Міністерство освіти та науки

Прикарпатський національний університет

Фізико-технічний факультет

Кафедра комп’ютерної інженерії та електроніки

Лабораторна робота № 6

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;  
  
public class Algorithms {  
  
 */\*\*  
 \* 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;  
 }  
   
  
 */\*\*  
 \* Розширений алгоритм Евкліда  
 \*/* public static int[] extendedEuclidAlgorithm(int a, int b) {  
 int u\_p = 1, v\_p = 0;  
 int [] res = {0, 1};  
 while (a % b != 1) {  
 int q = (int) a / b;  
 int u\_n = u\_p - q \* res[0];  
 int v\_n = v\_p - q \* res[1];  
 u\_p = res[0];  
 v\_p = res[1];  
 res[0] = u\_n;  
 res[1] = v\_n;  
 int r\_n = a % b;  
 a = b;  
 b = r\_n;  
 }  
 int q = a / b;  
 res[0] = u\_p - q \* res[0];  
 res[1] = v\_p - q \* res[1];  
 return res;  
 }

*/\*\*  
 \* 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;  
 }  
   
}

Написав клас, що здійснює шифрування методом Рабіна

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class RabinCipher {  
 private final int N;  
  
 public RabinCipher(int n) {  
 N = n;  
 }  
  
 public List<Integer> cipher(List<Integer> inputList) {  
 final List<Integer> cipheredList = new ArrayList<>();  
 for (Integer m : inputList) {  
 final int c = Algorithms.*discretePower*(m, 2, N);  
 cipheredList.add(c);  
 }  
 return cipheredList;  
 }  
  
}

Написав клас, для формування додаткового ключа:

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class RabinSecret {  
 private final int p;  
 private final int q;  
 private final int n;  
  
 public RabinSecret(int p, int q) {  
 this.p = p;  
 this.q = q;  
 this.n = p \* q;  
 }  
  
 public List<Integer> formSecret(List<Integer> input, List<Integer> cipheredInput) {  
 List<Integer> result = new ArrayList<>();  
 final int p\_x = (p + 1) / 4;  
 final int q\_x = (q + 1) / 4;  
 for (int i = 0; i < input.size(); i++) {  
 int c = cipheredInput.get(i);  
 int m\_p = Algorithms.*discretePower*(c, p\_x, p);  
 int m\_q = Algorithms.*discretePower*(c, q\_x, q);  
 int[] euclidRes = Algorithms.*extendedEuclidAlgorithm*(p, q);  
 final int y\_p = euclidRes[0], y\_q = euclidRes[1];  
  
 int r1 = (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n;  
 if (r1 < 0) {  
 r1 = n - r1;  
 }  
 int r2 = n - r1;  
 int r3 = (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n;  
 if (r3 < 0) {  
 r3 = n - r3;  
 }  
 int r4 = n - r3;  
  
 result.add(r1 + r2 + r3 + r4 - input.get(i));  
 }  
 return result;  
 }  
  
}

Ключ формується наступним чином:

1. Сумуємо вектор коефіцієнтів, що отримає користувач при дешифруванні
2. Віднімаємо значення повідомлення

Написав клас, для дешифрування повідомлення:

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class RabinDecipher {  
 private final int p;  
 private final int q;  
 private final int n;  
  
 public RabinDecipher(int p, int q) {  
 this.p = p;  
 this.q = q;  
 this.n = p \* q;  
 }

List<Integer> decipher(List<Integer> inputList, List<Integer> secret) {  
 List<Integer> decipheredList = new ArrayList<>();  
 final int p\_x = (p + 1) / 4;  
 final int q\_x = (q + 1) / 4;  
 for (int i = 0; i < inputList.size(); i++) {  
 int c = inputList.get(i);  
 int m\_p = Algorithms.*discretePower*(c, p\_x, p);  
 int m\_q = Algorithms.*discretePower*(c, q\_x, q);  
 int[] euclidRes = Algorithms.*extendedEuclidAlgorithm*(p, q);  
 final int y\_p = euclidRes[0], y\_q = euclidRes[1];  
  
 *// k % n = n - k % n якщо k < 0* int r1 = (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n;  
 if (r1 < 0) {  
 r1 = n - r1;  
 }  
 int r2 = n - r1;  
 int r3 = (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n;  
 if (r3 < 0) {  
 r3 = n - r3;  
 }  
 int r4 = n - r3;  
  
 decipheredList.add(r1 + r2 + r3 + r4 - secret.get(i));  
 }  
 return decipheredList;  
 }  
  
}

Клас для демонстрації роботи програми:

package solution;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
  
public class RabinDemo {  
  
 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) {  
  
 *// Генеруємо 60 перших простих чисел, залишаємо тільки такі, що a mod 4 = 3 і обираємо два довільних випадково* List<Integer> primeNumbers = Algorithms  
 .*generatePrimeNumbers*(60)  
 .stream()  
 .filter(a -> a % 4 == 3)  
 .collect(Collectors.*toList*());  
 List<Integer> inputValues = *pickRandomTwo*(primeNumbers);  
  
 final int p = inputValues.get(0);  
 final int q = inputValues.get(1);  
  
 System.*out*.println("p = " + p);  
 System.*out*.println("q = " + q);  
  
 final int N = p \* q;  
  
 RabinCipher cipher = new RabinCipher(N);  
 RabinSecret secret = new RabinSecret(p, q);  
 RabinDecipher decipher = new RabinDecipher(p, q);  
  
 Scanner scanner = new Scanner(System.*in*);  
 while (scanner.hasNext()) {  
 String inputText = scanner.nextLine();  
 if ("stop".equals(inputText)) {  
 break;  
 }  
 final List<Integer> inputList = *stringToIntList*(inputText);  
 final List<Integer> cipheredList = cipher.cipher(inputList);  
 final List<Integer> secretKey = secret.formSecret(inputList, cipheredList);  
 final List<Integer> decipheredList = decipher.decipher(cipheredList, secretKey);  
 System.*out*.println("Вхідний текст: " + *intListToString*(inputList));  
 System.*out*.println("Зашифрований текст: " + *intListToString*(cipheredList));  
 System.*out*.println("Розшифрований текст: " + *intListToString*(decipheredList));  
 System.*out*.println("Співпадає " + inputList.equals(decipheredList));  
 }  
 System.*out*.println("Програму завершено");  
 }  
  
}



Рисунок 1. Демонстрація роботи програми

Висновок: на цій лабораторній роботі я засвоїв алгоритм Рабіна.