Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №4

з дисципліни «Криптографія»

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

Виконали:
студенти 3 курсу ФТІ
групи ФБ-81
Близнюк Микола та Мишкін Артем
Перевірили:
Чорний О.

Київ – 2020

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 пошукувипадкового простого числа з заданого інтервалуабо заданої довжини, використовуючи датчик випадкових чисел татестиперевірки на простоту.В якості датчика випадкових чисел використовуйтевбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна ізпопередніми пробними діленнями. Тестинеобхідно реалізовувати власноруч, використання готових реалізацій тестівне дозволяється.

2. За допомогою цієї функціїзгенерувати дві пари простих чисел q,p і q1, p1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб qp≤q1p1; p і q–прості числа для побудови ключівабонента A, p1 і q1 –абонента B.

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ q,p,d та відкритий ключ e,n. За допомогою цієї функції побудувати схеми RSA для абонентів Ai B–тобто, створити та зберегти для подальшого використання відкриті ключі),(n1,e1)петасекретні d і d1.

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів Ai 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 Mi знайти криптограму для абонентів Аи B, перевірити правильність розшифрування. Скласти для Ai Bповідомлення 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 nk<<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.

Перевірка числа на простоту за допомогою теста Мили Рабіна (256):

Функція generate_key_pair(key_length)
приймає 1 параметр – довжину ключа

Потім за допомогою функції random_prime(key_length)
Обирається число на проміжку $(2^{key\_length} - 1\ ,\ 2^{key\_length+1} - 1)$
Якщо воно не парне - перевіряємо на простоту за Мілі Рабіном
Якщо воно парне – додаємо одиницю і перевіряємо на простоту за Мілі Рабіном

```
.....
Rabin says 1274509149553101377296535687161411036121021323907747553987018621171891362842 11 is not prime number
We found a prime number p = 22066440721201928969757560772028692497337902707132000029527597203122813078 1643
Rabin says 2299981326512823600289141451383835330082212900591587666252317074847788099539043 is not prime number
.....
We found a prime number q = 12137636583858583740998516941848801106759124018619656140302667219910692023828 7
```

Якщо два простих числа виявилися однаковими – генеруємо просте число допоки вони не будуть різними.

Генерування public та private ключів:

Після знаходження двох простих чисел. Обчислюємо fi(q*p).
Знаходимо n = p*q
Та знаходимо public exponent на проміжку (2, fi(q*p)) таку що gcd(e, fi(q*p)) = 1
І знаходимо d як обернений $e^{-1} mod\ fi(q * p)$.
Отримуємо пари (e, n) – public key
(d, n) – private key

```
Public key:
    Public exponent: 23057184056333291781466356685839842368106678278882237728702910768114026146332577307026164950934723983919813445462579661143259526970190261533137085963205743
    Modulus: 26783443817320732401842843464391237786684424141173699324661897434120525376697421414655828206276440527736649932576088311566240402933739466791387869125365541
Public key(hex):
    Public exponent: 0x1b83d2b03793ffa1748c84d8cd28fb6846d9cefc1d80f4f4f667c3c909a7181713f31fce31914d73c8c88a7737018d2e3f7fc6084d04aa3614b5ccb4d3f207c6f
    Modulus: 0x1ff62bf0f9750488b7db5474f2c012703a6cd250b50fa27d0dc55c0b00125dfd40a573431236a5e1cfc9cf8f4b44759a372b46bca7dfc7acd52fdd22363ad4f25
Public key(site-hex):
    Public exponent: 01b83d2b03793ffa1748c84d8cd28fb6846d9cefc1d80f4f4f667c3c909a7181713f31fce31914d73c8c88a7737018d2e3f7fc6084d04aa3614b5ccb4d3f207c6f
    Modulus: 01ff62bf0f9750488b7db5474f2c012703a6cd250b50fa27d0dc55c0b00125dfd40a573431236a5e1cfc9cf8f4b44759a372b46bca7dfc7acd52fdd22363ad4f25

Private key:
    Secret: 42417333830853249579552231698783743816567113299000705080325903182185977925771555200946324625310874883269472911849635616728396308738956826653777008770342 07
    Modulus: 26783443817320732401842843464391237786684424141173699324661897434120525376697421414655828206276440527736649932576088311566240402933739466791387869125365541
Private key(hex):
    Secret: 0x81950f98d6245253825a107ef52a62617953a3dcdaf313eb7a87453f80d013aa5d91dcff11b931641c8af902152d3de07cdc850ea15696a04dafda45905e2df
    Modulus: 0x1ff62bf0f9750488b7db5474f2c012703a6cd250b50fa27d0dc55c0b00125dfd40a573431236a5e1cfc9cf8f4b44759a372b46bca7dfc7acd52fdd22363ad4f25
Private key(site- hex):
    Secret: 081950f98d6245253825a107ef52a62617953a3dcdaf313eb7a87453f80d013aa5d91dcff11b931641c8af902152d3de07cdc850ea15696a04dafda45905e2df
    Modulus: 01ff62bf0f9750488b7db5474f2c012703a6cd250b50fa27d0dc55c0b00125dfd40a573431236a5e1cfc9cf8f4b44759a372b46bca7dfc7acd52fdd22363ad4f25
```

Кодування відкритого повідомлення

Функція encrypt(message, public_key) приймає два параметри: повідомлення та public key

Кодування повідомлення відбувається за принципом швидкого піднесення до степеня за модулем (схема зліва на право)

```
Open message: 100
    hex: 0x64
    site-hex: 064

Encrypted message: 24761435222675989460804207801580486193836612291701358786456991913048664982346804543784430593845001337244365093896034328707077593049850448608797387993160258
    hex: 0x1d8c75fcb7c8f3479b6775396ffa8b58d0b4b930a2dd58ce8c5666ac03a366e7e427178f8af68e2b26f85ea311e9a02b1cde14a031e735c89b8a643864ac49a42
    site-hex: 01d8c75fcb7c8f3479b6775396ffa8b58d0b4b930a2dd58ce8c5666ac03a366e7e427178f8af68e2b26f85ea311e9a02b1cde14a031e735c89b8a643864ac49a42
```

Декодування

Функція decrypt(enc_message, private_key) приймає два параметри: зашифроване повідомлення та private key

Декодування повідомлення відбувається за принципом швидкого піднесення до степеня за модулем (схема зліва на право)

Decrypted message: 100
hex: 0x64
site-hex: 064

Отимання підпису

Функція sign(original_message, private_key) приймає два параметри: повідомлення та private key
Підпис повідомлення відбувається за принципом швидкого піднесення до степеня за модулем (схема зліва на право)

Message signature: 21794419905264450175002831929858673023079907478564213898455545410896875503477865194654436570113104307787680212154023149970677450262762178359148610372646050
hex: 0x1a020e7c9dd4472fb56188742b5f092e06fd42df145ba3c06d3903b572094d95d8a96eb794672fb2c0b3d982a00bf378049945008a0ee50961e0b8417009e34a2
site-hex: 01a020e7c9dd4472fb56188742b5f092e06fd42df145ba3c06d3903b572094d95d8a96eb794672fb2c0b3d982a00bf378049945008a0ee50961e0b8417009e34a2

Перевірка підпису реалізовано локально:

Функція verify(message, signature, public_key) приймає три параметри: повідомлення, signature повідомлення та public key
Перевірка повідомлення відбувається за принципом швидкого піднесення до степеня за модулем (схема зліва на право)

Local signature verification: True

Такий самий функціонал реалізован також за допомогою API

API signature verification: {'verified': True}

Протокол конфіденційного розсилання ключів з підтвердженням справжності

Два користувачі А і Б генерують public та private ключі. Користувач А знає свої ключі((e, n), (d, n)) та відкритий ключ (e1, n1) користувача Б.

A Public key:
Public exponent: 9347823124663215555692283613712037611544367840527665785246958520031122687700850035650773880779485474251447918893923121407095680974791870409539223857745711
Modulus: 27368826171478153995687829808056855014376092905082684163788721813404063904745573121969011505229817109607617477752266110104903294326367767281030396435679481
A Public key(site-hex):
Public exponent: 0b27b3528be75b2d7441dd2b3ceac1a07c05416a23f28253a4af57d1a47f787deacea4d5681f4218d4f8e25c5d86c6cec659159d5c1f31b8b98c2497563d94f2f
Modulus: 020a90092211410741e086b80e7edaaa1475fcb9c38877b48e2685ec335d9c34bf628e29a7558bdde8bdc3cfbf53aebc705070979e26080c1c59737659f89ec8f9

A Private key:
Secret: 34090001988222051766916805372004063288967752712833114018326066359729590972831901216700064478088082154573727398189007496131323271557762111238181781965006 51
Modulus: 27368826171478153995687829808056855014376092905082684163788721813404063904745573121969011505229817109607617477752266110104903294326367767281030396435679481
A Private key(site- hex):
Secret: 04116d88b204455e4622f83f8376d0a2f85592261fbd0b9897d61ae9858d315dff027f7178fc9bb34bea53f1c07534e16b7bd24c8613e392e1c327ab0fbf8f70ab
Modulus: 020a90092211410741e086b80e7edaaa1475fcb9c38877b48e2685ec335d9c34bf628e29a7558bdde8bdc3cfbf53aebc705070979e26080c1c59737659f89ec8f9

B Public key:
Public exponent: 4194653826742297272752348590098869234502444564023063939734793907769338337964506844868221468027816205819025680817502579431225562212155366164342315412794531 3
Modulus: 1209521927083089396563118465880851356967261956349971322819610447288449826078311079819898966107298501749608382216033597801146998864951894911236554837466010 01
B Public key(site-hex):
Public exponent: 0320e6687aaa55cbfb52eb8c90f6e37c54ed12619162435c2f367032f78af442fdc034bab91cc5fa05b9f44fc40d44714ec73252e1a15512c044a10b32c3e2be61
Modulus: 09056200b776217b43c8bf5c20c166adfca207a488b4289fc946eabd9e95c9f1c7f1465e1edd4a40c2e0683cbe12823de1a1786c53d9cad001de605f3f472ac829

B Private key:
Secret: 2267258638120988792776883707651759663114951923488190218731917795806905408140848861257293143881987525500341745339119992696458062842435901664301316665258822 1
Modulus: 1209521927083089396563118465880851356967261956349971322819610447288449826078311079819898966107298501749608382216033597801146998864951894911236554837466010 01
B Private key(site- hex):
Secret: 01b0e54b32dd23b6dbbad0f0e0bc3571ae3a017c9f4d5811f380ef300b5b6b13651d962b1fe3d469fa6bb31c74238100a2effec471aa27a0bdc7875983e8da00bd
Modulus: 09056200b776217b43c8bf5c20c166adfca207a488b4289fc946eabd9e95c9f1c7f1465e1edd4a40c2e0683cbe12823de1a1786c53d9cad001de605f3f472ac829

Обирається випадкове число k і користувач А використовує функцію send_key(k, a_private, b_public), де формується повідомлення (k1, s1)

k is 5328395870512060119874141620818118381363496497051889280707056532536133212457 1

User A put

k: 532839587051206011987414162081811838136349649705188928070705653253613321245 71
s: 170885645121439505449802168195399852462116226454389093414106912203342332896665822890560197153558084975973649804993674833079520832966068107317784739153458 75

User A forms

k1: 755563347656280279837143388176409504824128732458713482251417935413668125014116523260318974931572341277701904558247989431639534675255909731532799866083085 53
s1: 920601322062691934845723091734760291952370146939690669222482995255612669102563146796185882132975398732264675154646498337094647037266909403826405409716597 99

Користувач Б використовує функцію receive_key(povidomlennia, public_key_a, public_key_b, private_key_b). Де він знаходить конфіденційність (k, s) і за допомогою відкритого ключа Користувача А перевіряє підпис

User B finds

k: 532839587051206011987414162081811838136349649705188928070705653253613321245 71
s: 170885645121439505449802168195399852462116226454389093414106912203342332896665822890560197153558084975973649804993674833079520832966068107317784739153458 75

User B check if A signature is valid – True

Такий самий функціонал реалізован також за допомогою API

API public key: {'modulus': 'ACAF2B410D07CA3F164EEBB1A41D02171F08E159D2DCB9626D59A43C7C62F1A0ACA971B61973A88FED0090D00851176BDD21861C251790B439C01BFB6E589147', 'publicExponent': '10001'}
Answer: {'key': '9E89F56EE1A6A716ECC08D297F9C6C1E6E830E15D851D955995E86FDFA4BF4', 'verified': True}

Висновки

Під час виконання 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, організували використання цієї системи для засекреченого зв'язку й електронного підпису, вивчели протокол розсилання ключів.