

Міністерство освіти і науки України

Національний технічний університет  
України "Київський політехнічний інститут  
імені Ігоря Сікорського"

Фізико-технічний інститут

# Криптографія

Лабораторна робота №4

Виконав студент групи ФБ-13

Нійозов Рустам

Київ 2023

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

*Мета та основні завдання 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 і рекомендації щодо виконання 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.
2. За допомогою цієї функції згенерувати дві пари простих чисел  $p, q$  і  $1 < p, q$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p \neq q$ ;  $p \neq 1$ ;  $q \neq 1$  – прості числа для побудови ключів абонента А,  $1 < p < q$  – 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  $(e, n)$ ,  $(d, p, q)$  і 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(). Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою

<http://asymcryptwebservice.appspot.com/?section=rsa>.

Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

### Хід роботи:

1. Із тестом Міллера-Рабіна проблем багато не виникло. Для перевірки чи правильно зроблено тест, було використано сайт: <https://planetcalc.ru/8995/>

2. Згенерував числа, так, щоб  $pq \leq p_1q_1$ :

$p = 114893576615012767532631617462266153531956267357783648861491082899568397195091$

$q = 58900183899818136254803171219179457679020479081697934423863264607746985268011$

$p_1 = 112988788668750763609699926596140978736789755768783595000344553991545243073851$

$q_1 = 113329302157219497734891659533221069041902924052190802623646108652814389334531$ .

```
Ключі для абонента А:
e : 548848422418996062495089148477084702058954251492622100854654701695303018647667245272344361226607040778538938920915429711711645734599948716253451370108309
n : 5776652998028945879143037534466072593348237941885877611128385733878086272215701709838134684636447626992338453384372440068102673357895225856604895722414910
d : 5635433351151399782244711914198570411493922718998230959199763608887139982225703380529292862764493092751504953008973169373641519247802414250891097943074429
Ключі для абонента В:
e1 : 4910482717211608771597943683709543277099624464813473845024758038244921721873238352625676568769941984170694392351889579890686916125303502912683883362982663
n1 : 6075215413231783801888152898588052578227769128836765921205952323845087129937342509445887205625363107385539749627674161951279527502340146009872476949280613
d1 : 77715469673190082403743337302338321198394750730809819347469429441137324594273879419820669781935450688358498550939107398880125073688800676865828332666279
Message: 64029779378354421335813372386423114703471661547956615893337914215314554952658936298392952548061088274981678455783047870325730402515233716474478497315642
Encrypted message: 1129232939497407213171801478089927172699355901902114340997922096881082906887412653827948065810749367658210779059998827774594015571932940184042241191201535
Decrypted message: 64029779378354421335813372386423114703471661547956615893337914215314554952658936298392952548061088274981678455783047870325730402515233716474478497315642
Key k: 4337940918961117558795295893153051764390809641449998406529409327888196835243951601669214530295063058183911285877981184169131384589301662176658025125314687
(579674711339238002258132578928098786774504911550155241800295113074319529278962337120748393584836066338589940258135737126298773381239209075140887541470671, 20471032456911883093640582086472906959717691
00260448752111839170691156567666243472560067916831361621472387308019981301128651183622677126328164019071560009)
Authentication succeeded.
(1129232939497407213171801478089927172699355901902114340997922096881082906887412653827948065810749367658210779059998827774594015571932940184042241191201535, 64029779378354421335813372386423114703471661
547956615893337914215314554952658936298392952548061088274981678455783047870325730402515233716474478497315642)
Verification successful.
64029779378354421335813372386423114703471661547956615893337914215314554952658936298392952548061088274981678455783047870325730402515233716474478497315642
239084941111324639416817542337403103810
1
```

перевірка на сайті:

## Get server key

Key size

128

Get key

Modulus


CBEE2A447C72D371B704C18C72128ABD

Public exponent

10001

Створено ключі для Абонента

## Encryption

 Clear

Modulus

CBEE2A447C72D371B704C18C72128ABD

Public exponent

10001

Message

15

Bytes

Encrypt

Ciphertext

B3DE15B11DDE107F56E6FFB0F1A12A42

Зашифровано повідомлення 21, що у 16-ковому представленні буде 15

За допомогою коду отримано наступне:

```
n = 271069907083902572793476553998716799677
# CBEE2A447C72D371B704C18C72128ABD (у 16-ковій)
e = int('10001', 16)
some_message = 21 #(15 у 16-ковій)
A = SubscriberKey('A', e, n, d=None)
print(A.encrypt(21))
# 239084941111324639416817542337403103810
```

239084941111324639416817542337403103810

PS: C:\Program Files\Криптографія\lab\11\lab4\

# Decimal to Hexadecimal converter

From

Decimal

▼

To

Hexadecimal

▼

Enter decimal number

23908494111132463941681754233740310:

10

= Convert

✕ Reset

↕ Swap

Hex number (32 digits)

B3DE15B11DDE107F56E6FFB0F1A12A42

16

Як видно, це те саме число, що й на сайті

## Decryption

✖ Clear

Ciphertext

B3DE15B11DDE107F56E6FFB0F1A12A42

Bytes

▼

Decrypt

Message

15

## Sign

✖ Clear

Message

15

Bytes


▼

Sign

Signature

4B6CB046FB009D1F4278AC84EF27A871

## Verify

 Clear

Message

15

Bytes

Signature

4B6CB046FB009D1F4278AC84EF27A871

Modulus

CBEE2A447C72D371B704C18C72128ABD


Public exponent

10001

Verify

Verification

true




```
signed_message = int('4B6CB046FB009D1F4278AC84EF27A871', 16)
print(A.verify(some_message, signed_message))
```

Вивід:

```
259084941111524659410817542557405105810
1
PS C:\3_kurs\криптографія\lab\11lab4>
```

## Send key

 Clear

Modulus

CBEE2A447C72D371B704C18C72128ABD

Public exponent

10001

Send

Key

981DFB007ED61D81754EF79CC569C0CF


Signature

5B2156CF227A1C05

## Receive key

Key	981DFB007ED61D81754EF79CC569C0CF
Signature	5B2156CF227A1C05
Modulus	CBEE2A447C72D371B704C18C72128ABD
Public exponent	10001

---

Key	5B2156CF227A1C05
Verification	true 

### Висновок:

По ходу роботи я ознайомився та використав на практиці тест Міллера-Рабіна для перевірки текстів на простоту. Також дізнався про методи генерації ключів для криптосистеми RSA. Практично по-працював з системою RSA організував секретний зв'язок та обмін даними за електронним підписом. Перевірів правильність нашої системи завдяки онлайн ресурсу.