Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 3

По дисциплине СМЗКС

Тема: «Протоколы аутентификации»

Выполнил:

Студент 4 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Хацкевич М.В.

Брест, 2022

**Цель работы:** приобретение практических навыков работы с протоколами аутентификации.

**Задание:**

Разработать приложение с реализацией Протокола Нидхэма и Шредера.

**Алгортим:**

1. A https://rudocs.exdat.com/data/77/76876/76876_html_m6934ea8b.pngKDC: IDA || IDB || N1  
2. KDC https://rudocs.exdat.com/data/77/76876/76876_html_m6934ea8b.pngA: EKa [KS || IDB || N1 || EKb [KS || IDA] ]  
3. A https://rudocs.exdat.com/data/77/76876/76876_html_m6934ea8b.pngB: EKb [KS || IDA]  
4. B https://rudocs.exdat.com/data/77/76876/76876_html_m6934ea8b.pngA: EKS [N2]  
5. A https://rudocs.exdat.com/data/77/76876/76876_html_m6934ea8b.pngB: EKS [f (N2)]

**Код программы:**

|  |
| --- |
| **class A:** |

import java.io.\*;

import java.math.BigInteger;

import java.net.Socket;

import java.net.UnknownHostException;

public class A {

private static final String N1 = String.valueOf((int) (Math.random() \* 1000000));

static String modifyN2(String N2) {

return String.valueOf(new BigInteger(N2).add(BigInteger.ONE));

}

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

try (Socket socket = new Socket("localhost", 3345);

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

DataInputStream ois = new DataInputStream(socket.getInputStream())) {

System.out.println("Client connected to socket.");

String replyFromKDC2 = "";

while (!socket.isOutputShutdown()) {

System.out.println("Client start writing in channel...");

Storage.printLine();

Thread.sleep(1000);

String clientCommand1 = String.join(Storage.DELIMITER, Storage.ID\_A, Storage.ID\_B, N1);

oos.writeUTF(clientCommand1);

oos.flush();

System.out.println("Client sent message " + clientCommand1 + " to server.");

Thread.sleep(1000);

System.out.println("Client sent message & start waiting for data from server...");

Thread.sleep(2000);

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

replyFromKDC2 = ois.readUTF();

System.out.println("replyFromKDC2 = " + replyFromKDC2);

Storage.printLine();

break;

}

connectWithB(replyFromKDC2);

System.out.println("Closing connections & channels on clientSide - DONE.");

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

static void connectWithB(String replyFromKDC) {

String decryptedReplyFromKDS = Scrambler.decrypt(Storage.KEY\_A, replyFromKDC);

System.out.println("Decrypted reply from KDS: " + decryptedReplyFromKDS);

String[] array = decryptedReplyFromKDS

.split(Storage.DELIMITER);

String ks = array[0];

String idB = array[1];

String N1FromKDC = array[2];

if (N1.equals(N1FromKDC)) {

System.out.println("Got N1 from KDS - " + N1FromKDC);

} else {

System.out.println("Expected: " + N1);

System.out.println("Got: " + N1FromKDC);

throw new RuntimeException("N1 from KDS is wrong!!!");

}

System.out.println("KS: " + ks);

System.out.println("ID\_B: " + idB);

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

String messageToB3 = array[3];

boolean established = false;

try (Socket socket = new Socket("localhost", 3346);

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

DataInputStream ois = new DataInputStream(socket.getInputStream())) {

System.out.println("A connected to socket B.");

while (!socket.isOutputShutdown()) {

if (!established) {

System.out.println("A start writing in channel...");

oos.writeUTF(messageToB3);

oos.flush();

System.out.println("A sent message " + messageToB3 + " to B.");

System.out.println("A sent message & start waiting for data from server...");

Thread.sleep(2000);

System.out.println("reading B...");

String replyFromB4 = ois.readUTF();

String decryptedN2 = Scrambler.decrypt(ks, replyFromB4);

System.out.println("Decrypted N2 from B = " + decryptedN2);

String modifiedN2 = modifyN2(decryptedN2);

oos.writeUTF(Scrambler.encrypt(ks, modifiedN2));

oos.flush();

established = true;

} else {

System.out.println();

System.out.println("Connection established!");

System.out.println();

while (!socket.isOutputShutdown()) {

if (br.ready()) {

Thread.sleep(1000);

String clientCommand = br.readLine();

oos.writeUTF(clientCommand);

oos.flush();

System.out.println("A sent message " + clientCommand + " to B.");

Thread.sleep(1000);

if (clientCommand.equalsIgnoreCase("quit")) {

System.out.println("A kill connections");

Thread.sleep(2000);

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

String in = ois.readUTF();

System.out.println(in);

break;

}

System.out.println("Client sent message & start waiting for data from B...");

Thread.sleep(2000);

String in = ois.readUTF();

System.out.println(in);

}

}

}

}

System.out.println("Closing connections & channels on clientSide - DONE.");

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**Class B:**

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.net.ServerSocket;

import java.net.Socket;

public class B {

private static final String N2 = String.valueOf((int) (Math.random() \* 1000000));

static void parseReplyFromAModifiedN2(String ks, String modifiedN2) {

modifiedN2 = Scrambler.decrypt(ks, modifiedN2);

String receivedN2 = new BigInteger(modifiedN2).subtract(BigInteger.ONE).toString();

if(receivedN2.equals(N2)){

System.out.println("Received N2 is equal to real N2");

}

}

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

System.out.println("B is started!");

boolean established = false;

try (ServerSocket server = new ServerSocket(3346)) {

Socket client = server.accept();

System.out.print("Connection accepted.");

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

DataInputStream in = new DataInputStream(client.getInputStream());

while (!client.isClosed()) {

if (!established) {

System.out.println("Server reading from channel A");

String replyFromA3 = in.readUTF();

System.out.println("READ from A message - " + replyFromA3);

String decryptedReplyFromA3 = Scrambler.decrypt(Storage.KEY\_B, replyFromA3);

String[] arrayReplyFromA3 = decryptedReplyFromA3.split(Storage.DELIMITER);

String ks = arrayReplyFromA3[0];

String idA = arrayReplyFromA3[1];

System.out.println("KS: " + ks);

System.out.println("ID\_A: " + idA);

Storage.printLine();

System.out.println("Server B writing to channel");

out.writeUTF(Scrambler.encrypt(ks, N2));

System.out.println("B send - " + N2 + " to A - OK");

out.flush();

String replyFromAModifiedN2 = in.readUTF();

parseReplyFromAModifiedN2(ks, replyFromAModifiedN2);

established = true;

} else {

System.out.println();

System.out.println("Connection established!");

System.out.println();

while (!client.isClosed()) {

System.out.println("B reading from channel");

String entry = in.readUTF();

System.out.println("READ from A message - " + entry);

if (entry.equalsIgnoreCase("quit")) {

System.out.println("A initialize connections suicide ...");

out.writeUTF("B reply - " + entry + " - OK");

out.flush();

Thread.sleep(3000);

break;

}

out.writeUTF("B reply - " + entry + " - OK");

System.out.println("B Wrote '" + entry + "'to A.");

out.flush();

}

}

}

System.out.println("Client A disconnected");

in.close();

out.close();

client.close();

System.out.println("Closing connections & channels - DONE.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Class KDC:**

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

public class KDC {

public static void main(String[] args) {

try (ServerSocket server = new ServerSocket(3345)) {

Socket client = server.accept();

System.out.print("Connection accepted.");

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

System.out.println("DataOutputStream created");

DataInputStream in = new DataInputStream(client.getInputStream());

System.out.println("DataInputStream created");

Storage.printLine();

while (!client.isClosed()) {

System.out.println("Server reading from channel");

String clientCommand1 = in.readUTF();

System.out.println("READ from client message - " + clientCommand1);

String ks = Scrambler.generateKs();

System.out.println("Generated KS: " + ks);

Storage.printLine();

System.out.println("Server try writing to channel");

String kdsCommand2 = getKdsCommand2(ks, clientCommand1);

//

// Thread.sleep(2000);

out.writeUTF(kdsCommand2);

System.out.println("KDC -> A - " + kdsCommand2 + " - OK");

System.out.println("Server Wrote message to client.");

Storage.printLine();

out.flush();

break;

}

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

System.out.println("Closing connections & channels.");

in.close();

out.close();

client.close();

System.out.println("Closing connections & channels - DONE.");

} catch (IOException e) {

e.printStackTrace();

}

}

private static String getKdsCommand2(String ks, String clientCommand1) {

String[] array = clientCommand1.split(Storage.DELIMITER);

String idA = array[0];

String idB = array[1];

String N1 = array[2];

String messageForB = Scrambler.encrypt(Storage.KEY\_B, String.join(Storage.DELIMITER, ks, idA));

return Scrambler.encrypt(Storage.KEY\_A, String.join(Storage.DELIMITER, ks, idB, N1, messageForB));

}

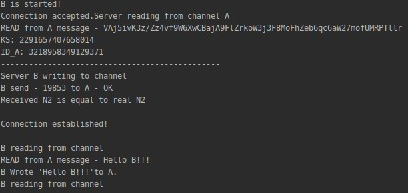
}

**Результаты работы программы:**

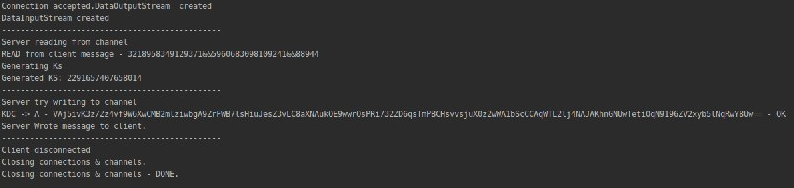
**A:**

****

**B:**

****

**KDC:**

****

**Вывод:** приобрел практические навыки работы с протоколами аутентификации.