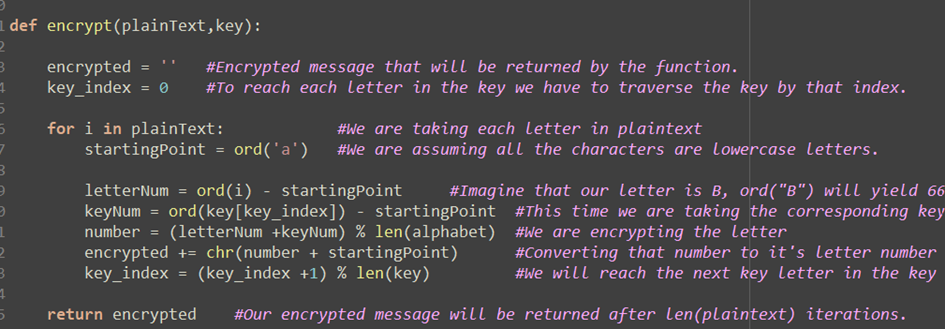
**CENG 471 - HW1**

İlayda Özel 260201037 - Göktay İncekara 250201012

**Introduction**

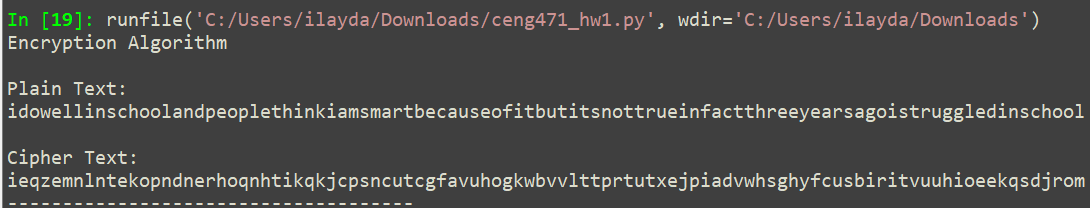
In this report we will try to explain encryption and decryption algorithms and the algorithm to find the key, given ciphertext and plaintext. Furthermore, with the help of the break algorithm we will analyze the relationship between the key length and execution time.

**Encryption Algorithm**

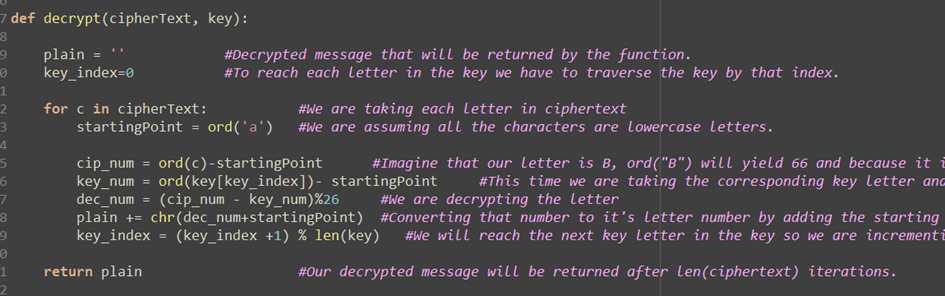


In Vigenere Shift encryption algorithm, we have a plaintext and a key. The first element of the plaintext is shifted with the first element of the key. And the second is with the second element of the key. Each time the next element of the plaintext and the key is being used. If the key comes to the end, we start from the first element of the key again. This process is repeated until the end of the plaintext.

We find the ASCII char number of the both plaintext letters and key letters, we discard them from the ASCII number of “a”, so we can reach to the index numbers of the alphabet such as “a”=0, “b”=1”…”z”=26. We add the key’s ASCII number and the plaintext’s ASCII number. If the result is greater than 26, which is the number of letters in the alphabet, we take the mod 26 of the result, add with the ASCII number of “a” and convert this number to its ASCII char representative.

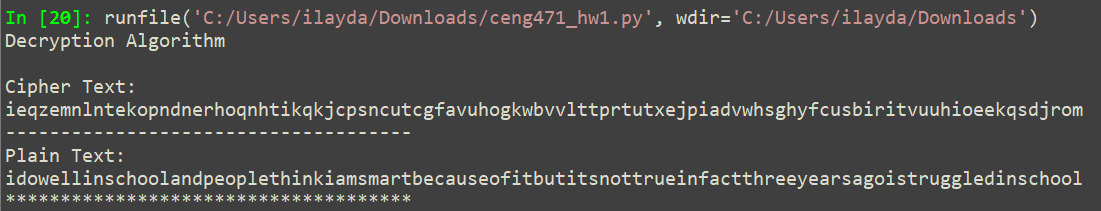


**Decryption Algorithm**

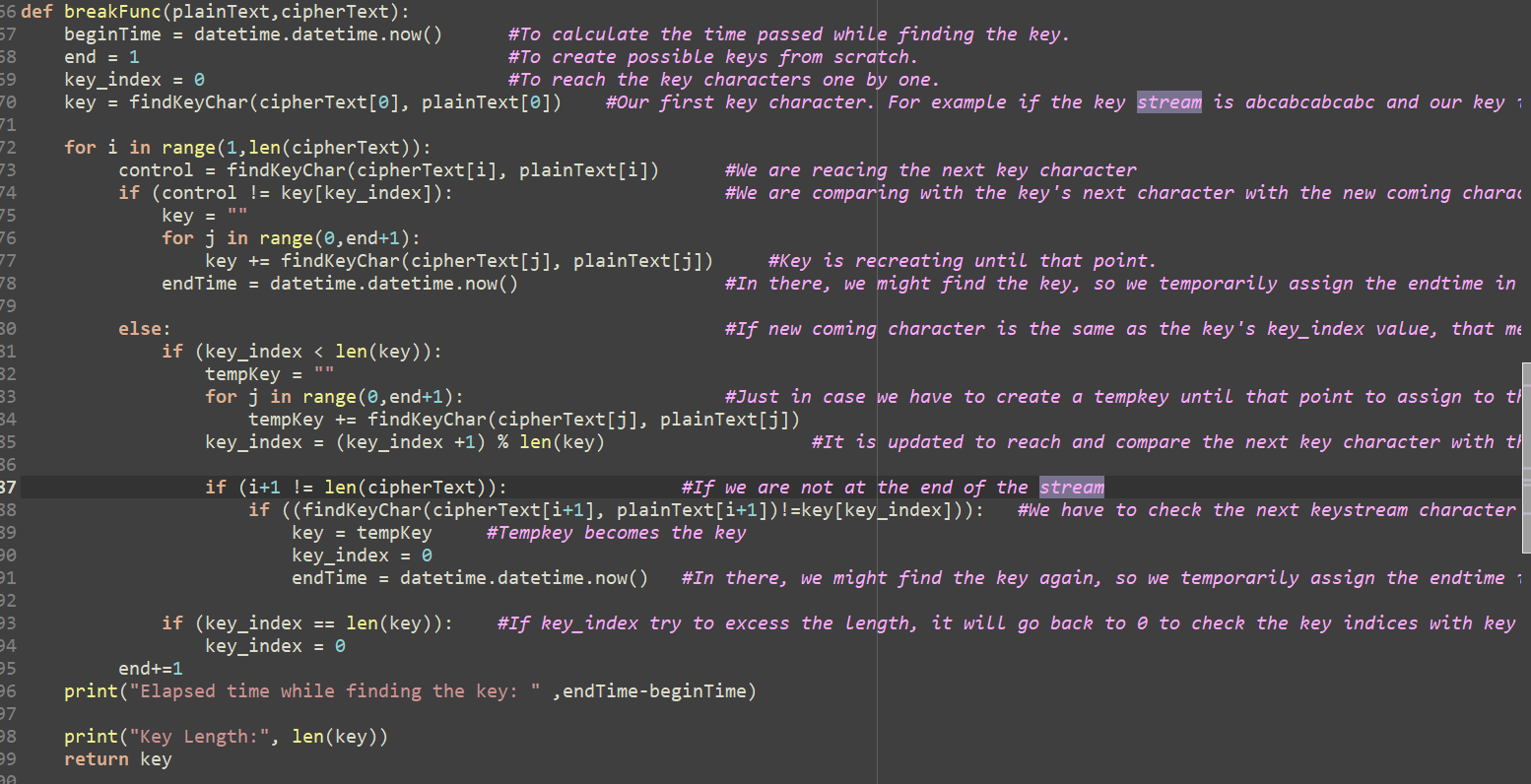


In Vigenere Shift decryption algorithm, we have a ciphertext and a key. The first element of the ciphertext is shifted with the first element of the key. And the second is with the second element of the key. Each time the next elemet of the plaintext and the key is being used. If the key comes to the end, we start from the first element of the key again. This process is repeated until the end of the plaintext.

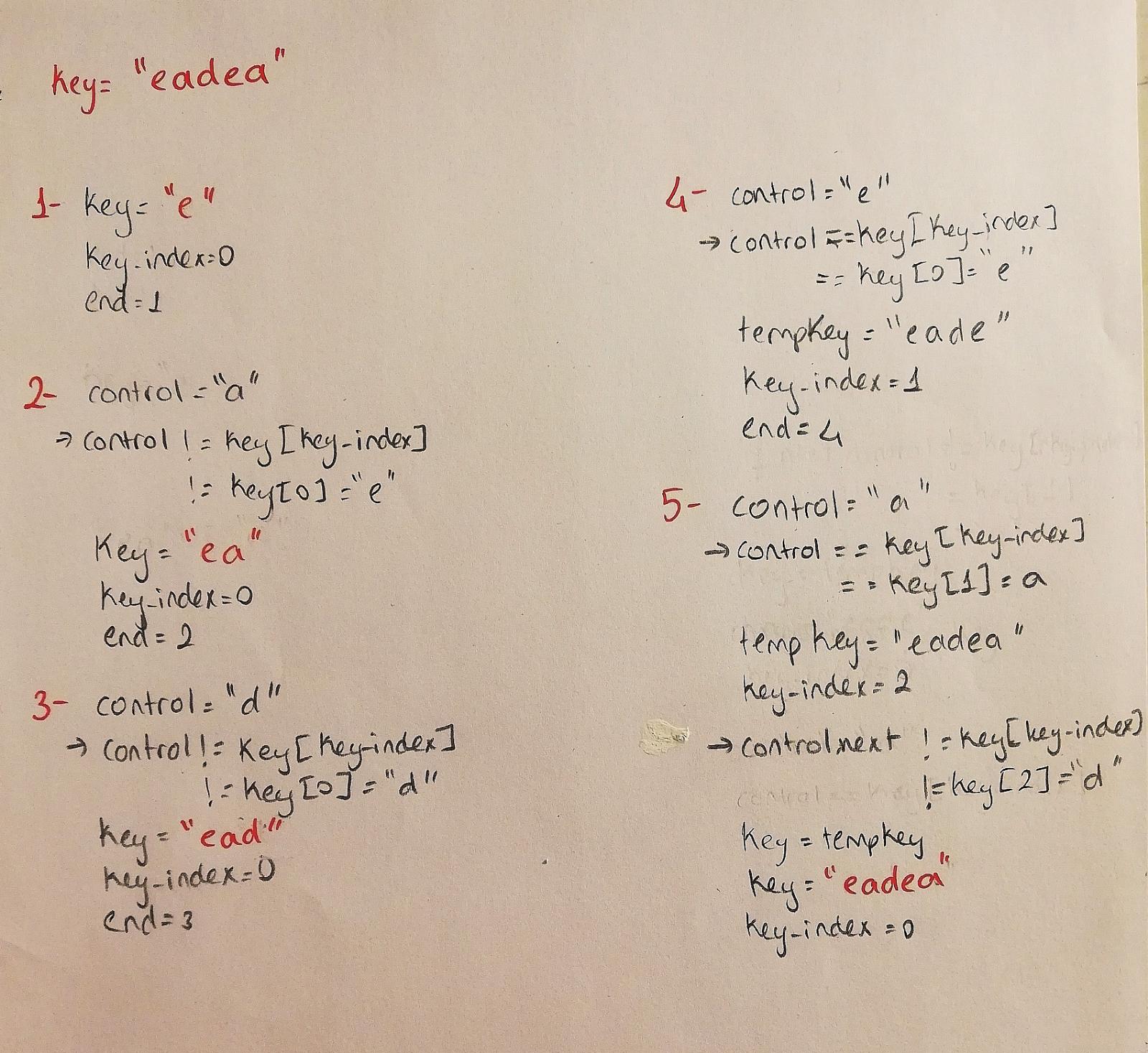
We find the ASCII char number of the both plaintext letters and key letters, we discard them from the ASCII number of “a”, so we can reach to the index numbers of the alphabet such as “a”=0, “b”=1”…”z”=26. We add the key’s ASCII number and the plaintext’s ASCII number. If the result is greater than 26, which is the number of letters in the alphabet, we take the mod 26 of the result, add with the ASCII number of “a” and convert this number to its ASCII char representative.

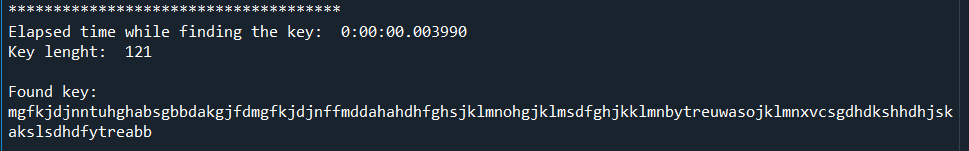


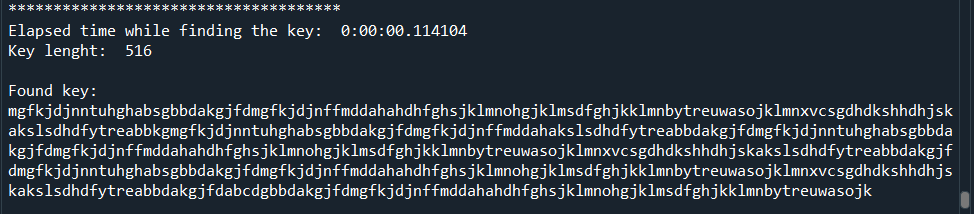
**Finding Key Function**

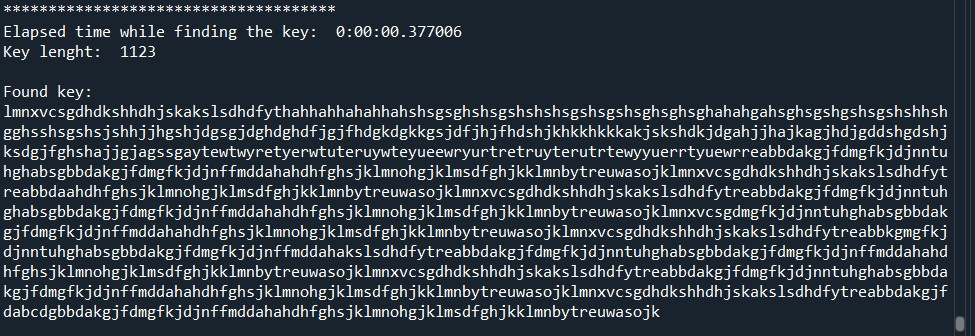


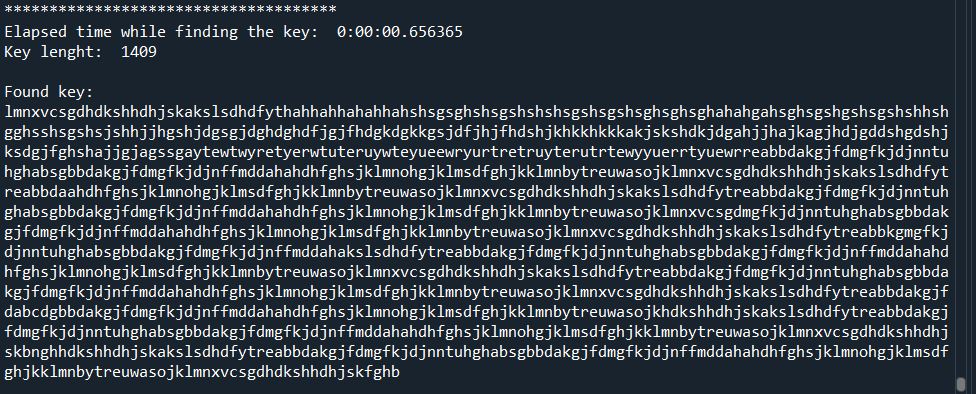
A step by step trace for creating the key “eadea“

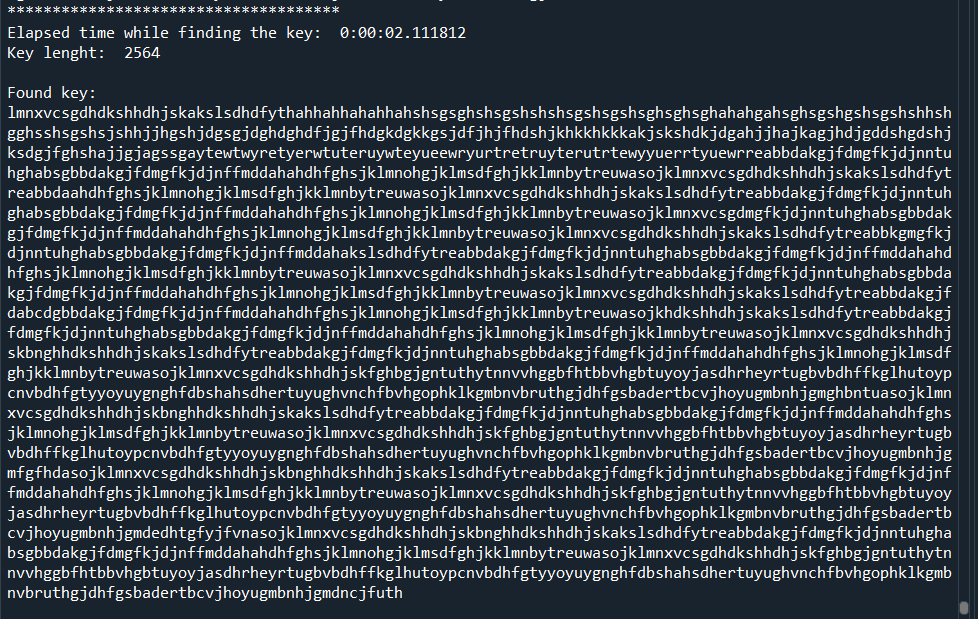


In the first try to break the key, We have a 121 character long key. To find the key, the time needed is 0.003990 second.

In the second try to break the key, We have a 516 character long key. To find the key, the time needed is 0.114104 second. Notice that when we increase the key length 4 times, the time increased nearly 29 times.

In the third try to break the key, We have a 1123 character long key. To find the key, the time needed is 0.377006 second. Notice that when we increase the key length 2,17 more times, the time increased 3,3 times.

In the fourth try to break the key, We have a 1409 character long key. To find the key, the time needed is 0.656365 second. Notice that when we increase the key length 1.25 more times, the time increased 1.74 times.

Finally, to break the key, We have a 2564 character long key. To find the key, the time needed is 2.111812 seconds. Notice that when we increase the key length 1,81 more times, the time increased more than 2 seconds. If we compare it with the first example, when we increase the key length 20,677 times, the time increased 529,27 times.

To sum up, when we have a longer key, it takes much time to break it.