МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Захист програмного забезпечення

Лабораторна робота 3

Використання функцій криптографічного інтерфейсу Windows для захисту інформації

Мета роботи: оволодіти навиками криптографічного захисту інформації.

Перевірив:

_____

Виконав:
студент III курсу
групи ФБ-01
Сахній Н.Р.

Київ 2023

Варіант №11:

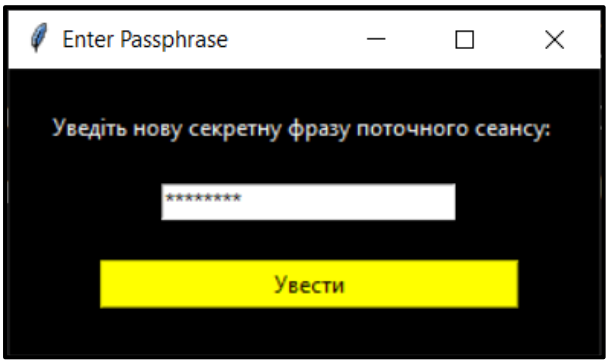
№	Тип симетричного шифрування	Використовуваний режим шифрування	Додавання до ключа випадкового значення	Використовуваний алгоритм хешування
11	Блоковий	Зворотній зв'язок по шифротексту	Так	MD4

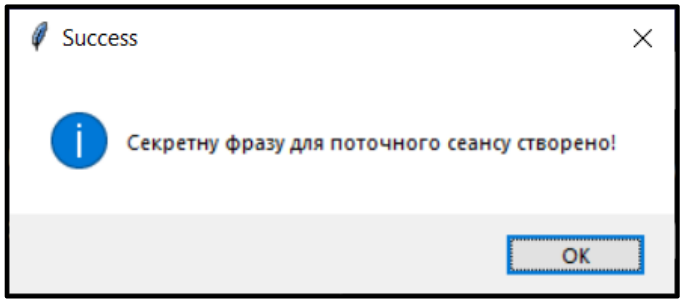
Завдання:

У програму, розроблену при виконанні лабораторних робіт №1 і №2, додамо засоби захисту від несанкціонованого доступу до файлу з обліковими даними зареєстрованих користувачів:

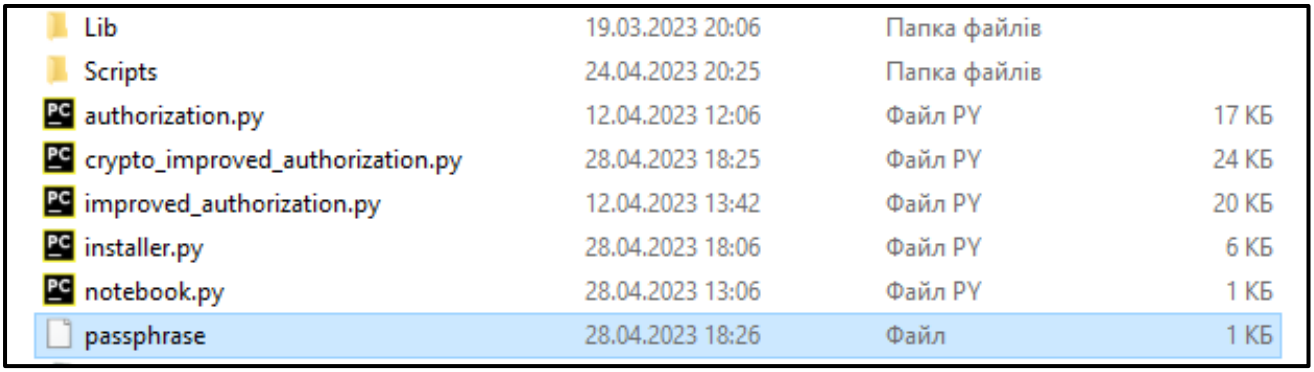
– Файл з обліковими записами зашифровуватиметься за допомогою функцій CryptoAPI з використанням сеансового ключа, генерованого на основі введеної адміністратором правильної фрази ↓

- Перший запуск програми після встановлення

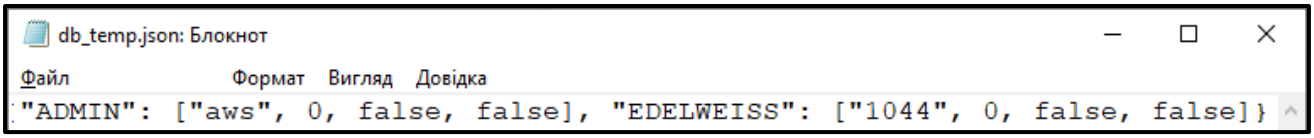




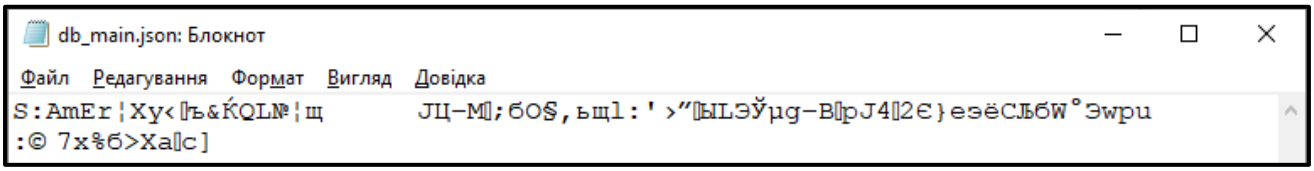

```
/* І одразу після уведення фрази було створено файл "passphrase" */
```




```
/* Під час роботи з програмою буде створено файл "db_temp.json" */
```

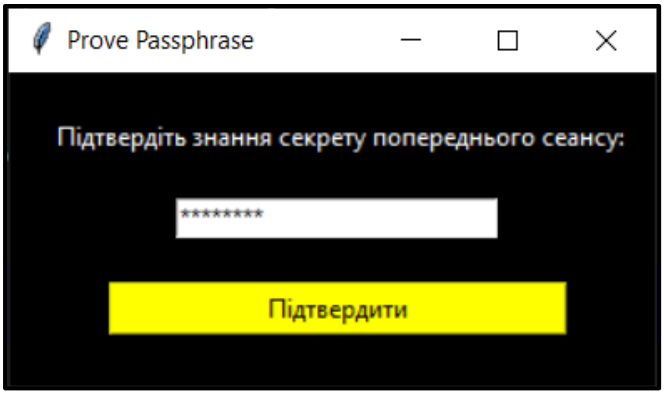


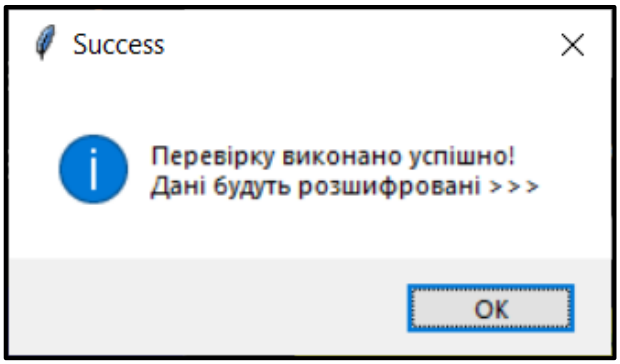
/* По завершенню роботи, дані шифруються у файл “ db_main.json” */

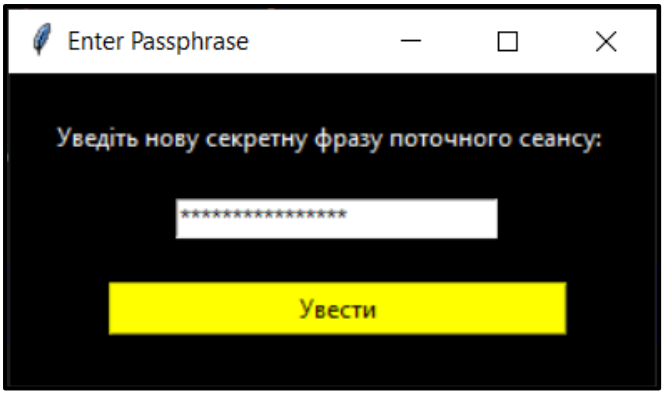


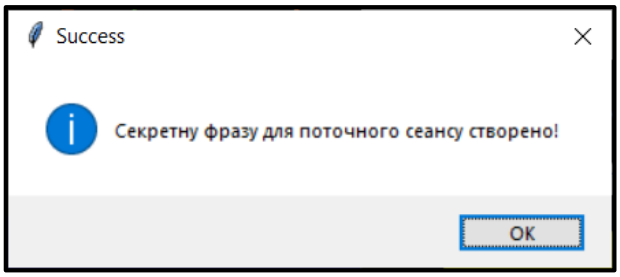
– Під час запуску програми, файл з обліковими записами розшифровуватиметься в тимчасовий файл, який після завершення роботи програми буде знову зашифрований для обліку можливих змін в облікових записах користувачів («старий» вміст файлу із обліковими записами, при цьому, стиратиметься) ↓

- Усі наступні запуски програми на виконання

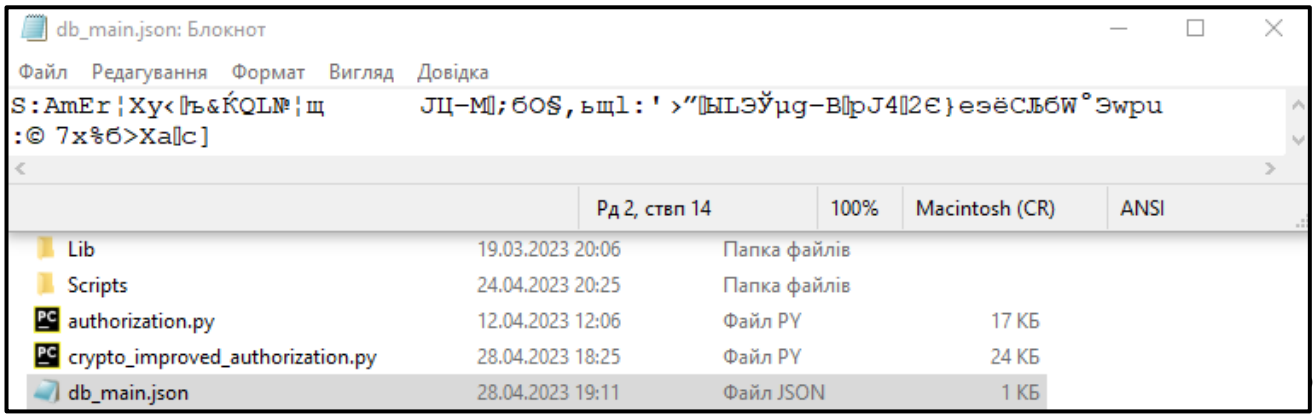


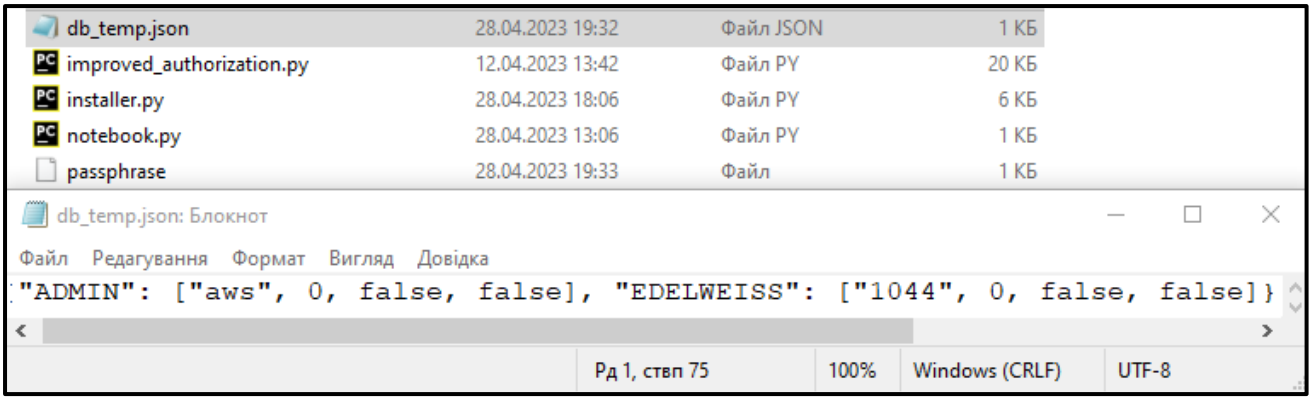




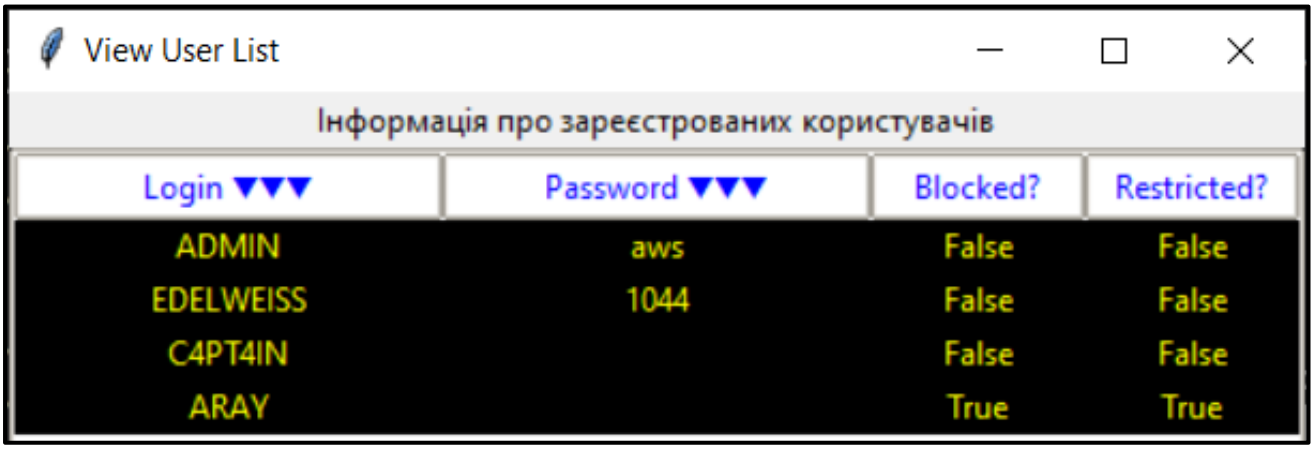


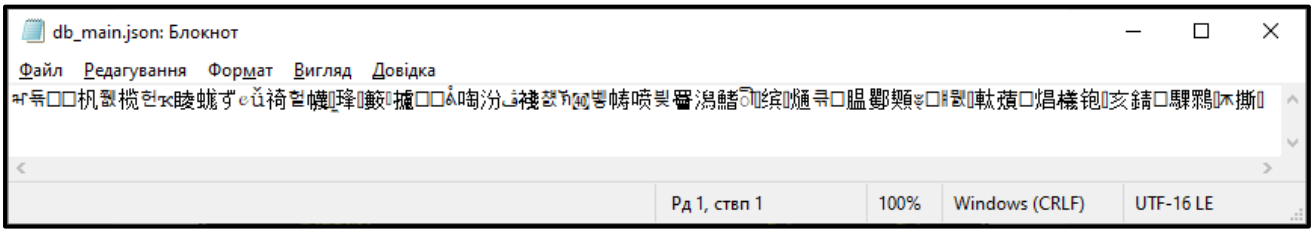
/* Під час роботи із програмою, у каталозі проекту будуть наявні обидва файли, тобто основний “db_main.json” із зашифрованими даними та тимчасовий “db_temp.json” із розшифрованими записами */





/* Отже, взаємодіючи із програмою, користувач використовує тимчасовий файл, а по завершенню всі дані про облікові записи будуть зашифровані із використанням секретної фрази поточного сеансу */





***** Для реалізації цих завдань необхідно було дописати декілька нових функцій, що будуть зашифровувати основний файл бази даних облікових записів користувачів за допомогою відповідних алгоритмів шифрування та хешування, що були отримані по варіанту до програмного коду із попередніх лабораторних робіт:

```
#########################################################################################################


from Crypto.Hash import MD4
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
```

```
def save_passphrase(root, secret):
    with open(passphrase, "wb") as fp:
        iv, md4_hash = get_random_bytes(16), MD4.new(bytes(f"{secret}", "utf-8")).digest()
        fp.write(iv + md4_hash)
    messagebox.showinfo("Success", "Секретну фразу для поточного сеансу створено!")
    register(root)
```

```
def scan_passphrase():
    with open(passphrase, "rb") as fp:
        iv, md4_hash = fp.read(16), fp.read(16)
    return iv, md4_hash


def compare_passphrase(root, secret):
    new_pass = MD4.new(bytes(f"{secret}", "utf-8")).digest()
```

```
    iv, old_pass = scan_passphrase()

    if new_pass == old_pass:
        messagebox.showinfo("Success", "Перевірку виконано успішно!\n"
                                       "Дані будуть розшифровані >>>")
        if os.path.isfile(db_main):
```

```
            decrypt_database()

        if os.path.isfile(db_temp):
```

```
            global user_db
            user_db = load_database()
        else:
            dump_database()

        enter_or_prove(root, "create")
    else:
        messagebox.showerror("Error", "Неправильний сеансовий ключ!")
```

```
def encrypt_database():
    with open(db_main, "wb") as fp:
        plaintext = open(db_temp, "r").read()
        iv, key = scan_passphrase()

        cipher = AES.new(key, AES.MODE_CFB, iv)

        fp.write(cipher.encrypt(pad(plaintext.encode("utf-8"), AES.block_size)))
        os.remove(db_temp)
```

```python
def decrypt_database():
    with open(db_temp, "wb") as fp:
        encrypted = open(db_main, "rb").read()
        iv, key  = scan_passphrase()

        cipher = AES.new(key, AES.MODE_CFB, iv)

        fp.write(unpad(cipher.decrypt(encrypted), AES.block_size))
```

```python


def enter_or_prove(root, action):
    create = ["Enter", "Уведіть нову секретну фразу поточного сеансу:", "Увести"]
    prove = ["Prove", "Підтвердіть знання секрету попереднього сеансу:", "Підтвердити"]
    wordlist = create if action == "create" else prove

    create_window(root, f"310x150")
    root.title(f"{wordlist[0]} Passphrase")
    root.configure(background="black")
```

```python

    phrase_label = tk.Label(root, bg="black", fg="white", text=wordlist[1])
    phrase_label.place(x=20, y=20)
    phrase = tk.StringVar()
    phrase_entry = tk.Entry(root, width=25, textvariable=phrase, show="*")
    phrase_entry.place(x=80, y=60)

    button = tk.Button(root, bg="yellow", fg="black", text=wordlist[2], width=30, command=lambda:
                       save_passphrase(root, phrase.get()) if action == "create"
                       else compare_passphrase(root, phrase.get()))
    button.place(x=48, y=100)
    return None
```

```python


root = tk.Tk()
user_db = {"ADMIN": ["", 0, False, False]}

passphrase = "./passphrase"
db_main = "./db_main.json"
db_temp = "./db_temp.json"
```

```python

if os.path.exists(passphrase):
    enter_or_prove(root, "prove")
else:
    enter_or_prove(root, "create")

root.mainloop()
```