

# Digitale Zertifikate Grundlagen und Anwendungen

PC-Treff-BB Aidlingen

Günter Waller

# Agenda

- Ziele der Informationssicherheit: C I A
- Grundlagen Verschlüsselung
- Asymmetrische Verschlüsselung
- RSA
- Hash, digitale Signatur
- PKI
- Worauf basiert Vertrauen? (Trust)
- Anwendungsfälle (Übersicht)
- Webserver
- Client Identifikation: SSO, SSH, VPN
- DNSSEC
- Qualifizierte elektronische Signatur
- Links

# Ziele der Informationssicherheit: C I A

## Zentrale Schutzziele der Informationssicherheit

- #1 C = Confidentiality (Vertraulichkeit)
- #2 I = Integrity (Unversehrtheit [der Daten])
- A = Availability (Verfügbarkeit)

## Weitere Schutzziele

- #3 Authenticity (Echtheit, Vertrauenswürdigkeit)
- #4 Non-repudiation (Verbindlichkeit)
- #5 Accountability (Zurechenbarkeit)
- Anonymität

Mittel zur Zielerreichung #1-#5: Im Kern Kryptografie

# Grundlagen Verschlüsselung

- Was brauche ich für eine Verschlüsselung?
  - Eine Abbildung (Funktion im mathematischen Sinne), umgesetzt i. d. R. In einem **Algorithmus**
    - Für alle gleich, idealerweise standardisiert, nicht geheim
    - Umkehrbar (Ver- und Entschlüsselung)
  - Ein Geheimnis (Secret), i.d.R. ein **Schlüssel**
    - Nur dem- oder denjenigen bekannt, welche die betroffenen Daten kennen sollen
- Problem Schlüsselaustausch
  - Annahme: Daten werden über unsicheren Kommunikationskanal ausgetauscht
    - Out-of-band Übermittlung des Schlüssels – oder ...

# Asymmetrische Verschlüsselung

- Problem: Vermeidung der Übertragung des Schlüssels über das unsichere Medium
- Lösung: Es gibt 2 Schlüssel
  - Encryption Key  $\mathcal{E}$  zur Verschlüsselung
  - Decryption Key  $\mathcal{D}$  zur Entschlüsselung
- $\mathcal{E}$  und  $\mathcal{D}$  bilden ein Paar
- $\mathcal{E}$  muß nicht geheim gehalten werden - im Gegenteil.
- $\mathcal{D}$  muß geheim bleiben, gehört nur einer Person.
- $\mathcal{D}$  darf nicht leicht aus  $\mathcal{E}$  herzuleiten sein.
- Die Verschlüsselungsfunktion ist eine Einwegfunktion (one-way function): Eine Richtung geht leicht, die andere schwer.
- Hat man eine Zusatzinformation ( $\mathcal{D}$ ), geht es leicht. Dann spricht man von einer Falltürfunktion (trap door function).

## Beispiel: RSA

- Geschichte: Anfang der 70er Jahre wurde beim britischen GCHQ ein asymmetrisches Verfahren entwickelt, aber nie veröffentlicht.
- 1976: Diffie und **Hellman** (Stanford University) veröffentlichen ein theoretisches Paper über Public Key Kryptographie. Zu dem Zeitpunkt betrachten sie das Problem als ungelöst.
- 1977: **Rivest, Shamir und Adleman** (MIT) stießen beim Versuch, obiges Papier zu widerlegen auf ein funktionierendes Verfahren: RSA. Der Artikel zeigt auch schon auf, dass das Verfahren sowohl für Signaturen als auch für Vertraulichkeit (Verschlüsselung) geeignet ist.
- Daraus wird 1983 ein Patent und es entsteht eine **Firma** für Security Produkte. Das Patent ist allerdings seit 2000 abgelaufen.

# RSA Algorithmus 1/3

- Der öffentliche Schlüssel besteht aus einem Paar ganzer Zahlen (e,n).
- Zu verschlüsselnde Nachricht M (Message) durch eine Zahl repräsentieren ( $<n$ ). Kodierung ist egal. Längere Nachricht in mehrere kleine zerlegen.
- Mit der Verschlüsselungsfunktion E (Encrypt) ermittelt man nun die verschlüsselte Nachricht C (Ciphertext):

$$C = E(M) = M^e \pmod{n}$$

- Das Gegenstück, der private Schlüssel, besteht aus einem weiteren Paar (d,n) mit dem gleichen n.
  - Dazu gehört die Entschlüsselungsfunktion D (Decrypt):
- $$M = D(C) = C^d \pmod{n}$$
- Wichtig: Die Länge von C wird durch E nicht größer als n-1.

## RSA Algorithmus 2/3

- Bis dahin klingt es einfach. Der Aufwand steckt
  - a) im Finden zweier großer Primzahlen  $p$  und  $q$  als Startwert für die Schlüsselerzeugung <sup>1</sup>
  - b) in der Erzeugung der Schlüssel:
    - i. [Beschreibung](#) in Wikipedia
    - ii. Mathematischer Beweis im RSA-Originalartikel
  - c) in der Schlüssellänge, d.h. im erforderlichen Rechenaufwand

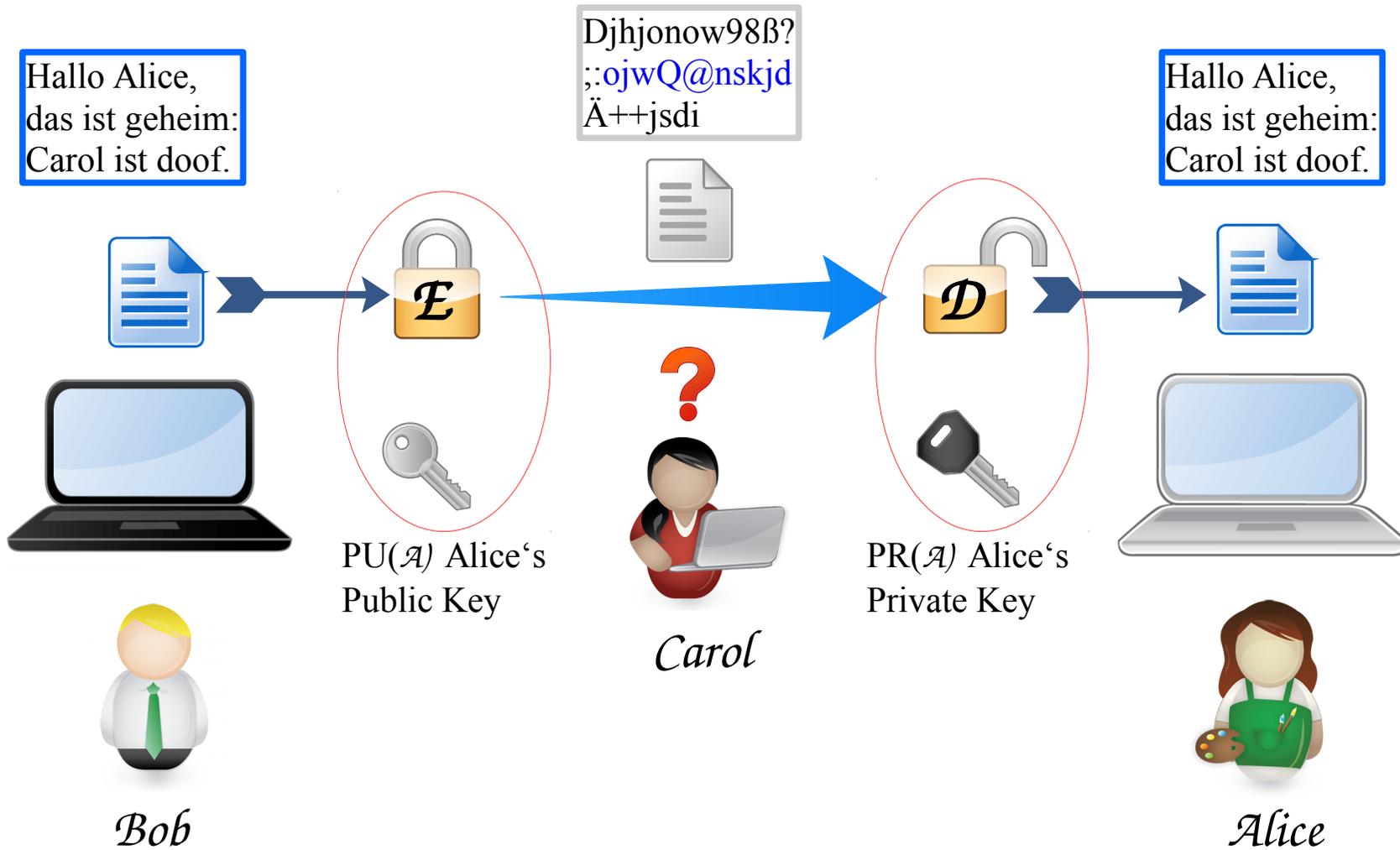
---

<sup>1</sup> Es gilt  $n = p \times q$ . Man erzeugt Zufallszahlen in der gewünschten Länge und testet, ob sie prim sind. Im Mittel braucht man dafür 115 Versuche. Wir verwenden Zahlen, die zu groß sind, um das mit Brute Force durchzuprobieren. Es gibt keinen Beweis, sondern nur Tests, die teilbare Zahlen erkennen. Davon werden sehr viele ausgeführt, die alle negativ ausgehen müssen. Es bleibt ein geringes, vernachlässigbares Restrisiko, dass man eine nicht-Primzahl erwischt hat. Dann wird  $D(E(M)) \neq M$ .

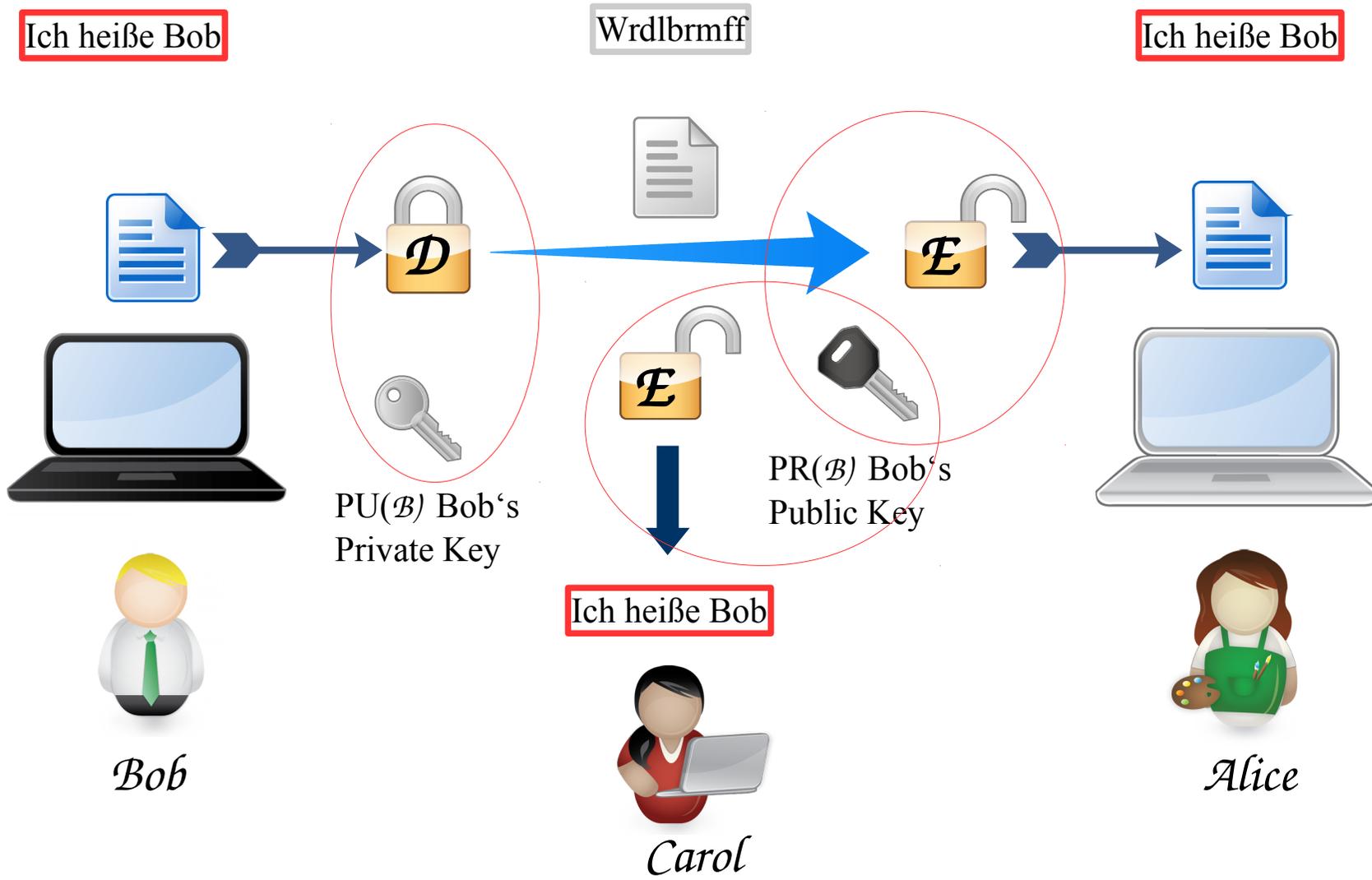
## RSA Algorithmus 3/3

- Beschleunigung der Rechnung mit dem **Chinesischen Restsatz** (CRT-RSA)
- Das unmodifizierte RSA-Verfahren ist angreifbar, weil es deterministisch ist, d.h. der gleiche Klartext gibt immer das gleiche Chifftrat. Damit sind Angriffe möglich, ohne den Schlüssel zu faktorisieren. Lösung: Der Klartext wird mit einem Zufallswert ergänzt (Padding) und dann verschlüsselt.
- Aus Effizienzgründen wird RSA in der Regel in Hybridverfahren mit symmetrischen Verfahren kombiniert. Es wird zufällig ein Sitzungsschlüssel für ein symmetrisches Verfahren erzeugt, der per RSA verschlüsselt und zusammen mit der Nachricht übertragen wird.
- Für die Sicherheit von RSA sind Primzahlen mit mehreren hundert Dezimalstellen (mindestens 2048 Bit) erforderlich. Damit können symmetrische Schlüssel jeder üblichen Länge verschlüsselt werden (z.B. AES mit einer Schlüssellänge von 128, 168 oder max. 256 Bit)
- Further Reading: <http://www.muppetlabs.com/~breadbox/txt/rsa.html>

# Geheime Daten, verschlüsselt übertragen über ein unsicheres Netz



# Öffentliche Daten, vom Absender signiert, Sender-Identität und Integrität bewiesen



# Hash – was ist das?

Eine Hash-Funktion bildet Daten beliebiger Länge auf Daten einer festen Länge ab.

- Eigenschaften einer kryptografischen Hash-Funktion:
  - Deterministisch
  - Leicht zu berechnen
  - Praktisch nicht umkehrbar
  - Kleine Änderungen am Input bewirken einen völlig anderen Output
  - Praktisch unmöglich, ein zweites Input-Datum zu finden, das den gleichen Output erzeugt
- Anwendungen in kryptografischen Verfahren, insbes. als **digitale Signaturen** und für Authentifizierung (Password Hashing). Nicht-kryptografische Anwendungen: Hash-Tables, digitale Fingerabdrücke, Prüfsummen (bei Downloads).

# Hash – Praxis

- Die älteren Hash-Verfahren (MD4, MD5, SHA-0, u.a.) gelten inzwischen als unsicher. Gegen SHA-1 wurde 2017 eine Kollision berechnet.
- Aktuell sind:
  - SHA2: Oberbegriff für SHA-224, SHA-256, SHA-384 und SHA-512. Die Zahl steht für die Länge des Hashwertes.
  - SHA3: Zukunftssicherer als SHA-2, verwendet andere Technik. Soll bereitstehen, wenn eines Tages SHA-2 gebrochen wird.
- Verschärfungen bei Passwörtern (zur Erschwerung von Brute-Force-Angriffen mit Rainbow Tables von bekannten Hashwerten):
  - Salt: Klarnachricht wird um einen Zufallswert erweitert, dann wird Hash gebildet. PW geht nicht im Klartext durchs Netz.
  - Pepper: Server ergänzt Klartext um einen geheimen Wert, der nicht zusammen mit den Hashes gespeichert wird.
  - Grundsätzlich sollten PW **nie** im Klartext gespeichert werden.

# Prinzip der digitalen Signatur

Hier werden Hash- und Verschlüsselungs-Funktion zu einem Verfahren zusammengefügt.

Siehe z.B.: <http://www.itwissen.info/Digitale-Signatur-digital-signature-DSig.html>

# PKI – Was ist das? Wozu dient das?

## PKI = Public Key Infrastructure

- Ziel (allgemein): **Übermittlung des öffentlichen Schlüssels** (public key) in **asymmetrischen Kryptosystemen**, Garantie der Echtheit bzw. der **Korrektheit der zugehörigen Identität**.
- Mittel zum Zweck: **Digitales Zertifikat**. Signiert vom **Aussteller**.
  - Authentizität der Signatur durch Überprüfung des Zertifikates des Ausstellers.
  - usw. usw.
  - Am Ende (oder Beginn) dieser Kette (**Trust Chain**) muss ein Zertifikat stehen, dem man auf andere Art und Weise, also ohne weiteres Zertifikat vertraut: das Root-Zertifikat.
    - Selbst ausgestellt, selbst als vertrauenswürdig importiert
    - Vom Browserhersteller vorinstalliert
    - Vom Betriebssystem vorinstalliert

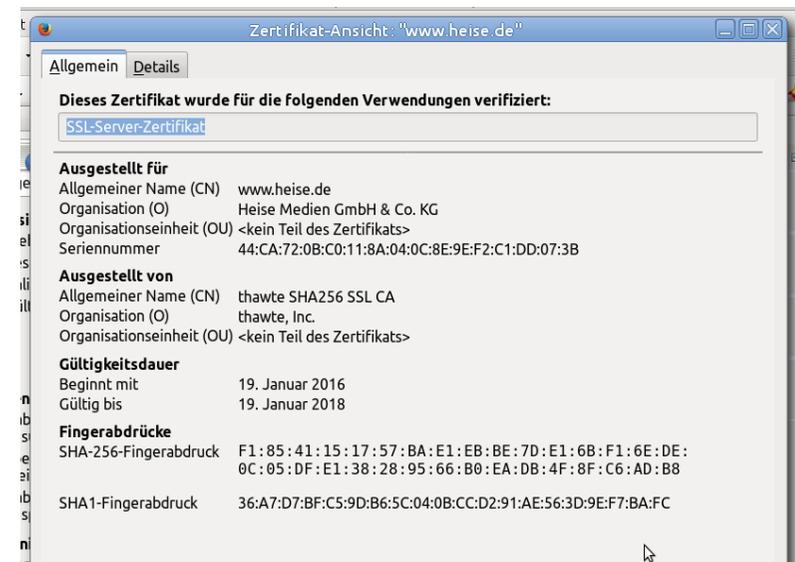
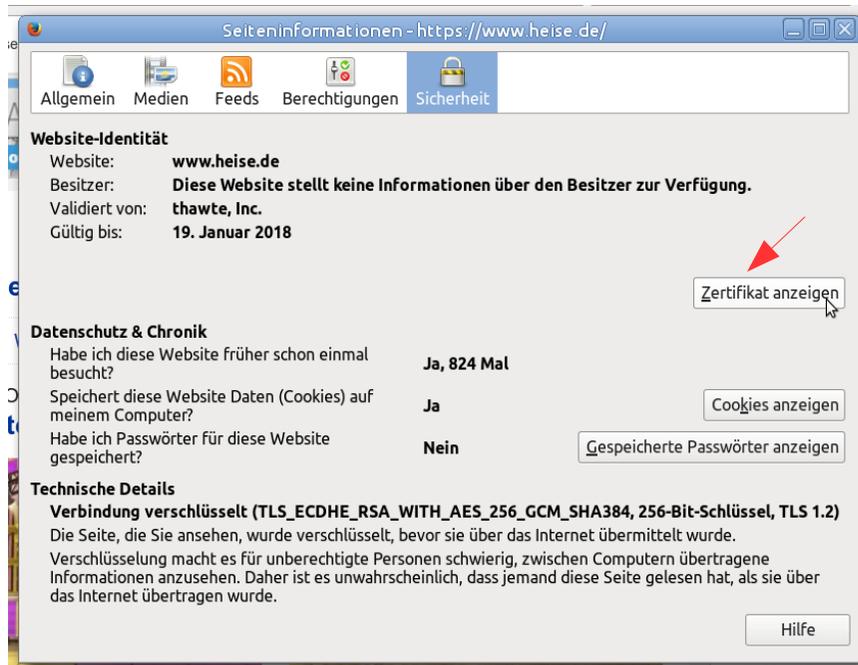
# Demo

- <https://www.heise.de/>

## 1. Im Browser auf das Schloss klicken

- a) > Verbindungsdetails
- b) Weitere Informationen

## 2. Zertifikat anzeigen



## Demo (Forts.)

- Walkthrough durch die Details:
  - Zertifikatshierarchie
    - Root-Zertifikat ist selbstsigniert
  - Version (**X.509** V3), Seriennummer, Signaturalgorithmus (**PKCS**), Validität, Inhaber (LDAP-Attribute)
  - Erweiterungen: CRL-Verteilungspunkte (Download CRL), Zertifizierungsstellen-Informationen-Zugriff (u.a. OCSP-Link)
  - CRL ansehen mit openssl:

```
openssl crl -inform DER -noout -text -in tg.crl > crl_view.txt
```

# Root-Zertifikat ist selbstsigniert

The screenshot shows a Windows Certificate Viewer window titled "Zertifikat-Ansicht: 'www.heise.de'". It has two tabs: "Allgemein" (selected) and "Details".

**Zertifikats-hierarchie**

- thawte Primary Root CA - G3
  - thawte SHA256 SSL CA
    - www.heise.de

**Zertifikats-Layout**

- thawte Primary Root CA - G3
  - Zertifikat
    - Version
    - Seriennummer
    - Zertifikatsunterzeichnungs-Algorithmus
    - Aussteller**
    - Validität
      - Nicht vor
      - Nicht nach
    - Inhaber
    - Angaben zum öffentlichen Schlüssel des Inhabers

**Feld-Wert**

```
CN = thawte Primary Root CA - G3
OU = "(c) 2008 thawte, Inc. - For authorized use only"
OU = Certification Services Division
O = "thawte, Inc."
C = US
```

Two red arrows originate from the "Aussteller" field in the "Zertifikats-Layout" section and point to the "thawte Primary Root CA - G3" entry in the "Zertifikats-hierarchie" section and the "CN = thawte Primary Root CA - G3" line in the "Feld-Wert" section, illustrating that the root certificate is self-signed.

Taskbar: en - Mozilla..., 2017\_Zertifikate, guenter@T510: ~/mat..., Zertifikat-Ansicht: "w... Erinnerung

# Trust für Root-Zertifikat kommt vom Browser

The screenshot shows the Windows Certificate Management console (Zertifikatverwaltung) with the 'Zertifizierungsstellen' tab selected. A table lists certificates and their cryptographic modules. The 'thawte Primary Root CA - G3' certificate is highlighted. A dialog box titled 'CA-Zertifikat-Vertrauenseinstellungen bearbeiten' is open, showing the trust settings for this certificate. The 'Dieses Zertifikat kann Websites identifizieren' checkbox is checked, while the others are unchecked.

Zertifikatsname	Kryptographie-Modul
TeliaSonera Root CA v1	Builtin Object Token
▼thawte, Inc.	
thawte Primary Root CA	Builtin Object Token
thawte Primary Root CA - G2	Builtin Object Token
<b>thawte Primary Root CA - G3</b>	<b>Builtin Object Token</b>
thawte EV SSL CA - G3	
thawte DV SSL CA - G2	

Das Zertifikat "thawte Primary Root CA - G3" repräsentiert eine Zertifizierungsstelle.

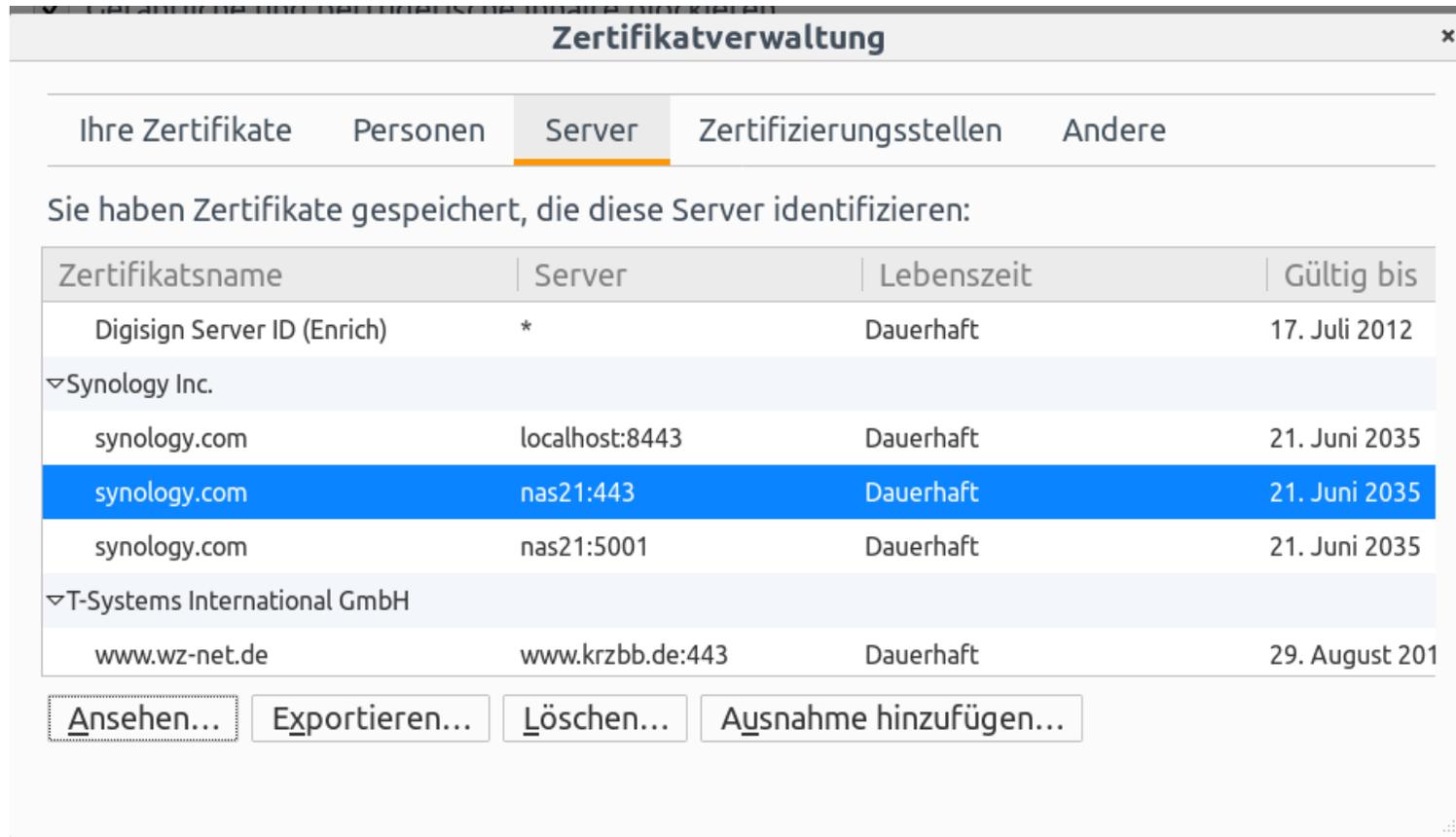
Vertrauenseinstellungen bearbeiten

- Dieses Zertifikat kann Websites identifizieren.
- Dieses Zertifikat kann Mail-Benutzer identifizieren.
- Dieses Zertifikat kann Software-Hersteller identifizieren.

Abbrechen OK

# Ausnahmen im Browser hinzufügen

Für eigenes NAS Ausnahme(n) hinzugefügt.



**Zertifikatverwaltung**

Ihre Zertifikate   Personen   **Server**   Zertifizierungsstellen   Andere

Sie haben Zertifikate gespeichert, die diese Server identifizieren:

Zertifikatsname	Server	Lebenszeit	Gültig bis
Digisign Server ID (Enrich)	*	Dauerhaft	17. Juli 2012
▽Synology Inc.			
synology.com	localhost:8443	Dauerhaft	21. Juni 2035
synology.com	nas21:443	Dauerhaft	21. Juni 2035
synology.com	nas21:5001	Dauerhaft	21. Juni 2035
▽T-Systems International GmbH			
www.wz-net.de	www.krzbb.de:443	Dauerhaft	29. August 201

# Bestandteile einer PKI

- **Digitale Zertifikate:** Digital signierte elektronische Daten, die sich zum Nachweis der Echtheit von Objekten verwenden lassen (Definition aus Wikipedia). Enthält Verfallsdatum.
- Zertifizierungsstelle (CA) stellt das Zertifikat aus.
- Registrierungsstelle (RA) bekommt Antrag, prüft die Richtigkeit der Angaben (Beispiele: siehe Let's Encrypt, oder Überprüfung Ausweis), informiert CA.
- Zertifikatssperrliste (CRL) enthält die Zertifikate, die vor Ablauf ihrer Gültigkeit widerrufen werden. Client (z.B. Browser, sollte dies prüfen).
- Verzeichnisdienst (LDAP Server) für Zertifikate und CRLs.
- Validierungsdienst (VA) führt Gültigkeitsprüfungen in Echtzeit durch (z.B. OCSP), falls die Anwendung dies erfordert.
- Subscriber (Zertifikatsinhaber), Participant (Nutzer, der vertraut)

# Dateiformate für Zertifikate

- Laut [Wikipedia](#):
  - **.CER** – DER- oder Base64-kodiertes Zertifikat
  - **.CRT** – DER- oder Base64-kodiertes Zertifikat
  - **.CSR** – Base64-kodierte Zertifizierungsanfrage des öffentlichen Schlüssels (plus weitere Metadaten des Besitzers) an eine CA, umschlossen von „-----BEGIN CERTIFICATE REQUEST-----“ und „-----END CERTIFICATE REQUEST-----“
  - **.DER** – DER-kodiertes Zertifikat
  - **.P12** – PKCS#12, kann öffentliche Zertifikate und private Schlüssel (Kennwort-geschützt) enthalten.
  - **.P7B** – Siehe .p7c
  - **.P7C** – PKCS#7-signierte Datenstruktur ohne Dateninhalt, nur mit Zertifikat(en) oder Sperrliste(n)
  - **.PEM** – Base64-kodiertes Zertifikat, umschlossen von „-----BEGIN CERTIFICATE-----“ und „-----END CERTIFICATE-----“
  - **.PFX** – Siehe .p12
- DER = Distinguished Encoding Rules
- CSR = Certificate Signing Request

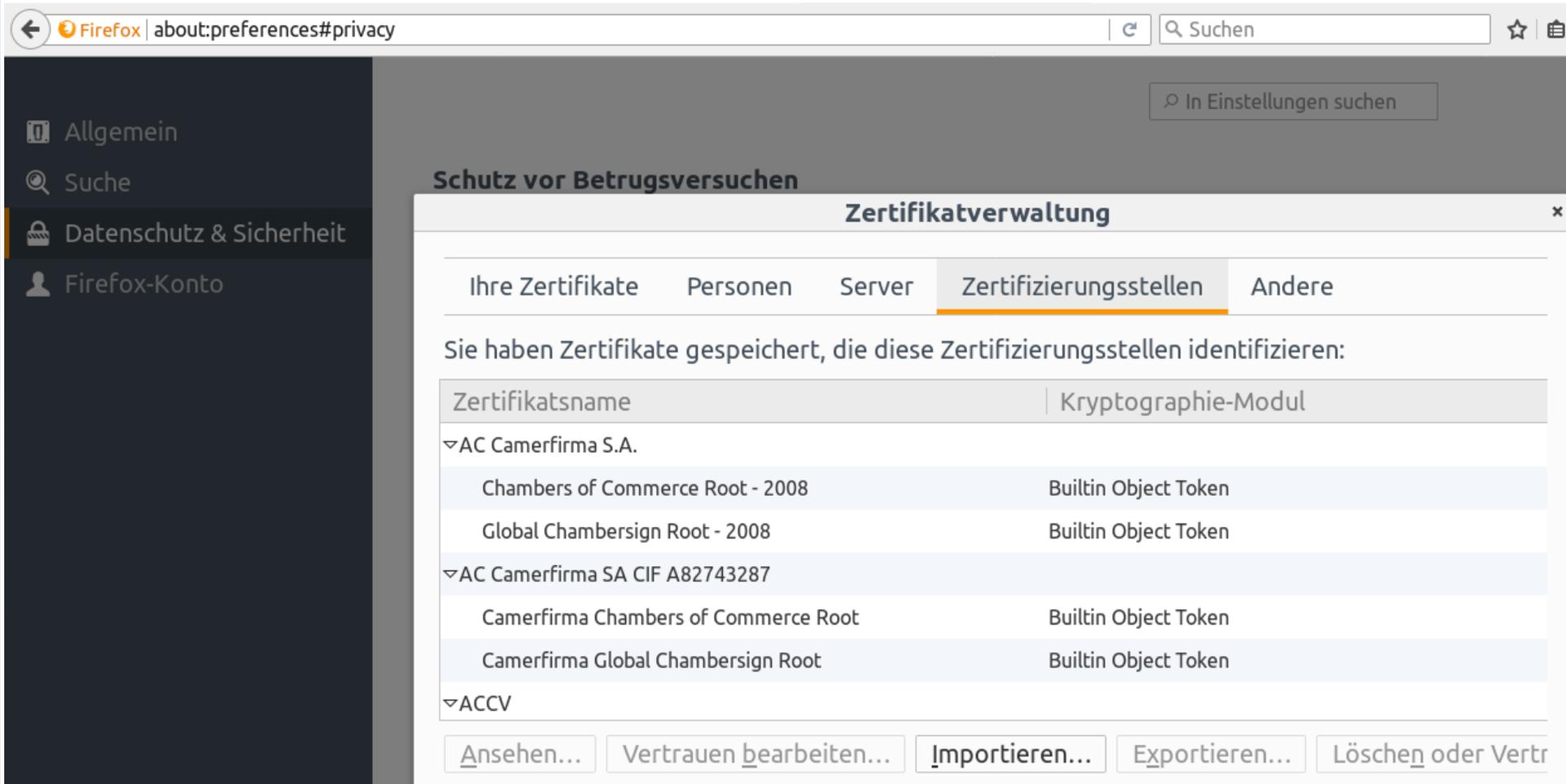
## Worauf basiert Vertrauen (Trust)?

Bei dem beschriebenen System handelt es sich um ein hierarchisches Gebilde. Das Vertrauen steht und fällt damit, dass Vertrauen in die Ur-CA am Anfang der Kette besteht, der Root-CA.

Laut Wikipedia wird dieses Feld von einer kleinen Zahl von Unternehmen dominiert, genannt werden Symantec (ex: VeriSign), Comodo, GoDaddy (Webhoster), die 3/4 des Marktes für TLS-Zertifikate von Webservern halten sollen. In der Vergangenheit gab es gelegentlich Zwischenfälle, bei denen eine Root-CA kompromittiert wurde. Als Folge mußten die entsprechenden Zertifikate aus den Browsern, Betriebssystemen entfernt werden, z.T. auch händisch.

- [Diginotar](#) (2011)
- [Google und Symantec](#) (aktuell)

# Beispiel: Firefox Zertifikatverwaltung



Firefox | about:preferences#privacy

In Einstellungen suchen

**Schutz vor Betrugsversuchen**

**Zertifikatverwaltung**

Ihre Zertifikate   Personen   Server   **Zertifizierungsstellen**   Andere

Sie haben Zertifikate gespeichert, die diese Zertifizierungsstellen identifizieren:

Zertifikatsname	Kryptographie-Modul
<ul style="list-style-type: none"> <li>AC Camerfirma S.A. <ul style="list-style-type: none"> <li>Chambers of Commerce Root - 2008</li> <li>Global Chambersign Root - 2008</li> </ul> </li> <li>AC Camerfirma SA CIF A82743287 <ul style="list-style-type: none"> <li>Camerfirma Chambers of Commerce Root</li> <li>Camerfirma Global Chambersign Root</li> </ul> </li> <li>ACCV</li> </ul>	Builtin Object Token

Ansehen...   Vertrauen bearbeiten...   Importieren...   Exportieren...   Löschen oder Vertr

# Anwendungsfälle

- <https://de.wikipedia.org/wiki/RSA-Kryptosystem#Anwendungsgebiete>
- Internet- und Telefonie-Infrastruktur: **X.509-Zertifikate**
- Übertragungs-Protokolle: **IPsec, TLS, SSH, WASTE**
- E-Mail-Verschlüsselung: **OpenPGP, S/MIME**
- Authentifizierung französischer Telefonkarten
- Kartenzahlung: EMV
- RFID Chip auf dem deutschen Reisepass
- Electronic Banking: HBCI
- Single Sign-On (SAML, OpenId, OAuth)
- **VPN**
- **DNSSEC**

## „Schlüsselverwendung“ - öffentlicher Schlüssel

- **Digitale Signatur:** für digitale Signaturen, nicht für Nichtabstreitbarkeit, d.h. eher kurzfristiger Charakter.
- **Nichtabstreitbarkeit:** für digitale Signaturen eines Nichtabstreitbarkeitsservice, z.B. Notariatsservice.
- **Schlüsselverschlüsselung:** für die Verschlüsselung von anderen Schlüsseln oder Sicherheitsinformationen.
- **Datenverschlüsselung:** zur Verschlüsselung von Benutzerdaten (außer andere Schlüssel).
- **Schlüsselvereinbarung:** Diffie Hellman Algorithmus soll für die Schlüsselvereinbarung verwendet werden.
- **Zertifikatsignatur:** für die Verifikation von Signaturen auf Zertifikaten, d.h. sinnvoll bei CA-Zertifikaten.
- **CRL-Signatur:** für die Verifikation von Signaturen auf CRLs, sinnvoll bei CA-Zertifikaten.
- **Nur Verschlüsselung/nur Entschlüsselung:** nur mit Schlüsselvereinbarung nach Diffie Hellman sinnvoll.

# Erweiterte Schlüsselerwendung

In der Abteilung „Extended Key Usage“ des Zertifikates finden sich wichtige Punkte (hier: von Microsoft definiert):

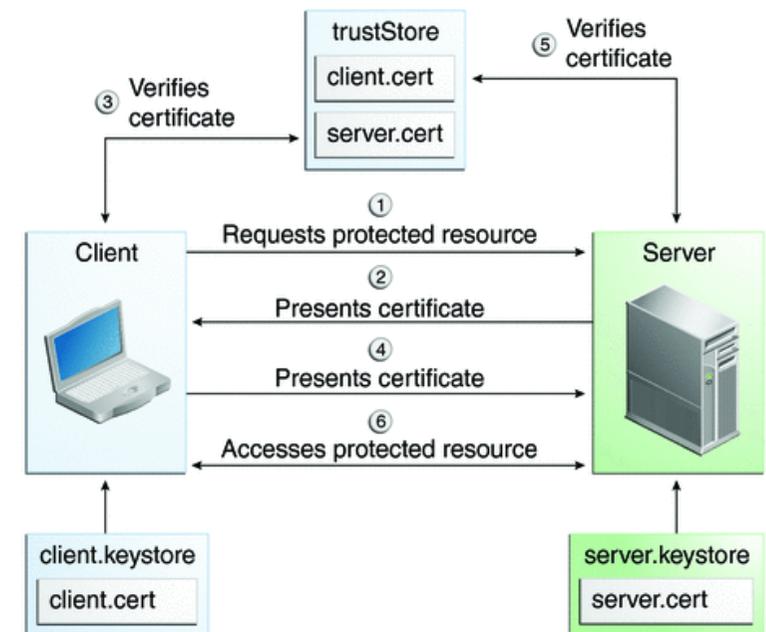
- Serverauthentifizierung (Webserver)
- Clientauthentifizierung

u.a.

Quelle (z.B.): <https://www.cryptas.com>

# Webserver

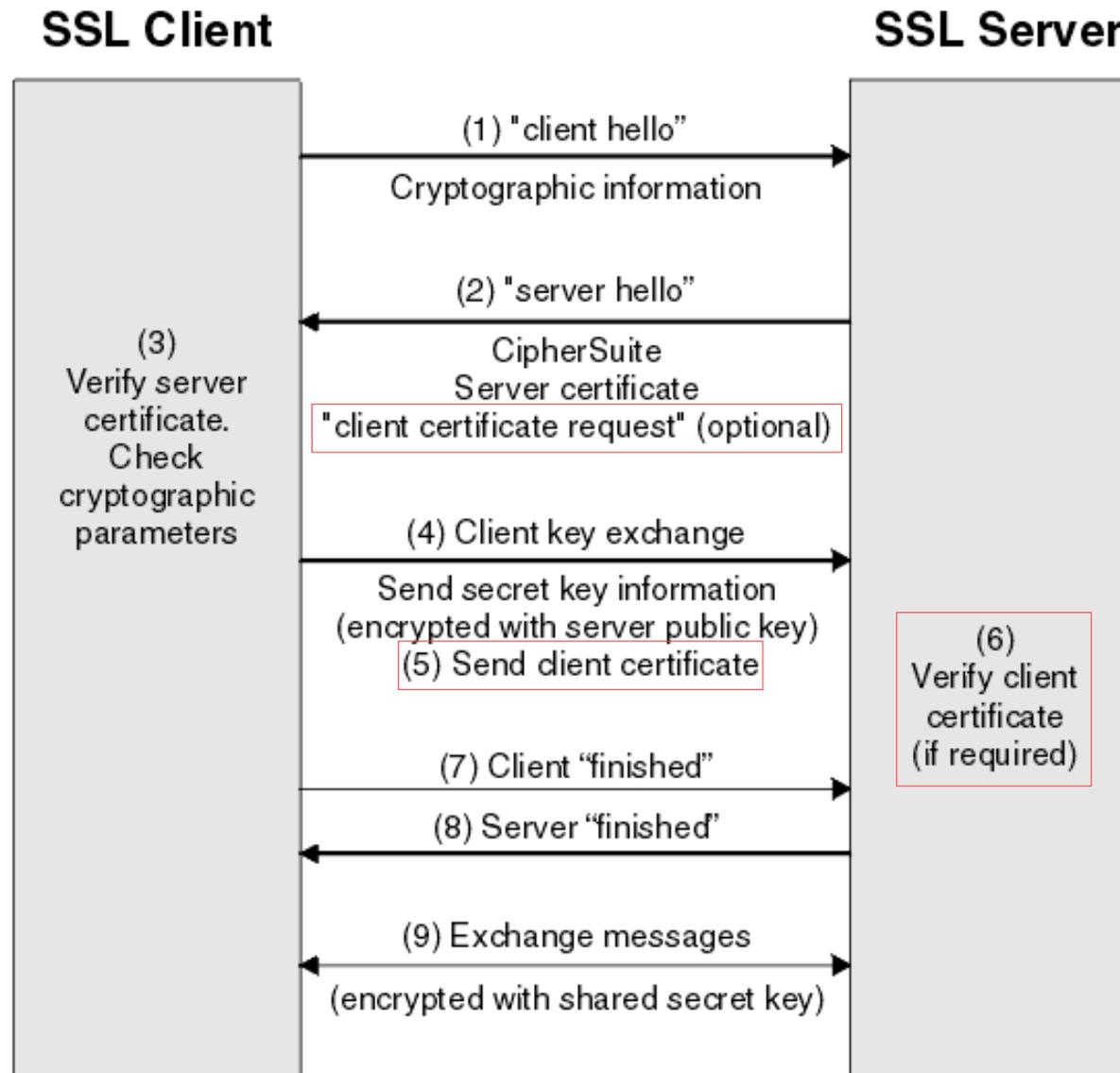
- Der Server identifiziert sich bei HTTPS mit einem Serverzertifikat. Bei HTTP gibt es keine Gewähr, dass es sich nicht um einen Fake Server handelt.
- Auch der Client kann sich gegenüber dem Server mit einem Zertifikat identifizieren, d.h. in dem Fall authentifizieren. Das Zertifikat enthält dann in Attributen die Identität. Der Server validiert das Zertifikat und prüft die Identität gegen sein Benutzerverzeichnis. Es fließt kein Passwort. Gleichzeitig dient das Zertifikat dem TLS-Verbindungsaufbau.



# TLS Handshake

- Verständigung auf das verwendete Protokoll
- Auswahl Krypto-Algorithmus
- Austausch und Validierung der Zertifikate
- Generierung gemeinsamer Schlüssel
- Nutzdaten symmetrisch verschlüsselt

# TLS Handshake mit Client Zertifikat



## Single Sign-On

- **SAML**, ein auf XML-formatierten Tokens aufbauendes SSO-Verfahren, enthält digitale Zertifikate als **Bestandteil**.
- **OAuth**, ein Autorisierungsprotokoll zwischen Client, User (Resourceowner) und Resourceserver erlaubt dem User, an den Client Rechte zu vergeben, ohne diesem das eigene password am Server geben zu müssen. Das Mittel ist ein Access Token, der zeitlich und funktional limitiert ist.
  - Mein Beispiel: YouTube App auf dem Smart TV verlangt von mir, mich auf einem anderen Gerät bei Google anzumelden und dort den angezeigten Code einzugeben, so dass die App unter meiner Id agieren kann.
  - Für Banking wird jetzt **OAuth mit Client-Zertifikaten** verknüpft, um es wasserdicht zu machen.

# OAuth am TV



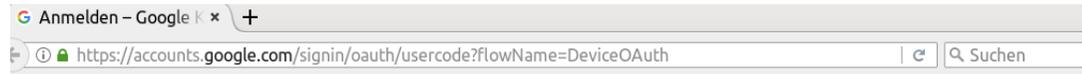
Suchen

## Anmeldung bei YouTube

Wenn du dich anmeldest, kannst du deine Kontoinhalte (z. B. Abos und Playlists) auf diesem Fernseher aufrufen.

1. Öffne die folgende Seite auf deinem Smartphone oder auf deinem Computer:  
**youtube.com/activate**
2. Melde dich in deinem Google-Konto an. Gib danach folgenden Code ein:

**QKLZ-ZBDZ**



Google

### Gerät verbinden

Geben Sie den auf Ihrem Gerät angezeigten Code ein

Code eingeben

QKLZ-ZBDZ

WEITER

Google

### Willkommen

YouTube on TV möchte

- YouTube-Konto verwalten
- Leih- und Kaufverlauf abrufen und verwalten

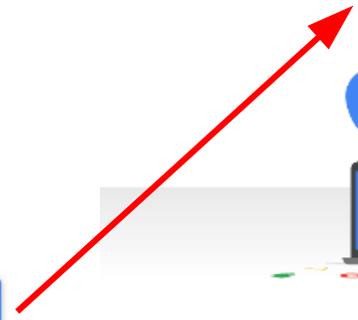
YouTube on TV die Erlaubnis dazu erteilen?  
Sie können die [Nutzungsbedingungen](#) und [Datenschutzrichtlinien](#) dieser App einsehen. Unter [Mein Konto](#) können Sie diese und jede andere mit Ihrem Konto verknüpfte App entfernen

ABBRECHEN ZULASSEN

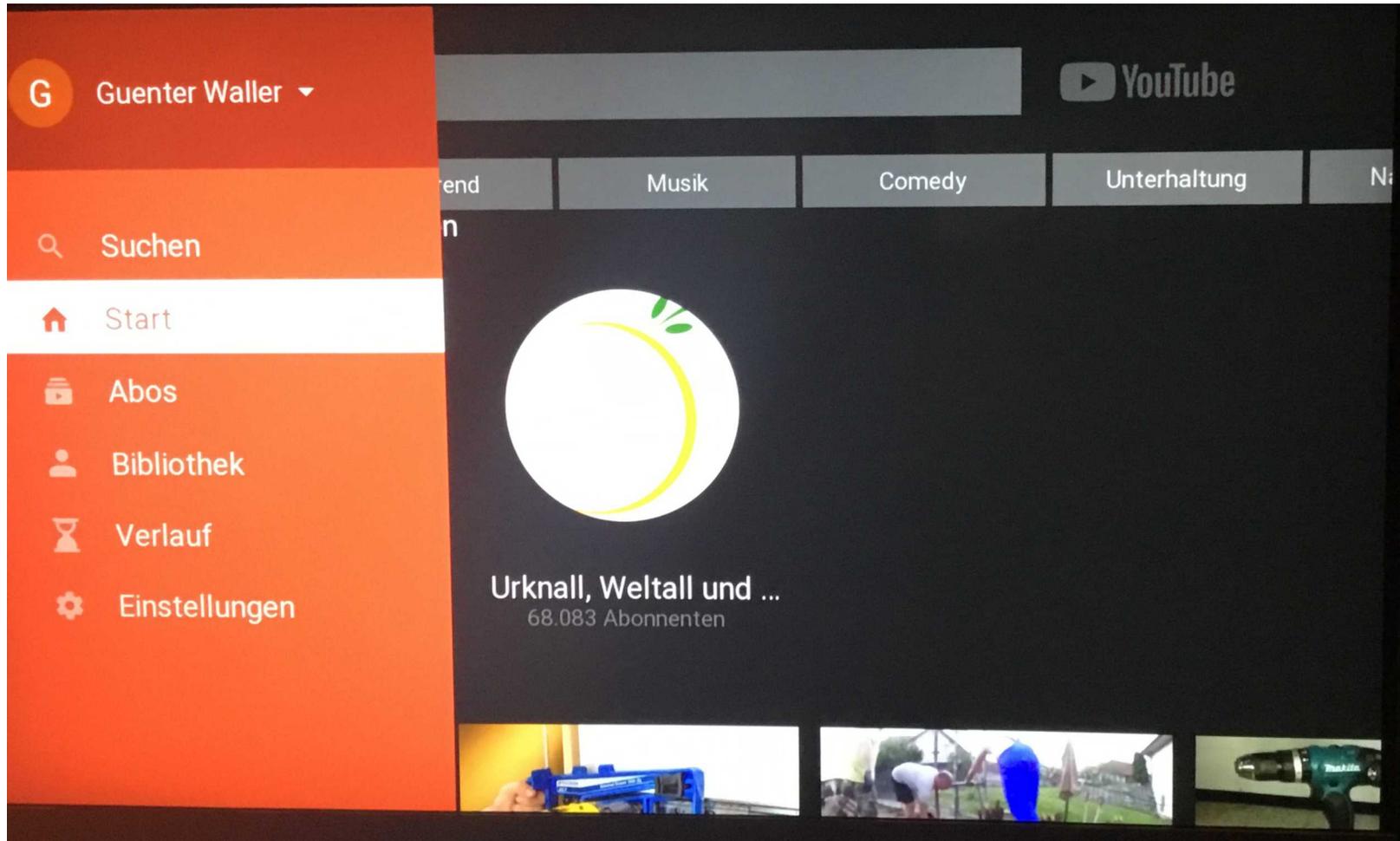
Google

### Fertig! Das Gerät ist verbunden.

Fahren Sie auf Ihrem Gerät fort  
Um alle Apps zu sehen, die mit Ihrem Google-Konto verknüpft sind, rufen Sie [Mein Konto](#) auf



# OAuth am TV



# SSH

- Rückgriff auf OpenWRT Vortrag von 2013
  - Inzwischen wurde allerdings der SSH Service Dropbear durch OpenSSH ersetzt, daher gibt es serverseitig leichte Abwandlungen:
    - `/root/-ssh/authorized_keys` (neuer Pfad)
    - Keytype `ssh-dss` nicht mehr unterstützt
  - Auf dem Linux Client deshalb `ssh-rsa` erzeugt:
    - `ssh-keygen -t rsa`
    - Eintrag in `~/.ssh/config`
      - `PubkeyAcceptedKeyTypes ssh-rsa`

# SSH mit Zertifikat

- Schritte pro Client:
  - Erzeugen Public/Private Key Pair
    - Dabei Passphrase für spätere Zugriffe festlegen
  - Übertragen des Public Key an den Router
  - Auf dem Router den Key als berechtigten Client-Key eintragen
  - Testen – beim Login wird (grafisch) nach der vorher festgelegten Passphrase gefragt, es geht kein Passwort über die Leitung.
- Android Client: App „VX ConnectBot“
- IOS: App „Termius“





Cancel Edit Host Save

Alias Router Lokal

Hostname 192

Group >

Tags >

Backspace as CTRL+H

SSH

Use SSH

Port

Username

Password

Key RSA-00

Cancel New Key Save

Generate Paste Import

Key Name RSA-01

Passphrase

Type RSA >

Bits 2048 >

Cipher AES-128 >

Save passphrase

Specify the number of bits in the key to create. For RSA keys, the minimum size is 768 bits and the default is 2048 bits. Generally, 2048 bits is considered sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2.

# IOS App „Termius“

Cancel Save

Key Name RSA-00

Passphrase

Private

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEowIBAAKCAQEAt17MouFAEpi3sJ2SJPXJe27O9RpetHU/  
cpg3aCFfngVnUyNQ  
6MNCeOGI8I0bilEottVY4144Cw96jf1HcL/2J/QjiZhq9+cQgyjiJcdd+S/8x0vC  
kzvp77WqxiwcmnPzjy9/2WIOE9gTKM82rsuCYTzFrQA8aOPQN0fDqXK2HYDuE  
ovZ  
bJMUF/CGOEg4tvvGcl0EXP98nTOoux18syj9HKI107nd0CCgI9UdMP2D9aG/  
wiet
```

.....

9rc

```
NpuUuam7CaX0XXNII1+5AOEsGgNE70QfOSD1GRKZqLFDIYHaL/  
EILnZPuldTZynL  
NHhF1GzJdiqF4/x+l0zGik9YoTc9z82VWdkz7hJqv3hyn7lv4wj7  
-----END RSA PRIVATE KEY-----
```

Public

```
ssh-rsa  
AAAAB3NzaC1yc2EAAAADAQABAAQAC3Xsyi4UASmLewnZlk9cl7bs71GI60d  
T9ymDdolV+eBwdTI1Dow0J44YjwjRulGsi21VjjXjgLD3qN/Udww/  
Yn9COJmGr35xCDK0Ilx235L/zHS8KTO+nvtaRGLByac/OPL3/  
ZYg4T2BMozzauy4JhPMWtADxo49A3R8OpcrYdgo4Si/  
NskxQX8IY4SDi2+8ZyXQRc/3ydM6i7HXyzKP0cqXXTud3QIKAj1R0w/YP1ob/  
COy3Z2txlfbMnX4TYPNqNSAwG6K3Hw+oGJ8jHC2x7bZooqyTP1MADbqvpl5A  
7rlkelqIZgTtfZ1DbA5K3u1LR/RPUBus1gE1H4TRap+0H Generated By Termius
```

## OpenVPN (1/4)

- OpenVPN verwendet zertifikatsbasierte Authentifizierung
- Mein OpenWRT-Router (Turris Omnia) hat eine vereinfachte Konfiguration über die Web-Oberfläche
  - Vereinfachtes Installieren OpenVPN Paket
  - CA generieren (dauert etwas). Ergebnis im Verzeichnis `/etc/ssl/ca/openvpn`

```
-rw-r--r--      1 root      root      6704 Sep  3 00:32 01.crt
-rw-r--r--      1 root      root      1582 Sep  3 00:32 01.csr
-r-----       1 root      root      3268 Sep  3 00:32 01.key
-rw-r--r--      1 root      root      6704 Sep  3 00:32 01.pem
-rw-r--r--      1 root      root      1862 Sep  3 00:32 ca.crt
-r-----       1 root      root      3272 Sep  3 00:32 ca.key
```

## OpenVPN (2/4)

Es wird dann auf der gleichen Seite eine Konfiguration vorgeschlagen. Ich habe sie so gelassen und „applied“. Dabei geht es um die Networking-Aspekte.

### Previous settings

If you haven't tried to set up OpenVPN server on our router yet, you can safely proceed to "**Apply configuration**" button.

Otherwise if you've tried to set up OpenVPN outside this plugin, there is a chance that your configuration might collide with the configuration created by this plugin. Therefore you might need to disable the old configuration first.

Configuration enabled	<input checked="" type="checkbox"/>		<b>Current settings</b>
OpenVPN network	<input type="text" value="10.111.111.0/24"/>		Network: 10.111.111.0/24
All traffic through vpn	<input type="checkbox"/>		Device: tun_turris
Use DNS from vpn	<input checked="" type="checkbox"/>		Protocol: udp
<input type="button" value="Apply configuration"/>			Port: 1194
			Route: 192.168.1.1/24

Note that when you trigger "**Apply configuration**" button you might lose the connection to the router for a while. This means that you might need to reopen this admin page again.

## OpenVPN (3/4)

In der Konfigurationsdatei finden sich folgende, mit Zertifikaten befassten Zeilen:

```
option ca '/etc/ssl/ca/openvpn/ca.crt'  
option crl_verify '/etc/ssl/ca/openvpn/ca.crl'  
option cert '/etc/ssl/ca/openvpn/01.crt'  
option key '/etc/ssl/ca/openvpn/01.key'  
option dh '/etc/dhparam/dh-default.pem'
```

OpenVPN arbeitet nur auf Zertifikatsbasis. Unter Turris/Foris werden die Clientzertifikate auf dem Server erzeugt und dann - ggf. mit Nachbearbeitung - auf die Clients gebracht. Dateien pro Client:

```
-rw-r--r-- 1 root root 6911 Sep 3 01:04 02.crt  
-rw-r--r-- 1 root root 1582 Sep 3 01:04 02.csr  
-r----- 1 root root 3272 Sep 3 01:04 02.key  
-rw-r--r-- 1 root root 6911 Sep 3 01:04 02.pem
```

## OpenVPN (4/4)

Die mit „Get Config“ herunterladbare Datei enthält sowohl Konfigurationsparameter als auch den öffentlichen sowie den privaten Schlüssel. Das ist eher unüblich (unsicher?), erspart aber dem Client, einen privaten Schlüssel selbst erzeugen zu müssen.

### Client configuration

Here you can create and revoke the client capability to connect to your OpenVPN network.

Client name

Create

Client	Status		
Stylus2	active	Get Config	Revoke
ipad	active	Get Config	Revoke

Be sure to check, that the server IP address provided in you configuration file actually matches the public IP address of your router. You can set this address manually when the autodetection fails.

Router address

To apply the client configuration you need to download it and put it into the OpenVPN config directory or you might try to open it using your OpenVPN client directly. You might need to restart your client afterwards.

## DNSSEC (1/2)

Damit DNS-Abfragen als authentisch überprüft werden können, gibt es den Standard DNSSEC (Domain Name System Security Extensions), siehe RFC 3833.

Ein DNS-Teilnehmer kann verifizieren, dass die erhaltenen DNS-Zonendaten tatsächlich identisch sind mit denen, die der Ersteller der Zone autorisiert hat. DNSSEC wurde als Mittel gegen Cache Poisoning (Fälschen von DNS-Antworten) entwickelt. Es sichert die Übertragung von Resource Records durch digitale Signaturen ab. Eine Authentifizierung von Servern oder Clients findet nicht statt.

Vertraulichkeit ist bei DNSSEC nicht vorgesehen, DNS-Daten werden daher nicht verschlüsselt.

Seit 2010 ist DNSSEC auf allen 13 Rootservern eingeführt.

Die Verifikation muss nicht zwingend in jedem Endgerät passieren, sondern beim DNS-Resolver (i.d.R. beim Provider).

Es gibt für jede Zone einen Hauptschlüssel (KSK, Key Signing Key) und einen Arbeitsschlüssel (ZSK, Zone Signing Key). Ganz oben steht der Root-KSK. Näheres bei [Heise](#).

## DNSSEC (2/2)

Damit Nameserver möglichst wenig belastet werden, hat man sich für kurze krypto- grafische Schlüssel entschieden – mit der Folge, dass diese immer mal wieder getauscht werden müssen, damit Angreifer möglichst wenig Zeit zum Knacken bekommen.

Aktuell soll erstmals seit 2010 der Root-KSK getauscht werden. Dieser Tausch wurde jetzt von ICANN verschoben (vom 11.10.17 mindestens nach Q1 2018), weil Tests ergeben haben, dass viele Provider den neuen Schlüssel noch nicht eingespielt haben, und auch das Protokoll, was einen automatischen Wechsel ermöglichen würde, nicht implementiert haben. Eine bestimmte Resolver-Software (Name nicht bekannt) kann das überhaupt noch nicht.

# Qualifizierte elektronische Signatur

Beschreibung auf [Wikipedia](#).

Rechtlich gesehen wird eine QeS benötigt, wenn gesetzlich für einen Vorgang die Schriftform (nicht zu verwechseln mit der Textform) vorgeschrieben ist. Beispiel: Grundstücksgeschäfte.

Tief blicken lässt die offizielle Liste der Anbieter bei der [Bundesnetzagentur](#), insbesondere die Liste der ehemaligen Anbieter.

Wer die eID-Funktion des Personalausweises nutzt, braucht ein Lesegerät in der teuren Komfort-Ausführung. Das Zertifikat kostet jedes Jahr neu.

Hier ein [Erfahrungsbericht von 2014](#). Spricht Bände. Für Privatleute völlig uninteressant.

## Nützliche Links

- Grundsatzartikel Wikipedia zu Zertifikaten
  - [https://de.wikipedia.org/wiki/Digitales\\_Zertifikat](https://de.wikipedia.org/wiki/Digitales_Zertifikat)
- Grundsatzartikel Wikipedia zu Signaturen
  - [https://de.wikipedia.org/wiki/Digitale\\_Signatur](https://de.wikipedia.org/wiki/Digitale_Signatur)
-  Diese Verbindung ist nicht sicher
- Signaturrecht
  - <https://www.signaturrecht.de/>
- Pros und Cons
  - Blog einer Firma, die mit Zertifikaten Geld verdient ([Beispielbeitrag](#))
  - Dan Kaminski, Security-Experte ([Vortrag CCC 2009](#))



Diese Verbindung ist nicht sicher

Der Inhaber von [www.signaturrecht.de](http://www.signaturrecht.de) hat die Website nicht richtig konfiguriert. Firefox hat keine Verbindung mit dieser Website aufgebaut, um Ihre Informationen vor Diebstahl zu schützen.