

Міністерство освіти і науки України
Національний технічний університет «ХПІ»
Навчально-науковий інститут комп'ютерних наук та інформаційних
технологій
Кафедра комп'ютерної інженерії та програмування

ЗВІТ

з лабораторної роботи № 3
з дисципліни «Сучасні технології безпечного програмування»
«АСИМЕТРИЧНЕ ШИФРУВАННЯ. АЛГОРИТМ RSA»

Виконав:
студент гр. КН-Н9226
Кулик Д.І.

Перевірів:
Бульба С. С.

Харків – 2022

Мета роботи: Дослідити і реалізувати механізм асиметричного алгоритму шифрування RSA.

Індивідуальне завдання

Розробити додаток обміну таємними посиланнями між двома клієнтами за допомогою алгоритму шифрування RSA

- Реалізувати алгоритм генерації ключів (public / private keys) для алгоритму RSA. Створити ключі заданої довжини (напр. 1024 біт)
- Реалізувати та продемонструвати роботу алгоритму шифрування та дешифрування повідомлення RSA
- Підтвердити роботу реалізованого алгоритму шляхом порівняння результату кодування з існуючим алгоритмом (наприклад, використовуючи утиліту openssl або вбудовані системи шифрування обраної мови програмування)

Хід роботи

Алгоритм RSA є асиметричним алгоритмом шифрування. Асиметричний насправді означає, що він працює з двома різними ключами, тобто відкритим ключем і закритим ключем. Як видно з назви, відкритий ключ надається кожному, а закритий ключ залишається закритим.

Важливі фрагменти програми

- Select two large prime numbers, p and q .
- Multiply these numbers to find $n = p \times q$, where n is called the modulus for encryption and decryption.
- Choose a number e less than n , such that n is relatively prime to $(p - 1) \times (q - 1)$. It means that e and $(p - 1) \times (q - 1)$ have no common factor except 1. Choose "e" such that $1 < e < \varphi(n)$, e is prime to $\varphi(n)$.
 $\text{gcd}(e, \varphi(n)) = 1$
- If $n = p \times q$, then the public key is $\langle e, n \rangle$. A plaintext message m is encrypted using public key $\langle e, n \rangle$. To find ciphertext from the plain text following formula is used to get ciphertext C .
 $C = m^e \bmod n$
Here, m must be less than n . A larger message ($> n$) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that:
 $D_e \bmod \{(p - 1) \times (q - 1)\} = 1$
Or
 $D_e \bmod \varphi(n) = 1$
- The private key is $\langle d, n \rangle$. A ciphertext message c is decrypted using private key $\langle d, n \rangle$. To calculate plain text m from the ciphertext c following formula is used to get plain text m .
 $m = c^d \bmod n$

Рисунок 1 – Алгоритм створення ключів

```

def gen_pq(self, bits):
    """
    generate keypair (p, q)
    """
    assert bits >= 512, 'key length must be at least 512 bits'
    l = bits >> 1

    while True:
        p = prime.randprime_bits(l)
        if prime.is_probable_prime(p, None, l // 8):
            break

    while True:
        q = prime.randprime_bits(bits - l)
        if p != q and prime.is_probable_prime(q, None, l // 8):
            break

    self.p = p
    self.q = q
    self.N = p * q
    self.phi = (p - 1) * (q - 1)

```

Рисунок 2 – Функція для генерації пар ключей

```

def encrypt_data(self, data):
    bs = self.key.block_size - 1
    data_stream = (data[i:i + bs] for i in range(0, len(data), bs))
    return b''.join(self.encrypt_block(block) for block in data_stream)

def decrypt_data(self, data):
    useCRT = self.key._can_crt
    bs = self.key.block_size
    data_stream = (data[i:i + bs] for i in range(0, len(data), bs))
    return b''.join(self.decrypt_block(block, useCRT)[:bs - 1] for block in data_stream).rstrip(b'\x00')

```

Рисунок 3 – Функції шифрування та дешифрування даних

```

def my_rsa(text, bits):
    key = RSAKey(bits=bits)
    public_key = (key.e, key.N)
    private_key = (key.d, key.N)

    print(f'Розроблений RSA: public_key          : {public_key}')
    print(f'Розроблений RSA: private_key         : {private_key}')
    cipher = RSA(key)

    encrypted = cipher.encrypt_data(text.encode())

    print(f'Розроблений RSA: Заданий текст          : {text}')
    print(f'Розроблений RSA: Зашифрований текст       : {encrypted.hex()}')

    decrypted = cipher.decrypt_data(encrypted)
    print(f'Розроблений RSA: Розшифрований текст      : {decrypted.decode()}')

```

Рисунок 4 – Виклик власної реалізації алгоритму

```

def rsa_lib(text, bits):
    public_key, private_key = rsa.newkeys(bits)
    print(f'RSA_LIB: public_key          : {public_key}')
    print(f'RSA_LIB: private_key         : {private_key}')

    encrypted = rsa.encrypt(text.encode(), public_key)

    print(f'RSA_LIB: Заданий текст          : {text}')
    print(f'RSA_LIB: Зашифрований текст       : {encrypted.hex()}')

    decrypted = rsa.decrypt(encrypted, private_key)

    print(f'RSA_LIB: Розшифрований текст      : {decrypted.decode()}')

```

Рисунок 5 – Виклик реалізації алгоритму з бібліотеки RSA

Результати роботи програми

```
C:\Users\Daniil\PycharmProjects\stbp\Scripts\python.exe C:\Users\Daniil\PycharmProjects\stbp\LABS\kulyk03/main.py
Розроблений RSA: public_key          : (65537,
46406989123565506199700675815486663837700591488210211282789358246992761502850153310534089289681032431429096055751294208452158755567603978455694396637125164570,
011109047252231054829794245886102027361436931548358573467688925900553275662355866934897)
Розроблений RSA: private_key         :
(1955428238471796776942390043249865300331266054208958274028307868823267938570894279064343616177947892936683839021578634307957892541616771999105849769037660098,
8823214019510296219732845124428358006726603636650330179919312548984777311059454640320193,
46406989123565506199700675815486663837700591488210211282789358246992761502850153310534089289681032431429096055751294208452158755567603978455694396637125164570,
011109047252231054829794245886102027361436931548358573467688925900553275662355866934897)
Розроблений RSA: Заданий текст       : Daniil Kulyk
Розроблений RSA: Зашифрований текст  :
b7eed5d4ecc3f8ce48d9829fc2d4e0da2b6b61ee83ce695594472fb2db3eb43b5f1f3dab58d5331b5d2da92dcaff67feb2bf5ebcc20ece8851a9843031af0ec12927531776a9b102daa57df16bc354,
804fcaff073a2c88404e8516723f46b6fc01
Розроблений RSA: Розшифрований текст : Daniil Kulyk

RSA_LIB: public_key                  : PublicKey
(9459052401537671362332681443914981685453325548346965021691910417527044182863191061319405493275435908085697322586469833261340765971045122082861459368209740996,
619215266082435876482884938883119714929173168245291482241037796409756655659832939336307, 65537)
RSA_LIB: private_key                 : PrivateKey
(9459052401537671362332681443914981685453325548346965021691910417527044182863191061319405493275435908085697322586469833261340765971045122082861459368209740996,
619215266082435876482884938883119714929173168245291482241037796409756655659832939336307, 65537,
38080418475390988483419361157247490240474928859664971715568207952154284407382460741543127475255062178453856319197006283590523638460725163042894334673054929011,
25001126460942732479018450542198771915318084938598908819933671961676528796380275883393,
54132825090273715247546434089794169690620694476952450358379527641274128273214365588941547322084045314744471173473541460199333161875795151523288961594117269254,
1747378302492699804866266907115069155042575115956456900869829589073744073621937364304736988656676744149452572424542763431271477495300584676955287)
RSA_LIB: Заданий текст               : Daniil Kulyk
RSA_LIB: Зашифрований текст          :
407b94ab5b23ce5cb693fe408e9e389a084a779d8d6fdb22ee063dabddda55b53b78c149464e6a927d852456a595abc817b426cc3a1a3676a96ff9b8cb6b6df4e7b0e97dc1c9aa643be9821a0c02ae1,
7b79a5ff6e72919b6e3add95896342d816
RSA_LIB: Розшифрований текст         : Daniil Kulyk
```

Рисунок 6 – Результат виконання програми

При порівнянні можемо побачити що результат виконання реалізацій алгоритму однаковий.

Висновки: в результаті виконання лабораторної роботи було досліджено і реалізовано механізм асиметричного алгоритму шифрування RSA. В результаті порівняння власної реалізації алгоритму з вже реалізованими була виявлена ідентичність роботи, що доводить коректність першого.