Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Combination of Cryptography Algorithm Knapsack and Run Length Enconding (RLE) Compression in Treatment of Text File

To cite this article: Paska Marto Hasugian *et al* 2020 *J. Phys.: Conf. Ser.* **1573** 012017

View the article online for updates and enhancements.

IOP | ebooks™

Bringing together innovative digital publishing with leading authors from the global scientific community.

Start exploring the collection–download the first chapter of every title for free.

This content was downloaded from IP address 181.215.77.10 on 28/07/2020 at 13:46

Combination of Cryptography Algorithm Knapsack and Run Length Enconding (RLE) Compression in Treatment of Text File

Paska Marto Hasugian[1*], Pandi Barita Nauli Simangunsong[2], Muhammad Iqbal Panjaitan[2], Dewi Wahyuni [2], Syarifah Fadillah Rezky[2]

[1]STMIK Pelita Nusantara Medan, Indonesia
[2]Universitas Imelda Medan, Indonesia

*Email: paskamarto86 @gmail.com

Abstract. The Knapsack algorithm cryptographic method is asymmetric cryptography in which the encryption key is different from the decryption key. In addition to text file security issues, the issue of the size of a text file is also a consideration. Large text files can be compressed by doing the compression process. Run Length Encoding (RLE) algorithm is an algorithm that reduces the size of a text file, if the text has a lot of repetition of characters. The combination of Knapsack and RLE algorithms can guarantee Text files cannot be seen by unauthorized users and can guarantee text files can be stored in low capacity media files. In this study, the authors made a combination of knapsack and RLE algorithm in text files. In the Knapsack algorithm there will be an increase in the size of the text file, this can be seen in the example of a case where the size of the plaintext (the original message) is 9 bytes, then after the encryption process the text file size becomes 7 bytes. Because of that the use of a combination of encryption and data compression is better because the file becomes smaller than the combination of data compression and encryption. Plaintext that has a lot of repetition of characters will be well compressed.

Keywords: Cryptography, Knapsack, RLE

1. Introduction

Cryptography is a method for sending secret messages so that only the intended recipient of the message can see, delete, disguise or read the message. The word cryptography comes from the Greek words kryptos which means hidden and grapein which means to write. The original message is called plaintext and the encoded message is called ciphertext. The message that has been encapsulated and sent is called a cryptogram. The process of changing plaintext into ciphertext is called encryption. Turning the ciphertext process into a plaintext is called decryption. Anyone involved in cryptography is called a cryptographer. On the other hand, the study of mathematical techniques for trying to defeat cryptographic methods is called cryptanalysis. Cryptanalysts are people practicing cryptanalysis. In addition to data security that needs to be considered is the speed of sending data [1]. This delivery

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

speed depends on the size of the information. One solution to this problem is to compress the data before sending and then the recipient will reconstruct it back into the original data (decompression).
The process of encoding plaintext into ciphertext is called encryption, also called enciphering. While the process of returning ciphertext to plaintext is called decryption or dechipering. Encryption and decryption can be applied to messages sent or stored messages. The term encryption of data in motion refers to the encryption of data transmitted through communication channels, while the term decryption of data at rest refers to the encryption of data that is in storage [2].

The Knapsack algorithm is a public-key cryptographic algorithm. The security of this algorithm lies in the difficulty of solving the knapsack problem. Knapsack means sack. Sacks have a limited load capacity. Items are placed in the bag only to the maximum capacity limit of the bag[3], [4].

The types of Knapsack Algorithms are as follows:
1. 0/1 Knapsack problem Each item consists of only one unit and may be taken or not at all
2. 0 / n Knapsack problem Each item consists of n for units and may be taken or not at all
3. Bounded Knapsack problem Every item is available n units and the number is limited
4. Unbounded Knapsack problem Every item is provided with more than one unit and the amount is unlimited
5. Fractional Knapsack problem Goods may be taken in the form of fractions or some.

The problems with the Knapsack Algorithm are as follows[5]:
Given the weight of the knapsack is M. It is known that n objects with their respective weights are w1, w2, ..., wn. Determine the value of b such that:

M = b1w1 + b2w2 + ... + bnwni

Which in this case, bi is 0 or 1. If b = 1, it means that object i is inserted into the knapsack, otherwise if bi = 0, object i is not entered. In algorithm theory, the knapsack problem belongs to the NP-complete group. Issues that are NP-complete cannot be solved in a polynomial time order.

Data compression is a general term for various algorithms and programs developed to overcome the problem of data compression[6]. The purpose of data compression is to reduce redundancy in stored data or transmitted data, thereby increasing effective data density. Three data compression algorithms are LZW, Arithmetic Coding, and Run-Length Encoding (RLE), where the RLE algorithm is assisted by the Burrows Wheeler Transform (BWT) algorithm to maximize its performance. These algorithms were chosen because they are all in the category of lossless algorithms, and represent each coding technique. The algorithm that has the best performance based on average ratio and compression / decompression processing time is LZW algorithm. Algorithm RLE + BWT is very effective for files that have many of the same character sets, but bad for files with irregular (random) contents. Just like the RLE + BWT algorithm, the LZW algorithm is also bad for files whose contents are random, but effective for plain text files, while the Arithmetic Coding algorithm is effective for all test files whether they are random or not.[7]

2. Research Method

In this study the data used are secondary data. The author obtained data from literature review and articles that the author obtained from supporting libraries, information from the internet, and journals related to text file data, security and cryptography as well as file compression techniques[8].

The system development method used is the Rapid Application Development System (RAD), which is a software development method that was created to reduce the time needed to design and implement information systems to produce a very short development cycle. This RAD model is an adaptation of a linear sequential model where rapid development is achieved using a component-based construction approach[9][10]. So, if the needs are well understood, the RAD process allows developers to create functional systems that are intact in a very short period of time (± 60 to 90 days).

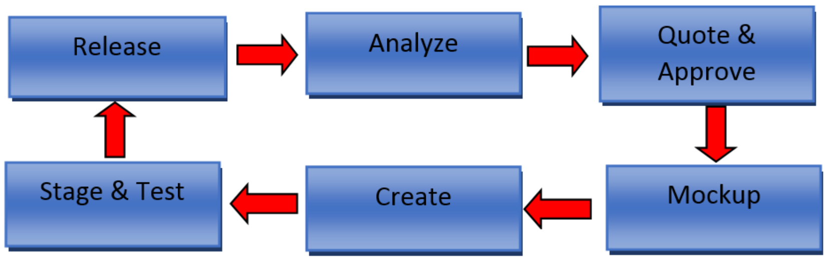


Figure 1. RAD Software Development Diagram[9][10]

3. Result and Discussion
Before the design stage of a system is carried out, it is necessary to analyze the system to be built. Systems analysis is a term that collectively describes the initial phases of system development. Basically, system analysis is a stage that is intended to create a comprehensive understanding of the system so that a description of the needs, work methods, and data flow that will be done by the system is obtained. This will help simplify the system implementation process. This system is expected to help solve problems in data security, so that the data remains unmodified or read by people who do not have access rights. In this system the text file will be encrypted using the Knapsack algorithm. Furthermore, for text data compression, it is performed using the RLE algorithm. It is hoped that this text file will be safe and the file size can be smaller than before. The main problem of this research is how to combine the knapsack and RLE algorithm in a text file. So hopefully this text file is maintained and the file size can be smaller than before.

Reduce the size of the repeating string character. This loop is usually called Run. Usually RLE encodes a run of symbols into two bytes, number and symbol. RLE can compress all types of data regardless of the contents of the information, but the contents of (the data to be compressed affects the compression ratio. RLE cannot achieve high compression ratios compared to other compression methods, but is easy to implement and fast to execute [6]. RLE is supported by most of the bitmap file formats such as TIFF, BMP, PCX.

Example:
S = 11111111111111100000000000000000001111
Can be represented as 1 as many as Fifteen, 0 as many as nineteen, and four for 1, namely (15, 1), (19, 0), (4, 1). because the maximum number of repetitions is 19, which can be represented by 5 bits, can be coded as bit streams (01111.1), (10011.0), (00100.1) [1].
Another example of the RLE algorithm:
1. KKKKKKK
   It is repeating the character "K" 7 times, this can be said as run length because it repeats the character 7 times
2. ABCDEFH
   Is an inaccurate example, because it does not experience repetition on each character. The seven characters above are seven different characters. This character cannot be said to be Run Length.
3. ABABBBC
   From the example three characters, there is the repeated character "B" 3 times, this character can already be said to be run length

The technique of the run length encoding algorithm is to abbreviate writing of a code, for completion of example 1 is to be able to write "KKKKKKKK" to ('r', '7', 'K') for shorter written "r7k" provided that number 7 is obtained because the character symbol "k" has repeated 7 times.
The solution for example two, ABCDEFG, cannot repeat the character on this symbol so that the writing can be changed to (‘n’’7’, "ABCDEFG") and shorter n7ABCDEFG.

3.1. Key Rise Process
To determine the sequence of secret keys, you must enter parameters as examples of these parameters then enter in the following formula:
w (3) = (distance) + w (1) + w (2); .... (1)
w (4) = (distance + 1) + w (1) + w (2) + w (3);
w (5) = (distance + 2) + w (1) + w (2) + w (3) + w (4);
w (6) = (distance + 3) + w (1) + w (2) + w (3) + w (4) + w (5);
w (7) = (distance) + w (1) + w (2) + w (3) + w (4) + w (5) + w (6);

where the initial value is w (1) = 2. So we get:
w (3) = 4 + 2 + 3 = 9
w (4) = (4 + 1) +2 +3 +9 = 19
w (5) = (4 + 2) + 2 + 3 + 9 + 19 = 39
w (6) = (4 + 3) + 2 + 3 + 9 + 19 + 39 = 79
w (7) = 4 + 2 + 3 + 9 + 19 + 39 + 79 = 155

then from these calculations it was found that the secret key was 2,3,9,19,39,79,155
Below is a view of the source code of the secret Revive Key process:
To determine the sequence of public keys, determine the Modulus value m which is a number that is greater than the sum of all elements in the sequence and the value of n is a value that has no attachment to m. In this system the value of m is generated randomly: m = 352 n = 5 calculated using the formula:
Kp = w (Cr) .n mod m .... (2)
Information:
Kp = row of Public Keys
w (Cr) = value of each secret key row
so we get: (2.5) mod 352 = 10
(3.5) mod 352 = 15
(9.5) mod 352 = 45
(19.5) mod 352 = 95
(39.5) mod 352 = 195
(79.5) mod 352 = 43
(155.5) mod 352 = 71
Rows of Public Keys are 10.15, 45, 95, 195, 43, 71

3.2 Compression and Encryption Process
For example the implementation in this case, for example to encrypt the plaintext 'AAAAAA'. Then you will get: Compression using RLE, plaintext "A" has repeated 7 times, then the compression of plaintext "AAAAAAA" is A7. To encrypt compressed plaintext, A7:
The 'A' character in ASCII has a binary value of 1000001
The '7' character in ASCII has a binary value 0110111
1st plaintext block: 1000001
Public key: 10 15 45 95 195 43 71
Cryptograms: (1x10) + (0x15) + (0x45) + (0x95) + (0x195) + (0x43) + (1x71) = 81
2nd plaintext block: 0110111
Public key: 10 15 45 95 195 43 71
Cryptograms: (0x10) + (1x15) + (1x45) + (0x95) + (1x195) + (1x43) + (1x71) = 369
Then the ciphertext of plaintext A7 is 81, 369

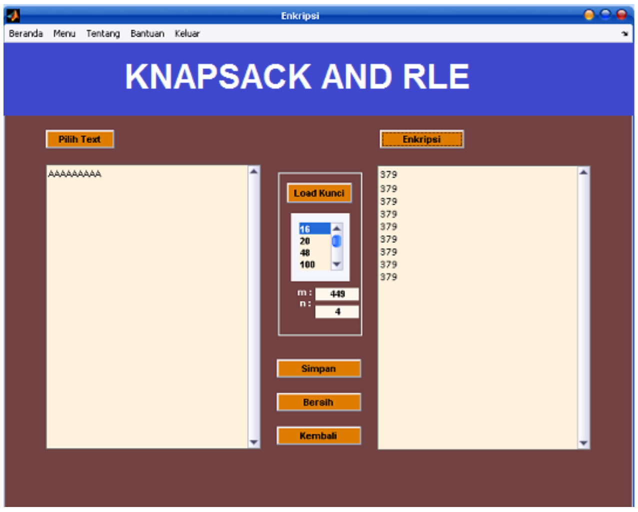
3.3. Encryption and Compression Process
The initial process in this stage is data encryption first then data compression. The public key, secret key and plaintext example are the same as the data above.
Plaintext = AAAAAA
Public Key = 10 15 45 95 195 43 71
Secret Key = 2 3 9 19 39 79 155
Value of m = 352
Value of n = 5
Value of n-1 = 141

Plaintext encryption:
Plaintext block A: 1000001
Public key: 10 15 45 95 195 43 71
Cryptograms: (1x10) + (0x15) + (0x45) + (0x95) + (0x195) + (0x43) + (1x71) = 81
Then the ciphertext is 81, 81, 81, 81, 81, 81, 81

Plaintext compression: 81 repeats seven times, so the compression is 81 7
To restore the initial plaintext, it decompresses the data first and then decrypts the data.

3.4. Decryption and Decompression Process
The repetition of 81 is the object that repeats and 7 is the value of the recurrence, then 81 is obtained seven times. So that the initial plaintext can be obtained is "81 81 81 81 81 81 81". 81 81 81 81 81 81 81 81 is not an initial plaintext, to determine the text data decryption required: 81. 141 mod 352 = 157 = 2 + 155, corresponds to 1000001 is A Then the initial plaintext is obtained "AAAAAAA"

System implementation phase is the stage of making software, the next stage of the testing system design stage. The stage is carried out to translate the design based on the results of test analysis in a language understood by the computer as well as the application of software in the actual situation. Implementation of the results of the analysis and design stages can be seen from the appearance of the system interface as follows:
This form is where users interact with the system in compressing text can be seen in Figure 2 below.



AAAAAAAAA

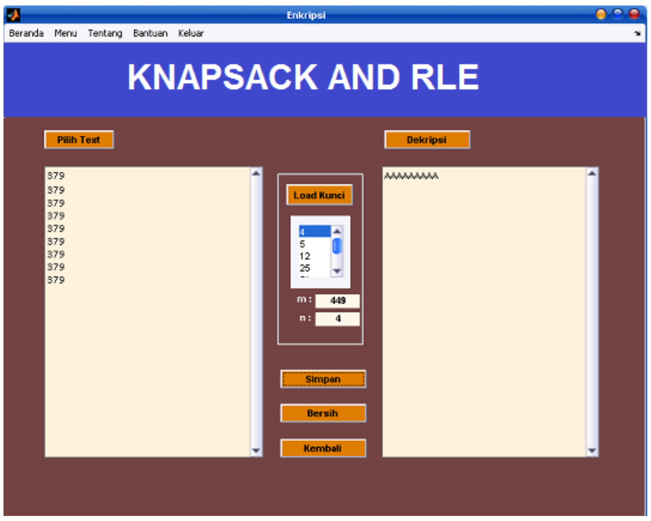
```
16
20
48
100
```

m : 449
n : 4

```
379
379
379
379
379
379
379
379
379
```

Figure 2. Display Encryption Testing

This form is where users interact with the system in decrypting text can be seen in Figure 3 below, which returns ciphertext to plaintext.



```
379
379
379
379
379
379
379
379
379
```

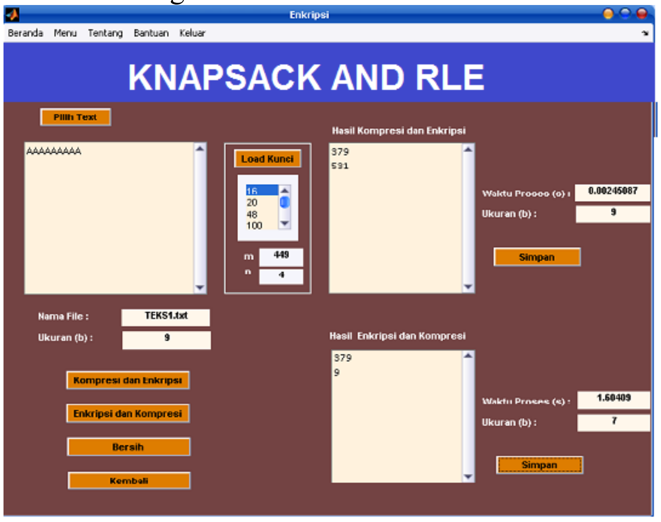
```
4
5
12
25
```

m : 449
n : 4

AAAAAAAAA

Figure 3. Display Decryption Testing

In the picture above, the user is required to enter an encrypted file called File Encryption. To continue, the user presses the load key to display the decryption key and the Decryption key to return the text to the original.
This form is the core of the purpose of this study, which combines Knapsack and RLE algorithms in text files. Specifically for the Knapsack and RLE combination testing, two different texts were tested, TEXT1.txt containing "AAAAAAAAA" and TEXT2.txt containing "ABABABABA". Testing of TEXT1 and TEXT2 can be seen in Figure 4 below.



AAAAAAAAA

```
16
20
48
100
```

m 449
n 4

Nama File : TEKS1.txt
Ukuran (b) : 9

Hasil Kompresi dan Enkripsi

```
379
531
```

Waktu Proses (s) : 0.00245087
Ukuran (b) : 9

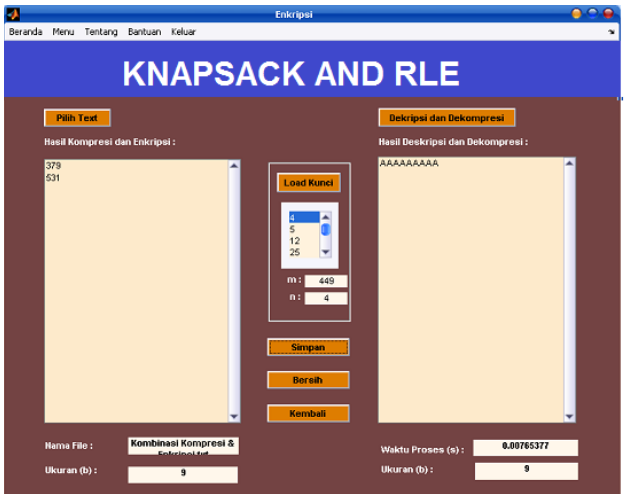
Hasil Enkripsi dan Kompresi

```
379
9
```

Waktu Proses (s) : 1.60409
Ukuran (b) : 7

Figure 4. Display of Combination Testing for TEXT1

This form is a display to return the results of a Combination of Compression and Encryption in a text file. Decryption and Decompression Testing is also done with the two texts above. tEKS2.txt can be seen in Figure 5 below.



Hasil Kompresi dan Enkripsi :

```
379
531
```

4
5
12
25

m : 449
n : 4

Hasil Deskripsi dan Dekompresi :

```
AAAAAAAAA
```

Nama File : Kombinasi Kompresi & Enkripsi.txt
Ukuran (b) : 9
Waktu Proses (s) : 0.00765377
Ukuran (b) : 9

Figure 5. Display of the Decryption and Decompression Test for TEXT1.

4. Conclusion

After testing, the following conclusions can be drawn

1. Text Compression using the Run Length Encoding algorithm can be compressed well by the system if there are many sequential letters in the text. As for text that has little or no repetition of letters at all, compression using this algorithm is not done well because the file size increases from the original size.
2. Text encryption using the Knapsack algorithm can secure messages properly.
3. Combination takes precedence Encryption then Text compression is better used because the combination of the two successfully compresses well compared to prioritizing Compression then encrypting it even though the time to execute plaintext is longer.
4. The combination of Knapsack and RLE Algorithms in the Text file is appropriate if in a plaintext (the original message) has a lot of repetition of characters..

References

[1] R. Munir, “Kriptografi,” *Inform. Bandung*, 2006.
[2] T. Limbong and P. D. P. Silitonga, “Testing the Classic Caesar Cipher Cryptography using of Matlab,” *Int. J. Eng. Res. Technol.*, vol. 6, no. 2, pp. 175–178, 2017.
[3] G. I. Sampurno, E. Sugiharti, and A. Alamsyah, “Comparison of Dynamic Programming Algorithm and Greedy Algorithm on Integer Knapsack Problem in Freight Transportation,” *Sci. J. Informatics*, vol. 5, no. 1, p. 49, May 2018.
[4] R. Munir, “Algoritma Knapsack,” pp. 0–18, 2004.
[5] A. Kataria, “Algorithm for fractional knapsack problem,” 2018. [Online]. Available: https://www.includehelp.com/algorithms/fractional-knapsack-problem.aspx. [Accessed: 14-Dec-2019].
[6] H. K., A. A., V. U., and V. K., “New Modified RLE Algorithms to Compress Grayscale Images with Lossy and Lossless Compression,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 7, pp. 250–255, 2016.
[7] A. Mubark and A. Ibrahim, “Comparison Between (RLE And Huffman) Algorithmsfor Lossless Data Compression,” no. 3, pp. 1808–1812, 2015.
[8] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif dan R&D.* Bandung: PT Alfabet, 2016.
[9] M. Devi, “Rapid Application Development,” 2014. [Online]. Available: http://machlizadevi.blog.binusian.org/2014/06/03/rapid-application-development/.

[Accessed: 13-Dec-2019].
[10] X. Franch, L. Lopez, S. Martínez-Fernández, M. Oriol, P. Rodríguez, and A. Trendowicz, “Quality-Aware Rapid Software Development Project: The Q-Rapids Project,” 2019, pp. 378–392.