**УКРАЇНИ МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ**

**КАФЕДРА КОМП’ЮТЕРНИХ НАУК**

**ПРАКТИЧНА РОБОТА № 8**

з дисципліни "Захист інформації"

Тема: **"Алгоритм RSA"**

Варіант № 14

**Перевірив Коробченко О.В.**

**Студентка групи Індн – 31с Пархомчук А. К.**

Суми – 2017

**Мета**

Виробити вміння та навички шифрування даних за допомогою шифру алгоритму RSA.

**Завдання**

1. Ознайомитися з алгоритмом шифрування даних RSA.

2. Виконати програмну реалізацію шифрування і дешифрування даних за допомогою алгоритму RSA, згідно свого варіанту.

3. Протестувати роботу програми.

4. Оформити звіт.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **A** | **B** | **Алфавіт** |
| 14 | 10000 | 15000 | Малі букви англійського алфавіту, спецсимвол (=) і цифри |

**Теоретична частина**

Алгоритм RSA запропонували в 1978 три автори: Р. Ривест (Rivest), А. Шамір (Shamir) і А. Адлеман (Adleman). Алгоритм отримав свою назву за першими літерами прізвищ його авторів. Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, який може працювати як в режимі шифрування даних, так і в режимі електронного цифрового підпису.

Надійність алгоритму ґрунтується на складності факторизації великих чисел і складності обчислення дискретних логарифмів.

У криптосистемі RSA відкритий ключ КА, секретний ключ КВ, повідомлення М і криптограма С належать множині цілих чисел

*,*  (1)

де ***N*** – модуль:

*.* (2)

***Р*** і ***Q*** - випадкові великі прості числа.

Для забезпечення максимальної безпеки вибирають ***Р*** і ***Q*** однакової довжини і зберігають у таємниці.

Множина ***ZN*** з операціями додавання і множення по модулю ***N*** утворює арифметику по модулю ***N***.

Відкритий ключ ***КА*** вибирають випадковим чином так, щоб виконувалися умови:

, (3)

, (4)

де  – функція Ейлера, показує кількість позитивних чисел в інтервалі від 1 до ***N***, які взаємно прості з ***N***:

.

Умова (4) означає, що відкритий ключ ***КА*** і функція Ейлера  повинні бути взаємно простими. Далі, використовуючи розширений алгоритм Евкліда, обчислити секретний ключ ***KB*** з порівняння:

**  (5)

або

.

Це можна здійснити, завдяки тому що одержувач ***В*** знає пару простих чисел (***P, Q***) і може легко знайти . Зауважимо, що ***KB*** і ***N*** повинні бути взаємно простими.

Відкритий ключ ***КА*** використовують для шифрування даних, а секретний ключ ***KB*** - для розшифрування. Перетворення шифрування визначає криптограму ***C*** через пару (відкритий ключ ***КА***, повідомлення ***М***) відповідно до такої формули:

**.** (6)

Звернення функції , тобто визначення значення ***М*** за відомим значенням ***С***, ***КA*** і ***N***, практично нездійсненне при ***N*** > 2512. Однак зворотну задачу, тобто задачу розшифрування криптограми ***C***, можна вирішити, використовуючи пари (секретний ключ ***KB***, криптограма ***C***) за формулою:

.  (7)

Процес розшифрування можна записати так:

. (8)

Підставляючи в (8) значення (6) і (7), одержуємо:

,

або

**.** (9)

Величина  відіграє важливу роль в теоремі Ейлера, яка стверджує: якщо , то

,

або в більш загальній формі

. 10)

Зіставляючи вирази (9) і (10), одержуємо



або, що те ж саме, .

Саме тому для обчислення секретного ключа ***KB*** використовують співвідношення (5). Таким чином, якщо криптограму



піднести до ступеня KB, то в результаті відновлюється вихідний відкритий текст М, так як



Таким чином, одержувач ***В***, що створює криптосистему, захищає два параметри: *секретний ключ* ***KB*** і *пару чисел* (***P, Q***), добуток яких дає значення модуля ***N***. З іншого боку, одержувач ***В*** відкриває значення модуля ***N*** і відкритий ключ ***КА***. Противнику відомі тільки значення ***КА*** і ***N***. Якби він зміг розкласти число ***N*** на множники ***Р*** і ***Q***, то він дізнався б "потайний хід" - трійку чисел {***Р, Q, КA***}, обчисливши значення функції Ейлера



і визначивши значення секретного ключа ***KB***. Однак, як було сказано вище, розкладання дуже великого ***N*** на множники обчислювальної неможливі (за умови, що довжини обраних ***Р*** і ***Q*** становлять не менше 100 десяткових знаків).

Розглянемо алгоритм шифрування і розшифрування в криптосистемі RSA. Припустимо, що користувач ***А*** хоче передати користувачеві ***В*** повідомлення в зашифрованому вигляді, використовуючи криптосистему RSA. У такому випадку користувач ***А*** виступає в ролі відправника, а користувач ***В*** - в ролі одержувача. Як зазначалося вище, криптосистему RSA повинен сформувати одержувач повідомлення, тобто користувач ***В***. Розглянемо послідовність дій користувача ***В*** і користувача ***А:***

**1.** Користувач ***В*** вибирає два довільних великих простих числа ***Р*** і ***Q***;

**2.** Користувач ***В*** обчислює значення модуля ***N***

;

**3.** Користувач ***В*** обчислює функцію Ейлера (8)

;

**4.** Користувач ***В*** вибирає випадковим чином значення відкритого ключа ***КА*** з урахуванням виконання умов:

, ;

**5.** Користувач ***В*** обчислює значення секретного ключа ***КB***, використовуючи розширений алгоритм Евкліда при вирішенні порівняння

;

**6.** Користувач ***В*** пересилає користувачеві ***А*** пару чисел (***N, КA***) по незахищеному каналу.

Якщо користувач ***А*** хоче передати користувачеві ***В*** повідомлення ***М***, він виконує наступні кроки:

* користувач ***А*** розбиває вихідний відкритий текст ***М*** на блоки, кожен з яких може бути представлений у вигляді числа *Мi=*0,1,2,...,*N*-1 ;
* користувач ***А*** шифрує текст, представлений у вигляді послідовності цифр ***М***, за формулою

;

* користувач ***А*** відправляє криптограму

*C*1*, С*2*, С*3,...,*Ci, ...*

користувачу ***В;***

* користувач ***В*** розшифровує прийняту криптограму

*C*1*, С*2*, С*3,...,*Ci, ...,*

використовуючи секретний ключ ***КB***, по формулі

.

У результаті буде отримана послідовність чисел ***Mi***, яка представляє собою вихідне повідомлення ***М***. Щоб алгоритм RSA мав практичну цінність, необхідно мати можливість без істотних витрат генерувати великі прості числа, вміти оперативно обчислювати значення ключів ***КA*** і ***КB***.

**Вихідні дані**

Алфавiт: abcdefghijklmnopqrstuvwxyz0123456789=

P: 13879

Q: 14639

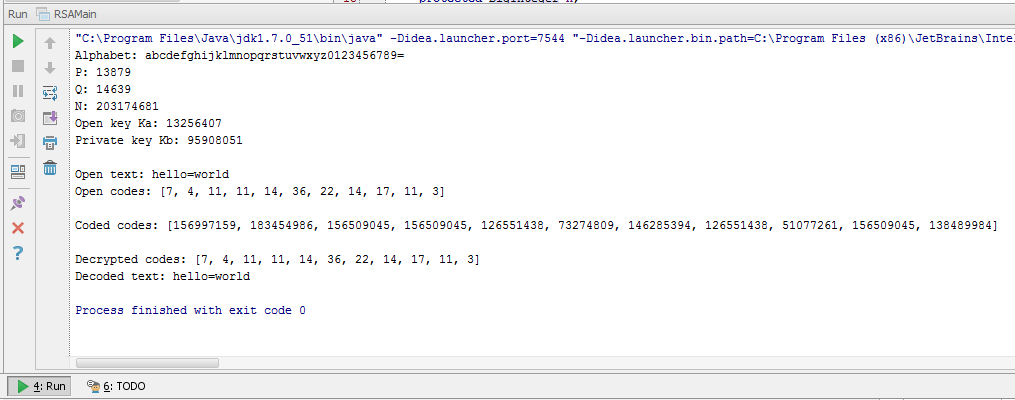
N: 203174681

Вiдкритий ключ *Ka*: 13256407

Закритий ключ *Kb*: 95908051

Текст для шифрування: hello=world

**Скриншоти роботи програми**



**Вихідний текст програми**

***RSAMain.java***

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.Arrays;

public class RSAMain {

public static final String FILE\_INPUT\_TXT = "res/input.txt";

private static String alphabet = "abcdefghijklmnopqrstuvwxyz0123456789=";

private static int[] codeDecode(int[] inputText, RSAUser user) {

int[] outputText = new int[inputText.length];

for (int i = 0; i < inputText.length; i++) {

BigInteger temp = new BigInteger("" + inputText[i]);

outputText[i] = temp.modPow(user.getKey(), user.getN()).intValue();

// System.out.printf("# %d - code: %d - pow(key: %d, N: %d)=%d\n", i, inputText[i], user.getKey(), user.getN(), outputText[i]);

}

return outputText;

}

private static int[] getOpenText(String fileName) {

BufferedReader file = null;

int[] array = null;

try {

ArrayList<Integer> positions = new ArrayList<Integer>();

file = new BufferedReader(new FileReader(fileName));

int c;

while((c = file.read())!=-1) {

positions.add(new Integer(c));

}

file.close();

array = new int[positions.size()];

int i = 0;

for(Integer element: positions) {

array[i] = alphabet.indexOf((char)element.intValue());

i++;

}

} catch(IOException e) {

e.printStackTrace();

} finally {

return array;

}

}

private static void intArray2String(int[] array, String message) {

System.out.print(message + ": ");

for(int i: array) {

System.out.print(alphabet.charAt(i));

}

System.out.println();

}

public static void main(String[] args) {

System.out.println("Alphabet: " + alphabet);

RSAGenerator obj = new RSAGenerator();

BigInteger openKey = obj.Ka;

BigInteger privateKey = obj.Kb;

BigInteger N = obj.n;

RSAUser userA = new RSAUser(openKey, N);

RSAUser userB = new RSAUser(privateKey, N);

// System.out.println("Openkey: " + openKey);

// System.out.println("PrivateKey: " + privateKey);

// System.out.println("N value: " + N);

int[] openText = getOpenText(FILE\_INPUT\_TXT);

intArray2String(openText, "Open text");

System.out.println("Open codes: " + Arrays.toString(openText));

System.out.println();

int[] closeText = codeDecode(openText, userA);

System.out.println("Coded codes: " + Arrays.toString(closeText));

System.out.println();

int[] decodedText = codeDecode(closeText, userB);

System.out.println("Decrypted codes: " + Arrays.toString(decodedText));

intArray2String(decodedText, "Decoded text");

}

}

***RSAGenerator.java***

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.Random;

public class RSAGenerator {

public static final String FILE\_PRIME\_TXT = "res/prime.txt";

public static final String FILE\_KEYS\_TXT = "res/keys.txt";

private BigInteger p;

private BigInteger q;

protected BigInteger n;

private BigInteger phi;

protected BigInteger Ka;

protected BigInteger Kb;

private static final int a = 10000; //10000

private static final int b = 15000; //15000

public RSAGenerator() {

ArrayList<BigInteger> mas = generateAB();

writeFile(mas, FILE\_PRIME\_TXT);

generatePQ(mas);

System.out.println("P: " + p);

System.out.println("Q: " + q);

generateN();

System.out.println("N: " + n);

generatePhi();

// System.out.println("PHI: " + phi);

getKa(generateKeys(FILE\_KEYS\_TXT));

System.out.println("Open key Ka: " + Ka);

getKb();

System.out.println("Private key Kb: " + Kb);

System.out.println();

}

private void writeFile(ArrayList<BigInteger> array, String fileName) {

BufferedWriter file = null;

try {

file = new BufferedWriter(new FileWriter(fileName));

for (BigInteger element : array) {

file.write(element + "\r\n");

}

file.close();

} catch (IOException e) {

System.out.print(e.toString());

}

}

private boolean isPrime(BigInteger value) {

if (value.equals(new BigInteger("2"))) {

return true;

} else if (value.mod(new BigInteger("2")).equals(BigInteger.ZERO)) {

return false;

}

BigInteger to = value.divide(new BigInteger("3"));

BigInteger i = new BigInteger("3");

for (; i.compareTo(to) != 1; i = i.add(new BigInteger("2"))) {

if (value.mod(i).equals(BigInteger.ZERO)) {

return false;

}

}

return true;

}

private ArrayList<BigInteger> generateAB() {

ArrayList<BigInteger> arrayList = new ArrayList<BigInteger>();

for (int i = 0; i < b - a; i++) {

BigInteger currentValue = new BigInteger(new Integer(a + i).toString());

if (isPrime(currentValue)) {

arrayList.add(currentValue);

}

}

return arrayList;

}

private void generatePQ(ArrayList<BigInteger> array) {

Random r = new Random();

p = array.get(r.nextInt(array.size()));

q = array.get(r.nextInt(array.size()));

}

private void generateN() {

n = p.multiply(q);

}

private void generatePhi() {

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

}

private ArrayList<BigInteger> generateKeys(String fileName) {

BufferedWriter file;

StringBuilder builder = new StringBuilder();

ArrayList<BigInteger> arrayKeys = new ArrayList<BigInteger>();

try {

file = new BufferedWriter(new FileWriter(fileName));

for (BigInteger i = new BigInteger("2"); i.compareTo(phi) != 1; i = i.add(BigInteger.ONE)) {

if (i.gcd(phi).equals(BigInteger.ONE)) {

// System.out.println("<generateKeys> i: " + i);

arrayKeys.add(i);

builder.append(i).append("\r\n");

}

}

file.write(builder.toString());

file.close();

} catch (IOException e) {

e.printStackTrace();

} finally {

return arrayKeys;

}

}

private void getKa(ArrayList<BigInteger> arrayKeys) {

Random r = new Random();

Ka = arrayKeys.get(r.nextInt(arrayKeys.size()));

}

private void getKb() {

Kb = Ka.modInverse(phi);

}

}

**Результати роботи програми**

Alphabet: abcdefghijklmnopqrstuvwxyz0123456789=

P: 13879

Q: 14639

N: 203174681

Open key Ka: 13256407

Private key Kb: 95908051

Open text: hello=world

Open codes: [7, 4, 11, 11, 14, 36, 22, 14, 17, 11, 3]

Coded codes: [156997159, 183454986, 156509045, 156509045, 126551438, 73274809, 146285394, 126551438, 51077261, 156509045, 138489984]

Decrypted codes: [7, 4, 11, 11, 14, 36, 22, 14, 17, 11, 3]

Decoded text: hello=world

**Висновки**

В ході даної лабораторної роботи була створена програмна реалізація шифрування шифром RSA. Результати тестування показують правильність роботи створеної програми. Використання файлових потоків введення-виведення дає можливість шифрування текстів будь-якого розміру досить швидко.