Journal Majelis Paspama
Volume 01 Number 02 2023 (No 15-21)
DOI 10.
E.SSN 2987-1379 (Online)

Steganography With End Of File

Zuvernius Sitohang[1], Yohana Desma R. Munte[2], Pembinata Taringan[3], Hiskia Hasugian[4], Trihadi pinem[5]
Universitas Katolik Santo Thomas Medan dan I. Setia Budi No.479, Tj. Sari, Kec. Medan Selayang, Kota Medan, Sumatera Utara 1

Article Info
Corresponding Author:
Zuvernius Sitohang
E-mail:
zuvernius@gmail.com

ABSTRACT
STEGANOGRAPHY WITH END Steganography is the science and art of hiding secret messages in other messages, so that the existence of the message cannot be known by third parties. One method used in steganography is the EOF (End of File) message insertion method, which is inserting data at the end of a file by providing a special mark to identify the beginning and end of the data. This technique is used to insert data as needed, and the steganographed file does not appear to change visually or audibly, even though the file size becomes larger. In this regard, this technique can be applied to protect the copyright of digital products, such as software or multimedia, through digital watermarking techniques that use the EOF method to insert information or messages as permanent identification in digital files. In addition, in an effort to improve security, a technique for combining Blowfish cryptography with Hexa steganography is also proposed. This technique has advantages over DCS steganography, namely fewer pixel color changes, so that the inserted data is more difficult to detect and easier to maintain even if the stego image is processed or edited. The results of the study show that the combination of Blowfish cryptography with Hexa steganography is more effective than DCS steganography in protecting the inserted data. The test was carried out by analyzing the robustness of Hexa steganography using the PNSR test to determine the level of change in the original image and the inserted image.
Keywords: Steganography, Cryptography, EOF

INTRODUCTION

Crime and criminals today involve information and communication technology. The use of computers, mobile phones, email, the internet, and other digital devices can invite various parties to commit crimes based on communication technology. Therefore, techniques are needed to secure the sending of text messages so that people cannot tap or take text messages to commit crimes.

Cryptography can be the answer to this problem. As a science that has been applied to data security, cryptography can be used to secure important data in a file. The data contained in the file is encrypted or encrypted to be converted into certain symbols so that only certain people can know the contents of the data.(Rohmanu, 2017).

Steganography is a method to hide a message in another message where others do not know that the message has a more important message in it. The purpose of developing Steganography is to keep the message data inserted in digital image media from being detected by other applications that there is a secret text message in the digital image.

One of the techniques for inserting messages into images is the end of file method.(Yudharta Pasuruan, 2022)which uses the Least Significant bit (LSB) algorithm. LSB steganography is one of the easiest steganography methods to implement, namely by changing the least significant bit in a file. Changes to the Least Significant Bit (LSB) will only result in almost insignificant changes to the file. Implementation of the Least Significant Bit (LSB) technique and steganography techniques has been proven to provide advantages in providing good stego image quality and maintaining imperceptibility.(Mido and friends, 2022).

METHOD

The general design flow of the Steganography method with the End Of File method can be seen in Figure 1.

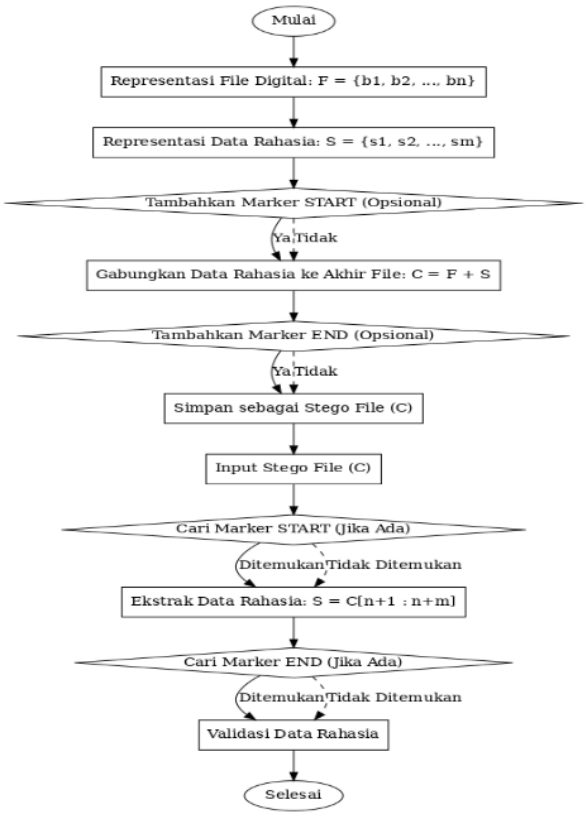

Figure 1 Insertion flow with EOF method

Explanation of Figure 1 Insertion Flow with EOF Method

1. Sender Initiates Process: The sender initiates the steganography process to hide the secret data in a cover file.
2. Confidential File and Data Representation: The sender has a digital file that will be used as a data hiding place Cover File (F), The sender has confidential data that he wants to hide in the cover file. Confidential Data (D)
3. Embedding Process: Add START Marker (Optional): The sender can add START marker before the secret data if needed. Merge Secret Data: The sender merges the secret data (S) into the cover file (F) using the merge operation C = F + S. Add END Marker (Optional): The sender can also add END marker to indicate the end of the secret data. Save Stego File: The sender saves the stego file (C) containing the cover file and secret data.

4. Sending Stego File to Recipient: The sender sends the Stego File (C) to the recipient via secure media such as email.
5. Recipient Receives Stego File The recipient receives a stego file (C) containing the secret data hidden in a cover file.
6. Extraction Process (Secret Data Retrieval) The receiver checks whether there is a START marker indicating the start of the secret data. The receiver extracts the secret data contained in the stego file (C) starting from byte n+1 to n+m. The receiver checks whether there is an END marker to ensure the end of the secret data. The receiver checks whether the extracted data is valid and as expected.
7. Receiver Gets Secret Data: After the extraction process is complete, the receiver gets the secret data hidden by the sender. The steganography process is complete with the secret data successfully hidden and retrieved.

A4 Size

All articles are written in 10pt Nunito font. The title of the article is written in 12pt Nunito font. The author's name is written in 10pt Nunito font. The author's affiliation is written in 8pt Nunito font. Corresponding authors must include an email address. The article title and abstract are written in one column (one column) and aligned left and right. The author's name is written without including a title.

RESULTS AND DISCUSSION

Ascii Byte Representation

Known data, Main File : "HELLO WORD"

Table 1. Data representation

Character	Byte (Ascii)
H	72
English	69
I	76
I	76
O	79
Space	32
We	87
O	79
R	82
I	76
D	68

Main file in Bytes F = 72, 69, 76, 76, 79, 32, 87, 79, 82, 76, 68
The main file length is 11 bytes. Secret Data : : "1234"

Table 2. Byte representation

Character	Byte (Ascii)
1	49
2	50
3	51
4	52

Secret data in Bytes S = {49, 50, 51, 52}. Secret data length (mmm): m=4

Embedding Process (Secret Data Insertion)

Basic formula: C = F + S

Description: Result file = Main file appended with secret data behind it. Merge Bytes from the main file and secret data:

Table 3. Merge Bytes from the public file and secret data

index	Byte (Ascii)	Character	Information
1	72	H	Main File
2	69	English	Main File
3	76	I	Main File
4	76	I	Main File
5	79	O	Main File
6	32	SPACE	Main File
7	87	We	Main File
8	79	O	Main File
9	82	R	Main File
10	76	I	Main File
11	68	D	Main File
12	49	1	Confidential Data
13	50	2	Confidential Data
14	51	3	Confidential Data
15	52	4	Confidential Data

Value Substitution:

Value = {72,69,76,76,79,32,87,79,82,76,68} + {49,50,51,52},

Result: C={72,69,76,76,79,32,87,79,82,76,68,49,50,51,52}

After insertion, the resulting file, C="HELLO WORLD1234". Where the main file remains the same and the secret data is added at the end of the file. The length of the resulting file: C=n+m=11+4=15 bytes.

Extraction Process (Secret Data Retrieval)

Table 4. File Length Identification

Data type	long
Main file (n)	11
Confidential Data (m)	4

The secret data is located from bytes (n+1) to (n+m)

Table 5. Extract bytes from the result file

calculation	Results
Initial index (n+1)	11 + 1 = 12
Final index (n+m)	11 + 4 = 15

table 6.extract bytes from index 12 to 15

Index (position in file)	Byte (ascii)	Character
12	49	1
13	50	2
14	51	3
15	52	4

Extracted Result Bytes: S = "1234"

S value={49,50,51,52}

Table 7. Convert bytes to text

Byte (ascii)	character
49	1
50	2
51	3
52	4

The text we get is: "1234". The final result is the original text entered earlier: "1234".

Verify Results

1. Main File: Bytes: 72, 69, 76, 76, 79, 32, 87, 79, 82, 76, 68→ "HELLO WORLD"
2. Secret Data:Bytes: 49, 50, 51, 52→ "1234"
3. Result File :Bytes: 72, 69, 76, 76, 79, 32, 87, 79, 82, 76, 68, 49, 50, 51, 52→ "HELLO WORLD1234"

Extract Secret Data: Bytes: 49, 50, 51, 52 → "1234"

Testing with Python

Insert the main file and secret data. The first step is the process of inserting secret data. We insert the main file and the secret message to be inserted. Here, all characters need to be converted to ASCII or binary code to simplify the calculation process, because the insertion process is based on the ASCII code of the characters we entered earlier. This is the main file converted to ASCII.

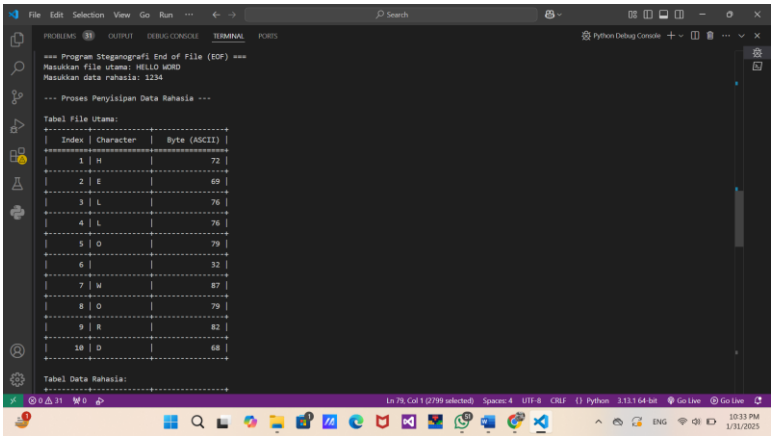

Figure 1. converted to ASCII

Secret Data Table containing characters to be converted to ASCII. The secret data must also be converted to ASCII or binary. Later, the converted ASCII codes will be added up, or in other words, the calculation process will be carried out using the existing formula, namely by adding the ASCII code of the main file with the ASCII code of the secret message.

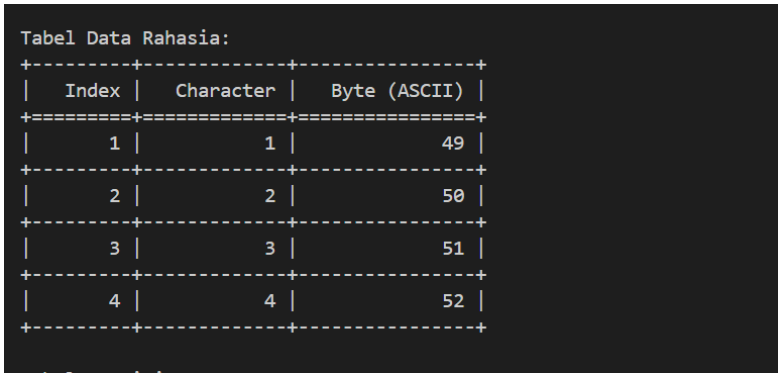

Figure 2. Add the ASCII code of the main file with the ASCII code of the secret message.

The Embedding process is presented in this table, where all the processed calculation results are displayed. This is an addition process where the main file will be embedded with the secret message. However, we need to provide an indication whether each character belongs to the main file or the secret message to facilitate the extraction process. In this process, the main file appears before the secret data.

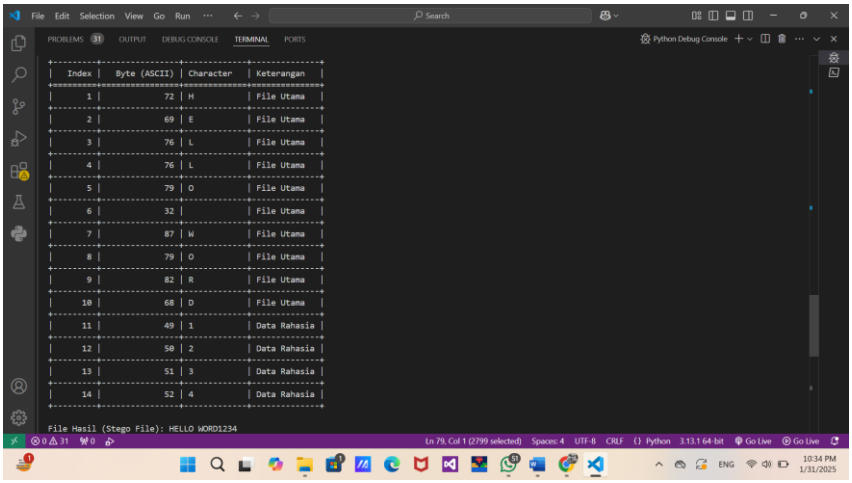

Figure 3. calculation results

Finally, the embedded message is extracted again. The final process is extraction, where the message we inserted in the previous step is extracted from the file. This process is the reverse of the insertion process—while insertion involves addition, the extraction process involves subtraction.

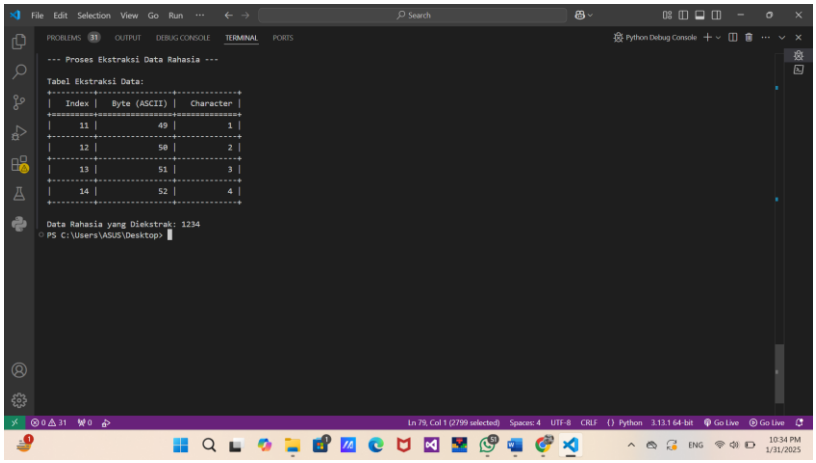

Figure 4. Re-extracted

CONCLUSION

The EOF method is more suitable for short files. This is because: Computational Complexity: The process of adding and subtracting ASCII codes for each character can be very complicated and time-consuming when applied to large files. Risk of Error: The more data, the higher the risk of errors during the insertion and extraction process, especially if there are no clear markers. File Size: Adding ASCII data can increase the file size, which may be inefficient for large files. However, if this method is to be applied to large files, optimization is needed, such as using a more efficient algorithm or special markers (such as EOF) to simplify the extraction process.

REFERENCE

Ari Anti, U., Harsa Kridalaksana, A., & Marisa Khairina, D. (2017). *VIDEO STEGANOGRAPHY USING LEAST SIGNIFICANT BIT (LSB) AND END OF FILE (EOF) METHODS*. *12*(2).

Cahyono, A. D., Yasin, M., & Malang, U. N. (2019). *Steganography implementation using end of file (EOF) method in data security (Case study on AVI, MP3, and JPEG files)*.

Edisuryana, M., Isnanto, R. R., & Somantri, M. (n.d.). *STEGANOGRAPHY APPLICATION ON BITMAP FORMAT IMAGE USING END OF FILE METHOD*.

Fauzi, A. (2019). ANALYSIS OF TEXT MESSAGE COMBINATION INTO AUDIO FILES UTILIZING DATA ENCRYPTION STANDARD ALGORITHM AND END OF FILE METHOD. *Journal of Kaputama Informatics Engineering (JTIK)*, *3*(1).

Kusumaningsih, D., Pudoli, A., & Rahmadan, I. (2017). *IMPLEMENTATION OF RIJNDAEL ALGORITMA CRYPTOGRAPHY AND END OF FILE METHOD STEGANOGRAPHY FOR DATA SECURITY* (Vol. 9, Issue 1).

Mido, A. R., Iman, E., & Ujianto, H. (2022). *ANALYSIS OF IMAGE EFFECT ON THE COMBINATION OF RSA CRYPTOGRAPHY AND LSB STEGANOGRAPHY*. *9*(2), 279-286. https://doi.org/10.25126/jtiik.202294852

Rohmanu, A. (2017). CRYPTOGRAPHY AND STEGANOGRAPHY IMPLEMENTATION WITH DES ALGORITHM METHOD AND END OF FILE METHOD. *SIMANTIK Informatics Journal*, *2*(1). www.jurnal.stmikcikarang.ac.id

Yudharta Pasuruan, U. (2022). *Application of the End Of File Steganography Method to Insert Messages in Digital Images Rahmad Zainul Abidin.* http://sipi.usc.edu/database/database.php

Darwis, D., & KISWORO, K. (2017). Teknik Steganografi untuk Penyembunyian Pesan Teks Menggunakan Algoritma End Of File. *Explore: Jurnal Sistem Informasi dan Telematika*, *8*(2), 331304.

Minarni, M., Ikram, A., Warman, I., & Swara, G. Y. (2023). Implementasi Algoritma Vigenere Cipher Dan End Of File Pada Steganografi Video. *Jurnal Minfo Polgan*, *12*(1), 432-441.

Edisuryana, M., Isnanto, R. R., & Somantri, M. (2013). Aplikasi Steganografi Pada Citra Berformat Bitmap Dengan Menggunakan Metode End Of File. *Transient: Jurnal Ilmiah Teknik Elektro*, *2*(3), 734-742.