Journal of Theoretical and Applied Information Technology

30th November 2023. Vol.101. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SECURING AND COMPRESSING TEXT FILES USING THE AES 256 ALGORITHM AND LEVENSTEIN CODE

HANDRIZAL[1], FAUZAN NURAHMADI[2], MUHAMMAD FADLI[3]

[1,2,3]Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia

E-Mail: handrizal@usu.ac.id

ABSTRACT

In using technology, humans cannot exchange information without the internet. However, along with the development of technology, Data security, and confidentiality are significant issues to be considered in communication since there is a negative impact in the form of data tapping, which results in data being seen and information being taken or possessed by persons who do not have access rights. Methods for securing data are needed in this case. Cryptography is extremely appropriate in the field of data security. The AES algorithm is a security system for protecting data. However, in this case, the impact of cryptography is to create large files, so compression techniques are needed. As a result, a compression method is needed, specifically the Levenstein code, required to compress ciphertext in smaller size. The test used 2 types of characters, namely homogeneous and heterogeneous characters. The test parameters used Ratio of Compression, Compression ratio, space-saving, and bitrate. From the test results, it was found that encryption with AES 256 experienced an increase in the number of characters, but the resulting character length has the same ciphertext length. Testing with the Ratio of Compression parameter shows that homogeneous characters have a smaller percentage value of 62.09% than heterogeneous characters 62.22%. The compression Ratio test for homogeneous characters has the same ratio as heterogeneous characters, namely 1.61. In space saving test, the percentage of homogeneous characters is 37.91% greater than heterogeneous characters at 37.78%. Testing based on Bitrate, homogeneous characters with heterogeneous characters has the same value of 4.97. Moreover, in The time comparison, homogeneous characters are faster than heterogeneous characters on a 1000-character test with a homogeneous time of 1.0907 ms and a heterogeneous time of 1.7975 ms.

Keywords: *AES 256, Cryptography, Compression, Levenstein Code*

1. INTRODUCTION

In using technology, humans cannot exchange information without the internet. However, along with the development of technology, Data security, and confidentiality are significant issues to be considered in communication since there is a negative impact in the form of data tapping, which results in data being seen and information being taken or possessed by persons who do not have access rights. Methods for securing data are needed in this case. Cryptography is extremely appropriate in the field of data security.

The AES algorithm is a secure algorithm for protecting data. This AES was first published by NIST (National Institute of Standards and Technology) where it replaced the DES algorithm which was considered outdated and easy to break into. AES is proven to be immune to conventional attacks (linear and differential attacks) that use statistics to crack passwords, and every encryption and decryption process must perform 10 turns or 10 rounds in carrying out security or opening the security.

In the AES modification, there are certain changes made in the key-encryption process by improving the encryption quality and also improving the existing effect on the chipper image to be produced. The difference in key length will affect the number of rounds in the AES algorithm [1]. Research that has been done on Modified AES for text and Image Encryption explained that the results of AES modifications have increased efficiency due to faster timing and reduced CPU usage where this AES algorithm also produces higher effects that can improve algorithm

performance [2]. For the Levenstein algorithm, Vladir Levenstein developed a code in 1968 that could be used for non-negative integers. There are several encoding steps in searching for this algorithm code. The steps in decompression with this algorithm are to enter the result of the compression text, the result of the string bit of the Levenstein code that becomes the value of the file is changed back to binary, returns the binary to the original string bit by reading the last 8 bits, the reading result is a decimal number, declare the reading with n and then remove the bit in the last part by 7 + n.

The number of studies described above means that the security and confidentiality of this data is a factor that needs to be considered, especially with the study and development of technology today. It also relates to data compression which is done to facilitate with some cryptographic or compression methods.

2. SCOPE AND LIMITATION

The scope and limitations of this research are as follows:

- The data used are text files with UTF-8 encoding format, such as *.txt, and *html.
- The cryptographic method used is the Aes 256 Algorithm and the Levenstein Code Algorithm.
- The measurement parameters are calculating the compression ratio, compression ratio (CR), space-saving (SS), and bit rate.
- The programming language to be used is C#.

3. LITERATURE REVIEW

3.1. Cryptography

Etymologically the word cryptography comes from the Greek language which consists of two words krupto which means hidden and grafh which means writing. This can mean that cryptography is a rule in the encryption and decryption of data or messages. The message to be encrypted is commonly called plaintext. After that, an algorithm is used to encrypt the message so that the plaintext that has been encoded is called ciphertext/ cipher text. Classical Cryptography In this algorithm, conventional or symmetric encryption techniques are used namely substitution techniques and transposition techniques. However, the algorithm is outdated because it is so easy to crack that it is no longer safe [3]. Modern Cryptography This algorithm focuses on the level of the algorithm and the key used. It generally operates in bit mode compared to character mode. Modern cryptography is an improvement that refers to cryptography.

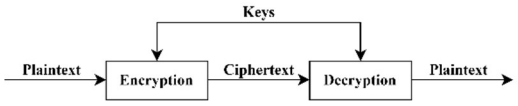

Figure 1. Asymmetric cryptographic schemes

3.2. Compression

Compression is a process of converting a set of data into a form of code or symbol that is used as a utilization to minimize data storage and shorten the time in transferring data. Data compression in simple terms has characteristics that can be likened to a process to replace a string in the form of a collection of characters into a new string with the same content but with less length or size.

3.2.1. Lossy Compression.

This lossy compression is a type of compression that can cause the data resulting from the compression to be lost or cannot be restored like the original data [4]. This compression can be used for images or videos, but it is not suitable for use in text form. Examples of lossy compression algorithms are CS&Q (coarser sampling and/or quantization), JPEG, and MPEG. The Lossy Compression Scheme can be seen in Figure 2.

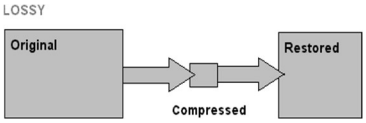

Figure 2. Lossy Compression Scheme

3.2.2. Lossless Compression.

Lossless Compression is a type of compression that does not allow the difference between the initial data (before compression) and the loss after compression. Lossless compression has a smaller degree of compression but with complete data accuracy between before and after the compression

process. Lossless compression can work by using repeated patterns on messages that the compression process will carry out and the process of coding those patterns is carried out efficiently. Examples of algorithms in lossless compression are Run-Length, Huffman, Delta, Lempel Ziv Welch (LZW), Elias Omega Code, Levenstein Code, and others. The Lossless Compression Scheme can be seen in Figure 3.

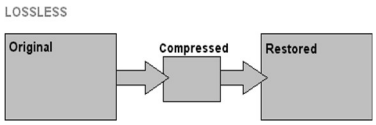

Figure 3. Lossless Compression Scheme

3.3. Test Parameters

Several factors are often used in analyzing the quality of a data compression technique as follows:

1. Ratio of Compression (Rc)

   Ratio of compression (Rc) is the percentage of data size after the compression process is carried out with the size of the data before the compression process is carried out.

$$Rc = \frac{\text{Data Size After Compression}}{\text{Data Size Before Compression}} \times 100\%$$

2. Compression Ratio (Cr)

   Compression ratio (Cr) is a comparison of the size of the data before the compression process with the data after the compression process.

$$Cr = \frac{\text{Data Size Before Compression}}{\text{Data Size After Compression}}$$

3. Space Savings (Ss)

   Space Savings (Ss) is a percentage of storage space savings.

$$Ss = \left(1 - \frac{\text{Data Size After Compression}}{\text{Data Size Before Compression}}\right) \times 100\ \%$$

4. Bit rate

   Bit Rate is the average number of bits used to encode a single character, formulated by the compressed bit size divided by the number of unique characters in the text.

$$\text{Bit Rate} = \frac{\text{Number of bits compressed}}{\textit{Number of Unique Characters In Text}}$$

3.4. Research Problem

There are several problems in this research, namely:

- Negative impacts in the form of data tapping which results in data being seen and information taken or owned by people who do not have access rights.
- This also relates to data compression which is done to make it easier with several cryptographic or compression methods.

4. METHOD

3.1. Advanced Encryption Standard Algorithm (AES)

The AES algorithm is a secure algorithm for protecting data. This AES was first published by NIST (National Institute of Standards and Technology) where it replaced the DES algorithm which was considered outdated and easy to break into. AES is proven to be immune to conventional attacks (linear and differential attacks) that use statistics to crack passwords, and every encryption and decryption process must perform 10 turns or 10 rounds in carrying out security.

3.1.1. AES Algorithm Encryption Process

There are 4 types of byte transformations for the AES algorithm encryption process, namely SubBytes, ShiftRows, and AddRoundKey. The AES encryption process can be described in the picture below :

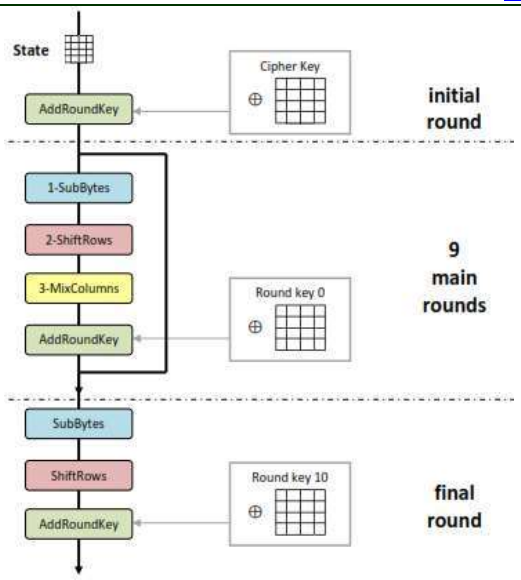

Figure 4. AES Algorithm Encryption Process Illustration [5]

3.1.2. AES Algorithm Decryption Process

The cipher transformation can be reversed and implemented in the opposite direction to produce an inverse cipher that can be easy to understand. The byte transformations used are InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey.

The decryption process of the AES algorithm can be seen in Figure 5 below :

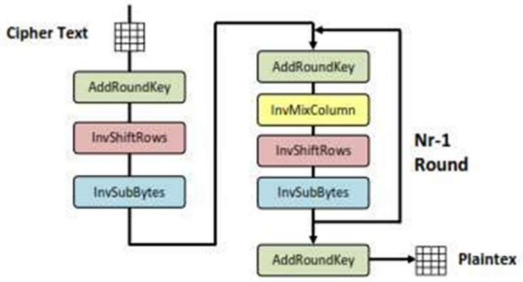

Figure 5. AES Decryption Process Illustration [5]

3.2. Levenstein Code Algorithm

This little-known code for non-negative integers was created in 1968 by Vladimir Levenshtein. Both encoding and decoding are multi-step processes [6]. The Levenstein process code for n=0 is 0. As for the positive number N, do the following :

1. Denote the first number of C by 1. Initialize the current code on an empty string.
2. Take the binary value of n without the previous value of 1. Add it to the current code.
3. Denote M as the number of bits added in stage 2.
4. If M ≠ 0, add C with 1 and do step 2 back, but with a value of M instead of n.
5. If M = 0, add 1 followed by 0 on C to the current code and stop.

The decoding process is carried out with the following steps:

1. Denote C with a sequential value of number 1 before the first number 0.
2. If C = 0, the decoding value is zero, stop.
3. Denote N=1, and repeat step 4 (C-1) times.
4. Read N bits, add 1, and set the result of the bitstring to N (thereby removing the previous value from N). The string entered for N in the last iteration is its decoding value.

Table 1. Levenstein Code

n	Levenstein Code
0	0
1	10
2	1100
3	1101
4	1110000
5	1110001
6	1110010
7	1110011
8	11101000
9	11101001
10	11101010
11	11101011
12	11101100

3.3. General Architecture

General Architecture is an overview of how a system runs and the features that support the resolution of a problem. The general architecture in Figure 6 below can be seen in that the user inputs a

text file and then processes it by the system with the Aes-256 Algorithm and also the Leveinstein Code Algorithm. Then the output of the file is decompressed by the system and produces a decompressed text result file.

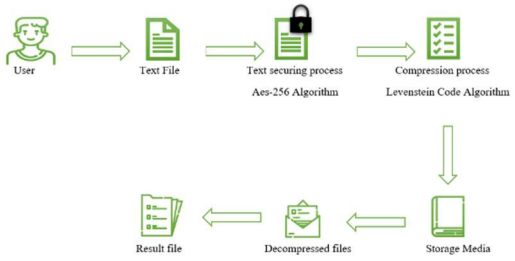

Figure 6. General Architecture of the System

5. RESULTS AND DISCUSSIONS

4.1. System implementation

At this stage the system is built using the C# programming language and using the Sharp Develop 5.1 application, in the process of building this system will be divided into three forms in the C# language including the Main Page, Encryption-Compression Page, and Decompression-Decryption Page.

4.1.1. Testing Encryption – compression

The initial stage in carrying out the compression encryption process is to press the encryption-compression menu.

After that, follow these steps:

1. Press the "browse" button to open the File Dialog, and select the text file (*.txt) as the file you want to encrypt and compress
2. The user must enter the key for encryption in the key textbox.
3. If you have entered the key, the user can press the encryption button to carry out the encryption process and the result in the form of ciphertext will be seen on the right side, namely the ciphertext textbox.
4. On the compression side, you can see ciphertext characters that are ready to carry out the compression process
5. The user can press the compression button to perform the compression process. If the compression has been completed, information will appear in the form of the ratio of compression, compression ratio, space-saving, bitrate, and running time which can be seen in Figure 7.

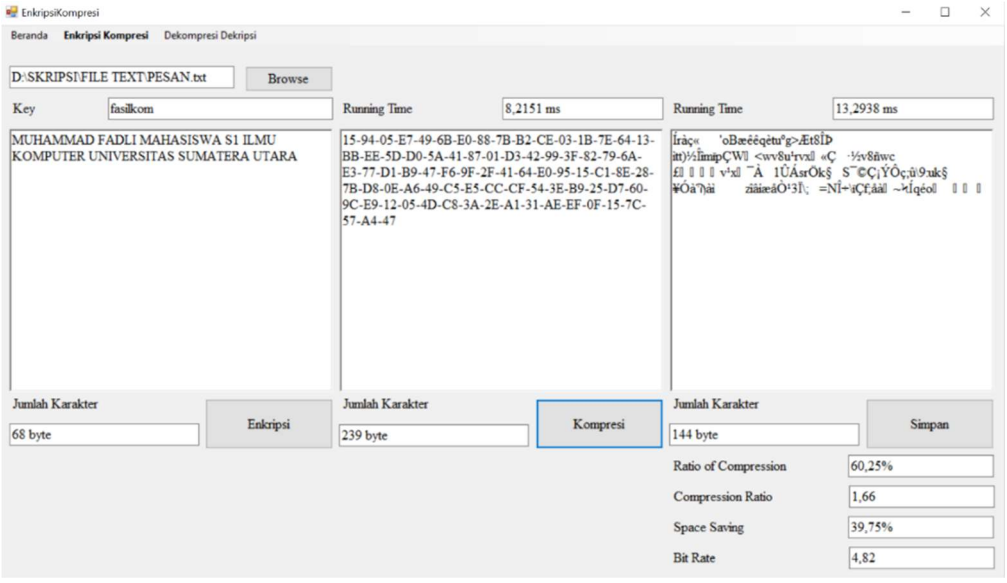

Figure 7. Compression Encryption Text File View

4.2. Test Results

Table 2. AES test results with Homogeneous Strings

Number of Characters	Character Type	Key	Amount of Ciphertext	Encryption Time (ms)	Decryption Time (ms)
100	1	fasilkom	335	0,4592	0,4326
200	1	fasilkom	623	0,5279	0,5047
300	1	fasilkom	911	0,6259	0,6025
400	1	fasilkom	1247	0,7661	0,6422
500	1	fasilkom	1535	0,9043	0,7397
1000	1	fasilkom	3023	2,0456	1,0907

Table 3. Levenstein Code test results with Homogeneous Strings

Amount of Ciphertext	Compression result (Number of Characters)	Ratio Of Compression (%)	CR	SS (%)	Bitrate	Compression Time (ms)	Decompression Time (ms)
335	202	60,3	1,66	39,7	4,82	0,5768	1,9756
623	384	61,64	1,62	38,36	4,93	1,4389	3,7521
911	565	62,02	1,61	37,98	4,96	1,4395	6,4339
1247	774	62,07	1,61	37,93	4,97	1,6926	11,5864
1535	948	61,76	1,62	38,24	4,94	2,0529	14,0352
3023	1877	62,09	1,61	37,91	4,97	4,0384	29,4348

4.2.1. Testing systems with Homogeneous Strings

For the results of homogeneous string testing performed by the AES 256 and Levenstein Code algorithms in Table 2, and Table 3 above, it can be seen that the comparison between the amount of text generated by the AES 256 cryptographic process has increased the number of ciphertext characters. The characters tested were 100 characters originally to 335, 200 to 623, 300 to 911, 400 to 1247, 500 to 1535, and 1000 to 3023 characters. Thus, there is an increase in the number of characters, so compression techniques are needed to reduce the bit size and number of characters.

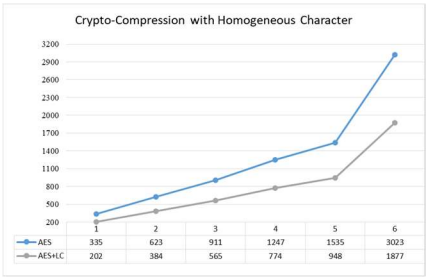

Figure 8. Ciphertext Comparison Chart with Homogeneous Compression

Table 4. AES test results with Heterogeneous Strings

Number of Characters	Character Type	Key	Amount of Ciphertext	Encryption Time (ms)	Decryption Time (ms)
100	36	fasilkom	335	0,5291	0,4513
200	36	fasilkom	623	0,3642	0,5237
300	36	fasilkom	911	0,6514	0,5755
400	36	fasilkom	1247	0,7835	0,6495
500	36	fasilkom	1535	0,8839	1,4267
1000	36	fasilkom	3023	2,1483	1,7975

Table 5. Levenstein Code test results with Heterogeneous Strings

Amount of Ciphertext	Compression result (Number of Characters)	Ratio Of Compression (%)	CR	SS (%)	Bitrate	Compression Time (ms)	Decompression Time (ms)
335	197	58,81	1,7	41,19	4,7	0,5778	2,1545
623	376	60,35	1,66	39,65	4,83	0,9409	3,8894
911	563	61,8	1,62	38,2	4,94	1,6403	6,8557
1247	768	61,59	1,62	38,41	4,93	1,3883	13,3486
1535	953	62,08	1,61	37,92	4,97	2,0523	14,8454
3023	1881	62,22	1,61	37,78	4,98	4,2732	26,8320

4.2.2. Testing system with Heterogeneous Strings

For the results of heterogeneous string testing performed by the AES 256 and Levenstein Code algorithms in Table 4, and Table 5 above, it can be seen that the comparison between the amount of text generated by the AES 256 cryptographic process has increased the number of ciphertext characters. The characters tested were from the original 100 characters to 335, 200 to 623, 300 to 911, 400 to 1247, 500 to 1535, and 1000 to 3023 characters. Because of the increase in characters, compression techniques are used to reduce the bit size and number of characters.

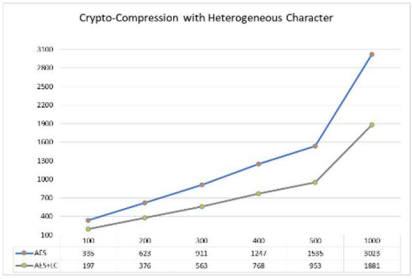

Figure 9. Ciphertext Comparison Chart with Heterogeneous Compression

4.2.3. Independent testing with t-test sample

Based on tests that have been carried out with homogeneous strings and heterogeneous strings, it was found that there was a difference in values between the two. Therefore, to corroborate the results of the test, it is necessary to do a t-test. In this study, the t-test used was the Independent t-test. Independent sample t-test is a test used to determine the difference in mean or mean values between two free groups or two groups that do not belong to the same subject. The requirements for conducting this test are from different groups, the data type is numeric, and for the data scale, namely intervals and ratios, the data is normally distributed.

In this case, testing will be carried out with different groups between homogeneous strings and heterogeneous strings that will be analyzed with SPSS.

The results of the tests that have been carried out with the t-test can be described in the table below :

Table 6. Independent t-test results

α = 0,05	Results
Mean	Homogen = 4570.80 Heterogen = 4567.57
t count	0.004
df	58
Sig (2-tailed)	0,997
Mean difference	3.133

From the table above, we can see for the interpretation of the test results from the independent t-test. The average value of testing with homogeneous strings is 4570.80, while for testing with heterogeneous strings, it is 4567.67. So in conclusion, there is a difference in the average test results. To prove whether the difference is

significant (real) or not, it can be interpreted by way of decision-making.

One of the decisions made in the independent t-test is by looking at the Sig. (2-tailed) value of the SPSS output as in Figure 4.24 obtained 0.997. If Sig. (2-tailed) > p value then H0 is accepted and Ha is rejected, and if Sig. (2-tailed) < p value then H0 is rejected and Ha is accepted.

So in this case it is found that the value is 0.997 > 0.05 so H0 is accepted and Ha is rejected. This means There is no significant difference between the average test value of homogeneous strings and heterogeneous strings.

5. CONCLUSION

Based on the results of the literature study, implementation, and testing carried out, the following conclusions are obtained:

The AES 256 algorithm and the Levenstein Code algorithm in securing & and compressing text files, it was found that the implementation was carried out by testing the system which was then successfully implemented and encoded into scrambled messages until successfully decrypted into the original message and also compressed and decompressed the text file.

Tests on homogeneous and heterogeneous characters based on the Ratio of Compression parameter showed that homogeneous characters had a smaller percentage value of 62.09% than heterogeneous characters, namely 62.22%.

The Compression Ratio test shows that the Levenstein Code algorithm with homogeneous characters has the same ratio as heterogeneous characters, namely 1.61.

Testing space saving with the Levenstein Code algorithm with homogeneous and heterogeneous characters, the results show that the percentage of homogeneous characters is 37.91% greater than heterogeneous characters 37.78% because the characters tested have similar data and take up less memory space.

Tests on homogeneous and heterogeneous characters based on Bitrate, homogeneous characters with heterogeneous characters have the same value, namely 4.97

REFERENCES:

[1] B. Langenberg, H. Pham, and R. Steinwandt, "Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit," *IEEE Trans. Quantum Eng.*, vol. 1, pp. 1–12, 2021, doi: 10.1109/tqe.2020.2965697.

[2] Priyanka Sharma, "A New Image Encryption using Modified AES Algorithm and its Comparision with AES," *Int. J. Eng. Res.*, vol. V9, no. 08, pp. 194–197, 2020, doi: 10.17577/ijertv9is080083.

[3] D. Ariyus, *Pengantar Ilmu Kriptografi : Teori Analisis dan Implementasi*. Yogyakarta: CV Andi Offset, 2008.

[4] D. Putra, "Pengolahan Citra." Andi, Yogyakarta, 2010.

[5] A. Prameshwari and P. N. Sastra, "Implementasi Algoritma Advanced Encryption Standard (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen," *J. Eksplora Inform.*, vol. 8, 2018.

[6] D. Salomon, "Data Compression: The Complete Reference." IEEE Signal Processing Magazine, Northridge, 2007.