CRYPTOGRAPHY LABORATORY FILE
CS-511

Submitted to:
Dr. Samayveer Singh
Assistant Professor
Department of Computer Science

Submitted by:
Shashi Shekhar Azad
Roll No.: 23203029
M. Tech. CSE 1st Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DR. B. R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR

Objective 12

Write a program to implement the SHA-1 Hash function to validate data integrity.

Procedure:

1. **Initialization**
   1.1. Initialize five 32-bit variables (A, B, C, D, E) with specific constant values defined by the SHA-1 algorithm.
   1.2. Prepare the input data by padding it with zeros to ensure its length is congruent to 448 modulo 512 bits.
2. **Message Processing**
   2.1. Break the padded message into 512-bit blocks.
   2.2. Process each block in a loop.
3. **Block Processing**
   3.1. Break the 512-bit block into 16 words of 32 bits each.
   3.2. Extend the 16 words into 80 words using a specific bitwise operation and constants.
4. **Main Loop**
   4.1. Iterate through the 80 words in the block.
   4.2. Perform logical and arithmetic operations on the variables (A, B, C, D, E) using the words and constants.
5. **Update Variables**
   5.1. Update the variables (A, B, C, D, E) based on the results of the main loop.
   5.2. Rotate and shift operations are applied to these variables.
6. **Hash Value Calculation**
   6.1. After processing all blocks, concatenate the final values of (A, B, C, D, E) to obtain the hash value.
   6.2. The hash value is typically represented as a hexadecimal number.
7. **Message Digest**
   7.1. The resulting hash value is the SHA-1 hash of the input data.

Code:

```
import java.util.Scanner;
public class SHAHashing {
   public static int messLength = 0;
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a message: ");
      String word = sc.nextLine();
      String binary = strToBinary(word);
      messLength = binary.length();
      calculateMod(word, binary);
      sc.close();
   }
```

```
public static String strToBinary(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 String hexToBinary(String hexString) {
    if (!hexString.matches("^[0-9A-Fa-f]+$")) {
        throw new IllegalArgumentException("Invalid hexadecimal string: " + hexString);
    }
    long decimalValue = new java.math.BigInteger(hexString, 16).longValue();
    String binaryString = Long.toBinaryString(decimalValue);
    int padding = hexString.length() * 4 - binaryString.length();
    if (padding > 0) {
        binaryString = String.format("%" + padding + "s", "").replace(' ', '0') + binaryString;
    }
    return binaryString;
}
public static void calculateMod(String word, String binary) {
    int binaryMessageLength = word.length() * 8 - 8;
    String endBitLength = calculateMessageLength(binaryMessageLength + 8);
    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";
        binary = binary.replaceAll("\\s+", "");
        createMessageLength(binary, onePadded, binaryZeros, endBitLength);
    }
    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();
    }
    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;
            }
        }
        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) {
    return ((x << shift) | (x >>> (32 - shift)));
}
private static int hash1 = 0x67452301;
private static int hash2 = 0xEFCDAB89;
private static int hash3 = 0x98BADCFE;
private static int hash4 = 0x10325476;
private static int hash5 = 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) {
    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++) {
            intArray[j] = leftrotate(intArray[j - 3] ^ intArray[j - 8] ^ intArray[j - 14] ^ intArray[j - 16], 1);
        }
        int A = hash1;
```

```
int B = hash2;
int C = hash3;
int D = hash4;
int E = hash5;
int t = 0;
for (int x = 0; x <= 19; x++) {
    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;
    }
    hash1 += A;
    hash2 += B;
    hash3 += C;
    hash4 += D;
    hash5 += E;
}
String hash1Length = Integer.toHexString(hash1);
String hash2Length = Integer.toHexString(hash2);
String hash3Length = Integer.toHexString(hash3);
String hash4Length = Integer.toHexString(hash4);
String hash5Length = Integer.toHexString(hash5);
if (hash1Length.length() < 8) {
    StringBuilder hash1L = new StringBuilder(hash1Length);
    hash1L.insert(0, 0);
    hash1Length = hash1L.toString();
} else if (hash2Length.length() < 8) {
    StringBuilder hash2L = new StringBuilder(hash2Length);
    hash2L.insert(0, 0);
    hash2Length = hash2L.toString();
} else if (hash3Length.length() < 8) {
    StringBuilder hash3L = new StringBuilder(hash3Length);
    hash3L.insert(0, 0);
    hash3Length = hash3L.toString();
} else if (hash4Length.length() < 8) {
    StringBuilder hash4L = new StringBuilder(hash4Length);
    hash4L.insert(0, 0);
    hash4Length = hash4L.toString();
} else if (hash5Length.length() < 8) {
    StringBuilder hash5L = new StringBuilder(hash5Length);
    hash5L.insert(0, 0);
    hash5Length = hash5L.toString();
}
String hh = hash1Length + hash2Length + hash3Length + hash4Length + hash5Length;
```