

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”
Фізико-технічний інститут

Лабораторна робота № 4
з предмету «Криптографія»

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

Виконали:
Студенти 3 курсу,
ФТІ, групи ФБ-05
Супрун Максим, Алькова Аліна

Київ – 2022

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 і рекомендації щодо 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 і довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб p і q – прості числа для побудови ключів абонента A , і – абонента B . $q \nmid p-1$, $q \nmid p+1$, $q \nmid p-2$, $q \nmid p+2$, $q \nmid p-4$, $q \nmid p+4$

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ та відкритий ключ. За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі, та секретні d_A , d_B , (pk_A) , (pk_B) , (sk_A) , (sk_B) , (n) , (e) , (d) , (p) , (q)

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 n . $n \nmid 0$

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

За допомогою методичних вказівок реалізували усі потрібні для виконання функції. Серед них тест Міллера-Рабіна, генерація простих чисел, генерація ключових пар, функції шифрування/дешифрування, створення підпису, його верифікація. Окремо додали функції пошук нсд за розширеним алгоритмом Евкліда та пошук оберненого елемента з 3 лаби, функції конвертації тексту в десятичну систему та навпаки, а для проведення тестів - конвертації з хексу та навпаки. Особливо критичних проблем під час виконання не виникало. Усі функції реалізуються строго до, поданих у вказівках, формулах. Єдине іноді тест Міллера-Рабіна пропускав прості числа, але помилка була швидко виявлена та виправлена.

Перейдемо до виконання. Спочатку у нашій схемі з'являються юзери А та Б. (Подалі та у коді ми іменуємо їх по класиці для легшого сприйняття як Аліса(той хто надсилає) та Боб(отримувач)). Аліса та Боб по черзі генерують пари відкритих та закритих ключей.

---Alice:

p: 66940350573416764237533868426136458258565776292343662957536111610763892040179

q: 59989047430891953102761416942880075497059566499752767885049082593057268089363

Alice's public key: [4015687865589233620396443097936281946319589225464015050725107692124014561400627785408109841506920570459826653968766433560727510297543813946197655358515977, 1753944344984322061147632679802677961121946114671474579501661691711458669467694851284524825023479665890085779788455875769650742117461132531209023932618613]

Alice's private key: 2310417493218980967624041392268888328929314437947963603686121498571471626938640935075228896971761441901403791093674951632274764461924749640546919337234777

---Bob:

p: 110582499002401853067925970165563507376162886381782711375157884867316635547551

q: 67694477173663661453187260364098520323641033330507644862420814953188575111263

Bob's public key: [7485824454524776856664627389149684169284513374240097942512184826992995524330011686653258821772299183975909121357102365619550296930068781870733250752166913, 429990471718017111406765094168112264836554863970090386428523378482860266528539312557727987037282400792536838187509991033006825760479585394510387358379341]

Bob's private key: 3055490734156402247340182284179922670576554073178589265472605141386393163228725184352981811815605338316516492194781565123112257170123846734756207386694561

Відповідно масив відкритого ключа складається з [n,e] та масив закритого ключа з [d,p,q].

Вважається, що відкриті ключі так як вони самі по собі відкриті відомі обом юзерам.

Далі Аліса попередньо обирає повідомлення, за допомогою публічного ключа Боба та свого приватного шифрує повідомлення, підписує, формує повідомлення [k1,s1] та надсилає Бобу.

Боб отримує повідомлення, розшифровує за допомогою свого закритого ключа, за допомогою відкритого ключа e Аліси перевіряє її підпис.

--Ready to send message

--Sending this ->

[5383390308571295562903770047128169938687977299663276224210780671005328543879353375257039640902788244757827512469735192455467637235525083319446763475914084, 4729382801463043901252622430869616575452765948357518205031249697686706456742787855215343970390043955842175990609929081340937116426827664070681823363118836]

--Message was recieved

(!)verification succeeded

--Message from receiver ->

Catch me, if u can

Далі йдуть тести функцій за допомогою рекомендованого сайту.

Тест1 - Функція шифрування

```
#TEST1 - ENCRYPTION FUNCTION (ENCRYPT LOCAL, DECRYPT ON SERVER)
server_key_n = '8EFC6EED4D95A432ECB4EAECFEDCB5EB0B5BEBA4EA060B9C8264708D6AB73179'
server_key_n = from_hex(server_key_n)
#print(server_key_n)
server_key_e = '10001'
server_key_e = from_hex(server_key_e)

serv_key_pub = [server_key_n, server_key_e]

test_message = "Glory to Ukraine"
test_message = convert(test_message)

encrypted = to_hex(encrypt(test_message, serv_key_pub))
print('Text is encrypted!')
print("Here is your ciphertext --> ", encrypted)

Text is encrypted!
Here is your ciphertext --> 1853f208b0159888d65d13c9d3befd0c41d6c13280e7a7dd452e0a06af88020e
```

Get server key

Key size

256

Modulus

8EFC6EED4D95A432ECB4EAECFEDCB5EB0B5BEBA4EA060B9C8264708D6AB73179

Public exponent

10001

Decryption

Ciphertext

1853f208b0159888d65d13c9d3befd0c41d6c13280e7a7dd452e0a06af88020e

Text

Message

Glory to Ukraine

Розшифроване повідомлення збігається.

Тест2 – функція дешифрування.

```
#TEST2 - DECRYPTION FUNCTION (DECRYPT LOCAL, ENCRYPT ON SERVER+ENCRYPT LOCAL)
print("Alice's public key: ",a_public,"\n")
my_key_n = to_hex(a_public[0])
#print(my_key_n)
my_key_e = to_hex(a_public[1])
#print(my_key_e)

tmessage = "Catch me if you can"
server_ciphertext = '3A4CE3D550E6A9CEE818DB740A8D05734ADB95E9E08E2BC3C7A824A05A8574BAE1D746C53A48CDB31179950CB555773C007B782CC509'

encrypted = to_hex(encrypt(convert(tmessage), a_public))
print(encrypted,"\n")

print("Alice's private key: ",a_private,"\n")
nserver_ciphertext = from_hex(server_ciphertext)
convert_totext(decrypt(nserver_ciphertext, a_private))
```

Encryption

Modulus	4cac45fe98b41925b86a39320da496eca0c33bbc6802c8fd7bd0483c85f7110508bb8f3e512502d3c8adb6da4d2b39	
Public exponent	217d19f3ad7853a8a64ff6c935e8ce825afba23129f4f9c2c4fa73f51ce66c1c987f8bb949c837f74d7f006c031c40b0d5	
Message	Catch me if you can	Text ▾
	<input type="button" value="Encrypt"/>	
Ciphertext	3A4CE3D550E6A9CEE818DB740A8D05734ADB95E9E08E2BC3C7A824A05A8574BAE1D746C53A48CDB3117	

Alice's public key: [4015687865589233620396443097936281946319589225464015050725107692124014561400627785408109841506920570459826653968766433560727510297543813946197655358515977, 1753944344984322061147632679802677961121946114671474579501661691711458669467694851284524825023479665890085779788455875769650742117461132531209023932618613]

3a4ce3d550e6a9cee818db740a8d05734adb95e9e08e2bc3c7a824a05a8574bae1d746c53a48cdb31179950cb555773c007b782cc5090045281fc11d65ff7fe9

Alice's private key: [2310417493218980967624041392268888328929314437947963603686121498571471626938640935075228896971761441901403791093674951632274764461924749640546919337234777, 66940350573416764237533868426136458258565776292343662957536111610763892040179, 59989047430891953102761416942880075497059566499752767885049082593057268089363]

'Catch me if you can'

Бачимо, що збігається як шифртекст в обох випадках шифрування(це була додаткова перевірка), так і наша функція правильно розшифровує повідомлення.

Тест3 – функція верифікації підпису.

Sign

Message	Hi	Text ▾
	<input type="button" value="Sign"/>	
Signature	64D9B32067F8840071AEF3504F00F3868DABF3D9C2862126041C350025D8C17B	

```

: #TEST3 - VERIFY FUNCTION (SIGN ON SERVER, VERIFY LOCAL)
server_sign = '64D9B32067F8840071AEF3504F00F3868DABF3D9C2862126041C350025D8C17B'
server_sign = from_hex(server_sign)
server_key_n = '8EFC6EED4D95A432ECB4EAECFEDCB5E80B5BEBA4EA060B9C8264708D6AB73179'
server_key_n = from_hex(server_key_n)
#print(server_key_n)
server_key_e = '10001'
server_key_e = from_hex(server_key_e)
serv_key_pub = [server_key_n, server_key_e]
mess = 'Hi'
mess = convert(mess)
verify(mess,server_sign,serv_key_pub)

(!)verification succeeded

```

Верифікація проходить

Тест4 - підпис

```

: #TEST4 - SIGN FUNCTION (SIGN LOCAL, VERIFY ON SERVER)
print("Alice's private key: ",a_private,"\n")
mess = 'Hi'
mess = convert(mess)
print(to_hex(sign(mess,a_private)), "\n")
print("Alice's public key: ",a_public,"\n")
my_key_n = to_hex(a_public[0])
print(my_key_n)
my_key_e = to_hex(a_public[1])
print(my_key_e)

```

Alice's private key: [2310417493218980967624041392268888328929314437947963603686121498571471626938640935075228896971761441901403791093674951632274764461924749640546919337234777, 66940350573416764237533868426136458258565776292343662957536111610763892040179, 59989047430891953102761416942880075497059566499752767885049082593057268089363]

988e11976ba3f7596c388abb08c03bc1ba0083a82d3ddc2fef3350d18bdd0cd92f99db7ddbc1c390a57a043d5541148d005efe80d25d0a9b6d0ad75b5f6d451

Alice's public key: [4015687865589233620396443097936281946319589225464015050725107692124014561400627785408109841506920570459826653968766433560727510297543813946197655358515977, 1753944344984322061147632679802677961121946114671474579501661691711458669467694851284524825023479665890085779788455875769650742117461132531209023932618613]

4cac45fe98b41925b86a39320da496eca0c33bbc6802c8fd7bd0483c85f7110508bb8f3e512502d3c8adb6da4d2b390bb4da0b3a61ae8dbb546a8130a5e7ab09217d19f3ad7853a8a64ff6c935e8ce825afba23129f4f9c2c4fa73f51ce66c1c987f8bb949c837f74d7f006c031c40b0d5284344bba9ca02ae025de2afac5775

Verify

✕ Clear

Message

Hi

Text

Signature

988e11976ba3f7596c388abb08c03bc1ba0083a82d3ddc2fef3350d18bdd0cd92f99db7ddbc1c390a57a043d554114

Modulus

4cac45fe98b41925b86a39320da496eca0c33bbc6802c8fd7bd0483c85f7110508bb8f3e512502d3c8adb6da4d2b39

Public exponent

217d19f3ad7853a8a64ff6c935e8ce825afba23129f4f9c2c4fa73f51ce66c1c987f8bb949c837f74d7f006c031c40b0d5

Verify

Verification

true

✓

Тут також все добре.

Висновки

Під час виконання лабораторної роботи ми ознайомилися зі способами генерації простих чисел, а також відомими тестами, що допомагають визначати їх. Окрім того ознайомились з системою захисту інформації на основі криптосхеми RSA та спробували відтворити у коді процедури шифрування, дешифрування генерацію електронного підпису та його верифікацію. Вивчили протокол розсилання ключів. Набуті навички знадобляться у професійній діяльності та подальшому вивченні курсу.