МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Комп’ютерний практикум №4

з дисципліни Криптографія:

«Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем»

Виконали:

Студенти групи ФБ-96

Сендецький Костянтин

Твердохлібов Денис

Київ 2021

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.

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 A, 1 p і q1 – 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) , (,) 1 n1 e та секретні d і d1 .

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 $0 < k < n$.

Порядок виконання роботи

1. Реалізовуємо функцію генерації випадкового великого числа. У цій же функції перевіряємо це цисло на простоту за допомогою теста Міллера-Рабіна. Якщо тест не проходить, генерується нове число

```python
def miller_rabin(p):
    k = 50
    d = p - 1
    s = 0
    while d % 2 ==0:
        d//=2
        s+=1
    for _ in range(k):
        a = randint(2, p - 1)
        if gcd(a, p) > 1:
            return False
        x = pow(a, d, p)
        if x == 1 or x == p - 1:
            continue
        for _ in range(1, s):
            x = pow(x, 2, p)
            if x == 1:
                return False
            if x == p-1:
                break
        return False
    return True
```

```python
def prime(length):
    n0 = 2**(length-1)
    n1 = 2**length - 1
    # print("prime from ", n0, " to ", n1)
    x = randint(n0, n1)
    if x % 2 == 0:
        x += 1
    while not miller_rabin(x):
        x += 2
    return x
```

2. Реалізована функція генерацію ключових пар. За допомогою функціїї генерації випадкового великого числа генеруємо числа **p** та **q**. Знаходимо модуль **n = q*p** та застосовуэмо функцію Ойлера до **n (fi_n = (p-1)*(q-1))**. Знаходимо секретний ключ **d = e^{-1}mod(fi_n).** Та генеруэмо ключові пари: відкритий ключ(**e, n**), секретний ключ(**d,n**), де **e -2^{16}+1,** відкритий ключ.

В нашому випадку генеруємо ключі для абонентів А та В

```
def generator(len):
    p = prime(len)
    q = prime(len)
    e = pow(2, 16) + 1
    if p == q:
        q = prime(len)
    n = q * p
    fi_n = (p - 1) * (q - 1)
    while gcd(fi_n, e) != 1:
        e = randint(3, fi_n - 1)
    d = pow(e, -1, fi_n)
    open_key = e, n
    secret_key = d, n
    return open_key, secret_key
```

3. Реалізували функції шифрування, розшифрування, та створення повідомлення з цифровим підписом для абонентів А і В.

```
def encrypt(m, open_key):
    return pow(m, open_key[0], open_key[1])


def decrypt(c, secret_key):
    return pow(c, secret_key[0], secret_key[1])


def sign(m, secret_key):
    return encrypt(m, secret_key)


def verification(sign, msg, open_key):
    if msg == pow(sign, open_key[0], open_key[1]):
        return True
    else:
        return False
```

4. Вибираємо випадкове повідомення(число). За допомогою згенерованих ключів шифруємо, дешифруємо повідомлення та знаходимо підпис. Перевіряємо підпис функцією верифікації.

Абонент А маючи свій ключ та відкритий ключ абонента В формує повідомлення (k1, s1) та відправляє його до В.

За допомогою свого приватного ключа, B може розшифрувати повідомлення та перевірити підпис A за його публічним ключем

```
A
e 65537
d 567437963939029713301322930525601387203092755970744227576223765021886759216413580162685021840026120217978723134578142300416435104237787385669506881013840 9
n 627352168471814000482958279595403659251815006377655528908920288947660051600375181199307539430806140241384953494729632218837608111183172837574062841035326 9
B
e 65537
d 519756619307103371412729896368533073101126138509411795049746887745865272564395935650413467718141548272614856887003712454379755660271404463131827402899781 3
n 632030606912137186237611637782810131029381273578092973600060521053915434976411282256187410186485437942699908650396542023815494151753188030798354983344270 9
Open text:  585474761158097739836601321480554571061554690992527544638840518642872290374 59
Signature:  49389088042983749052164881668252539478587904547629060943157987984405338048669676496673278440815726869833688969809702976512954649515617400440500509954650 59
Encrypt Signature:  32734904967666050205182600537835609016585681310570733115150418411597413575797833364188079038003739563255454164130182883978536100644173795349141562519911 92
Encrypted msg:  428211097643780424078609538873202385806844688264786900376780143380074863768681429099469301512136973863540315839649825672624005162774865778679117870141277 2
Decrypted msg:  585474761158097739836601321480554571061554690992527544638840518642872290374 59
Decrypted sign:  49389088042983749052164881668252539478587904547629060943157987984405338048669676496673278440815726869833688969809702976512954649515617400440500509954650 59
Verification:  True
```

Перевірка за допомогою сайта

Get server key

Clear

Key size 256

Get key

Modulus D2358B0795616D393B67C1151715CC2C06F1FF53AEC760286AA1F49369139E67

Public exponent 10001

Encryption

Clear

Modulus D2358B0795616D393B67C1151715CC2C06F1FF53AEC760286AA1F49369139E67

Public exponent 10001

Message abobus Text

Encrypt

Ciphertext CA98B947424EE5B1FB5EEEC7B348A80D3FB3E3655BAC53D5DD0532A85D411AE3

Decryption

Clear

Ciphertext CA98B947424EE5B1FB5EEEC7B348A80D3FB3E3655BAC53D5DD0532A85D411 Text

Decrypt

Message abobus

Sign

Clear

Message abobus Text

Sign

Signature 9F3F0F0315391B91E6A266AE6526126ABA590E8537F0571D06F83AF51E6A9FEF

Clear

Message abobus Text

Signature 9F3F0F0315391B91E6A266AE6526126ABA590E8537F0571D06F83AF51E6A9FEF

Modulus D2358B0795616D393B67C1151715CC2C06F1FF53AEC760286AA1F49369139E67

Public exponent 10001

Verify

Verification true ✔

```
e -  10001
n -  D2358B0795616D393B67C1151715CC2C06F1FF53AEC760286AA1F49369139E67
Message  abobus
Ciphertext CA98B947424EE5B1FB5EEEC7B348A80D3FB3E3655BAC53D5DD0532A85D411AE3
Sign -  9F3F0F0315391B91E6A266AE6526126ABA590E8537F0571D06F83AF51E6A9FEF
Sign is True
```

Висновок: Ознайомилися з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомилися з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи засекречений зв'язок й електронний підпис, вивчили протокол розсилання ключів.