad, Iraq

Najlaa Mohammad Hussein
Computer Science Department
Baghdad University
Baghdad, Iraq

Suha Abdul Raheem Khoja
Electronic and communication
Engineering Department
Baghdad University
Baghdad, Iraq

***Abstract*—Scrambling is widely used to protect the security of data files such as text, image, video or audio files; however, it is not the most efficient method to protect the security of the data files. This article uses NTRU public key cryptosystem to increase the robustness of scrambling of sound files. In this work, we convert the sound file into text, and then scramble it in the following way: first, we encrypt the header of the sound file then, scramble the data of the file after the header in three stages. In each stage we scramble the data of the sound file and keep the original order of data in an array then, the three arrays are encrypted by the sender and sent with the encrypted header to the receiver in one file, while the scrambled data of the sound file is sent to the receiver in another file. We have tested the proposed method on several sound files; the results show that the time of encryption and decryption is reduced to approximately one-third, or less, compared to encrypting the file using NTRU.**

Keywords—NTRU; public key; cipher; sound; scramble; segment

I. INTRODUCTION

One of the classical encryption ciphers is transposition cipher. In transposition cipher, symbols of plaintext remain the same, but their original sequence is changed in a systematic way. Transposition ciphers are widely used before computer age. With the advent of technology the researchers have invented more complex ciphers like NTRU, which is a relatively new public key cryptosystem. NTRU was developed by three mathematicians J. Hoffstien, J. H. Silverman and J. Piper at the rump session of crypto in 1996. [1] NTRU (Nth degree truncated polynomial ring units) is the first secure public key cryptosystem not based on factorization or discrete logarithm problems. [2] It is a lattice-based public key cryptosystem; its security comes from the interaction of the polynomial mixing system with the independence of reduction modulo to relatively prime numbers.[3] The basic idea of NTRU is finding the shortest vector problem in a lattice. The base of NTRU operations are objects in a truncated polynomial ring R=z[x]/(xN-1) with convolution multiplication and all polynomials in the ring have integer coefficients and degree at most N-1. a = a0 + a1x + a2x2 + … + aN-2xN-2 + aN-1xN-1.[4]The main characteristic is that the polynomial multiplication is the most complex operation that is much faster than other asymmetric cryptosystems such as RSA, ElGamal and Elliptic curve cryptography.[5]

In this paper we introduce a new transposition (scrambling) method with the assistance of NTRU algorithm. In our method sound files will pass through more than one stage of scrambling. The result is a more complex permutation that is not easily reconstructed thus the sound files become significantly much more secure. [6]

II. RELATED WORK

Sadiq, Hussein and Khoja proposed two methods to enhance the NTRU algorithm which they have used for encrypting sound files after converting the sound into text. In the proposed methods the message is encrypted one character at a time. Since NTRU encrypts only prime numbers, only the first 7 bits of each character are encrypted. In method I NTRU algorithm is enhanced by adding the result obtained from calculating a mathematical equation of one variable to the message and then the resulted encrypted bit is fed-back and added to the next bit of the message in the next step; this procedure is repeated for the subsequent bits of the message. In method II NTRU algorithm is enhanced by adding the subsequent states of LFSR (Linear Feedback Shift Register) to the subsequent bytes of the message. The proposed methods are tested on several sound files; the results show that the proposed methods maintain approximately the same original method encryption and decryption time while generating a more complex encryption.[7] Jaspreet Kaur and Er. Kanwal Preet Singh [8] used three different kinds of algorithms NTRU, RSA and RINGDAEL for speech encryption and decryption by first converting the speech into text, and then the text is converted into cipher text. The performance is analyzed for these three algorithms respectively. The parameters calculated are encryption, decryption, delay time, complexity, packets lost and security levels. In these three algorithms, encryption decryption and delay time are varying according to the number of bits per second. On the other hand, complexity and packets lost are approximately the same. There are no packets lost during transmitting and receiving the data. Also Jaspreet Kaur and Er. Kanwal Preet Singh [9] used three different kinds of techniques i.e. MD-5, SHA-2 and RINGDAEL for speech encryption, where the speech is first converted into text then the text is converted into cipher text. At the end, the performances of these three techniques are analyzed, respectively.

III. NTRU

A. Prameters

NTRU depends on 3 integer parameter (N, p, q), where N is a prime integer, p and q are relatively prime integers and q is larger than p. [10]

B. Key Generation

To generate the public key, two random polynomials f and g are chosen in the ring R with the restriction that their coefficients are small, usually in {-1, 0, 1}. Another symbol is imported here: L(d1, d2), which means a set of polynomials with d1 coefficients are 1, d2 coefficients are -1 and the rest are 0. f usually chosen from Lf (df, df-1) and g from Lg (dg, dg). Then fp (the inverse of f modulo p) and fq (the inverse of f modulo q) are computed with the property that: f*fp=1(mod p) and f*fq=1(mod q). If f doesn't have inverses, another f should be chosen. The pair of polynomials f and fp should be kept as the private key and the public key h can be computed by h = p * fq * g (mod q). [11]

C. Encryption

The plaintext m is a polynomial with coefficients taken mod p. A random polynomial r is chosen with small coefficients. The cipher text is

e = r * h + m (mod q). [12]

D. Decryption

To decrypt e, the polynomial a is computed first

a = f * e (mod q)

The coefficients of a must be chosen from the interval [-q/2, q/2]. Then the original message can be computed by

m = fq * a (mod q). [10]

IV. THE PROPOSED METHOD

In this method the sender and receiver agree upon four steps of scrambling to the sound, according to random arrays that will be generated and encrypted by the sender and sent to the receiver in a metadata file. The sender then sends highly scrambled sound file to the receiver. This scrambling hides all the information in the file and makes it very difficult to predict and or discover without needing to NTRU encrypting the metadata file.

The sender starts the encryption process by encrypting the header of the sound file and writes it as the first part of the metadata file.

The sound file is split into a number of segments. The segment length (abbreviated SL) indicates the number of bytes in one segment. This length is chosen by the sender and it can vary due to the application and the size of the sound file. SL is encrypted and written to the metadata file.

Then the sender uses the proposed method to start the first step of scrambling the generated segments, which consists of dividing the segments of the file into parts (namely "Data Blocks", DB).The number of data blocks is encrypted using NTRU and written to the metadata file. Data blocks are then scrambled among themselves in a random way, to follow the order of a random array (namely, "Pointers to Data Block", PTDB) which has a length equal to the number of the data blocks; where the indexes of this array are indicators to the new order of the data blocks and the values of this array are used as indicators to the original order of the data block. This array is encrypted using NTRU and written to the metadata file.

Blocks will now be processed one at a time. The length of data blocks (abbreviated LDB) varies and is chosen randomly at run time to hide it. LDB for each data block is encrypted using NTRU and written to the metadata file.

The block size is chosen to satisfy a maximum value restriction to reduce the number of multiplications for both the encryption and the decryption operations as well as to decreases the size of the resulting encrypted file. The block size should also satisfy a minimum, as a very small DB cannot ensure enough scrambling. These maximum and minimum lengths are specified by the sender.

The second step of scrambling is applied to segments inside each data block among themselves to follow the order of a random array (namely "Pointers to Segments", PTS) which has a length equal to the number of segments in that data block, where the indexes of this array are indicators to the new order of segments inside the data block and the values of this array are used as indicators to the original order of segments within the data block. These arrays (one per data block) are encrypted using NTRU and written to the metadata file.

The third step of this method requires that, for each data block, the bytes inside each segment be scrambled among themselves according to a random array (namely," Pointers to Bytes within segment", PTB), which has a length equal to the number of bytes in the segment (SL). Where the indices of this array are indicators to the new order of bytes inside each segment and the values of this array are used as indicators to the original order of segments within the data block. The method uses the same scrambling (i.e. the same PTB) for all segments within one data block. Thus, only one PTB is required per data block. PTB are encrypted using NTRU and are written to the metadata file.

The highly scrambled sound data is written to a data file one byte from each segment at a time according to PTS array

Finally, the sender sends to the receiver the two generated files which are: the data file containing the scrambled sound data and the metadata file which contain the encrypted information.

Example: a very small size sound file of 74 bytes is used in this example to illustrate the proposed method. The ASCII of data inside the file after the header is "34 56 128 5 133 22 200 215 99 122 21 78 37 152 163 23 213 183 172 224 174 53 34 183 89 156 172 211 94 11", segment length is chosen to be 3 bytes, minimum Length of Data Block is 2 segments and maximum Length of Data Block is 6 segments.

The encryption process begins by encrypting the first 44 bytes, the header of the sound file using NTRU, then the four steps of scrambling are applied to the rest of the file, (the remaining 30 bytes of data). According to the method the file is divided into three data block, with the first block containing 3 segments, the second block containing 5 segments, and the last block containing 2 segments as illustrated in fig. (1)

Then the four stages of scrambling are applied as follows:

In the first step the order of the data block is changed where the second data block becomes the first, third data block becomes second and first data block becomes third, PTDB = [2, 3, 1], is used. This step is illustrated in fig. (2).

The second step in data scrambling changes the order of segments within each data block according to the block's PTS arrays as illustrated in fig. (3).

A PTS is randomly created for each data block to be:

1) *For the first Data Block, the PTS 1 is [2, 3, 1].*
2) *For second Data Block, the PTS 2 is [5, 3, 4, 1, 2].*
3) *For the third Data Block, the PTS 3 is [2, 1].*

The third step of data scrambling changes the order of bytes within data segments as shown in fig. (4), (PTBs), which was randomly created to be:

1) *For the first Data Block, PTB 1 is [2, 3, 1].*
2) *For the second Data Block, PTB 2 is [2, 1, 3].*
3) *For the third Data Block, PTB 3 is [3, 1, 2].*

The fourth and last scrambling step is write to the resulted data file one byte form each segment at time and with accordance to the block's PTB, i.e. for data block one with PTB = [2, 3, 1], the second bytes from all segments in that block are written first, then the third bytes from all the segments in that block are written, lastly the first bytes from all segments are written to the file. This step is shown in fig. (5), while the contents of the metadata file is shown in fig. (6)

V. Experimental Results

In this work we convert the sound file into text; we can apply this method to any sound file after storing it in a text editor. This method is applied to the original NTRU algorithm [13, 14] (namely original method) and the proposed method.

We convert the sound file into text via ISO-8859-1: 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1, is part of the ISO/IEC 8859 series of ASCII-based standard character encodings, which is intended for "Western European" languages [15].

We test the original and proposed methods on 25 wave sound files of sizes ranging from 10 KB to 1MB. The encryption and decryption time in seconds is computed for each one of the 25 files 25 times, and then average of computation is taken to enhance the accuracy of the computation.

Fig. (7) Displays the effect of file size on the time of encryption and decryption of the original method, Fig. (8) Displays the effect of file size on the time of encryption and decryption of the proposed method with SL 8, Fig. (9) Displays the effect of file size on the time of encryption and decryption of the proposed method with SL 12, Fig. (10) Displays the effect of file size on the time of encryption and decryption of the proposed method with SL 16, Fig. (11) Displays the relationship between file size and encryption time to the original and proposed methods and Fig. (12) Displays the relationship between file size and decryption time to the original and proposed methods.

VI. Conclusions and Future Work

The proposed method scrambles the sound file in a complex and efficient manner while reducing the time of encryption and decryption depending on SL. When SL equals 8, the time is reduced to approximately one-third of the time needed to encrypt and decrypt the same data using NTRU encryption method. Increasing SL to 12 and 16 had an effect of decreasing the time of encryption and decryption (up to 80% of the original encryption and decryption time).

For future research, we proposed a method that combines between NTRU algorithm and ECC algorithm to produce a new stronger cipher system that incorporates the advantages of the two algorithms.

References

[1] Ranjan, R., Baghel, A. S., Kumar, S., "Improvement of NTRU Cryptosystem", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 9, September 2012.

[2] Narasimham, C., Pradhan, J., "Evaluation of Performance Characteristics of Cryptosystem Using Text Files" Journal of Theoretical and Applied Information Technology, JATIT, 2008.

[3] Challa, N., Pradhan, J., "Performance Analysis of Public Key Cryptographic Systems RSA and NTRU" International Journal of Computer Science and Network Security. Vol.7 No. 8 August 2007, pp 87- 96.

[4] Kumar, R. G. V. S., Jumar, N. K., Sekhar, C. P., Numma, B. V. V. S., Kumar, V. B., "Modified Mutual Authentication and Key Agreement Protocol Based on NTRU Cryptography for Wireless Communications", International Journal of Computer Science and Network (IJCSN), Volume 1, Issue 4, August, 2012.

[5] Gupta, N., Ghosh, D., "Implementation of NTRU PKCS using Array for polynomials up to degree 147 with private key protection algorithm using XOR function only", IEEE 2008.

[6] Stallings, W., "Cryptography and Network Security Principles and Practice", Fifth Edition, Pearson, Boston, 2011.

[7] Sadiq, A. T., Hussein, N. M., Khoja, S. A., " Proposal for Two Enhanced NTRU", International Journal of Advanced Computer Science and Applications, Vol. 5, No. 5, 2014.

[8] Kaur, J., Singh, K. p. "Comparative Study of Speech Encryption Algorithms Using Mobile Applications", International Journal of Computer Trends and Technology, Vol.4, Issue. 7, Jully 2013, pp. 2346 - 2350.

[9] Kaur, J., Singh, K. p. "Speech to Text Encryption Using Cryptography Techniques", International Journal of Innovative Research and Development, Volume 2, Issue. 4, pp. 274 - 283, April 2013.

[10] Jha, R., Saini, A, K., "A Comparative Analysis and Enhancement of NTRU Algorithm for Network Security and Performance Improvement" International Conference on Communication Systems and Network Technologies, 2011.

[11] Shen, X., Du, Z., Chen, R., "Research on NTRU Algorithm for Mobile Java Security" IEEE 2009, pp.366 - 369.

[12] Wei, S., Zhuo, Z., "Research on PKI Model Based on NTRU", International Symposium on Electronic Commerce and Security, IEEE, 2008.

[13] O'Rourke, C. M., "Efficient NTRU Implementations" Thesis, April, 2002.

[14] Yadav, S. K., Bhardwaj, K.,"On NTRU Implementation: An Algorithmic Approach", Proceedings of the 4th National Conference; INDIA, 2010.

[15] ISO /IEC JTC 1/SC 2/WG 3 7bit and 8bit codes and their extension SECRETARIAT: ELOT, 1998.

First Data Block	First Segment	First byte	34
		Second byte	56
		Third byte	128
	Second Segment	First byte	5
		Second byte	133
		Third byte	22
	Third Segment	First byte	200
		Second byte	215
		Third byte	99
Second Data Block	First Segment	First byte	122
		Second byte	21
		Third byte	78
	Second Segment	First byte	37
		Second byte	152
		Third byte	163
	Third Segment	First byte	23
		Second byte	213
		Third byte	183
	Fourth Segment	First byte	172
		Second byte	224
		Third byte	174
	Fifth Segment	First byte	53
		Second byte	34
		Third byte	183
Third Data Block	First Segment	First byte	89
		Second byte	156
		Third byte	172
	Second Segment	First byte	211
		Second byte	94
		Third byte	11

Fig. 1. Dividing the file in to segments and then to three data block

Second Data Block	First Segment	First byte	122
		Second byte	21
		Third byte	78
	Second Segment	First byte	37
		Second byte	152
		Third byte	163
	Third Segment	First byte	23
		Second byte	213
		Third byte	183
	Fourth Segment	First byte	172
		Second byte	224
		Third byte	174
	Fifth Segment	First byte	53
		Second byte	34
		Third byte	183
Third Data Block	First Segment	First byte	89
		Second byte	156
		Third byte	172
	Second Segment	First byte	211
		Second byte	94
		Third byte	11
First Data Block	First Segment	First byte	34
		Second byte	56
		Third byte	128
	Second Segment	First byte	5
		Second byte	133
		Third byte	22
	Third Segment	First byte	200
		Second byte	215
		Third byte	99

Fig. 2. The first step of scrambling (scrambling the dbs)

		First byte	53
		Second byte	34
	Fifth Segment	Third byte	183
		First byte	23
		Second byte	213
	Third Segment	Third byte	183
		First byte	172
		Second byte	224
	Fourth Segment	Third byte	174
		First byte	122
		Second byte	21
	First Segment	Third byte	78
		First byte	37
		Second byte	152
Second Data Block	Second Segment	Third byte	163
		First byte	211
		Second byte	94
	Second Segment	Third byte	11
		First byte	89
		Second byte	156
Third Data Block	First Segment	Third byte	172
		First byte	5
		Second byte	133
	Second Segment	Third byte	22
		First byte	200
		Second byte	215
	Third Segment	Third byte	99
		First byte	34
		Second byte	56
First Data Block	First Segment	Third byte	128

Fig. 3. The second step of scrambling (scrambling segments within dbs)

		Second byte	34
		First byte	53
	Fifth Segment	Third byte	183
		Second byte	213
		First byte	23
	Third Segment	Third byte	183
		Second byte	224
		First byte	172
	Fourth Segment	Third byte	174
		Second byte	21
		First byte	122
	First Segment	Third byte	78
		Second byte	152
		First byte	37
Second Data Block	Second Segment	Third byte	163
		Third byte	11
		First byte	211
	Second Segment	Second byte	94
		Third byte	172
		First byte	89
Third Data Block	First Segment	Second byte	156
		Second byte	133
		Third byte	22
	Second Segment	First byte	5
		Second byte	215
		Third byte	99
	Third Segment	First byte	200
		Second byte	56
		Third byte	128
First Data Block	First Segment	First byte	34

Fig. 4. The third step of scrambling (scrambling bytes within segments)

	Fifth Segment		34
	Third Segment		213
	Fourth Segment		224
	First Segment		21
	Second Segment	Second byte	152
	Fifth Segment		53
	Third Segment		23
	Fourth Segment		172
	First Segment		122
	Second Segment	First byte	37
	Fifth Segment		183
	Third Segment		183
	Fourth Segment		174
	First Segment		78
Second Data Block	Second Segment	Third byte	163
	Second Segment		11
	First Segment	Third byte	172
	Second Segment		211
	First Segment	First byte	89
	Second Segment		94
Third Data Block	First Segment	Second byte	156
	Second Segment		133
	Third Segment		215
	First Segment	Second byte	56
	Second Segment		22
	Third Segment		99
	First Segment	Third byte	128
	Second Segment		5
	Third Segment		200
First Data Block	First Segment	First byte	34

Fig. 5. The fourth step of scrambling which results in the order of data that will be written to the data file

1		encryption of the header of the file	
2		encryption of the Segment length (3)	25 1063 579 868 1299 926 1387 1
3		encryption of the number of Data Block (3)	25 1063 579 868 1299 926 1387 1
4		encryption of Pointer to Data Block [2, 3, 1]	25 1063 579 868 1299 926 1387 0 25 1063 579 868 1299 926 1387 1 25 1063 579 868 1299 926 1386 1
5.a	Second DB	encryption of the actual length of Data Block (5)	25 1063 579 868 1299 927 1386 1
5.b		encryption of PTB [2,1,3]	25 1063 579 868 1299 926 1387 0 25 1063 579 868 1299 926 1386 1 25 1063 579 868 1299 926 1387 1
5.c		encryption of PTS [5,3,4,1,2]	25 1063 579 868 1299 927 1386 1 25 1063 579 868 1299 926 1387 1 25 1063 579 868 1299 927 1386 0 25 1063 579 868 1299 926 1386 1 25 1063 579 868 1299 926 1387 0
5.a	Third DB	encryption of the actual length of Data Block (2)	25 1063 579 868 1299 926 1387 0
5.b		encryption of PTB [3,1,2]	25 1063 579 868 1299 926 1387 1 25 1063 579 868 1299 926 1386 1 25 1063 579 868 1299 926 1387 0
5.c		encryption of PTS [2,1]	25 1063 579 868 1299 926 1387 0 25 1063 579 868 1299 926 1386 1
5.a	First DB	encryption of the actual length of Data Block (3)	25 1063 579 868 1299 926 1387 1
5.b		encryption of PTB [2,3,1]	25 1063 579 868 1299 926 1387 0 25 1063 579 868 1299 926 1387 1 25 1063 579 868 1299 926 1386 1
5.c		encryption of PTS [2,3,1]	25 1063 579 868 1299 926 1387 0 25 1063 579 868 1299 926 1387 1 25 1063 579 868 1299 926 1386 1

Fig. 6. The contents of the metadata file

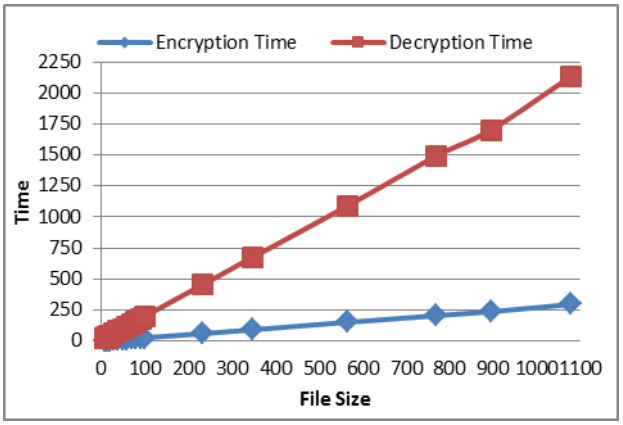

Fig. 7. The effect of file size on the time of encryption and decryption of the original method

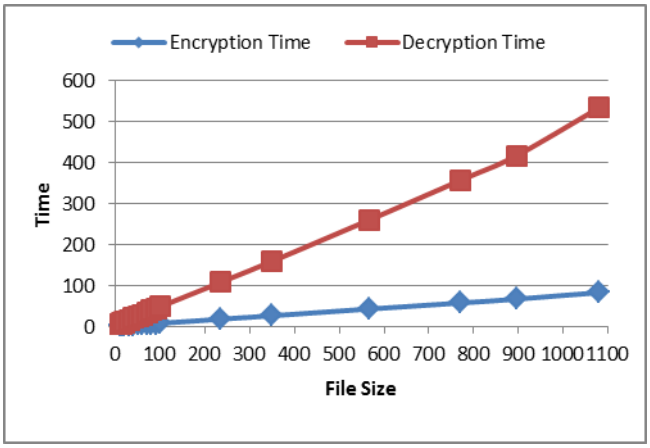

Fig. 8. The effect of file size on the time of encryption and decryption of the proposed method with sl 8

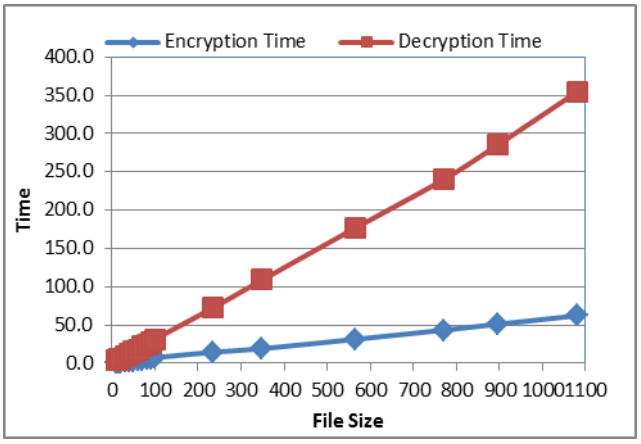

Fig. 9. The effect of file size on the time of encryption and decryption of the proposed method with sl 12

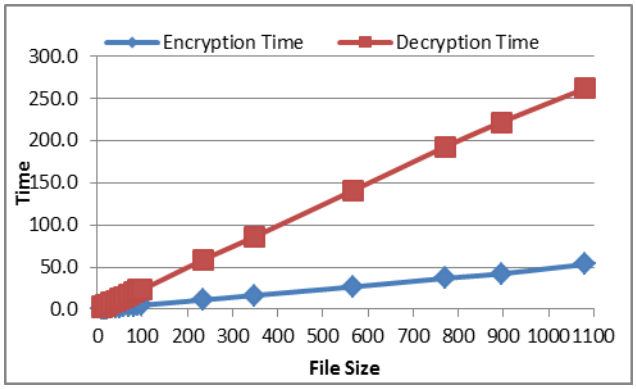

Fig. 10. The effect of file size on the time of encryption and decryption of the proposed method with sl 16

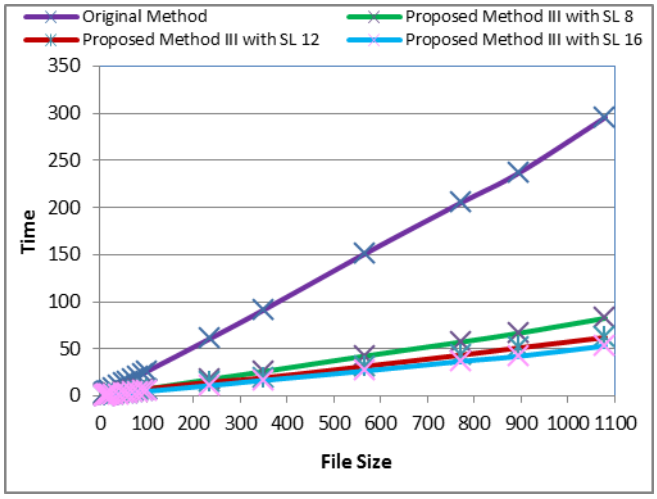

Fig. 11. Compression for the effect of file size on time of encryption between the original method and the proposed method

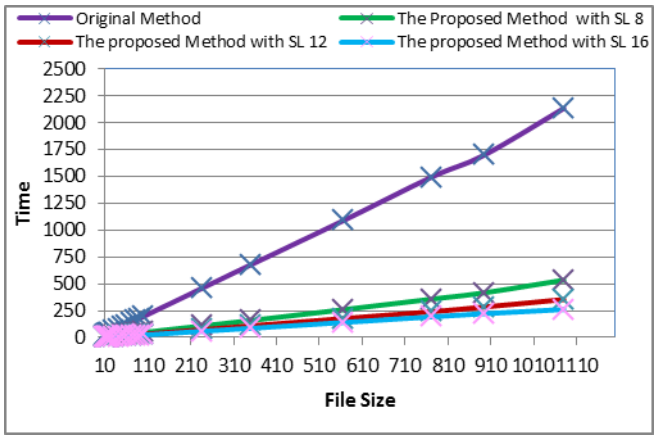

Fig. 12. Compression for the effect of file size on time of decryption between the original method and the proposed method