ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Môn học : An ninh máy tính

Báo cáo đồ án 1

Giảng viên hướng dẫn : Lê Giang Thanh

Lê Hà Minh

Thành viên nhóm : Nguyễn Thanh Phong - 20127274

Trần Thanh Hòa - 20127163

MỤC LỤC

I. Thông tin thành viên nhóm

A. Thành viên nhóm

Họ và tên	MSSV	Email
Nguyễn Thanh Phong	20127274	ntphong20@clc.fitus.edu.vn
Trần Thanh Hòa	20127163	tthoa20@clc.fitus.edu.vn

B. Phân công công việc

Họ và tên	Công việc	Ghi chú
Nguyễn Thanh Phong	- Mục A phần 1, 2, 3 - Mục B phần 2, 3, 8, 9, 10 - GUI 20%	**100%**
Trần Thanh Hòa	- Mục A phần 4, 5,6,7 - Mục B phần 1, 4, 5, 6, 7 - GUI 80%	**100%**

II. Các chức năng đã thực hiện

A. Xây dựng một module gồm các chức năng chính sau:

Công việc	Mức độ hoàn thành
1. Cho phép phát sinh một khoá bí mật Ks của thuật toán AES	**100%**
2. Mã hoá tập tin sử dụng thuật toán AES với khoá Ks	**100%**
3. Giải mã tập tin sử dụng thuật toán AES với khoá Ks	**100%**
4. Phát sinh một cặp khoá Kprivate và Kpublic của thuật toán RSA	**100%**
5. Mã hoá một chuỗi sử dụng thuật toán RSA sử dụng khoá Kpublic	**100%**
6. Giải mã một chuỗi sử dụng thuật toán RSA sử dụng khoá Kprivate	**100%**
7. Tính giá trị hash của một chuỗi sử dụng thuật toán SHA-1, SHA-256	**100%**
Tổng	**100%**

B. Xây dựng một ứng dụng để sử dụng các chức năng của module ở mục A, gồm các chức năng chính sau:

Công việc	Mức độ hoàn thành
1. Người dùng chọn tập tin cần mã hoá (tập tin P)	**100%**
2. Hệ thống phát sinh khoá bí mật Ks và mã hoá tập tin P thành tập tin C bằng thuật toán AES	**100%**
3. Hệ thống phát sinh cặp khoá Kprivate và Kpublic của thuật toán RSA và mã hoá khoá Ks bằng khoá Kpublic, output là chuỗi Kx.	**100%**
4. Hệ thống lưu lại chuỗi Kx kèm theo giá trị hash SHA-1 của Kprivate (gọi là HKprivate). Có thể xuất thành file C.metadata, với C là tên của tập tin C ở trên, cấu trúc tập tin là tuỳ chọn (XML, JSON, Plain text…).	**100%**
5. Hệ thống kết xuất khoá Kprivate cho người dùng (có thể xuất ra file).	**100%**
6. Người dùng chọn tập tin cần giải mã (tập tin C)	**100%**
7. Người dùng nhập khoá Kprivate (có thể chọn từ file)	**100%**
8. Hệ thống kiểm tra giá trị hash SHA-1 của Kprivate có trùng với HKprivate không?	**100%**
9. Giải mã chuỗi Kx để có được Ks dùng Kprivate.	**100%**
10. Dùng Ks giải mã tập tin C thành tập tin P.	**100%**
Tổng	**100%**

III. Giải thích sơ lược về code

A. AES

Đầu vào là 1 key ngẫu nhiên có giá trị 16 (ký tự) và chuỗi ký tự (tùy ý giá trị) sau đó cả 2 sẽ được chuyển thành chuỗi Bytes để thực hiện thao tác mã hóa giải mã
Chuỗi đầu vào sẽ được phân thành các khối 16 bytes sau đó sẽ thực hiện quy trình
Phải mở rộng key (expand key) từ key ban đầu thành 11 keys

Quá trình Encrypt:

- Đầu tiên add sub key vào plaintext

- Chạy vòng lặp từ 1 đến 9 thực hiện quy trình sub byte - shift row - mix column - add round key
- Vòng cuối cùng sẽ sub byte - shift row - add round key cuối cùng

```python
def enc(key, data):

    # Padd the data with \x00 and break it into blocks of 16
    pad = bytes(16 - len(data) % 16)

    if len(pad) != 16:
        data += pad
    grids = break_in_grids_of_16(data)

    # Expand the key for the multiple rounds
    expanded_key = expand_key(key, 11)

    # And apply the original key to the blocks before start the rounds
    # Work with integers from now
    temp_grids = []
    round_key = extract_key_for_round(expanded_key, 0)

    for grid in grids:
        temp_grids.append(add_sub_key(grid, round_key))

    grids = temp_grids

    # encrypt flow
    #
    for round in range(1, 10):
        temp_grids = []

        for grid in grids:
            sub_bytes_step = [[lookup(val) for val in row] for row in grid]
            shift_rows_step = [rotate_row_left(
                sub_bytes_step[i], i) for i in range(4)]
            mix_column_step = mix_columns(shift_rows_step)
            round_key = extract_key_for_round(expanded_key, round)
            add_sub_key_step = add_sub_key(mix_column_step, round_key)
            temp_grids.append(add_sub_key_step)

        grids = temp_grids

    # A final round without the mix columns
    temp_grids = []
    round_key = extract_key_for_round(expanded_key, 10)

    for grid in grids:
        sub_bytes_step = [[lookup(val) for val in row] for row in grid]
        shift_rows_step = [rotate_row_left(
            sub_bytes_step[i], i) for i in range(4)]
        add_sub_key_step = add_sub_key(shift_rows_step, round_key)
        temp_grids.append(add_sub_key_step)

    grids = temp_grids

    # Just need to recriate the data into a single stream before returning
    int_stream = []

    for grid in grids:
        for column in range(4):
            for row in range(4):
                int_stream.append(grid[row][column])

    return bytes(int_stream)
```

(Quá trình Encrypt)

Quá trình Decrypt :

Cũng tương tự quá trình Encrypt nhưng ngược lại

- Đầu tiên thực hiện vòng cuối cùng của quá trình Encrypt (sub byte - shift row - add round key cuối cùng)
- Chạy vòng lặp 9 về 1 thực hiện quy trình invert sub byte -invert shift row - invert mix column - add round key)

- Cuối cùng là đổi ngược quá trình add sub key vào plaintext

Sau mỗi quá trình Encrypt và Decrypt chuyển các khối 16 bytes thành chuỗi bytes như bình thường

B. RSA

Tạo khóa:

Bước 1: Chọn hai số nguyên tố lớn p và q.

Bước 2: Tính n = p * q, n là một số nguyên tố rất lớn.

Bước 3: Tính hàm Euler của n, ký hiệu là φ(n) hoặc phi(n), bằng (p - 1) * (q - 1).

Bước 4: Chọn một số nguyên e sao cho 1 < e < φ(n) và e là số nguyên tố cùng nhau với φ(n). e được chọn là khóa công khai (public key).

Bước 5: Tìm số nguyên d sao cho (d * e) % φ(n) = 1. d được chọn là khóa bí mật (private key).

Mã hóa:

```python
def rsa_encrypt_string(message, e, n):
    plaintext = [ord(char) for char in message]
    ciphertext = [str(pow(char, e, n)) for char in plaintext]
    return ' '.join(ciphertext)
```

Bước 1: Chọn một thông điệp cần được mã hóa dưới dạng một số nguyên m, m < n.

Bước 2: Tính c = (m^e) % n. Giá trị c là mã hóa của m và được gửi đi.

Giải mã:

```python
def rsa_decrypt_string(ciphertext, d, n):
    encrypted_values = ciphertext.split(' ')
    decrypted_values = [pow(int(char), d, n) for char in encrypted_values]
    plaintext = ''.join([chr(decrypted_char) for decrypted_char in decrypted_values])
    return plaintext
```

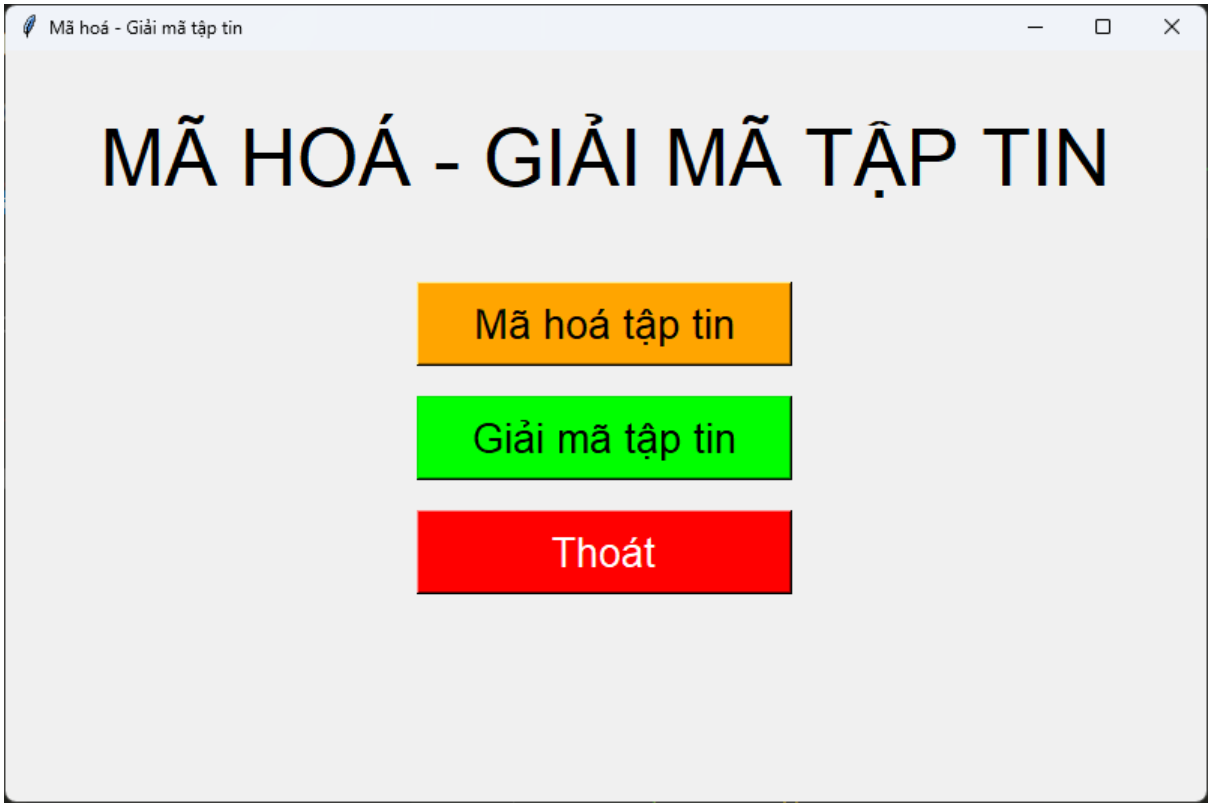
Bước 1: Nhận được mã hóa c.

Bước 2: Tính m = (c^d) % n. Giá trị m là thông điệp ban đầu trước khi được mã hóa

C. GUI

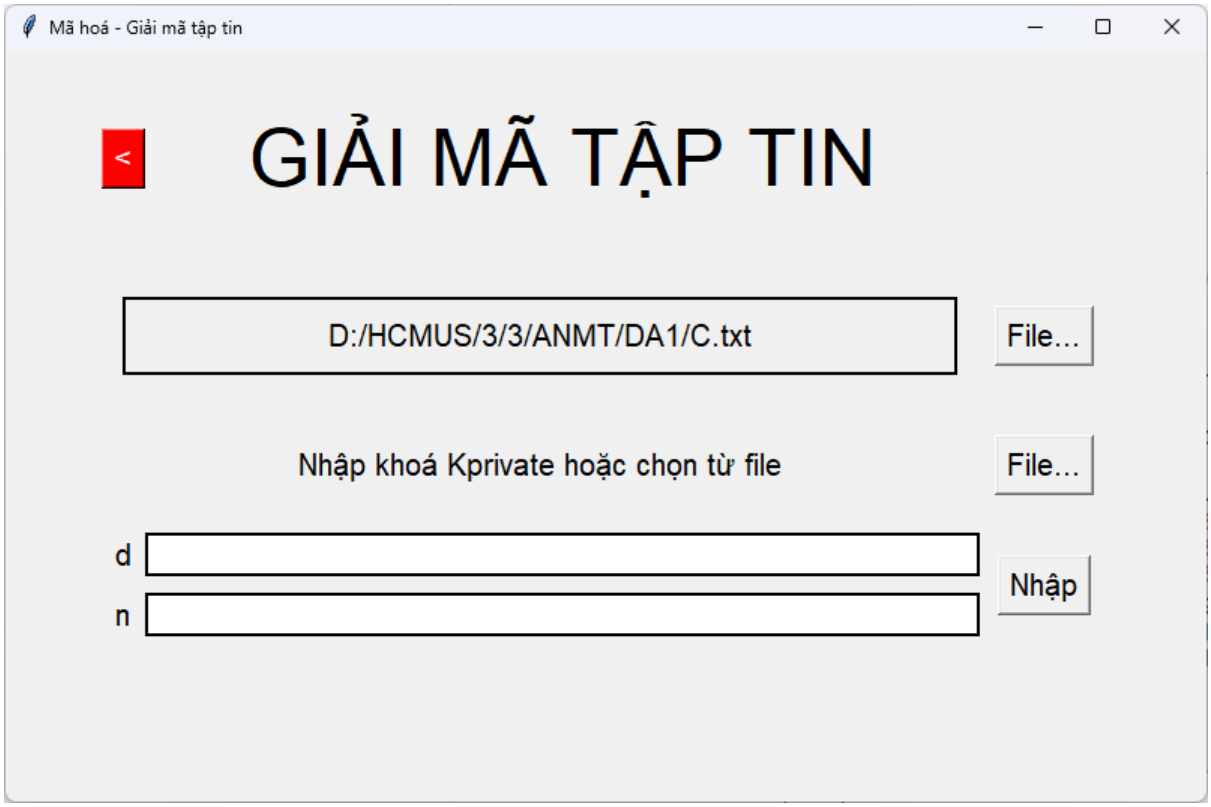
Thư viện sử dụng: Tkinter (giao diện), threading (đa luồng)

Giao diện chính:



Giao diện Mã hoá tập tin:

Giao diện Giải mã tập tin:

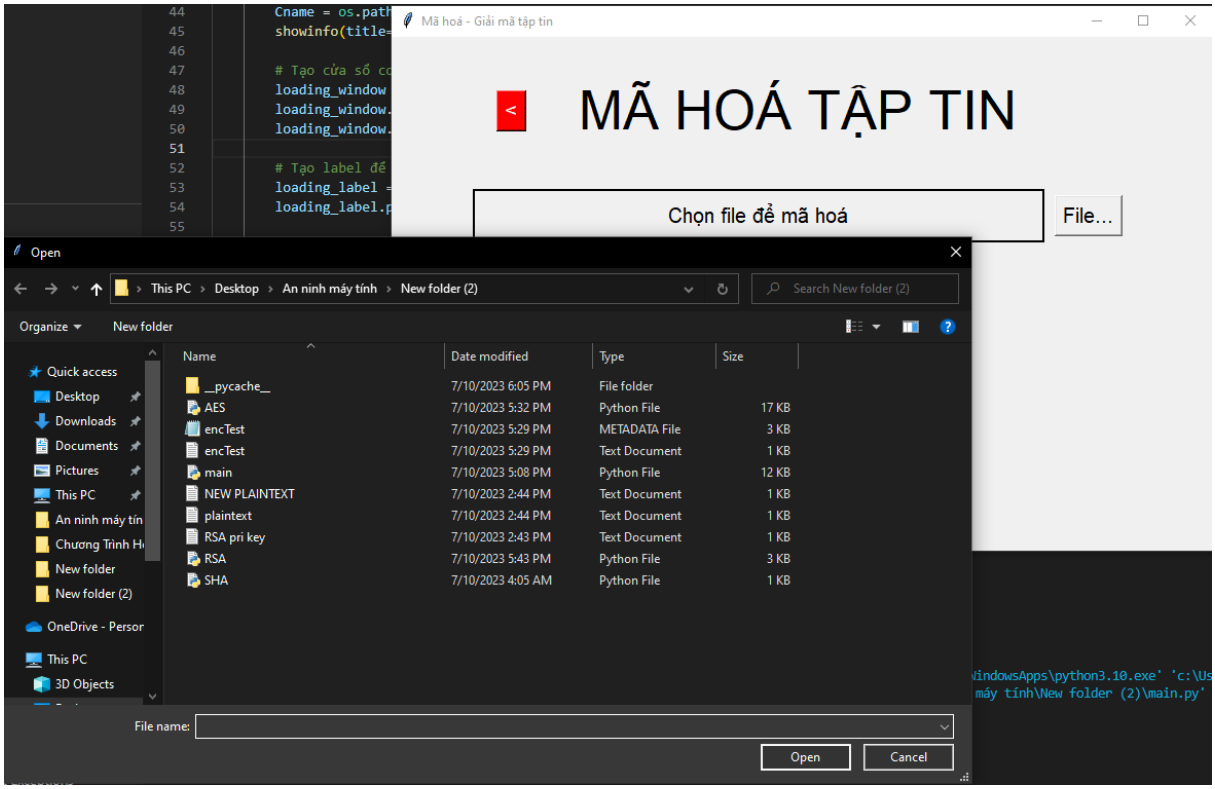


D. Xử lý các yêu cầu của đề bài

1. Người dùng chọn tập tin cần mã hoá (tập tin P)

Đoạn code này sẽ hiển thị giao diện để cho người dùng chọn file để mã hóa và đây là giao diện của người dùng khi chọn file.

```
def read_file_P():
    # Hiển thị hộp thoại để chọn tệp
    global P
    P = askopenfilename()
    if P:
        encrypt_display_lable1.config(text=P)
        encrypt_display_lable2.grid(row=2, column=0, columnspan=2, pady=0)
        encrypt_button3.grid(row=2, column=2, pady=10, padx=10)
    else:
        encrypt_display_lable1.config(text="Chọn file để mã hoá")
        showinfo(title="Chọn file để mã hoá", message="Chọn lại file")
```



2. Hệ thống phát sinh khoá bí mật Ks và mã hoá tập tin P thành tập tin C bằng thuật toán AES

Sau khi chọn file thành công, code này sẽ đọc, lưu giá trị dữ liệu để tiến hành mã hóa theo các bước đã nêu ở phần AES sau đó là viết vào file mã hóa

```
def encrypt_file(plantext_file_name,ciphertext_file_name,byte_key):
    # ghi chuỗi từ file
    with open(plantext_file_name, 'r') as file_plaintext:
        plain_text = file_plaintext.read()
    byte_plaintext = string_to_bytes(plain_text)# trans to byte string
    encrypted_text = enc(byte_key, byte_plaintext)

    with open(ciphertext_file_name, 'wb') as file_ciphertext:
        file_ciphertext.write(encrypted_text)
```

3. Hệ thống phát sinh cặp khoá Kprivate và Kpublic của thuật toán RSA và mã hoá khoá Ks bằng khoá Kpublic, output là chuỗi Kx.

Hàm gen_rsa_keypair() sẽ tạo 2 cặp khóa Kprivate và Kpublic, Ks sẽ được mã hóa thành Kx trong hàm rsa_encrypt_string()

```
58          Kprivate, Kpublic = RSA.gen_rsa_keypair()
59          n = Kprivate[1]
60          Kx = RSA.rsa_encrypt_string(Ks, Kpublic[0], Kpublic[1])
```

4. Hệ thống lưu lại chuỗi Kx kèm theo giá trị hash SHA-1 của Kprivate (gọi là HKprivate). Có thể xuất thành file C.metadata, với C là tên của tập tin C ở trên, cấu trúc tập tin là tuỳ chọn (XML, JSON, Plain text…).

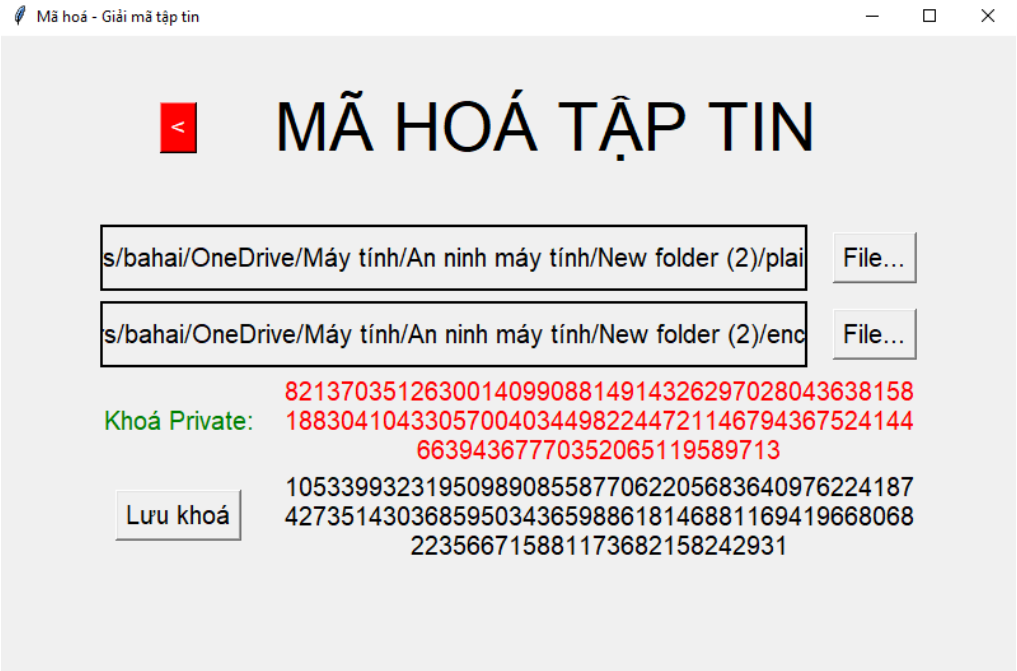
Tiến hành hash giá trị d trong Kprivate để có được HKprivate, lưu HKprivate và Kx thành file.metadate với Kx ở trên mà HKprivate ở dưới

```
def gen_key_encrypt():
    global Kprivate
    Kprivate, Kpublic = RSA.gen_rsa_keypair()
    n = Kprivate[1]
    Kx = RSA.rsa_encrypt_string(Ks, Kpublic[0], Kpublic[1])
    HKprivate = SHA.sha1_hash_string(str(Kprivate[0]))
    with open(Cname + ".metadata", "w") as file:
        file.write(Kx)
        file.write('\n')
        file.write(HKprivate)
    file.close()
```

5. Hệ thống kết xuất khoá Kprivate cho người dùng (có thể xuất ra file).

GUI sẽ hiện đường dẫn để lưu Kprivate và cho người dùng chọn file để lưu sau khi bấm vào “Lưu Khóa”.

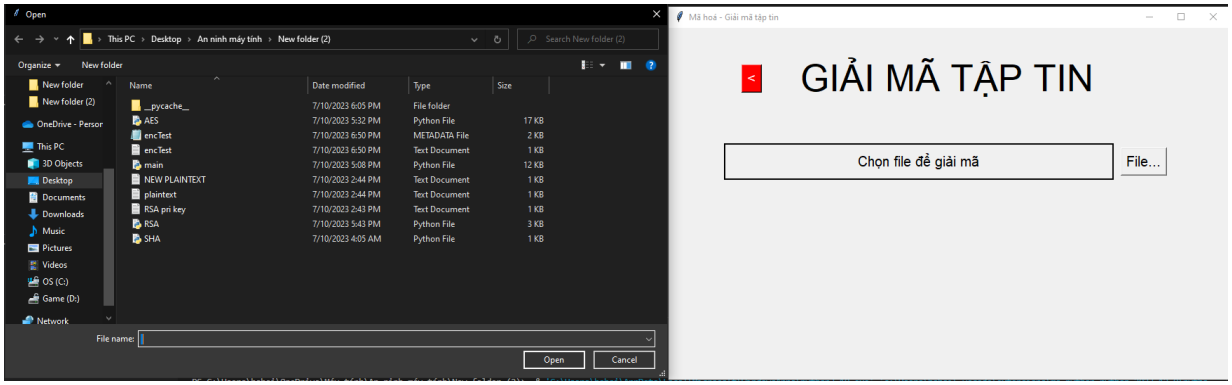
```
def write_Kprivate():
    pri_file = asksaveasfilename()
    if pri_file:
        with open(pri_file, "w") as file:
            file.write(str(Kprivate[0]))
            file.write("\n")
            file.write(str(Kprivate[1]))
        file.close()
        showinfo(title="Lưu khoá Kprivate", message="Lưu thành công")
    else:
        showinfo(title="Lưu khoá Kprivate", message="Lưu không thành công")
```



(ảnh giao diện)

6. Người dùng chọn tập tin cần giải mã (tập tin C)

```
def read_file_C():
    global C, Cname
    C = askopenfilename()
    Cname = os.path.splitext(os.path.basename(C))[0]
    if C:
        decrypt_display_lable1.config(text=C)
        decrypt_lable2.grid(row=2, column=0, columnspan=2, pady=0)
        decrypt_button3.grid(row=2, column=2, pady=20, padx=10)
        decrypt_display_lable3.grid(row=3, column=0, pady=0)
        decrypt_display_lable4.grid(row=4, column=0, pady=0)
        entry1.grid(row=3, column=1, pady=2)
        entry2.grid(row=4, column=1, pady=2)
        decrypt_button4.grid(row=3, rowspan=2, column=2, pady=20, padx=10)
    else:
        encrypt_display_lable1.config(text="Chọn file để giải mã")
        showinfo(title="Chọn file để mã hoá", message="Chọn lại file")
```

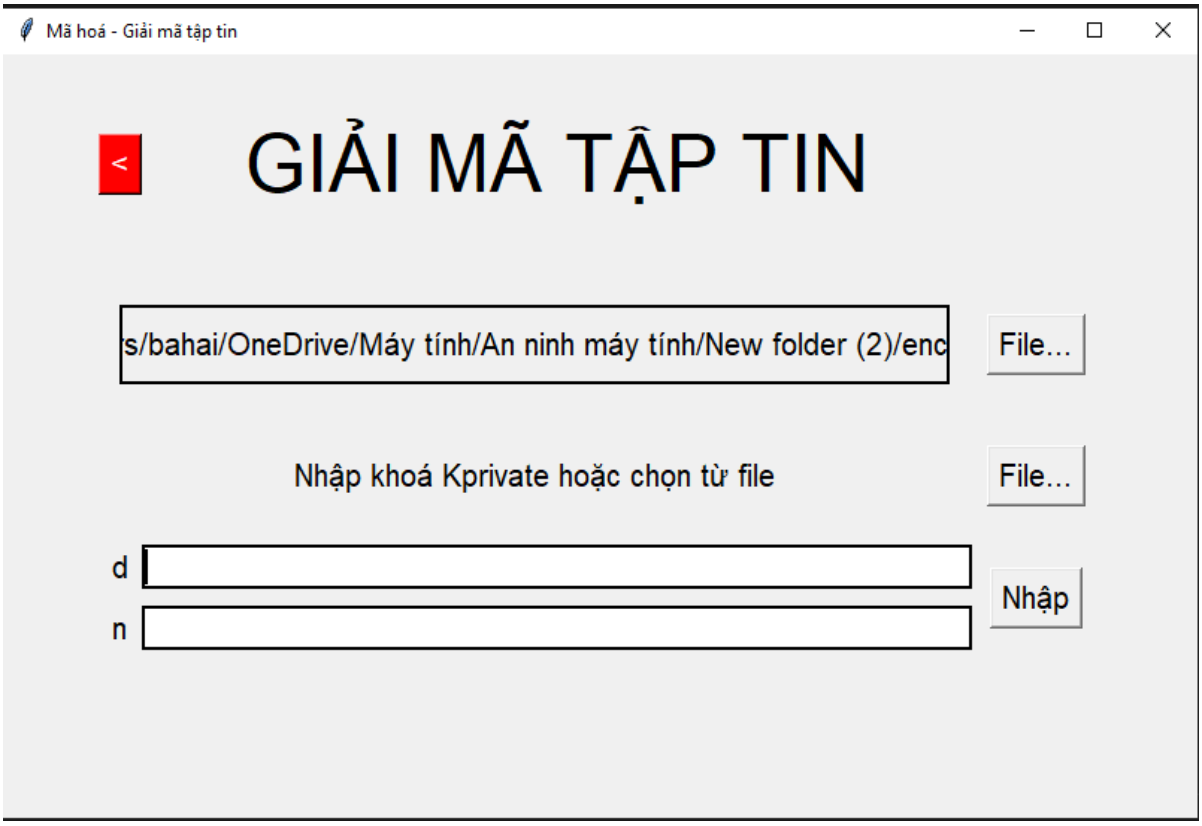


7. Người dùng nhập khoá Kprivate (có thể chọn từ file)

Giao diện yêu cầu chọn file hoặc nhập d, n từ bàn phím và sẽ kiểm tra khóa hợp lệ hay không

```
def read_Kprivate():
    global Kprivate
    pri_file = askopenfilename()
    if pri_file:
        with open(pri_file, "r") as file:
            d = int(file.readline())
            n = int(file.readline())
            Kprivate = (d, n)
            write_file_P()
        file.close()
    else:
        showinfo(title="Đọc khoá Kprivate", message="Đọc không thành công")

def check_input():
    global Kprivate
    if entry1.get().isdigit():
        if entry2.get().isdigit():
            Kprivate = (entry1.get(), entry2.get())
            write_file_P()
        else:
            showinfo(title="Nhập key", message="Hãy nhập số n")
    else:
        showinfo(title="Nhập key", message="Hãy nhập số d")
```



8. Hệ thống kiểm tra giá trị hash SHA-1 của Kprivate có trùng với HKprivate không?

Đọc file chứa Kx và HKprivate đã làm ở phần 4. sau đó kiểm tra HKprivate và giá trị hash mới có giống nhau hay không

```
136     metadata_file = Cname + ".metadata"
137     if metadata_file:
138         with open(metadata_file, "r") as file:
139             Kx = file.readline()
140             HKprivate = file.readline()
141         file.close()
142         Kprivate_hash = SHA.sha1_hash_string(str(Kprivate[0]))
143         if Kprivate_hash != HKprivate:
144             showinfo(title="Giải mã tập tin", message="Giải mã thất bại")
```

9. Giải mã chuỗi Kx để có được Ks dùng Kprivate.

Sử dụng hàm RSA decrypt để giải mã chuỗi Kx thành Ks dùng Kprivate

```
Ks = RSA.rsa_decrypt_string(Kx, Kprivate[0], Kprivate[1])
```

10. Dùng Ks giải mã tập tin C thành tập tin P.

Sau khi kiểm tra giá trị hash SHA-1 của Kprivate có trùng với Hkprivate thì thực hiện giải mã tập tin C đã chọn thành tập tin P bằng thuật toán AES decrypt.

```
145     else:
146         Ks = RSA.rsa_decrypt_string(Kx, Kprivate[0], Kprivate[1])
147         byte_key = AES.string_to_bytes(Ks)
148         showinfo(title="Giải mã tập tin", message="Giải mã thành công\nHãy lưu file giải mã")
149         P = asksaveasfilename()
150         AES.decrypt_file(C, P, byte_key)
151         decrypt_to_main()
```

IV. Tham khảo

https://medium.com/wearesinch/building-aes-128-from-the-ground-up-with-python-8122af44ebf9

https://vi.wikipedia.org/wiki/Advanced_Encryption_Standard