



EXERCISES — Binary Cipher

version #7be580532266ed398481e31366afcc24b1950c2a



**The way is lit. The path is clear.
We require only the strength to follow it.**

Copyright

This document is for internal use at EPITA ([website](#)) only.

Copyright © 2022-2023 Assistants <assistants@tickets.assistants.epita.fr>

The use of this document must abide by the following rules:

- ▷ You downloaded it from the assistants' intranet.*
- ▷ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

1	Goal	3
2	my_xor_crypt	3
3	my_rol_crypt	4

*<https://intra.assistants.epita.fr>

File Tree

```
binary_cipher/  
├── rol_crypt.c  (to submit)  
└── xor_crypt.c  (to submit)
```

Authorized headers : You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- stddef.h

Compilation : Your code must compile with the following flags

- -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function : None

1 Goal

In this exercise, you are going to manipulate directly the bits underlying your data. For this, you will implement two functions: one to cipher a text with a xor, the other with a rotation. They both act on data indicated by the data pointer, with length `data_len`. To cipher, they use a key pointed by the `key` pointer, of length `key_len`. When ciphering, if the data is longer than the key, start again the ciphering from the beginning of the key.

For every functions, you must modify the data in-place.

2 my_xor_crypt

The prototype is the following:

```
/*  
** XOR byte by byte the data of size data_len with the key of size  
** key_len.  
*/  
void my_xor_crypt(void *data, size_t data_len, const void *key, size_t key_len);
```

The key is repeated as many times as necessary to have the same length as the data. Each byte of the data is combined with the corresponding byte of the key.

For example:

```
char text[] = "Example";  
my_xor_crypt(text, 7, "KEY", 3)
```

```
message :  E    x    a    m    p    l    e
key :      K    E    Y    K    E    Y    K
result :  E^K  x^E  a^Y  m^K  p^E  l^Y  e^K
```

3 my_rot_crypt

The prototype is the following:

```
/*
** Rotate byte by byte the data of size data_len with the key of size
** key_len.
** Rotating a byte by another means adding them together with potential
** overflow.
*/
void my_rot_crypt(void *data, size_t data_len, const void *key, size_t key_len);
```

For example:

```
message :  E    x    a    m    p    l    e
key :      K    E    Y    K    E    Y    K
result :  E+K  x+E  a+Y  m+K  p+E  l+Y  e+K
```

The key is repeated as many times as necessary to have the same length as the data. Each byte of the data is combined with the corresponding byte of the key.

As the security of the data is not the objective here, you can leave the data and the key written in clear in the code.

You should work on the data using `unsigned char`.

The way is lit. The path is clear. We require only the strength to follow it.