КРИПТОГРАФІЯ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

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

Мета та основні завдання 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:

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.
2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p_1q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 – 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), (e_1, n_1) та секретні d і d_1.
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 $0 < k < n$.

Хід роботи:

1. Напишемо функцію пошуку випадкового простого числа заданої довжини:

```
def RandomPrime(length = 256):
    while 1==1:
        num = randint(1 << (length - 1), (1 << length) - 1)
        if MillerRabinTest(num):
            return num
```

Для перевірки на простоту будемо використовувати тест Міллера-Рабіна:

```
def MillerRabinTest(num, k = 100):
    if num % 2 == 0:
        return False
    s = 0
    d = num - 1
    while d % 2 == 0:
        s += 1
        d //= 2
    for _ in range(k):
        x = randint(1, num)
        g, _, _ = euclid(num, x)
        if g != 1:
            return False
        xd = pow(x, d, num)
        if xd == 1 or xd == num - 1:
            continue
        xr = xd
        for _ in range(1, s):
            xr = pow(xr, 2, num)
            if xr == 1:
                return False
            elif xr == num - 1:
                break
        else:
            return False
    return True
```

3-5. Створимо пару ключів для абонентів A та B, так щоб $pq \leq p_1q_1$:

```
p = RandomPrime()
q = RandomPrime()
p1 = RandomPrime()
q1 = RandomPrime()
if p*q > p1*q1:
    public_keyA, dA = GenerateKeyPair(p1, q1)
    public_keyB, dB = GenerateKeyPair(p, q)
else:
    public_keyA, dA = GenerateKeyPair(p, q)
    public_keyB, dB = GenerateKeyPair(p1, q1)
print("Ключі абонента A: "+str(public_keyA)+" "+str(dA))
print("Ключі абонента B: "+str(public_keyB)+" "+str(dB))
```

```
Ключі абонента A: [84594143740397146535015092165759343530963030295887675666950916530004529285068563791138398156075708827252249426250687447741704466298823539660659670923896691, 65537] 41025003745991462964773863226945530241613103924412133538652989610039122240372242878755384546192668955866457680626936351289444932135193138553520236054433329
Ключі абонента B: [92888302781729262865722518666917395838012482099810345248927231606075479947937451966499283151663760853625898532731672038721334482020385662921685191227777757, 65537] -28559429040875301688272407207201817692844523159607221214976024487883499716967525103510516182720248708505356786602000715659007293674327988638558304495025 27
```

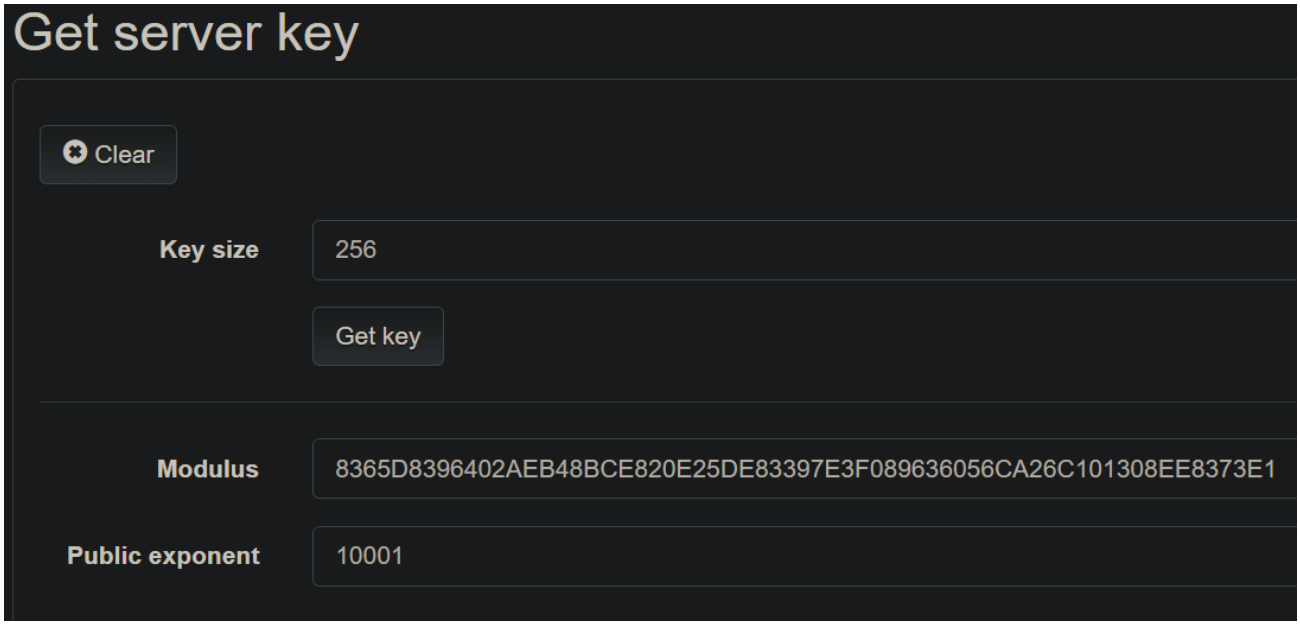
Зашифруємо, розшифруємо, підпишемо та перевіримо підпис. Також перевіримо протокол конфіденційного розсилання ключів.

```
Повідомлення: 17026404016525622772318570133002405784148693544096491268951466348619461222882433453460949743331753198888045508330864613424198844417935824016549607849254939
Абонент A зашифрував повідомлення: 47719425308004993830206092909129909897514964225234563195891130299032392790587437426723985836529965256528941862224901375447340719115557306215897994645992 72
Абонент B розшифрував повідомлення: 17026404016525622772318570133002405784148693544096491268951466348619461222882433453460949743331753198888045508330864613424198844417935824016549607849254939
Абонент A підписав повідомлення: 16422197493259116838034904660542185387424318083474772616416807164168951664665327961248136260080153918004557784327514794188953102820882491825070553848509 5
Абонент B перевірив підпис: True
Секретний ключ k: 693253635407090143489264359286772994860247886562188454200826845504909733936344633377791761581318006829577720767093778476047515469204717232906737645396348 0
Абонент A відправив секретний ключ k: 12340879471170784703706997619703950346431202810256053899109282996101536885453263635642319561740980379372597394239876377671492748128740673695892132303043343524828143960233715188963505693048327828859463348076801949931025091073858502185018417390792387075544813849826181425145862950985436095435633323652253229101
Абонент B перевірив автентифікацію: 16422197493259116838034904660542185387424318083474772616416807164168951664665327961248136260080153918004557784327514794188953102820882491825070553848509 5
Перевірка пройшла успішно!
Абонент B розшифрував секретний ключ k: 693253635407090143489264359286772994860247886562188454200826845504909733936344633377791761581318006829577720767093778476047515469204717232906737645396348 0
```

Все працює як потрібно.

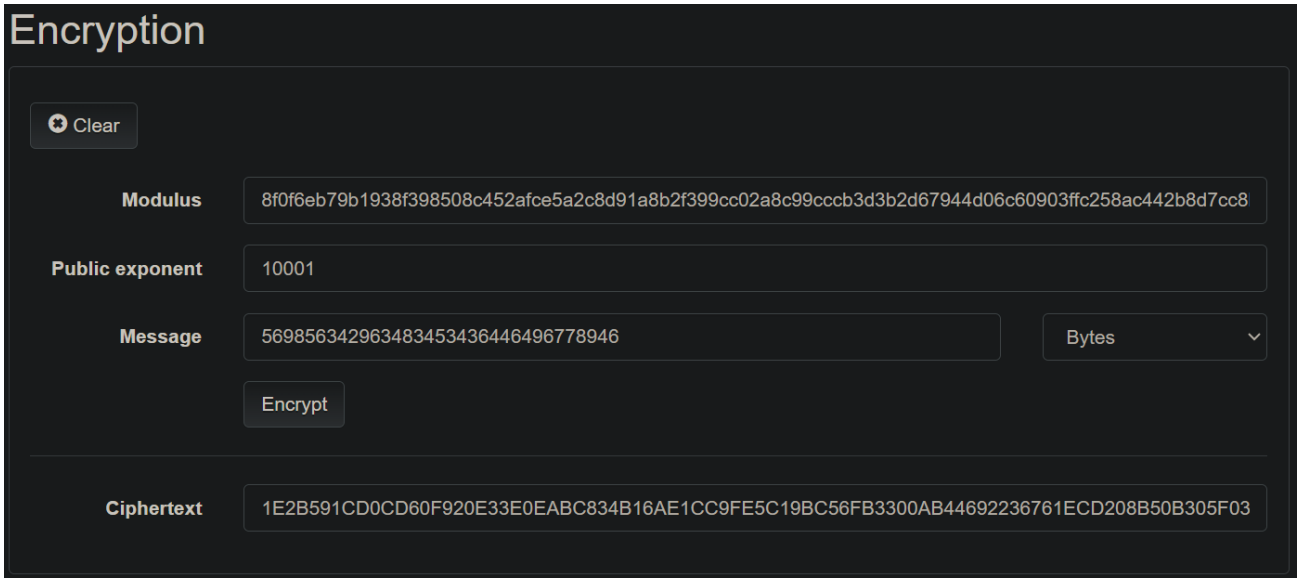
Тестування з сервером

Сгенеруємо ключі для сервера та для себе:




```
Our public key: n=0x8f0f6eb79b1938f398508c452afce5a2c8d91a8b2f399cc02a8c99cccb3d3b2d67944d06c60903ffc258ac442b8d7cc8bb7c29c84b53d5e58884c667bd880691 e=0x10001
Enter server Modulus: 8365D8396402AEB48BCE820E25DE83397E3F089636056CA26C101308EE8373E1
Enter server exponent: 10001
```

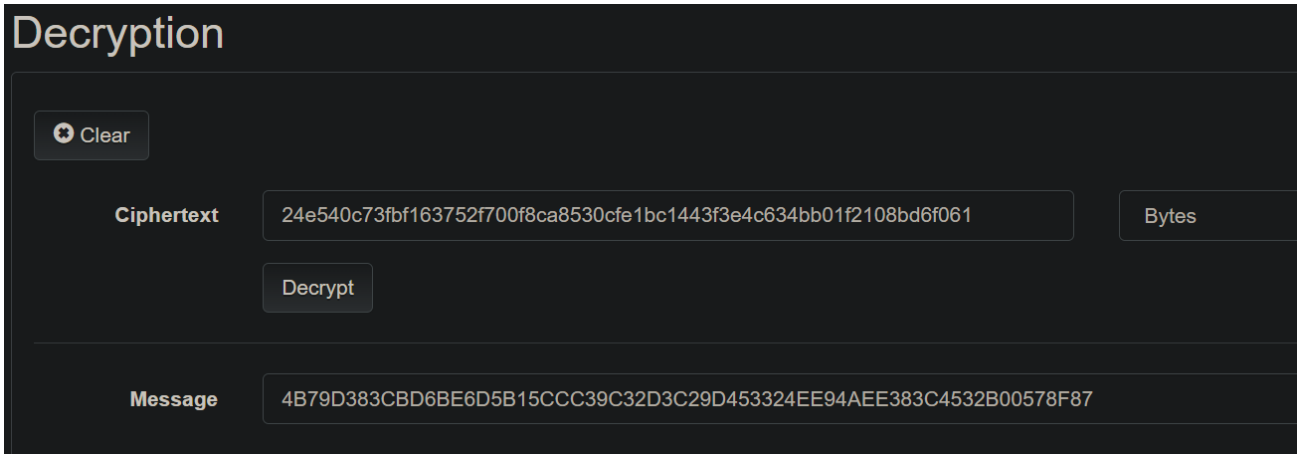
Розшифруємо повідомлення з сервера:



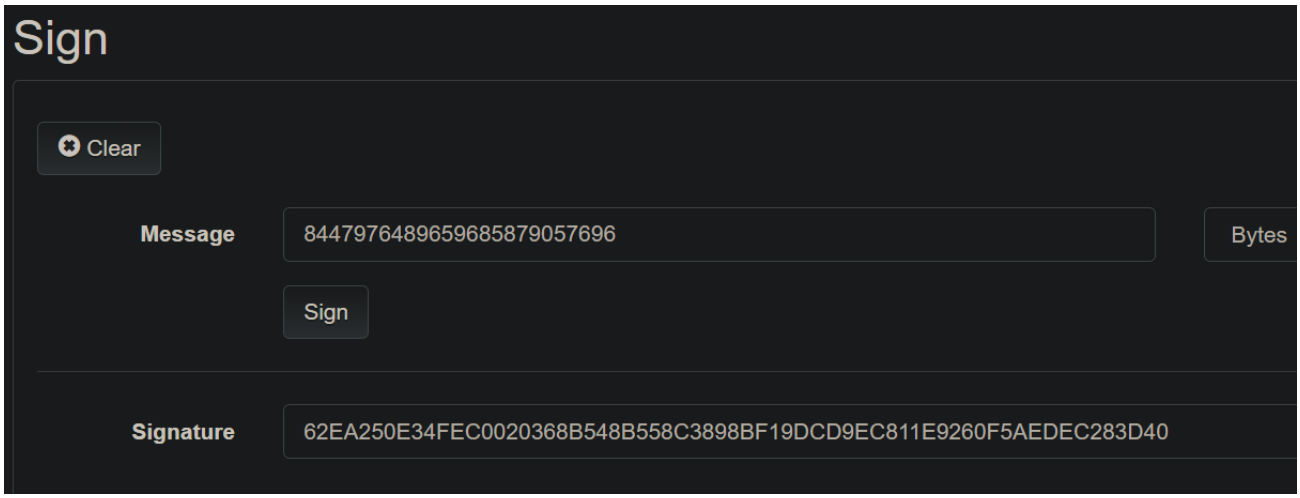

```
Decrypting message from server
Enter ciphertext: 1E2B591CD0CD60F920E33E0EABC834B16AE1CC9FE5C19BC56FB3300AB44692236761ECD208B50B305F036F7AEDDE321C0A665D891D01470B0001B8F1F63642B0
Decrypted message: 0x569856342963483453436446496778946
```

Зашифруємо повідомлення для сервера:

```
Encrypting message for server
Message: 0x4b79d383cbd6be6d5b15ccc39c32d3c29d453324ee94aee383c4532b00578f87
Encrypted message: 0x24e540c73fbf163752f700f8ca8530cfe1bc1443f3e4c634bb01f2108bd6f061
```



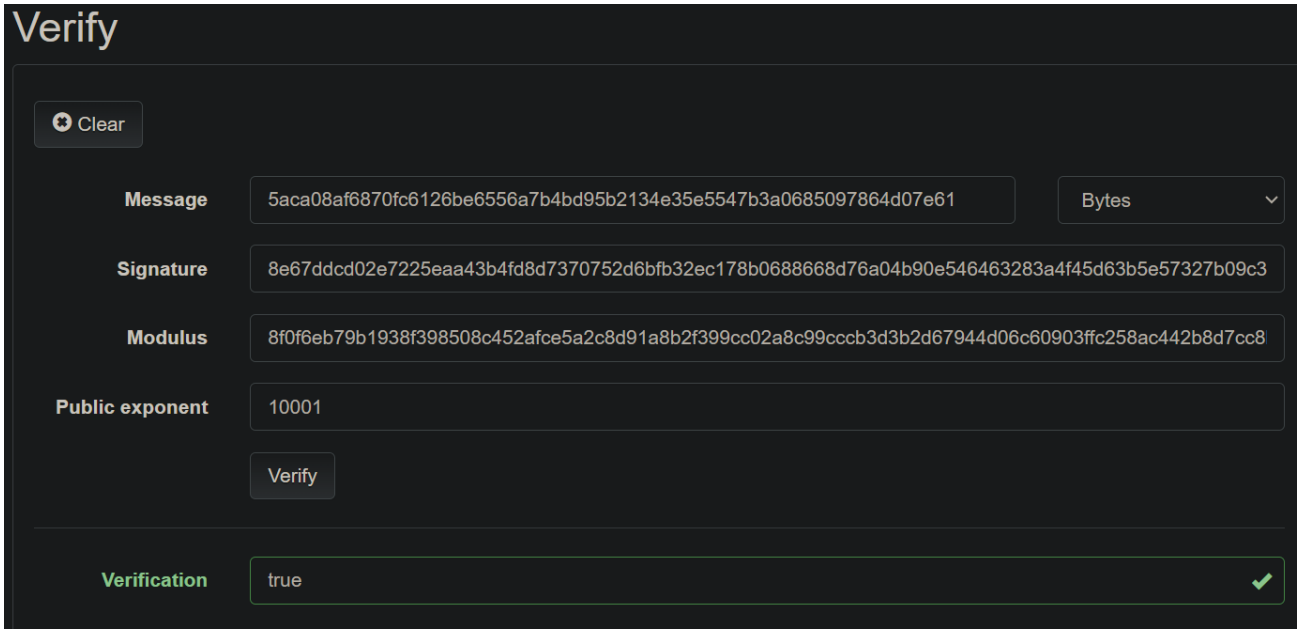
Перевіримо підпис повідомлення, отриманого з сервера:




```
Verifing signature from server
Enter server message: 844797648965968587905769 6
Enter server signature: 62EA250E34FEC0020368B548B558C3898BF19DCD9EC811E9260F5AEDEC283D40
Verification successful!
```

Підпишемо повідомлення для сервера:

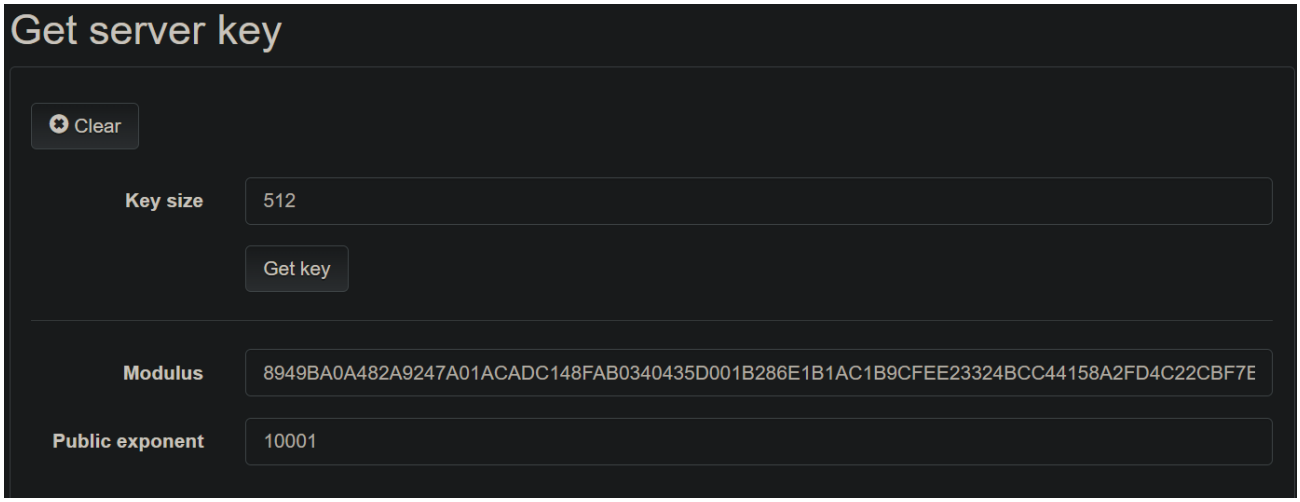
```
Signing message for server
Message: 0x5aca08af6870fc6126be6556a7b4bd95b2134e35e5547b3a0685097864d07e61
Signed message: 0x8e67ddcd02e7225eaa43b4fd8d7370752d6bfb32ec178b0688668d76a04b90e546463283a4f45d63b5e57327b09c3c3c4e9bc6ee075990e3732107996e3d733f
```



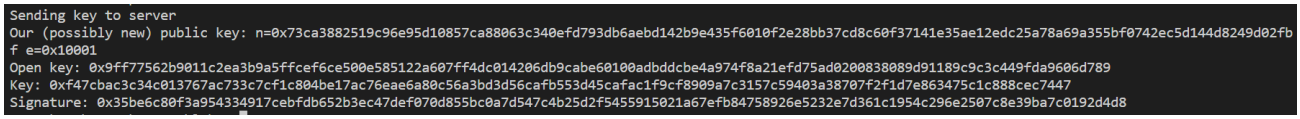
Отримаємо секретне значення з сервера:

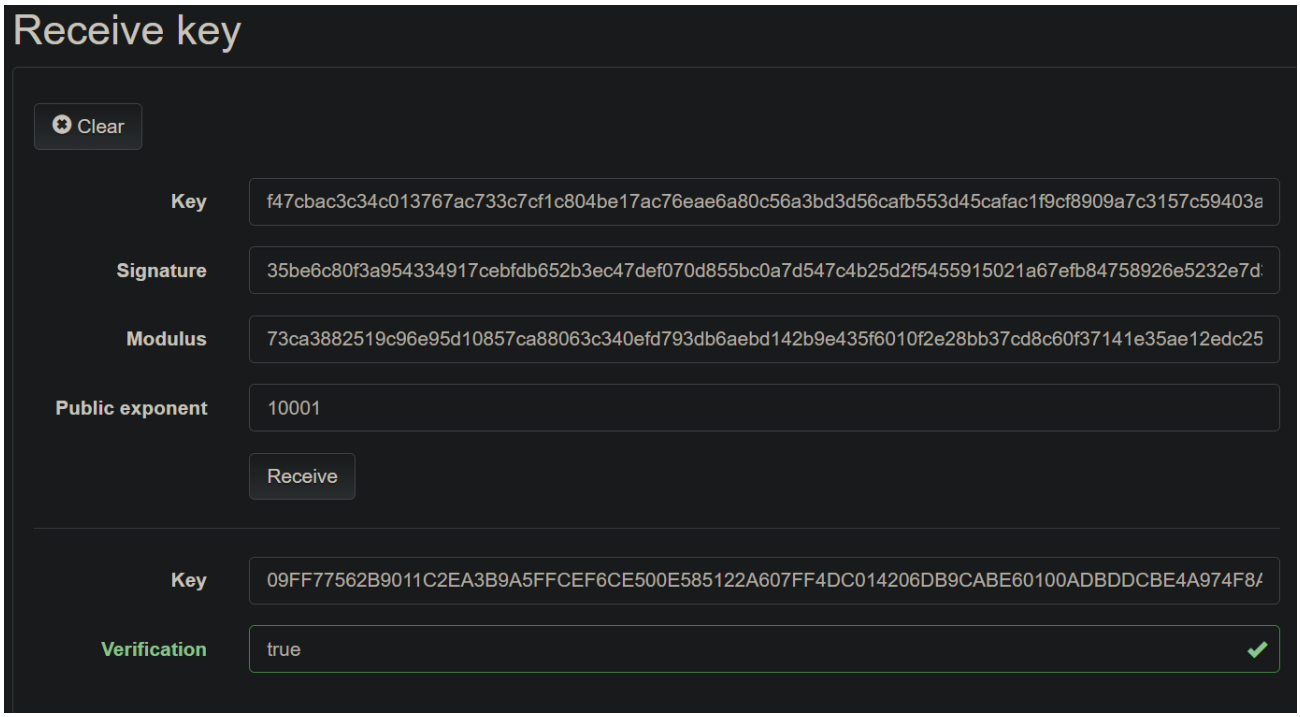

```
Receving key from server
Enter server key: 1643615DFEFE1E44AE85184726CC5728BDD4F8A238ACF05B34423C6F6AE7606A1328675344C1F956B32E06F9FA3A8113EDD09796DCFB0C65AF8A6B6C018EC819
Enter server signature: 52F932ECBFFE6C436FD8FC782E8273A2C3EBD71A5EAC7263183E733FFAFB7681852624D5B2967586AD7F576C4C37DC8409C123748A7D8E57AE81CCCF1D4C57E0
Verification successful!
Server key: 0x8e5240e2923a0a9
```

Після багатьох неуспішних спроб відправлення секретного значення, я зрозумів потрібно було брати більший ключ для сервера:



Відправимо секретне значення на сервер:





Висновок:

В ході цієї лабораторної я навчився використовувати криптосистему RSA та алгоритм електронного підпису.