**Additional Experiments:**

**1. AIM:** Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

**DESCRIPTION:**

The MD5 message-digest algorithm is widely a used hash function producing a 128-bit hash value. Although MD5 was initially designed to be as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption. It remains suitable for other non-cryptographic purposes, for example for determining the partition for a particular key in a partitioned database.

**ALGORITHM:**

1.The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit words), the message is padded so that its length is divisible by 512

2.first a single bit,1 is appended to the end of message and followed by as many as zeros required to bring the length of the message upto 64 bits fewer than a multiple of 512

3.remaining bits are filled up with 64 bits representing length of original message, modulo 264

4.MD5 operates on 128-bit state, divided into four 32-bit words, denoted A, B, C and D

5.algorithm then uses 512-bit message block to modify the state

6.message block consists of 4 stages called rounds

7.each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation

8.There are four possible functions, a different one is used in each round:

F(B, C, D)=(B˄C)˅(¬B˄D)

G(B, C, D)=(B˄D)˅(C˄¬D)

H(B, C, D)=B  C  D

I(B, C, D)=C(B˅¬D)

,˄,˅,¬ denote the XOR, AND, OR and NOT operations respectively.

**PROGRAM:**

import java.security.\*;

public class MD5

{

public static void main(String[] a)

{

// TODO code application logic here

try

{

MessageDigest md = MessageDigest.getInstance("MD5");

System.out.println("Message digest object info: ");

System.out.println(" Algorithm = " +md.getAlgorithm());

System.out.println(" Provider = " +md.getProvider());

System.out.println(" ToString = " +md.toString());

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("MD5(\"" +input+"\") = " +bytesToHex(output));

System.out.println("");

}

catch (Exception e) {

System.out.println("Exception: " +e); }

}

public static String bytesToHex(byte[] b) {

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBufferbuf = new StringBuffer();

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

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

buf.append(hexDigit[b[j] & 0x0f]); }

return buf.toString(); } }

**OUTPUT:**

Message digest object info:

Algorithm = MD5

Provider = SUN version 1.6

ToString = MD5 Message Digest from SUN, <initialized>

MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B

**2. AIM:** Write a java program for RC5 symmetric algorithm.

**DESCRIPTION:**

RC5 is a symmetric-key block cipher notable for its simplicity. Designed by Ronal Rivest in 1994, RC stands for “Rivest Cipher”, or alternatively, “Ron’s Code”. The Advanced Encryption Standard candidate RC6 was based on RC5.

Key sizes: 0 to 2040 bits(128 suggested)

Block sizes: 32, 64 or 128 bits(64 suggested)

**ALGORITHM:**

1.Initialization of constants P and Q. RC5 makes use of 2 magic constants P and Q whose value is defined by the word size w.

2.Converting secret key K from bytes to words. Secret key K of size b bytes is used to initialize array L consisting of c words where c = b/u, u = w/8 and w = word size used for that particular instance of RC5.

3.Initializing sub-key S. Sub-key S of size t=2(r+1) is initialized using magic constants P and Q.

4.Sub-key mixing. The RC5 encryption algorithm uses Sub key S. L is merely, a temporary array formed on the basis of user entered secret key.

5.Encryption. We divide the input plain text block into two registers A and B each of size w bits. After undergoing the encryption process the result of A and B together forms the cipher text block. RC5 Encryption Algorithm:

1. One-time initialization of plain text blocks A and B by adding S[0] and S[1] to A and B respectively. These operations are mod.

2. XOR A and B. A=A^B

3. Cyclic left shift new value of A by B bits.

4. Add S[2\*i] to the output of previous step. This is the new value of A.

5. XOR B with new value of A and store in B.

6. Cyclic left shift new value of B by A bits.

7. Add S[2\*i+1] to the output of previous step. This is the new value of B.

8. Repeat entire procedure (except one-time initialization) r times.

5. XOR B with new value of A and store in B.

6. Cyclic left shift new value of B by A bits.

7. Add S[2\*i+1] to the output of previous step. This is the new value of B.

8. Repeat entire procedure (except one-time initialization) r times.

import javax.crypto.spec.\*;

import java.security.\*;

import javax.crypto.\*;

public class Main

{

private static String algorithm = "RC5";

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

{

toEncrypt = "The shorter you live, the longer you're dead!";

System.out.println("Encrypting...");

byte[] encrypted = encrypt(toEncrypt, "password");

System.out.println("Decrypting...");

String decrypted = decrypt(encrypted, "password");

System.out.println("Decrypted text: " + decrypted);

}

public static byte[] encrypt(String toEncrypt, String key) throws Exception

{

// create a binary key from the argument key (seed)

SecureRandom sr = new SecureRandom(key.getBytes());

KeyGenerator kg = KeyGenerator.getInstance(algorithm);

kg.init(sr);

SecretKey sk = kg.generateKey();

// create an instance of cipher

Cipher cipher = Cipher.getInstance(algorithm);

// initialize the cipher with the key

cipher.init(Cipher.ENCRYPT\_MODE, sk);

// enctypt!

byte[] encrypted = cipher.doFinal(toEncrypt.getBytes());

return encrypted;

}

public static String decrypt(byte[] toDecrypt, String key) throws Exception

{

// create a binary key from the argument key (seed)

SecureRandom sr = new SecureRandom(key.getBytes());

KeyGenerator kg = KeyGenerator.getInstance(algorithm);

kg.init(sr);

SecretKey sk = kg.generateKey();

// do the decryption with that key

Cipher cipher = Cipher.getInstance(algorithm);

cipher.init(Cipher.DECRYPT\_MODE, sk);

byte[] decrypted = cipher.doFinal(toDecrypt);

return new String(decrypted);

}

}

Output

ENTER PLAIN TEXT RC5PROGRAM

ENTER KEY TEXT F

ENCRYPTED: ??-??‚±?‚μFJ|

DECRYPTED: RC5 PROGRAM