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-33 Ольшевський Богдан,

ФБ-25 Степура Нікіта

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
def miller_rabin_test(n, k=20):
    """
    Тест Міллера-Рабіна на простоту числа.

    Args:
        n (int): Число для перевірки
        k (int): Кількість раундів тестування

    Returns:
        bool: True якщо число ймовірно просте, False якщо складене
    """
    # Перевірка тривіальних випадків
    if n < 2:
        return False
    if n == 2 or n == 3:
        return True
    if n % 2 == 0:
        return False

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

    # Виконуємо k раундів тестування
    for _ in range(k):
        a = random.randint(2, n - 2)

        # Перевіряємо НСД(a, n)
        if math.gcd(a, n) != 1:
            return False

        # Обчислюємо a^d mod n
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue

        # Перевіряємо послідовність квадратів
        for _ in range(s - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False

    return True
```

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 ≤ p1q1 ; p і q – прості числа для побудови ключів абонента A, p1 і q1 – абонента B.

p= 48567968306380870604893231928375565109999867986982953555518657836663878595709
q= 41114984726175839504123089274205806665507535906835052448065721822642262646757
p1=902047222671537382281339588851323312656799833403315535344289366692887665406 31
q1=705600617369614129407526477278650785608132877532680837703947637333216677438 37
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 .

$$n = p * q$$

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

```
=== КОРИСТУВАЧ A ===
Відкритий ключ:
  n = 664367118910885192475272626338363380834785812473531030210340735972378330577380713621709006830777771858319177649921297307580609602373782542562587417217668 7
  e = 65537
Закритий ключ:
  d = 248464807429479470643589606963292284728635736511073829294222369633687731853017345147071254676237521726136308085444245461700827915750489764903742640695675 3
  p = 938641891778383028341445879378197447499373359082010151073000347777975051520509
  q = 707796151791342413456705412371379067439354447174955330940718116979270915772 43
```

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 $C = M^e mod n$

Розшифрувати $M = C^d mod n$

Цифровий підпис створити $S=M^d mod n$

Перевірити цифровий підпис $M=S^e mod n$

```
=== ТЕСТУВАННЯ ШИФРУВАННЯ ===
Оригінальне повідомлення: 57404603602095860386516336838900086290838048244899142868090876401039762111
1180654803057946056202479776226091533 47
Шифрування коректне: True
3976211180654803057946056202479776226091533 47
Шифрування коректне: True
3976211180654803057946056202479776226091533 47
Шифрування коректне: True
3976211180654803057946056202479776226091533 47
Шифрування коректне: True

3976211180654803057946056202479776226091533 47
Шифрування коректне: True

 === ТЕСТУВАННЯ ЦИФРОВИХ ПІДПИСІВ ===
 Цифровий підпис: 464543826078590707490517961099750667084142096085336297890003770338534243021212781
 9917152000379055015697594927619933492959927288387988103002787177756671046
 Перевірка підпису: True
```

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 $0<k<n$.

Повідомлення для обміну

Програма генерує випадкове число M, яке виконує роль повідомлення, що користувач A надсилає користувачу B. Це число знаходиться у межах від 0 до n_A.

Зашифроване повідомлення

Повідомлення M шифрується відкритим ключем користувача B (n_B, e_B).
Завдяки цьому лише B може його розшифрувати, адже тільки він має відповідний закритий ключ d_B.
Цей етап забезпечує конфіденційність обміну.

Зашифрований підпис

Користувач A створює цифровий підпис свого повідомлення за допомогою свого закритого ключа (d_A).
Потім цей підпис додатково шифрується відкритим ключем користувача B.
Таким чином досягається автентичність і захист від підробки повідомлення.

Перевірка автентичності
Після отримання повідомлення користувач B розшифровує його, отримує підпис і перевіряє, чи дійсний він для цього повідомлення.
Якщо перевірка успішна, програма виводить повідомлення:

```
=== БЕЗПЕЧНИЙ ОБМІН ПОВІДОМЛЕННЯМИ ===
Повідомлення для обміну: 57404603602095860386516336838900086290838048244899142868090876401055303776603402409781624577498417661696244223976211180654803057946056202479776226091533347
Зашифроване повідомлення: 5111954121004692291952133184780062051957950922009825161303834559432983068305509277144992627512950829073188252243725922002009478211089723833311108661408409
Зашифрований підпис: 232847532700242857086324480183484675500061229078010340715821605326710171470750262865831538791967306570684361776803224655251780014355282071891601046740743
Повідомлення автентичне: True
```

Як можна побачити, повідомлення автентичні, тому програма вивела true.

Висновки:

У нашій роботі була створена функція для генерації випадкових простих чисел заданої довжини, що поєднує тест Міллера-Рабіна з попередніми пробними діленнями для перевірки простоти чисел. Такий підхід забезпечує отримання простих чисел високої якості, що відповідають вимогам криптографії. Генерація пар простих чисел завдовжки понад 256 біт дозволила забезпечити стійкість системи до атак.

Були реалізовані функції для роботи з RSA, включаючи генерацію ключових пар, шифрування, розшифрування, а також створення та перевірку цифрового підпису. Особлива увага була приділена коректному виконанню операцій піднесення до степеня за модулем, що здійснювалося за допомогою схеми Горнера для підвищення ефективності при роботі з великими числами. Ці функції забезпечують необхідні криптографічні операції в системі RSA, гарантуючи конфіденційність, автентифікацію та цілісність повідомлень.

На останньому етапі був розроблений і протестований протокол конфіденційної передачі ключів з використанням RSA. Протокол передбачає шифрування та підписування повідомлень відкритими та закритими ключами абонентів, що забезпечує підтвердження автентичності відправника та захист даних у відкритому каналі. Тестування з випадковими ключами та повідомленнями підтвердило правильність реалізації та рівень безпеки обміну, що демонструє повну функціональність системи для використання у реальних сценаріях.