

**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**

Варіант 4

Виконали:

Студенти ФБ-05

Берега О. А. Ковбель Д. О.

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. 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  $1 < p, q$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p \nmid q-1$  ;  $q \nmid p-1$  ;  $p \nmid q$  – прості числа для побудови ключів абонента А,  $1 < p < q < 1$  – абонента В. 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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі  $(e, n)$  ,  $(n, e)$  та секретні  $d$  і  $d_1$  .
  1. 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 и В, перевірити правильність розшифрування.
- Скласти для А і В повідомлення з цифровим підписом і перевірити його.
1. 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.

## Хід роботи

В роботі ми використовували функції з бібліотеки random: randint(), randrange(), аби згенерувати число розміром 256 за завданням. Перевірка на простоту, здійснюється в декілька етапів: ділення на відомі перші 13 простих чисел(якщо випадкове число ділилось на них відповідно воно не було простим); тест Міллера-Рабіна(в якому знайшли розклад нашого числа, а потім перевірка простоти за основою випадково вибраного числа x). Реалізована функція test\_millrab() повертає True/False значення у відповідності чи є число сильно псевдопростим, чи ні.

### Результати виконання:

Keys for Alice:

Open Keys for Alice:

e =

1520188411089749086588356197963003471329963876787546165202182583361245891888828  
975369532472140628660313313672279511612432268571677242053323170155711181989

n =

5131840839399378920948186496443110110286192423767372829154563295794505909320912  
318455704057852738989389634829469567060751702008328757598001677986669400413

Secret key for Alice:

d =

4520566060774125141993329159705093317044048027884319394090034853823308154832975  
154420378337823768929845766981578175777689514853696507817491278934493027469

p =

71803258844346529515225014083482695735913915220461279405209551052827658333947

q =

71470862492802265514062467637545951677637419918222972619441030833536121663879

Keys for Bob:

Open keys for Bob:

e1 =

5615155250541475272222034832959503995439091817509057978292838860746153969287720  
550032428095690120698174881263280359738804058005009959645665778719191281931

n1 =

8723848363757664555956258811776813777062671802106940098594845003461139599097939  
466789546847280002418031439135602754827164380634647511535349019865234823557

Secret key for Bob:

d1 =

2451073639877136378373449654359806760368968383024689860208459061338923387973286  
731074909088989150185201802148930462909901516761145783217167562123934039051

p1 =

81167929424041692171444899034362152914067872376891205535766212507886022717647

q1 =

107479005879059511671108287151410921789636890258334614308579113820271880199531

```
Open Keys for Alice:
p = 719185040267875032639991217183645158320888768513711343876828944836316474461503806671703916782588543247458746614871723085345137750914934030564
q = 940276912764339885472702997836446613446229411973306268394949901741952202999485696945481828174702495320279873122039638952014239350794931281170
Secret key for Alice:
d = 886976051823967206100749420123464784653099001096797737288484235219077017966534448139468108497087648936916939116160627953262371088925750306998
e = 85308054440195894882564054977679000108625210253133581861372610102080509946271
f = 110221352360521692561651559070489544449416109503195929942712001452770291253909

Keys for Bob:
Open keys for Bob:
p1 = 47364885164630866424782809540406026665123175779630216250740535880504531951381482435745601843518458913370011061002161128086196947674783585946
q1 = 9649797397697071250399106730933498788485640517235362358058541573584960127515592140918195919173867541155708139853521991451073703985849076625
Secret key for Bob:
d1 = 82994451056395374589807466818279389542522432995641912095914904469186161135434943788894606460321539111759020480441173644979273876398896254299
e1 = 99240001509671376480169565921135390858940201826421457569403028376312347001443
f1 = 97236973507670250003111431819924360401115809082834052977831243720327853188963

Beginning key= 16456158630107189330863348540720343140356636339124942459913611191507075600130651884895547840470514513639484702891366844013767348124

Message: 5232301083891101740561287117316618677367254054551317093097880232732789510910891220199273014089961729772235130772551317068953267726238254

Comet 44100744310569991473278141226684172310589422838060311925039779891060979653783794424005874801464784616806590580716270412269895037976598756510
Decrypted key = 164561586301071893308633485407203431403566363391249424599136111915070756001306518848955478404705145136394847028913668440137673481

Key recieved

Decrypt: 4410074431056999147327814122668417231058942283806031192503977989106097965378379442400587480146478461680659058071627041226989503797659875
Encrypt: 52323010838911017405612871173166186773672540545513170930978802327327895109108912201992730140899617297722351307725513170689532677262382541
Miller function: 940276912764339885472702997836446613446229411973306268394949901741952202999466144004801756415958073758875056267583834820038063996
Text check: True
```

## Висновок:

При виконанні лабораторної роботи номер 4, ознайомились з декількома способами генерації простих чисел, що й так вже були відомі, але вперше з практичним використанням. Окрім цього, було засвоєно інформацію про систему захисту інформації на основі RSA, і також не забули про протокол розсилання ключів.