TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN

ASSIGNMENT #01.04

Bộ môn: An toàn và phục hồi dữ liệu

Sinh viên thực hiện:
20120576 – Nguyễn Bửu Thạch
21120419 – Vũ Thành Công

Thành phố Hồ Chí Minh, tháng 10/2024

Mục lục

I. Mã đối xứng AFFINE

1. Về thuật toán Affine

Mã Affine là một loại mã thay thế đơn ký tự, trong đó mỗi chữ cái trong bảng chữ cái được ánh xạ thành số tương ứng, được mã hóa bằng một hàm toán học đơn giản và được chuyển đổi trở lại thành một chữ cái. Công thức được sử dụng là mỗi chữ cái được mã hóa thành một chữ cái khác và ngược lại.

Toàn bộ quá trình dựa trên hoạt động modulo m (độ dài của bảng chữ cái được sử dụng). Trong mã affine, các chữ cái của bảng chữ cái có kích thước m trước tiên được ánh xạ thành các số nguyên trong phạm vi 0 … m-1.

'Khóa' cho mật mã Affine bao gồm 2 số, chúng ta sẽ gọi chúng là a và b. Đối với bài toán này, ta tiến hành mã hóa 26 chữ cái và 03 ký tự thường dùng trong văn bản (dấu chấm '.' + dấu phẩy ',' + khoảng trắng ' ') thành các nội dung mới (cũng nằm trong bộ 29 ký tự trên)

Khóa /công thức mã hóa: $Ci = F(Pi) = (a*Pi + b) \bmod 29$ (cặp khóa a, b là hai số nguyên)

Khóa /công thức giải mã: $Pi = F^{-1}(Ci) = a\_inv*(Ci – b) \bmod 29$ ($a\_inv*a \equiv 1 \bmod 29$)

2. Ý nghĩa các hàm

a) Quá trình mã hóa

Bước 1: Nhập một đoạn văn bản cần mã hóa vào

Bước 2: Nhập mật khẩu:

- Khi nhập mật khẩu, dữ liệu ở dạng string, sau đó nhờ hàm

  convert_to_ascii_numbers(input_string)

  để chuyển mật khẩu sang dạng số nguyên bằng cách chuyển đổi các ký tự thành mã ASCII sau đó kết hợp các số đó thành một chuỗi.
- Mật khẩu sẽ được đem đi hash qua hàm

  hash_password(password)

  bằng cách sử dụng SHA-256 để tiến hành băm mật khẩu
- Sau đó qua hàm

  save_password_to_file(hashed_password, filename)

  giá trị mật khẩu vừa được hash sẽ lưu vào file *Password.txt*

Bước 3: Tính giá trị a, b

Giá trị của a và b được tính dựa trên password được nhập vào, với hàm:

calculate_a_b_fromThePassWord(passwd)

qua đó, a và b lần lượt là phần dư của password với 2048 và 1024. Nếu a đồng dư với 29, a sẽ được tăng thêm 25 đơn vị ($a = a + 25$). Kích thước của a và b cũng được tùy chỉnh bằng cách thay đổi số 2048 và 1024.

Bước 4: Tiến hành mã hóa

Hàm mã hóa là:

affine_encrypt(plaintext, a, b, outputFile)

Trước hết, với 29 ký tự được mã hóa bao gồm từ a đến z, ký tự '.' ',' ' '. Và tất cả các ký tự còn lại sẽ được giữ nguyên (có phân biệt hoa thường)

Với đoạn văn bản được nhập vào, ký tự nào thỏa điều kiện sẽ tiến hành tính toán giá trị index của ký tự đó, theo quy ước là a = 0, b = 1, ... z = 25, các ký tự đặc biệt '.' ',' ' ' lần lượt là 26 27 28. Sau đó sẽ tiến hành tính toán qua công thức:

*(a * num + b) % 29*

Nếu không thỏa điều kiện, ký tự đó sẽ được giữ nguyên. Cuối cùng tất cả ký tự sẽ được cộng vào chuỗi cipherText.

Tiến hành tạo ra thêm {a} byte nội dung ngẫu nhiên gắn thêm vào đầu bản mã và {b} byte nội dung ngẫu nhiên gắn thêm vào cuối bản mã. Kết quả sẽ được *encode()* vào chuỗi *final_ouput* rồi ghi vào file Affine.sec dưới dạng binary.

b) Quá trình giải mã

Bước 1: Quá trình bắt đầu bằng việc chọn file để giải mã

Bước 2: Sau khi chọn được file, tiến hành nhập password.

- Nếu sau khi hash giá trị *password* và đối chiếu với file *Password.txt* có giá trị *password* đã nhập đúng, tiến hành bước tiếp theo.
- Nếu sai, tiến hành trả về **Wrong password!**

Bước 3: Tính toán a và b

Tương tự quá trình mã hóa, giá trị của a và b được tính dựa trên *password* được nhập vào với cách thức tương tự quá trình mã hóa.

Bước 4: Tiến hành giải mã

Hàm giải mã là:

affine_decrypt(inputFile, a, b)

Bằng cách loại bỏ các bit dư thừa a và b, ta có được giá trị *cipherText* cần. Sau đó tiến hành tính số nghịch đảo của a bằng:

mod_inverse(a, m)

Số nghịch đảo của a là 1 số x nào đó sao cho *(a * x) % m == 1*. Với m = 29.

Các quy ước tương tự như quá trình giải mã, chỉ tính toán đối với các ký tự từ a đến z, ký tự '.' ',' ' ' (các ký tự hợp lệ). Còn lại sẽ tiến hành giữ nguyên. Các giá trị hợp lệ được tính toán như sau:

*(a_inv * (num - b)) % 29*

Sau đó cộng dồn vào chuỗi text và ta có kết quả chuỗi kết quả sau khi đã giải mã được in ra màn hình.

3. Hướng dẫn sử dụng chương trình

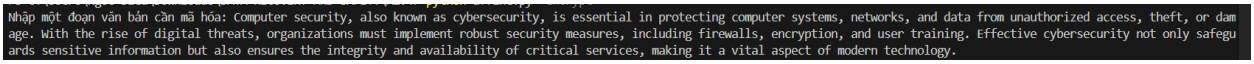
Môi trường: Visual Studio Code

Ngôn ngữ: Python

a) Mã hóa

Để thực hiện mã hóa, nhập lệnh: *python affine.py -encrypt*

Lúc này chương trình đưa ra yêu cầu nhập một đoạn văn bản:

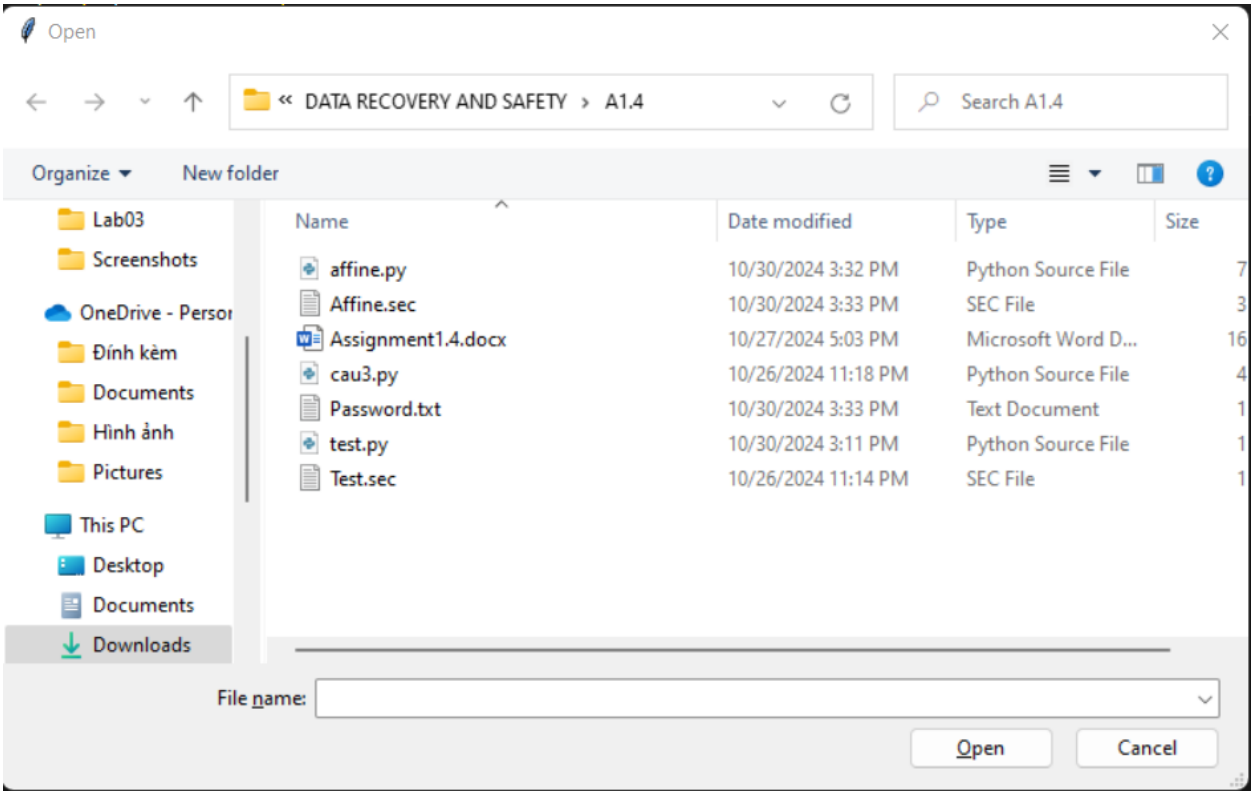


Sau đó tiến hành nhập mật khẩu mã hóa.

b) Giải mã

Để thực hiện giải mã, nhập lệnh: *python affine.py -decrypt*

Lúc này, chương trình sẽ hiện ra màn hình yêu cầu chọn file cần giải mã:



Khi chọn được file, màn hình trả về:

```
Select the cipher file to decrypt:
Affine.sec
```

Sau đó, tiến hành nhập mật khẩu cho để giải mã:

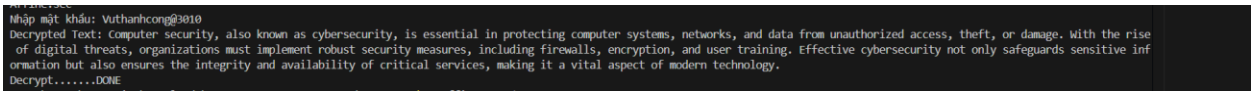
4. *Kết quả thực hiện được*

a) Mã hóa

Quá trình mã hóa đã diễn ra thành công, kết quả đã được lưu vào file *Affine.sec*

b) Giải mã

- Đối với trường hợp nhập đúng mật khẩu:



- Đối với trường hợp nhập sai mật khẩu:

II. Mã bất đối xứng RSA

1. Nguyên lý của RSA

RSA sử dụng hai khóa: khóa công khai (public key) và khóa riêng tư (private key).

- **Khóa công khai** dùng để mã hóa dữ liệu và có thể được chia sẻ công khai.
- **Khóa riêng tư** dùng để giải mã dữ liệu và phải được giữ bí mật.

Cặp khóa trong RSA được tạo ra theo các bước sau:

- Tạo hai số nguyên tố ngẫu nhiên lớn p và q.
- Tính $n = p * q$ và $\varphi = (p-1) \times (q-1)$
- Chọn số nguyên e sao cho $1 < e < \varphi$ và $gcd(e, \varphi) = 1$
- Tính d là nghịch đảo của e theo modulo φ, tức là $d \times e \equiv 1(mod\ \varphi)$.
- Cặp khóa công khai là (n, e) và khóa bí mật là d

Phương thức mã hoá:

Bên A tạo cặp khóa và gửi khóa công khai (n, e) cho bên B để bên B thực hiện việc mã hóa P thành C (bản mã – cipher). Công thức thực hiện:

$$C = F(P) = P^e \ (mod\ n)$$

Phương thức giải mã:

Khi bên A nhận được bản mã C thì sử dụng khóa bí mật d để giải mã ra bản gốc:

$$P = F^{-1}(C) = C^d \ (mod\ n)$$

2. Ý nghĩa của các hàm

a) Hàm generate_rsa_keys(p, q)

Đầu vào: 2 số nguyên tố p và q

Thuật toán:

- Tính $n = p \times q$
- Tính $\varphi = (p-1)(q-1)$
- Chọn e sao cho $1 < e < \varphi$ và $\gcd(e, \varphi) = 1$
- Tính d là nghịch đảo của e theo modulo φ (Viết 1 hàm riêng để tính d)

Đầu ra:

- publicKey: Cặp khoá công khai (n,e)
- privateKey: Cặp khoá riêng tư (n,d)

b) Hàm rsa_encrypt(plainText, publicKey, outputFile)

Đầu vào:

- plainText: chuỗi văn bản gốc cần mã hoá
- publicKey: cặp khoá công khai (n,e)
- outputFile: tên của file mà cipherText sẽ được ghi vào

Thuật toán:

- Chuyển đổi từng ký tự trong plainText sang ASCII
- Mã hoá mỗi ký tự (dưới dạng byte) với công thức $pow(byte, e, n)$
- Ghi từng byte mã hoá xuống outputFile

c) Hàm rsa_decrypt(inputFile, privateKey)

Đầu vào:

- inputFile: tên file chứa cipherText cần giải mã
- privateKey: Cặp khoá riêng tư (n,d)

Thuật toán:

- Đọc các byte mã hoá từ inputFile
- Giải mã mỗi byte với công thức $pow(char, d, n)$
- Chuyển mỗi byte thành những ký tự tương ứng và nối chúng thành chuỗi plainText

3. *Hướng dẫn sử dụng:*

a) Tạo khoá:

```
# Generate RSA keys
publicKey, privateKey = generate_rsa_keys(31847,28579)
print("Public Key:", publicKey)
print("Private Key:", privateKey)
```

b) Mã Hóa Dữ Liệu:

```
# Plain text to encrypt
plainText = "Ma hoa RSA"

# Encrypt the plainText and write to a file
outputFile = "encrypted.txt"
rsa_encrypt(plainText, publicKey, outputFile)
```

c) Giải mã dữ liệu:

```
# Read cipherText from a file and decrypt
decryptedText = rsa_decrypt(outputFile, privateKey)
print("Decrypted message:", decryptedText)
```

4. Kết quả thu được:

Tạo cặp khoá công khai và cặp khoá riêng tư:

```
Public Key: (910155413, 421861417)
Private Key: (910155413, 619501189)
```

Dữ liệu sau khi được mã hoá và ghi xuống file

encrypted.txt

```
60524114
136413748
650785939
637227672
407451959
136413748
650785939
136201652
596415676
553289469

```

Giải mã cipherText đã đọc được từ file và in ra màn hình

```
Public Key: (910155413, 421861417)
Private Key: (910155413, 619501189)
Decrypted message: Ma hoa RSA
```

III. Giải mã khi không có khóa

1. Giải mã khi không có khóa (Affine).

Bằng việc vét cạn các trường hợp có thể xảy ra, chương trình xảy ra với 2 vòng lặp:

- Vòng lặp thứ nhất để thử giá trị a từ 1 đến 5000. Mỗi lần lặp sẽ liên tục tính giá trị nghịch đảo của a và tìm kiếm giá trị a cho phù hợp.
- Vòng lặp thứ 2 được lồng trong vòng lặp thứ nhất, và lần lượt thử các giá trị a và b đến khi nào tìm ra kết quả.

Hàm sử dụng:

decrypt_without_a_b(inputFile, left, right)

Với left và right lần lượt dùng để điều chỉnh giá trị mà hàm sẽ vét cạn.

Trong bài này, chúng em chạy left = 5000, right = 5000. Độ phức tạp xấp xỉ $O(5000^2)$.

2. Sử dụng AI để nhận diện kết quả trả về.

Sử dụng tới tool trong thư viện ast để nhận diện xem liệu đây có phải là source code python hợp lệ hay không.