Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO LAB 6

Môn học: Mật mã học

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: chủ đề môn học

Ngày báo cáo: xx/xx/20xx

Nhóm: XX (nếu không có xoá phần này)

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT101.XXXX.YYYY

STT	Họ và tên	MSSV	Email
1	Phạm Nguyễn Hải Anh	21520586	
2	Nguyễn Nhật Quân	21522497	

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Câu hỏi 01	1%	
2	Câu hỏi 02	100%	
3	Câu hỏi 03	100%	
4	Câu hỏi 04	100%	
5	Câu hỏi 05	60%	
6	Câu hỏi 06		
7	Câu hỏi 07		
8	Câu hỏi 08		

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

Để chạy được các file python trong bài nộp, cần cài đặt:

thư viện pycryptodome

thư viện gmpy2

1. Kịch bản 01/Câu hỏi 01

Khóa được cho là 1 trong các khóa "possibly weak" (Recommendation for the triple data encryption algorithm (TDEA) block cipher (nist.gov)). Khóa dạng này sẽ tạo ra 4 subkeys, thay vì 16 subkeys (mỗi subkey 48 bits, tức 6 bytes)

Tuy đây là câu dễ nhưng hướng giải sẽ phức tạp.

Sử dụng PyCryptoDome để tạo ra các subkey của key được cho:

1. Subkey 0 (K1): 3A6D6C94C75A
2. Subkey 1 (K2): 797C40A819C5
3. Subkey 16 (K17): B183EF1F9DD1
4. Subkey 17 (K18): 3AF59E08A4A3

Để khóa được đề cho trở thành khóa yếu, nghĩa là tồn tại plaintext m sao cho:

$$\text{Encrypt}(\text{Encrypt}(m)) = m$$

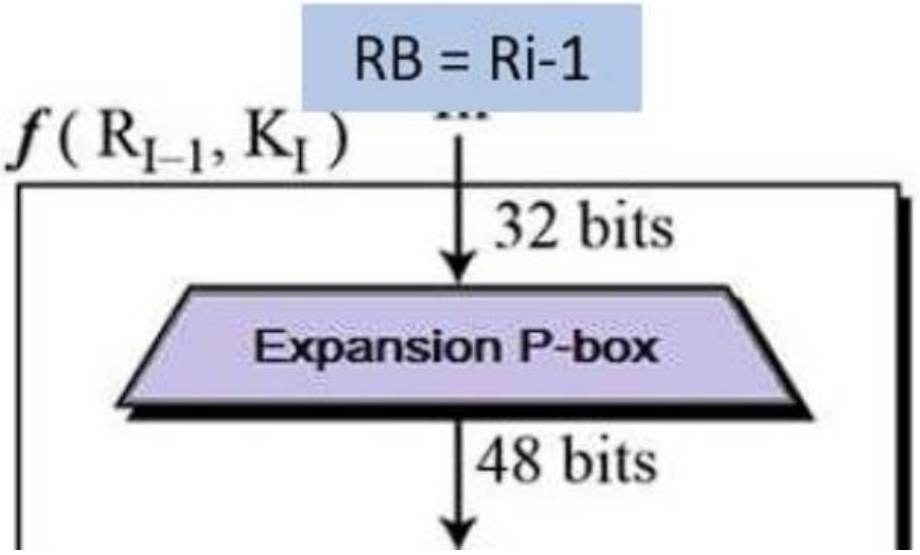
Không chắc hàm Encrypt() phải là DES hay không vì web trên là TDEA (Tripple DES).

Lời giải này sẽ đi tìm 1 plaintext 64-bit, giải theo hướng vận hành của DES.

Đầu tiên, 64-bit plaintext hoán vị:

```
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]
```

rồi chia thành 2 32-bit blocks, Lấy block phải mở rộng cho đủ 48-bit:



Nhưng em không nghĩ bài này sẽ giải như thế nên em đã thay các key yếu và bán yếu trên mạng làm plaintext rồi bruteforce (file bt1.py) nhưng không thành công

2. Kịch bản 02 (file bt2.py)

Ta tìm key phù hợp bằng cách thử từng key trên wiki (Weak key - Wikipedia) vào việc giải mã đoạn ciphertext:

```
Last login: Tue Jun  6 01:29:12 2023 from 192
pngha2@ansibletarget1:~$ python3 bt2.py
wrong key:  1F1F1F1F0E0E0E0E
wrong key:  E0E0E0E0F1F1F1F1
wrong key:  011F011F010E010E
wrong key:  0101010101010101
wrong key:  FEFEFEFEFEFEFEFE
wrong key:  0000000000000000
wrong key:  FFFFFFFFFFFFFFFF
wrong key:  E1E1E1E1F0F0F0F0
wrong key:  1E1E1E1E0F0F0F0F
wrong key:  1F011F010E010E01
wrong key:  01E001E001F101F1
semi weak key des - khoá nửa yếu trong des
key:  E001E001F101F101
wrong key:  01FE01FE01FE01FE
wrong key:  FE01FE01FE01FE01
wrong key:  1FE01FE00EF10EF1
wrong key:  E01FE01FF10EF10E
wrong key:  1FFE1FFE0EFE0EFE
wrong key:  FE1FFE1FFE0EFE0E
wrong key:  E0FEE0FEF1FEF1FE
wrong key:  FEE0FEE0FEF1FEF1
```

Vậy key cần tìm là 0xE001E001F101F101

3. Câu hỏi 3:

Trường hợp sử dụng CBC:

Cần tính toàn vẹn (integrity): các block phụ thuộc với nhau, từ đó ta có thể kiểm tra nếu block nào thay đổi.

Dữ liệu đầy đủ: Các data sẽ đi liền với nhau vì tính chất liền khối của CBC. Không thể cắt ngang dữ liệu.

Ví dụ: Sử dụng trong việc mã hóa toàn bộ một file.

Trường hợp sử dụng OFB:

Đồng bộ 2 bên: OFB đồng bộ luồng giữa bên gửi và bên nhận

Truy cập bất kỳ: OFB cho phép giải mã hoặc mã hóa 1 đoạn bất kỳ của dữ liệu

Ví dụ: Sử dụng cho các ứng dụng streaming, ứng dụng thời gian thực

4. Câu hỏi 4: (file bt4.py)

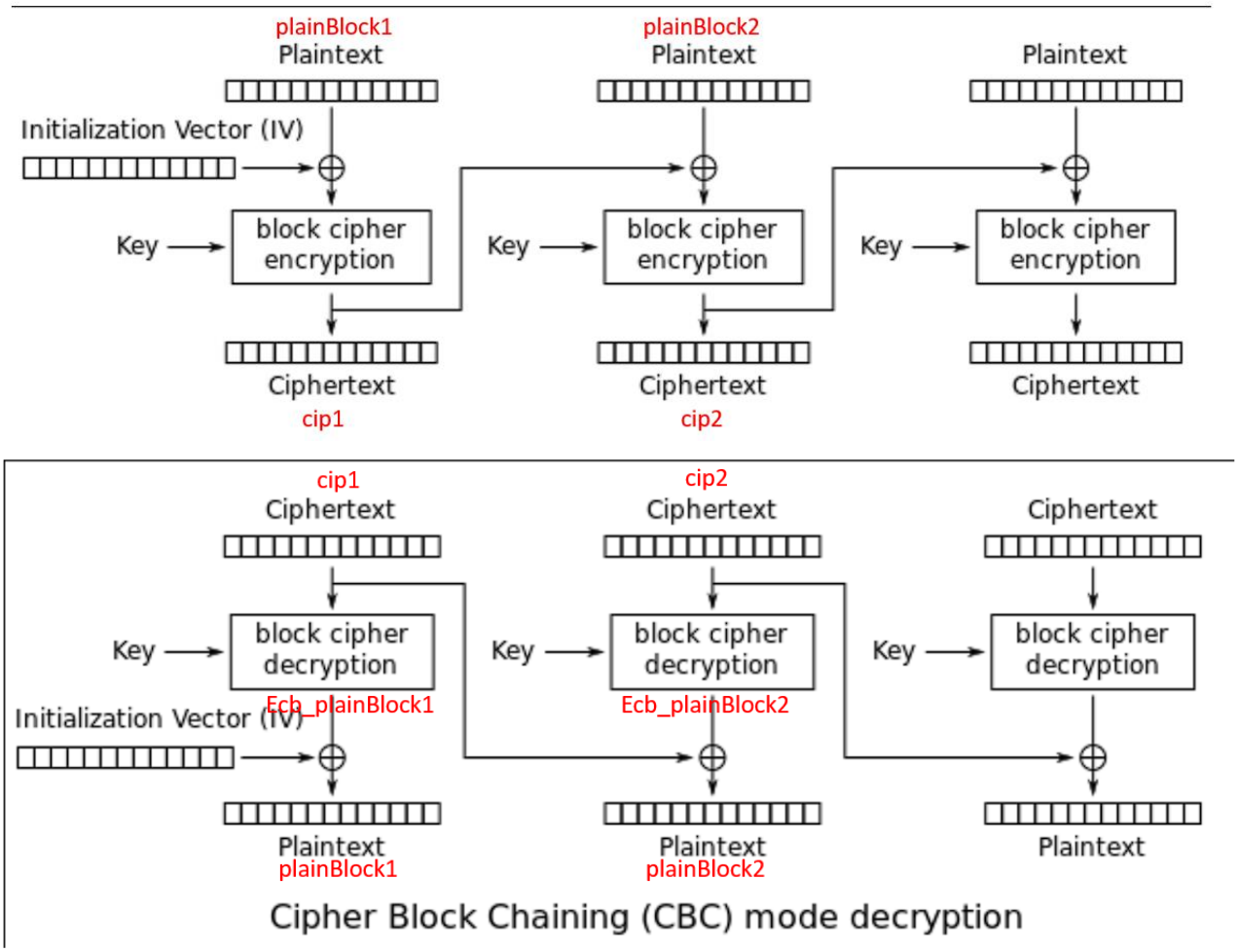
Đổi các thông số từ base64 sang hex:

```
cipher:
5c491d9b80063faa102b439f2a137b6bf7d305bf97c9432528e1c02a83d6ef0533adc6ee2678f8d236827d556294798b53557f757225b5417c76c0210bc2c304
820186f2b30ee72f611c20d47c17f73030c6679122af6a16cebe7a2f34b00cde9ec7418b2177e93a713de7ec2512534497ea68eec557a2cb4f6d06919aeaa256
8a117133b96001f92b96a9aa01fabb951fd54f9821fa23c3ef7dd0c0fb46c4b755c2e28b521cc48373d5f97c3a14b3bf7aa0fa47b897803402cdaf5b3d57613e
545ade6649f6db965c95d73e5deb9e063ca0e2ecb44b0265b5ea18bc5fed5775

iv:
c88c199f44545ef970f65c6d5610ae3a

plaintext2:
854fbabf2c97ed9850b7193e76d43ffceca8a638e3264bc2f190f2fc0a7b128a4c546b9fe3a180914686e05bf6b7cf3013ceae0f9dd19bf25846fe3b05b41ae3
90f8111d524976e15c37a44559adb66ae64ae38b9f2eb45a035e59a01964db1063ae0ef756fd0561bd925a625dc84fb1f2b22ce552579ff9d11dac895c572b34
f6841b87aa398ceb01a985fcba89cab731b81213d7a482974cb6c769a1839b518e13ff799a5940e39c9c6a6f8b66bc56ef7d92ab968d2538f0f58d14dbaf02dc
5ac893a60310ee1460ac8f39fbe78085cf39fe058f57fbf4bd2f745073e89d05
```

Ta có ciphertext và plaintext2 đều có 224 bytes, iv 16 bytes, nghĩa là sẽ có 14 blocks.

Logic giải sẽ vận hành như hình dưới:

Với plainBlock là các khối plaintext gốc cần tìm, cip là các khối ciphertext, ecb_plainBlock là các khối plaintext2. Các khối này đều là 128 bit



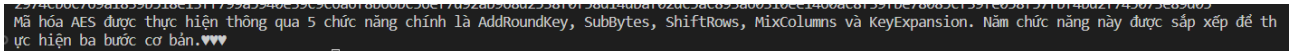
plainBlock1 = ecb_plainBlock1 xor IV

plainBlock2 = ecb_plainBlock2 xor cip1

và cứ thế cho tới:

plainBlock22 = ecb_plainBlock22 xor cip21

Trước khi thực thi code, ghép IV vào trước cip rồi dùng zip để xor từng byte:



Plaintext cần tìm: "Mã hóa AES được thực hiện thông qua 5 chức năng chính là AddRoundKey, SubBytes, ShiftRows, MixColumns và KeyExpansion. Năm chức năng này được sắp xếp để thực hiện ba bước cơ bản.♥♥♥"

5. Câu hỏi 5:

```
e = 65537
n = 0x99EFA9177387907EB3F74DC09A4D7A93ABF6CEB7EE102C689ECD0998975CEDE29F3CA951FEB5ADFB9282879CC666E22DCAFC07D7F89D762B9AD55320
ct = 0x339BE515121DAB503106CD190897382149E032A76A1CA0EEC74F2C8C74560B00DFFC0AD65EE4DF4F47B2C9810D93E8579517692268C821C67249464
k = 2**1025 - 3

with open("fact.json", "rb") as f:
    # curl 'http://factordb.com/api?query=2^1025-2' -o fact.json
    fact = json.load(f)
fs = [gmpy2.mpz(p) for p, _ in fact["factors"]]
print(fs)


def factor():
    cur = 2

    # better version
    for i in range(1, len(fs)):
        for c in combinations(fs, i):
            s = reduce(mul, c)
            if isprime(s + 1):    "isprime" is not defined
                cur = gmpy2.powmod(cur, s + 1, n)
                g = gmpy2.gcd(cur - 1, n)
                if 1 < g < n:
                    p = g
                    q = n // p
                    return p, q


p, q = factor()
phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
pt = gmpy2.powmod(ct, d, n)
print(int(pt).to_bytes((int(pt).bit_length() + 7) // 8, "big"))
```

```
> py .\bai5.py
[mpz(2), mpz(3), mpz(5), mpz(17), mpz(257), mpz(641), mpz(65537), mpz(274177), mpz(2424833), mpz(6700417), mpz(67280421310721), mpz(123892
6361552897), mpz(5964598127497217), mpz(57046892006851290547 21), mpz(74556028256478842083373957362004549187833663426 57), mpz(934616397153
5797776916355819960689658405123754163818858028032 1), mpz(74164006262753080152478714190193747405994078109751902390582131614441575950470500 8
092818711693940737)]
b'CTF{Recycling_Is_Great}'
```

6. Câu hỏi 6

Vì e=13 là 1 số mũ nhỏ nên có nhiều cách để tấn công (tìm lại plaintext). Bắt đầu từ trường hợp m^e < n, với m là plaintext.

Do m^e < n nên m^e mod n = m^e. Vậy chỉ cần căn bậc e (e=13) là ra lại plaintext mà không cần n.

Sử dụng hàm root của thư viện gmpy2:

```
with gmpy2.local_context(gmpy2.context(), precision=800) as ctx:
 ctx.precision += 800
 root = gmpy2.root(cipher,e)
```

Chạy file:

```
pngha2@ansibletarget1:~$ python3 bt6_demoi.py
191837232265290072623428073582671913806789486696369999277269281.0
b'wao. m^e < n is so bad !!!'
```

7. Câu hỏi 7

Đầu tiên, em lấy dữ liệu theo của file uit.png và 1 trong các file signature dưới dạng hình thì thấy có sự trùng hợp ở phần đầu của dữ liệu

```
image:
89504E470D0A1A0A0000000D4948445200000DA000001050806000000 8502F995000000097048597300000EC400000EC401952B0E1B
0000200049444154789CE4BD77BC5D45D5F8FD9D73EF4DEF9D544249420B0408844E80D0A4F722A080A080A2A0A062C1868F0AA2D205
4104A437A9A193D04BE81020B48424A4F7DCDC76D6FBC7DAB3F7EC3933FB9C1BD4E7F97DDEE113EED97B4F596BCD5AB3CA3423228293
4404630CFF7F4AFF4E9CFFAFD16F6DE0F94FE1506BBDEDC9B7720D3CFDA6F0D45BF0D6A7B06099A12C0683D0A7BBB0D130D87113D863
0B43DF1E608CF95FE923E30B9A45C0052404987D574BDE588AE5AD5687FB3DD43EF0A561A85697FF6D6D71F1F3A69D2120802D6A6B08
F583FFBEBD8C548D9EFEB7227EF09F5DF6AA85567EBED8F3EA26E1F6A9C20D8FC3FC658652C980D39624143306CA2274ED08876C2F9C
BCB7A177B7EAF8B8EDD93643B884CA84F21AD154F1315479B55444A0585D2EA26EB9A232B5C01842D6679A6AB054A3498871E20C2634
B7C2B255B07039CC5E282C5C0EF3971AE62F1556AE31AC5A23343643B90C65319405EA4B8231D0B101BA743274EF24F4ED61E8DF5318
DA0F761D6B28994AD86AA1430CD75ABEB9F887E8D91EDAC4EA8ED1F9932F849FDD20BCF9A92103C1FE10C0C2EDBED27743FBC12F8E15
B6DAD054C5A5889EB5D0C6C5DD94CBE5AA92D45E81FBBF50F63F518F5F5F11739605E62E86F7660AD33F87F766C167F361F10AC39A66
682B03C6D554B90139DF1E2049060106F6847DB616F6DF163618EC300830638EF0C16CD868180CEB6768A8FFF7E1BBB6DF6BCD534BFA
700E9C7E19CC5F66EB55BA190063C5CCA1A50193BCB4EFBA74845F1F2FECB6F97FCF7C0C9A8E7EAA665EC546865AEB589B7C5FB67C91
C959ABE9EA8FD6F397C2CB1F082F4C87691FA9B66A69031C33C6A9257DAFF662F2EC352B0EE78C1D291CB9936AB1AE9D32585734C2A3
D3847B5F8077661A5ADBA0AE047D7BC026C3853DB714F61D5F57952645F489BDAF85DEFF0E170160DE12E1B80B61DED2CC4A48889314
70EA3290A3B1F328025D3B09579E01634756C7A51653B91AFC1582B6B68C5E6B077C199FEE3F59A696C1C44F6D6DC227F38427DF8027
DF84E99F1BD554496A371D2B644FE8D401761B2B7C7537D878B849EB6C6B13DE9B05B73F233CF69A61E51AB2C2085D3AC28431AAF5B6
DFC8D0A943064B5B9BD0568686FAB0CFD71E3A55F373AAA21C31DDFCBADADA846F5F293CFBAEC10E48AE65885FC60A15894623F37F35
5402C3FB97B9F95C43B7CEC57C5AE482D4921FA0DECD141BA98B9CDC5883B5BC8F29D3B89F93958939A6B5F84F7E5B7EBB2158EC6F41
4DC2475E151E790DDE9B65129324E979C76E91C425CF4C9B84B6494703CE809CB76FFAF78403270887ED6818D43B8369D92AE1C93784
DB9F85B73FCD980E0C7575C2C6C384FDB68149E30C7DBAE7856BE60278F24DE1B1D7D44FDC6D73F8CA36306628D4D755D2D84D6E7F14
0DA26E7EBF3F624219CAEBD7F7C41BC273EF59029398D1AE560B98A6AEA213FFB5307381E1FAC785D3F7ABE4E75AB4959B8A7C4F11A9
1E75ACC6A8ED49FF2E6DF9BF919A5A84573E10EE78069E9BAE7E16C6F8961E6EB4301530FBC1E220197B387C030823070A47ED02FB8E
37F4E892D1FDB3F970D773C2832F1BE62FD351D956D6B707ECBEB970D004C398E15057CACA7DB1049E7E4B78F85578EB53352BADA00B
42A70638684299EF1F5AA24343751720C5E63FCE0B92FA54AD6D70E29F84373F2991B318DBDD268AB9D36E8F2EC243BF84AE9DFE7DB8
8468581F4238A61EBF8CC0F99A2826C83E80A1E7180CB59821ED356957340A0FBF22DC36153E9C6310D14857DAE158CD653354BF25BF2
8294DA3149FE547925C3ED26C3D53C9C38D6D0B9A3D2BFB54D78E363E1962930E52DC39A168349C4A464D4673B647BD8653343CFAE99
B5B06A8DF0DC7BC27D2FC2CB1F185635194AE9086BA8AF13361921ECB639ECBCA96158FF12F575790DD31E93303418C7DEB5B6C1D255
B074A5B060392C582A2C59A9EF96AC84E5AB85C6E612CD2DB0A6594DDC9636F8745E323C88496867A38AD6A2C8FC3593FC75E2213A28
```

```
signature:
89504E470D0A1A0A0000000D4948445200000DA000001050806000000 8502F995000000097048597300000EC400000EC401952B0E1B
0000200049444154789CE4BD77BC5D45D5F8FD9D73EF4DEF9D544249420B0408844E80D0A4F722A080A080A2A0A062C1868F0AA2D205
4104A437A9A193D04BE81020B48424A4F7DCDC76D6FBC7DAB3F7EC3933FB9C1BD4E7F97DDEE113EED97B4F596BCD5AB3CA3423228293
4404630CFF7F4AFF4E9CFFAFD16F6DE0F94FE1506BBDEDC9B7720D3CFDA6F0D45BF0D6A7B06099A12C0683D0A7BBB0D130D87113D863
0B43DF1E608CF95FE923E30B9A45C0052404987D574BDE588AE5AD5687FB3DD43EF0A561A85697FF6D6D71F1F3A69D2120802D6A6B08
F583FFBEBD8C548D9EFEB7227EF09F5DF6AA85567EBED8F3EA26E1F6A9C20D8FC3FC658652C980D39624143306CA2274ED08876C2F9C
BCB7A177B7EAF8B8EDD93643B884CA84F21AD154F1315479B55444A0585D2EA26EB9A232B5C01842D6679A6AB054A3498871E20C2634
B7C2B255B07039CC5E282C5C0EF3971AE62F1556AE31AC5A23343643B90C65319405EA4B8231D0B101BA743274EF24F4ED61E8DF5318
DA0F761D6B28994AD86AA1430CD75ABEB9F887E8D91EDAC4EA8ED1F9932F849FDD20BCF9A92103C1FE10C0C2EDBED27743FBC12F8E15
B6DAD054C5A5889EB5D0C6C5DD94CBE5AA92D45E81FBBF50F63F518F5F5F11739605E62E86F7660AD33F87F766C167F361F10AC39A66
682B03C6D554B90139DF1E2049060106F6847DB616F6DF163618EC300830638EF0C16CD868180CEB6768A8FFF7E1BBB6DF6BCD534BFA
700E9C7E19CC5F66EB55BA190063C5CCA1A50193BCB4EFBA74845F1F2FECB6F97FCF7C0C9A8E7EAA665EC546865AEB589B7C5FB67C91
C959ABE9EA8FD6F397C2CB1F082F4C87691FA9B66A69031C33C6A9257DAFF662F2EC352B0EE78C1D291CB9936AB1AE9D32585734C2A3
D3847B5F8077661A5ADBA0AE047D7BC026C3853DB714F61D5F57952645F489BDAF85DEFF0E170160DE12E1B80B61DED2CC4A48889314
70EA3290A3B1F328025D3B09579E01634756C7A51653B91AFC1582B6B68C5E6B077C199FEE3F59A696C1C44F6D6DC227F38427DF8027
DF84E99F1BD554496A371D2B644FE8D401761B2B7C7537D878B849EB6C6B13DE9B05B73F233CF69A61E51AB2C2085D3AC28431AAF5B6
DFC8D0A943064B5B9BD0568686FAB0CFD71E3A55F373AAA21C31DDFCBADADA846F5F293CFBAEC10E48AE65885FC60A15894623F37F35
5402C3FB97B9F95C43B7CEC57C5AE482D4921FA0DECD141BA98B9CDC5883B5BC8F29D3B89F93958939A6B5F84F7E5B7EBB2158EC6F41
4DC2475E151E790DDE9B65129324E979C76E91C425CF4C9B84B6494703CE809CB76FFAF78403270887ED6818D43B8369D92AE1C93784
DB9F85B73FCD980E0C7575C2C6C384FDB68149E30C7DBAE7856BE60278F24DE1B1D7D44FDC6D73F8CA36306628D4D755D2D84D6E7F14
0DA26E7EBF3F624219CAEBD7F7C41BC273EF59029398D1AE560B98A6AEA213FFB5307381E1FAC785D3F7ABE4E75AB4959B8A7C4F11A9
1E75ACC6A8ED49FF2E6DF9BF919A5A84573E10EE78069E9BAE7E16C6F8961E6EB4301530FBC1E220197B387C030823070A47ED02FB8E
37F4E892D1FDB3F970D773C2832F1BE62FD351D956D6B707ECBEB970D004C398E15057CACA7DB1049E7E4B78F85578EB53352BADA00B
42A70638684299EF1F5AA24343751720C5E63FCE0B92FA54AD6D70E29F84373F2991B318DBDD268AB9D36E8F2EC243BF84AE9DFE7DB8
8468581F4238A61EBF8CC0F99A2826C83E80A1E7180CB59821ED356957340A0FBF22DC36153E9C6310D14857DAE158CD653354BF25BF2
8294DA3149FE547925C3ED26C3D53C9C38D6D0B9A3D2BFB54D78E363E1962930E52DC39A168349C4A464D4673B647BD8653343CFAE99
B5B06A8DF0DC7BC27D2FC2CB1F185635194AE9086BA8AF13361921ECB639ECBCA96158FF12F575790DD31E93303418C7DEB5B6C1D255
B074A5B060392C582A2C59A9EF96AC84E5AB85C6E612CD2DB0A6594DDC9636F8745E323C88496867A38AD6A2C8FC3593FC75E2213A28
```

Sự giống nhau này tiếp tục cho đến gần cuối của 2 dữ liệu

Image:

```
A65CB82E70EF9EC0CD9BDF157ABC2B9A0751FFBFF615F5DCBDF4BD3C7C2BED83E83B195E5A852FD2050777B09E9E1724686A63B25198
EA46793704F322E60A1F4FA63AFBD94AA264EC4626D36410D597E9F0422F3B7204A8542C6C6FB3EBDA7EF4A2F8EA569F71F2C7E93A98
E26C5651364EFDF0219817351EBBD175B05EB77C05EB46E9502F37F56F99840B9EB4C47D8EFBF59B1E52904E9049539ECE938997A87E
4C6D8365E1F57A745B930EBA95CFC4C77E938E5796CB095CBB46181D254C4D599103B35B3EA3525C3F71CFCAF46E1AB051BA35A56E07
BB5766A219D5779C9CC109A8DB1F7090EFA09EFE05E6EE459ED7372F20000000000494544AE426082
Ln 2, Col 90371 100% Windows (CRLF) UTF-8
```

Signature:

```
EA46793704F322E60A1F4FA63AFBD94AA264EC4626D36410D597E9F0422F3B7204A8542C6C6FB3EBDA7EF4A2F8EA569F71F2C7E93A98
E26C5651364EFDF0219817351EBBD175B05EB77C05EB46E9502F37F56F99840B9EB4C47D8EFBF59B1E52904E9049539ECE938997A87E
4C6D8365E1F57A745B930EBA95CFC4C77E938E5796CB095CBB46181D254C4D599103B35B3EA3525C3F71CFCAF46E1AB051BA35A56E07
BB5766A219D5779C9CC109A8DB1F7090EFA09EFE05E6EE459ED7372F20000000000494544AE42608200CD27786ECAE50DC9AB4F0FF5
1D75DA8D38482DEA00F15C5A76369E11093AEFD62CAAA203819ECB05C1
Ln 2, Col 90455 100% Windows (CRLF) UTF-8
```

Và rõ ràng là signature có độ dài của dữ liệu lớn hơn image nên có thể suy ra phần signature là chuỗi khác nhau ở phía sau của các file signature dưới dạng ảnh sau khi bỏ đi phần prefix giống nhau.

Đoạn code dưới đây thực hiện việc đọc file uit.png dưới dạng base64 encode và dưới dạng binary sau đó lần lượt đọc các file chữ kí dưới dạng binary và lọc đi phần prefix giống nhau và verify bằng chữ kí và public key được đọc từ file.

```
int main(int argc, char *argv[])
{
    ECDSA<ECP, SHA1>::PublicKey key;
    LoadPublicKey("./ec.public.key", key);
    string prefix = ReadData("./uit.png");
    cout << "prefix: " << hex(prefix) << endl;
    string message = GetMessage("./uit.png");
    std::filesystem::path directoryPath = "./signature";
        if (std::filesystem::exists(directoryPath) && std::filesystem::is_directory(directoryPath)) {
        std::vector<std::string> filePaths;

        for (const auto& entry : std::filesystem::directory_iterator(directoryPath)) {
            if (entry.is_regular_file()) {
                filePaths.push_back(entry.path().string());
            }
        }

        // Access the file paths in the vector for further processing
        for (const auto& filePath : filePaths) {
            string signature = ReadData(filePath);
            removePrefix(signature, prefix);
            if (VerifyMessage(key, message, signature))
                cout << "Yayyyy valid signature in: " << filePath << endl;
            else
                cout << "Invalid" << endl;
        }
    }
    else {
        std::cout << "Invalid directory path." << std::endl;
        return 1;
    }
}
```

Chạy code:

```
Invalid
ya sua, valid signature in: ./signature\3449febf4cfb168d1002d84e0da97a56.png
ya sua, valid signature in: ./signature\5ecf653a589b3980a8a166e404069edc.png
ya sua, valid signature in: ./signature\62b7f87696d98b562540881d83fdba18.png
Invalid
Invalid
Invalid
Invalid
ya sua, valid signature in: ./signature\d27a1a4ae33ab2dc6a510b4093c270ce.png
Invalid
```

Có 4 chữ kí được verify

8. Câu hỏi 8

Bài này trước khi sử lý, hàm hash chia message ra làm các block 32 bytes. Nếu 2 đoạn message chỉ có độ dài sau khi chia ra là 1 block thì hàm hash sẽ an toàn, tuy nhiên nếu có ít nhất 1 message lớn hơn 1 block thì sẽ xảy ra xung đột. Trong bài này, em chia ra làm 2 message 1 block và 2 blocks.

Các từ viết tắt:

scr: scramble_block() function

mix_: block sau khi đi qua hàm scramble

Init: initial state

a1: block thứ 1 của message a

b1, b2: block thứ 1, 2 của message b

f: thực hiện một xáo trộn của block thứ 2

Đề bài yêu cầu: hash1 = hash2

```
for i,b in enumerate(msg_blocks):
    mix_in = scramble_block(b)
    for _ in range(i):
        mix_in = rotate_right(mix_in, i+11)
        mix_in = xor(mix_in, X_bytes)
        mix_in = rotate_left(mix_in, i+6)
    if (i % 2 != 0):
        mix_in = xor(mix_in, rotate_right(mix_in, 12))
    initial_state = xor(initial_state,mix_in)
    initial_state = scramble_block(initial_state)
```

=> scr(mix_a1 ^ init) = scr(scr(mix_b1 ^ init) ^ f(mix_b2))

=>mix_a1 ^ init = scr(mix_b1 ^ init) ^ f(mix_b2))

Sau đó em sẽ chọn a1 và b2 tuy nhiên chỉ chọn 31 bytes do hàm pad nếu đủ 32 bytes thì sẽ pad thêm 32 bytes vào nữa

```
def pad(data):
    padding_len = (BLOCK_SIZE - len(data)) % BLOCK_SIZE
    if padding_len == 0:
        padding_len = BLOCK_SIZE
    return data + bytes([padding_len]*padding_len)
```

Ta có thể tính được b1 sau khi chọn ra a1 và b2 như code bên dưới:

```
initial_state = xor(Y_bytes, Z_bytes)

bl_a1 = os.urandom(31)
bl_a1_pad = pad(bl_a1)

mix_in_a1 = scramble_block(bl_a1_pad)
hash1 = xor(mix_in_a1, initial_state)

bl_b2 = os.urandom(31)
bl_b2_pad = pad(bl_b2)
mix_in_b2 = scramble_block(bl_b2_pad)
mix_in_b2 = rotate_right(mix_in_b2, 1+11)
mix_in_b2 = xor(mix_in_b2, X_bytes)
mix_in_b2 = rotate_left(mix_in_b2, 1+6)
mix_in_b2 = xor(mix_in_b2, rotate_right(mix_in_b2,12))

scr_mix_in_b1 = xor(hash1, mix_in_b2)
mix_in_b1 = reverse_scramble_block(scr_mix_in_b1)
mix_in_b1 = xor(mix_in_b1, initial_state)
bl_b1 = reverse_scramble_block(mix_in_b1)


mess1 = bl_a1.hex()
mess2 = (bl_b1+bl_b2).hex()
```

Chạy code:

```
> py .\baitap_8.py
m1:  6233ac616ba75956b5b16377269777c033c799c0ef30e52496136acb6c2c39
m2:  ab882558cefb8060834c0422c46d6b2df741fbe574298d38974ebc4e8a0ece0aeaf2f30388acfd2c9a6f
7f0617909004821e789775372c24adaae2cc9fd7e7
Please enter a hex encoded messages m1:
6233ac616ba75956b5b16377269777c033c799c0ef30e52496136acb6c2c39
Enter a hex encoded messages m2:
ab882558cefb8060834c0422c46d6b2df741fbe574298d38974ebc4e8a0ece0aeaf2f30388acfd2c9a6f7f061
7909004821e789775372c24adaae2cc9fd7e7
----
hash m1: f1e3e8e88966818ec03b0ece86c2f3eb0041b7d4b32f28cbc33f39e75b54aa6c
hash m2: f1e3e8e88966818ec03b0ece86c2f3eb0041b7d4b32f28cbc33f39e75b54aa6c
----
Congratulations on finding the collision value f1e3e8e88966818ec03b0ece86c2f3eb0041b7d4b3
2f28cbc33f39e75b54aa6c
quanp ~\code\Cryptography_Lab6\nt219-n21-antn\nt219-n21-antn  12.129s
>
```

9. Kịch bản 03

- Tài nguyên:
- Mô tả/mục tiêu:
- Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).

  *Ví dụ: [*NT101.K11.ANTT*]-Session1_Group3.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT