

Протокол лабораторної роботи №4

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

Варіант №7

Виконав

Студент 3 курсу

Групи ФБ-13

Короткевич Іван

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.

Код програми:

Створення 256 битного числа та перевірка числа на простоту:

```
#Generate random 256 bit number
def number_256bit():
    random_number = secrets.randbits(256)
    return random_number

#Function to check if number is prime
def check_prime(p, k=15):

    if p % 2 == 0 or p % 3 == 0 or p % 5 == 0 or p % 7 == 0:
        return False

    s = 0
    d = (p - 1)
    while d % 2 == 0:
        d //= 2
        s += 1

    for i in range(k):
        x = randrange(2, p - 1)

        if gcd(x, p) > 1:
            return False

        a = pow(x, d, p)
        if a == 1 or a == -1:
            continue

        for _ in range(1, s):
            a = pow(a, 2, p)

            if a == p - 1:
                break

        else:
            return False

    return True
```

Знаходження простого числа:

```
def select_prime(number):  
    if number % 2 == 0:  
        x = number + 1  
    else:  
        x = number  
  
    i = 1  
    while i < number / 2:  
        if check_prime(x) == True:  
            return x  
  
        x += 2*i  
        i += 1  
  
    return x
```

Шифрування, дешифрування, створення підпису та перевірка повідомлення:

```

def GenerateKeyPair(p, q):
    n = p*q
    phi_n = (p-1)*(q-1)
    exp = secrets.randbits(16)
    e = select_prime(exp)
    d = inverse_elem(e, phi_n)
    return n, e, d

#Encrypt message
def Encrypt(M, e, n):
    C = pow(M, e, n)
    return C

#Sign the message
def Sign(M, d, n):
    S = pow(M, d, n)
    return S

#Decrypt the message
def Decrypt(C, d, n):
    M = pow(C, d, n)
    return M

#Verify integrity of the message
def Verify(M, S, e, n):
    if M == pow(S, e, n):
        return True
    return False

```

Відправлення та отримання ключа:

```
#Function that sends common secret
def SendKey(k, d, e1, n, n1):
    k1 = pow(k, e1, n1)
    S = pow(k, d, n)
    S1 = pow(S, e1, n1)
    return k1, S1

#Function that recieves common secret
def RecieveKey(k1, S1, d1, n1):
    k = pow(k1, d1, n1)
    S = pow(S1, d1, n1)
    return k, S
```

Приклад роботи програми:

```
Ciphertext for B: 57063827247076009810695030921508137434513230017667915687868718798664433720172321593745795133873601692482649375477579750436855030155735159148980796358625
e1: 39937
Modulus1: 769625199170794701664345553169374517444442819983028145370202946236470436559336066352585573743313720570536793295147322093549471975999577417579404728439321
Signature for b: 42902871052326726444313267348625031507551592211868510303600892215484102009014651298743880928622353017753222479498235948477260422493757545562462013493819
Ciphertext for A: 30142657851040080403183369005828688061252384038639657754592497139746706620657475126388072380955103812184749150918832095891417222812875033139304196271103
e: 42689
Modulus: 65564096273902823884059956202733932579622108857686132583503288603452810165591383825653080676215561310692777114749324479666943664297466683969876688643457
Signature for b: 3106166992519797363478028823190144300083573165466186098736363506966410561204310532796935822790519589643285475892861669866166937642596176510149413667966726
Verification result for B: True
e: 42689
Modulus: 65564096273902823884059956202733932579622108857686132583503288603452810165591383825653080676215561310692777114749324479666943664297466683969876688643457
Plaintext: 67890
Verification result: True
e1: 39937
Modulus1: 769625199170794701664345553169374517444442819983028145370202946236470436559336066352585573743313720570536793295147322093549471975999577417579404728439321
Plaintext: 12345
k1 3562664273455145991909602098377691544669562679311109864406255789411558083099910800066182922395502078220384776350278161902876403695027445835412972167348743
S1: 64465228681205818257121649577235164601800145574537934370570361570455818267519375737534366931277426870349051524299161533920195565300239489411879039136210
k: 12345
Verification result: True
```

Програма може зберігати ключі у спеціальному файлі keys.json

Перевірка роботи:

Для перевірки роботи значення треба було перевести в 16-ти систему числення:

Encryption

Modulus
140787B02162022C2ADBC00D6F9FA22422ECA2376F9F41D4DE906DA462B4537D8A31287089D0E579F36EC

Public exponent
A6C1

Message
3039

Bytes
▼

Ciphertext
935590EA98B958C59826B0EE44548134CE313DA71FB7D5E6383757492E753384FAFEA9826C460941D03FB9

Client public(n) and private key: 65564096273902823884059956202733932579622108857686132583503288603452810165591383825653080676215561310692777114749324479666943664297466683969876688643457 17061808557398544602310245108954795310900278931341452057207665513264711474368093945214197839896088088179550927084958594350722911921556846412006121519425
Message from server: 0x3039

Decryption

✖ Clear

Ciphertext

8F859C04FBBD2C786A261D1D4A08214EEA4B2F8D90E369BA5669641AF08442

Bytes

Decrypt

Message

3039

Message for server: 0x3039
Encrypted message for server: 64916804813803711410043531971233661144765847521899449030167718427959832625686

Enter decimal number

64916804813803711410043531971233661

10

= Convert

✖ Reset

↕ Swap

Hex number (64 digits)

8F859C04FBBD2C786A261D1D4A08214EE
A4B2F8D90E369BA5669641AF0844216

16

Sign

✖ Clear

Message

3039

Bytes

Sign

Signature

B3F54154177B0EA2DCE2B69C7499C2A727F7C39AC50F5D540055B3122795EE41

Enter hex number

B3F54154177B0EA2DCE2B69C7499C2A727

16

= Convert

✕ Reset

↕ Swap

Decimal number (77 digits)

813973283078962545998265008375472
229966049856971614988365299992131

10

Decimal from signed 2's complement

Message signature from server: 81397328307896254599826500837547222996604985697161498836529999213175917375041
Verification result: True

Signature from client: 16836862083266965416175260939188169723743324124294925725168075148523430050471367389281514692954537034503167970426283650416237401540369291396205161714197
Message: 0xbdf
Client public key: 42689 65564096273902823884059956202733932579622108857686132583503288603452810165591383825653080676215561310692777114749324479666943664297466683969876688643457

Enter decimal number

42689

10

= Convert

✕ Reset

↕ Swap

Hex number (4 digits)

A6C1

16

Enter decimal number

65564096273902823884059956202733932

10

= Convert

× Reset

↕ Swap

Hex number (127 digits)

140787B02162022C2ADBC00D6F9FA224
22ECA2376F9F41D4DE906DA462B4537D

16

Enter decimal number

16836862083266965416175260939188169

10

= Convert

× Reset

↕ Swap

Hex number (126 digits)

524BFF8330311BF871313DEF81F8FE029
389DE088422FE9CFDB9623C3023E0F2E7

16

Verify

🗑 Clear

Message

BDF

Bytes



Signature

524BFF8330311BF871313DEF81F8FE029389DE088422FE9CFDB9623C3023E0F2E72EDF55A7E702B913B39C

Modulus

140787B02162022C2ADBC00D6F9FA22422ECA2376F9F41D4DE906DA462B4537D8A31287089D0E579F36EC

Public exponent

A6C1

Verify

Verification

true



Send key

Clear

Modulus

140787B02162022C2ADBC00D6F9FA22422ECA2376F9F41D4DE906DA462B4537D8A31287089D0E579F36EC

Public exponent

A6C1

Send

Key

0139DF22D672353D6F3A07568E9A869C479D0E8E688CB9184ABD4708ECE052A4293FAB526D9EBEF0CB89;

Signature

EFB9EE0FED86BA2DF0AC4411295022B633D549C3DA3CFBA05D413D131119CE73776390E6F391E1DCB599

Server key and signature: 5927640984702541738 12067161012964312297537222011419250313893791394002094073294103030284123662332
Client private key: 17061808557398544602310245108954795310900278931341452057207665513264711474368093945214197839896088088179550927084958594350722911921556846412006121519425
Server public key 65537 96616247683602825070798563963110556653091141416585574575630299175132795599309
Verification result: True