Notice

During the lab session, we made an announcement for you to use Linux environment for project 1.

But, we found out some of you are using macOS environment.

There are some minor differences between macOS gcc and Linux gcc.

We will not take this into consideration while we are grading.

If you are still willing to use macOS environment, you will face the problem in endian conversions.

You can use 3rd party libraries for endian conversions.

Plus, you are allowed to refer to open-source codes for checksum calculation, cypher calculation and endian conversion (regardless of which OS environment you are using.

But, please make sure to cite the source in your comment.

Notice

When your input text is larger than 10MB (more exactly, 10MB - header size,) you have to split the text chunk into multiple messages.
Between two adjacent messages, continuity problem would happen. (In 'cake' example, the encryption of previous message ends with a key character 'a', and the encryption of following message starts with 'c')

In that case, each chunk of text should be encrypted as it is encrypted by one message transaction (e.g., a hypothetical 20MB message.)

We recommend you to make your client count the number of alphabetic characters in the text chunk, so it can properly request the following messages. (In the 'cake' example, the key of second message would be 'keca'.)

We understand that it is unintuitive for remote encryption client to inspect data before sending it to the server. We are sorry for students who are confused by this unintuitive solution. Thanks for Youngjae Min who asked the first question about this in QnA board.

FAQ

1. How do I submit?

2. You said that you are going to grade by:

```
./client -h 143.248.56.16 -p 4000 -o 0 -k asdf < test.txt > a.txt

./client -h 143.248.56.16 -p 4000 -o 1 -k asdf < a.txt > b.txt

diff test.txt b.txt
```

But then, is it guaranteed that test.txt does not contain any uppercase letters?

Thank you very much.

ANS

1. Upload your zip file to klms

2. We will grade with -i option (ignore case option).

FAQ

I have a question regarding implementation of server.

It seems like your server re-initalizes the keyword shift for each packet, but it also seems reasonable to preserve the keyword shift while the connection between the server and the client is maintained.

For example, suppose that our keyword is "abcd", and our message is "aaaaa₩naaaaa". For some reason, the client may want to split the message into two parts, send them separately, and merge together. Now suppose that the client split the message with line break(₩n). If we implement the server in the former way (way your server was implemented), the client will receive "abcda₩nabcda". But what the client actually wants is "abcda₩nbcdab".

So what policy should we follow?

ANS

It depends on implementation, but simply remembering last used character at the client side and changing "keyword" at second part of message.

Server side should be same as ours.