**LAB 1**

**Fuzzing - Reverse Engineering - Cryptography**

Họ tên và MSSV: Nguyễn Hoàng Anh

Lớp: 10\_ĐH\_CNPM1

Link youtube: List Thực hành BMMMTVHT:

<https://www.youtube.com/playlist?list=PLQjJWCgilYX-yTERBkNgYxIA4L9P2JY71>

***(Video sau khi chỉnh sửa, cắt ngắn sẽ được up lên sau ạ)***

***\*Bài lab chưa xong nên ngắn ạ…***

Lưu ý: sử dụng tài khoản là mã số sinh viên (đã tạo ở Câu 1.4 - Lab01) để thực hiện tất cả các câu trong bài thực hành. Tất cả các câu trong bài thực hành này được thực hiện trên Kali Linux

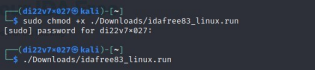
**Câu 1: Thực hiện kỹ thuật Reverse Engineering với công cụ IDA Free**

1.1. Cài đặt IDA Free vào Kali Linux. Tải file cài đặt (ví dụ: idafree83\_linux.run):

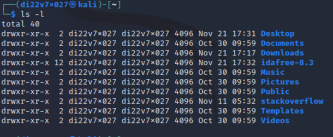
**$sudo chmod +x ./Downloads/idafree83\_linux.run**

**$./Downloads/idafree83\_linux.run**

➢ Tài và cài đạt IDA



➢ Thực hiện các bước theo yêu cầu. IDA Free sẽ được cài đặt ở thư mục cá nhân của người dùng.

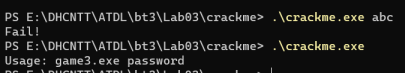


SVTH: ………………….. MSSV:

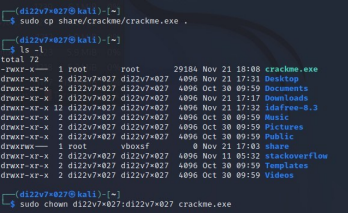
……………….

1.2. Nhấp vào link này 126 Proj 2x: Reverse Engineering with IDA Pro Freeware (10-40 pts.) tải các file  về tạo file Lab03.zip.

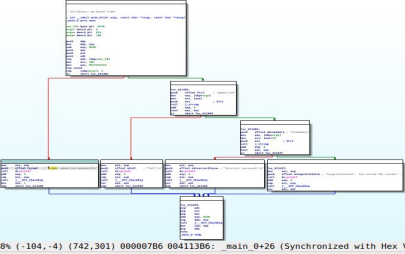
Trên máy Windows, tải file Lab03.zip. Giải nén sẽ được thư mục crackme chứa 02 file crackme.exe và  msvcr100d.dll. Chạy file crackme.exe ở môi trường CMD với vài giá trị đầu vào khác nhau và xem kếtquả.

1.3. Di chuyển file crackme.exe vào máy ảo Kali Linux (sử dụng chức năng Shared Folders hoặc Drag and Drop). Chạy IDA Free và mở file crackme.exe trên môi trường IDA Free.

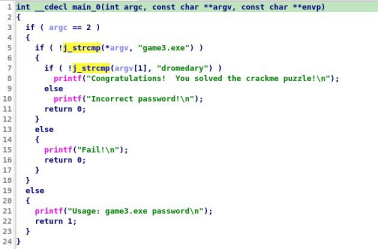
➢ **Dùng lệnh để copy file: $sudo cp share/crackme/crackme.exe .**

****

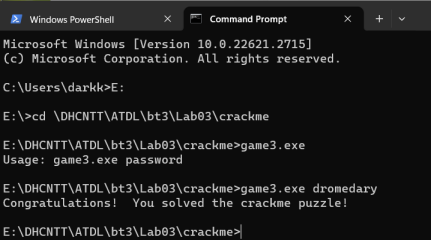
1.4. Trên thanh menu, sử dụng chức năng “Search, Text” tìm kiếm với từ khóa “usage”. Trên cửa sổ “IDA - View A” chọn “Fit window”. Chọn “Text view” hoặc “Graph view” để xem mã Assembly của chương trình.



1.5 Trên thanh menu, sử dụng chức năng “View, Sub view, Generate pseudo code” (F5) để sinh mã giả của  crackme.exe. Trên cửa sổ “Pseudocode - A”, sử dụng chức năng “Synchronize with” để đồng bộ với “IDA - View A” và “Hex - View 1”.



1.6 Phân tích mã giả của crackme.exe để có thể chạy chương trình in ra thông báo "Congratulations! You solved the crackme puzzle!”. Chạy file crackme.exe ở môi trường CMD của máy Windows in ra thông báo  trên.

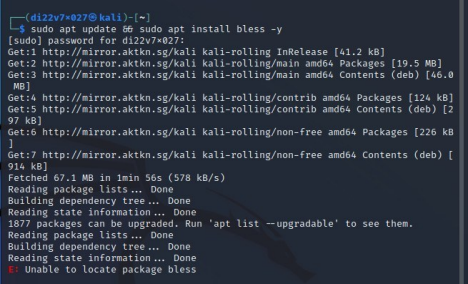


1.7. Cài đặt công cụ Bless hex editor vào Kali Linux:

➢ Cài đặt Bless

**$ sudo apt update && sudo apt install bless –y**

Nếu trong quá trình cài đặt bless xảy ra lỗi như hình bên dưới thì làm như sau

➢ Cách xử lý lỗi khi cài bless

✓ B1: **$sudo nano /etc/apt/sources.list**

✓ B2: Xoá toàn bộ dữ liệu trong file

✓ B3: Copy đoạn mã bên dưới dán vào file -> Lưu lại

deb http://http.kali.org/kali kali-rolling main contrib non-free non-free-firmware deb http://http.kali.org/kali kali-last-snapshot main contrib non-free non-free

firmware

deb http://http.kali.org/kali kali-experimental main contrib non-free non-free

firmware

deb http://http.kali.org/kali kali-bleeding-edge main contrib non-free non-free firmware

deb-src http://http.kali.org/kali kali-rolling main contrib non-free non-free

firmware

✓ B4: **$sudo apt-get upgrade**

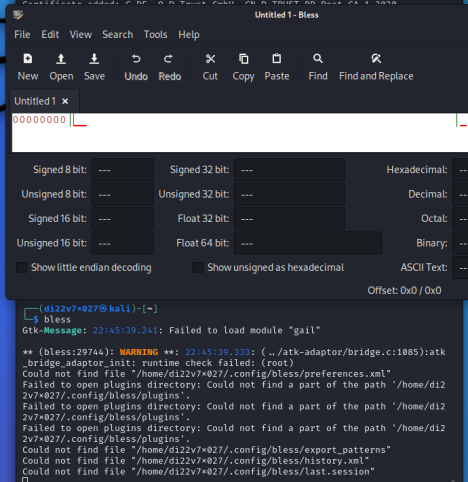
✓ B5: **$sudo apt-get update**

✓ B6: **$sudo apt-get install checkinstall**

✓ B7: **$sudo apt install bless –y**

✓ B8: Sừ dụng lệnh **$bless** để kiểm tra cài đặt thành công chưa ?

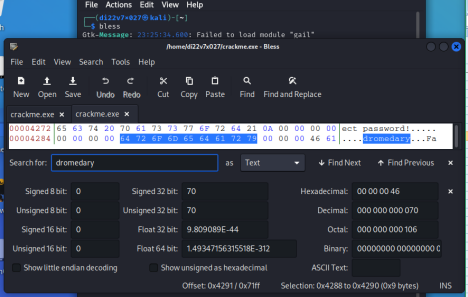
➢ Cài đặt thành công

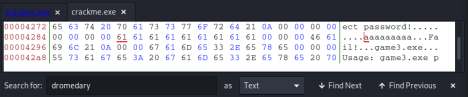


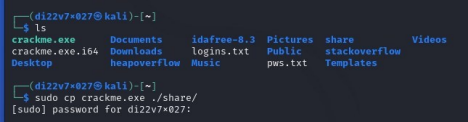
1.8. Sử dụng công cụ Bless hex editor để thay đổi giá trị password cần nhập cho crackme.exe thành “aaaaaaaaa”. Di chuyển file crackme.exe từ Kali Linux qua máy Windows. Chạy file crackme.exe ở môi trường CMD của máy Windows in ra thông báo trên "Congratulations! You solved the crackme puzzle!” ➢ Gõ lệnh: **$bless** để mở công cụ chỉnh sửa

➢ Tiếp theo vào **Open** mở file **crackme**

➢ **Search -> Find ->**tìm kiếm từ **dromedary** chọn **as Text**



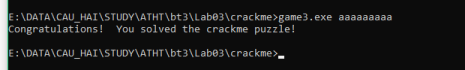
➢ Thay đổi đổi mật khẩu từ **dromedary** bằng **aaaaaaaaa -> Save - >** tắt công cụ **bless **

➢ Sao chép di chuyển tập tin **crackme** vào máy **Windows** thông qua thư mục **share **

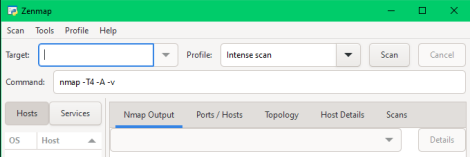
➢ Vào thư mục **crackme** trong máy Windows xoá file **game3 cũ**

➢ Copy file **crackme** vào thư mục **crackme** và đổi tên thành **game3**

➢ Mở CMD của Windows và thực thi file **game3.exe** với mật khẩu là **aaaaaaaaa**

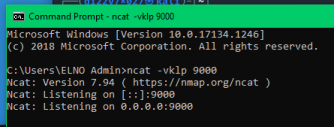
**Câu 2: Thực hiện kỹ thuật Fuzzing với công cụ Spike**

➢ Cài đặt công cụ nmap (Zenmap) trên máy Windows.

➢ Tắt tường lửa trên máy Windows

➢ Thực hiện dịch vụ Netcat ở môi trường CMD của máy Windows:

Gõ lệnh: **>ncat -vklp 9000**

****

➢ Thay đổi cấu hình mạng của máy ảo Kali Linux sao cho có thể giao tiếp mạng tới máy Windows (Sử dụng mạng Internet được chia sẻ từ điện thoại )

➢ IP của máy Windows 192.168.188.177

➢ Kiểm tra ping 192.168.188.177 (Tuỳ theo từng máy )

- Tạo file simple1.spk với nội dung bên dưới:

➢ $nano simple1.spk

➢ Nội dung file

s\_string("Hello, world!");

- Thực hiện lệnh sau để fuzzing dịch vụ Netcat. Quan sát kết quả dịch vụ Netcat nhận được. ➢ $ generic\_send\_tcp 192.168.188.177 9000 simple1.spk 0 0

➢ Kết quả dịch vụ Netcat

cleqr

➢ Tạo file simple2.spk với nội dung bên dưới:

**$nano simple2.spk**

➢ # Nội dung file

s\_string\_variable("Hello, world!");

- Thực hiện lệnh sau để fuzzing dịch vụ Netcat. Quan sát kết quả dịch vụ Netcat nhận được. ➢ $ generic\_send\_tcp 192.168.188.177 9000 simple2.spk 0 0

➢ Kết quả dịch vụ Netcat

- Tải và giải nén file Lab03.zip được thư mục vulnserver. Thực thi file vulnserver.exe ở môi trường CMD của máy Windows.

- Trên máy Kali Linux, nối kết vulnserver trên máy Windows:

**$nc 192.168.188.177 9999**

**$HELP**

**$EXIT**

➢ Tạo file trun.spk với nội dung bên dưới:

**$nano trun.spk**

➢ Nội dung file

**s\_readline();**

**s\_string("TRUN ");**

**s\_string\_variable("COMMAND");**

- Thực hiện lệnh sau để fuzzing dịch vụ vulnserver. Quan sát kết quả sẽ thấy vulnserver dừng hoạt động sau một thời gian ngắn:

**$ generic\_send\_tcp 192.168.188.177 9999 trun.spk 0 0**

➢ Kết quả dịch vụ vulnserver

Cài đặt công cụ Immunity Debugger vào máy Windows. Sử dụng công cụ để mở file và thực thi file vulnserver.exe. Tiếp tục fuzzing dịch vụ vulnserver. Quan sát giao diện để thấy thanh ghi EIP chứa giá trị “41414141” (có lỗi buffer overflow).

➢ Mở file vulnserver trên công cụ **Immunity Debugger**

➢ Tiếp tục **$generic\_send\_tcp 192.168.188.177 9999 trun.spk 0 0**

➢ Kết quả dịch vụ **Immunity Debugger** thấy thanh ghi EIP chứa giá trị “41414141” (có lỗi buffer overflow).

- Khai thác lỗi buffer overflow trên vulnserver.exe theo hướng dẫn. (Không bắt buộc thực hiện) **Câu 3: Giải thuật băm và tấn công mật khẩu**

3.1. Tìm giá trị băm của chuỗi "@ntoanhethong\_ct222" sử dụng giải thuật MD5 và SHA512 sử dụng công cụ md5sum và sha512sum:

➢ **$ echo -n "@ntoanhethong\_ct222" | md5sum**

➢ **$ echo -n "@ntoanhethong\_ct222" | sha512sum**

➢ Sử dụng một trang web online cho phép thực hiện giải thuật băm, ví dụ :https://www.pelock.com/products/hash-calculator để kiểm tra kết quả.

3.2. Sử dụng công cụ john và hashcat để dịch ngược giá trị băm "b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46 df5f1326af5a2ea6d103fd07c95385ffab0cacbc86"

# Xác định giải thuật băm

➢ **$ hashid**

**"b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7 785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86"**

➢ $ **echo -n**

**"b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7 785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86" > hash**

➢ **Hiển thị nội dung: $ cat hash**

➢ # Sử dụng John the Ripper

**$john --format=raw-sha512 \--wordlist=/usr/share/wordlists/metasploit/password.lst hash**

➢ # Sử dụng hashcat

**$hashcat -a 0 -m 1700 ./hash /usr/share/wordlists/metasploit/password.lst**

Không đủ dung lượng bộ nhớ cho hình thức tấn công này. Vì vậy ta tắt máy ảo và tăng dung lượng bộ nhớ RAM trên máy ảo lên. Sau đó chạy lại lệnh trên

➢ Sử dụng một trang web cho phép dịch ngược giá trị băm, ví dụ: https://crackstation.net/, để kiểm tra kết quả.

3.3. Tạo một file zip “abc.zip” với mật khẩu mở file là "abc123". Sử dụng john và hashcat để dò mật khẩu của file:

abc123

➢ Trích xuất giá băm của mật khẩu

$ **zip2john abc.zip > hash**

➢ Dò mật khẩu:

**$ john --wordlist=/usr/share/wordlists/metasploit/password.lst hash**

# Chỉnh sửa file hash cho phù hợp định dạng của hashcat

Xoá đi đoạn đầu đoạn cuối trong file hash chỉ lấy “$zip…..zip$”

**# hashcat -a 0 -m 13600 hash /usr/share/wordlists/metasploit/password.lst**

3.4. Sử dụng công cụ john dò mật khẩu người dùng:

➢ Tạo người dùng mới “newuser” với mật khẩu “qwerty”

**$ sudo adduser newuser**

➢ Trích xuất giá trị băm của mật khẩu người dùng

**$ sudo cat /etc/shadow | grep "newuser" > hash**

➢ Dò mật khẩu:

**$ john --format=crypt --wordlist=/usr/share/wordlists/metasploit/password.lst hash**

**Câu 4: Tìm hiểu giải thuật AES**

➢ Cài và tạo môi trường ảo cho python:

➢ **$ sudo apt update && sudo apt install python3-venv –y**

➢ **$ python -m venv lab03\_04**

➢ Cài đặt module pycryptodome:

**$ ./lab03\_04/bin/pip install pycryptodome**

➢ Tải và giải nén file Lab03.zip được thư mục aes. Sử dụng lệnh bên dưới copy file **$sudo cp –r ./share/aes ./**

**$ls-l** lúc này quyền thuộc root

➢ **Chuyển quyền cho người dùng user: $sudo chown –R di22v7x027:di22v7x027 aes**

Tạo tập tin aes\_ecb.py để mã hóa tập tin tux.bmp theo giải thuật AES-ECB. Thực thi aes\_ecb.py: ➢ Tạo tâp tin **$nano ./aes\_ecb.py**

➢ **$./lab03\_04/bin/python aes\_ecb.py**

from Crypto.Cipher import AES

key = b"aaaabbbbccccdddd"

cipher = AES.new(key, AES.MODE\_ECB)

with open("./aes/tux.bmp", "rb") as f:

clear = f.read()

clear\_trimmed = clear[54:-12]

ciphertext = cipher.encrypt(clear\_trimmed)

ciphertext = clear[0:54] + ciphertext + clear[-12:]

with open("./aes/tux\_ecb.bmp", "wb") as f:

f.write(ciphertext)

➢ Kiểm tra chiều dài tập tin mã hoá

➢ Thực hiện chỉnh sửa code mã hoá tập tin

➢ Gõ lệnh thực thi chương trình mã hoá tập tin ➢ Xem lại tập tin mã hoá theo giải thuật ECB 

Tập tin ban đầu

Tập tin đã được mã hoá 

Tạo tập tin aes\_cbc.py để mã hóa tập tin tux.bmp theo giải thuật AES-CBC. Thực thi aes\_cbc.py: ➢ Tạo tập tin: **$nano aes\_cbc.py**

from Crypto.Cipher import AES

key = b"aaaabbbbccccdddd"

iv = b"0000111122223333"

cipher = AES.new(key, AES.MODE\_CBC, iv)

with open("./aes/tux.bmp", "rb") as f:

clear = f.read()

clear\_trimmed = clear[54:-12]

ciphertext = cipher.encrypt(clear\_trimmed)

ciphertext = clear[0:54] + ciphertext + clear[-12:]

with open("./aes/tux\_cbc.bmp", "wb") as f:

f.write(ciphertext)

➢ Copy đoạn code bên trên paste vào tập tin vừa tạo

➢ Thực thi chương trình mã hoá **$./lab03\_04/bin/python aes\_cbc.py**

➢ Xem lại tập tin được mã hoá theo giải thuật CBC 

Tập tin ban đầu

Tập tin đã được mã hoá

Viết code giải mã tập tin secret.png đã được mã hóa theo giải thuật AES CBC. Biết key mã hóa là "aaaabbbbccccdddd" và IV là "0000111122223333". Lưu ý: phần header của tập tin không được mã hóa.

➢ Tạo tập tin giải mã: **$nano aes\_key\_cbc.py**

from Crypto.Cipher import AES

key = b"aaaabbbbccccdddd"

iv = b"0000111122223333"

cipher = AES.new(key, AES.MODE\_CBC, iv)

with open("./aes/tux\_cbc.bmp", "rb") as f:

encrypted\_data = f.read()

header\_size = 54 # Kích thước header của file bmp

footer\_size = 12 # Kích thước footer của file bmp

# Phân chia dữ liệu thành header, ciphertext và footer

header = encrypted\_data[:header\_size]

ciphertext = encrypted\_data[header\_size:-footer\_size]

footer = encrypted\_data[-footer\_size:]

# Giải mã dữ liệu

decrypted\_content = cipher.decrypt(ciphertext)

# Ghi dữ liệu đã giải mã ra tập tin mới

with open("./aes/decrypted\_tux.bmp", "wb") as f:

f.write(header + decrypted\_content + footer)

➢ Copy đoạn code bên trên paste vào tập tin vừa tạo

➢ Thực thi chương trình mã hoá **$./lab03\_04/bin/python aes\_key\_cbc.py** ➢ Xem lại tập tin được giải mã theo giải thuật cbc 

Tập tin đã được giải mã

Tập tin đã được mã hoá 

**Câu 5: Tìm hiểu giải thuật Diffie-Hellman và RSA**

**5.1.** Tìm hiểu giải thuật Diffie-Hellman trao đổi khóa bí mật qua video. Tìm khóa bí mật sk, với g = 23, p=5, a=4 (khóa bí mật của Alice), b=10 (khóa bí mật của Bob).

Để tính khóa bí mật chung (shared secret key) trong giao thức trao đổi khóa Diffie-Hellman, chúng ta sử dụng các thông số như sau:

p: Mô-đun prime (5)

g: Phần tử nguyên thủy (23)

a: Khóa bí mật của Alice (4)

b: Khóa bí mật của Bob (10)

Theo giao thức Diffie-Hellman, cả Alice và Bob sẽ thực hiện các bước sau:

Alice tính *A=ga* mod  *p*

Bob tính *B=gb* mod  *p*

Cả hai cùng trao đổi giá trị A và B

Alice tính *sk=Ba* mod *p*

Bob tính *sk=Ab* mod *p*

Ta bắt đầu tính toán:

Bước 1 (Alice tính *A*): *A=ga* mod *p=234* mod *5*

*A* = 1 mod 5

Bước 2 (Bob tính *B*): *B=gb mod p=2310 mod 5*

*B* = 4 mod 5

Sau khi trao đổi giá trị A và B:

Bước 4 (Alice tính *sk*): *sk = Ba mod p = 44 mod 5*

*sk* = 1 mod 5

Bước 5 (Bob tính *sk*): *sk = Ab mod p=110 mod 5*

*sk* = 1 mod 5

Vì *sk* của cả hai đều bằng 1, đó chính là khóa bí mật chung (shared secret key). Trong trường hợp này, giá trị của *sk* bằng 1.

Khóa bí mật sk= 1

Có thể tìm hiểu thêm về giải thuật Diffie-Hellman qua video sau.

**5.2.** Truy cập đến địa chỉ http://people.cs.pitt.edu/~kirk/cs1501/notes/rsademo/index.html để tạo khóa, mã hóa, giải mã sử dụng giải thuật RSA. Sau đó mô tả ngắn gọn cách RSA tạo khóa, mã hóa và giải mã.

➢ **Mô tả cách RSA tạo khoá, mã hoá, giải mã**

**RSA** là một thuật toán mã hóa và giải mã dựa trên khóa công khai **(public key)** và khóa bí mật **(private key)**. Quá trình trong **RSA** bao gồm ba bước chính: tạo khóa, mã hóa và giải mã. - **Tạo Khóa (Key Generation):**

Chọn hai số nguyên tố lớn ngẫu nhiên **p** và **q**.

Tính tích của hai số nguyên tố này: n = p \* q. Đây là modulus.

Tính hàm Euler của n: **phi(n) = (p - 1) \* (q - 1).**

Chọn một số nguyên **e (1 < e < phi(n))** sao cho **e** và **phi(n)** là số nguyên tố cùng nhau và được sử dụng làm khóa công khai.

Tìm số nguyên **d** sao cho **(d \* e) mod phi(n) = 1**. Số **d** được sử dụng làm khóa bí mật. - **Mã Hóa (Encryption):**

Dùng khóa công khai **(e, n)**, chuyển đổi thông điệp **(plaintext)** thành **cipher text** bằng cách sử dụng công thức: **ciphertext = (plaintext^e) mod n.**

- **Giải Mã (Decryption):**

Sử dụng khóa bí mật **(d, n),** giải mã cipher text để nhận lại **plaintext** ban đầu bằng công thức: **plaintext = (ciphertext^d) mod n.**

**- Tóm lại:**

Khóa công khai **(public key):** Gồm hai thành phần **(e, n)** được chia sẻ với người dùng khác để mã hóa thông điệp.

Khóa bí mật **(private key):** Gồm hai thành phần **(d, n)** được bảo mật, chỉ người nhận có thể sử dụng để giải mã thông điệp đã được mã hóa.

**RSA** dựa trên tính khó của việc phân tích một số nguyên lớn thành các thừa số nguyên tố. An toàn của thuật toán dựa trên khả năng tính toán nhanh chóng khi mã hóa nhưng rất khó để giải mã mà không có khóa bí mật tương ứng.

**5.3.** Với p = 23, q = 41, e = 7 giải mã dữ liệu đã được mã sử dụng RSA sau: ”171 ,362 ,466 ,50 ,333 ,913 ,429 ,50 ,547 ,466 ,906 ,554 ,809 ,91 ,736 ,906 ,24 ,118”. Trong đó các con số là mã unicode của các kí tự được mã hóa.

SVTH: ………………….. MSSV: ……………

➢ **PHÂN TÍCH ĐỀ BÀI**

Để giải mã dữ liệu đã được mã hóa sử dụng **RSA**, chúng ta sẽ sử dụng khóa công khai **(e, n)** và khóa bí mật **(d, n)** đã được tạo từ p và q cho trước.

Trước tiên, chúng ta cần tính **n**, **phi(n)**, và **d** từ **p** và **q**:

• **p = 23**

• **q = 41**

• **e = 7**

• Tính **n**: **n = p \* q = 23 \* 41 = 943**

• Tính phi**(n)**: **phi(n) = (p - 1) \* (q - 1) = 22 \* 40 = 880**

• Tìm **d**: Để tìm **d**, ta cần tìm số nguyên **d** sao cho **(d \* e) mod phi(n) = 1**. Sử dụng thuật toán Euclid mở rộng hoặc thuật toán dùng lũy thừa nhanh để tính **d**. Trong trường hợp này, giả sử **d = 503**.

- Sau khi có khóa bí mật **(d, n**), chúng ta có thể giải mã dữ liệu đã được mã hóa.

Dữ liệu đã được mã hóa: **[171, 362, 466, 50, 333, 913, 429, 50, 547, 466, 906, 554, 809, 91, 736, 906, 24, 118]**

- Sử dụng khóa bí mật **(d, n)**, ta sẽ giải mã từng số trong dãy mã hóa để nhận lại các ký tự tương ứng theo mã Unicode:

➢ Tạo tập tin giải mã: **$nano aes\_RSA.py**

ciphertext = [171, 362, 466, 50, 333, 913, 429, 50, 547, 466, 906, 554, 809, 91, 736, 906, 24, 118] d = 503

n = 943

plaintext = ""

for num in ciphertext:

decrypted\_num = (num \*\* d) % n

plaintext += chr(decrypted\_num)

print(plaintext)

➢ Copy đoạn code bên trên paste vào tập tin vừa tạo

➢ Thực thi chương trình: $**./lab03\_04/bin/python aes\_RSA.py**

➢Kết quả là: **……**

---HẾT---