Advances in Intelligent Systems Research, volume 172
Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)

Improving the SMS Security and Data Capacity Using Advanced Encryption Standard and Huffman Compression

LAURENTINUS[1], Harrizki Arie PRADANA[2], Dwi Yuny SYLFANIA[3], Fransiskus Panca JUNIAWAN[4]

[1]*laurentinus@atmaluhur.ac.id, Department of Computer Science, STMIK Atma Luhur, Jl. Jend. Sudirman, Pangkalpinang, Indonesia*
[2]*harrizkiariep@atmaluhur.ac.id, Department of Computer Science, STMIK Atma Luhur, Jl. Jend. Sudirman, Pangkalpinang, Indonesia*
[3]*dysylfania@atmaluhur.ac.id, Department of Computer Science, STMIK Atma Luhur, Jl. Jend. Sudirman, Pangkalpinang, Indonesia*
[4]*fransiskus.pj@atmaluhur.ac.id, Department of Computer Science, STMIK Atma Luhur, Jl. Jend. Sudirman, Pangkalpinang, Indonesia*

ABSTRACT

The development of technology that increasingly rapidly has a significant influence on telecommunications. One feature of cellular telephones is SMS (Short Message Service). Still, this facility in the form of SMS (Short Message Service) has a vulnerability in the way of information leakage. Therefore an encryption application is proposed using OOP (Object Oriented Programming) method with Waterfall's model and (Unified Modeling Language) as tools. To the maintenance of the confidentiality of a message, it required AES (Advanced Encryption Standard) cryptography. Cryptography AES (Advanced Encryption Standard) had functions to encode messages into the form of ciphertext. The results of characters from encryption will usually have more than before because there is an addition to the cipher on each encrypted data. Therefore, Huffman algorithm compression is needed for text compression so that the results of encrypted messages become less. The purpose of this study is to maintain the confidentiality of messages. The results obtained in this study are differences in the number of text encryption characters AES (Advanced Encryption Standard) with the addition of Huffman compression, which is 17.35%. The implementation of this message application was successfully implemented using the AES algorithm and the Huffman algorithm. The message is not can be read by the service center, and the number of characters sent messages can compress into fewer.

Keywords: *SMS, cryptography, compression, advances encryption standard, Huffman*

1. INTRODUCTION

The mobile phone provides various features, one of the most popular feature is SMS (Short Message Service). SMS is a one of mobile phone service that allows users to communicate with each other by sending short messages in the form of text instantly and for a small fee. SMS is not securing the messages so we are not suppose to send a sensitive or confidential message. Therefore we need an application that can guarantee the security of sending SMS messages because there are millions of unprotected messages. The selection of the right algorithm is another important aspect, seen from the level of importance of the message. A good algorithm makes encryption unpredictable so cannot be disassembled using any method.

AES algorithm is an encryption method that uses asymmetry keys and licenses to use confidentiality [16]. SMS services have used in every field, such as banking for m-banking, e-commerce, and personal transactions.

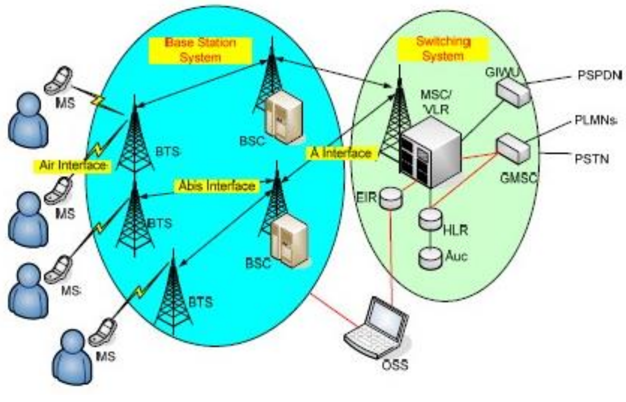


Figure 1. GSM service structure [15]

In addition to discussing cryptography [8]-[10], this study also discusses data compression. how much encrypted data can be compressed using the huffman algorithm.

A research conducted with a comparative analysis of the performance and security of the most useful algorithms:

RC6 (Rivest Cipher) and RSA (Rivest Shamirdan Adleman). This study aims to determine the complexity of encryption and decryption algorithms better. For proof, it develops Android-based SMS encryption and decryption prototypes. The results show that all messages can be encrypted using the key generated before sending to the recipient. Encrypted messages also have better security and time performance[1]. Another research proposes an encrypted JPEG image capture scheme. Retrieval is based on the Huffman code in the JPEG bitstream. There are three parties involved, namely the content owner, cloud server, and authorized users. The testing results shows that the proposed scheme can ensure confidentiality, integrity, and format compatibility of the JPEG image. Besides taking pictures in terms of various factors the quality of the image is still fairly effective [2]. Huffman is also applied for image compression based on Huffman coding [11]-[12]-[14]. Conducted a research by taking variables based on eye image compression and image compression methods. Compression is performed at the stage of image filtering by wavelet transforms to remove excessive information in the image. The proposed Huffman method is used in the image encoding process. From the test results obtained that the image format size JPEG images can be reduced in the same image effect [3]. Other research proposes a fast algorithm for decoding Huffman based on the recursion Huffman tree. This research was conducted by focusing on the efficiency of Huffman's decoding time. To speed up the decoding process numerical interpretation is used. The results of tests carried out from the test file show that the average number of symbols decoded at one time for the proposed method ranges from 1.91 to 2.13 with the processing unit condition is 10. From the experimental comparison shows that the decoding time of the proposed method increased rapidly. This is based on a comparison of conventional binary tree search methods with the compressed-level Huffman decoding method [4]. By using other encryption methods, a research has also been conducted which aims to improve SMS security using RSA cryptography. SMS is applied as a media for complaints for the public in making complaints against fraud in the regional head election. From the Avalanche Effect test results obtained an average of 10.44%. While from the brute force testing with two digit private key 32,768 brute force trials can be performed with a key search time of 3.7 miliseconds per key [5].

METHOD

In making a message security application with cryptographic algorithm AES (Advanced Encryption Standard) and the Huffman algorithm, the prototype model is used, which is a looping system development model that provides a systematic and structured approach in developing a system. The following are the stages in the prototype model that the author uses in building applications.

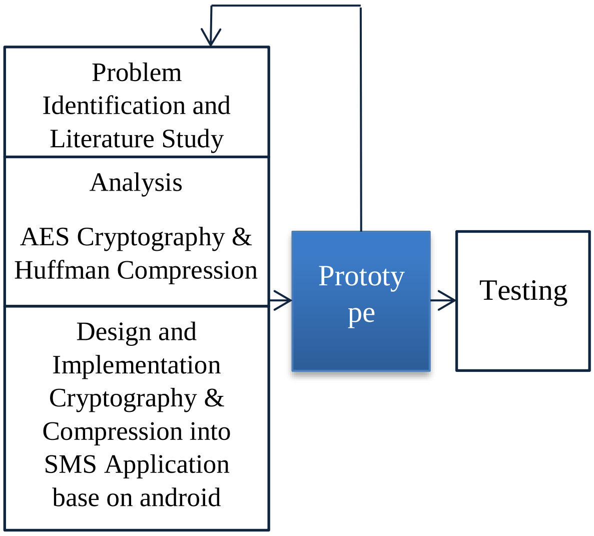

Problem Identification and Literature Study
Analysis
AES Cryptography & Huffman Compression
Design and Implementation Cryptography & Compression into SMS Application base on android
Prototy pe
Testing

Figure 2. Research State

Huffman Algorithm

Huffman algorithm is a simple compression algorithm compiled by David Huffman in 1952 [6]. This algorithm included in the type of lossless data compression, which is data compression that does not eliminate or change the number of bytes and stored according to the original data [13].

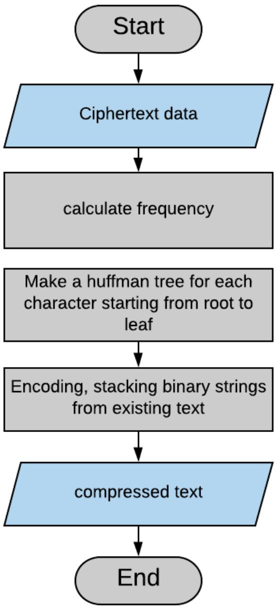

Start
Ciphertext data
calculate frequency
Make a huffman tree for each character starting from root to leaf
Encoding, stacking binary strings from existing text
compressed text
End

Figure 3. Huffman Compression Flowchart [7]

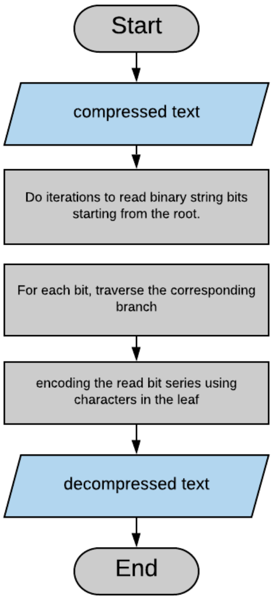

Start
compressed text
Do iterations to read binary string bits starting from the root.
For each bit, traverse the corresponding branch
encoding the read bit series using characters in the leaf
decompressed text
End

Figure 4. Huffman Decompression Flowchart [7]

AES Algorithm

AES algorithm that was socialized by the National Institute of Standards and Technology (NIST) in November 2001 created as a new encryption standard that developed from the DES (Data Encryption Standard) algorithm through several stages of comparison with other algorithms. Vincent Rijmen coined this algorithm, and Joan Daemen became the winners of the new algorithm replacement contest replacement DES [9]. AES algorithm uses substitution, permutation, and the number of cycles applied to each block to be encrypted and decrypted. For each spin, AES uses a different key. The key to each round is called the round key. AES works in the form of bytes or characters, the block size for the AES algorithm is 128 bits (16 bytes).

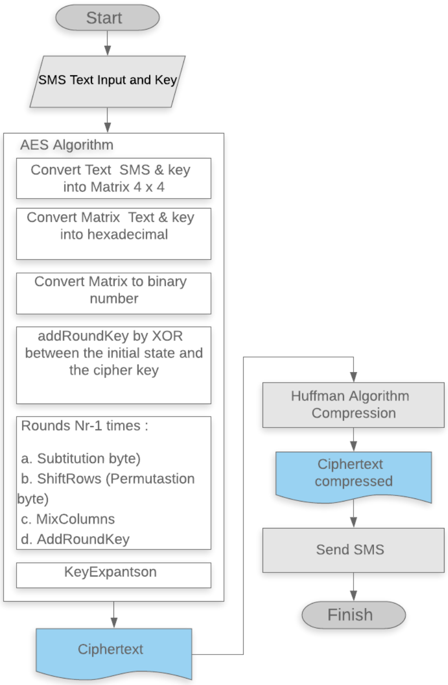

Start
SMS Text Input and Key
AES Algorithm
Convert Text SMS & key into Matrix 4 x 4
Convert Matrix Text & key into hexadecimal
Convert Matrix to binary number
addRoundKey by XOR between the initial state and the cipher key
Rounds Nr-1 times :
a. Subtitution byte)
b. ShiftRows (Permutastion byte)
c. MixColumns
d. AddRoundKey
KeyExpantson
Ciphertext
Huffman Algorithm Compression
Ciphertext compressed
Send SMS
Finish

Figure 5 AES Encryption Process Flow Chart [17]

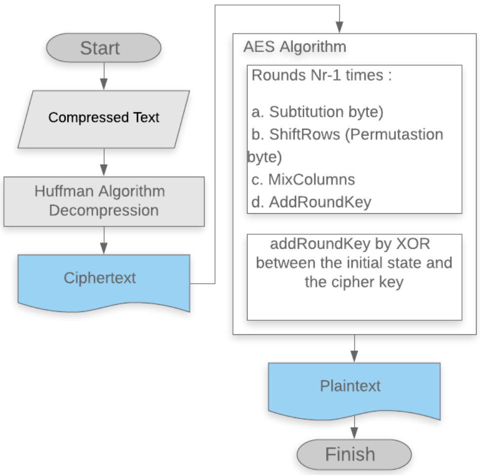

Start
Compressed Text
Huffman Algorithm Decompression
Ciphertext
AES Algorithm
Rounds Nr-1 times :
a. Subtitution byte)
b. ShiftRows (Permutastion byte)
c. MixColumns
d. AddRoundKey
addRoundKey by XOR between the initial state and the cipher key
Plaintext
Finish

Figure 6 AES Decryption Process Flow Chart [17]

RESULT

This process discusses the application of AES cryptography with Huffman compression in the SMS messaging system.

AES Encryption Process

AddRound*Key*

a. Perform XOR logic or initial round between the initial plaintext / state and the cipher key.
b. Round of Nr-1 times. The process carried out in each series are:
    1) SubBytes or substitution bytes using the substitution table (SBox).
    2) ShiftRows or through permutations of data bytes from different array columns.
    3) MixColumns or randomize data in each state array column.
    4) AddRoundKey or perform XOR operations between data and keys.
c. Final Round is the process for the last round of SubBytes, ShiftRows, and AddRoundKey
d. Key Expansion
   The AES algorithm executes a primary key and creates a key expansion to produce a key schedule. The key represented as a word (w [i]). Similar to the state, but the state element is the cipher key. The key expansion required is AES Nb (Nr + 1) word, so that in AES-128 requires 4 (10 + 1) word = 44 words. Some steps that are carried out to create a key schedule are rotword (), subWord, and Rcon (). Rotword () is if w [i] represented by an array of rows or columns into rows (transpose), then it can be described by sliding once to the left in the array position as shift rows () does in the second row. For example w [i] = (a0, a1, a2, a3), we get Rotword (w [i]) = (a1, a2, a3, a0). subword (), which

substitutes every byte converted to hexadecimal form with S-box table as well as what SubBytes () does. e.g. w [i] = CF4F3C09, by substituting into the S-Box table get Subword (w [i]) = 8A84EB01, where CF becomes 8A, 4F becomes 84, 3C becomes EB, and 09 becomes 01. Rcon [i] is a fixed component (Constanta) word of the round during the calculation process of expansion into the key schedule. The value on the AES-128 that uses ten times the rotation.

```
 public static byte[] encryptSMS(String secret key, String message){
  try {
   byte[] returnArray;
   Key key = generateKey(secretkey);


    private static Key generateKey(String secretkey) throws
    Exception {
            Key key = new SecretKeySpec(secretkey.getBytes(),"AES");
            return key;
   }
   Cipher c = Cipher.getInstance("AES");
   returnArray = c.doFinal(message.getBytes());
   return returnArray;
 } catch (Exception e) { e.printStackTrace(); byte[]  returnArray = null; return
  returnArray;
 }
}
```

Huffman Compression Proccess

a. Count the Frequency

Tree is a concept that describes a directed graph that is connected and does not contain trajectories

.

```
int[] Frequency = newint[257];
Arrays.fill(Frequency, 1);
FrequencyTable  freqtable = new FrequencyTable(Frequency);
HuffmanEncoder enc_cipher = new HuffmanEncoder(out);
enc_cipher.codeTree = freqtable.buildCodeTree();
int x = 0;
while (true) {
          int c = in.read();
          if (c == -1)
          break;
          enc_cipher.write(c);

          freqtable.increment(c);
          x++;
          if(x < 262144 &&isPowerOf2(i)||x % 262144 == 0)
            enc_cipher.codeTree =freqtable.buildCodeTree();
          if (x % 262144 == 0)
           freqtable = new FrequencyTable(frequency);
  }
enc_c.write(256);
```

b. Make a Huffman Tree for each character from root to leaf to 1 Huffman Tree The Huffman code is a prefix code that consists of a collection of binary codes and has the characteristic that no binary code is the starting point for other binary codes.

The prefix code draws as a binary tree that contains values or labels. On the left branch, the binary tree is labeled 0 while on the right branch, and it is labeled 1. The series of bits formed at each line from root to leaf is a prefix code for the paired characters to produce a Huffman tree. Proses encoding

```
public CodeTree buildCodeTree() {
  Queue<NodeWF> pqueue = new  PriorityQueue< NodeWF >();
  for (int i=0; i< freq.length;i++) {
    if (freq[i] > 0)
     pqueue.add(new NodeWF (new  Leaf(i),i, freq [i]));
    }
  for(int i = 0; i< freq.length&&pqueue.size()< 2; i++){
    if (i >= freq.length || freq [i] == 0)
                pqueue.add(new NodeWF (new Leaf(i), i, 0));
  }
  if (pqueue.size() < 2)
  thrownew AssertionError();
  while (pqueue.size() > 1) {
                NodeWF nf_1 = pqueue.remove();
                NodeWF nf_2 = pqueue.remove();
                pqueue.add(new NodeWF (
                newInternalNode(nf_1.node, nf_2.node),
                Math.min(nf_1.lowestSymbol, nf_2.lowestSymbol),
                nf_1. freq + nf_2. freq));
  }
  Returnnew CodeTree((InternalNode)pqueue.remove().node, freq.length);
}
```

Huffman Decompression Process

It is known that the code for a symbol / character cannot be the beginning of another symbol code to avoid ambiguity in the decompression process. Because each Huffman code generated is unique, the decoding process can be done easily.

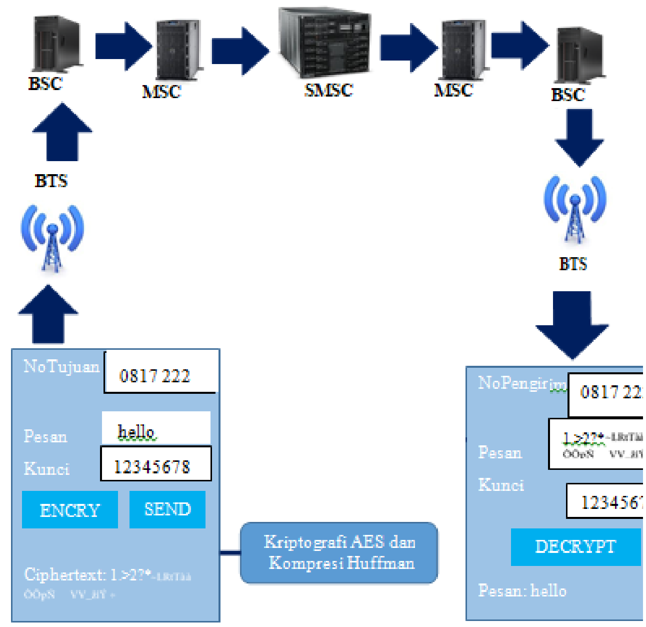
a. Do iterations to read binary string bits starting from the root.
b. For each bit traverse the corresponding branch
c. Code a series of bits that have been read with characters in the leaf.

```
int[]Frequency = new int[257];

Arrays.fill(Frequency, 1);

FrequencyTable freqtable = new FrequencyTable(Frequency);

HuffmanDecoder dec = new HuffmanDecoder(in);

dec.codeTree = freqtable.buildCodeTree();

int count = 0;

while (true) {

  int symbol = dec.read();

  if (symbol == 256)  // EOF symbol

  break;

  out.write(symbol);


  freqTable.increment(symbol);

  count++;

  if (count < 262144 && isPowerOf2(count) || count % 262144 == 0)  // Update code tree

  dec.codeTree = freqtable.buildCodeTree();

  if (count % 262144 == 0)  // Reset frequency table

  freqtable = new FrequencyTable(Frequency);

}
```

AES Decryption Process

```
public static byte[] decryptSMS(String KunciRahasiaString, byte[] PesanTerenkripsi)throws Exception {

  generateKey(KunciRahasiaString);


    private static Key generateKey(String KunciRahasiaString) throws

    Exception { Key key = new SecretKeySpec(KunciRahasiaString.getBytes(), "AES");

  return key;

  }


  Cipher c = Cipher.getInstance("AES");

  c.init(Cipher.DECRYPT_MODE, key);

  byte[] decValue = c.doFinal(PesanTerenkripsi);

  return decValue;

}
```

3.5 Implementation



NoTujuan 0817 222
Pesan hello
Kunci 12345678
Ciphertext: 1.>2?*-LRrTää ÕÕpN VV_äîY »

NoPengirim 0817 22
Pesan 1.>2?*-LRrTä ÕÕpN VV_äîY
Kunci 123456
Pesan: hello

Figure 7. System Design

Main Menu AES Application

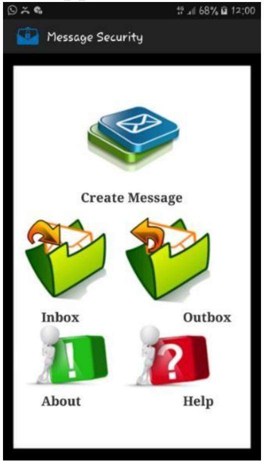


Figure 8. *Main Menu*

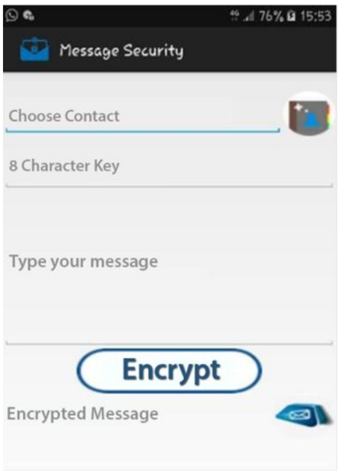
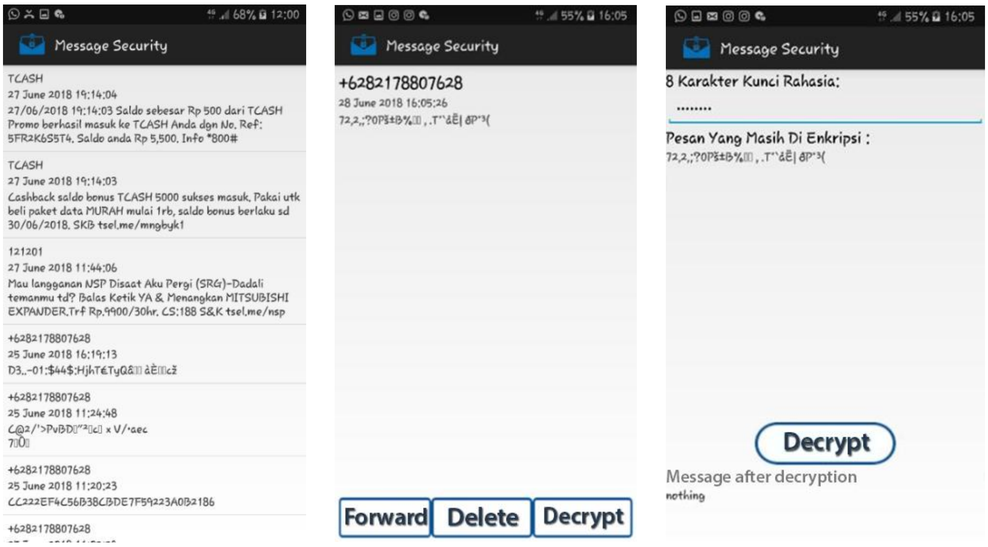


Figure 9. Create Message Form



TCASH
27 June 2018 19:14:04
27/06/2018 19:14:03 Saldo sebesar Rp 500 dari TCASH Promo berhasil masuk ke TCASH Anda dgn No. Ref: 5FR2K6S5T4. Saldo anda Rp 5,500. Info *800#

TCASH
27 June 2018 19:14:03
Cashback saldo bonus TCASH 5000 sukses masuk. Pakai utk beli paket data MURAH mulai 1rb, saldo bonus berlaku sd 30/06/2018. SKB tsel.me/mngbyk1

121201
27 June 2018 11:44:06
Mau langganan NSP Disaat Aku Pergi (SRG)-Dadali temanmu td? Balas Ketik YA & Menangkan MITSUBISHI EXPANDER.Trf Rp.9900/30hr. CS:188 S&K tsel.me/nsp

+6282178807628
25 June 2018 16:19:13
D3..-01:44:HjhT€TyQâ àÈcž

+6282178807628
25 June 2018 11:24:48
C@2/'>PvBD"²c x V/·aec
7Ò

+6282178807628
25 June 2018 11:20:23
CC222EF4C56B38CBDE7F59223A0B2186

+6282178807628

+6282178807628
28 June 2018 16:05:26
72,2,;?0P$±B%, .T"`áÈ| ðP"³(

8 Karakter Kunci Rahasia:
....

Pesan Yang Masih Di Enkripsi :
72,2,;?0P$±B%, .T"`áÈ| ðP"³(

Message after decryption
nothing

Figure 10. Inbox and Message Decryption Form

TESTING

At this stage, measurement of system performance and functional testing.

Table 1. Encryption and Compression Results

Character	AES algorithm without compression	AES Algorithm with Huffman Compression
H	B114E95845481ED3BF7C187 D2D02D93B	@-,^|bV`H"Ÿ ?!. ^{1}SŠ -€
halo, apa kabar?	608A8BB7C4A5548CCD0F6 271AF32586108C81B39D5F2 366FA2C3AFDF047A5E2A	4,4=0xxZjNl ÙÑÃæè2/ ìHñ®î_á ç¿ü Yœ_ñ9ç·þ%p
apa yang kamu lakukan akhir pekan ini?	DC1FFA08711128A5D5E D3613BBD8CF9D9A803B DA7E6957D24E57AEE750 775BCD9F50AD00180252 487B7E0924F03BBC5D	B?+B@pL^RD BŠË.,4=ĐÑ Œ0 øsŒ ° äœÁ7 p7pãÏ_yóÂŒùBg;1Ýå3››{Ç[ŒŸùô¶BYÖà
Kalo kamu tidak ada aktivitas pada waktu akhir pekan, maukah kamu mengikuti kontes lari?	3D7567EB9DFCF2B62A2816 1C89026CB2B076F0CF859D 91906F632F7BC1D49BB12E 3E8C4F71264595DD044A398 EB39D68442708AB6C195B2 2294019CB0F485F6A50A588 D2ED2437DE0636C1FA8133 8CD3EF3A9C43330C929956 A9573A6FE2	1A20.0zvVpÜØÜÑA¥cGÆz .D`oRÝõ í^{1} |8ÓÀ Üö ¯=Â§ ^{3},W±uÂ§KlËŸùBêúz}/gx_¯gw }oôôýåm}}>ï3¿öYeMvgÕéÓÖú Bä'^{1}Æ2c ;.2!ò:õ%x]j±Ve>>JC®°n3ÏÁ;¥ w x$\

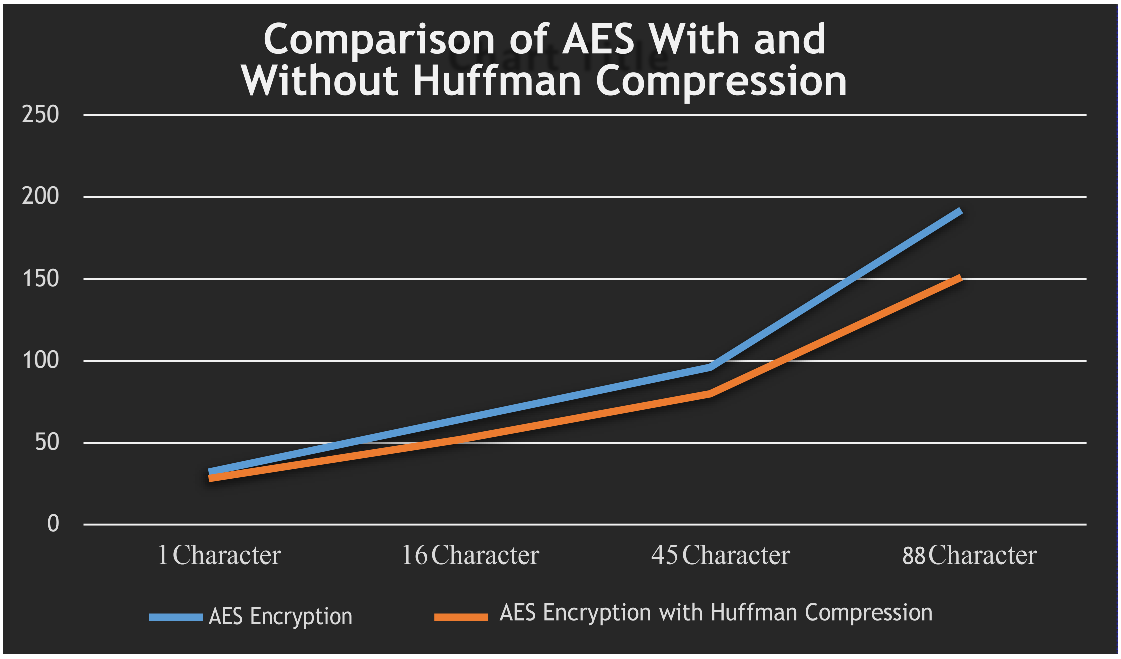


Figure 11. Graph Comparison of the Number of Character AES Encryption and AES Encryption with Huffman Compression

Table 2. *Presentation Compression Effisiency*

Number of Characters	Encryption	Encryption with Compression	percentage
1	32	28	12,5%
16	64	52	18,8%
45	96	80	16.7%
88	192	151	21,4%
			17,35%

SUMMARY

The conclusion of the AES algorithm research is, that the text encoding in the Android-based SMS application has increased the amount of text. Resulting in an increased cost of sending messages, but with the addition of Huffman compression so that the lack of AES algorithm can be covered and can be accepted by the public because it is easier to operate and cheaper. Encryption test results are: AES algorithm cryptography can be applied to secure SMS and get unreadable ciphertext results, and the level of compression efficiency of Huffman gets 17.35% more efficient than without compression.

REFERENCES

[1] T. Mantoro, Laurentinus, N. Agani, and M. A. Ayu, “Improving the security guarantees, authenticity and confidentiality in short message service of mobile applications,” in *Proceedings of 2016 4th International Conference on Cyber and IT Service Management, CITSM 2016*, 2016.

[2] Liang, Haihua & Zhang, Xinpeng & Cheng, Hang, “Huffman-code based retrieval for encrypted JPEG images”, Journal of Visual Communication and Image Representation. 61. 10.1016/j.jvcir.2019.03.021, 2019.

[3] Yuan, Shuyun & Hu, Jianbo,” Research on image compression technology based on Huffman coding”. Journal of Visual Communication and Image Representation. 59. 10.1016/j.jvcir.2018.12.043, 2018.

[4] Lin, Yih-Kai & Huang, Shu-Chien & Yang, Cheng-Hsing., “A fast algorithm for Huffman decoding based on a recursion Huffman tree”, Journal of Systems and Software. 85. 974–980. 10.1016/j.jss.2011.11.1019, 2012.

[5] D. Y. Sylfania, F. P. Juniawan, L. Laurentinus, and H. A. Pradana, “SMS Security Improvement using RSA in Complaints Application on Regional Head Election’s Fraud”, *Jurnal Teknologi dan Sistem Komputer*, vol. 7, no. 3, pp. 116-120, Jul. 2019. https://doi.org/10.14710/jtsiskom.7.3.2019.116-120

[6] Q. Zhou, K. W. Wong, X. Liao, and Y. Hu, “On the security of multiple Huffman table based encryption,” *J. Vis. Commun. Image Represent.*, 2011.

[7] M. Ramakrishnan and R. Sujatha, “Cf-Huffman code based hybrid signcryption technique for secure data transmission in medical sensor network,” *Int. J. Appl. Eng. Res.*, 2015.

[8] N. N. Mohamed, H. Hashim, Y. M. Yussoff, M. A. M. Isa, and S. F. S. Adnan, “Compression and encryption technique on securing TFTP packet,” in *ISCAIE 2014 - 2014 IEEE Symposium on Computer Applications and Industrial Electronics*, 2015.

[9] C. A. Sari, G. Ardiansyah, D. R. I. Moses Setiadi, and E. H. Rachmawanto, “An improved security and message capacity using AES and Huffman coding on image steganography,” *Telkomnika (Telecommunication Comput. Electron. Control.*, 2019.

[10] D. Kapoor Sarmah and N. Bajpai, “A new horizon in data security by Cryptography &; Steganography,” *Int. J. Comput. Sci. Inf. Technol.*, 2010.

[11] J. H. Pujar and L. M. Kadlaskar, "A new lossless method of image compression and decompression using Huffman coding techniques," *J. Theor. Appl. Inf. Technol.*, 2010.

[12] J. S. Vitter, “Design and Analysis of Dynamic Huffman Codes,” *J. ACM*, 1987.

[13] E. Satir and H. Isik, “A Huffman compression based text steganography method,” *Multimed. Tools Appl.*, 2014.

[14] R. Arshad, A. Saleem, and D. Khan, “Performance comparison of Huffman Coding and Double Huffman Coding,” in *2016 6th International Conference on Innovative Computing Technology, INTECH 2016*, 2017.

[15] Laurentinus, “Implementasi Kriptografi Dan Kompresi SSM Menggunakan Algoritma RC6 Dan Algoritma Huffman Berbasis Android,” *Jurnal Ilmiah Informatika Global*, 2017.

[16] E. Andreas, “Aplikasi Kriptografi File DOC, DOCX, JPG dan PDF dengan Metode AES dan Kompresi Huffman,” *Kriptografi AES Dengan Kompresi Huffman*, 2014.

[17] D. Darwis, R. Prabowo, and N. Hotimah, “Kombinasi Gifshuffle, Enkripsi AES dan Kompresi Data Huffman untuk Meningkatkan Keamanan Data,” *J. Teknol. Inf. dan Ilmu Komput.*, 2018.