JURNAL RISET INFORMATIKA **P-ISSN: 2656-1743 |E-ISSN: 2656-1735**
Vol. 2, No. 4 September 2020 DOI: https://doi.org/10.34288/jri.v2i4.153

INSERTING MESSAGE SECRET ON FILE DATA BANK USING STEGANOGRAPHY ENGINEERING WITH EOF (END OF FILE) METHOD

Michael Sitorus[1], Deki Satria[2]

Department of Information System[1], Departement of System and Information Technology[2]
Institute of Technology and Business Bank Rakyat Indonesia (BRI Institute)[1,2]
michael.sitorus@bri-institute.ac.id[1], deki.satria@bri-institute.ac.id[2]

Abstrak

Dalam pertukaran informasi tidak disadari ada informasi yang penting. Sering beredar informasi yang penting tapi dianggap tidak penting seperti informasi rekening nasabah bank. Ancaman keamanan terhadap informasi cukup banyak seperti interruption, interception, modifikasi, dan fabrication. Banyak pencurian data informasi oleh pihak yang tidak bertanggungjawab. Penelitian ini bertujuan untuk mengamankan data informasi berupa file citra digital yang akan disisipkan pesan rahasia dengan membangun aplikasi untuk sistem keamanan pesan rahasia di Bank dengan menerapkan steganografi menggunakan metode End Of File (EOF). Aplikasi Steganografi yang mampu menyisipkan pesan rahasia pada data file citra digital. Hasil penelitian ini adalah mampu merahasiakan keberadaan dari sebuah pesan tersembunyi. Ketika orang lain yang menerima file gambar, mereka hanya dapat melihat gambar saja secara kasat mata. Jadi semua pesan tidak terlihat dan tidak akan dapat terbaca oleh hacker atau orang lain sebelum mengetahui sisi keamanannya dan password untuk membuka atau membaca pesan rahasianya.

Kata kunci: Steganografi, End Of File (EOF), Citra Digital, Keamanan, Bank, Hacker

Abstract

Exchanging information does not realize that is important information. Circulates often important information insignificant but such as account bank customer information. The threat of security to information such as interruption, interception, modification, and fabrication. Thefts of information data by irresponsible parties. This research aims to secure information data in the form file digital image to be inserted with a secret message on building application for a secret message security system in the Bank with applying steganography using the End Of File (EOF) method. Steganography application capable of inserting secret messages in digital image data files. The results of the research can keep the existence of the message hidden. When other people receive image files, they can only see the images with the naked eye. So all messages are invisible and will not be read for hackers or other people before knowing the security side and open of password or reading the secret message.

Keywords: Steganography, End Of File (EOF), Digital Image, Security, Bank, Hacker

INTRODUCTION

A lot of people do not know that data theft is a threat and easy to do. Therefore peoples tend to send their data unsafely through the internet. Data shared on the internet, usually public, can be accessed anywhere, anytime, and in any size without looking at its importance.

Bank, as a business entity, should follow the advancement of technology. Therefore data integration became one of the primary concerns in their daily activity. The data need to be integrated, such as client data, business data, and other critical data. These vital data can be in picture format (Sa'adah & Purqon, 2016). There was a lot of secrets data that need to be encrypted or hide. The business could use steganography as a way to obscure private data.

Steganography is an art and science about hiding messages into a media such as pictures and sounds. Therefore no one knows about the data hidden in the media except the sender and receiver (Nurmaesah et al., 2018)(Sitorus, 2015a).

There were two crucial things needed to implement steganography. They are the message and the media itself. Steganography is used to obscure the private data and protect the data (Sitorus, 2015b). The media used to obscure the data won't be changed; therefore, the media will look the same before and after the process (Ariyus, 2009).

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

RESEARCH METHOD

This research using steganography. We can identify excess bit using this method implicitly by forecasting or explicitly using a calculation. We can use this excesses bit to hide our private message in the file.
There were four Steganography Algorithm widely used, which are: 1) Least Significant Bit (LSB); 2) End of File (EOF); 3) Domain Transformation; and 4) Spread Spectrum Encoding.

Previous research in this area conducted using the EOF method and Caesar chipper to encrypt data into the file. The application developed based on how substitution work on the Caesar Chipper method to conduct the encryption and EOF to add the message in the end bit of the file (Indriyono, 2016).

Before EOF come to light, the LSB Algorithm was widely used by the researcher. But the main downside of LSB is when the private/hidden message was larger than the container file. This problem made the steganography impossible to conduct. In EOF methods, the hidden message got special marking at the end of the file as an identifier (Anggraini & Sakti, 2014). We can use this algorithm or technique to hide messages without changing the file size (Masri et al., 2019).

Research Time and Place

This research was conducted in BRI Pasar Minggu and BRI Institute in August 2020.

Research Population and Samples

The population of this research is the staff of BRI Pasar Minggu and the BRI Institute. The details of Population and Sample is:

a. The population is all the staff of BRI Pasar Minggu and BRI Institute. The total staff is ten people.
b. The sample is part of the population we consider to be able to represent the population. The sample was chosen using random sampling. In this research, we used five files with a different extension.

Research Plan

This research tried to develop an application that will add a hidden or private message into an image file. This research is a randomized pretest-posttest control group design. We use SDLC Waterfall as Development Methods. The Waterfall model steps can be seen in Figure 1.

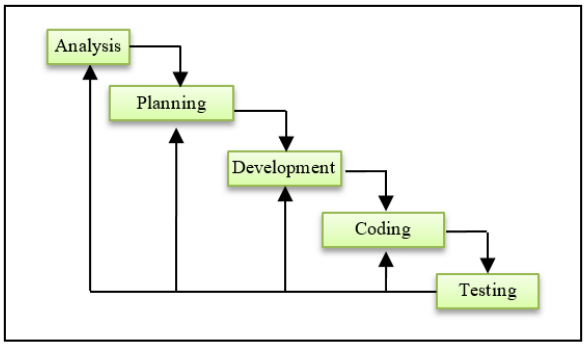


Figure 1. *Waterfall Model*

1) Analysis
   In this step, the user requirement analysis step we conducted. This step was important because we can find what the user needs is. The problem we tried to solve in this research was how to add a hidden message in the image file using JPG, GIF, and BMP extension. EOF Method was used to solve this problem.
2) Plan
   In this step, we tried to solve the problems found in the analysis step. In this step, we made the development model using UML. The models are based on the result of requirement analysis.
3) Model Development
   In this step, we will create the UI mockup of the system based on the analysis result.
4) Coding
   The coding process can be seen as a translation from human language to computer language. This is the main process of software development, where the programmer implements the analysis result in the application.
5) Testing
   Test steps were conducted to check whether the implementation met the user expectation. This step was also conducted to make sure there were no errors in the coding.

Data Gathering Methods

We use qualitative methods to gather data, such as interview, observation Focus Group Discussion (FGD), and Document

RESULT AND DISCUSSION

A. Analysis

We can use steganography to hide data and enhance the hidden capability of the message (Anindyawati & Suryani, 2012). From the analysis result, we found that EOF is the best method in this case. EOF won’t increase the file size event when we add hidden text messages in the fie. This EOF methods add a special identifier at the end of the

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v2i4.153

image file as a data header in the file and Flag at the end of the file.

B. Planning

System planning needs to have steps that focused on design programs or software, including data structure, software architecture, interface, and code procedure (Nasution et al., 2017). We used UML to design the software model. We use three UML models, which are Use Case Diagram, Activity Diagram, and Sequence Diagram.

1. *Use Case Diagram*

The use case diagram represents how the user will use the system as an actor. The use case of this system can be seen in Figure 2.

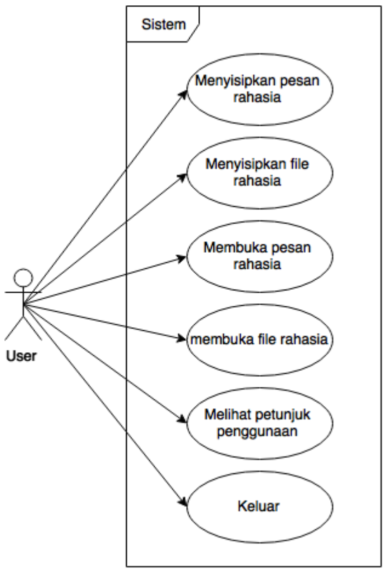


Figure 2. *Use Case Diagram*

In figure 2, users login to the main page to use the application as intended by the user. On the main page, the user can hide the message to the image file, opening the hidden message in the image file, reading the user guide, and log out from the system.

2. *Activity Diagram*

Activity diagram show activity in the system in the form of action set, how each action is executed, decision, and how the action end (Suendri, 2018). Activity diagram show workflow or activity or business process of the system (Hendini, 2016). Activity Diagram eases the analysis step to determine the activity that needs to be executed. The activity diagram of this application can be seen in figure 3. The activity starts with choosing the image that will become the main file. The next step is to choose "stego image" and insert the password for the hidden message; next, the user inserts the hidden message and then clicking insert to combine the image file and message.

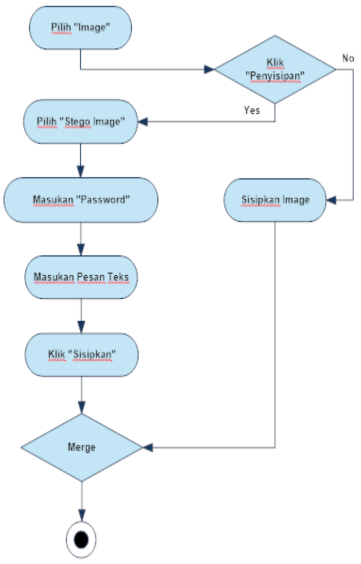


Figure 3. *Activity Diagram* Text Message Insertion

Figure 4 shows that the insertion of the hidden message starts from click "*Penyisipan*". After that, the user clicks "Stego Image" and enter the password for the message. Stego image is a process to insert a message into the image file. "*Sisipkan"* use to execute image and data file combinations.

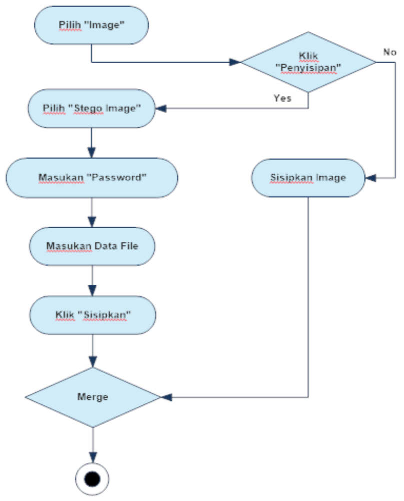


Figure 4. *Activity Diagram* Data File Insertion

3. *Sequence Diagram*

Sequence Diagram is a popular tool in software development, especially in the OOP (Object Oriented Programming) approach. The analyst used this diagram to show how objects interacted in the system in some timeline (Heriyanto, 2018),(Novita & Sari, 2015).

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

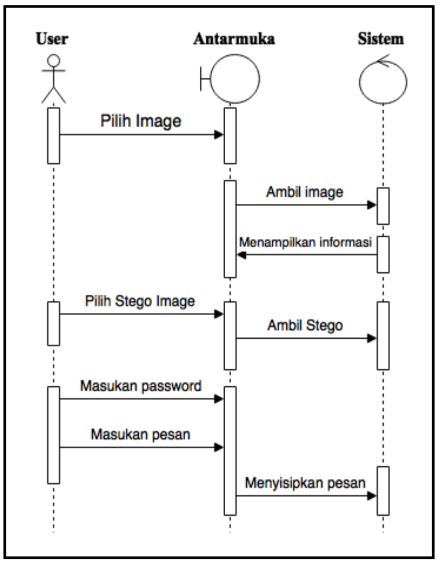


Figure 5. *Sequence Diagram* Text Message Insertion

Figure 5 explains text insertion, where the process started from choosing the main image file as the message container. There will be informed if the file has been used as a container from another steganography process. The next step user will choose "Stego Image" to insert the message and password.

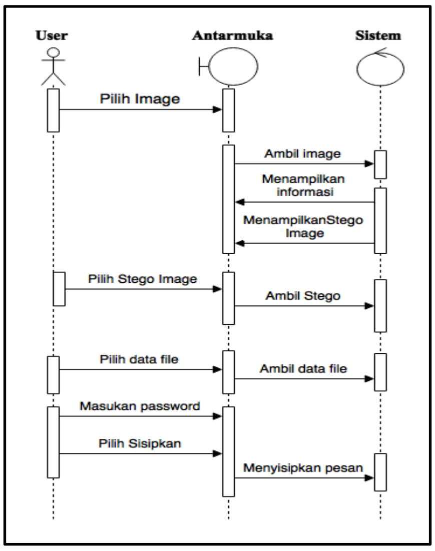


Figure 6. *Sequence Diagram* Data File Insertion

Figure 6 shows data file insertion where the process starts from choosing the image as a message container. This step will also check if the container image has been used as a container before, and the message will be added.

C. Development

The result of this research was an application that could be used to add a message to a file. The UI of these apps is divided into five main menus with a specific function. The UI can be seen in figure 7.

1. Main page: this page will show all the functionality of the system.
2. Insert: in this menu, there was a button to insert the message into the image file. The hidden message in text form. Insert button to choose hidden message file in docs extension (doc, Docx, pdf, Xls)
3. Extraction: in this menu, there was an extraction button to extract hidden text messages and file extraction to open secret documents.
4. User Guide: Users will use this menu when they need the user guide of the application.
5. Profile: in this menu, the user can see the information about the steganography and author button to show information about the author.

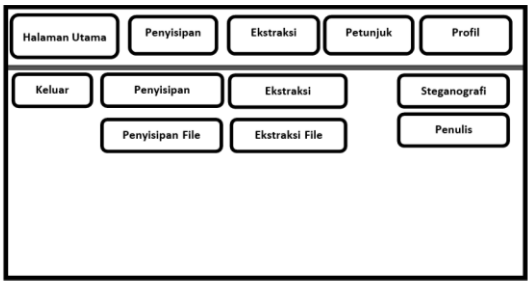


Figure 7. Home page design

Figure 8 shows the UI design for text message insertion. On this page, there were Master Image Text Book and Stego Image, which used to display data in folders. Message Text Box used to write the message, and Password Text Box used to insert the password. The insertion process is executed if all text box filled correctly.

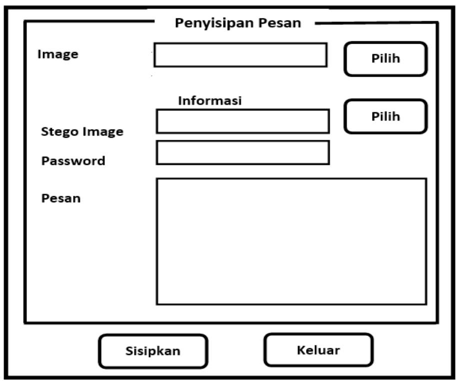


Figure 8.UI Design for message Insertion

In the text message insertion from above, four buttons have different functions:

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

In figure 8, there were four buttons which are:
1. Select the button to select the image from the directory in the computer
2. Select the button in the “Stego Image” used to choose the image as the output.
3. Insert Button used to insert a hidden message and execute the steganography process.
4. Exit Button to close the systems.

Figure 9 shows UI for document insertion. The first frame is Image Frame to show real Image File and Stego Frame to show manipulated image. There were also Text Boxes in the Stego Image and Real Image, which used to display data in the folder. Data File Text box used to show data in the folder, Password Text Box used to enter the password. The steganography process is executed if all the box filled.

In the insertion form, there were five buttons which are:
1. Select the button to choose an image from the computer.
2. Select Button in the Stego Image Button to select an image as output.
3. Select buttons in the data file used to choose the file to be inserted into the main image.
4. Insert Button used to execute the steganography.
5. Exit Button to exit the process.

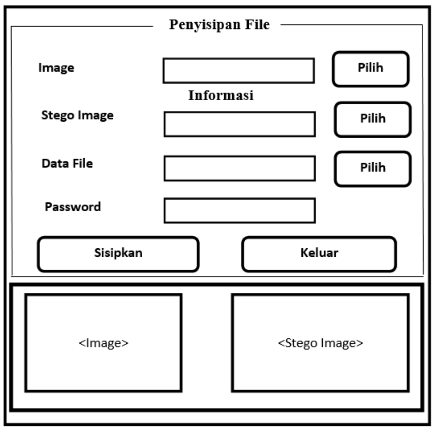


Figure 9. UI Design Insertion

In Figure 10, is the interface design of the message extract form in the form of a text message. There is a text box on the Stego Image to display data from the folder, the text box on the password is used to enter the password, the Message Text box is to display a secret message. The extraction process can be done when the Stego Image and Password text boxes have been filled in. Figure 10 shows the UI design for the message extraction form. There was a text box in the stego image to display the data from the directory, a password text box to show the password of the message, and a message text box to display the hidden message. The extraction process executed if all the text box filled
1. “Select” button in the stego image form used to take the manipulated image from the directory.
2. The “Extract” button is used to extract the hidden message from the file.
3. Exit Button to exit the Form

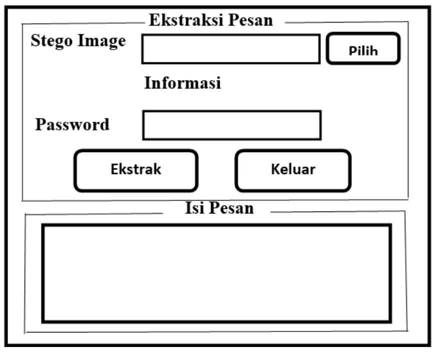


Figure 10. UI Design for Extraction

Figure 11 shows the design of the message extraction form in document format. There were text boxes to display data from directory, text box to display password, and a frame to display the stego image. The extraction executes if all the form is filled.

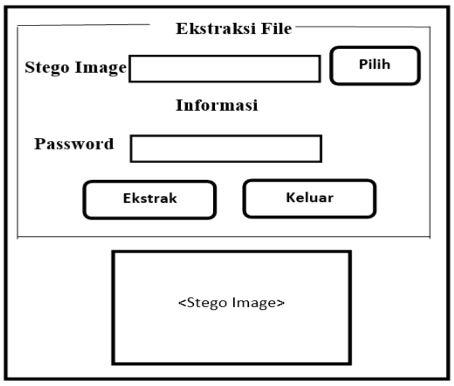


Figure 11. File Extraction UI Design

In this design, there were three buttons, which are:
1. “Select” Button in Stego Image to select the image from the computer directory
2. “Extract” button to process the extraction process.
3. “Exit” button to close the form.

D. Coding

This process was the implementation process where the design was translated into software code. The code is displayed below.

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

```
// encrypt text messages
if(features== UEM || features== CEM){
Cipher cipher= Cipher.getInstance("DES");
SecretKeySpec spec= new
SecretKeySpec(password.substring(0,
8).getBytes(), "DES");
cipher.init(Cipher.ENCRYPT_MODE, spec);
messageArray= cipher.doFinal(messageArray);
messageSize= messageArray.length;
}

// converts 32-bit message size into byte
array
tempByte= new byte[4];
for(i=24, j=0; i>=0; i-=8, j++)
{
tempInt= messageSize;
tempInt>>= i;
tempInt&= 0x000000FF;
tempByte[j]= (byte) tempInt;
}

//put 4 bytes messageSize array into master
file
writeBytes(tempByte);

// insert message
writeBytes(messageArray);
DataOutputStream out= new
DataOutputStream(new FileOutputStream(outputFile)
out.write(byteArrayOut,0,byteArrayOut.length);
byteOut.writeTo(out);
out.close();
{ catch(EOFException e) }
{ catch(Exception e) }
message= "Oops!!\nError: "+ e.toString();
e.printStackTrace();
return false;
message= "Sukses menyisipkan dalam '"+
outputFile.getName()+ "'.";
return true;

// insert data file
public SteganoInformation(File file)
this.file= file;
isEster= false;
if(!file.exists())
{ starter= null;
  return; }
if(file.getName().equals("Sec#x&y"))
{ isEster= true;
  return; }
byteArray= new byte[(int) file.length()];
DataInputStream in= new DataInputStream(new
FileInputStream(file));
in.read(byteArray, 0, (int) file.length());
in.close();
{ catch(Exception e) }
starter= null;
return;

// get the length of the original file
name= new byte[4];
String fileName= file.getName();
String fileExtension=
fileName.substring(fileName.length()-3,
fileName.length());
if(fileExtension.equalsIgnoreCase("jpg"))
inputMarker= steganografi.OFFSET_JPG;
else f(fileExtension.equalsIgnoreCase("png"))
inputMarker= steganografi.OFFSET_PNG;
else
inputMarker= steganografi.OFFSET_GIF_BMP_TIF;
retrieveBytes(name, byteArray, inputMarker);
dataLength= 0;
for(i=24,j=0; i>=0; i-=8,j++){
temp= name[j];
temp&= 0x000000FF;
temp<<= i;
dataLength|= temp;}
inputMarker= dataLength;
if(dataLength<0 || dataLength>file.length()){
starter= "Invalid";
return; }
```

E. Testing

In this step, we check the implemented modules, whether it met the requirement or not. After that, we conduct a test to check the UI design, whether it met the intended design or not (Guntoro, 2020).

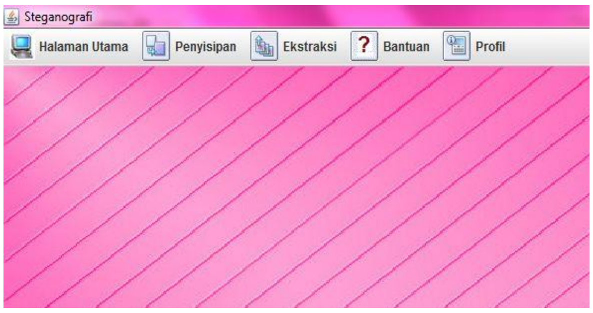


Figure 12. Steganography system UI

The test result from Figure 12 shows that the UI/UX design met the requirements. The interview and observation conducted to the BRI staff show that they feel satisfied with the UI.

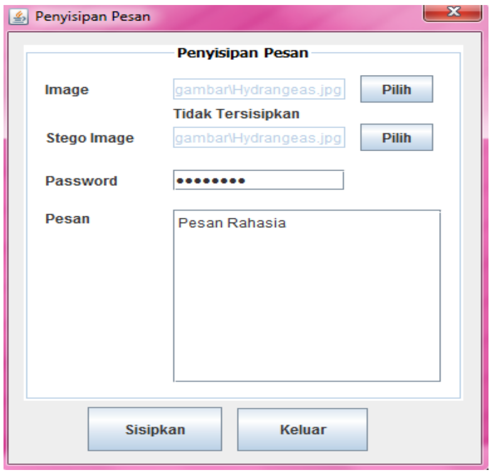


Image: gambar\Hydrangeas.jpg
Tidak Tersisipkan
Stego Image: gambar\Hydrangeas.jpg
Password: ••••••••
Pesan: Pesan Rahasia

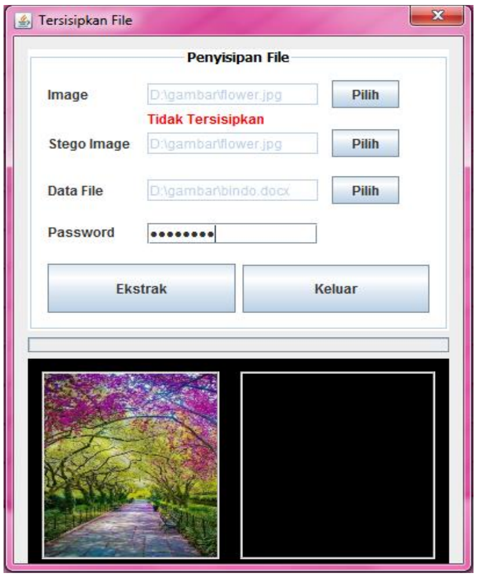
Figure 13.Message Insertion Form

The test result of Figure 13 shows there was a message hidden in the file. We chose the file and inserted the hidden message to test this form. If we chose the image first and then added the message, the system will display “Message been added” automatically. If the system can not add the message, the system will display “Message cant be added” in the form, as shown in Figure 14. After all

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v2i4.153

the text boxes are filled, the system can execute the algorithm. The result of the system was the same file but with slightly larger file size.



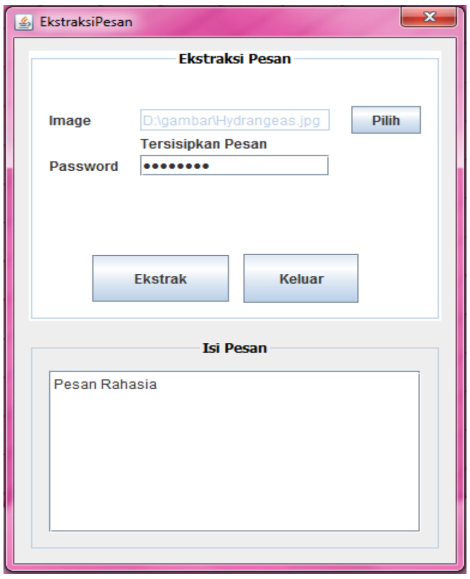
Tersisipkan File

Penyisipan File

Image: D:\gambar\flower.jpg [Pilih]
Tidak Tersisipkan
Stego Image: D:\gambar\flower.jpg [Pilih]
Data File: D:\gambar\bindo.docx [Pilih]
Password: ••••••••

[Ekstrak] [Keluar]

Figure 14.File Insertion form UI



EkstraksiPesan

Ekstraksi Pesan

Image: D:\gambar\Hydrangeas.jpg [Pilih]
Tersisipkan Pesan
Password: ••••••••

[Ekstrak] [Keluar]

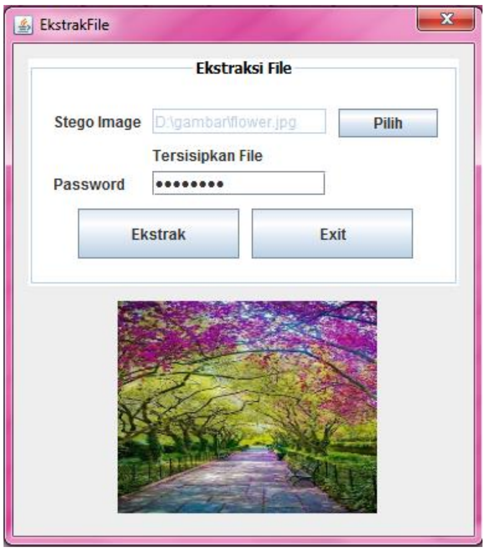
Isi Pesan

Pesan Rahasia

Figure 15. Extract Message Form UI

Figure 15, shows that we could only see the hidden message through the form. The first step to see the message is selecting the file with the hidden message. Next, the user needs to insert the password and click the extract button. The prerequisite for the extraction is to make sure the file has the hidden message.

Figure 16, shows that the user needs to make sure the file has a hidden message. In this figure, we test whether we can extract the message or not.



EkstrakFile

Ekstraksi File

Stego Image: D:\gambar\flower.jpg [Pilih]
Tersisipkan File
Password: ••••••••

[Ekstrak] [Exit]

Figure 16. Tampilan Form Ekstrak File

"Extract file" form has the same process as the extracted message where the first step is selecting the file, insert the password, and click extract to see the hidden message.

CONCLUSION AND SUGGESTION

Conclusion

Steganography using EOF proved to be able to insert a hidden message into an image file, could display the hidden message for example for the Banking process.

Suggestion

For the next research, the EOF algorithm can be combined with another algorithm, whether to secure the data or the file. The user of the system can be tested more broadly in another conventional Bank.

REFERENCE

Anggraini, Y., & Sakti, D. V. S. Y. (2014). Penerapan Steganografi Metode End Of File (EOF) dan Enkripsi Metode Data Encryption Standard (DES) Pada Aplikasi Pengamanan Data Gambar Berbasis Java Programming. *Konferensi Nasional Sistem Informasi 2014*, 1743.

Anindyawati, N., & Suryani, E. (2012). Pembangunan Aplikasi Penyembunyian Pesan Menggunakan Metode End Of File (EOF) ke dalam Citra Digital Terhadap Pesan yang Terenkripsi Dengan Algoritma RSA.

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License

ITSMART, *1*(1), 5–12. https://doi.org/https://doi.org/10.20961/itsmart.v1i1.576

Ariyus, D. (2009). *Keamanan Multimedia*. Andi Publisher.

Guntoro. (2020). *Metode Waterfall*. Materi Kuliah SI.

Hendini, A. (2016). Pemodelan UML Sistem Informasi Monitoring Penjualan dan Stok Barang (Studi Kasus: Distro Zhezha Pontianak). *Jurnal Khatulistiwa Informatika*, *IV*(2), 107–116.

Heriyanto, Y. (2018). Perancangan Sistem Informasi Rental Mobil Berbasis Web Pada PT. Apm Rent Car. *Intra-Tech*, *2*(2), 64–77.

Indriyono, B. V. (2016). Implementasi Sistem Keamanan File dengan Metode Steganografi EOF dan Enkripsi Caesar Cipher. *SISFO*, *6*(1).

Kristanto, A. (2004). Rekayasa Perangkat Lunak (Konsep Dasar). In *Gava Media* (1st ed.). Gava Media.

Masri, M., Masri, M., Widya, H., & Yuhendri, D. (2019). Perancangan Aplikasi Penyisipan Pesan Pada Pixel Citra Menggunakan Metode End Of File. *Journal of Electrical Technology*, *4*(3), 178–184.

Nasution, Y. R., Johar, A., & Coastera, F. F. (2017). Aplikasi Penyembunyian Multimedia Menggunakan Metode End Of File (EOF) dan Huffman Coding. *Jurnal Rekursif*, *5*(1).

Novita, R., & Sari, N. (2015). Sistem Informasi Penjualan Pupuk Berbasis E-Commerce. *TEKNOIF*, *3*(2).

Nurmaesah, N., Lestari, T., & Retno Mariana, A. (2018). APLIKASI STEGANOGRAFI UNTUK MENYISIPKAN PESAN DALAM MEDIA IMAGE | Nurmaesah | Jurnal TAM (Technology Acceptance Model). *Jurnal TAM (Technology Acceptance Model)* , *8*(1), 13–17. http://ojs.stmikpringsewu.ac.id/index.php/JurnalTam/article/view/82

Sa’adah, N., & Purqon, A. (2016). Perbandingan Hasil Deteksi Tepi Pada Citra Kanker Payudara Dengan Menggunakan Metode Canny Dan Metode Ant Colony Optimization (ACO). *Simposium Nasional Inovasi Dan Pembelaran Sains (SNIPS 2016)*.

Sitorus, M. (2015a). Teknik Steganography Dengan Metode Least Significant Bit (LSB). *Jurnal Ilmiah Fakultas Teknik LIMIT’S*, *11*(2), 54–59. https://doi.org/10.13140/RG.2.2.14942.23362

Sitorus, M. (2015b). Aplikasi Keamanan Data Dengan Teknik Steganografi Menggunakan Metode End Of File (EOF). *Proceedings of the 1st Informatics Conference*, *1*(1). https://doi.org/10.13140/RG.2.2.28364.00643

Suendri. (2018). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan). *ALGORITMA: Jurnal Ilmu Komputer Dan Informatika*, *3*(1).

The work is distributed under the Creative Commons Attribution-NonCommercial 4.0 International License