دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

مبانی امنیت و اطلاعات

(پاییز ۱۴۰۱)

تمرین عملی دوم

محمد چوپان ۹۸۳۱۱۲۵

گزارش پروژه :

توضیحات فایل ها و کد ها درون کد موجود می باشد.

توضیحات کد :

در ابتدا یک فایل با اسم handleFile.py برای کار با فایل ایجاد کرده و تمامی توابع موجودی که برای کار با فایل نیاز داریم را پیاده سازی میکنیم . برای خواندن کلید ، خواندن ورودی، خواندن متن رمزنگاری شده و نوشتن در مقدار های جدید همچنین ۲ فایل برای ذخیره سازی بردار ورودی و کلید همراه با سالت نیز اضافه میکنیم.

به این منظور که در هر بار رمزنگاری کلید ها عوض می شود برای نگهداری آن ها هم ۲ فایل کپی برای کاربر در نظر میگیریم. این توابع به خواندن و نوشتن در فایل کمک می کنند.

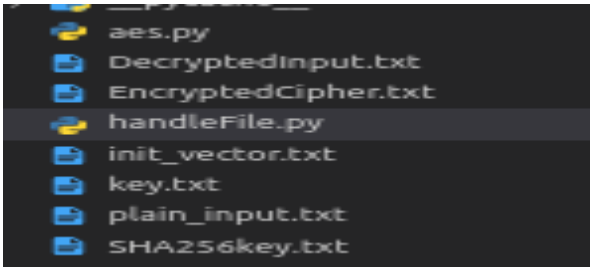
به صورت زیر :

```python
#Read key file from text file
def read_key() -> str:
    """Read key file from text file
    Returns:
        string: key to be used for encryption/decryption
    """      You, 4 hours ago • feat :doc for Readfile.py …
    try:
        with open('key.txt', 'r') as f:
            key :str = f.read()
            if(key == ''):
                print("Key file is empty")
                exit()
            return key

    except FileNotFoundError:
        print("Key file not found")
        exit()
#Read cipher file from text file
def read_cipher() -> str:
    """Read cipher file from text file
    Returns:
        string: cipher text to be decrypted
    """
    try:
        with open('EncryptedCipher.txt', 'rb') as f:
            cipher: str = f.read()
            if(cipher == ''):
                print("cipher file is empty")
                exit()
        return cipher

    except FileNotFoundError:
        print("Cipher file not found")
        exit()
#Read input file from text file
def read_input() -> str:
    """Read input file from text file

    Returns:
        string: input text to be encrypted
    """
    try:
        with open('plain_input.txt', 'r') as f:
            input :str = f.read()
            if(input == ''):
                print("Input file is empty")
                exit()
        return input
    except FileNotFoundError:
        print("Input file not found")
        exit()
```

```python
#Write encrypted to text file
def write_encrypted(encrypted: bytes) -> bool:
    """Write encrypted to text file

    Args:
        encrypted (bytes): encrypted text
    """
    try:
        with open('EncryptedCipher.txt', 'wb') as f:
            f.write(encrypted)

        return True
    except:
        return False

#Write decrypted to text file
def write_decrypted(decrypted: bytes) -> None:
    """Write decrypted to text file

    Args:
        decrypted (bytes): decrypted text
    """
    try:
        with open('DecryptedInput.txt', 'wb') as f:
            f.write(decrypted)
        return True
    except:
        return False

def write_key(key: bytes) -> None:
    """Write key to text file

    Args:
        key (bytes): key to be written to text file
    """
    try:
        with open('SHA256key.txt', 'wb') as f:
            f.write(key)
        return True
    except :

        return False

def write_init_vector(init_vector: int) -> None:
    """Write initial vector to text file

    Args:
        init_vector (int): initial vector to be written to text file
    """
    try:
        with open('init_vector.txt', 'w') as f:
            f.write(str(init_vector))
        return True
    except:
        return False
```

```python
def read_init_vector() -> str:
    """Read initial vector from text file

    Returns:
        string: initial vector to be used for decryption
    """
    try:
        with open('init_vector.txt', 'r') as f:
            init_vector: str = f.read()
            if(init_vector == ''):
                print("init_vector file is empty")
                exit()
        return int(init_vector)
    except FileNotFoundError:
        print("init_vector file not found")
        exit()
def read_256_key() -> bytes:
    """Read 256 bit key from text file

    Returns:
        string: 256 bit key to be used for decryption
    """
    try:
        with open('SHA256key.txt', 'rb') as f:
            key: str = f.read()
            if(key == ''):
                print("key file is empty")
                exit()      You, 2 hours ago • feat: read iv and 256 key a
        return key
    except FileNotFoundError:
        print("key file not found")
        exit()

def copy_256_key() -> None:
    """Copy 256 bit key to key.txt
    """
    try:
        with open('SHA256key.txt', 'rb') as f:
            key: str = f.read()
            if(key == ''):
                print("key file is empty")
                exit()
        with open('SHA256key_copy.txt', 'wb') as f:
            f.write(key)
    except FileNotFoundError:
        print("key file not found")
        exit()
def copy_init_vector() -> None:
    """Copy initial vector to init_vector.txt
    """
    try:
        with open('init_vector.txt', 'r') as f:
            init_vector: str = f.read()
            if(init_vector == ''):
                print("init_vector file is empty")
                exit()
        with open('init_vector_copy.txt', 'w') as f:
            f.write(init_vector)
    except FileNotFoundError:
        print("init_vector file not found")
        exit()
```

در کل پروژه فایل ها به صورت زیر است :



فایل key.txt :

برای کلید اولی

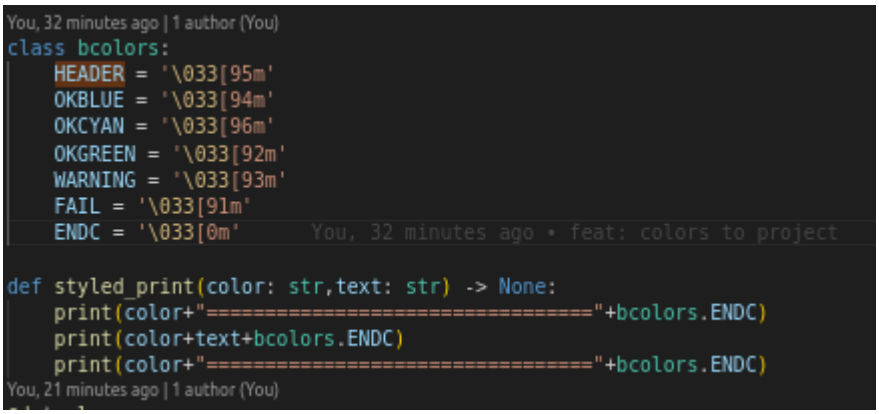
فایل SHA256key.txt :

فایل کلید به همراه سالت

فایل init_vector.txt فایل بردار ها است.

فایل های دیگر هم از نام گذاری آن ها مشخص است.

درون فایل aes.py که فایل اصلی ما است به شکل زیر است :



یک کلاس و تابع پرینت برای پرینت های رنگی

کلاس AES_CTR که مقدار کلید و بردار را نگهداری میکند توضیح توابع درون عکس ها موجودند.

```python
@dataclass
class AES_CTR:
    """AES_CTR class for encrypting and decrypting files using AES in counter mode
    Attributes:
        key: key for AES encryption
        init_vector: initial vector for AES encryption
    """
    key: str = read_key()      You, 3 hours ago • feat: initialize AES_CTR class …
    initial_vector :  str = ""
```

```python
    def create_salt(self) -> bytes:
        """Create a random salt for the key
        Returns:
            random salt
        """
        return urandom(16)

    def convert_key_256 (self ) -> None:      You, 3 hours ago • feat: initialize AES_CTR
        """Convert the key to 256 bits
        """
        salt : bytes = self.create_salt()
        self.key = pbkdf2.PBKDF2(self.key, salt).read(32)
    def show_key_hex(self) -> None:
        """Show the key in hex
        """
        styled_print(bcolors.OKGREEN,f'Algorithm key is : {binascii.hexlify(self.key)}')
    def write_key(self) -> None:
        """Write key to text file
        """
        if(write_key(self.key)):
            styled_print(bcolors.OKGREEN,"Key written to file")
        else:
            styled_print(bcolors.FAIL,"Error writing key to file")
    def create_init_vector(self) -> None:
        """Create a random initial vector for AES encryption
        """
        self.initial_vector = secrets.randbits(256)
    def write_init_vector(self) -> None:
        """Write initial vector to text file
        """
        if(write_init_vector(self.initial_vector)):
            styled_print(bcolors.OKGREEN,"Initial vector written to file")
        else:
            styled_print(bcolors.FAIL,"Error writing initial vector to file")
```

```
def read_256_key(self) -> None:
    """Read key from text file
    """
    self.key = read_256_key()
def read_init_vector(self) -> None:
    """Read initial vector from text file
    """
    self.initial_vector = read_init_vector()
def encrypt_initalize(self) -> None:
    """Initialize the encryption
    """
    self.convert_key_256()
    self.show_key_hex()
    self.write_key()
    self.create_init_vector()
    self.write_init_vector()
def decrypt_initalize(self) -> None:
    """Initialize the decryption
    """
    self.read_256_key()
    self.show_key_hex()
    self.read_init_vector()
```

دو تابع initialize برای این هستند که اگر رمزنگاری هست کلید تولید کند و اگر رمزگشایی قبلی ها را از فایل ها بخواند.

کلاس Encrypy :

```python
class Encrypt(AES_CTR):
    """Encrypt class for encrypt
    Attributes:
        key: key for AES encryption
        init_vector: initial vector for AES encryption
        plaintext: plaintext for AES encryption
        aes: pyaes object for AES encryption
    """
    def __init__(self) -> None:
        """Initialize Encrypt class
        """
        super().__init__()
        super().encrypt_initalize()
        self.aes :object = pyaes.AESModeOfOperationCTR(self.key, pyaes.Counter(self.initial_vector))
        self.plaintext : str = read_input()
    def encrypt(self) -> str:
        """Encrypt the plaintext
        Returns:
            encrypted string
        """
        return self.aes.encrypt(self.plaintext)
    def write_cipher(self, cipher: bytes) -> None:
        """Write cipher to text file
        Args:
            cipher (bytes): encrypted string
        """
        if(write_encrypted(cipher)):
            styled_print(bcolors.OKGREEN,"Cipher written to file")
        else:
            styled_print(bcolors.FAIL,"Error writing cipher to file")
    def show_cipher_hex(self, cipher: bytes) -> None:
        """Show the cipher in hex
        Args:
            cipher (bytes): encrypted string
        """
        styled_print(bcolors.OKGREEN,f'Encrypted cipher is : {binascii.hexlify(cipher)}')
    def encrypt_file(self) -> None:
        """Encrypt the file
        """
        cipher = self.encrypt()
        self.show_cipher_hex(cipher)
        self.write_cipher(cipher)
```

کلاس Decrypt:

```
You, 23 minutes ago | 1 author (You)
class Decrypt(AES_CTR):
    """Decrypt class for decrypt
    Attributes:
        key: key for AES decryption
        init_vector: initial vector for AES decryption
        ciphertext: ciphertext for AES decryption
        aes: pyaes object for AES decryption
    """
    def __init__(self) -> None:
        """Initialize Decrypt class
        """
        super().__init__()
        super().decrypt_initalize()
        self.aes : object = pyaes.AESModeOfOperationCTR(self.key, pyaes.Counter(self.initial_vector))
        self.ciphertext : str = read_cipher()
    def decrypt(self) -> str:
        """Decrypt the ciphertext
        Returns:
            decrypted string
        """
        return self.aes.decrypt(self.ciphertext)
    def write_decrypted(self, decrypted: str) -> None:
        """Write decrypted to text file
        Args:
            decrypted (str): decrypted string
        """
        if(write_decrypted(decrypted)):
            styled_print(bcolors.OKGREEN,"Decrypted written to file")
        else:
            styled_print(bcolors.FAIL,"Error writing decrypted to file")
    def show_decrypted_hex(self, decrypted: str) -> None:
        """Show the decrypted in hex
        Args:
            decrypted (str): decrypted string
        """
        styled_print(bcolors.OKGREEN,f'Decrypted cipher is : {decrypted}')
    def decrypt_file(self) -> None:
        """Decrypt the file
        """
        decrypted = self.decrypt()
        self.show_decrypted_hex(decrypted)
        self.write_decrypted(decrypted)
```

تابعی برای نمایش هدر برنامه :

```
def print_title() -> None:
    styled_print(bcolors.HEADER,"AES-CTR Encryption and Decryption")
```

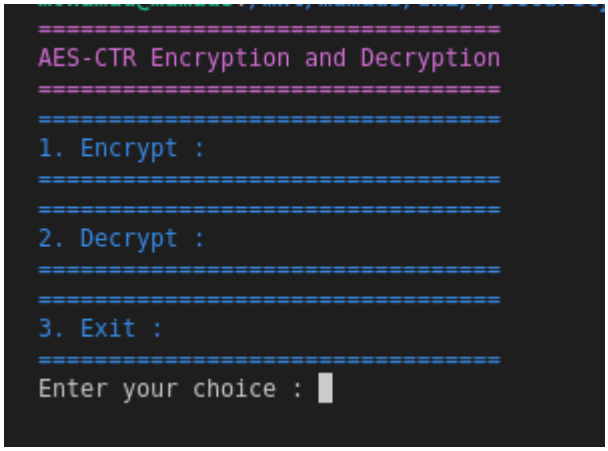
تابعی برای ورودی:

```
def get_inputs() -> list:
    print_title()
    styled_print(bcolors.OKBLUE,"1. Encrypt : ")
    styled_print(bcolors.OKBLUE,"2. Decrypt : ")
    styled_print(bcolors.OKBLUE,"3. Exit : ")
    inputs = input("Enter your choice : ")
    return inputs
```

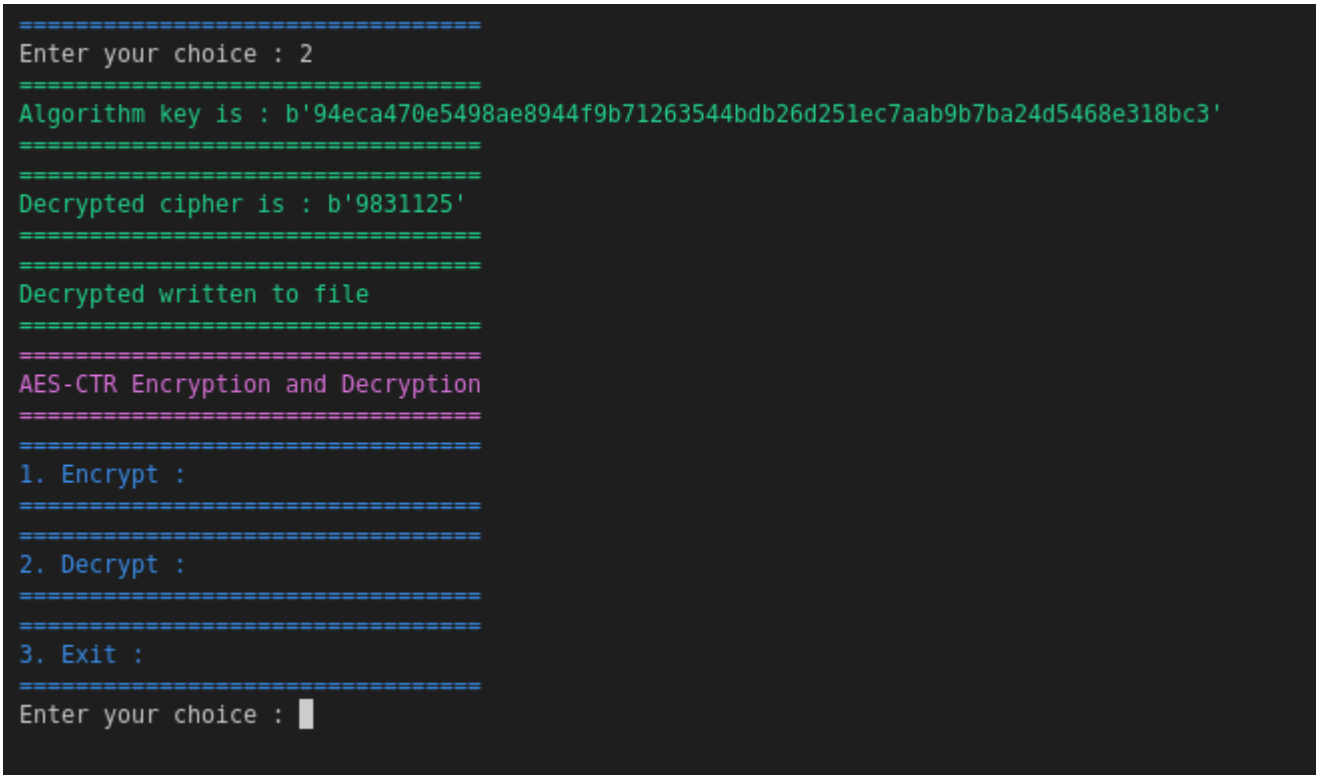
و تابعی برای پردازش ورودی ها و main

```
def handle_inputs(inputs: list) -> None:

    if(inputs == "1"):
        styled_print(bcolors.WARNING,"Are you sure you want to encrypt the file after encrypt keys are resseted? (y/n)")
        choice = input("Enter your choice : ")
        if(choice == "y"):
            styled_print(bcolors.WARNING,'Do you want to copy the key and initial vector to new file? (y/n)')
            new_choice = input("Enter your choice : ")
            if(new_choice == "y"):
                copy_256_key()
                copy_init_vector()
                styled_print(bcolors.OKGREEN,"Key and initial vector copied to new file")
            encrypt = Encrypt()
            encrypt.encrypt_file()
        else:
            print(bcolors.OKCYAN,"Exiting")
            exit()

    elif(inputs == "2"):
        decrypt = Decrypt()
        decrypt.decrypt_file()
    elif(inputs == "3"):
        styled_print(bcolors.OKCYAN,"Exiting")
        exit()
    else:
        styled_print(bcolors.FAIL,"Invalid choice")

def main() -> None:
    while(True):
        inputs = get_inputs()
        handle_inputs(inputs)

if __name__ == "__main__":
    main()
```

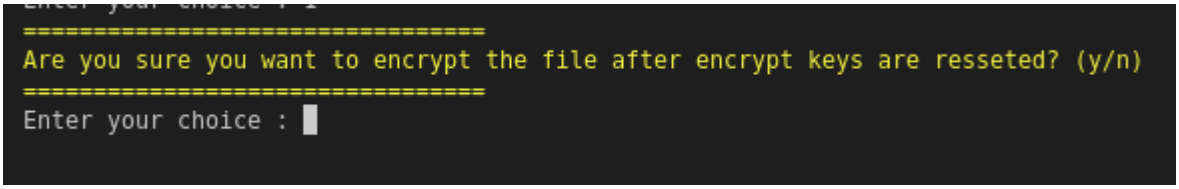
در نهایت در صورت اجرای برنامه به این شکل است که در صورت اجرا فایل :



اگر گزینه ۳ انتخاب شود از برنامه خارج می شود در صورت انتخاب گزینه ۲ :

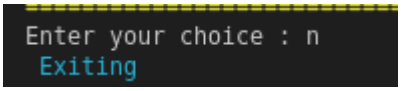


در صورت انتخاب گزینه ۱ :

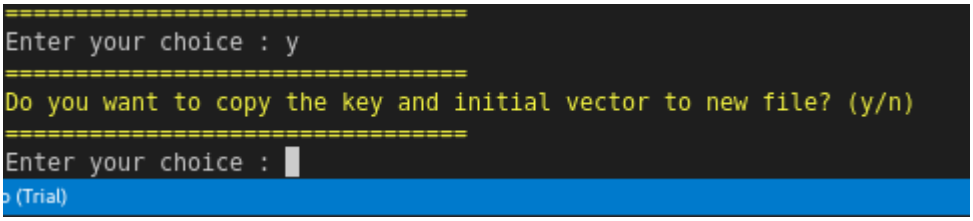


اخطار می دهد که کلید ها از اول تولید می شوند آیا می خواهید ادامه دهید.

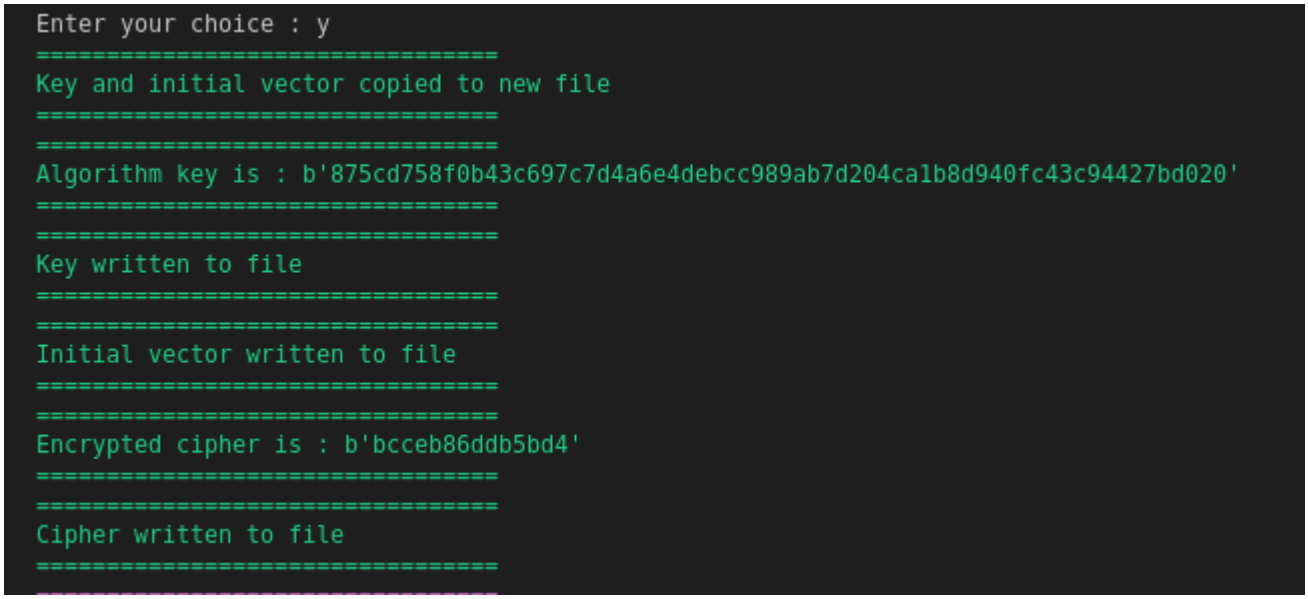
اگر خیر بزنیم از برنامه خارج می شود .



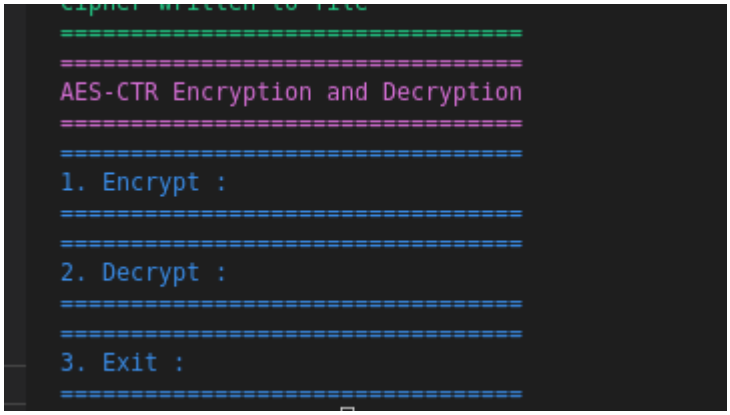
اگر بله بزنیم :



میگوید که آیا میخواهید کلید های قبلی را کپی کنید : اگر بله بزنیم کلید ها را در دو فایل کپی جایگذاری میکند.



و فرایند جدد تکرار می شود :



تمامی فایل ها در پروژه موجود است.

همچنین در صورت نبود فایل ها key.txt و plain_input برای رمزنگاری با خطا مواجه می شویم.