Міністерство освіти і науки України

Національний технічний університет «ХПІ»

Навчально-науковий інститут комп’ютерних наук та інформаційних технологій

Кафедра комп’ютерної інженерії та програмування

ЗВІТ

з лабораторної роботи № 4

з дисципліни «Сучасні технології безпечного програмування»

«ВИКОРИСТАННЯ МНЕМОНІЧНИХ ФРАЗ ДЛЯ ФОРМУВАННЯ КЛЮЧІВ ШИФРУВАННЯ»

Виконав:

студент гр. КН-Н922б

Кулик Д.І.

Перевірив:

Бульба С. С.

Харків – 2022

Мета роботи: Дослідити і реалізувати механізм використання мнемонічних фраз для формування ключів шифрування

Індивідуальне завдання

- Використовуючи алгоритм bip39, створити seed генератора псевдовипадкових чисел за допомогою мнемонічної фрази та стосовні ключі шифрування.
- Зашифрувати текст
- Використовуючи раніше створену мнемонічну фразу, відновити ключі шифрування на дешифрувати текст. Вдосконалитись, що оригінальний та дешифрований тексти однакові.

З.І., для додаткових балів необхідно реалізувати підтримку україномовних мнемонічних фраз.

Хід роботи

BIP описує реалізацію мнемонічного коду або мнемонічного речення — групи слів, які легко запам’ятати — для створення детермінованих гаманців. Він складається з двох частин: створення мнемоніки та перетворення її на бінарне початкове значення. Пізніше це початкове число можна використовувати для створення детермінованих гаманців за допомогою BIP-0032 або подібних методів.

Важливі фрагменти програми

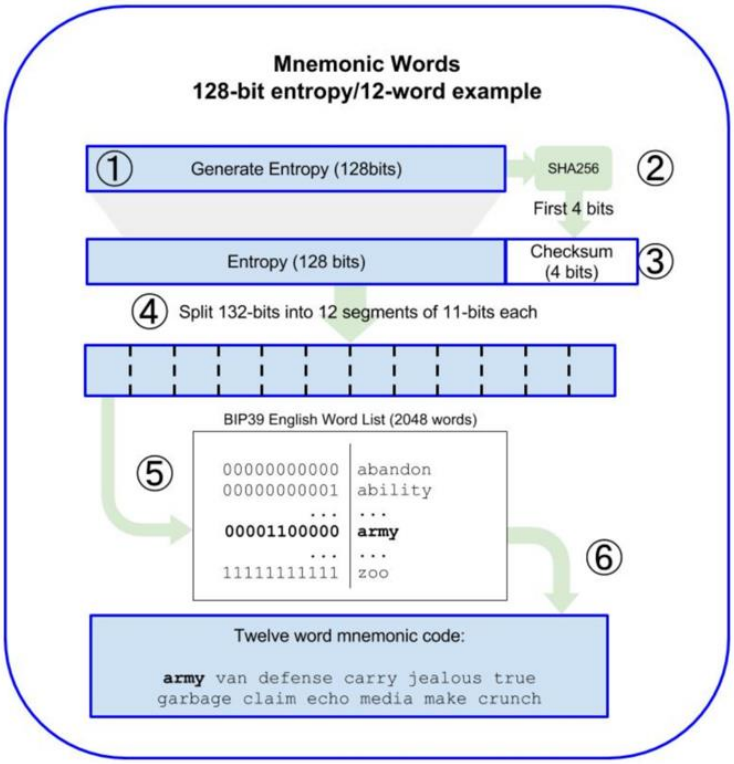

Рисунок 1 – Алгоритм створення ключів

```
def get_2048_words(lang):
    if lang == 'en':
        df = pd.read_csv(
            './kulyk04/en_words.txt',
            names=['words']
        ).reset_index().set_index('words')

        nums = df.to_dict()['index']
    elif lang == 'ua':
        df = pd.read_csv(
            './kulyk04/ua_words.txt',
            names=['words']
        ).reset_index().set_index('words')

        nums = df.to_dict()['index']

    return nums
```

Рисунок 1 – Зчитування слів зі словників

```
def get_seed(mnemonic: str, passphrase: str = "") -> bytes:
    decode_phrase(mnemonic)
    mnemonic = normalize_string(mnemonic)
    passphrase = "mnemonic" + normalize_string(passphrase)
    mnemonic_bytes = mnemonic.encode("utf-8")
    passphrase_bytes = passphrase.encode("utf-8")
    stretched = hashlib.pbkdf2_hmac(
        "sha512", mnemonic_bytes, passphrase_bytes, PBKDF2_ROUNDS
    )
    print(f'GENERATED SEED       : {stretched.hex()}')
    return stretched
```

Рисунок 3 – Генерація чисел

```
def encrypt(text, mnemonic):
    rsa = RSA.generate(1024, randfunc=PRNG(get_seed(mnemonic)))
    public_key = rsa.public_key().export_key('PEM')
    print(f'PUBLIC KEY           : {public_key}')
    cipher = PKCS1_OAEP.new(RSA.import_key(public_key))
    encrypted_text = cipher.encrypt(text)

    return encrypted_text
```

Рисунок 4 – Функція шифрування

```
def decrypt(encrypted_text, mnemonic):
    rsa = RSA.generate(1024, randfunc=PRNG(get_seed(mnemonic)))
    private_key = rsa.export_key('PEM')
    print(f'PRIVATE KEY          : {private_key}')
    cipher = PKCS1_OAEP.new(RSA.import_key(private_key))
    decrypted_text = cipher.decrypt(encrypted_text)

    return decrypted_text
```

Рисунок 5 – Функція дешифрування

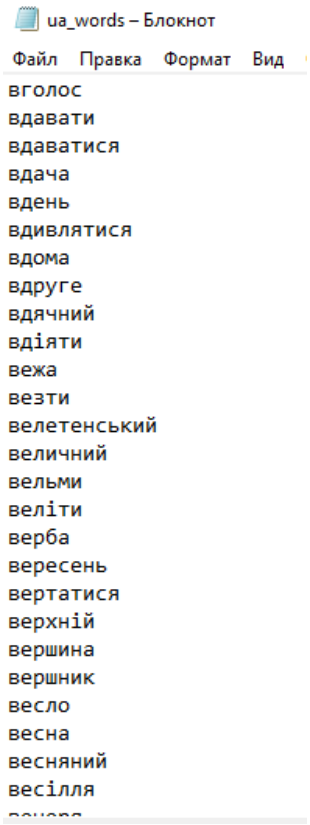

Рисунок 6 – Словник українських слів

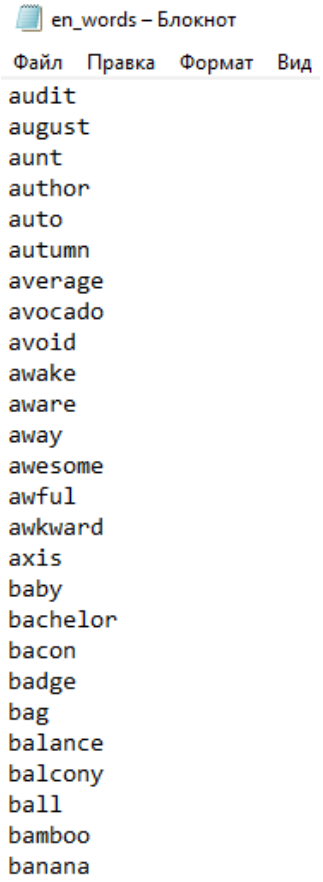

Рисунок 7 – Словник англійських слів

Результати роботи програми

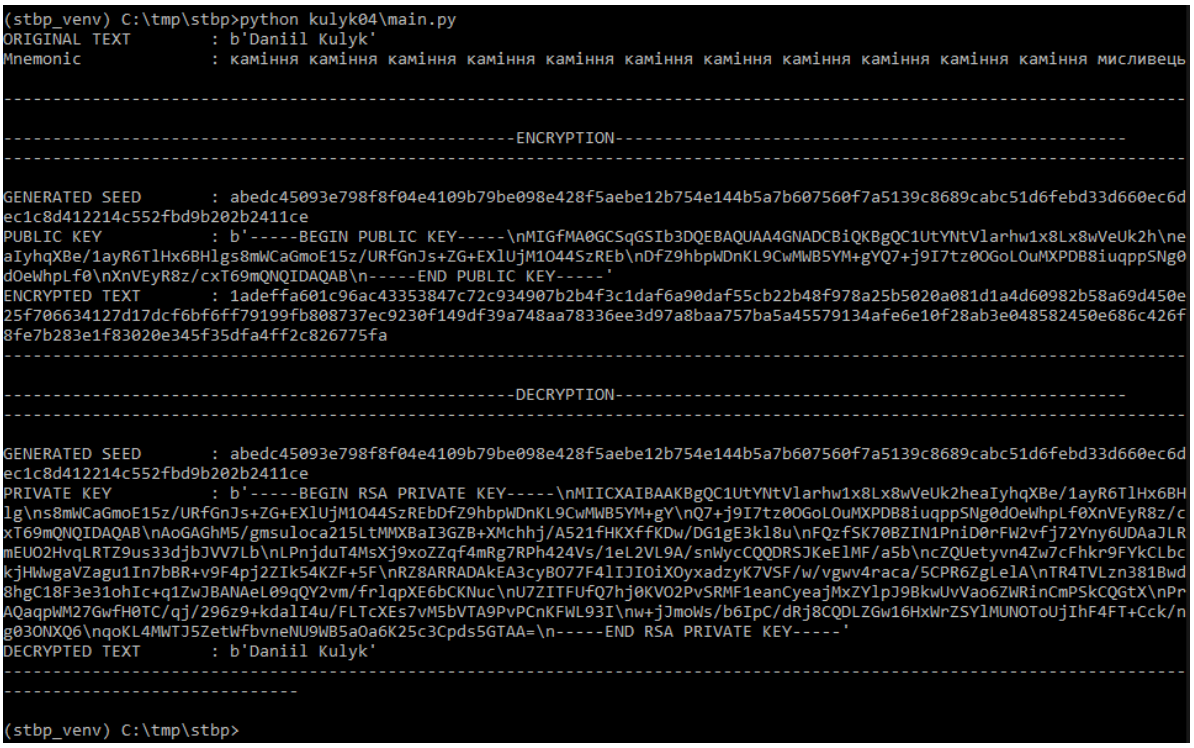

Рисунок 8– Результат виконання програми для ua мнемонічних фраз

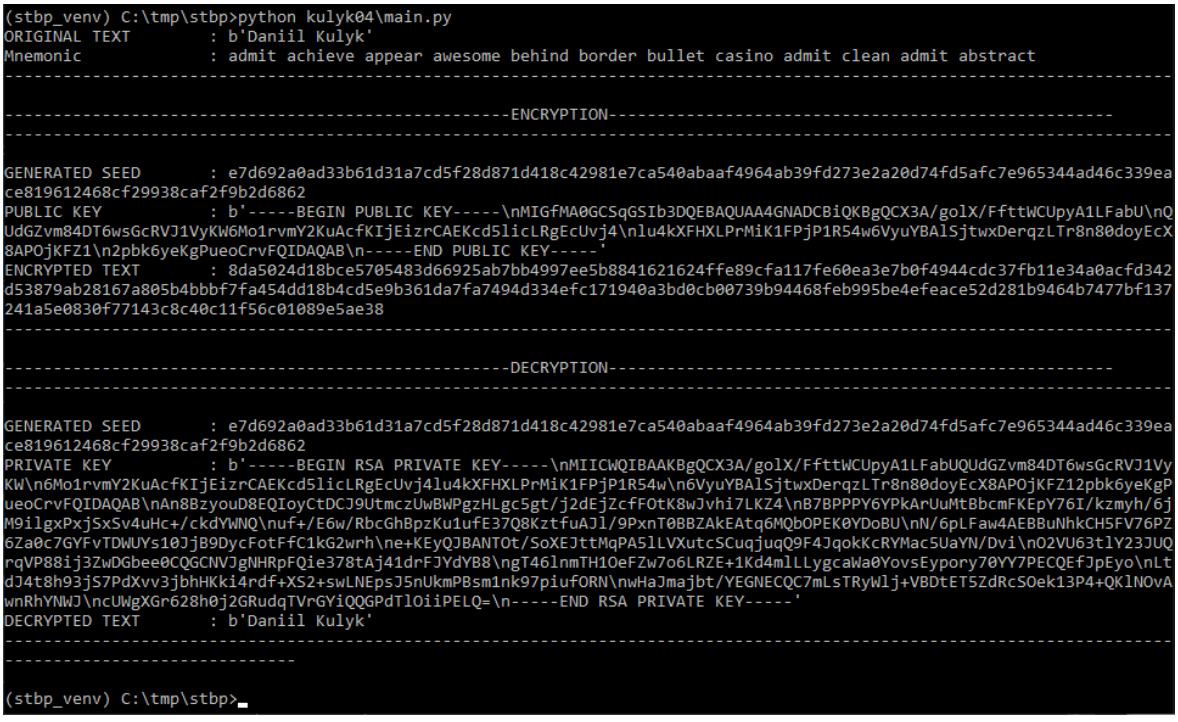

Рисунок 9 – Результат виконання програми для en мнемонічних фраз

Висновки: в результаті виконання лабораторної роботи було досліджено і реалізовано механізм використання мнемонічних фраз для формування ключів шифрування RSA.