TASKS:
1. Establish communication between master and slave ECU
2. Send data from Master to Slave
3. Generate a key and encrypt data using the key
4. Store key in secure memory location
5. Implement SecOC(adding 1bit of freshness value + 1 bit of authenticator)

<u>MASTER ECU</u>

```
Import hmac
import hashlib
import FreshnessManager
import cryptography
from cryptography.fernet import Fernet
key_data=Fernet.generate_key()
#print("Key:",key_data)

Master_id=12267

file=open('key_value','wb')
file.write(key_data)
file.close()

Data=" 'Temp':100 and 'Pressure':90 "

if __name__=='__main__':
        FreshnessValue=FreshnessManager.Freshnessval()
        print('Freshnessvalue:',FreshnessValue)

        FreshnessManager.FreshnessCounter()
        new_message=FreshnessValue+Data

        #print('Data with freshness value:', new_message)

encoded=Data.encode()
f=Fernet(key_data)
encrypted=f.encrypt(encoded)
print("Encrypted Data:",encrypted)

file=open('encrypted_data','wb')
file.write(encrypted)
file.close()

Securedata= bytes(Data,'utf-8') + key_data + bytes(FreshnessValue,'utf-8')
MAC = hmac.new(key_data ,Securedata, hashlib.sha1)

secure_data=str(MAC.digest())+" "+str(encrypted)+" "+str(FreshnessValue)
print("Secure data format=",secure_data)

import pyAesCrypt,os
buffersize=256*1024
password=input("Enter your password: ")
pyAesCrypt.encryptFile("key_value","key_value.aes",password,buffersize)
os.remove("key_value")
```

SLAVE ECU

```
import cryptography
from cryptography.fernet import Fernet
import pyAesCrypt,os

Master_ID=int(input("ECU_ID for access :"))
file=open('ECU_ID')
Master_id=file.read()
file.close
if Master_ID==int(Master_id):
  print("AUTHENTICATION CONFIRMED")
  file=open('encrypted_data','rb')
  encrypted=file.read()
  print('Data from MASTER:',encrypted)
  #key1=str(input("Enter key for decryption :"))
  password="1234"
  password1=input("Enter password for key access: ")
  buffersize=256*1024
  if password1==password:
    pyAesCrypt.decryptFile("key_value.aes","key_valueout",password,buffersize)
    print("File decrypted")

    file=open('key_valueout','rb')
    key_data=file.read()
    print("key is:",key_data)
  else:
    print("No access to key flie")
  key1=str(input("Enter key for decryption :"))
  file=open('key_valueout','rb')
  key=file.read()
  file.close
  f1=Fernet(key)
  if key1==str(key):
     file=open('encrypted_data','rb')
     encrypted=file.read()
     decrypted=f1.decrypt(encrypted)
     original_data=decrypted.decode()
     print ("Decrypted Data:",original_data)
  else:
     print("INVALID KEY")

else:
  print("UNAUTHORIZED ACCESS")
```