НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
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:
Філонов Дмитро
Швайка Олексій

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.

Постановка задачі:

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 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.

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 $0 < k < n$.

Хід роботи

Спочатку написали функцію для генерації простих чисел з заданого діапазону довжини. Спочатку випадкове число генерується за допомогою random.randint(), а за допомогою теста Міллера-Рабіна перевіряли чи є це число просто

```
def find_rand_int(max_len, min_len=256):
    if max_len < min_len:
        return 1

    while True:
        p = randint(2**min_len, 2**max_len)
        k, d, s = 30, p-1, 0

        while d % 2 == 0:
            d = d // 2
            s += 1

        simple = True
        for _ in range(k):
            x = randint(2, p)
            if gcd(x, p) > 1:
                simple = False
                break

            if pow(x, d, p) in {1, p-1}:
                continue

            for r in range(1, s):
                if pow(x, d * (2**r), p) == p-1:
                    break
            else:
                simple = False

        if simple == False:
            continue
        return p
```

Потім згенерували дві пари простих чисел p, q і p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq ≤ p1q.

```
def pq_and_p1q1_create(max_len):
    a = []
    for _ in range(4):
        a.append(find_rand_int(max_len))
    a.sort()
    return a[0], a[1], a[2], a[3]
```

Також написали клас, який має метод для генерації ключових пар RSA

```
class RSA:
    def __init__(self, p_, q_):
        self.p_ = p_
        self.q_ = q_
        self.public_key, self.private_key = self.find_rsa()

    def find_rsa(self):
        n = self.q_ * self.p_
        On = (self.q_ - 1)*(self.p_ - 1)

        while True:
            e = randint(2, On - 1)
            if gcd(e, On) == 1:
                break
        d = pow(e, -1, On)
        return (n, e), (d, self.p_, self.q_)
```

Де (n, e) відкрита пара ключів (d, q, p) приватні ключі

Також написали функції для шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису

```
def encrypt(self, msg, reciver_public_key):
    return pow(msg, reciver_public_key[1], reciver_public_key[0])

def decrypt(self, msg, reciver_secret_key):
    return pow(msg, reciver_secret_key[0], reciver_secret_key[1] * reciver_secret_key[2])

def sign(self, msg, sender_secret_key):
    return (pow(msg, sender_secret_key[0], sender_secret_key[1] * sender_secret_key[2]), msg)

def verify(self, msg, sender_public_key):
    return msg[1] == pow(msg[0], sender_public_key[1], sender_public_key[0])
```

Організували роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA

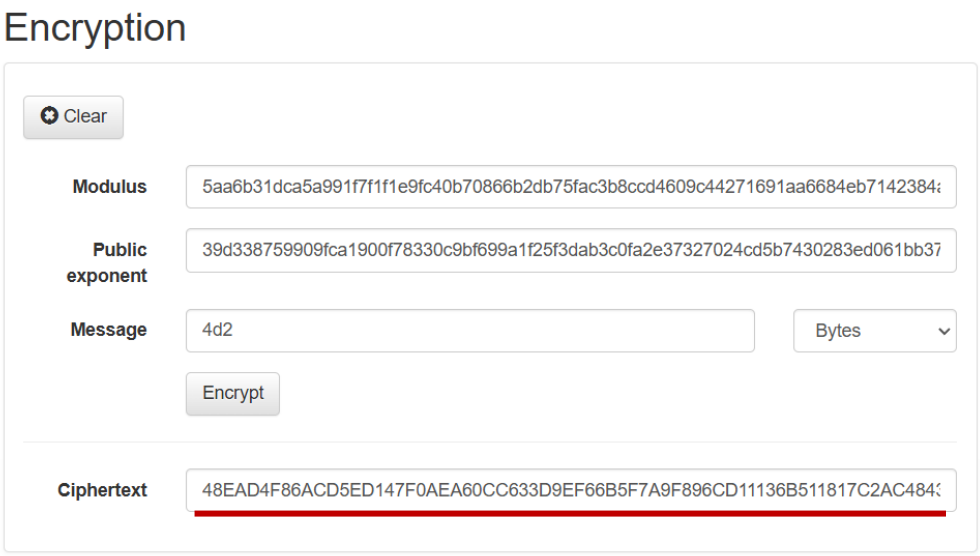
```python
class A_abonent(RSA):
    def sendkey(self, reciver_public_key):
        k = randint(0, self.private_key[1] * self.private_key[2])
        s = self.sign(k, self.private_key)
        s1 = self.encrypt(s[0], reciver_public_key)
        k1 = self.encrypt(k, reciver_public_key)
        return k1, s1


class B_abonent(RSA):
    def receivekey(self, msg, sender_public_key):
        k = self.decrypt(msg[0], self.private_key)
        s = self.decrypt(msg[1], self.private_key)
        if self.verify((s, k), sender_public_key):
            print('Успішний обмін ключами')
        else:
            raise ValueError('Невдалий обмін')
```

Демонстрація створення публічних і приватних ключів, а також демонстрація обміну ключами, шифрування розшифрування повідомлення

```
PS E:\kpi\crypt\4> & C:/Users/dddddd/AppData/Local/Programs/Python/Python313/python.

91797345976698328381
      q1:139795796206955134490035745932211383362824756393331934727593624276544246879189708674966032 1

   RSA:
    1)A:
       SECRET key
       d: 0x2a02d03773cf23646bdaaca0feb70280a99df95e9ec9a99a62fd88e2c424533ad2551aadcc7c94f15d4eaa5d2f476a1fedc6e23638b87aaaa3e602315bb85fd9d15a070a90027f494953

       PUBLIC key
       n: 0x4d00221f93bdc261748f8f677c117683e30c1ef902c805000b4e7d3d212f0edb71f780cba81916b63e3bab3c1f9db3005734412ddb0655a3c928c767b99439886fcf7d0b784d8eefa15c7
       e: 0x3fe4f5840540632b79dbc862c2ff1cfa0ce7fd3a52cca5056947ad9f08b66bf46bce0a695ca2a03e82a1da0505d2bb5f78d5c83ec08947f42f51ae98f8e11a449ed1606d2a15d28a19433

    2)B:
       SECRET key
       d: 0x4d36982c2fb1ee6a572987c68df9a5a7e4af8fa68dd444529c834e9b8e1431ca328e36b73771ab9fd7e86fefa2b6d8105150921f6dc54cdf04d63dfa5243699386de6b2a13f2f374e8d72d

       PUBLIC key
       n: 0x5aa6b31dca5a991f7f1f1e9fc40b70866b2db75fac3b8ccd4609c44271691aa6684eb7142384a90105e740efe483bac4d716c8837f79f5ff6bbb68f6d6d04bf56578c607233e54b1f71f5d
       e: 0x39d338759909fca1900f78330c9bf699a1f25f3dab3c0fa2e37327024cd5b7430283ed061bb37f1a8b2c0fd94150567d78fe9cf35372797d7973060c3a6318b6f75daad6e6c2e82bc862a5


Обмін ключами між абонентами A та B
Успішний обмін ключами

       Початок обміну повідомленнями між абонентами A та B
       Вихідне повідомлення (A):
           0x4d2


       Зашифроване повідомлення (A -> B):
           0x48ead4f86acd5ed147f0aea60cc633d9ef66b5f7a9f896cd11136b511817c2ac4843af2804022b26dab0001a421511bed9ee4596e2c8118741f6f6720f2aaefba89707f65cfc279f5c8db8

Розшифроване повідомлення (B): 0x4d2
```

З скріншоту видно що після шифрування та розшифрування ми отримуємо початкове повідомлення

На сайті перевіримо щоб правильно відбувався процес шифрування повідомлення, з цього видно що Chipertext і значення з консолі співпадає(підкреслено червоним)



Висновки

Після виконання даної лабораторної роботи засвоїли роботу RSA. Створили різні типи ключів: приватні і публічні. Навчилися шифрувати та розшифрвовувати цифровий підпис. Перевірили здобуті навички в тестовом середовищем http://asymcryptwebservice.appspot.com/?section=rsa