Міністерство освіти і науки України

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:

Студенти групи ФБ-22

Дажук Павло, Копилов Сергій

Київ – 2024

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.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq <= p_1q_1; p і q – прості числа для побудови ключів абонента A, p_1, q_1 – абонента B.

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), (e_1,n_1) та секретні d і d_1.

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.

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.

Хід 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.

```
def is_prime_trial_division(n, small_primes):
    """Перевірка на простоту пробними діленнями."""
    if n < 2:
        return False
    for p in small_primes:
        if n % p == 0:
            return n == p
```

```
    return True


def miller_rabin_test(p, k=20):
    """
    Виконує імовірнісний тест Міллера-Рабіна для перевірки числа p на
простоту.
    :param p: Число для перевірки
    :param k: Кількість ітерацій
    :return: True, якщо p є простим, інакше False
    """
    # Тривіальні перевірки
    if p <= 1:
        return False
    if p <= 3:
        return True
    if p % 2 == 0:
        return False

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

    # Допоміжна функція для на сильну псевдопростоту за основою x
    def is_strong_pseudoprime(x):
        """
        Перевіряє, чи є p сильно псевдопростим за основою x.
        :param x: Випадкова основа
        :return: True, якщо p є сильно псевдопростим за основою x
        """
        # Обчислюємо x^d mod p
        x = modular_pow(x, d, p)
        if x == 1 or x == p - 1:
            return True

        # Повторне зведення в квадрат до s-1 разів
        for _ in range(s - 1):
            x = modular_pow(x, 2, p)
            if x == p - 1:
                return True
            if x == 1:
                return False

        return False

    # Основний алгоритм
    for _ in range(k):
        # Крок 1: Вибір випадкової основи x
        x = random.randint(2, p - 2)

        # Якщо НСД(x, p) > 1, то p складене
        if gcd(x, p) > 1:
            return False

        # Крок 2: Перевірка на сильну псевдопростоту за основою x
        if not is_strong_pseudoprime(x):
            return False

    return True  # Якщо всі k перевірок пройдено, вважаємо p простим
```

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq <= p_1q_1; p і q – прості числа для побудови ключів абонента A, p_1, q_1 – абонента B.

```
def generate_random_prime(start, end):
    """Генерація випадкового простого числа у заданому інтервалі."""
    # Список малих простих чисел для пробного ділення
    small_primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

    while True:
        # Генерація випадкового числа в заданому інтервалі
        candidate = random.randint(start, end)

        # Перевірка пробними діленнями
        if not is_prime_trial_division(candidate, small_primes):
            continue

        # Перевірка тестом Міллера-Рабіна
        if miller_rabin_test(candidate):
            return candidate


def generate_prime_pair(bits):
    """Генерація пари простих чисел з заданою довжиною."""
    p = generate_random_prime(2**(bits-1), 2**bits - 1)
    q = generate_random_prime(2**(bits-1), 2**bits - 1)
    return p, q
```

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), (e_1,n_1) та секретні d і d_1.

```
def GenerateKeyPair(bits=256):
    """
    Генерація ключових пар для RSA.
    :param bits: Довжина простих чисел (в бітах).
    :return: Кортеж ((n, e), (d, p, q)), де
             (n, e) — відкритий ключ,
             (d, p, q) — секретний ключ.
    """
    # Крок 1: Генерація простих чисел p і q
    p, q = generate_prime_pair(bits)

    # Крок 2: Обчислення n = p * q і φ(n) = (p-1) * (q-1)
    n = p * q
    phi_n = (p - 1) * (q - 1)

    # Крок 3: Вибір e (взаємно простого з φ(n))
    e = 65537

    # Крок 4: Обчислення d, оберненого до e за модулем φ(n)
    d = mod_inverse(e, phi_n)

    # Повернення відкритого і секретного ключів
    return (n, e), (d, p, q)
```

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 М і знайти криптограму для абонентів А и В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.

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 $0 < k < n$.

```
if __name__ == "__main__":
    while True:
        # Генерація пар простих чисел для A
        public_key_a, private_key_a = GenerateKeyPair(256)

        # Генерація пар простих чисел для B
        public_key_b, private_key_b = GenerateKeyPair(256)

        # Перевірка умови pq ≤ p1q1
        if private_key_a[1] * private_key_a[2] <= private_key_b[1] *
private_key_b[2]:
            break

    # Створення випадкового повідомлення для тесту
    message = generate_random_message(public_key_a[0])

    # Шифрування для A і B
    ciphertext_a = Encrypt(message, public_key_a)
    ciphertext_b = Encrypt(message, public_key_b)

    # Розшифрування для A і B
    decrypted_a = Decrypt(ciphertext_a, private_key_a)
    decrypted_b = Decrypt(ciphertext_b, private_key_b)

    # Створення цифрового підпису для A і B
    signature_a = Sign(message, private_key_a)
    signature_b = Sign(message, private_key_b)

    # Перевірка цифрового підпису
    is_valid_a = Verify(signature_a, message, public_key_a)
    is_valid_b = Verify(signature_b, message, public_key_b)

    # Випадковий ключ для передачі
    key_to_send = generate_random_message(public_key_a[0])

    # Відправлення ключа
    encrypted_key, encrypted_key_signature = SendKey(public_key_b,
private_key_a, key_to_send)

    # Прийом ключа
    received_key, verify_message = ReceiveKey(encrypted_key,
encrypted_key_signature, public_key_a, private_key_b)

    with open("rsa_results.txt", "w", encoding="UTF-8") as file:
        file.write(f"Перевірка шифрування, розшифрування, підпису, перевірки
```

```
підпису\n")
        file.write(f"Відкритий ключ A: {public_key_a}\n")
        file.write(f"Секретний ключ A: {private_key_a}\n\n")
        file.write(f"Відкритий ключ B: {public_key_b}\n")
        file.write(f"Секретний ключ B: {private_key_b}\n\n")
        file.write(f"Оригінальне повідомлення: {message}\n\n")
        file.write(f"Зашифроване повідомлення для A: {ciphertext_a}\n")
        file.write(f"Зашифроване повідомлення для B: {ciphertext_b}\n\n")
        file.write(f"Розшифроване повідомлення для A: {decrypted_a}\n")
        file.write(f"Розшифроване повідомлення для B: {decrypted_b}\n\n")
        file.write(f"Цифровий підпис A: {signature_a}\n")
        file.write(f"Цифровий підпис B: {signature_b}\n\n")
        file.write(f"Перевірка цифрового підпису для A: {is_valid_a}\n")
        file.write(f"Перевірка цифрового підпису для B: {is_valid_b}\n\n\n")

        file.write(f"Перевірка відправлення та отримання ключа\n")
        file.write(f"Ключ для передачі: {key_to_send}\n")
        file.write(f"Відправлені ключ та підпис: {encrypted_key,
encrypted_key_signature}\n")
        file.write(f"Отриманий ключ: {received_key}\n")
        file.write(verify_message)
```

Результати:

```
Перевірка шифрування, розшифрування, підпису, перевірки підпису
Відкритий ключ A:
(774526292791214135673802689355012506906318138773427227591058157001159236572021256250146946823220
224283590934451079134834710355366223678960049428088645513 9, 65537)
Секретний ключ A:
(375569084007690221059520204846314943573491068130685625915379666010648021408085488740935292136343
305144206970321342911187006614031030307801714027984947 9273,
765291572064368858229018389005986276590641949190285804649909792481870171391 89,
101206693117230427681084527046754017481895385120594891509901644962305188258551)

Відкритий ключ B:
(826872995427863745911122176827864985920330166557739386271294979300497695472505883040984797029125
107529576221593925989125390761539054843670728880979808 4707, 65537)
Секретний ключ B:
(550462210926991888099185947675708706085388175179633057412306462790506339255216838914661469395269
844262905409827238603419513653403701255784819679399396 6945,
727854448166766267037702466856424072339866367926832489464136681656088544106 43,
113604168733253425761226189136661123367362751689113736786602018763502394942449)

Оригінальне повідомлення:
319723249432376901118550618260796132792426956601622358271460522173308075692491929976244510031588 8
016242618585424137325405297224941363323632526417329287474

Зашифроване повідомлення для A:
328486245044996230159514910925995025315276079893242614068608639803935268854991331871249403092467 9
900210036404685670221409094176705431068255746301543627 37
Зашифроване повідомлення для B:
444219085323866740826884659057220584702092870013010450782121167113664120359128183234911768349978 0
911740394551455480546299976084350745397556878909349187525

Розшифроване повідомлення для A:
319723249432376901118550618260796132792426956601622358271460522173308075692491929976244510031588 8
016242618585424137325405297224941363323632526417329287474
Розшифроване повідомлення для B:
319723249432376901118550618260796132792426956601622358271460522173308075692491929976244510031588 8
016242618585424137325405297224941363323632526417329287474

Цифровий підпис A:
363268141347472377361473883606065318341952936576154938181003687802749436925232120518373963750861 7
748852976758095298289508257107835743362644990753235388787
Цифровий підпис B:
189722349976863494955336141500029818799622570237799070240441393225289005215820043125305144791050 9
794213821752513518338588647904187117003636966413593912512

Перевірка цифрового підпису для A: True
Перевірка цифрового підпису для B: True


Перевірка відправлення та отримання ключа
```

```
Ключ для передачі:
153792238076706746203000023939251750552410680695260318039991516977067957710120178716755790536457521505230432080418627628839565028716179951887895378508977 3
Відправлені ключ та підпис:
(441004688237425622731099945817915788707591974340025310553267499331647508251347860954458874519011328594434212320901456967211589457838342254043446580387962 7,
702681325833602378273466615825442004098602855146118027941972386638543100650258759572622074540926108564128967732894971643414019836730755719283955178349353 8)
Отриманий ключ:
153792238076706746203000023939251750552410680695260318039991516977067957710120178716755790536457521505230432080418627628839565028716179951887895378508977 3
Підпис і ключ є дійсними. Перевірка успішна.
```

Висновки

У цій роботі досліджувалася криптосистема RSA, зокрема методи генерації ключів та електронного підпису, а також організація захищеного зв'язку за допомогою цієї системи. Задача полягала в написанні функцій для пошуку простих чисел, генерації пар простих чисел для ключів RSA, а також у створенні криптосистеми для двох абонентів. У роботі реалізовано процеси шифрування, розшифрування та створення цифрового підпису, а також протокол для конфіденційного розсилання ключів з підтвердженням справжності за допомогою RSA. Ключові етапи включали створення випадкових простих чисел, перевірку їх на простоту за допомогою тестів Міллера-Рабіна, генерацію пари ключів для двох абонентів, а також перевірку коректності обміну зашифрованими повідомленнями і підписами. Робота дозволила на практиці ознайомитися з принципами функціонування асиметричних криптосистем і продемонструвати реалізацію захищеного зв'язку за допомогою RSA.