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»

Варіант 6

Виконали:

Студенти 3 курсу,

ФТІ, групи ФБ-92

Друзь Данило

Чикрій Кирило

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.

Постановка задачі:

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 p1 , q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq <= p1q1; p і q – прості числа для побудови ключів абонента A, p1 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) , (e1 ,n1) та секретні d і d1

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 □ n.

Хід роботи:

В ході виконання роботи, були створені функції знаходження простого числа в діапазоні, шифрування, дешифрування, надсилання, та отримання повідомлення, а також був описаний процес автентифікації співбесідника. Основні для цього функції це функція знаходження простих чисел (включаючи і тест Міллера-Рабіна) та функції зведення числа в степінь за модулем (реалізована за схемою Горнера).

Єдиною проблемою в даній лабораторній роботі була функція, що тестувала кандидатів на просте число. Вона зайняла найбільший час написання, все інше було набагато легше.

Результати

Data:

Sender Public Key: e =
81392535834141116297852232238634724587383319788511081374589664935585181510033753780349945308877062542482841664342607226211289633607805630386030842275523279
n =
10240602304677163484526559680401311268278782060542938141061474841333768953693455489699523233522361179787248488441772242783241385836068695575864687683372079

Sender Secret Key: d =
80304202595318060738750365235248162271704068433885103367354634406743068849946330618982079049192111097320616755964824347452716758978278543792731414573540313
p =
10131802291443625226199667148974734989677505953526759960717174567287767920117
3 q =
10107384658823653316756959211453967730384634878970367309519158545741029498872
3

Data:

Receiver Public Key: e =
95804082077566947784419625767153434737091153397676382980764391453603061393902299867907655399200916189436836954321813948319470485061866766310931130106546133
n =
11702773790462097053440912658412589984891087495292025391479303432611383876433545124018266564094794538185516704260293391861693740570438942616630635758089113

Receiver Secret Key: d =
11332089722267423046353171476732100302254630708948775530033795987392539822703602703903964206102527578047428104063665545466324887037714041198237539026565857 p =
10532299374388797139118769576680332841098978641038372348659801815998933197627
1 q =
11111318976480597854350107006191931072006564131918342611909911202615649204930
3

Generated message:
47715263362300779509979828465353559915847896835295987107314004628996813789287361446719966414478059563550233773009511623734574415688547527137631756368773
1

Authentification...

Authentification success

Encrypted:
89083712365042729156979954286018978915139363678937331110192156862289951604661306936824905810524513609201919291022819444778712408894039822028521061917690
8

Signed:
29444243344335920954402858047143587314604805629896193466678683920022916800230102810842803934380914286316920863052271730776969877727068059229136100768203412

Checking...

Look`s like original

Got message:
477152633623007795099798284653535599158478968352959871073140046289968137892873614467199664144780595635502337730095116237345744156885475271376317563687731

Перевірка

Для перевірки правильності виконання використаємо сайт https://asym-crypt-study.herokuapp.com/, який згенерував нам відкритий ключ за яким ми зможемо створити підпис повідомлення та зашифрувати це повідомлення. Далі пройдемо верифікацію повідомлення за отриманими параметрами та зашифруємо повідомлення в нашому коді.

Get server key

Clear

Key size 256

Get key

Modulus D35913665679F190F20040892336193A61BD6767FE1E97085EAE

Public exponent 10001

RSA Testing Environment

Server Key
Encryption
Decryption
Signature
Verification
Send Key
Receive Key

Sign

Clear

Message 35 Bytes

Sign

Signature 22BB6DD8F9E6C8EEEB73B19388F8B51811815109B12A2F86A0

Server Key
Encryption
Decryption
Signature
Verification
Send Key
Receive Key

Encryption

Clear

Modulus D35913665679F190F20040892336193A61BD6767FE1E97085EAE

Public exponent 10001

Message 35 Bytes

Encrypt

Ciphertext AFEB2FAF61A64F6EC8716187F7AFA332290F20320E07C40D96

```python
n = int("D35913665679F190F20040892336193A61BD6767FE1E97085EAB0DB2A52517C7",16)
e = int("10001",16)
print("Public keys: n=", n, "e =", e)
message = 53
print("Message in hex:", hex(message)[2:])
encrypted = encrypt((e, n), message)
# signature = sign()
print("Encrypted message:", hex(encrypted)[2:])
print("Signature:", int("22BB6DD8F9E6C8EEEB73B19388F8B51811815109B12A2F86A042E6722938D33B", 16))
print("Verifycation:", verify((e, n), message, int("22BB6DD8F9E6C8EEEB73B19388F8B51811815109B12A2F86A042E6722938D33B", 16)))
```

```
PS C:\Programming\III курс\Crypto\crypto-FB-9\cp4\druz_chikriy_fb-92_cp4> python lab4.py
Public keys: n= 955953943320366550004415793251293953604215202184526982860478489385417481317 83 e = 65537
Message in hex: 35
Encrypted message: afeb2faf61a64f6ec8716187f7afa332290f20320e07c40d96f5ea17ccae97ec
Signature: 157097953929625953751651622554007815370717088740685337193344001354276709875 79
Verifycation: True
PS C:\Programming\III курс\Crypto\crypto-FB-9\cp4\druz_chikriy_fb-92_cp4>
```

Висновки

Під час даного лабораторного 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, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.