«Захист програмного забезпечення»

**Лабораторна робота №3**

ВИКОРИСТАННЯ ФУНЦІЙ КРИПТОГРАФІЧНОГО ІНТЕРФЕЙСУ WINDOWS ДЛЯ ЗАХИСТУ ІНФОРМАЦІЇ

ВАРІАНТ 2

Виконав:

Студент групи ФБ-01

Корабельський Тарас

**Мета роботи**: Оволодіти навиками криптографічного захисту інформації.

**Завдання:**

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

- файл з обліковими записами повинен бути зашифрований за допомогою функцій CryptoAPI з використанням сеансового ключа, генерованого на основі введеної адміністратором правильної фрази +

- при запуску програми файл з обліковими записами повинен розшифровуватися в тимчасовий файл, який після завершення роботи програми повинен бути знову зашифрований для обліку можливих змін в облікових записах користувачів («старий» вміст файлу облікових записів при цьому стирається) +

2. Варіанти використання алгоритмів шифрування і хешування при виклику функцій CryptoAPI вибираються відповідно до виданого викладачем завдання.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип симетричного шифрування | Використовуваний  режим шифрування | Додавання до ключу випадкового значення | Використовуваний алгоритм хешування |
| 2 | Потоковий | - | Так | MD2 |

**Хід Роботи:**

Для програми був обраний блоковий алгоритм шифрування «**AES**», який використовує секретний ключ.

Функція, яка отримує кодову фразу, хешує її через MD2 і зберігає у файл:

def generate\_session\_key(key):

    # Генерація ключа сеансу за допомогою хеш-алгоритму MD2

    session\_key = MD2.new(key.encode()).digest()

    return session\_key

def write\_session\_key(key):

    with open("secret", "wb") as f:

        f.write(key)

Функція шифрування AES:

# Open the input and output files and encrypt the data

def encrypt\_file(key):

    # Генерація випадкового вектора ініціалізації

    iv = os.urandom(16)

    # Створення шифру AES за допомогою кодової фрази та вектора ініціалізації

    cipher = AES.new(key, AES.MODE\_CBC, iv)

    with open(temp\_file, "rb") as f\_in, open(db\_file, "wb") as f\_temp:

        f\_temp.write(iv)

        for chunk in iter(lambda: f\_in.read(4096), b""):

            if len(chunk) % 16 != 0:

                # Додавання PKCS7, якщо розмір чанку не кратний 16 байтам

                padding\_length = 16 - len(chunk) % 16

                chunk += bytes([padding\_length] \* padding\_length)

            encrypted\_chunk = cipher.encrypt(chunk)

            f\_temp.write(encrypted\_chunk)

    os.remove(temp\_file)

Функція розшифрування AES:

# Open the temporary file and decrypt the data

def decrypt\_file(key):

    with open(db\_file, "rb") as f\_temp, open(temp\_file, "wb") as f\_out:

        iv = f\_temp.read(16)

        cipher = AES.new(key, AES.MODE\_CBC, iv)

        for chunk in iter(lambda: f\_temp.read(4096), b""):

            decrypted\_chunk = cipher.decrypt(chunk)

            if len(chunk) < 4096:

                # Видалення PKCS7,якщо чанк є останнім фрагментом

                padding\_length = decrypted\_chunk[-1]

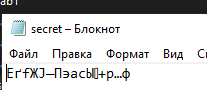
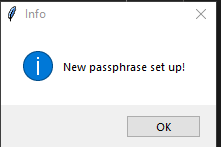
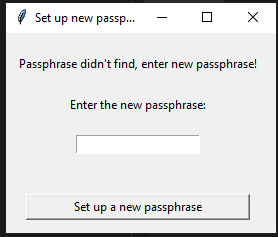
                decrypted\_chunk = decrypted\_chunk[:-padding\_length]

            f\_out.write(decrypted\_chunk)

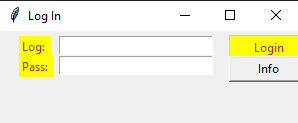
    os.remove(db\_file)

**Робота програми:**

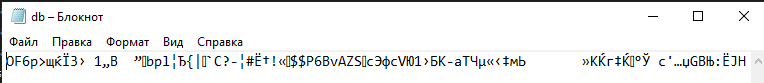
При першому запуску програми треба створити кодову фразу для сеансу:



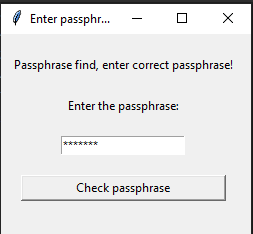
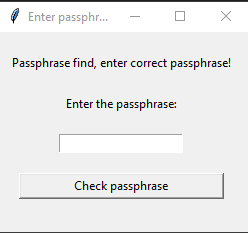
Після чого викликається початкове вікно з першої лабораторної:



Під час роботи створюється json-файл для роботи основної програми, який зашифрофується в db файл:



У наступних запусках запрошується кодова фраза з минулого запуску, з потребою ввести нову фразу для даного сеансу:



Після закінчення знову відбувається шифрування новим ключем та видалення БД.

**Висновок:** Під час виконання даної лабораторної роботи, я познайомився та навчився використовувати бібліотеки, які можуть використовуватися для криптографічного захисту інформації.