International Journal of Computer Networks and Applications (IJCNA)
Volume 2, Issue 3, May – June (2015)

RESEARCH ARTICLE

Design and Develop an Approach for Integrating Compression and Encryption on Textual Data

Amit Jain
Sir Padampat Singhania University, Udaipur, India

Avinash Panwar
Sir Padampat Singhania University, Udaipur, India

Divya Bhatnagar
Sir Padmpat Singhania University, Udaipur, India

Abstract – The primary intention of this research is to design and develop an approach for integrating compression and encryption on textual data. This improved approach can provide the better security with the aid of encryption and also, can provide better compression with the help of encoding procedure. The advantage of combining the encoding procedure with the encryption technique is that the attacks relevant to storage can be easily avoided since the data is fully within encoded domain also; the storage space required will be less due to good compression ratio. With the advantages of this, we have planned to utilize the dictionary-based encoding procedure and proposed encryption method to comcryption protocol (Compression + encryption). For integrating the compression and encryption, dictionary-based compression and the encryption based on permutation, sorting and reordering will be used. Dictionary-based methods are capable of giving better compression rate as like huffman encoding also follows dictionary-based method. Also, permutation, sorting and reordering-based methods can provide better security if these operations are included in the encryption protocol.

Index Terms – Encoding, Encryption, CHE, CSE, Decoding, Decryption.

1. INTRODUCTION

The prime function of compression is to acquire methods in such a way that the actual data consume less space. Therefore, compressed data is efficient, since lower amount of memory space is required, also data travel faster from source to destination from a disk or through network link due to the lessening in the practice in case of slow devices. Since, now a days the processor speed is enhanced much faster than transferring through a disk or through internet transmission speeds, so the time require to encode and decode the data (although is quiet low) is not an important factor [1-5]. Transferring compressed data from source to destination is consisting of two processes: a) Source process, b) Destination process. The responsibility of the source process is to compress the required data and then transmit it by communication network. Then, the responsibility of the destination process is to accept the compressed data and decompressed it. The total responsibility is to make an encoded file to the disk and then to transmit it and at the receiving end decode it. In some case, encoding and transferring the data happened faster than receiving and decoding begin. But in some condition, especially in real-time transmission applications, where the sender and receiver processes at the same time, i.e. the data (encoded) transferring should be started without preprocessing the total file at the sender end and at the receiving end the data must be decoded the file as soon as the file arrives [6-10]. The above real-time data transferring is used when the communication is over a network. This type of procedure can be used when there is talk/chat protocols or remote login used, i.e. where short messages are necessary during the transmission time.

The theory of adding up data compression with encryption [11-15] of data is quiet useful, while transmission time needed and optimization of memory space required, due to the compression the redundancy in the plaintext decreased making the data resistant to statistical methods of cryptanalysis. Applying cryptographic characteristics in a compression algorithm is not a safe option, since the amount of compression achieved and amount of security cannot be achieved at the same time [16-21]. Furthermore, due to rise in computer-related technologies, which helps the cryptanalyst to make planned attacks with high success rate, also brute force attacks can be done in a quick period of time. Thus, according to cryptographic point of view, the security of data should be higher and should be based on an algorithm where brute force attack have minimum chance of success.

In this paper we propose an approach for integrating compression and encryption on textual data. Initially the text data is preprocessed to consider the repeated letters as one and assign the corresponding frequency to it. Thereafter the data is compressed and encoded using our condition based Huffman encoding (CHE) technique. A dictionary is then formed which is used to retrieve the original text data. The dictionary is then encrypted using our complex shuffle encryption (CSE)

RESEARCH ARTICLE

technique which uses a 128 bit key for encryption. The encoded data, the encrypted dictionary and the key is send to the receiver. The receiver first decrypt the dictionary based on our complex shuffle decryption (CSD) technique that uses the received key. The decrypted dictionary is then used to decode the data and the receiver can read the original text. The major contributions of our work are as follows:

Designing an effective compression technique: The compression technique is designed using CHE which converts the generated code based on conditions and the converted code is used to compress the original data effectively based on different condition.

Designing an effective encryption technique: The encryption technique is designed using CSE which uses 128 bit key. The key is split into three different numbers of bits and the three separated bits would do different functions each to encrypt the data effectively and to enhance the security.

Integrating the designed compression and encryption techniques on textual data: Both the compression and the encryption techniques were integrated to reduce the storage and to secure the data effectively.

This paper is organized as follows: the second section shows the motivation of our work and the third section explains our proposed technique and the fourth section shows the results we obtained for our work and the fifth section concludes our technique.

2. MOTIVATION OF OUR WORK

The encoding of data would reduce the required storage and the encryption of the data would provide secured transmission. If we integrate both the techniques, the attacks relevant to storage can be easily avoided because the data is fully encoded and the storage space required will be less due to good compression ratio. The ultimate objective of improving security and compression can be done through the following ways. At first, an intelligent dictionary will be created from the input text data based on the unique strings and frequency. Then, encoding of the input text data will be done using intelligent dictionary. Then, the dictionary will be encrypted in an effective way so that no one can obtain the original data back without decrypting the dictionary. The encryption of dictionary and then encoding will be done based on private key (of 128 bit size) which is essential at the receiver to retrieve original text data back. This way will be more secure and compressed against all the attacks except the stolen attack of key. This stolen key attack can be avoided with permutation-based encrypting the encoded text data. So, over all, the original text data will be encoded using the dictionary-based technique as per the method devised to bring more compression ratio. Then, the dictionary will be encrypted and then encoded using the method devised including the several operations like, permutation, sorting and reordering. This will give the private key that will be again used to encrypt the encoded text data. So, in the receiver side, it has the private key and encrypted dictionary and encoded text data. Then, the private key will be utilized i) to obtain the dictionary back to original form and, ii) the encrypted data back to encoded data. Finally, the original text will be obtained from decrypted dictionary and decrypted encoded data using the decoding procedure.

3. PROPOSED METHOD

This section explains the proposed technique of integrating compression and encryption on textual data. The Figure 1 and Figure 2 shows the process in transmitter side and receiver side of our proposed technique.

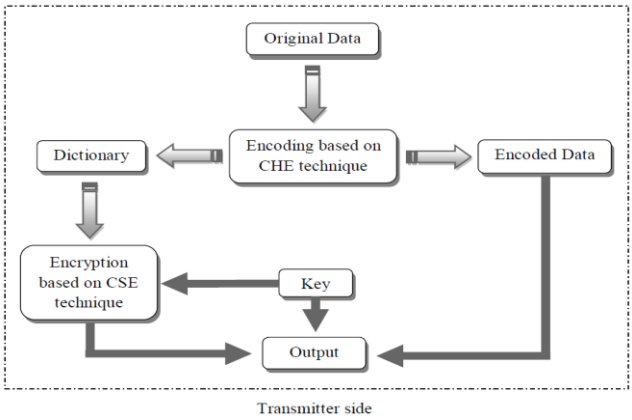


Figure 1 Process in transmitter side

In this Figure 1, the original data i.e. the given input data is initially encoded using condition based Huffman encoding (CHE) technique and a dictionary is created based on it. The dictionary contains the essential information to retrieve the encoded data. The dictionary is encrypted using a complex shuffle encryption (CSE) technique to hide the information. The CSE technique uses a key to encrypt the data. The encoded data, the key and the encrypted data are the output from the transmitter side.

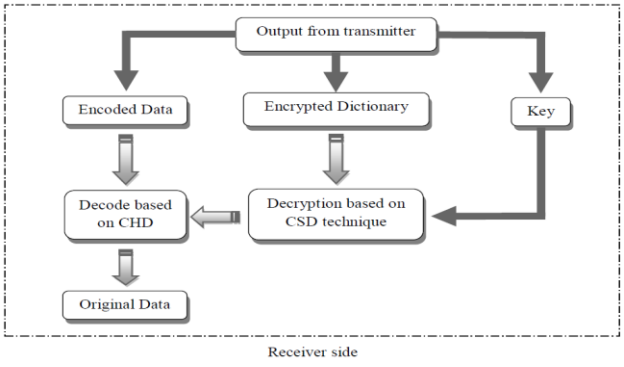


Figure 2 Process in receiver side

RESEARCH ARTICLE

The Figure 2 explains as follows: the output from the transmitter i.e. the encoded data, the encrypted dictionary and the key are received by the receiver side. The encrypted dictionary is then decrypted using complex shuffle decryption (CSD) technique based on the key received and using the decrypted dictionary, the encoded data is decoded based on condition based Huffman decoding (CHD) technique. The decoded data is the original data.

3.1. CHE Technique

This section shows the condition based Huffman encoding (CHE) technique. The original data is encoded using condition based Huffman encoding technique. The condition based Huffman encoding technique has three processes which are Huffman code generation for the original data, code conversion of condition based sequence and encoding. It is explained as follows:

Huffman code generation

The Huffman code generation is as follows: initially the repeated letters from the word is taken and assigned the frequency value to it. Thereafter, the generation of code is done by combining the letters of least two frequencies and giving zero's and one's values to it. Assigning zero's and one's values based on two least frequency letters is done step by step until last combination. After generating zeroes and ones for every combination, the Huffman code for each letter is formed from the bottom end of the tree formed.

Code conversion of condition based sequence

The code conversion of condition based sequence is done after generating Huffman code for each letter in the words. The process of code conversion of condition based sequence is as follows: initially the letters used to generate the Huffman code is arranged in ascending order based on the length of the code. Thereafter, the code conversion process is done by taking the letter that has highest length of Huffman code i.e. the last letter after the arrangement as constant and check with the least length letters and verify whether the least length letter is the preceding letter of the constant letter in the original data. If yes, assign the Huffman code of the least length letter to the constant letter. If no, check with next least length letter and if no letters in the series forms the combination, there would not be any change. Similarly do this process on the prior highest length Huffman coded letter.

Encoding

The encoding process is done based on the combination of letters used in the code conversion of condition based sequence and the preceding letter in the original data. The encoding process is as follows: initially the combination of the letters used for the code conversion process and the preceding letter of the combination of the letters used for the code conversion process is checked to decide whether the code formed using code conversion process is to be considered or not. If the Huffman code of the preceding letter of the combination of the letters used for code conversion is ends with zero, then the code formed in the code conversion process is considered and if it is not ends with zero, we won't consider the code formed based on the code conversion process. After this verification, a code is formed for the original data. The final code is the encoded data based on condition based Huffman encoding technique.

3.2. Example of CHE Technique

The whole process of condition based Huffman encoding is explained by an example as follows: consider 'optoipoa' is an original data and from this original data, the Huffman code is formed for each letter in the original data. The formation of code is shown in Figure 3

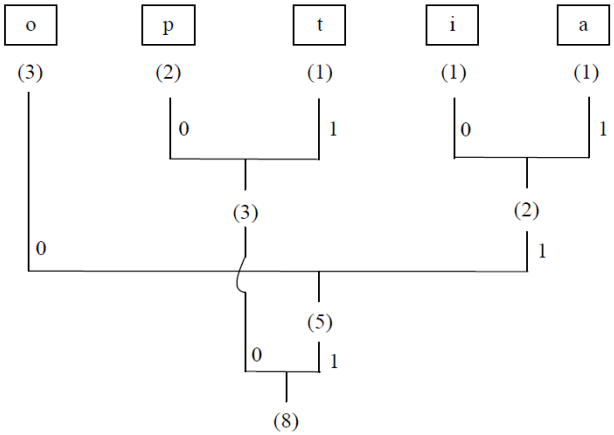
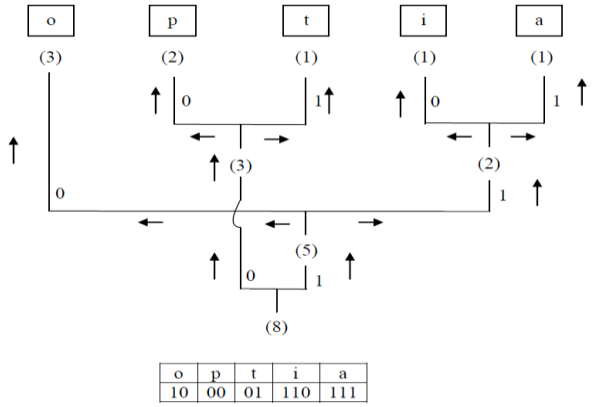


Figure 3 Formation of Huffman code

The Figure 3 is explained as follows: initially the repeated letters are considered for one time and mark the frequency of the letter in the original data i.e. the numeric term 3 below the letter 'o' represents that the letter 'o' repeated three times in the original data. Thereafter, two letters with least frequency is taken and assigns zero and one to it and represents the total frequency of two letters below it. Similarly, this process is done by taking the next two letters and assign zeros and ones until last step. The Huffman code is then formed for each letter by considering the corresponding branches of zeros and ones from the last step to the first. From the Figure 3, the Huffman code formed for the letter 'o' is 10; and the Huffman code formed for the letter 'p' is 00; and the Huffman code formed for the letter 't' is 01; and the Huffman code formed for the letter 'i' is 110; and the Huffman code formed for the letter 'a' is 111. Eventually, the Huffman code for the original data 'optoipoa' is '1000011011000010111'. The Figure 4 shows the direction of Huffman code formed for each letter.

RESEARCH ARTICLE



o	p	t	i	a
10	00	01	110	111

Figure 4 Direction of Huffman code formation

After generating the Huffman code for each letter, the code conversion of condition based sequence is done by arranging the least length letters first. The code conversion of condition based sequence process is used to compress the data. Here, in Figure 4 we obtained the least length letters first and we don't need to change it. Thereafter, the highest length letters are compared with the least length letters by checking whether the least length letter is the preceding letter of the highest length letter in the original data. Therefore, the highest length letter 'a' would compared with the least length letter 'o' to check whether 'o' is the previous letter of 'a' in the original data 'optoipoa'. From the original data we can see that the letter 'o' which has two length code comes before the letter 'a' which has three length code and therefore the Huffman code of 'o' is assigned to the letter 'a'. The Figure 5 shows the letters with its code after the code conversion process of the letters 'o' and 'a'.

o	p	t	i	a
10	00	01	110	10

Figure 5 After code conversion process

Similarly the code conversion process is done between the prior highest length letter and the least length letters. When comparing all the other letters in the original code, the letters 'i' and 'p' comes sequent in the original data but the three length coded letter 'i' comes before the two length coded letter 'p' and therefore it has not satisfied the condition. So the code conversion will not do between the letters 'i' and 'p' of the original code.

After the code conversion process, the encoding is done based on the Huffman code of the preceding letter of the letters which are used for code conversion. The encoding is the process that fixes the code converted in the code conversion process. The code conversion is done between the sequence of letter 'o' and 'a' in the original code 'optoipoa'. Therefore, the Huffman code of the previous letter 'p' is considered to fix the converted code between the letters 'o' and 'a'. To fix the converted code, the value of last length of the letter 'p' should be zero i.e. the Huffman code of 'p' should ends with zero. Here, the Huffman code of 'p' is 00. Therefore, the code converted between 'o' and 'a' after the code conversion process is considered for encoding the data. If the Huffman code of 'p' ends with 1, then the code converted between 'o' and 'a' after the code conversion process would not be considered and the Huffman code obtained which is before the code conversion process is considered as encoded data. Finally, the CHE based encoded data for the original data 'optoipoa' is '10000110110001010'.

3.3. Dictionary formation

A dictionary is generated to retrieve the encoded data and it is send to the receiver after encrypting it. The dictionary contains the letters considered to generate the Huffman code with its frequency in the original data. The original data we taken for our example is 'optoipoa'. The Figure 6 shows the dictionary generated for the original data taken for our example.

Letters	Frequency
o	3
p	2
t	1
i	1
a	1

Figure 6 Dictionary generated for the original data

The dictionary word generated from the data given is 'optia' that would have its frequency after each letters which is as 'o3p2t1i1a1'.

3.4. Complex Shuffle Encryption

The dictionary is then encrypted using complex shuffle encryption (CSE) technique and it is transmitted to the receiver side. The complex shuffle encryption is done based on a 128 bit key. The key is split into three different divisions to do different jobs. The Figure 7 shows the structure of 128 bit key.

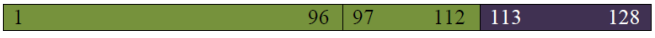


Figure 7 Structure of 128 bit key

In this 128 bit key, the first bit to ninety sixth bits are used for shuffling; and from ninety seventh bit to one twelfth bit is used to convert the letter based on ASCII code; and the one thirteenth bit to one twenty eight bits contain the information about position of the dictionary data after retrieval process. The encryption of the dictionary word 'o3p2t1i1a1' with its frequency is as follows:

RESEARCH ARTICLE

Initially, the dictionary word is rotated clockwise based on the length of the word i.e. if the dictionary word is four length, then it would be rotated clockwise four times. Thereafter, the rotated words are sorted based on the alphabet and the last letter of the sorted words is chosen to form a different word. This process is called reorder process and it is shown in the Figure 8

Rotate clockwise	Sort	Reordered dictionary
o3p2t1ila1	1a1o3p2t1i	itapo11132
1o3p2t1ila	1ila1o3p2t	
a1o3p2t1il	1o3p2t1ila	
1a1o3p2t1i	2t1ila1o3p	
ila1o3p2t1	3p2t1ila1o	
1ila1o3p2t	a1o3p2t1il	
t1ila1o3p2	ila1o3p2t1	
2t1ila1o3p	o3p2t1ila1	
p2t1ila1o3	p2t1ila1o3	
3p2t1ila1o	t1ila1o3p2	

Figure 8 Reorder Process of dictionary word

After a different word is formed from the dictionary word, we use the 128 bit key to complete the encryption process. The three different jobs of the key are interchange process, rotation process and location identification.

Interchange Process (1 to 96 bits):

The zeros and ones are generated randomly in the first ninety six bits of the 128 bits key. The 96 bits are then split into twelve cells and each cell has 8 bits and each cell is converted into integers. Therefore we would get twelve integer values. The integer values of two after two cells of the twelve cells are then used to exchange the letters of the corresponding positions. For instance if the first cell has the integer value as 1 and the second cell has the integer value as 4, then the corresponding letters in the first and fourth position of the reordered dictionary word is interchanged i.e. the 'itapo11132' is formed as 'ptaio11132'. Similarly, the letters are interchanged based on next two integers until the integers in eleventh and twelfth cells.

Rotation Process (97 to 112 bits):

The process in 97 to 112 bits is as follows: alike 1 to 96 bits, the zeros and ones are generated randomly in 97 to 112 bits and it is split into two cells and each cell has eight bits of binary data. The binary data in both cells are converted in to integers and using the integer values the letters in corresponding positions are changed based on ASCII code. For instance the word obtained after 1 to 96 bits process is 'ptaio11132' and assume that the integer value of the first cell of 97 to 112 bits is 3, then the ASCII code of the letter in the third position is taken and added with the integer value and the solution is then converted into character and placed it on the same position. Therefore, the letter in the third position would get changed. In 'ptaio11132', the third letter is 'a' and the decimal value of ASCII is 97 which is added with 3 gives the solution as 100. The character'd' has the decimal value of the ASCII as 100 and therefore, we replace the 'a' as'd'. The word is then changed as 'ptdio11132'. Thereafter, the integer in the second cell is considered and the integer value we assume as 7. The ASCII code of the letter in the seventh position is subtracted with it and the solution is converted into character and placed in the same position. It is explained as follows: the seventh letter of 'ptdio11132' is '1' and the decimal value of ACII of '1' is 49 which is subtracted with 7 gives the solution as 42. The 42 is the decimal value of the ASCII character '*' and it is replaced in seventh position. Eventually, the encrypted dictionary is 'ptdio1*132'.

Location Identification (113 to 128 bits):

The 113 to 128 bits contains the information about position of the dictionary word after retrieval process. Therefore, the retrieval process is also done simultaneously with the encryption process to identify the position of the dictionary word. The Figure 9 shows the retrieval process of the dictionary word 'o3p2t1i1a1'.

C1	i	t	a	p	o	1	1	1	3	2
S1	1	1	1	2	3	a	i	o	p	t
C2	il	tl	a1	p2	o3	1a	1i	1o	3p	2t
S2	1a	1i	1o	2t	3p	a1	il	o3	p2	tl
C3	ila	tli	a1o	p2t	o3p	1a1	1il	1o3	3p2	2tl
S3	1a1	1il	1o3	2tl	3p2	a1o	ila	o3p	p2t	tli
C4	ila1	tlil	a1o3	p2tl	o3p2	1a1o	1ila	1o3p	3p2t	2tli
S4	1a1o	1ila	1o3p	2tli	3p2t	a1o3	ila1	o3p2	p2tl	tlil
C5	ila1o	tlila	a1o3p	p2tli	o3p2t	1a1o3	1ila1	1o3p2	3p2tl	2tlil
S5	1a1o3	1ila1	1o3p2	2tlil	3p2tl	a1o3p	ila1o	o3p2t	p2tli	tlila
C6	ila1o3	tlila1	a1o3p2	p2tlil	o3p2tl	1a1o3p	1ila1o	1o3p2t	3p2tli	2tlila
S6	1a1o3p	1ila1o	1o3p2t	2tlila	3p2tli	a1o3p2	ila1o3	o3p2tl	p2tlil	tlila1
C7	ila1o3p	tlila1o	a1o3p2t	p2tlila	o3p2tli	1a1o3p2	1ila1o3	1o3p2tl	3p2tli1	2tlila1
S7	1a1o3p2	1ila1o3	1o3p2tl	2tlila1	3p2tli1	a1o3p2t	ila1o3p	o3p2tli	p2tlila	tlila1o
C8	ila1o3p2	tlila1o3	a1o3p2tl	p2tlila1	o3p2tlil	1a1o3p2t	1ila1o3p	1o3p2tli	3p2tli1a	2tlila1o
S8	1a1o3p2t	1ila1o3p	1o3p2tli	2tlila1o	3p2tlila	a1o3p2tl	ila1o3p2	o3p2tli1	p2tlila1	tlila1o3
C9	ila1o3p2t	tlila1o3p	a1o3p2tli	p2tlila1o	o3p2tlila	1a1o3p2tl	1ila1o3p2	1o3p2tl1il	3p2tli1a1	2tlila1o3
S9	1a1o3p2tl	1ila1o3p2	1o3p2tlil	2tlila1o3	3p2tli1a1	a1o3p2tli	ila1o3p2t	o3p2tli1a	p2tlila1o	tlila1o3p
C10	ila1o3p2tl	tlila1o3p2	a1o3p2tlil	p2tlila1o3	o3p2tlila1	1a1o3p2tli	1ila1o3p2t	1o3p2tl1ila	3p2tli1a1o	2tlila1o3p
S10	1a1o3p2t1i	1ila1o3p2t	1o3p2tlila	2tlila1o3p	3p2tli1a1o	a1o3p2tlil	ila1o3p2t1	*o3p2tli1a1*	p2tlila1o3	tlila1o3p2

Figure 9 Retrieval process of the dictionary word

The Figure 9 explains as follows: the C1 contains the reordered dictionary word i.e. 'itapo11132' which is arranged in horizontal order. It is then sorted S1; and the letters in C1 and S1 are merged in C2 and again the letters in the C2 are sorted in S2. The C3 is obtained by merging the C1 and S2. Similarly, the process is repeated until the length of the dictionary word. In our example, the dictionary word is ten lengths and so, the process is done until S10. In S10, the dictionary word 'o3p2t1i1a1' is obtained in the eighth position which is italicized. The encryption would be stopped at this point and the encrypted data with the key and encoded data is send to the receiver.

RESEARCH ARTICLE

3.5. Complex Shuffle Decryption

The received encrypted data is then decrypted using complex shuffle decryption (CSD) technique. It is done by doing the reverse process of encryption based on the key received. The decryption process is explained as follows: initially the two integers in the bits from 97 to 112 is taken and do the reverse process of the process done while encryption i.e. while encryption the first integer is added with the ASCII code, but in decryption we have to subtract it; and the second integer is subtracted with the ACSII code while encryption, but here it is added with the ASCII code. Thereafter, the twelve cells from the bits 1 to 96 are taken to do the reverse process i.e. instead of taking integers from left to right cells (1st to 12th cell) in encryption process, the integers from right to left (12th to 1st cell) is taken to do the decryption process. Therefore the different word formed from the dictionary word is obtained and the retrieval process done at the encryption side is processed at the decryption side. The position information in the key (113-128 bits) is used to retrieve the original dictionary which is the decrypted data. The decrypted dictionary is then used to decode the original data based on condition based Huffman decoding (CHD) technique. The CHD would do the reverse process of CHE. The Figure 10 shows the algorithm of the whole process of our proposed technique; and the Figure 11 shows the algorithm of CSE technique; and the Figure 12 shows the algorithm of CSD technique.

Algorithm of our proposed technique:

Input: Original Data
Output: Original Data

Transmitter Side:
1. *Start*
2. Get the original data
3. *For* each letter in the original data
4. Check for repeated letters
5. *If* repeated letter exists
6. Convert the similar repeated letters as one and assign frequency value
7. *Else*
8. Keep the letter as it is and assign the frequency as one
9. *End if*
10. *End for*
11. Take the letters with its frequency
12. *For* each two least frequency letters
13. Assign 0's and 1's until last combination
14. *End for*
15. *For* each letter
16. Generate Huffman code based on 0's and 1's from last branch to first of corresponding letter
17. *End for*
18. Arrange the letters in ascending order based on the length of the code
19. *For* each highest length letter
20. Check with each least length letter
21. *If* any of the least length letter comes previous to the position of the highest length letter of original data
22. Assign the code of least length letter to the highest length letter to compress the data
23. *Else*
24. Ignore
25. *End if*
26. *If* the code of previous letter of the least and highest length letters used for code conversion ends with zero
27. Fix the converted code for encoding
28. *Else*
29. Don't consider the converted code for encoding
30. *End if*
31. *End for*
32. Compressed encoded data would be formed
33. Generate a dictionary which is used to decode the encoded data
34. Generate 128 bit key
35. Encrypt the dictionary based on CSE technique
36. Send the compressed encoded data, encrypted dictionary and key to the receiver

Receiver Side:
37. Decrypt the encrypted dictionary based on CSD technique
38. Decode the encoded original data using the decrypted dictionary
39. Original data would be obtained
40. *Stop*

Figure 10 Algorithm of our proposed technique

Algorithm of CSE technique:

Input: Dictionary word
Output: Encrypted dictionary

1. *Start*
2. Get the dictionary word
3. Generate 128 bit key
4. Take the first 96 bits from the 128 bit key
5. *For* each bit
6. Generate 0's and 1's randomly
7. *End for*
8. Split 8 bits as one cell and therefore we would get 12 cells
9. *For* each cell
10. Convert binary to decimal
11. *End for*
12. Take two after two cells from left to right
13. *For* each two cells
14. Interchange the letters in the corresponding positions of the decimal value
15. *End for*
16. Take 97 to 112 bits
17. *For* each bit
18. Generate 0's and 1's
19. *End for*
20. Split 8 bits as one cell and therefore 2 cells would be obtained
21. Convert binary to decimal
22. Take the ASCII code of the letter of the corresponding position to the first decimal value
23. Add the decimal value with ASCII code and convert it to character and place the new character on the same position
24. Take the ASCII code of the letter of the corresponding position to the second decimal value
25. Subtract the decimal value with ASCII code and convert it to character and place the new character on the same position
26. The 113 to 128 bits contain the information about the position of the original dictionary based on retrieval process
27. Encrypted dictionary would be obtained based on the 128 bit key
28. *Stop*

RESEARCH ARTICLE

Figure 11 Algorithm of CSE technique

Algorithm of CSD technique:

Input: Encrypted dictionary
Output: Dictionary word

1. *Start*
2. Get the encrypted dictionary
3. Take the two decimal values of 97 to 112 bits
4. Take the ASCII code of the letter of the corresponding position to the first decimal
5. Subtract the decimal value with the ASCII code and convert it to character
6. Take the ASCII code of the letter of the corresponding position to the second decimal
7. Add the decimal value with the ASCII code and convert it to character
8. Take the twelve decimal values of the first 96 bits
9. Take two after two cells from right to left
10. *For* each two cells
11. Interchange the letters in the corresponding positions of the decimal value
12. *End for*
13. Do retrieval process
14. Use 113 to 128 bits of the key to identify the correct position
15. Original dictionary word would be obtained
16. *stop*

Figure 12 Algorithm of CSD technique

4. RESULTS AND DISCUSSIONS

This section shows the experimental outcomes of our proposed technique. The proposed technique is experimented based on execution time for encoding and decoding, compression ratio, execution time for encryption and decryption, and the number of words identified by hacker. We experimented the encoding and the encryption processes separately, because we don't have any existing techniques that uses encoding and encryption techniques together. The encoding process is compared with an existing encoding technique and the encryption process is compared with an existing encryption technique. Finally, we compared our proposed technique with the existing encoding and encryption techniques together in terms of execution time and memory consumption. We used jdk 1.7 with system configuration as follows: i3 processor that clocks at 3.06GHz with 4GB of RAM.

4.1. Experimental Setup

To experiment the proposed technique, we took ten different sentences as input from two different documents. The ten different sentences are then encoded using our proposed condition based Huffman encoding (CHE) technique and using existing normal Huffman encoding technique. After encoding the documents based on our proposed encoding technique and the existing Huffman encoding technique, the compression ratio is calculated for both the techniques and compared. This is the first phase of comparison. In the second phase of comparison, we compared the number of words a hacker identified from the encrypted data based on randomly generated key. It is done as follows: the ten different sentences we given as input for encode process is taken as input for the encryption process in our experimentation. The ten different sentences are encrypted using our proposed technique and using the existing AES technique. The proposed encryption technique and the existing AES technique is compared using a randomly generated key by considering a hacker knows the length of the key and trying to decrypt the encrypted data to see the original data. In the third phase of comparison, the proposed technique is compared with the existing encoding technique and the existing encryption technique together.

4.2. Performance comparison

This section shows the performance comparison of our proposed technique. The performance is compared for three phases which are encoding, encryption and combining both. For encoding, we compared our proposed encoding technique with the existing Huffman encoding technique in terms of execution time and compression ratio; and for encryption, we compared our proposed encryption technique with the existing AES technique in terms of execution time and total word retrieved by randomly generated key. In the third phase, we compared our proposed technique with the existing technique (i.e. by combining the existing encoding and existing encryption techniques) in terms of execution time and memory consumption.

4.2.1 Evaluation metric

This section explains the evaluation of compression ratio used for our comparison. The compression ratio is the ratio of total memory taken by the input original data to the total memory taken after encoding it. It is shown by a formula given below:

$$CR = \frac{M\left(D^{input}\right)}{M\left(D^{encoded}\right)}$$

In the above equation, CR denotes the compression ratio; and $M\left(D^{input}\right)$ denotes the memory taken by the input data; and $M\left(D^{encoded}\right)$ denotes the memory taken after encoding the original input data. If $M\left(D^{encoded}\right)$ is less, then the value of CR would be high. This denotes that the data is compressed well.

4.2.2 Comparison based on encoding

RESEARCH ARTICLE

This section shows the comparison between the proposed (CHE) encoding technique and the existing Huffman encoding technique. The comparisons are done based on execution time and compression ratio.

Documents	Encoding time (in ms)		Decoding time (in ms)	
	Normal Huffman Encoding	Proposed Encoding Technique	Normal Huffman Decoding	Proposed Decoding Technique
1	21	30	44	98
2	17	32	31	43
3	30	35	114	159
4	25	34	31	119
5	25	41	52	101
6	17	41	17	63
7	44	48	64	67
8	19	20	46	90
9	25	40	125	148
10	30	61	59	157

Table.1 Execution time for encoding and decoding

The Table.1 shows the execution time taken for encoding and decoding using our proposed CHE technique and using the existing Huffman encoding technique. Here, the ten documents represent the ten different sentences we taken for our experimentation. The time calculated for our experimentation is in ms and this Table.1 shows that our proposed technique took more time to execute for both encoding and decoding. This is because our proposed technique does more process than the normal Huffman encoding technique to compress the data. The Table.2 shows the compression ratio obtained between the proposed encoding technique and the existing Huffman encoding technique.

Documents	Compression ratio	
	Normal Huffman Encoding	Proposed Encoding Technique
1	0.7323	0.7337
2	0.7431	0.7444
3	0.7251	0.7257
4	0.7328	0.734
5	0.7272	0.7288
6	0.7312	0.7325
7	0.734	0.7355
8	0.7345	0.7365
9	0.7309	0.7319
10	0.7387	0.7399

Table.2 Compression ratio comparison

The Table.2 contains the compression ratio values using our proposed encoding technique and the existing Huffman encoding technique for the ten different sentences we gave as input. It clearly shows that our proposed technique compressed the data better than the existing technique for all the ten different sentences we given as input.

4.2.3 Comparison based on encryption

This section shows the comparison between our proposed (CSE) encryption technique and the existing AES encryption technique. The comparisons are done based on execution time and the number of words retrieved by a hacker using randomly generated key.

Documents	Encryption time (in ms)		Decryption time (in ms)	
	AES Algorithm	Proposed Technique	AES Algorithm	Proposed Technique
1	1224	284	729	128
2	1201	255	578	99
3	2217	422	913	200
4	1176	243	705	121
5	1856	328	919	176
6	989	215	739	134
7	1962	364	501	82
8	1006	221	596	97
9	1257	288	701	124
10	1293	260	796	139

Table.3 Execution time for encryption and decryption

The Table.3 shows the execution time taken for encryption and decryption using our proposed technique and the existing AES technique. The execution time values shown in this table are measured in milliseconds and it is calculated for all the ten different sentences we given as input. While decrypting the

RESEARCH ARTICLE

encrypted data, the system would use different numbers of randomly generated keys and if it identified any words, the system would check with the dictionary and so the processing time would get increase. If different words are identified using different keys, it would also increase the processing time. For instance, consider the first and second documents are of same size and assume in the first document two different words are identified by two different keys and in the second document two different words are identified using same key. Here, the execution time for decrypting the first document would be more than the execution time of the second document. The Table.4 shows the number of words identified by the hacker based on randomly generated key.

Documents	max words discovered	
	AES Algorithm	Proposed Encryption Technique
1	27	14
2	21	0
3	28	12
4	26	0
5	27	0
6	14	0
7	13	0
8	16	0
9	26	0
10	24	0

Table.4 Number of words identified by hacker

The number of words identified by the hacker shown in this Table.4 is taken by considering the words after decryption present in the dictionary as original word. The process of using a randomly generated key is as follows: Initially ten keys are generated based on the length and based on the genetic algorithm the generated keys are given for crossover process and it is then given for mutation process. Therefore, we would get thirty keys and the thirty keys are used to decrypt the encrypted data to get the original data. The fitness is calculated for each key and the fitness is the number of decrypted words present in the general dictionary using corresponding keys. Thereafter, ten best keys are selected from the thirty keys based on the fitness and the process is repeated until the iteration number we set. A best key is chosen based on the best fitness to finally decrypt the encrypted data.

Here in this Table.4 when using the first sentence, the hacker identified 27 words on the encrypted data that used AES encryption technique and the hacker identified 14 words on the encrypted data that used our proposed technique. When the second sentence is given as input for encryption, the hacker identified 21 words using AES encryption technique and he couldn't find any words using our proposed technique. When the third sentence is given as input for encryption, the hacker identified 28 words using AES encryption technique and the hacker identified 12 words using our proposed technique. When all the other sentences are given as input, the hacker identified some words on the AES based encrypted data, but the hacker could not find any word on the encrypted data that used our proposed encryption technique. The Table.4 clearly shows that our proposed technique is better compared to the existing AES technique.

4.2.4 Comparison based on encoding and encryption together

This section shows the comparison between our proposed technique and the existing encoding and encryption techniques together based on execution time and memory taken for execution. In the previous two comparisons, we compared the proposed encoding with the existing encoding technique and the proposed encryption with the existing encryption technique to compare the effectiveness of our proposed technique. But here, the proposed technique is compared with the existing technique (i.e. by combining the existing encoding and the existing encryption techniques). The Table.5 shows the execution time comparison between our proposed technique and the existing technique (combining existing encoding and encryption techniques).

Documents	Execution Time (in ms)	
	Existing Technique	Proposed Technique
1	2018	540
2	1827	429
3	3274	816
4	1937	517
5	2852	646
6	1762	453
7	2571	561
8	1667	428
9	2108	600
10	2178	617

Table.5 Execution time comparison

RESEARCH ARTICLE

This Table.5 shows that our proposed technique consumed more time than the existing technique (existing encoding technique and existing encryption technique together), because our proposed technique do more process than the existing technique to effectively compress the data and for secured transmission. The Table.6 shows the memory taken for our proposed technique and the existing technique (combining existing encoding and encryption techniques).

Documents	Memory (in bits)	
	Existing Technique	Proposed Technique
1	5044840	4231112
2	4882600	3647000
3	5452472	2994912
4	4669000	1440728
5	4160352	3454616
6	5494464	1987456
7	3664272	2536912
8	4636816	1585256
9	4108432	1178904
10	3740720	2466184

Table.6 Memory consumption comparison

The Table.6 clearly shows that our proposed technique consumed less memory than the existing technique. This indicates that our technique compressed the data effectively.

5. CONCLUSION

In this paper we have proposed an approach to integrate compression and encryption for textual data to reduce the required space and to enhance the security. Here, initially the text data is preprocessed by considering the repeated letters as one and assigned corresponding frequency to it. Thereafter, our proposed condition based Huffman encoding (CHE) technique is applied to compress and encode the data and a dictionary that contains essential information to retrieve the original data is formed based on it. The dictionary is then encrypted based on our proposed complex shuffle encryption (CSE) technique using a 128 bit key. The encoded data, the encrypted dictionary and the key is then sent to the receiver. The receiver first decrypted the dictionary based on complex shuffle decryption (CSD) technique that uses the received key for decryption; and the original data is decoded based on condition based Huffman decoding (CHD) technique that uses the decrypted dictionary to decode the original data. Our proposed encoding technique is compared with the existing Huffman encoding technique based on compression ratio and it showed that our technique is better than the existing technique. We also compared our proposed encryption technique with the existing AES technique based on number of words retrieved by a hacker and it showed that our technique performed better. The performance is also compared between the proposed technique and the existing technique (by combining the existing Huffman encoding and existing AES techniques) based on execution time and memory consumption and it showed that our technique consumed less memory than the existing technique. In terms of compression ratio, in terms of hacking and in terms of memory usage, our technique performed better than the existing techniques and showed that our technique reduced the storage space and enhanced the security.

REFERENCES

[1] Tamar Shoham, David Malah and Slava Shechtman, “Quality Preserving Compression of a Concatenative Text-To-Speech Acoustic Database”, IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 20, NO. 3, PP. 1056-1068, 2012.

[2] M. Mitra, J.N. Bera and R. Gupta, "Electrocardiogram compression technique for global system of mobile-based offline telecardiology application for rural clinics in India", IET Science, Measurement and Technology, Vol. 6, No. 6, PP. 412–419, 2012.

[3] L. Robert and R. Nadarajan, "Simple lossless preprocessing algorithms for text compression", IET Software, Vol. 3, No. 1, PP. 37–45, 2009.

[4] Sebastian Wandelt and Ulf Leser, "FRESCO: Referential Compression of Highly Similar Sequences", IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, VOL. 10, NO. 5, PP. 1275-1288, 2013.

[5] Shuhui Wang and Tao Lin, "Compound image compression based on unified LZ and hybrid coding", IET Image Processing, Vol. 7, No. 5, PP. 484–499, 2013.

[6] Miguel A. Martínez-Prieto, Joaquín Adiego and Pablo de la Fuente, "Natural Language Compression on Edge-Guided text preprocessing", Information Sciences, Vol. 181, PP. 5387-5411, 2011.

[7] W. Oliveira Jr., E. Justino and L.S. Oliveira, "Comparing compression models for authorship attribution", Forensic Science International, Vol. 228, PP. 100-104, 2013.

[8] Ashutosh Gupta and Suneeta Agarwal, "A fast dynamic compression scheme for natural language texts", Computers and Mathematics with Applications, Vol. 60, PP. 3139-3151, 2010.

[9] Ana Granados, David Camacho and Francisco Borja Rodríguez, "Is the contextual information relevant in text clustering by compression?", Expert Systems with Applications, Vol. 39, PP. 8537-8546, 2012.

[10] M.Baritha Begum and Y.Venkataramani, "LSB Based Audio Steganography Based On Text Compression", Procedia Engineering, Vol. 30, PP. 703-710, 2012.

[11] Ranjan Bose and Saumitr Pathak, "A Novel Compression and Encryption Scheme Using Variable Model Arithmetic Coding and Coupled Chaotic System", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I, VOL. 53, NO. 4, PP. 848-857, 2006.

[12] Deep Vardhan Bhatt and Gerhard P. Hancke, "Secure Internet Access to Gateway Using Secure Socket Layer", IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 55, NO. 3, PP. 793-800, 2006.

[13] Maruti Satti and Subhash Kak, "Multilevel Indexed Quasigroup Encryption for Data and Speech", IEEE TRANSACTIONS ON BROADCASTING, Vol. 55, No. 2, PP. 270-281, 2009.

[14] Neri Merhav, "Perfectly Secure Encryption of Individual Sequences", IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 59, NO. 3, PP. 1302-1310, 2013.

RESEARCH ARTICLE

[15] Frédérique Oggier and Miodrag J. Mihaljevi´, "An Information-Theoretic Security Evaluation of a Class of Randomized Encryption Schemes", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 2, PP. 158-168, 2014.
[16] Francisco Salcedo-Campos, Jesús Díaz-Verdejo and Pedro García-Teodoro, "Segmental parameterisation and statistical modelling of e-mail headers for spam detection", Information Sciences, Vol. 195, PP. 45-61, 2012.
[17] Sherif Sakr, "XML compression techniques: A survey and comparison", Journal of Computer and System Sciences, Vol. 75, PP. 303-322, 2009.
[18] Nir Nissim, Robert Moskovitch, Lior Rokach and Yuval Elovici,
[19] "Novel active learning methods for enhanced PC malware detection in windows OS", Expert Systems with Applications, 2014.
[20] Simson L. Garfinkel, "Digital media triage with bulk data analysis and bulk_extractor", Computers & Security, Vol. 32, PP. 56-72, 2013.
[21] Baiying Lei, Ee-Leng Tan, Siping Chen, Dong Ni, Tianfu Wang and Haijun Lei, "Reversible watermarking scheme for medical image based on differential evolution", Expert Systems with Applications, Vol. 41, PP. 3178-3188, 2014.
[22] Osama Ahmed Khashan and Abdullah Mohd Zin, "An Efficient Adaptive of Transparent Spatial Digital Image Encryption", Procedia Technology, Vol. 11, PP. 288-297, 2013.