

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Khoa Công Nghệ Thông Tin

-----o0o-----



MÔN HỌC: An Ninh Máy Tính

Giáo viên hướng dẫn:

ThS.Lê Giang Thanh

ThS.Lê Hà Minh

BÁO CÁO ĐỒ ÁN.

Nhóm thực hiện:

Lê Tiến Hùng - 19127412

Trần Nam Khánh - 19127441

Phạm Duy Tiến - 19127577

TP. HỒ CHÍ MINH, NGÀY 03 THÁNG 07 NĂM 2022

MỤC LỤC

I.Phân công công việc:	2
II.Chức Năng	2
1. Đăng ký tài khoản người dùng	2
2. Phát sinh cặp khóa bất đối xứng	5
3. Cập nhật thông tin tài khoản	5
4. Mã hoá tập tin	6
5. Giải mã tập tin	7
6. Ký trên tập tin.	8
7. Xác nhận chữ ký trên tập tin.	10
III.Các vấn đề, giải pháp	12

I. Phân công công việc:

Tên	Mục đã làm	Phần trăm
Lê Tiến Hùng	Mục 1,2,3	100%
Phạm Duy Tiến	Mục 4, 5	100%
Trần Nam Khánh	Mục 6, 7	100%

II. Chức Năng

1. Đăng ký tài khoản người dùng

- Hàm genSalt(): tạo salt và tạo password kết hợp với satl

```
def genSalt(password, satl=None):  
    if satl is None:  
        ALPHABET = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"  
        chars = []  
        for i in range(16):  
            chars.append(random.choice(ALPHABET))  
        satlN = "".join(chars)  
        return password + '_ANMT_' + satlN, satlN  
    return password + '_ANMT_' + satl, satl
```

- Hàm insertUser(): thêm thông user vào database

```
29 def insertUser(email, name, dOBird, address, phone, passwordHash, privateKey, nonce, publicKey, salt):  
30     inserts = f"EXEC SP_INS_USER '{email}', '{name}', '{dOBird}', '{address}', {phone}, {passwordHash}, {privat  
31     insert = connect.cursor()  
32     insert.execute(inserts)  
33     insert.commit()  
34     insert.close()  
35
```

_ Hàm genKeyPair(): sinh khóa public Key và Lấy 16 byte từ password để mã hóa private key

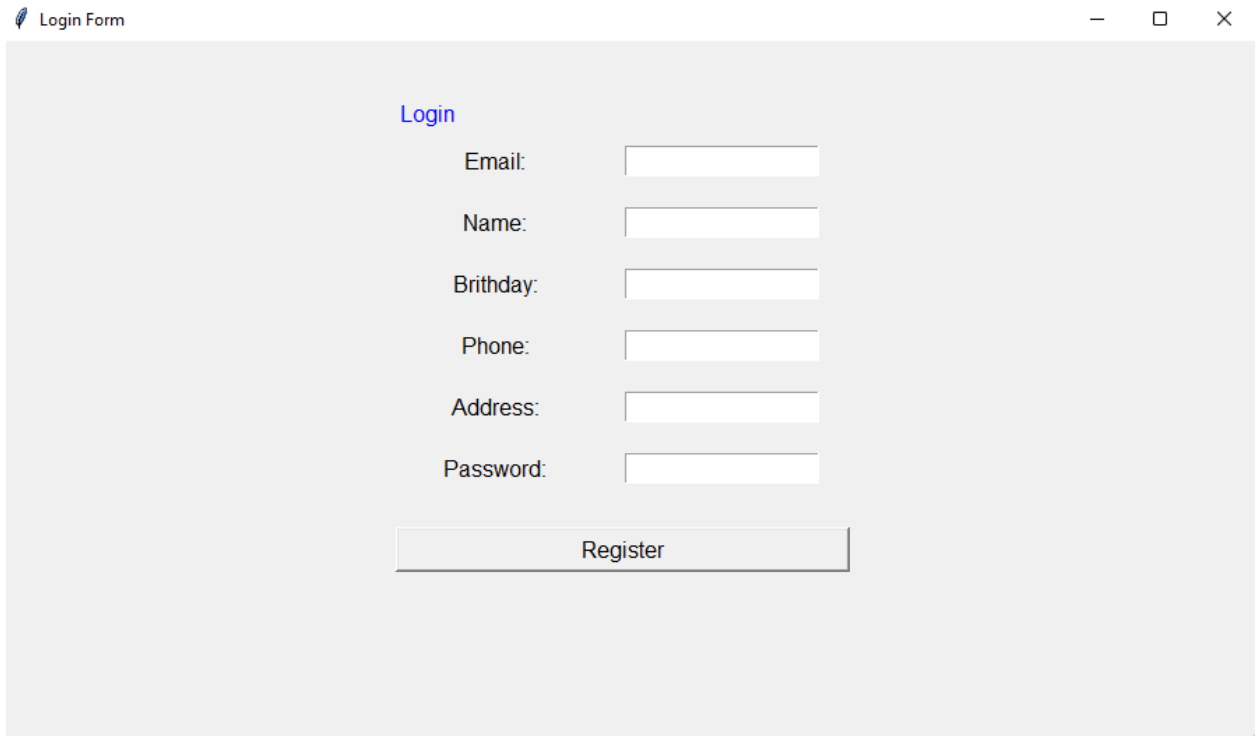
```
def genKeyPair(size, passphrase):  
    keypair = RSA.generate(size)  
    passphrase = passphrase + "0123456789101112"  
    passphrase = passphrase[:16]  
    cipher = AES.new(passphrase.encode('utf-8'), AES.MODE_EAX)  
    nonce = cipher.nonce  
    ciphertext, tag = cipher.encrypt_and_digest(keypair.export_key())  
    return keypair.public_key().export_key(), ciphertext, nonce
```

- Hàm đăng ký

```
7 def signUp(email, name, dOBird, address, phone, password):
8     passwordSalt, salt = gen_key.genSalt(password)
9     passwordHash = hex(int(SHA256.new(passwordSalt.encode('utf-8')).hexdigest(), 16))
10    publicKey, priKey, nonce = gen_key.genKeyPair(1024, password)
11    # convert to hex
12    publicKey = '0x' + publicKey.hex()
13    priKey = '0x' + priKey.hex()
14    nonce = '0x' + nonce.hex()
15    db.insertUser(email, name, dOBird, address, phone,
16                  passwordHash, priKey, nonce, publicKey, salt)
17
```

- + Dòng 8: tạo passwordSalt và salt.
- + Dòng 9: hash 16 byte passwordSalt làm passwordHash.
- + Dòng 10: tạo public key và private key đã được mã hóa bằng thuật toán AES.
- + Dòng 15: insert thông tin lên database.

- UI:



Login Form

Login

Email:

Name:

Brithday:

Phone:

Address:

Password:

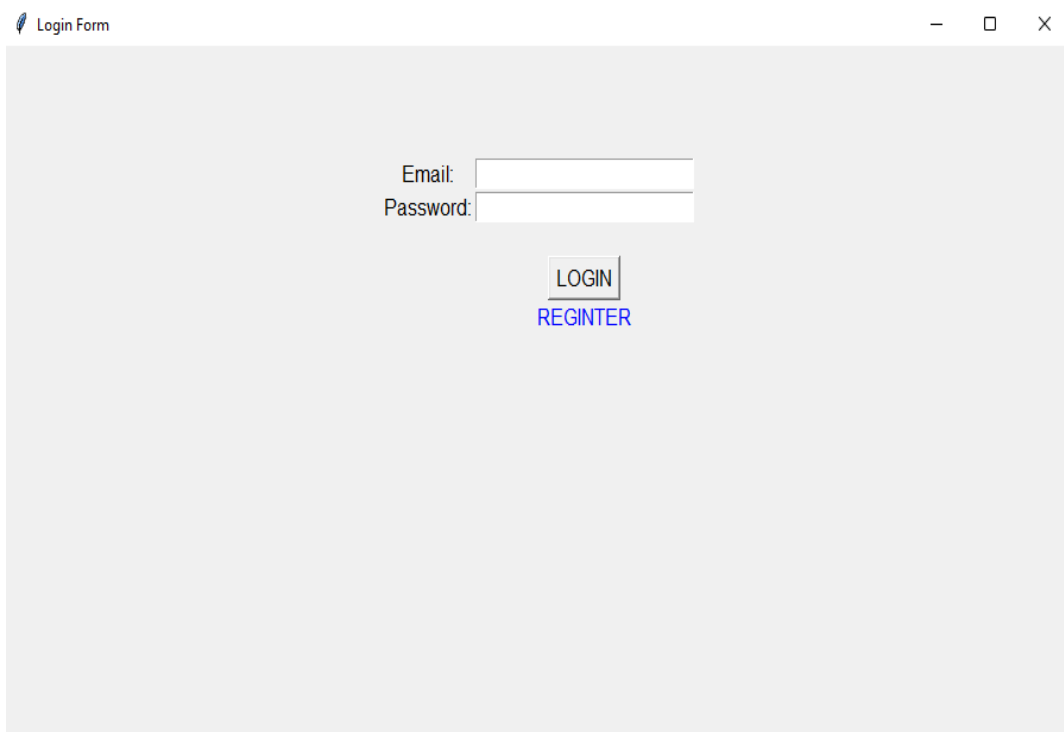
- Đăng nhập

```
13 def checkUserPassword(email, password):
14     passwordSalt, salt = gen_key.genSalt(password)
15     passwordHash = hex(int(SHA256.new(passwordSalt.encode('utf-8')).hexdigest(), 16))
16     print(passwordHash)
17     query = f"SELECT * FROM USERINFO WHERE EMAIL = '{email}' AND MATKHAU = {passwordHash}"
18     exec = connect.cursor()
19     exec.execute(query)
20     if exec.rowcount != 0:
21         data = exec.fetchall()
22         exec.close()
23         return True, data
24     else:
25         exec.close()
26         return False, ""
27
```

+ Dòng 15: Hash lại password

+ Dòng 17 đến 16: so sánh email và password hash có đúng trong database không.

Nếu có trả về data người dùng và đăng nhập thành công.



The screenshot shows a web browser window titled "Login Form". Inside the window, there is a login form with two input fields: "Email:" and "Password:". Below the "Password:" field, there are two buttons: "LOGIN" and "REGISTER". The "REGISTER" button is highlighted in blue.

2. Phát sinh cặp khóa bất đối xứng

```
6 def genKeyPair(size, passphrase):
7     keypair = RSA.generate(size)
8     passphrase = passphrase + "0123456789101112"
9     passphrase = passphrase[:16]
10    cipher = AES.new(passphrase.encode('utf-8'), AES.MODE_EAX)
11    nonce = cipher.nonce
12    ciphertext, tag = cipher.encrypt_and_digest(keypair.export_key())
13    return keypair.public_key().export_key(), ciphertext, nonce
```

Dòng 7: Sinh khóa public key và private key kích thước Size = 2048 bits.

Dòng 8: padding password.

Dòng 9: lấy 16 byte đầu của password làm key session.

Dòng 12: mã hóa private key bằng AES với key session.

3. Cập nhật thông tin tài khoản

- Hàm handleNewPass(): giải mã mã private bằng password cũ và mã hóa lại private key bằng password mới

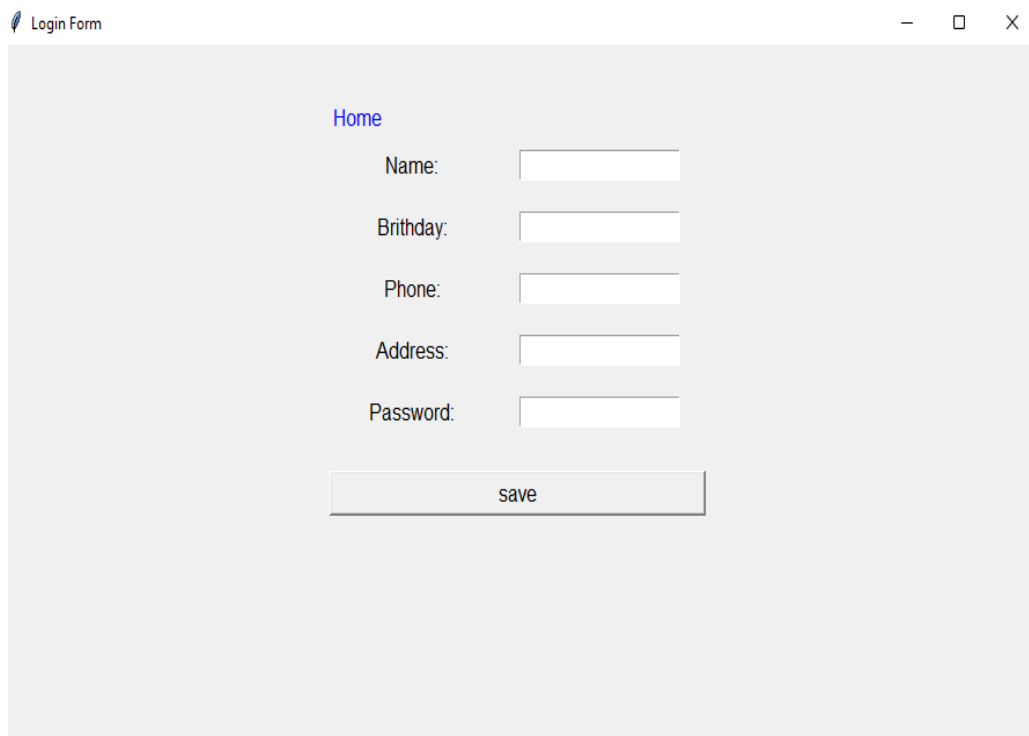
```
30 def handleNewPass(ciphertext, oldpassphare, oldnonce, newpassphare):
31     prvateKey = decryptKey(ciphertext,oldpassphare,oldnonce)
32     return encryptKey(prvateKey,newpassphare)
33
```

- updataInfo():

```
25
26 def updateInfo(data,email, name, dOBird, address, phone,oldPass , newPass):
27     password_salt, salt = gen_key.genSalt(newPass, satl=data[0][9])
28     # hashed_string = hex(int(hashlib.sha256("hung".encode('utf-8')).hexdigest(), 16))
29     password_hash = hex(int(SHA256.new(password_salt.encode('utf-8')).hexdigest(), 16))
30     oldKeyCipher = data[0][6]
31     oldNonce = data[0][7]
32     privateK, nonceK = gen_key.handleNewPass(oldKeyCipher, oldPass, oldNonce, newPass)
33     db.updateUserData(email, name, dOBird, address,
34         phone, password_hash, '0x' + privateK.hex(), '0x' + nonceK.hex())
35
```

Dòng 33: update thông tin lại database.

- UI:



Home

Name:

Brithday:

Phone:

Address:

Password:

4. Mã hoá tập tin

```
7 def encry_file(filename ,publicKey):
8     #gen Ksession
9     Ksession = gk.genKsession()
10    with open(filename, "rb") as f:
11        data = f.read()
12    ciphertext_data, nonce = gk.encryptKey(data,Ksession)
13    pub_key = publicKey
14    encry_Ksession = rsa.encrypt(str(Ksession).encode('ascii'),pub_key)
15    encry_data_and_key = {"data":ciphertext_data, "Ksession": encry_Ksession, "nonce":nonce}
16
17    return str(encry_data_and_key)
```

Dòng: 7: cho phép sinh một Key Session để mã hóa tập tin.

Dòng 10,11: đọc tập tin dưới dạng binary.

Dòng 12: sử thuật toán mã hóa AES để mã hóa tin.

Dòng 14: Sử dụng Public key để mã hóa Key Session

Dòng 15: cấu trúc tập tin theo dạng Dictionary và return về string Dictionary

Login Form

khanh

[EDIT INFORMATION](#)

[DIGITAL SIGNATURE & VERIFY](#)

To Email:

Open File

SEND

MY FILE

input1.jpg
input2.txt

OPEN

- Chọn email của người nhận và mở file cần chia sẻ sau chọn SEND sẽ lưu data vào database của người người nhận.

5. Giải mã tập tin

```

20 def decry_file(cipherFile, Kpri):
21     data_and_key = ast.literal_eval(cipherFile)
22     Ksession = cr.decry_RSA(data_and_key["Ksession"],Kpri)
23     Nonce = data_and_key["nonce"]
24     return gk.decryptKey(data_and_key["data"],Ksession,Nonce)

```

- Dòng 21: chuyển string Dictionary thành Dictionary
- Dòng 21: Sử dụng private key để giải mã Key session
- Dòng 24 return tập tin được giải mã



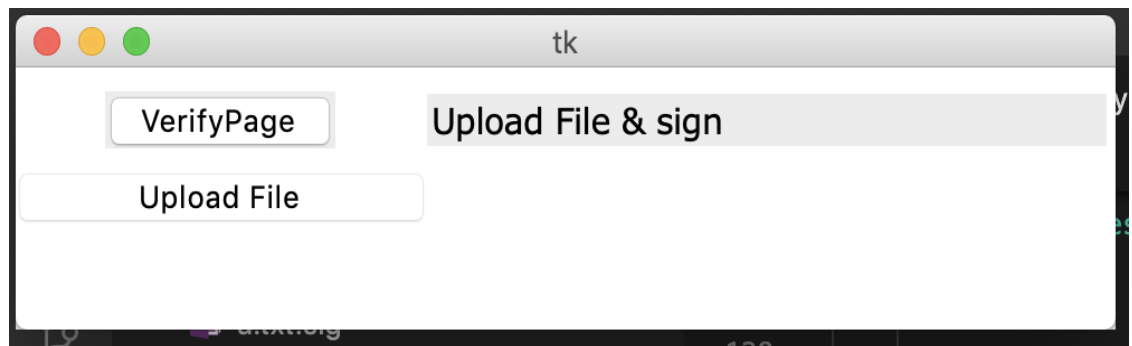
- Chọn tên file cần mở sau đó chọn open, file được giải mã từ function giải mã file sau đó sẽ lưu vào thư mục của người dùng.

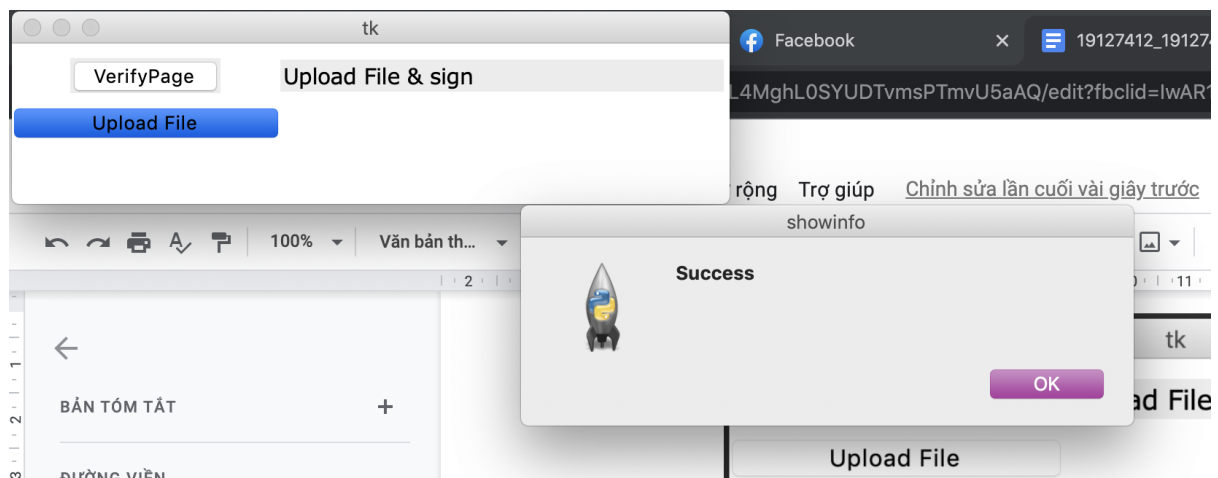
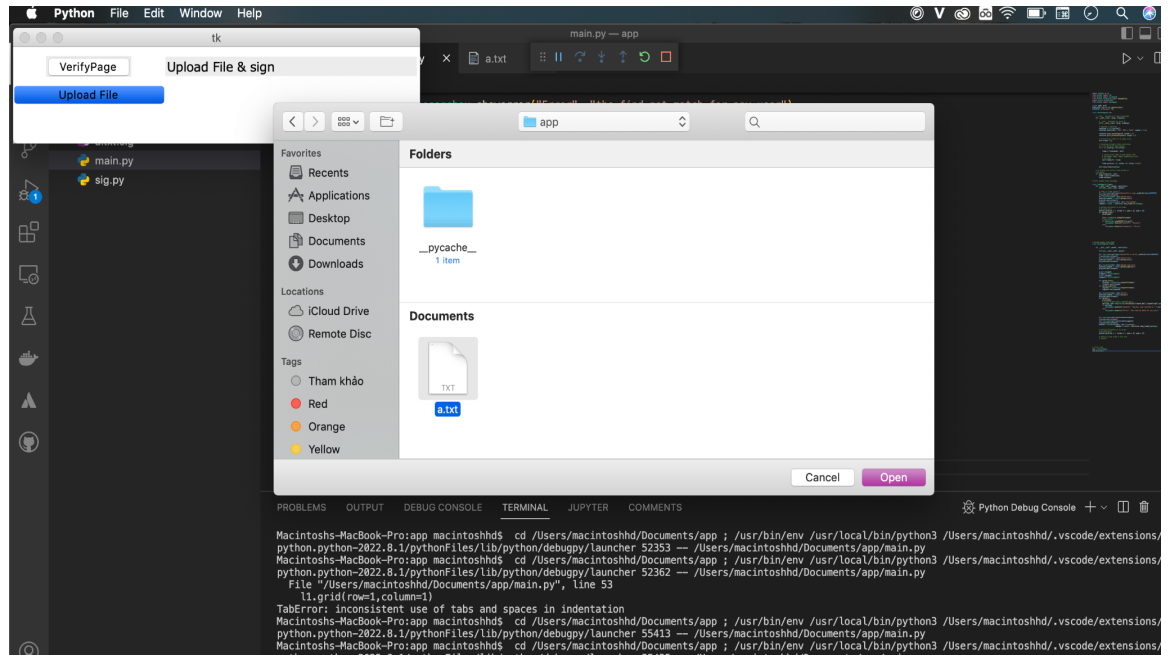
6. Ký trên tập tin.

- Đọc tập tin lựa chọn với dạng base64 encoded.
- Thư viện rsa có hỗ trợ hàm kí và xác thực bằng các hàm băm khác nhau. Em sẽ sử dụng hàm `rsa.verify(param1, param2, param3)` để kí. Với `param1` là 1 đoạn string đã được encoded, `param2` là `privateKey`, `param3` là thuật toán hash sử dụng.
- Sau đó lưu kết quả vào file với đuôi là `.sig`

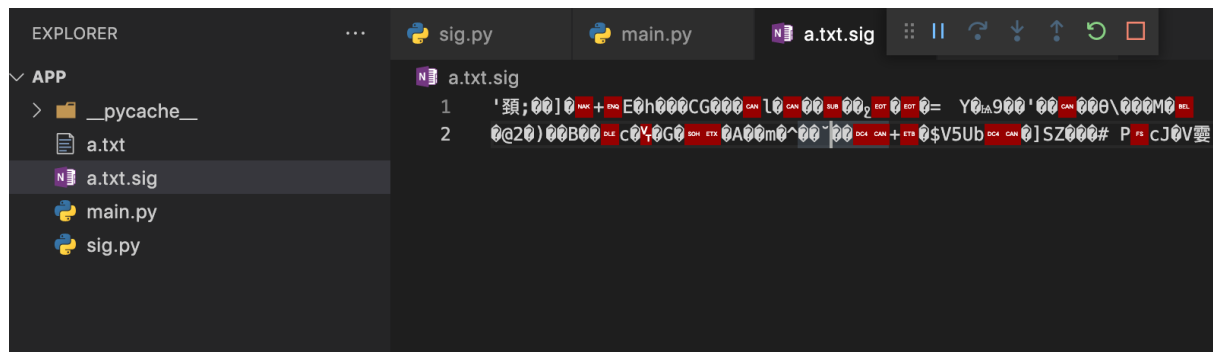
```
def signSHA256(filepath,privateKey):
    signFile = filepath.split('/')[-1]
    outputPath = filepath.replace(filepath.split('/')[-1],signFile+'.sig')
    # print(outputPath)
    try:
        with open(filepath, "rb") as image_file:
            encoded_string = base64.b64encode(image_file.read())
            sign = rsa.sign(encoded_string, privateKey, 'SHA-256')
            print('sign',sign)
            file = open(outputPath,'wb')
            file.write(sign)
            file.close()
        return True
    except:
        return False
```

Demo:





Kết quả: sẽ có 1 file xuất hiện ngay thư mục của app



7. Xác nhận chữ ký trên tập tin.

- Ta sẽ đưa vào 2 tập tin. Một là tập tin gốc, hai là tập tin kí.

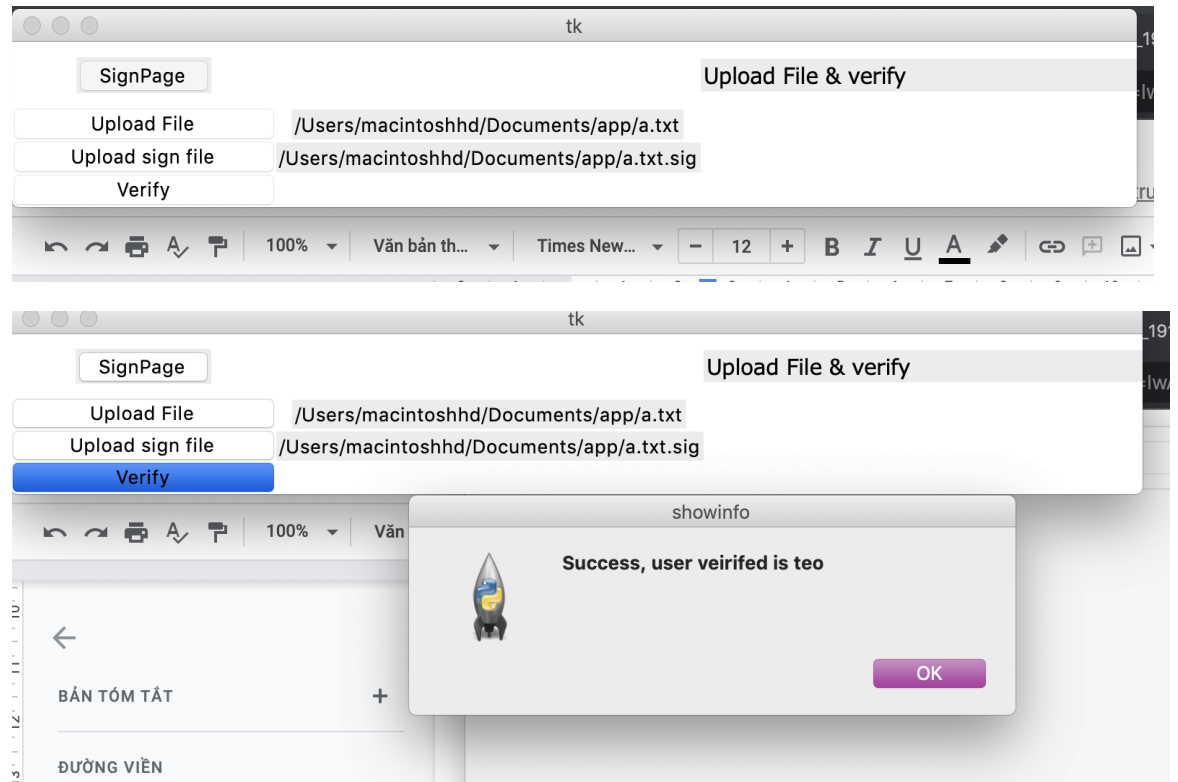
- Tập tin gốc sẽ được đọc dưới dạng base64, còn chữ kí sẽ được đọc theo dạng byte.
- Sử dụng hàm `rsa.verify(pa1,pa2,pa3)` để xác thực. Với `pa1` encoded của tập tin gốc, `pa2` là bytearray của file chữ kí, `pa3` là `publicKey`. Hàm sẽ trả về hàm hash được sử dụng nếu verify thành công, nếu sai sẽ throw lỗi không xác thực.
- Ở đây để demo, em sẽ sử dụng 1 dict gồm các user được hardcode với các `publicKey` khác nhau. Hàm sẽ trả về `True` và tên user nếu đúng và ngược lại `False`, '' nếu sai.

```

27 def verifySign(filepath,signpath,publicKey):
28     print(publicKey)
29     # print('p',filepath)
30     # print('s',signpath)
31     p1,k1 = generate_key()
32     p2,k2 = generate_key()
33     p3,k3 = generate_key()
34     p4,k4 = generate_key()
35     list_user = {
36         'user1' : p1,
37         'user2' : p2,
38         'user3' : p3,
39         'teo' : publicKey
40     }
41
42     with open(filepath, "rb") as image_file:
43         encoded_string = base64.b64encode(image_file.read())
44     file = open(signpath,'rb')
45     sign = file.read()
46     # print('sign',sign)
47     for user in list_user:
48         # print(user,user.values())
49         try:
50             rsa.verify(encoded_string, sign,list_user[user])
51             return True, user
52         except:
53             continue
54     return False, ''

```

Demo:



III. Các vấn đề, giải pháp

- **Vấn đề:** Nhóm do chưa làm việc với nhau nên còn gặp các vấn đề như chưa lên được kế chi tiết và làm timeline công việc cho các thành viên và trong quá trình ghép code còn do đặt tên hàm chưa rõ khiến cho việc ghép code và sửa lỗi khó khăn.
- **Giải pháp:** các nhóm họp và làm việc cùng trong quá trình kiến ghép code, cùng nhau đưa ra ý đóng góp cho nhau để hoàn thành công việc