## Task 2: Password-Based key Derivation and AES-GCM

* *(a) Introduction and High-Level Description*
  + The purpose of this task is encrypt a message using a password and AES-GCM with an authentication tag.
  + With a correct password, the DISCO board can decrypt the message from encrypt data. If the password is incorrect, it will not decrypt the correct message.
  + The key of this task is get from the password. Using PBKDF2 with a hash function.
* *(b) Low Level Description*
  + The password which student input need to be processed so that it can be used as a key for encryption. The method to get a key in this task is to use PBKDF2 to derive a 256-bit key with a random salt. The salt will decide what output we get. Here student create a 128-bit random salt.
  + A SHA-256 hashing function is used here to get the random key. The hash process can be done multiple times. Students here hash it for 5 times.

correct = mbedtls\_pkcs5\_pbkdf2\_hmac(&ctx1,input,lengths,hash\_key,128,5,256,midkey);

* + The PBKDF2 funciton need to be initialized. The code is:

    extern const mbedtls\_md\_info\_t mbedtls\_sha256\_info;

    int resultCode = mbedtls\_md\_setup(&ctx1, &mbedtls\_sha256\_info, 1);

* + After the key is generated, it is set as a key which can be used by AES-GCM.

mbedtls\_gcm\_setkey(&ctx\_gcm, MBEDTLS\_CIPHER\_ID\_AES, midkey, 256)

* + The program will read the message from USART 1, putty from the computer, the encrypt of the message is:

mbedtls\_gcm\_crypt\_and\_tag(&ctx\_gcm, MBEDTLS\_GCM\_ENCRYPT, lengths, iv, 12, addition, 4,inputMessage, messageEc, 16, tagBuf)

* + Here **iv** should be a random data 12 bytes long data buffer. However, student set a fixed data buffer, which may cause some concern in consider of safety. **Addition** data buffer is for the increment of processing data. Students make the **tagBuf** to be empty, this will make the encrypt function work.
  + Then student will request the password again. Use the same procedure to generate the key. And then the program will decrypt the encrypted message use this key. If the password is correct, the message will be decrypted correct, otherwise it will not be decrypted.

mbedtls\_gcm\_auth\_decrypt(&ctx\_gcm,messageLen,iv,12,addition,4,tagBuf, 16,messageEc,backoutput);

* *(c) Results and Analysis:*
  + The program worked as expected
  + As talked to TA, the iv in this part should be random generated 12-byte data buffer. Student set a fixed data buffer, which may course some issue about safety.

Conclusion

In this lab, student learned how to use Advanced Encryption Standard to encrypt message and decrypt the message. This encrypt operation involves the probability, logic to make the message encrypted and cannot be read.

To make the message not be read by hackers, it is necessary to generate a key to encrypt the message, and the key cannot to be tried out. The first task use a random generated key. In the second task, hash function will make the password to become a key. This is a more safe option because the repeat hashing can prevent brute method to tryout the key.

The application of the AES can keep the message transfer between microprocessor safe. The data is more difficult to be stolen. The protection of data has been used in many data transfer procedure today.