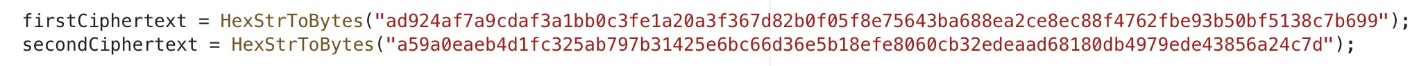
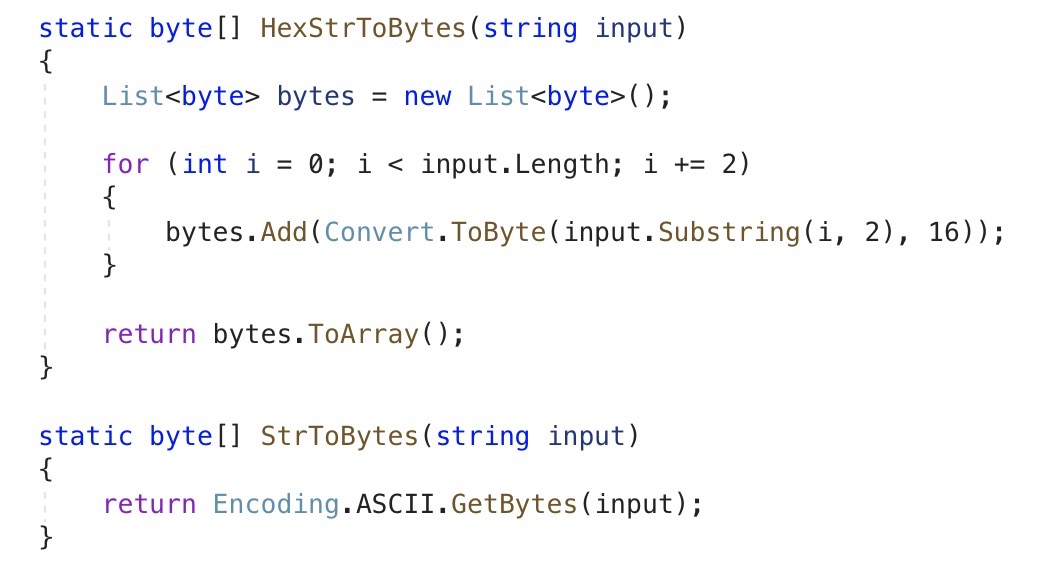
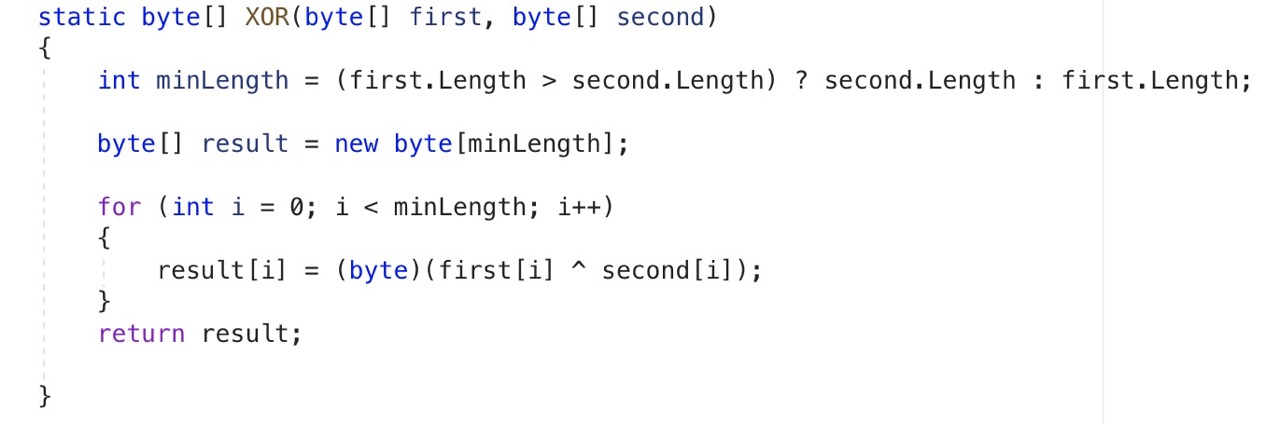
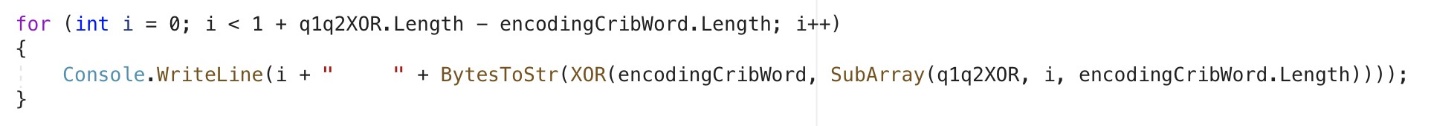
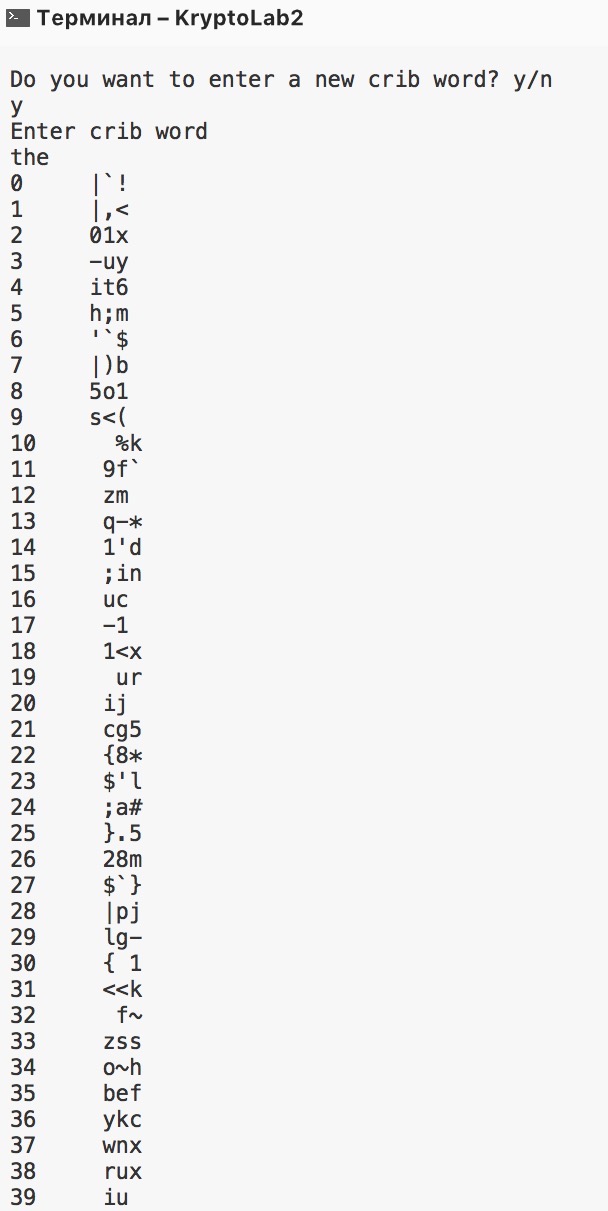
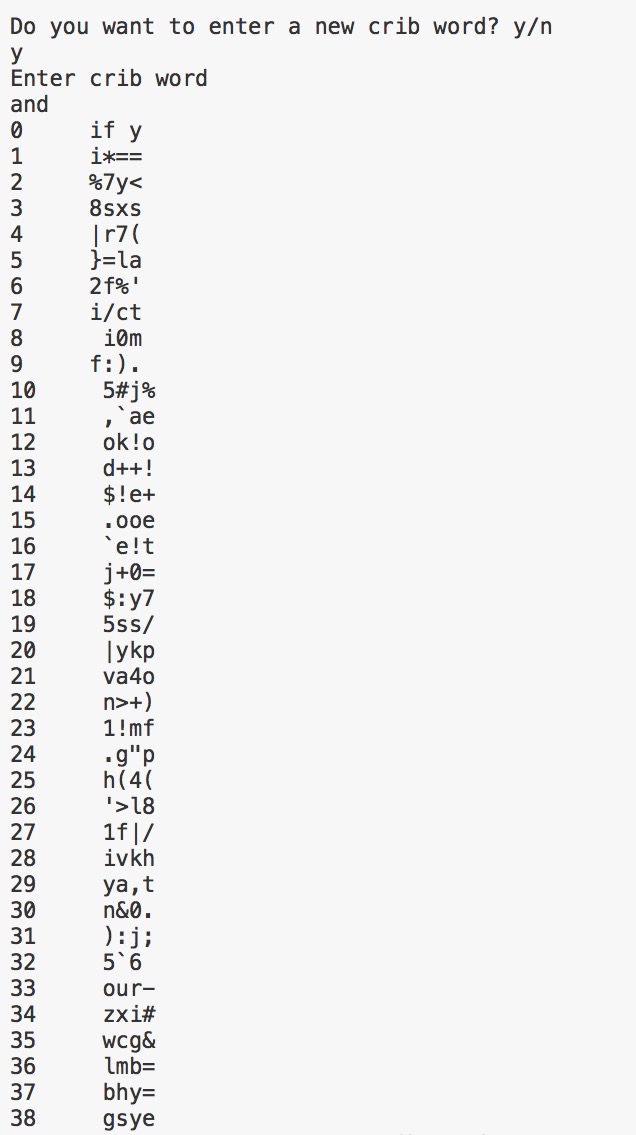
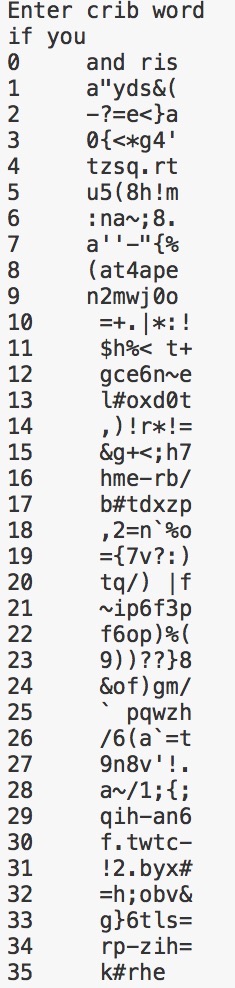
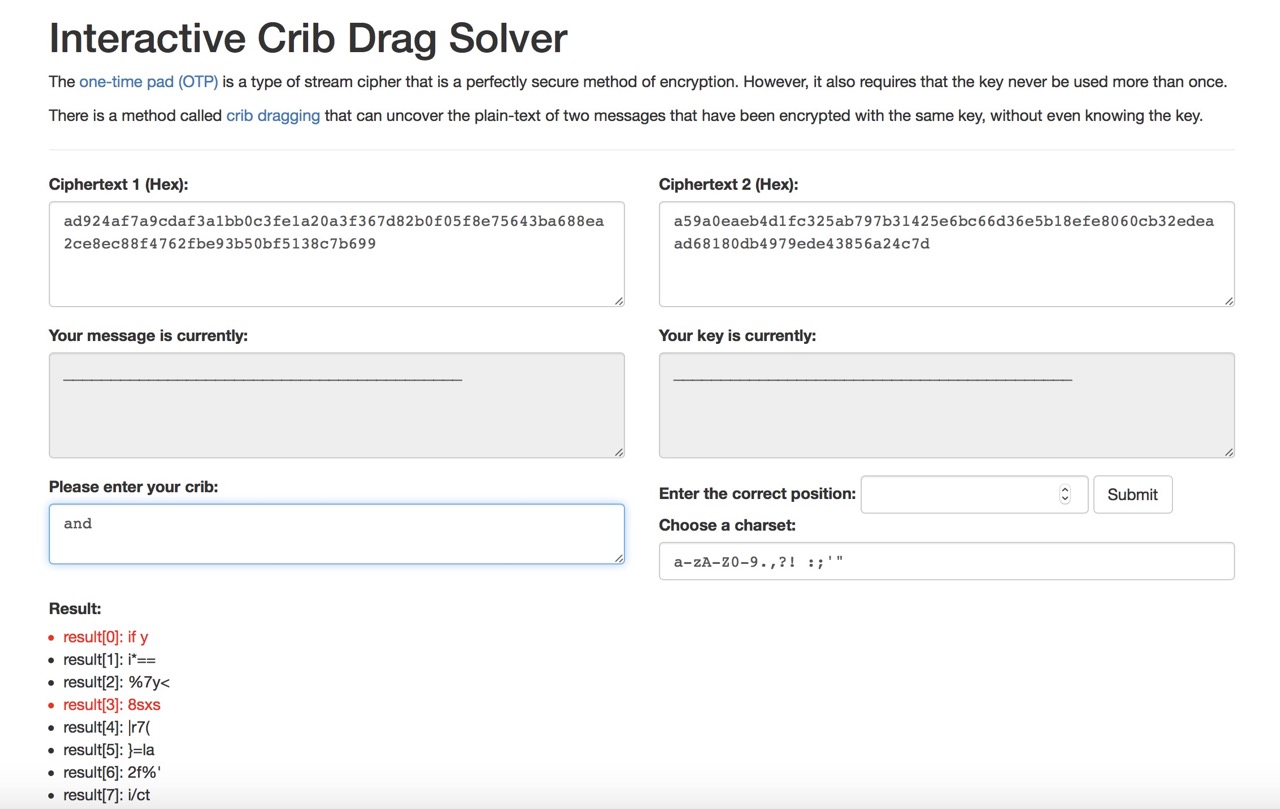
# Report

**Theoretical part:**  
The one time pad (OTP) is a type of stream cipher that is a secure method of encryption. It’s simple to implement and is secure as long as the length of the key is greater than or equal to the length of the message. That’s its major downfall. However, it also requires that the key never be used more than once. This laboratory work shows what happens when you re-use a key to encrypt more than one message.  
We will use a method called crib dragging to uncover the plain-text of two messages that have been encrypted with the same key, without even knowing the key.  
  
Say we send messages A and B of the same length, both encrypted using same key, K. The stream cipher produces a string of bits C(K) the same length as the messages. The encrypted versions of the messages then are:  
  
E(A) = A xor C  
E(B) = B xor C  
where xor is performed bit by bit.  
Say an adversary has intercepted E(A) and E(B). He can easily compute:  
E(A) xor E(B)  
However, xor is commutative and has the property that X xor X = 0 (self-inverse) so:  
E(A) xor E(B) = (A xor C) xor (B xor C) = A xor B xor C xor C = A xor B  
E(A) xor E(B)  
However, xor is commutative and has the property that X xor X = 0 (self-inverse) so:  
  
E(A) xor E(B) = (A xor C) xor (B xor C) = A xor B xor C xor C = A xor B  
  
That means, if A and B were encrypted using the same key, we can decrypt A xor B  
  
**Algorithm by example:**  
Input :  
Using first two quotes from <https://docs.google.com/document/d/19vgZtvDN4_StEgVEM9MjfxnqfayByLNMD7PFJgvZv7c/edit>  
[](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/ciphertext.jpg?raw=true)

1. Guess a word that might appear in one of the messages  
   For this point we might check <https://en.wikipedia.org/wiki/Most_common_words_in_English>  
   First time we'll be use the most popular word "the"
2. Encode the word from step 1 to a hex string  
   Using the following method: [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/strToBytes.jpg?raw=true) [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/strToBytesFunc.jpg?raw=true)
3. XOR the two cipher-text messages  
   If your cipher-texts have different length, we'll take the length of the shortest one, so the "tail" of the longer will be cut. However, we can aslo decrypt this "tail" using the same method.  
   [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/q1q2XOR.jpg?raw=true)  
   [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/XORfunc.jpg?raw=true)
4. XOR the hex string from step 2 at each position of the XOR of the two cipher-texts (from step 3)  
   [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/q1q2XORcribword.jpg?raw=true)
5. When the result from step 4 is a readable text, we guess the English word and expand our crib search.  
   -> If the result is not readable text, we try an XOR of the crib word at the next position.  
   Now we can see the result :  
   [](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/resultOfWordThe.jpg?raw=true)

As we see, in some ocassions it returns combination of readable letters, but the whole word is still hard to recognize. So we should try another crib, for example "and " - as we consider, phrases must contain spaces, so we can try enter words with them  
  
The result :  
[](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/resultOfWordAnd.jpg?raw=true)  
So we've got readable text "if y" on the [0] position of the text (offset equals zero)  
We then repeat the process, guessing what “if y” might be when expanded and then XOR that result with the XOR of the cipher-texts. Let's check crib “if you ” - convert it to a hex string, and XOR it with the XOR of the cipher-texts, we’ll get “and ris”.  
[](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/resultOfWordIfyou.jpg?raw=true)  
So using this method we can find all other words for our ciphertexts  
  
There are also existing calculators that can be used for this work, like  
<https://lzutao.github.io/cribdrag/>  
example of using this program - we've got the same result  
[](https://github.com/YaJProgrammist/KryptoLab2/blob/main/Screenshots/existedSolverResult2.jpg?raw=true)