

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Криптографія
Лабораторна робота №4
Варіант 1

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

Виконали:
студенти 3 курсу ФТІ
групи ФБ-05
Качур Ілля Ковальов Данііл

Перевірила:
Селюх П.В.

Київ – 2022

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 зашкерееного зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

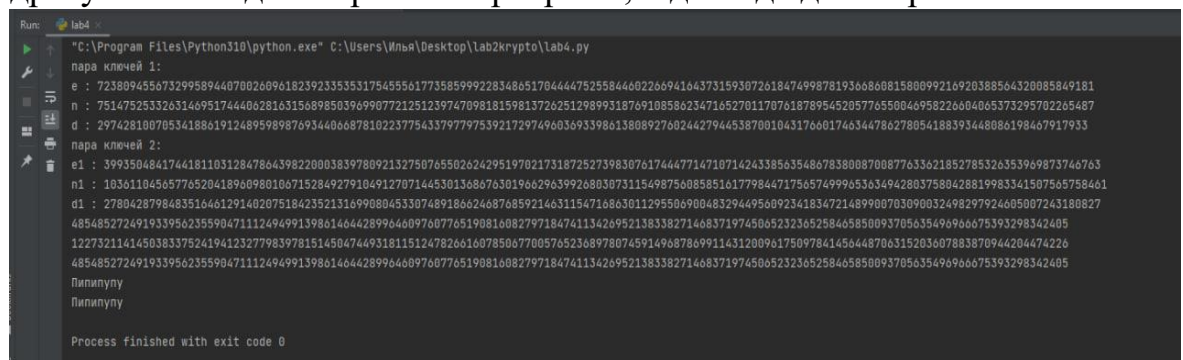
Постановка задачі

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 p_1, q_1 , p, q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p_1q_1$; p і q – прості числа для побудови ключів абонента А, p_1 і q_1 – абонента В.
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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , (e_1, n_1) та секретні d і d_1 .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 А і В, перевірити правильність розшифрування. Скласти для А і В 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 $0 < k < n$. Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція `Encrypt()`, яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: `GenerateKeyPair()`, `Encrypt()`, `Decrypt()`, `Sign()`, `Verify()`, `SendKey()`, `ReceiveKey()`

Хід роботи:

1. Із тестом Міллера-Рабіна були труднощі, важко зрозуміти було алгоритм роботи, через те що є багато потрібних кроків які плутали нас по ходу виконання.
2. Згенерували прості числа, прикріплюю:
Abonent a-
(81396464488284250301106248806993989400089559858023686139697821246942365117497,
115148115947708827990720226878931510361227217733056371507849470626294143347091)
Abonent b-
(101466692580136235453411273357081425385801915309410009764744604784046688168237,
101406899543669370215218980305628353937571149559180213492095772618306858939183))
3. У ході самого виконання були проблеми з шифруванням, бо ми не знали що шифрування відбувається з допомогою відкритих ключів для того кого шифрують (із цим нам підказали та врятували). Також дратувала швидкість роботи програми, відповідь даю скріном:



```
Run: lab4
"C:\Program Files\Python310\python.exe" C:\Users\Міня\Desktop\lab2krypto\lab4.py
пара ключей 1:
e : 72380945567329958944070026096182392335353175455561773585999228348651704447525584602266941643731593072618474998781936686081580099216920388564320085849181
n : 751475253326314695174406281631568985039699077212512397470981815981372625129899318769108586234716527011707618789545205776550046958226604065373295702265487
d : 2974281007053418861912489598987693440668781022377543379779753921729749603693398613808927602442794453870010431766017463447862780541883934488086198467917933
пара ключей 2:
e1 : 3993504841744181103128478643982200038397809213275076550262429519782173187252739830761744477147107142433856354867838008700877633621852785326353969873746763
n1 : 10361104565776520418960980106715284927910491270714453013686763019662963992680307311549875608585161779844717565749996536349428037580428819983341507565758461
d1 : 2780428798483516461291402075184235213169908045330748918662468768592146311547168630112955069084832944956092341834721489900703090032498297924605007243180827
4854852724919339562355904711124949913986166442899646097007765190816082797184741134269521383382714683719745065232365258465850093705635496966675393298342405
1227321141650383375241941232779839781514504744931811512478266160785067700576523689780745914968786991143120096175097841456448706315203607883870944204474226
4854852724919339562355904711124949913986166442899646097007765190816082797184741134269521383382714683719745065232365258465850093705635496966675393298342405
Пилипуну
Пилипуну
Process finished with exit code 0
```

Перевірка системи RSA

Get server key

✖ Clear

Key size

256

Get key

Modulus

BDAFB62ED0B290497205D4768C4BE3F9ED6F0FE125A77B268F2871F7DB813E47

Public exponent

10001

Encryption

✖ Clear

Modulus

BDAFB62ED0B290497205D4768C4BE3F9ED6F0FE125A77B268F2871F7DB813E47

Public exponent

10001

Message

42

Bytes ▼

Encrypt

Ciphertext

7DD95417588D2D3272D294856681467C317FFF19BE570EAEF11563FBF86EA074

Decryption

✖ Clear

Ciphertext

7DD95417588D2D3272D294856681467C317FFF19BE570EAEF11563FBF86EA07

Bytes ▼

Decrypt

Message

42

Sign

Message

42

Bytes

Signature

2DE786C271DB6F9E96CD9DECB722BD3C7BB9DC3B0712BB3E44D1C14A1E7E6AF5

Verify

Message

42

Bytes

Signature

2DE786C271DB6F9E96CD9DECB722BD3C7BB9DC3B0712BB3E44D1C14A1E7E6AF5

Modulus

BDAFB62ED0B290497205D4768C4BE3F9ED6F0FE125A77B268F2871F7DB813E47

Public exponent

10001

Verification

true

✓

Перевіряю правильність коду

```
final()
# exp = '10001'
mod = 'BDAFB62ED0B290497205D4768C4BE3F9ED6F0FE125A77B268F2871F7DB813E47'
#mod = '42'
n = int(mod, base=16)
#85797583998544640711737405761966135740046954643331660431393077356296392490567 mod 16
#print(n)
e = int('10001', 16)
b = e, n
message = 66 #42 в 16
print(Encrypt(b,66)) # 56923092262062670029369080183419769797982230129887409317214508866303825977460 answer
message_2 = int('2DE786C271DB6F9E96CD9DECB722BD3C7BB9DC3B0712BB3E44D1C14A1E7E6AF5', 16)
print(Verify(b, message, message_2))
# cyphertext = '7DD95417588D203272D294856681467C317FFF19BE570EAEF11563F8F86EA074'
# decrypted = '42'
# signature = '2DE786C271DB6F9E96CD9DECB722BD3C7BB9DC3B0712BB3E44D1C14A1E7E6AF5'
```

```
56923092262062670029369080183419769797982230129887409317214508866303825977460
True
```

Та додатково

Исходное основание

10

Основание системы счисления исходного числа

Исходное число

5692309226206267002936908018341976979798223

Число которое необходимо преобразовать

Основание результата

16

Основание системы счисления переведенного числа

Переведенное число

7DD95417588D2D3272D29485668146

7C317FFF19BE570EAEF11563FBF86E

A074

Як бачимо усе збігається, значить усе тіп топ

Висновок

У ході виконання ми ознайомились з тестом Міллера-Рабіна, та асиметричною криптографією, а саме криптосистемою RSA.