Міністерство освіти і науки України Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

Криптографія

Лабораторна робота No 4
Варіант - 6

Виконали: студенти групи ФБ-13

Клименко Д. О. Стягайло Д. А.

Київ 2023

Мета: 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 і рекомендації щодо 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.

```
def __prime_test(num, k):
        if num == 2 or num == 3: return True
        if num < 5 or num % 2 == 0: return False
        # step 0
        d = num - 1
        s = 0
        while d % 2 == 0:
            d = d // 2
            s += 1

        for _ in range(k):
            # step 1
            a = random.randint(2, num - 1)
            b0 = __horner_pow(a, d, num)
            c = d
            while c != num - 1:
                if b0 == 1 or b0 == num - 1: break;
                b = (b0 ** 2) % num
                if b == 1: return False
                b0 = b
                c = c * 2
            if c == num - 1: return False
        # step 3
        return True
```

Помилка тесту буде 1/4^k, в random prime K=150, отже помилка 1/4^150.

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.

```
[*]Init users:
   p1:37987493707993602374838455834350219382790973458663144050686790515793385043 37
   q1:63292177030860205954562725589559488353335275739843305038907808543946438025893
   p2:81995260661316850295690770162855491654883648397658955562530943859217115064049
   q2:99775898341063814800624669863016220311642549093232140340242866822277672310041
```

кандидатів що не пройшли було дуже багато, вивели їх чисто один раз для протоколу, в основний код не додавали, щоб вивід не засмічувати:

```
784446072938141360286499338267990744885969982344060987
855117542696745904609 15
625616468187838817635624670088307170445911830336963366
022707401076936324786 92
770991428216746301194681606609868236850706319700909128
368666951238192543657 96
107579659638438849357589723286733199186010683774945966
385396062416666193027953
725061052006198891774322601605420631345037007174534028
446276579121785903419 07
187290750733961517636405719711712813459994640811460231
674578431711352163602 65
243503771582711646656491043218101025859266779335887957
898765222719851338193 06
106076916854973037409951958014787419948942371734492437
313471916359677785322 440
714800232292032720677819043480932936388853505644066022
248644107423085598937 04
105154718711064354028748869126594015389396954282301062
110412208928769290991010
```

```
460520427773918592257741135990834929873590953856621755
282639065080619049740 80
105740609880746100379772054620831674148779875260904133
824929315742181096728 799
925269845792114279931610864772442493059116931300088153
552593289556664783724 89
109989407076374077753958106442558533203931749533788508
593876352498223445198 119
113288257314817536317107315194625619259015427369427517
955349846973319281713 361
255981794841046341164477987355429184482873629454682280
509829627065074686047 04
558639230193641633767112362350102352248343369162592740
649981633061531222394 70
177295156389396378684407003323941433935496449233174310
218053525005527601449 93
429997134606210621932009870916464453896029348177426335
191857432251506730874 15
170630855404683748152979223788591722284630583811997576
572572838995198209757 44
439251761040079108839204209369279817333173998205021225
428856774004085478506 80
763722407104006263101285747667899659754007425808759390
714622260310091171000 49
565282059285569523870830937830317078134886985981333550
189054198708158181437 80
```

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 .

```
[*]A public key:
    ['0x49733ec331f398bd03839edd144fd86122954225daa521fc25b71ebca5e5fffee9b71d825537745fc821d98ea4044f2da40fb1dc2
b9b3b15b7b304adf303775', '0x3d4c6f1a22fdc1ac15b930a0ded38d577fb410eab7dbad76581472b43657da2baade71666befdd282f312
598697f1164fefd0bbf1feb33abe7f803bf8ccb3ed']
[*]B public key:
  ['0x9c34a1d007e385beab60fd14e46f93d7ee3661f7e2581525a583466a709c1e3fe09edaf0d2cfbf9d2e4b9df2bb696cec11b16740a8c
f9eb790edf7b4f7821289', '0x78900073c8c4cb9596493eea76b0f0226eef4156817074a73595f439eeebecee6ffc95892ee10f0b09f815
9926343cb6cf1148176dd35f5d8284e325bd2e6b9f']
[*]A private key:
    ['0x86604154f9675cc3bc65d0a46c05b0500a46ed58612ae6bdafc13fc7e53b891', '0x8bee1a4b5527c58c62d32e04e41d8459ee93
22b2bd8a3f1d3c4413621cc722a5', '0x44f6fd25aa0da221e756e7307c89846e95628b23f4a7de8874297df2a4e929be683da34d7e93688
e53c14c53c32bba6911700995cd6d725e9a8c41a70179765']
[*]B private key:
  ['0xb547ac43b8c47e1b6c4fc60b50f36c9e3fd47ab4cfff159ee3edc7117d6726f1', '0xdc97283e2b9d3a5627c3428ec9257ab47a5ab
fba83939dba540ff2364d159519', '0x7aa3cc800260db7f58639d52ae8377804ff809cba4e273ec2c3f36af224c264b3175cc0022d2a746
22f2d76c4488d6b5e93213466f994889c97f72aab21366df']
```

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 М і знайти криптограму для абонентів А и В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.

Реалізовані функції:

```python
def encrypt(msg_encoded, pbk):
	n, e = pbk
	return __horner_pow(msg_encoded, e, n)
```

```python
def decrypt(msg, prk):
	p, q, d = prk
	n = p * q
	return __horner_pow(msg, d, n)
```

```python
def sign(msg, prk):
	p, q, d = prk

	return __horner_pow(msg, d, p * q)
```

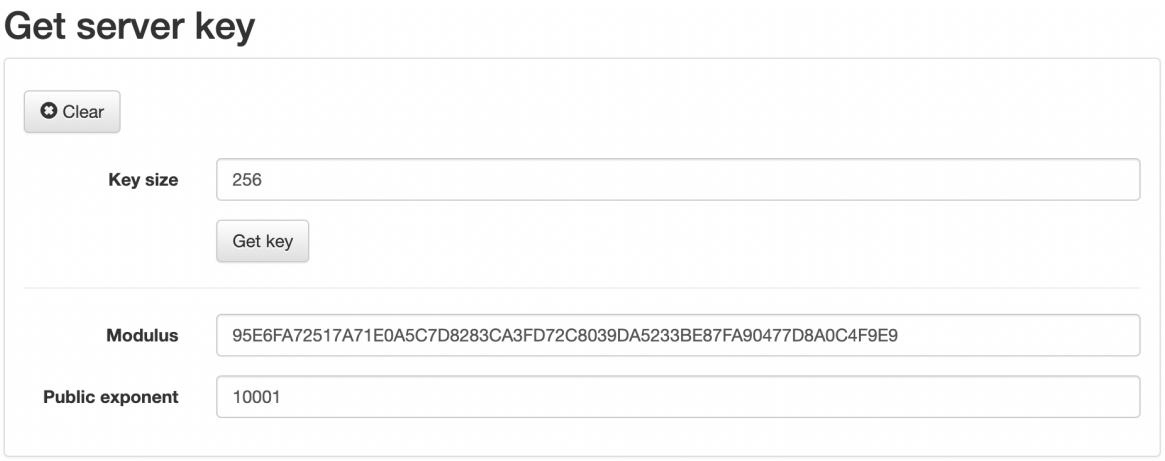
```python
def verify(msg, pbk, signature):
	n, e = pbk
	sgn = __horner_pow(signature, e, n)

	return msg == sgn
```

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.

```
[*]A and B exchange public keys:
[*]A wanna send msg:
    0xd2bea560f3e2e4109c98f5aabea310b3adcf2dd3be298e4008995f2b8b22338a
[*]Encrypted msg:
   0x1798b42143678242608fe67c60c477b6db177ffb59e440fb7dd33544375af820cb7fb5599044b2186d1ee4d1e82cee24ef4e067fb8ac
62ae9583b4b117ee77ea
[*]Signature:
    0x2e476b02dee6d1c15bbac00e6673d936dc7acc521587da3407cf27795700bd2e04d7de9c574f0603f8dfa413cc7c0b5cf23170375d4
0004e2541210f2e9a9b5
[*]B receive msg
[*]Decrypted msg:
   0xd2bea560f3e2e4109c98f5aabea310b3adcf2dd3be298e4008995f2b8b22338a
```

Перевірка на сайті:

1) Створимо публічний ключ серверу та додамо її в наш код:



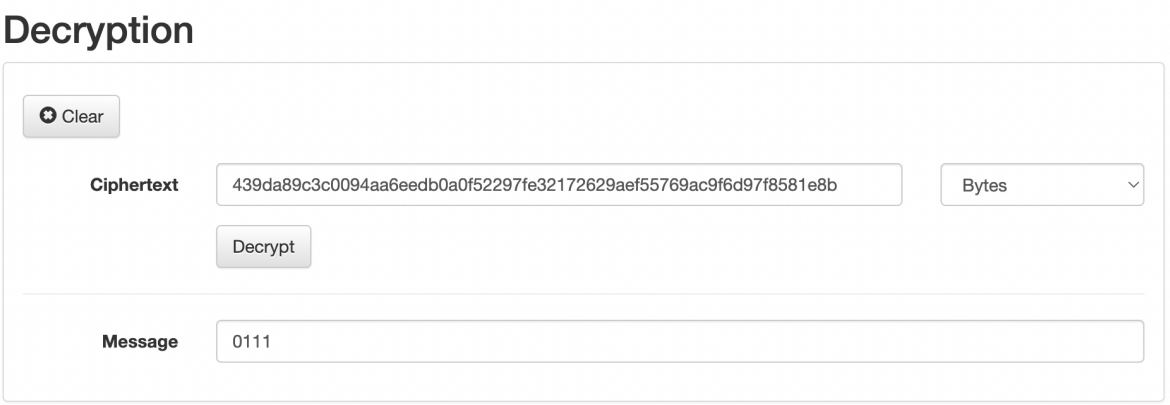

```
server_pbk
=(int("95E6FA72517A71E0A5C7D8283CA3FD72C8039DA5233BE87FA90477D8A0C4F9E9",16)
,int("10001",16))
```

2) Зашифруємо число “111” за допомогою цього публічного ключа

```
msg=int("111",16)


encrypted = rsa.encrypt(msg, server_pbk)
signature = rsa.sign(msg, prk)
print('[*]Encrypted:\n  ',hex(encrypted),'\n[*]Signature:\n
',hex(signature))
```

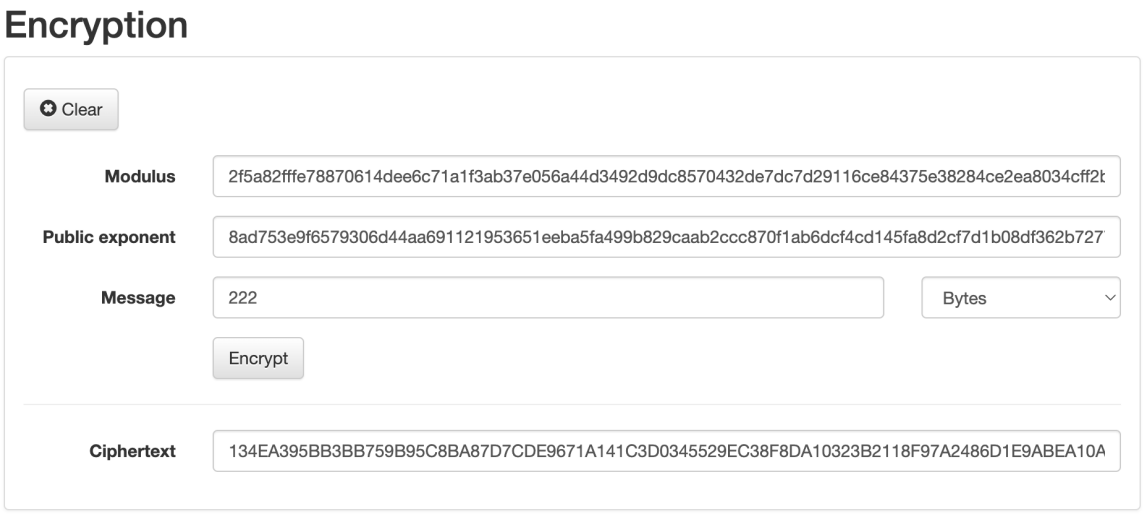
```
[*]Encrypted:
   0x439da89c3c0094aa6eedb0a0f52297fe32172629aef55769ac9f6d97f8581e8b
```

3) Розшифруємо на сайті (розшифровка відбувається за допомогою приватного ключа який ми не знаємо)



4) Тепер зашифруємо деяке число “222” на сайті за допомогою нашого публічного ключа (завчасно згенерованому)
наш публічний ключ:

```
pbk =
(int('0x2f5a82fffe78870614dee6c71a1f3ab37e056a44d3492d9dc8570432de7dc7d29116
ce84375e38284ce2ea8034cff2bc9707029df750ee2ea330d8c4c4b8229f',16),
int('0x8ad753e9f6579306d44aa691121953651eeba5fa499b829caab2ccc870f1ab6dcf4cd
145fa8d2cf7d1b08df362b7277f28953dfe9b120d75b8f6353bad1ba3',16))
```

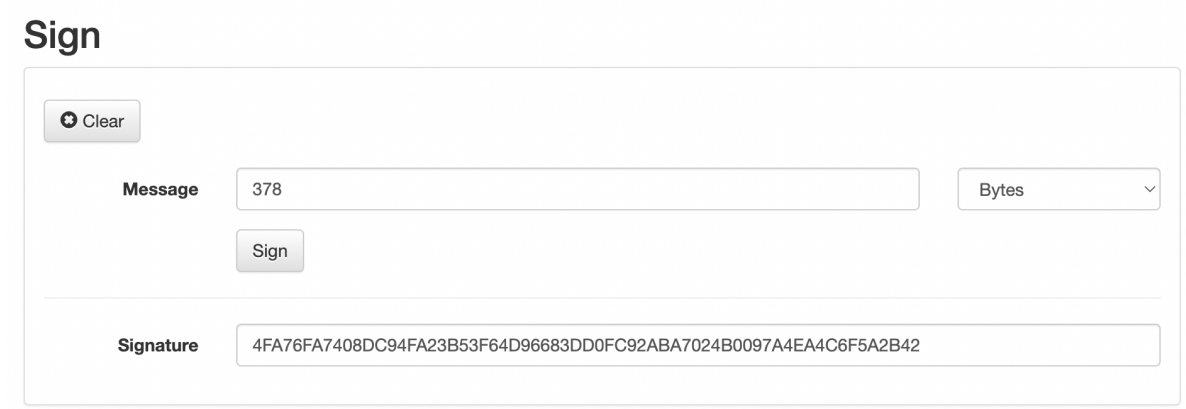


5) і розшифруємо за допомогою нашого приватного ключа

```
enc_msg =
int("134EA395BB3BB759B95C8BA87D7CDE9671A141C3D0345529EC38F8DA10323B2118F97A2
486D1E9ABEA10A41C65E0705D6FA2843AA20B8AEAF9CB473E3A2FA23F",16)
dec = rsa.decrypt(enc_msg,prk)
print('[*]Decrypted:\n  ',hex(dec)
```

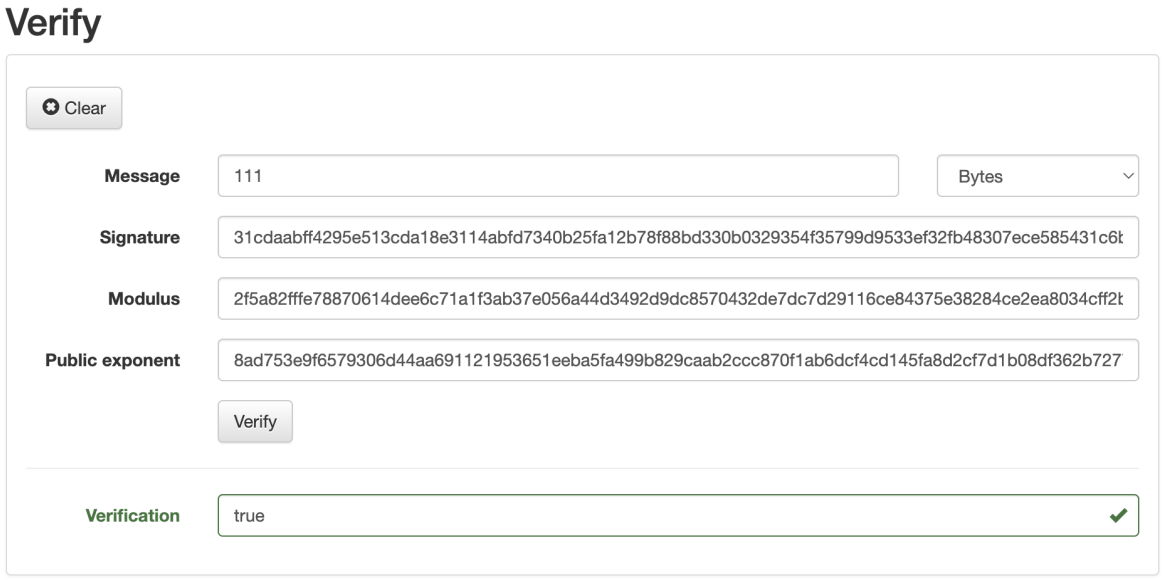
```
[*]Decrypted:
    0x222
```

6) А тепер верифікуємо в себе електроний підпис серверу (щоб впевнетись що ми отримуємо повідомлення від серверу) і навпаки, верифікуємо свій електроний підпис на сайті




```
server_sig =
int("461EE1A380771F48D90465996A9C4FD5A1C98DED4ECFA421070AC03BF1D29559",16)
sig_msg = int("333",16)
verify = rsa.verify(sig_msg,server_pbk,server_sig)
print('[*] Is verified: ',verify)
```

```
[*] Is verified:  True
```



Висновок: ми знайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомились з системою захисту інформації на основі криптосхеми RSA, організували передачу повідомлень з використанням цієї системи засекреченого зв'язку й електронного підпису