암호분석 HW1

Substitution Cipher

20175204 김영범
2020년 4월 4일

Ex1 : 암호키에 대한 유효성을 확인하는 함수를 작성하고, 이를 이용하여 암복화하는 예제를 작성하여라

```
def Key_Validation(key):
    check_alpha_number = list() #먼저 알파벳의 숫자를 파악하기 위한 0으로 초기화된 26개의 배열을 만듭니다.
    for i in range(len(Alphabet)):
        check_alpha_number.append(0)# 0으로 초기화
    for char in key:#key안에 들어있는 알파벳을 하나씩 검사하고 순서를 이용해서 위에 선언한 배열에 1을 더해줍니다.
        alpha_idx = Alphabet.find(char.upper()) #(ex. b가 key에 있었다면 chek_alpha_number[2]번째 값에 1을 더해주는식)
        check_alpha_number[alpha_idx]  =  check_alpha_number[alpha_idx] + 1
        if(check_alpha_number[alpha_idx] >1):#만약 1을 넘어간다면 중복되는 알파벳이 키에 있다는 뜻이므로 에러메세지를 출력하게 합니다
            print("There are overlapping alphabet in key ")
    print("this key is safe for Substitution_Encryprion")# 아닌경우에는 치환암호를 구현할 key에 적합하다고 출력해줍니다
    return
```

```
~/vscode/Crypto_Analysis master*
❯ python3 EX1.py
this key is safe for Substitution_Encryprion
plain_txt =  hello world!
cipher_txt =  iqcct ftlch!
recovetxt =  hello world!
```

기존에 수업시간에 구현한 암복호화 함수들은 직접 구현해보았고, 추가적으로 암호키에 대한 유효성을 확인하는 Key_Validation 함수를 만들었습니다. 먼저 암호키에 대한 유효성을 판단하기 위해서는 다음과 같은 두가지 조건을 생각해야할 것입니다.

[1 : 암호화가 정상적으로 이루어져야 함.]
[2: 공격자가 키를 보았을때 쉽게 외우지 못하는 무작의성이 보장되어야함.]

1을 위해서는 사용자가 세팅한 26자리의 key문자열에 중복되는 알파벳이 없어야 할것입니다. 그걸 바탕으로 구현하였습니다. 위의 실행파일을 보면 “this key is safe for Substitution Encryption”이라 적혀있는데 이는 중복되어있는 알파벳이 없음을 의미합니다. 2를 위해서는 사용자가 세팅한 key에 대중적으로 사용된는 단어가 존재 하는지 확인해야합니다. 즉 키에 존재하는 알파벳을 2자리에서 26자리의 연속된 단어로 분할 하고, 단어사전과 대조해보아야합니다. 하지만 구현하는데 어려움에 봉착했습니다. 이는 앞으로 강의를 들으며 파이썬을 더 공부하여 구현하겠습니다.

Ex2 : 암호키를 랜덤하게 생성하는 함수를 작성하고, 이를 이용하여 암복호화하는 예제를 제시하라

```
def Key_generation():#알파벳 26개를 이용해서 키를 랜덤으로 만들어주는 함수입니다.
    temp = list() # random함수는 문자열에서 사용이 불가능 하니까 list를 이용하려고 변수를 넣었습니다.
    for i in Alphabet:# temp list에 알파벳을 하나씩 추가시킵니다.
        temp.append(i)
    random.shuffle(temp) #random.suffle함수를 이용해서 temp list에 있는 26개를 무작위로 섞어줍니다.
    usrkey = ''
    for j in temp:#usrkey를 선언하고 무작위로 섞인 값들을 대입시켜줍니다.
        usrkey += j
    return usrkey
```

```
~/vscode/Crypto_Analysis master*
❯ python3 EX2.py
this key is safe for Substitution_Encryprion
key           =  JPTSWOCEYABZMHXQINKULFGDVR
red_plain_txt =  hello world!
cipher_txt    =  ewzzx gxnzs!
recovetxt     =  hello world!
```

암호키를 랜덤하게 생성하는 함수 : Key_generation 를 만들었습니다. 알파벳 26개를 이용해서 키를 랜덤으로 만들어주는 함수입니다. 파이썬에는 random모듈이 존재합니다. 그중에서 random.suffle은 리스트에 있는 내용들을 전부 무작위로 섞어주는 역할을 합니다. 이를 이용하였습니다. 물론 Key_generation 함수를 이용해 key를 생성한 다음에는 HW1_Ex1에서 구현한 Key_Validation 함수를 이용해 생성한 key의 유효성을 판단하였습니다. 위의그림은 EX2.py의실행 화면입니다. 키생성 및 암복호화가 정상적으로 이루어짐을 확인할 수있습니다.

EX3. 텍스트 파일에서 평문을 읽어 암호문 파일을 만드는 예제와 이를 복호화 하는 예제를 작성하여라

인터넷에서 영어 공포소설의 앞부분을 발췌해 my_text.txt파일에 저장시킨후, 수업시간에 구현한 암복호화 함수, 그리고 Ex1,Ex2에서 구현한 키 생성,검증 함수를 이용하였습니다. 결과적으로 나온 파일은 my_cipher_msg.txt 파일과, recovered_msg.txt 파일 입니다.
Ex3.은 실행화면에서 출력되는 부분이 별도로 존재 하지않기 때문에 따로 사진으로 올리지 않겠습니다. 물론 my_text.txt 파일과 실행하면서 생긴 my_cipher_msg.txt 파일, recovered_msg.txt 파일은 별도 첨부파일에 있습니다.
항상 암호화를 실시할때 key를 랜덤하게 생성하기때문데, 실행시켰을때 만들어지는 my_cipher_msg.txt는 기존에 들어있는 내용과는 다를 수 있습니다.
하지만 복호화 시킨 recovered.msg.txt는 동일합니다.

감사합니다. :)