

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ**  
**УНІВЕРСИТЕТ УКРАЇНИ**  
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**  
**імені Ігоря Сікорського»**  
**«ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ»**

**КРИПТОГРАФІЯ**

*Комп'ютерний практикум №4*

Виконали

студенти 3-го курсу групи ФБ-22

Лаптев Д. М. та Проскурня А. С.

Бригада №5

Перевірів/-ла: \_\_\_\_\_

Київ 2025

## 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.

## 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  $p \neq q$ ;  $p$  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  $(e, n)$ ,  $(p, q)$  і  $n$  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().

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

Згенеруємо 2 пари простих чисел довжиною 256 біт:

$p=111037008699443494319846787257631372789995452751970566491626186746427276878843$

$q=77213024704802583634157729708137941185979461185084197953657676385895076285181$

$p1=111224351447191284199010921433359767377223566654084985981081557289664235478837$

$q1=84080817030089233023024835269383448526219000866175631491814728840915823678059$

Згенеруємо публічні та приватні ключі абонентів А та В.

Відкритий ключ абонента А: (e, n) =

(2238525690251961136637344078371838105438432066378209107894448446397125586554647135874581643541963499364765832154106216127990103619928312731173141893554123,

8573503295857509923938416470442077851388989055726637479266345560549570425255737807589105165498414015801671773830005434289019936108143251035039524653325583)

Секретний ключ абонента А: (d, p, q) =

(2502632654446164099471923137609512597795768901033766674775644255218756295380474184452057702819798732651554522715183212475698547402555389608021596769551347,

111037008699443494319846787257631372789995452751970566491626186746427276878843,  
77213024704802583634157729708137941185979461185084197953657676385895076285181)

Відкритий ключ абонента В: (e, n) =

(27523579304756039532527119828668349646059390570892115279669929033569241565

499736288074646886462276998847907609444529923298457360398023235254324122092  
917,  
935183434332163095926137698188854816561683946217642637570096524843684386484  
656511114380094846472853212501335236920217137373049220499853696045678239573  
7383)  
Секретний ключ абонента В: (d, p, q) =  
(79965026771377443064661142912651031632246970037660212962171757310074779836  
149902525084007761090797419917470139719583527284153428802599721525970656427  
62981,  
111224351447191284199010921433359767377223566654084985981081557289664235478  
837,  
840808170300892330230248352693834485262190008661756314918147288409158236780  
59)

Кандидати, що не пройшли (у файлі invalid\_pairs.txt):

Пара (591685010865335511778190725118711868472951683028463697359815758820456541  
36621,  
72023644916261476594598242851188881171337901708443133544077836804689578510379  
) не підходить, оскільки  
11103700869944349431984678725763137278999545275197056649162618674642727687884  
3 \*  
77213024704802583634157729708137941185979461185084197953657676385895076285181  
>  
59168501086533551177819072511871186847295168302846369735981575882045654136621  
\*  
72023644916261476594598242851188881171337901708443133544077836804689578510379  
Пара (951384796108522985786603292681688698643499060930593336863000556459702222  
59803,  
75102122379898967141477216296670509351504733478227524774763898560059771283071  
) не підходить, оскільки  
11103700869944349431984678725763137278999545275197056649162618674642727687884  
3 \*  
77213024704802583634157729708137941185979461185084197953657676385895076285181  
>  
95138479610852298578660329268168869864349906093059333686300055645970222259803  
\*  
75102122379898967141477216296670509351504733478227524774763898560059771283071  
Пара (823827216081936189712461802297157848691051311975851075906676988885270939  
48381,  
96718819537327757134604773601145028776305996749605340266920742061345676004721  
) не підходить, оскільки  
11103700869944349431984678725763137278999545275197056649162618674642727687884  
3 \*  
77213024704802583634157729708137941185979461185084197953657676385895076285181  
>  
82382721608193618971246180229715784869105131197585107590667698888527093948381  
\*  
96718819537327757134604773601145028776305996749605340266920742061345676004721

Відкрите повідомлення M =

103417908096753293107748111040875722128999286030092950873465642585967

Зашифруємо повідомлення ключем абонента А, потім розшифруємо та звіримо із вхідним повідомленням, аналогічно зробимо і з абонентом В:

```
# # Шифрування та розшифрування повідомлення
cipher_A = encrypt(M, public_key_A)
print(f"Криптограма для абонента А: {cipher_A}")
l_A = decrypt(cipher_A, private_key_A)
print(f"Розшифроване повідомлення абонента А: {l_A}")
cipher_B = encrypt(M, public_key_B)
print(f"Криптограма для абонента В: {cipher_B}")
l_B = decrypt(cipher_B, private_key_B)
print(f"Розшифроване повідомлення абонента В: {l_B}")
if l_A == M and l_B == M:
    print("Повідомлення розшифровані вірно")
else:
    print("Повідомлення розшифровані невірно")
```

Криптограма для абонента А:

64320582272355091235366633840765033816813679228766001231327907873373708060252  
66401775763144202605838049484242508257040959482318599328439072640535961098732

Розшифроване повідомлення абонента А:

103417908096753293107748111040875722128999286030092950873465642585967

Криптограма для абонента В:

82244466265147021625087050664951066983080895313574764198264370619253180825173  
72421941870026701782883361982371093524476114133967797072608304336838839227161

Розшифроване повідомлення абонента В:

103417908096753293107748111040875722128999286030092950873465642585967

Тепер підпишемо і верифікуємо підписи кожного із абонентів А та В:

```
# # Підпис та перевірка повідомлення
signature_A = sign(M, private_key_A)
print(f"Підпис абонента А: {signature_A}")
verified_message_A = verify(M, signature_A, public_key_A)
print(f"Перевірене повідомлення абонента А: {verified_message_A}")
signature_B = sign(M, private_key_B)
print(f"Підпис абонента В: {signature_B}")
verified_message_B = verify(M, signature_B, public_key_B)
print(f"Перевірене повідомлення абонента В: {verified_message_B}")
```

Підпис абонента А:

36680258591357190988909194521402739141498881418299364219273367387604939347082  
13591691988132661796266158061360025133178776525141247238647417617970734528278

Перевірене повідомлення абонента А: True

Підпис абонента В:

36511600079024304846230411149304974252879310492867026271071067508913351188826  
97094713861688034228068639196626319317224799634384739418902247465213365404680

Перевірене повідомлення абонента В: True

Протокол конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA

Ми генеруємо випадковий ключ, потім слідують коментарям на скріншоті

```

# Вибір випадкового ключа k
k = random.randint(1, min(public_key_A[0], public_key_B[0]) - 1)
print(f"Випадковий ключ k: {k}")

# Відправник (A) відправляє ключ k отримувачу (B)
encrypted_key, signature = send_key(k, private_key_A, public_key_B)
print(f"Зашифрований ключ: {encrypted_key}")
print(f"Підпис: {signature}")

# Отримувач (B) отримує ключ k від відправника (A)
received_key = receive_key(encrypted_key, signature, public_key_A, private_key_B)
print(f"Отриманий ключ: {received_key}")

# Перевірка правильності отриманого ключа
assert k == received_key, "Отриманий ключ невірний!"
print("Отриманий ключ правильний.")

```

Результат наших дій (із терміналу):

Випадковий ключ k:

17169020391989422808950862693884704663127872828983557422235501488390534626203  
13267053206202211596949207786152763187264589650205147067546914786728177156196

Зашифрований ключ:

69916787318876707583031827069851826333858042052682415781708397545753777783893  
70937062028750137270328605159854442817106551613950997288350595672145888625335

Підпис:

57562701495758105821782309523057969488822379442340222850784254774932389728735  
54587813159436142572426920554509381637354374141612308792506781036098924542821

Отриманий ключ:

17169020391989422808950862693884704663127872828983557422235501488390534626203  
13267053206202211596949207786152763187264589650205147067546914786728177156196

Отриманий ключ правильний.

А тепер перевіримо шляхом взаємодії із тестовим середовищем Asym Crypto Lab Environment.

## RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

### Get server key

Clear

Key size256

Get key

Modulus84C8997CE309B5A64EFCE5A3C99D244DD2919F42FEBBDC21171117E13EBFCE7

Public exponent10001

Отримали Відкритий ключ сервера (e, n) (подані у хексі)

## Шифрування

Відкрите повідомлення M:

103417908096753293107748111040875722128999286030092950873465642585967

Криптограма для абонента A:

0x7acf3ee7097c790505691140423cfeca5f449a03413e28474701d927852467cb1b792627bb6  
461109fafbfd24d35d8964f9c3bd00f3360ff0d932ff66ebd59ec

*Примітка (около години чахли над кодом і середовищем, бо тупили і не могли зрозуміти, що усе у середовище і з нього виходить у хексі, потім пішло легше і швидше (у документації не знайшли вказівок щодо систем числення))*

Дані для вводу у середовище:

Відкрите повідомлення M:

0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Відкритий ключ абонента A: (e, n) =

(0x2abdaf86d0582c372438df9f0648b752de0866bbe55e45e62ee199ea500780631c7896afd87  
877f5febf28bef53d7300234d9c835df2bce0ce5fc1c368bflfcb,  
0xa3b269426bd82c38be2d560c240e29d192fe18f87855dff0048c3bd3a5d668c014f0c5bff21bf  
d0786615d827e80ca13aa0325af783dc049a534232c7a7b5d0f)

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Encryption

Clear

Modulus

A3B269426BD82C38BE2D560C240E29D192FE18F87855DFF0048C3BD3A5D668C014F0C5BFF21BFD078661f

Public exponent

2ABDAF86D0582C372438DF9F0648B752DE0866BBE55E45E62EE199EA500780631C7896AFD87877F5FEBF2

Message

3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Bytes

Encrypt

Ciphertext

7ACF3EE7097C790505691140423CFECA5F449A03413E28474701D927852467CB1B792627BB6461109FAFBF1

Шифротексти сходяться.

## Розшифрування

Відкрите повідомлення M:

0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Криптограма для серверу:

0xd1f2280381514dc1d32accb4828875df743cf4332b91df24a8138c2734b09fc

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Decryption

Clear

Ciphertext

d1f2280381514dc1d32accb4828875df743cf4332b91df24a8138c2734b09fc

Bytes

Decrypt

Message

03D603023584881441108275675FD4C8358051B6B81BF3D64C7EA88F6F

Усе правильно, ми отримали вихідне повідомлення.

## Підписання



# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Sign

Clear

Message

3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Bytes

Sign

Signature

0A1BF7D0A0C35103B5EA1E0478106252A3F942178DC571CD6C0142DDFB675AB2

```
# # Перевірка взаємодію із тестовим середовищем Asym Crypto Lab Environment
public_key_serv = (60059724760081170307872390905278238563065865687640799498300153216416031436007, 65537)

# public_key_A = (8573503295857509923938416470442077851388989055726637479266345560549570425255737807589105
# n = public_key_A[0]
# e = public_key_A[1]
# print(f"Відкритий ключ абонента A: (e, n) = ({hex(e)}, {hex(n)})") # Переводимо у гексадецимальну систему
# # Шифрування
# cipher_A = encrypt(M, public_key_A) #Шифруємо ключем абонента A
# print(f"Криптограма для абонента A: {hex(cipher_A)}")

# cipher_serv = encrypt(M, public_key_serv) #Шифруємо ключем серверу
# print(f"Криптограма для серверу: {hex(cipher_serv)}")
signature_serv = 0x0A1BF7D0A0C35103B5EA1E0478106252A3F942178DC571CD6C0142DDFB675AB2
verified_message_serv = verify(M, signature_serv, public_key_serv)
print(f"Перевірене повідомлення серверу: {verified_message_serv}")
```

Відкрите повідомлення M:  
0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f  
Перевірене повідомлення серверу: True

## Підтвердження

Підпис абонента A:  
0x4608efda717688f7f365e760cb3b780580d6b5ff252f8baabdc74ea4217528d38c7c18d5cc616b60343bdca890c7b442542b6e2d0d6ce2d795df037d52ef7716

# RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

## Verify

Clear

Message

3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Bytes

Signature

4608efda717688f7f365e760cb3b780580d6b5ff252f8baabdc74ea4217528d38c7c18d5cc616b60343bdca890c7b44

Modulus

a3b269426bd82c38be2d560c240e29d192fe18f87855dff0048c3bd3a5d668c014f0c5bff21bfd0786615d827e80ca13

Public exponent

2abdaf86d0582c372438df9f0648b752de0866bbe55e45e62ee199ea500780631c7896afd87877f5febf28bef53d730f

Verify

Verification

true

## Висновки

У ході виконання комп'ютерного практикуму ми ознайомилися з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомилися з системою захисту інформації на основі криптосхеми RSA, організацією з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.