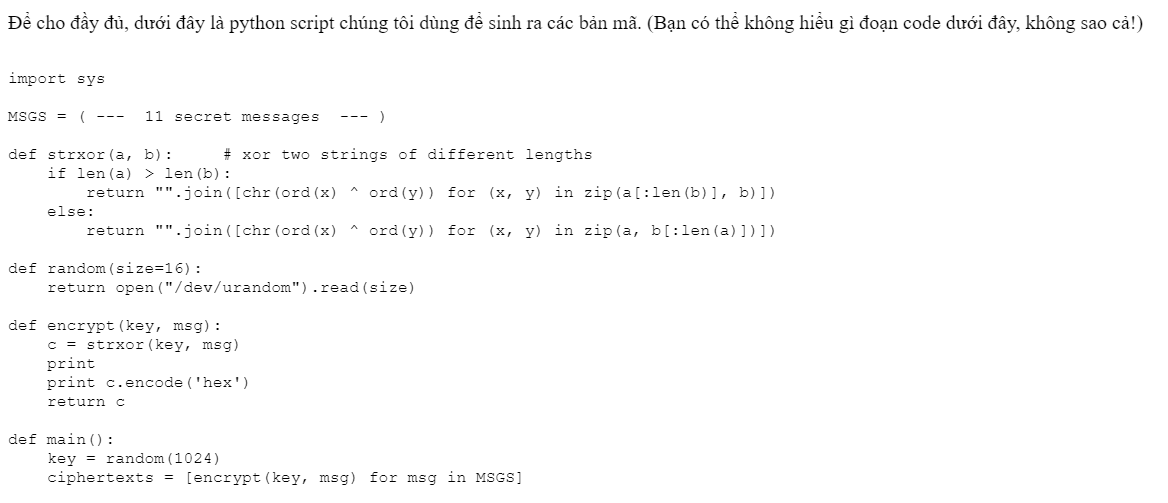
Project 1 – Mật mã ứng dụng

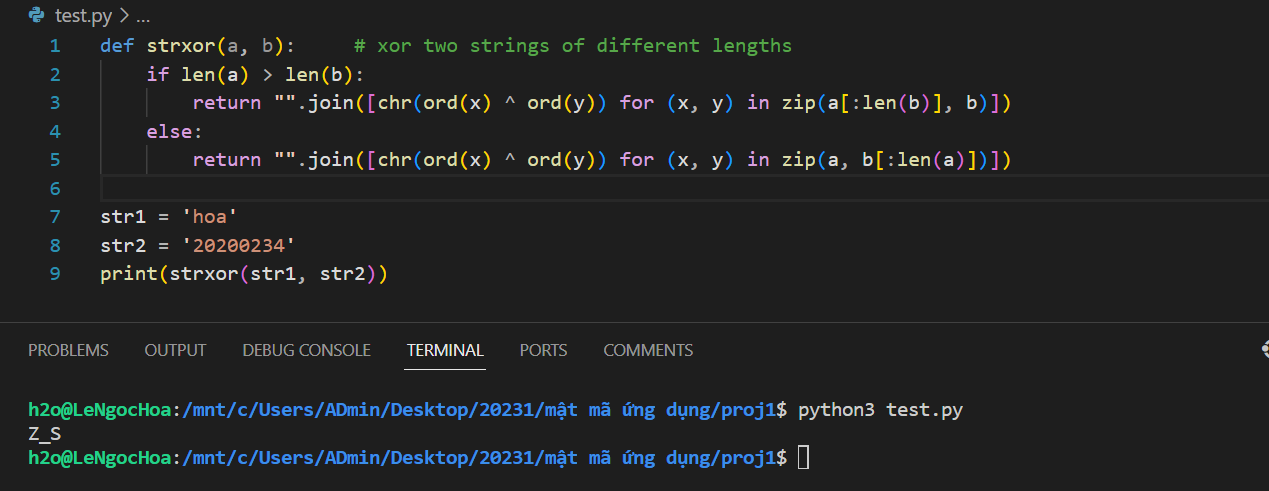
Lê Ngọc Hoa – 20200234

1. Phân tích đề bài

Mật mã many times pad – Sử dụng một key để mã hóa (trường hợp này là XOR) cho nhiều message. Mã nguồn được đưa ra bởi đề bài:

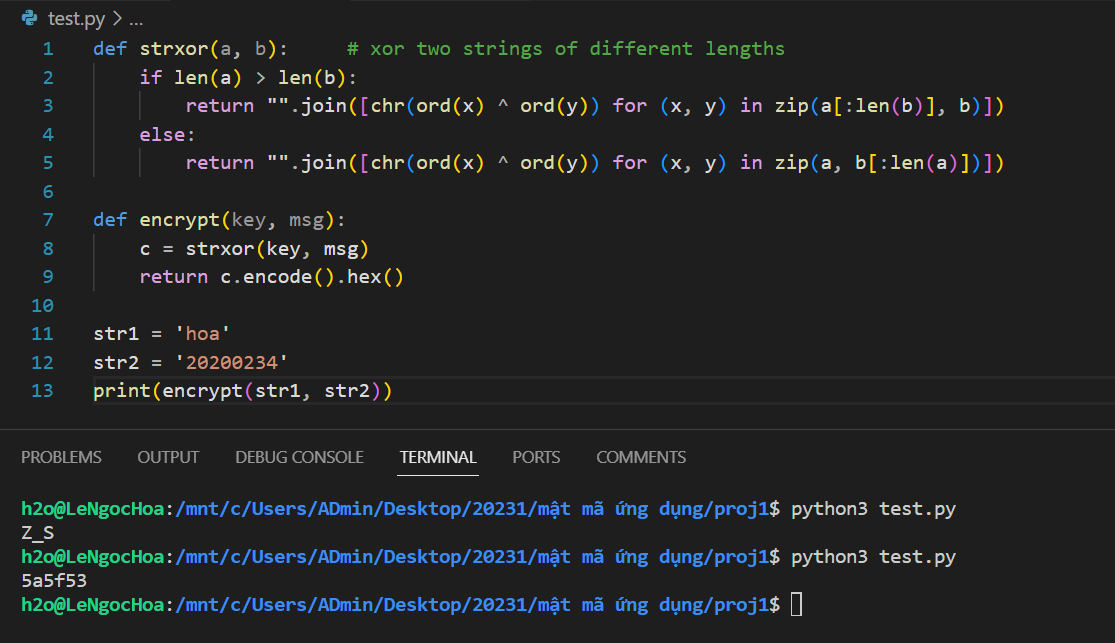


Hàm strxor() thực hiện XOR hai chuỗi với nhau, kết quả có độ dài bằng độ dài của chuỗi nhỏ hơn.



Hàm random() được sử dụng tương tự với vai trò sinh key ngẫu nhiên, đề bài sử dụng key có độ lớn 1024

Hàm encrypt() sử dụng hàm strxor() sau đó mã hóa kết quả thành dạng hex (em có code lại cho phù hợp với ngôn ngữ python3 và thư viện quen thuộc)



Tổng hợp lại, toàn bộ mã nguồn đề bài thực hiện sinh một khóa ngẫu nhiên có độ lớn 1024 và mã hóa 11 message (đều sử dụng key này) lưu trong tuple MSGS, trong đó message thứ 11 chính là mật mã cần tìm.

1. Ý tưởng, hướng thực hiện

Chúng ta xem xét kết quả khi XOR hai ký tự c1, c2 với nhau khi chúng thuộc [A-Za-z] hoặc là dấu cách. Ở đây, ký hiệu mã thập phân trong ASCII của [a-z] là 97-122, [A-Z] là 65-90, dấu cách (space) là 32. Chúng ta có bảng sau biểu diễn kết quả dạng mã thập phân của hai ký tự c1 và c2 khi XOR với nhau:

|  |  |  |  |
| --- | --- | --- | --- |
| ⊕ | A-Z (65-90) | a-z (97-122) | Space (32) |
| A-Z (65-90) | ≤32 | ≤64 | ≥65 |
| a-z (97-122) | ≤64 | ≤32 | ≥65 |
| Space (32) | ≥65 | ≥65 | 0 |

Từ bảng trên có thể thấy một điều đặc biệt là, khi một trong hai ký tự c1 hoặc c2 là space thì kết quả phép XOR ≥65 và ngược lại cũng đúng. Trong trường hợp kết quả bằng 0 thì cả hai ký tự là space.

Tuy nhiên, cần chú ý một số trường hợp đặc biệt, ví dụ với x ^ y, nếu x là một ký tự đặc biệt, y là một ký tự alphabet, thì vẫn có thể xảy ra trường hợp x ^ y >= 65. Và nếu x = y thì x ^ y = 0.

Bài toán có thể chia làm hai bài toán nhỏ:

Bài toán 1: Tìm ra các vị trí là space trong tất cả các chuỗi.

Bài toán 2: Từ vị trí i của chuỗi x là space, sử dụng phép toán XOR tìm ra chính xác ký tự ở vị trí i của các chuỗi còn lại.

1. Giải quyết hai bài toán nhỏ

Trước hết, chúng ta để ý rằng chuỗi cần giải mã có độ dài 166, tức plaintext có độ dài 83, nên chúng ta chỉ cần các chuỗi còn lại có độ dài 166 ký tự đầu (ở dạng hex) là đủ, biến đổi 1 chút về list gồm các ciphertext:

A screen shot of a computer code

Description automatically generated

1. **Giải quyết bài toán 1**

Xét ký tự ở vị trí i của ciphertext c1 (plaintext a) và ciphertext c2 (plaintext b), khi XOR c1[i] và c2[i] với nhau, thì kết quả bằng:

c1[i] ^ c2[i] = (a[i] ^ key[i]) ^ (b[i] ^ key[i]) = a[i] ^ b[i]

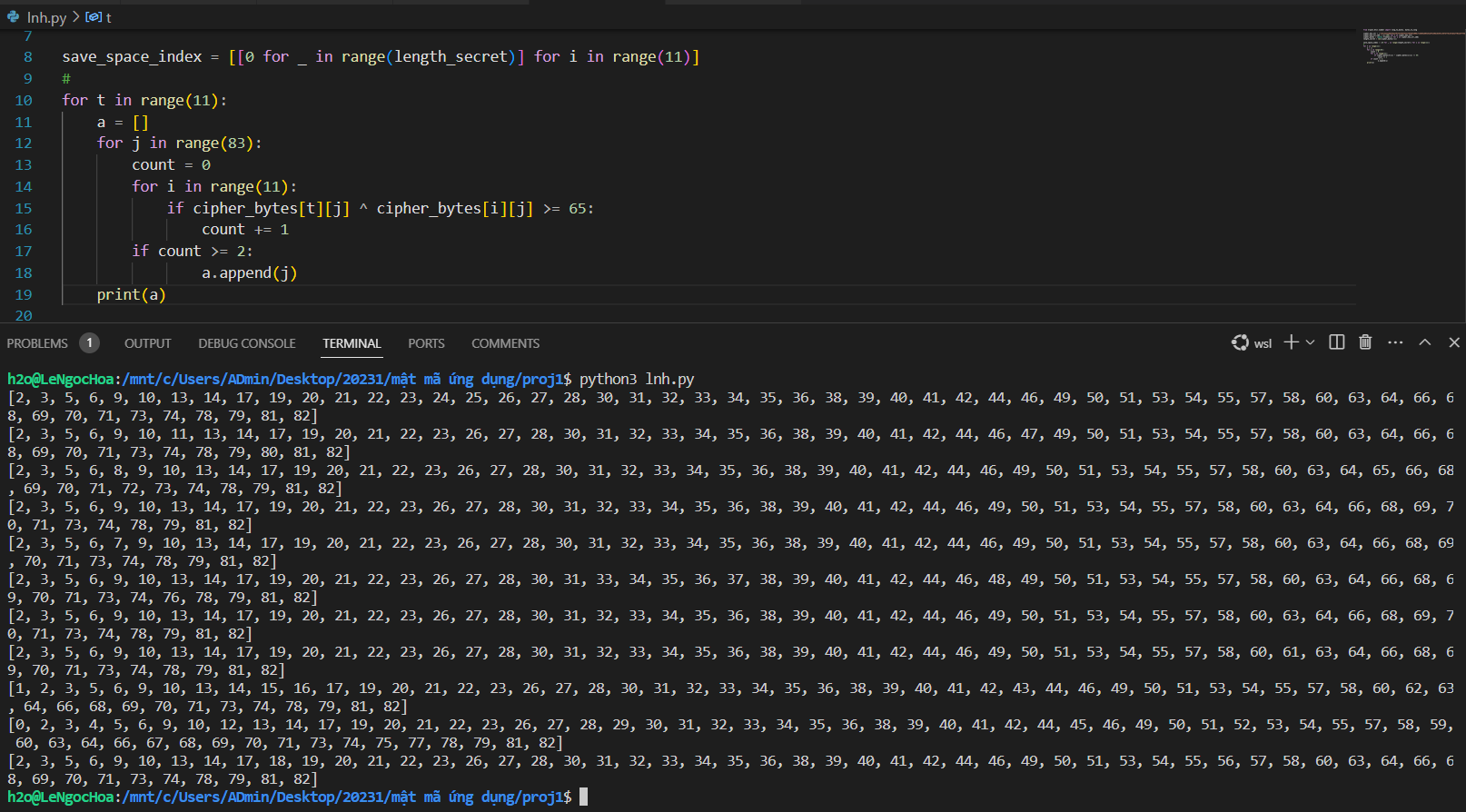
Nếu c1[i] ^ c2[i] có giá trị >= 65 thì có thể xảy ra một số trường hợp:

* Một trong hai ký tự a[i] và b[i] là space
* Một trong hai ký tự a[i] và b[i] là ký tự đặc biệt

Bởi vậy, cố định một chuỗi, để kiểm tra vị trí i của chuỗi này có phải space hay không, chúng ta thực hiện XOR lần lượt với các byte ở cùng vị trí i ở chuỗi khác, bằng cách đặt một biến count, đếm số lượng các chuỗi khác với vị trí i sau khi XOR có kết quả >= 65, chúng ta có thể vét cạn toàn bộ vị trị là space của chuỗi này (sẽ thừa) tùy vào giá trị count mà chúng ta quy ước.

Thử nghiệm với bộ input ciphertext cụ thể trong trường hợp này, khi giá trị count được chọn còn nhỏ, sau khi thử nghiệm, thì với điều kiện count >= x (x < 4) sẽ bị sót một số vị trí space. Khi chọn count >= 4 thì có thể vét cạn toàn bộ vị trí space nhưng bị thừa. Khi chọn count >= x (x > 4) thì sẽ giảm dần các vị trí space sai nhưng sẽ sót mất một số vị trí space đúng. Chẳng hạn:

Kết quả kiểm tra với điều kiện count >= 2, chúng ta có quá nhiều “vị trí space” (thừa nhiều)

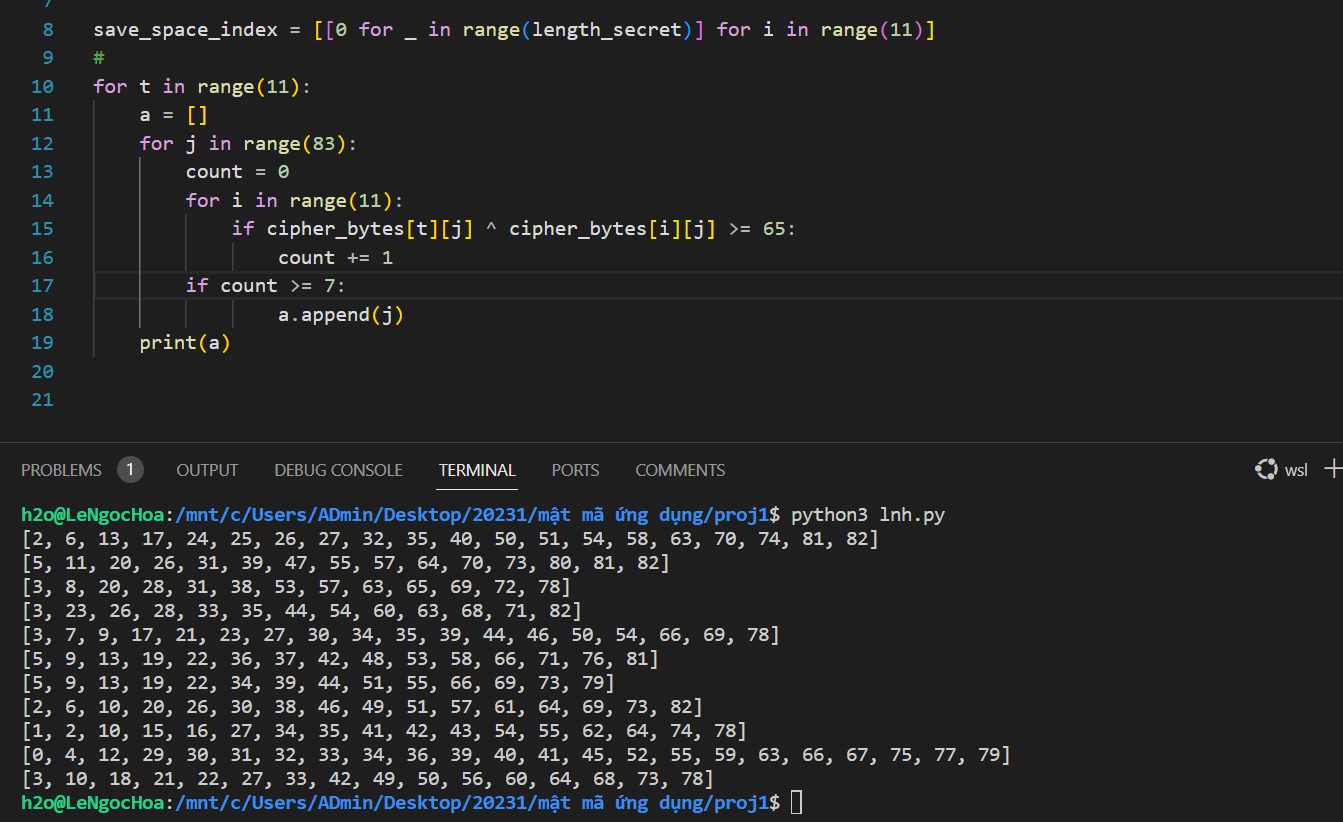


Kết quả kiểm tra với điều kiện count >= 4, số lượng vị trí space đã được giảm, tuy nhiên vẫn còn thừa một số vị trí sai

A screenshot of a computer program

Description automatically generated

Kết quả kiểm tra với điều kiện count >= 7, đã giảm đi nhiều vị trí space sai, tuy nhiên cũng “bỏ sót” một số vị trí space đúng.



Về cách lựa chọn giá trị x trong điều kiện kiểm tra count >= x, tùy thuộc vào tính chất mỗi bộ dữ liệu đầu vào chúng ta có thể chọn giá trị x khác nhau, chẳng hạn, khi message chủ yếu là các ký tự [a-Z] thì x có thể chọn nhỏ, khi message chứa nhiều ký tự lạ nằm ngoài [a-Z] như chữ số, ký tự đặc biệt thì chúng ta có thể chọn x lớn hơn. Nhìn chung, chúng ta có thể chọn x có giá trị ở tầm giữa số lượng message đầu vào nhằm đưa số lượng vị trí space bị thiếu và số lượng vị trí space bị thừa về mức tối thiểu.

1. **Giải quyết bài toán 2**

Trước hết chúng ta đã có danh sách vị trí các space của secret message, tiếp theo chỉ cần tìm ra các ký tự còn lại. Khi biết vị trí i ciphertext c1 (plaintext a) là space, tức a[i] là space, xét thêm ciphertext c2 (plaintext b), ta có:

c1[i] ^ c2[i] = (a[i] ^ key[i]) ^ (b[i] ^ key[i]) = a[i] ^ b[i]

Có: b[i] = c1[i] ^ c2[i] ^ 32

Bởi vậy, để tìm ký tự ở vị trí i của secret message, chúng ta tìm kiếm message j có vị trí i là space, tiến hành XOR ngược lại theo công thức trên.

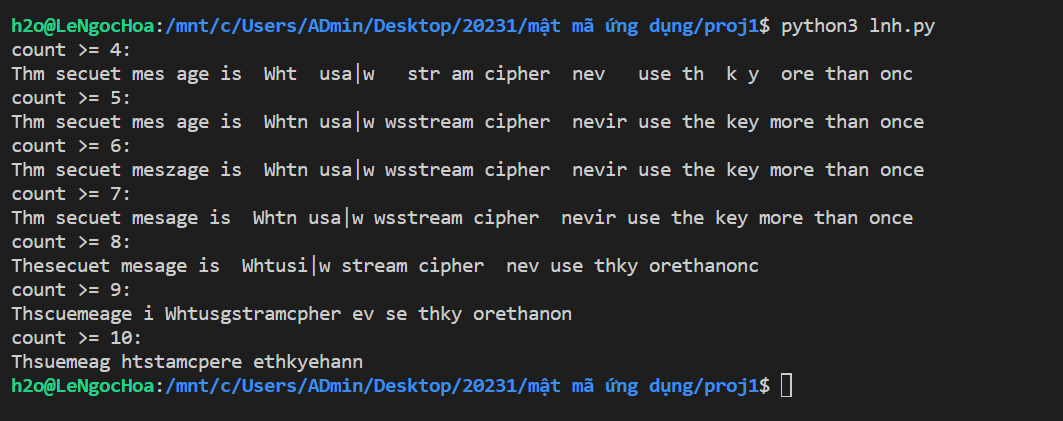
A computer screen shot of a computer code

Description automatically generated

Trong đó có một chút tối ưu bằng cách kiểm tra tính printable của ký tự temp được tìm ra khi đó.

1. Nhận xét kết quả

Kết quả chưa được tối ưu do cần giải quyết nhiều trường hợp về ký tự đặc biệt, đồng thời phụ thuộc vào tính chất cấu tạo của các message được mã hóa. Với một số điều kiện khác nhau về kiểm tra count >= x, chúng ta có các kết quả secret message sau khi giải mã thu được như sau:



Plaintext:

The secret message is: When using a stream cipher, never use the key more than once