

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

## **Криптографія**

### **Комп'ютерний практикум №4**

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

**Перевірила:**  
Селюх П. В.

---

**Виконали:**  
студенти III курсу  
групи ФБ-95  
Гурджия В.  
групи ФБ-94  
Золотов І.

## 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:

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  $q, r$ ,  $p_1, p_2$ ,  $q, r$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p_1 q r p_2 \leq p_1 q$  – прості числа для побудови ключів абонента А,  $p_1 p_2$  – абонента В.
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ ( $q, r$ ) та відкритий ключ ( $e, n$ ). За допомогою цієї функції побудувати схеми RSA для абонентів А і В – тобто, створити та зберегти для подальшого використання відкриті ключі ( $n, e$ ), ( $p_1, p_2$ ) та секретні  $d_1, d_2$ .
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  $M$  і знайти криптограму для абонентів А і В, перевірити правильність розшифрування. Скласти для А і В 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 виконання роботи:

Робота самої програми:

```
For A
private key:
d: 6029736401173757717827639103823301614850510273438013244111087164415022918984995514453996385169526465390871385268750394460581406858144797733221850205009021
p: 7271829197274482453877753233432182808941884357672733152904186093330489550321
q: 88955769248589846188968710732889288229594312404793609568470034061083049157383
public key:
n: 6468711600879071924246179718821953919591843743939493634350697450186385043279724187800570288624698629770812482000126607187513919644167967236878045627169943
e: 513507430985236079477804952153842213238631583702944311559221585861093979126822630499921346144595460953527859334132433340415661256609370734969774372685781

For B
private key:
d: 4601085647480492657947095296878563367278182137523418723471505907328090249655436540842811078546182367770495151101976951808320089859234247715305450586203435
p: 95866094104187888817657844814438194115050386349941005387381897087459370924707
q: 70341896766479388095142780808013861067646827605546582753265100020889087133303
public key:
n: 6743402894882382787720559119766957462680356799048169394536805829497662822650452556041555070433401295361681545379116787800969978649665261361071494885217221
e: 1941550152640427917055712967100031486614845878804833674054779258416829052040445572481602170490838604872886905776053593511592838867378347993469513480552183

Enter Message: 969
Encrypted Text: 2413413451415367782176768562071932070029259219430038455659672455907596890106106605654559773002616080337674779831744978854722460439135524535917397942271555

Signature:
M: 2413413451415367782176768562071932070029259219430038455659672455907596890106106605654559773002616080337674779831744978854722460439135524535917397942271555
S: 3845509602928914415424453606567568874160850196100085685305266002562292308460384163502953853234801466245520903195977874904972956974540752814989189954120181
Signature Verified :)
Decrypted Text: 969
```

Генеруємо значення  $p_1$   $q_1$  і  $p_2$   $q_2$ , далі передаємо у функцію RSA і за допомогою неї ми отримуємо приватні та відкриті ключі. Потім проходить шифрування і розшифрування і перевірка цифрового підпису, і як бачимо перевірка пройшла успішно.

Наступний пункт буде перевірка через сайт коректності роботи програми.

## RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

### Get server key

Clear

Key size256

Get key

ModulusB6BF58FA6F86D27142A6A3720C195835912CA7851645EFA7A93896D43C0D1627

Public exponent10001

## RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

### Encryption

Clear

ModulusB6BF58FA6F86D27142A6A3720C195835912CA7851645EFA7A93896D43C0D1627

Public exponent10001

Message123456789

Bytes

Encrypt

Ciphertext6FF7BC2A5269F88193BC69BD2B2DD2A2A1C33A4DFC42F8A29D607613C322A5B9

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Sign

Clear

Message

123456789

Bytes

▼

Sign

Signature

83820C75DCC1BC6B0A4E60F07ED75DA78698D50D1A7F76BF98DCABC81BE91C2B

```
Verify Site:
Encrypted Text: 50644436087052988430619912332573088492047365257577971552923239288615565698489
Encrypted message: 0x6ff7bc2a5269f88193bc69bd2b2dd2a2a1c33a4dfc42f8a29d607613c322a5b9
Sign: 0x83820c75dcc1bc6b0a4e60f07ed75da78698d50d1a7f76bf98dcabc81be91c2b
Signature Verified :)
```

Наступна перевірка через сайт.

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Encryption

Clear

Modulus

4be038721ce8600afc0ea13ce219b31aa87b56c3b781b1369592c42855f5c238e9b803d5ae9e52f850e4366550a21

Public exponent

7bbf8582c26de211e4d71a06be15dbceb97f3aa3dad46d0e6d91e40e02fe5fbbba05b6232397a1f61f546036183040

Message

1311977

Bytes

▼

Encrypt

Ciphertext

23C7AEC6CF9483371F0D64446D9BF8D6E6F7450890F9C9F552CFCFF86C7B70E5579EC15637DD7E0871A;

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Verify

Clear

Message

1311977

Bytes

▼

Signature

593ffea901851d78c3e1670613de35fa442f3bd14d49247c2fc7305a517a29b3a2b6a4cf07edb107c6f5ba69184fc47

Modulus

4be038721ce8600afc0ea13ce219b31aa87b56c3b781b1369592c42855f5c238e9b803d5ae9e52f850e4366550a21

Public exponent

7bbf8582c26de211e4d71a06be15dbceb97f3aa3dad46d0e6d91e40e02fe5fbbba05b6232397a1f61f546036183040

Verify

Verification

true

✓

```

Signature Verified :)
Verify Site:
private key:
d: 3262714192779016818512247387835914243891521932280628588159380230871236857130804984864930344023364863137873577586050403909266602850608845665688983168360267
p: 66841549772294029533547681096809318815954044661547181014529770260901008576383
q: 59453159316964042068509888014655455975856314121276202005536238285437547091473
public key:
n: 3973941307604978526558508421376429280876190211271896718499894932414965853194045562156722328897359683912409455968880417502881485972113040098123610008482159
e: 25317248102399431051572429459966774854245920800713468268022660394736144561463386305584925930398760034486385889271403266642218647342803565184276784026275
hex(e): 0x7bbf8582c26de211e4d71a06be15dbceb97f3aa3dad46d0e6d91e40e02fe5fbbba05b6232397a1f61f54603618304076f01c9826af8915c7c0e5f061c512a3
hex(n): 0x4be038721ce8600afc0ea13ce219b31aa87b56c3b781b1369592c42855f5c238e9b803d5ae9e52f850e4366550a21fd5fc61771b3d4b8e8bbe541ec5c6e2716f
enter C: 23C7AEC6CF94833371F0D64446D9BF8D6E6F7450890F9C9F552CFCFF86C7B70E5579EC15637DD7E0871A7D2D30B552A285F5EC314E6C7D995D1BB3E89BC3349B
Decrypted Text: 19994999
Decrypted message: 0x1311977
Signature:
M: 19994999
S: 292150044833404561359135444451721792420728609713123988256603415303117433577933688258447320761238481785923065023034684363442446460928834945848059110598777
signature: 0x593ffea901851d78c3e1670613de35fa442f3fbd14d49247c2fc7305a517a29b3a2b6a4cf07edb107c6f5ba69184fc47464ef5b4b75b6cc2b1390d4ba8fa879
Press any key to continue . . .

```

## Висновок:

Ми навчилися використовувати RSA для шифрування і дешифрування текстів і повідомлень а також використовувати і перевіряти цифрові підписи. Також отримали навички роботи з алгоритмом Міллера-Рабінна і схеми Горнера.