НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра Інформаційної Безпеки

Лабораторна робота №4 дисципліни

”КРИПТОГРАФІЯ”

Підготували:

студенти групи ФБ-03

Борох Іван

Жигун Анастасія

Перевірив:

Чорний Олег Миколайович

Київ 2022

Тема роботи: Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем

Мета роботи: ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Порядок виконання роботи

Хід роботи

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.

Пошук випадкового простого числа:

```
def random_search(n_max, n_min=0):
    while True:
        x = randint(n_min, n_max)
        if miller_rabin_test(x):
            return x
```

Тест Міллера-Рабіна:

```
def miller_rabin_test(num):
    if num % 2 == 0 or num % 3 == 0 or num % 5 == 0 or num % 7 == 0 or num % 11 == 0 or num % 13 == 0:
        return False

    d = num - 1
    s = 0
    while d % 2 == 0:
        d = d // 2
        s += 1

    x = randint(2, num - 2)

    if gcd(x, num) > 1:
        return False
```

```
    if pow(x, d, num) == 1 or pow(x, d, num) == -1:
        return True

    for i in range(1, s - 1):
        x = (x ** 2) % num
        if x == - 1:
            return True
        if x == 1:
            return False
    return False
```

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq p1q1 ; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – абонента B.

Генерація простих чисел:

```
# random prime nums
A_numbers = (random_search(2 ** 258, 2 ** 256), random_search(2 ** 258, 2 ** 256))
B_numbers = (random_search(2 ** 258, 2 ** 256), random_search(2 ** 258, 2 ** 256))
```

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p,q) та відкритий ключ (n,e) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , (,) 1 n1 e та секретні d і d1.

Генерація ключів:

```
# keys
open_A, secret_A = create_keys(A_numbers[0], A_numbers[1])
open_B, secret_B = create_keys(B_numbers[0], B_numbers[1])
```

```
def create_keys(p, q):
    e = 2 ** 16 + 1
    n = p * q
    fi = (p - 1) * (q - 1)

    d = pow(e, -1, fi)

    open_key = (n, e)
    secret_key = (d, p, q)

    return open_key, secret_key
```

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів A и B, перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.

```python
def encryption(data, open_key_B):
    return pow(data, open_key_B[1], open_key_B[0])


def decryption(data, secret_key_B):
    return pow(data, secret_key_B[0], secret_key_B[2] * secret_key_B[1])


def make_signature(data, secret_key_A):
    return pow(data, secret_key_A[0], secret_key_A[2] * secret_key_A[1])


def authentication(de_data, de_signed, open_key_A):
    if de_data == pow(de_signed, open_key_A[1], open_key_A[0]):
        return True
    return False
```

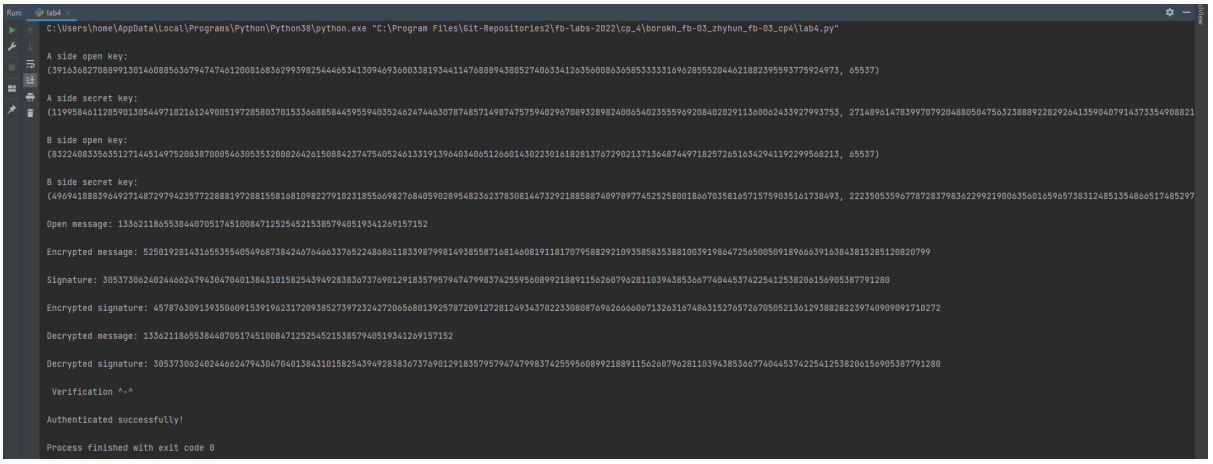
```python
# message encryption
encrypted = encryption(message, open_B)
print(f"\nEncrypted message: {encrypted}")
# signature creation
signed = make_signature(message, secret_A)
print(f"\nSignature: {signed}")
# signature encryption
en_signed = encryption(signed, open_B)
print(f"\nEncrypted signature: {en_signed}")
```

```python
# data decryption
decrypted = decryption(encrypted, secret_B)
print(f"\nDecrypted message: {decrypted}")
# signature description
de_signed = decryption(en_signed, secret_B)
print(f"\nDecrypted signature: {de_signed}")

# verification
print("\n Verification ^-^\n")
if authentication(decrypted, de_signed, open_A):
    print("Authenticated successfully!")
else:
    print("Wrong signature!!!")
```

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа 0 k n.

Результат виконання:



Проблеми, які виникали та шляхи їх вирішення:

Висновок

У ході виконання даного лабораторного практикуму ми ознайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA, системою захисту інформації на основі криптосхеми RSA,

реалізували з використанням цієї системи засекречений зв'язок й електронний підпис, а також розібрались із протоколом розсилання ключів.

В результаті опрацювання методичних вказівок та практичної частини ми вивчили принципи роботи криптосистеми RSA та алгоритму електронного підпису, а ще ознайомились з методами генерації параметрів для асиметричних криптосистем.