

بخش اول:

در این قسمت از کتابخانه های زیر استفاده شده است به منظور گفته شده :

```
# Used libraries:
import argparse # For command line argument processing
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC # For
deriving key
from cryptography.hazmat.backends import default_backend # For selecting
cryptography backend
from cryptography.hazmat.primitives import hashes # For using hash functions
from base64 import urlsafe_b64encode, urlsafe_b64decode # For performing
Base64 operations
import os # For generating random numbers and interacting with files
```

از تابع های زیر به منظور های گفته شده است:

یک کلاس به منظور عملکرد کلی سیستم و تعریف تابع ها:

```
class PasswordManager:
    """
    PasswordManager class for managing encrypted passwords.
    """
```

تعریف کلید اصلی برای رمز کردن رمز عبور ها و ذخیره آنها در زمان شی سازی از کلاس :

```
def __init__(self, key):
    """
    Initializes the PasswordManager with a master key.

    Parameters:
    - key (str): The master key for password encryption and decryption.
    """
    self.key = key
```

تابع رمزنگاری با استفاده از AES و PBKDF2 :

```
def encrypt_password(self, password):
    """
    Encrypts a password using PBKDF2 key derivation and AES encryption.

    Parameters:
    - password (str): The password to be encrypted.

    Returns:
    - str: The encrypted password.
    """
    kdf = PBKDF2HMAC(
```

```

        algorithm=hashes.SHA256(),
        length=32,
        salt=os.urandom(16),
        iterations=100000,
        backend=default_backend()
    )
    key = urlsafe_b64encode(kdf.derive(self.key.encode()))
    cipher_text = urlsafe_b64encode(password.encode())
    return key + cipher_text

```

تابع رمزگشایی:

```

def decrypt_password(self, encrypted_password):
    """
    Decrypts an encrypted password using the master key.

    Parameters:
    - encrypted_password (str): The encrypted password.

    Returns:
    - str: The decrypted password.
    """
    key = urlsafe_b64encode(self.key.encode())
    cipher_text = urlsafe_b64decode(encrypted_password)
    return urlsafe_b64decode(cipher_text[len(key):]).decode()

```

تابع ذخیره سازی:

این تابع نام و کامنت مربوط به رمز رمزنگاری شده را به همراه خود رمزنگاری رمز در فایل تکستی ذخیره میکند.

```

def save_password(self, name, comment, password):
    """
    Saves a new password entry to a text file.

    Parameters:
    - name (str): The name associated with the password.
    - comment (str): A comment or description for the password.
    - password (str): The password to be saved.
    """
    encrypted_password = self.encrypt_password(password)
    with open('passwords.txt', 'a') as file:
        file.write(f"{name}:{comment}:{encrypted_password}\n")

```

تابع های نمایش کل رمزهای ذخیره شده یا نمایش یک رمز با نام داده شده:

```

@staticmethod
def show_passwords():
    """

```

```

Shows all passwords stored in the text file.
"""
with open('passwords.txt', 'r') as file:
    for line in file:
        data = line.strip().split(':')
        if len(data) >= 3:
            name, comment, _ = data[:3]
            print(f"Name: {name}, Comment: {comment}")
        else:
            print(f"Invalid line: {line}")

def show_password(self, name):
    """
    Shows details of a specific password.

    Parameters:
    - name (str): The name associated with the password.
    """
    with open('passwords.txt', 'r') as file:
        for line in file:
            data = line.strip().split(':')
            if data[0] == name:
                decrypted_password = self.decrypt_password(data[2])
                print(f"Name: {data[0]}, Comment: {data[1]}, Password: {decrypted_password}")

```

تابع های بروزرسانی و حذف رمز در فایل ذخیره سازی:

```

def update_password(self, name, new_password):
    """
    Updates the password for a specific entry.

    Parameters:
    - name (str): The name associated with the password.
    - new_password (str): The new password to be updated.
    """
    with open('passwords.txt', 'r') as file:
        lines = file.readlines()
    with open('passwords.txt', 'w') as file:
        for line in lines:
            data = line.strip().split(':')
            if data[0] == name:
                updated_line = f"{name}:{data[1]}:{self.encrypt_password(new_password).decode()}\n"
                file.write(updated_line)
            else:
                file.write(line)

    @staticmethod
    def delete_password(names):
        """
        Deletes password entries for specified names.

```

```

Parameters:
- names (list): A list of names for passwords to be deleted.
"""
with open('passwords.txt', 'r') as file:
    lines = file.readlines()
with open('passwords.txt', 'w') as file:
    for line in lines:
        data = line.strip().split(':')
        if data[0] not in names:
            file.write(line)

```

تابع اصلی برای تعریف نحوه گرفتن ورودی در cmd:

```

def main():
    """
    Main function for handling command-line interface operations.
    """
    parser = argparse.ArgumentParser(description="Password Manager CLI")
    parser.add_argument("--newpass", help="Create a new password", nargs=3)
    #parser.add_argument("--c", help="Comment for the password", nargs=1)
    #parser.add_argument("--key", help="User simple password", nargs=1)
    parser.add_argument("--showpass", help="Show all passwords",
action="store_true")
    parser.add_argument("--sel", help="Show a specific password", nargs=1)
    parser.add_argument("--update", help="Update a password", nargs=1)
    parser.add_argument("--delete", help="Delete a password", nargs='+') #
Allow one or more names
    parser.add_argument("--test", help="run test part", action="store_true")

    args = parser.parse_args()

    key = input("Enter your master password: ")
    password_manager = PasswordManager(key)

    '''
    if args.newpass and args.c and args.key:
        name = args.newpass
        comment = args.c
        user_password = args.key
        password_manager.save_password(name, comment, user_password)
    '''

    if args.newpass:
        name, comment, simple_password = args.newpass
        password_manager.save_password(name, comment, simple_password)
    elif args.showpass:
        password_manager.show_passwords()
    elif args.sel:
        name = args.sel[0]
        password_manager.show_password(name)
    elif args.update:
        name = args.update[0]
        new_password = input("Enter the new password: ")

```

```
        password_manager.update_password(name, new_password)
    elif args.delete:
        names = args.delete
        password_manager.delete_password(names)
    elif args.test:
        password_manager.save_test()
    else:
        print("Invalid command. Use --help for usage information.")

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

تست سناریو های مختلف:

ساخت رمز جدید:

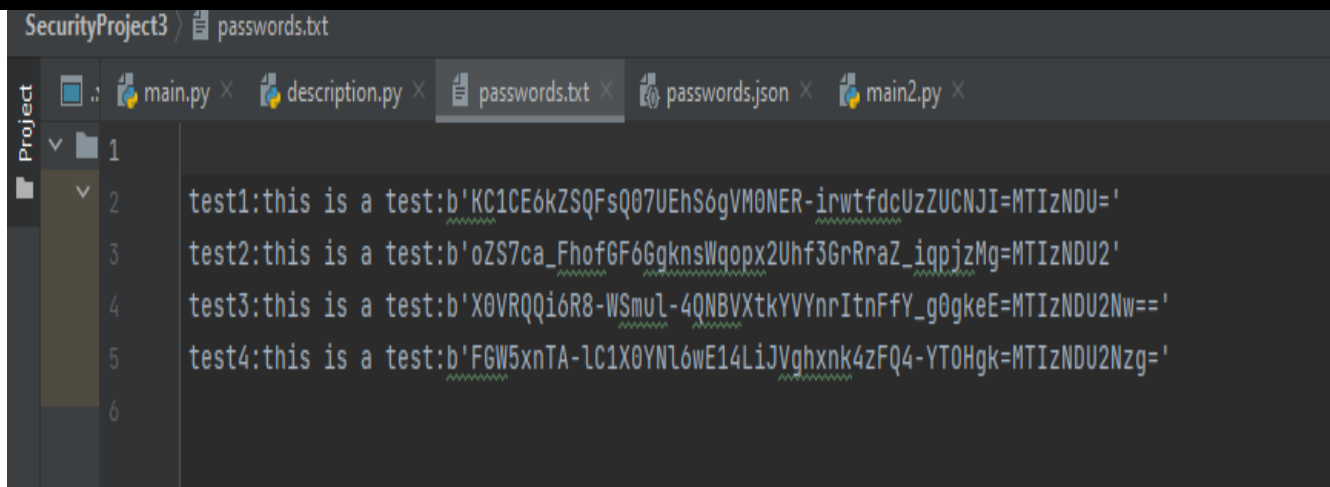
لازم به ذکر است که در کامند لاین دیگر سویچ های `--c` و `--key` پیاده سازی نشده اند و هر سه پارامتر به سویچ `--newpass` پاس داده میشوند.

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --newpass "test1" "this is a test" "12345"  
Enter your master password: 123
```

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --newpass "test2" "this is a test" "123456"  
Enter your master password: 1234
```

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --newpass "test3" "this is a test" "1234567"  
Enter your master password: 12345
```

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --newpass "test4" "this is a test" "12345678"  
Enter your master password: 123456
```



The screenshot shows the PyCharm IDE interface with the 'SecurityProject3' project open. The 'passwords.txt' file is selected in the editor. The file contains four lines of text, each representing a password entry with a name, a comment, and a hashed password. The lines are numbered 1 through 6 in the left margin.

```
1 test1:this is a test:b'KC1CE6kZSQFsQ07UEhS6gVM0NER-irwtfdcUzZUCNJI=MTIzNDU='  
2 test2:this is a test:b'oZS7ca_Fhof6F6GoknsWqopx2Uhf3GrRraZ_igpjzMg=MTIzNDU2'  
3 test3:this is a test:b'X0VRQQi6R8-WSmu1-4QNBVXtkYVYnrItNffY_g0gkeE=MTIzNDU2Nw=='  
4 test4:this is a test:b'FGW5xnTA-lC1X0YNl6wE14LiJVghxnk4zFQ4-YTOHgk=MTIzNDU2Nzg='  
5  
6
```

نمایش همه رمزها:

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --showpass  
Enter your master password: 123  
Invalid line:  
  
Name: test1, Comment: this is a test  
Name: test2, Comment: this is a test  
Name: test3, Comment: this is a test  
Name: test4, Comment: this is a test  
  
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>
```

نمایش یک رمز با اسم خاص:

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --sel "test1"
Enter your master password: 123
Name: test1, Comment: this is a test, Password: b'hvswRzsGU346Kxf1DDrw1m7tdt3E9zVSxZFm8AV5RZ4=MTIzNA=='

C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>
```

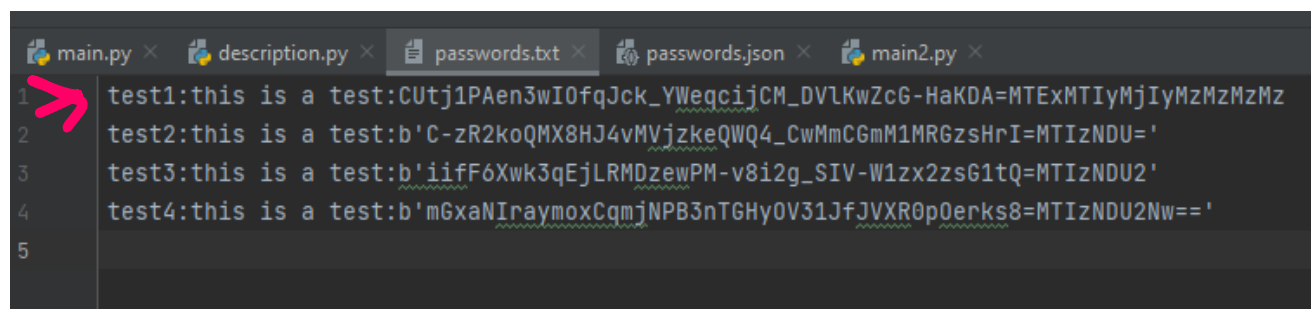
\* در رابطه با نمایش رمز بصورت رمز شده این نکته باید لحاظ شود که میدانیم که رمز باید بصورت رمز شده در فایل ذخیره شود که این امر رعایت شده است و در زمان نمایش باید رمزگشایی شود و نمایش شود و برای این امر تابع رمزگشایی هم نوشته شده اما چونکه به مشکلات زاد و حل نشدنی از جهت انکودینگ و دیکودینگ، کار با فرمت utf-8 و ... برخورد نمودم نتوانستم فرم رمزگشایی شده را نمایش دهم و همان فرم رمز شده نمایش داده میشود. تابع رمزنگاری ونحوه نمایش را هم تغییر دادم اما مشکل را حل نکرد. این کدها کامنت شده اند.

بروزرسانی یک رمز:

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --sel "test1"
Enter your master password: 123
Name: test1, Comment: this is a test, Password: b'hvswRzsGU346Kxf1DDrw1m7tdt3E9zVSxZFm8AV5RZ4=MTIzNA=='

C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --update "test1"
Enter your master password: 123
Enter the new password: 11112222333333

C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>
```



```
main.py × description.py × passwords.txt × passwords.json × main2.py ×
1 test1:this is a test:CUTj1PAen3wI0fqJck_YWeqciJCM_DVlKwZcG-HaKDA=MTExMTIyMjIyMzMzMzMz
2 test2:this is a test:b'C-zR2koQMX8HJ4vMVjzkeQWQ4_CwMmCGmM1MRGzsHrI=MTIzNDU='
3 test3:this is a test:b'iifF6Xwk3qEjLRMDzewPM-v8i2g_SIV-W1zx2zsG1tQ=MTIzNDU2'
4 test4:this is a test:b'mGxaNIraymoxCqmjNPB3nTGHYOV31JfJVXR0p0erks8=MTIzNDU2Nw=='
5
```

همانطور که میبینیم رمز رمزنگاری شده تغییر کرده است.

### حذف یک نمونه:

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --delete "test1"
Enter your master password: 123

C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>
```

```
1 test2:this is a test:b'C-zR2koQMx8HJ4vMVjzkeQWQ4_CwMmCGmM1MRGzsHrI=MTIzNDU='
2 test3:this is a test:b'iifF6Xwk3qEjLRMDzewPM-v8i2g_SIV-W1zx2zsG1tQ=MTIzNDU2'
3 test4:this is a test:b'mGxanIraymoxCgmjNPB3nTGHy0V31JfJVXR0p0erks8=MTIzNDU2Nw=='
4
```

## بخش دوم:

در این قسمت یک تابع دیگر پیاده سازی کردیم که در هر بار پیمایش کلید رمزنگاری را شماره متغیر داخل حلقه قرار داده و ورودی "0000" را با همان الگوریتم رمزنگاری قسمت قبل رمز میکند. سویچ test- برای این قسمت تعریف شده است.

```
C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>python main.py --test
Enter your master password: 123

C:\Users\MMNazari1380\PycharmProjects\SecurityProject3>
```

محتوای فایل رمزها:



```
main.py x statsgen.py x output_stats.txt x description.py x run_statsgen.py x passwords.txt x test.txt x passwords.json x main2.py x
1 b'TDzmgn9eBTDQhV6Wo3xsNysoS77hmzpdbrGUJ4HeX4I=MDAwMA== '
2 b'vWeCxbWR3unJ2o7PfJX0ixwN58tIfJQnmDuIsv81fIA=MDAwMA== '
3 b'SLqFHM1631Uc608GK2YSsUZScrRg4jyfUql2iFFs5c4=MDAwMA== '
4 b'DjaKwVgCynG9mzGL6-GND_lPqK8wuT0uH0ZEs7Er_M=MDAwMA== '
5 b'V8cx0K20FaMI_wxRvDzF65fE_Q6kLVTxShK9xSwZD0U=MDAwMA== '
6 b'aKUULaZcCNJfj0do5Q9A2dL_UFK3sX1nQUl07VENFj0=MDAwMA== '
7 b'7pUdyNoXBWMB1ttxEZRi8jKG0TGKfHBRaCuaX1D759U=MDAwMA== '
8 b'fqUijEh0LcmFFrxJi8LRsko-dIXg91pWLzpQ_603S0=MDAwMA== '
9 b'pUUpYkrb29x9Aud28Qh26KI_Q0_2Lbp-0ssPjRcCqeU=MDAwMA== '
10 b'stG-MSyS7cIBpoHZVQmaIASg1gmqMgcJs-j4zjZLA7s=MDAwMA== '
11 b'IP4ZyEBdmvXP8xYxubIRX_05CA7C1Tpm0u5z5v3ETkM=MDAwMA== '
12 b'gg-eI-x8lDDbxAUyJMRKROP-H0bh9G0SPRYueih5Wc=MDAwMA== '
13 b'-n5hcMWpWlpjREtUXj_Q43upK0tL2Ly3qVH6qxbqXI0=MDAwMA== '
14 b'04n2Lf4viZwjh-GPD2rWgF0epr1_1N0ACr-XBdS07w=MDAwMA== '
15 b'u0TysHd909B3dqmanJNzqJr0yQnWe9p03KHZL-0L4IA=MDAwMA== '
16 b'XbDZKwWohTFG9V2SN9dTmCFH6FuBJTppSDdw9XHGA4=MDAwMA== '
17 b'x6AmrhgiNLUwohh4ng4rTzh2a4TJqWiX2ZL4ew0jIKk=MDAwMA== '
18 b'6uYJVd_y7yy4_0Sn2CtvzZFq1q39unHim62MkUrUEms=MDAwMA== '
19 b'kMT_8zF4t3tk-uX_o30bcNeWYI3vCJgKn7qMxEI7fYs=MDAwMA== '
20 b'FXi5uqpw_BuG0z6FVknzD0UY0xLTkiutpo0Vut6nJU=MDAwMA== '
21 b'6BmMgERD6WNWUVVfxUqZmRoPpoP9nUfUitiKricmapc=MDAwMA== '
22 b'LYQmLvcY0Jsectow_ZZLVkic_a0m_Em04bUHCMvKJpbo=MDAwMA== '
23 b'iPLlyfs-BRLZuPAfY2-NwKd_QMmEIT39a-sTFu6N0sQ=MDAwMA== '
24 b'8aP4pEYL4UjjZAA6hragVE7n3iYFGUE8U1Fcn164yfa=MDAwMA== '
25 b'4VmJadPrdeSE-kmD6x2ECjPowkh3tPbivVujTu9Jhwk=MDAwMA== '
26 b'3J7rUr1IILFPAK6oiEYdondImo5ZicEV0bTUVs1cJu4=MDAwMA== '
27 b'tHsFyUI4XLZGDuF1p-UDoUQCgFLPoDNwN1AKs2F0_Hk=MDAwMA== '

Run | TODO | Problems | Terminal | Python Packages | Python Console | Event Log
Packages: 'options' (47 minutes ago) 7:56 Python 3.9 (SecurityProject3)
```

اما با توجه به فرمت رمزهای ذخیره شده در برنامه برای statsgen قابل ارزیابی نیستند اما برنامه اجرا شده و فایل خروجی ساخته شده است. لازمه ذکر است که فایل statsgen ارایه شده دارای مشکل بود و مشکلات آن برطرف شد و سپس برنامه تست شد.

```
C:\Users\MMWazari1380\PycharmProjects\SecurityProject3>python statsgen.py --minlength 8 --maxlength 20 --charset loweralpha,upperalpha,numeric,special --simplemask stringdigit,allspecial --output output_stats.txt test.txt
```

```
StatsGen 0.0.3  
[+] iphelix@thesprawl.org
```

```
[*] Analyzing passwords in [test.txt]  
[*] Saving advanced masks and occurrences to [output_stats.txt]  
[+] Analyzing 0% (0/10000) of passwords  
NOTE: Statistics below are relative to the number of analyzed passwords, not the total number of passwords
```

```
[*] Length:
```

```
[*] Character-set:
```

```
[*] Password complexity:  
[+] digit: min(None) max(None)  
[+] lower: min(None) max(None)  
[+] upper: min(None) max(None)  
[+] special: min(None) max(None)
```

```
[*] Simple Masks:
```

```
[*] Advanced Masks:
```

```
C:\Users\MMWazari1380\PycharmProjects\SecurityProject3>
```