Лабораторна робота 3

Використання функцій криптографічного інтерфейсу Windows для захисту інформації

Приходько Юрій ФБ-12, варіант 13

Мета роботи

Оволодіти навиками криптографічного захисту інформації.

Зміст 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 ключа, генерованого на основі введеної адміністратором парольної фрази;
   - парольна фраза зберігається в реєстрі під час інсталяції;
   - при запуску програми файл з обліковими записами повинен розшифровуватися в тимчасовий файл, який після завершення роботи програми повинен бути знову зашифрований для фіксації можливих змін в облікових записах користувачів («старий» вміст файлу облікових записів при цьому видаляється).
2. Варіанти використання алгоритмів шифрування і хешування при виклику функцій CryptoAPI вибираються відповідно до виданого викладачем завданням.

№	Тип симетричого шифрування	Використовуваний режим шифрування	Додавання до ключу випадкового значення	Використовуваний алгоритм хешування
13	Блоковий	Електронна кодова книга	Да	MD5

Хід роботи

У ході аналізу завдання лабораторної роботи були зроблені наступні висновки.

Зважаючи на те що отримання навичок використання криптографічного інтерфейсу саме Windows не є основною метою лабораторної роботи, а також в зв'язку з технічними

аспектами виконання попередніх лабораторних робіт, а саме обраної мови розробки python, для виконання роботи був обраний більш гнучний та зручний у данному випадку інтерфейс надання криптопрімтивів, а саме модуль pycryptodome.

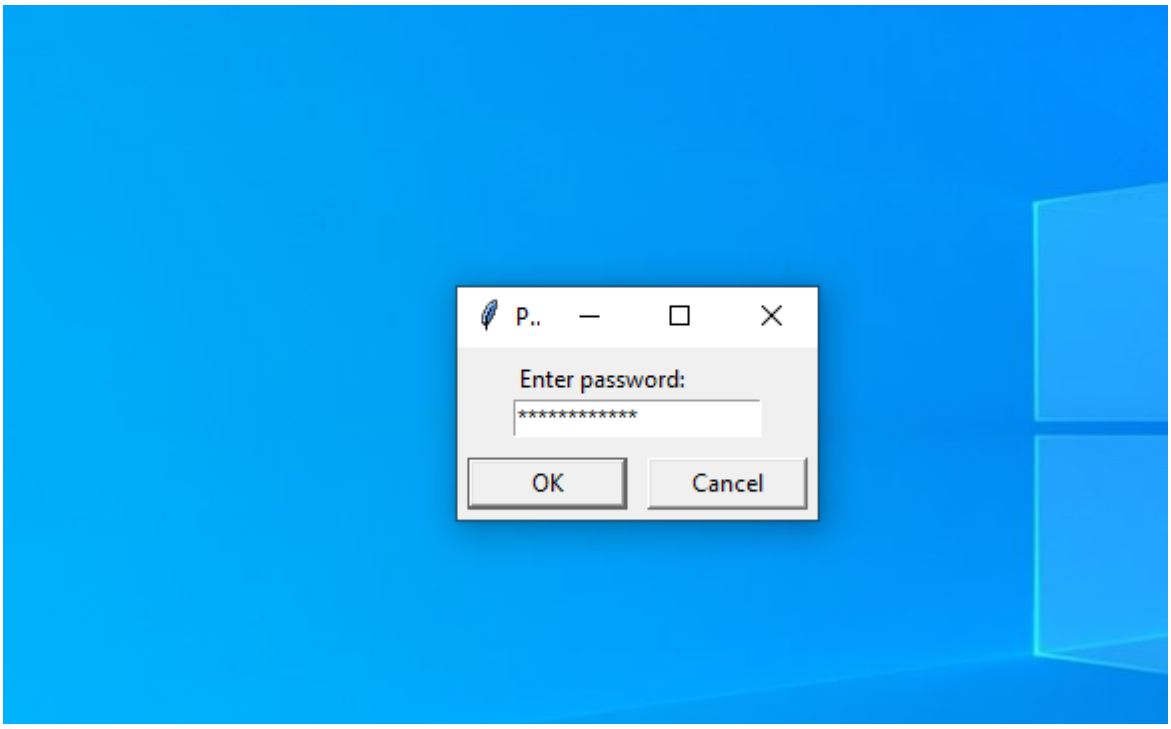
При встановленні застосунку використовуючи встановщик з другої лб, у користувача має запитатись пароль, який має бути захешований алгоритмом MD5 зазначеним у варіанті, і збережений в реєстр. Також до паролю додається випадкове значення при генерації ключа, отже і при збереженні паролю ми використаємо сіль при роботі з MD5. Вона також буде зберігатись.

Має бути реалізований криптографічний інтерфейс застосунку що буде зашифровувати і розшифровувати файл, `database.json` використаний для зберігання інформації про облікові записи. Алгоритмом шифрування обрано AES в режимі ECB як того вимагає варіант завдання.
На основі паролю введеного користувачем має бути отриманий ключ шифрування для алгоритму AES. Для цього використаємо PBKDF2 (Password-Based Key Derivation Function), з сіллю.
Для забезпечення достатнього рівня безпеки при роботі з ненадійним MD5 на пароль для розшифрування даних акаунтів встановлено обмеження на мінімальну довжину в 12 символів, наявність мінімум 1 заголовної літери, цифри та символу.

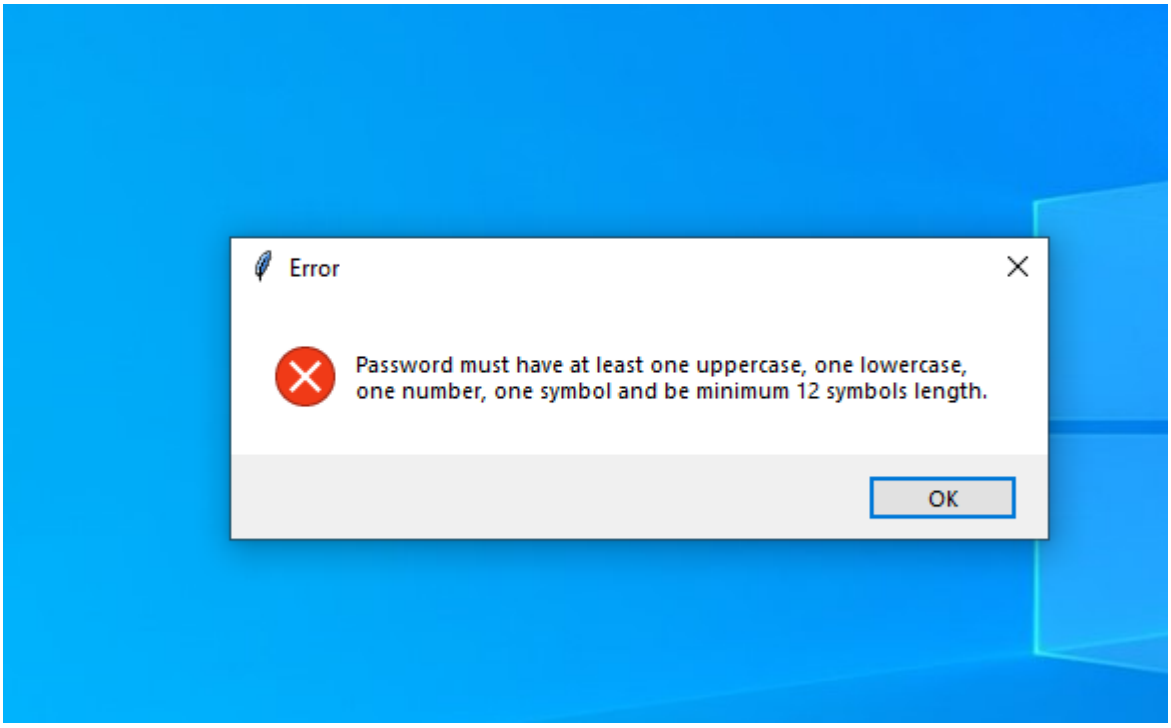
З міркуваннь безпеки було вирішено не створювати, під час роботи застосунку, тимчасовий файл з розшифрованими даними, а тримати їх в пам'яті, при цьому розшифрування і зашифрування при збереженні нових даних відбуваються в реальному часі.
Також з міркуваннь безпеки було вирішено змінювати сіль для хешування а також для створення ключа при кожному перезапуску застосунку, так як можливість ввійти та вийти в акаунт при роботі з ним, дозволяє тривалий час його не перезапускати.
При завершенні програми вся база даних перешифровується ключем згенерованим за допомогою парольної фрази та нової солі, таким чином сесійний ключ шифрування не буде повторюватись і не може бути вганадний чи перебраний за оптимальний час. Це зроблено у випадку якщо зловмиснику вдається отримати сесійний ключ для симетричного шифрування, а не парольну фразу.

В результаті виконання лабораторної роботи був написаний контроллер що відповідає за обробку криптографічних операцій, а також було видозмінено деякі частини файлів що вже існували. Важливі ділянки коду будуть надані далі, вихідні коди в повному обсязі доступні за посиланням

При встановленні застосунку, у користувача запитують кодову фразу, яка має обмеження на довжину і вміст, вона зберігатиметься в зашифрованому вигляді в реєстрі.

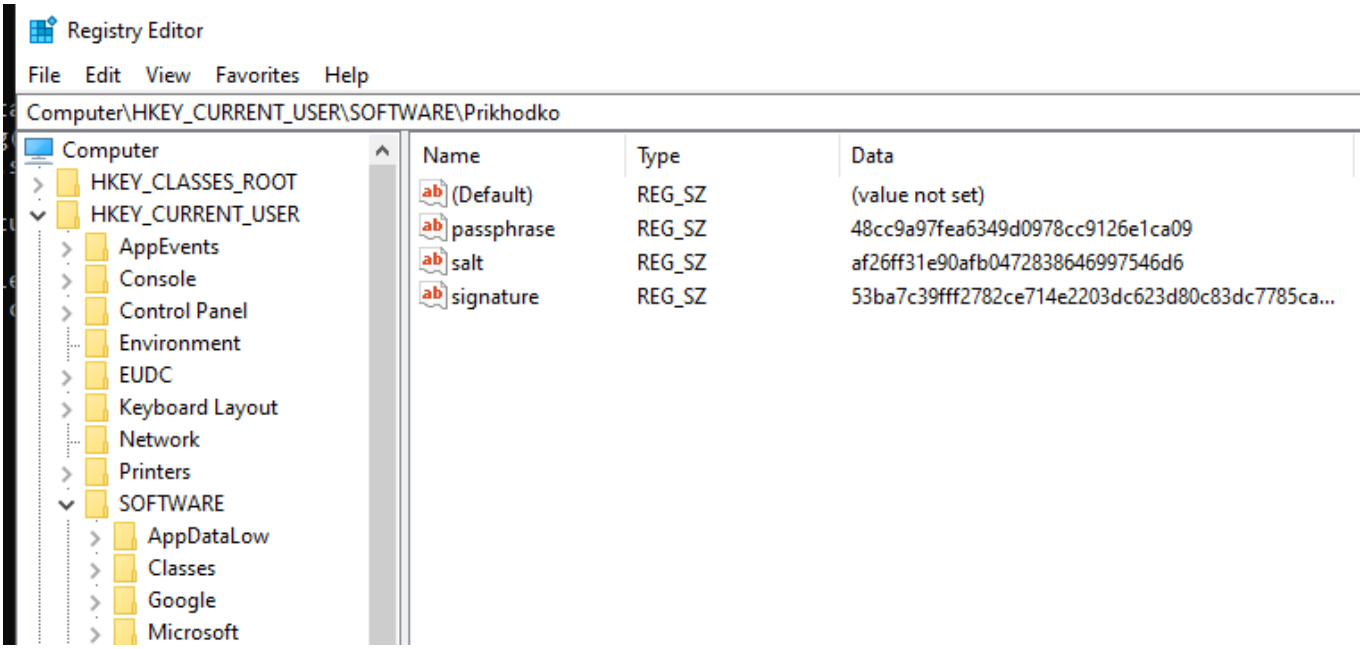


При введенні не достатньо надійного паролю, користувач отримає відповідне повідомлення

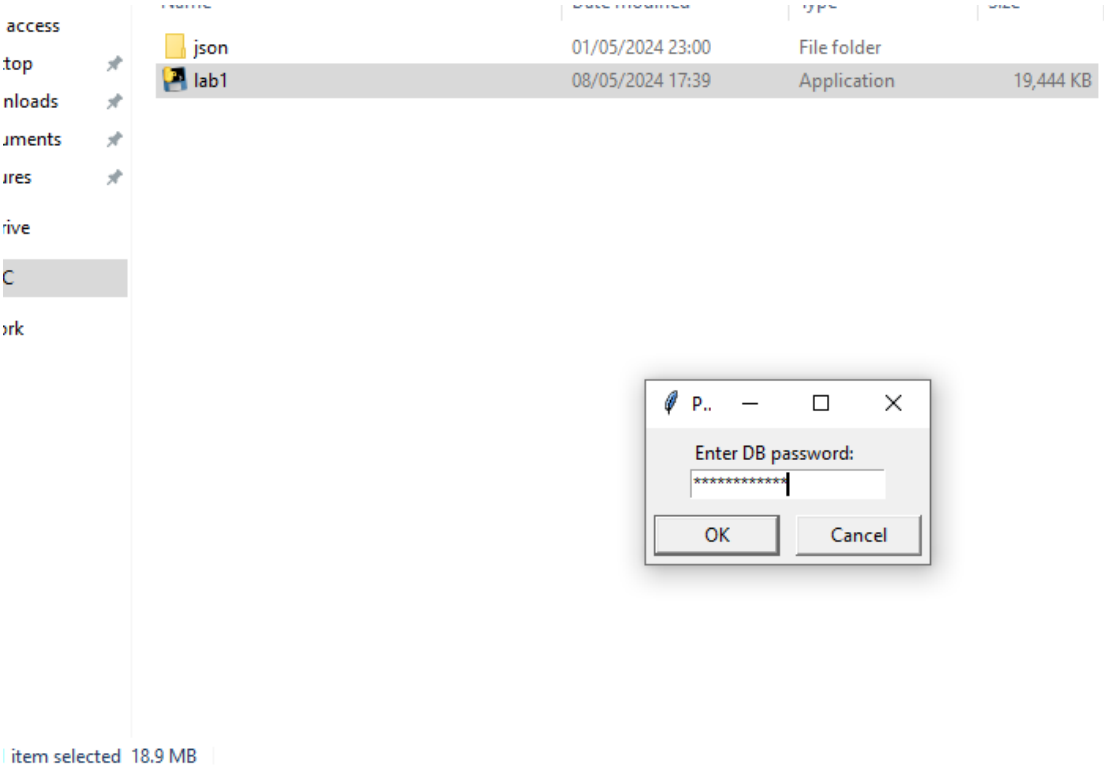


Якщо пароль було введено і підтверджено успішно, додаток встановиться як це було у 2

лабораторній роботі. Збережені дані можна перевірити відкривши реєстр.

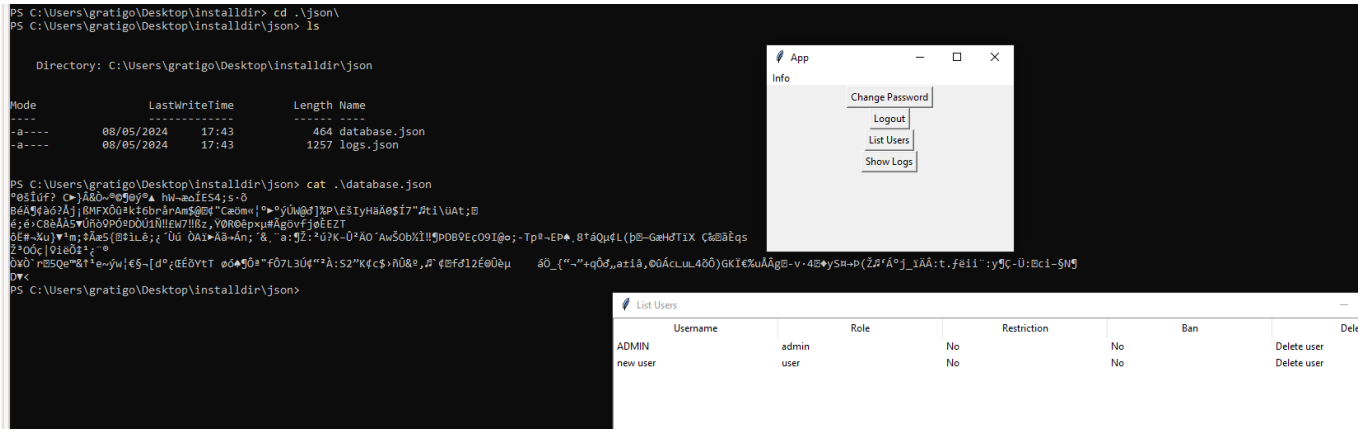


При запуску застосунку, необхідно буде ввести контрольну фразу що було обрана під час встановлення. Якщо вона введена неправильно, додаток закриється.



Якщо пароль введено правильно користувач отримає доступ до функціоналу застосунку. При цьому база даних зберігається в зашифрованому стані весь час, сесійний ключ

змінюється при перезапуску.



При завершенні застосунку, ключ зміниться і база даних буде перешифрована, відповідно до ключа створеного за допомогою нової солі.

Вихідний код для функцій в `crypto_controller.py`

```python
from hashlib import md5
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Protocol.KDF import PBKDF2
from json import dumps
import winreg
import os


def checkPass(password:str):
    key = winreg.CreateKey(winreg.HKEY_CURRENT_USER, r"Software\Prikhodko")
    stored_phrase, _ = winreg.QueryValueEx(key, 'passphrase')
    stored_salt, _ = winreg.QueryValueEx(key, 'salt')

    if md5((password + stored_salt).encode()).hexdigest() != stored_phrase:
        return None

    salt = os.urandom(16)
    hashed_sess = md5((password + salt.hex()).encode()).hexdigest()
    winreg.SetValueEx(key, 'passphrase', 0, winreg.REG_SZ, hashed_sess)
    winreg.SetValueEx(key, 'salt', 0, winreg.REG_SZ, salt.hex())
    winreg.CloseKey(key)
    return stored_salt


def decrBase(password:str, salt:str, data:bytes) -> str:
    key = PBKDF2(password, salt, dkLen=32, count=100000)
    cipher = AES.new(key, AES.MODE_ECB)
    data = unpad(cipher.decrypt(data), AES.block_size).decode()
    return data

def encrBase(password:str, salt:str, data:str) -> str:
    key = PBKDF2(password, salt, dkLen=32, count=100000)
    cipher = AES.new(key, AES.MODE_ECB)
    data = cipher.encrypt(pad(data.encode(), AES.block_size))
    return data

def encrEndBase(password:str, data:dict) -> bytes:
    regkey = winreg.CreateKey(winreg.HKEY_CURRENT_USER, r"Software\Prikhodko")
    salt, _ = winreg.QueryValueEx(regkey, 'salt')
    winreg.CloseKey(regkey)
    key = PBKDF2(password, salt, dkLen=32, count=100000)
    cipher = AES.new(key, AES.MODE_ECB)
    data = cipher.encrypt(pad(dumps(data).encode(), AES.block_size))
    return data
~
~
~
~
.\controllers\crypto_controller.py
```