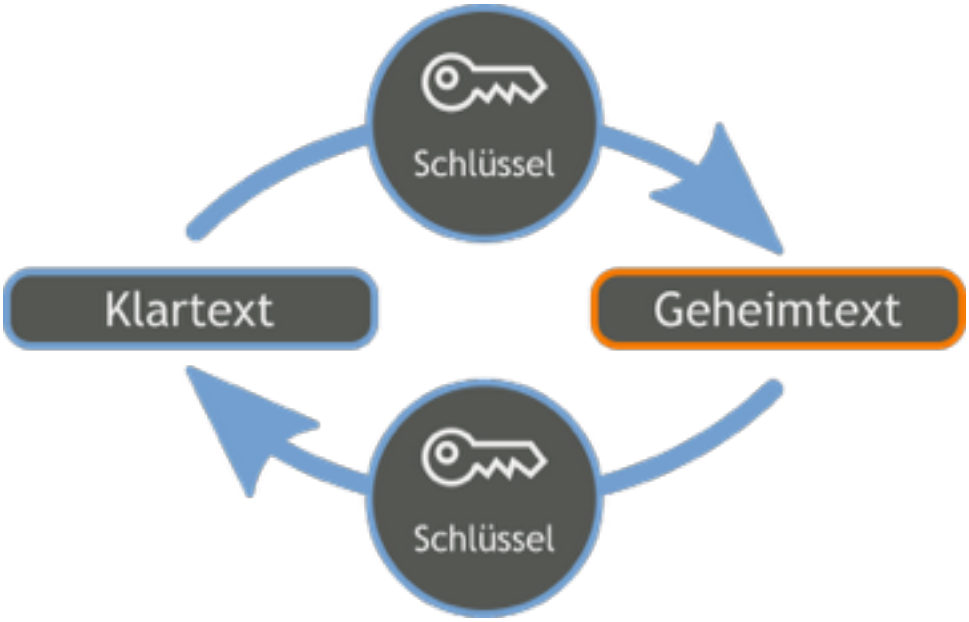
Verschlüsselung

Brunner Helmuth, Ari Ayvazyan

19 Feb 2015



Aufgabenstellung

Kommunikation [12Pkt]
Programmieren Sie eine Kommunikationsschnittstelle zwischen zwei Programmen (Sockets; Übertragung von Strings). Implementieren Sie dabei eine unsichere (plainText) und eine sichere (secure-connection) Übertragung. Bei der secure-connection sollen Sie eine hybride Übertragung nachbilden. D.h. generieren Sie auf einer Seite einen privaten sowie einen öffentlichen Schlüssel, die zur Sessionkey Generierung verwendet werden. Übertragen Sie den öffentlichen Schlüssel auf die andere Seite, wo ein gemeinsamer Schlüssel für eine synchrone Verschlüsselung erzeugt wird. Der gemeinsame Schlüssel wird mit dem öffentlichen Schlüssel verschlüsselt und übertragen. Die andere Seite kann mit Hilfe des privaten Schlüssels die Nachricht entschlüsseln und erhält den gemeinsamen Schlüssel.

Sniffer [4Pkt]
Schreiben Sie ein Sniffer-Programm (Bsp. mithilfe der jpcap-Library http://jpcap.sourceforge.net oder jNetPcap-Library http://jnetpcap.com/), welches die plainText-Übertragung abfangen und in einer Datei speichern kann. Versuchen Sie mit diesem Sniffer ebenfalls die secure-connection anzuzeigen.

Info
Gruppengröße: 2 Mitglieder
Punkte: 16

- Erzeugen von Schlüsseln: 4 Punkte
- Verschlüsselte Übertragung: 4 Punkte
- Entschlüsseln der Nachricht: 4 Punkte
- Sniffer: 4 Punkte

Arbeitsdurchführung

Im ersten Schritt haben wir eine einfach Socket- Client - Server Verbindung aufgebaut.

In weiterer Folge ist das Programm dann immer um weiter Funktionen erweitert worden.

Wir haben eine Asynchrone Verschlüsselung implementiert.

Weiters gibt es eine unverschlüsselte Übertragung, hier kann der Paketinhalt mit einem Sniffer gelesen werden.

z.B mit Wireshark.

Die Nachricht ist: “Hello, who are you?”

```
0000  02 00 00 00 45 00 00 4a  58 2c 40 00 40 06 00 00   ....E..J X,@.@...
0010  7f 00 00 01 7f 00 00 01  22 b8 d7 3d 2b 09 ac 26   ........ "..=+..&
0020  d8 67 d4 62 80 18 31 d7  fe 3e 00 00 01 01 08 0a   .g.b..1. .>......
0030  2a 39 89 c5 2a 39 89 c1  74 00 13 48 65 6c 6c 6f   *9..*9.. t..Hello
0040  2c 20 77 68 6f 20 61 72  65 20 79 6f 75 3f         , who ar e you?
```

Bei der verschlüsselten Übertragung kann die Nachricht nicht so einfach als plain Text gelesen werden.

Überblick über die verwendete Verschlüsselung

Der Server erzeugt einen Private und Public - Key. Den Public Key übermittelt der Server an den Client.

Nun erstellt der Client mithilfe des Public - Keys einen Sharedkey, diese wird dann dem Server übermittelt.

Jetzt entschlüsselt der Server den Shared Key mit dem Private Key.
Nun gibt es einen Session Key mit dem die Nachrichten verschlüsselt werden können.

Folgendes Code Snippet zeigt einen Test, wie die Verschlüsselung arbeitet:

```
//Server
//The server generates a private + public key
AsyncKeyCommunication serverAsyncKey = new AsyncKeyCommunication();
//The public key ( serverAsyncKey.getPublicKey() )is sent to the client
PublicKey transmittedPublicKey = serverAsyncKey.getPublicKey();

//Client
//The client generates a sharedKey that will be encrypted and sent to the
server
AsyncKeyCommunication clientAsyncKey = new
AsyncKeyCommunication(transmittedPublicKey);
SharedKeyCommunication clientSharedKeyCommunication = new
SharedKeyCommunication();
//The client encrypts the shared key with the public key
byte[] clientSharedKey = clientSharedKeyCommunication.getKey();
byte[] encryptedSharedKey = clientAsyncKey.encrypt(clientSharedKey);
//The client transmits the encrypted shared key to the server
byte[] transmittedEncryptedSharedKey = encryptedSharedKey;

//Server
//The server decrypts the encrypted shared key, using its private key
byte[] decryptedSharedKey =
serverAsyncKey.decrypt(transmittedEncryptedSharedKey);
//The server uses the decrypted shared key for further encrypted communication
final SharedKeyCommunication serverSharedKeyCommunication = new
SharedKeyCommunication(decryptedSharedKey);

//Communication – Everything is now set up
//
//Server
String messageFromServer = "Hey There!";
//The server encrypts the message using the shared key
byte[] encryptedMsgFromServer =
serverSharedKeyCommunication.encrypt(messageFromServer.getBytes());
//The server transmits this message to the client
byte[] transmittedMsgFromServer = encryptedMsgFromServer;

//Client
//The client decrypts the message
byte[] decryptedMsgFromServer =
clientSharedKeyCommunication.decrypt(transmittedMsgFromServer);
String resultingMessage=new String(decryptedMsgFromServer);
```

```
//Final check
Assert.assertTrue(messageFromServer.equals(resultingMessage));
```

Ausführung

Dies ist die Konsolen Ausgabe der Übertragung:

[INFO] 08:56:50 ConnectServer:51 - Message: Hello, who are you?
[INFO] 08:56:50 ConnectServer:56 - Sender Start

[INFO] 08:56:50 ConnectClient:44 - Receiver Start

[INFO] 08:56:50 ConnectServer:76 - Public key sent

[INFO] 08:56:50 ConnectClient:58 - Publickey recived
[INFO] 08:56:50 ConnectClient:74 - Send the response

[INFO] 08:56:50 ConnectServer:96 - SharedKey recevied
[INFO] 08:56:50 ConnectServer:102 - Message will be send
[INFO] 08:56:51 ConnectServer:114 - Uncrpyted Message send: Hello, who are you?
[INFO] 08:56:51 ConnectServer:115 - Encrpyted Message send: v)`LnPsWgm"m}

[INFO] 08:56:51 ConnectClient:104 - Encrypted Message received: v)`LnPsWgm”m}
[INFO] 08:56:51 ConnectClient:107 - Unencrypted Message received: Hello, who are you?