

Міністерство освіти і науки 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 групи ФБ-93

Флекевчук Д.І

Перевірила:

Селюх П.В.

Київ – 2021

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.

Завдання

- 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.
- За допомогою цієї функції згенерувати дві пари простих чисел p_0, q_0 і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $p_0q_0p_1q_1$, де p_0, q_0 - прості числа для побудови ключів абонента А, p_1, q_1 - абонента В.
- Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (e, n) . За допомогою цієї функції побудувати схеми RSA для абонентів А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e_0, n_0) , (e_1, n_1) та секретні d_0 і d_1 .
- 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 М і знайти криптограму для абонентів А і В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.
- 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 $0 < k < n$.

Хід роботи:

Частина 1:

Для генерації чисел використовую бібліотеку `random` мови `python` завдяки ній отримаю випадкові числа заданого розміру. На сильну простоту перевіряю тестом Міллера-Рабіна, для пришвидшення роботи алгоритму, було використано постулат Бертрана.

Частина 2:

Створив клас User, який на даному етапі розробки зберігає закритий ключ та своє ім'я.

За допомогою методу для генерації чисел тимчасово надаю користувачам ключі.

Створюю клас RSA, клас RSA дозволяє зберігати відкриті ключі за іменами користувачів, а також породжувати нові об'єкти класу User, та надавати цим екземплярам класу їх закриті ключі та імена. Також користувач може змінити ключ на ключ меншого розміру за потреби(для протоколу обміну ключами).

Частина 3:

Створюю клас RSA, клас RSA дозволяє зберігати відкриті ключі за іменами користувачів, а також породжувати нові об'єкти класу User, та надавати цим екземплярам класу їх закриті ключі та імена. Також користувач може змінити ключ на ключ меншого розміру за потреби(для протоколу обміну ключами).

Частина 4:

Реалізую функції шифрування розшифрування та для верифікації за цифровим підписом.

Демонстрація роботи

Шифрування та розшифрування.

```
184
185     print(toString(
186         Alice.decrypt(
187             Bob.encrypt(
188                 toNumber("test message"), Alice.name
189             )
190         )
191     )
192 )
193
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4> python main.py
test message
PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4>

Верифікація за підписом

```
181 RSA_SERVER = RSAServer()
182 Alice = RSA_SERVER.addUser("Alice")
183 Bob = RSA_SERVER.addUser("Bob")
184
185 print(Alice.verify(
186     Bob.sign(1111), Bob.name
187 ))
188 )
189
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4> python main.py
True
PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4>

Протокол обміну ключами.


```
RSA_SERVER = RSAServer()
Alice = RSA_SERVER.addUser("Alice")
Bob = RSA_SERVER.addUser("Bob")

Alice.receiveKey(
    Bob.sendKey(Alice.name)
)
```

```
PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4> python main.py
changed
key received 64458420785742623010480950151687941742504142976293236535240471715460742323633759801453050427558289736248075793167332206443038847383775929882727455817688
PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4> python main.py
key received 636249793921140628232914535523138944924565711262903923228748133000081810400145115482733064675434918088775203984512173851523198224049433674824709609181189
PS C:\Users\Данил\Desktop\crypto-FB-9\cp4\flekevchuk_fb-93_cp4> python main.py
```

Тести на сайті

Генерую ключі для одного з абонентів на сайті



Key size

256

Get key


Modulus

C51B6DE4CCC1023B5D9D5021D21D3AB393BA806E04AB610C1989A7F4FA873575

Public exponent

10001

І) Я зашифрував повідомлення(test message) для абоненту з сайту та віддав на сайт ШІТ



Ciphertext

7FAF4372D7B34814EFF162EDABD87D7D29BCE41E0BDB86DDBDC6662932E6I

Text

▼

Decrypt

Message

test message

II)Зашифрував повідомлення на сайті та розшифрува у себе у себе

Clear

Modulus

9C1298617B3A52482C9268646298B18C013D7C1CBE1F6A790E75148CC464A206A78D2FC09790E4E022DDE

Public exponent

295DDDC67B9EB880FBD0AD8F98FB99E1B997014C8A6B6D2007AA8954F571530A411B57677AD1787F7D07A

Message

test message

Text

Encrypt

Ciphertext

7F6CA3D04CB4FAE0D456230234D8A8180F94742D9F256732EF5BCDC7695CE6D75EC0DA823830E56F1C9D

III)Згенерував підпис та надав його разом з відкритими ключами абоненту на сайті

Clear

Message

test message

Text

Signature

6B444EEC2F4EB349A5E5EFEA799BB738F2003BA6655F5709B69C89AE9405D0286344F24EE0422501C87D5I

Modulus

83B3C72E99E4BF49234BDBD9918AB74F9566B1E841EB2762C09860BE70EB2625F93D50544393ABBF3CAB&

Public exponent

2C7FD19F557BE4598F6BFBB2AE94094AC87698CE8035759CFA47490042783F5DBD731D364ADDA789469B8

Verify

Verification

true

IV)Згенерував підпис абонента на сайті, перевінив підпис у себе.

Clear

Message

test message

Text

Sign

Signature

3B763D4A8336C1613E39297A964D98205B266B11DCC2F5248FDF855474668862

Висновки

В ході виконання цієї лаю. роботи я навчився перевіряти числа на сильну простоту, з методами розподілення та генерації ключів в протоколі RSA. Після отримання

теоретичних відомостей я розробив на основі RSA систему, організував засекречений зв'язок, та електронний підпис та протокол передачі ключів .