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:
ФБ-31 Караман Любов,
ФБ-31 Голомовза Дар’я

2025

**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.

```python
# Тест Міллера-Рабіна
def miller_rabin(p, k=5):
    # Перевірка особливих випадків: числа 2 та 3 завжди прості
    if p == 2 or p == 3:
        return True

    # Відсіюємо числа менші за 2 та парні числа (крім 2)
    if p < 2 or p % 2 == 0:
        return False

    # Крок 0: розклад p-1 = d * 2^s
    # Знаходимо такі d та s, що d - непарне, а s - степінь двійки
    s, d = 0, p - 1
    while d % 2 == 0:
        s += 1          # Збільшуємо степінь двійки
        d //= 2         # Ділимо d на 2 поки воно парне

    # Крок 1: k раундів перевірок з різними випадковими основами
    for _ in range(k):
        x = random.randint(2, p - 2)  # вибір випадкової основи
        g = gcd(x, p)                 # знаходимо НСД(x, p)

        # Якщо x і p не взаємно прості (НСД ≠ 1), то p складене
        if g != 1:
            return False

        # Крок 2: перевірка сильної псевдопростоти
        x_power = pow(x, d, p)  # обчислюємо x^d mod p

        # Якщо x^d ≡ 1 (mod p) або x^d ≡ -1 (mod p), переходимо до наступного раунду
        if x_power == 1 or x_power == p - 1:
            continue

        # Перевіряємо послідовність x^(d*2), x^(d*4), ..., x^(d*2^(s-1))
        for r in range(s - 1):
            x_power = pow(x_power, 2, p)  # обчислюємо x^(2^r * d) mod p
```

```
=== ТЕСТУВАННЯ ФУНКЦІЇ МІЛЛЕРА-РАБІНА ===
miller_rabin(2) = True (ПРОСТЕ)
miller_rabin(3) = True (ПРОСТЕ)
miller_rabin(4) = False (СКЛАДЕНЕ)
miller_rabin(17) = True (ПРОСТЕ)
miller_rabin(25) = False (СКЛАДЕНЕ)
miller_rabin(97) = True (ПРОСТЕ)
miller_rabin(100) = False (СКЛАДЕНЕ)

==========
=== ГЕНЕРАЦІЯ 16-БІТНОГО ЧИСЛА ===
16-бітне число: 50023
```

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p1 , q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p1q1$ ; p і q – прості числа для побудови ключів абонента A, p1 і q1 – абонента B.

p = 703102508600718975894934874661308204200070619461596263313148721207370340336 51

q = 726610323374860025616034470062001477783935618601951213643125224515681431858 41

p1 = 100557293597230021161707360369430771008852436463763663016456000739586315309357

q1 = 102457560898635210812066898328209636448788089346424503355182272170438952359831

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), (e1, n1) та секретні d і d1 .

$$n = p * q$$
$$d = e^{-1} mod(\varphi(n))$$

```
3. ГЕНЕРАЦІЯ КЛЮЧОВИХ ПАР...
Абонент A:
  Відкритий ключ: e = 65537
                  n =
                  5108815411400437174920114549486751048464889903733412665038043186550311691864336450134058878429712505787872016652444802523748053251810077515472721940735491
  Закритий ключ: d =
  4009676035158830690546804584920732009092398827812312850016964880702326662716000426807021281780964191024800031013634773860983889191695301446034373515189473
  p = 703102508600718975894934874661308204200070619461596263313148721207370340336 51
  q = 726610323374860025616034470062001477783935618601951213643125224515681431858 41

Абонент B:
  Відкритий ключ: e1 = 65537
                  n1 =
                  1030285503254013545741460657822963058169782156221063193798693240541305541869797802328570089466484842961672759751265505153669993329749781609844629794523866 7
  Закритий ключ: d1 =
  8599363308283533723790170353170560484145635858743668120906101736553390366635442433071323776111321490497575364810891612061990630459400537913463534365865513
  p1 = 100557293597230021161707360369430771008852436463763663016456000739586315309357
  q1 = 102457560898635210812066898328209636448788089346424503355182272170438952359831
```

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.

*Зашифрувати C = Memodn*

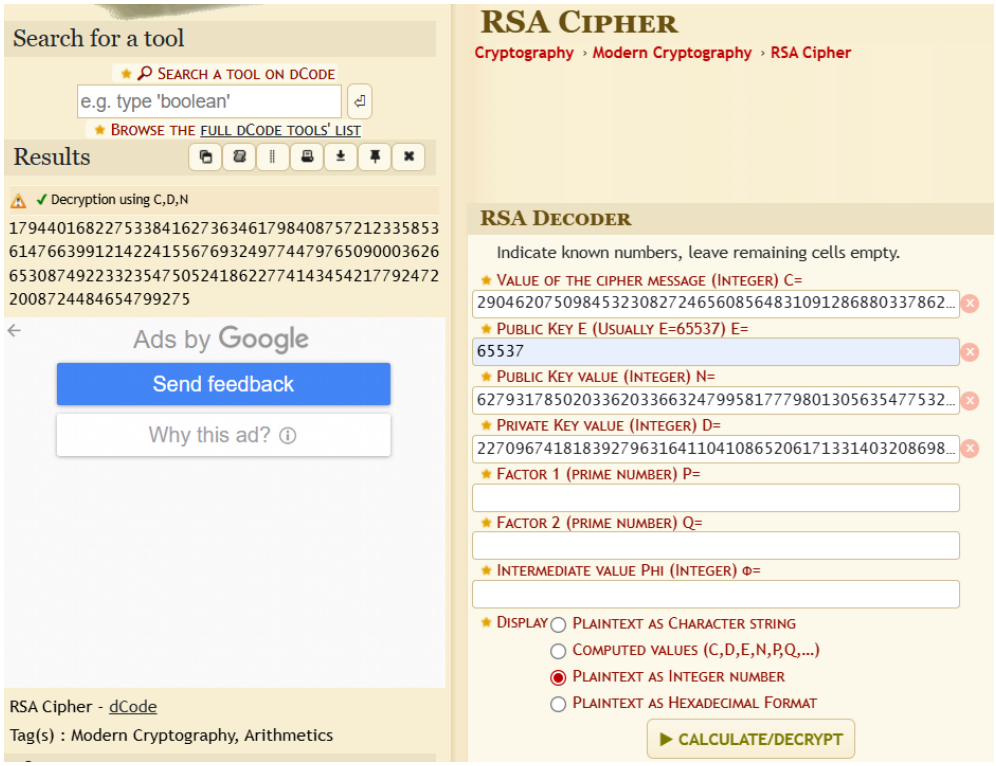
*Розшифрувати M = Cdmodn*

*Цифровий підпис створити S=Mdmodn*

```
4. ШИФРУВАННЯ, РОЗШИФРУВАННЯ ТА ЦИФРОВИЙ ПІДПИС
Випадкове повідомлення M = 179440168227533841627363461798408757212335853614766399121422415567693249774479765090003626653087492233235475052418622774143454217792472200872448465479927 5

Шифротекст для A: C = 290462075098453230827246560856483109128688033786283143071193924626815792508259752136379644042975943839837881043327750775310534314790276222927638217386105 8
Розшифроване повідомлення A: 179440168227533841627363461798408757212335853614766399121422415567693249774479765090003626653087492233235475052418622774143454217792472200872448465479927 5
Перевірка коректності: True

Шифротекст для B: C1 = 283862350678317658861052951782347175687926805529677229614176751195981982652712432573042629513541194873454508890249327506604839264027570431319503553686857 3
Розшифроване повідомлення B: 179440168227533841627363461798408757212335853614766399121422415567693249774479765090003626653087492233235475052418622774143454217792472200872448465479927 5
Перевірка коректності: True

Цифровий підпис A: S = 34908273538870818331711511417121388605026344398599784395059377205005584424184150584362740240642124790193861693387356660810267060880651372543495000464273 04
Перевірка підпису A: True

Цифровий підпис B: S1 = 28457465379846571854972882670862746760286699285573193629891784176661434027438343218502746602659693433874054455434177190491446855744316521020531395103798 98
Перевірка підпису B: True
```

Бачимо, що

```
Випадкове повідомлення M = 179440168227533841627363461798408757212335853614766399121422415567693249774479765090003626653087492233235475052418622774143454217792472200872448465479927 5

Шифротекст для A: C = 290462075098453230827246560856483109128688033786283143071193924626815792508259752136379644042975943839837881043327750775310534314790276222927638217386105 8
```




*Виконали перевірку на сайті та впевнилися що алгоритм працює коректно.*

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.

Абонент $A$ формує повідомлення $(k_1, S_1)$ і відправляє його $B$, де

$$k_1 = k^{e_1} \bmod n_1, \quad S_1 = S^{e_1} \bmod n_1, \quad S = k^d \bmod n.$$

Абонент $B$ за допомогою свого секретного ключа $d_1$ знаходить (конфіденційність):

$$k = k_1^{d_1} \bmod n_1, \quad S = S_1^{d_1} \bmod n_1,$$

і за допомогою відкритого ключа $e$ абонента $A$ перевіряє підпис $A$ (автентифікація):

$$k = S^e \bmod n.$$

```
5. ПРОТОКОЛ КОНФІДЕНЦІЙНОГО РОЗСИЛАННЯ КЛЮЧІВ

Повідомлення k = 474992224497537139703480946218362365456416797252256367844804953329445257689218581839253914281417846191958160707491563806936243425469213128428613037677991
7

A створює цифровий підпис для k: S = 231118027531495826478997195105011743323558974325877402988244665999828624097855188378137701191063333982869901730901132251460319157792606514016316968620348
0

A формує повідомлення для B:
  Зашифрований ключ (k1) = 393138777260262848925963815620536277300946491848082797288267150102922439949002123317635325920083704351627726009304301862356034522707596881465254199838563
1
  Зашифрований підпис (S1) = 115354965436361516091041005127640701683553083057184800587761679931286373926407922919060842220943332152070670539732382407314862825552172477462595538346547

B отримує повідомлення від A і розшифровує його:
  Розшифрований ключ k = 474992224497537139703480946218362365456416797252256367844804953329445257689218581839253914281417846191958160707491563806936243425469213128428613037677991
7
  Розшифрований підпис S = 231118027531495826478997195105011743323558974325877402988244665999828624097855188378137701191063333982869901730901132251460319157792606514016316968620348
0

Перевірка цифрового підпису A (автентифікація): True

Протокол успішно виконано: ключ k = 474992224497537139703480946218362365456416797252256367844804953329445257689218581839253914281417846191958160707491563806936243425469213128428613037677991
7 отримано конфіденційно та підпис перевірено.
```

## Висновки:

У роботі була реалізована функція генерації випадкових простих чисел із застосуванням тесту Міллера-Рабіна та пробних ділень, що забезпечує отримання криптографічно стійких чисел довжиною понад 256 біт. Було створено ключові пари RSA та реалізовані операції шифрування, розшифрування, цифрового підпису та перевірки підпису, що гарантує конфіденційність, автентичність і цілісність повідомлень. На завершальному етапі розроблено протокол безпечної передачі ключів, який підтвердив коректність роботи системи та можливість безпечного обміну інформацією через відкритий канал.