Họ và tên: Nguyễn Trung Kiên
MSSV: 20205090
IT-E6-K65

Tấn công Many-Time Pad

Dữ liệu đầu vào gồm 11 bản mã được mã hóa bơi 1 key 1024 bytes. Mỗi bản mã được tạo bằng cách sử dụng hàm strxor() key với plaintext có sẵn.

$$plaintext[i] \oplus key = ciphertext[i]$$

Khai báo:

1. ciphertext[] chứa 11 bản mã đầu vào.
2. target chứa bản nã cần giải.

```
import binascii

ciphertexts = [
     "315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cdf2d3aff021dfff5b403b510d0d0455468aeb98622b137c
     "234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef4081ffde5ad46a5069d9f7f543bedb9c861bf29c
     "32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e51d04e0e7b207b70b9b8261112bacb6c866a232c
     "32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad3306ef5021eafe1ac01a81197847a5c68a1b78769a37bc
     "3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d9607cee021dafe1e001b21ade877a5e68bea88d61b93ac
     "32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8de287be84d07e7e9a30ee714979c7e1123a8bd9822a33ec
     "32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba47b01c909ba7696cf606ef40c04afe1ac0aa8148dd066592ded9f8774b529c
     "315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54ce56801d943ba708b8a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f8
     "271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4aad6e04d513e96d99de2569bc5e50eeeca709b50a8a987f4264edb6896fb537c
     "466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b3399ccfafb61d0315fca0a314be138a9f32503bedac8067f03ac
     "32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b57bc
 ]

target = "32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b
 l= len(ciphertexts)
```

Sau đó ta chuyển đổi các chữ số hexadecimal thành các ký tự tương ứng asscii:

$$ciphertexts = [binascii.unhexlify(x) \text{ for } x \text{ in } ciphertexts]$$

Do thuật toán XOR, nếu nhiều dữ liệu được mã hóa bằng cùng một chuỗi khóa, kẻ tấn công có thể bẻ khóa dữ liệu mà không cần đoán chuỗi khóa.

Giả sử rằng A và B là các chuỗi dữ liệu văn bản thuần túy và key là khóa được sử dụng để mã hóa dữ liệu A và B. Dữ liệu được mã hóa A được đại diện bởi EA (EA = A_key), dữ liệu được mã hóa B bởi EB (EB = B_key) và ký hiệu bằng phép toán XOR.

Sử dụng cùng một giá trị kết quả XOR,ta có thể nhận được:

$$EA \oplus EB = (A \oplus key) \oplus (B \oplus key) = A \oplus B \oplus (key \oplus key) = A \oplus B$$

Tất cả các chuỗi văn bản rõ có thể được giải mã miễn là bất kỳ chuỗi văn bản rõ nào được biết đến:

$$EA \oplus EB \oplus B = A \oplus B \oplus B = A$$

Hàm strxor() để xor 2 string trên hệ hexadecimal trực tiếp, trả về kết quả có độ dài bằng độ dài string ngắn hơn.

```
# This method allows XOR operations on hexadecimal strings directly
def strxor(a, b):
    return "".join([chr(x ^ y) for x, y in zip(a, b)])

def stror1(a, b):
    if len(a) > len(b):
       return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a[:len(b)], b)])
    else:
       return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b[:len(a)])])
```

Đối với mỗi văn bản mật mã và văn bản mã còn lại, thống kê phân bố xác suất của vị trí tương ứng trong các bản ghi ở trên về các vị trí có thể, sử dụng trên 70% ở đây cho chính xác, ghi vị trí tương ứng vào khóa.

```
# Khả năng xuất hiện vị trí space
def space_possibility(s):
    newspace = []
    for position in range(400):
        count = 0
        for idx_i in range(l - 1):
            if position in s[idx_i]:
                count += 1
        if count > 7:
            newspace.append(position)
    return newspace
```

Trong bảng ASCII, phạm vi khoảng trắng trong 16-digit là 0x20, phạm vi 16 chữ số A-Z là 0x41-0x5A và phạm vi A-Z trong 16-digit là 0x61 - 0x7A. Ta có thể thấy rằng một chữ cái viết hoa khác biệt với một khoảng trắng hoặc có thể nhận được một chữ cái viết thường tương ứng; một ký tự thường khác biệt với một khoảng trắng hoặc có thể nhận được một ký tự viết hoa tương ứng.

Đối với các chuỗi dữ liệu loại trừ hoặc tiếp theo theo từng cặp, ta xác định vị trí [A-Z] hoặc [a-z] hoặc [0] và vị trí của mỗi chuỗi có thể là khoảng trắng trong văn bản thuần túy.

```
def find_letter(s):
    #Trả về vị trí của một chữ cái đã cho trong một chuỗi đã cho
    position = []
    for index in range(len(s)):
        if (s[index] >= 'A' and s[index] <= 'Z') or (s[index] >= 'a' and s[index] <= 'z') or s[index] == chr(0):
            # space_counts[index] += 1
            position.append(index)
    return position
```

Từ đó, chúng ta có thể thực hiện các phép toán exclusive or exclusive operations (xor) trên 11 cặp bản mã để xác định các vị trí có thể có của các khoảng trắng trong các bản rõ khác nhau.

$$EA \oplus EB = (A \oplus key) \oplus (B \oplus key) = A \oplus B \oplus (key \oplus key) = A \oplus B$$

Khi có đủ thông tin vị trí, thông tin chính cho vị trí tương ứng có thể được lấy bằng cách độc quyền hoặc thao tác với bản mã và dấu cách tại vị trí tương ứng (tại thời điểm này m = space nên c_space = m_key_space = key).

```
# Vị trí có thể tìm thấy các ký tự khoảng trắng
def find_space(ciphertexts):
    space_possib = [] # Lưu các khoảng trắng có thể
    # Dùng vòng lặp kép cho exclusive operations
    for ciphertext_i in ciphertexts:
        space = []
        for ciphertext_j in ciphertexts:
            if ciphertext_i == ciphertext_j:
                continue
            xor_res = strxor(ciphertext_i, ciphertext_j)
            # Ghi lại vị trí của các ký tự khoảng trắng có thể có trong mỗi plaintext
            space.append(find_letter(xor_res))

        space_possib.append(space_possibility(space))
    return space_possib
```

Sử dụng space location và space của mỗi dữ liệu được đề cập ở trên, sau đó thực hiện XOR và đặt vào vị trí khóa tương ứng, và cuối cùng ta có key.

```
# Tính toán để nhận được khóa tương ứng
def m_key(cipher):
    space_possib = find_space(cipher)
    key = [0] * 200
    for cipher_idx in range(11):
        for position in range(len(space_possib[cipher_idx])):
            idx = space_possib[cipher_idx][position] #Hexadecimal, biểu diễn bởi 2 bits
            a = cipher[cipher_idx][idx]
            key[space_possib[cipher_idx][position]] = a ^ ord(" ") # key (ascii code)

    key_str = ""
    for k in key:
        key_str += chr(k)    #chuyển hệ
    return key_str
```

Kết quả của chương trình: result = *'the secret message is: When using a stream cipher, never use the key more than once'*