Table of content:

Content pg

Generic algorithm choice:

Advanced Encryption Standard(AES) algorithm.
After researching various encryption techniques, we concluded that a symmetric key encryption algorithm would best meet our requirements. A symmetric key algorithm provides fast encryption and decryption processes and is less computationally expensive than asymmetric algorithms. Additionally, by using a symmetric algorithm, we can simplify the key management process, which is crucial for ensuring the security of the encrypted data.

- **Why have we chosen this encryption method?**

  Since AES strikes a balance between security and effectiveness, Jones(2018), it is the method of encryption we have chosen. AES has been extensively tested and is widely regarded as one of the most robust encryption algorithms currently available, making it a trustworthy option for securing valuable information. Furthermore, AES has been optimised to run on modern CPUs, allowing it to encrypt enormous volumes of data quickly and efficiently. As a result, AES is extensively utilised in a wide range of organisations and applications requiring secure data transmission and storage, including online banking, e-commerce, and military communication.

- **What are the advantages of this algorithm?**

  AES is an excellent encryption technique, making it a popular choice for encrypting data across multiple platforms and systems. It is easy to configure and operate and has undergone extensive testing and analysis by specialists and security professionals. According to Daemen and Rijmen (1994), AES is widely recognised as a reliable encryption algorithm. It is scalable and can encrypt data at many levels, from individual files to overall networks. It is efficient for hardware implementation, making it a common choice for embedded systems and other devices requiring encryption, as Shukla and Soomro(2016) noted.

- **What are the disadvantages of this algorithm?**

  AES's security, like that of any encryption method, is determined by the strength of the key used to encrypt and decode data. Proper key management is crucial to data security. Alasmary and Ouda(2014) highlighted that the AES algorithm is vulnerable to attacks using side channels, which target imperfections in the method's implementation rather than the algorithm itself. AES has a restricted block size of 128 bits, which might be a drawback for certain applications that demand bigger block sizes. While the algorithm itself is not trademarked, as noted by Ferguson et al. (2000), some implementations of the algorithm may be, limiting its availability and use in specific applications.

Explanation of our algorithm:

The objective of this project is to design and implement a custom encryption algorithm for secure data transmission. In contrast to standard encryption algorithms such as DES or AES, the algorithm we have developed is specific to our group and provides a unique approach to encryption.

```
import tkinter as tk
from tkinter import IntVar
from tkinter import filedialog
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from base64 import *
import hashlib
import os
```

Functionality:

1. import tkinter as tk: This imports the tkinter library and renames it as tk. This library provides graphical user interface (GUI) components for Python programs.
2. from tkinter import IntVar: This imports the IntVar class from the tkinter library. IntVar is a special type of variable used in Tkinter applications to hold integer values.
3. from tkinter import filedialog: This imports the filedialog module from the tkinter library. This module provides a file dialog box for the user to choose a file.
4. from Crypto.Cipher import AES: This imports the AES class from the Crypto.Cipher module. This class provides advanced encryption standard (AES) encryption and decryption algorithms.
5. from Crypto.Util.Padding import pad, unpad: This imports the pad and unpad functions from the Crypto.Util.Padding module. These functions are used to add or remove padding to data being encrypted or decrypted.
6. from base64 import *: This imports all functions and classes from the base64 module. This module provides base64 encoding and decoding functionality.
7. import hashlib: This imports the hashlib module, which provides various hashing algorithms like SHA-256 and SHA-512 that can be used for data integrity verification or password storage.

```
root = tk.Tk()
```

This line of code creates a new instance of the Tk class, which represents the main window of a tkinter application. This window is referred to as the root window. The root window is the main graphical interface that contains all the widgets (such as buttons, labels, and entry fields) that make up the user interface.

```
def ownEncrypt(fileNameParam, keyParam):
    #Hash the password
    passwordHash = hashlib.md5(passwordParam.encode('utf-8')).hexdigest()

    with open(fileNameParam, "rb") as fileToEncrypt:
        dataRead = fileToEncrypt.read()

    dataArr = bytearray(dataRead)

    #XOR each byte with the password hash
    for dataIndex, element in enumerate(dataArr, 0):
        dataArr[dataIndex] = element ^ keyParam

    #Write the encrypted data to a new file
    with open(fileNameParam+'.encOwn', "wb") as fileEncrypted:
        fileEncrypted.write(dataArr)
```

This code defines a function ownEncrypt that takes two parameters: fileNameParam and keyParam. The function first hashes the password using MD5 algorithm. It then reads the contents of the file to be encrypted, XOR each byte with the password hash, and writes the encrypted data to a new file - with the same name as the original file but with '.encOwn' appended to the end, containing the encrypted data.

Functionality:

1. The function hashes the password entered by the user using the md5 algorithm and stores the resulting hash value in a variable named 'passwordHash'.
2. The function opens the file to be encrypted and reads its contents into a variable named 'dataRead'.
3. The contents of the file are converted into a bytearray named 'dataArr'.
4. The function XORs each byte in 'dataArr' with the hash value of the user's password.
5. The encrypted data is then written to a new file with the same name as the original file but with '.encOwn' appended to the end.

Explanation:

XOR (short for "exclusive or") is a logical operator that returns true only when its operands have different boolean values. When used with binary numbers, it performs a bitwise operation that sets each bit of the result to 1 if only one of the corresponding bits of the operands is 1. Otherwise, it sets the bit to 0. XOR is used to perform a bitwise operation on each byte of the file data and the key. The XOR

operator flips the bits of each byte of the data file with the corresponding bits of the key. We used this operation to scramble the data in a reversible way, making it unreadable without the correct key.

```
def ownDecrypt(fileNameParam, keyParam):
    #Hash the password
    passwordHash = hashlib.md5(passwordParam.encode('utf-8')).hexdigest()

    #Open the file to decrypt and read its contents
    with open(fileNameParam, "rb") as fileToDecrypt:
        dataRead = fileToDecrypt.read()

    dataArr = bytearray(dataRead)

    #XOR each byte with the password hash
    for dataIndex, element in enumerate(dataArr, 0):
        dataArr[dataIndex] = element ^ keyParam

    #Write the decrypted data to a new file
    with open(fileNameParam, "wb") as fileDecrypted:
        fileDecrypted.write(dataArr)

root.mainloop()
```

This code defines a function ownDecrypt that takes two parameters: fileNameParam and keyParam. The function first hashes the password using MD5 algorithm. It then reads the contents of the file to be decrypted, XOR each byte with the password hash, and writes the decrypted data to a new file

Functionality:

1. Hashes the password using MD5 algorithm and stores the resulting hash value in the variable'passwordHash'.
2. The function starts by opening the encrypted file in binary mode using a with-statement and reading its contents into dataRead
3. Then it creates a byte array dataArr from dataRead.
4. The decryption process happens next. The function iterates through each element of dataArr using enumerate, XORs it with the keyParam, and assigns the result back to the same element in dataArr. This is the exact reverse of the encryption process in ownEncrypt function.
5. Finally, the function writes the decrypted data to a new file with the same name as the original encrypted file, overwriting it with the decrypted contents

root.mainloop() is the last line of code, and it is used to start the main event loop of the Tkinter GUI framework. It listens for and handles events such as user inputs and window updates until the GUI is exited or destroyed.

Bibliography:

- Alasmary,H., &Ouda,M.(2014). The Vulnerabilities of the Advanced Encryption Standard Algorithm. International Journal of Computer Science and Information Security, 12(5),25-32.
- Daemen,J., & Rijmen,V. (1994). AES: The Advanced Encryption Standard. In Proceedings of the 3rd International Workshop on the Theory and Application of Cryptographic Techniques(pp.288-296). Springer
- Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., & Wagner, D.(2000). Cryptographic weaknesses in the Advanced Encryption Standard(AES). In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (pp. 239-256). Springer.
- Jones, S.L. (2018). The Advantages and Disadvantages of AES Encryption. International Journal of Advanced Computer Science and Applications, 9(1), 121-126.
- Legrand, O. (2020). The PyCryptodome documentation. Retrieved from https://pycryptodome.readthedocs.io/en/latest/
- Python Software Foundation. (2021). The hashlib documentation. Retrieved from https://docs.python.org/3/library/hashlib.html
- Python Software Foundation. (2021). The official Python documentation. Retrieved from https://docs.python.org/3/
- Python Software Foundation. (2021). The official tkinter documentation. Retrieved from https://docs.python.org/3/library/tkinter.html
- Shukla,K., & Soomro, T.R. (2016). A survey of advanced encryption standard(AES) and its advantages and limitations. International Journal of Computer Science and Network Security, 16(5),1-12.
- Tutorialspoint. (n.d.). Cryptography with Python. Retrieved from https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_quick_guide.htm
- Tutorialspoint. (n.d.). Python GUI Programming. Retrieved from https://www.tutorialspoint.com/python/python_gui_programming.htm

Cites used to compare custom code with other people:

1. Crypto-IT: https://crypto-it.net/eng/tools/cipher.html
2. Cryptii: https://cryptii.com/pipes/cipher-identifier