Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Криптографія

Лабораторна робота №4

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

Виконали:
студентки 3 курсу ФТІ
групи ФБ-93
Шрейдер Марія
Жембровська Олена

Перевірила:
Селюх П.В.

Київ - 2021.

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 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.

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.

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 .

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().

Хід роботи

Протокол роботи конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA

```
def protocol():
    Mabel = Node()
    Dipper = Node()

    Dipper.GenerateKeyPair(0)
    Mabel.GenerateKeyPair(Mabel.n)

    print('\nMabel keys')
    Mabel.PrintKeys()
    print('\nDipper keys')
    Dipper.PrintKeys()

    k = random.randint(0, 2**256)
    print('\nkey', k)
    pack = Mabel.SendKey(k,Dipper.e,Dipper.n)
    print('Mabel send ', pack)
    result = Dipper.ReceiveKey(pack,Mabel.e,Mabel.n)
    print('Dipper receive', result)
```

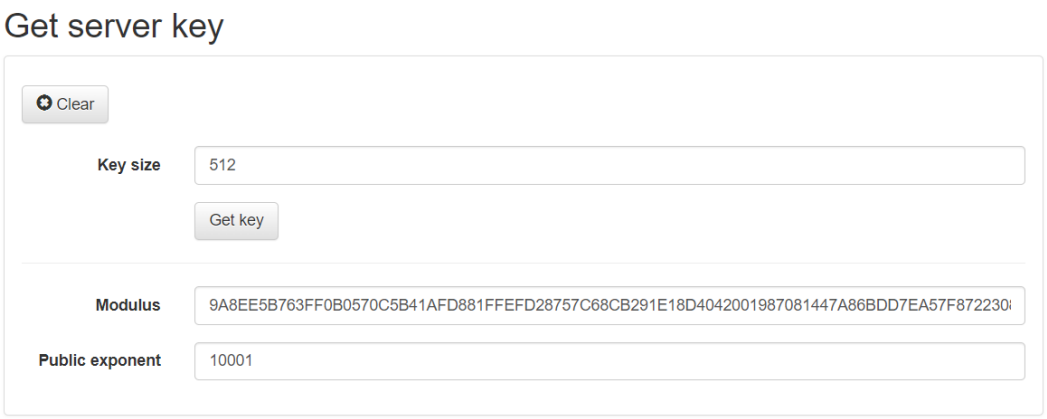
```
p 67687881910149923459484130871727592948891326099908625404631514557378058035167
q 92730195380697678842940035626534079060157788032090046978227526502335029597897
p 60307148030481693803001661434110619314611727973806071836838291651034651223797
q 88453144808326250731410787499287510899025544329916239754966882728770056792803

Mabel keys
e 41187855360479646930369081542960329200158000364615600273964736523120676209302235252058250266240560830171466586958308330733055103267013497174521663474270l
n 53343568977173645111042654693099263360735352107157326224503217665031683250806957082696155908149268059306728530241733254781560878002686570762328168119329911
d 11219174132747209191652144218295305825252853681142390711882245362948622664258969170678050887031126159369473337038594254510471385059572858711787742189535773

Dipper keys
e 29385266762605177968764375510128634449430005687846012063955263393477193222614414674901639458881266973187868971983655155473721785027053131065869476546174077
n 62767105144337944106221441368705634740394988450841797522122673129450220163443669673157220976872271660776388337629733205748774091552330132139667298952437999
d 55186649046245084690240248778845982222002525282313081175673434508355214127574223910759087246870911787970927888858356569683577661005796454041979995283486871

key 61530740377693162910734608932583457497317322294478869887170664355655721832796
Mabel send
 (28581080211474984287934866954933368714955232965396561305401554334530111802057740706312313894233177250154953580249364070341290451561547483352782832830622211,
  34123767015460380809950498390877523307754720247255587228673296720228909079093549286789918287176309185718383931128686841683644806874747235476988180257600041)
Dipper receive 61530740377693162910734608932583457497317322294478869887170664355655721832796
```

Функція check()

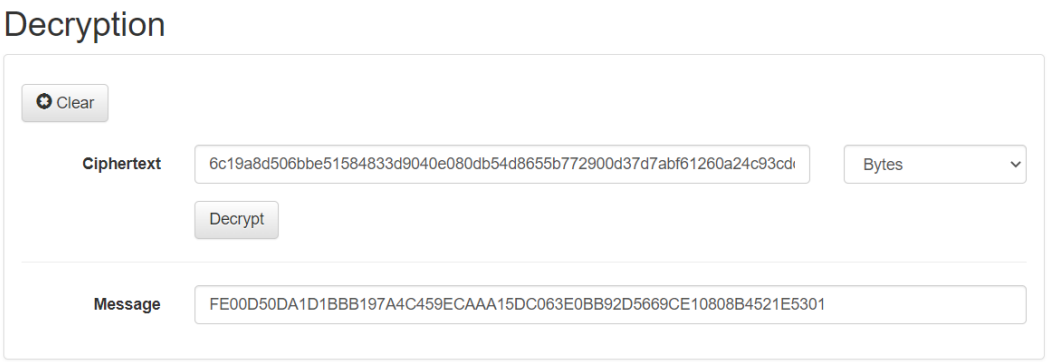
Перевіряє роботу програми, шляхом взаємодії з тестовим середовищем http://asym-crypt-study.herokuapp.com/

```
def check():
    A = Node()
    server_n =
int(0x9A8EE5B763FF0B0570C5B41AFD881FFEFD28757C68CB291E18D4042001987081447A86BDD7EA57F8722308B5ECB695F1B8B57702331C834EA76773224FA51693)
    server_e = int(0x10001)

    #генеруємо ключі, які будемо використовувати нижче
    #A.GenerateKeyPair(server_n)
    #A.PrintKeys()

    A.e =
10852746501111419120795960929289968488613208275606288276151660071245230394717794773237389989976950293838562598240507342259497868077712481899388858960780149
    A.n =
11729174846780466515621400509184851154650274595662807386553577455135392784735642578294635291713690950972273380803161955115964315406336833363293791988506049
    A.d =
10395809668393900846290703319055430003363574478006090934093666592738743330019249614087926613845539922756948307649748442660101450656799819642150514339729889
    print('\nmy keys')
    print('e ', hex(A.e)[2:], '\nn ', hex(A.n)[2:], '\nd ', hex(A.d)[2:])

    msg = 114888933979642393893806692060425606139553765450567466106219951259984737489665
    print('\nmsg ', hex(msg)[2:])

    print('\n-encrypt here with server keys, decrypt on the server-')
    print('encrypted msg', hex(A.Encrypt(msg,server_e,server_n))[2:])

    print('\n-encrypt on the server with my keys, decrypt here-')
    emsg =
0xD6A64F78722422EAB4E0AC03BFEB8029ACF0CB50C929BF05D09C707BC22F7C768EE7D50197CF0C7B2017EAE1E393506BD807D5D037FF57DB909120659FF3ADA9
    print('encrypted msg ', emsg)
    print('decrypted msg ', hex(A.Decrypt(emsg,A.d,A.n))[2:])

    print('\n-create a sign here and verify on the server-')
    print('my sign ', hex(A.Sign(msg))[2:])

    print('\n-create a sign on the server and verify on the here-')
    server_sign =
0x2A777F1355107E54D60F3C9B1EEEA89489DFCB5E88CB99A85F42C025C086D90B86CEDD6E77E209772D211AB4876B31F4AB17DDA9266EB3CED5E1618F14269335
    print('server sign ', server_sign)
    print(A.Verify(msg,int(server_sign),server_e,server_n))
```

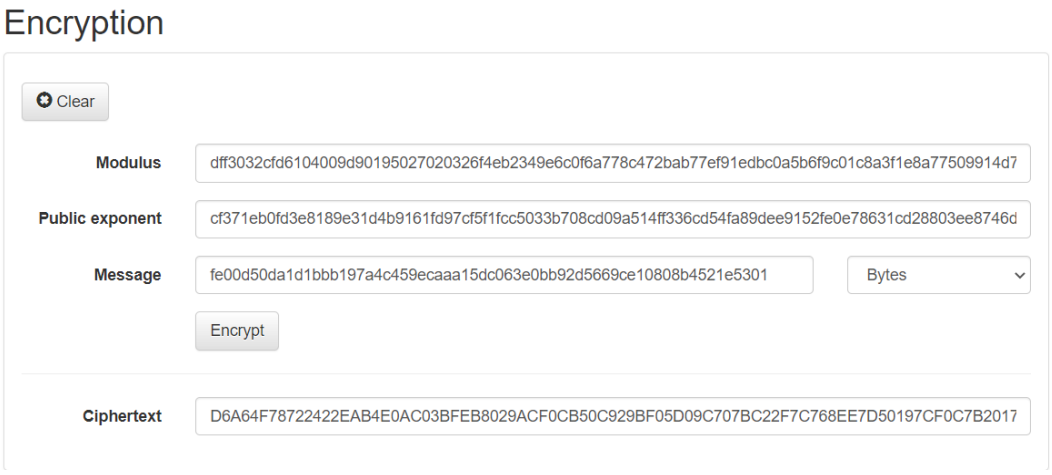
Генеруємо ключ на сайті



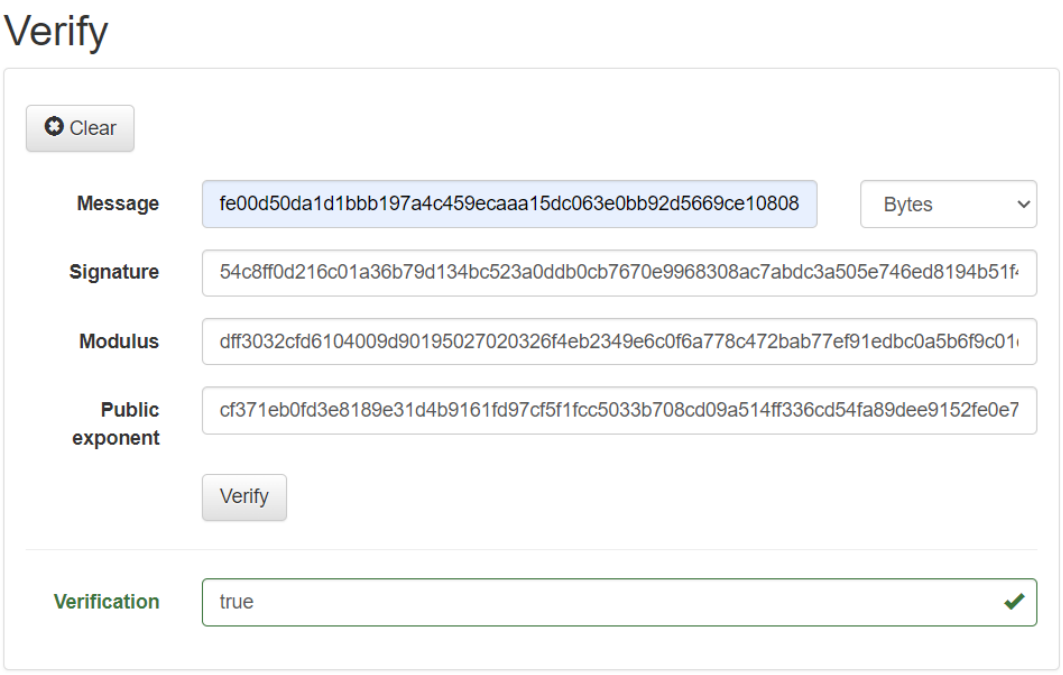
Розшифровуємо, зашифроване програмою повідомлення з ключами сервера



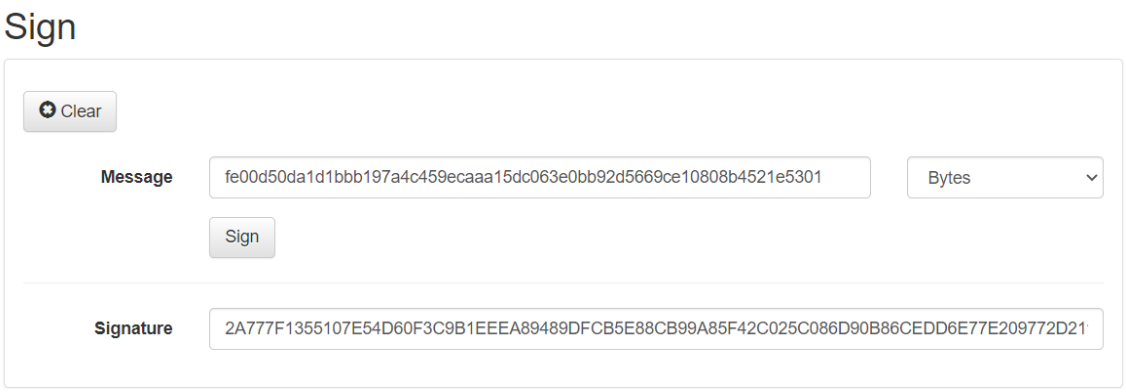
Шифруємо повідомлення, яке потім розшифруємо в програмі за допомогою свого секретного ключа.



Перевіряємо свій підпис, сгенерований програмою за допомогою нашого секретного ключа.



Отримуємо підпис сервера, який потім перевіримо в програмі.



Вивід програми

```
my keys
e  cf371eb0fd3e8189e31d4b9161fd97cf5f1fcc5033b708cd09a514ff336cd54fa89dee9152fe0e78631cd28803ee8746d436b63eb877babacea79b07ea311b75
n  dff3032cfd6104009d90195027020326f4eb2349e6c0f6a778c472bab77ef91edbc0a5b6f9c01c8a3f1e8a77509914d7e3a80ce8a81b718bc9064aa518f85dc1
d  c67da8bfa737c507e40c1bbe9de6055d8c8fa3a910de42917dc3116512c4d2b2be5b4c0c5e4ef719b0253d5e7c86a0eaa187c2c51b1657276d3ca09f8dd9d1e1

msg  fe00d50da1d1bbb197a4c459ecaaa15dc063e0bb92d5669ce10808b4521e5301

-encrypt here with server keys, decrypt on the server-
encrypted msg 6c19a8d506bbe51584833d9040e080db54d8655b772900d37d7abf61260a24c93cddc89ea1cdfa133e368450e221c06f76214bcb967805b242f351c3165677e8

-encrypt on the server with my keys, decrypt here-
encrypted msg  1124211437923769483937803760733292404456032218822189864540909163778299064181140340424792216392087698486810214665959975678530783
decrypted msg  fe00d50da1d1bbb197a4c459ecaaa15dc063e0bb92d5669ce10808b4521e5301

-create a sign here and verify on the server-
my sign  54c8ff0d216c01a36b79d134bc523a0ddb0cb7670e9968308ac7abdc3a505e746ed8194b51f4e6ebec33eef9613a19cad29663c16e861667cc83a15fe22815cd

-create a sign on the server and verify on the here-
server sign  2224165885791828946911743887072903437385214029664079973850822541214149837463805774508939823532078922047808374125576381572930647996
True
```

Висновки: в ході виконання лабораторної роботи ознайомились з тестами пошуку випадкового простого натурального числа та реалізували один з них, тестом Міллера-Рабіна. А також реалізували протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника за допомогою криптосистеми RSA.