

Original Research Article

Plain Text Encoding/Decoding Technique Using a Combination of Huffman and Run-Length Algorithms.

ABSTRACT

This work presents an idea of combining encoding/decoding, and compression algorithms. The sole purpose is to study the effect of this combination on input textual data, and the result that could be achieved and how far one can go in encoding and compressing data. The proposed combination is that of Huffman and Run-Length algorithms. This study also focuses on the sequence of applying the two algorithms to see if it has an effect on the output data or not. Moreover, it is shown why these two algorithms were chosen and each of them contribute to the overall result.

Keywords: Encryption techniques, Huffman and Run-Length Algorithms, Compression techniques, encoding, decoding.

1 INTRODUCTION AND RELATED WORK

In recent days dealing with data no matter of their forms like plain text or binary data is a concrete problem. Data can be stored in local devices or resources or transmitted across networks. So, what if transmission path is unsafe or the storage devices can be hacked? The common answer is that data will be insecure. From this point of view the trouble can be figured. The major problems are the size and safety of data. In fact, the problem could not be solved unless having an algorithm or technique that could facilitate the process of maintaining data by scaling down the size and representing the data in some other secure form without losing the original data. So using cryptography will help solving the mentioned problem and accomplishing the main goal.

Cryptography is the art of attaining security by encoding or transforming data to make it non-readable for the community. Encoding is a subject which deals with the secret transmission of messages/data between two parties. Cryptography enables the transmitter and receiver to store delicate information and transmit it across insecure networks so it cannot be read by anyone except the authorized recipient. Furthermore, Cryptography is considered as a branch of both mathematics and computer science, and is associated with information theory, computer security and engineering too. Cryptography is the potent tool for controlling against many forms of security threats [4].

In fact, one of the most significant problems facing the digital data in general is the size of data and its protection. All types of data should be stored, sent, and received in a secure form with little size as much as it can be.

Many algorithms are presented to solve the problem as Lossless and lossy compression techniques “size of data” or encryption/decryption algorithms “data protection”.

One of the simplest form or technique of lossless data compression is Run-length encoding (RLE). It reduces strings by recurring characters to a single character. That means the run of characters “plain text source” is replaced with the number of the same characters and a single character which is the RLE principle. The RLE performance is based on a sequence of identical values of the input data [6-7].

Some other type or technique of lossless data compression is Huffman encoding. The Huffman algorithm is the one of earliest data-compression and encryption algorithms. It developed by David A. Huffman in 1979. The Huffman algorithm yields a variable and fixed-length binary code depending on the probabilities of each symbol of a source alphabet. The proof [3] that Huffman code can be surprisingly difficult to cryptanalyze helped us make a decision of using Huffman encoding as a second main step of changing the form of original plain text and as a private-key “closed key” of encoding/decoding. The final step is to cipher the data obtained from the previous step using RLE algorithm. As well, the RLE algorithm will be applied to reduce the size obtained data.

Our technique process can be summarized as follows: the original plain text will be encoded as ASCII code in binary form, then represented as a binary code using Huffman encoding. Finally, the RLE will check the repeating string of characters that obtained the Huffman file to produce non-repeated data with new cipher form. That will be done because of the properties of Huffman and RLE algorithms that have been discussed.

This paper will attempt to propose a technique to encrypt and compress the original plain text using Huffman and RLE algorithms respectively. That means the aim of this paper is to propose an algorithm or technique that presents a way of reducing the data size and encrypts it using two of Lossless techniques.

Compression Using Huffman encoding article [1] discussed the various techniques available for Lossless compression. The analysis of Huffman algorithm and comparison with various Lossless compression techniques as Arithmetic, LZW and Run Length Encoding has been stated. Comparison of common algorithms with Huffman Encoding has been performed from different points of view as the basis of their use in different applications and their advantages and disadvantages. The main conclusion is that the Huffman algorithm is used in JPEG compression. It produces optimal and compact code, but relatively slow. Huffman algorithm is based on a statistical model that adds to overhead. As well, the researcher concluded that arithmetic encoding is really effective for more frequently occurring sequences of pixels with fewer bits and reduces the file size dramatically. RLE is simple to implement and fast to execute. LZW algorithm is more adept to practice for TIFF, GIF and Textual Files.

Comparison of lossless data compression algorithms for text data article [2] tested six lossless data compression algorithms and compares their performance. A set of selected algorithms such as Huffman Encoding, Shannon Fano Algorithm, RLE, LZW, and Adaptive Huffman Algorithm are examined and followed out to evaluate the performance in compressing text data. Although they are tested for ten text files with different file sizes and different contents. The compression behavior depends on the category of the compression algorithm: lossy or lossless. Compression ratio, factor and time, and saving percentage are used to evaluate the performance of discussing lossless algorithms. It is important to mention that the performance depends on the type and the structure of the input source file. The main article's conclusion is that the Shannon Fano algorithm can be viewed as the most efficient algorithm comparing to other discussed algorithms. The reset value of this algorithm exhibit in an acceptable scope and it better results for the files with large size. Besides the article depicts the all discussed algorithms work well, except Run length encoding algorithm. Finally, the LZW algorithm does not work well for large file size.

The procedure of converting images into text format can be done. Lossless Huffman Encoding Technique for Image Compression and Reconstruction Using Binary Trees discussed the way of saving a black and white image, which its pixels of different shades of grey. Each pixel has a number value corresponding to the brightness or darkness of the shade. That stands for the pixel colored with black is 0 and white is 255, and all the numbers in between the black and white colors are shades of grey. So, each pixel is coded as some integer from 0 to 255 [9].

A digital image can be coded with respect to a model using Huffman encoding. It's well recognized that the Huffman's algorithm generates minimum redundancy codes compared to other algorithms. The presented scientific article presents a compression and decompression technique based on Huffman encoding and decoding for scan testing to reduce test data volume, test application time. The main conclusion is that the image compression method is well suited for grey scale (black and white) bitmap images [8].

Breaking a Huffman Code article examines the problem of deciphering a file that has been Huffman coded. The authors find that a Huffman code can be surprisingly difficult to cryptanalyze. The article introduces the analysis of the situation for a three-symbol source alphabet in details [3].

As a matter of fact, an image or audio file can be converted to a form of plain text to use as an input source. For instance, it is possible to convert any image into editable text file with software such as JiNa OCR Image to text. Furthermore an intermediate programmer can produce an application that will read each pixel in an image. Then save the information about the pixels that been read in text form.

The major contribution of this paper is developing an algorithm that will encode/decode digital data and reduce the input file size.

The sections of this paper are formed as follows: Part I shows the introduction, problem, goal, and related work. Part II represents suggested technique that will be used in the experiment. Part III discusses the experiment on some case studies. Part IV represents the conclusions and future work.

2 METHODOLOGY

2.1 Technique

Foremost of all, the algorithm can be described as the set of well-defined instructions that allows a computer to execute a specific task in a specific order [5] “well-defined procedure”. The technique is a procedure to complete a task.

Figure 1 shows the process sequence that manipulates the data presented as a plain text. Figure 2 depicts the steps of the proposed technique.

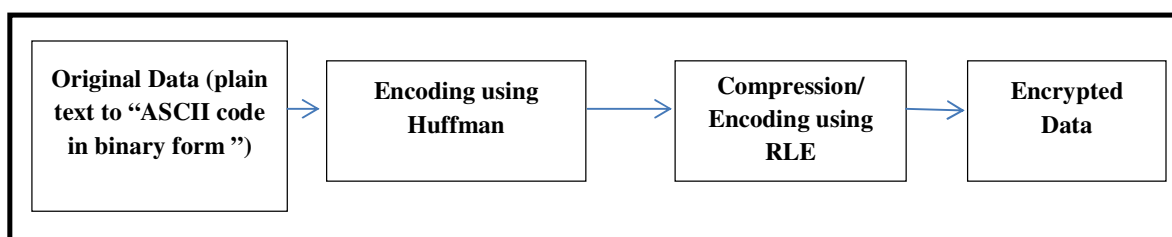


Fig. 1. Process of Encryption using Huffman and RLE techniques

Step 1: Start.
 Step 2: Recognize the type of original data “source”.
 Step 3: If the type of original data is plain text, go to step 5.
 Step 4: If the type of original data is non-plain text (e.g. Image), then represent the original data in plain text form.
 Step 5: Encode the original plain text to the ASCII code in a binary form.
 Step 6: Represent the original plain text in Huffman form.
 Step 7: Represent the data that’s been encoded using the Huffman (Step 6) in RLE form.
 Step 8: Compare the size and form of the original data with the size and form of output data in step 7
 Step 8: Calculate the saving percentage %.
 Step 9: Stop.

Fig. 2. Encoding/Decoding technique - steps

Figure 3 shows the application that was developed to implement the upper technique using NetBeans IDE. The developed application allows the user to pick a text file and represent it in Huffman and/or RLE form. All the obtained data will be saved in Huffman file and/or RLE text file. The information about the input/output text file as title, size, date, time, and used algorithm will be saved in the database to be used in comparison step.

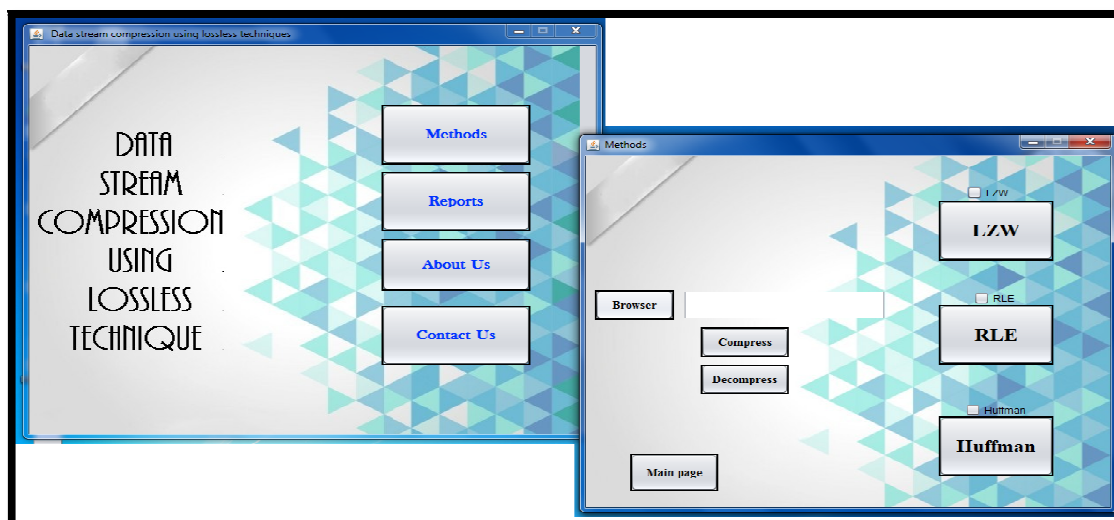


Fig. 3. Screenshots of the developed application

2.2 Experiments

This section contains the comparison table of experiments that were done on different original data. The tables (Table 1 and Table 2) show the original plain text, its size, and output data “in final step”. It is quite apparent that the data and its size change throughout the executed process.

The formula [2] that used to calculate the shrinkage of the source data as a percentage is:

$$\text{Saving percentage} = \frac{\text{Size before compression} - \text{Size after compression}}{\text{Size before compression}} \% \quad (1)$$

2.3 Results And Discussion

As seen in Table 1 the original data is changed to a different form after implementing Huffman and RLE algorithms respectively. Actually, implementing RLE directly on original data will change the input data in RLE form, but using Huffman coding followed by RLE to encode and find the repeated characters “recurring characters to a single character”, will change the input “original” data to a form that is characterized with more compression factor.

But what will happen if the suggested sequence process is changed, using the same input as in table 1; i.e. implementing RLE then Huffman algorithms on the same original data?

As examined in Table 2, the original data is changed to a different form after implementing RLE and Huffman algorithms respectively. But it is obvious, that the size of the output data is greater than the size of the output data that’s been obtained previously in table 1. From the point of view of encoding/decoding and the output data size, this proves that the suggested sequence process “Huffman-RLE” gives better results. Likewise, table 1 shows that better outcomes will be held if the type of the input data is ASCII code in binary form.

In fact, examining the idea of using the Huffman coding before the RLE algorithm could be explained as follows: the Huffman binary code of a character will not always take the same form of a binary code, it depends on its frequency and position in a word. For example, the Huffman binary code of “o” in the word “room” is 1 and the Huffman binary code of “o” in “room and moon” is 11, whereas the Huffman binary code of “o” in “domain+” is 00. Therefore, the outcomes of Huffman coding is not a static value. In addition the Huffman coded text differs from ASCII encoded text. For example the Huffman coded text of “room” is 110010, and the ASCII encoded text is 01110010011011110110111101101101. So, these two mentioned reasons give the suggested sequence process more validity.

As a consequence of what’s been discussed above, the character will have a different binary form using the Huffman algorithm.

Table 1. Encryption and data size changing using Huffman then RLE algorithms respectively

Source data type	Source data view (plain text)	Output data view (Encryption) “Implementing Huffman and RLE algorithms respectively”	Encoding/ Decoding (pass or fail)	Original size Bytes	Output size Bytes	Saving Percentage %	Size reducing (pass or fail)
The String “Alphabet”	Princess Nourah bint Abdulrahman University	130101021031201013031 201202120610101010140 104104101061303120410 610120213012013014010 410410101304102120510 102130103101061201031 201061010210	<u>Pass</u>	43	138	-220	Fail
ASCII code in binary form Of “Princess Nourah bint Abdulrahman University”	01010000011100 10011010010110 11100110001101 10010101110011 01110011001000 00010011100110 11110111010101 11001001100001 01101000001000 00011000100110 10010110111001 11010000100000 01000001011000 10011001000111 01010110110001 11001001100001 01101000011011 01011000010110 11100010000001 01010101101110 01101001011101 10011001010111 00100111001101 10100101110100 01111001	101051302102120102101 201302120312012021010 130212013021202106102 130212014013010101302 102120410120105106120 310212010210120130213 010410610510120310212 021031301010120120313 021021204101201041201 201012041012013031061 010101012013021201021 013012021202101013021 021302120120102101301 03140210	<u>Pass</u>	344	281	18	<u>Pass</u>

Table 2. Encryption and data size changing using RLE and Huffman algorithms respectively

Source data type	Source data view (plain text)	Output data view (Encryption) “Implementing RLE and Huffman algorithms respectively”	Encoding/ Decoding (pass or fail)	Original size Bytes	Output size Bytes	Saving Percentage %	Size reducing (pass or fail)
The string “Alphabet”	Princess Nourah bint Abdulrahman University	01010100010001111110 11010111000011111010 10110111100010111010 10010010110100101101 10100011101011100101 01110101001101111110 11010111101010111010 10111010100110101000 01011011011001001000 11101011100101100001 11010110101011101010 10101101011111101100 11011111001000111100 01111110111101010110 0	<u>Pass</u>	43	281	-553	Fail
ASCII code in binary form Of “Princess Nourah bint Abdulrahm an University ”	01010000011100 10011010010110 11100110001101 10010101110011 01110011001000 00010011100110 11110111010101 11001001100001 01101000001000 00011000100110 10010110111001 11010000100000 01000001011000 10011001000111 01010110110001 11001001100001 01101000011011 01011000010110 11100010000001 01010101101110 01101001011101 10011001010111 00100111001101 10100101110100 01111001	01000010001100001011 01011110001111011100 10001111000010111001 01101011110111011011 01110010111011110000 10000101101011110111 00101101011110111011 11000110001100011110 11010111101110010110 01001011010010000100 00101101011110001111 01110110011000010111 00100011000010001100 01101110110110001111 01110010001111000010 11100101101011110110 10010001100110001100 01100011000010000101 11011011000111101110 11110001101101101001 00001000010111001011 10110110110101111000 11110111011001100001 01110010001100110111 001011110010000101110 11001100001011100101 10101101100011000110 00010000100001000010 11100101101011110111 00100011110000101101 00101110111101110111 10000100001011010111 10001111011010111101 11001011100100011110	<u>Pass</u>	344	714	-107	Fail

		00010110100100011011					
		01100101111000					

2.3.1 Results

The two tables presented above show some results that could be summarized as follows:

1. Encoding/Decoding is passed because of changing the original data view as seen in output data view field. Which means the main aim is passed.
2. Encoding the input using Huffman encoding before RLE is implemented and makes the process more reliable.
3. When the input data is an ASCII code in binary form, the reduction test is pass and not fail. For example, the second experiment in table 1 shows that the size of the original data which is in binary format was decreased up to 18%. But in the first experiment, the size of the original data which is in text format is 43 bytes and the size of output data is 138 bytes. This means, the size of original data has increased 220%.
4. Implementing the RLE before Huffman coding of the original data gives bad results as seen in table 2.

3 CONCLUSIONS AND FUTURE WORK

The proposed technique can be utilized to encode and decode data and to increase security level during data communication. The encoded data is surprisingly difficult to cryptanalyze. The outcome is totally different compared with the original data. The decoding cannot be attained without using the Huffman-RLE algorithm sequence. So as a result, Encoding/decoding is done and passed if the input is an ASCII code in binary form, and the proposed technique reduces the size of data. This technique may be improved using the adaptive Huffman encoding technique.

REFERENCES

1. Mamta Sharma. Compression Using Huffman Encoding. IJCSNS International Journal of Computer Science and Network Security. 2010; 10(5): 133-141.
2. S.R. Kodituwakku, U.S. Amarasinghe. Comparison of lossless data compression algorithms for text data. Indian Journal of Computer Science and Engineering. ISSN : 0976-5166, 2001; 1(4) 416-425.
3. David W. Gillman, Mojdeh Mohtashemi, and Ronald L. Rivest. On Breaking a Huffman Code. IEEE TRANSACTIONS ON INFORMATION THEORY, 1996; 42(3): 972-976.
4. Nigam Sangwan. Text Encryption with Huffman Compression. International Journal of Computer Applications (0975-8887). 2012; 54(6): 29-32.
5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. Third printing, MIT Press. 2002.
6. A. Nagarajan, K. Alagarsamy. An Enhanced Approach in Run Length Encoding Scheme", International Journal of Engineering Trends and Technology. ISSN: 2231-5381. 2011; 2(1): 43-47.
7. S. Sarika, S. Srilali. Improved Run Length Encoding Scheme For Efficient Compression Data Rate. Int. Journal of Engineering Research and Applications. ISSN : 2248-9622. 2013; 3(6): 2017-2020.
8. Pujar, Jagadish H.; Kadlaskar, Lohit M. A new lossless method of image compression and decompression using Huffman encoding technique. Journal of Theoretical and Applied Information Technology. 2010; 15 (1/2): 18-23.
9. Mridul K. Mathur, Seema Loonker, Dheeraj Saxena. Lossless Huffman coding technique for image compression and reconstruction using binary trees. ISSN:2229-6093. 2012; Vol 3 (1): 76-79.