**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

# КРИПТОГРАФІЯ

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

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

**Варіант 5**

Виконали:

студенти гр. ФБ-11

Цема В.В.

Ципун Р.Г.

Перевірила

Селюх П. 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. 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 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 – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , ( , ) 1 n1 e та секретні d і d1
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. 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 $0 < k < n$.

## Хід роботи

Реалізовано спілкування з сервером(автоматичне в коді).

У **GET rsa/encrypt** ми відправляємо свій публічний ключ та текст у відповідь отримуємо зашифрований текст від сервера і розшифровуємо його своїм приватним ключем.

У **GET rsa/decrypt** ми отримуємо публічний ключ, відправляємо текст зашифрований отриманим публічним ключем, потім сервер розшифровує його своїм приватним ключем та відправляє результат.

У **GET rsa/sendKey** ми отримуємо публічний ключ від сервера, потім відправляємо йому свій публічний, далі сервер генерує рандомний ключ який підписує своїм приватним та шифрує нашим публічним, далі ми його отримуємо зашифроване повідомлення та розшифровуємо його нашим приватним ключем та перевіряємо його підпис за допомогою публічного ключа сервера.
У **GET rsa/receiveKey** аналогічно попередньому тільки навпаки, ми шифруємо, а сервер розшифровує та перевіряє підпис нашим публічним ключем.

```
+----------------------------------------+
|    Communication with test server      |
+----------------------------------------+
```

**GET rsa/serverKey:**

**[*] Server PublicKey: (107464227467192207278250860640564362747294135989074029995671612725891427103289, 65537)**

**GET rsa/encrypt:**

**[*] Alice PublicKey: (28948159952287608483212848226456051925021177033883163454457603911424010127043, 65537)**
**[*] Message (bytes): b'Hello, Biden! I love RSA'**
**[*] Server received public key and starts encrypting...**
**[*] Server response: {'cipherText': '0FFEBA466BBCAE5B38483A578A13059B50610D8B618B5E660078CF9B2994D347'}**
**[*] Decrypted message (long): 177514930725334766330658296572186733075692323754153759553**
**[*] Decrypted message: Hello, Biden! I love RSA**

**GET rsa/decrypt:**

**[*] Server PublicKey: (107464227467192207278250860640564362747294135989074029995671612725891427103289, 65537)**
**[*] Message (bytes): b'Biden, do you love RSA?'**
**[*] Encrypted message (long): 24339808825232608147308582487517588798297369869591584123936959413865778301 16**
**[*] Server response: {'message': 'Biden, do you love RSA?'}**

**GET rsa/sign:**

**[*] Server PublicKey: (107464227467192207278250860640564362747294135989074029995671612725891427103289, 65537)**
**[*] Message (bytes): b'Biden'**
**[*] Server response: {'signature': 'B0CF1A856F4DFFBDF8FADBEE18C566E1906B8253DFEC0D5275C79086D4ECB0FA'}**
**[*] Signature verification: True**

**GET rsa/verify:**

**[*] Alice PublicKey: (28948159952287608483212848226456051925021177033883163454457603911424010127043, 65537)**
**[*] Message (bytes): b'Hello! Biden!'**
**[*] Alice signed message: (573581676307306226155360861750 5, 21467580260633122884097801001709057289110004317971400782706930851533756460661)**
**[*] Server received signature and starts verification...**
**[*] Server response: {'verified': True}**

**GET rsa/sendKey:**

**[*] Server PublicKey: (107464227467192207278250860640564362747294135989074029995671612725891427103289, 65537)**
**[*] Alice PublicKey: (28948159952287608483212848226456051925021177033883163454457603911424010127043, 65537)**

**[*] Server sent encrypted key: {'key': '379CB8FE2540AAC8D0F12AB4818F877EC00629E75DC689BCE81D879F0FECEAE4', 'signature': '2A31E3F4F75555BE536B9F03104C87C445D5BB065E6C60BB28DCD2D0389C2DB1'}**
**[*] Signature verification: True**
**[*] Decrypted key: 12066838560286176212**

**GET rsa/receiveKey:**

**[*] Server PublicKey: (107464227467192207278250860640564362747294135989074029995671612725891427103289, 65537)**
**[*] Alice PublicKey: (28948159952287608483212848226456051925021177033883163454457603911424010127043, 65537)**
**[*] Alice signed key: (1337228322, 221258789830876161857641271926351309051527812616357095827426353718801606470 47)**
**[*] Alice encrypted key with server public key: ('c4b9f5e5a00b5347fcaadebfff8396b03142d78af92f631491094518e29b6909', '31032cbbbc39db2fdd08b93f27a7bae2f9183ee7e1e4d69aaccde114cf2a0444')**
**[*] Server received key and starts verification...**
**[*] Server response: {'key': '4FB47C22', 'verified': True}**
**[*] Key: 1337228322**

**Висновки:** У ході виконання даної лабораторної роботи, ми зрозуміли, як працює криптосистема типу RSA, навчились організовувати засекречений зв’язок і генерувати цифровий підпис з використанням цієї системи. Ми також ознайомились з рядом сучасних методів, що використовуються для тестування великих чисел на простоту і навчились використовувати її на практиці