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
	Лаптєв Д. М. та Проскурня А. С.
	Бригада №5
Перевірив/-ла:	

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 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 e1 e2 e3 e4 e4 e4 e5.
- 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 п. 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 Епстурт(), 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=1110370086994434943198467872576313727899954527519705664916261867464272768 78843

 $\underline{q} = 7721302470480258363415772970813794118597946118508419795365767638589507628\\5181$

<u>p1</u>=111224351447191284199010921433359767377223566654084985981081557289664235 478837

<u>q1</u>=840808170300892330230248352693834485262190008661756314918147288409158236 78059

Згенеруємо публічні та приватні ключі абонентів А та В.

<u>Відкритий ключ абонента А:</u> (e, n) =

(2238525690251961136637344078371838105438432066378209107894448446397125586554647135874581643541963499364765832154106216127990103619928312731173141893554123,

857350329585750992393841647044207785138898905572663747926634556054957042525 573780758910516549841401580167177383000543428901993610814325103503952465332 5583)

<u>Секретний ключ абонента А:</u> (d, p, q) =

(25026326544461640994719231376095125977957689010337666747756442552187562953 804741844520577028197987326515545227151832124756985474025553896080215967695 51347,

111037008699443494319846787257631372789995452751970566491626186746427276878 843,

772130247048025836341577297081379411859794611850841979536576763858950762851 81)

<u>Відкритий ключ абонента В:</u> (e, n) =

(27523579304756039532527119828668349646059390570892115279669929033569241565)

499736288074646886462276998847907609444529923298457360398023235254324122092

935183434332163095926137698188854816561683946217642637570096524843684386484 656511114380094846472853212501335236920217137373049220499853696045678239573 7383)

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

(7996502677137744306466114291265103163224697003766021296217175731007477983614990252508400776109079741991747013971958352728415342880259972152597065642762981,

111224351447191284199010921433359767377223566654084985981081557289664235478837.

840808170300892330230248352693834485262190008661756314918147288409158236780 59)

Кандидати, що не пройшли (у файлі invalid_pairs.txt):

Пара (591685010865335511778190725118711868472951683028463697359815758820456541 36621,

72023644916261476594598242851188881171337901708443133544077836804689578510379) не підходить, оскільки

11103700869944349431984678725763137278999545275197056649162618674642727687884

77213024704802583634157729708137941185979461185084197953657676385895076285181

59168501086533551177819072511871186847295168302846369735981575882045654136621

 $72023644916261476594598242851188881171337901708443133544077836804689578510379 \\ \Pi \text{apa} (95138479610852298578660329268168869864349906093059333686300055645970222259803,$

75102122379898967141477216296670509351504733478227524774763898560059771283071) не підходить, оскільки

11103700869944349431984678725763137278999545275197056649162618674642727687884

77213024704802583634157729708137941185979461185084197953657676385895076285181

95138479610852298578660329268168869864349906093059333686300055645970222259803

96718819537327757134604773601145028776305996749605340266920742061345676004721) не підходить, оскільки

11103700869944349431984678725763137278999545275197056649162618674642727687884 3 *

77213024704802583634157729708137941185979461185084197953657676385895076285181

82382721608193618971246180229715784869105131197585107590667698888527093948381

96718819537327757134604773601145028776305996749605340266920742061345676004721

Відкрите повідомлення М =

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 A: {cipher_A}")

1_A = decrypt(cipher_A, private_key_A)

print(f"Розшифроване повідомлення абонента A: {1_A}")

cipher_B = encrypt(M, public_key_B)

print(f"Криптограма для абонента A: {cipher_B}")

1_B = decrypt(cipher_B, private_key_B)

print(f"Розшифроване повідомлення абонента B: {1_B}")

if 1_A == M and 1_B == M:

   print("Повідомлення розшифровані вірно")

else:

   print("Повідомлення розшифровані невірно")
```

Криптограма для абонента А:

64320582272355091235366633840765033816813679228766001231327907873373708060252 66401775763144202605838049484242508257040959482318599328439072640535961098732 Розшифроване повідомлення абонента А:

Криптограма для абонента А:

82244466265147021625087050664951066983080895313574764198264370619253180825173 72421941870026701782883361982371093524476114133967797072608304336838839227161 Розшифроване повідомлення абонента В:

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 A: {signature_A}")
verified_message_A = verify(M, signature_A, public_key_A)
print(f"Перевірене повідомлення абонента A: {verified_message_A}")
signature_B = sign(M, private_key_B)
print(f"Підпис абонента B: {signature_B}")
verified_message_B = verify(M, signature_B, public_key_B)
print(f"Перевірене повідомлення абонента B: {verified_message_B}")
```

Підпис абонента А:

36680258591357190988909194521402739141498881418299364219273367387604939347082 13591691988132661796266158061360025133178776525141247238647417617970734528278 Перевірене повідомлення абонента A: 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 слідуємо 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 (А) відправляє ключ k отримувачу (В)
encrypted_key, signature = send_key(k, private_key_A, public_key_B)
print(f"Зашифрований ключ: {encrypted_key}")
print(f"Підпис: {signature}")

# Отримувач (В) отримує ключ k від відправника (А)
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 к:

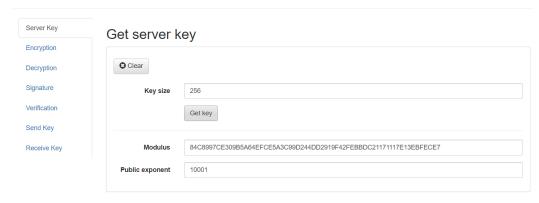
17169020391989422808950862693884704663127872828983557422235501488390534626203 13267053206202211596949207786152763187264589650205147067546914786728177156196 Зашифрований ключ:

69916787318876707583031827069851826333858042052682415781708397545753777783893 70937062028750137270328605159854442817106551613950997288350595672145888625335 Пілпис:

5756270149575810582178230952305796948882237944234022285078425477493238972873554587813159436142572426920554509381637354374141612308792506781036098924542821 Отриманий ключ:

1716902039198942280895086269388470466312787282898355742223550148839053462620313267053206202211596949207786152763187264589650205147067546914786728177156196 Отриманий ключ правильний.

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



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

Шифрування

Відкрите повідомлення М:

103417908096753293107748111040875722128999286030092950873465642585967 Криптограма для абонента А:

0x7acf3ee7097c790505691140423cfeca5f449a03413e28474701d927852467cb1b792627bb6461109fafbfd24d35d8964f9c3bd00f3360ff0d932ff66ebd59ec

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:

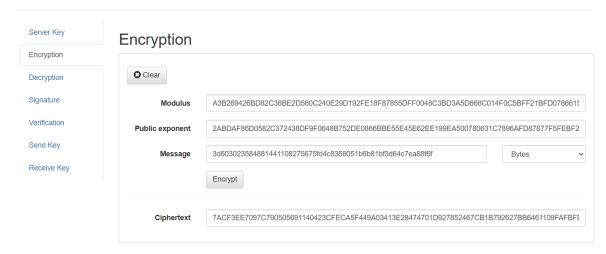
Відкрите повідомлення М:

0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

Відкритий ключ абонента А: (e, n) =

(0x2abdaf86d0582c372438df9f0648b752de0866bbe55e45e62ee199ea500780631c7896afd87877f5febf28bef53d7300234d9c835df2bce0ce5fc1c368bf1fcb,

0xa3b269426bd82c38be2d560c240e29d192fe18f87855dff0048c3bd3a5d668c014f0c5bff21bf d0786615d827e80ca13aa0325af783dc049a534232c7a7b5d0f)

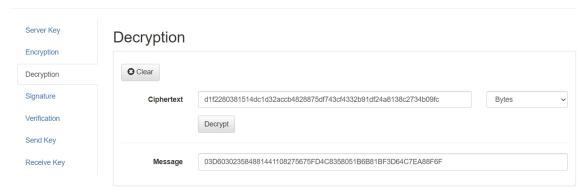


Шифротексти сходяться.

Розшифрування

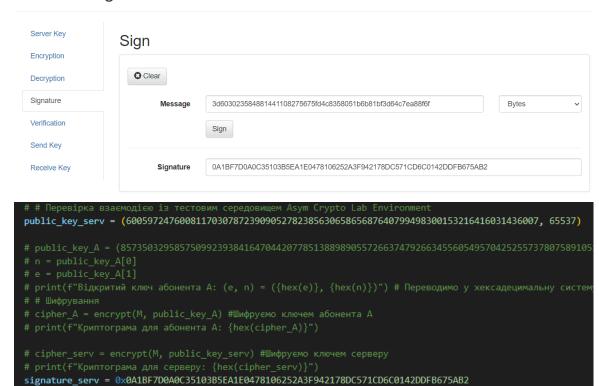
Відкрите повідомлення М: 0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f Криптограма для серверу: 0xd1f2280381514dc1d32accb4828875df743cf4332b91df24a8138c2734b09fc

RSA Testing Environment



Усе правильно, ми отримали вихідне повідомлення.

Підписання



Відкрите повідомлення М:

0x3d603023584881441108275675fd4c8358051b6b81bf3d64c7ea88f6f

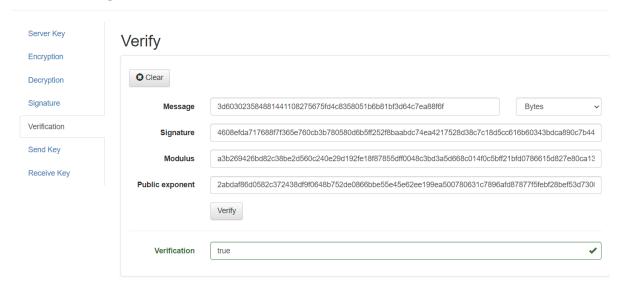
verified_message_serv = verify(M, signature_serv, public_key_serv) print(f"Перевірене повідомлення серверу: {verified_message_serv}")

Перевірене повідомлення серверу: True

Підтвердження

Підпис абонента А:

0x4608efda717688f7f365e760cb3b780580d6b5ff252f8baabdc74ea4217528d38c7c18d5cc616b60343bdca890c7b442542b6e2d0d6ce2d795df037d52ef7716



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.