International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Enhancing Security in RSA Cryptosystem Using Burrows-Wheeler Transformation and Run Length Encoding

A. Devi*[1], K. Mani[2]

*[1]Department of Computer Science, Bangalore University/KCMS College, Bangalore, Karnataka State, India

[2]Department of Computer Science, BharathidasanUniversity/NMC, Thiruchirapalli, Tamil Nadu State, India

ABSTRACT

While transmitting the plaintext, to speed up the transmission and eliminating redundancy, it is necessary to compress the text. As the original form of plaintext is altered when compression is used, the eavesdropper may not easily be cracked the correct plaintext which produces first level of security. Once the compression has been done, encoding is performed to form the different format so that fewer bits will be used to represent the original plaintext thereby size of the original text is reduced which produces the second level of security. For that the Burrows-Wheeler Transform lossless compression technique is used in this paper to transform the plaintext and the transformation permutes the order of characters. To reduce the redundancy and to increase the efficiency of algorithm, move-to-front transformation is performed in BWT. Further, the transformed code is again compressed using run length encoding and then it is encrypted using RSA public-key algorithm in which block size is determined dynamically in this paper. The experimental results clearly show that the increasing of encryption and decryption time and enhancing the security of RSA.

Keywords: BWT, Move-to-Front, Run-Length Encoding, RSA.

I. INTRODUCTION

Encryption ensures security to access confidential data by an authorized recipient and avoid accessing the data from third party. Encryption is used to protect data when transmitting the data across networks against eavesdropping of network traffic by unauthorized users [12]. It is also used to protect sensitive information by encoding and transforming the information into an unreadable ciphertext and the ciphertext may be decrypted into a readable form using key. Secret-key and public-key cryptography[13] are the two types of cryptosystems in which secret-key cryptography uses the same key for both encryption and decryption whereas in public-key cryptography, secret-key and public-key is used by each user. The public-key is used for encryption and the secret-key is used for decryption. The most popular public-key algorithm is RSA public-key cryptosystem. The design of a robust encryption algorithm in cryptanalysis is used to find and correct any weaknesses [11]. Encryption provides only security but not increasing the transmission speed [9]. To increase the speed, the data compression technique is efficient to remove the redundant character strings in a file in which the compressed file has uniform distribution of characters and it provides shorter ciphertext. It also reduces the time to encrypt and decrypt the message.

There are three types of compression models available namely static, semi-static and adaptive. Static model does not depend on the data which is being compressed and it is a fixed model known by compressor and decompressor[7]. Semi-static model is constructed from the data to be compressed and it

is also a fixed model. An adaptive model is a function of previously compressed part of the data. The Burrows-Wheeler transformation [8] developed by Michael Burrows and David Wheeler in 1983, also called as block sorting and it is based on a permutation of the input sequence. Compression and indexing are the two important applications in BWT. In [10] BWT, first the entire sequence is to be coded. For that the original sequence must be cyclically shifted from right to left and arranged in the form of lexicographic order. Then, the sequence is converted into binary and Run-Length Encoding (RLE) is applied to increase the security level in RSA algorithm. The compressed code is applied for RSA public-key algorithm to encrypt the code.

The rest of the paper is organized as follows. Background and related work is discussed in section 2. Section 3 presents the proposed methodology in BWT with Move To Front (mtf). An example for proposed methodology is illustrated in section 4. Experimental results are discussed in section 5. Finally, section 6 ends with conclusion.

II. BACKGROUND AND RELATED WORK

Data Compression Methodologies for Lossless Data and Comparison between Algorithms presented by S.Porwal, Y.Chaudhary, J.Joshi, M. Jain[1]. They compared the performance of Huffman and Arithmetic encoding. After comparison of two techniques they concluded that the compression ratio for arithmetic encoding is better than the Huffman encoding and also found that the channel bandwidth and time is reduced much better than Huffman encoding. But the compression speed is very less in arithmetic encoding than the Huffman coding. U.Khurana and A.Koul[2] provides Text Compression And Superfast searching and they proved that it is an efficient technique providing high compression ratios and faster search through the text. S.Kaur and V.S.Verma[3] implemented a Design and Implementation of LZW Data Compression Algorithm. The author has implemented a finite state machine by using LZW data compression algorithm and he proved that text data is effectively compressed. A Data Compression using Huffman based LZW Encoding Technique is presented by Md.RubaiyatHasan[4] for transmitting a digital image from a digital data source to a digital data receiver. The author has proved that it provides better transmission speed and saves time. Rajan.S.Jamgekar et.al[5] implemented a File Encryption and Decryption Using Secure RSA. It shows that MREA algorithm is used to encrypt files and transmit encrypted files to other end where it is decrypted. But it works for smaller file size whereas it takes more time for larger file size. In [6], Monisha Sharma et.al described about a novel Approach of Image Encryption and Decryption by using partition and Scanning Pattern. The author has proposed a lossless encryption of image and also gives access to variable lengths of the encryption keys.

III. PROPOSED METHODOLOGY

The proposed methodology consists of three phases. In phase 1, the proposed mtf-BWT is used in performing the compression of plaintext. RLE is used in phase 2 for further compression of compressed text obtained in phase 1. Then, the decimal form of binary encoded compressed form of plaintext is encrypted using dynamically generated block size of RSA in phase 3.

3.1 Proposed MTF-BWT

This phase consist of three steps viz., Shifting of plaintext characters towards left circularly, arrange them in lexicographic order and formation of move to first(mtf) .

3.1.1 Shifting Of Plaintext Characters Towards Left

Let M be a plaintext, and $m_i \in M$, i=1,2,3,…,n where n denotes the total number characters in M(length of M). Suppose there are totally d distinct characters denoted as $m_1,m_2,..,m_d$ with $m_1 \neq m_2 \neq \ldots \neq m_d$ in M and there are $n_1,n_2,\ldots,n_d$ times distinct characters occur respectively. Then M=

$$\sum_{i=1}^{d} ni = n$$

Fill $m_i \in M$ in tabular form and shift one position towards left circularly as shown in Table 1.

Table 1. ORIGINAL MESSAGE m_i<<<1 , i-1,2,…,n

0	m_1	m_2	m_3	…	m_{n-1}	m_n
1	m_2	m_3	m_4	…	m_n	m_1
2	m_3	m_4	m_5	…	m_1	m_2
…	…	…	…	…	…	…
n-2	m_{n-1}	m_{n-2}	m_{n-3}	…	m_{n-1}	m_{n-2}
n-1	m_n	m_{n-1}	m_{n-2}	…	m_2	m_1

It is the original BWT which is modified as right shift circularly shown in Table 2.

Table 2. SHIFTING m_i>>>1

0	m_1	m_2	m_3	…	m_{n-1}	m_n
1	m_n	m_1	m_2	…	m_{n-2}	m_{n-1}
2	m_{n-1}	m_{n-2}	m_1	…	m_{n-3}	m_{n-2}
…	…	…	…	…	…	…
n-2	m_3	m_4	m_5	…	m_1	m_2
n-1	m_2	m_3	m_4	…	m_n	m_1

Table 2 shows the message $m_i \in M$ is shifted one position towards right circularly. In row 0 of Table 2 consists of original M. Shift one position towards right circularly all the elements in 0^{th} row so that row 1 is obtained. In general $(n-1)^{th}$ row is obtained by shifting each element of $(n-2)^{th}$ row one position towards circular right. The steps involved in algorithm1 show that the order of sequence will be different from original BWT.

Algorithm 1: Circular shifting of plaintext characters towards right

```
String Function shift_right(M)
begin{Main}
    1.  L←length(M)
    2.  for i ←1 to L
    begin
        2.1   i←1
        2.2   while(i ≤ L) then
        begin
        2.3   if( i==L) then
          M[1] ←M1[i]
             //M1[i] ←M[1]
          else
            M[i+1] ←M1[i];i+1
            M1[i] ←M[i+1];i←i+1
        end {if}
        2.4   end {while}
        2.5   print(M)
        2.6   M1←M
    3.  end {for i}
    4.  return M1
end {main}
```

3.1.2 Arrange Them In Lexicographic Order

After forming Table 2, sort the Table 2 in lexicographical order. For that first identify the number of blank spaces b_k in M. If the number of blank space occurs in M is k, then the number of words in M is k+1. Let the blank spaces are denoted as $b_1,b_2,\ldots,b_k$ and the blank spaces $b_i \in M$ occurs at position p_i, i=1,2,…,k. After that identify the distinct alphabets where d≤ n from M which is shown in algorithm 2 and sort them in ascending order.

Algorithm 2: Finding the distinct characters from M

```
int position function distinct(M)
begin
    1.  L←length(M)
    2.  U←sort(unique(M))
    3.  j←1; k←1
    4.  for i in 1 to length(L_U)
     begin
        4.1   for j in 1 to length(M)
        begin
      4.2 if L_U[i]=M[j]
                pos←j
                posa[k]←pos
                k←k+1
            end {if}
        4.3   end {for j}
    5.  end {for i}
    6.  return posa
```

end {main}

Let the alphabet a_l occurs at position l_k, k=1, 2,…,d with $p_i \neq l_k$. Further, $a_1 < a_2 < \ldots < a_d$. Now start with b_1 at p_1, put all the characters that appear after b_1, proceed up to the last character before b_1 occurs at position p_1-1and fill them in 0^{th} row of Table 3.

Table 3. Lexicographical Order

0	b_1	m_{p1+1}	m_{p1+2}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p1-1}
1	b_2	m_{p2}	m_{p2+1}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p2-1}
…	…	….	…	…	….	…	…	…	…	…
p_{n-1}	b_n	m_{pn}	m_{pn+1}	…	m_{n-1}	m_n	m_1	m_2	…	m_{pn-1}
P_n	m_{p2}	m_{p2+1}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p2-1}	b_2
…	…	….	…	…	….	…	…	…	…	…
p_{i-1}	b_{i-1}	m_{pi-1}	m_{pi}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p-1-1}
p_{i-1+1}	m_{pi-1}	m_{pi}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p-1-1}	b_{i-1}
…	…	…	…	…	…	…	…	…	…	…
p_i	b_i	m_{bi}	m_{bi+1}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p-1}
p_{i+1}	m_{bi}	m_{bi+1}	…	m_{n-1}	m_n	m_1	m_2	…	m_{p-1}	Bi

Similarly, start with b1, proceed up to the position p_2-1, fill the corresponding characters in 1^{st} row. Repeat the said process for other blank spaces too and the process in terminated when the last b_k is processed. Once all b_k are processed now start with a_l and repeat the said process till a_d are processed. It is shown in Table 3 and it is shown in algorithm 3.

Algorithm3: Forming the source character in Lexicographical Order

String Lexi Function Lexicographic(M)

begin {main}

1. posa←distinct(M)
2. l←length(posa)
3. k←1
4. for i in 1 to l
   begin
   4.1 st←posa[i]+1
   4.2 st←posa[i]+1
   end{for i}
5. if st ≤ end
   k←k+1
   lex[k] ←M[st]
   st←st+1
   end {if}
6. for j=1 to posa[i]-1
   begin
   6.1 k←k+1
   6.2 lex[k] ←M[j]
   end{for j}
7. Loop←1
8. for m←1 to l
   begin
   8.1 x[Loop][m] ←lex[m]
   8.2 Loop←Loop+1
   end {for m]
9. return x[Loop][m]

end {main}

The main difference between the existing and proposed methodology is that in the proposed methodology, first sequence is always considered for mtf whereas in the existing methodology the last sequence is considered.

Form a table with order n**r** x n**d,** Where **nr ≥ nd** and **nr** is determined on the basis of first sequence of lexicographical order. The source alphabet is $\{n_b, m_1, m_2, \ldots, m_d\}$. To fill the value in row 1, the top

position of (0, 0) must be filled by b_1 (the first blank space) and the succeeding columns in rows r_1 are filled with other alphabets from source alphabet. Search the position of first alphabet in the previous row and move the element from that position to the top position, i.e., (1, 0) and fill the rest of the alphabet in the succeeding position. The process is repeated until all the n distinct elements are filled from the source sequence. In order to determine the final sequence, the elements must be considered from the moving position of previous row to the top position in the same row. Now, the final sequence is represented as n_1, n_2, n_3,…,n_n. It is noted that the number of characters in final sequence stored as same. It is shown in algorithm 4.

Algorithm4: For finding the move to front

```
String FS move-to-front(mtf)
begin{main]
 1. ds←sort(unique(M))
 2. dsp←unique-pos(ds);
 3.  p←0; j←0; en←0;
 4.  for k ← 0 to length(fs)-1
    begin
     4.1 if k=dsp[j] and k<length(dsp) then
       begin
            enc[p] ←en
            j←j+1; en←en+1;
        else
              enc[p] ←0; p←p+1
       end {if}
   end{for k}
 3.  return enc
end {main}
```

After finding the first sequence, it is converted into binary which is then used in RLE.

3.2 Run Length Encoding (RLE)

RLE is a technique used to reduce the size of repeating string of characters. The repeating string is called run. It replaces sequences of the same data values within a file by a count number and a single value. For example, the bit stream , 11111111111111100000000000000000001111 is compressed using RLE as 15119041. In the proposed method, mtf based BWT and RLE are used before encryption. For performing encryption and decryption, the standard RSA is used and the proposed methodology is now termed as RLE-EBWT-RSA.

3.3 RLE-EBWT-RSA

In original RSA, the block size taken is either 2048 or 4096 bits and it is always constant. In the modified RSA block size is determined on the basis of modulus m, $m= p \times q$ and it depends on the size of p and q where p and q are prime numbers. As p and q are accepted as input, the block size is based on only m and the block size denoted as BS is unpredictable which produces the first level of security. Further, original P is completely changed using the proposed methodology before it is encrypted the eavesdropper may not predict the correct P from C which increases second level of security. The steps involved in modified RSA algorithm are shown in algorithm 5.

Algorithm 5: Modified RSA algorithm based on mtf based BWT

```
int IP  function  modified_RSA(M)
begin {main}
// Te first 6 steps for key generation part, step 7 is
//generating IP after using the proposed
methodology. 8th step is performed by the sender and
9th is performed by receiver
```

1. Generate two large distinct primes p, and q, most probably both are of same size
2. Compute $n \leftarrow p \times q$; $\phi(n)=(p-1)\times(q-1)$
3. convert n into binary $nb \leftarrow (n)_2$
4. Compute BS ← length(nb) //BS-block size
5. Select the random integer e, $1 < e < \phi(n)$) such that $\gcd(e, \phi(n))=1$
6. Find d, $d \times e \equiv (1 \bmod \phi(n))$
7. IP ← RLE(BWT_mtfM))
8. i←BS

   8.1 repeat

   begin

Locate i digits from IP, say IP_i

To encrypt, compute $C_i \leftarrow IP_i^e \bmod n$,

$(1< IP_i < n)$

To decrypt, compute $IP_i \leftarrow C_i^d \bmod n$

return IP_i

$i \leftarrow i+BS$

end {repeat}

8.2 until ($i \leq$length(IP))

end{main}

After getting IP by the receiver the steps from 3.1 to 3.3 are reversed, the receiver gets the original plaintext.

IV. PROPOSED METHODOLOGY – AN EXAMPLE

In order to understand the relevance of the work, let M is "KANNAN BABA". Now d=5 with m_1='K', m_2='A', m_3='N', m_4=" ", m_5="B" and n_1=1, n_2=4,n_3=4,n_4=1, n_5=2. Then sorted order of m_i, i=1,2,..,5 is {blank, A, B, K, N}. Since n==11, form the original BWT matrix of order 11×10. First, fill M in 0^{th} row then shift one position towards left circularly. The process is repeated for the last character "A" in M. It is shown Table 4.

Table 4. Original Bwt

0	K	A	N	N	A	N	ϕ	B	A	B	A
1	A	N	N	A	N	ϕ	B	A	B	A	K
2	N	N	A	N	ϕ	B	A	B	A	K	A
3	N	A	N	ϕ	B	A	B	A	K	A	N
4	A	N	ϕ	B	A	B	A	K	A	N	N
5	N	ϕ	B	A	B	A	K	A	N	N	A
6	ϕ	B	A	B	A	K	A	N	N	A	N
7	B	A	B	A	K	A	N	N	A	N	ϕ
8	A	B	A	K	A	N	N	A	N	ϕ	B
9	B	A	K	A	N	N	A	N	ϕ	B	A
10	A	K	A	N	N	A	N	ϕ	B	A	B

After forming Table 4, again fill M in 0^{th} row of Table 5. Start with first alphabet in sorted list i.e., A in row 0, fill the next row of table 5 shifting towards circularly right row 0 so that row 1 is obtained. In row 1, now the next alphabet in sorted order is B. Start with B fills all the characters which occur in row 1 by shifting towards right circularly. The process is repeated until the last character "N" is processed. The resultant is shown in Table 5.

Table 5. Permutation For Modified Bwt

0	K	A	N	N	A	N	ϕ	B	A	B	A
1	A	K	A	N	N	A	N	ϕ	B	A	B
2	B	A	K	A	N	N	A	N	ϕ	B	A
3	A	B	A	K	A	N	N	A	N	ϕ	B
4	B	A	B	A	K	A	N	N	A	N	ϕ
5	ϕ	B	A	B	A	K	A	N	N	A	N
6	N	ϕ	B	A	B	A	K	A	N	N	A
7	A	N	ϕ	B	A	B	A	K	A	N	N
8	N	A	N	ϕ	B	A	B	A	K	A	N
9	N	N	A	N	ϕ	B	A	B	A	K	A
10	A	N	N	A	N	ϕ	B	A	B	A	K

The second step is to sort the sequence in the lexicographical order shown in table 6. Table 6, consists of the cyclically shifted sequences which are in the lexicographical order.

Table 6. Lexicographic Order For Modified Bwt

0	ϕ	B	A	B	A	K	A	N	N	A	N
1	A	B	A	K	A	N	N	A	N	ϕ	B
2	A	K	A	N	N	A	N	ϕ	B	A	B
3	A	N	ϕ	B	A	B	A	K	A	N	N
4	A	N	N	A	N	ϕ	B	A	B	A	K
5	B	A	B	A	K	A	N	N	A	N	ϕ
6	B	A	K	A	N	N	A	N	ϕ	B	A
7	K	A	N	N	A	N	ϕ	B	A	B	A
8	N	ϕ	B	A	B	A	K	A	N	N	A
9	N	A	N	ϕ	B	A	B	A	K	A	N
10	N	N	A	N	ϕ	B	A	B	A	K	A

Table 7 shows the first and last sequence of the sorted elements.

Table 7. First And Last Sequence Of Elements

F_i	ϕ	A	A	A	A	B	B	K	N	N	N
L_i	N	B	B	N	K	ϕ	A	A	A	N	A

i- Element of First and Last sequence

Further, the sequence is applied to mtf coding scheme to encode the sequence. For that consider the first sequence F= "ϕAAAABBKNNN". Normally, the

last sequence L is considered for mtf. But, in the proposed methodology, the first sequence F is considered so that the number of moving position of the each element to the 0^{th} position is very less. This is because F gets the repeated element several times. The source alphabet A based on F is { ϕ,A,B,K,N }. In this coding scheme, the first element is blank space which is already in the top position and there is no change in the position. Thus, 0 is placed in the sequence. The second element is A from the sequence moving from first position into the top of the list. This is encoded as 1.The next subsequent three elements from sequence are A and hence it is not necessary to encode the those elements instead and three zeros in the sequence. The next element is B moving towards the top of the list from second position and it is encoded as 2. Again B is repeated and it is considered as zero. After B, the element K is moving towards the top of the list and it is encoded as 3.The last three elements from the sequence is N which is moving towards the top of the list from the position four and it is encoded as 4.The two elements out of three are zeros. The final sequence from this coding scheme is 01000203400. The mtf for first sequence and last sequence are shown in Table 8 and Table 9 respectively.

Table 8. Mtf Coding Scheme For First Sequence

NO.	I	II	III	IV
0	ϕ	A	B	K
1	A	ϕ	A	B
2	B	B	ϕ	A
3	K	K	K	ϕ
4	N	N	N	N

Table 9. Move To Front Coding Scheme For Last Sequence

NO.	I	II	III	IV	V	VI	VII	VIII	IX
0	ϕ	N	B	N	K	ϕ	A	N	A
1	A	ϕ	N	B	N	K	ϕ	A	N
2	B	A	ϕ	ϕ	B	N	K	ϕ	ϕ
3	K	B	A	A	ϕ	B	N	K	K
4	N	K	K	K	A	A	B	B	B

The first sequence is "ϕAAAABBKNNN". Once the mtf for the first sequence is 01000203400. After applying RLE, it is compressed as 0130203420. It is treated as IP plaintext for encryption. Similarly, the last sequence is "NBBNKϕAAANA" and its mtf is 43014340031 and its compressed form is 4301433201. It is observed that the number of digits are more in last sequence than the number digits in first sequence.

To perform encryption and decryption using modified RSA, let p=50053 and q=50069. Then n=2506103657, ϕ(n)= 2506003536, $(n)_2$=1001 0101 0110 0000 0001 1011 0110 1001. Thus, block size BS=32 bits. Let e= 56989. Using Extended Euclidean Algorithm d is computed as 2472671653. Let M="KANNAN BABA". Using the proposed methodology $IP_1(M)$ =0130203420 where the number of bits for IP=27. As 27< 32, BS=1. To encrypt C_1 =$(130203420)^{56989}$ mod 2506103657 =638232499 and to decrypt IP_1=$(638232499)^{2472671653}$ mod 2506103657=130203420. After obtaining IP, the reverse process, i.e., decompression and decoding are performed to get M="KANNAN BABA".

V. RESULTS AND DISCUSSIONS

The proposed methodology is implemented in VC++ where in which different size of plaintext is taken. The security level produced by original RSA, BWT-RSA, RLE-BWT-RSA and RLE-EBWT-RSA are measured using All Block Cipher (ABC) Universal Hackman tool. Also the time taken for encryption and decryption for different file sizes are noted without using modified BWT and after using modified BWT and they are shown in Table 10 and Table 11 respectively and their corresponding graphical representation are shown in Figure 1, Figure 2, Figure 3 and Figure 4 respectively.

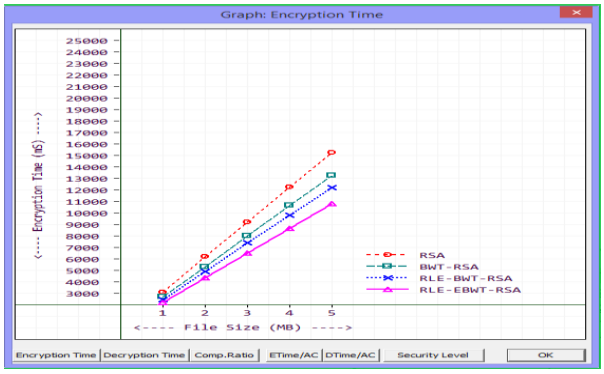

Figure 1. Encryption Time Before Modified BWT

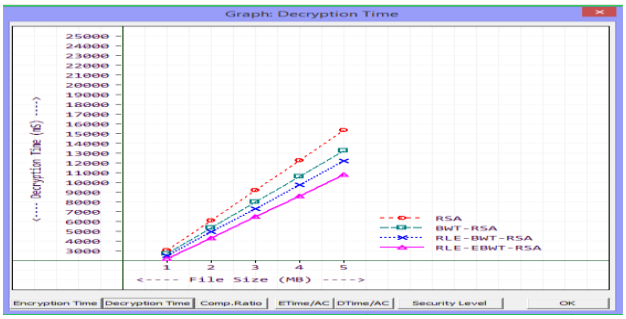

Figure 2. Decryption Time Before Modified BWT

The encryption time and decryption time for RSA algorithm after using modified BWT is analyzed for various sizes of plaintext. The encryption time and decryption time of 1MB file size is 3109 and 3133.The encryption and decryption time for 16 MB is 48957 and 48925. After applying BWT encoding with RSA algorithm, the encryption and the decryption time for 1MB and 16 MB are 2699,42386 and 2659,42422. After applying the compression using RLE-BWT with RSA, the encryption and decryption time varies from 39155 to 39127 for 16MB. For RLE with enhanced BWT-RSA, the encryption time and the decryption time lies between 34652 and 34647 of 16MB file size of plaintext as shown in table 11. The experimental results clearly show that RLE-EBWT-RSA outperforms than the other counterpart.

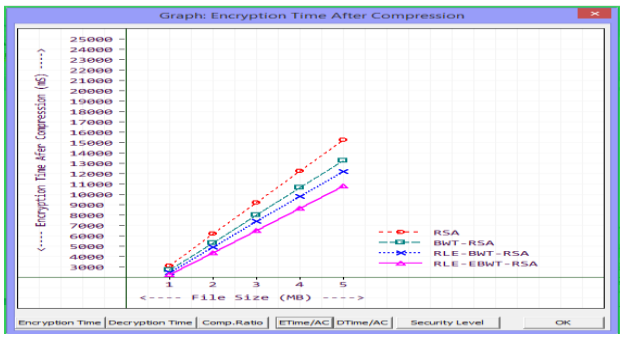

Figure 3. Encryption Time After Modified BWT with mtf

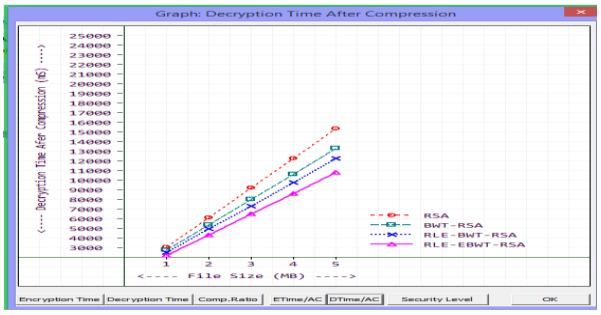

Figure 4. Decryption Time After Modified BWT with mtf

Table 10. Encryption Time And Decryption Time Before Using Modified Bwt

Method	Encryption Time(ms) File size					Decryption Time(ms) File size				
	1MB	2MB	4MB	8MB	16MB	1MB	2MB	4MB	8MB	16MB
RSA	3109	6156	12275	24527	48952	3123	6180	12243	24447	48915
BWT-RSA	2699	5324	10668	21241	42381	2649	5376	10601	21237	42412
RLE-BWT-RSA	2521	4979	9825	19574	39141	2454	4920	9874	19597	39122
RLE-EBWT-RSA	2173	4328	8714	17291	34638	2220	4409	8652	17327	34642

Table 11. Encryption Time And Decryption Time After Using Modified Bwt

Method	Encryption Time(ms) File Size					Decryption Time(ms) File Size				
	1MB	2MB	4MB	8MB	16MB	1MB	2MB	4MB	8MB	16MB
RSA	3109	6168	12277	24528	48957	3133	6190	12247	24451	48925
BWT-RSA	2699	5336	10670	21242	42386	2659	5386	10605	21241	42422
RLE-BWT-RSA	2531	4981	9829	19586	39155	2466	4924	9879	19600	39127
RLE-EBWT-RSA	2183	4330	8718	17303	34652	2232	4413	8657	17330	34647

The compression ratios are shown in Table 12 and the corresponding graphical representation is shown in Figure 5.

Table 12. Compression Ratio

Method	Compression Ratio(%) File size				
	1MB	2MB	4MB	8MB	16MB
RSA	100	100	100	100	100
BWT-RSA	100	100	100	100	100
RLE-BWT-RSA	84	88	88	88	86
RLE-EBWT-RSA	83	87	87	87	85

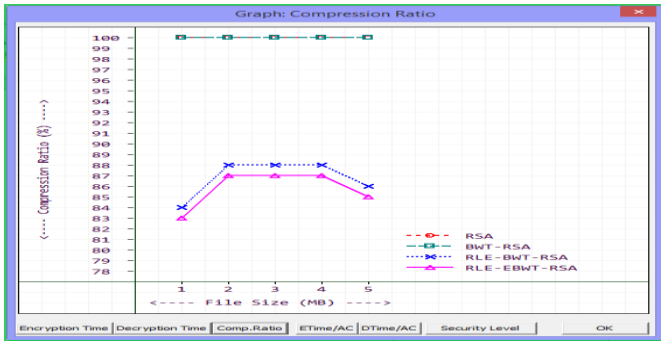

Figure 5. Compression Ratio

Table 13 shows the security level for various existing and proposed algorithms. Because of run length encoding is applied after encoding using BWT and the first sequence is considered for mtf in BWT rather than last sequence ,the security level of RLE-EBWT-RSA is increased upto 93% for 16MB compared to RLE-BWT-RSA with 90%. The various levels of security is shown in figure 6.

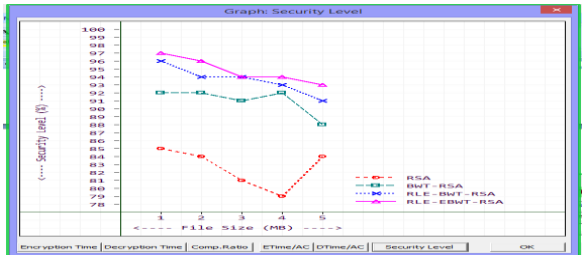

Figure 6. Security Level

Table 13. Security Level

Method	Security Level(%) File Size				
	1MB	2MB	4MB	8MB	16MB
RSA	84	85	85	82	78
BWT-RSA	95	93	90	89	90
RLE-BWT-RSA	95	94	94	91	90
RLE-EBWT-RSA	96	96	95	94	93

VI. CONCLUSION

A novel mtf based BWT methodology has been proposed to generate the binary code for the plaintext M and it will produce shorter code for M when it is compared with existing mtf based BWT. The shorter code has further been compressed using RLE. As compression is performed two times for M, the size of M is drastically reduced which results in increasing the speed of transmitting the plaintext. Also the original form of M has been completely changed using the proposed methodology, the eavesdropper may not predict the correct M from C which increases the security of any cryptographic

algorithms. Further, the block size is determined dynamically only on the basis of modulus m, which enhances the security as opposed to block size is fixed like 2048 or 4098 bits normally used in well known standard RSA public-key cryptosystems. The experimental results strengthen the increasing the speed and enhancing the security level using the proposed methodology.

VII. REFERENCES

[1]. S.Porwal, Y.Chaudhary, J.Joshi, M. Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science and Innovative Technology (IJESIT) ,Vol.2, Issue 2, March 2013.

[2]. U.Khurana and A.Koul, "Text Compression and Superfast Searching", Thapar Institute Of Engineering and Technology, Patiala, Punjab, India-147004.

[3]. S.Kaur and V.S.Verma, "Design and Implementation of LZW Data Compression Algorithm", International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, July 2012.

[4]. Md.RubaiyatHasan, "Data Compression using Huffman based LZW Encoding Technique", International Journal of Scientific & Engineering Research Vol.2, Issue 11, November-2011.

[5]. Rajan.S.Jamgekar, Geeta Shantanu Joshi, "File Encryption and Decryption Using Secure RSA", International Journal of Emerging Science and Engineering (IJESE), Vol-1, Issue-4, February 2013.

[6]. Monisha Sharma, Chandrashekhar Kamargaonkar, Amit Gupta, "A Novel Approach of Image Encryption and Decryption by using partition and Scanning Pattern", International Journal of Engineering Research & Technology (IJERT), Vol. 1, Issue 7, September- 2012.

[7]. https://en.wikipedia.org/wiki/Data_compression

[8]. D. Adjeroh, T. Bell, and A. Mukherjee, "The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching", Springer, 1 edition, July 2008.

[9]. K.Mani and A.Devi, "Enhancing Security in Cryptographic Algorithms Based on IENCCRS Scheme", IJAER,Vol.10, No.82, 2015.

[10]. J.Seward,"On the performance of bwt sorting algorithms", In Data Compression Conference, pages 173–182, IEEE Computer Society, 2000.

[11]. https://en.wikipedia.org/wiki/Elgamal.

[12]. William Stallings ,"Cryptography and Network Security, Principles and Practices", Fourth Edition , November, 2005.

[13]. Hans Delfs and Helmut Knebl, "Introduction to Cryptography Principles and Applications", Springer-Verlag, Berlin, Heidelberg, 2001.