Міністерство освіти і науки України Національний технічний університет України “Київський політехнічний інститут ім. Ігоря Сікорського” Фізико-технічний інститут

КРИПТОГРАФІЯ
КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4
Криптоаналіз афінної біграмної підстановки
Варіант 7

Виконали студенти:
ФБ-23 Лишиленко Ангеліна
ФБ-23 Тіщенко Олександр

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 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq < p1q1 ; p і q – прості числа для побудови ключів абонента А, 1 p і q1 – абонента В.
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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , (e1 ,n1) та секретні d і d1 .
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.

Виконання роботи

1.

```
def gen_number(start, end):
    while True:
        num = random.randint(start, end)
        if miller_r(num):
            return num
```

Генеруємо випадкове просте число в діапазоні від 10 до 100 та перевіряємо його за допомогою тесту Міллера-Рабіна

```
# перевірка простоти методом Міллера-Рабіна
def miller_r(n, k=5):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False


    s, d = 0, n - 1
    while d % 2 == 0:
        d //= 2
        s += 1

    for _ in range(k):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue

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

    return True
```

Спочатку перевірка простих випадків:

- Якщо число n менше або дорівнює 1, воно не є простим.
- Якщо n дорівнює 2 або 3, воно просте.
- Якщо n є парним і більше 2, воно не є простим

Далі число n−1 розкладається у вигляді $2^s \cdot d$, де d — непарне число, а s — кількість множників 2. Це необхідно для наступних кроків. Наприклад, якщо n−1=24, то d=3 і s=3, оскільки $24=2^3 \cdot 3$.

Після цього вибирається випадкове число a в діапазоні від 2 до n−2. Це число використовується для перевірки, чи n може бути простим. Обчислюється x= a^d mod n. Якщо x=1 або x=n−1, число може бути простим, і тест переходить до наступної ітерації.

Якщо x не дорівнює 1 чи n−1, то x підноситься до квадрату (обчислюється x^2 mod n). Якщо під час цих обчислень x стає рівним n−1, число теж може бути простим. Якщо ні, число точно є складеним. Цей процес повторюється кілька разів (визначається параметром k), щоб зменшити ймовірність помилки. Якщо після всіх перевірок число не виявилося складеним, воно вважається ймовірно простим.

2. Генеруємо випадкове просте число заданої довжини у бітах

```python
# Функція для пошуку випадкового простого числа заданої довжини у бітах
def gen_bits(bit_length):
    while True:
        # Генеруємо випадкове число заданої довжини
        num = random.getrandbits(bit_length)
        num |= (1 << (bit_length - 1)) | 1  # Забезпечуємо старший і молодший біти = 1

        # Перевіряємо простоту
        if  miller_r(num):
            return num
```

та дві пари простих чисел (p, q) та (p1, q1), що задовольняють умову pq <= p1q1

```python
# Функція для генерації двох пар простих чисел, що задовольняють умову pq <= p1q1
def gen_pairs(bit_length):
    while True:
        # Генерація випадкових простих чисел для пари A та пари B
        p = gen_bits(bit_length)
        q = gen_bits(bit_length)
        p1 = gen_bits(bit_length)
        q1 = gen_bits(bit_length)

        # Перевірка умови pq <= p1q1
        if p * q <= p1 * q1:
            return (p, q), (p1, q1)
```

Результат

```
Випадкове просте число в діапазоні: 73
Пара для абонента A: (109945379359341482631862801754932534533480695692497551578306374967936299786103,
833815265083479333894132407571122982388474417775566610066204386398545260958 09)
Пара для абонента B: (904700456708508162116097672426088316997442438327955967639806650622191773378 07,
107998945711700521224585807760209409622675136731241515558733445060442850005503)
```

3. Функція генерації ключових пар для RSA. Функція GenerateKeyPair(), використовує попередню функцію gen_bits для генерації p і q, e взяли за значенням 65537 (2^16 +1), для знаходження d написали функцію mod_inverse(), яка вираховує обернений елемент до e за модулем ф(n). Потім сформували (n,e)-відкритий та (d, p,q)-секретний ключі для A і B.

```
Відкритий ключ A: (65537, 115400899338505383312076295094962770579109651716487529310959152016178171675781360323100992660349967005335695256940997363039448363953888779768099469431272 67)
Секретний ключ A: (485642736737412220844265715354543725311174480727028405083579262493888028871933505788206113627861277726196481767068895163727540507531397003162135500362291 3, 10330553720771266876601128954715406033404743756498850743681705100122254996307 3, 11170833864062195609850710416057846175909628290928144774714236299622465159657 9)
Відкритий ключ B: (65537, 687365629782159620076413670096260399478547142677411036841783048012367096397318947967113823823319856824042376805638568452512218047866850367623618293321646 3)
Секретний ключ B: (219004252795844714082359489224102619917169581858294091281158304859029820404841067193663297535149136354800419764990871184361250090203424680028098807319657 7, 915229427381401428846048112786365049605390921232299020898634870677062153558 39, 751030953789158858846021511627024827949434266375468307514081337106387142648 17)
```

4. Функцію Encrypt() записали для зашифрування повідомлення відкритим ключем (n, e). Використали формулу $C = M^e \bmod n$,
Функцію Decrypt() записали для розшифрування повідомлення секретним ключем d. Використали формулу $M = C^d \bmod n$.
Функцію Sign() записали для підписання повідомлення секретним ключем d. Використали формулу $S = M^d \bmod n$.
Функцію Verify() записали для перевірки цифрового підпису відкритим ключем (n, e). Використали формулу $M = S^e \bmod n$.

```
Повідомлення: 39237601051
Зашифроване повідомлення для A: 64669217354776173311462441582503583372533890270229284981812751867034615883062329245102110007390973657032328808380190630609811504213366853665689957247615 42
Розшифроване повідомлення для A: 39237601051
Зашифроване повідомлення для B: 625894012690414562701854846054143881411593874339536921172376892934726617910334038751674875809104083527262229691082980672817740853797195597733608939698472
Розшифроване повідомлення для B: 39237601051
Підпис для A: 481223518465218828324067403992216416537504694331742434730410732778788846365937847505964363639576876773250711660652122560520136303316906160587600354258694 0
Перевірка підпису A: True
Підпис для B: 171976806379608632054756374184187812788485752771792832635447923844593911546956912244374651164161256347004986616506906244939776949730286475163525218445369 7
Перевірка підпису B: True
```

Перевірка з тестовим середовищем:

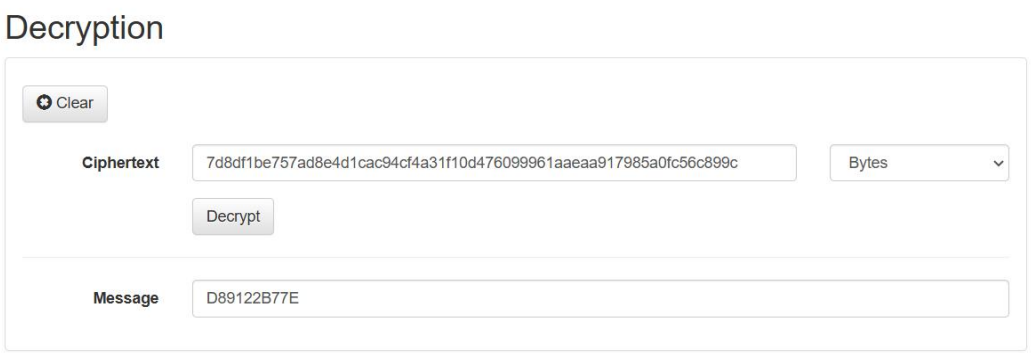
```python
# Перевірка з тестовим середовищем:

M = int('d89122b77e', 16)
key = (int('10001', 16), int('9877D22722D834616FDD95CB4FEEEC9CCF36BD7163B46900BDF1AA7FFF1690F5', 16))
signature = int('83E2795046D06B0BF78BDF411C4D1F2065C7236CB5E77BDE3ADE7347C3E9D4DF', 16)
X = Encrypt(M, key)
Y = Verify(signature, M, key)
print(hex(X)[2:])
print(Y)
```

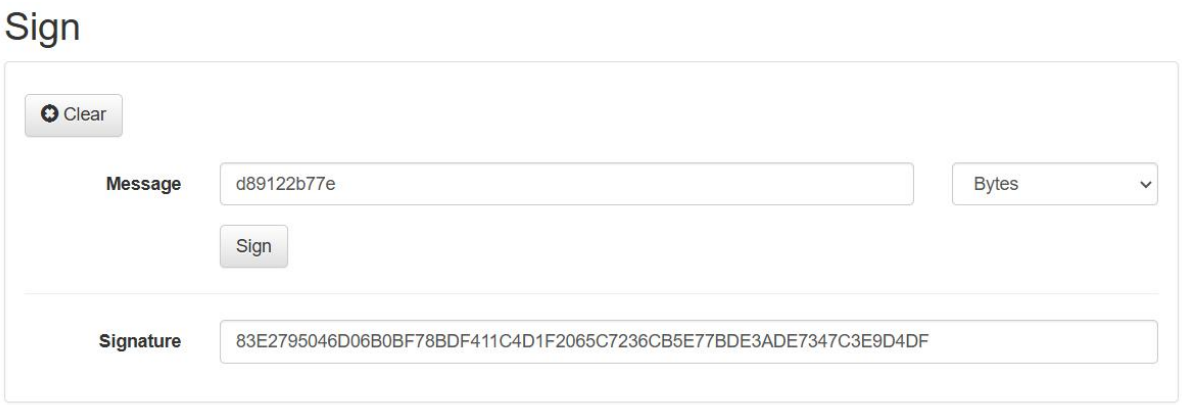
Encrypt:

```
7d8df1be757ad8e4d1cac94cf4a31f10d476099961aaeaa917985a0fc56c899c
```

Decrypt:



Sign:



Verify:

True

5. Робота протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA.

Написали дві функції: SendKey(), ReceiveKey().

```
Секретний ключ k: 40337974006609287035938442580262585979577937306746472541824867579619760455161095666461074831675228665271071704942496321667444584746701460976200897143234 16
Відправлені дані: k1 = 64591162788532956453107160327192205114734730813362661113987563334636245184343161845115246159934364699595751758500922455206392344784634115502371506879167 03, S1 = 23310888259617191415671172619903700351195509921600813058734322923454817773640621668371125850618019409582172378721127844313700085847515370275655066825497 4
Отриманий ключ: 40337974006609287035938442580262585979577937306746472541824867579619760455161095666461074831675228665271071704942496321667444584746701460976200897143234 16
Перевірка автентичності: Успішно
```