



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»  
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ  
Кафедра Інформаційної Безпеки

## Криптографія

### Комп'ютерний практикум №4

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

##### ***Мета:***

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.

Перевірів:

\_\_\_\_\_

Виконали:

студенти II курсу

групи ФБ-01

Приходько І.Ю.

та Сахній Н.Р.

Київ 2022

ФБ-01 (Приходько Ігор та Сахній Назар)

## 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  $p_1, q_1$  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  $A$ ,  $p_1$  і  $q_1$  – 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  $(e, n)$ ,  $(e_1, n_1)$  та секретні  $d$  і  $d_1$ .
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  $A$  і  $B$ , 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  $0 < k < n$

--	--	--

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 зашифрувати власною реалізацією повідомлення та розшифрувати його на сервері. А ось для того, щоб перевірити процес розшифрування, необхідно зашифрувати на сервері повідомлення та розшифрувати його локально за допомогою реалізованої власної програми.

## Хід роботи:

### 1. Параметри криптосистеми RSA для абонентів А і В

```
Run: Python Crypto_CP4 X
D:\KPI\Криптографія\venv\Scripts\python.exe D:/KPI/Криптографія/Crypto_CP4.py
Enter size value of prime number in bits:
>>> 256
```

#### 1.1. Дві пари великих випадково згенерованих простих чисел $p$ , $q$ та $p_1$ , $q_1$

```
[x]> Pair of prime numbers for user A
p = 73658276694353937118748368055007068495544245084587683525780165251043332960347
q = 90800893407433479359562734507428754067437046774732317962563954865575438337693

[x]> Pair of prime numbers for user B
p1 = 102291923378229545379170139715309800863431153387856439694634952839680326220979
q1 = 71704808744587266076116508744254837884781035588443982139709780823452442907139
```

#### 1.2. Пари чисел $n$ , $e$ та $n_1$ , $e_1$ – відкриті ключі, а $d$ , $d_1$ – секретні ключі

```
[***]> Generation RSA key pair for user A
• Public key of user A:
e = 5270023878744421694744650594986873481822573605741604423129772654864030161878419014628155709763616295410276088225247743497212826901161013148792955722684229
n = 668823730699273505658214506652910226783376935263070191576102318939822175410423317672185342977996780578615151274695598133993075404627760290767394364459471
• Secret key of user A:
d = 5079785859191561769949430918751633160948393653388365451217032890805803804968055714413761755723465144967313528495635045389402102552937545036679978333530693
p = 73658276694353937118748368055007068495544245084587683525780165251043332960347
q = 90800893407433479359562734507428754067437046774732317962563954865575438337693

[***]> Generation RSA key pair for user B
• Public key of user B:
e1 = 5291953132109432363495538702964525739991872485343669719370762194494739085148172594478778465063391853938498805086535175860803832388352107212362267379982817
n1 = 73348228019519245012042072622043466783677485948500949072002014314868968931910022321315950693379564765214247660188500841859740758026308467650628490669081
• Secret key of user B:
d1 = 652422293622976249792861380050115657510079114766452064666849576864258698878414195353173173773736783012228636614680572205375473170587613804823669234718493
p1 = 102291923378229545379170139715309800863431153387856439694634952839680326220979
q1 = 71704808744587266076116508744254837884781035588443982139709780823452442907139
```

#### 1.3. Чисельні значення прикладів ВТ, ШТ, цифрового підпису для А та В

```
Message --> 1817196599923362879975509007400366641185522376054168333156645469855048367873087443066562529902146188836000612659275537791512765787225986992724456527002

• User B encrypt message for user A: 11126706202270680562797926599939504086323744576599943633015124845640655189418051523581632284088616439420399751108190307636735917355746623861463174675627
[!] Digital signature of user B: 6936419080019040012358231840184664521641702451052319271120678988367728700025989394264468861994051780317077506792703644581221674566555134630085858376277
• User A encrypt message for user B: 1944812090673021786257202583951589135391042352212716817839881445828501728466192128035992778037214214508671528506101099989738956512367369852826791421
[!] Digital signature of user A: 1496144513245117328968200078140696135867521291528666584908994100104109116748956238621258580998658500503258305009481755238670747737026425518416201060340

[*] User A decrypt message from user B after successful signature verification: 1817196599923362879975509007400366641185522376054168333156645469855048367873087443066562529902146188836000612659275537791512765787225986992724456527002
[*] User B decrypt message from user A after successful signature verification: 1817196599923362879975509007400366641185522376054168333156645469855048367873087443066562529902146188836000612659275537791512765787225986992724456527002
```

#### 1.4. Протокол розсилення ключів з підтвердженням справжності, чисельні значення характеристик на кожному кроці

```
Key --> 31413346404592198409510405332484831760006305390675482093884904992618251083683379461967467236997722785109803755879566924359083389625293188352649524466887

> User A forms message (M1, S1) => (173817473154934518731146961662546061282116648789882745596263987488966549142569575743641232300512884249931089451269848928584215720984460334729396363652605, 157850968922321981796221665584298005859545292152690576064450)
> User B using own secret key finds shared key <
[*]- Key was successfully received from authenticated user: 31413346404592198409510405332484831760006305390675482093884904992618251083683379461967467236997722785109803755879566924359083389625293188352649524466887

> User B forms message (M1, S1) => (3257739412373027368486810435956211323317988051044941023486649027274132996974930106381358575188718921199925484996630408365593743557292736915424508333226, 6109225697094739311742201281628376738546390032966999816768150)
> User A using own secret key finds shared key <
[*]- Key was successfully received from authenticated user: 31413346404592198409510405332484831760006305390675482093884904992618251083683379461967467236997722785109803755879566924359083389625293188352649524466887

Process finished with exit code 0
```

## Висновки:

У ході виконання лабораторної роботи були здобуті практичні навички захисту інформації на основі криптосхеми RSA. З використанням цієї системи було організовано засекречений зв'язок та процес верифікації повідомлення за цифровим підписом користувача.

Ми дізналися про алгоритми перевірки числа на простоту, використовуючи при цьому відповідні тести перевірки чисел на простоту. А також ознайомилися з методами генерації ключів для асиметричної криптосистеми типу RSA.

За протоколом розсилання ключів була перевірялася конфіденційність надісланої інформації та автентичність відправника.