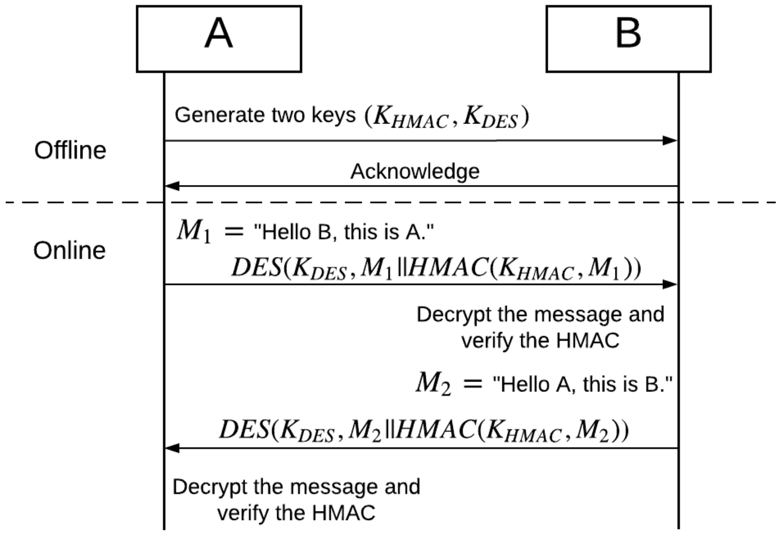
Lab 3: Implementation and Application of HMAC

HMAC is a keyed-hash type of message authentication code (MAC), involving a hash function and a secret key. It can simultaneously provide the data integrity and the authentication of a message. According to the different underlying hash functions MD5, SHA-1, SHA-256, etc., the algorithm is termed HMAC-MD5, HMAC-SHA1, HMAC-SHA256, etc.

Task

It is an **individual** work using socket programming. Client ***C*** and Server ***S*** share a key for HMAC in an offline manner (e.g., a local file). Client ***C*** then generates a message, gets the HMAC digest of this message, and encrypts the message along with its HMAC to obtain ciphertext. Then ***C*** sends this ciphertext to Server ***S***. Server ***S*** decrypts received ciphertext and then verifies the integrity of the received message by generating another HMAC with the shared HMAC key and matches the two HMACs. Client ***C*** and Server ***S*** switch the roles and do the above again. All the transmitted messages should be encrypted with DES.

A simple protocol for the above process may be like:


A
B
Offline
Online
Generate two keys (K_{HMAC}, K_{DES})
Acknowledge
M_1 = "Hello B, this is A."
$DES(K_{DES}, M_1 \| HMAC(K_{HMAC}, M_1))$
Decrypt the message and verify the HMAC
M_2 = "Hello A, this is B."
$DES(K_{DES}, M_2 \| HMAC(K_{HMAC}, M_2))$
Decrypt the message and verify the HMAC

Here A is client, B is server.

- M_1 is the message A wants to send to B.
- $HMAC(K_{HMAC}, M_1)$ is the result after applying HMAC algorithm to message M_1 using key K_{HMAC}.
- $M_1 \| HMAC(K_{HMAC}, M_1)$ is the **concatenation** of message M_1 and $HMAC(K_{HMAC}, M_1)$.
- $DES(K_{DES}, M_1 \| HMAC(K_{HMAC}, M_1))$ is the result of applying DES algorithm to the above concatenated string using key K_{DES}. It is also the ciphertext that sent to B.
- Once B receives $DES(K_{DES}, M_1 \| HMAC(K_{HMAC}, M_1))$, decrypt it using K_{DES} to get $M_1 \| HMAC(K_{HMAC}, M_1)$. Split it to obtain M_1 and $HMAC(K_{HMAC}, M_1)$.
- Use obtained M_1 and shared K_{HMAC} to generate a new HMAC $HMAC(K_{HMAC}, M_1)$.
- Compare $HMAC(K_{HMAC}, M_1)$ and $HMAC(K_{HMAC}, M_1)$ to verify the integrity of M_1.
- Then B starts to send another message to A just like what A did.

Steps

1. ***S*** and ***C*** share two keys. One is for HMAC, the other is for DES. Dump them to files.
2. ***S*** and ***C*** set up a simple chat program. (Socket)
3. ***S*** and ***C*** both load HMAC key and DES key from the files on startup of their programs.
4. ***S*** and ***C*** exchange messages. The messages should be entered by you through keyboard in the console, **NOT** by hardcoding.

Both sides should display:

1. Shared keys for HMAC and DES
2. Plain message and HMAC before concatenation
3. Ciphertext to be sent
4. Received ciphertext

5. Plain message and HMAC after decryption and separation
6. Receiver calculated HMAC, which uses K_{HMAC} and received plain message+
7. Verification result (by comparing two HMACs)

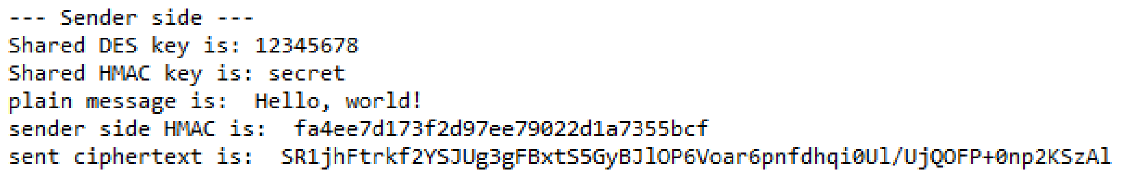
Submission

Submit the following files in a **zip** onto Canvas.

1. All of your source code files for this lab. Code comments will be helpful for the TA to understand your code.
2. A README file including information:
   a) Which language and external libraries you use.
   b) Which IDE you use.
   c) Detailed steps about how to run your code to finally obtain the required outputs. TA will check and run your code based on this instruction. If your code cannot be successfully executed, points will be deducted. **This is the most important part used to evaluate your work, so please be as specific as you can.**
3. A lab report including screenshots of each step of the entire code execution procedure, **as clear and specific as you can**. Alternatively, you can choose to use screen recording to record the entire code execution procedure and submit the recorded file.

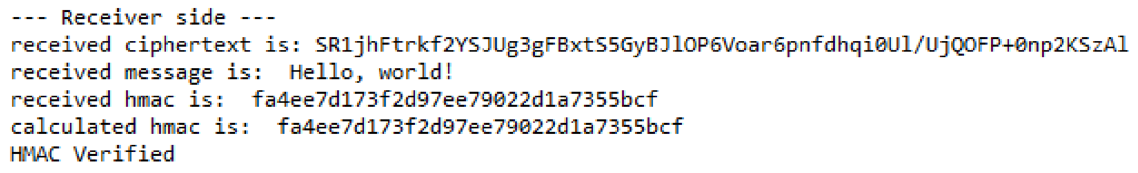
Screenshots example for a single communication

A sender sends "Hello, world!" to a receiver:



```
--- Sender side ---
Shared DES key is: 12345678
Shared HMAC key is: secret
plain message is:  Hello, world!
sender side HMAC is:  fa4ee7d173f2d97ee79022d1a7355bcf
sent ciphertext is:  SR1jhFtrkf2YSJUg3gFBxtS5GyBJlOP6Voar6pnfdhqi0Ul/UjQOFP+0np2KSzAl
```

The receiver receives the ciphertext from the sender and finally verify the HMAC:



```
--- Receiver side ---
received ciphertext is: SR1jhFtrkf2YSJUg3gFBxtS5GyBJlOP6Voar6pnfdhqi0Ul/UjQOFP+0np2KSzAl
received message is:  Hello, world!
received hmac is:  fa4ee7d173f2d97ee79022d1a7355bcf
calculated hmac is:  fa4ee7d173f2d97ee79022d1a7355bcf
HMAC Verified
```

Note that this is only an example of a single communication. You need to show the bi-directional communication results in the report. Do not forget to verify HMACs.

Submission Due: 11am, February 23, 2021.