((گزارش پروژه دوم مبانی امنیت اطلاعات))

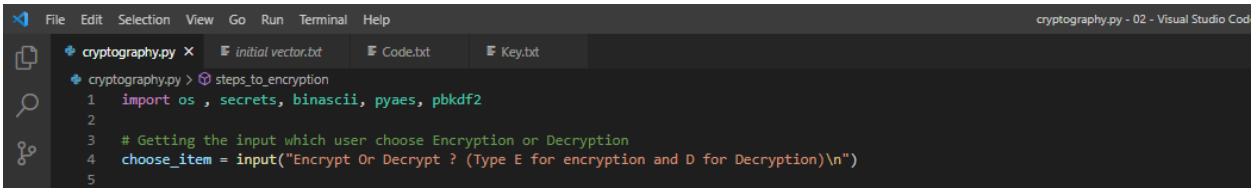
گردآورنده : عرفان ماجدی 9831099

مبانی امنیت اطلاعات

دکتر شهریاری

توضیحات کد پروژه)

در ابتدا باید کتابخانه های موردنیاز را **import** کنیم و سپس با توجه به دستور کار پروژه باید یک ورودی از کاربر بگیریم که اگر **E** وارد کرد همان **encryption** است یعنی میخواهد عملیات رمزنگاری را انجام دهد و اگر **D** وارد کرد همان **Decryption** است به این معنا که میخواهد عملیات رمزگشایی را انجام دهد. حال برای این دو عملیات دو تابع جداگانه تعریف شده است که براساس ورودی کاربر هر کدام از آنها وظایف خود را انجام می دهند و همچنین یک تابع برای ایجاد فایل ها درست کردیم . در ادامه به توضیح این توابع خواهیم پرداخت ولی در ابتدا کد ورودی و شرط رفتن به توابع را باهم می بینیم :




```
# the conditions which was set in project description
if choose_item == 'E' :
    steps_to_encryption()
if choose_item == 'D' :
    steps_to_decryption()
```

همان طور که از کد هم پیداست اگر کاربر **E** وارد کرد وارد تابع **steps_to_encryption** می شویم و اگر **D** را به عنوان ورودی وارد کرد وارد تابع **steps_to_decryption** خواهیم شد. حال می خواهیم در ادامه با نحوه کارکرد هرکدام آشنا شویم .

تابع steps_to_encryption ( )

تابع اولی که میخواهیم درباره ی آن صحبت کنیم در واقع به صورت کلی کاری که میکند این است که plaintext را به ciphertext با استفاده از یک key تبدیل می کند. حال کد این تابع را باهم می بینیم :

```
# In this function we encrypt the plaintext
def steps_to_encryption() :

    # reading the key
    with open('Key.txt', 'r') as key_file :
        key = key_file.readline()

    # print(key)
    # generate a random salt in each code running
    salt = os.urandom(16)
    # adding the salt to the key
    key_ = pbkdf2.PBKDF2(key, salt).read(32)
    # printing the encryption key in hex
    print('AES encryption key:', binascii.hexlify(key_))

    # creating the initial vector
    iv = secrets.randbits(256)
    # opening the plaintext we want to encrypt
    with open('Code.txt', 'r') as code_file :
        plaintext = code_file.readline()

    # using AES with CTR mode for encryption
    AES_with_CTR_mode = pyaes.AESModeOfOperationCTR(key_, pyaes.Counter(iv))
    ciphertext = AES_with_CTR_mode.encrypt(plaintext)
    print('ciphertext: ', ciphertext)
    print('Encrypted: ', binascii.hexlify(ciphertext))
    # pass the values to write_file function
    write_file(binascii.hexlify(ciphertext), binascii.hexlify(key_), iv)
```

در ابتدای این تابع فایل Key را میخوانیم که محتوای آن کلیدی است که در دستور پروژه قرار داده شده است و میخواهیم از آن در رمزنگاری و رمزگشایی استفاده کنیم. سپس با استفاده از متد urandom در کتابخانه ی OS یک salt به اندازه 16 بایت ایجاد می کنیم و با استفاده از کتابخانه ی pbkdf2 این salt را به key اضافه می کنیم و سپس 32 بایت آن که همان 256 بیت است را می خوانیم و در اخر این کلید جدید که به صورت بایت است را با استفاده از متد hexlify از کتابخانه ی binascii به hex تبدیل می کنیم و به کاربر نشان می دهیم. در مرحله ی بعد initial vector را با استفاده از متد randbits از کتابخانه ی secrets به اندازه 256 بیت تولید می کنیم . حال وقت آن است که فایلی که میخواهیم رمزنگاری کنیم را باز کنیم و آن را بخوانیم و در یک متغیر به نام plaintext ذخیره کنیم. در مرحله ی اخر باید با استفاده از کتابخانه ی pyaes.AESModeOfOperationCTR که شی این کلاس است به آن key_ و یک شمارنده براساس initial vector بدهیم و سپس آن را در یک متغیر ذخیره می کنیم . حال plaintext را با استفاده از

متد **encrypt** رمزنگاری می کنیم و خود **ciphertext** و حالت **hex** آن را به کاربر نمایش می دهیم. در آخرین مرحله چون میخواهیم برای **key_** ، **ciphertext** و **iv** فایل های جداگانه بسازیم یک تابع **write_file** ایجاد می کنیم و این سه متغیر را به آنها می دهیم. در ادامه با این تابع نیز آشنا می شویم .

تابع **make_file**)

تابعی که میخواهیم درباره ی آن صحبت کنیم وظیفه اش ایجاد فایل است که در ادامه کد آن را می بینیم :

```
# In this function we create necessary files for created ciphertext, key and initial vector
def write_file(ciphertext, password, iv) :
    with open('Encrypted Text.txt', 'wb') as encrypted_file :
        encrypted_file.write(ciphertext)
    encrypted_file.close()

    with open('Encrypted Key.txt', 'wb') as encrypted_key_file :
        encrypted_key_file.write(password)
    encrypted_key_file.close()

    with open('initial vector.txt', 'w') as iv_file :
        # convert initial vector to string
        iv_file.write(str(iv))
    iv_file.close()
```

این تابع سه ورودی دارد که به ترتیب **ciphertext, password, iv** است. کاری که انجام می دهیم این است که برای هر ورودی یک فایل ایجاد کرده و در آن فایل این ورودی ها را ذخیره می کنیم و سپس آن را می بندیم . به عنوان مثال برای ورودی **iv** یک فایل **txt** به نام **initial vector** درست کردیم و **iv** را به صورت **string** در آن ذخیره کردیم زیرا **iv** یک عدد صحیح است و سپس فایل تولید شده را بستیم . در اخر به سراغ تابع **steps_to_decryption** می رویم.

تابع **steps_to_decryption**)

در این تابع کاری که به صورت کلی انجام می دهیم این است که ciphertext را به plaintext ای که در اول برنامه داشتیم تبدیل کنیم. کد این تابع به صورت زیر است :

```
in this function we decrypt the ciphertext
def steps_to_decryption() :

    # reading ciphertext from its file and print it
    with open('Encrypted Text.txt', 'rb') as encrypted_text_file :
        ciphertext = encrypted_text_file.readline()
        ciphertext = binascii.unhexlify(ciphertext)
    print("ciphertext:", ciphertext)

    # reading Encrypted Key from its file and print it
    with open('Encrypted Key.txt', 'rb') as key_encrypted_file :
        encrypted_key = key_encrypted_file.readline()
        encrypted_key = binascii.unhexlify(encrypted_key)
    print("Encrypted key: ", encrypted_key)

    # reading the initial vector from its file and print it
    with open('initial vector.txt', 'r') as initial_vector_file :
        iv = initial_vector_file.readline()
        # convert initial vector to integer
        iv = int(iv)
    print("Initial Vector: ", iv)

        #read_decrypt.close()

    #read_decrypt = str.encode(read_decrypt)
    #print(read_decrypt)

    # AES wit CTR mode
    AES_with_CTR_mode = pyaes.AESModeOfOperationCTR(encrypted_key, pyaes.Counter(iv))
    # decrypted the ciphertext
    decrypted = AES_with_CTR_mode.decrypt(ciphertext)
    # print the initial text which we started with
    print('The Initial Text is :', decrypted)
```

در این تابع کاری که در ابتدای امر می کنیم این است که هر سه فایلی که در تابع قبل درست کردیم را باز می کنیم ، می خوانیم و در متغیری ذخیره می کنیم و در آخر هم آن ها را برای اطمینان به کاربر نشان می دهیم . فقط توجه کنید که چون initial vector باید به صورت عدد صحیح استفاده شود ، مقداری که از فایل می خوانیم را به عدد صحیح تبدیل کردیم. سپس مانند قسمت encryption یک AES با mode کاری CTR درست می کنیم ولی در اینجا به جای متد encrypt از متد decrypt استفاده می کنیم و ciphertext را به آن می دهیم و plaintext ایجاد شده را به کاربر نمایش می دهیم. در این جا کار ما در پروژه به اتمام می رسد :)