08 – Secure Sockets

//// Design of Distributed Systems
//

Dipl.-Inf. André Langer

*VSR.Informatik.*TU-Chemnitz.*de*

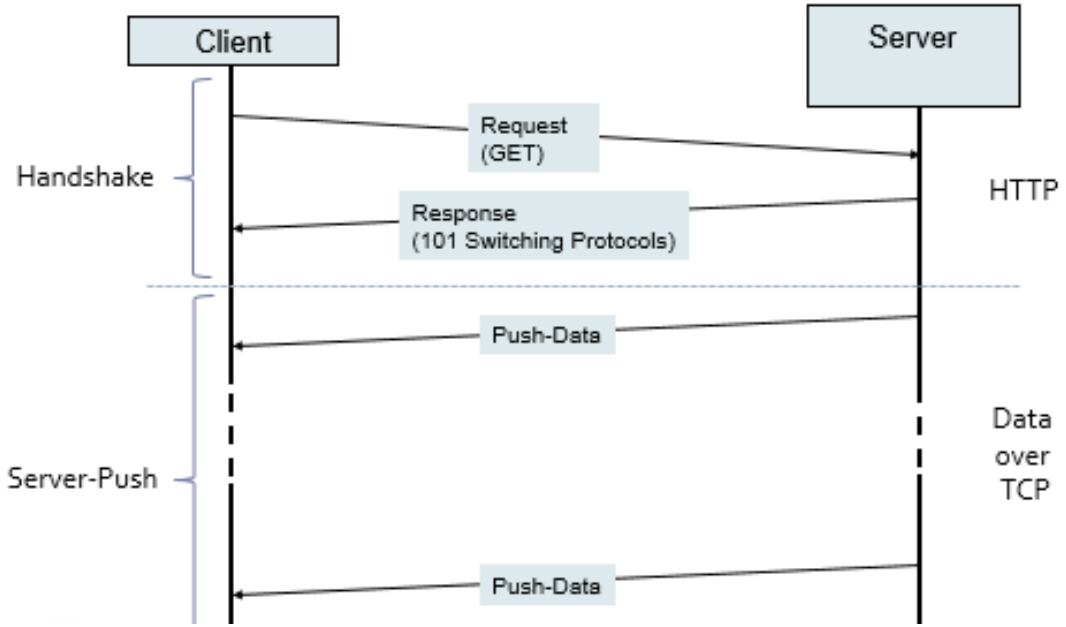
1 Repetition

We were looking for a solution so that a server application can actively send messages to a specific client at any time

→ WebSockets

- A specialized **WebSocket Service** runs on server-side
- The Client initializes with a **HTTP request** a connection with the Server
- After the server response, the **TCP connection is not closed** but remains open as persistent and full-duplex
- The client and the server can exchange further messages via the established TCP connection by using an **arbitrary appliation layer communication protocol**
- The client **browser API** has to support WebSocket Communication

WebSocket Principle



? So what's the big deal?

A not-closed TCP connection 😐

Sounds similar to HTTP Keep-alive

NO!

Advantages

- Server can actively use the connection (bi-directional)
- No HTTP overhead
- No delay due to polling
- Supported by many Web browsers
  Example: Google Chrome (JavaScript):

- ```
  //Socket öffnen und Daten empfangen
  var s = new WebSocket(host);
  s.onmessage = function (e) {…};
  …
  //Daten senden
  var xxx = inputBox.value;
  s.send(xxx);
  ```
  ```

Disadvantages?

- Client still has to initialize the connection
- Complex protocol
- We need a specialized service on server-side

- So is there any alternative?

2 SSE

SSE Advantages

- Operate directly via HTTP without any special implementation
- Directly handled by the web browser
  - Content-Type: text/event-stream
- Re-connect possibilities
- Arbitrary events

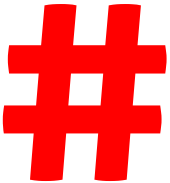
You should be able to explain the differences between

- AJAX Polling
- Long Polling / „Comet“
- Server-Sent Events
- WebSockets

3 Secure Sockets

What is SSL / TLS

So far, we normally send application data as plain text

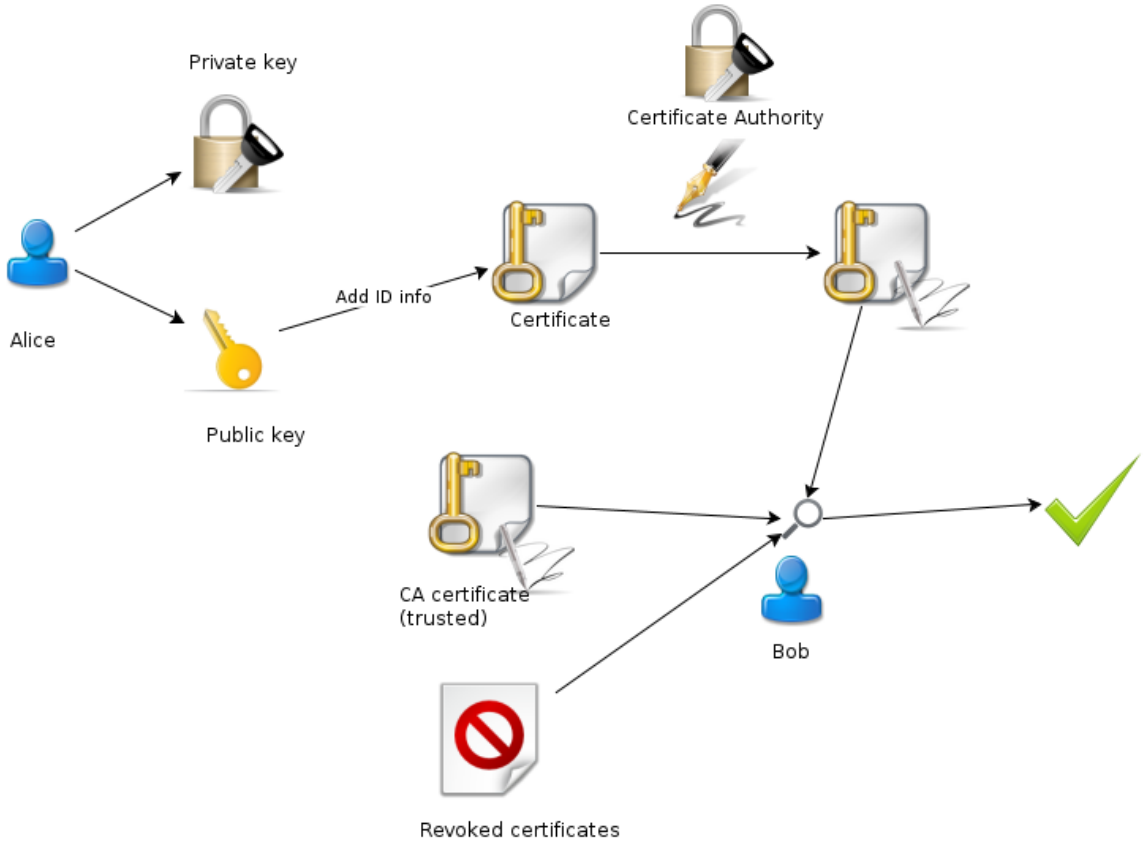


We want to establish an encrypted link between a server and a client

So we need a communication protocol that specifies how to secure the channel and encrypt our data

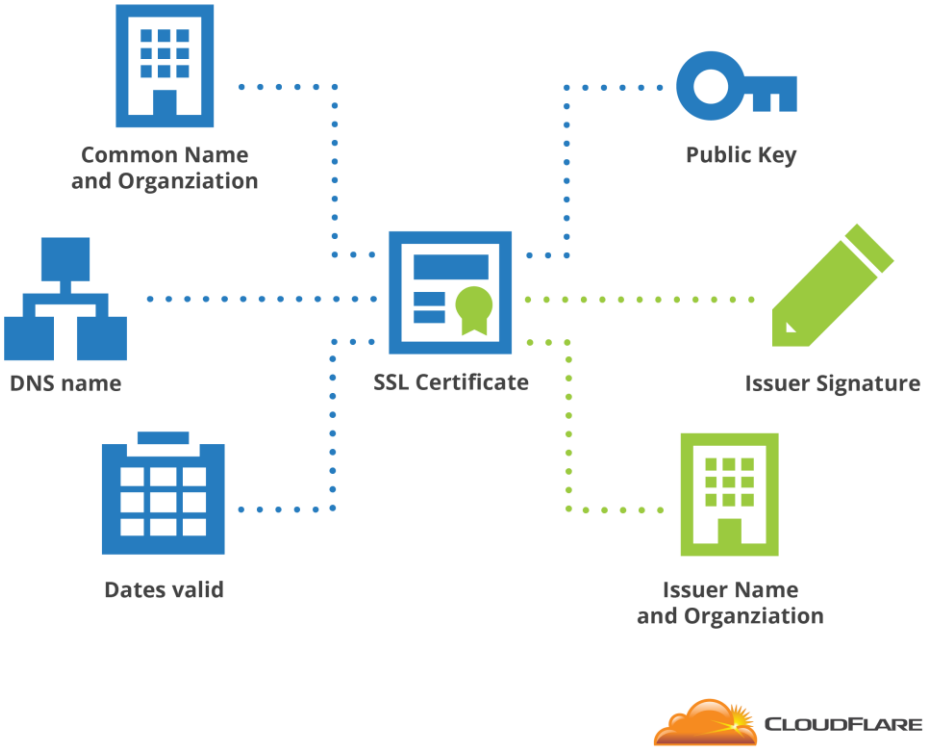
SSL/TLS Overview

- Secure Sockets Layer (SSL)
  - Version 1.0 by Netscape Communications (1994)
- Transport Layer Security (TLS)
  - IETF-standard from the year 1999 (RFC 2246)
- Network protocol for secure data transfer
- Since Version 3.0 SSL is being further developed under the name TLS
  - Minor differences between SSL 3.0 & TLS 1.0
  - TLS 1.0 is presented as SSL 3.1
  - Currently TLS 1.2



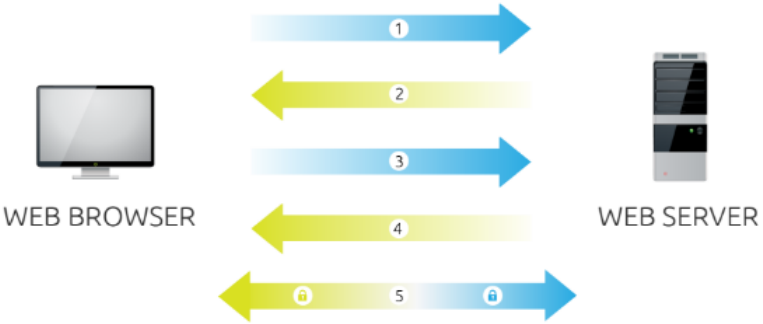
Source: http://swift.siphos.be/aglara/images/04-ca_certificate.png

The anatomy of a certificate



Source: https://blog.cloudflare.com/content/images/2015/06/illustrations-ssl-blog-june-2015-02.png

SSL/TLS Overview



1. **Browser** connects to a web server (website) secured with SSL (https). Browser requests that the server identify itself.
2. **Server** sends a copy of its SSL Certificate, including the server's public key.
3. **Browser** checks the certificate root against a list of trusted CAs and that the certificate is unexpired, unrevoked, and that its common name is valid for the website that it is connecting to. If the browser trusts the certificate, it creates, encrypts, and sends back a symmetric session key using the server's public key.
4. **Server** decrypts the symmetric session key using its private key and sends back an acknowledgement encrypted with the session key to start the encrypted session.
5. **Server** and **Browser** now encrypt all transmitted data with the session key.

UPPER LAYERS	7	**Application Layer** ✓ Message format, Human-Machine Interfaces
	6	**Presentation Layer** ✓ Coding into 1s and 0s; encryption, compression
	5	**Session Layer** ✓ Authentication, permissions, session restoration
TRANSPORT SERVICE	4	**Transport Layer** ✓ End-to-end error control
	3	**Network Layer** ✓ Network addressing; routing or switching
	2	**Data Link Layer** ✓ Error detection, flow control on physical link
	1	**Physical Layer** ✓ Bit stream: physical medium, method of representing bits

Source: http://nhprice.com/wp-content/uploads/2013/03/1-tutorial-osi-7-layer-model1.gif

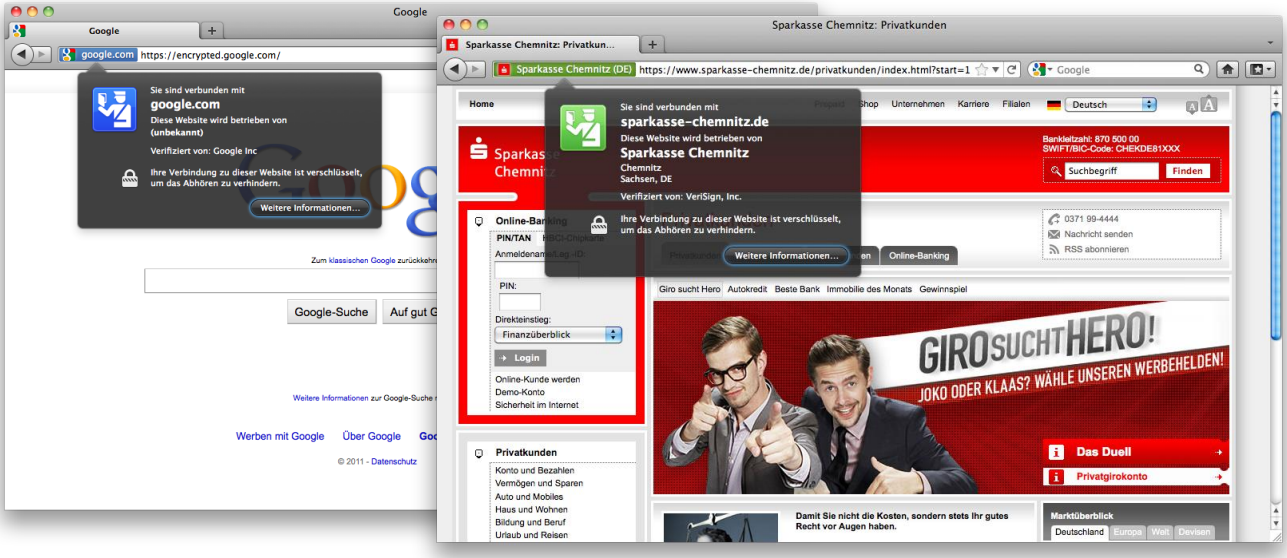
SSL/TLS Architecture

- In OSI-model in layer 6
- In TCP/IP-model
  - Above the Transport layer (i.e. TCP,...)
  - Below the Application layer (i.e. HTTP,...)
- Basic idea: generic security layer
- Protocol consists of 2 layers:

Handshake Protocol	Change Cipher Spec Protocol	Alert Protocol	Application Data Protocol
Record Protocol			

Hypertext Transfer Protocol Secure

- HTTP with additional transmission encryption by SSL/TLS
- Standard-Port: 443



4 WSS

WSS

- WebSockets over SSL/TLS
- Prefer wss:// over ws://
- Protects against man-in-the-middle attacks

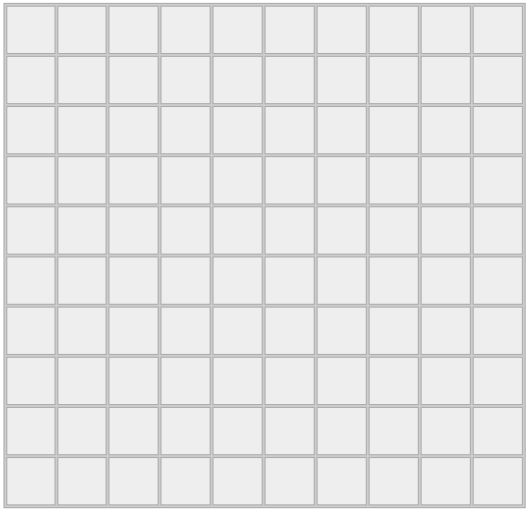
However, this only secures the connection channel, not the transmitted data

WSS Best Practises

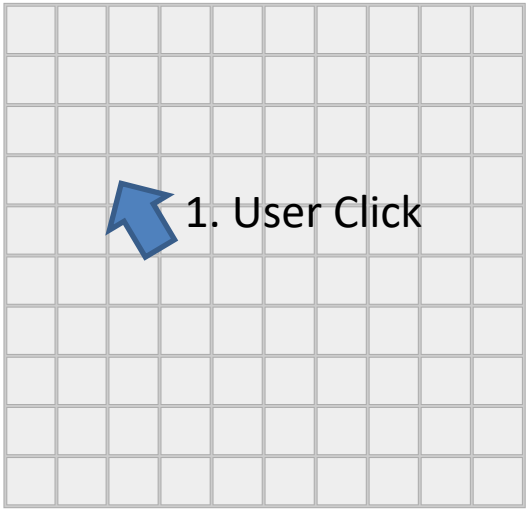
- Validate client input
- WS Sub-Protocol Inspection
- Validate server data
- Authentication / Authorization
- Check Origin Header

5 The Click Game

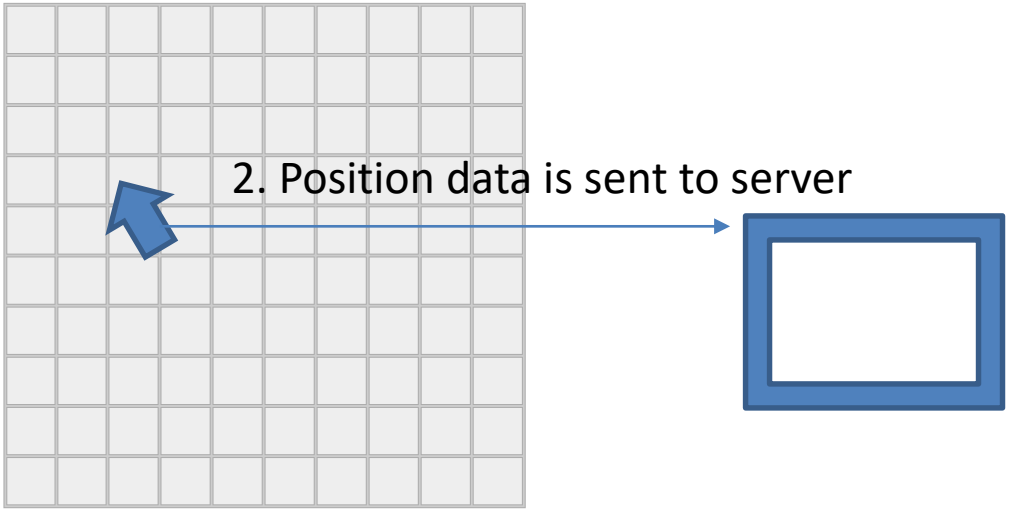
Basic Scenario



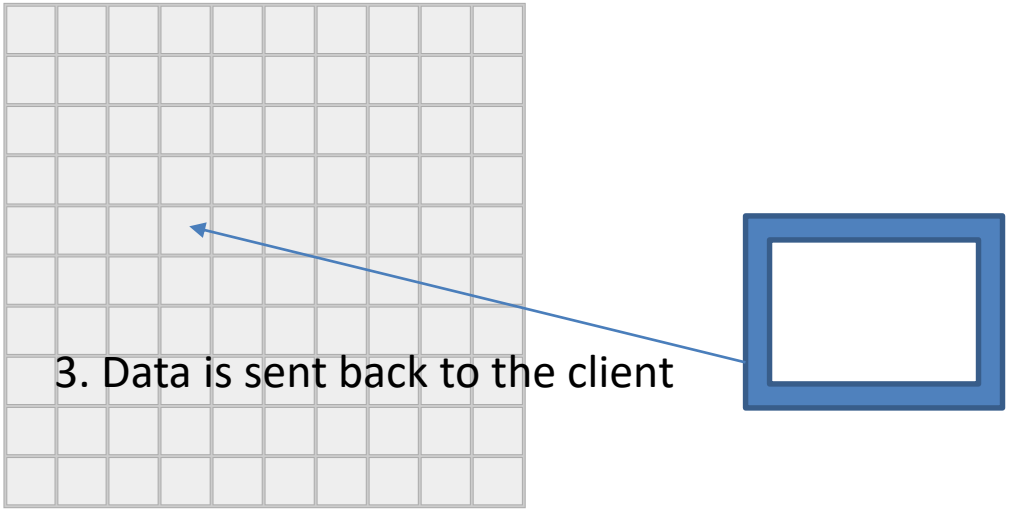
Basic Scenario



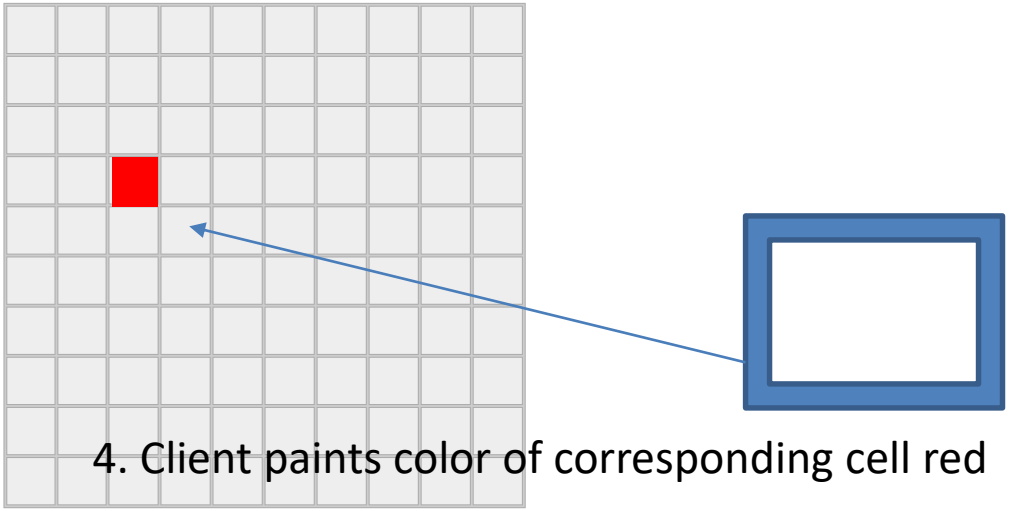
Basic Scenario



Basic Scenario



Basic Scenario



Simple Task:

- Implement the requirement using AJAX and traditional server side technologies

Advanced Task:

- Implement the requirement using WebSockets

6 ToDo

Next week, we will talk about server-side development

! Please bring a **laptop** with you and install Microsoft **Visual Studio** Community 2015 in advance

VSR

Thank You!

André Langer@informatik.tu-chemnitz.de

VSR.*Informatik*.TU-Chemnitz.*de*