НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Фізико-Технічний Інститут

Звіт
із лабораторної роботи №4
із дисципліни «Криптографія»
на тему

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

Виконав:
студент групи ФБ-13
Берчук В.В.

Київ – 2023

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.
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
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.
6. Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою: http://asymcryptwebservice.appspot.com/?section=rsa

Хід роботи

1. Реалізація функції пошуку випадкового простого числа заданої довжини, з використанням модулю random у python, та реалізація тесту Міллера – Рабіна. Тест Міллера – Рабіна (з пробними діленнями):

```
def MillerRabin(p, k = 100):
    primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
    for prime in primes:
        if p % prime == 0:
            return False

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

    for i in range(k):
        x = random.randint(2, p - 1)
        if gcd(x, p) > 1:
```

```
            return False

        if pow(x, d, p) != 1 and pow(x, d, p) != p - 1:
            for r in range(1, s - 1):
                xr = pow(x, d * (2 ** r), p)
                if xr == - 1:
                    break
            else:
                return False
    return True
```

Генератор:

```
def PrimeGen(bits):
    while True:
        p = random.randint(2 ** (bits - 1), 2 ** bits)
        if MillerRabin(p):
            return p
```

2. Результат функції генерування простих p,q,p1,q1 (pq ≤ p1q1):

```
p, q = PrimeGen(256), PrimeGen(256)
p1, q1 = PrimeGen(256), PrimeGen(256)
while p * q > p1 * q1:
    p1, q1 = PrimeGen(256), PrimeGen(256)
```

```
TASK 2 ------------------------------------------------------------------------------------
p = 81723094709594421838509035835382732081035740766125061536136950738479856823863
q = 10558189118676149110819254065805995182552875713020659862325471367264575692240 7
----------------------------------------
p1 = 93913289501781030990769012113304825749925599076234491463745193736515794414147
q1 = 91909471974802288076616285189083625641893003099368262395862199103391600961279
```

3. Функція генерації ключових пар для RSA:

```
def GenerateKeyPair(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = 65537 #2^16 + 1
    d = pow(e, -1, phi)
    return (d, p, q), (n, e)

A_private, A_public = GenerateKeyPair(p, q)
B_private, B_public = GenerateKeyPair(p1, q1)
```

```
TASK 3 ------------------------------------------------------------------------------------------------------------------------------------------------------
Відкритий ключ абонента A:
(86284788930738019267013760989996609476192200171303514257040609501512740228849858910958922340767813498221503129401602784401678544948435202070757687569982 41, 65537)
Відкритий ключ абонента A (hex):
(A4BF2054A27FAC0FA3C277375A67A04B442E7D6EBF18845D377D793848BF98245B90E4C296724C924B6C22C38417AA00F1F220928E1ECD0CF3A7EBEB005A1061, 10001)
----------------------------------------
Відкритий ключ абонента B:
(86315208495254376033340022428404072369081752093755519860274654749827225593450130835781051168216749710138875807513411101688378700360106288468873038368140 13, 65537)
Відкритий ключ абонента B (hex):
(A4CDFEBC91A09E5BE892A627C0208279CDD287781A25FF25280A8184F4B913BA6DBF520BDC7F94100CA2EFEEC4DC53C7EAC2315A42034287BA9BE52FB58B4ABD, 10001)
----------------------------------------
Таємний ключ абонента A:
(64515117066451385021014210538044369707987399374310920634504233437495266745411126587509653786682976857856428880830209574835465334382896938652616513313949 85, 8172309470959442 1
Таємний ключ абонента A (hex):
(7B2E5514333457D8929F52D8568668FA1E0ECCC6C3A7A740B3BEE3A64BEEC5E823D55E793E694FB627F2FE0C88AFF4998134F80CE763968AB323E342A7B42DA9, B4ADA1E7801919162DA5E121AA69ACCDFF5E917E0C2
----------------------------------------
Таємний ключ абонента B:
(72791785683266156659393467805902996410680674374319450496017820366635655146318760071116200378130768673416535238601995421761425889256024048208454087602708 29, 9391328950178103 6
Таємний ключ абонента B (hex):
(8AFBE2395AC2CEEA09A02B19D8D051D8B0F83C20D0447D00F3DCB04FD52683F8597FFE70ECBD142EAFE54E70B7367619F58320F69929A4B7645F9411A7BA8FED, CFA109C6835074803DE52190DEB5CA3029DF7907E3 9
```

4. Функції шифрування, дешифрування, підписування та верифікації:

```python
def Encrypt(m, pubkey):
    n, e = pubkey
    c = pow(m, e, n)
    return c

def Decrypt(c, privkey):
    d, p, q = privkey
    m = pow(c, d, p * q)
    return m

def Sign(m, privkey):
    d, p, q = privkey
    s = pow(m, d, p * q)
    return s

def Verify(m, s, pubkey):
    n, e = pubkey
    v = pow(s, e, n)
    return v == m

m = random.randint(10**50,  10**60)
c1 = Encrypt(m, A_public)
m1 = Decrypt(c1, A_private)
c2 = Encrypt(m, B_public)
m2 = Decrypt(c2, B_private)
s1 = Sign(m, A_private)
s2 = Sign(m, B_private)
```

```
TASK 4 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Відкрите повідомлення: 255484001285841813193748811632962783914514963052318582616393
Відкрите повідомлення (hex): 28B3716EA1443720B18F0F014007FDEF080130D4116243B149
----------------------------------------
Шифротекст для A: 475385617333606903253903385800862775921300346371670632951145833350502522420894669401905036885279444042520068330758786817778904950582274495121791529536115
Шифротекст для A (hex): 5AC45D6D0D0625994C6627CE83B1C739001C9553BE1D6788A81560FD6B8CA0DBD4FF96813C5563A08849905BDE89F88460DCD53CA654AD29B628522B0A8AE87E
----------------------------------------
Розшифроване повідомлення для A: 255484001285841813193748811632962783914514963052318582616393
Розшифроване повідомлення для A (hex): 28B3716EA1443720B18F0F014007FDEF080130D4116243B149
----------------------------------------
Шифротекст для B: 504081147651347357298275639324880432865292197353336358468473954525987656951336176359393739921480980052272336393596528384981956895441725654683795706847020 5
Шифротекст для B (hex): 603EF926E6B6029D3C057A8FA3A925D98ECB87B13B46DF0ECCD525C4FFCA91A6C996D35AEA3331F65B4DDADD7E27F3EA82A5E7DF53B4701B388D4DA2056437BD
----------------------------------------
Розшифроване повідомлення для B: 255484001285841813193748811632962783914514963052318582616393
Розшифроване повідомлення для B (hex): 28B3716EA1443720B18F0F014007FDEF080130D4116243B149
----------------------------------------
Цифровий підпис абонента A: 67262794998027876237229568723381832570969206416557478540909942210122351476382653668572373616953149550211408562752315085220734020170467508621640985
Цифровий підпис абонента A (hex): 806D5E8CCE1A4C8213D78EDBE18AB2EE2FC17D6248DC911FBADD36C26004DB7F548AFF73B3547D10211044097999F7172CC8C5881598610C7F7946CA88E929D
Верифікація підпису для абонента A:  True
----------------------------------------
Цифровий підпис абонента B: 77663504546966334785252601824177200172997884260513219539262115998253959905660560099417800122900439217915001653814146337796897607467621991724085273
Цифровий підпис абонента B (hex): 944921590EFAD7F8A2FD00806A2C6E33CDEE5BDE0D34D939F7BA38BE43BBED50331E81E11B7C8ADDDB19F38355A721576100FC4D4D2846254C97AB0055DBE186
Верифікація підпису для абонента B:  True
```

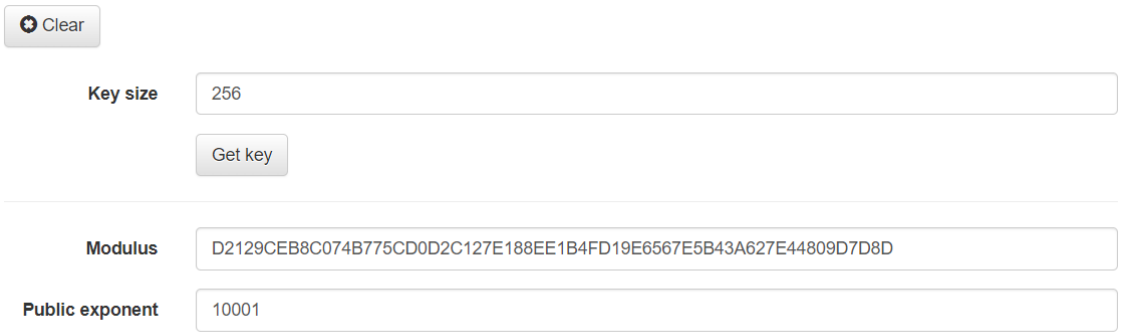
5. Функції надсилання та отримання ключа:

```python
def SendKey(k, B_public, A_private):
    k1 = Encrypt(k, B_public)
    s = Sign(k, A_private)
    s1 = Encrypt(s, B_public)
    return k1, s1

k = random.randint(1, A_public[0] - 1)
k1, s1 = SendKey(k, B_public, A_private)

def ReceiveKey(k1, s1, A_public, B_private):
    k = Decrypt(k1, B_private)
    s = Decrypt(s1, B_private)
    check = Verify(k, s, A_public)
    return k, check

k_res, check = ReceiveKey(k1, s1, A_public, B_private)
```

```
TASK 5 ----------------------------------------------------------------------------------------------------------------------------------------------------------------
Відправлений k1: 822118634476803916915429097076781987915465742690916497117396847315489594922694340515382993969754105745404377231081402048142684355859483688407651815056929 6
Відправлений k1 (hex): 9CF85260837DCDACE43EC2C26F617F0C92AA8C1A56B8FB108A72BE2FE945E63B6F914640820D36B1F9D105FF2ECC4DF060F0405B7F10C3997E48B3135A70FD50
----------------------------------------
Відправлений підпис s1: 2929213369105968291481662470751557768473390264454B1166019833932556364171127916221018384182110914617935123140329205531288114045417759812428576081906327
Відправлений підпис s1 (hex): 37EDB25AF496A03B5286802F54BA8E22B29DE1B9BD3B0BA909062414B4287C0A59C7BB8CE0DCF3177779B97254BB99A4E7B63E0F8B46DC17F37F8C034E2DB317
----------------------------------------
Отриманий k: 530664846429263008200472277765619696901474061080326015809983910724636128730333119908328627737765147167972121240036038944902489591229277119914742180494972 1
Отриманий k (hex): 65525B7D30BA4259AE7935467FFE6FFF8894BA6AEA6E56C550F1DF4F3EC5CBEA79F1F7D4BCEDEB18668F91FE221DBF49C05DF118FB504163C1096D961F4A70D9
Перевірка підпису: True
```

6. **Перевірка шифрування за допомогою http://asymcryptwebservice.appspot.com/?section=rsa**

Генеруємо ключ

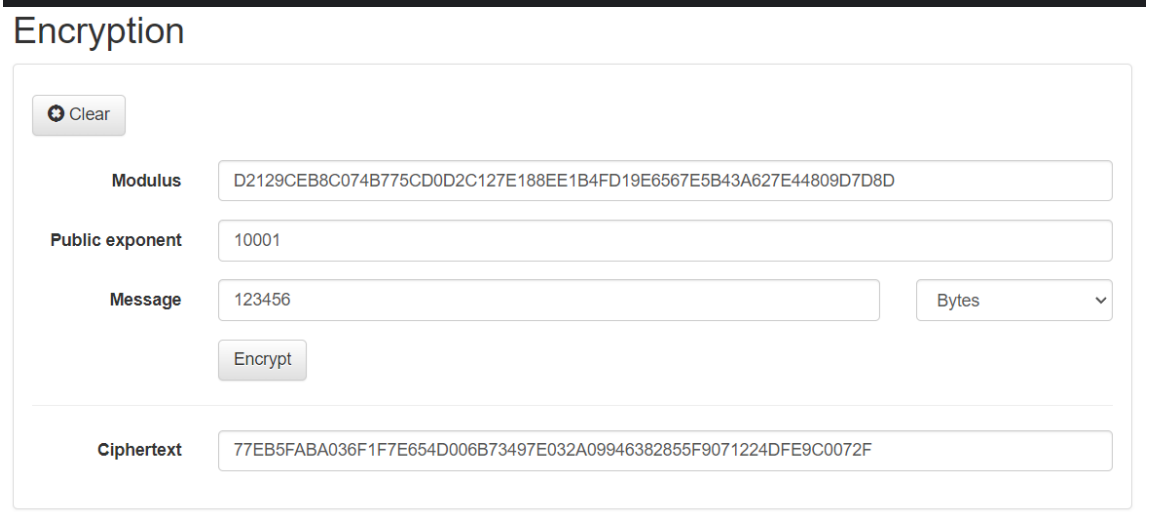


Записуємо його в код і шифруємо:

```python
def Encrypt(m, pubkey):
    n, e = pubkey
    c = pow(m, e, n)
    return c

n = int('D2129CEB8C074B775CD0D2C127E188EE1B4FD19E6567E5B43A627E44809D7D8D', 16)
e = int('10001', 16)
m = int('123456', 16)

A_public = (n, e)
c = Encrypt(m, A_public)
print("Шифротекст для A:", c)
print("Шифротекст для A (hex):", hex(c).upper()[2:])
```

Результат:

```
Шифротекст для A: 542410983345580155268503493994060120240447031411318487222193767208181269809 11
Шифротекст для A (hex): 77EB5FABA036F1F7E654D006B73497E032A09946382855F9071224DFE9C0072F
```



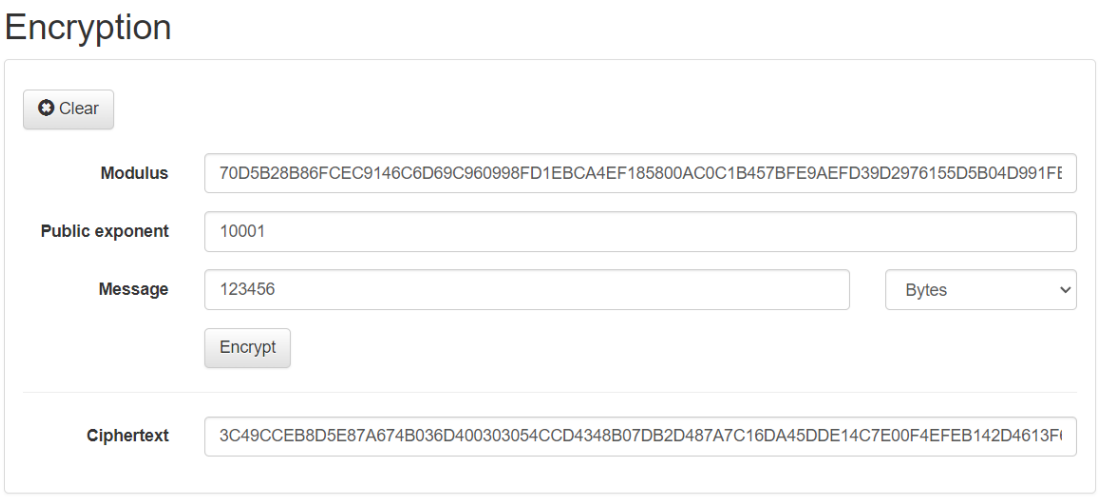
Перевірка дешифрування.

Згенеруємо ключ:

```python
def GenerateKeyPair(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = int('10001',16)
    d = pow(e, -1, phi)
    return (d, p, q), (n, e)

p, q = generate_prime(256), generate_prime(256)
A_private, A_public = GenerateKeyPair(p, q)

print(f"p = {p}")
print(f"q = {q}")
print('-' * 40)
print("Відкритий ключ абонента A (hex):")
print("(" + ", ".join(map(lambda x: format(x, 'X'), A_public)) + ")")
```

```
p = 79136224948021213494718846588753851226796486518108280320309030705511475779491
q = 74676744728551147756554419351947208262382653025549952175697849293644730061097
----------------------------------------
Відкритий ключ абонента A (hex):
(70D5B28B86FCEC9146C6D69C960998FD1EBCA4EF185800AC0C1B457BFE9AEFD39D2976155D5B04D991FE14553B726042 3F84233BD45127CF992C2BE3BD23DC1B, 10001)
----------------------------------------
Таємний ключ абонента A:
(42099740623580151193122320467973979329951944222214446841242413074310024994373919165879233367090599167686857914791161378903858791301236470903451154354744 33, 791362249480212134
----------------------------------------
```

Введемо його на сервер:



Дешифруємо результат:

```
def Decrypt(c, privkey):
    d, p, q = privkey
    m = pow(c, d, p * q)
    return m

c = int('3C49CCEB8D5E87A674B036D400303054CCD4348B07DB2D487A7C16DA45DDE14C7E00F4EFEB142D4613F6CAF5DB48A42D3423061334D78BF09B3DF639D480ADA8',16)
priv_key = (420997406235801511931223204679739793299519442222144468412424130743100249943739191658792333670905991676868579147911613789038587913012364709034511543547443, 791
m1 = Decrypt(c, priv_key)
print("Розшифроване повідомлення для A (hex):", hex(m1).upper()[2:])
```

```
Розшифроване повідомлення для A (hex): 123456
```

Перевірка підпису.

Підпишемо повідомлення:

```python
def Sign(m, privkey):
    d, p, q = privkey
    s = pow(m, d, p * q)
    return s

1 usage
def GenerateKeyPair(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = int('10001',16)
    d = pow(e, -1, phi)
    return (d, p, q), (n, e)

p, q = generate_prime(256), generate_prime(256)
A_private, A_public = GenerateKeyPair(p, q)
print("Відкритий ключ абонента A (hex):")
print("(" + ", ".join(map(lambda x: format(x, 'X'), A_public)) + ")")
print('-' * 40)
print("Таємний ключ абонента A:")
print(A_private)
print('-' * 40)
m = int('123456', 16)
s1 = Sign(m, A_private)
print("Цифровий підпис абонента A (hex):", hex(s1).upper()[2:])
```

```
Відкритий ключ абонента A (hex):
(702EEB216647AC9F6DDDBA8B64F771F90EDAA46F63DDADA6DB23D0A6CACE7FECFCF29C6EED4E0214518D0C4C28322A4A4D674AE161F31EF44C44300D82EC908B, 10001)
----------------------------------------
Таємний ключ абонента A:
(363771444540933466030166534213079075939790661457674498428724112326921736365863228763724092509720299763997513220798141046866502447031412363287364043663667 3, 650015812364074502
----------------------------------------
Цифровий підпис абонента A (hex): 35EC736B67957BEE713659443B270C2B707411FA85B2E503DE1CB204D24BB6B04DAC2E7515B135026DF3D6EBE6710ADDFF904FD13C31B47A6B94B33A6F3AFE49
```

Verify

Clear

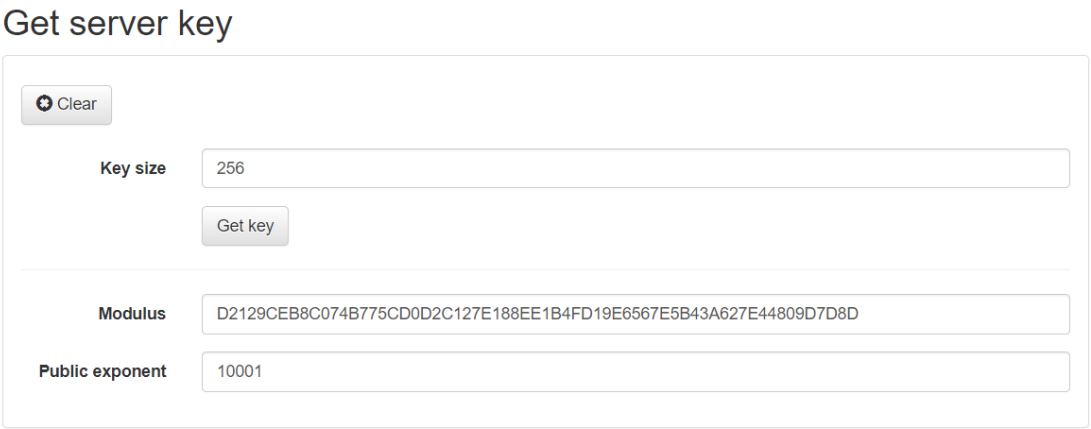
Message	123456	Bytes
Signature	35EC736B67957BEE713659443B270C2B707411FA85B2E503DE1CB204D24BB6B04DAC2E7515B135026DF3D	
Modulus	702EEB216647AC9F6DDDBA8B64F771F90EDAA46F63DDADA6DB23D0A6CACE7FECFCF29C6EED4E021451	
Public exponent	10001	

Verify

Verification true ✔

Перевірка верифікації.

Генеруємо ключ та підписуємо повідомлення:




```
m = int('123456',16)
s = int('41226C27C239B942A27D0714D982356A9C0A7533AB3E7DC5ACCEA75656B8B072',16)
n = int('D2129CEB8C074B775CD0D2C127E188EE1B4FD19E6567E5B43A627E44809D7D8D',16)
e = int('10001',16)

1 usage
def Verify(m, s, n, e):
    v = pow(s, e, n)
    return v == m

print(Verify(m, s, n, e))
```

```
True
```

Висновок

У ході виконання даного практикуму було реалізовано основні алгоритми та процедури криптосистеми RSA - генерацію ключів, шифрування, розшифрування, підпис та верифікацію. Також було продемонстровано застосування RSA для безпечного обміну ключами між абонентами. Робота дозволила отримати практичні навички використання асиметричних криптосистем.