Cryptography

Programmmieren eines einfachen Verschlüsselungsprogrammes

Janeczek, Mair TGM 5AHITT

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc410051403)

[DezSys06 - Verschlüsselung 2](#_Toc410051404)

[Anforderungsanalyse 3](#_Toc410051405)

[Designüberlegung 4](#_Toc410051406)

[Technologiebeschreibung 5](#_Toc410051407)

[Sockets 5](#_Toc410051408)

[Verschlüsslungsmethoden 5](#_Toc410051409)

[AES 5](#_Toc410051410)

[DES 5](#_Toc410051411)

[RSA 5](#_Toc410051412)

[Sniffer 5](#_Toc410051413)

[Aufwandsabschätzung und Arbeitszeitaufzeichnung 6](#_Toc410051414)

[Arbeitsdurchführung 7](#_Toc410051415)

[Testbericht 8](#_Toc410051416)

[Conclusio 9](#_Toc410051417)

[Quellenangabe 10](#_Toc410051418)

# Aufgabenstellung

## DezSys06 - Verschlüsselung

**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](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

# Anforderungsanalyse

* **Es werden zwei Akteure (Client-Server) für die sinnvolle Kommunikation benötigt.**
* **Die Kommunikation muss mittels Ports ermöglicht werden.**
* **Der Kommunikationsfluss muss mittels eines Sniffers bewiesen werden können.**
* **Der übertragene Text muss mittels einer Verschlüsselungsmethode (AES), vor dem übertragen, verschlüsselt worden sein.**

# Designüberlegung

Wir haben 3 Akteure:

* Client
* Server
* Sniffer

Diese Akteure werden alle Statisch von einer Hauptklasse aufgerufen. Diese Akteure öffnen einen Socket und Kommunizieren über diesen. Der Server lässt einen privateKey und einen publicKey generieren und schickt seinen publicKey zu den Client. Dieser erstellt einen sharedKey und verschlüsselt jenen mit dem publicKey, um ihn danach wieder zurückzuschicken, wo er wieder entschlüsselt wird. Dabei ist die Art und Wiese wie der Key verschlüsselt und erstellt wird mittels eines Strategy-Patterns dynamisch definiert.

# Technologiebeschreibung

## Sockets

## Verschlüsslungsmethoden

### AES

### DES

### RSA

## Sniffer

# Aufwandsabschätzung und Arbeitszeitaufzeichnung

**Janeczek:**

**Some Excel-table**

**Mair:**

**Some Excel-table**

# Arbeitsdurchführung

Wir haben begonnen indem wir ein Designkonzept in Form eines UML-Diagrammes erstellt haben. Allerdings mit der Intention es nicht zu dynamisch zu gestalten, um es nicht unnötig groß zu gestalten (Wir wollen immerhin rechtzeitig fertig werden). Dabei haben wir beim Design mit dem nötigsten angefangen und dann ein paar dynamische Erweiterungen hinzugefügt. Nachdem wir den ersten Ansatz unseres Designs erstellt haben, haben wir die Aufgaben fürs erste geteilt. Einer schrieb die Dokumentation während der andere sich mit dem Source Code beschäftigt. …

# Testbericht

# Conclusio

# Quellenangabe

[1] **Title,** Autor**,** Link