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-21 Худоба Арсен,
ФБ-21 Шабанов Кирило

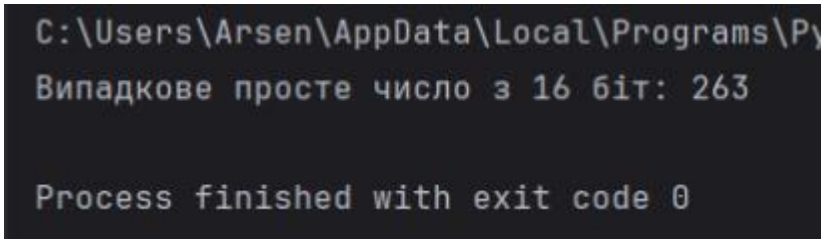
Мета роботи:

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.

```
# Тест Міллера-Рабіна
def miller_rabin(p, k=5):
    if p == 2 or p == 3:
        return True
    if p < 2 or p % 2 == 0:
        return False

    # Крок 0: розклад p-1 = d * 2^s
    s, d = 0, p - 1
    while d % 2 == 0:
        s += 1
        d //= 2

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

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

        # Крок 2: перевірка сильної псевдопростоти
        x_power = pow(x, d, p)  # x^d mod p
        if x_power == 1 or x_power == p - 1:
            continue

        for r in range(s - 1):
            x_power = pow(x_power, 2, p)  # x^(2^r * d) mod p
            if x_power == p - 1:
                break
        else:
            return False  # Якщо не знайдено псевдопростоти, p складене

    return True  # Якщо p пройшло всі k раундів, воно, ймовірно, просте
```



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=36646164541686624240937833154972327964008617161515222144838114780214993351409

q=31539962308096287988814609326825756936315439947047156766760061841746726548927

p1=73431616661604875049686928558414853172421655489331888049826565905174368641603

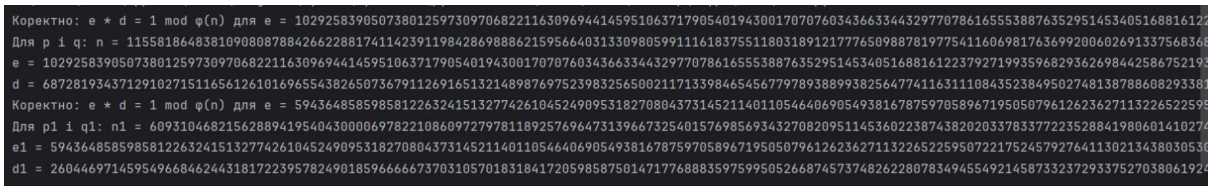
q1=82976583645640818895092178503362825123374069319439494124111219775406714256373

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 .

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

USER A Public key	e	10292583905073801259730970682211630969441459510637179054019430017070760343663344329770786165553887635295145340516881612237927199359682936269844258675219336
	n	11558186483810908087884266228817411423911984286988862159566403133309805991116183755118031891217776509887819775411606981763699200602691337568368892342888143

USER A Private key	d	68728193437129102715116561 26101696554382650736791126 91651321489876975239832565 00211713398465456779789388 99382564774116311108435238 4 950274813878860829338 1
	p	36646164541686624240937833 15497232796400861716151522 21448381147802149933514 09
	q	31539962308096287988814609 32682575693631543994704715 67667600618417467265489 27
USER B Public key	e1	59436485859858122632415132 77426104524909531827080437 31452114011054640690549381 67875970589671950507961262 36271132265225950722175245 79276411302134380305300 7
	n1	60931046821562889419540430 00069782210860972797811892 57696473139667325401576985 69343270820951145360223874 38202033783377223528841980 60141027429433429568591 9
USER B Private key	d1	26044697145954966846244318 17223957824901859666667370 31057018318417205985875014 71776888359759950526687457 37482622807834945549214587 33237293375270380619244 7
	p1	73431616661604875049686928 55841485317242165548933188 80498265659051743686416 03
	q1	82976583645640818895092178 50336282512337406931943949 41241112197754067142563 73



Бачимо що Повідомлення для A, M = 4148839703010821568637612144303780480229407909454044657922614664857072450047251506549403727494194118673917486505857912661874891139594046374384726530881 40

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

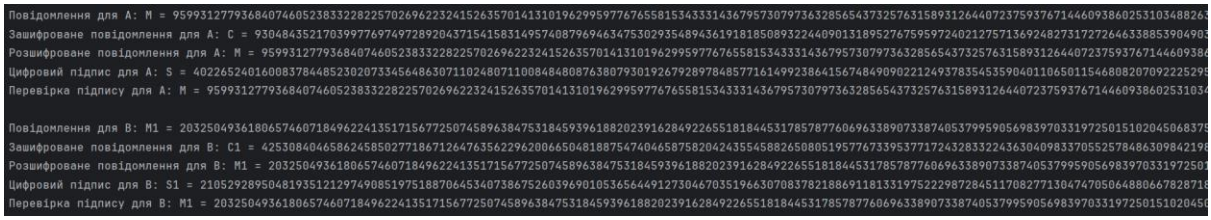
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише 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 $M = S^e mod n$



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 А формує повідомлення k1, s1 і відправляє його В, де

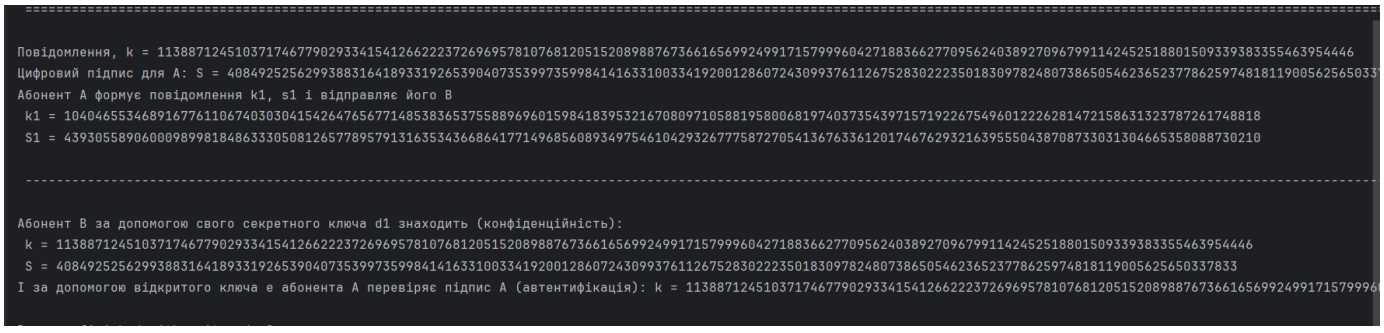
$$k_1 = k^{e_1} \bmod n_1, \quad S_1 = S^{e_1} \bmod n_1, \quad S = k^d \bmod n.$$

Абонент В за допомогою свого секретного ключа d1 знаходить (конфіденційність):

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

і за допомогою відкритого ключа е абонента А перевіряє підпис А (автентифікація):

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



Висновки:

У роботі була розроблена функція генерації випадкових простих чисел заданої довжини, яка використовує тест Міллера-Рабіна в комбінації з попередніми пробними діленнями для перевірки чисел на простоту. Такий підхід гарантує отримання простих чисел високої якості, необхідних для криптографічних цілей. Генерація двох пар простих чисел довжиною понад 256 біт дозволила забезпечити належну стійкість системи до атак. Використання власної реалізації тесту забезпечило розуміння алгоритму його роботи та дозволило адаптувати його до специфічних потреб завдання.

Було реалізовано функції для роботи з RSA, зокрема генерацію ключових пар, шифрування, розшифрування, створення та перевірку цифрового підпису. Особливу увагу приділено правильній реалізації операцій піднесення до степеня за модулем, яка виконувалася за допомогою схеми Горнера, що забезпечує ефективність навіть для великих чисел. Ці функції дозволяють виконувати всі необхідні криптографічні операції в системі RSA, забезпечуючи конфіденційність, автентифікацію та цілісність повідомлень між абонентами.

На завершальному етапі було розроблено та протестовано протокол конфіденційної передачі ключів за допомогою алгоритму RSA. Протокол передбачає використання відкритих та закритих ключів абонентів для шифрування та підпису повідомлень, що гарантує підтвердження автентичності відправника і захист переданих даних у відкритому каналі. Проведене тестування для випадково згенерованих ключів та повідомлень підтвердило коректність реалізації та забезпечення безпеки обміну. Це демонструє повну функціональність розробленої системи, яка може бути використана у реальних сценаріях.