مبانی امنیت – پروژه دوم

بخش اول و دوم

کلید (که در مرحله هشتم گفته شد برابر *AUT*ICTSec*2022* باشد) در فایل env. قرار داده شده و با استفاده از دو کتابخانه os و dotenv خوانده میشود. برای خواندن پسورد و ساختن یک salt رندوم دو تابع زیر تعریف شده اند:

```python
def get_encryption_key() -> str:
    """ Returns the encryption key from the .env file """
    return os.getenv("ENCRYPTION_KEY")


def generate_salt() -> str:
    """ Returns a random salt """
    salt = os.urandom(16)   # 128-bit salt
    return base64.b64encode(salt).decode()
```

همانطور که مشخص است، salt ساخته شده 128 بیت داشته و پسورد به صورت پیش فرض از فایل env. خوانده میشود.

بخش سوم

با استفاده از کتابخانه PBKDF2، دو رشته salt و key با یکدیگر ترکیب شده و یک رشته 256 بیتی خروجی داده میشود. این کد در قالب تابع زیر تعریف شده:

```python
def enforce_encryption_key(key: str, salt: str) -> str:
    """ Enforces the encryption key by adding the salt to it, extend to 256 bits and hexlify it """
    return PBKDF2(key, salt).read(32) # 256-bit key
```

برای نمایش خروجی این تابع فایل main.py را اجرا کنید.

بخش چهارم

تابع get_vi برای تولید مقدار بردار اولیه نوشته شده:

```python
def get_vi() -> bytes:
    """ Returns a random initialization vector """
    return secrets.randbits(128)   # 128-bit IV
```

این تابع مقدار VI برابر 16 بایت (128 بیت) باز میگرداند که از جنس bytes است.

بخش پنجم

سه تابع برای رمزنگاری کردن رشته با استفاده از کلید و VI، تابع دیگر برای ذخیره سازی بایت باینری، و تابع سوم برای خواندن مقدار مورد نیاز از فایل txt. داخل فایل نوشته شده اند:

مبانی امنیت – پروژه دوم

```python
def read_plaintext() -> str:
    """ Reads the plaintext from the .env file """
    with open("plaintext.txt", "r") as f:
        return f.read()


def encrypt_plaintext(pbp: str, vi: int, plaintext: str) -> bytes:
    """ Encrypts the plaintext using AES-256-CBC """
    return pyaes.AESModeOfOperationCTR(pbp, pyaes.Counter(vi)).encrypt(plaintext)


def write_ciphertext_to_file(ciphertext: bytes, path: str = "ciphertext.txt") -> str:
    """ Saves the ciphertext to a file as binary """
    with open(path, "wb") as f:
        f.write(ciphertext)
```

همانطور که مشاهده میشود تابع read_plaintext محتویات فایل را بازگردانده، سپس تابع encrypt_plaintext آن را رمزنگاری کرده و در آخر تابع write_ciphertext_to_file آن را در فایل ذخیره میکند.

بخش ششم

مطابق شکل زیر تو تابع برای decryption و خواندن مقدار رمزنگاری شده از فایل نوشته شده اند:

```python
def decrypt_ciphertext(pbp: str, vi: int, ciphertext: bytes) -> str:
    """ Decrypts the ciphertext using AES-CTR """
    return pyaes.AESModeOfOperationCTR(pbp, pyaes.Counter(vi)).decrypt(ciphertext).decode()


def read_ciphertext_from_file(path: str = "ciphertext.txt") -> bytes:
    """ Reads the ciphertext from a file as binary """
    with open(path, "rb") as f:
        return f.read()
```

ابتدا تابع read_ciphertext_from_file اجرا شده و به عنوان خروجی مقدار رمزنگاری شده را از فایل برمیگرداند. سپس تابع decrypt_ciphertext آن را به رشته اصلی برمیگرداند.

بخش هفتم و هشتم

انجام شد.

بخش نهم

با اجرای فایل interactive.py میتوانید به صورت تعاملی با برنامه ارتباط برقرار کنید.

نکات:

- فایل رمز در env. ذخیره میشود.
- برای اجرا یکباره کل برنامه و توابع، فایل main.py را اجرا کرده و برای برنامه تعاملی، فایل interactive.py را اجرا کنید.