Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4
з дисципліни «Криптографія»

Виконав:

студент 3 курсу

НН ФТІ групи ФБ-25

Черняк Денис

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

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.
Хід 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

```python
def is_prime(p, primes=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47], k=10):
    for prime in primes:
        if p % prime == 0:
            return False

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

    for _ in range(k):
        x = random.randrange(2, p)
        y = pow(x, d, p)

        if gcd(x, p) != 1:
            return False

        if y in (1, p - 1):
            continue

        for _ in range(s - 1):
            y = pow(y, 2, p)
            if y == 1:
                return False
            elif y == p - 1:
                break
        else:
            return False
    return True
```

```python
def GeneratePrime(bits):
    while True:
        x = random.getrandbits(bits) | (1 << (bits - 1)) | 1

        if is_prime(x):
            return x
        else:
            print(f"Кандидат {x} не є простим числом")
```

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 <= p1q1 ; p і q – прості числа для побудови ключів абонента A, p1і q1 – абонента B.

```python
bits = 256

while True:
    p = GeneratePrime(bits)
    q = GeneratePrime(bits)
    p1 = GeneratePrime(bits)
    q1 = GeneratePrime(bits)
    print(f"Перевірка потенційних пар простих чисел: p={p}, q={q}, p1={p1}, q1={q1}")

    if p * q <= p1 * q1:
        break
```

```
p=655912456017684411688876707286354032744716413921458494309008665286384083400 81, q=648939714631719841656838980550648549545679121705949431947117366571713215763 81
p1=107740228391240460592605454953777649222984130858105416316492715152871651856499, q1=115742475952286724694221719023248338316218497255350632599297569798007447167859
```

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), (e1, n1) та секретні d і d1.

```python
def GenerateKeyPair(p, q):
    n = p * q
    f = (p - 1) * (q - 1)
    e = 65537
    d = pow(e, -1, f)
    return ((n, e), (d, p, q))
```

```
Ключі абонента A:
Відкритий ключ (n, e): (42564764203150661391987786608907197057834584200834610546867864425183750566354485479260574753557435808891636028155090173729585320591093814475824544652208 61, 65537)
Секретний ключ (d, p, q): (15979728374114147319658627976952754572393641907278861677192477788753177578078850971946052509182260897757745474885838011240736385269416739624235236087 0273, 6559124560176844116888767072863540327447164

Ключі абонента B:
Відкритий ключ (n, e): (12470120793667028441629897130558921418069024483965177169274498922098986569219075035916487336045374484910864423031660929076649974693659840454459957033065641, 65537)
Секретний ключ (d, p, q): (11064551690674545735703167953095217670334525063560661189760167726933733143111161905782781672722494122364288449854520017592846766852188328995177577274894953, 1077402283912404605926054549537776492229841
```

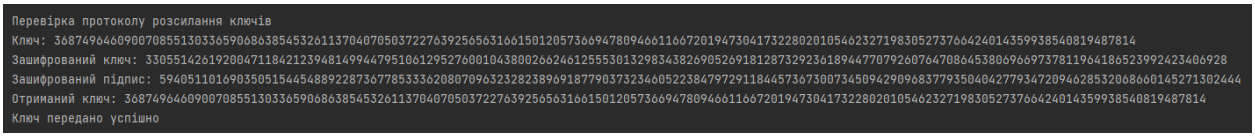
4.Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання.

```python
def Encrypt(message, public_key):
    n, e = public_key
    return pow(message, e, n)

def Decrypt(ciphertext, private_key):
    d, p, q = private_key
    n = p * q
    return pow(ciphertext, d, n)

def Sign(message, private_key):
    d, p, q = private_key
    n = p * q
    return pow(message, d, n)

def Verify(message, signature, public_key):
    n, e = public_key
    return pow(signature, e, n) == message
```

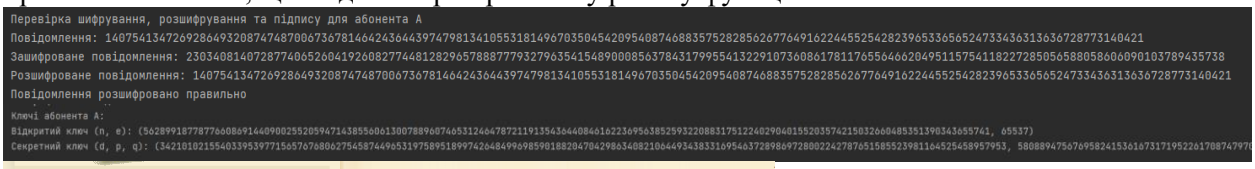
```
Перевірка шифрування, розшифрування та підпису для абонента A
Повідомлення: 2825682470092735238276710132173472110270990615305772202332658492219369305684095272617387710166422148882765486628682836385836728228675851305707681076513 59
Зашифроване повідомлення: 1593543118498211626589901691912754244389586244458059829488054079978345829287961433817795871865749497993630132817014135992052605853910096120111541763773832
Розшифроване повідомлення: 2825682470092735238276710132173472110270990615305772202332658492219369305684095272617387710166422148882765486628682836385836728228675851305707681076513 59
Повідомлення розшифровано правильно
Цифровий підпис: 3207991420139171490268539864398220108119969588542691956282647159637305705958835115229415919146644229496743804551730943123873582214410095228751761389474573
Підпис дійсний
Перевірка шифрування, розшифрування та підпису для абонента B
Повідомлення: 6682677235237046817337297491103165346768877865648950896157015596105979161679418590476008536550957789622806965785720927837588146066978468745340883495955674
Зашифроване повідомлення: 8430250194935758276622398794615302545062625652269719395184832911572396648126176564200047018335126963036818294340264934232367504247768339087244869959525 24
Розшифроване повідомлення: 6682677235237046817337297491103165346768877865648950896157015596105979161679418590476008536550957789622806965785720927837588146066978468745340883495955674
Повідомлення розшифровано правильно
Цифровий підпис: 2659374681760764439596055825519186590107977846225397781558912966802263508257257366647334202768096996775957411886095806811392519558933102557700682720679311
Підпис дійсний
```

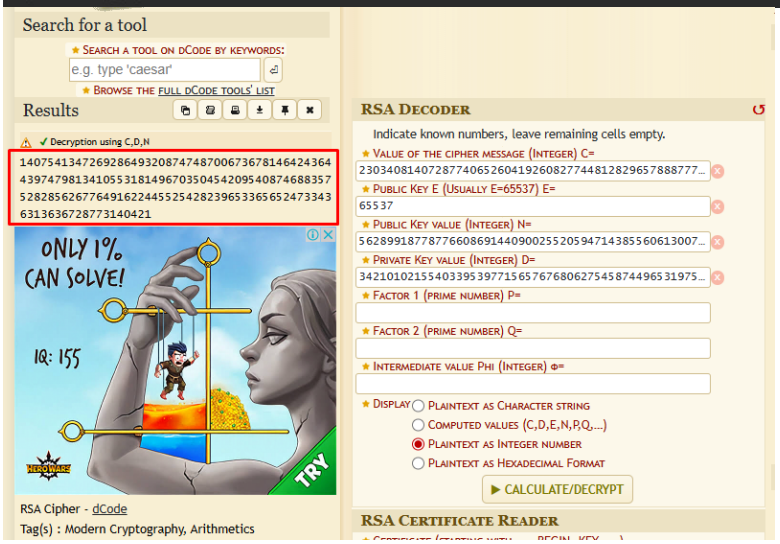
5.За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа $0 < k < n$.

```python
def SendKey(key, sender_private_key, receiver_public_key):
    k1 = Encrypt(key, receiver_public_key)
    signature = Sign(key, sender_private_key)
    signature1 = Encrypt(signature, receiver_public_key)
    return k1, signature1

def ReceiveKey(encrypted_key, encrypted_signature, receiver_private_key, sender_public_key):
    k = Decrypt(encrypted_key, receiver_private_key)
    signature = Decrypt(encrypted_signature, receiver_private_key)
    if Verify(k, signature, sender_public_key):
        return k
    return None
```



Перевіряю на сайті: https://www.dcode.fr/rsa-cipher
Як можна побачити, повідомлення зашифроване абонентом А, правильно розшифровується його приватним ключем, що свідчить про правильну роботу функцій:





Висновок:

У ході виконання лабораторної роботи я ознайомився з принципами функціонування криптосистеми RSA та алгоритму цифрового підпису. Реалізував функції для генерації простих чисел за допомогою тесту Міллера–Рабіна, створення ключових пар RSA (відкритого та закритого ключів), а також функції шифрування, розшифрування, підписання та перевірки цифрового підпису. Також було реалізовано протокол конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу.