**FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING**

**Department of Computer Engineering**

**Course , Subject & Experiment Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Academic Year** | **2020-21** | **Estimated Time** | **02 - Hours** |
| **Course & Semester** | **T.E. (CMPN)- Sem VI** | **Subject Name & Code** | **CSS - (CSL604)** |
| **Module No.** | **02 – Mapped to CO-2** | **Chapter Title** | **Key Management Techniques** |

|  |  |
| --- | --- |
| **Practical No:** | **2** |
| **Title:** | **Implementation of Diffie- Hellman Key exchange algorithm and Simulation of Man In the Middle attack** |
| **Date of Performance:** |  |
| **Date of Submission:** |  |
| **Roll No:** |  |
| **Name of the Student:** |  |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Rubric** | **Grade** |
| **1** | **On time submission**  **Or  completion  (2)** |  |
| **2** | **Preparedness(2)** |  |
| **3** | **Skill (4)** |  |
| **4** | **Output (2)** |  |

**Signature of the Teacher:**

**Date:**

**Title: Implementation of Diffie- Hellman Key exchange algorithm and Simulation of Man In the Middle attack.**

**Lab Objective** :

This lab provides insight into:

* The working of Diffie – Hellman Key Exchange Protocol.

**Reference** : “Cryptography and Network Security” B. A. Forouzan

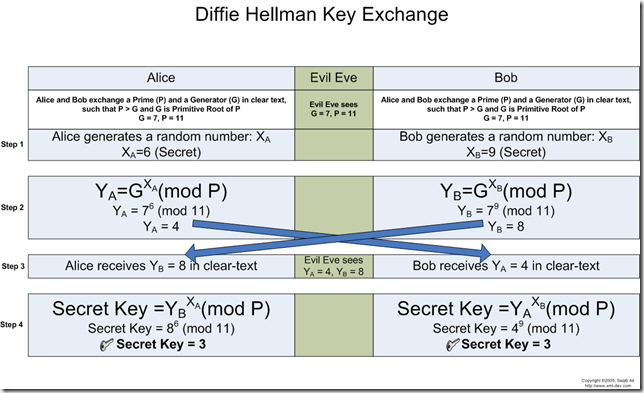
“Cryptography and Network Security” Atul Kahate

**Prerequisite:** Any programming Language andKnowledge of Symmetric Key cryptography.

**Theory:**

Diffie-Hellman is a way of *generating* a shared secret between two people in such a way that the secret can't be seen by observing the communication.

This is particularly useful because you can use this technique to create an encryption key with someone, and then start encrypting your traffic with that key. And even if the traffic is recorded and later analyzed, there's absolutely no way to figure out what the key was, even though the exchanges that created it may have been visible.



**Man – In – The –Middle Attack**

Let us take the example illustrated by Diffie-Hellman to discuss the Man-in-the-Middle Attack. Let us that Eve is in the middle of Alice and Bob. Eve does not need the value of x or y to attack the protocol. She can fool both Alice and Bob by the following process.

**Diagram

Description automatically generated**

1. Alice choose a, calculate A=g^a mod p

2. Eve, the intruder, interpret A, she chooses z, calculate Z=g^z mod p, and sends Z to both Alice and Bob.

3. Bob choose b, calculate B=g^b mod p, and sends B to Alice; B is interpreted by Eve and never reaches Alice.

4. Alice and Eve calculate the same key g^az mod p, which become a shared key between Alice and Eve. Alice however think that it is a key shared between Bob and herself.

5. Eve and Bob calculate the same key g^bz mod p, which become a shared key between Eve and Bob. Bob, however, thinks that it is a key shared between Alice and himself.

This situation is called man-in-the-middle attack.

|  |
| --- |
| **Practical and Real Time Applications**   * Used as a method of exchanging cryptography keys for **use** in symmetric encryption algorithms like AES * Public key encryption schemes based on DF – ElGamal encryption * Password-authenticated key agreement * [public key infrastructure](https://en.wikipedia.org/wiki/Public_key_infrastructure) -  It is possible to use DF as part of PKI |

**Input:**

**Implementation of Diffie- Hellman Key exchange algorithm**

**(Server)**

import java.net.\*;

import java.io.\*;

public class Serverd

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

private DataOutputStream out = null;

public int p = 23;

public int g = 5;

private int a = 4;

private int A = (int)Math.pow(g, a)%p;

public int B;

private int key1;

public Serverd(int port)

{

try

{

server = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String Ainstring = String.valueOf(A);

out.writeUTF(Ainstring);

B = Integer.parseInt(in.readUTF());

System.out.println("Public key of server (A) = "+A);

System.out.println("Public key received from client = "+B);

key1 = (int)Math.pow(B,a)%p;

System.out.println("Key = "+key1);

System.out.println("");

System.out.println("Closing connection");

socket.close();

in.close();

out.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Serverd serverd = new Serverd(7000);

}

}

**(Client)**

import java.net.\*;

import java.io.\*;

public class Clientd

{

private Socket socket = null;

private DataOutputStream out = null;

private DataInputStream in = null;

public int p = 23;

public int g = 5;

private int b = 3;

private int B = (int)Math.pow(g, b)%p;

public int A;

private int key2;

public Clientd(String address, int port)

{

try

{

socket = new Socket(address, port);

System.out.println("Connected");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

A = Integer.parseInt(in.readUTF());

String Binstring = String.valueOf(B);

out.writeUTF(Binstring);

System.out.println("Public key of client (B) = "+B);

System.out.println("Public key received from server = "+A);

key2 = (int)Math.pow(A, b)%p;

System.out.println("Key = "+key2);

System.out.println("");

System.out.println("Closing connection");

out.close();

socket.close();

in.close();

}

catch(UnknownHostException u)

{

System.out.println(u);

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Clientd clientd = new Clientd("127.0.0.1", 7000);

}

}

**Output:**

**Graphical user interface, text

Description automatically generated**

**Simulation of Man In the Middle attack**

**(Server)**

import java.net.\*;

import java.io.\*;

public class Server

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

private DataOutputStream out = null;

public int p = 23;

public int g = 5;

private int a = 4;

private int A = (int)Math.pow(g, a)%p;

public int B;

private int key1;

public Server(int port)

{

try

{

server = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String Ainstring = String.valueOf(A);

out.writeUTF(Ainstring);

B = Integer.parseInt(in.readUTF());

System.out.println("Public key of server (A) = "+A);

System.out.println("Public key received by server = "+B);

key1 = (int)Math.pow(B,a)%p;

System.out.println("Key = "+key1);

System.out.println("");

System.out.println("Closing connection");

socket.close();

in.close();

out.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Server server = new Server(5000);

}

}

**(Attacker)**

import java.net.\*;

import java.io.\*;

public class Attacker

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

private DataOutputStream out = null;

public int p = 23;

public int g = 5;

private int ea = 7;

private int eb = 9;

private int EA = (int)Math.pow(g, ea)%p;

private int EB = (int)Math.pow(g, eb)%p;

public int A;

public int B;

private int key\_server;

private int key\_client;

public Attacker(String address, int port)

{

try

{

socket = new Socket(address, port);

System.out.println("Connected");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

A = Integer.parseInt(in.readUTF());

String EBinstring = String.valueOf(EB);

out.writeUTF(EBinstring);

System.out.println("Public key of attacker sent to server sender(EB) = "+EB);

System.out.println("Public key received from server(A) = "+A);

key\_server = (int)Math.pow(A, eb)%p;

System.out.println("Server Key = "+key\_server);

System.out.println("");

System.out.println("Closing connection");

System.out.println("");

out.close();

socket.close();

in.close();

}

catch(UnknownHostException u)

{

System.out.println(u);

}

catch(IOException i)

{

System.out.println(i);

}

}

public Attacker(int port)

{

try

{

server = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String EAinstring = String.valueOf(EA);

out.writeUTF(EAinstring);

B = Integer.parseInt(in.readUTF());

System.out.println("Public key of attacker sent to client sender(EA) = "+EA);

System.out.println("Public key received from client(B) = "+B);

key\_client = (int)Math.pow(B,ea)%p;

System.out.println("Client Key = "+key\_client);

System.out.println("");

System.out.println("Closing connection");

socket.close();

in.close();

out.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Attacker aclient = new Attacker("127.0.0.1", 5000);

Attacker aserver = new Attacker(6000);

}

}

**(Client)**

import java.net.\*;

import java.io.\*;

public class Client

{

private Socket socket = null;

private DataOutputStream out = null;

private DataInputStream in = null;

public int p = 23;

public int g = 5;

private int b = 3;

private int B = (int)Math.pow(g, b)%p;

public int A;

private int key2;

public Client(String address, int port)

{

try

{

socket = new Socket(address, port);

System.out.println("Connected");

System.out.println("");

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

A = Integer.parseInt(in.readUTF());

String Binstring = String.valueOf(B);

out.writeUTF(Binstring);

System.out.println("Public key of client (B) = "+B);

System.out.println("Public key received by client = "+A);

key2 = (int)Math.pow(A, b)%p;

System.out.println("Key = "+key2);

System.out.println("");

System.out.println("Closing connection");

out.close();

socket.close();

in.close();

}

catch(UnknownHostException u)

{

System.out.println(u);

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Client client = new Client("127.0.0.1", 6000);

}

}

**Output:**

**Graphical user interface, text

Description automatically generated**

**Conclusion:**

Using the public keys of server and client same keys are generated at client and server side using Diffie-Hellman key exchange algorithm. The attacker in man-in-the-middle-attack successfully exchanges the keys with client and server.

|  |
| --- |
| **Conclusion:** |
| The program was tested for different sets of inputs.  Program is working               SATISFACTORY              NOT SATISFACTORY    ( Tick appropriate outcome) |

|  |
| --- |
| **Post Lab Assignment:** |
| 1. In the Diffie- Hellman protocol , what happens if x and y have the same value, that is, Alice and Bob have accidentally chosen the same number? Are A and B (values exchanged by Alice and Bob to each other) the same? Do the session keys calculated by Alice and Bob have the same value? Use an example to prove your claims.   🡪 If x and y both have same values, the values A and B exchanged by Alice and Bob are same, the session keys calculated by Alice and Bob are also same.  Eg:-Let p=23  g=5  Let Alice choose the number(x)=9  Let Bob choose the number(y)=9  A(calculated by Alice)=gxmodp=59mod23=11  B(balculated by Bob)=gymodp=59mod23=11  K(key generated by Alice)=Bxmodp=119mod23=5  K(key generated by Bob)=Aymodp=119mod23=5  Therefore, the session keys calculated by Alice and Bob are same.   1. How to secure Diffie-Hellman from Man-in –the –Middle attack?   🡪 One way to secure Diffie-Hellman from Man-in-the-Middle attack is to use digital signature. Digital Signature is encrypted using private key of sender. Agreeing upon digital signature ensures proper authentication. The attacker cannot create digital signature. The receiver will come to know if the message is changed by the attacker. |