ENCRYPTION PROJECT 2023

Group 18

Group Members

CHIKEKA, MILANI 40100499

HANSA, ZAI 41425626

PHOSA, Selaelo 37685775

SOMARU, AMIEL 37379283

CONTENTS

KEY WORDS

- DES – Data Encryption standard
- AES-Advanced Encryption Standard

WHAT IS DES ALGORITHM

Is an algorithm that utilizes the use of the symmetric conversion of 48-block cipher text blocks to 64-bit plaintext blocks. The DES algorithm uses a symmetric key that enables users to encrypt and decrypt data using the same key.

The stages of encryption in DES:

1. Key transformation
2. Expansion permutation
3. S-Box permutation
4. P-Box permutation
5. XOR and swap

THE ADVANTAGES OF DES

1. It is simple to integrate into both software and hardware
2. The 8 S-boxes used in each round are kept secret, and it is also hard for anyone to learn their design, making an attack even more difficult

THE DISADVANTAGE OF DES

1. The key size of DES is one of its drawbacks.
2. DES performs the same set of algorithms on the plaintext block 16 times during the course of its 16 rounds.

WHAT IS AES ALGORITHM

AES is a systematic encryption technique that can be used to safeguard confidential data. AES is a block cipher with a 128-bit length. Additionally, AES supports keys with lengths ranging from 128, 192, or 256 bits.

The stages of encryption in AES:

1.) Sub Bytes
2.) Shift Rows
3.) Mix Columns
4.) Add Round Key

THE ADVANTAGES OF AES

1. Uses larger key lengths with lengths of 128,192,256 bits which makes more challenging for hackers

THE DISADVANTAGES OF AES

1. Utilizes basic algebra
2. Every block is consistently and uniformly encrypted.
3. Hard to implement onto hardware
4. Due to the large key size, it requires more power to encrypt and decrypt

HOW DES COMPARES TO AES ALGORITHM

The block is split into two halves before being encrypted using the Feistel notion, which is used by the DES structure. While AES uses the permutation-substitution technique, which encrypts a block using both permutation and substitution.

WHAT IS PASSWORD HASHING?

The process by which the specified key or character is transformed into an integer. It's shortened, which helps you find the string it was assigned to for security purposes. An integrity check on a piece of information is the primary purpose of hashing.

THE ADVANTAGES OF PASSWORD HATCHING

Allows for an effective sorting of large amounts of data. You have a secure storage and management password.

THE CODE USED FOR OUR ENCRYPTION PROJECT

```python
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox
from Cryptodome.Cipher import DES
from Cryptodome.Random import get_random_bytes
import os
import hashlib


class Encryption:
    def __init__(self, master):
        self.master = master
        self.key = None

    def create_gui(self):
        self.master.title("Encryption Example")



        # Create UI elements
        self.heading_label = tk.Label(
            self.master, text="Encryption", font=("Helvetica", 16, "bold")
        )
        self.heading_label.pack(pady=10)

        self.file_frame = tk.Frame(self.master)
        self.file_frame.pack(pady=10)

        self.select_label = tk.Label(self.file_frame, text="Select File:")
        self.select_label.pack(side=tk.LEFT)

        self.file_path_var = tk.StringVar()
        self.file_path_entry = tk.Entry(
            self.file_frame, textvariable=self.file_path_var, width=30
        )
        self.file_path_entry.pack(side=tk.LEFT)

        self.browse_button = tk.Button(
            self.file_frame, text="Browse", command=self.browse_file
        )
        self.browse_button.pack(side=tk.LEFT)

        self.key_frame = tk.Frame(self.master)
        self.key_frame.pack(pady=10)

        self.key_label = tk.Label(self.key_frame, text="Encryption Key:")
        self.key_label.pack(side=tk.LEFT)

        self.key_var = tk.StringVar()
        self.key_entry = tk.Entry(self.key_frame, textvariable=self.key_var,
width=30)
        self.key_entry.pack(side=tk.LEFT)

        global v
        v = tk.IntVar()

        tk.Radiobutton(root,
                    text="DES",
                    padx = 20,
                    variable=v,
                    value=1).pack(pady=5)

        tk.Radiobutton(root,
                    text="Own Algorithm",
                    padx = 20,
                    variable=v,
                    value=2).pack(pady=5)
```

```python
    self.encrypt_button = tk.Button(
        self.master, text="Encrypt", command=self.encrypt_file
    )
    self.encrypt_button.pack(pady=10)

    self.decrypt_button = tk.Button(
        self.master, text="Decrypt", command=self.decrypt_file
    )
    self.decrypt_button.pack(pady=10)

def browse_file(self):
    try:
        file_path = filedialog.askopenfilename()
        self.file_path_var.set(file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Failed to browse file: {str(e)}")


def encrypt_file(self):
    file_path = self.file_path_var.get()
    key = self.key_var.get()

    if file_path and key:

        if v.get() == 1:
            try:
                # Read the contents of the file
                with open(file_path, "rb") as file:
                    plaintext = file.read()

                # Generate a random 8-byte initialization vector (IV)
                iv = get_random_bytes(8)
```

```
                    # Create a DES cipher object with the key and mode
                    cipher = DES.new(key.encode(), DES.MODE_CBC, iv)

                    # Pad the plaintext to be a multiple of 8 bytes
                    padded_plaintext = self.pad(plaintext)

                    # Encrypt the padded plaintext
                    ciphertext = cipher.encrypt(padded_plaintext)

                    # Create a new file path for the encrypted file
                    encrypted_file_path = file_path + ".encrypted"

                    # Write the IV and ciphertext to the new file
                    with open(encrypted_file_path, "wb") as file:
                        file.write(iv + ciphertext)

                    # Delete the original file
                    os.remove(file_path)

                    messagebox.showinfo("Success", "File encrypted successfully")
                    self.clear_textboxes()
                except Exception as e:
                    messagebox.showerror("Error", f"Encryption failed: {str(e)}")
            else:
                # Create a new file path for the encrypted file
                encrypted_file_path = file_path + ".encrypted"

                # Hash the password
                hashed_password = hashlib.sha256(key.encode()).digest()

                with open(file_path, "rb") as file_in, open(encrypted_file_path,
"wb") as file_out:
                    while True:
                        # Read a chunk of data from the input file
                        chunk = file_in.read(1024)
                        if not chunk:
                            break
                        # Encrypted chunks
                        encrypted_chunk = bytearray() #create an array of bytes
                        for byte in chunk:
                            # XOR each byte of the data with a byte from the
hashed password
                            encrypted_byte = byte ^
hashed_password[len(encrypted_chunk) % len(hashed_password)]
                            encrypted_chunk.append(encrypted_byte)
                        # Write encrypted file to the output file.
                        file_out.write(bytes(encrypted_chunk))

                # Delete the original file
                os.remove(file_path)

                messagebox.showinfo("", "ENCRYPTED SUCCESSFULLY")
                self.clear_textboxes()

        else:
            messagebox.showwarning("Warning", "Please select a file and enter a
key")

    def decrypt_file(self):
        file_path = self.file_path_var.get()
        key = self.key_var.get()

        if file_path and key:
            if v.get() == 1:
                try:
                    # Read the contents of the encrypted file
                    with open(file_path, 'rb') as file:
                        ciphertext = file.read()
                    # Extract the IV and ciphertext from the file
                    iv = ciphertext[:8]
                    ciphertext = ciphertext[8:]

                    # Create a DES cipher object with the key and mode
                    cipher = DES.new(key.encode(), DES.MODE_CBC, iv)

                    # Decrypt the ciphertext
                    decrypted_text = cipher.decrypt(ciphertext)

                    # Unpad the decrypted text
                    unpadded_text = self.unpad(decrypted_text)

                    # Create a new file path for the decrypted file
                    decrypted_file_path = file_path[:-10]

                    # Write the decrypted text to the new file
                    with open(decrypted_file_path, 'wb') as file:
                        file.write(unpadded_text)

                    messagebox.showinfo("Success", "File decrypted successfully")
                    self.clear_textboxes()

                    # Delete the original file
                    os.remove(file_path)
                except Exception as e:
                    messagebox.showerror("Error", f"Decryption failed: {str(e)}")
            else:
                # Hash the password
                hashed_password = hashlib.sha256(key.encode()).digest()

                # Create a new file path for the decrypted file
```

```
                decrypted_file_path =
os.path.splitext(file_path)[0].replace(".encrypted", "")

                with open(file_path, "rb") as file_in, open(decrypted_file_path,
"wb") as file_out:
                    while True:
                        chunk = file_in.read(1024)
                        if not chunk:
                            break
                        decrypted_chunk = bytearray()
                        for byte in chunk:
                            decrypted_byte = byte ^
hashed_password[len(decrypted_chunk) % len(hashed_password)]
                            decrypted_chunk.append(decrypted_byte)
                        file_out.write(bytes(decrypted_chunk))

            # Delete the original file
            os.remove(file_path)

            messagebox.showinfo("", "DECRYPTED SUCCESSFULLY")
            self.clear_textboxes()

        else:
            messagebox.showwarning("Warning", "Please select a file and enter a
key")


    def pad(self, data):
        padding_size = DES.block_size - (len(data) % DES.block_size)
        padding = bytes([padding_size] * padding_size)
        return data + padding

    def unpad(self, data):
        padding_size = data[-1]
        return data[:-padding_size]

    def clear_textboxes(self):
        self.file_path_var.set("")
        self.key_var.set("")

if __name__ == "__main__":
    root = tk.Tk()
    app = EncryptionExample(root)
    app.create_gui()
    root.mainloop()
```

CODE DESCRIPTION

This is our Python program that provides a graphical user interface (GUI) to encrypt and decrypt files using either the Data Encryption Standard (DES) or a custom encryption algorithm based on a user-provided password. Here's an overview of the code:

The program imports the required modules, including tkinter for the GUI, filedialog and messagebox for file selection and error handling, and DES and get_random_bytes from Cryptodome.Cipher for the DES encryption.

A class Encryption is defined, which initializes the master GUI and sets the encryption key to None.

The **create_gui method** is defined to create the GUI elements, including labels, entries, radio buttons, and buttons. The GUI allows the user to select a file to encrypt/decrypt, enter an encryption key, choose an encryption algorithm, and perform the encryption/decryption operations.

The **browse_file method** is defined to open a file selection dialog when the "Browse" button is clicked. If a file is selected, its path is set to the file_path_var variable.

The **encrypt_file method** is defined to perform the encryption operation. It first retrieves the file path and encryption key from the GUI. If both are provided, the method checks the selected encryption algorithm.

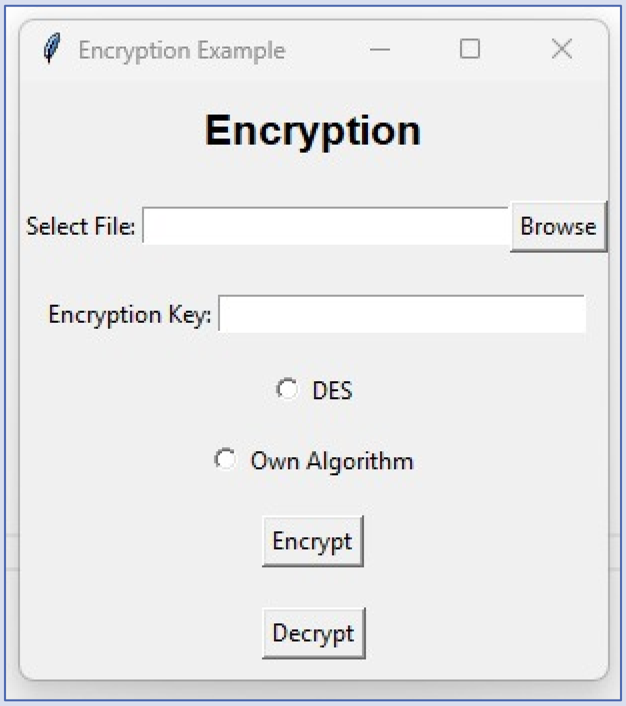
- If DES is selected, it reads the contents of the file, generates a random 8-byte initialization vector (IV), creates a DES cipher object with the key and mode, pads the plaintext to be a multiple of 8 bytes, encrypts the padded plaintext, creates a new file path for the encrypted file, writes the IV and ciphertext to the new file, deletes the original file, and shows a success message box.
- If the custom algorithm is selected, it hashes the password using SHA256, reads the input file and encrypts it byte by byte using the XOR operator with a byte from the hashed password, writes the encrypted file to the output file, deletes the original file, and shows a success message box.

The **decrypt_file method** is defined to perform the decryption operation. It retrieves the file path and encryption key from the GUI and checks the selected encryption algorithm.

- If DES is selected, it reads the contents of the encrypted file, extracts the IV and ciphertext, creates a DES cipher object with the key and mode, decrypts the ciphertext, removes the padding, creates a new file path for the decrypted file, writes the plaintext to the new file, deletes the original file, and shows a success message box.
- If the custom algorithm is selected, it reads the input file and decrypts it byte by byte using the XOR operator with a byte from the hashed password, writes the decrypted file to the output file, deletes the original file, and shows a success message box.

The pad method is defined to pad the plaintext with spaces to be a multiple of 8 bytes for DES encryption.

GUI



DEMONSTRATION VIDEO

Attached in zip file for guidance

BIBLIOGRAPHY

1. Simplilearn (2020). *What is DES? The Data Encryption Standard Explained*. [online] Simplilearn.com. Available at: https://www.simplilearn.com/what-is-des-article.
2. Geeksforgeeks (2021). *Advanced Encryption Standard (AES)*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/advanced-encryption-standard-aes/.
3. ak, A. (2016). *Lecture 8: AES: The Advanced Encryption Standard Lecture Notes on " Computer and Network Security "*. [online] Available at: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf.
4. www.rfwireless-world.com. (n.d.). *Advantages of AES | disadvantages of AES*. [online] Available at: https://www.rfwireless-world.com/Terminology/Advantages-and-disadvantages-of-AES.html#:~:text=Drawbacks%20or%20disadvantages%20of%20AES&text=%E2%9E%A8It%20uses%20too%20simple
5. T, N. (2020). *What is Data Encryption Standard (DES)? Definition, Working, Advantages and Disadvantages*. [online] Binary Terms. Available at: https://binaryterms.com/data-encryption-standard-des.html#:~:text=64%2Dbit%20ciphertext.- [Accessed 15 May 2023].
6. Miya, A. (2021). *What is DES Algorithm? - Use My Notes*. [online] Available at: https://usemynotes.com/what-is-des-algorithm/.