

گزارش پروژه سوم – امنیت اطلاعات

- بخش اول : پیاده سازی یک ابزار مدیریت گذرواژه و ساخت گذرواژه های پیچیده

در این بخش از برنامه ما قصد داریم کلید و اطلاعات و نام یک گذرواژه را گرفته، با استفاده از کلید آن را رمز کنیم و ذخیره کنیم. برای این منظور دستور `newpass` بصورت زیر پیاده سازی شد:

```
if args.newpass:
    name, comment, key = args.newpass
    if os.path.exists(passwords_file):
        decrypt_passwords_file(key)
    enc = save_password(name, comment, key)
    print('New Password added!')
    print(f'Encrypted Password : {enc}')
    encrypt_passwords_file()
```

ابتدا چک میکنیم اگر فایل رمزها وجود دارد با استفاده از کلید دریافتی آن را رمزگشایی میکنیم. اگر هم وجود نداشت با فراخوانی تابع `save_passwords` این فایل ایجاد میشود. در این تابع که بصورت زیر است:

```
def save_password(name, comment, key):
    generated_password = generate_password(name, comment, key)
    encrypted_password = encrypt(generated_password, key)
    entry = {"name": name, "password": encrypted_password, "comment":
comment, "key": key}
    with open(passwords_file, "a") as file:
        file.write(json.dumps(entry) + "\n")

    return encrypted_password
```

ابتدا یک فایل متنی با استفاده از تابع `generate_passwords` میسازیم:

```
def generate_password(name, comment, key):
    # Generating a random salt
    salt = ''.join(random.choices(string.ascii_letters + string.digits,
k=16))

    # Combining name, comment, key, and salt for password generation
    password_input = f"{name}_{comment}_{key}_{salt}"
    generated_password = hashlib.sha256(password_input.encode()).hexdigest()

    return generated_password
```

که همانطور که مشاهده میشود با انتخاب رندوم یک سالت و ترکیب بخش های مختلف رمز و در نهایت هش کردن آن متن رمز را میسازد و برمیگرداند.

در گام بعدی این متن و کلید به تابع `encrypt` فرستاده میشود:

```
def encrypt(text, key):
    key = derive_key(key)
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(text.encode())
    return base64.b64encode(cipher.nonce + tag + ciphertext).decode()
```

در این تابع ابتدا از کلید ورودی یک کلید که مناسب رمزنگاری AES باشد با استفاده از تابع `derive_key` ایجاد کرده و با استفاده از الگوریتم AES آن را رمز میکنیم و برمیگردانیم. تابع مشتق گیرنده کلید نیز به صورت زیر است:

```
def derive_key(password, salt=b'some_salt', iterations=100000,
key_length=32):
    key = hashlib.pbkdf2_hmac('sha256', password.encode(), salt, iterations,
key_length)
    return key
```

در نهایت `save_passwords` رمز را برگردانده و به کاربر نمایش داده میشود، سپس فایل رمزها رمز و ذخیره میشود. خروجی این بخش بصورت زیر است:

```
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --newpass "git" "pass for my git" "1234"
New Password added!
Encrypted Password : 5+VPYbZ70oAviQndz1PVa3+eIPs6+HaLJBnfPZw9g/XN1Xqme1ou6h5YshQ0peH7/HusF3CHDIRQ55TE7m4DY6fAtmK8U8UdkS5PRer1uHefyL44a5xb+Nd+aZZ70oHf
```

در بخش بعدی میخواهیم اطلاعات یک رمز مشخص را ببینیم. برای این منظور دستور `sel` پیاده سازی شده است:

```
elif args.sel:
    name, key = args.sel
    decrypt_passwords_file(key)
    print('Related data: ')
    select_password(name)
    encrypt_passwords_file()
```

در این بخش پس از دریافت نام رمز و کلید اصلی با فراخوانی تابع `select_password` پس از رمزگشایی اطلاعات برگردانده شده سپس فایل مجدداً رمز نگاری میشود.

```
def select_password(name):
    with open(passwords_file, "r") as file:
        for line in file:
            entry = json.loads(line)
            if entry["name"] == name:
                print(f"Password: {entry['password']}, Comment: {entry['comment']}")
                break
```

در این تابع به سادگی پس از لود کردن داده های جیسون اطلاعات مربوط به رمز مدنظر برگردانده میشود.

خروجی این بخش بصورت زیر است:

```
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --sel "git" "1234"
Related data:
Password: Tz8T2R0vdgNRzXRrWgquYgtVmz6TEU0l0dZZQgt/moq6TZRdn6WuQipSV4TZKDRHufX0H6B0+afbisb56vAm40cs+rbvFeX/kVY05gWKQaqw9JQ3UERM5pfjiubLZzD, Comment: pass for my git
```

در بخش بعدی می‌خواهیم رمز را اپدیت کنیم به این منظور دستور `update` تعریف شده است:

```
elif args.update:
    name, key = args.update
    decrypt_passwords_file(key)
    update_password(name)
    encrypt_passwords_file()
```

در این دستور پس از رمز گشایی فایل اسم آن را به تابع `update_passwords` برمی‌گردانیم.

```
def update_password(name):
    global key
    with open(passwords_file, "r") as file:
        entries = [json.loads(line) for line in file]

    updated_entries = []

    for entry in entries:
        if entry["name"] == name:
            print(f'Previous Password: {entry["password"]}')
            # Generate a new password for the existing entry
            new_generated_password = generate_password(entry["name"],
            entry["comment"], entry["key"])

            # Update the password field in the entry
            entry["password"] = encrypt(new_generated_password, entry["key"])

            print(f'New Password: {entry["password"]}')

    updated_entries.append(entry)

    with open(passwords_file, "w") as file:
        for entry in updated_entries:
            file.write(json.dumps(entry) + "\n")

    print(f"Password for '{name}' updated!")
```

در این بخش پس از شناسایی رمز خواسته شده به شیوه تولید رمز اولیه رمز جدیدی برای آن ساخته میشود که همانطور که قبلاً گفتیم بعلت اینکه بر اساس سالت رندوم و عملکرد هش در گام اول و ساختن کلید مشتق برای AES که با سالت رندوم است جلو میرود هرگز تکراری نخواهد بود.

خروجی این بخش:

```
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --update "git" "1234"
Previous Password: Tz8T2R0vdgNRzXRrWgquYgtVmz6TEU0L0dZZQgt/moq6TZRdn6WuQipSV4TZKDRHufX0H6B0+afb1sb56vAm40cs+rwbvFeX/kVY05gWKQaqw9JQ3UERM5pfjiubLZzD
New Password: sz4Ksai+9e1fppDn0VAVaFlhZtPfBJzk6jUpBTvtvKMeVFWwLJ3baLLPs7Sn+oG077JYct9y/2t/RRUzhAU2FoSXM2xVGpCtMCUx0YZbEaXRZXGbWqxKoIQ7ggyWEgeu
Password for 'git' updated!
```

در بخش بعدی ما قصد حذف کردن یک رمز را داریم برای این منظور دستور delete پیاده سازی شده است:

```
elif args.delete:
    name, key = args.delete
    decrypt_passwords_file(key)
    delete_password(name)
    encrypt_passwords_file()
    print('Deleted Successfully!')
```

پس از رمزگشایی تابع delete_password فراخوانی شده و پس از حذف فایل مجددا رمزنگاری میشود.

```
def delete_password(name):
    with open(passwords_file, "r") as file:
        entries = [json.loads(line) for line in file]
    with open(passwords_file, "w") as file:
        for entry in entries:
            if entry["name"] != name:
                file.write(json.dumps(entry) + "\n")
```

در این تابع پس از شناسایی رمز خواسته شده آن را حذف میکنیم. خروجی این بخش:

```
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --showpass "1234"
Showing Passwords..
Name: git, Password: sz4Ksai+9e1fppDn0VAVaFlhZtPfBJzk6jUpBTvtvKMeVFWwLJ3baLLPs7Sn+oG077JYct9y/2t/RRUzhAU2FoSXM2xVGpCtMCUx0YZbEaXRZXGbWqxKoIQ7ggyWEgeu, Comment: pass for my git
Name: web, Password: v35xKKmX0afHhJrxilHSko8NsMtNRDID/F68DBNcjMNnzcYcLJcqr58T4FS0Ga3urFWFtTnL4gxIfefeGerxHkz3BNA6taHQ0Q/vxScydkLTge30cmk4IQp2Zs/HzTu7, Comment: pass for my website
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --delete "git" "1234"
Deleted Successfully!
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project3> python passmanager.py --showpass "1234"
Showing Passwords..
Name: web, Password: v35xKKmX0afHhJrxilHSko8NsMtNRDID/F68DBNcjMNnzcYcLJcqr58T4FS0Ga3urFWFtTnL4gxIfefeGerxHkz3BNA6taHQ0Q/vxScydkLTge30cmk4IQp2Zs/HzTu7, Comment: pass for my website
```

لازم به ذکر است که الگوریتم رمزنگاری XOR برای فایل در نظر گرفته شده:

```
# Encrypt passwords.txt using XOR encryption
def encrypt_passwords_file():
    with open(passwords_file, "r") as file:
        plaintext = file.read()
    encrypted_data = xor_encrypt(plaintext, key)
    with open(passwords_file, "w") as file:
        file.write(encrypted_data)
```

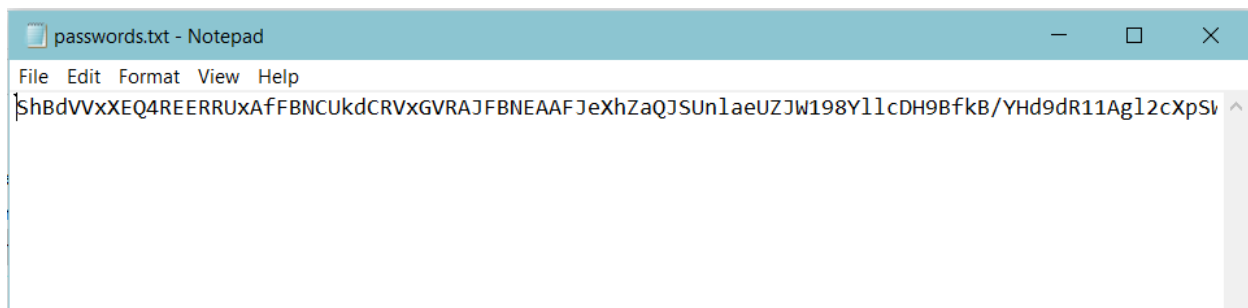
```
def xor_encrypt(data, key):
    # Convert data and key to bytes
    data_bytes = data.encode()
    key_bytes = key.encode()

    # Repeat the key to match the length of data
    repeated_key = key_bytes * (len(data_bytes) // len(key_bytes)) +
    key_bytes[:len(data_bytes) % len(key_bytes)]

    # Perform XOR operation
    encrypted_bytes = bytes([data_byte ^ key_byte for data_byte, key_byte in
    zip(data_bytes, repeated_key)])

    # Encode the result as base64 for storage
    return base64.b64encode(encrypted_bytes).decode()
```

که با استفاده از کلید داده های فایل رمزها را رمزنگاری میکنید و خروجی بصورت زیر است:



برای رمزگشایی هم:

```
# Decrypt passwords.txt using XOR decryption
def decrypt_passwords_file(key):
    with open(passwords_file, "r") as file:
        encrypted_data = file.read()
        decrypted_data = xor_decrypt(encrypted_data, key)

    with open(passwords_file, "w") as file:
        file.write(decrypted_data)
```

که از تابع زیر استفاده میکند.

```
def xor_decrypt(data, key):
    encrypted_bytes = base64.b64decode(data)
    key_bytes = key.encode()

    # Repeat the key to match the length of data
    repeated_key = key_bytes * (len(encrypted_bytes) // len(key_bytes)) +
    key_bytes[:len(encrypted_bytes) % len(key_bytes)]

    # Perform XOR operation
    decrypted_bytes = bytes(
        [encrypted_byte ^ key_byte for encrypted_byte, key_byte in
        zip(encrypted_bytes, repeated_key)])
```

```
# Decode the result to get the plaintext
decrypted_data = decrypted_bytes.decode()

return decrypted_data
```

بخش دوم - استفاده از ابزار statsgen برای تحلیل گذرواژه های تولید شده توسط ابزار پیاده سازی شده

توسط ماژول generator.py رمزهای خواسته شده را تولید و در فایل تست ذخیره میکنیم:

```
# generator.py
import string
import random
from passmanager import *

def generate_passwords():
    # Set the common parameters
    name = "generated"
    comment = "auto-generated password"
    key = "0000"

    # Create a list to store passwords
    passwords = []

    # Generate 1000 passwords
    for i in range(0, 1000):
        name = f'generated-{i}'
        comment = f'auto-generated password-{i}'
        generated_password = generate_password(name, comment, key)
        encrypted_password = encrypt(generated_password, key)
        passwords.append(encrypted_password)
        print(f'Password{i} generated!')
    print('Generation is Done!')

    # Write passwords to the test.txt file
    with open("test.txt", "w") as file:
        for password in passwords:
            print(f'writing password: {password}')
            file.write(f"{password}\n")
    print('Done')

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

فایل تست بصورت زیر است:

```
test.txt - Notepad
File Edit Format View Help
p59rjgE/Mqhpayw0FYAQVhntQ46kZDMY6fzr7mU1IXCZE01z+d1i0cVuBec+PAesazwuSPTDbto1qDxAhRl/cyHqKua9YHamePvJk01jltz2zuqubVKmt/BHX4bI
kekvt30vTZp2+xcvMEkxwlnSND6amUQhH5LaqLTIAU92/0zVZ5+MrB8qM9nN6qHqCJQFR/bnRmf3q4FD7IRZL/capQhJc7wPvcJgmUgGppNI89vAhYzW0hmrMIEJ1neA
2/IEKqGqgr5EqE1Z5ARr1CR6/G6zmZr1Uy6H3/PP9h6hlyc9K5CE5IA7Kdx6/FAS8syVKZE3W11Wt3Kt3q/HeAvavrWVhYqCqzRqZ1JCUUnqo/H3T7mPGRtUKB/+BR
HfGfDh0IhB1LdR3JQfCO/ggn9Sj807u0d0A7P3ksQJwHfba0eHPBuz2TCmMCLJ1VIGctdMhPw313CSXfucVhVAMt6v50Qf2ncR1p7D0XZGR+Uc7ptae8EHsmCuZ
Juu6hu0i5sduLufK0DSCJ1Q+ZV0annhRdxw5t6a8T0Wk1DV3yEG80gkvkwe6GqV7eY59TKYngZnyhyHaeZPKfSogf3CX1o2FecDQ71Tn1p1TNPRLVafmsnJ3t1RiDYVvq
nk3p0Ung779y/hXW1PyhmNZ1f5i0YMS8rTy78y0cQTZk8q76hNzE83Z91tkQpndTQO/ARQ043Jog/UCaW3h3ZtpuZUjh+11vdZ58pWQAO1lDbWxYqVz7pJNCK6x
i7JpCu81p3AAbnrssqJmYi0tawMtx90NmhHC9H2SLCTPRJhtVmbCDm76wn9AFzLYVY1eozMaZ3Q5kchZqwpDk6/H9XX1Net6kf7GCAU1tetw+dI1s1G0R3z3XU1Luki
D6Zf85rPB0G1FTNAxfKfuzl1eXv/GmnuIdjo9U8K1DCTPRJhtVmbCDm76wn9AFzLYVY1eozMaZ3Q5kchZqwpDk6/H9XX1Net6kf7GCAU1tetw+dI1s1G0R3z3XU1Luki
6apJ5i015s+dy3jed3rTrC92t0a7b57hYue+K8AFkhm0GKXRP00j1pGw0ZyevV1FpChamB1iCUDGSEPLH2AZXEE1JXkpROPB3mdAwblVPHt9mXfkQTHB1XoavLOPwD
VXx0ZRP+RFWYf8J6x1b50uBm1E+PgRCG1m8aorBu2h9PKtbyNt5H0jnhvKvWts800wQvkb6vW580ZrX/IJ1t4Cqugk5473C9Ns5d0yrQLqafYi56NFH6ZpY2td11
70LzYGnUvP5w8/J/GTqxcvLgT7j+n6rH58J10310AI+92132FV+kvu+OpB130zL3m1gAmOXH69hMpcnnz1UXdcjy0hVum9BCyDgrexG0C3o14BFogpF81jSwuH1RvL1
brX21whg7R8L06BexdNM3+07AnBcPsT0ZC+ANGGpG2tas12jpf5rBp1qYmBUUpYf8dpIYb19sr4H00+FkDZ5adF2181L1SgKusq6962KymGMRh850kt/I0vWqX83I
gubJc5V75Z4H/UaA7sw3KVENs11QVU00j5F36q+hR0T16A3ZdSPNM2d1msfG6acZa1CUXG+JyUHTgAGuHfPpt+sdRfF/carubqe56fCe1WgZ6gKneB85AheNjVFAFRD
ua/CsLBY3Eon1p3+A800rXK0ex1Layhgw3aoQC+s92cKfT6P1E71s17ubXbWfM525a17hKbW10KS/0xgucub0b0hM9ne0kus09c1/jy9S3RF2KX0opJ2c+Yc0y
0TUNCKK6tsic1CfFrIy0mbaeU5V63Y0cMfQTj7DfKsZVScZv6cmXhUnzV1Mq5oVnnJ1U4iF2o5FFnAvayOd1+trVvY11+orZ43ahf5QgAWXf8BecngK0KndW5W0eGf
nM3app5smurZ8RHQMSs85FAXr+HMAqhrd3t+9vKqGfjgavGMH1NntiImekan9z7RMO180531IEUse+abPhtS358HISm12LfuosqWQK056KXZ0E8Y3Qsn192KhwuqHy
scFff1uYvocyHhVhsSrg5CVBhu9g01Z89YCZqhkE9ZBu6LkLJ2m0/XjQm1BBTPlJ1e260zi5TYW8IXV1/RRvygakaJAGB6uWf5bJYEYUDK65XJEPE57+Izrpxz
uyDD/1ve3K1rZqLp5X0wB5P0H0XAZF7jCmW23m08e7KG90ehi f6Q/prbLur2+8uM81L8k08T1Yt1Lq24xc12g39j3bmo5dwfK014qY9R8WZwanik//KAE0FrYngD
8V3x11/5ZnrVzK+UJ225W8pEYVrFVnyym59vBdwYYZ/q/0s0du16aYUvSYGudKErNoLEgEM8BL0vTQmaactU7XqrdowWarkqEBU1Pm19x9t5T68Z+R9wrrD9
W0UaYUvP5CnrmqF/IXJ225780UVD5/7HT101oylvfGpWdtA+F05ax8uMkX9YTLKJ0M3XoZxmtJ5K2u7HnggaHeFlsBZZT8TJUBv1RLTMAqk20050GvnrN40MRHs7R5bYCO
Qds50V6b6NPL9Gukpnqush1ThZyBPMK9KznmIEFsTwsrPKAm+t+TKSMKMuL87VUgeLvcWLRGoXTUapdcGATMG91I5INB05YqaJE93YgqVqJX0f726nHF1u0a2FHTY
xzZkukXLT7T2AF3V+LQ60hNcdMwAhZ7BXU80qK6GfK5XyTq/Sbw0MrvrDUHJC61crrwn/6XGN1T61U0sG456sQah0wX811W1j1yAw0KvPxf9uM6e5dxF4B500W
E4uADP1kvZYVhUgVDFDof5ZMP5U2jFfP1YmmlavJyJkZnAmnmfJ0D0B5Zearj0f4ZV41e8B1T712+ooU4f5k011+KCC0Aa0fYUBC3UuX1T0423m5C7Erya
v425rQT+YVY0KKZ0K09+nw5fA590qWwH/x13b+1qEK0J3JBHvGPTEUEbkiubmNcAe70H8BeyZGmG0L5ERV34Dkaoy61f2tmvCBoaBub1f5ZnsqPb1YAX3wdy31m
+nKupEsoVQngH0Vh0ksUfZHT1HAASZ1wZkhDFKSLBLwCvZdU9MSBLV2VKA4VgdeVE1rE18P1ZnmD1AH8PmZf0D165VxxvZ+owM4e62cf/Hma5Vzh0hgn4R7Pj6uWu
1kLNBHj8jKEP51tpK55WjR8RnuCvuc5VW/YAsJ/5259XEET1ZD0Afnur6Cq3ikt51eZqvqfQh5Gk2xxh1aDEf6emoTgWv/1yN1BQn0e5b1HF7RbJ/x21UdgWdM
RNDY9Xdo1t5WGNMaLvJnab/wsr0xkA6UEx+xi/vQ5Gze552xiame9efC4GcCOBugsfW5X+6p+aeWu7NCX00vHRZyrQcP8f5IHPsTPcOPrS4C5a5CbyKUbz+18y1lgCM
za6nht1V//1fEdpZS10SAQ4shVcFTUDZPoeF9bZ0xwFV3OYx9nkiYH+VCAqFw0UwJN14uswFR5zmz1FngubYRtFwdtDjYruxU+Hge5Wz9J2MEKpgQ0p3+dxx+Q
0aPXU323E8+pUNG1Y5S2+4z1hN11xjBfQ4d9au8q/7jKDCPvcI+AGuVfVc/WZbFkDYs1S4OM6fQEIIMH93YOSd011hWz08T8u8P1Z6+adyfPvdh1w6j6d7uZK
ETL1s1arsvVJ05GCLAYTEJBEJDE1ZpLPLFjXTVDjmlktf5ET5u2uWcSt1DzuhPKQYhpxC88+1p9UyGw0kWU5Jfo1KjEudEwbmmr0/pU7sz8W10Z0d8smnKRoCED1p
GORqR1CHCSVY26/hD9R7L16n9MKMPSf1BRKc+K8RQ0vC94xtsrMUC4kfW6sFb1KNSCKf15DR1TQN285VY7TgQ0kyDu1HNVZHCW4UPyus67Cd0gXl/Yccdehqe/ab
ZQ0rUDZvesFZmZ50zXtAtA9sShKrfb5W0P55u/1AAK3T7nZf3Tj2hyB70XyAvLAW3v3xbWp3d50v6JY9Rw10avR+0j3GCGeH4P1W144cWPKWwC8CZpRyJundW6+
/vfDuaXmPvK0K6c5h5S5XVSKfba0hP53Wf1FSMBUkXh/as04wKsXvP0DYc+Ap5+H0f6C66ccZMSBLv8zJP09TnZ56U7H1z1bz0hhb/uoTHH0u138K1nG
AHXKZfB0eafN4pN1NS5R1DfZQq1Yj5Vh6H70hde2avCJ1LV5on3KfjBAWwB1QRf5P6Vf92X599n1HawKRF1wk15nh3ZguV000VhX2hZQf1ewbZ8001GXQ/1wI+a
1gqF03zpq0BmuZ2T04MEqLPxQ0JdGEs613G07W/gKZVA/ANV1HdpC2P3HT1xBKjnx0eOzBxagY0Q/d4SUMS5WkSp13MrvVYQrZszZh0FZ0uR6KvHBPCH8+jqkRhhW
BAXtFpG7K2061EBdC8B8Rgn1nvDmB45F510KK05M9VnmHf1s051RkGzcXa+a20a1392a08du82DFXoap10f20F9rnhwseo32at0uA8hG5iZrF0eoeuHmPi
```

در گام بعدی خروجی بصورت زیر است:

```
[*] Analyzing passwords in [test.txt]
[+] Analyzing 100% (1000/1000) of passwords
NOTE: Statistics below is relative to the number of analyzed passwords, not total number of passwords

[*] Length:
[+] 128: 100% (1000)

[*] Character-set:
[+] all: 98% (989)
[+] mixedalphanum: 01% (11)

[*] Password complexity:
[+] digit: min(8) max(34)
[+] lower: min(35) max(70)
[+] upper: min(34) max(70)
[+] special: min(0) max(14)

[*] Simple Masks:
[+] othermask: 100% (1000)
```

تحلیل آماری از فایل `test.txt` مطابق با خروجی اسکریپت StatsGen به صورت زیر است:

طول رمزها:

- همه رمزها در فایل `test.txt` دارای طول ۱۲۸ کاراکتر هستند.

مجموعه کاراکترها:

98٪ - از رمزها شامل تمامی کاراکترها (عددی، حروف کوچک و بزرگ، ویژه) هستند.

1٪ - از رمزها شامل ترکیبی از حروف کوچک و عدد می‌باشند.

پیچیدگی رمزها:

- تعداد اعداد در رمزها حداقل ۸ و حداکثر ۳۴ می‌باشد.
- تعداد حروف کوچک در رمزها حداقل ۳۵ و حداکثر ۷۰ می‌باشد.
- تعداد حروف بزرگ در رمزها حداقل ۳۴ و حداکثر ۷۰ می‌باشد.
- تعداد کاراکترهای ویژه در رمزها حداقل ۰ و حداکثر ۱۴ می‌باشد.

الگوهای ساده:

100٪ - از رمزها الگوهای پیچیده‌ای دارند که با `othermask` نمایش داده می‌شوند. این الگوها از ترکیب‌های مختلف از اعداد، حروف کوچک و بزرگ، و کاراکترهای ویژه تشکیل شده‌اند.

نکته:

این تحلیل بر اساس رمزهای موجود در فایل `test.txt` انجام شده است. اگر تعداد کل رمزها در فایل بیشتر از ۱۰۰۰ می‌باشد، آمارها نسبت به تعداد آن رمزها محاسبه شده‌اند.

بخش امتیازی – رمز با اندازه متفاوت

با تغییر کد به شکل زیر در فایل `passmanagerPlus.py` رمزها با طول متفاوت تولید میشوند:

```
def generate_password(name, comment, key):
    # Generating a random salt
    salt = ''.join(random.choices(string.ascii_letters + string.digits,
    k=16))

    # Generating a random password length between 8 and 20 characters
    password_length = random.randint(8, 20)

    # Generating a random password with the specified length
    password_input = f"{name}_{comment}_{key}_{salt}"
    generated_password =
    hashlib.sha256(password_input.encode()).hexdigest()[ :password_length]

    return generated_password
```

حالا ۱۰۰۰ رمز را توسط فایل `generatorPlus.py` تولید میکنیم و به فایل `stategen` میدهیم:

```
[*] Analyzing passwords in [testPlus.txt]
[+] Analyzing 100% (1000/1000) of passwords
    NOTE: Statistics below is relative to the number of analyzed passwords, not total number of passwords

[*] Length:
[+]          56: 24% (244)
[+]          64: 23% (233)
[+]          68: 22% (228)
[+]          60: 21% (217)
[+]          72: 07% (78)

[*] Character-set:
[+]          all: 96% (960)
[+]          mixedalphanum: 04% (40)

[*] Password complexity:
[+]          digit: min(1) max(21)
[+]          lower: min(13) max(40)
[+]          upper: min(9) max(39)
[+]          special: min(0) max(9)

[*] Simple Masks:
[+]          othermask: 100% (1000)
```

همانطور که مشاهده میشود رمزها در طولهای مختلف تولید شده اند و پیچیدگی بالاتر رفته است.

بخش امتیازی – طراحی رابط کاربری

با استفاده از کد بخش قبلی (بخش passmanagerGui.py) طراحی شد. نمونه:

