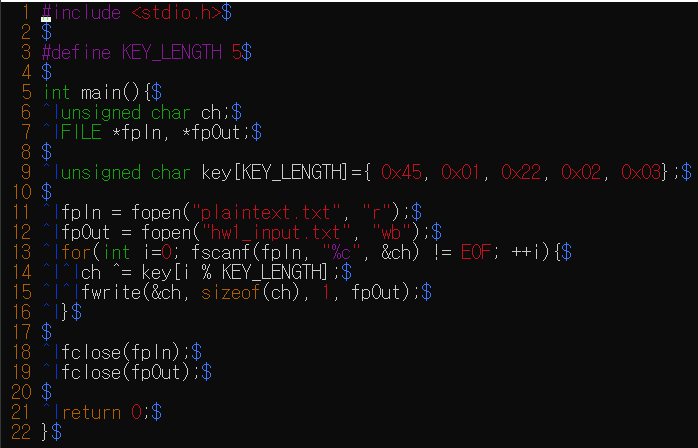
정보보호개론 Assignment 1

2016310936 우승민

이번 과제는 vigenere variant cipher를 사용하여 암호화 시킨 plain text를 해독하는 tool을 만드는 과제입니다.

우선 암호화 코드는 아래와 같습니다.



KEY\_LENGTH 와 그만큼의 key 값이 주어지면 입력 받은 plaintext를 key와 xor시켜 hw1\_input으로 암호화합니다.

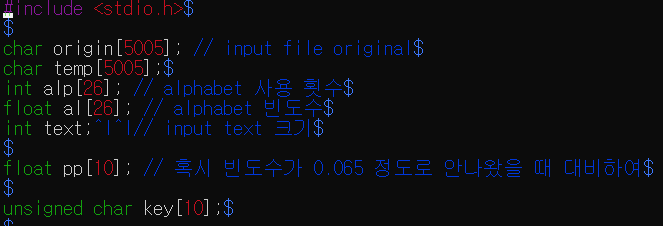
이 코드의 암호해독 코드를 만들기 위해서는 크게 2가지 일을 수행해야합니다.

1. KEY\_LENGTH 값 찾기

2. 사용된 KEY 찾기

위 2가지 일을 수행하면 찾은 KEY\_LENGTH 와 KEY를 통해 암호화된 text를 해독할 수 있습니다.

우선 제 코드의 main 문과 전역변수부터 설명하겠습니다.



각 변수에 대해 설명하면 origin은 암호화된 text를 저장하는 용도로 사용되고, temp는 나중에 key를 찾으면서 변형된 text를 저장하기 위해 사용됩니다.

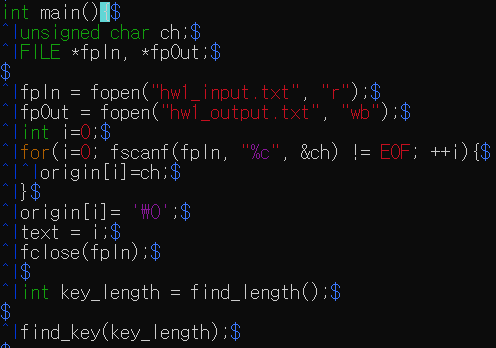
alp와 al array는 암호화된 text에서 alphabet 빈도수를 측정하기 위해 사용됩니다.

text 변수는 입력 받은 text의 크기를 저장하는데 사용됩니다.

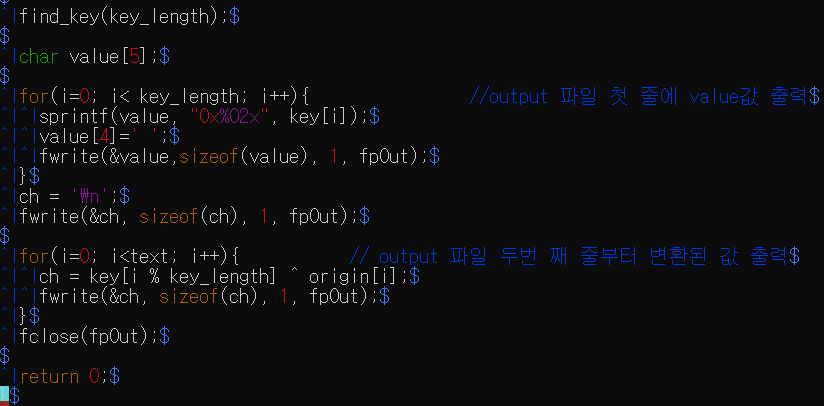
pp array는 KEY\_LENGTH에 따라 합산된 빈도수를 저장하기 위해 사용됩니다.

key array는 찾게 된 KEY\_VALUE 값을 저장하기 위해 사용됩니다.

다음으로 main에 대해 설명하겠습니다.

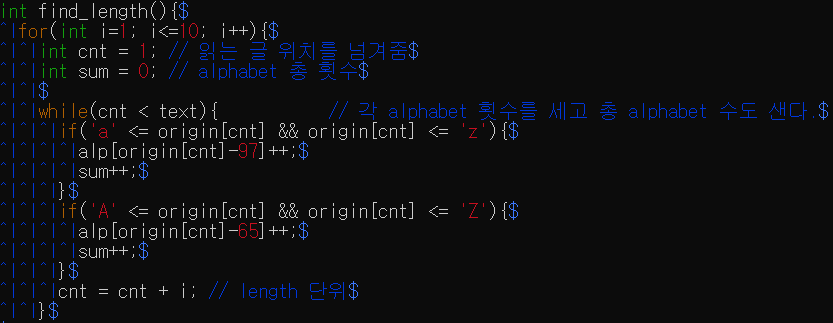


우선 입력 받은 text를 origin에 저장하고 key\_length에 find\_length() 함수를 통해 KEY\_LENGTH를 찾은 후 저장하게 하였고, 이어서 find\_key(int ) 함수를 통해 KEY\_VALUE 값들을 찾게 해주었습니다.

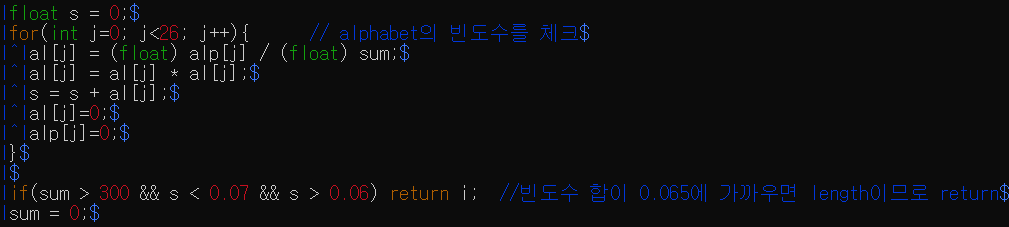


KEY\_VALUE 값을 전부 찾으면, output 파일 상단에 순서대로 출력한 이후 다음 줄부터 해독한 text를 출력하게 만들어주었습니다.

다음으로 KEY\_LENGTH 값을 찾는 함수인 find\_length() 대해 설명하겠습니다.

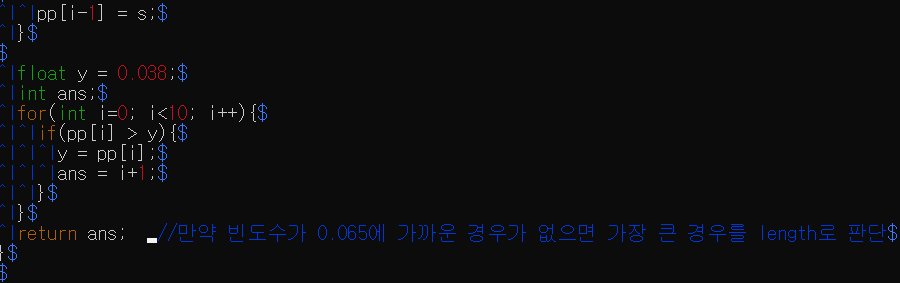


우선 KEY\_LENGTH가 1부터 10까지 가능하기 때문에 for문도 1부터 10까지 반복하게 해주었고, origin에 저장한 text를 예상 KEY\_LENGTH ( i ) 번째 글자마다의 alphabet 사용 횟수를 저장해주었습니다. sum은 총 alphabet 사용 횟수입니다.



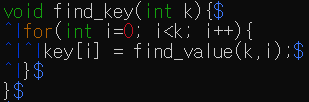
이후 al array에 (alphabet 마다의 사용 횟수) / (총 alphabet 사용 횟수) = 빈도 를 저장해 주었고, 제곱하였습니다. s에 빈도 제곱 값을 총합을 저장하고, 만약 0.065에 가까운 값이 나오면 빠른 해독을 위해 바로 return하여 함수를 끝내도록 하였습니다.

만약 sum이 300보다 작다면 표본이 너무 적다고 판단되기에 return하지 않도록 하였습니다.



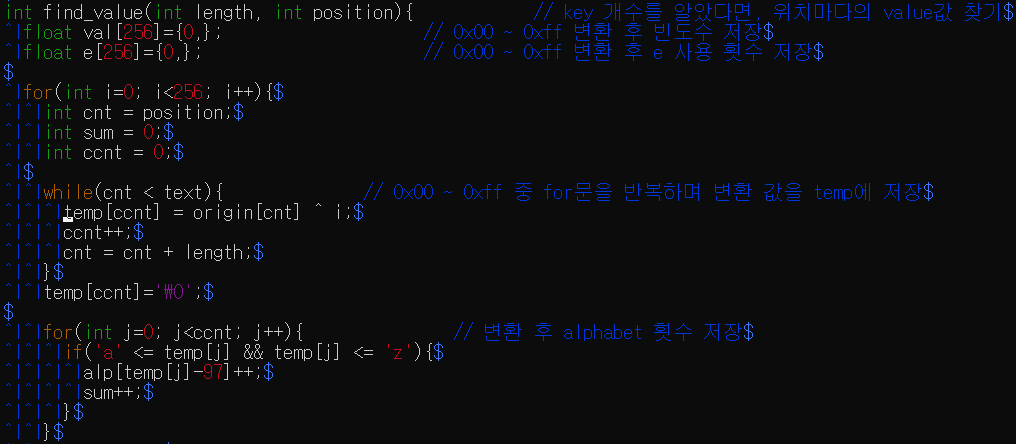
만약 0.065에 가까운 값이 없으면 pp array에 빈도 수 합을 저장하고 for문이 끝나면 1 부터 10까지의 빈도 수 중 가장 크게 나온 값을 KEY\_LENGTH로 정하게 하였습니다.

다음으로 find\_key( int )은 아래와 같습니다.



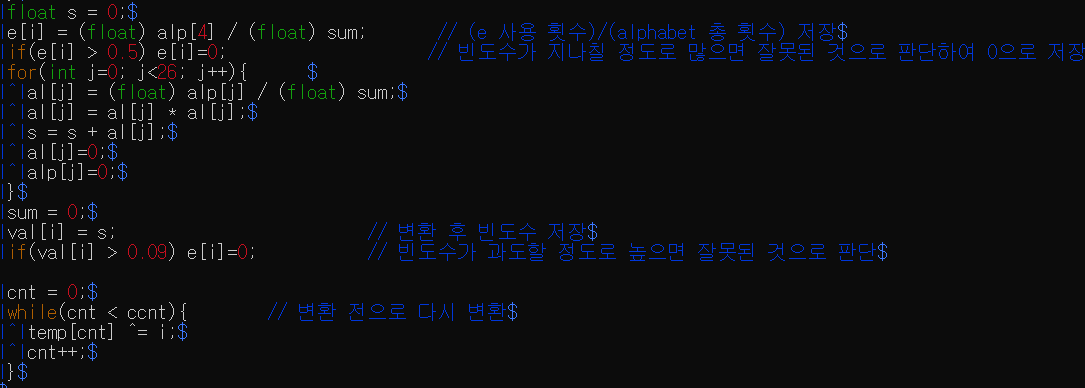
KEY\_LENGTH만큼 for문을 돌며 각 위치에 맞는 value를 찾기 위해 find\_value ( int, int ) 함수를 사용하게 됩니다. 처음 넘겨주는 인자는 length의 길이이고, 두번째 인자는 몇 번째의 value 인지를 넘겨줍니다.

다음으로 KEY\_VALUE 값을 찾는 함수인 find\_value( int )에 대해 설명하겠습니다.



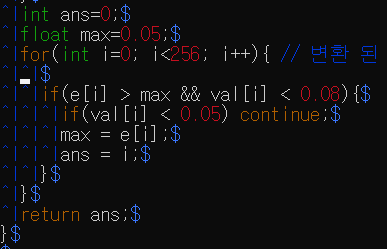
우선 초반 부분은 find\_length() 함수와 비슷하게 진행됩니다. 다른 점은 temp에 position 번째의 글자를 시작으로 length마다의 글자를 i(0x00~0xff)와 xor한 값을 저장하게 하였습니다. 그 후 find\_length()와 동일하게 alphabet 사용횟수와 총 alphabet 사용횟수를 저장하게 해주었습니다.

새로운 array인 val은 0x00~0xff까지의 값에 따른 빈도수 합을 저장하기 위해 사용하였고, e는 alphabet ‘e’의 사용 횟수를 저장하기 위해 사용되었습니다.



그 후 e array에 alphabet ‘e’의 사용 빈도를 저장하였습니다. 만약 0.5보다 클 경우는 과도하게 나왔기 때문에 표본이 부족하여 생긴 결과일 수도 있으므로 0으로 바꾸었습니다. 이후엔 (alphabet 사용 횟수) / ( 총 alphabet 사용 횟수) = 빈도 를 구한 후 빈도의 총합을 제곱하여 저장해주었습니다. 여기서는 val array에 저장하였습니다. 만약 빈도 총합이 0.09 초과하면 이것 또한 표본의 문제로 발생한 것이라 판단해 e array에 저장한 값을 0으로 하였습니다.

그 후 다시 temp에 변형 전 상태를 저장하였습니다.

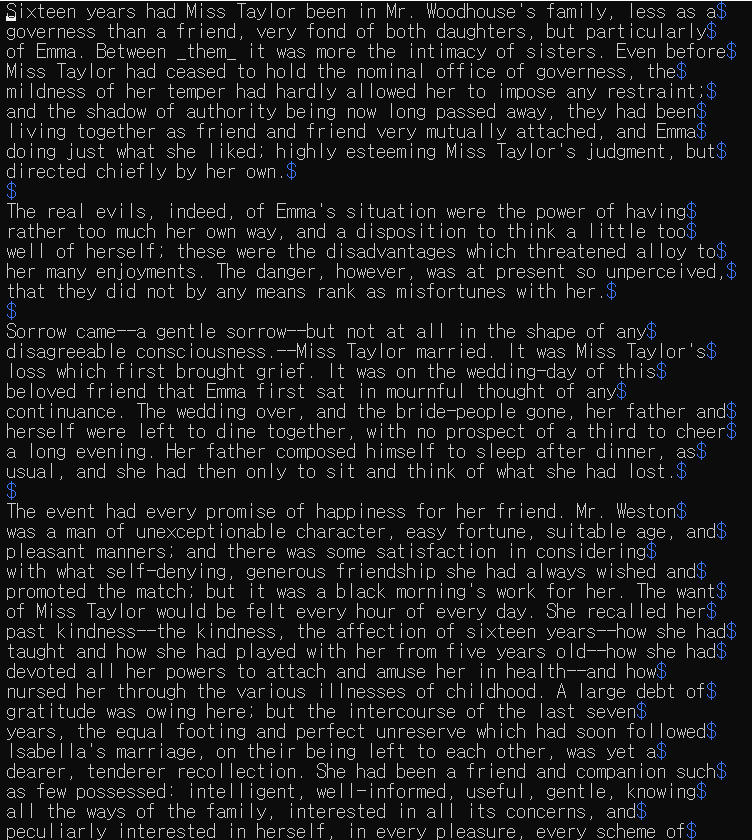


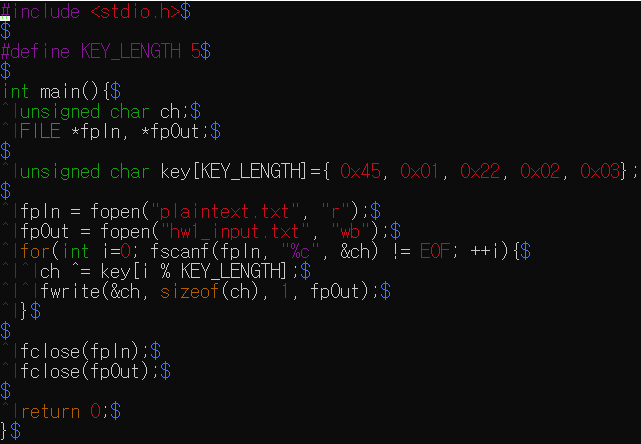
마지막으로 for문 (0x00 ~ 0xff) 까지 빈도수 제곱 합 ‘e’ 사용 빈도 측정이 끝나면, 전체의 값을 다시 비교하여 KEY\_VALUE를 찾는 과정입니다.

제가 선택한 value는 ‘e’의 사용횟수가 가장 크고, 빈도수 제곱 합이 0.05~0.08 안에 들어오면 적절하다고 생각하였습니다.

제 code에 대해 평가하자면 처음 입력되는 plaintext의 크기가 1000 bytes에서 5000 bytes 사이의 값으로 제한되어 있기에 사용된 text에 따라 표본이 부족하여 정확도가 부족하다고 생각합니다.

실행화면을 끝으로 설명 마치겠습니다.





암호화 code

암호화된 text 입니다.





실행 화면입니다.

해독한 text입니다.

