Pengaplikasian Block Cipher dan Steganografi untuk Merahasiakan Citra Rekam Medis Milik Pasien

Yunianti Khotimah - 18317014
Program Studi Teknik Biomedis
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: yuniantikhotimah@gmail.com

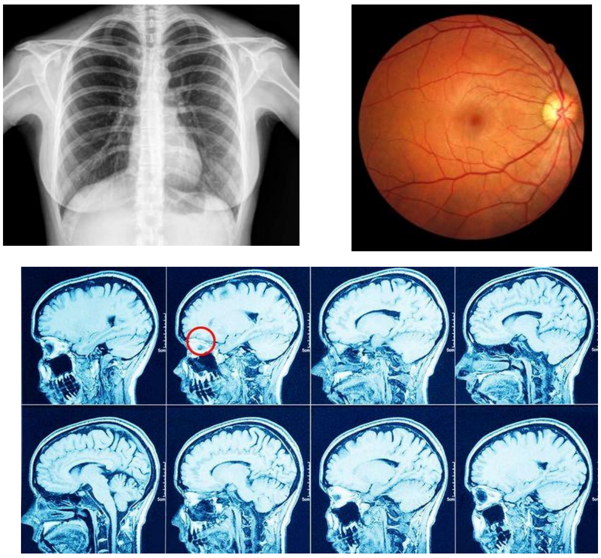
***Abstrak*—Perkembangan teknologi telah membuat dunia ini bergerak ke arah era digital. Begitu pula yang terjadi dalam bidang medis, dimana perkembangan teknologi telah membantu para tenaga medis dalam melakukan diagnosis penyakit pasien melalui alat-alat modern seperti X-Ray, MRI, dan lain sebagainya. Hasil diagnosis ini umumnya berupa citra rekam medis yang sifatnya rahasia, informasinya hanya boleh diberikan kepada pasien yang bersangkutan atau pihak tertentu yang sudah disetujui pasien tersebut. Karena sifatnya yang rahasia, maka unit pelayanan kesehatan seperti rumah sakit perlu menyimpan data citra medis ini dalam suatu mekanisme yang terjamin keamanannya dari serangan pihak luar. Salah satu untuk melindungi data ini adalah dnegan mengenkripsinya sehingga pihak luar tidak dapat dengan mudah mendapatkan informasi di dalamnya. Pada makalah ini, akan diajukan suatu program enkripsi sederhana menggunakan cipher blok berantai untuk melindungi data citra rekam medis pasien. Agar keamanan lebih kuat, data yang sudah terenkripsi akan disamarkan keberadaannya dengan teknik steganografi berupa mengubah format file menjadi suatu file teks.**

Kata kunci—citra medis, rahasia, cipher blok berantai, steganografi, keamanan

I. PENDAHULUAN

Rekam medis merupakan suatu catatan dokumen yang berisikan riwayat kesehatan pasien yang dapat berupa identitas pasien, hasil pemeriksaan, hasil pengobatan, dan pelayanan lainnya yang telah diberikan pada pasien oleh tenaga medis. Rekam medis ini beraneka ragam jenisnya mulai dari yang tersimpan dalam bentuk digital hingga yang tertulis atau tercetak. Rekam medis bersifat sangat rahasia karena merupakan suatu bentuk data pribadi seseorang. Umumnya, unit pelayanan kesehatan menyimpan data ini dalam suatu database dan informasinya hanya dapat diberikan kepada pasien dan pihak tertentu yang sudah disetujui oleh pasien. Karena sifatnya yang sangat rahasia, maka unit pelayanan kesehatan seperti rumah sakit misalnya perlu memberikan sistem keamanan khusus untuk data ini.

Salah satu data rekam medis yang cukup rentan keamanannya adalah citra medis digital. Citra medis digital merupakan hasil foto pemeriksaan menggunakan suatu alat diagnosis tertentu seperti hasil radiologi, funduskopi, MRI, dan lain sebagainya. Karena sifatnya yang digital, membuat data ini rentan terhadap serangan hacker maupun pencurian data diluar sana. Untuk melindungi data ini, diperlukan adanya suatu sistem enkripsi sehingga isi dan informasinya tidak dapat diperoleh dengan mudah.



Gambar 1 Citra medis digital berupa hasil radiologi, funduskopi, dan MRI

Pada makalah ini, akan diusulkan suatu sistem keamanan sederhana untuk data citra medis digital yang mengaplikasikan cipher blok berantai untuk proses enkripsinya (kriptografi) dan menyamarkannya menjadi suatu file teks untuk menyembunyikan bentuk file aslinya (steganografi). Sistem keamanan ini akan diwujudkan dalam suatu program GUI sederhana untuk memudahkan tenaga medis dalam menggunakannya.

II. DASAR TEORI

A. Kriptografi Modern

Kriptografi merupakan suatu ilmu khusus untuk menyembunyikan pesan-pesan agar terjaga kerahasiaannya. Pada kriptografi, pesan-pesan yang akan dirahasiakan ini disebut dengan plainteks yang kemudian akan disandikan (enkripsi) menjadi suatu cipherteks sehingga arti dari pesan tersebut tidak dapat dibaca oleh sembarang pihak. Pesan yang telah dienkripsi (cipherteks) ini dapat dikembalikan menjadi plainteks semula melalui proses dekripsi. Jadi, secara umum terdapat dua buah fungsi pada kriptografi yaitu enkripsi dan dekripsi yang dapat diformulasikan sebagai berikut.

$$E(P) = C$$

$$D(C) = P$$

$$D\big(E(P)\big) = P$$

dengan E merupakan fungsi enkripsi, D merupakan fungsi dekripsi, P adalah plainteks, dan C adalah cipherteks.

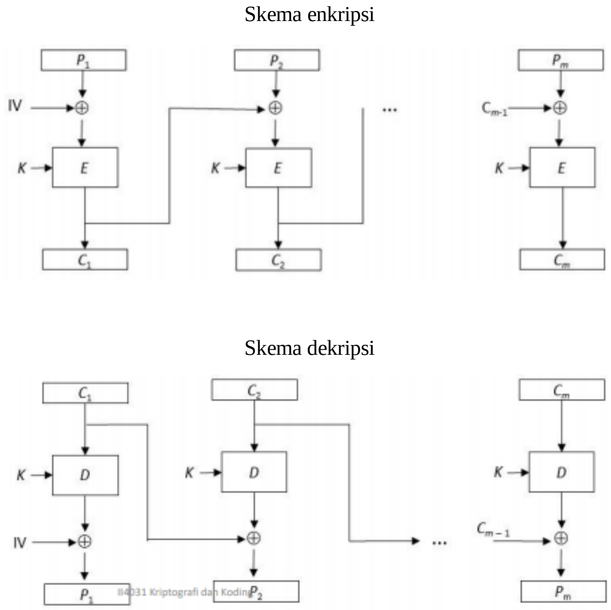
Gambar 2 Kriptografi pada zaman Mesir Kuno (atas) dan kriptografi modern (bawah)

Kriptografi sudah mulai digunakan pada zaman Mesir Kuno 4000 tahun yang lalu dan masih terus berkembang hingga saat ini. Ilmu ini berkembang dari kriptografi klasik yang mulanya hanya berupa huruf alphabet melalui pena dan kertas hingga menjadi kriptografi modern yang berbasis mode bit atau byte melalui komputer digital yang sering ditemui saat ini. Pada kriptografi modern data-data direpresentasikan dalam bentuk binernya berupa angka 0 dan 1 dengan operasi yang sering dilakukan adalah operasi XOR. Kriptografi modern dapat dibagi menjadi dua jenis berdasarkan basis operasi bitnya yaitu cipher alir dan cipher blok.

B. Block Cipher

Cipher blok atau *block cipher* merupakan salah satu jenis kriptografi modern yang beroperasi pada suatu blok bit, berbeda dengan cipher alir yang beroperasi pada bit tunggal. Pada cipher blok fungsi enkripsi dan dekripsi dilakukan dalam blok per blok sehingga panjang dari blok cipherteks akan sama dengan panjang blok plainteks. Terdapat beraneka ragam jenis mode operasi dalam cipher blok, salah satunya adalah cipher blok berantai atau *cipher block chaining* (CBC).

Pada cipher blok berantai terdapat suatu ketergantungan antar blok karena hasil enkripsi dari suatu blok akan diumpan balikkan ke blok selanjutnya sehingga suatu blok cipherteks tidak hanya bergantung pada blok plainteksnya melainkan juga bergantung pada seluruh blok plainteks sebelumnya. Skema sederhana proses enkripsi dan dekripsi dari cipher blok berantai adalah sebagai berikut.



Gambar 3 Skema enkripsi (atas) dan dekripsi (bawah) dari cipher blok berantai

Adanya ketergantungan antar blok ini membuat hasil blok-blok cipherteks dari blok-blok plainteks yang sama akan memiliki hasil yang berbeda dan membuatnya menjadi lebih kuat dari serangan kriptanalisis.

C. Steganografi

Steganografi adalah ilmu menyembunyikan pesan dengan suatu cara sedemikian sehingga orang luar tidak dapat mendeteksi keberadaan pesan tersebut. Tujuan utama dari kriptografi adalah menyembunyikan isi pesan sedangkan tujuan utama dari steganografi adalah menyembunyikan keberadaan pesan. Sama seperti kriptografi, steganografi juga telah ada sejak zaman kuno dan terus berkembang hingga sekarang (steganografi modern). Teknik steganografi modern (digital) yang umum dilakukan adalah menyembunyikan pesan digital di dalam dokumen digital lain seperti pesan teks yang disembunyikan pada file gambar.

Steganografi dapat digabung dengan kriptografi untuk memperkuat keamanan pesan yang akan dirahasiakan. Teknik penggabungan ini dilakukan dengan cara pesan rahasia dienkripsi terlebih dahulu dengan suatu teknik kriptografi kemudian hasil enkripsi disembunyikan dalam suatu cara tertentu dengan teknik steganografi.

Pada makalah ini, saya mengajukan penggabungan kriptografi dan teknik penyamaran (steganografi) berupa penyembunyian pesan dengan cara meyimpannya dalam bentuk format yang berbeda, pada kasus ini citra digital (file gambar) yang disamarkan menjadi suatu teks terenkripsi (file teks) sehingga orang luar tidak menduga bahwa ternyata teks terenkripsi tersebut sebenarnya berisi citra digital (gambar).

III. Rancangan Dan Implementasi

A. *Tampilan GUI pada Program*

Program dibuat dalam bahasa Phyton dengan tampilan GUI sederhana untuk memudahkan tenaga medis dalam menggunakannya. Program dapat menerima input dari pengguna berupa citra rekam medis (gambar) yang akan dienkripsi ataupun file teks terenkripsi yang ingin didekripsi kembali menjadi citra (gambar). Ketika input berupa citra rekam medis, maka output yang diberikan pada pengguna akan berupa file teks terenkripsi begitu pula yang terjadi jika sebaliknya. Kedua file (citra medis dan file teks terenkripsi) dapat disimpan kembali oleh pengguna. Berikut ini adalah rancangan tampilan antarmuka program yang akan dibuat.

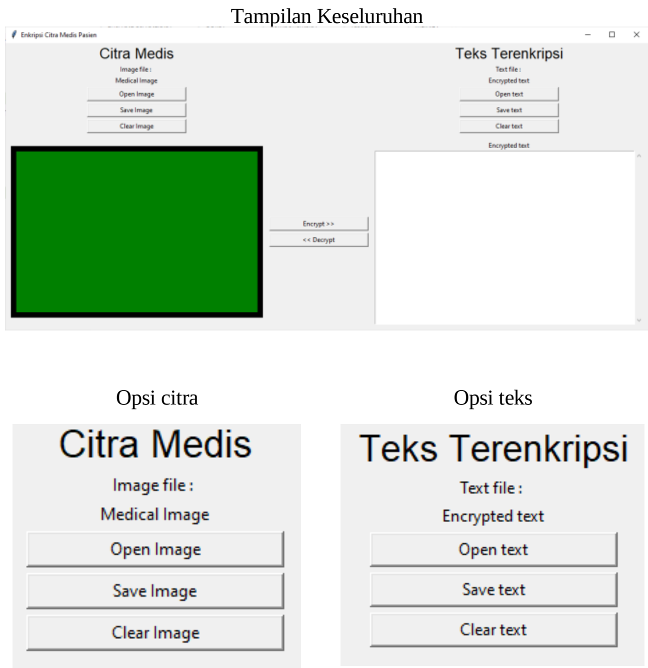
Row / Column	0	1	2	3	4	5	6
0	Judul 1 (Citra)				Judul 2 (File teks)		
1	File citra yang dibuka				File teks yang dibuka		
2	Opsi pilihan citra				Opsi pilihan teks		
3	Tampilan Citra			Opsi enkripsi dan dekripsi	Tampilan Teks		
4							
5							
6							

Gambar 4 Rancangan tampilan GUI

Secara sederhana tampilan GUI terbagi menjadi dua bagian yaitu sisi kiri yang akan menampilkan citra beserta opsi yang dapat dilakukan dan sisi kanan yang akan menampilkan teks enkripsi beserta opsi yang dapat dilakukan. Pada bagian tengah antar keduanya akan terdapat tombol enkripsi dan dekripsi untuk mengubah citra medis menjadi teks terenkripsi atau sebaliknya. Untuk opsi terdapat 3 jenis pilihan yang dapat dilakukan masing-masing untuk citra dan teks yaitu:

1. Membuka file citra/teks
2. Menyimpan file citra/teks
3. Menghapus file citra/teks yang ditampilkan

Tampilan GUI diimplementasikan dalam 3 file Python yaitu "Component.py" yang berisikan fungsi untuk membangkitkan frame teks dan frame tombol pada GUI, "GUI.py" yang berisikan tampilan GUI secara keseluruhan beserta fungsi-fungsi yang dijalankan ketika tombol tertentu ditekan, serta "Main.py" yang merupakan program utama untuk menjalankan GUI. Adapun kode program untuk ketiga file ini terlampir pada Appendix (bagian akhir) dari makalah ini. Berikut ini adalah tampilan GUI ketika program dijalankan (implementasi).

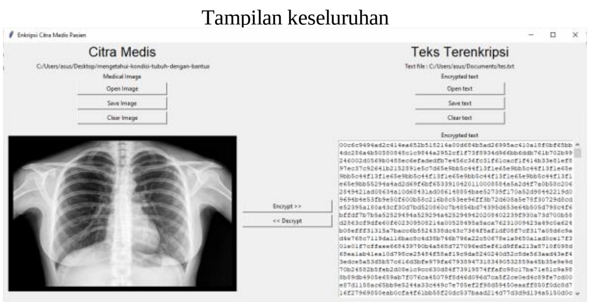


Gambar 5 Tampilan GUI program (implementasi)

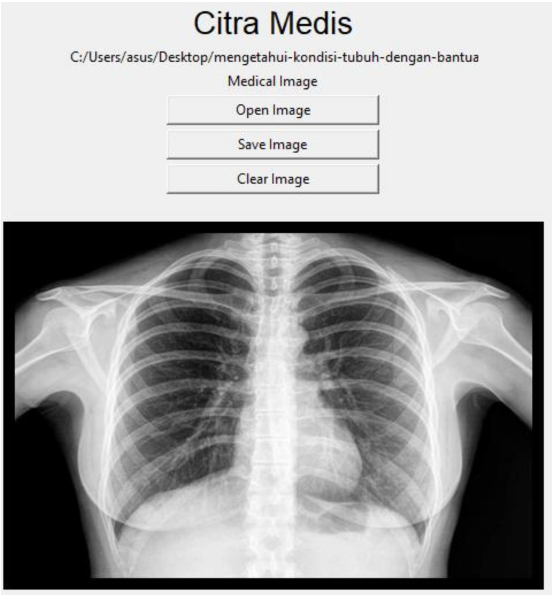
B. *Opsi Citra dan Teks*

Seperti yang telah disebutkan sebelumnya, terdapat 3 jenis opsi/pilihan yang disediakan untuk masing-masing citra dan teks yaitu membuka file, menyimpan file, dan menghapus file (gambar 5). Implementasi ketiga fungsi opsi ini ditulis pada file Python "GUI.py" yang terlampir pada Appendix.

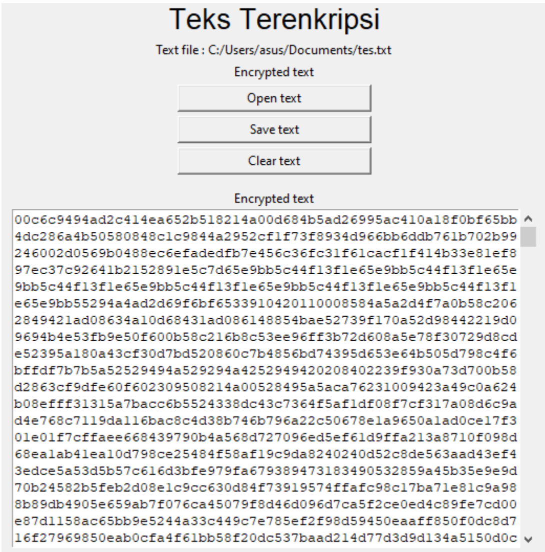
Opsi pertama (membuka file) dirancang dengan sedemikian rupa sehingga file yang terbuka akan langsung diperlihatkan pada frame tampilan kepada pengguna (kiri untuk tampilan citra dan kanan untuk tampilan teks). Berikut ini adalah contoh tampilan yang dihasilkan ketika membuka file citra medis dan file teks.



Tampilan citra

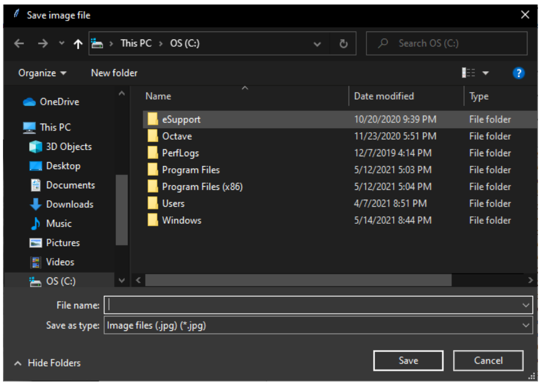


Tampilan teks



Gambar 6 Contoh tampilan citra medis dan teks terenkripsi yang terbuka

Opsi kedua (menyimpan file) dirancang dengan sedemikian rupa sehingga baik tampilan citra ataupun teks terenkripsi yang saat itu dilihat oleh pengguna dapat disimpan dalam format masing-masing (png atau jpg untuk citra medis dan txt untuk teks terenkripsi). Ketika pengguna memilih opsi ini, akan muncul jendela penyimpanan yang memperbolehkan pengguna untuk memilih lokasi penyimpanan dan nama penyimpanan sesuai yang ia inginkan.

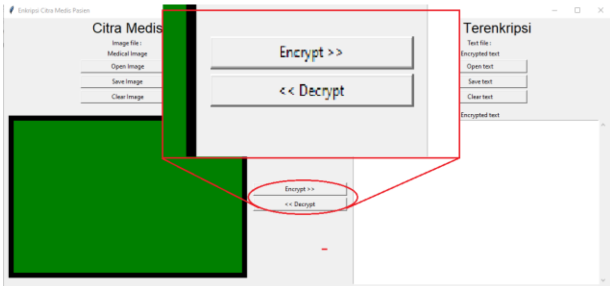


Gambar 7 Jendela penyimpanan

Opsi ketiga (menghapus file) dirancang dengan sedemikian rupa sehingga ketika opsi ini dipilih, maka direktori file dan tampilan yang saat itu diperlihatkan pada pengguna akan terhapus sehingga tampilan program akan kembali ke tampilan default seperti semula (gambar 5).

C. *Enkripsi-Dekripsi dan Penyamaran Pesan*

Proses enkripsi-dekripsi dilakukan dengan cipher blok berantai dengan panjang 1 bloknya adalah 8 bit (1 karakter). Proses enkripsi hanya dilakukan untuk mengenkripsi citra medis menjadi file teks terenkripsi bukan sebaliknya, begitu pula dengan proses dekripsi yang hanya dilakukan untuk mendekripsi file teks terenkripsi menjadi citra medis kembali. Bit 11111111 yang bernilai 255 digunakan sebagai vektor inisiasi dan fungsi enkripsi-dekripsi hanya memanfaatkan pergeseran bit sederhana. Hal ini dinilai cukup aman meski fungsi enkripsi-dekripsi hanya sederhana karena mengingat implemenetasi menggunakan blok berantai sehingga antar blok saling memiliki keterkaitan. Program enkripsi-dekripsi diimplementasikan dalam file Pyhton “EncryptDecryptLib.py” yang terlampir pada Appendix (bagian akhir makalah ini).



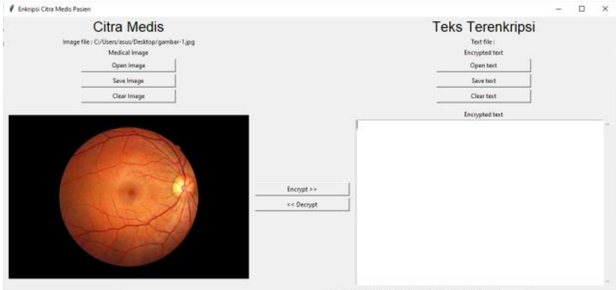
Gambar 8 Tampilan tombol enkripsi-dekripsi

Berikut ini adalah skema atau tahapan rancangan proses enkripsi dan penyamaran pesan yang diajukan.

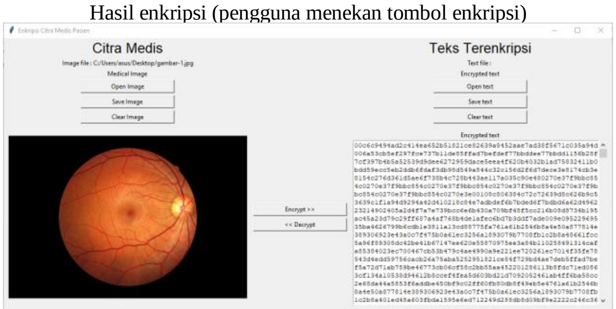
1. Citra medis yang dibuka pengguna dibaca sebagai *file of bytes* dan nilai masing-masing byte ini dimasukkan dalam sebuah blok array (sebagai plainteks)
2. Blok plainteks pertama kemudian di-XOR-kan dengan vektor inisiasi (11111111) dan hasilnya masuk ke dalam fungsi enkripsi berupa pergeseran bit 3 langkah ke kiri (rotasi sirkuler) menjadi blok cipherteks pertama

3. Hasil blok cipherteks ini kemudian menjadi nilai dari vektor inisiasi yang akan di-XOR-kan dengan blok plainteks selanjutnya. Langkah 2-3 ini diulangi hingga semua blok plainteks telah berhasil terenkripsi
4. Hasil tiap blok cipherteks kemudian disamarkan (steganografi) menjadi suatu file teks dalam bentuk karakter heksadesimal sehingga tiap blok (8-bit) akan direpresentasikan dalam 2 karakter heksadesimal dan ditampilkan pada pengguna serta dapat disimpan

Membuka file citra medis



Hasil enkripsi (pengguna menekan tombol enkripsi)

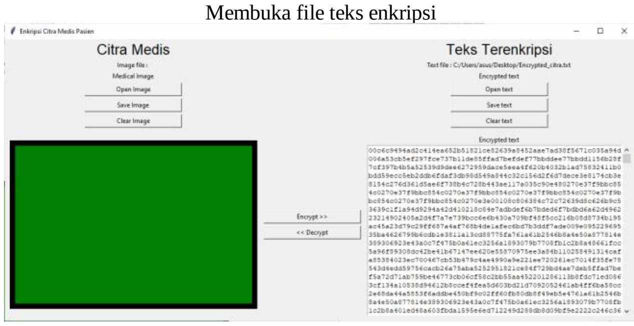


Gambar 9 Contoh proses enkripsi citra medis

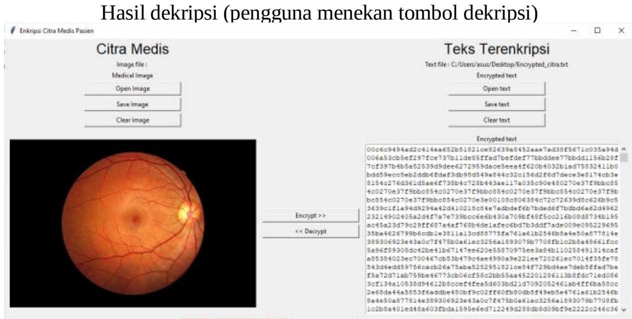
Berikut ini adalah skema atau tahapan rancangan proses dekripsi yang diajukan.

1. File teks yang dibuka pengguna dibaca sebagai suatu string heksadesimal yang akan dibuka penyamarannya dimana tiap 2 karakter heksadesimal akan menjadi 1 blok cipherteks
2. Bit pada blok cipherteks pertama masuk ke fungsi dekripsi yaitu penggeseran 3 langkah ke kanan secara sirkuler kemudian di-XOR-kan dengan vektor inisiasi (11111111) dan menjadi blok plainteks pertama
3. Blok cipherteks yang sudah didekripsi ini kemudian menjadi nilai dari vektor inisisasi yang akan di-XOR-kan dengan blok cipherteks selanjutnya untuk didekripsi. Langkah 2-3 ini diulangi hingga semua blok cipherteks berhasil didekripsi
4. Hasil akhirnya akan berupa blok plainteks semula dalam array bytes yang dapat disimpan dan ditampilkan dalam format gambar kembali

Membuka file teks enkripsi



Hasil dekripsi (pengguna menekan tombol dekripsi)



Gambar 10 Contoh proses dekripsi citra medis

IV. Hasil Dan Pembahasan

A. *Tampilan GUI dan Pilihan Opsi pada Program*

Untuk tampilan GUI dinilai telah sesuai dengan rancangan yang diinginkan. Tampilan GUI dinilai sederhana dan nama tombol-tombol dinilai cukup jelas menggambarkan kegunaan tombol tersebut. Diharapkan dengan tampilan yang cukup simpel ini, program ini mampu membantu para tenaga medis yang bekerja di unit layanan kesehatan untuk melindungi data citra rekam medis pasien.

Mungkin yang bisa dikembangkan pada program ini untuk lebih memudahkan para tenaga medis yang menggunakannya adalah menambahkan opsi untuk dapat membuka multi file (banyak file sekaligus), karena umumnya database yang disimpan oleh unit layanan kesehatan pasti jumlahnya banyak sedangkan program ini masih memiliki kekurangan dimana pemrosesan hanya dapat dilakukan terhadap single file (satu file saja).

B. *Keamanan Enkripsi-Dekripsi*

Dari segi keamanan, program ini dinilai memiliki tingkat keamanan yang sedang, cukup aman jika hanya digunakan untuk menyembunyikan citra rekam medis pasien. Meskipun fungsi enkripsi-dekripsi yang dibuat hanya sederhana menggunakan XOR dan penggeseran bit secara sirkuler, namun dengan mengaplikasikan cipher block berantai membuat antar blok saling bergantung satu sama lain sehingga serangan kriptanalisis dapat diminimalisir.

Jika diingingkan keamanan yang lebih tinggi, keamanan program ini masih dapat dikembangkan dengan mengubah fungsi enkripsi-dekripisi menjadi lebih kompleks. Perubahan ini dapat berupa pengaplikasian jenis cipher tertentu maupun penambahan suatu skema pasangan kunci public-privat yang hanya diketahui oleh pihak unit layanan kesehatan dan pasien.

APPENDIX

Komponen GUI (Component.py)

```
import tkinter as tk
import tkinter.scrolledtext as st

class TextFrame:
    def __init__(self,title,width=50,height=5):
        # Constructor for text frame
        # Components : title and input field (big text)
        # Input :
        #           title(string) for title
        #           width(int) and height(int) for input field
 dimensions
        self.frame = tk.Frame()

        self.label = tk.Label(master=self.frame,text=title)
        self.label.pack()

        self.entry = st.ScrolledText(master=self.frame,width=w
idth,height=height)
        self.entry.pack()

class ButtonListFrame:
    def __init__(self,title,labels,width=20):
        # Constructor for list of buttons frame
        # Components : title and button list
        # Input :
        #           title(string) for title
        #           labels(list of strings) for button label
        #           width(int) for button width
        self.frame = tk.Frame()

        self.label = tk.Label(master=self.frame,text=title)
        self.label.pack()

        self.button_list = []
        for label in labels:
            new_button = tk.Button(master=self.frame,text=labe
l,width=width)
            new_button.pack(padx=2,pady=2)
            self.button_list.append(new_button)
```

Implementasi GUI (GUI.py)

```
import tkinter as tk
import tkinter.scrolledtext as st
import tkinter.filedialog as fd
import tkinter.messagebox as mb
import PIL.Image
import math

from PIL import ImageTk
from tkinter import *
from Components import *

from EncryptDecryptLib import *

class GUI:
    def __init__(self,parent):
        #--- init ---#
        self.parent = parent
        self.parent.title("Enkripsi Citra Medis Pasien")

        #--- define grid ---#
        self.parent.columnconfigure([0,1,2,3,4,5,6],weight=1)
        self.parent.rowconfigure([0,1,2,3,4,5,6],weight=1,mins
ize=10)

        #--- title 1 ---#
        self.title_frame = tk.Frame()
        tk.Label(master=self.title_frame,text="Citra Medis",fo
nt=("Arial",20)).pack()
        self.title_frame.grid(row=0,column=0,columnspan=3,rows
pan=1)

        #--- title 2 ---#
        self.title_frame = tk.Frame()
        tk.Label(master=self.title_frame,text="Teks Terenkrips
i",font=("Arial",20)).pack()
        self.title_frame.grid(row=0,column=4,columnspan=3,rows
pan=1)

        self.fileImage = ""
        #--- file name 1---#
        self.filename1 = tk.Label(master=self.parent,text="Ima
ge file : " + self.fileImage,width=50)
        self.filename1.grid(row=1,column=0,columnspan=3)

        self.fileText = ""
        #--- file name 2---#
        self.filename2 = tk.Label(master=self.parent,text="Tex
t file : " + self.fileText,width=50)
        self.filename2.grid(row=1,column=4,columnspan=3)

        #--- image frame ---#
        image_button_list = ["Open Image","Save Image","Clear
Image"]
        self.image_frame = ButtonListFrame(
            title = "Medical Image",
            labels = image_button_list,
            width = 25
        )
        self.image_frame.button_list[0].bind("<Button-
1>",lambda event,text="image": self.OpenFile(event,text))
        self.image_frame.button_list[1].bind("<Button-
1>",lambda event,text="image": self.SaveFile(event,text))
        self.image_frame.button_list[2].bind("<Button-
1>",lambda event,text="image": self.Clear(event,text))
        self.image_frame.frame.grid(row=2,column=0,columnspan=
3,rowspan=1)

        #--- teks frame ---#
        text_button_list = ["Open text","Save text","Clear tex
t"]
        self.text_frame = ButtonListFrame(
            title = "Encrypted text",
            labels = text_button_list,
            width = 25
        )
        self.text_frame.button_list[0].bind("<Button-
1>",lambda event,text="text": self.OpenFile(event,text))
        self.text_frame.button_list[1].bind("<Button-
1>",lambda event,text="text": self.SaveFile(event,text))
        self.text_frame.button_list[2].bind("<Button-
1>",lambda event,text="text": self.Clear(event,text))
        self.text_frame.frame.grid(row=2,column=4,columnspan=3
,rowspan=1)

        #--- image ---#
        self.container_im = Frame(parent, bg='black', width=45
0, height=300)
        self.container_im.grid(row=3,column=0,columnspan=3,row
span=4,pady=10,padx=10)
        self.canvas_for_image = Canvas(self.container_im, bg='
green', height=300, width=450, borderwidth=0, highlightthickne
ss=0)
        self.canvas_for_image.grid(row=3, column=0, sticky='ne
sw',columnspan=3,rowspan=4,pady=10,padx=10)
        self.image = []
        self.image_on_canvas = self.canvas_for_image.create_im
age(0, 0, image=self.image, anchor='nw')
        self.image_byteintarray = []

        #--- document ---#
        self.document = TextFrame(
            title="Encrypted text",
            width=60,
            height=20
        )
        self.document.frame.grid(row=3,column=4,columnspan=3,r
owspan=4,pady=10,padx=10)
```

```
        #--- button frame ---#
        self.button_frame = tk.Frame()
        self.button_frame.grid(row=3,column=3,columnspan=1,rowspan=4)

        #--- encrypt button ---#
        self.encrypt_button = tk.Button(master=self.button_frame,text="Encrypt >>",command=self.Encrypt,width=25)
        self.encrypt_button.pack(padx=2,pady=2)

        #--- decrypt button ---#
        self.decrypt_button = tk.Button(master=self.button_frame,text="<< Decrypt",command=self.Decrypt,width=25)
        self.decrypt_button.pack(padx=2,pady=2)

    def Encrypt(self):
        image_byteintarray = self.image_byteintarray

        # Check for validity
        if (len(image_byteintarray)==0): # Empty document
            mb.showinfo(title="Alert",message="Please insert image")
        else:
            text_byteintarray = EncryptImage(image_byteintarray)
            text_hexstring = ByteIntArrayToHex(text_byteintarray)

            self.document.entry.delete("1.0",tk.END)
            self.document.entry.insert("1.0",text_hexstring)

    def Decrypt(self):
        document = self.document.entry.get("1.0",tk.END)[:-1]

        if (len(document)==0): # Empty document
            mb.showinfo(title="Alert",message="Please insert text")
        else:
            text_byteintarray = HexToByteIntArray(document)
            image_byteintarray = DecryptText(text_byteintarray)

            self.image_byteintarray = image_byteintarray

            savename = 'temp.png'
            output_file = open(savename, "wb")

            for byteint in image_byteintarray:
                output_file.write(byteint.to_bytes(1,byteorder='little'))
            output_file.close()

            self.image = PIL.Image.open(savename)
            self.image = ImageTk.PhotoImage(self.image.resize((450, 300), PIL.Image.ANTIALIAS))
            self.canvas_for_image.itemconfig(self.image_on_canvas, image = self.image)


    def Clear(self,event,text):
        if (text=="text"):
            self.fileText = ""
            self.filename2["text"] = "Text file : " + self.fileText
            self.document.entry.delete("1.0",tk.END)
            self.document.entry.insert("1.0","")
        elif (text=="image"):
            self.fileImage = ""
            self.filename1["text"] = "Image file : " + self.fileImage
            self.image = []
            self.canvas_for_image.itemconfig(self.image_on_canvas, image = self.image)
            self.image_byteintarray = []

    def OpenFile(self,event,text):
        # Open file using open file dialog

        filename=""
        if (text=="text"):
            # Take filename
            filename = fd.askopenfilename(
                initialdir = "/",
                title = "Select " + text + " file",
                filetypes = [("Text files (.txt)","*.txt"),("All files","*.*")]
            )
        elif (text=="image"):
            # Take filename
            filename = fd.askopenfilename(
                initialdir = "/",
                title = "Select " + text + " file",
                filetypes = [("Image files (.jpg)","*.jpg"),("Image files (.png)","*.png"),("All files","*.*")]
            )

        if (filename!=""): # If filename is chosen
            content = OpenFileAsByteIntArray(filename)
            content_bytes = bytes(content)

            if (text=="image"): # For image
                self.filename1["text"] = "Image file : " + filename
                self.fileImage = filename
                self.image_byteintarray = content

                savename = 'temp.png'
                output_file = open(savename, "wb")

                for byteint in content:
                    output_file.write(byteint.to_bytes(1,byteorder='little'))
                output_file.close()

                self.image = PIL.Image.open(savename)
                self.image = ImageTk.PhotoImage(self.image.resize((450, 300), PIL.Image.ANTIALIAS))
                self.canvas_for_image.itemconfig(self.image_on_canvas, image = self.image)

            elif (text=="text"): # For text
                self.filename2["text"] = "Text file : " + filename
                self.fileText = filename

                self.document.entry.delete("1.0",tk.END)
                self.document.entry.insert("1.0",content_bytes)

        return "break"

    def SaveFile(self,event,text):
        # Save file using save file dialog

        filename=""
        if (text=="text"):
            # Take filename
            filename = fd.asksaveasfilename(
                initialdir = "/",
                title = "Save " + text + " file",
                filetypes = [("Text files (.txt)","*.txt"),("All files","*.*")],
                defaultextension = [("Text files (.txt)","*.txt"),("All files","*.*")]
            )
        elif (text=="image"):
            # Take filename
            filename = fd.asksaveasfilename(
                initialdir = "/",
                title = "Save " + text + " file",
                filetypes = [("Image files (.jpg)","*.jpg"),("Image files (.png)","*.png"),("All files","*.*")],
                defaultextension = [("Image files (.jpg)","*.jpg"),("Image files (.png)","*.png"),("All files","*.*")]
            )

        if (filename!=""): # If file name is chosen
            file = open(filename,"wb")
            if (text=="text"): # For document, insert the document
                document = self.document.entry.get("1.0",tk.END)[:-1]
```

```
                document_byteintarray = StringToByteIntArray(d
ocument)
                for byteint in document_byteintarray:
                    file.write(byteint.to_bytes(1,byteorder='l
ittle'))
            elif (text=="image"): # For signature, insert the 
signature
                for byteint in self.image_byteintarray:
                    file.write(byteint.to_bytes(1,byteorder='l
ittle'))
            file.close()

        return "break"
```

Implementasi Enkripsi/Dekripsi (EncryptDecryptLib.py)

```
import math
import hashlib

# Function to left
# rotate n by d bits
def leftRotate(n, d):
    INT_BITS = 8
    return (n << d) & 0xFF | (n >> (INT_BITS - d))

# Function to right
# rotate n by d bits
def rightRotate(n, d):
    INT_BITS = 8
    return (n >> d) | (n << (INT_BITS - d)) & 0xFF

def ByteIntArrayToHex(byteintarray):
    hexString = ''
    for byteint in (byteintarray):
        temp = (hex(byteint)[2:])
        if (len(temp)==1):
            hexString = hexString + '0' + temp
        else:
            hexString = hexString + temp
    return hexString

def HexToByteIntArray(hexString):
    byteintarray = []
    for i in range (len(hexString)//2):
        temp = '0x'
        temp = temp + hexString[i*2:i*2+2]
        temp = int(temp,16)
        byteintarray.append(temp)
    return byteintarray

def StringToByteIntArray(string):
    # Mengubah string menjadi array of integer (byte) sesuai d
engan ascii/utf-8
    # Input : string
    # Output : array of integer (byte) dari string
    byteint_array = []

    for char in string:
        byteint_array.append(ord(char))

    return byteint_array

def ByteIntArrayToString (byteint_array):
    # Mengubah string menjadi array of integer (byte) sesuai d
engan ascii/utf-8
    # Input : array of integer (byte)
    # Output : string
    string = "".join([chr(value) for value in byteint_array])

    return string

def OpenFileAsByteIntArray(filename):
    # Membuka file dengan nama filename per byte lalu menyimpa
nnya menjadi array of integer (byte)
    # Input : filename
    # Output : array of integer (byte) dari isi file

    # Buka file
    input_file = open(filename,"rb")

    # Baca isi file per byte hingga habis
    byteint_array = []
    byte = input_file.read(1)
    while (byte):
        # Ubah byte tersebut menjadi integer yang sesuai lalu 
masukkan ke array
        byteint = int.from_bytes(byte,byteorder='little')
        byteint_array.append(byteint)
        byte = input_file.read(1)

    # Tutup file
    input_file.close()

    return byteint_array

def EncryptImage(image_byteintarray):
    init = 255
    text_byteintarray = []
    for byteint in (image_byteintarray):
        xor = init^byteint
        enc = leftRotate(xor, 3)
        text_byteintarray.append(enc)
        init = enc
    return text_byteintarray

def DecryptText(text_byteintarray):
    init = 255
    image_byteintarray = []
    for byteint in (text_byteintarray):
        dec = rightRotate(byteint, 3)
        xor = init^dec
        image_byteintarray.append(xor)
        init = byteint
    return image_byteintarray
```

Program Utama (main.py)

```
from GUI import *

window = tk.Tk()
GUI(window)
window.mainloop()
```

Link Repository Github

https://github.com/Kyunianti19/Makalah-Kripto-18317014

Acknowledgment

Puji syukur kepada Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan makalah ini. Penulis juga ingin menyampaikan ucapan terima kasih kepada Bapak Rinaldi Munir selaku dosen pengajar mata kuliah II4031 Kriptografi dan Koding atas ilmu yang telah diberikan. Selain itu, penulis juga ingin menyampaikan terima kasih kepada orang tua dan teman-teman yang tak bisa penulis sebutkan satu persatu atas dukungan yang mereka berikan kepada penulis.

References

[1] R. Munir, “Pengantar Kriptografi STI” https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Pengantar-Kriptografi-STI-(2021).pdf (diakses pada 24 Mei 2021)

[2] R. Munir, “Kriptografi Modern” https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Kripto-modern-2021.pdf (diakses pada 24 Mei 2021)

[3] R. Munir, “Block Cipher”

https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Block-Cipher-2021.pdf (diakses pada 24 Mei 2021)

[4] R. Munir, “Steganografi”
https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Steganografi-2021.pdf (diakses pada 24 Mei 2021)

[5] Artikel KM, “Mengenal Rekam Medis Pasien di Rumah Sakit”
https://krakataumedika.com/info-media/artikel/mengenal-rekam-medis-pasien-di-rumah-sakit#:~:text=Rekam%20medis%20adalah%20berkas%20yang,yang%20telah%20diberikan%20kepada%20pasien. (diakses pada 24 Mei 2021)

[6] K. Adrian, “Mengetahui Kondisi Tubuh dengan Pemeriksaan X-Ray”
https://www.alodokter.com/mengetahui-kondisi-tubuh-dengan-bantuan-x-ray (diakses pada 25 Mei 2021)

[7] S. Evani, “Diagnosis Retinopati”
https://www.alomedika.com/penyakit/oftalmologi/retinopati/diagnosis (diakses pada 25 Mei 2021)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Yogyakarta, 25 Mei 2021

Yunianti Khotimah (18317014)