WEEK -8

8.Write a program in Java to implement Secure Hash Algorithms ?

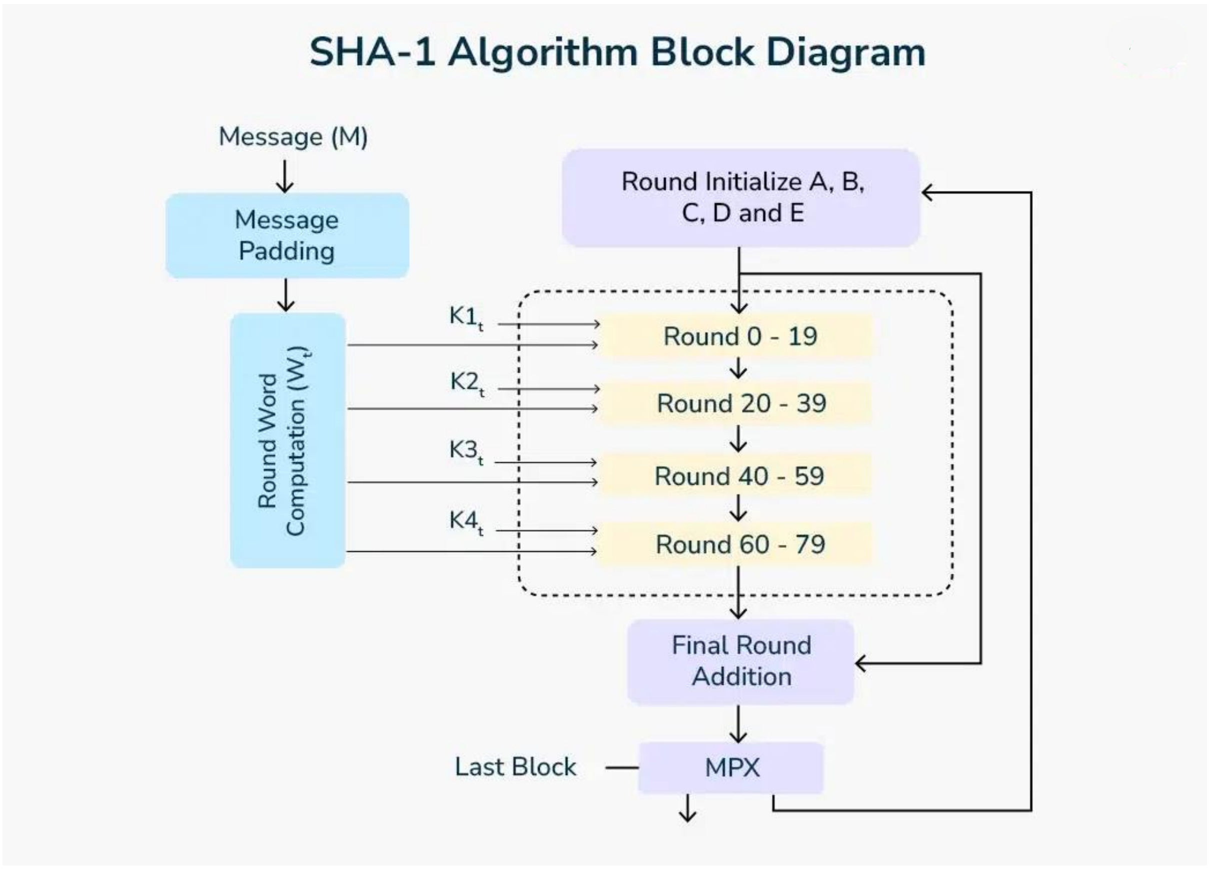
AIM : To implement Secure Hash Algorithm using java.

OBJECTIVE: To understand the the secret key generation using Secure Hash Algorithm using java

THEORY:

SHA-1 or Secure Hash Algorithm 1 is a cryptographic algorithm that takes an input and produces a 160-bit (20-byte) hash value. This hash value is known as a message digest. This message digest is usually then rendered as a hexadecimal number which is 40 digits long. It is a U.S. Federal Information Processing Standard and was designed by the United States National Security Agency. SHA-1 is been considered insecure since 2005. Major tech giants browsers like Microsoft, Google, Apple, and Mozilla have stopped accepting SHA-1 SSL certificates by 2017.

ALGORITHM BLOCK DIAGRAM :



HOW SHA WORKS :

How SHA-1 Works

The block diagram of the SHA-1 (Secure Hash Algorithm 1) algorithm. Here's a detailed description of each component and process in the diagram:

Components and Process Flow:

1. Message (M):
    - The original input message that needs to be hashed.
2. Message Padding:
    - The initial step where the message is padded to ensure its length is congruent to 448 modulo 512. This step prepares the message for processing in 512-bit blocks.
3. Round Word Computation (WtW_tWt):
    - After padding, the message is divided into blocks of 512 bits, and each block is further divided into 16 words of 32 bits. These words are then expanded into 80 32-bit words, which are used in the subsequent rounds.
4. Round Initialize (A, B, C, D, and E):
    - Initialization of five working variables (A, B, C, D, and E) with specific constant values. These variables are used to compute the hash value iteratively.
5. Round Constants (KtK_tKt):
    - SHA-1 uses four constant values (K1K_1K1, K2K_2K2, K3K_3K3, K4K_4K4), each applied in a specific range of rounds:
        - K1K_1K1 for rounds 0-19
        - K2K_2K2 for rounds 20-39
        - K3K_3K3 for rounds 40-59
        - K4K_4K4 for rounds 60-79
6. Rounds (0-79):
    - The main computation loop of SHA-1, divided into four stages (each corresponding to one of the constants K1K_1K1 to K4K_4K4). In each round, a combination of logical functions and operations is performed on the working variables (A, B, C, D, and E) using the words generated in the previous step.
7. Final Round Addition:
    - After all 80 rounds, the resulting values of A, B, C, D, and E are added to the original hash values to produce the final hash.
8. MPX (Multiplexing):
    - Combines the results from the final round addition to form the final message digest.

Summary of Steps:

- Input (Message M): The process starts with the input message MMM.

- Message Padding: The message is padded to meet the length requirements.
- Word Computation: The padded message is divided into blocks and further into words, which are expanded for use in the rounds.
- Initialization: Initial hash values are set.
- Round Processing: The main loop performs 80 rounds of computation using the message words and round constants.
- Final Addition: The results from the rounds are added to the initial hash values.
- Output (Hash Value): The final message digest is produced.

Cryptographic Hash Functions in Java

To calculate cryptographic hash values in Java, the MessageDigest class is used, which is part of the java.security package. The MessageDigest class provides the following cryptographic hash functions:

- MD2
- MD5
- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

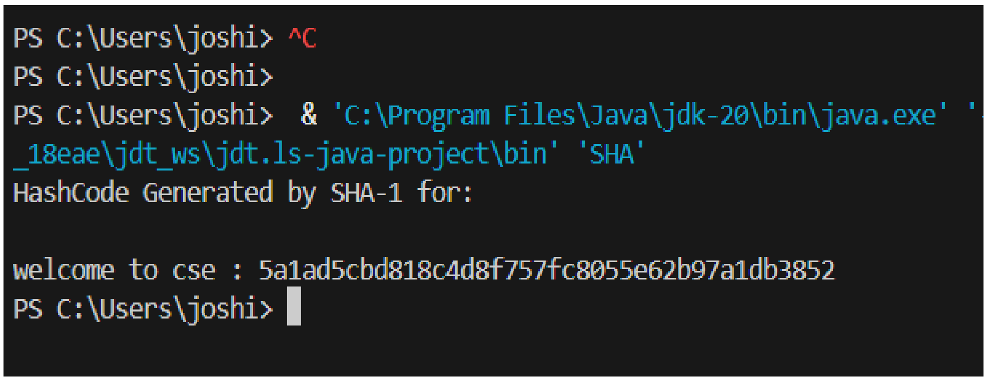
These algorithms are initialized in static method called getInstance(). After selecting the algorithm the message digest value is calculated and the results are returned as a byte array. BigInteger class is used, to convert the resultant byte array into its signum representation. This representation is then converted into a hexadecimal format to get the expected MessageDigest.

PROGRAM :

```
// Java program to calculate SHA-1 hash value

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;


public class  SHA {

   public static String encryptThisString(String input) {
    try {
        // getInstance() method is called with algorithm SHA-1
        MessageDigest md = MessageDigest.getInstance("SHA-1");

        // digest() method is called
        // to calculate message digest of the input string
        // returned as array of byte
        byte[] messageDigest = md.digest(input.getBytes());

        // Convert byte array into signum representation
        BigInteger no = new BigInteger(1, messageDigest);

        // Convert message digest into hex value
        String hashtext = no.toString(16);

        // Add preceding 0s to make it 40 digits long
        while (hashtext.length() < 40) {
            hashtext = "0" + hashtext;
        }

        // return the HashText
        return hashtext;
    }

    // For specifying wrong message digest algorithms
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
```

```
    // Driver code
    public static void main(String args[]) throws NoSuchAlgorithmException {
        System.out.println("HashCode Generated by SHA-1 for:");

        String s1 = "welcome to cse";
        System.out.println("\n" + s1 + " : " + encryptThisString(s1));


    }
}
```

OUTPUT:



```
PS C:\Users\joshi> ^C
PS C:\Users\joshi>
PS C:\Users\joshi>  & 'C:\Program Files\Java\jdk-20\bin\java.exe' '
_18eae\jdt_ws\jdt.ls-java-project\bin' 'SHA'
HashCode Generated by SHA-1 for:

welcome to cse : 5a1ad5cbd818c4d8f757fc8055e62b97a1db3852
PS C:\Users\joshi>
```