

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**

**Виконали:**  
Студенти III курсу  
Групи ФБ-95  
Пашинський М.О.  
Бурчак Б.Ю.  
**Перевірила:**  
Селюх П.В.

## 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  $q, p$ ,  $p_1, q_1$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p_1 q_1 \leq p < q$  – прості числа для побудови ключів абонента А,  $p_1 q_1$  – абонента В.
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ  $d$ ,  $q d$  та відкритий ключ  $e$ . За допомогою цієї функції побудувати схеми RSA для абонентів А і В – тобто, створити та зберегти для подальшого використання відкриті ключі  $e_A, e_B$ ,  $d_A, d_B$  та секретні  $d_A, d_B$ .
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:

Для початку потрібно було реалізувати генератор випадкових простих чисел розміру 256 біт. Для цього було написано функції:

**Generate\_Random\_Simple\_Num()** - що генерує випадкове число розміру 256 біт.

**Miller\_Rabin\_Test()** - що перевіряє згенероване число на простоту.

Наступним етапом була генерація пар випадкових чисел **p, q** (для Аліси) та **p1, q1** (для Боба), що потрібні для генерації відкритого та приватного ключів для обох абонентів.

**Generate\_P\_Q()** – генерує p, q та p1, q1

**RSA\_Key\_Gen()** – генерує ключі

Отримані параметри записуються в файл:

```
Pair for A:
p: 90708139651171628521232124273227547177346217781423756111247506238698070547837
q: 69566688813520377697879885024586665621243724118908870864034909974094452652647
Pair for B:
p1: 112073977340162542552662738928888367198598165969755897719223786608606282998547
q1: 85601641391936357891585261876706867992620076350667944010392771273899070821367
A Public key:
e: 1593609335864786365937483709963217862265130443027889099543435345531761924154913260449715584697769244348342623657127607090935773025991525368154830613635179
n: 6310264923966405545081634361743364697147281305733994345722743220237583762655046555180171749847431139074099118261555279802159059154451869753201949158174539
A Private key:
d: 4522213385477028961123614658200377460290674310063362232363147357633911629627446226121638658575859302062910972443461941395340480380557896798631884759887691
p: 90708139651171628521232124273227547177346217781423756111247506238698070547837
q: 69566688813520377697879885024586665621243724118908870864034909974094452652647
B Public key:
e: 527591614076627887274032297325005264525943581316156640341196425542330879325385642962646531723936336879469490971753749367478183608539205572833424870679687
n: 9593716417640595342367622524208473025050354835400421605791321503930938086326213981721147604562565834718011766807012406521751921654491148546951498957553749
B Private key:
d: 4043872437711644014070494210451681590349933062601195739876694460432256395587984772102038380587765184343257725280970017452056760653142606613729102021881759
p: 112073977340162542552662738928888367198598165969755897719223786608606282998547
q: 85601641391936357891585261876706867992620076350667944010392771273899070821367
```

Далі були написані функції:

**RSA\_Encryption()** – шифрує повідомлення

**RSA\_Decryption()** – розшифровує повідомлення

**RSA\_Sign\_Gen()** – генерує ЦП

**RSA\_Verify\_Sign()** – перевіряє ЦП

Наступним кроком була реалізація обміну повідомленнями між Алісою і Бобом. Для цього було написано функції:

**Send\_Message()** – ф-я приймає відкритий ключ одержувача. За допомогою цього ключа відправник шифрує повідомлення, підписує його та передає отримувачу набір даних (**шифротекст, ЦП та свій публічний ключ**).

**Receive\_Message()** – ф-я приймає відкритий ключ відправника. За допомогою цього ключа отримувач перевіряє ЦП та розшифровує повідомлення своїм приватним ключем.

```
-----
A --> B
Message to transfer:
4365561660646124106916620765706448782813780097007055890069238212397218543191109883305233835178375324676609206941782296348036437669863851635084922906743970
Message after encryption:
1312774379950029068949495126281332771596174655949135733949654477589981505683927997830565563046035032745624210332294425607149094356128624699449321612747478
Digital Signature:
5916202516484008480409937162751182919551877179530819121455794122847510658910711709907207110463978539727751189935059313745896037998786085999418160462230604
Signature Verification successfull
Message after decryption:
4365561660646124106916620765706448782813780097007055890069238212397218543191109883305233835178375324676609206941782296348036437669863851635084922906743970
-----
```

Останнім завданням була перевірка правильності написаних в ході роботи ф-й за допомогою сайту <https://asym-crypt-study.herokuapp.com>. Для цього була написана ф-я `Server_Check()`:

RSA Testing Environment

Отримання значення ключа з сайту

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Get server key

Clear

Key size

256

Get key

Modulus

84B29D74642F5E9CFA52E4677818722C6220A71C64FBB569610602B74356B603

Public exponent

10001

```
# server key
e = int("10001", 16)
n = int("84B29D74642F5E9CFA52E4677818722C6220A71C64FBB569610602B74356B603", 16)
serv_key = e, n
```

RSA Testing Environment

Шифрування на сайті

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Encryption

Clear

Modulus

84B29D74642F5E9CFA52E4677818722C6220A71C64FBB569610602B74356B603

Public exponent

10001

Message

666

Bytes

Encrypt

Ciphertext

6D38B4C0A33C83D251190A75BB7EA940B65A0DE9904348134C756C0FE064C9CB

```
# encryption
M = int("666", 16)
C = RSA_Encryption(M, serv_key)
print("Encrypted message: " + str(hex(C)))
```

Encrypted message: 0x6d38b4c0a33c83d251190a75bb7ea940b65a0de9904348134c756c0fe064c9cb

Digital Sign: 0x23991792ce76804af686d5ca8821878bf8841c846df2b79bf9162747835949dd

Signature Verification successfull

PS D:\КПИ\3 КУРС\КРИПТА\LABS\LABA\_4>

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Decryption

Clear

Ciphertext

6D38B4C0A33C83D251190A75BB7EA940B65A0DE9904348134C756C0FE064C9CB

Bytes

Decrypt

Message

0666

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Verify

Clear

Message

666

Bytes

Signature

23991792CE76804AF686D5CA8821878BF8841C846DF2B79BF9162747835949DD

Modulus

84B29D74642F5E9CFA52E4677818722C6220A71C64FBB569610602B74356B603

Public exponent

10001

Verify

Verification

true

**Висновок:**  
В ході виконання даної лабораторної роботи ми ознайомились з методами генерації ключів для асиметричної криптосистеми типу RSA та алгоритмом перевірки чисел на простоту Мюллера-Рабіна, навчились шифрувати, підписувати, розшифровувати та перевіряти цифрові підписи за схемою RSA.