Practical No – 03
Message Authentication Codes

Aim : Implement algorithms to generate and verify message authentication codes (MACs) for ensuring data integrity and authenticity.

Source code:

```
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class MD5 {
    public static String toHexString(byte[] hash) {
    BigInteger number = new BigInteger(1, hash);
    StringBuilder hexString = new StringBuilder(number.toString(16));
    while (hexString.length() > 32) {
       hexString.insert(0, '0');
    }
    return hexString.toString(); }
  public static void main(String args[])throws NoSuchAlgorithmException
  {
    System.out.println("Hashcode Generated by MD5 for:");
       String s1= "Information and security";
       MessageDigest md=MessageDigest.getInstance("MD5");
       byte[] hash=md.digest(s1.getBytes(StandardCharsets.UTF_8));
       System.out.println("Message Digest: "+s1+":"+toHexString(hash));
   }}
```

Output:

```
run:
Hashcode Generated by MD5 for:
Message Digest: Information and security:c971095f58c8c7aa2aba10c9f61ebd82
BUILD SUCCESSFUL (total time: 0 seconds)
```

Source code:

```
import hashlib
result=hashlib.md5(b'good')
result=result.hexdigest()
print('Message Digest', result)
```

Output:

```
Message Digest 755f85c2723bb39381c7379a604160d8
```

Source code:

```
import hashlib
str = input('Enter String to encode :')
result = hashlib.sha1(str.encode())
result = result.hexdigest()
print("Output of SHA1 ", result)
```

Output:

```
Enter String to encode :hello
Output of SHA1  aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
```

Source code:

```
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto import Random
```

```
def generate_signature(private_key,message):
    key = RSA.importKey(private_key)
    hashed_message = SHA256.new(message.encode('utf-8'))
    signer = PKCS1_v1_5.new(key)
    signature = signer.sign(hashed_message)
    return signature
def verify_signature(public_key,message,signature):
    key  = RSA.importKey(public_key)
    hashed_message = SHA256.new(message.encode('utf-8'))
    verifier = PKCS1_v1_5.new(key)
    return verifier.verify(hashed_message,signature)


random_generator = Random.new().read
key_pair = RSA.generate(2048,random_generator)
public_key = key_pair.publickey().export_key()
private_key = key_pair.export_key()
message = "Hello world!"
signature = generate_signature(private_key, message)
print("Generated Signature: ", signature)
is_valid = verify_signature(public_key, message, signature)
print("Signature Verification Result:", is_valid)
```

Output:

Generated Signature:
b"w6\xdf\x95\x18\xd9\x98\xf5)I\x9d\xc6\x0e&\n<\x0c\xa7O\xcd\xc1\x0f\x0c\xf7\x02\xa3\x17%)\x12\x96!Bn\xa2\x83\x88S\x02\xfd-
\xee\x1a|\xbc\xf5\xab\xfd\xcb\xe3$\xff\xa1\xea)\x84b\xe9\xf1!\r\xb7\x17\x15\xe9\xef\xe7\xd7\x048Y\xdbf\xda\xbe\x9aMt\xdf\xc5\x8d\xd5\xcc\x1d|[\xe2>'\xad\x82\xb1\xd3'\xc4\xabl\xbd9\x94\x03O\xe6\xffz\xfc\xe0\xb9\xdbV\x143\xc97^f,j\xc2\xbb`\x07\x99s{\xd4$\xa9&\x8c_U\x8e\xb3T4\xeb\xdf0\x99\x93\xbd\xc2\x98\x9b4\x17\x81\x99\xf9\x07\x89?\xaa\x99Z\n\xa8\xaa\x03X%\xbe\x8d\xee\xb1\xfbT\xdaD\x00\x86\xa2^\x8f\xecL!\xc9\xca<\x17\x1f\xe2\xd0\x8bd,i\xe3\x0c\xec\xcb\x11\x06\x92a\xa0\xe67\xfbLJ\x1c\xc1\xcb\x8a\xb8\xcck\x02W~\x9a\x9b*\xf7\xaf2\xa4\xee)2\xa4\xca\xb6P\x98\xe2q\x83\x84\xed\xc3k\xc5{h5:\xc0n\xb8\x9c\x1b\x00\x91\xf4\x03\xd0\xd1C\x86\xc05"

Signature Verification Result: True