

گزارش پروژه اول امنیت اطلاعات

فاز ۱)

در این بخش برای پیاده سازی خواسته های مسئله به صورت زیر شروع میکنیم:

```
def main():
    password = input("Enter the password: ")

    try:
        mode = int(input("Enter the mode (1 for standard, 2 for first char, 3 for partial): "))

        if mode == 1:
            standard_mode(password)
        elif mode == 2:
            first_char_mode(password)
        elif mode == 3:
            partial_mode(password)
        else:
            print("Invalid mode. Please enter 1, 2, or 3.")
    except ValueError:
        print("Invalid input. Please enter a valid integer for the mode.")
```

ابتدا از کاربر رمز مدنظر گرفته شده سپس با توجه به مود انتخابی او کار را ادامه میدهیم:

در صورتی که مود انتخابی ۱/ استاندارد باشد:

```
def standard_mode(password):
    char_set = int(input("Enter the character set (1 for numbers, 2 for lowercase letters, 3 for both): "))

    possibilities, calculation_time = calculate_possibilities(password, char_set)

    if possibilities is not None:
        print(f"Number of password possibilities: {possibilities}")
        print(f"Estimated calculation time: {calculation_time:.10f} seconds")
```

از کاربر خواسته میشود تا فضای رمز را انتخاب کند سپس با استفاده از تابع دیگری احتمالات و رمز گشایی آغاز و محاسبه میشود:

```
def calculate_possibilities(password, char_set):
    length = len(password)

    if char_set == 1:
        characters = string.digits # Only numbers
    elif char_set == 2:
        characters = string.ascii_lowercase # Only lowercase letters
    elif char_set == 3:
        characters = string.digits + string.ascii_lowercase # Numbers and
lowercase letters
    else:
        print("Invalid character set. Please enter 1, 2, or 3.")
        return None

    start_time = time.time()

    found = False
    num_iterations = 0

    while not found:
        num_iterations += 1
        random_guess = ''.join(random.choice(characters) for _ in
range(length))
        print(f"Trying password: {random_guess}")

        if random_guess == password:
            found = True
            print(f"Found the password: {password}")

    end_time = time.time()

    calculation_time = (end_time - start_time)

    password_count = len(characters) ** length

    return password_count, calculation_time
```

در این بخش با توجه به فضای رمز فرایند رمز گشایی انجام شده و زمان آن از قبل از شروع تا بعد از پیدا کردن رمز اندازه گیری شده و به کاربر نمایش داده میشود:

```
Trying password: 8308
Trying password: k6g4
Trying password: kgv8
Trying password: yize
Trying password: 12sa
Found the password: 12sa
Number of password possibilities: 1679616
Estimated calculation time: 60.3790404797 seconds
```

در صورت انتخاب مود دوم از تابع زیر استفاده خواهد شد:

```
def first_char_mode(password):
    length = len(password)
    first_char = password[0]
    char_set = int(input("Enter the character set (1 for numbers, 2 for
lowercase letters, 3 for both): "))

    possibilities, calculation_time =
calculate_possibilities_first_char(length, first_char, char_set, password)

    if possibilities is not None:
        print(f"Number of password possibilities: {possibilities}")
        print(f"Estimated calculation time: {calculation_time:.10f} seconds")
```

که برای انجام محاسبات از کد زیر استفاده کرده و همانند مرحله قبلی اما با داشتن اولین کاراکتر جلو می‌رود:

```
def calculate_possibilities_first_char(length, first_char, char_set,
password):
    if char_set == 1:
        characters = string.digits # Only numbers
    elif char_set == 2:
        characters = string.ascii_lowercase # Only lowercase letters
    elif char_set == 3:
        characters = string.digits + string.ascii_lowercase # Numbers and
lowercase letters
    else:
        print("Invalid character set. Please enter 1, 2, or 3.")
        return None

    start_time = time.time()

    found = False
    num_iterations = 0

    while not found:
        num_iterations += 1
        random_password = first_char + ''.join(random.choice(characters) for
_ in range(length - 1))
        print(f"Trying password: {random_password}")

        if random_password == password:
            found = True
            print(f"Found the password: {password}")

    end_time = time.time()

    calculation_time = (end_time - start_time)

    password_count = len(characters) ** (length - 1)

    return password_count, calculation_time
```

```

Trying password: ss3
Trying password: s2t
Trying password: s8x
Trying password: s2m
Trying password: s5a
Found the password: s5a
Number of password possibilities: 1296
Estimated calculation time: 0.0241315365 seconds

```

در صورت انتخاب مود سوم علاوه بر طول رشته اصلی بخشی از آن هم نمایش داده خواهد شد:

```

def partial_mode(password):
    length = len(password)
    k = int(input("Enter the value of k (number of characters to reveal): "))
    partial = password[:k]
    char_set = int(input("Enter the character set (1 for numbers, 2 for lowercase letters, 3 for both): "))

    possibilities, calculation_time = calculate_possibilities_partial(length, k, partial, char_set, password)

    if possibilities is not None:
        print(f"Number of password possibilities: {possibilities}")
        print(f"Estimated calculation time: {calculation_time:.10f} seconds")

```

برای انجام محاسبات از تابع زیر استفاده خواهد شد:

```

def calculate_possibilities_partial(length, k, partial, char_set, password):
    if char_set == 1:
        characters = string.digits # Only numbers
    elif char_set == 2:
        characters = string.ascii_lowercase # Only lowercase letters
    elif char_set == 3:
        characters = string.digits + string.ascii_lowercase # Numbers and lowercase letters
    else:
        print("Invalid character set. Please enter 1, 2, or 3.")
        return None

    start_time = time.time()

    found = False
    num_iterations = 0

    while not found:
        num_iterations += 1
        random_password = partial + ''.join(random.choice(characters) for _
in range(length - k))

```

```

    print(f"Trying password: {random_password}")

    if random_password == password:
        found = True
        print(f"Found the password: {password}")

    end_time = time.time()

    calculation_time = (end_time - start_time)

    password_count = len(characters) ** (length - k)

    return password_count, calculation_time

```

خروجی:

```

Enter the password: d54s
Enter the mode (1 for standard, 2 for first char, 3 for partial): 3
Enter the value of k (number of characters to reveal): 2
Enter the character set (1 for numbers, 2 for lowercase letters, 3 for both): 3
Trying password: d5sm
Trying password: d56p
Trying password: d54s
Found the password: d54s
Number of password possibilities: 1296
Estimated calculation time: 0.0072724819 seconds

```

فاز ۲)

ابتدا کتابخانه های لازم را اضافه میکنیم. کتابخانه فانت یک پیاده سازی متقارن رمزنگاری است.

```
from cryptography.fernet import Fernet
```

فانکشن زیر یک کلید تولید میکند که برای رمزنگاری و رمزگشایی مورد استفاده خواهد بود:

```
def generate_key():
    key = Fernet.generate_key()
    print(f"Generated key: {key.decode()}")
    return key

def load_key():
    return open("secret.key", "rb").read()

def save_key(key):
    with open("secret.key", "wb") as key_file:
        key_file.write(key)
```

برای رمز نگاری از تابع زیر استفاده خواهد شد:

```
def encrypt(filename, key):
    fernet = Fernet(key)

    with open(filename, "rb") as file:
        file_data = file.read()

    encrypted_data = fernet.encrypt(file_data)

    encrypted_filename = f"{filename}.encrypted"
    with open(encrypted_filename, "wb") as encrypted_file:
        encrypted_file.write(encrypted_data)
```

همانطور که مشاهده میشود فایل باز شده و پس از رمز نگاری نسخه جدید ذخیره خواهد شد

```
def decrypt(filename, key):
    fernet = Fernet(key)

    with open(filename, "rb") as file:
        encrypted_data = file.read()

    decrypted_data = fernet.decrypt(encrypted_data)

    decrypted_filename = filename[:-len(".encrypted")]
    with open(decrypted_filename, "wb") as decrypted_file:
        decrypted_file.write(decrypted_data)
```

برای رمزگشایی از تابع بالا استفاده خواهد شد

برای آغاز برنامه بصورت زیر عمل خواهیم کرد:

```
def main():
    try:
        mode = int(input("Enter the mode (1 for encryption, 2 for decryption): "))

        if mode == 1: # Encryption
            key_option = int(input("Enter 1 to generate a key, or 2 to use an existing key: "))

            if key_option == 1:
                key = generate_key()
                save_key(key)
            elif key_option == 2:
                key = load_key()
            else:
                print("Invalid option for key. Please enter 1 or 2.")
                return

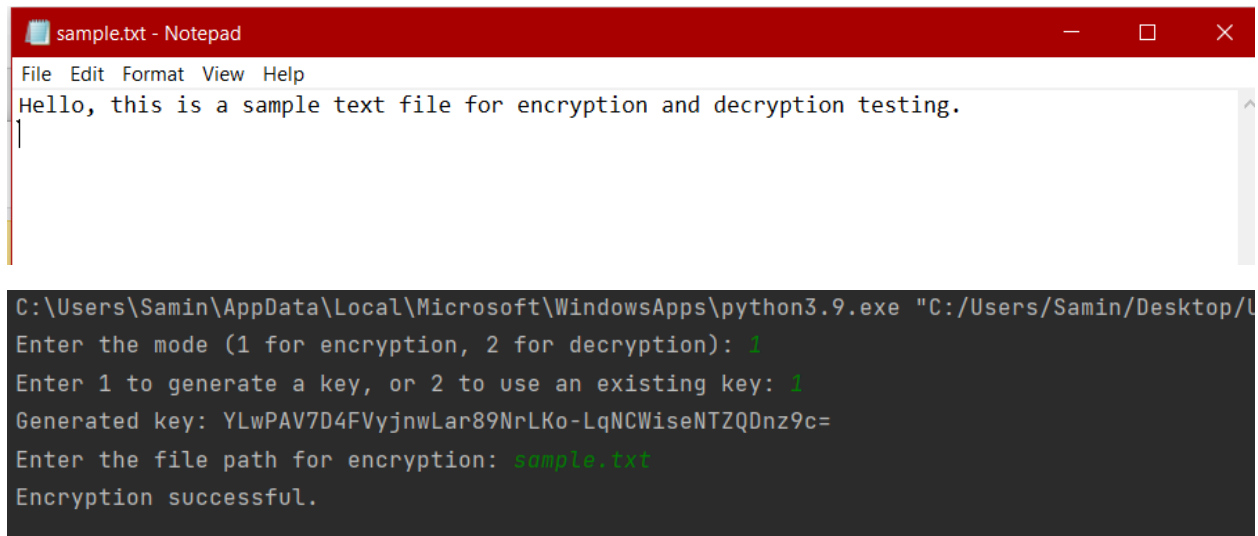
            filename = input("Enter the file path for encryption: ")
            encrypt(filename, key)
            print("Encryption successful.")

        elif mode == 2: # Decryption
            key = input("Enter the decryption key: ")
            filename = input("Enter the file path for decryption: ")
            decrypt(filename, key)
            print("Decryption successful.")

        else:
            print("Invalid mode. Please enter 1 or 2.")
    except ValueError:
        print("Invalid input. Please enter valid integers.")
```

ابتدا مسیر فایل مدنظر از کاربر گرفته شده، سپس از او میپرسیم کلید مدنظر دارد یا نه، اگر داشته باشد از کلید خودش وگرنه از کلید تولیدی برنامه استفاده خواهد شد. درصورت انتخاب رمزگشایی کلید از کاربر گرفته شده و رمز گشایی انجام میشود.

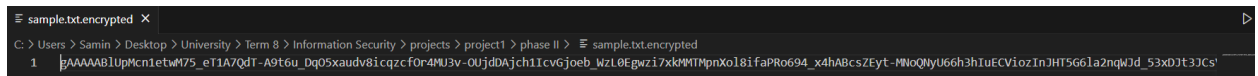
مثال) متن زیر را در یک فایل تکست ذخیره کردیم:



The image shows two windows. The top window is a Notepad application titled 'sample.txt - Notepad'. It contains the text: 'Hello, this is a sample text file for encryption and decryption testing.' The bottom window is a terminal window showing the execution of a Python script. The output is as follows:

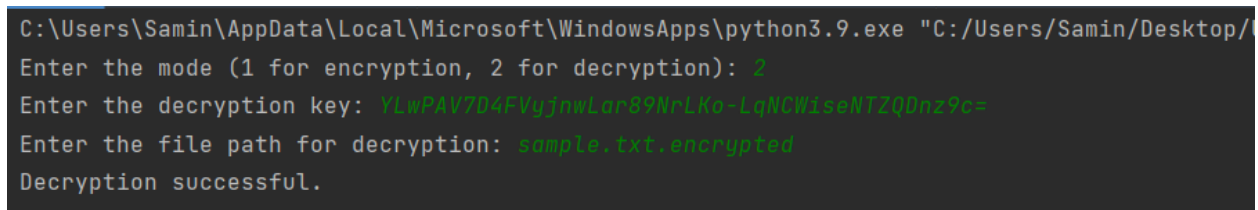
```
C:\Users\Samin\AppData\Local\Microsoft\WindowsApps\python3.9.exe "C:/Users/Samin/Desktop/U
Enter the mode (1 for encryption, 2 for decryption): 1
Enter 1 to generate a key, or 2 to use an existing key: 1
Generated key: YLwPAV7D4FVyjnwLar89NrLKo-LqNCWiseNTZQDnz9c=
Enter the file path for encryption: sample.txt
Encryption successful.
```

پس از رمز نگاری بصورت زیر درمیآید:



The image shows a file explorer window titled 'sample.txt.encrypted X'. The address bar shows the path: 'C:\Users\Samin\Desktop> University > Term 8 > Information Security > projects > project1 > phase II > sample.txt.encrypted'. The file list shows a single file named 'sample.txt.encrypted' with a size of 1 KB.

حالا درخواست رمز گشایی میدهیم:



The image shows a terminal window with the following output:

```
C:\Users\Samin\AppData\Local\Microsoft\WindowsApps\python3.9.exe "C:/Users/Samin/Desktop/U
Enter the mode (1 for encryption, 2 for decryption): 2
Enter the decryption key: YLwPAV7D4FVyjnwLar89NrLKo-LqNCWiseNTZQDnz9c=
Enter the file path for decryption: sample.txt.encrypted
Decryption successful.
```



The image shows a Notepad application titled 'sample.txt_decrypted.txt - Notepad'. It contains the text: 'Hello, this is a sample text file for encryption and decryption testing.'

فایل رمز گشایی شده بصورت بالا خواهد بود.