

**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**

Виконали:

студенти гр. ФБ-14

Цибулено-Сігов І. М.

Татаренко А. О.

Перевірила

Селюх П. В.

Київ 2023

## 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 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  $1 < p < q$ ;  $p$  і  $q$  – 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  $(e, n)$ ,  $(e, n)$  і  $1 < e < n$  та секретні  $d$  і  $1 < d$ .
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  $0 < k < n$ .

При генерації підходящих під умови криптосистеми випадкових простих чисел зіткнулися з проблемою, коли такі 4 числа генерувалися дуже довго. Це було пов'язано з тим, що генерація останнього числа -  $q_2$  залежала від попередніх чисел, і це число могло мати дуже малий інтервал, тільки попавши в який, воно б задовільнило рівняння  $p \cdot q < p_1 \cdot q_1$ . Вирішили цю проблему обмеженням кількості регенерацій числа  $q_2$  (5), якщо за таку кількість  $q_2$  не задовольнятиме рівняння, всі числа будуть регенеровані.

Через умову криптосистеми щодо  $p$   $q$  та  $p_1$   $q_1$ , кандидати можуть відкидатися:

```
Не пройшов q2 - 80134130270402594791740897332352202994236215239971943436339165285193902610423
Не пройшов q2 - 42039859030633443063851639058323548946666709122867443708426234008238286801339
Не пройшов q2 - 91507773162476816070272939056889164418829096231538599195577484375748357128611
Не пройшов q2 - 79162912736260702962844790083935057847586757596308712920721035291770933734723
Не пройшов q2 - 88996286057809281532230717002127299053467662220811043324650508257729375222243
Не пройшли:
p1 - 44415751289139210982411491570579972112453785533325599486655748339256589839759
q1 - 96085064245935949831893874137027458676171358728043968112523223926612439050987
p2 - 25184690510580494655624164489673533868199575447636937246746625318682471398283
```

В результаті, отримали пари ключів:

```
p = 54096515714858767869060490161884628381484798405909961334005818854896734155263
q = 105932333402322260306492737095672084874380938216814489499485139843186256832323
Public key A: [4745213874248839508210974984093027427981022283928761030086140801156643292747360079188215851
Private key A: [540678243594083348028824222169980460367715790893579843144514543694285473497223352241785831
p = 95504194371631507434182771786847392898508845554858822865728493648973593947671
q = 62035423351955603901592320145511541745286458219792982762070983430161438387459
Public key B: [1652008258348724429237157065670755554506131164684272457287846838673864986426510727914651664
Private key A: [192841940005690899700238199983245412671696871345556664582557215523418025737663870690464160
```

Приклад роботи:

```
p = 81741685423504868327290902154702275927002153810626010350078719949298451644207
q = 3568546114528506710685957160971290332247982100024005907263853669784624096423
Public key A: [1155361539488447669430872255120581813447368385475883847879317260433281236759787801920290507133179300314884033810186482327222828454230034267787259443143721, 290
Private key A: [100310836200813789893690112026922426029891467923666659283397171580747525866300528904169983138853404918492132978253250746435636842694740345208981858254657, 81
p = 77354199677901158278324497417034039934358297512303060894323206972242408998687
q = 54811686183625319349331689459012166815838075389437763480979343364528389972743
Public key B: [1011571079342716915705698675161266381274093656807229844905590667228571146894118161366965754192780241293367066790551046730017320079740264078915284417528785, 423
Private key B: [4036678060148842567256124495947948920508820200802658497978899721908246064215753888436286378734415442265807571093335732279561953077032210194749276948873825, 77
A is sending k to B ...
k = 769
(k1, s1): [938572739214393314806305943716902608237593067853707882924282676612522149581582263512044700693827132220157922635877483617215162185911784478221655245356344, 11356503
B receives k from A ...
k is verified
Received k = 769
```

Server:

Send key, Receive key потребують правильної роботи всіх інших написаних нами функцій, тому достатньо виконати тести лише цих високорівневих процедур:

Send key:

```
==== SendKey test (Server ReceiveKey) ====
k : 59, (0x3b)
Key : 0x9222c9a2945ad8c28374521d1fab5347815f452931f9ff91b0e654e751b2cf716c939ede70c25d542a008876d8940975416acc9f920879816fffd29394bd2cfa0
Signature : 0x2632214ee6d3d6ee24fd4853cec3d89a62916b1af85cd260a72bcd0684a9b2a1c79382dd3bfc0d3ef25101e7c1b533e834619be089d6eb57ccdb5b2af09a29d6
Modulus : 0x585b36926868ee8e52039b4e7d0070aebafca2ddfec5176622a4bd6a3e6e63a114cb08f208e8da5b60138b0159dd3840e62145fde16c3c435c6eabc9ad02db91
Public exponent: 0x177f7d52ed4e7fe71222d95cdf4eb214f6cc6500c00d87ecb54b70f52f67a4ff9d99784b3930a37c936f8d5155ab398f8cc66c1e6fbbccdec129999ad2db6a55
```

Receive key

✖ Clear

Key

9222c9a2945ad8c28374521d1fab5347815f452931f9ff91b0e654e751b2cf716c939ede70c25d542a008876d894097

Signature

2632214ee6d3d6ee24fd4853cec3d89a62916b1af85cd260a72bcd0684a9b2a1c79382dd3bfc0d3ef25101e7c1b533

Modulus

585b36926868ee8e52039b4e7d0070aebafca2ddfec5176622a4bd6a3e6e63a114cb08f208e8da5b60138b0159dd3

Public exponent

177f7d52ed4e7fe71222d95cdf4eb214f6cc6500c00d87ecb54b70f52f67a4ff9d99784b3930a37c936f8d5155ab398f8

Receive

Key

3B

Verification

true

✔

Receive key:

```
==== ReceiveKey test (Server SendKey) ====
Modulus : 0x585b36926868ee8e52039b4e7d0070aebafca2ddfec5176622a4bd6a3e6e63a114cb08f208e8da5b60138b0159dd3840e62145fde16c3c435c6eabc9ad02db91
Public exponent: 0x177f7d52ed4e7fe71222d95cdf4eb214f6cc6500c00d87ecb54b70f52f67a4ff9d99784b3930a37c936f8d5155ab398f8cc66c1e6fbbccdec129999ad2db6a55
k is verified
k : 18235923996296727097
```

Send key

✖ Clear

Modulus

585b36926868ee8e52039b4e7d0070aebafca2ddfec5176622a4bd6a3e6e63a114cb08f208e8da5b60138b0159dd3

Public exponent

177f7d52ed4e7fe71222d95cdf4eb214f6cc6500c00d87ecb54b70f52f67a4ff9d99784b3930a37c936f8d5155ab398f8

Send

Key

0E2DC499494214202C08E3D0DE38FE7D6309D158DE5D5802B3B7F58F38278A279DA8CBB7100D98F73C73C

Signature

399BF5B0E8A5F96555DC7BF688A2408FF8CBF2E42DD21257271B13713C1ABAB17B101873B1B7D8AF2E08C

**Висновки:**

В ході виконання лабораторної роботи ми вивчили й використали на практиці методи знаходження простих великих чисел, генерації ключів для асиметричної криптосистеми типу RSA та системи електронних підписів.

В результаті виконання лабораторної отримали імітацію обміну повідомлення з шифруванням, підписом та його підтвердженням.