암호분석 HW1 2번-20192233 박진철

(a)

```python
#2-(a)
Alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

#단순치환암호 암호화
def subst_encrypt(key, msg):
    result = ''
    InSet = Alphabet
    OutSet = key
    for ch in msg:
        if ch.upper() in InSet:
            idx = InSet.find(ch.upper())
            if ch.isupper():
                result += OutSet[idx].upper()
            else:
                result += OutSet[idx].lower()
        else:
            result += ch
        #메시지에서 알파벳 위치의 문자를 동일한 위치의 키값으로 변경
    return result
```

```python
#단순치환암호 복호화
def subst_decrypt(key, msg):
    result = ''
    InSet = key
    OutSet = Alphabet
    for ch in msg:
        if ch.upper() in InSet:
            idx = InSet.find(ch.upper())
            if ch.isupper():
                result += OutSet[idx].upper()
            else:
                result += OutSet[idx].lower()
        else:
            result += ch
        #메시지에서 키값 위치의 문자를 동일한 위치의 알파벳으로 변경
    return result
#=============================================
```

단순치환암호의 암호화/복호화하는 함수들을 SubstLib.py 라이브러리에 저장

(b)

```python
#2-(b)
def randomkey(msg):
    alpha_list = list(msg)
    random.shuffle(alpha_list) #26개의 대문자를 랜덤하게 섞음
    shuffled = ''.join(alpha_list)
    return shuffled

def findright(msg):
    for i in range(0, len(msg)):
        for j in range(i+1, len(msg)):
            if msg[i] == msg[j]:
                return False
            else:
                return True
#============================================
```

1. randomkey: 받은 msg를 랜덤하게 섞는 함수

2. findright: 중복되는 값이 있는지 체크하고, 중복된 값이 있으면 False, 없으면 True 반환

(c)

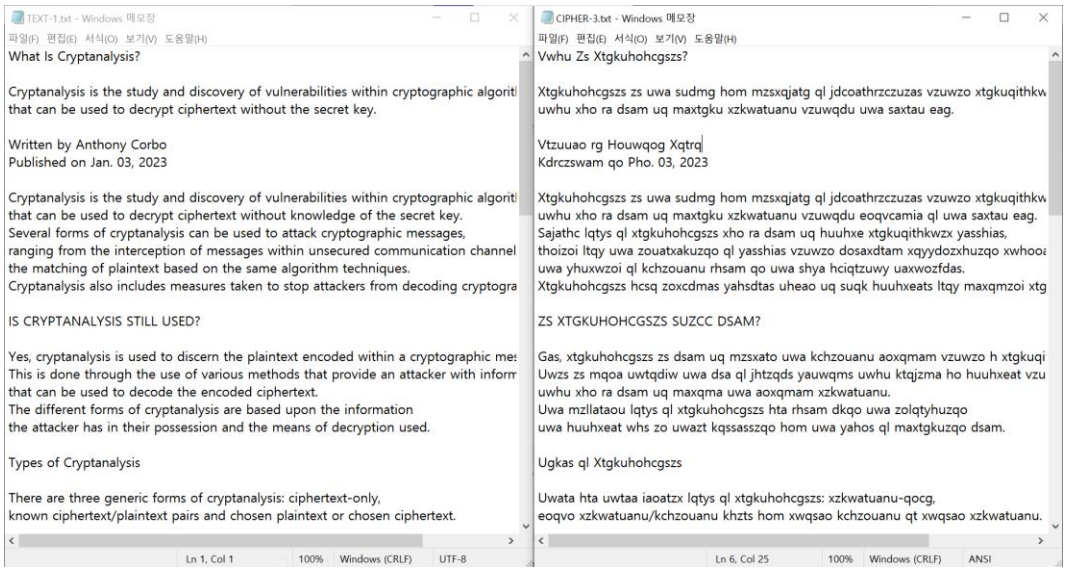
```
22  #2-(c)
23  in_file = "TEXT-1.txt"
24
25  InFileObj=open(in_file, 'rt', encoding='UTF8')
26  PT = InFileObj.read()
27  InFileObj.close()
28
29  key = randomkey(UpAlphabet) #랜덤하게 섞인 값을 키로 설정
30
31  if findright(key) == True: #유효한 키 값이라면, 암호화
32      CT = SubstLib.subst_encrypt(key, PT)
33      out_file = 'CIPHER-3.txt'
34      OutFileObj = open(out_file, 'w')
35      OutFileObj.write(CT)
36      OutFileObj.close()
37  else: #유효하지 않은 키값이라면, 오류메시지 출력
38      print('InValued key')
39  #=============================================
```

TEXT-1.txt의 데이터를 PT에 저장

알파벳을 랜덤하게 섞은 문자열을 key로 설정

유효한 키라면, 암호화하여 암호문을 CIPHER-3.txt에 저장

유효하지 않은 키라면, 오류 메시지만 출력



정상적으로 CIPHER-3.txt에 암호문이 저장됨

(d)

1. 빈도 사전 만들기

```python
#2-(d)
def getLetterCount(message):
    letterCount = {'A':0, 'B':0, 'C':0, 'D':0, 'E':0, 'F':0, 'G':0, 'H':0,
                   'I':0, 'J':0, 'K':0, 'L':0, 'M':0, 'N':0, 'O':0, 'P':0,
                   'Q':0, 'R':0, 'S':0, 'T':0, 'U':0, 'V':0, 'W':0, 'X':0,
                   'Y':0, 'Z':0}
    for char in message.upper():
        if char in UpAlphabet:
            letterCount[char] += 1
    return letterCount
```

문자열에서 각 알파벳의 출현 횟수를 계산하는 함수 getLetterCount 생성

2. 빈도 찾는 함수 만들기

```python
def findfreq(message):
    letter2freq = getLetterCount(message) #메시지의 빈도 사전 만들기

    #사전 반대로 바꾸기
    freq2letter = {}
    for char in UpAlphabet:
        if letter2freq[char] not in freq2letter:
            freq2letter[letter2freq[char]] = [char]
        else:
            freq2letter[letter2freq[char]].append(char)

    #동일한 출현빈도인 문자들을 정렬
    for freq in freq2letter:
        freq2letter[freq].sort(key=ETAOIN.find, reverse=False)
        #작은 값을 앞쪽에 정렬
        freq2letter[freq] = ''.join(freq2letter[freq])


    freqPairs = list(freq2letter.items()) #사전을 리스트로 전환
    freqPairs.sort(key=getItemAtIndexZero, reverse=True)
    #배열의 첫번째 값을 기준으로 정렬

    #빈도순서를 문자열로 바꾸기
    freqOrder = []
    for freq_pair in freqPairs:
        freqOrder.append(freq_pair[1])
        freq_order_str = ''.join(freqOrder)

    return freq_order_str #빈도순서를 반환
```

메시지 빈도사전을 만들고 키와 값을 서로 바꿔줌

키에서 작은 값을 앞쪽에 정렬한 후, 배열의 첫번째 값을 기준으로 정렬함

정렬된 사전을 문자열로 바꿔줌

3. 평문과 암호문 영문자 빈도 조사

```
88    print('Key=', key)
89
90    CTfreq=findfreq(CT)
91    print('CT frequency=', CTfreq)
92
93    PTfreq=findfreq(PT)
94    print('PT frequency=', PTfreq)
```

키, 암호문, 평문의 빈도 조사

```
Key= HRXMALIWZPECYOQKFTSUDJVNGB
CT frequency= AUHZOSQTXWCKMGIYDLVNERJFBP
PT frequency= ETAINSORCHLPDYGMUFWXKBVQZJ
```

(c)에서의 암호문과 평문을 비교하면, key값과 맞다는 것을 알 수 있음

일반적인 영문자의 빈도순서: ETAOINSHRDLCUMWFGYPBVKJXQZ

->평문의 빈도순서와 흡사함