Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Лабораторная работа №2

“SHA 1 ”

Преподаватель: O.Cerbu

Выполнил: Маруневич Николай группа I1902

Кишинев 2022

Class sha1.java

import java.util.Scanner;  
  
public class sha1 {  
  
 public static int *messLength* = 0;  
  
 public static void main(String[] args) {  
  
 //Getting the word  
 System.*out*.println("Insert a word a phrase to be hashed");  
 Scanner sc = new Scanner(System.*in*);  
 String word = sc.nextLine();  
 System.*out*.println("Plain Text: " + word);  
  
 //Converting the word to binary  
 String binary = *convertToBinary*(word);  
 *messLength* = binary.length();  
 *calculateMod*(word, binary);  
  
 }  
  
 public static String convertToBinary(String word) {  
  
 byte[] bytes = word.getBytes();  
 StringBuilder binary = new StringBuilder();  
  
 for (byte b : bytes) {  
 int val = b;  
 for (int i = 0; i < 8; i++) {  
 binary.append((val & 128) == 0 ? 0 : 1);  
 val <<= 1;  
 }  
 binary.append(' ');  
 }  
  
 return binary.toString();  
  
 }  
  
 public static void calculateMod(String word, String binary) {  
  
 int binaryMessageLength = word.length() \* 8 - 8; //the -8 will be taken into account below.  
 String endBitLength = *calculateMessageLength*(binaryMessageLength+8); //add back 8 for accuracy  
 int subMod = endBitLength.length();  
 int temp = (binaryMessageLength) % 512;  
  
 if (432 - temp < 0) {  
 int x = 512 - temp;  
 temp = x + 440 + temp + 64;  
 } else {  
 temp = 432 - temp;  
 }  
  
 int binaryZeros = temp;  
 String onePadded = "10000000"; //add back the removed 8  
 binary = binary.replaceAll("\\s+", ""); //remove spaces  
 *createMessageLength*(binary, onePadded, binaryZeros, endBitLength); //creates the 512 bit message  
  
 }  
  
 public static String calculateMessageLength(int bitLength) {  
  
 String tempBitsLength = Integer.*toBinaryString*(bitLength);  
 StringBuilder sb = new StringBuilder(tempBitsLength);  
 int temp = 64 - tempBitsLength.length();  
  
 while (temp > 0) {  
 sb.insert(0, 0);  
 temp--;  
 }  
  
 return sb.toString();  
  
 }  
  
 //create complete message  
 public static String createMessageLength(String message, String paddedOne, int zeros, String endLength) {  
  
 StringBuilder messageBinary = new StringBuilder(message);  
 messageBinary.insert(messageBinary.toString().length(), paddedOne);  
  
 while (zeros > 0) {  
 messageBinary.insert(messageBinary.toString().length(), 0);  
 zeros--;  
 }  
  
 messageBinary.insert(messageBinary.toString().length(), endLength);  
 String m = *printMessage*(messageBinary.toString());  
 m = m.replaceAll("\\s+", "");  
 int[] mArray = new int[m.toString().length()/32];  
  
 for (int i = 0; i < m.toString().length(); i+=32) {  
 mArray[i/32] = Integer.*valueOf*(m.substring(i+1, i+32),2);  
 if(m.charAt(i) == '1'){  
 mArray[i/32] |= 0X80000000;  
 }  
 System.*out*.printf("Decimal(iterator), String(Binary), Hex values of input: %d %s %x\n", i, m.substring(i, i+32),mArray[i/32]);  
 }  
  
 *hash*(mArray);  
 return messageBinary.toString();  
  
 }  
  
 public static String printMessage(String message) {  
  
 StringBuilder sb = new StringBuilder(message);  
 int num = message.length();  
  
 while (num > 0) {  
 if (num % 32 == 0) {  
 sb.insert(num, " ");  
 }  
 num--;  
 }  
  
 return sb.toString();  
  
 }  
  
 private static int leftrotate(int x, int shift) { //leftrotate function  
  
 return ((x << shift) | (x >>> (32 - shift))); //>>> is an UNSIGNED shift compared >> which is not  
  
 }  
  
 //instance variables  
 private static int *h1* = 0x67452301;  
 private static int *h2* = 0xEFCDAB89;  
 private static int *h3* = 0x98BADCFE;  
 private static int *h4* = 0x10325476;  
 private static int *h5* = 0xC3D2E1F0;  
 private static int *k1* = 0x5A827999;  
 private static int *k2* = 0x6ED9EBA1;  
 private static int *k3* = 0x8F1BBCDC;  
 private static int *k4* = 0xCA62C1D6;  
  
  
 private static String hash(int[] z) {  
  
 //Extend the sixteen 32-bit words into eighty 32-bit words  
 int integer\_count = z.length;  
 int[] intArray = new int[80];  
 int j = 0;  
  
 for(int i = 0; i < integer\_count; i += 16) {  
 for(j = 0; j <= 15; j++)  
 intArray[j] = z[j+i];  
 for ( j = 16; j <= 79; j++ ) {  
 //w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1  
 intArray[j] = *leftrotate*(intArray[j - 3] ^ intArray[j - 8] ^ intArray[j - 14] ^ intArray[j - 16], 1);  
 //System.out.printf("J: %d ARRAY: %x\n", j, intArray[j]);  
 }  
  
 // calculate A,B,C,D,E:  
 int A = *h1*;  
 int B = *h2*;  
 int C = *h3*;  
 int D = *h4*;  
 int E = *h5*;  
 int t = 0; //temp  
  
 for ( int x = 0; x <= 19; x++ ) {  
 //temp = leftrotate(a leftrotate 5) + f(t) + e + w[i] + k  
 t = *leftrotate*(A,5)+((B&C)|((~B)&D))+E+intArray[x]+*k1*;  
 E=D; D=C; C=*leftrotate*(B,30); B=A; A=t;  
 }  
 for ( int b = 20; b <= 39; b++ ) {  
 t = *leftrotate*(A,5)+(B^C^D)+E+intArray[b]+*k2*;  
 E=D; D=C; C=*leftrotate*(B,30); B=A; A=t;  
 }  
 for (int c = 40; c <= 59; c++ ) {  
 t = *leftrotate*(A,5)+((B&C)|(B&D)|(C&D))+E+intArray[c]+*k3*;  
 E=D; D=C; C=*leftrotate*(B,30); B=A; A=t;  
 }  
 for ( int d = 60; d <= 79; d++ ) {  
 t = *leftrotate*(A,5)+(B^C^D)+E+intArray[d]+*k4*;  
 E=D; D=C; C=*leftrotate*(B,30); B=A; A=t;  
 }  
  
 *h1*+=A; *h2*+=B; *h3*+=C; *h4*+=D; *h5*+=E;  
  
 }  
  
 String h1Length = Integer.*toHexString*(*h1*);  
 String h2Length = Integer.*toHexString*(*h2*);  
 String h3Length = Integer.*toHexString*(*h3*);  
 String h4Length = Integer.*toHexString*(*h4*);  
 String h5Length = Integer.*toHexString*(*h5*);  
 //System.out.println(h1Length.length());  
  
 //Integer.toHexString does not include extra leading 0's  
 if(h1Length.length() < 8) {  
 StringBuilder h1L = new StringBuilder(h1Length);  
 h1L.insert(0,0);  
 h1Length = h1L.toString();  
 } else if(h2Length.length() < 8) {  
 StringBuilder h2L = new StringBuilder(h2Length);  
 h2L.insert(0,0);  
 h2Length = h2L.toString();  
 } else if(h3Length.length() < 8) {  
 StringBuilder h3L = new StringBuilder(h3Length);  
 h3L.insert(0,0);  
 h3Length = h3L.toString();  
 } else if(h4Length.length() < 8) {  
 StringBuilder h4L = new StringBuilder(h4Length);  
 h4L.insert(0,0);  
 h4Length = h4L.toString();  
 } else if(h5Length.length() < 8) {  
 StringBuilder h5L = new StringBuilder(h5Length);  
 h5L.insert(0,0);  
 h5Length = h5L.toString();  
 }  
  
 //result  
 String hh = h1Length + h2Length + h3Length + h4Length + h5Length;  
 System.*out*.println("Result: " + hh);  
  
 return null;  
 }  
  
}

Результат:

