

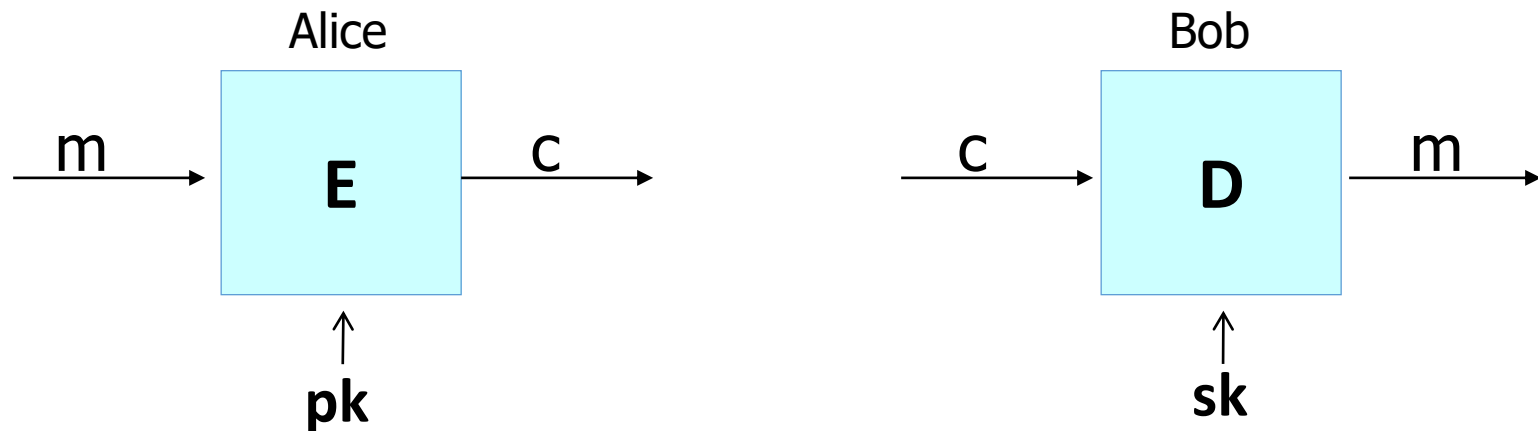
# IT-Sicherheit

## Public Key Verschlüsselung

Version vom 21.11.2017

# Public Key Verschlüsselung

- Bob erzeugt Schlüsselpaar (PK, SK)
- Bob überträgt PK an Alice



Typische Verwendung:

- sichere Übertragung von Schlüsseln für symmetrische Verschlüsselung
- signieren von Dokumenten (Verschlüsseln mit SK, jeder kann entschlüsseln und erkennen, mit wessen SK verschlüsselt wurde)

# Definition

Ein Public Key Cipher besteht aus drei Algorithmen (G, E, D):

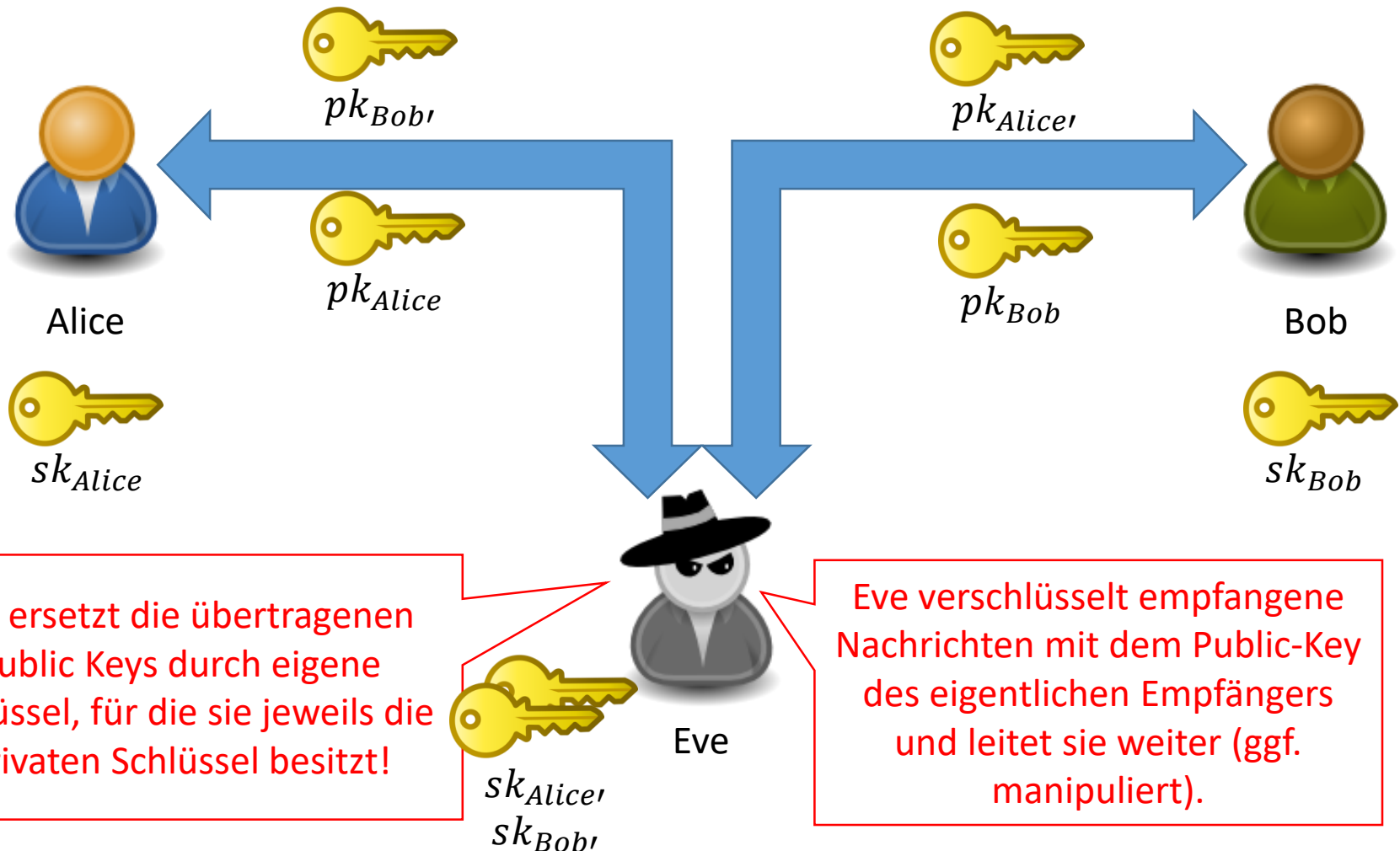
- $G()$   
randomisierter Algorithmus, der ein Schlüsselpaar (sk, pk) ausgibt.
- $E(pk, m)$   
randomisierter Algorithmus, der zu einem Klartext m einen Ciphertext c ausgibt
- $D(sk, m)$   
deterministischer Algorithmus, der zu einem Ciphertext c den Klartext m ausgibt (oder anzeigt, dass der Ciphertext nicht zu dem privaten Schlüssel passt)

Konsistenzbedingung: für alle von G erzeugten Schlüsselpaare und alle erlaubten Klartexte muss gelten:  $D(sk, E(pk, m)) = m$

# RSA: Rivest-Shamir-Adleman

- öffentlicher Schlüssel ist Zahlenpaar  $(e, N)$
- privater Schlüssel ist Zahlenpaar  $(d, N)$
- Algorithmus E:  $E(e, N, m) := m^e \bmod N = c$
- Algorithmus D:  $D(d, N, c) := c^d \bmod N = m$
- Schlüsselerzeugungsalgorithmus G:
  1. Wähle zwei große Primzahlen  $p$  und  $q$ . Dann ist  $N = pq$
  2. Wähle  $e$  teilerfremd zu  $(p - 1)(q - 1)$
  3. Ermittle  $d$ , so dass gilt  $e \cdot d \equiv 1 \pmod{(p - 1)(q - 1)}$   
(über erweiterten euklidischen Algorithmus)

# Problem: Eigentümerprüfung bei Schlüsselpaaren



# Schutz öffentlicher Schlüssel

## Certificate Authorities

- Durch digitale Unterschriften beglaubigen Certificate Authorities den Eigentümer öffentlicher Schlüssel.
- Den Certificate Authorities wird global vertraut.

Alle Browser und verbreiteten Betriebssysteme haben Listen mit vertrauenswürdigen CAs.

## Web of Trust

- Durch direkte Vorlage eines Identitätsnachweises beglaubigt ein Teilnehmer des Systems die Identität eines anderen.
- Falls einem Teilnehmer vertraut wird, dann wird auch dessen Beglaubigungen vertraut.
- Bis zu einer bestimmten Ebene ist das Vertrauen transitiv.

Bekannteste Implementierungen:  
PGP, GnuPG

# Digitale Unterschrift

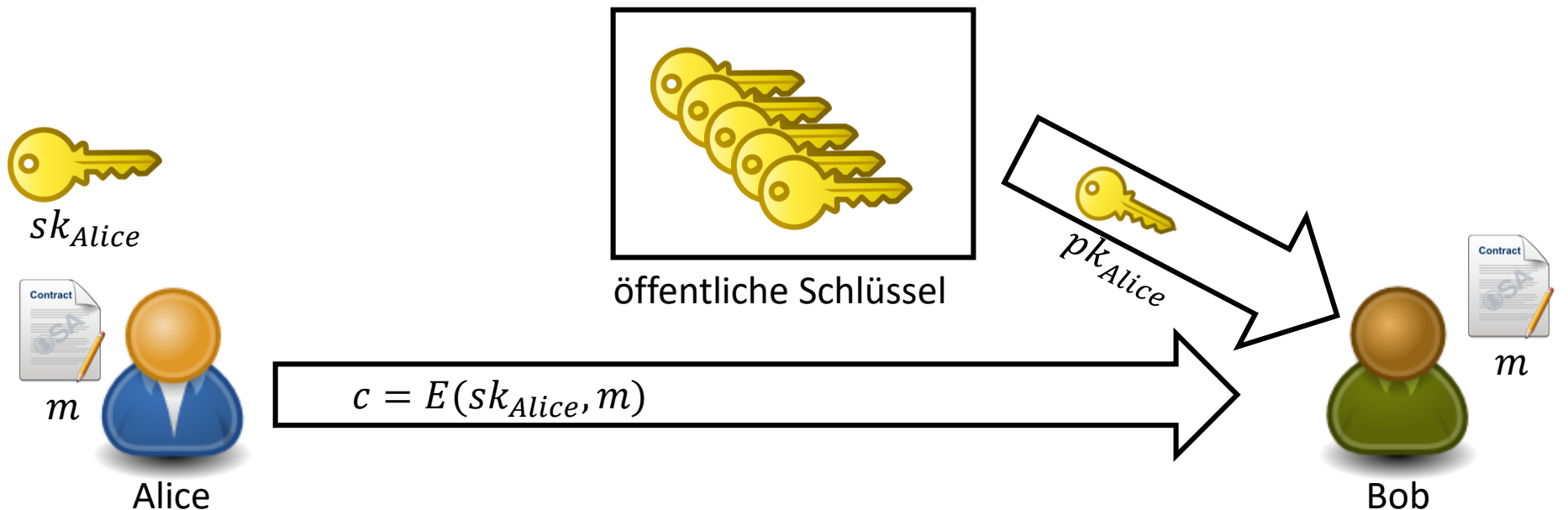
- Ziele und Anforderungen

1. Jeder kann das Dokument lesen. Vertraulichkeit ist keine Anforderung an die digitale Signatur.
2. Niemand kann das Dokument unbemerkt verändern.
3. Jeder kann den Ersteller des Dokuments identifizieren.



# Elektronische Unterschrift (Beispiel RSA)

Der Verschlüsselungsmechanismus mit öffentlichen/privaten Schlüsseln kann eine elektronische Unterschrift erzeugen.

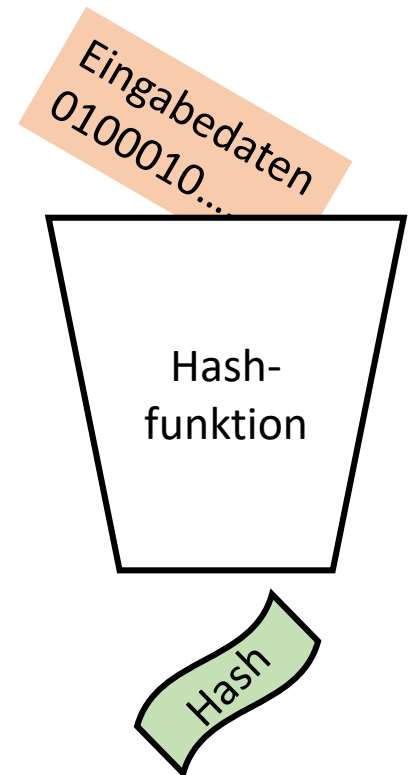


1. Alice verschlüsselt die Nachricht mit ihrem **privaten** Schlüssel  $sk_{Alice}$
2. Bob besorgt sich den zugehörigen **öffentlichen** Schlüssel  $pk_{Alice}$  und verifiziert, dass die Nachricht nur von Alice kommen kann.

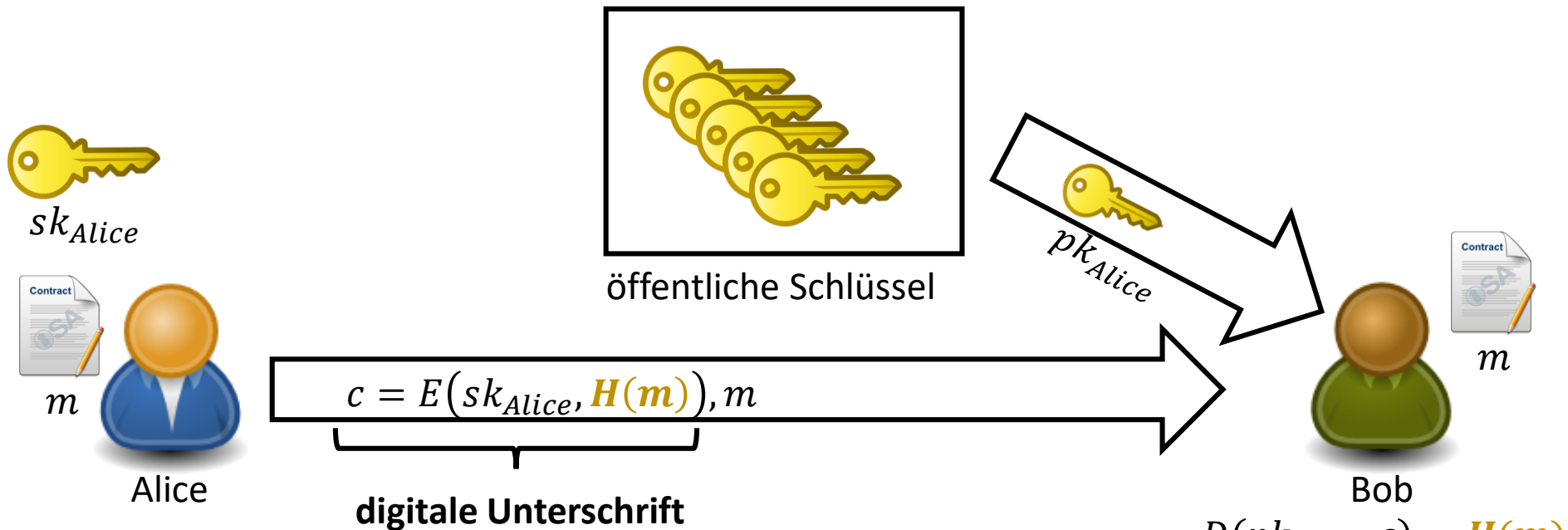


# Hashfunktionen

- Vertraulichkeit der Nachricht ist bei der elektronischen Unterschrift nicht gefordert → Die Verschlüsselung der gesamten Nachricht ist nicht notwendig.
- Vor der Verschlüsselung wird die Nachricht mit Hilfe einer **Hashfunktion** auf eine feste Länge reduziert.
  - Hashfunktionen sind Funktionen, die zu einer Eingabe beliebiger Länge einen charakteristischen Wert fester Länge zurückliefern.
  - Für sichere Hashfunktionen gilt, dass
    1. für einen bekannten Hashwert kein Algorithmus existiert, der dazu passende Eingabedaten erzeugt und schneller ist als Brute-Force.
    2. es keinen Algorithmus gibt, der zwei verschiedene Eingabedaten mit gleichem Hashwert erzeugen kann und schneller ist als Brute-Force.



# Elektronische Unterschrift (revisited)



1. Alice bildet den Hashwert der Nachricht und verschlüsselt ihn mit ihrem **privaten** Schlüssel  $sk_{Alice}$ . Das Ergebnis überträgt sie **mit der Nachricht** an Bob.
2. Bob berechnet den Hashwert der empfangenen Nachricht, besorgt sich den zugehörigen **öffentlichen** Schlüssel  $pk_{Alice}$  und verifiziert, dass die Nachricht nur von Alice kommen kann.

# Trusted Third-Party Konzept: Zertifizierungsstellen

- Situation
  - Jeder Teilnehmer vertraut einer oder mehreren zentralen Zertifizierungsstellen.
  - Der Schlüssel eines Teilnehmers wird von genau einer Zertifizierungsstelle beglaubigt.  
→ Das Ergebnis ist ein Zertifikat.
- Vertrauen bedeutet
  - Jeder Teilnehmer hat den public Key der Zertifizierungsstelle vertrauenswürdig erhalten und vertraut der Stelle, dass sie ordentliche Zertifikate ausstellt.
  - Der Owner Trust in vertraute Zertifizierungsstellen ist immer voll, er muss nicht weiter berücksichtigt werden.

Bezeichnung für das Zertifikat von Bob:  $CA_X\langle B \rangle$  oder  $CA_X\langle B, K_B \rangle$   
Die Zertifizierungsstelle  $CA_X$  beglaubigt den Public Key  $K_B$  von Bob.

# Bestandteile einer Public Key Infrastruktur

- **Zertifikate**
  - binden mit einer digitalen Signatur öffentliche Schlüssel untrennbar an ihren Eigentümer
- **Zertifizierungsstelle**
  - ist ein besonders gut geschütztes System, mit dessen privatem Schlüssel geprüfte Zertifizierungsanträge unterschrieben werden
- **Registrierungsstelle (RA – Registration Authority)**
  - authentifiziert die Antragsteller und prüft die Antragsberechtigung
  - erzeugt die Schlüsselpaare (oder erhält den öffentlichen Schlüssel vom Antragsteller authentisch)
  - übergibt geprüfte Anträge an die CA zur Unterzeichnung
- **Validierungs- und Rückrufdienst**
  - stellt regelmäßig Informationen zur Gültigkeit der ausgestellten Zertifikate zur Verfügung
- **Verzeichnisdienst**
  - stellt die fertigen Zertifikate öffentlich (oder einer Nutzergruppe) zur Verfügung

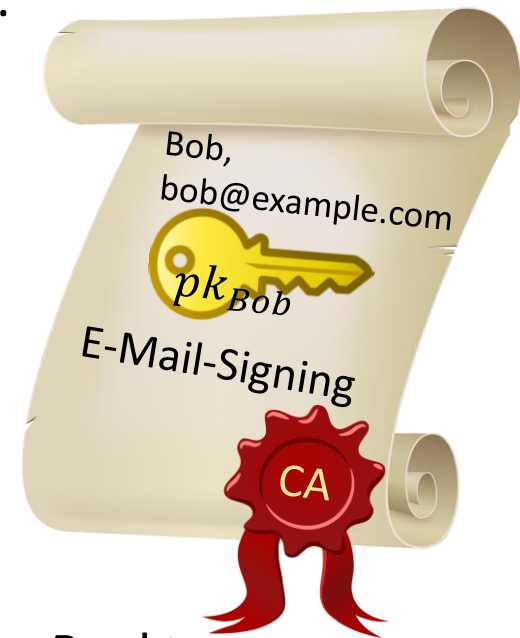
# Zertifikate

Zertifizierungsstellen (Certificate Authorities, CAs) erstellen Beglaubigungen in Form von X.509 Zertifikaten.

Die wesentlichen Inhalte sind:

- **Name des Eigentümers**
- **Public Key** des Eigentümers
- Name der **ausstellenden CA**
- **Gültigkeitsdauer**
- mögliche **Verwendungszwecke**
- (optional) weitere **Attribute**
- digitale **Signatur des Ausstellers** unter all diese Punkte

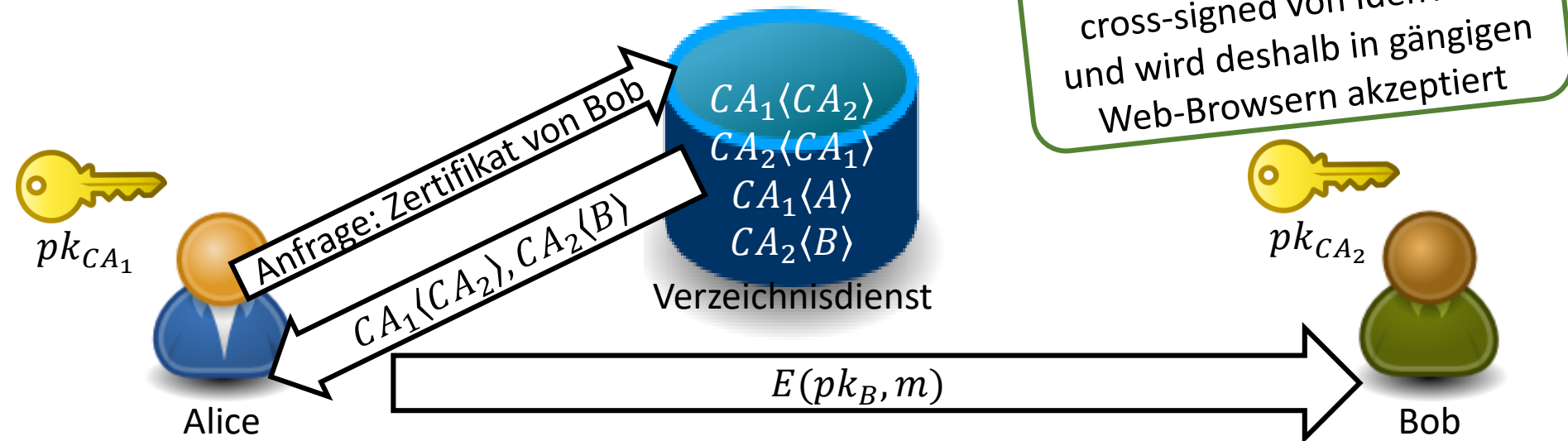
Beispiele für Verwendungszwecke sind E-Mail-Signatur, E-Mail-Verschlüsselung, SSL Server, Code-Signing, Erstellung von Zertifikaten (ist selbst eine CA)



# Cross Zertifikate

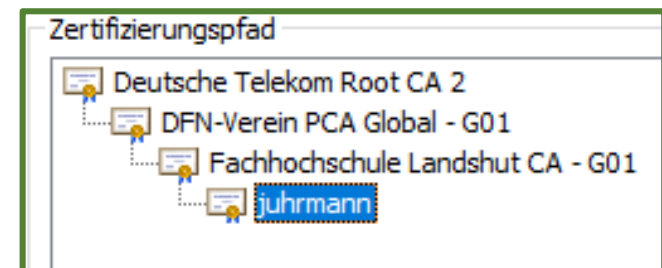
- Möglicherweise sind zwei Teilnehmer (Alice, Bob) bei verschiedenen Zertifizierungsstellen zertifiziert.
- Sie können in diesem Fall sicher kommunizieren, wenn sich beide CAs gegenseitig vertrauen, d.h. sich gegenseitig zertifiziert haben
- $CA_1\langle CA_2 \rangle$  und  $CA_2\langle CA_1 \rangle$  sind **Cross Zertifikate**

Beispiel: Let's encrypt ist cross-signed von IdenTrust und wird deshalb in gängigen Web-Browsern akzeptiert



# CA Hierarchien

- Das Vertrauen von CAs ist transitiv:  
Vertraut  $CA_1$  der  $CA_2$  und vertraut  $CA_2$  der  $CA_3$ , dann vertraut auch  $CA_1$  der  $CA_3$ .
- allgemein ist die Zertifikatskette  $CA_1\langle CA_2\rangle, CA_2\langle CA_3\rangle, \dots, CA_n\langle B\rangle$  äquivalent zu dem einzelnen Zertifikat  $CA_1\langle B\rangle$
- ➔ Wenn Alice  $pk_{CA_1}$  besitzt und die Zertifikatskette, dann kann sie auch  $pk_B$  vertrauen
- ➔ Es können Hierarchien aufgebaut werden: CAs lassen ihre Schlüssel von übergeordneten CAs beglaubigen.
- ➔ An der Spitze der Hierarchie steht eine Root-CA, die nicht mehr beglaubigt ist.



# Speicherung von Zertifikaten

- Self-Signed Certificates

- Public Keys von Root-CAs werden meist im Format eines Zertifikats aufbewahrt.
- Dieses Zertifikat wird von der CA selbst signiert (Subject = Issuer)
- **ACHTUNG: Selbstsignatur hat keine Sicherheitsrelevanz und kann aus beliebiger Quelle kommen!**

Aussteller	Deutsche Telekom Root CA 2, ...
Gültig ab	Freitag, 9. Juli 1999 13:11:00
Gültig bis	Mittwoch, 10. Juli 2019 00:59:00
Antragsteller	Deutsche Telekom Root CA 2, ...
Öffentlicher Schlüssel	RSA (2048 Bits)
Parameter für öffentlichen ...	05 00
Schlüsselkennung des Antra	31 c3 79 1b ba f5 53 d7 17 e0

- Zertifikatsspeicher

- In Anwendungen und Betriebssystemen werden Zertifikate in Zertifikatsspeichern untergebracht. Diese enthalten i.d.R. eigene Schlüsselpaare (privater Schlüssel ist meist passwortgeschützt), Nutzerzertifikate anderer Personen, Zertifikate von Zwischenzertifizierungsstellen, Root-Zertifikate
- Zertifikate mit eigenen Schlüsselpaaren können auf spezieller Hardware gespeichert werden. Diese Hardware kann ggf. das Schlüsselpaar direkt erzeugen → Auslesen des privaten Schlüssels ist nicht möglich.

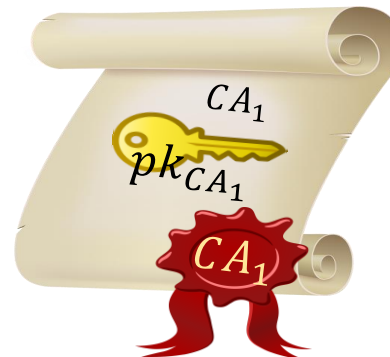
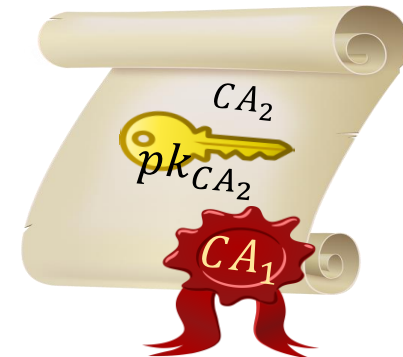
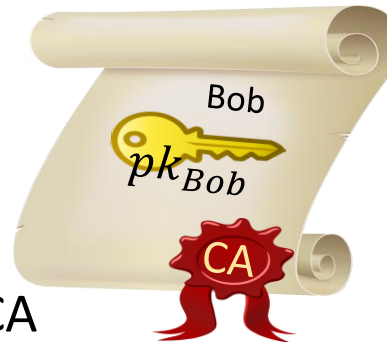




# Begriff „Zertifikat“

Der Begriff „Zertifikat“ wird unterschiedlich verwendet:

1. eigene Zertifikate  
öffentlicher Schlüssel ist signiert  
privater Schlüssel ist verschlüsselt gespeichert
2. andere End-Zertifikate  
signiert von einer CA  
privater Schlüssel fehlt
3. Intermediate CA  
signiert von einer anderen CA  
privater Schlüssel fehlt
4. Trusted Root CA  
selbstsigniert  
privater Schlüssel fehlt  
**Signatur nicht prüfbar!**

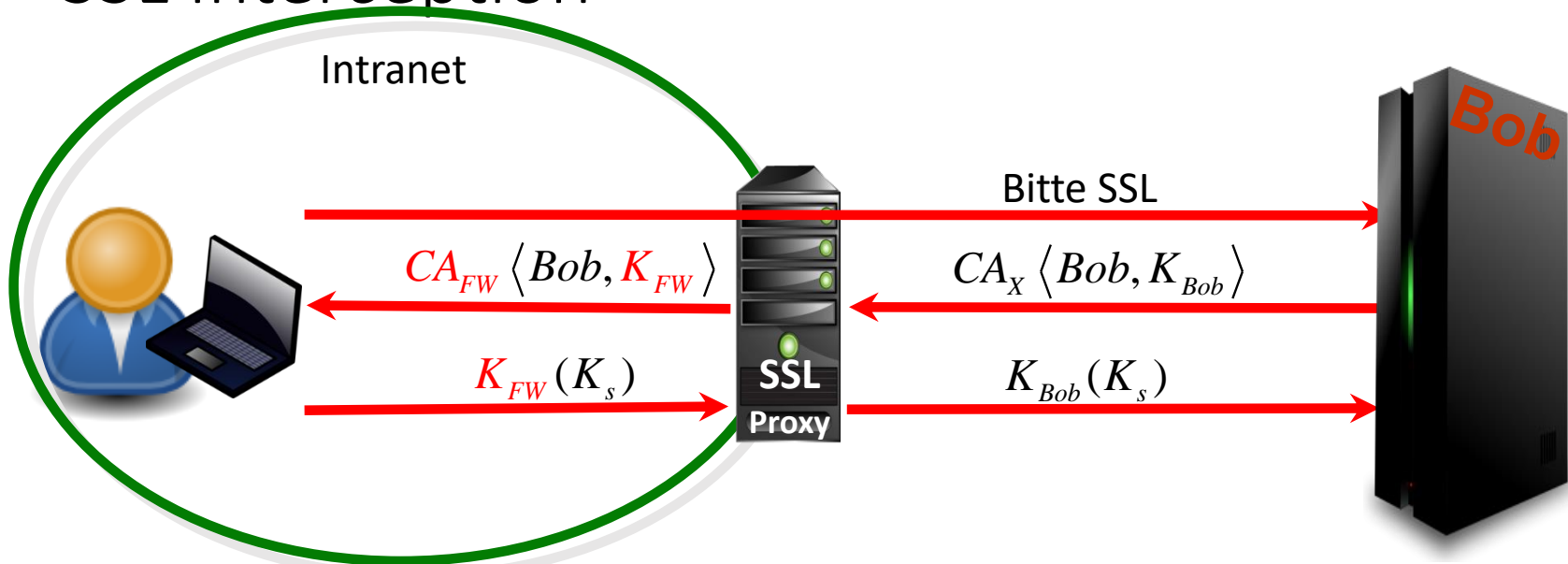


# Gültigkeitsdauer von Zertifikaten

- Die typische Gültigkeitsdauer von Zertifikaten liegt im Bereich von Jahren
  - Nutzerzertifikate: ca. 1 – 3 Jahre
  - Serverzertifikate: ca. 3 – 5 Jahre
  - Rootzertifikate: ca. 10 – 30 Jahre
  - Vollautomatisch ausgestellte/erneuerte Zertifikate (z.B. von Let's Encrypt) haben eine typische Gültigkeitsdauer von mehreren Monaten.
- Für manche Zwecke werden kurzlebige Zertifikate verwendet
  - für Single-Sign-On stellt der Authentifizierungsserver an authentifizierte Personen kurzlebige Zertifikate aus (ca. 1 Stunde bis 1 Tag), damit kann sich der Eigentümer ggü. Servern im Bereich des Authentifizierungsservers legitimieren
  - Ein SSL-Proxy in einer Firewall kann eine SSL-Verbindung über die Firewall unterbrechen und selbst ein Serverzertifikat ausstellen. Gültigkeitsdauer typ. im Bereich von Minuten.

kurzlebige Zertifikate werden  
nicht von regulären CAs  
ausgestellt

# SSL Interception



- Alice möchte mit Bob eine SSL Verbindung aufbauen (über die Firewallgrenze hinweg)
- Bob sendet sein echtes Zertifikat an Alice.
- Der SSL-Proxy ersetzt dieses durch ein kurzlebiges Zertifikat mit Bobs Namen und einem eigenen Schlüssel (eigentlich eine Fälschung)
- Alice sendet  $K_s$  an Bob, verschlüsselt mit dem FW-Key (sollte eigentlich für Bob verschlüsselt sein!)
- Der Proxy entschlüsselt  $K_s$  und sendet ihn weiter, jetzt verschlüsselt für Bob
- Der Proxy kann die mit  $K_s$  geschützte SSL-Verbindung zwischen Alice und Bob mitlesen.  
Er ist ein **Trusted Man in the Middle**.

# Trusted Man in the Middle

- Voraussetzungen
  - Mitarbeiter/Betriebsrat müssen informiert sein und zustimmen.
  - Der Public Key der „Firewall CA“ darf nur in die Rechner im Intranet importiert werden!
  - **Niemals darf die „Firewall CA“ von einer Root-CA zertifiziert sein, die in Browsern/Betriebssystemen verankert ist!**
- Risiken
  - Der Datenschutz wird verletzt. Der Proxy kann verschlüsselte Verbindungen mitlesen, die nicht für ihn bestimmt sind.
  - Mit der Firewall-CA können **Zertifikate für alle Domänen** der Welt ausgestellt werden.
  - **Client kann nicht mehr selbst bestimmen, welcher CA er vertraut!**
- Nutzen
  - Prüfen des Netzwerkverkehrs (Virencheck, Kommunikation von Schadsoftware, Upload von Firmengeheimnissen, etc.)
  - Ende-zu-Ende-Verschlüsselung widerspricht dem Konzept von Firewalls

# Security Policy

- CAs sind für alle Kommunikationspartner vertrauenswürdige Parteien
  - Um das Vertrauen herzustellen muss jede CA eine **Security Policy** erstellen.
  - Diese enthält die Regeln nach denen die CA Zertifikate erstellt
  - Ein Internet Standard (rfc3647) beschreibt die notwendigen Inhalte in zwei Dokumenten
    - **CP (Certificate Policy)**: wird von der Root CA erstellt, gilt für die Root CA und für die gesamte CA-Hierarchie unterhalb dieser Root.
    - **CPS (Certificate Practice Statement)**: wird von jeder CA erstellt und beschreibt wie in der CA die Regeln der CP umgesetzt werden.
- Wesentliche Inhalte der CP und des CPS
  - Die Architektur der Zertifizierungsstelle.
  - Die Verwendung der Zertifikate.
  - Wie findet die Identifikation und die Authentifizierung der Antragsteller statt?
  - Wo und wie werden die Schlüssel erzeugt, verteilt und aufbewahrt.
  - Wie ist die Ablauforganisation (Anträge, Ausstellung, Erneuerung, Widerruf...)
  - Welche organisatorischen und technischen Sicherheitsmaßnahmen werden durchgeführt?

# Extended Validation Zertifikate

- 2007 vom CA-Browser Forum entwickelt
  - Das CAB-Forum ist ein Zusammenschluss der wichtigsten Browser Hersteller und CA-Anbieter
- Ziel: mehr Vertrauen in Zertifikate schaffen durch gründlichere Überprüfung der Antragsteller
  - Überprüft wird:
    - Identität
    - Geschäftsadresse
    - Rechtsstatus
    - Eigentum an der Domäne
    - Zeichnungsberechtigung der antragstellenden Person (Prokura)
  - Nicht überprüft wird:
    - Betreibt der Antragsteller eine anständige Firma
- Im Browser werden EV-Zertifikate in der Adresszeile gekennzeichnet (andere Farbe oder Einblendung des Namens)



# Widerruf von Zertifikaten

- Revocation von Zertifikaten
  - Zertifikate müssen für ungültig erklärt werden, z.B. wenn sie kompromittiert werden (Revocation)
  - Die CA muss Ihren Anwendern den Status der ausgestellten Zertifikate zur Verfügung stellen
- CRL (Certificate Revocation List)
  - Eine Liste mit Seriennummern widerrufenen Zertifikate, digital unterschrieben von der CA
  - Diese wird von der CA regelmäßig erstellt und steht zum Download bereit.
  - Nutzer müssen diese herunterladen und in ihren Anwendungen installieren
- OCSP (Online Certificate Status Protocol)
  - Erlaubt eine Realzeitabfrage nach dem Status eines Zertifikats
  - Die CA betreibt dazu einen OCSP-Responder. Die URL steht meistens im Zertifikat
  - Die Anfrage nach der Gültigkeit des Zertifikats X hat eine vom Responder signierte Antwort zur Folge der Form „Zertifikat X ist gültig/revokiert/Status unbekannt“

# OCSP versus CRL

- CRL
  - Vorteile: Geringe Server Belastung, Verwendung ist offline möglich
  - Nachteile: Nicht immer ganz aktuell, der Anfragende erfährt eigentlich zu viel (ganze Liste), die Listen sind manchmal sehr groß.
  - CRL wird immer weniger verwendet.
- OCSP
  - Vorteil: Die Response erfolgt in Realzeit, aktueller Status ist garantiert
  - Nachteile: Online Verbindung ist nötig, Die Responder sind stark belastet (DFN 2013 ca. 1,4 Mio. Responses pro Tag). CA erfährt IP und Verbindungswunsch des Anfragenden.
  - OCSP wird von vielen Anwendungen mangelhaft integriert (bleibt die Response aus, wird das Zertifikat in der Regel als gültig erklärt).
  - EV-Zertifikate erfordern OCSP-Bestätigung.
- TLS 1.2 erlaubt **OCSP-Stapling**
  - Der Server dessen Zertifikat der Client überprüfen will stellt regelmäßig, (z.B. einmal täglich) eine OCSP-Anfrage für sich selbst.
  - Beim Aufbau einer TLS Verbindung wird diese Response zusammen mit dem Zertifikat an den Client geschickt.



# Die SSL Krise

- SSL/TLS (https) ist der Hauptanwender für Zertifikate
  - Auch wenn das Protokoll schon alt ist, werden immer wieder Fehler gefunden (z.B. Heartbleed 2014, Logjam 2015), die verwendeten Algorithmen und Versionen müssen regelmäßig überprüft werden.
  - Das Protokoll selbst ist ausgereift, bei Versionen ohne bekannte Fehler kaum zu brechen.
- Angreifer sucht sich schwächere Glieder in der Kette: Er greift CAs an.
  - Comodo 2011
    - Accounts von RA Mitarbeitern wurden gehackt und damit unzulässige Zertifikate erstellt (u.a. mail.google.com)
  - Diginotar 2011
    - Die CA hatte einen remote Admin Zugang (!), dieser wurde gehackt. Damit wurden 531 Zertifikate gefälscht, meist für soziale Netzwerke, aber auch für \*.\*.com
    - Diginotar ist inzwischen insolvent
  - Türktrust 2012
    - Türktrust verkauft zwei CA-Zertifikate (für SubCAs), die beliebige Zertifikate ausstellen können.
    - Eines davon wird als SSL-Proxy verwendet und stellt weltweit gültige Zertifikate z.B. für Google aus.
- Welchen CAs kann ich trauen?
  - In den Browsern/Betriebssystemen sind über 100 Root-CAs aus allen möglichen Ländern der Erde verankert (u.a. in China, Deutschland, Finland, Litauen, Spanien, Südafrika, Taiwan, Türkei, Ungarn, USA).
  - Alle sind im Browser gleich vertrauenswürdig. Haben Sie alle das gleiche Vertrauen verdient?

Lenovo Superfish Adware ... X +

https://www.us-cert.gov

lenovo certificate

Official website of the Department of Homeland Security

**US-CERT**  
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

HOME ABOUT US PUBLICATIONS ALERTS AND TIPS RELATED RESOURCES C'VP

**Alert (TA15-051A)** [More Alerts](#)

**Lenovo Superfish Adware Vulnerable to HTTPS Spoofing**

Original release date: February 20, 2015 | Last revised: February 24, 2015

Print Tweet Send Share

**Systems Affected**

Lenovo consumer PCs that have Superfish VisualDiscovery installed.

**Overview**

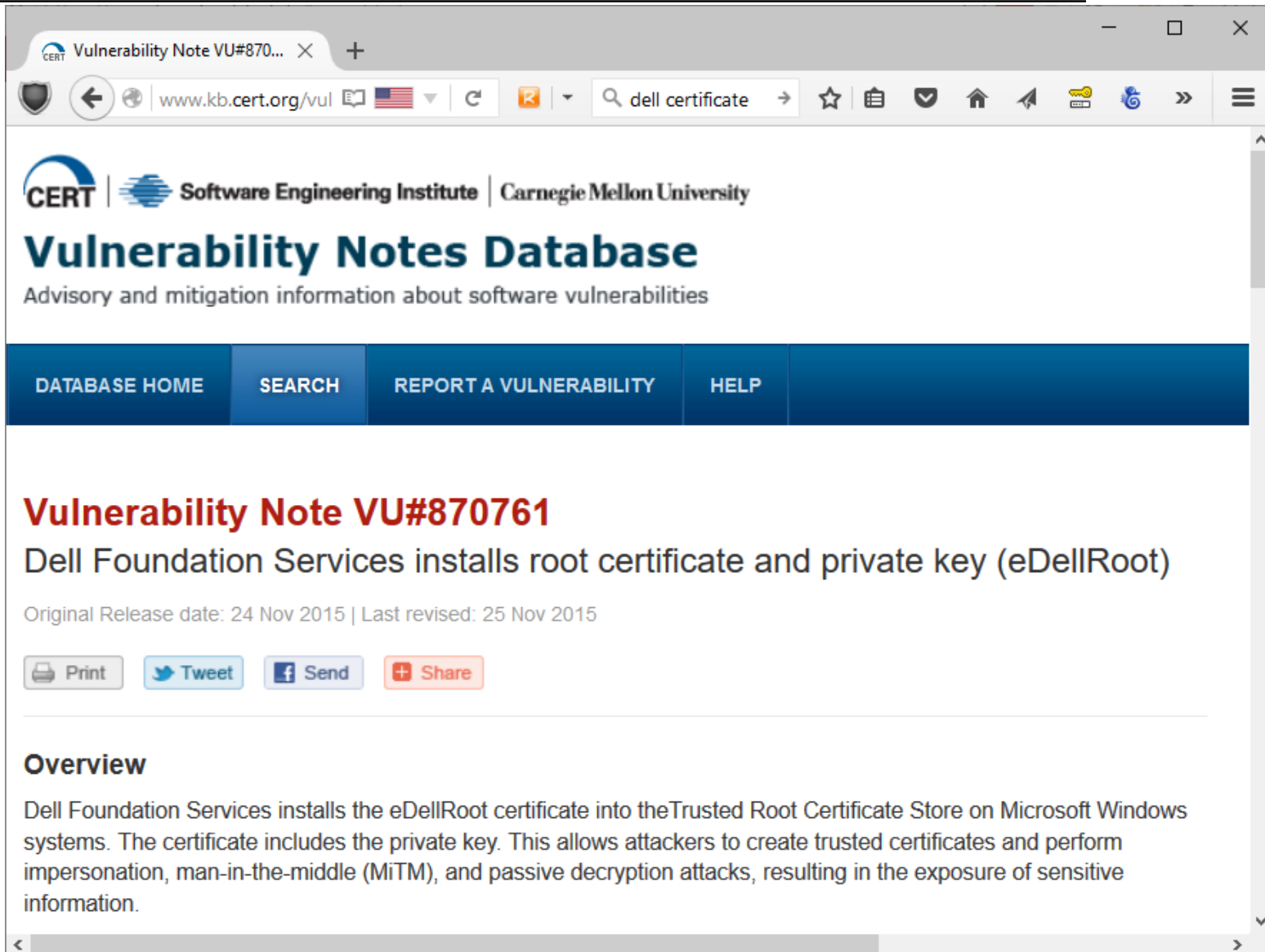
Superfish adware installed on some Lenovo PCs install a non-unique trusted root certification authority (CA) certificate, allowing an attacker to spoof HTTPS traffic.

**Description**

Starting in September 2014, Lenovo pre-installed Superfish VisualDiscovery on some PCs to display targeted advertisements. In order to intercept encrypted connections, browser-based encrypted traffic to the Internet is intercepted, decrypted, and replaced with targeted advertisements. **the software installs a trusted root CA certificate for Superfish.**

**the private key can easily be recovered from the Superfish software**

Although [Lenovo has stated](#) they have discontinued the practice of pre-installing Superfish VisualDiscovery, the systems that came with the software already installed will continue to be vulnerable until corrective actions have been taken.



The screenshot shows a web browser window displaying the CERT Software Engineering Institute Vulnerability Notes Database. The page title is "Vulnerability Note VU#870761" and the subtitle is "Dell Foundation Services installs root certificate and private key (eDellRoot)". The page includes a navigation bar with links to "DATABASE HOME", "SEARCH", "REPORT A VULNERABILITY", and "HELP". Below the navigation bar, the vulnerability note is displayed, including the original release date (24 Nov 2015) and the last revised date (25 Nov 2015). The note is titled "Vulnerability Note VU#870761" and "Dell Foundation Services installs root certificate and private key (eDellRoot)". The overview section states: "Dell Foundation Services installs the eDellRoot certificate into the Trusted Root Certificate Store on Microsoft Windows systems. The certificate includes the private key. This allows attackers to create trusted certificates and perform impersonation, man-in-the-middle (MiTM), and passive decryption attacks, resulting in the exposure of sensitive information."

Vulnerability Note VU#870... X +

www.kb.cert.org/vul

CERT Software Engineering Institute | Carnegie Mellon University

## Vulnerability Notes Database

Advisory and mitigation information about software vulnerabilities

[DATABASE HOME](#) [SEARCH](#) [REPORT A VULNERABILITY](#) [HELP](#)

### Vulnerability Note VU#870761

#### Dell Foundation Services installs root certificate and private key (eDellRoot)

Original Release date: 24 Nov 2015 | Last revised: 25 Nov 2015

[Print](#) [Tweet](#) [Send](#) [Share](#)

#### Overview

Dell Foundation Services installs the eDellRoot certificate into the Trusted Root Certificate Store on Microsoft Windows systems. The certificate includes the private key. This allows attackers to create trusted certificates and perform impersonation, man-in-the-middle (MiTM), and passive decryption attacks, resulting in the exposure of sensitive information.

# Lösungsansätze

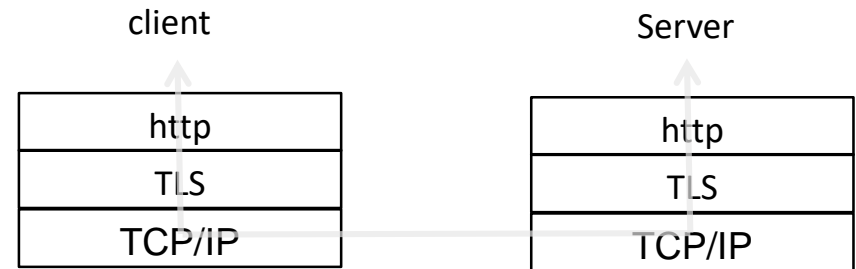
- Gehackte Zertifikate revokieren
  - Aber Zertifikatsprüfung im Browser ist oft mangelhaft, für den Benutzer kaum zu handeln
- Updates der Browser
  - Kurzfristig: Browser Patches mit Blacklists
  - CA-Pinning: von häufig verwendeten Diensten (z.B. Google) wird der Hashwert des bekannten Ausstellerzertifikats gespeichert (gepinnt). Kommt ein Zertifikat von einem falschen Aussteller, so wird das erkannt.
  - Pinning mittlerweile in RFC7469 standardisiert, aber kaum verwendet.
- CA/Browser – Forum Aktivitäten
  - 2012: **Baseline Requirements** für Zertifikate, die in Browsern/Betriebssystemen verankert sind:
    - Lebensdauer von Server Zertifikaten maximal 5 Jahre
    - Antragsteller müssen nachweisen, dass sie die Domäne kontrollieren, für die ein Zertifikat beantragt wird.
    - Die CA muss ein mindestens jährliches Audit von einer unabhängigen Organisation durchführen lassen
    - OCSP-Responder Adressen müssen in den Zertifikaten stehen.
- EU-Regulierung von Zertifizierungsdiensten ist geplant
  - Im Wesentlichen: CAs müssen für den Missbrauch ausgestellter Zertifikate haften

## • Was ist SSL/TLS

- SSL/TLS ist eine Protokollschicht die zwischen das Transportprotokoll und Internetanwendungen gesteckt wird und nach Aushandlung eines Schlüssels die Daten zwischen IP-Quelle und IP-Ziel verschlüsselt.
- Voraussetzung: eine existierende X.509 Public Key Infrastruktur (PKI) .
- Häufig werden eigene Ports für die mit SSL erweiterten Anwendungen gewählt:
  - https: port 443, imaps:port 993 Idaps:port 636, ...
- Alternativ kann häufig bei Verwendung des Standardports mit dem Kommando „STARTTLS“ TLS aktiviert werden.

## • Geschichte

- SSL (Secure Socket Layer)
  - Entwickelt von Netscape 1994 (SSL 1.0)
- TLS (Transport Layer Security)
  - Aufbauend auf SSL 3.0 von Netscape wurde 1999 ein IETF Standard entwickelt: TLS 1.0
  - 2006: Version 1.1
  - 2008: Version 1.2
  - 2017: Version 1.3 („working draft“)



# Arbeitsweise von TLS

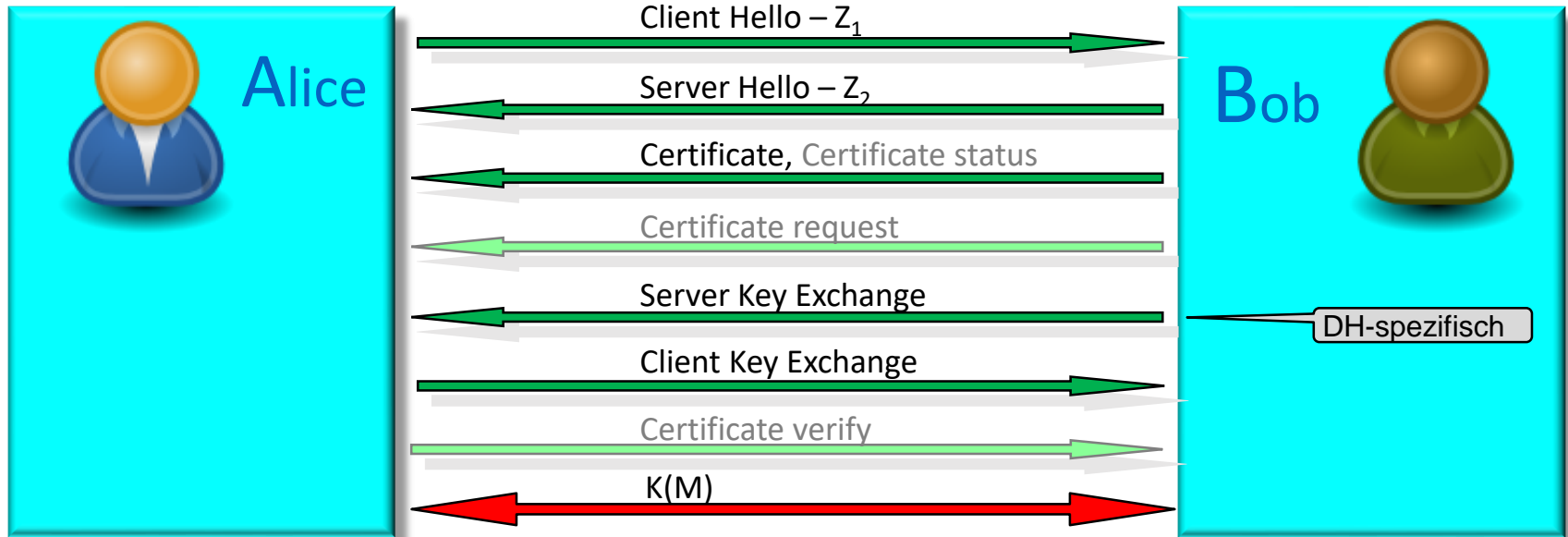
- Das Handshake Protokoll

- Beim Verbindungsaufbau wird eine „**Ciphersuite**“ zwischen Client und Server ausgehandelt
- Jede Ciphersuite enthält Algorithmen
  - Zum Schlüsselaustausch/Schlüsselvereinbarung
  - Zur digitalen Signatur
  - Zur Nutzdaten-Verschlüsselung, inklusive Verschlüsselungsmodus
  - Zum Hashen der Daten die signiert werden müssen.
- Im Standard sind ca. 50 Cipher Suites festgelegt, diese werden durch einen ASCII-String gekennzeichnet.
- In jeder Version gibt es mindestens eine „MUST“ Ciphersuite, die jede TLS Implementierung unterstützen muss.
  - Dadurch wird gewährleistet, dass alle TLS fähigen Dienste miteinander verschlüsselt sprechen können.

- Verschlüsselte Kommunikation

- Ergebnis des Handshake Protokolls
  - Die beiden Partner kennen einen gemeinsamen symmetrischen Session Key
  - Alle weiteren Nachrichten werden damit verschlüsselt
  - Der Server hat sich beim Client authentifiziert

# Handshake Protokoll im Detail, am Beispiel TLS\_ECDHE\_RSA\_WITH\_AES256\_GCM\_SHA256



- **Client Hello**: Von Alice unterstützte Ciphersuites und nonce  $Z_1$
- **Server Hello**: eine von Bob ausgewählte Ciphersuite und nonce  $Z_2$
- **Certificate**: Bob's Zertifikat (mit Bob's RSA Key) Certificate status (optional, TLS 1.2: OCSP-Status)
- **Certificate Request**: Anforderung des Client Zertifikats, falls Client Authentifizierung gewünscht
- **Server Key Exchange**: Bobs public DH-Key, unterschrieben von Bob mit seinem privaten RSA Key
- **Client Key exchange**: Alices public DH-Key, unsigniert

optional

optional

- **Certificate verify**: Alices RSA Zertifikat, mit Signatur über bisherigen Austausch (insbes.  $Z_1, Z_2$ )

Alice verifiziert Bobs DH Key. Alice und Bob erzeugen mit DH ein  $K_0$  (PreMasterSecret) und aus  $K_0, Z_1, Z_2$  den AES-Schlüssel  $K$  zur Nutzdatenverschlüsselung. Bob ist authentifiziert!

optional

Bob verifiziert die Unterschrift die er von Alice erhalten hat – damit ist Alice bei Bob authentifiziert