K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch:_B4	**Roll No.: 1721022**
Experiment No. 2	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title : Implementation of Integrity goal in suitable Application using Hash algorithm

Objective: A hash function is any function that can be used to map data of arbitrary size onto data of a fixed size.

Expected Outcome of Experiment:

CO	Outcome
CO3	Apply Cryptographic Hash functions system security

Books/ Journals/ Websites referred:

https://www.wikipedia.org/

https://www.geeksforgeeks.org/

https://crypto.interactive-maths.com/route-cipher.html

https://crypto.interactive-maths.com/playfair-cipher.html

https://www.includehelp.com/data-structure-tutorial/hashing.aspx

https://www.includehelp.com/data-structure-tutorial/hashing.aspx

Abstract:-

Hashing:

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value. It is also used in many encryption algorithms.

Data Integrity

Data integrity is the overall completeness, accuracy and consistency of data. This can be indicated by the absence of alteration between two instances or between two

updates of a data record, meaning data is intact and unchanged. Data integrity is usually imposed during the database design phase through the use of standard procedures and rules. Data integrity can be maintained through the use of various error-checking methods and validation procedures.

Message Authentication

In information security, message authentication or data origin authentication is a property that a message has not been modified while in transit (data integrity) and that the receiving party can verify the source of the message. Message authentication does not necessarily include the property of non-repudiation

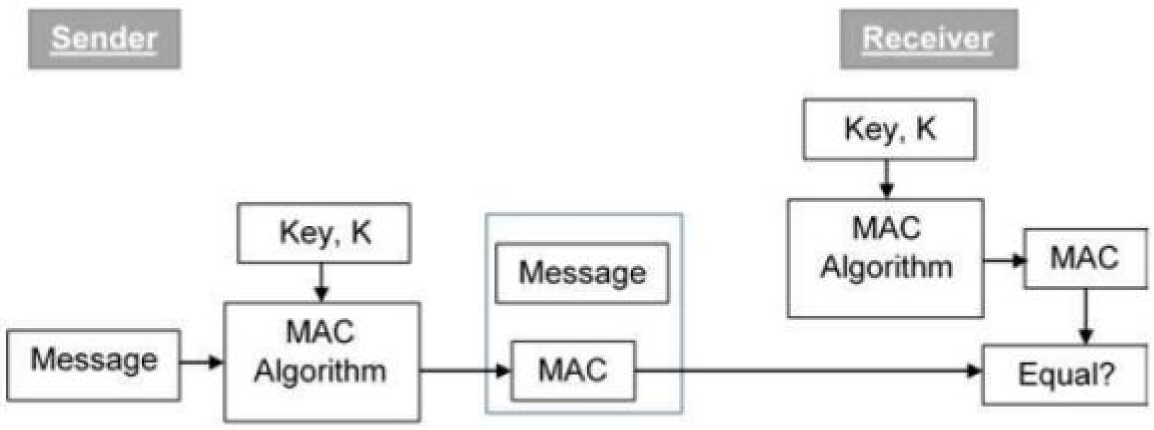
Related Theory: -

Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Detail process is given below

• The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.

• Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.

• The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.

• On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re- computes the MAC value.

• The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.

• If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Implementation Details:

```
import java.util.*; import
java.io.*;

class Hashing {


     public static String myHashAlgorithm(String message, String key) throws
Exception{

                //Step 1: msg | key at binary level byte[] msg
                = message.getBytes("UTF_16");

                String msgBin = ""; for (int i=0;
                i<msg.length; i++) {

                     msgBin += String.format("%8s", Integer.toBinaryString((msg[i] +
256) % 256)).replace(' ', '0');

                } byte[] keyBytes = key.getBytes("UTF_16");

                String keyBin = ""; for (int i=0;
                i<keyBytes.length; i++) {


                     keyBin += String.format("%8s", Integer.toBinaryString((msg[i] +
256) % 256)).replace(' ', '0');

} String keyAdjusted = String.format("%" + msgBin.length() + "s",keyBin).replace(' ',
'0');

                String andString = "";

                for (int i=0; i<msgBin.length(); i++) {

                     andString += Integer.parseInt(msgBin.charAt(i)+"") |
Integer.parseInt(keyAdjusted.charAt(i)+"");

                }

                System.out.println(andString.length());
                //Adjusting the AND'ed string to a fixed 128 bit length String
                andStringAdjusted = String.format("%128s", "").replace(' ', '0');

                  for (int i=0; i<andString.length(); i+=128) {
                          String andStringNew = "",a;

                        if (i+127 > andString.length()) {

                             a = String.format("%128s", andString.substring(i)).replace('
', '0');

                        }else{

                             a = String.format("%128s", andString.substring(i,
i+127)).replace(' ', '0');

                        } for (int j=0; j<a.length(); j++) {

                             andStringNew += Integer.parseInt(a.charAt(j)+"",2) ^
Integer.parseInt(andStringAdjusted.charAt(j)+"",2);

                        } andStringAdjusted = andStringNew;

                } andString = andStringAdjusted;

                int grpLength = andString.length() / 4;
                for (int j=0; j<4; j++) {

                        //Step 2: Form 4 groups A,B,C,D String A =
                        andString.substring(0,grpLength);


                        String B = andString.substring(1*grpLength,2*grpLength);
                        String C = andString.substring(2*grpLength,3*grpLength);

                        String D = andString.substring(3*grpLength,4*grpLength);


                        //Step 3: (B AND C) ^ (~B AND D)
                        String res = "";

                        for (int i=0; i<grpLength; i++) {

                             res        +=       (Integer.parseInt(B.charAt(i)+"")         &
Integer.parseInt(C.charAt(i)+""))       ^       (~Integer.parseInt(B.charAt(i)+"")         &
Integer.parseInt(D.charAt(i)+""));
```

```
                        }


                        //Step 4: (A+res)
                        String newA = "";

                        for (int i=0; i<res.length(); i++) {

                             newA                                                    +=
Integer.toBinaryString(Integer.parseInt(A.charAt(i)+"",2)                            +
Integer.parseInt(res.charAt(i)+"",2));

                        }


                    //Step 5: (newA) & 2^grplength (Since the length increases after
adding both so we keep the length)

res = ""; String grplengthBin = String.format("%" + newA.length() +
"s",String.format("%" + grpLength + "s","").replace(' ', '1')).replace(' ', '0');

                        for (int i=0; i<grplengthBin.length(); i++){

                             res += Integer.parseInt(newA.charAt(i)+"") &
Integer.parseInt(grplengthBin.charAt(i)+"");

                        } newA = res.substring(res.length()-grpLength);

                        //Step 6: D,Res,B,C and repeat String ans=
                        newA + "" + B + "" + C + "" + D;

                    andString = ans.substring(grpLength) + ans.substring(0,
grpLength);

                }


                String hashStr = ""; for (int i=0;
                i<andString.length(); i+=4) {

                     hashStr +=
Integer.toHexString(Integer.parseInt(andString.substring(0+i, i+4), 2));

                } return hashStr;

        }
```

```
	public static void main(String[] args) throws Exception {

		String s1 = fileRead("test1.txt");

		String s2 = fileRead("test2.txt");
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter the key");
		String key = sc.nextLine();

		System.out.println("Sent Message: "+s1);
		String hash1 = myHashAlgorithm(s1,key);

	      System.out.println(" Sent Message Hash - " + hash1);// this is the hash of
the key and the message to be sent together

		System.out.println("Received Message: "+s2);
		String hash2 = myHashAlgorithm(s2,key);

	      System.out.println(" Received Message Hash - " + hash2);//this is the
hash of the key and the message that is received together.

	      if (hash1.equals(hash2)) {//this checks whether both the messages are
same or not

			System.out.println("Both messages are same. Message
Authenticated.");

		}else{

			System.out.println("Both are different. Message
Unauthenticated.");

		}

	}


	public static String fileRead(String filename) throws Exception{

		File file = new File(filename);


		Scanner sc = new Scanner(file);
		String filecontents = "";

		while (sc.hasNextLine()) {
```

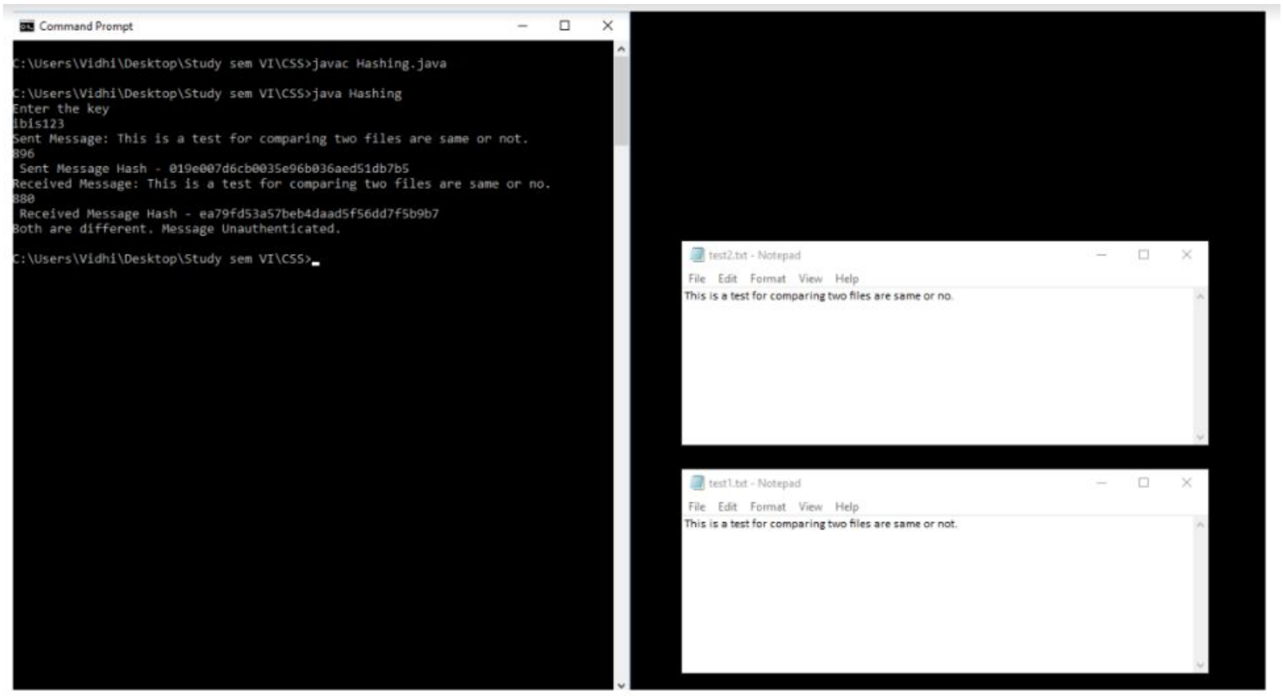
```
                    filecontents += sc.nextLine();

                } return filecontents;

        }


}
```

Outpu
t:



```
Command Prompt

C:\Users\Vidhi\Desktop\Study sem VI\CSS>javac Hashing.java

C:\Users\Vidhi\Desktop\Study sem VI\CSS>java Hashing
Enter the key
ibis123
Sent Message: This is a test for comparing two files are same or not.
896
 Sent Message Hash - 019e007d6cb0035e96b036aed51db7b5
Received Message: This is a test for comparing two files are same or no.
880
 Received Message Hash - ea79fd53a57beb4daad5f56dd7f5b9b7
Both are different. Message Unauthenticated.

C:\Users\Vidhi\Desktop\Study sem VI\CSS>_
```

```
test2.txt - Notepad
File  Edit  Format  View  Help
This is a test for comparing two files are same or no.
```

```
test1.txt - Notepad
File  Edit  Format  View  Help
This is a test for comparing two files are same or not.
```

Conclusion:- Hence, We understood how To implement Message Authentication using hashing.

Date: ______________   **Signature of faculty in-charge**