Security Lab
Lab Assignment No. 8

Aim: Write a program to demonstrate integrity management by implementing message digest using SHA-256.

SHA-256 is a more secure and newer cryptographic hash function that was launched in 2000 as a new version of SHA functions and was adopted as FIPS standard in 2002. It is allowed to use a hash generator tool to produce a SHA-256 hash for any string or input value. Also, it generates 256 hash values, and the internal state size is 256 bit and the original message size is up to 264-1 bits.

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way. The 256-bit key makes it a good partner-function for AES. It is defined in the NIST (National Institute of Standards and Technology) standard 'FIPS 180-4'. NIST also provides a number of test vectors to verify correctness of implementation. There is a good description at Wikipedia.

Application:
Cryptography
Data Integrity

Code:

```
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate SHA hash value
class sha256 {
	public static byte[] getSHA(String input) throws NoSuchAlgorithmException {
		// Static getInstance method is called with hashing SHA
		MessageDigest md = MessageDigest.getInstance("SHA-256");
		// digest() method called to calculate message digest of an input and return array of byte
		return md.digest(input.getBytes(StandardCharsets.UTF_8));
	}

	public static String toHexString(byte[] hash) {
		// Convert byte array into signum representation
```

```
            BigInteger number = new BigInteger(1, hash);
            // Convert message digest into hex value
            StringBuilder hexString = new StringBuilder(number.toString(16));
            // Pad with leading zeros
            while (hexString.length() < 32) {
                hexString.insert(0, '0');
            }
            return hexString.toString();
        }

        // Driver code
        public static void main(String args[]) {
            try {
                System.out.println("HashCode Generated by SHA-256 for:");
                String s1 = "Ninad Rao";
                System.out.println("\n" + s1 + " : " + toHexString(getSHA(s1)));
                String s2 = "All that glitters is not gold";
                System.out.println("\n" + s2 + " : " + toHexString(getSHA(s2)));
            } catch (NoSuchAlgorithmException e) {
                // For specifying wrong message digest algorithms
                System.out.println("Exception thrown for incorrect algorithm: " + e);
            }
        }
}
```

Output:

```
PS D:\III Year Engineering\CNS Lab Experiments>  & 'c:\Users\Ninad Rao\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'D:\Java\jdk-11.0.12\bi
n\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\Ninad Rao\AppData\Roaming\Code\User\workspaceStorage\e22ac086d4ad1c01f25fb96aee96dda2\redhat.java\jdt_ws\CNS Lab Exp
eriments_37835a1c\bin' 'sha256'
HashCode Generated by SHA-256 for:

Ninad Rao : 8dd6cc58ad8cac5ad942218eb51f5d51e57ff7aa6d36e48f78dc3dae25891095

All that glitters is not gold : 9b8c48af3e20b9bff2ffb4bad61a64611675b96a1ab772be62b410d7118c0961
```

Conclusion: Thus we understand how to demonstrate integrity management by implementing message digest using SHA-256.