CRYPTOGRAPHY AND NETWORK SECURITY

ASSESSMENT – 03

23MID0019

JEEVANTH S

SHA-512 ALGORITHM

Qn.No	Questions	Marks
1	For the given plaintext "SATURDAY" determine the following using SHA-512 algorithm. (a) Find the length of padding bits (b) Determine the first twenty 64-bit words in hexadecimal format (W0 to W19) (c) Apply the majority function of round operation on buffers A, B and C (d) Apply the condition function of round operation on buffers E, F and G Consider the following values for the buffer. A 00111100 00111100 B 10001000 00010001 C 11001100 00110011 D 10011001 01100110 E 11100111 00011000 F 11110000 00001111 Theoretically verify the simulated results.	10

Code

```
private static final String[] BUFFER_VALUES = {
    "00111100 00111100", // A
    "10001000 00010001", // B
    "11001111 00110011", // C
    "10011001 01100110", // D
    "11100111 00011000", // E
```

```
        "11110000 00001111"  // F
    };
    public static void main(String[] args) {
        String input = "SATURDAY"; // Example input
        System.out.println("SHA-512 Implementation");
        System.out.println("=====================");
        System.out.println("Input: " + input);
        String binaryInput = stringToBinary(input);
        System.out.println("Input in binary: " + binaryInput);
        System.out.println("Input length in bits: " + binaryInput.length());
        int originalLength = binaryInput.length();
        int paddingLength = calculatePaddingLength(originalLength);
        System.out.println("\nPadding Information:");
        System.out.println("Original message length: " + originalLength + " bits");
        System.out.println("Padding bits to be added: " + paddingLength + " bits (1 followed by
" + (paddingLength - 1) + " zeros)");
        String paddedMessage = applyPadding(binaryInput, originalLength);
        System.out.println("Total length after padding: " + paddedMessage.length() + " bits");
        System.out.println("Number of 1024-bit blocks: " + (paddedMessage.length() / 1024));
        long[] words = calculateFirst20Words(paddedMessage);
        System.out.println("\nFirst 20 words (W0 to W19):");
        for (int i = 0; i < 20; i++) {
            System.out.printf("W%02d: %016x\n", i, words[i]);
        }

        calculateFunctions();
    }
    public static String stringToBinary(String input) {
        StringBuilder binary = new StringBuilder();
        for (char c : input.toCharArray()) {
```

```
        String binaryChar = Integer.toBinaryString(c);
        while (binaryChar.length() < 8) {
            binaryChar = "0" + binaryChar;
        }
        binary.append(binaryChar);
    }
    return binary.toString();
}
public static int calculatePaddingLength(int messageLength) {
    int k = 0;
    while ((messageLength + 1 + k + 128) % 1024 != 0) {
        k++;
    }
    return 1 + k; // 1 for the mandatory '1' bit plus k zeros
}
public static String applyPadding(String message, int originalLength) {
    StringBuilder padded = new StringBuilder(message);
    padded.append("1");
    while ((padded.length() + 128) % 1024 != 0) {
        padded.append("0");
    }
    String lengthBinary = Long.toBinaryString(originalLength);
    StringBuilder length128 = new StringBuilder();
    for (int i = 0; i < 64; i++) {
        length128.append("0");
    }
    while (lengthBinary.length() < 64) {
        lengthBinary = "0" + lengthBinary;
    }
```

```
        length128.append(lengthBinary);
        padded.append(length128);
        return padded.toString();
    }
    public static long[] calculateFirst20Words(String paddedMessage) {
        long[] words = new long[80]; // SHA-512 uses 80 words total
        for (int i = 0; i < 16; i++) {
            String word64 = paddedMessage.substring(i * 64, (i + 1) * 64);
            words[i] = Long.parseUnsignedLong(word64, 2);
        }
        for (int i = 16; i < 20; i++) { // Only calculating first 20 as requested
            long s0 = rightRotate(words[i-15], 1) ^ rightRotate(words[i-15], 8) ^ (words[i-15] >>>
7);
            long s1 = rightRotate(words[i-2], 19) ^ rightRotate(words[i-2], 61) ^ (words[i-2] >>> 6);
            words[i] = words[i-16] ^ s0 ^ words[i-7] ^ s1; // CHANGED: Using XOR instead of
addition
        }
        return words;
    }
    public static long rightRotate(long value, int positions) {
        return (value >>> positions) | (value << (64 - positions));
    }
    public static void calculateFunctions() {
        System.out.println("\nFunction Results:");
        System.out.println("================");
        long[] buffers = new long[6];

        for (int i = 0; i < 6; i++) {
            String binaryValue = BUFFER_VALUES[i].replace(" ", "");
            buffers[i] = Long.parseLong(binaryValue, 2);
```

```
    }
    long A = buffers[0], B = buffers[1], C = buffers[2];
    long majority = (A & B) ^ (A & C) ^ (B & C);
    System.out.println("Majority Function Result:");
    System.out.printf("Maj(A,B,C) = %04x\n", majority);
    long D = buffers[3], E = buffers[4], F = buffers[5];
    long choice = (D & E) ^ (~D & F);

    System.out.println("\nCondition Function Result:");
    System.out.printf("Ch(D,E,F) = %04x\n", choice);
  }
}
```

Output :

```
PS D:\Documents\Crypto> javac SHA512.java ; java SHA512
SHA-512 Implementation
======================
Input: SATURDAY
Input in binary: 0101001101000001010101000101010101010010010001000100000101011001
Input length in bits: 64

Padding Information:
Original message length: 64 bits
Padding bits to be added: 832 bits (1 followed by 831 zeros)
Total length after padding: 1024 bits
Number of 1024-bit blocks: 1
```

```
First 20 words (W0 to W19):
W00: 5341545552444159
W01: 8000000000000000
W02: 0000000000000000
W03: 0000000000000000
W04: 0000000000000000
W05: 0000000000000000
W06: 0000000000000000
W07: 0000000000000000
W08: 0000000000000000
W09: 0000000000000000
W10: 0000000000000000
W11: 0000000000000000
W12: 0000000000000000
W13: 0000000000000000
W14: 0000000000000000
W15: 0000000000000040
W16: 12c1545552444159
W17: 8008000000000201
W18: 1e6a85a3ede1b185
W19: 0200100100001004
```

```
Function Results:
=================
Majority Function Result:
Maj(A,B,C) = 8c31

Choice Function Result:
Ch(D,E,F) = e109
```

Manual Calculation :

ASSESSMENT - 08

SHA - 512 Algorithm.

plaintext : "SATURDAY".

a) Length of padding bits.

$$\left.\begin{array}{r}\text{length of plaintext} \\ \text{(in bits)}\end{array}\right\} = 8 \times 8 = 64 \text{ bits}$$

length of padding bits $= 64 + ? = 896 \bmod 1024$

$= 64 + \underline{832} = 896 \bmod 1024$

$$\boxed{\text{Length of padding bits} = 832}$$

b) First 20 64-bit words in hexadecimal format (W_0 to W_{19})

W_0 = SATURDAY = 01010011010000010101010001010101
01010010010001000100000101011001

= 53 41 54 55 52 44 41 59.

$W_1 = \underbrace{1000\ldots\ldots00}_{64 \text{ bits}}$ = 80 00 00 00 00 00 00 00

$$\left.\begin{array}{l} W_2 \\ W_3 \\ W_4 \\ \vdots \\ W_{13} \\ W_{14} \end{array}\right\} = 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$$

W_{15} = 00 00 00 00 00 00 00 40

$$W_{16} = W_{16-16} \oplus \sigma_0^{512}(W_{16-15}) \oplus W_{16-7} \oplus \sigma_1^{512}(W_{16-2})$$
$$= W_0 \oplus \sigma_0^{512}(W_1) \oplus W_9 \oplus \sigma_1^{512}(W_{14})$$

$$\sigma_0^{512}(W_1) = ROTR^1(W_1) \oplus ROTR^8(W_1) \oplus SHR^7(W_1)$$

$ROTR^1(W_1)$ = 40 00 00 00 00 00 00 00
$ROTR^8(W_1)$ = 00 80 00 00 00 00 00 00
$SHR^7(W_1)$ = 01 00 00 00 00 00 00 00
41 80 00 00 00 00 00 00

W_{16} = 53 41 54 55 52 44 41 59
41 80 00 00 00 00 00 00
12 C1 54 55 52 44 41 59

0101
0100
0001

011
001
010

$$W_{17} = W_1 \oplus \sigma_0^{512}(W_2) \oplus W_{10} \oplus \sigma_1^{512}(W_{15})$$

$$\sigma_1^{512}(W_{15}) = ROTR^{19}(W_{15}) \oplus ROTR^{61}(W_{15}) \oplus SHR^6(W_{15})$$

$ROTR^{19}(W_{15})$ = 00 08 00 00 00 00 00 00
$ROTR^{61}(W_{15})$ = 00 00 00 00 00 00 02 00
$SHR^6(W_{15})$ = 00 00 00 00 00 00 00 01
00 08 00 00 00 00 02 01

W_{17} = 80 00 00 00 00 00 00 00
00 08 00 00 00 00 02 01
80 08 00 00 00 00 02 01

$$W_{18} = W_2 \oplus \sigma_0^{512}(W_3) \oplus W_{11} \oplus \sigma_1^{512}(W_{16})$$

$\sigma_1^{512}(w_{16}) = ROTR^{19}(w_{16}) \oplus ROTR^{61}(w_{16}) \oplus SHR^{6}(w_{16})$

w_{16} = 0001001011000001 01010100010101010101
0010 0100 01000100 000101 $\boxed{011001}$

$ROTR^{19}$ = 10001000001010110010001001011000001010101
00010101010101010010 01000

$ROTR^{61}$ = 1001011000001010101000101010101010010
0100 01000100000101011 001000

$SHR^{6}(w_{16})$ = 0000 0000001001011000000101010 1000
10101010101001001000100010 0000101

$\sigma_1^{512}(w_{16})$ = 1E 6A 85 A3 ED E1 B1 85

w_{18} = 1E 6A 85 A3 ED E1 B1 85

$w_{19} = w_3 \oplus \sigma_0^{512}(w_4) \oplus w_{12} \oplus \sigma_1^{512}(w_{17})$

$\sigma_1^{512}(w_{17}) = ROTR^{19}(w_{17}) \oplus ROTR^{61}(w_{17}) \oplus SHR^{6}(w_{17})$

w_{17} = 100$\underline{0}$ 0000 00001000 0000 0000 0000 00000000
0000 0000 0000 0000000010 00 $\boxed{000001}$

$ROTR^{19}(w_{17})$ = 000000000100000000011000000000001
00000000 000000000000000000000 00000

$ROTR^{61}(w_{17})$ = 0000000001000000000000000000000000
000000000 00000000000 001000000001100

$SHR^{6}(w_{17})$ = 00000010000000000000001000000000000000
000000 0000000000000000000001000 .

$\sigma_1^{512}(w_{17})$ = 02 00 10 01 00 00 ϕ0 0A

c) Majority function of round operation on buffers A, B and C.

A - 00111100 00111100 .
B - 10001000 00010001
C - 11001100 00110011

Majority $(x, y, z) = (x \text{ AND } y) \oplus (y \text{ AND } z) \oplus (z \text{ AND } x)$

A AND B = 00001000 00010000
B AND C = 10001000 00010001
C AND A = 00001100 00110000

10001100 00110001

$\Rightarrow \boxed{8C\ 31}$

d) condition function of round operation on buffers E, F and D.

D - 10011001 01100110
E - 11100111 00011000
F - 11110000 00001111

condition $(x, y, z) = (x \text{ AND } y) \oplus (\bar{x} \text{ AND } z)$

(D AND E) = 10000001 00000000
($\bar{D}$ AND F) = 01100000 00001001

11100001 00001001

$\Rightarrow \boxed{E1\ 09}$

Result

Therefore the simulated results are theoretically verified by the manual calculation results