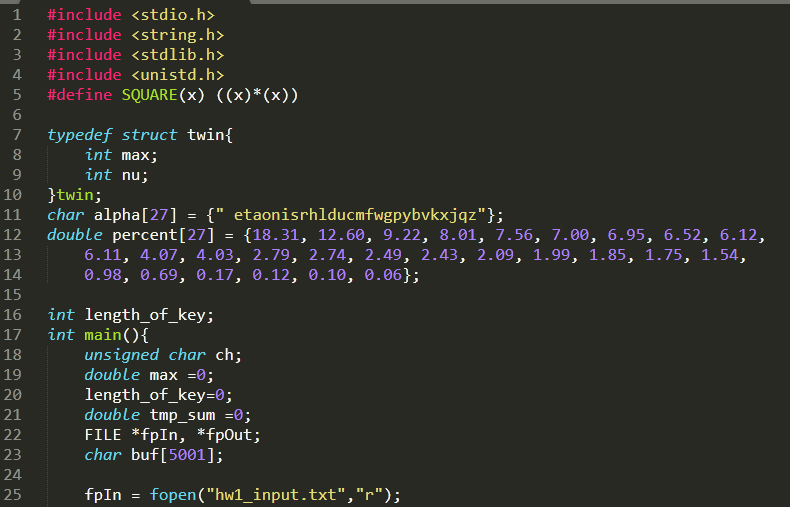
Assignment 1

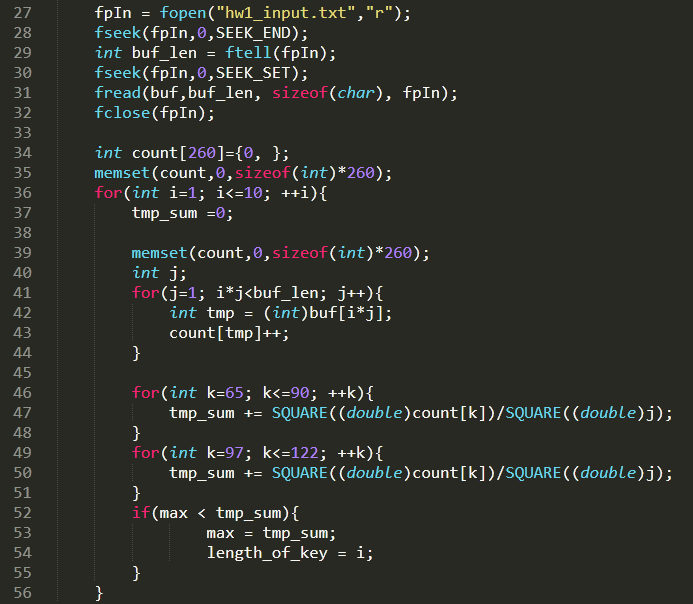
2016310932 배현웅

1. Overview
   1. Vigenere varint cipher Decryption is divided into two tasks. First task is to know the length of key. Second task is to distinguish each key value.
   2. To identify the length of key, I apply the statistics which alphabet distribution is not kind of uniform distribution. As the lecture note said, I use sigma of probabilities square. ( )
   3. After we find the key length, we also check distribution of Nth letter. This time, I multiply frequency of alphabet included ‘space’ and the probability of each alphabet which I find distribution of Nth letter.
2. Code description



This code consists of initiation of value, such as length\_of\_key, percent and alpha which stands for the frequency of alphabet. Alpha[7] is corresponds to percent[7]. (‘e’ : 12.60% frequency)

Char buf value is for input text whose name is hw1\_input.txt



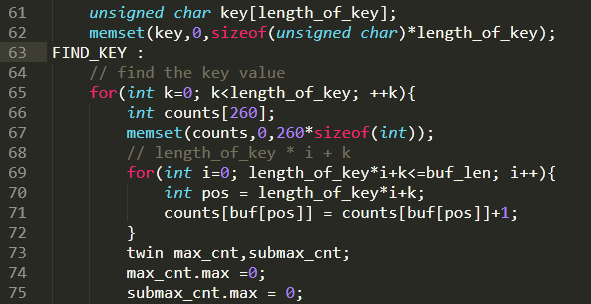
In line 27-32, I check file length by fseek and ftell incase of bumping into EOF value(in ASCII code 0x1A)

In line 36, to check possible length which is 1 to 10, I write for loop.

Line 41-44 : traverse all buf element(input context), then save frequency of each asci code into count array.

Line 46-51 : square of probability of each alphabet (lower case : 65(a)~90(z), capital : 97(A)~122(Z) is added into tmp\_sum

Line 52-54 : to compare value of each case (depends on key length) I take sigma of probabilities square. ( ) value and find the highest value among them. At that value, the highest possible key length is it.

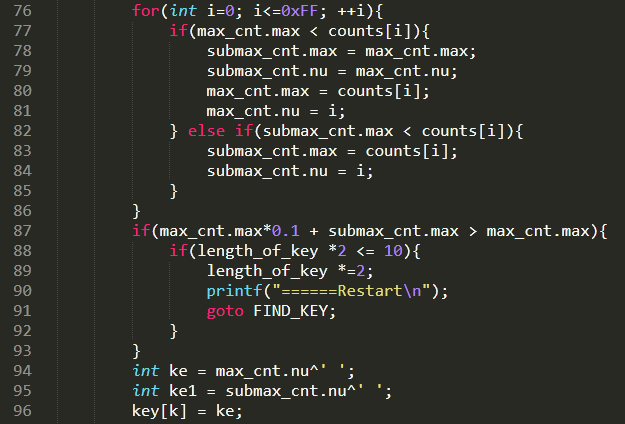


After we find the key length, as I comment on code, we should find the key value

Line 63 is label for correction of key length. I explain it later.

Line 65-72 : I take Nth letter in input text, and do similar thing as I find key length. However, this time I assume the repeated letter is ‘space’ which statistically occur often most

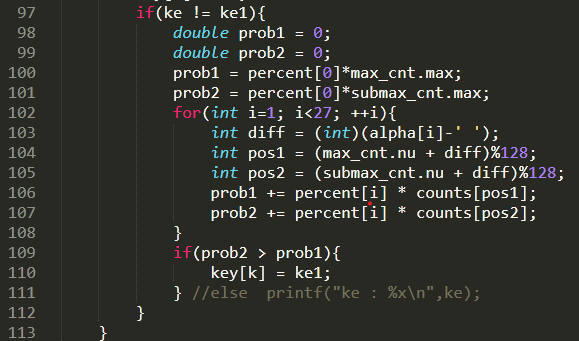




Line 76 : we find most frequent letter(=max) and 2nd frequent letter(=submax) which are candidate of ‘space’

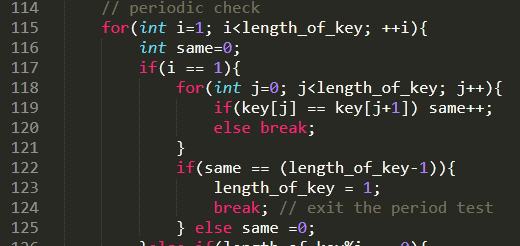
Line 87 – 93 : there is little difference b/w submax and max. There is unfolded key stream. So multiply 2, then do FIND\_KEY work again (ex key : 0x01 0x02 0x01 0x03)

Line 94-95 save to ke and ke1 value which are candidate of key



If two key candidate is different, I compare statistic values which is supposed to be ke or ke1is space

Compare sigma of alphabet probabilities square. ( ) which include ‘space’

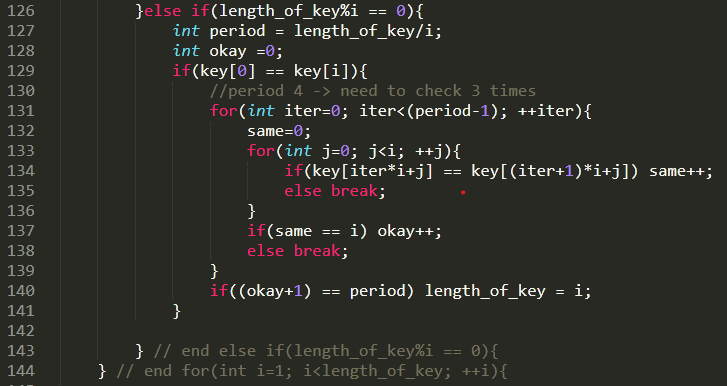


If there is repeated periodic value, we reduced to optimized way

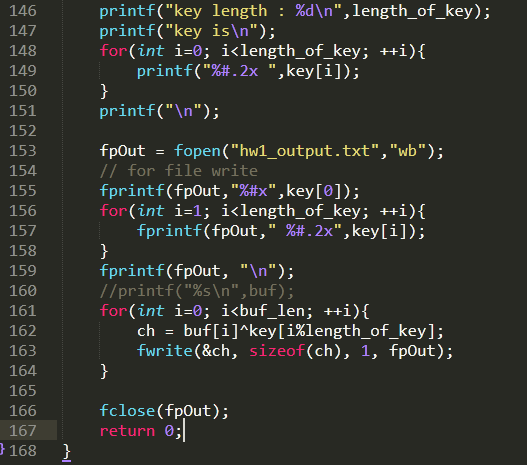
(ex key : 0x01 0x02 0x01 0x02 -> 0x01 0x02 ))

So, I Check all possible value which is lower than length\_of\_key

Line 117-125 : Test whether the key length is one



If key length is not one, then test possible key length which is aliquot(약수) of key length

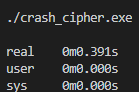


After find the key value and key length, write down to output value.

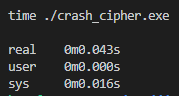
1. Performance Check

I Test time command two text which is encrypted by key length 10 for performance check

* Text which consists of 1485 letters



* Text which consists of 4775 letters



Worst case for my code runtime is about 0.391 seconds