Exercises — Binary cipher

version #

Assistants C/Unix 2022 <assistants@tickets.assistants.epita.fr>

Copyright

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

Contents

1 Binary cipher **3**

1.1 Goal . . . 3

1.2 my_xor_crypt . . . 3

1.3 my_rol_crypt . . . 4

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

1 Binary cipher

Files to submit:

- binary_cipher/xor-crypt.c
- binary_cipher/rol-crypt.c

Main function: none

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

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

1.1 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.

1.2 my_xor_crypt

The prototype is the following:

```c
/*
** 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:

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

```
messgae :   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
```

1.3 my_rol_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_rol_crypt(void *data, size_t data_len, const void *key, size_t key_len);
```

For example:

```
messgae :   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`.

It is my job to make sure you do yours.