МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



ЗВІТ

про виконання лабораторної роботи № 3

«Захист даних з допомогою комбінованих алгоритмів шифрування та застосування електронного цифрового підпису»

з дисципліни

«Технології захисту інформації»

Виконала:

студентка групи ІТ-31

Щербак Л. В. Прийняв викладач:

Досин Д.Г.

Львів 2022

**Мета роботи:** Вивчити принцип роботи асиметричного алгоритму шифрування на прикладі алгоритму RSA. Освоїти методику створення комбінованих алгоритмів шифрування, які поєднують переваги методів симетричної та асиметричної криптографії та навчитись застосовувати електронний цифровий підпис.

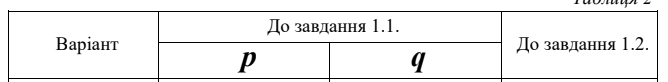
**Завдання роботи:**

1. Написати програму на мові java яка виконує операції:

1.1. Створення відкритого та закритого ключа при заданих для конкретного варіанту значеннях p та q (табл. 2). В якості числа e, яке входить в склад відкритого ключа, необхідно взяти найбільше просте число, менше p.

1.2. Використовуючи алфавіт (табл.1.), зашифруйте повідомлення з

допомогою створеного відкритого ключа. Повідомлення вибирається згідно з варіантом з табл..2.





1.3. Дешифрування зашифрованого тексту з допомогою створеного в 1.1.

закритого ключа. Шифрограма вибирається згідно з варіантом з табл. 3.



2. Оформити алгоритм функціонування програми та здійснити його опис

3. Результати подати у вигляді текстів програми та скрін шотів.

**Відповіді на контрольні питання:**

1. Поняття асиметричного шифрування даних.

У них для зашифрування даних використовується – «відкритий ключ», а для розшифрування – інший, «закритий, секретний ключ». При цьому розуміється, що ключ розшифрування не може бути визначений з ключа зашифрування. В асиметричних криптосистемах відритий ключ і криптограма можуть бути відправлені по незахищених каналах. Концепція таких систем заснована на застосуванні одно направлених функцій.

2. Особливості створення відкритого за закритого ключа.

Справа в тому, що замість відомого з давніх часів симетричного алгоритму шифрування, що передбачає використання однакових ключів як для шифрування, так і для розшифрування, при роботі з ЕЦП застосовується асиметричний крипто алгоритм, суть якого полягає у використанні двох різних ключів. Тоді як відкритий (публічний) ключ, що використовується для розшифрування, доступний всім, то закритий (таємний, особистий) – відомий лише відправнику.

3. Шифрування та дешифрування даних згідно з методом RSA.

На сьогоднішній день є найбільш використовуваним алгоритмом асиметричного шифрування. Його ефективність полягає в методі «первинної факторизації». По суті, обираються два різних випадкових простих числа заданого розміру (наприклад, 1024 біта кожне) і множаться, щоб створити ще одне гігантське число. Завдання полягає в тому, щоб визначити вихідні прості числа з помноженого гігантського. Виявляється, ця головоломка практично неможлива для сучасних суперкомп’ютерів, не кажучи вже про людей.

4. Поняття електронного цифрового підпису та способи його завдання?

Електронний цифровий підпис (або скорочено – ЕЦП) за правовим статусом прирівняний до власноручного підпису або печатки. ЕЦП – це дані в електронній формі, отримані за результатами криптографічного перетворення, які додаються до інших даних або документів і забезпечують їх цілісність та ідентифікацію автора.

5. Обчислення хеш образу повідомлення?

Хеш-функція використовується зокрема у структурах даних — хеш-таблицях, широко вживаних у програмному забезпеченні для швидкого пошуку даних. Хеш-функції використовуються для оптимізації таблиць та баз даних користуючись з того, що в однакових записів однакові значення хеш-функції. Такий підхід пошуку дублікатів ефективний у файлах великого розміру. Прикладом цього буде знаходження подібних ділянок у послідовностях ДНК. Криптографічна хеш-функція дозволяє легко перевірити, що деякі вхідні дані зіставляються із заданим значенням хешу, але, якщо вхідні дані невідомі, то навмисно важко відновити вхідне значення (або еквівалентну альтернативу), знаючи збережене значення хеш-функції. Це використовується для забезпечення цілісності переданих даних, і є будівельним блоком для HMACs, які забезпечують аутентифікацію повідомлень.

**Текст програми реалізації:**

// Java Program to Implement the RSA Algorithm

package lab3;

import java.math.\*;

import java.text.DecimalFormat;

class RSAEncryption {

// Буква А Б В Г Д Е Ї Ж З И Й К

// Код 01 02 03 04 05 06 07 08 09 10 11 12

// Буква Л М Н О П Р С Т У Ф Х Ц

// Код 13 14 15 16 17 18 19 20 21 22 23 24

// Буква Ч Ш Щ Є Ґ Ь І Ю Я Пробіл

// Код 25 26 27 28 29 30 31 32 33 34

// Цифра 0 1 2 3 4 5 6 7 8 9

// Код 35 36 37 38 39 40 41 42 43 44

static char[] symbol = new char[] {

'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ї', 'Ж', 'З', 'И', 'Й', 'К',

//01 02 03 04 05 06 07 08 09 10 11 12

'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц',

//13 14 15 16 17 18 19 20 21 22 23 24

'Ч', 'Ш', 'Щ', 'Є', 'Ґ', 'Ь', 'І', 'Ю', 'Я', ' ',

//25 26 27 28 29 30 31 32 33 34

'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'

//35 36 37 38 39 40 41 42 43 44

};

static int p;// = 13;// 1st prime number p

static int q;// = 17;// 2nd prime number q

static int n, e, d, z, j;

static String originalMsg;

static double[] encryptedMsg;

static char[] decryptedMsg;

// greatest common divisor (GCD) / Найбільший спільний дільник

static int gcd(int e, int z)

{

if (e == 0)

return z;

else

return gcd(z % e, e);

}

// get code by char

public static int getCodeByChar(final char c) {

final int length = symbol.length;

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

if (symbol[i] == c) {

return ++i;//gCode[i];

}

}

return 0;

}

// get char by code

public static char getCharByCode(final int c) {

final int length = symbol.length;

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

//if (gCode[i] == c) {

if ((i+1) == c) {

return symbol[i];

}

}

return ' ';

}

public static double[] encrypt(char[] msg, int e, int n) {

double[] result = new double[msg.length];

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

result[i] = ((Math.pow(getCodeByChar(msg[i]), e)) % n);

}

return result;

}

public static char[] decrypt(double[] msg, int d, int n) {

char[] result = new char[msg.length];

BigInteger N = BigInteger.valueOf(n);// converting int value of n to BigInteger

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

BigInteger C = BigDecimal.valueOf(msg[i]).toBigInteger();// converting double value of msg[i] to BigInteger

int code = (C.pow(d)).mod(N).intValue();

result[i] = getCharByCode(code);

}

return result;

}

public static void print(final String s, final char[] arr) {

System.out.print(s);

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

System.out.print(arr[i]);

}

System.out.println();

}

public static void print(final String s, final double[] arr) {

System.out.print(s);

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

System.out.print(new DecimalFormat("#0").format(arr[i]));

if (i < arr.length - 1) System.out.print(", ");

}

System.out.println();

}

public static void getKeysPair(int p, int q) {

System.out.println("Prime numbers (p, q) : ("+p+", "+q+")");

System.out.println();

n = p \* q;

z = (p - 1) \* (q - 1);//z - Euler function

System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

// e is for public key exponent

if (gcd(e, z) == 1) {

break;

}

}

System.out.println("- PublicKey (e, n) : ("+e+", "+n+")");

d = 0;

for (int i = 0; i <= 9; i++) {

int x = 1 + (i \* z);

// d is for private key exponent

if (x % e == 0) {

d = x / e;

break;

}

}

System.out.println("- PrivateKey (d, n) : ("+d+", "+n+")");

}

public static void main(String args[])

{

System.out.println("Examples from lection :");

getKeysPair(7, 13);

System.out.println();

System.out.println("PART 1 (encryption) :");

originalMsg = "ДВГУПС";

System.out.println("- Original message : "+originalMsg);

encryptedMsg = encrypt(originalMsg.toCharArray(), e, n);

print("- Encrypted message : ", encryptedMsg);

decryptedMsg = decrypt(encryptedMsg, d, n);

print("- Decrypted message : ", decryptedMsg);

System.out.println("PART 2 (decryption) :");

encryptedMsg = new double[] {31, 61, 23, 21, 75, 80};

print("- Original Encrypted message : ", encryptedMsg);

decryptedMsg = decrypt(encryptedMsg, d, n);

print("- Decrypted message : ", decryptedMsg);

encryptedMsg = encrypt(decryptedMsg, e, n);

print("- Encrypted message : ", encryptedMsg);

System.out.println();

System.out.println("Lab 3 :");

getKeysPair(13, 17);

System.out.println();

originalMsg = "БАГАЛ4";

System.out.println("- Original message : "+originalMsg);

encryptedMsg = encrypt(originalMsg.toCharArray(), e, n);

print("- Encrypted message : ", encryptedMsg);

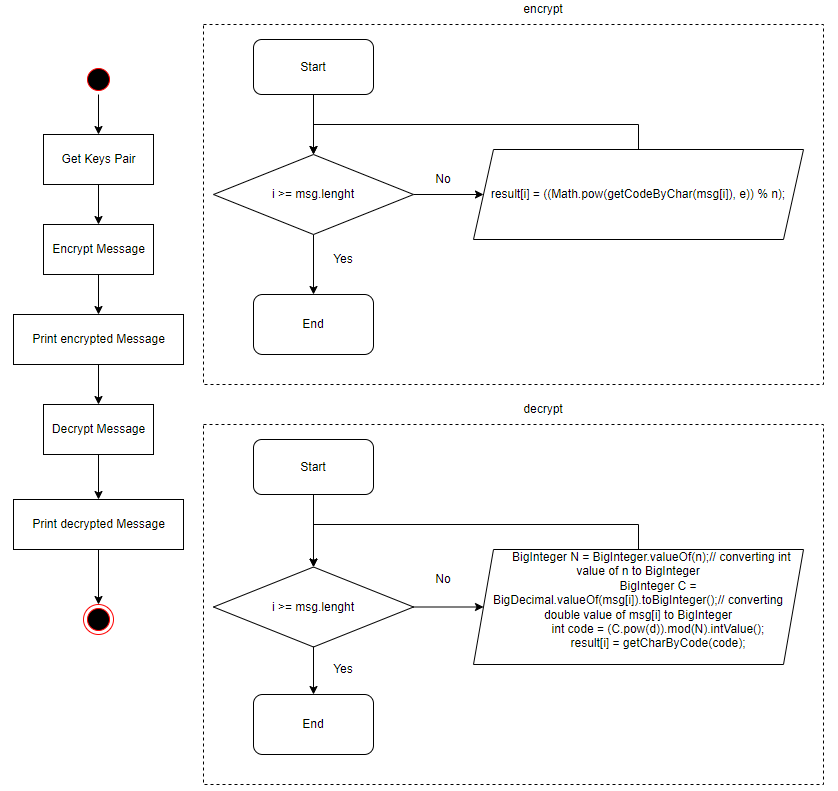
decryptedMsg = decrypt(encryptedMsg, d, n);

print("- Decrypted message : ", decryptedMsg);

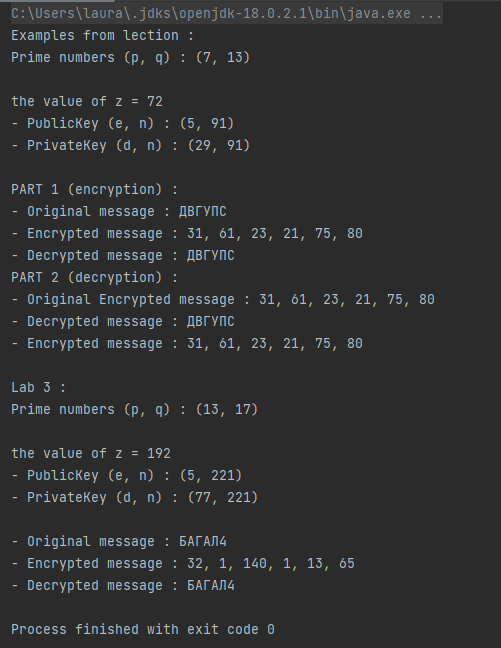
}

}

**Блок-схема:**



**Результати роботи програми:**



**Висновки:**

Розглянула принцип роботи асиметричного алгоритму шифрування на прикладі алгоритму RSA. Освоїла методику створення комбінованих алгоритмів шифрування, які поєднують переваги методів симетричної та асиметричної криптографії та навчилася застосовувати електронний цифровий підпис.