Many time pad attack

Le Huu Hai 20215570

March 2024

1 Ý tưởng

Giả sử $y_1,\ y_2$ là các bản mã, được mã hóa từ bản rõ $x_1,\ x_2$ bởi cùng 1 khóa k.
Vì vậy:

$$y_1 = x_1 \oplus k$$

$$y_2 = x_2 \oplus k$$

Khi xor hai bản mã, ta sẽ có thông tin về các bản rõ:

$$\begin{aligned} y_1 \oplus y_2 &= (x_1 \oplus k) \oplus (x_2 \oplus k) \\ &= (x_1 \oplus x_2) \oplus (k \oplus k) \\ &= x_1 \oplus x_2 \\ &:= z \end{aligned} \tag{1}$$

Đặt $x'_1,\ x'_2$ là các dự đoán của $y_1,\ y_2$. Bằng cách đoán x'_1, ta có thể tính ra x'_2 thông qua phép xor và kiểm tra cặp $x'_1,\ x'_2$ này bằng ngữ pháp tiếng Anh.

$$x'_2 = x'_1 \oplus z$$

Tổng kết, các bước thực hiện như sau:

- Xor các bản mã với nhau được tập các bản (gọi là z).
- Dự đoán từ có thể xuất hiện trong bản rõ (các từ có tần suất xuất hiện nhiều trong tiếng Anh). Các từ này nên có độ dài lớn nhất có thể để dễ dàng đoán từ được xuất hiện tiếp theo (có thể thêm dấu cách vào trước và sau từ dự đoán).
- Sau khi xor từ này với các bản z, lại dự đoán từ tiếp theo dựa trên kết quả và lặp lại bước trên.

2 Thực hiện

2.1 Chuẩn bị bản mã

```
# 10 bản mã
ciphertext = [
    '315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cdf2d3aff021dfff5b403b510d0d0455468aeb98622b137dae857553ccd8883a7bc37520e06e515d22c954eba5
    '234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef4081ffde5ad46a5069d9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919cf3a9f1cb7
    '32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e51d04e0e7b207b70b9b8261112bacb6c866a232dfe257527dc29398f5f3251a0d47e503c66e935de8
    '32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad3306ef5021eafe1ac01a81197847a5c68a1b78769a37bc8f4575432c198ccb4ef63590256e305cd3a9544ee4
    '3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d9607cee021dafe1e001b21ade877a5e68bea88d61b93ac5ee0d562e8e9582f5ef375f0a4ae20ed86e935de8
    '32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8de287be84d07e7e9a30ee714979c7e1123a8bd9822a33ecaf512472e8e8f8db3f9635c1949e640c621854eba0
    '32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba47b01c909ba7696cf606ef40c04afe1ac0aa8148dd066592ded9f8774b529c7ea125d298e8883f5e9305f4b44f915cb2bd05af5
    '315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54ce56801d943ba708b8a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f89ee04132ecb9298f5fd2d5e4b45e40ecc3b9d59e9
    '271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4aad6e04d513e96d99de2569bc5e50eeeca709b50a8a987f4264edb6896fb537d0a716132ddc938fb0f836480e06ed0fcd6e9759f40
    '466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b3399ccfafb61d0315fca0a314be138a9f32503bedac8067f03adbf3575c3b8edc9ba7f537530541ab0f9f3cd04ff50
]
# bản mã cần dịch
target = '32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b57bddef121336cb85ccb8f3315f4b52e301d16e9
```

Hình 1: Các bản mã

2.2 Các hàm sử dụng

```
1 usage
def dec_to_string(dec_value):
    hex_value = hex(dec_value).split('x')[-1]
    return hex_to_string(hex_value)


1 usage
```

Hình 2: Hàm chuyển đổi số nguyên sang kí tự ASCII

```
2 usages
def hex_to_string(hex_str):
    ans = ""
    for i in range(0, len(hex_str), 2):
        dec = int(hex_str[i:i + 2], 16)
        ans += chr(dec)
    return ans
```

Hình 3: Hàm chuyển đổi số Hexa sang kí tự ASCII

```
1 usage
def string_to_hex(str):
    ans = ''
    for c in str:
        dec = ord(c)
        hexa = hex(dec).split('x')[-1]
        if len(hexa) < 2:
            hexa = '0' + hexa
        ans += hexa
    return ans
```

Hình 4: Hàm chuyển đổi chuỗi kí tự ASCII sang chuỗi Hexa

```
def xor_two_cipher(str1, str2=target):
    length = min(len(str1), len(str2))
    xor_two_cipher_hex = ''
    for i in range(0, length, 2):
        xor_dec_value = int(str1[i: i + 2], 16) ^ int(str2[i: i + 2], 16)
        xor_hex_str = hex(xor_dec_value).split('x')[-1]
        if len(xor_hex_str) < 2:
            xor_hex_str = '0' + xor_hex_str
        xor_two_cipher_hex += xor_hex_str
    return xor_two_cipher_hex
```

Hình 5: Hàm XOR các bản mã

```
1 usage
def xor_with_guess(guess_str, str1):
    guess_hex = string_to_hex(guess_str)
    for i in range(len(str1)):
        xor_with_guess_hex = xor_two_cipher(guess_hex, str1)
        xor_with_guess_str = hex_to_string(xor_with_guess_hex)
        if len(xor_with_guess_str) == 0:
            continue
        print(str(i) + ":'" + xor_with_guess_str + "'", end=" ")
    print()
    return
```

Hình 6: Hàm đoán

2.3 Giải mã

Tạo mảng các bản z bằng cách sử dụng hàm XOR hai bản mã. Mục tiêu là giải mã bản mã "target" nên ta chỉ XOR bản mã này với 10 bản mã còn lại.

```
xor_list = []
for s in ciphertext:
    str_hex = xor_two_cipher(s)
    xor_list.append(str_hex)
```

Hình 7: Tạo mảng các bản z

```
1| 030d4543120b4314041754021753070902454e1c1e58452548545b00021a1d06475114411d000708410e4f0e19051c00005f0e4e32134511411b53044c070745460a060d4f1f4f060d4554061403420a1c4354
2| 111d09450145141d1018444d15011c0306074c10535f4e3d071c4e541d121d4e094f16001b1d01451505450c1b150545104943010813165241551000a521a0d17001811164e084f1d0300171a181e54004e4e45
3| 000000001d0c0017450048040b145300050a551d537145321109015155 1a1a4e094f160004115206131450170617 1a040244451c165606134e55171749020d45414b0916544d0014454615060217000c0f1116
4| 00000000100c131a000654081d075311150a441c105f44770a1c4e4155040c0f0c00044e10060b1515044f0d491104021d5e491a0d1b451e4f1a181600151b4547040a1d000c1c5206490400041c540a161745
5| 0d071000170a0d551154570c0b075315084542 1c0a1a41771b001a001a15490d0652414b160d0145071f4f0e491148020755001 90d1945015010100c4118011f451845104e4d1c06004118010f09000c0f1116
6| 000000521645020000545 41a0a530718170053491c5c00341a1c1e541a141b0f174818005e54060d0019001401190b0d525b490209560e174505531645171a005418450a410b0a5203521b0541174f1a1c4309
7| 000000521645020000545 41a0a530718170053491c5c003411151a4f1201081e0f595b001c1a1745105411749110409 1d5b534e111e0052671a0500521a05004e1f450d4f4d1a010000161a141a454f080c17
8| 030d4543120b4301001100190d165311080c4e1d534d48321a004e541d16490d0f4911001a0752100f0541131909480c140c414e12040a1c4755110c5454011600180017544d0e1c010017070f1d55020b1045
9| 15484d50010c151311110d06000a5a4147004e0a01435023010a00000610010b0a4541530715060012 4d1343081c0f0a0045540608054952 4e141e004c0d4804001b171643080b071745540e0e1c00080b0d00
10| 743c0d4553260c1c061d5308453c0b070817442d1a59543e070b0f520c53415c571057095310178acdec4e061a500b170b5c54014517165254 1d164541061c454f0d4559571f06060c4e13480e4e524f1d0c09
```

Hình 8: Các bản mã z

Bắt đầu bằng từ có tần suất xuất hiện cao nhất: "the" và "The". Ta đem từ này XOR với các bản z.

```
i=0
for list in xor_list:
    i+=1
    print(i, end_='| ')
    xor_with_guess( guess_str: "the", list)
```

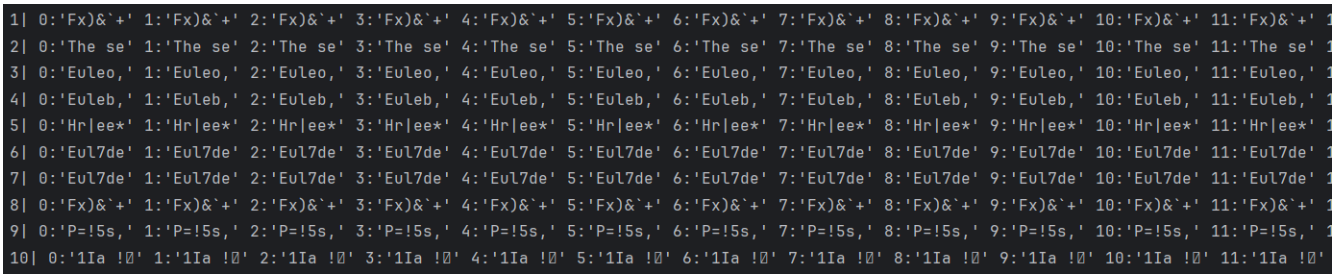
Hình 9: Thử XOR từ 'the' với các bản z

Quan sát thấy các bản z đều cho ra các kí tự có thể đọc được ở vị trí '0', nên từ "The" có thể đã xuất hiện trong bản rõ của bản mã "target" hoặc bản rõ của 1 trong 10 bản mã "cipher_text". Nhưng vì các kết quả từ các bản z là khác nhau, cộng với bản z là XOR của bản "target" với 1 bản mã bất kì, có thể kết luận "The" thuộc về bản rõ của "target".

```
1| 0:'We ' 1:'We ' 2:'We ' 3:'We ' 4:'We ' 5:'We ' 6:'We ' 7:'We ' 8:'We ' 9:'We ' 10:'We ' 11:'We ' 12:'We ' 13:'We ' 14:'We ' 15:'W
2| 0:'Eul' 1:'Eul' 2:'Eul' 3:'Eul' 4:'Eul' 5:'Eul' 6:'Eul' 7:'Eul' 8:'Eul' 9:'Eul' 10:'Eul' 11:'Eul' 12:'Eul' 13:'Eul' 14:'Eul' 15:'E
3| 0:'The' 1:'The' 2:'The' 3:'The' 4:'The' 5:'The' 6:'The' 7:'The' 8:'The' 9:'The' 10:'The' 11:'The' 12:'The' 13:'The' 14:'The' 15:'T
4| 0:'The' 1:'The' 2:'The' 3:'The' 4:'The' 5:'The' 6:'The' 7:'The' 8:'The' 9:'The' 10:'The' 11:'The' 12:'The' 13:'The' 14:'The' 15:'T
5| 0:'You' 1:'You' 2:'You' 3:'You' 4:'You' 5:'You' 6:'You' 7:'You' 8:'You' 9:'You' 10:'You' 11:'You' 12:'You' 13:'You' 14:'You' 15:'Y
6| 0:'The' 1:'The' 2:'The' 3:'The' 4:'The' 5:'The' 6:'The' 7:'The' 8:'The' 9:'The' 10:'The' 11:'The' 12:'The' 13:'The' 14:'The' 15:'T
7| 0:'The' 1:'The' 2:'The' 3:'The' 4:'The' 5:'The' 6:'The' 7:'The' 8:'The' 9:'The' 10:'The' 11:'The' 12:'The' 13:'The' 14:'The' 15:'T
8| 0:'We ' 1:'We ' 2:'We ' 3:'We ' 4:'We ' 5:'We ' 6:'We ' 7:'We ' 8:'We ' 9:'We ' 10:'We ' 11:'We ' 12:'We ' 13:'We ' 14:'We ' 15:'W
9| 0:'A (' 1:'A (' 2:'A (' 3:'A (' 4:'A (' 5:'A (' 6:'A (' 7:'A (' 8:'A (' 9:'A (' 10:'A (' 11:'A (' 12:'A (' 13:'A (' 14:'A (' 15:'A
10| 0:' Th' 1:' Th' 2:' Th' 3:' Th' 4:' Th' 5:' Th' 6:' Th' 7:' Th' 8:' Th' 9:' Th' 10:' Th' 11:' Th' 12:' Th' 13:' Th' 14:' Th' 15:'
```

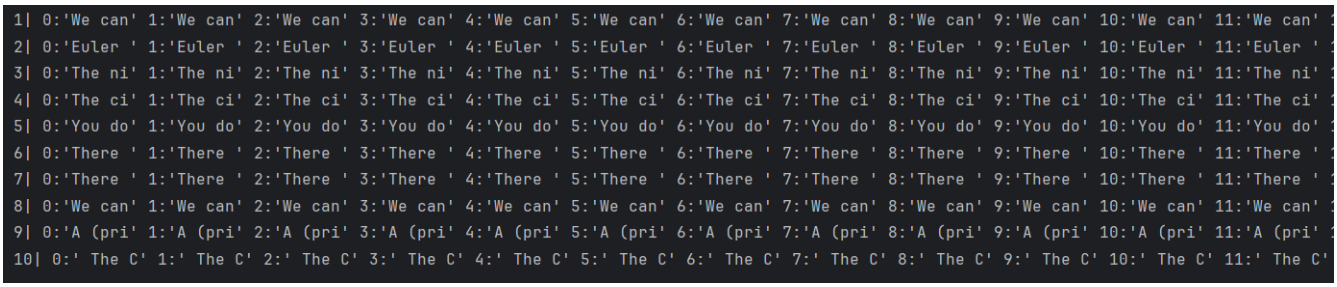
Hình 10: Kết quả khi XOR từ 'The' với các bản z

Trong kết quả của bản z thứ 2, ta thử mở rộng nó thành "Euler " và tiếp tục sử dụng hàm "xor_with_guess". Kết quả thu được ở dòng 2 sẽ là bản rõ của "target"



Hình 11: Mở rộng bản rõ của "target"

Bản rõ của "target" được mở rộng thành "The se". Tiếp tục sử dụng hàm "xor_with_guess" để mở rộng bản rõ của các bản z còn lại.

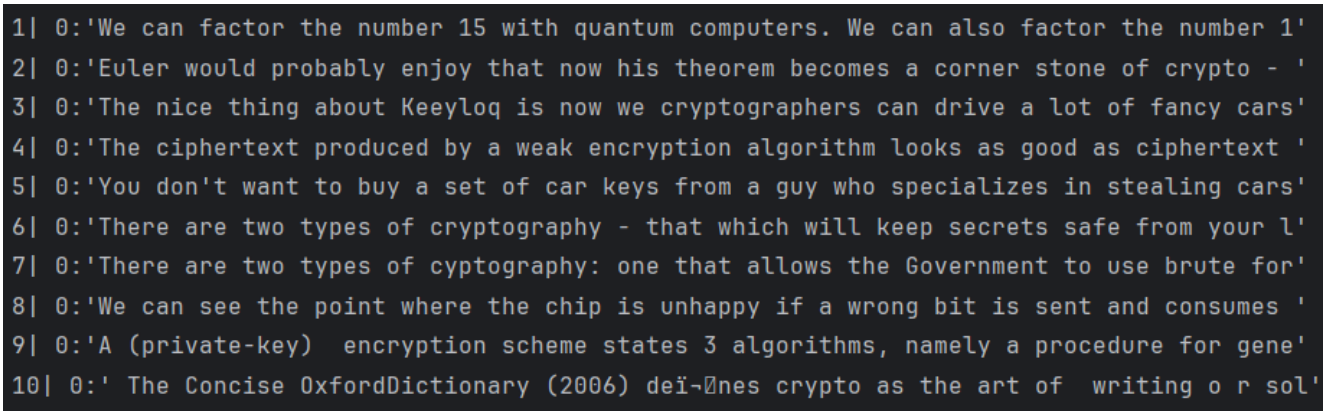


Hình 12: Mở rộng bản rõ của các bản z

Lặp lại các bước đoán và mở rộng đến khi đạt được độ dài của bản mã "target".

2.4 Kết quả

Bản rõ của "target":
"The secret message is: When using a stream cipher, never use the key more than once"
Bản rõ của các ciphertext (tính đến độ dài của "target"):



Hình 13: Các bản rõ