Cryptographic Attacks

CR432E - Cybersecurity with Python

Students must submit a code illustrating the solution of both parts. The code in question should be well commented. Students must also submit a live video to demonstrate the answers given in their codes. (Code and comments: 70 points, Explanation via demo: 30 points)

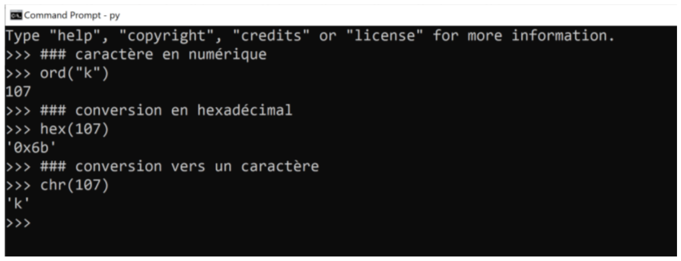
Part 1

Assuming you are a crypto analyst, you have intercepted the following encrypted message:

Message = "gpkyfe zj r kilcp nfeuviwlc crexlrxv"

You suspect that the entity sending the message sent a Caesar code. Your goal is to find the associated message in plain text (decrypted). Please consider these assumptions:

1. As a starting assumption, we will just have to use lowercase ASCII characters [a-z]. Other characters are unchanged during encryption.
2. Each character is encoded on one byte. As a reminder, a byte consists of 8 bits and a certain representation is used for the characters.
3. In this assignment, we will use hexadecimal notation: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F, the idea is to have 16 digits on 4 bits (24=16).
4. The following Python code illustrates character representations in hexadecimal (e.g., "k") is shown in Figure 1.



```
Command Prompt - py
Type "help", "copyright", "credits" or "license" for more information.
>>> ### caractère en numérique
>>> ord("k")
107
>>> ### conversion en hexadécimal
>>> hex(107)
'0x6b'
>>> ### conversion vers un caractère
>>> chr(107)
'k'
>>>
```

Figure 1 Conversions

5. To understand the encryption of the Caesar code, it involves using a number as a key. Considering the following figure, the key is equal to "2", the letter "a" will be encrypted as "c", the letter "b" will be encrypted as "d", etc. (Please refer to Figure 2)

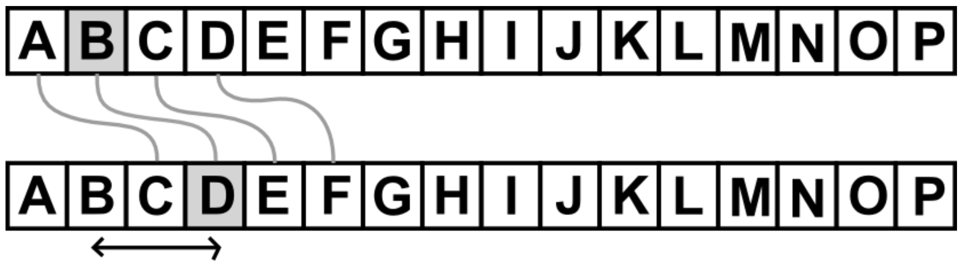


Figure 2 Code Caesar Example

Question 1 (5 points)

Please write an "encrypt" function which takes as input a (plain) text composed of a sequence of ASCII characters [a-z] and a key and which returns the encrypted text via the Caesar code.

Question 2 (5 points)

Please write a "decrypt" function that takes ciphertext and a key as inputs and returns the plaintext via Caesar code.

Encryption is the process of converting plaintext into ciphertext using a cipher and a key.
Unlike the key, the choice of the cipher is disclosed.

Question 3 (5 points)

Decryption is the reverse process of encryption, converting ciphertext back into plaintext using a cipher and a
Although the cipher would be public knowledge, recovering the plaintext without knowledge of the key should be impossible (infeasible).

What is the relationship between encryption and decryption?

Question 4 (5 points)

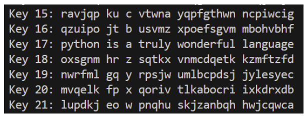
Using the "argparse" module, please create a setting for your script such that it can take a character string as input or a file containing text and return a message or an encrypted file as output.

Question 5 (5 points)

Please add to your script a "brute_force" function, which will take an encrypted message as input and return a dictionary having the encryption key values as keys and the plaintext values as values (e.g., {k0: plain_text0, k1: plain_text1, …, kn: plain_textn})

Question 6 (5 points)

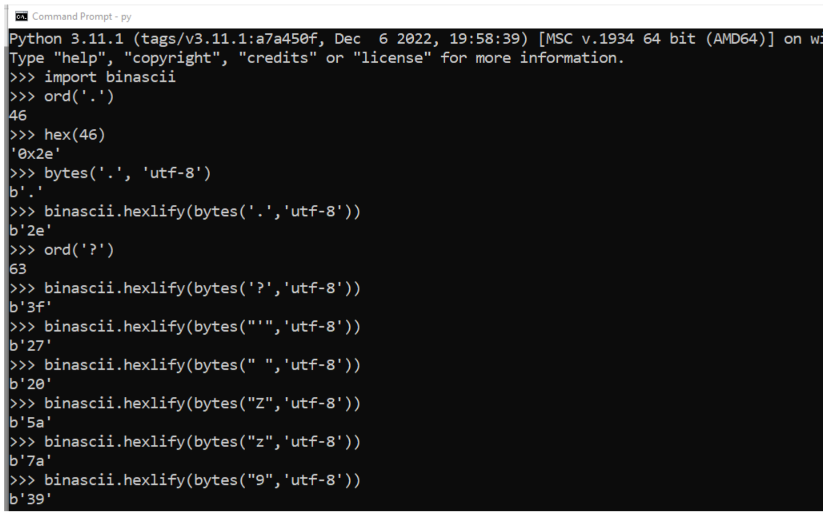
By testing the "brute_force" function with the intercepted message, please deduce the key value as well as the plaintext message.



```
Key 15: ravjqp ku c vtwna yqpfgthwn ncpiwcig
Key 16: qzuipo jt b usvmz xpoefsgvm mbohvbhf
Key 17: python is a truly wonderful language
Key 18: oxsgnm hr z sqtkx vnmcdqetk kzmftzfd
Key 19: nwrfml gq y rpsjw umlbcpdsj jylesyec
Key 20: mvqelk fp x qoriv tlkabocri ixkdrxdb
Key 21: lupdkj eo w pnqhu skjzanbqh hwjcqwca
```

Part 2

As an analyst, you intercept an encrypted file, or you suspect that the text is encrypted with substitution values on one byte. As a reminder, please refer to code snippets shown in Figure 3.



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on wi
Type "help", "copyright", "credits" or "license" for more information.
>>> import binascii
>>> ord('.')
46
>>> hex(46)
'0x2e'
>>> bytes('.', 'utf-8')
b'.'
>>> binascii.hexlify(bytes('.','utf-8'))
b'2e'
>>> ord('?')
63
>>> binascii.hexlify(bytes('?','utf-8'))
b'3f'
>>> binascii.hexlify(bytes("'",'utf-8'))
b'27'
>>> binascii.hexlify(bytes(" ",'utf-8'))
b'20'
>>> binascii.hexlify(bytes("Z",'utf-8'))
b'5a'
>>> binascii.hexlify(bytes("z",'utf-8'))
b'7a'
>>> binascii.hexlify(bytes("9",'utf-8'))
b'39'
```

Figure 3 Hex forms

Knowing that the letters used in this problem are the letters of the alphabet in upper and lower case, numbers as well as various punctuations, single space and “\n”. All symbols are encoded in “utf-8”. Substitution encryption consists of using a key in the form of a dictionary, where each symbol (e.g., letters) corresponds to a substitution value of one byte (8 bits). Here is an example, for “a” which is encrypted in “,”: {‘a’: b‘2c’, …}. Values are drawn at random for each letter and are meant to encrypt a single symbol. It is impossible to have the same key for two or more different symbols.

Even if a brute force attack is an option, we will prefer to carry out a cryptographic analysis. Therefore, we will calculate the frequency of characters in a plain text - representative in our case of the English language - as well as in an encrypted text. Assuming that the frequency distribution is similar in the two texts, we can match the encrypted symbol with the highest frequency with the plaintext symbol which has the highest frequency.

Question 7 (5 points)

Please write a function "frq_txt" which takes as input parameter a text (represented as a string) (see the text in the file "english_sample.txt") and will return a list of dictionaries or tuples (your choice), or each symbol is associated with the associated number of occurrences. The list must be ordered from the most frequent character to the least frequent characters.

Question 8 (5 points)

Please write a function "frq_cpr" which takes as input parameter a sequence of bytes (see the file "cipher_sample.bin") and will return a list of dictionaries or tuples (your choice), where each symbol is associated with the number of associated occurrences. The list must be ordered from the most frequent character to the least frequent characters.

Question 9 (5 points)

When matching the two ordered lists, please compare the plain text to the encrypted text. (No Python code, it would need an explanation)

one from a byte perspective and the other from a character perspective on a frequency analysis

Question 10 (5 points)

Since encryption is done via a key in dictionary form (e.g., key={'a': b'2d', …}), decryption is done with an inverted key (e.g., key={b'2d' : 'a', …}), based on the observation made in the previous question, please write a function “build_dec_key” which will cover the partial decryption key.

Question 11 (5 points)

Please implement a "guess_txt" function having as inputs the ciphertext as well as a partial decryption key of the most frequent characters made based on the frequencies of the base English text. If a character cannot be deciphered, you can match with “$”. The length of the partial key with which the test will be carried out is 17.

Question 12 (5 points, follow-up of question 11)

Please test the “guess_txt” function on the “cipher_sample.bin” file.

Question 13 (5 points)

Please deduce words or phrases from the frequency analysis of the “english_sample.txt” file, this will allow you to recover the decryption key.

Question 14 (5 points)

Please repeat the test using the decryption key on the “test.bin” file and deduce as many words as possible in plain text rom.