МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Фізико-технічний інститут

Криптографія
Комп’ютерний практикум No4

Виконав:
студент групи ФБ-05
Ємельянов Р.

Київ - 2022

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.

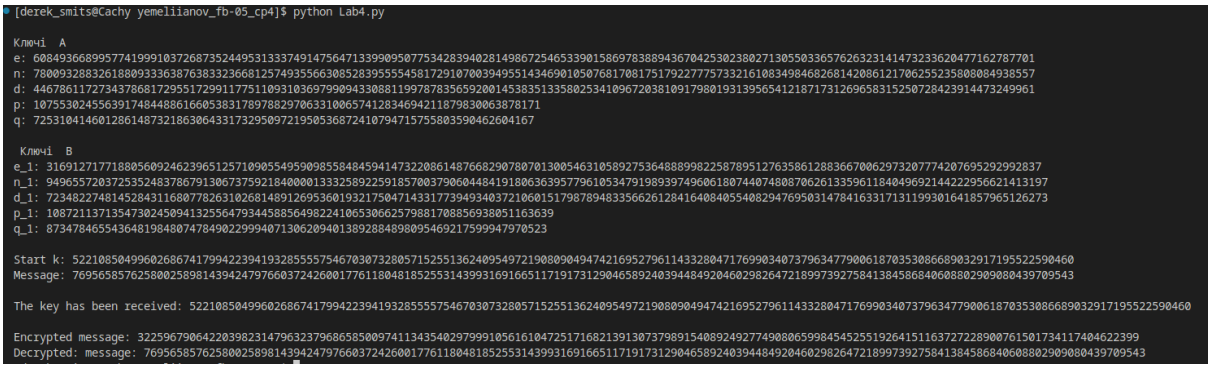
Постановка задачі:

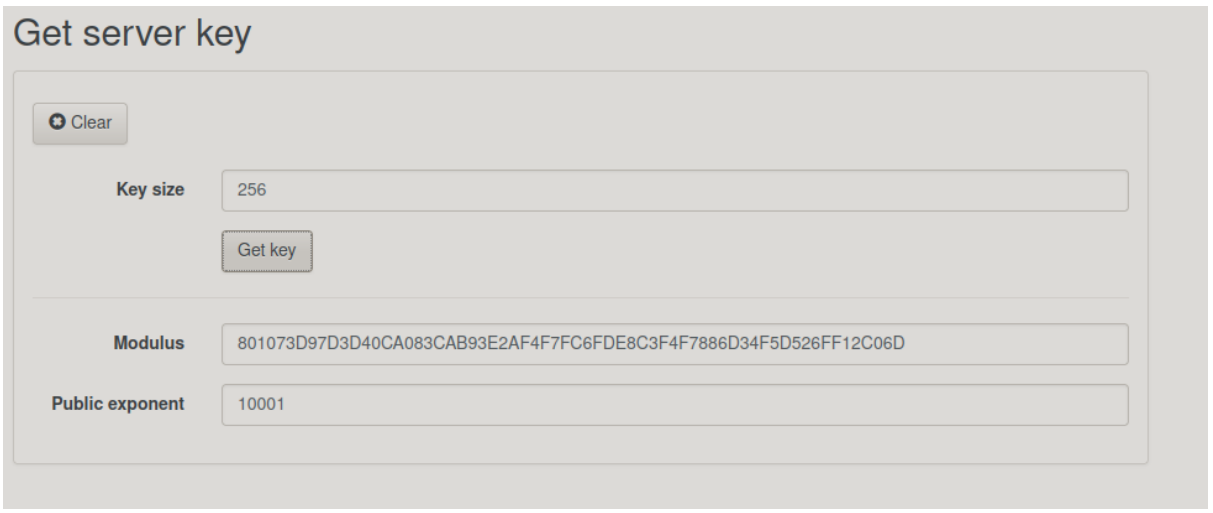
1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.
2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p1 , q1 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.
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) , (e1, n1) та секретні d і d1.
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
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 $0 < k < n$. Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція Encrypt(), яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату

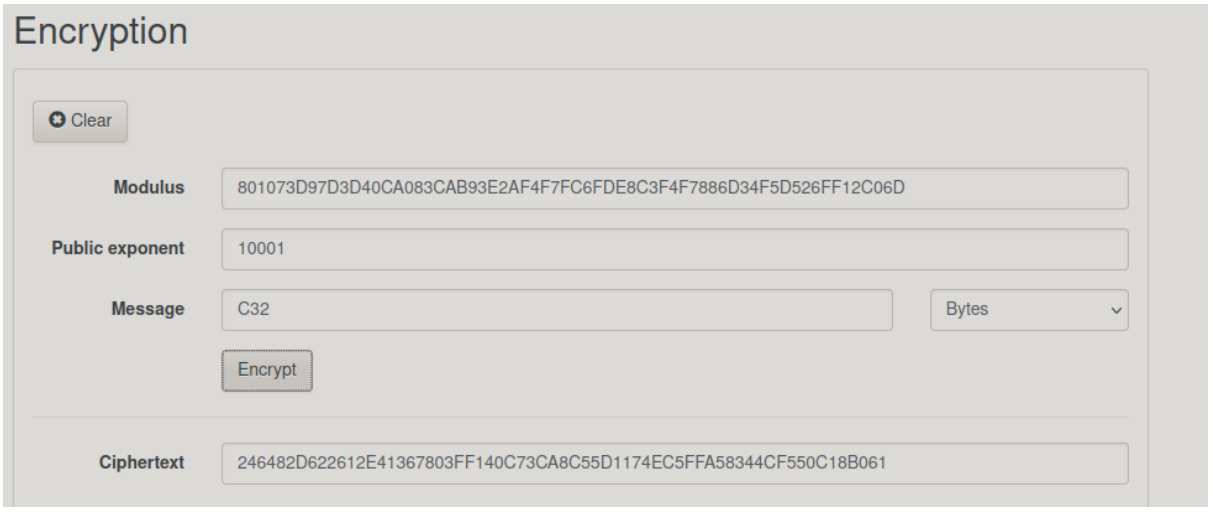
шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey(). Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально

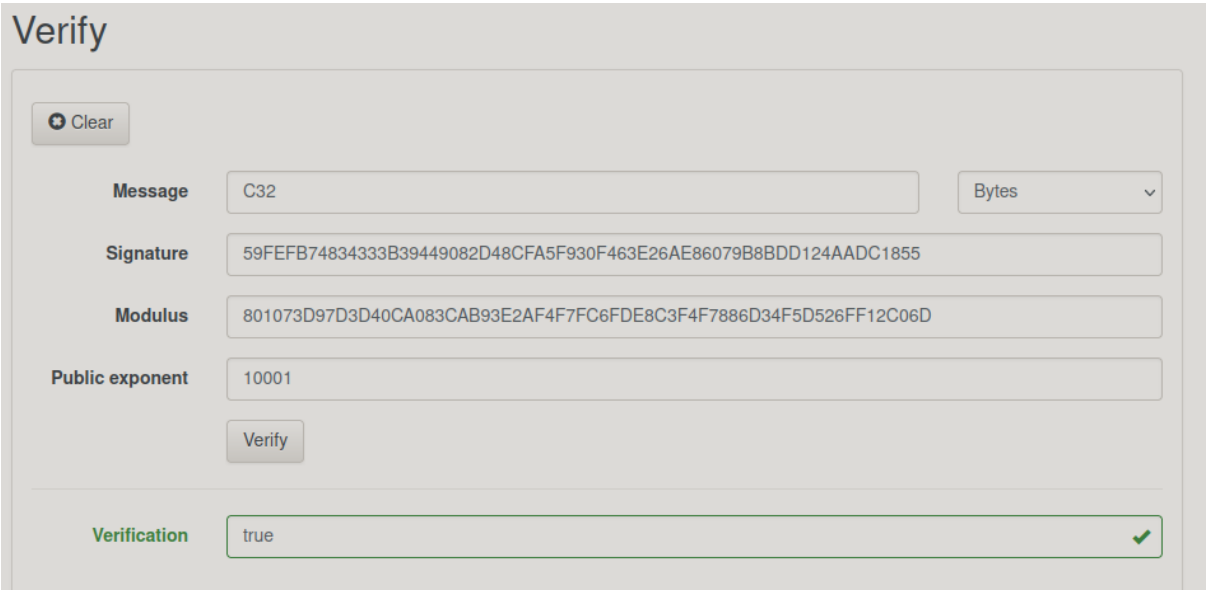
Хід роботи:

У нашій лабораторній роботі ми реалізували усі потрібні функції а також тести на них









Внесемо певні зміни у нашу програму аби перевірити отримані дані

```
def set_test_custom(self, m, s, e, n):
    self.m = int(m, 16)
    self.s = int(s, 16)
    self.e = int(e, 16)
    self.n = int(n, 16)

def test_case(self):
    print(f'n: {self.n}')
    print(f'e: {self.e}')
    print(f'mes: {self.m}')
    print(f'Ciphertext: {hex(self.encrypting(self.m, self.e, self.n))}')
    print(f'sign: {self.s}')
    if self.m == pow(self.s, self.e, self.n):
        print('Passed!')
    else:
        print('Failed!')
    return self.m == pow(self.s, self.e, self.n)
```

```
rsa_test.set_test_custom(
    "C32",
    "59FEFB74834333B39449082D48CFA5F930F463E26AE86079B8BDD124AADC1855",
    "10001",
    "801073D97D3D40CA083CAB93E2AF4F7FC6FDE8C3F4F7886D34F5D526FF12C06D"
    )
rsa_test.test_case()
```

```
n: 57925113736022094062863366039231680897275770885516528154985558985582962720877
e: 65537
mes: 3122
Ciphertext: 0x246482d622612e41367803ff140c73ca8c55d1174ec5ffa58344cf550c18b061
sign: 40706358157874831496918404139578035538507581592213491719683923675056736835
Passed!
```

Висновки:

В результаті виконання практикуму я ознайомився з системою захисту інформації на основі криптосхеми RSA. Було засвоєно і практично використано тест Міллера-Рабіна для перевірки чисел на простоту і генерації простих чисел. З допомогою Asym Crypto Lab Environment перевірив створені функції криптосхеми RSA.