دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

Department of
Computer Engineering

به نام خداوند جان و خرد

تمرین عملی تحویلی سوم

درس مبانی امنیت اطلاعات

امیرفاضل کوزه گر کالجی

9931099

نحوه اجرای برنامه

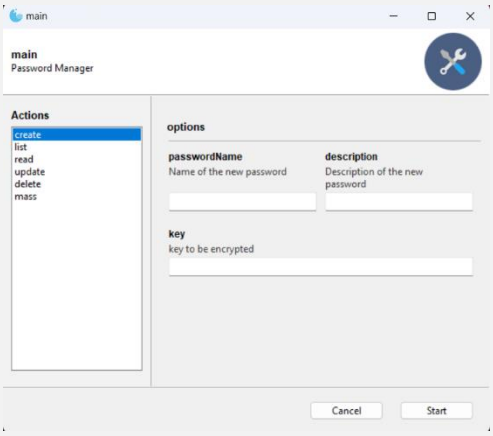
نصب پکیج ها

در گام اول به نصب پکیج های مورد استفاده در برنامه می‌پردازیم. برای اینکار لازم است موارد زیر را انجام دهید:

- وارد پوشه code شوید.
- دستور `pip install -r requirements.txt` را اجرا کنید.

این برنامه به دو نحو اجرا می شود:

- دارای رابط کاربری
- Cli



GUI

برای اجرای برنامه با رابط کاربری نیاز است تا موارد زیر را به ترتیب انجام دهید.

1. پس از نصب پکیج ها در گام اول، وارد پوشه code شوید.
2. دستور `python main.py` را اجرا کنید تا رابط کاربری نشان داده شود.

CLI

برای این کار نیاز است تا decorator مورد استفاده برای رابط کاربری را کامنت کنید. برای این کار:

- وارد پوشه cli/utils شوید و فایل argParser را باز کنید.
- خط شماره 5 که شامل decorator است را، حذف یا کامنت کنید.
- در ادامه دستورات پیاده سازی شده برنامه و کارکرد قطعات کد، به طور مفصل توضیح داده خواهند شد.

ساختار دستورات قابل استفاده

- دستور ساخت (create)
- `python main.py create -n <name> -c <description> -key <key>`

در این دستور نام، توضیحات و کلید مربوطه را می‌گیریم. سپس این کلید توسط الگوریتم AES رمزنگاری و به صورت یه فایل متنی آن را ذخیره می کنیم.

الگوریتم AES به دنبال کلید ذخیره شده در فایل می‌گردد. اگر آن را نیافت، آن را تولید می‌کند. در نهایت از این کلید برای رمزنگاری استفاده می‌کند:

```python
def __init__(self, key_file='./db/aes.key'):
    self.key_file = key_file
    if not os.path.exists(self.key_file):
        self.key = os.urandom(32)
        self.save_key()
    else:
        self.load_key()
```

تابع رمزنگاری نیز به شکل زیر پیاده سازی شده است:

```python
def encrypt(self, message):
    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(message.encode()) + padder.finalize()

    iv = os.urandom(16)
    cipher = Cipher(algorithms.AES(self.key), modes.CBC(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    encrypted_message = encryptor.update(padded_data) + encryptor.finalize()

    return base64.b64encode(iv + encrypted_message).decode('utf-8')
```

دمو:

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code (main)
$ python main.py create -n portal -c my student portal -key 1234
Your password is encrypted to: dYS+470rKh1Gvvkg3iPGWSHSHwcL7OBmCmZyQSVw//I=
```

portal.txt

code > db > keys > portal.txt

```
{
    "name": "portal",
    "description": "my student portal",
    "encrypted_password": "dYS+470rKh1Gvvkg3iPGWSHSHwcL7OBmCmZyQSVw//I="
}
```

- دستور خواندن (read)

```
• python main.py read <name>
```

این دستور، عملکرد بسیار ساده ای داشته، و صرفا در مخزن فایل های ما به دنبال رمز ذخیره شده مان می‌گردد. در نهایت آن رمز را به همرا اطلاعات دیگر ذخیره شده برمی‌گرداند.

دمو

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code
$ python main.py read portal
============================
Name:   portal
Description:    my student portal
Encrypted_password:     dYS+470rKh1Gvvkg3iPGWSHSHwcL7OBmCmZyQSVw//I=
============================
```

- بروزرسانی (update)

```
• python main.py update <name> -key <key>
```

با استفاده از این دستور می‌توانیم کلید رمز ذخیره شده فعلی خود را تغییر دهیم. برنامه در صورت موجود بودن رمز، آن را تغییر می‌دهد.

دمو

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code (main)
$ python main.py update portal -key helloworld
password for key: portal updated successfully!!
your new key encrypted value is: nz+YSG34EYZGLJUDez90q6jigvwRmfywL7XkE1tzOVk=

AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code (main)
$ python main.py update notavailable -key helloworld
this file isnt available yet, cant update it
```

portal.txt | Preview README.md

code > db > keys > portal.txt

```
1  {
2      "name": "portal",
3      "description": "my student portal",
4      "encrypted_password": "nz+YSG34EYZGLJUDez90q6jigvwRmfywL7XkE1tzOVk="
5  }
```

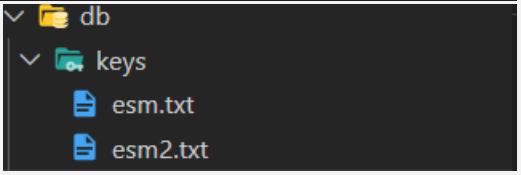
- حذف (delete)

```
python main.py delete <name>
```

با به کارگیری این دستور، با صرفا دادن نام رمز، فایل مربوط به اطلاعات ذخیره شده درباره آن به کلی از سیستم محلی حذف می‌شود.

دمو

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code (main)
$ python main.py delete portal
password deleted successfully
```



می‌توان مشاهده کرد که رمز مربوطه به کل حذف شده است.

- نشان دادن همه رمز ها (list)

```
python main.py list
```

این دستور برای مشاهده همه کلید های رمز شده در این سیستم است.

از پیش برخی رمز ها را تعریف کرده ایم. در ادامه کارکرد این دستور را مشاهده خواهیم کرد.

دمو

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/projects/3/code (main)
$ python main.py list
============================
1)
Name:   esm
Description:    in male mane
Encrypted_password:     i5Kh7jzF72hOa4NmXwzVWftzVjgsHV9FrsyZm2jZWl0=
============================
============================
2)
Name:   esm2
Description:    are kholase
Encrypted_password:     S3Md9paK3EzKHgw8hpuM5TBY8d/CfAbUDhHEXVW3v4k=
============================
```

بخش دوم

در این بخش به تولید و رمزنگاری انبوهی از گذرواژه ها خواهیم پرداخت و در نهایت با استفاده از ابزار statsgen این خروجی ها را بررسی خواهیم کرد.

فایل statsgen موجود در کورسز با ورژن فعلی پایتون تطابق نداشت پس مجبور به ایجاد برخی تغییرات در محتویات آن شدیم.

در ادامه به معرفی دستور دیگری از برنامه خود می‌پردازیم که وظیفه تولید مقدار بسیار زیادی نتیجه رمزنگاری شده از یک رشته دارد. رشته مورد استفاده "0000" است.

- دستور mass

```
python main.py mass -key <key>
```

این دستور یک کلید را گرفته و تقریبا ده هزار بار آن را رمزنگاری می‌کند.

تابع مربوط به رمزنگاری استفاده شده در این بخش با تابع رمزنگاری قبلی اندکی تفاوت دارد.

این تفاوت صرفا به آن دلیل ایجاد شده تا بتوان خروجی هایی به طول های مختلف به دست آورد.

```python
def encrypt_with_random_length(self, message):
    pad_length = random.randint(1, 16)
    random_pad = os.urandom(pad_length)
    padded_message = random_pad + message.encode()

    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(padded_message) + padder.finalize()

    iv = os.urandom(16)
    cipher = Cipher(algorithms.AES(self.key), modes.CBC(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    encrypted_message = encryptor.update(padded_data) + encryptor.finalize()

    return base64.b64encode(iv + encrypted_message).decode('utf-8')
```

دمو

```
AmirFazelK@WSN-476 MINGW64 /e/university/sem7/ICT/ICT-git/ICT/
$ python main.py mass -key 0000
```

mass.txt

code > db > mass_keys > mass.txt

```
1   ppWnvT2qqwf/wgMu3sYBVHwB73OEdQWvySDNuIDFeCo=
2   k2ePvOm+yjZo/ZIsQPl17565/2aBxdeIKbZUU6TSe+8=
3   KEBFxl61VqYmkQhBIjtTRh9UJcS80ZyqRZ50zNZlcN4=
4   0N2FFXauZzwv1B6P6E8/KQADxRJvRQNYc9MoFWMa6f+PElQ0FOl6gGZx6/iFJKyK
5   TrSMbX/uUIJbMbXGDPMym/4db57xxjJ+GzGzBE2SCAGK3NAQoODbg+b1BOyp/4IC
6   +g8wAqGrFoQa1vSxT94wR3CQ2fMUUdvCcngoU/HmWIk=
7   AdC1+d70GdlP9Q803OnfoEf9rqOxm1ljn6bBgfVPwtwphh2WR1AZhAqH79IF1MQ+
8   1kdwyK07VMzKTta0NuB1szFIe5RuWv/B1XhgqE5jPsNkvQAeCrFfBzJRKcS4Zl+Q
9   bPQdO/zao7ofze7WcHB9bR/q+I5esNDioFdEGoRQLjjXljJg4Wwlm4aS5zH1v4b2
10  fu8vNzKLzMiZfnPMSl4C4KIZCDmPIHgidyMtR+QmryLQuRnEeQFr9Re2WPbmNaeM
```

برای گام آخر، بیایید برنامه statsgen را برای فایل تکست خروجی اجرا کنیم تا از الگو های استفاده شده در آن، آگاه شویم:

```
     _ __   __ _  ___| | _
    | '_ \ / _` |/ __| |/ /
    | |_) | (_| | (__|   <
    | .__/ \__,_|\___|_|\_\
    | |
    |_| iphelix@thesprawl.org


[*] Analyzing passwords in [./db/mass_keys/mass.txt]
[+] Analyzing 100% (56/56) of passwords
    NOTE: Statistics below is relative to the number of analyzed passwords, not total number of passwords

[*] Length:
[+]                      44: 73% (41)
[+]                      64: 26% (15)

[*] Character-set:
[+]                     all: 92% (52)
[+]           mixedalphanum: 07% (4)

[*] Password complexity:
[+]                   digit: min(2) max(16)
[+]                   lower: min(12) max(35)
[+]                   upper: min(11) max(33)
[+]                 special: min(0) max(5)

[*] Simple Masks:
[+]               othermask: 100% (56)
```

```
[*] Advanced Masks:
[+] ?l?l?u?l?l?u?d?l?l?l?l?s?l?l?u?l?d?l?u?u?u?l?u?d?d?u?u?l?u?u?l?l?u?u?u?l?u?u?u?l?u?l?s: 01% (1)
[+] ?l?d?l?u?l?u?l?s?l?l?u?l?s?u?u?l?u?u?l?d?d?d?d?s?d?l?u?l?l?l?u?u?l?u?u?u?d?u?u?l?s?d?s: 01% (1)
[+] ?u?u?u?u?l?l?d?d?u?l?u?l?l?u?l?u?u?l?l?u?u?l?d?u?u?l?u?d?d?u?l?l?u?u?d?d?l?u?u?l?l?u?d?s: 01% (1)
[+] ?d?u?d?u?u?u?l?l?u?l?l?l?d?u?d?u?d?u?d?s?u?u?u?l?u?u?l?u?u?u?u?l?d?u?l?u?u?u?l?d?l?s?u?u?l?u?d?u?u?l?d?l?u?u?l?d?s?l?u?u?u?l?u: 01% (1)
[+] ?u?l?u?u?l?u?s?l?u?u?u?l?u?l?u?u?u?u?u?l?l?s?d?l?l?d?d?l?l?l?u?s?u?l?u?l?u?u?d?u?u?u?u?u?d?u?u?u?l?u?u?l?l?s?l?d?u?u?l?l?s?d?u?u: 01% (1)
[+] ?s?l?d?l?u?l?u?l?u?l?u?l?d?l?u?l?u?d?d?l?u?d?u?u?d?l?u?u?u?l?l?u?l?l?l?l?u?s?u?l?u?u?l?s: 01% (1)
[+] ?u?d?l?l?l?d?d?l?l?d?l?d?u?u?l?l?u?u?l?l?u?u?u?u?u?l?u?d?l?u?s?l?u?d?l?l?u?d?u?l?l?l?l?l?u?l?u?l?u?l?l?l?u?l?l?l?d?u?l?u?u?l?u?d: 01% (1)
[+] ?l?l?l?d?l?u?l?l?u?l?d?d?u?l?d?l?l?u?d?u?l?u?u?d?d?l?u?l?l?u?l?l?u?u?l?u?u?d?u?u?u?u?l?s: 01% (1)
[+] ?l?d?u?u?u?d?l?l?l?l?l?l?s?u?l?l?u?u?u?u?u?l?d?l?l?l?l?u?s?d?u?d?l?l?u?u?l?d?u?l?l?l?l?s: 01% (1)
[+] ?d?l?u?u?l?l?d?l?l?l?u?s?u?s?u?l?l?l?u?l?l?u?l?u?u?d?u?u?l?l?l?u?u?d?u?d?l?l?u?s?l?u?d?s: 01% (1)
[+] ?d?l?l?d?d?l?d?l?d?u?l?l?l?s?l?d?d?u?u?l?d?l?u?l?s?l?l?d?u?l?l?l?u?u?u?l?l?s?u?l?d?u?d?u?u?l?d?u?l?l?u?l?u?d?u?u?u?l?u?l?u?u?u?l: 01% (1)
[+] ?u?u?l?u?u?u?u?d?u?l?d?l?u?d?l?u?l?u?l?u?l?u?l?u?u?d?l?u?u?l?l?u?u?u?s?d?u?l?u?u?u?u?u?s: 01% (1)
[+] ?u?d?s?l?u?l?l?l?l?d?d?l?l?l?u?u?l?l?l?u?u?u?l?l?u?l?l?u?l?d?l?u?u?l?d?u?l?l?l?u?l?u?d?s: 01% (1)
[+] ?d?l?s?l?l?u?u?u?u?l?u?u?l?l?u?l?u?u?u?l?d?l?l?u?l?u?d?u?l?l?d?u?l?u?l?d?l?u?u?l?l?l?u?s: 01% (1)
[+] ?s?d?l?l?d?u?u?u?u?l?l?l?l?l?u?l?l?u?d?l?u?u?d?u?l?l?l?d?l?l?u?u?l?l?d?l?u?s?l?l?u?l?u?s: 01% (1)
[+] ?u?u?u?u?d?l?u?l?l?l?u?l?u?l?l?l?l?l?l?u?u?u?l?u?u?l?u?l?l?d?l?l?l?d?u?u?u?d?u?d?s?l?u?l?l?u?u?u?u?l?u?u?l?u?l?l?l?u?u?u?l?u?u?l: 01% (1)
[+] ?l?s?u?d?u?l?u?d?l?l?l?u?l?l?l?l?l?l?d?l?l?l?l?l?l?l?d?u?s?u?l?l?d?u?d?l?l?u?d?u?u?l?l?u?u?l?u?l?u?u?d?d?l?l?s?l?l?l?l?l?d?l?d: 01% (1)
[+] ?l?u?u?l?u?l?l?u?u?d?l?l?u?l?l?u?l?u?u?l?u?d?u?u?u?l?s?u?u?u?l?l?l?l?l?l?l?u?u?u?l?l?d?u?u?u?u?l?l?u?l?l?l?l?l?s?d?l?l?l?u?l?l: 01% (1)
[+] ?l?l?s?d?d?d?l?l?u?l?l?l?u?u?d?u?l?u?u?d?u?d?l?u?u?l?u?u?u?l?u?u?l?u?u?u?d?u?l?d?u?d?u?l?u?l?u?l?d?l?l?u?d?u?u?l?l?u?l?l?l?l?u?l: 01% (1)
[+] ?d?u?d?d?u?u?u?l?l?u?u?d?l?l?l?l?u?l?u?d?u?d?l?u?l?s?u?u?u?d?u?u?u?l?s?l?l?u?d?u?u?u?l?s: 01% (1)
[+] ?l?l?l?u?u?u?d?l?l?l?d?l?l?l?l?d?u?l?u?l?l?u?u?u?l?u?u?u?l?u?d?l?l?d?u?u?u?l?d?l?l?l?u?u?l?u?u?l?d?u?l?u?u?l?u?l?u?l?l?l?l?u?u?d: 01% (1)
[+] ?l?d?u?l?u?l?u?u?u?u?l?d?d?l?u?d?d?d?l?u?u?d?d?u?d?u?u?l?d?u?u?u?l?l?u?d?l?u?u?u?l?l?l?s: 01% (1)
[+] ?l?u?u?l?l?l?l?l?u?l?d?u?u?u?u?u?u?l?d?l?d?u?u?d?l?d?s?l?l?u?u?u?l?l?u?l?l?d?u?u?l?d?l?s: 01% (1)
[+] ?d?d?l?l?l?u?l?u?u?d?l?l?u?l?u?s?l?u?u?u?l?u?u?u?l?l?d?l?l?l?u?u?d?u?u?l?l?u?d?u?l?l?d?s: 01% (1)
[+] ?s?d?s?u?u?l?u?l?u?u?l?d?u?u?u?l?l?u?u?u?u?u?l?d?l?u?u?l?d?l?l?l?l?l?l?l?u?u?u?d?u?d?s: 01% (1)
[+] ?u?l?d?l?l?l?u?u?u?d?u?l?l?l?d?u?l?l?u?l?l?d?u?d?u?u?d?u?s?u?l?u?u?u?l?u?l?d?u?u?l?u?u?s: 01% (1)
[+] ?l?l?l?d?u?l?l?u?l?u?u?l?u?u?u?u?l?u?l?l?u?l?l?u?u?l?l?u?d?u?l?u?u?u?l?u?l?d?l?l?u?u?d?u?l?d?l?l?u?u?d?l?u?l?d?l?u?l?u?u?d?u?l?u?l: 01% (1)
[+] ?u?u?l?u?l?l?u?l?l?u?u?u?u?u?l?u?u?u?u?l?l?l?l?u?d?u?d?d?u?d?d?l?l?l?d?u?l?l?l?u?l?l?u?l?s: 01% (1)
[+] ?l?u?l?u?l?u?u?u?u?l?u?d?l?u?l?l?l?l?d?l?l?l?l?d?u?u?l?l?u?d?l?u?l?u?u?d?d?u?u?u?u?l?u?l?u?l?d?d?l?d?l?u?u?l?u?l?u?l?u?u?u?l?d?u?l: 01% (1)
[+] ?u?u?l?l?l?l?l?u?l?l?l?l?d?u?u?l?u?l?l?l?u?l?l?d?u?u?l?u?l?d?u?l?u?l?l?l?u?l?d?d?d?l?u?u?l?l?u?d?l?l?d?u?u?l?l?u?d?d?d?d?l?l?l?u: 01% (1)
[+] ?u?d?u?l?l?l?u?l?l?u?l?d?u?l?l?u?l?u?u?u?l?s?l?l?u?u?u?l?l?l?u?l?l?l?d?u?d?l?d?l?l?l?l?s: 01% (1)
[+] ?u?l?d?d?l?d?l?l?l?l?u?l?l?u?d?u?l?u?s?d?l?l?u?u?u?d?d?l?l?u?l?l?u?l?d?l?l?u?u?u?l?l?u?s: 01% (1)
[+] ?u?u?l?u?u?l?l?l?u?l?u?l?u?u?u?l?u?l?u?l?l?l?d?d?d?u?u?l?l?u?u?d?u?u?u?u?u?l?l?u?u?s: 01% (1)
[+] ?l?s?l?u?l?d?d?d?l?u?l?u?u?u?s?d?l?u?l?d?u?l?l?d?l?u?u?d?u?u?u?l?d?l?l?u?l?l?u?l?u?l?u?d?u?u?l?u?l?u?l?l?l?d?u?u?u?l?u?u?l?d?u?u: 01% (1)
[+] ?l?u?l?d?l?l?d?d?l?u?u?l?l?l?l?l?l?d?u?l?l?l?u?u?u?u?l?u?l?u?u?l?l?d?u?l?u?u?l?d?l?u?l?s: 01% (1)
[+] ?d?u?d?l?s?u?l?u?u?d?u?u?l?s?d?d?d?u?u?u?d?u?u?l?u?d?u?d?l?l?l?l?l?u?d?u?l?d?l?l?l?l?u?s: 01% (1)
[+] ?u?d?l?l?u?u?u?u?d?u?l?u?u?l?l?d?u?l?l?d?l?u?u?u?l?u?l?l?u?l?d?u?u?d?u?u?u?l?s?u?d?u?u?s: 01% (1)
[+] ?l?u?u?u?l?u?l?l?l?l?l?d?l?u?l?d?l?l?l?l?u?l?d?u?u?u?u?u?u?l?u?u?u?u?l?l?u?u?u?d?u?u?l?s: 01% (1)
[+] ?d?l?d?l?u?d?d?l?l?l?u?u?d?u?u?l?l?l?u?d?u?u?l?l?u?d?u?u?u?d?u?s?l?d?u?u?l?l?l?u?l?l?l?l?u?l?l?l?l?u?d?d?s?l?d?d?u?l?d?u?u?d?l?u?l?d?l?u?l: 01% (1)
[+] ?l?u?l?u?l?l?u?d?u?l?u?u?l?l?u?l?u?l?u?u?l?l?d?u?u?u?l?u?u?u?u?l?u?d?d?l?l?u?u?s?l?d?u?s: 01% (1)
[+] ?u?l?d?d?l?d?l?l?d?u?u?u?l?u?u?l?u?u?l?l?u?d?u?l?s?u?l?l?d?l?l?l?u?l?u?l?u?l?l?l?u?l?l?s: 01% (1)
[+] ?s?l?l?d?d?l?d?u?l?u?u?u?u?u?l?l?u?u?u?u?s?l?s?l?l?u?l?d?l?d?l?l?d?l?u?d?u?d?u?l?u?l?u?s: 01% (1)
[+] ?s?l?u?u?u?l?l?l?u?u?u?d?l?u?l?d?u?u?u?u?d?l?l?l?l?u?l?u?u?u?l?l?l?l?l?u?l?u?l?u?l?d?u?s: 01% (1)
```