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 Аль-Фітурі Асія,
ФБ-31 Гриб Вероніка

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.

Порядок і рекомендації щодо 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
def trial_division(n: int) -> bool:
    """Пробні ділення на малі прості числа"""
    small_primes = [
        2, 3, 5, 7, 11, 13, 17, 19, 23,
        29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71
    ]
    for p in small_primes:
        if n % p == 0:
            return n == p
    return True

def miller_rabin_test(n: int, k: int = 20) -> bool:
    """Ймовірнісний тест Міллера-Рабіна"""
    if n in (2, 3):
        return True
    if n < 2 or n % 2 == 0:
        return False

    # n - 1 = d * 2^s
    d = n - 1
    s = 0
    while d % 2 == 0:
        d //= 2
        s += 1

    for _ in range(k):
        a = random.randint(2, n - 2)
        x = mod_exp(a, d, n)

        if x == 1 or x == n - 1:
            continue

        is_composite = True
        for _ in range(s - 1):
            x = mod_exp(x, 2, n)
            if x == n - 1:
                is_composite = False
                break
        if is_composite:
            return False

    return True
```

```python
def is_prime(n: int, k: int = 20) -> bool:
    """Перевірка числа на простоту: пробні ділення + Міллер-Рабін"""
    if n < 2:
        return False
    if n in (2, 3):
        return True

    if not trial_division(n):
        return False

    return miller_rabin_test(n, k)

def generate_prime(bits: int) -> int:
    """Генерація випадкового простого числа заданої довжини (bits)"""
    while True:
        # Випадкове число заданої бітової довжини
        num = secrets.randbits(bits)
        # Ставимо старший біт, щоб точно було bits
        num |= (1 << (bits - 1))
        # Робимо непарним
        num |= 1

        if is_prime(num):
            return num
```

```
=== Завдання 1: генерація простих ===
Просте з інтервалу [1, 500]: 2
Просте 64-бітне число: 18227730479921718461
```

2. За допомогою цієї функції згенерувати дві пари простих чисел qp, і 1 1
,q p
довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб 1 1
qp pq ; p і q –
прості числа для побудови ключів абонента А, 1
p і 1
q – абонента В.

```
=== Завдання 2: дві пари простих p,q і p1,q1 ===
p  = 107889650830912351806203240758556388918815897482091182561311980001243871260263
q  = 898264376305312071274842456229712401234883219898112395565596647892052878334 51
p1 = 112425812363636134959145462828979707513189022157829775224567552882346656522 79
q1 = 115416157344739431582160926922024848219707760378947240414080694227290676685723

p*q  = 969134299134273779701970045747289056877948635417640645436754656095897593867808729451599279733746178367640735714328229618341453685619265059561835621845761 3
p1*q1 = 129757552493715798927205227161382548094864647528057442750843552776980107944007114731759473607469663998148808453912295322796589339746966108806144856817127 17
```

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ) , , (qp d та відкритий ключ) , (en . За
допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі) , (ne ,) , (1 1
n e та секретні d і 1
d .
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.

```
=== Завдання 3–4: RSA для A і B ===
Відкрите повідомлення M: 500403157700241479021762410754572157026300416295801062098641885324522913840066647144736566395906213432505936497199980218207405509310057036221758253098729
3
Криптограма для A: 681259278192735623013507308063640539732007514671074690639759793139258919217407184482543898813481635359738660528935986105085758185190598955919929625316775
6
Розшифроване A: 500403157700241479021762410754572157026300416295801062098641885324522913840066647144736566395906213432505936497199980218207405509310057036221758253098729
3
Криптограма для B: 116845825936699129336791687409762264072897633944523806459673949046837319021777523130712568709156141216500365704844099463626704011194204923384127977641084
7
Розшифроване B: 500403157700241479021762410754572157026300416295801062098641885324522913840066647144736566395906213432505936497199980218207405509310057036221758253098729
3
Цифровий підпис від A: 59888401238638401155694350052810548514621100715058982411164283378301952286640723642028181074662114722398103485947883397479065841307518692584846678613509
52
Перевірка підпису A: Дійсний
====================================
```

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 n k 0 .

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().

Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою
http://asymcryptwebservice.appspot.com/?section=rsa.
Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

```
=== Завдання 5: протокол розсилання ключа ===
Випадковий ключ k: 189396702220407177592674456118370877116070145482760693793834231899815468734597547806948705861008077662392447052680770236425865480172175581980538194561942 4

Відправлені значення від A до B:
k1 = 899874137728468526976854590631520855537721105785354452541879460549510500374971079165104144891058285408094103335490499039259301526287767436133558423033882 0
S1 = 383443625982820340691614453468740948843654073415054742791527427734297984714896294437150351713839645196837671952355347279577796268009045655972142695986773 0

B відновив ключ k_rec: 189396702220407177592674456118370877116070145482760693793834231899815468734597547806948705861008077662392447052680770236425865480172175581980538194561942 4
Перевірка підпису A: Дійсний
```

```
=== Перевірка через сайт dCode (RSA Cipher) для абонента A ===
Використовуємо значення, які вже обчислені програмою:
M  (відкрите повідомлення) = 500403157700241479021762410754572157026300416295801062098641885324522913840066647144736566395906213432505936497199980218207405509310057036221758253098729 3
C  (криптограма для A)     = 681259278192735623013507308063640539732007514671074690639759793139258919217407184482543898813481635359738660528935986105085758185190598955919929625316775 6
n  (модуль)                = 969134299134273779701970045747289056877948635417640645436754656095897593867808729451599279733746178367640735714328229618341453685619265059561835621845761 3
e  (публічний показник)    = 65537
d  (секретний показник)    = 182626739007099519039921419426873672161415164676562277356972702485379789802206290024840948195927219720076932106292420923683860315944921983067160004551847 3
```

Перевіримо через сайт:

Це те ж саме що на сайті, отже алгоритм працює коректно.

```
Як підставляти на https://dcode.fr/rsa-cipher :
  Value of the cipher message (Integer) C = C  (криптограма для A)
  Public key value (Integer) N            = n
  Public key value (Integer) E            = e
  Private key value (Integer) D           = d
  DISPLAY → 'Plaintext as integer number'
  Натиснувши CALCULATE/DECRYPT → результат має дорівнювати M.
```

```
Ті ж самі значення у шістнадцятковому форматі (для зручності):
n_hex = 95240AA08AAAD9463EB433903A4048295CCE0EA0799B2D5E345FE7725223154471DD091F517F8DD503D0AF35881
DF2EE5869CB521F4624037756FC9889D958FF
d_hex = 34A9E36EC948872CBCDB9911A6E0281B2300483DF6730B54269E6EF5F97715FCE94B546E6E09019E1AB337F3BEB
85F7CA5FE102632F0A6925FF2EFE88C06F811
C_hex = 542C4E64BB865D046A3F10FB2FAF869DFAEEBEDB377B4E39DB1E16DF7B193DBCE392A4F53C8A9437A0AC19187A6
AF85F5E94CBF94F053A36ED7E127BA2E33C5B
M_hex = 56BC69A10DAA8D6847748A8B7B686403E9F72BA632B9A2D61DC0A777DCAB1A24FCB6E4C288B963BC7B097FC554E
2AD22E0A35082F9A02BB7638E48A3B117B593
```

Висновки: У ході виконання нашої роботи була реалізована функція для генерації випадкових простих чисел заданої бітової довжини. Вона поєднує тест Міллера–Рабіна з попереднім пробним діленням, що дозволяє ефективно перевіряти простоту кандидатів. Такий підхід забезпечує отримання якісних простих чисел, придатних для використання у криптографічних задачах. Згенеровані пари простих чисел розміром понад 256 біт гарантують достатню криптостійкість проти сучасних атак.

Також були створені та протестовані функції, необхідні для роботи алгоритму RSA: формування пари ключів, операції шифрування і розшифрування, а також механізми створення та перевірки цифрового підпису. Особливу увагу приділено коректному виконанню модульного піднесення до степеня, що реалізовано із застосуванням схеми Горнера для оптимізації обчислень з великими числами. Ці компоненти забезпечують повний набір криптографічних можливостей RSA — конфіденційність, автентичність та цілісність даних.

На завершальному етапі було спроєктовано та перевірено протокол безпечної передачі ключів на основі RSA. Він використовує шифрування й підписування повідомлень відкритими та закритими ключами сторін, що дозволяє підтвердити особу відправника та захистити інформацію під час пересилання відкритими каналами. Перевірка протоколу на випадково згенерованих ключах і повідомленнях підтвердила правильність реалізації та відповідний рівень захищеності, що демонструє готовність системи до застосування у практичних умовах.