**AIM:** Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).

**PROGRAM:**

import java.math.BigInteger;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.SecureRandom; import

javax.crypto.spec.DHParameterSpec; import

javax.crypto.spec.DHPublicKeySpec; public

class DiffeHellman

{

public final static int

pValue = 47;

public final static int gValue = 71;

public final static int XaValue = 9;

public final static int XbValue = 14;

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

{

BigInteger p = new BigInteger(Integer.toString(pValue));

BigInteger g = new BigInteger(Integer.toString(gValue));

BigIntegerXa = new BigInteger(Integer.toString(XaValue));

BigIntegerXb = new BigInteger(Integer.toString(XbValue));

createKey();

intbitLength = 512; // 512 bits

SecureRandomrnd = new SecureRandom();

p = BigInteger.probablePrime(bitLength, rnd);

g = BigInteger.probablePrime(bitLength, rnd);

createSpecificKey(p, g);

}

public static void createKey() throws Exception

{

KeyPairGeneratorkpg = KeyPairGenerator.getInstance("DiffieHellman");

kpg.initialize(512);

KeyPairkp = kpg.generateKeyPair();

KeyFactorykfactory = KeyFactory.getInstance("DiffieHellman");

DHPublicKeySpeckspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("Public key is: " +kspec);

}

public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception

{

KeyPairGeneratorkpg = KeyPairGenerator.getInstance("DiffieHellman");

DHParameterSpecparam = new DHParameterSpec(p, g);

kpg.initialize(param);

KeyPairkp = kpg.generateKeyPair();

KeyFactorykfactory = KeyFactory.getInstance("DiffieHellman");

DHPublicKeySpeckspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),

DHPublicKeySpec.class);

System.out.println("\nPublic key is : " +kspec);

}

}

**OUTPUT:**

Public key is: javax.crypto.spec.DHPublicKeySpec@5afd29

Public key is: javax.crypto.spec.DHPublicKeySpec@9971ad

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

**PROGRAM:**

import java.security.\*;

public class SHA1

{

public static void main(String[] a)

{

try

{

MessageDigest md = MessageDigest.getInstance("SHA1");

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("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("SHA1(\"" +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]); }

returnbuf.toString(); }

}

**OUTPUT:**

Message digest object info:

Algorithm = SHA1

Provider = SUN version 1.6

ToString = SHA1 Message Digest from SUN, <initialized> SHA1("") =

DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 SHA1("abc") =

A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19D8424

0D3A89