Міністерство освіти та науки

Прикарпатський національний університет

Фізико-технічний факультет

Кафедра комп’ютерної інженерії та електроніки

Лабораторна робота № 7

З курсу “Захист інформації у комп’ютерних мережах та системах”

Виконав студент групи КІ-41

Воробій Віталій

Івано-Франківськ 2020

5 варіант

Програмна реалізація мовою *Java:*

Реалізував клас, що містить ряд ефективних алгоритмів, що будуть корисні у подальшому.

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class Algorithms {  
  
 */\*\*  
 \* Дискретне піднесення до степеня по модулю  
 \*/* public static int discretePower(int x, int y, int p) {  
 int res = 1;  
 x = x % p;  
 if (x == 0) return 0;  
 while (y > 0)  
 {  
 if ( (y & 1) == 1)  
 res = (res \* x) % p;  
 y = y >> 1;  
 x = (x \* x) % p;  
 }  
 return res;  
 }  
  
 */\*\*  
 \* Знаходження випадкового числа між двома межами  
 \* @param minQuote Мінімальне співвідношення між згенерованим  
 \* випадковим числом і правою межою  
 \*/*public static int findRandomInRangeWithMinQuote (  
 int a,  
 int b,  
 float minQuote)  
{  
 final Random r = new Random();  
 int leftBound = (int)( a + ((b - a) \* minQuote));  
 return leftBound + r.nextInt( Math.*abs*(b - leftBound));  
}

*/\*\*  
 \* Рекурсивний алгоритм Евкліда  
 \*/* public static int euclidGCD(int a, int b) {  
 if (b == 0) {  
 return a;  
 }  
 return *euclidGCD*(b, a % b);  
 }

*/\*\*  
 \* Знаходження взаємнопростого випадкового числа  
 \* в межах від 1 до p - 1  
 \*/* public static int findRandomCoprimeNumberBetween(int p) {  
 final Random r = new Random();  
 int a = 2 + Math.*abs*(r.nextInt(p - 3));  
 while (*euclidGCD*(p - 1, a ) != 1) {  
 a = 2 + Math.*abs*(r.nextInt(p - 3));  
 }  
 return a;  
 }  
  
 */\*\*  
 \* Функція для генерування простих чисел  
 \*/* public static List<Integer> generatePrimeNumbers(int length) {  
 List<Integer> numbers = new ArrayList<>();  
 for (int n = 2; numbers.size() != length; n++) {  
 boolean isPrime = true;  
 for (Integer r: numbers) {  
 if (n % r == 0) {  
 isPrime = false;  
 break;  
 }  
 }  
 if (isPrime) {  
 numbers.add(n);  
 }  
 }  
 return numbers;  
 }  
  
  
}

Написав клас конфігурації Ель Гамаля.

package solution;  
  
*/\*\*  
 \* Клас, що описує конфігурацію Ель Гамаля  
 \*/*public class ElHamal {  
 private static final float *MIN\_QUOTE* = 0.6f;  
 private final int p;  
 private final int q;  
 private final int x;  
 private final int y;  
  
 public ElHamal(int p, int q) {  
 this.p = p;  
 this.q = q;  
 this.x = calcX();  
 this.y = calcY();  
 }  
  
 */\*\*  
 \* Обрахунок значення x  
 \*/* private int calcX() {  
 return Algorithms.*findRandomInRangeWithMinQuote*(p, q, *MIN\_QUOTE*);  
 }  
  
 */\*\*  
 \* Обрахунок значення y  
 \*/* private int calcY() {  
 return Algorithms.*discretePower*(q, x, p);  
 }  
  
 public int getP() {  
 return p;  
 }  
  
 public int getQ() {  
 return q;  
 }  
  
 public int getY() {  
 return y;  
 }  
  
 public int getX() {  
 return x;  
 }  
}

Написав клас, для шифрування:

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ElHamalCipher {  
 private final ElHamal elHamal;  
  
 public ElHamalCipher(ElHamal elHamal) {  
 this.elHamal = elHamal;  
 }  
  
 public List<Integer> cipher(List<Integer> inputList) {  
 List<Integer> cipheredList = new ArrayList<>();  
 for (Integer m : inputList) {  
 CryptoPair cryptoPair = cipher(m);  
 cipheredList.add(cryptoPair.a);  
 cipheredList.add(cryptoPair.b);  
 }  
 return cipheredList;  
 }  
  
 */\*\*  
 \* Розрахунок криптограми  
 \*/* private CryptoPair cipher(int m) {  
 final int p = elHamal.getP();  
 final int q = elHamal.getQ();  
 final int y = elHamal.getY();  
  
 final int k = Algorithms.*findRandomCoprimeNumberBetween*(p);  
  
 *// Розрахунок криптограми* final int a = Algorithms.*discretePower*(q, k, p);  
 final int b = Algorithms.*discretePower*(y, k, p) + m % p;  
  
 return new CryptoPair(a, b);  
 }  
  
 */\*\*  
 \* Контейнер криптограми  
 \*/* private static class CryptoPair {  
 final int a;  
 final int b;  
  
 public CryptoPair(int a, int b) {  
 this.a = a;  
 this.b = b;  
 }  
 }  
  
}

У цьому випадку можна застосувати дискретне піднесення до степеня.

Написав клас для дешифрування повідомлення:

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ElHamalDecipher {  
 private final ElHamal elHamal;  
  
 public ElHamalDecipher(ElHamal elHamal) {  
 this.elHamal = elHamal;  
 }  
  
 public List<Integer> decipher(List<Integer> cipheredList) {  
 final int N = cipheredList.size();  
 final int x = elHamal.getX();  
 final int p = elHamal.getP();  
  
 final List<Integer> decipheredList = new ArrayList<>();  
  
 for (int i = 0; i < N - 1; i += 2) {  
 final int a = cipheredList.get(i);  
 final int b = cipheredList.get(i + 1);  
  
 final int m = b - (Algorithms.*discretePower*(a, x, p));  
 decipheredList.add(m);  
 }  
 return decipheredList;  
 }  
  
  
}

Таким чином можна застосувати дискретне піднесення до степеня.

Клас для демонстрації роботи програми:

package solution;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
public class ElHamalDemo {  
 private static final int *NUMBER\_OF\_PRIMES* = 5000;  
  
 */\*\*  
 \* Функція для вибору двох випадкових чисел зі списку  
 \*/* private static List<Integer> pickRandomTwo(List<Integer> inputList) {  
 Random random = new Random();  
 final int N = inputList.size();  
 int a = Math.*abs*(random.nextInt()) % N;  
 int b = Math.*abs*(random.nextInt()) % N;  
 while (b == a) {  
 b = Math.*abs*(random.nextInt()) % N;  
 }  
 return List.*of*(inputList.get(a), inputList.get(b));  
 }  
  
 private static List<Integer> stringToIntList(String text) {  
 final int N = text.length();  
 List<Integer> integers = new ArrayList<>();  
 for (int i = 0; i < N; i++) {  
 integers.add((int)text.charAt(i));  
 }  
 return integers;  
 }  
  
 private static String intListToString(List<Integer> integers) {  
 final int N = integers.size();  
 StringBuilder builder = new StringBuilder();  
 for (int v : integers) {  
 builder.append((char) v);  
 }  
 return builder.toString();  
 }  
  
 public static void main(String[] args) {  
  
 List<Integer> primes = Algorithms.*generatePrimeNumbers*(*NUMBER\_OF\_PRIMES*);  
  
 *// Фільтруємо прості числа менші 10 000* primes = primes  
 .stream()  
 .filter(prime -> prime > 10\_000)  
 .collect(Collectors.*toList*());  
  
 *// Залишаємо два випадкові* primes = *pickRandomTwo*(primes);  
 int a = primes.get(0);  
 int b = primes.get(1);  
  
 *// Встановлюємо p > q* final int p = Math.*max*(a, b);  
 final int q = Math.*min*(a, b);

System.*out*.println("p = " + p);  
 System.*out*.println("q = " + q);  
  
 final ElHamal elHamal = new ElHamal(p, q);  
 System.*out*.println("x = " + elHamal.getX());  
 System.*out*.println("y = " + elHamal.getY());  
  
 final ElHamalCipher cipher = new ElHamalCipher(elHamal);  
 final ElHamalDecipher decipher = new ElHamalDecipher(elHamal);  
  
 final Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Введіть текст: ");  
 while (scanner.hasNext()) {  
 final String inputText = scanner.nextLine();  
 if (inputText.equals("stop")) {  
 break;  
 }  
 List<Integer> inputList = *stringToIntList*(inputText);  
 List<Integer> ciphered = cipher.cipher(inputList);  
 List<Integer> deciphered = decipher.decipher(ciphered);  
  
 System.*out*.println("Вхідний текст: " + inputText);  
 System.*out*.println("Зашифрований текст: " + *intListToString*(ciphered));  
 System.*out*.println("Розшифрований текст: " + *intListToString*(deciphered));  
 }  
 System.*out*.println("Програму завершено");  
  
 }  
  
}



Рисунок 1. Демонстрація роботи програми

Висновок: на цій лабораторній роботі я засвоїв алгоритм Ель Гамаля.