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

Криптографія

Комп’ютерний практикум №4

Виконали:
Студенти групи ФБ-33
Назаренко Іван, Тимощенко Олександр

Київ – 2025

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.

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.

Генерація ключів A

```
==============================
   ГЕНЕРАЦІЯ КЛЮЧІВ A
==============================
Кандидати, що не пройшли: 133
Кандидати, що не пройшли: 23

Відкритий ключ A:
n = 5560613234258010662369324406113638545190330363806000167093875593159504705593450767233528383029308162859790505150397223395164748655676738957953138324534881
e = 1833277522489699276116639738680678551580695672626107621042309087386499012809333485347126117049443549611629593485243236004098714091734936086684182261396 9

Секретний ключ A:
p = 721962806482402337800508037266046247694000293455427037441235995174433320741581
q = 770207714903045938873083560396635413714616804965299564134751261902741008993 01
d = 868383160682211356978352327253081177350587171513075301339636849465009063126878245036260492889804900636855693939874540415108244024748140299686829555521 29
```

Генерація ключів B

```
==============================
   ГЕНЕРАЦІЯ КЛЮЧІВ B
==============================
Кандидати, що не пройшли: 79
Кандидати, що не пройшли: 17

Відкритий ключ B:
n = 9827145193026124781019668245713717036485543422938169980549751735683252037930007865954920481534412018428641338460887283847058280371929006987566275899044 57
e = 9140551351866422469403955936577209760738142855524129975731799729983940760342230220779837043337105639061306034305234558272653484477295591714385068826476713

Секретний ключ B:
p1 = 1047589512732317508557502859708711217735524573782423702958440173346719211 6927
q1 = 9380721240130607926489791868934446283407171993667245666223434377975815251839 1
d1 = 5878728986132038699712112098975705738488996468460543640691365699579796006009241109802350020836973284064167175144984624256632927573717098852457788541049557
```

Шифрування та розшифрування

```
==============================
   ШИФРУВАННЯ / РОЗШИФРУВАННЯ
==============================

Вихідне повідомлення (M):
64028662771995069597725154954015221385370367813040292583804650058717897542693491802686174726617262163437653260654102910725689066666799312675596953131902 4

Зашифроване повідомлення (C):
11479544105145489288985181211940400411458171131062375965968769695652559128903254631669148483681526322432408070746138741197623275221770022040976958883478 49

Розшифрування:
64028662771995069597725154954015221385370367813040292583804650058717897542693491802686174726617262163437653260654102910725689066666799312675596953131902 4 (вірно)
```

Цифровий підпис

```
==============================
ЦИФРОВИЙ ПІДПИС
==============================

[1] Повідомлення, яке підписуємо:
M = 65699636688680007509004787289267282142195825956986152353136954711444499386848613455032033405632415240317859447719256270851707077409462217429828166736205 86

[2] Формування підпису:
S = M^d_A mod n_A =
6984434913582022635916206657926661706004054270523912128090098817901827823744377739941355603409071203982587142730258170473906423107531943152475538773327881

[3] Перевірка підпису:
S^e_A mod n_A = 65699636688680007509004787289267282142195825956986152353136954711444499386848613455032033405632415240317859447719256270851707077409462217429828166736205 86
Очікується     = 65699636688680007509004787289267282142195825956986152353136954711444499386848613455032033405632415240317859447719256270851707077409462217429828166736205 86

Результат:
Успіх — підпис підтверджено
```

Передача секретного ключа

```
==============================
 ПРОТОКОЛ РОЗСИЛАННЯ СЕКРЕТНОГО КЛЮЧА
==============================

[1] Початковий секретний ключ k:
211378253914606903636343102788096545085217975020715682011256714684979996304589899041277143914475727627091799142476202073002375685591827620497712345187412 7

[2] Підпис S = k^d_A mod n_A:
501468847957044124393662180610784327608152067925209239174928791146760919549380068311586668714362121190149341379888811385723416446546660266753059577453809 6

[3] A шифрує k відкритим ключем B -> k1:
431066476732314749299148829306566277541391595998141418363208804260626755059446306757276604349415855148815214287762301301439494433646117640993152778177360 1

[4] A -> B надсилає (k1, S):
k1 = 431066476732314749299148829306566277541391595998141418363208804260626755059446306757276604349415855148815214287762301301439494433646117640993152778177360 1
S  = 501468847957044124393662180610784327608152067925209239174928791146760919549380068311586668714362121190149341379888811385723416446546660266753059577453809 6

[5] B розшифровує k1 своїм секретним ключем:
k2 = 211378253914606903636343102788096545085217975020715682011256714684979996304589899041277143914475727627091799142476202073002375685591827620497712345187412 7

[6] Детальна перевірка підпису:
S^e_A mod n_A = 211378253914606903636343102788096545085217975020715682011256714684979996304589899041277143914475727627091799142476202073002375685591827620497712345187412 7
Очікуване значення = 211378253914606903636343102788096545085217975020715682011256714684979996304589899041277143914475727627091799142476202073002375685591827620497712345187412 7

Результат перевірки:
Успіх — підпис підтверджено
```

Покроковий опис передачі ключа:

- Клієнт A та клієнт B генерують свої ключові пари RSA

- Клієнт A створює секретний ключ для передачі клієнту B.

```
# 1. A генерує секретний ключ k
k = random.randint(2, nA - 1)
```

- Клієнт A створює цифровий підпис S для ключа k та шифрує k відкритим ключем B

```
def SendKey(k, private_key_A, public_key_B):
    # A створює підпис S і шифрує k -> k1
    dA, pA, qA = private_key_A
    nA = pA * qA

    S = modexp(k, dA, nA)      # цифровий підпис
    nB, eB = public_key_B
    k1 = modexp(k, eB, nB)     # зашифрований ключ

    return k1, S
```

- Клієнт B отримує отримує k1, S. Розшифровує k1 своїм приватним ключем. Клієнт B перевіряє підпис S за допомогою відкритого ключа A

```
k2, valid = ReceiveKey(k1, S1, privB, pubA)
```

```
def ReceiveKey(k1, S, private_key_B, public_key_A):
    # B розшифровує k1 та перевіряє підпис S
    dB, pB, qB = private_key_B
    nB = pB * qB

    k = modexp(k1, dB, nB)

    nA, eA = public_key_A
    valid = (k == modexp(S, eA, nA))

    return k, valid
```

Висновки:
У цій лабораторній роботі ми реалізували криптосистему RSA, включаючи генерацію великих простих чисел і створення ключових пар. Було протестовано шифрування, розшифрування та цифровий підпис, що підтвердило коректність роботи алгоритму. Також ми побудували протокол безпечної передачі секретного ключа з перевіркою автентичності відправника.