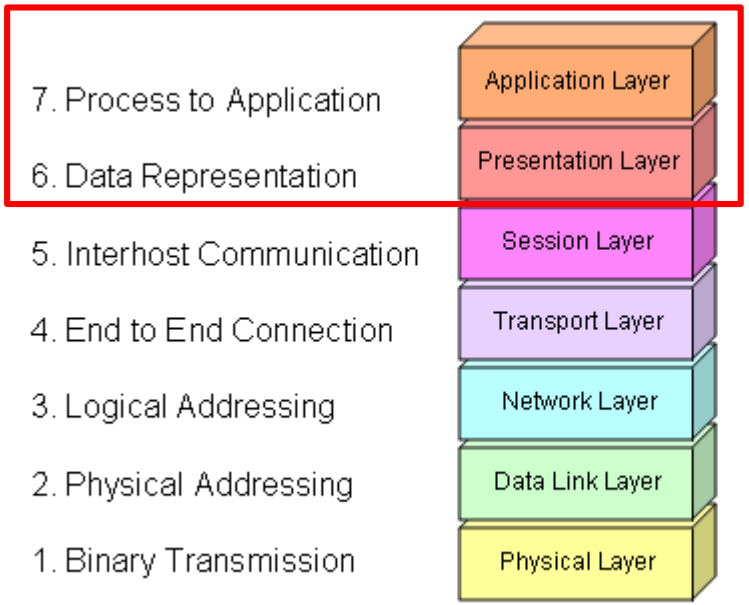
# IT-SICHERHEIT UND DATENSCHUTZ

## KAPITEL 6 - WEB-SICHERHEIT

buchmann@hft-leipzig.de

# LERNZIEL UND AUFBAU DIESES KAPITELS

- Internet-Anwendungen
  - allgemeiner Ablauf beim HTTP-Zugriff
- Angriffe
  - auf den Browser, Benutzer oder Server
- Kryptographische Protokolle
  - Schlüsselaustausch, Authentifizierung über Zertifikate, Signaturen




- Lernziele
  - Sie wissen, welche unterschiedlichen Angriffsziele auf Anwendungen mit Internet-Anbindung existieren.
  - Sie können erläutern, wieso dies praktisch immer etwas mit der Authentifizierung von Rechnern oder Personen zu tun hat.
  - Sie können Lösungsmöglichkeiten auf der Basis von kryptographischen Protokollen, Zertifikaten oder Signaturen bewerten und einsetzen.

# ZIELE DER IT-SICHERHEIT


- „Security triad“
  - **Vertraulichkeit**: Asset ist nur Autorisierten zugänglich
  - **Integrität:** Asset kann nur von Autorisierten modifiziert werden
  - **Verfügbarkeit**: Asset kann von Autorisierten genutzt werden

- ISO 7498-2 fügt hinzu
  - **Authentisierung**: Identität eines Senders wird überwacht
  - **Nichtabstreitbarkeit**: Sender kann nicht abstreiten, das eine Nachricht von ihm kam
- US Department of Defense fügt hinzu
  - **Auditierbarkeit**: Alle Aktionen mit dem Asset sind nachvollziehbar

# WEB-ANWENDUNGEN

# WEBBROWSER SIND EINFACH

- Sende einen Request an den Webserver
- Zeige das Ergebnis auf dem Bildschirm
- Verarbeite Links, Buttons, Textfelder, eingebettete Medien entsprechend
- Speichere Bookmarks und Cookies als Annehmlichkeit für den Nutzer

# WEBBROWSER SIND KOMPLIZIERT

- Herausforderungen für den Entwickler
  - Zahllose Plugins und Addons
  - Passive und aktive Inhalte in unterschiedlichen Sprachen (ActiveX, JavaScript,…) und Dialekten (HTML 1.0, XML, XHTML, HTML5,...)
  - Unterschiedliche Browser-, Betriebssytem und Plugin-Versionen
  -
- Herausforderungen für die IT-Sicherheit
  - Private Informationen aus Online-Banking, Beruf, Schule, Hobby
  - Browser kommuniziert über öffentliches Netz, darf auf jede Hardware und jede Ressource im Rechner mit den Rechten des Nutzers zugreifen
  - Mittlerweile bindet eine Webseite aktive Inhalte von hunderten von dritten Servern ein (Verhaltensanalyse, Werbe-Monitoring, Profiling, ...)

# LADEN EINER WEBSEITE

Input lines are **bold**
2x line break
is important!

/home/buchmann> **telnet dbis.ipd.uni-karlsruhe.de 80**
*Trying 129.13.182.137...*
*Connected to dbis.ipd.uni-karlsruhe.de.*
*Escape character is '^]'.*

request

**GET / HTTP/1.0**
**Referer: http://localhost/index.html**
**User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.1)  Firefox/2.0.0.1**
**Host: localhost**
**Accept: text/html, image/gif, image/jpeg, image/pjpeg, */***
**Accept-Language: en, de**

*(here follows the response from the server)*

# ANTWORT VOM WEBSERVER

Input lines are **bold**
2x line break
is important!

/home/buchmann> **telnet dbis.ipd.uni-karlsruhe.de 80**
*Trying 129.13.182.137...*
*Connected to dbis.ipd.uni-karlsruhe.de.*
*Escape character is '^]'.*
**GET / HTTP/1.0**

response

HTTP/1.1 200 OK
Date: 11 April 2018 08:46:39 GMT
Server: Apache/2.2.8 (Unix) DAV/2 mod_ssl/2.2.8 OpenSSL/0.9.8h PHP/5.2.6
X-Powered-By: PHP/5.2.6
Set-Cookie: PHPSESSID=74e6a1911499e9578534f31814c357d4; path=/
Expires: 19 Nov 2018 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" <…>
</html>*Connection closed by foreign host.*

# IM FOLGENDEN

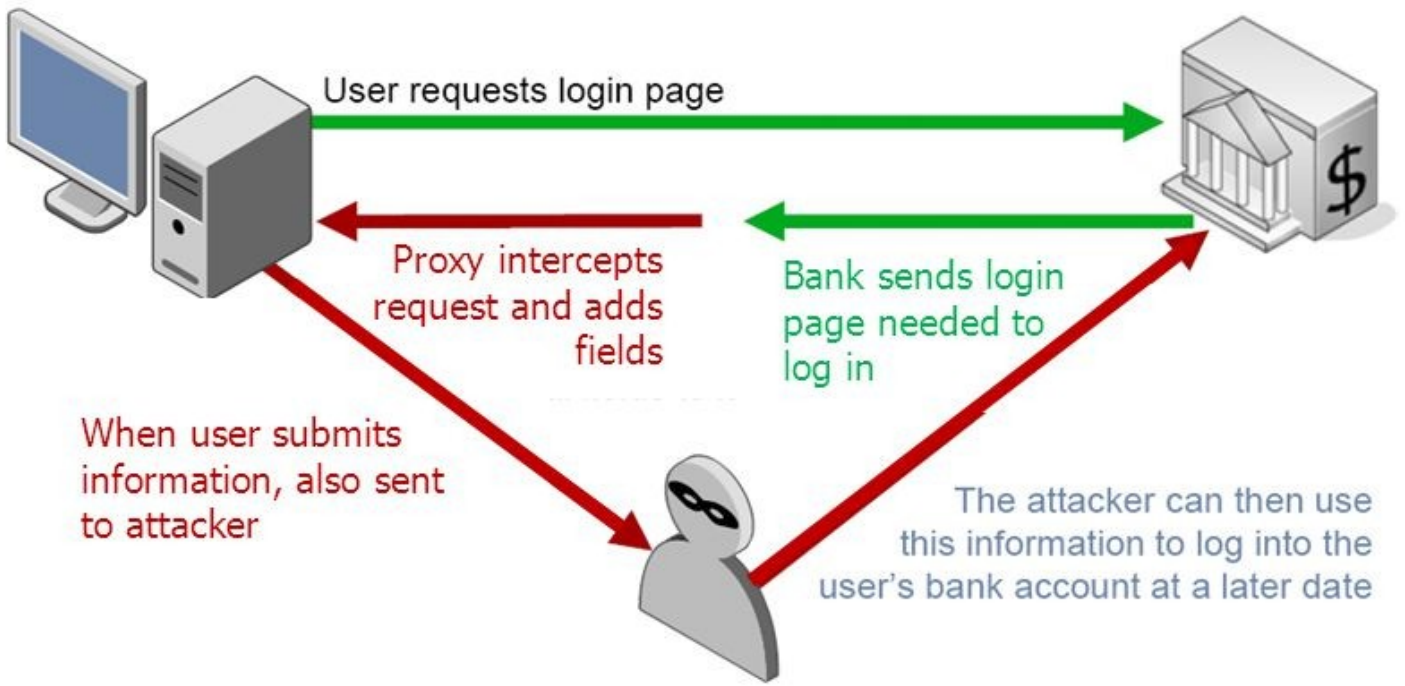
- Angriffe auf den Browser
  - Den Browser dazu bringen, das er etwas tut das er nicht sollte
- Angriffe auf den Nutzer
  - Den Nutzer dazu bringen, das er etwas tut das er nicht wollte
- Angriffe auf den Webserver
  - Den Server dazu bringen, das er etwas tut das er nicht sollte

# ANGRIFFE AUF DEN BROWSER

- Der Browser tut etwas, das er nicht soll

# MAN-IN-THE-BROWSER

- Browser von Schadcode infiziert, der die durchlaufenden Daten abfängt
  - Kann alles, was der Browser darf/Browser darf alles, was Benutzer darf
- Anmerkung: eine SSL-Verschlüsselung bzw. TLS (https) passiert im Browser
  - Daten abgefangen/manipuliert, bevor sie verschlüsselt werden



# BEISPIEL: TROJAN.SILENTBANKER

- 2008 entdeckt, erkennt >400 Banken-Webseiten

- Kann folgendes tun
  - Legitime Onlinebanking-Requests an Server des Angreifers umleiten
  - im Browser HTML-Quelltexte betrachteter Webseiten verändern
  - Onlinebanking-Requests des Nutzers verändern, bevor sie die Bank erreichen
  - Antworten des Onlinebanking-Servers an den Angreifer senden
  - Sich selber automatisch updaten
  - Nutzernamen, Passwörter und Inhalt der Zwischenablage aufzeichnen
  - Cookies und Zertifikate kopieren und versenden
  - Verzeichnisse der installierten Software außerhalb des Browsers erstellen und an den Angreifer versenden

# PAGE-IN-THE-MIDDLE

- Nutzer wird auf den Server des Angreifers umgeleitet
  - Angreifer-Server spielt dem Nutzer die angeforderte Seite vor, spielt der angeforderten Seite den Nutzer vor

# ZAHLLOSE VARIANTEN

- Email-Phishing (kennt jeder)
- DNS Spoofing, Router cache poisoning
  (Router ordnet falsche IP-Adresse einem DNS-Eintrag zu)
- DNS Injection
  (DNS-Server ordnet einer IP-Adresse einen falschen DNS-Eintrag zu)
- URL Redirection
  (CSS, JavaScript, Sicherheitslücken im Browser bringen diesen dazu, die falsche Seite anzuzeigen)
- Proxy-Angriffe
  (Cache-Zwischenspeicher im Web-Proxy enthält die falsche Seite)

# DOWNLOAD SUBSTITUTION

- Nutzer möchte Software herunterladen, erhält aber statt der Datei vom ausgewählten Server eine manipulierte Datei vom Angreifer
  - Angriff ist ähnlich zu Page-in-the-Middle
  - Großes Plus für den Angreifer: Nutzer möchte tatsächlich Software installieren und wird darum alle Sicherheitswarnungen wegclicken

# WARUM FUNKTIONIEREN ANGRIFFE AUF DEN BROWSER?

- Probleme bei **Identifizierung und Authentifizierung von Unbekannten**
  - Von welcher Person kommt eine Nachricht/Webseite wirklich?
  - Von welchem Rechner kommt eine Nachricht/Webseite wirklich?

- Problematisch, weil
  - Usability, User Experience im Internet sind wichtig!
    - Lieber unsicheres Online-Banking als nervige Authentifizierung?
  - Jede Rechner-zu-Rechner-Authentifizierung ist notwendigerweie begrenzt auf das Wissen der Rechner
  - Schadsoftware sitzt genau da, wo die Authentifizierung stattfindet
  - Beide Seiten einer Kommunikationsbeziehung sicher authentifizieren?
    - Bsp. Onlinebanking: 2-Faktor-Authentifizierung des Nutzers gegenüber Bank durch Pin und Handy/QR-Code, aber Authentifizierung der Bank gegenüber Nutzer nur über mickriges SSL-Zertifikat

# ANGRIFFE AUF DEN NUTZER

- Der Nutzer tut etwas, das er nicht will

# FALSCHER ODER IRREFÜHRENDER INHALT AUF WEBSEITEN

- Website Defacement
  - Angreifer ersetzt den regulären Inhalt einer Seite
- Fake-Webseite
  - Angreifer generiert eine gefälschte Webseite, die aber legitim aussieht, z.B. wie die Seite einer vertrauenswürdigen Regierungsorganisation
- Fake-Code
  - Angreifer platziert manipulierte Software-Downloads auf einer vertrauenswürdig aussehenden Seite
  - Schönstes Beispiel: „Sunburst“-Angriff auf SolarWinds (Update-Server von SolarWinds hatte Passwort „solarwinds123“, Angreifer manipulierte bereitgestellte Updates)

# SCHAD-/FREMDCODE AUF WEBSEITEN

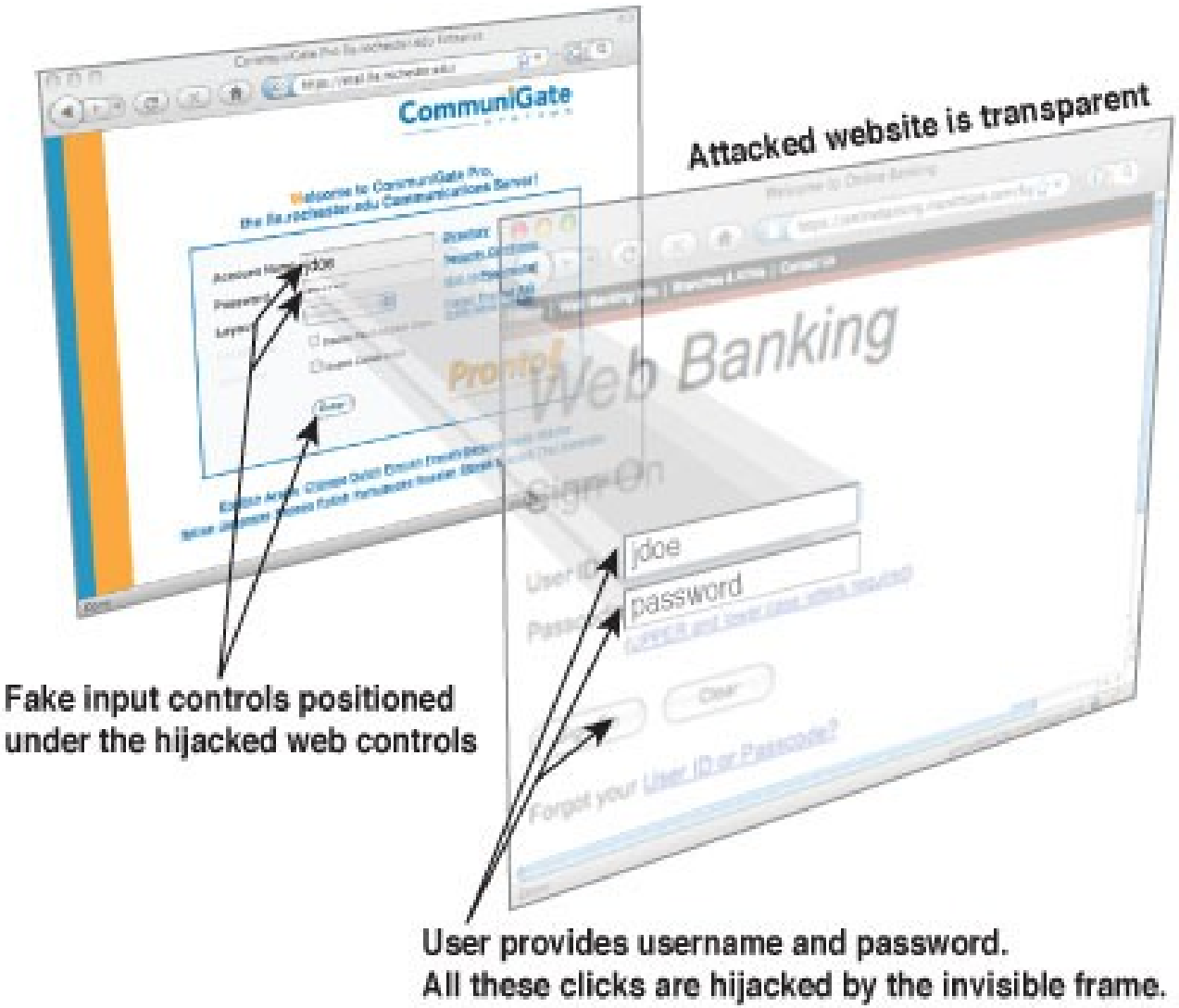
- Ersetze die Inhalte legitimer Webseiten
  - Möglichkeit, den Browser anzugreifen
    (durch Viren, Trojaner, ggf. durch vorspiegeln seriöser Downloads)

- Inhalte, die den Nutzer ausspähen
  - Web bugs, Analyse-Toolkits etc.,
    *mehr dazu im Datenschutzteil*

- Inhalt ist nicht das, was er zu sein scheint
  - Clickjacking

# CLICKJACKING

- HTML erlaubt transparente Webseiten bzw. transparente Layer über beliebigem Hintergrund
- Angreifer generiert unsichtbare, aber anklickbare Steuerelemente, die über legitimem Seiteninhalt liegen



# BEISPIEL: FACEBOOK-LIKEJACKING

## New Facebook Scam Involving Justin Bieber Punching Selena Gomez

June 13, 2011 by Tim Ollason  0 Comments

Facebook is the largest of all the Social Networking sites with more than 500 million users worldwide which is probably the reason that it's so often the victim of Facebook scams, virus and malware. Some of the recent reports we have

lol Justin Bieber Punches Selena!
lol Justin Bieber Rage Video!

(Click the PLAY Button Above To Watch The Video)

- Durch einen Klick auf den Play-Button wurde Like ausgelöst, anstelle das Video abzuspielen
- Funktioniert auf Facebook nicht mehr, aber auf anderen unvorbereiteten Webseiten, die aktive Inhalte von Dritten laden

# SCHUTZ VON WEBSEITEN

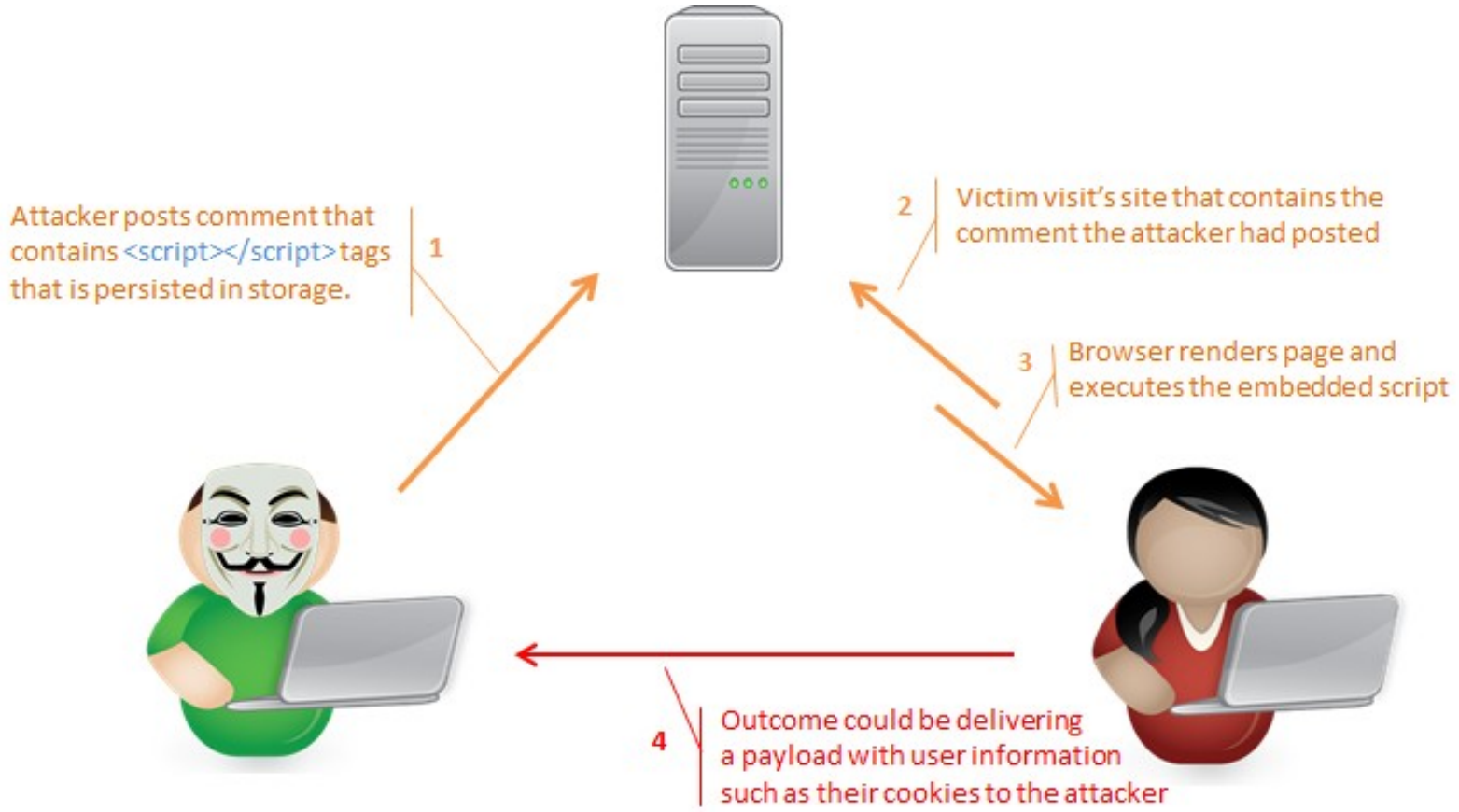
- Nutzer
  - Wenig Möglichkeiten, wenn der Server ein gültiges Zertifikat vorweist aber vom Angreifer manipuliert wurde
    *(„Seite sieht ungewohnt aus“)*
- Webseiten
  - *sollen* öffentlich sichtbar sein: nicht störende Sicherheitsmechanismen
  - Angreifer kann nicht nur Webseiten oder Downloads manipulieren, sondern auch Prüfsummen, Signaturen oder Zertifikate des Servers
    *(System so gut wie möglich absichern und SSL benutzen)*
- *Wir schauen uns später in diesem Kapitel an, wie weit der Schutz von Zertifikaten reicht und wie SSL funktioniert*

# ANGRIFFE AUF DEN WEBSERVER

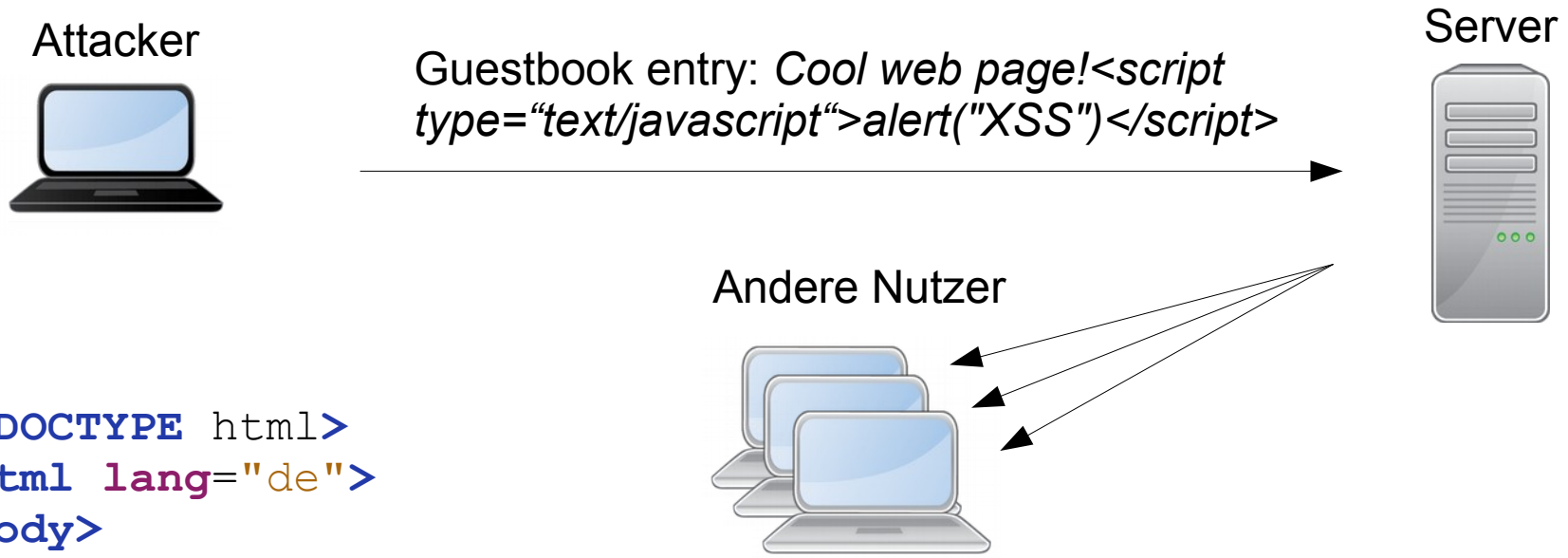
- Der Server tut etwas, das er nicht soll

# CROSS-SITE SCRIPTING

- Webseiten enthalten ausführbaren Code (JavaScript, Plugins, …) auf den der Angreifer Einfluss nehmen kann ohne den Server selbst anzugreifen

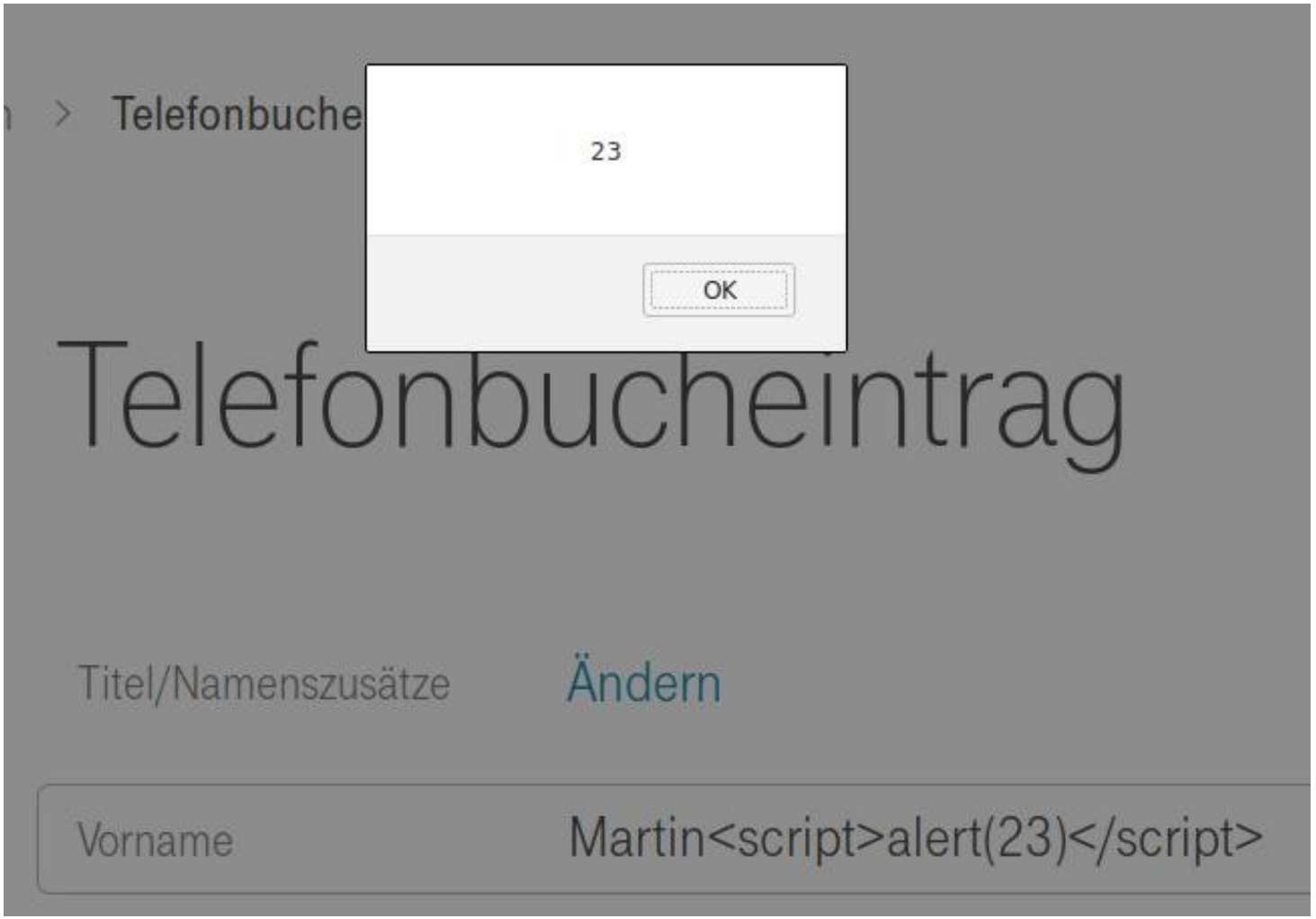


# BEISPIEL: EINFACHES GÄSTEBUCH




```
<!DOCTYPE html>
<html lang="de">
<body>
...sometext...
<h1>These are my guestbook entries</h1>
...
<ul><li>Awesome homepage!</li>
<li>Awesome homepage!</li>
...somemoretext...
<li>Awesome homepage!</li>
<li>Cool web page!<script type=?text/javascript?>alert("XSS")
 </script></li>
</ul>
</body>
```

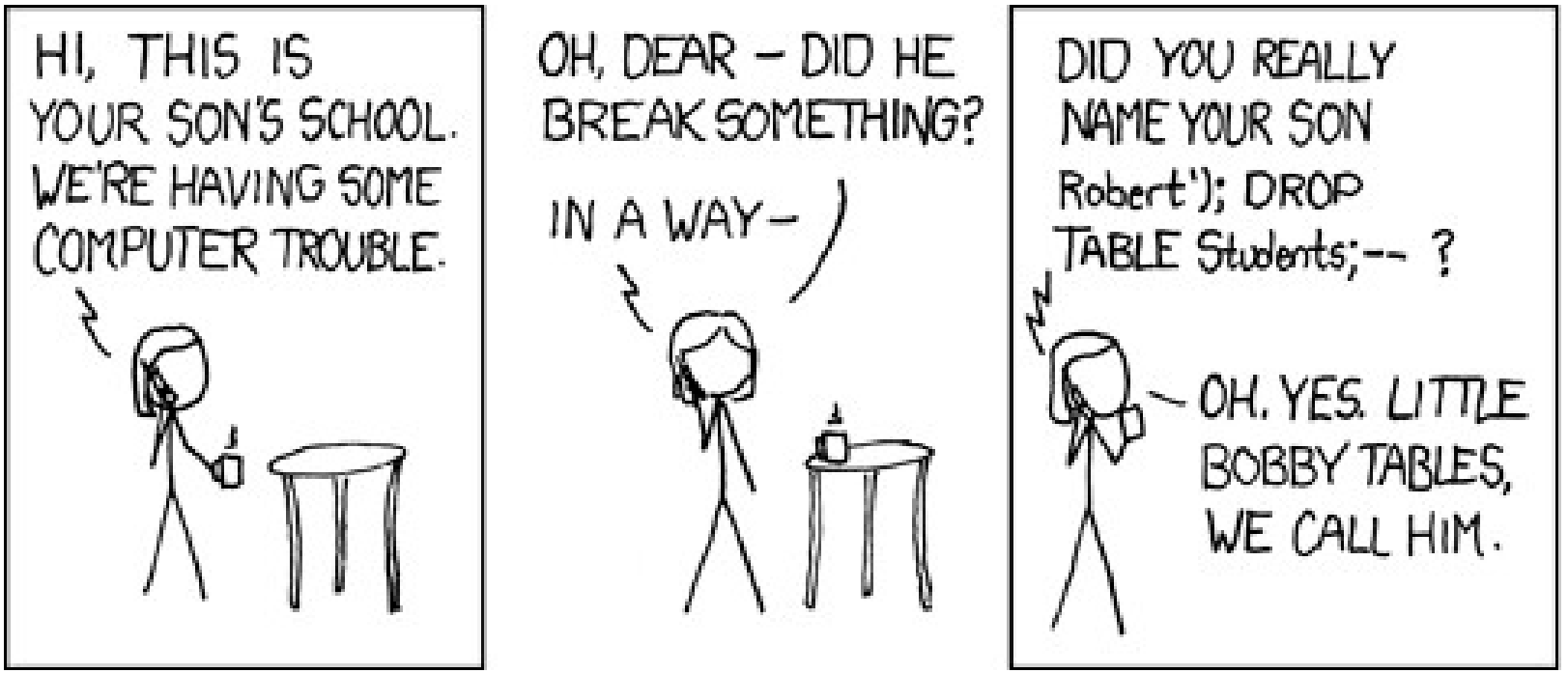
# DAS FUNKTIONIERTE NOCH AM 16. MÄRZ 2019 IM TELEKOM-KUNDENCENTER ;-)



# SQL INJECTION

- Wenn eine SQL-Datenbank im Backend verwendet wird:
  Gib SQL-Kommandos in Textfelder ein und warte was passiert
  - Angriff nutzt die so vom Betreiber geplante Interaktion

- Beispiel: Nutzer darf einen Nutzernamen in ein Textfeld eingeben



# FUNKTIONSWEISE EINER SQL INJECTION

- Typische SQL Query in der Datenbank
  ```
  SELECT * FROM customers WHERE name = 'Robert';
  ```

- Im Code im Backend steht
  ```
  SELECT * FROM customers WHERE name = '$Param';
  ```
  - *$Param* kommt vom Browser des Benutzers

- SQL injection: Browser sendet nicht
  ```
  Robert
  ```
  sondern Robert + Verknüpfung mit weiteren SQL-Befehlen
  ```
  Robert'; DELETE * FROM customers;
  ```
  beispielsweise in der URL kodiert
  ```
  http://www.victim.de/?param=Robert27%3B+DELETE+*+FROM+customers%3B
  ```

- Backend führt dann folgende SQL Query in der Datenbank aus
  ```
  SELECT * FROM customers WHERE name = 'Robert'; DELETE * FROM
  customers;
  ```

# WARUM SIND DIESE ANGRIFFE ERFOLGREICH?

- Übernahme ungeprüfter Parameter aus einer unsicheren Umgebung
  - Annahme, dass sich jeder im Internet so verhält wie vom Entwickler der Webanwendung geplant
- Angriffe dieser Art sind mit allen Systemen möglich, die Eingabeparameter aus dem Internet akzeptieren und intern Code ausführen
- Gegenmaßnahmen
  - *Immer jeglichen* Input auf Korrektheit prüfen!
  - Rechte des Webservers so weit wie möglich beschränken
    - Im Gegensatz zum Browser des Benutzers hat ein Webserver nur eine eingegrenzte Aufgabe und kann daher mit stark eingeschränkten Berechtigungen laufen

# SICHERE KOMMUNIKATION MIT UNBEKANNTEN?

- Deutlich weniger Angriffe, wenn wir wüssten mit welchem Rechner/welcher Person wir reden

# KRYPTOGRAPHISCHE PROTOKOLLE

- **Schlüsselaustausch**
- **Authentifizierung**
  - **Data origin authentication**
  - **Entity authentication**
- Secret splitting
- Secret sharing
- Time-stamping (global gültige, unverfälschte Zeitstempel)
- Key escrow (Nur berechtigte können Schlüssel wiederherstellen)
- Zero-Knowledge proofs (Besitz von Wissen nachweisen, ohne dieses Wissen preiszugeben)
- Blind signatures (Nachrichteninhalt signieren, ohne diesen Inhalt zu kennen)
- ...

# DATA ORIGIN AUTHENTICATION

**Data origin authentication** erlaubt die Prüfung, das eine Nachricht von einem bestimmten Absender stammt und unterwegs nicht verändert wurde.

- Bezug zur Kryptographie
  - Kryptographie kann Integrität sicherstellen
  - Beispiel: Alice publiziert ihren Public Key, und signiert eine Nachricht an Bob mit ihrem Private Key. Bob prüft Signatur mit Public Key von Alice.
    - Wenn Signatur (kryptographische Prüfsumme) korrekt ist, wurde die Nachricht nach dem Signieren nicht mehr verändert
    - Wenn die Signatur mit dem Public Key validiert werden konnte, musste sie jemand mit dem passenden Private Key erzeugt haben
    - Wenn der Public Key tatsächlich von Alice stammt UND Alice ihren Private Key geheim gehalten hat, war Alice die Absenderin der Nachricht

# ENTITY AUTHENTICATION

**Entity authentication** erlaubt die Prüfung der behaupteten Identität des Absender einer Nachricht

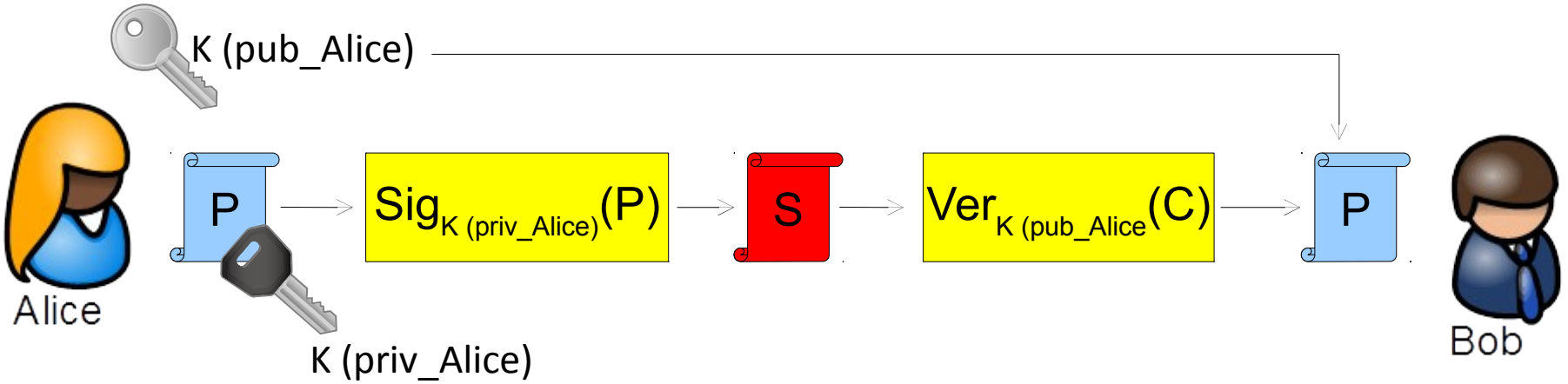
- Sie wissen bereits, wie direkte Authentifizierung funktioniert (Prüfung von *Wissen, Besitz, persönliche Merkmale*)

- Bezug zur Kryptographie
  - Alice gegenüber Bob und Bob gegenüber Alice über einen unsicheren Kanal authentifizieren
  - Herausforderungen
    - Alice und Bob sind sich nie begegnet, kein Shared Secret vereinbart
    - Alice muss sicher sein, dass eine Nachricht von Bob kommt und nicht von Carol, die eine Kopie einer alten Nachricht von Bob einschleust (Replay-Angriff)
    - Bob muss sicher sein, dass er direkt mit Alice kommuniziert und nicht mit Dave, der Bob gegenüber so tut als wäre er Alice und Alice gegenüber so als wäre er Bob. (Man-in-the-Middle-Angriff)

# BASIS: ELEKTRONISCHE SIGNATUREN

- Nachweis, das Nachricht unverändert von einer Quelle kommt
  - Authentizität und Integrität der Nachricht,
    mit Verschlüsselung kombiniert auch Vertraulichkeit

- Einfacher Ansatz: Reverse Public Key-Infrastruktur
  - Alice signiert eine Nachricht mit ihrem Private Key
  - Bob verifiziert die Nachricht mit Alice's Public Key



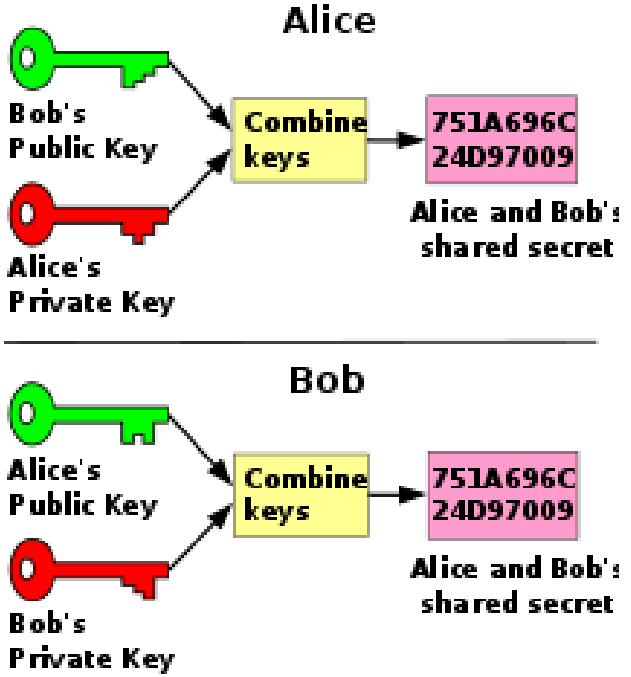
# ANFORDERUNGEN AN „GUTE“ SIGNATUREN

- Elektronische Signatur ersetzt handschriftliche Unterschrift
  → dieselben Anforderungen wie an eine gerichtsfeste Unterschrift

  - Signatur identifiziert Unterzeichner
  - Signatur bestätigt Authentizität des Dokuments
  - Mit Signatur erklärt Unterzeichner, das Text gültig und vollständig ist
  - Signatur ist eindeutig dem signierten Dokument zugeordnet
  - Signatur ist fälschungsssicher
  - Signatur kann nicht später zurückgezogen oder abgestritten werden

- Auch hier: **Noch keine Authentifizierung**!
  - Analoges Beipspiel: Wenn jemand für Sie auf Ihrem Personalausweis unterschreibt, wird dessen Unterschrift für die echte gehalten, nicht Ihre

# BASIS: SCHLÜSSELAUSTAUSCH

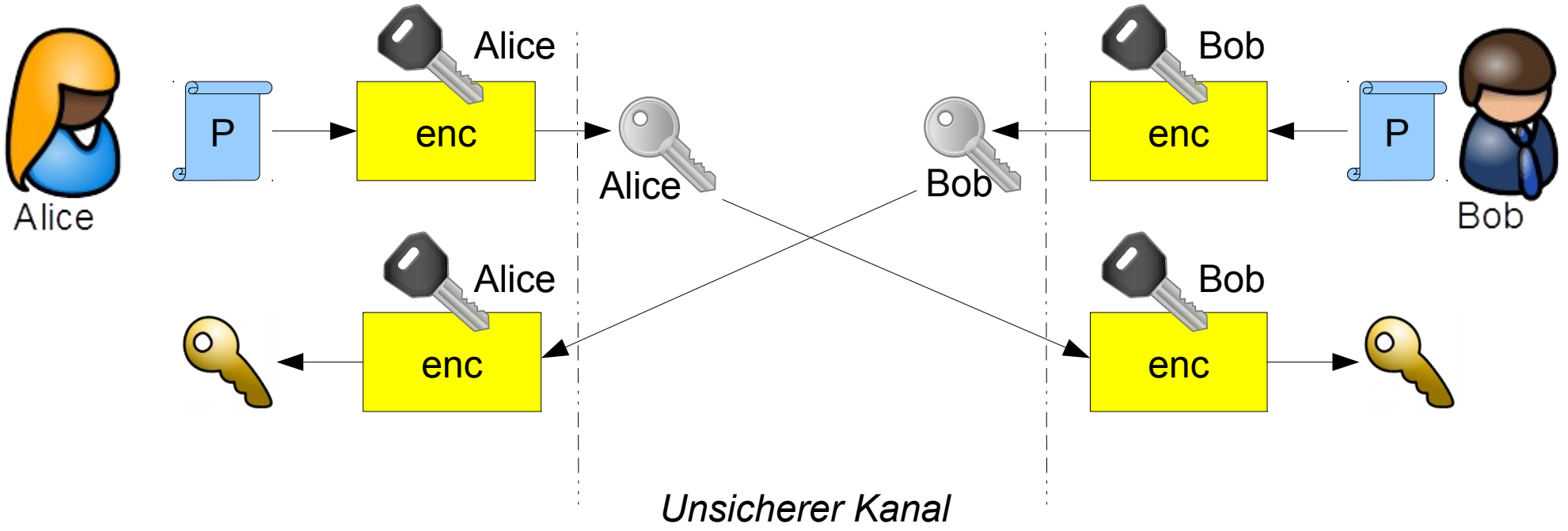



- **Diffie-Hellman** Protokoll
  - **Asymmetrisches** Verfahren, mit dem zwei Parteien einen geheimen Schlüssel über unsicheren Kanal aushandeln können
  - Nachfolgend **symmetrische** Verschlüsselung (Diffie-Hellmann ist nur Schlüsselaustausch, nicht Verschlüsselung selber)

- Principle
  - Alice kombiniert ihren Private Key mit Bobs Public Key
  - Bob kombiniert seinen Private Key mit Alices Public Key

    → Alice und Bob haben gemeinsam dasselbe Secret ohne dass private Informationen geflossen sind

# ILLUSTRATION




Symm. Key (der gesuchte Session Key)

Public Key (öffentlich)

Private Key (geheim)

P

Primzahl
(öffentlich)

# DISKUSSION DIFFIE-HELLMANN

- Empfänger einer Nachricht muss aktiv und verfügbar sein
  - Shared Secret kann nicht passiv von einer Seite vorbereitet werden
- Gut geeignet für eine Vollduplex-Verbindung, d.h., bidirektionalen Nachrichtenaustausch
  - Unnötig aufwendig für einseitigen Nachrichtenversand, da genügt RSA
- Wird u.a. von verschiedenen SSL-Versionen benutzt
- **Diffie-Hellman leistet keine Authentifizierung!**
  - Es muss bereits vorher sichergestellt sein, dass der Public Key von Alice und Bob auch *tatsächlich* zu den beiden Personen gehört!
    (Sonst: Man-in-the-Middle-Angriffe möglich)

# DIREKTE AUTHENTIFIZIERUNG: ZERTIFIKATE

- Authentifizierung ohne Trusted Third Party

# DIREKTE AUUTHENTIFIZIERUNG

- Alice und Bob authentifizieren sich gegenseitig ohne Hilfe eines Dritten
  - Beispielsweise mit **Zertifikaten**

- Vorteile:
  - Keine Trusted Third Party ist in die Kommunikation involviert, d.h., kann feststellen dass/wann Alice und Bob kommunizieren
  - Kein Single Point of Failure, kein Perfomanzproblem, skalierbar

- Nachteile:
  - Benötigt asymmetrische Kryptographie oder vorab über sicheren Kanal vereinbarte geheime Schlüssel

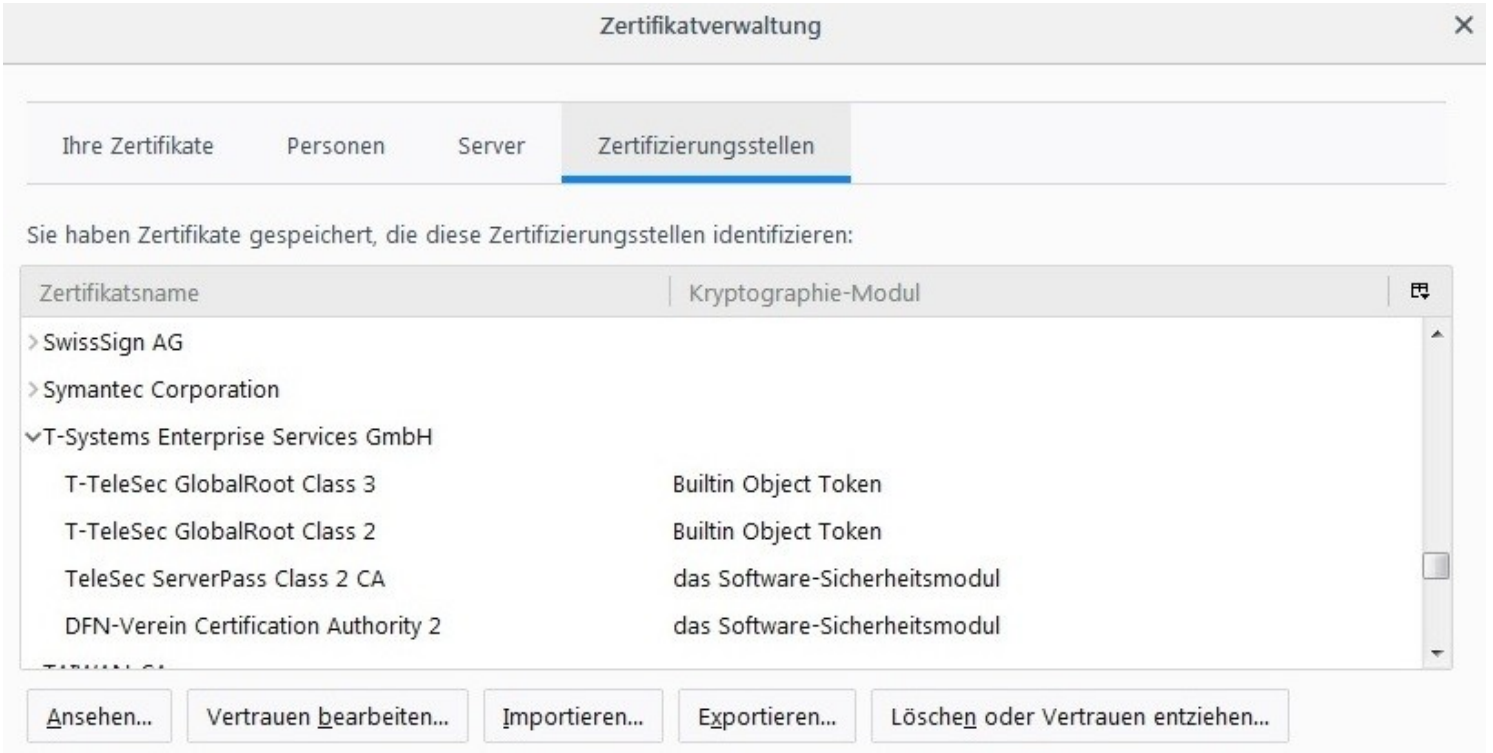
# ZERTIFIKATE

- Ein digital signiertes Dokument, mit dem der Aussteller bestätigt das ein bestimmter Public Key zu einem bestimmten Subjekt gehört
  - **signiertes Dokument:** Vom Aussteller mit digitaler Unterschrift signiert
  - **Aussteller:** Person oder Organisation, die das Zertifikat signiert hat
  - **Public Key:** Öffentlicher Schlüssel des Subjekts
  - **Subject:** Person oder Organisation, die der rechtmäßige Besitzer des Public Key/Private Key-Paares ist, um das es bei dem Zertifikat geht
  - Signatur des Ausstellers kann von einer **Certificate Authority** ebenfalls signiert worden sein von
  - **Certificate Chain**: Wenn Alice der CA von Bob vertraut, vertraut sie auch dem Zertifikat, mit dem Bob den Public Key von Carol bestätigt

- Operationen auf Zertifikaten:
  - können generiert, signiert, exportiert, importiert, vertraut und zurückgezogen werden

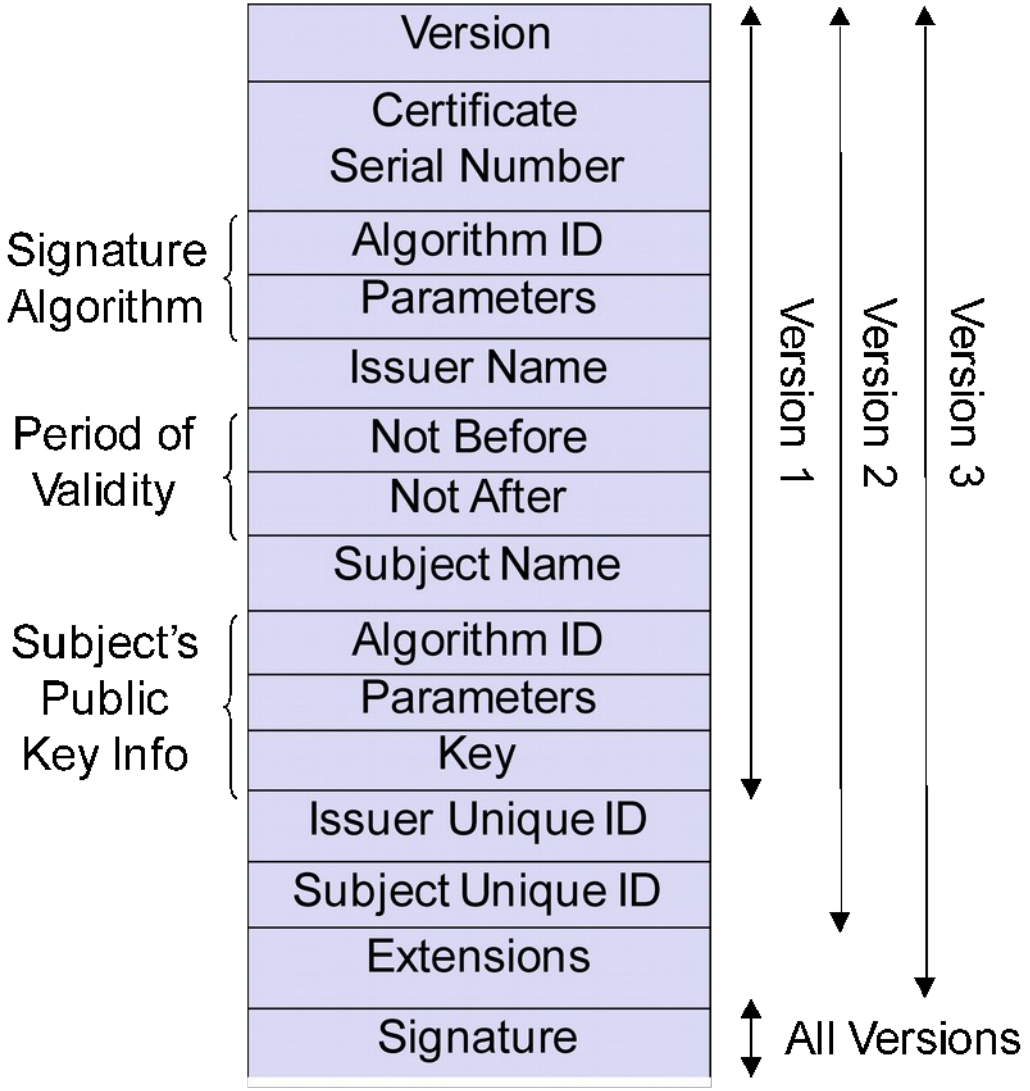
# WOHER BEKOMMT MAN EIN ZERTIFIKAT?

- Zwei Möglichkeiten
  1) Von einer Certificate Authority (CA, kostenfrei z.B. LetsEncrypt.org)




  2) Ein selbst signiertes Zertifikat erzeugen
     - keine CA involviert, d.h., Empfänger muss dem Zertifikat explizit vertrauen

# X.509 – ZERTIFIKATE




- Version - X.509 standard version number.
- Serial Number - A sequence number given to each certificate.
- Signature Algorithm Identifier - Name of the algorithm used to sign
- Issuer Name - Name of the issuer.
- Validity Period - Period during which this certificate is valid.
- Subject Name - Name of the owner of the public key.
- Subject Public Key Information - The public key

# X.509 AUTHENTIFIZIERUNG

- Wenn sich Alice gegenüber Bob authentifizieren will, sendet sie an Bob:
  - Zeitstempel (synchronisierte Uhren!)
  - Ihr signiertes Zertifikat CA<<A>>
  - Ein optionaler Session Key, der mit Bobs Public Key verschlüsselt ist

- Bob handelt wie folgt
  1) Verifiziert das Zertifikat (Gültigkeitsdauer, Signatur der CA)
  2) Holt Alices Public Key aus dem Zertifikat
  3) Verifiziert Alices Signatur und den Zeitstempel der Nachricht
  4) Entschlüsselt den Session Key mit seinem Private Key
  5) Kann nun sicher sein, dass Alice sein Kommunikationspartner ist

  *Für 2-Wege-Authentifizierung sendet Bob eine ebensolche Nachricht*

# X.509 CERTIFICATE CHAINS

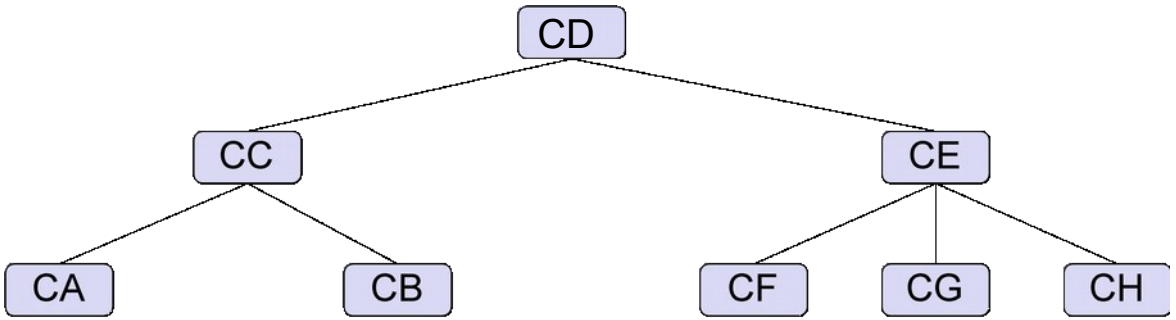
- Alice und Bob wollen kommunizieren. Das Zertifikat von Alice wurde von CA und das von Bob von CB unterzeichnet. Alice kennt weder Bob noch CB, und Bob kennt weder Alice noch CA
  - Alice und Bob können Public Keys ihrer Gegenüber nicht validieren

- Lösung: **Certificate chains** (Ketten von Zertifikaten)
  - Wenn CA und CB sich gegenseitig vertrauen, kann CA ein Zertifikat für CB signieren „CA<<CB>>“ und umgekehrt CB<<CA>>
    - Analoges Beispiel: Ausweisbehörden vertrauen sich gegenseitig, z.B. deutscher Reisepass wird von russischer Behörde anerkannt und umgekehrt
  - Wenn CA den öffentlichen Schlüssel von CB mit CA<<CB>> zertifiziert: Alice vertraut CA → CA vertraut CB → Alice vertraut auch CB
  - CB hat Zertifikat von Bob unterschrieben → Alice vertraut Zertifikat Bob d.h., Alice prüft CA<<CB>>, CB<<B>>

# X.509 CERTIFICATE CHAINS (FORTGESETZT)

- Certificate chains können beliebig lang werden
  - CA<<CC>>, CC<<CD>>, CD<<CE>>, CE<<CG>>, CG<<G>
    Alice kann Zertifikat von G validieren, obwohl sie nur CA vertraut
  - Wenn Alice CG<<G>> bekommt ist nicht immer offensichtlich, welche Zertifikate sie prüfen muss!
    - unter Umständen mehrere Wege möglich

- X.509 beschreibt Certificate Authorities als Zertifizierungshierarchie



# X.509 CERTIFICATE REVOCATION

- Schlimm: der Private Key von Alice ist kompromittiert
  (z.B. weil Eve ihn gestohlen hat oder weil er versehentlich öffentlich wurde)
- Alice muss zugehöriges Zertifikat *sofort zurückziehen* (**Revocation**)
  - Wenn nicht, könnte Eve sich als Alice ausgeben, bis die Gültigkeitsdauer des Zertifikats von allein abläuft

- Viel Schlimmer: der Private Key einer Certificate Authority ist kompromittiert
  - *Alle* Zertifikate, die mit diesem Key signiert wurden, müssen sofort zurückgezogen werden
  - Hacker könnte sich Zertifikate selbst unterschreiben, die von allen akzeptiert werden, die dieser Certificate Authority vertrauen

- Zurückziehen heißt: Zertifikat landet auf einer Ungültig-Liste, die regelmäßig von allen Beteiligten geprüft werden soll(te)
  - **Certificate revocation ist langsam und teuer**

# ARBITRATED AUTHENTICATION: KERBEROS

- Authentifizierung über Trusted Third Party

# ARBITRATED AUTHENTICATION

- Trusted Third Party (TTP) ist in jeden Authentifizierungsvorgang involviert
  - Z.B. mit **Kerberos**

- Vorteile:
  - Authentifizierung, ohne zuvor ein Secret auszutauschen oder Vertrauensbeziehungen zu Dritten zu vereinbaren
  - Es kann direkt (schnelle) symmetrische Verschlüsselung genutzt werden

- Nachteile:
  - TTP ist ein Single Point of Failure und kann Perfomanz beeinträchtigen
    - Verfügbarkeit der TTP ist kritisch
  - TTP sieht alle Authentifizierungen (und Kommunikationsvorgänge)
    - TTP ist ein wertvolles Angriffsziel

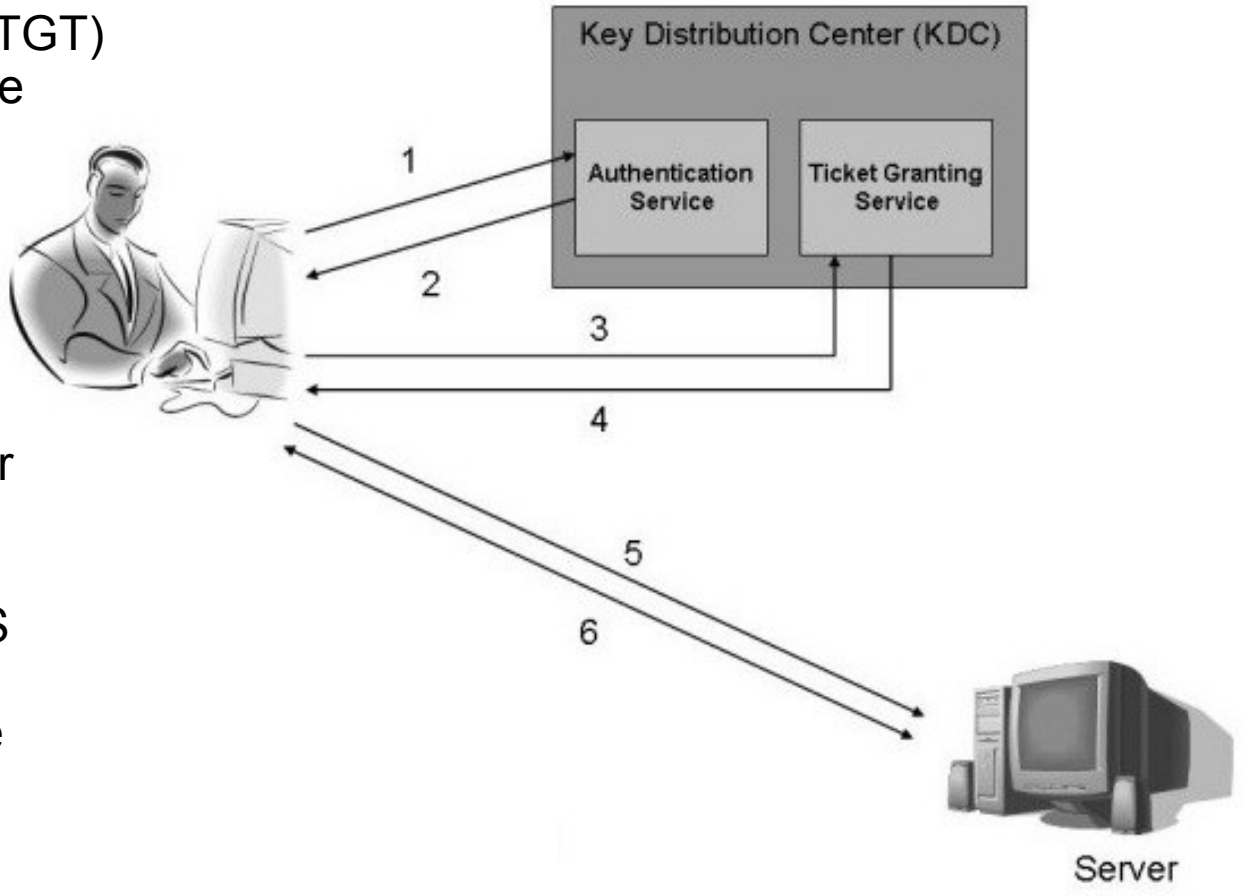
# KOMPONENTEN VON KERBEROS

- Am MIT in 80ern entwickelt, Grundlage für Microsoft Active Directory

1) **Authentifizierung**:
   - Alice authentifiziert sich gegenüber Authentication Server (AS), der dafür ein temporär gültiges Ticket-Granting-Ticket (TGT) ausstellt.
2) **Zugriffskontrolle**
   - Alice zeigt dieses TGT einem Ticket-Granting-Server (TGS), der die Zugriffsrechte von Alice für einen bestimmten Dienst S1 prüft.
   - Wenn Prüfung erfolgreich, stellt TGS Alice ein Service-Granting-Ticket SGT für Dienst S1 aus.
3) **Schlüsselaustausch**
   - AS generiert Session Key für Kommunikation von Alice mit TGS
   - TGS generiert Session Key für Kommunikation von Alice mit S1

# PROTOKOLLÜBERSICHT

1. Request ticket-granting ticket (TGT) From the Authentication Service (AS)
2. Receive the TGT from the AS
3. Show the TGT and request a service-granting ticket (SGT) from the Ticket-Granting Server (TGS)
4. Receive the SGT from the TGS
5. Submit SGT to network service
6. Use network service




Source: https://www.tecchannel.de/a/die-funktionsweise-von-kerberos,461645

# DISKUSSION KERBEROS

- Vorteile
  - Verteilte Architektur: In einem System aus vielen Servern Authentifizierung/Zugriffsrechte zentral in Kerberos pflegen
    - Jeder andere Dienst/Server muss nur das SGT auf Gültigkeit prüfen
  - Flexibilität, Skalierbarkeit: Jede der drei Protokollschritte kann auf einem anderen Server laufen

- Nachteile
  - Kerberos ist ein Single Point of Failure
  - Jeder Server/Service im System muss dem Kerberos-Server vertrauen
  - Herausforderungen im Design (Protokoll benötigt zwingend synchronisierte Uhren auf allen Servern, Virtualisierung kompliziert da Vertrauensbeziehungen an physischen Host gebunden)

# ABSCHLUSS

# ZUSAMMENFASSUNG

- Browser gehören zu den komplexesten Anwendungen überhaupt
- Sicherheit von Web-Anwendungen hängt wesentlich von zwei Faktoren ab
  - sichere Authentifizierung von Unbekannten (Rechnern, Personen)
  - Validierung *aller* Daten, die „von außen“ kommen
- Leistungen kryptographischer Protokolle
  - Direkte Authentifizierung (Zertifikate)
    - Trusted Third Party authentifiziert Zertifikat, welches die Kommunikationspartner authentifiziert
  - Vermittelte Authentifizierung (Kerberos)
    - Trusted Third Party authentifiziert die Kommunikationspartner direkt (aber ist ein Angriffsziel und erfährt auch, wer wann mit wem kommuniziert)

# MÖGLICHE PRÜFUNGSFRAGEN

- Erläutern Sie unterschiedliche Wege, mit denen ein Angreifer echt aussehende Kopien von Webseiten erzeugen und Anwender dazu bringen kann, diese anstelle der Originale zu benutzen.
- Entwickeln Sie einen Vorschlag, wie ein Webbrowser einen „Clickjack“-Angriff erkennen oder blockieren könnte.
- Welche der in diesem Kapitel erwähnten Angriffe betreffen das Schutzziel „Integrität“? Begründen Sie Ihre Antwort.
- Erläutern Sie die Vor- und Nachteile der vermittelten (arbitrated) Authentifizierung gegenüber der direkten Authentifizierung an einem eigenen Beispiel.
- Warum ist das Zurückziehen von Zertifikaten problematisch?