



# IT-Sicherheit

## 05-Hardware-Sicherheitsmodule

**Gerrit Kalkbrenner**

**[Gerrit.Kalkbrenner@hwr-berlin.de](mailto:Gerrit.Kalkbrenner@hwr-berlin.de)**

**Teile: Norbert Pohlmann**



# Hardware-Sicherheitsmodule

## Inhalt

- Ziele und Ergebnisse der Vorlesung
- Idee eines Hardware-Sicherheitsmoduls
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- Zusammenfassung



# Hardware-Sicherheitsmodule

## Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- Idee eines HSM
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- Zusammenfassung



## Ziele und Ergebnisse der Vorlesung

- Hardware-Sicherheitsmodule
  - Gutes **Verständnis** zu der Bedeutung von **Hardware-Sicherheitsmodule** im Bereich der Cyber-Sicherheit.
  - **Profundes Wissen** über die verschiedenen und aktuellen **Hardware-Sicherheitsmodule**.
  - Erlangen der **Kenntnisse** über prinzipielle Hardware-Sicherheitsmodule und zur **Umsetzung von konkreten Lösungen**.





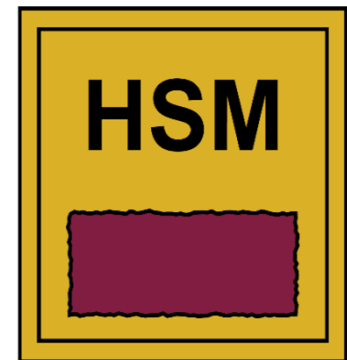
- Inhalt

## Hardware-Sicherheitsmodule

- Ziele und Ergebnisse der Vorlesung
- **Idee eines HSM**
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- Zusammenfassung

## Idee eines HSM

- Schutz vor Auslesen und Manipulation von sicherheitsrelevanten Informationen innerhalb eines geschützten Bereiches, meist Hardware
- Sicherheitsrelevante Informationen sind:
  - Geheime Schlüssel  
(für Verschlüsselung, Authentisierung, Signaturen, ..)
  - Programme  
(die nicht kopiert oder modifiziert werden dürfen)
  - Daten  
(z.B. Transaktionsdaten, die Werte darstellen)





- Inhalt

## Hardware-Sicherheitsmodule

- Ziele und Ergebnisse der Vorlesung
- Idee eines HSM
- **HSM: Smartcards**
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- Zusammenfassung

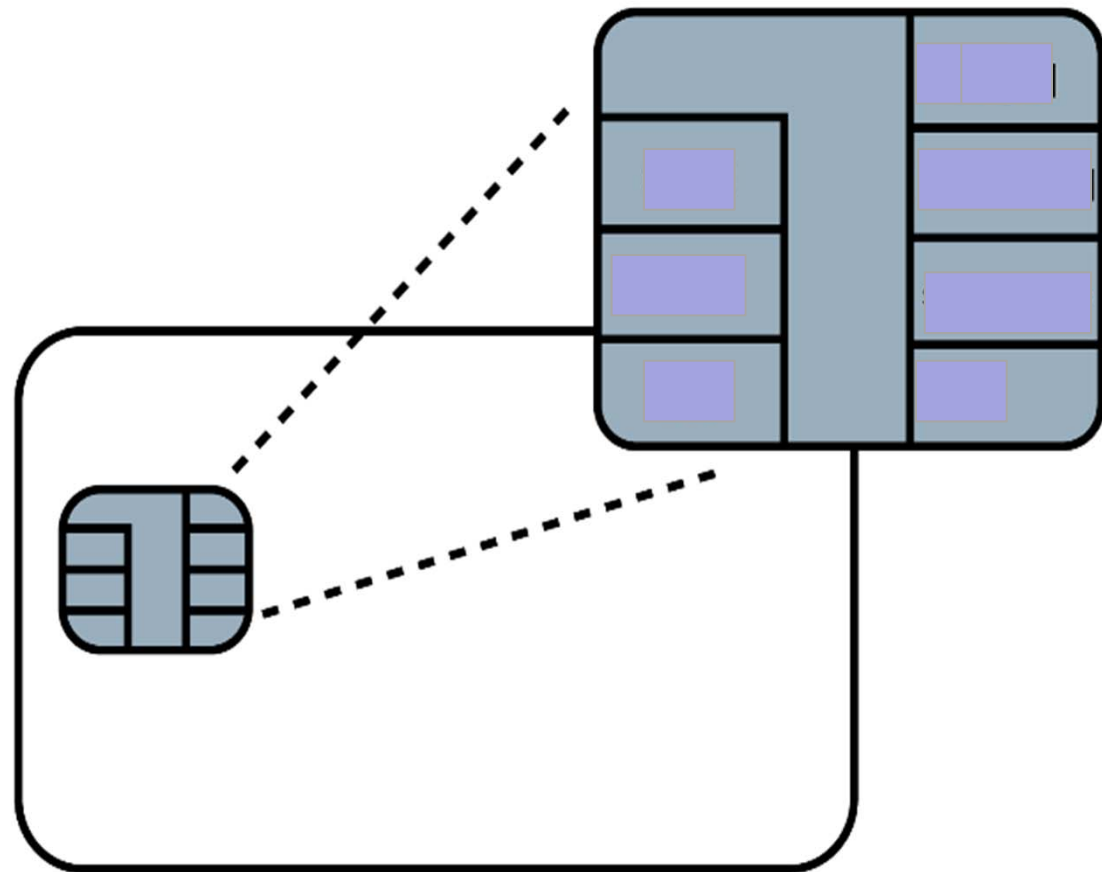


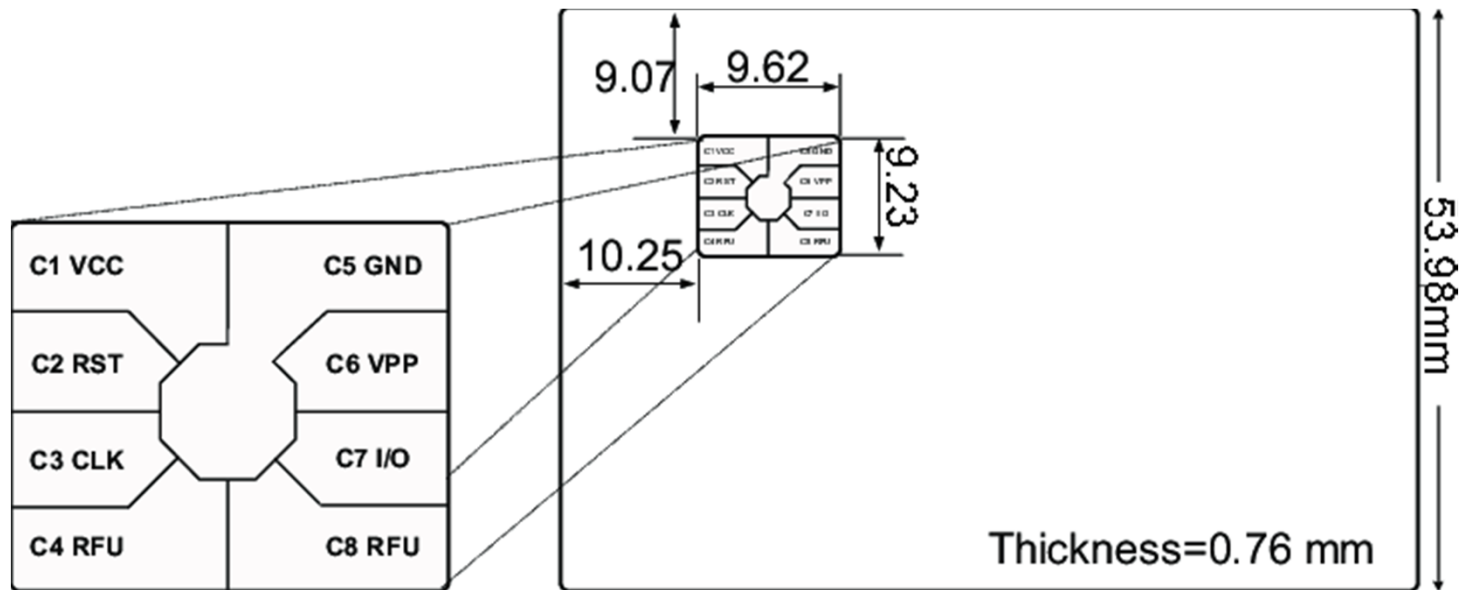
## **HSM: Smartcards • Überblick (1/2)**

- Eine SmartCard ist ein IT-System in der genormten Größe der EC-Karte (86 x 54 x 0,76 mm) mit Sicherheitsdienstleistungen.
- **Eine SmartCard enthält:**
  - eine CPU
  - RAM- und ROM-Speicher
  - ein »schlankes« Betriebssystem im ROM
  - eine I/O-Schnittstelle, über die die gesamte Kommunikation stattfindet (Kontaktflächen oder kontaktloses Interface)
  - ein EEPROM, auf das die geheimen Schlüssel, z. B. ein privater RSA-Schlüssel oder andere symmetrische Schlüssel, sowie persönliche Daten (Passworte etc.) sicher gespeichert sind
  - Sonstiges, beispielsweise einen Krypto-Prozessor.

# HSM: Smartcards

- Überblick (2/2)
- CPU
- RAM/ROM
- I/O-Schnittstelle
- EEPROM / Flash
- Krypto-Prozessor







## **HSM: Smartcards**

- Sicherheitsdienste
- Eine SmartCard stellt dem Nutzer in der Regel folgende Sicherheitsdienstleistungen zur Verfügung:
  - Laden und Entladen von Werteinheiten für elektronisches Bezahlen (auch ohne Krypto-Prozessor)
  - Kryptographische Anwendungen wie Digitale Signaturen usw.
  - Identifikation/Authentisierung des Nutzers (Aktivieren der SmartCard)
  - Single Sign On-Anwendungen (z. B. Passwort und PIN für unterschiedliche Anwendungen)
  - Sicheres Speichern von Daten auf der SmartCard
  - Lesen gespeicherter Servicedaten
  - Ausführen sonstiger Rechenoperationen



## HSM: Smartcards

- Sicherheitsmechanismen einer Smartcard (1)
  - **Smartcard Hardware:**
    - Unter- und Überspannungsdetektion
    - Erkennung niedriger Frequenzen
    - gescramblete Busse
    - Sensoren für Licht, Temperatur usw.
    - Passivierungs- bzw. Metallisierungsschichten über Bus- und Speicherstrukturen oder über der gesamten CPU
    - Zufallszahlengenerator in der Hardware
    - spezielle CPU-Befehle für kryptographische Funktionen
    - Speicherschutzfunktionen





## HSM: Smartcards

- Sicherheitsmechanismen einer Smartcard (2)
  - **Smartcard Software:**
    - Zugriffskontrolle auf Objekte
    - Zustandsautomaten, die in Abhängigkeit von Identifikations- und Authentisierungsmechanismen Befehle zulassen



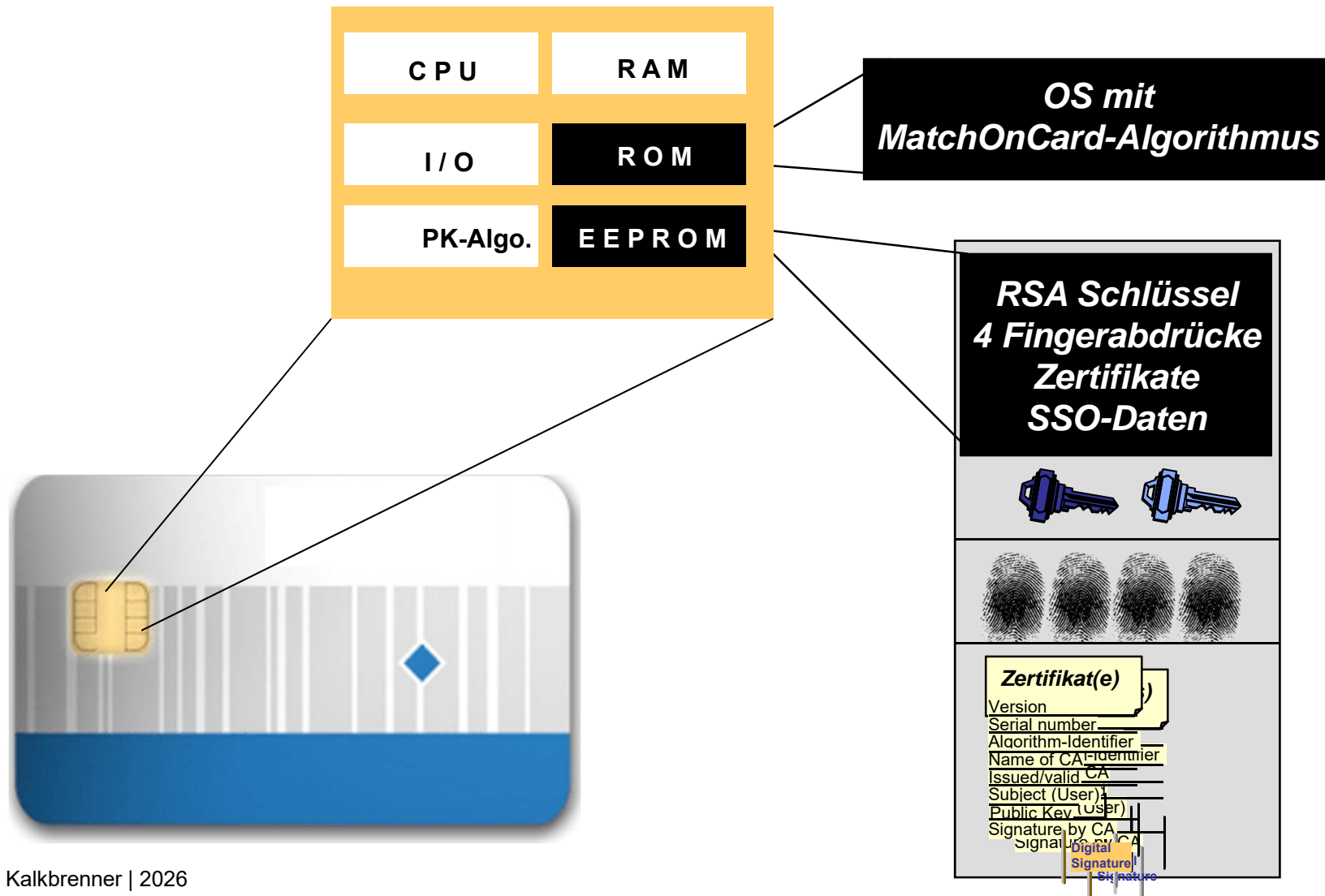
- Vorteile

## **HSM: Smartcards**

- Smartcards bieten erhöhte Sicherheit im Vergleich zu reinen Software Lösungen
- Die Sicherheit beruht auf:
  - Wissen (die PIN) und
  - Besitz (die Karte).
  - Geheime Schlüssel verlassen die Karte nie!
  - Alle geheimen Operationen finden direkt in der Karte statt.
  - Schlüssel können benutzt werden, ohne sie zu kennen
  - Geheime Daten sind manipulationssicher in der Karte gespeichert.
- Geschätzter Aufwand eines erfolgreichen Angriffs: 1 Mio. €



# HSM: Smartcards • Die biometrische Smartcard



- Alternative zur Smartcard  
**HSM: Smartcards**
  - **Yubico:**
    - FIPS certification
    - Secure manufacturing process
    - Easy to program own secrets
    - Tamper proof casing
    - Hardware two-factor authentication
    - AES encryption





- Inhalt

## Hardware-Sicherheitsmodule

- Ziele und Ergebnisse der Vorlesung
- Idee eines HSM
- HSM: Smartcards
- **HSM: Trusted Platform Module (TPM)**
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- Zusammenfassung

- Idee (1/2)

## HSM: Trusted Platform Module

- TPM ist ein kleines Sicherheitsmodul für alle Rechnersysteme (PC, Notebook, Smartphone, Drucker, Kühlschrank, usw.)



TPM

- Beispiel: IBM (Lenovo) hat seit längerem eine Notebook-Business-Lösung, auf der schon TPM vorhanden sind.
- Kosten kleiner als 1€ !



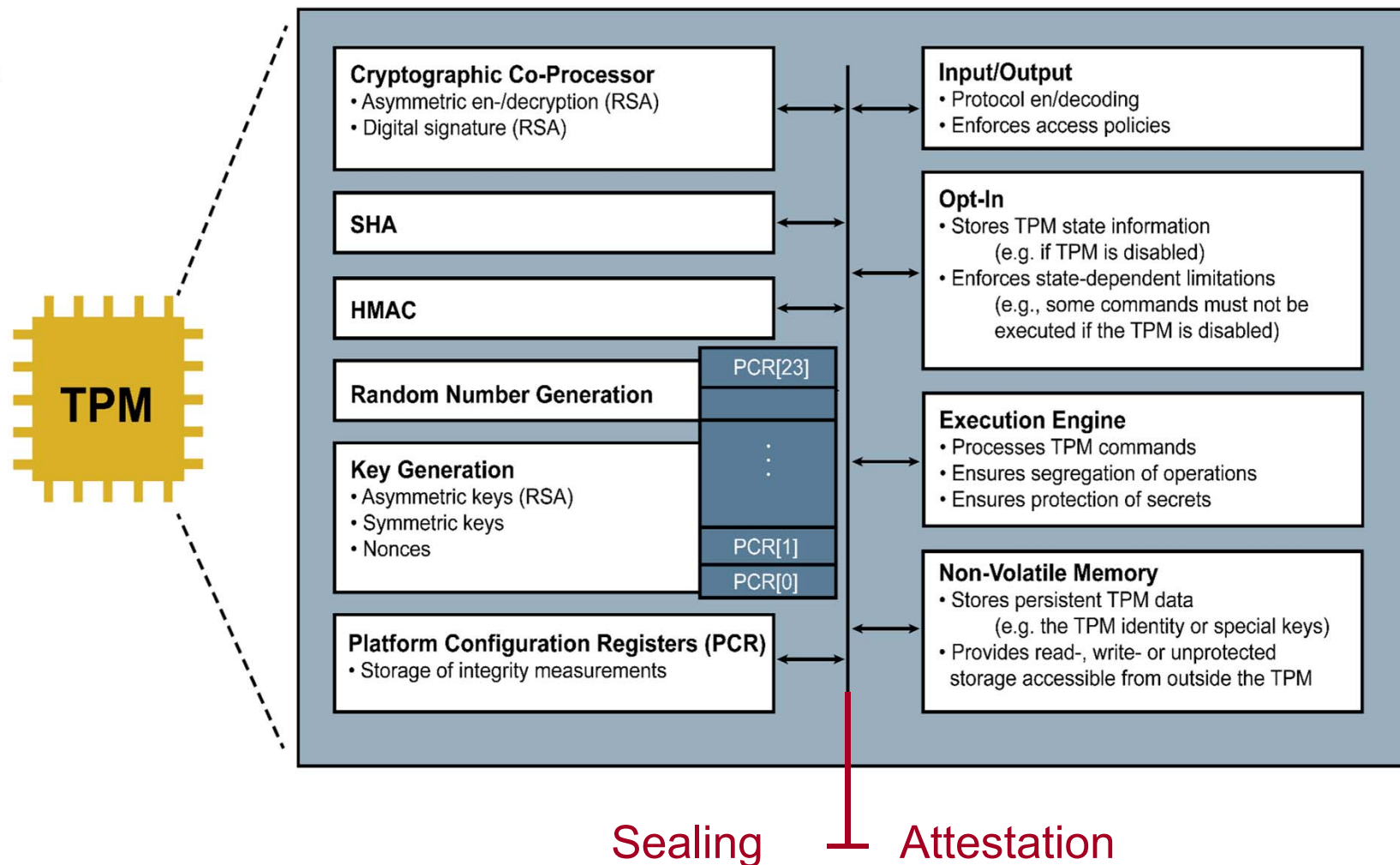
- Idee (2/2)

## **HSM: Trusted Platform Module**

- Gesteuert durch die Trusted Computing Group (TCG). Hauptmitglieder: Microsoft, Intel, HP, IBM, AMD, Sony, Oracle, aber auch Infineon, Utimaco, Lenovo...
- Einheitliche Standard-Software im TPM.
- Die einzelnen Unternehmen machen dann ihre eigene Lösung.
- Z.B. Microsoft: Next Generation Secure Computing Base (NGSCB)



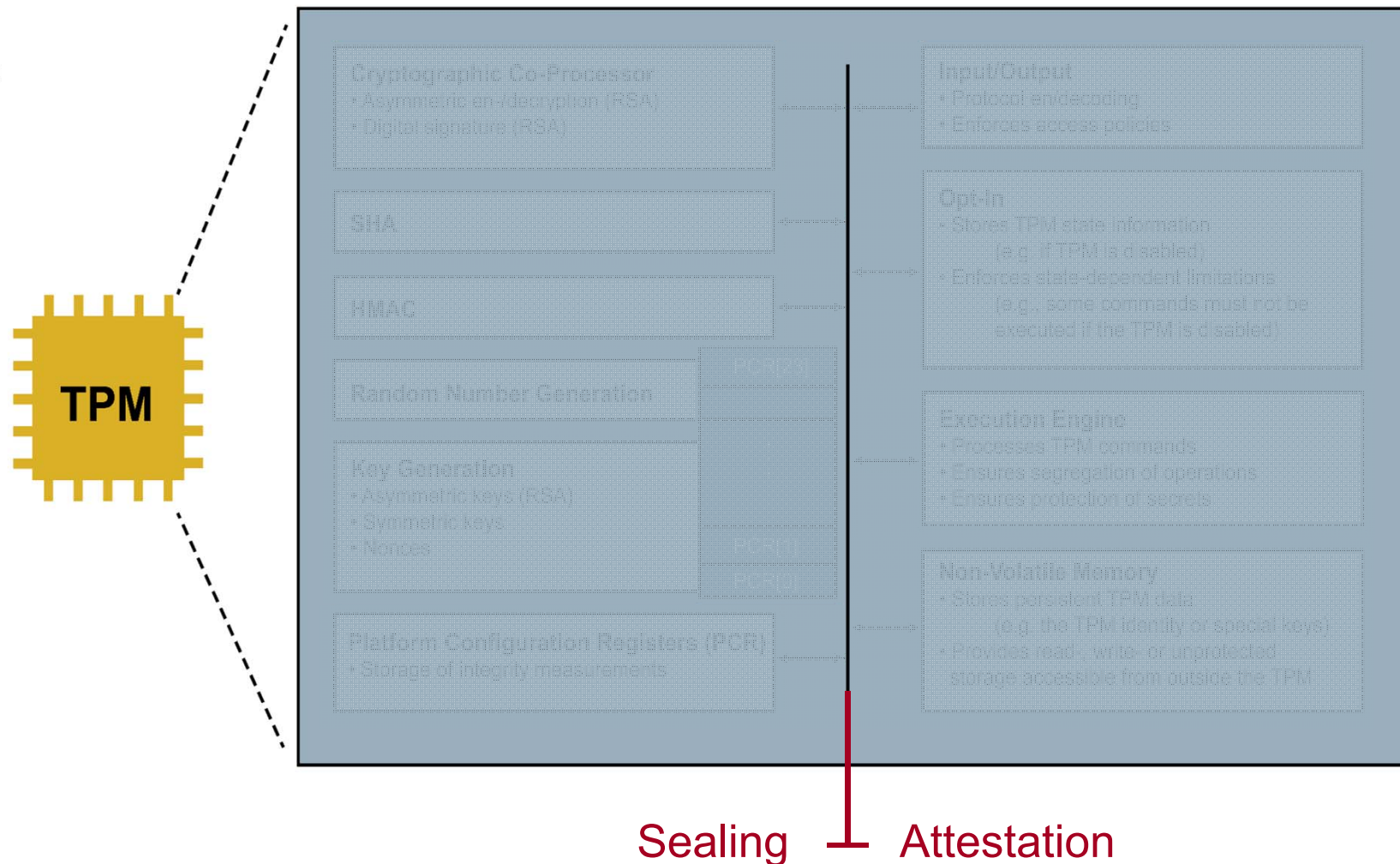
# HSM: Trusted Platform Module







# HSM: Trusted Platform Module



# HSM: Trusted Platform Module

- **Vorteile:**

- Sehr hohe Sicherheit bei geringer Investitionssumme (ein €).
- Sicherheit gleicht einer Smartcard.
- Microsoft Readiness in den meisten Fällen gegeben.
- Einfaches Sicherheitsmanagement durch Einbindung in eine Sicherheitsinfrastruktur (PKI, etc.).

- **Nachteile:**

- Intransparenz der TCG.
- Physikalische Backdoors möglich.



- Inhalt

## Hardware-Sicherheitsmodule

- Ziele und Ergebnisse der Vorlesung
- Idee eines HSM
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- **HSM: High-Level Security Module (HLSM)**
- Rahmenbedingungen
- Zusammenfassung

- Ziele

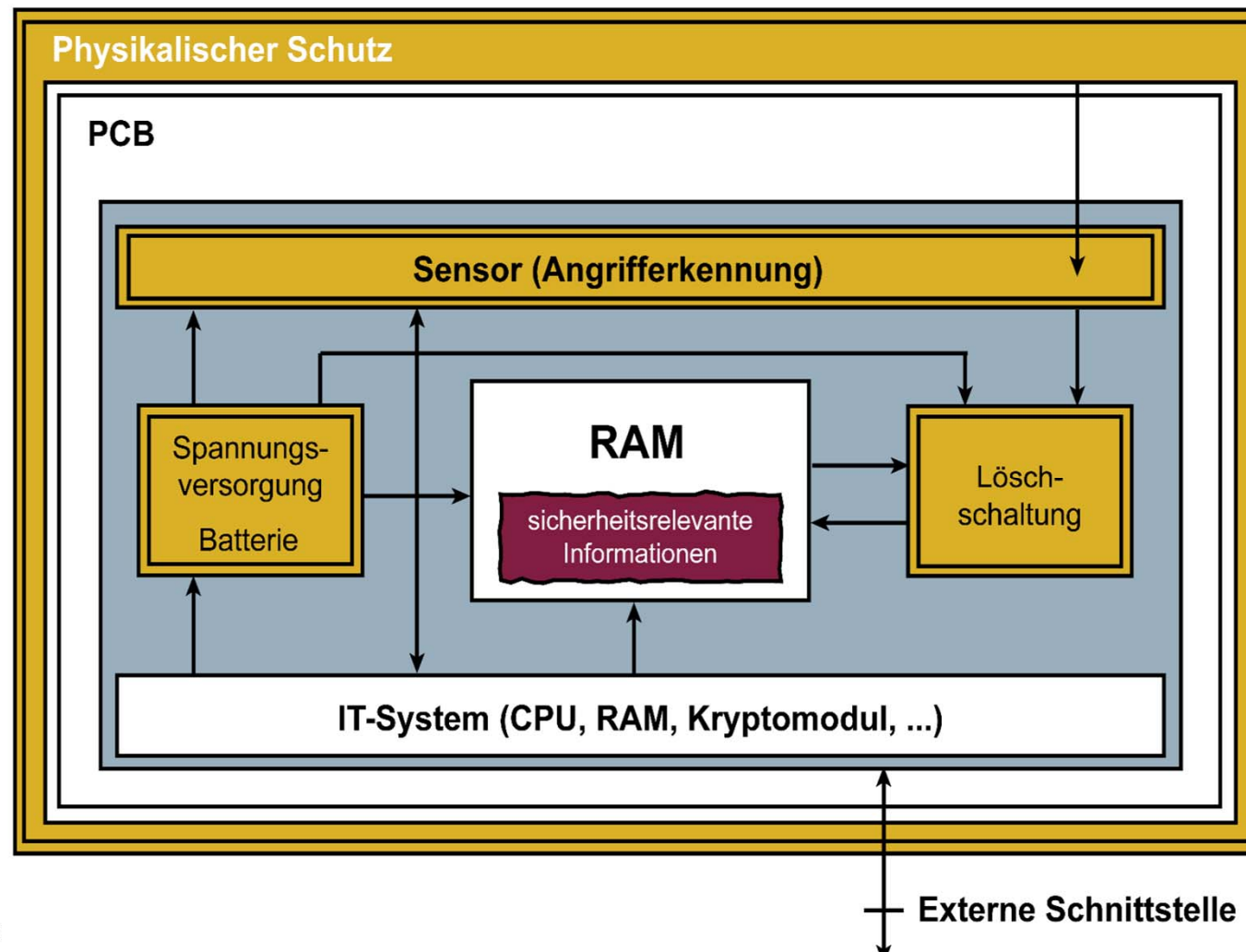
## **HSM: High-Level Security Module**

- High-security und high-performance Security Module für
  - besonders sichere, wertvolle Informationen (z.B. Master-Keys)
  - sehr hohe Performance-Anforderungen
- Wenn ein Angriff vom Sicherheitsmodul erkannt wird, sind die zu schützenden sicherheitsrelevanten Informationen innerhalb des Sicherheitsmoduls sofort aktiv zu löschen.



# HSM: High-Level Security Module

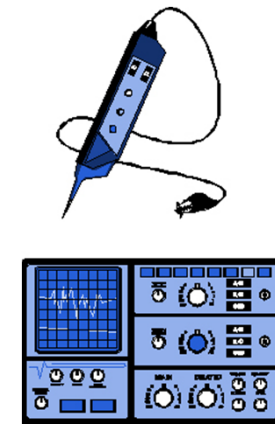
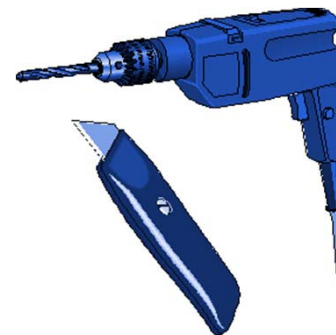
- Sicherheitsmechanismen





- **Potentielle Angriffe**  
**HSM: High-Level Security Module**

- Durchleuchten
- Temperatur Angriffe
- Mechanischen Attacke
- Chemischen Attacke
- Manipulation über Spannung





- Anforderungen (1/2)

## **HSM: High-Level Security Module**

- Grundanforderungen an Sicherheitsmodule in transaktionsbasierten Systemen:
  - Performance
  - Skalierbarkeit
  - Verfügbarkeit
  - flexible Schnittstellen zu den Host - Systemen
    - physikalisch: TCP/IP (100MBit, 1GBit, FDDI, ... )
    - logisch: Support von bestehenden Schnittstellen



- Anforderungen (2/2)

## **HSM: High-Level Security Module**

- Übergang der kryptographischen Hoheit an die Verantwortung eines Betreibers
- Umstellungsmöglichkeit auf neue kryptographische Verfahren
- Vertrauenswürdige Basis (z.B. geringe Anzahl an „Lines of Code“)





- Anwendungen (1/2)

## **HSM: High-Level Security Module**

- Geschätzter Aufwand eines erfolgreichen Angriffs: 5 Mio. €
  - z.B. Sicherung von Master-Schlüsseln
- Public Key Infrastruktur:
  - Schlüsselgenerierung (Signaturgesetz - Unterschrift!)
- Bankenumfeld:
  - Autorisierungsstationen (Freigabe von Geld)
  - Sicherheit für die Netzbetreiber  
(z.B. im Bereich ec, Mineralölunternehmen)



- Anwendungen (2/2)  
**HSM: High-Level Security Module**
- Industrie:
  - Schlüsselgenerierung für Auto-Schlüssel
  - Maut-Systeme (Abrechnung)
  - Authentifikation im Mobilfunknetz
  - Digitale Signatur von zentralen Prozessen (Rechnungen, usw.)



# Hardware-Sicherheitsmodule

## Inhalt

- Ziele und Ergebnisse der Vorlesung
- Idee eines HSM
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- **Rahmenbedingungen**
- Zusammenfassung



- Evaluierung und Zertifizierung  
**Rahmenbedingungen**
  - Nachweis der Hard- und Software-Sicherheit
  - unabhängige qualifizierte Organisationen
  - Beispiele für Standards:
    - FIPS 140-1
    - FIPS 140-2
    - CC Schutzprofil CWA 14167-2
  - Beispiele für Fragestellungen:
    - Erfüllt ein Zufallszahlengenerator alle notwendigen Eigenschaften, wie z.B. Gütekriterien, Streuung, Periodizität, Gleichverteilung?
    - Sind die Sicherheitsprotokolle sicher implementiert?



- **Key-Management (1/2)**  
**Rahmenbedingungen**
  - **Generelle Anforderungen:**
    - Keiner hat direkten Zugriff auf die geheimen Schlüssel.
    - Nutzung der Krypto-Funktionen (z.B. geheime Schlüssel) nur nach Autorisierung.
    - Definition und Veränderung von Funktionalitäten nur nach Autorisierung.
  - **Management von TPMs:**
    - Personalisierung des TPMs durch einen einzigartigen Endorsement Key (EK).
    - EK wird von einer öffentlichen PKI verwaltet/signiert.
    - Einfache Einbindung für verschiedene Anwendung (z.B. VPN-Systeme).



- **Key-Management (2/2)**  
**Rahmenbedingungen**
  - **Management nach dem Vier-Augen-Prinzip:**
    - Kritische Tätigkeiten sollten nicht von einer einzelnen Person durchgeführt werden.
    - Electronic Cash-Netze werden z.B. durch HLSMs miteinander verbunden.
    - Verwendung von unterschiedlichen Schlüsselsystemen.
    - Um-Verschlüsselung der Transaktionen nötig.
    - Schlüssel werden nach dem Vier-Augen-Prinzip eingegeben.



# Hardware-Sicherheitsmodule

- **Inhalt**
- Ziele und Ergebnisse der Vorlesung
- Idee eines HSM
- HSM: Smartcards
- HSM: Trusted Platform Module (TPM)
- HSM: High-Level Security Module (HLSM)
- Rahmenbedingungen
- **Zusammenfassung**



## Hardware-Sicherheitsmodule

- Zusammenfassung
  - **Einsatzumfeld einer SmartCard**
    - SmartCards werden typischerweise als Sicherheitskomponenten für **Personen** eingesetzt.
  - **Einsatzumfeld eines high-security und high-performance Security Modules**
    - High-security und high-performance Security Module werden typischerweise als **Sicherheitskomponenten für größere Rechnersysteme im Sicherheitsumfeld** eingesetzt.
  - **Einsatzumfeld von TPM**
    - TPMs werden überwiegend als **Sicherheitskomponenten für kleinere Rechnersysteme** eingesetzt.





# IT-Sicherheit

## 05 Open SSL



**Gerrit Kalkbrenner**  
**Gerrit.Kalkbrenner@hwr-berlin.de**



## Bibliotheken für Krypto-Software

- Proprietäre Implementierungen
- Bestandteil des Betriebssystems
- Hardware (TPM)

Weiterhin → OpenSSL

## Was ist es?

- robust
  - commercial-grade,
  - full-featured
  - general-purpose
  - Für Kryptographie und sichere Kommunikation.
- 
- command line application
  - kryptographische Funktionen
  - Generierung von Schlüsseln and Zertifikate

## Wo kommt es her?

- OpenSSL is komplett open source.
- Versionen ab 3.6 stehen unter der Apache v2 license.
- <https://www.openssl.org/source>
- Vorkompiliert für Windows:  
<https://www.heise.de/download/product/win32-openssl-47316/download>



# Arbeiten mit der Konsole

```
Win64 OpenSSL Command Prompt

OpenSSL 3.6.0 1 Oct 2025 (Library: OpenSSL 3.6.0 1 Oct 2025)
built on: Wed Oct  8 20:29:58 2025 UTC
platform: VC-WIN64A
options: bn(64,64)
compiler: cl /Z7 /Fdssl_static.pdb /Gs0 /GF /Gy /MD /W3 /wd4090 /nologo /O2 -DL_ENDIAN -DOPENSSL_PIC -D"OPEN
SSL_BUILDING_OPENSSL" -D"OPENSSL_SYS_WIN32" -D"WIN32_LEAN_AND_MEAN" -D"UNICODE" -D"_UNICODE" -D"CRT_SECURE_NO
_DEPRECATED" -D"_WINSOCK_DEPRECATED_NO_WARNINGS" -D"NDEBUG" -D_WINSOCK_DEPRECATED_NO_WARNINGS -D_WIN32_WINNT=0x
0502
OPENSSLDIR: "C:\Program Files\Common Files\SSL"
ENGINESDIR: "C:\Program Files\OpenSSL\lib\engines-3"
MODULESDIR: "C:\Program Files\OpenSSL\lib\openssl-modules"
Seeding source: os-specific
CPUINFO: OPENSSL_ia32cap=0x029ae3ffffebffff:0x0000000000000000:0x000000009c000000:0x0000000000000000:0x00000000
00000000

C:\Users\kalkbrenner>
```



## sha256

```
D:\ssl>openssl dgst -sha256 name.txt
```

```
SHA2-256(name.txt)=  
99671587e0856ec3a860511aa979e863d90dc8ee  
02b89ab2cce3077f242e106f
```



# AES

```
openssl enc -e -aes256 -in name.txt -out name.aes
```

```
openssl enc -d -aes256 -out name2.txt -in name.aes
```



## openssl enc -ciphers

-aes-128-cbc	-aes-128-cfb	-aes-128-cfb1
-aes-128-cfb8	-aes-128-ctr	-aes-128-ecb
-aes-128-ofb	-aes-192-cbc	-aes-192-cfb
-aes-192-cfb1	-aes-192-cfb8	-aes-192-ctr
-aes-192-ecb	-aes-192-ofb	-aes-256-cbc
-aes-256-cfb	-aes-256-cfb1	-aes-256-cfb8
-aes-256-ctr	-aes-256-ecb	-aes-256-ofb

...





# **IT-Sicherheit**

## **08 Transport Layer Security (TLS) Secure Socket Layer (SSL)**

**Gerrit Kalkbrenner**  
**Teile: Norbert Pohlmann**

**Gerrit.Kalkbrenner@hwr-berlin.de**



## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- **Einleitung**
- **Architektur und Protokolle**
- **Protokollablauf: Prinzipien, Schritte und Phasen**
- **Domänen-Zertifikaten**
- **TLS/SSL Authentifikationsmethoden**
- **TLS/SSL Anwendungsformen**
- **TLS/SSL Protokollmitschnitt**
- **Zusammenfassung**



## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- Zusammenfassung



# Ziele und Ergebnisse der Vorlesung

## → TLS/SSL-Technologie

- Gutes **Verständnis** für den Sinn und Zweck der **TLS/SSL-Technologie** und deren Anwendungsmöglichkeiten.
- Erlangen der **Kenntnisse** über die **Architektur**, Aufgaben, **Prinzipien** und **Sicherheitsmechanismen** der TLS/SSL-Technologie
- Gewinnen von **praktischen Erfahrungen** über die TLS/SSL-Technologie mit Hilfe eines Protokollmittschnittes



## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- **Einleitung**
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- Zusammenfassung



# Einleitung

## → Idee von TLS/SSL

- Da das **Internet offen** ist und die **Angriffsmöglichkeiten** sowie **Angriffswahrscheinlichkeiten** sehr groß sind, ist die Nutzung einer **verschlüsselten und integritätsgesicherten Kommunikation** zwischen Client und Server von besonderer Bedeutung.
- Sehr viele **Cyber-Sicherheitsaspekte** im Web, wie Eingabe von Passwörtern und Kreditkarten-Informationen haben mit der Einrichtung einer **vertrauenswürdigen Verbindung zwischen Client und Server** zu tun.
- **Der vorherrschende Ansatz für die Transportverschlüsselung** ist die Verwendung von **TLS (Transport Layer Security) / SSL (Secure Socket Layer) – TLS/SSL**.
- **TLS/SSL ist ein anwendungsunabhängiges Cyber-Sicherheitsprotokoll, das logisch auf einem Transportprotokoll aufsetzt.**



# Einleitung

## → Cyber-Sicherheitsfunktionen von TLS/SSL

- **Authentifikation** von Server und Client unter Verwendung von *asymmetrischen Verschlüsselungsverfahren* und *elektronischen Zertifikaten*.
- **Vertrauliche Client-to-Server (Ende-zu-Ende) Datenübertragung** mit Hilfe *symmetrischer Verschlüsselungsverfahren* unter der Nutzung eines *gemeinsamen Sitzungsschlüssels*.
- **Sicherstellung der Integrität und Authentizität** der transportierten Daten unter Verwendung des *HMAC-Verfahrens*.
- TLS/SSL bietet auch die Komprimierung der Daten an.

# Einleitung

## → Geschichte

- Die **Idee kam von Netscape**, die erste Version von SSL wurde 1994 veröffentlicht.
- 1999 wurde SSL von der **IETF als Standard** festgelegt und umbenannt zu Transport Layer Security (TLS).
- Da aber heute noch im Sprachgebrauch SSL fest verankert ist, wird im Weiteren immer von **TLS/SSL** gesprochen.

## Praktische Relevanz von TLS/SSL

- Da **TLS/SSL in allen wichtigen Internet-Technologien** wie Browsern, Web-Servern, E-Mail-Servern usw. eingebunden ist, wird TLS/SSL faktisch als **Standard für die Transportverschlüsselung** verwendet.





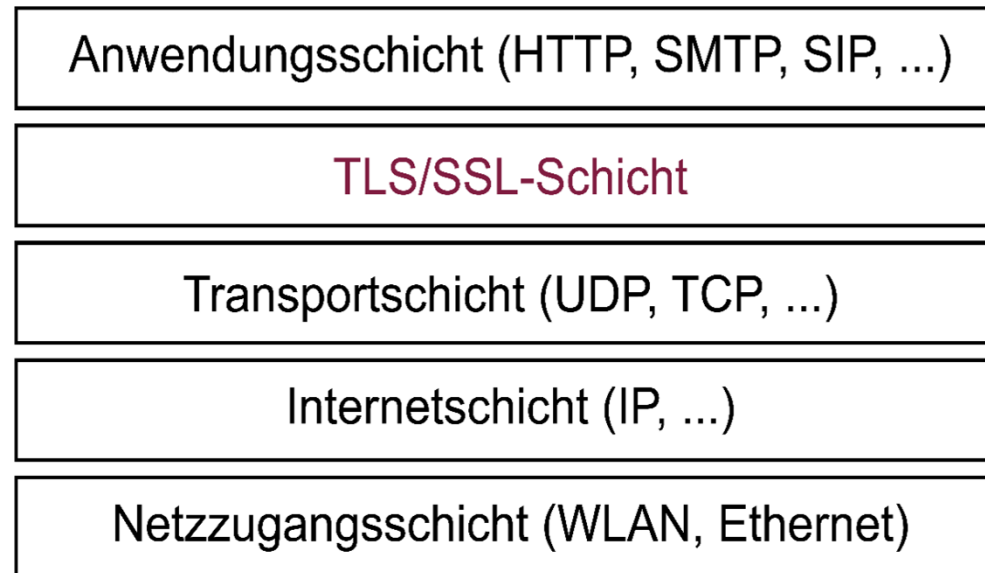
## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- **Architektur und Protokolle**
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- Zusammenfassung



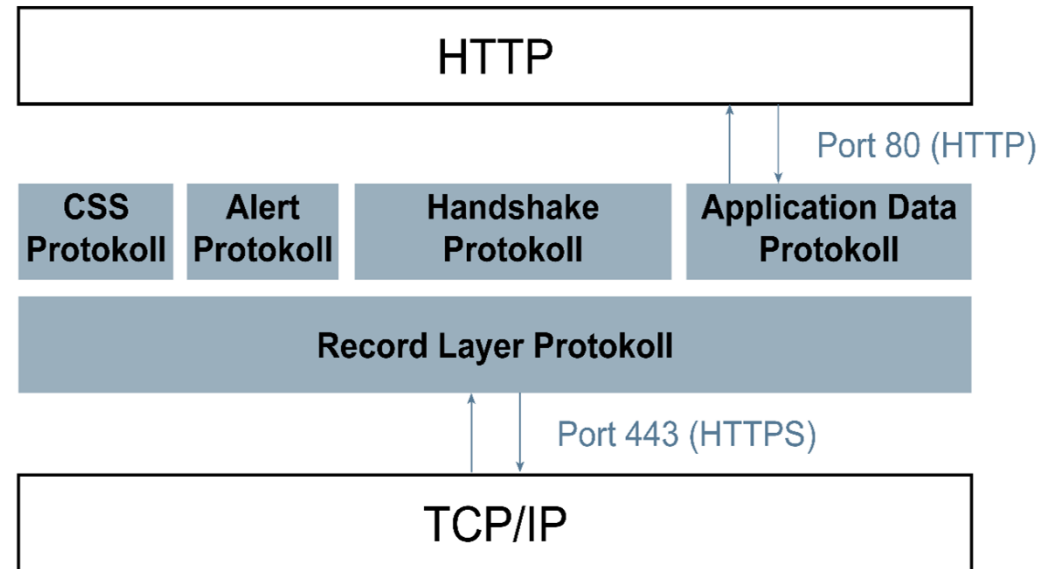
## → Kommunikationsarchitektur



- **TLS/SSL** kann eine Vielzahl **höherer Anwendungsprotokolle** unterstützen, wie HTTP, SMTP, SIP, IMAP, FTP, Telnet.
- Die genauen **Sicherheitseigenschaften** des TLS/SSL-Kanals werden **bei der Einrichtung** der *verschlüsselten* und *integritätsgesicherten Kommunikation* zwischen Client und Server **festgelegt**.
- Sicherheitseigenschaften sind: **Authentifikation von Client und Server** sowie **Integrität, Authentizität** und **Vertraulichkeit** der **Transportdaten**.



## → der TLS/SSL-Schicht



- **Die TLS/SSL-Schicht besteht aus zwei Teil-Schichten:**
  - höhere Schicht mit den **TLS/SSL-Teil-Protokollen** (CSS, Alert, Handshake, Application)
  - Record-Schicht mit dem **Record Layer-Protokoll**
- Der Client kann **durch die Wahl des Ports entscheiden**, ob die Kommunikation **TLS/SSL-gesichert sein soll** oder im Klartext:
  - Port 80: http-Kommunikation im Klartext
  - Port 443: http-Kommunikation TLS/SSL-gesichert



# Anwendung → der TLS/SSL-Schicht

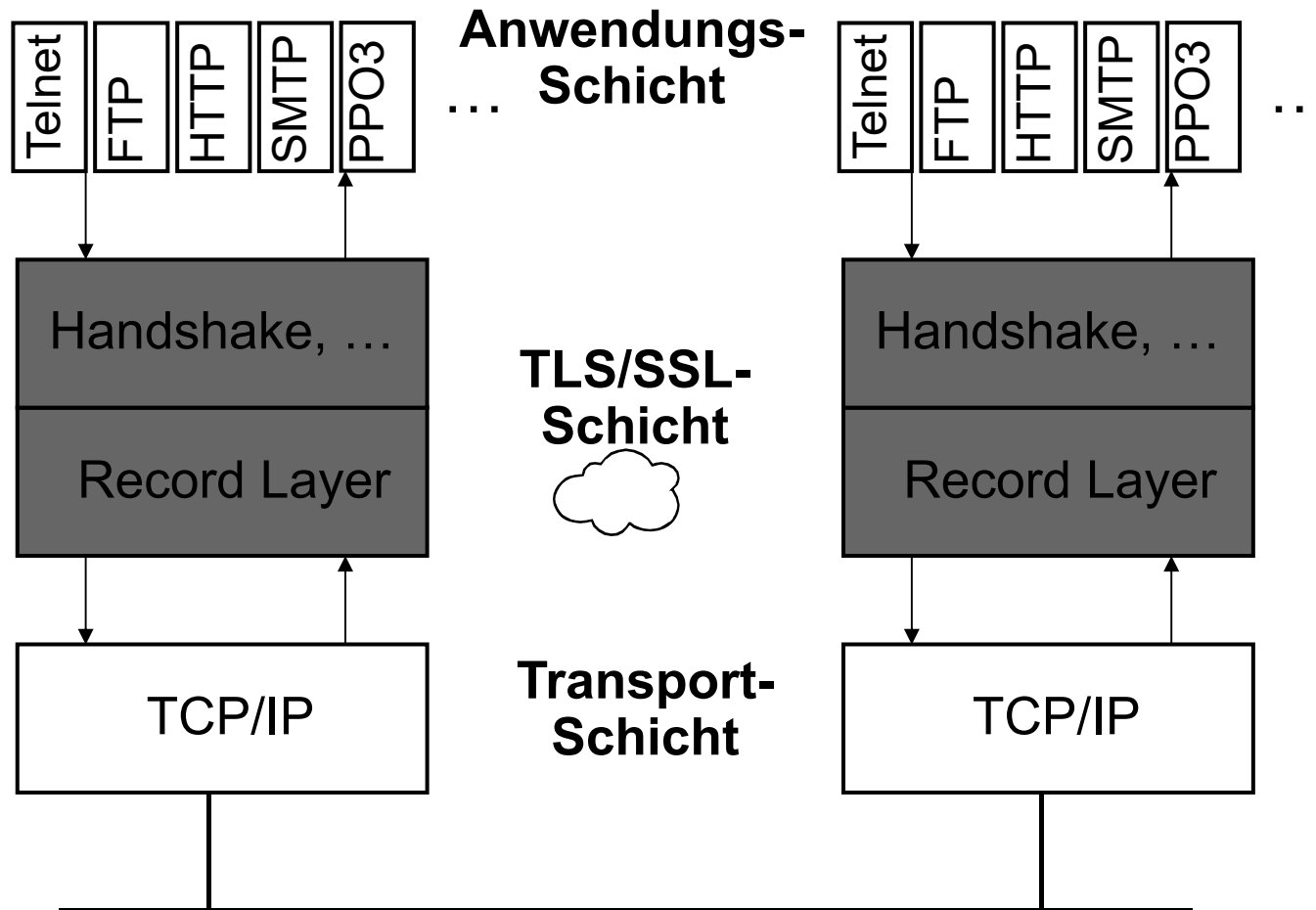
Kommunikationsanwendung	Port-Nummer	TLS/SSL Port-Nummer
Hypertext Transfer Protocol - HTTP	80	443
Simple Mail Transfer Protocol - SMTP	25	465
Internet Message Access Protocol - IMAP	143	993
Session Initiation Protocol – SIP	5060	5061
File Transfer Protocol – FTP	20, 21	989, 990
Teletype Network – TELNET	23	992
...	...	..

- für die Anwendungsprotokolle wird eine **weitere Port-Nummer** als **Transportadresse** definiert,
- Sie wird genutzt, wenn die Kommunikation zu diesen **Anwendungen TLS/SSL-gesichert** erfolgen soll.
- Anhand der **Port-Nummer** lässt sich erkennen, um welches ursprüngliche **Anwendungsprotokoll** es sich bei den übertragenen Daten handelt.  
(z.B. 443 → 80)



# TLS/SSL-Protokoll

## → Schichteneinordnung (1/2)



- Die TLS/SSL-Schicht befindet sich zwischen der Transport- und Anwendungsschicht.

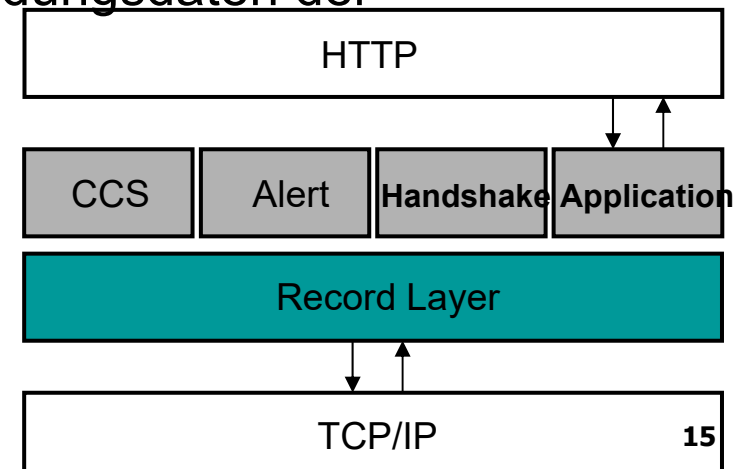


## → Schichteneinordnung (2/2)

- **TLS/SSL** übernimmt **zusätzlich** die Aufgaben der **Sitzungs- und Präsentationsschicht** (Schichten 5 und 6) des ISO/OSI-Modells.
- Ein wesentlicher Vorteil der Sitzungsschicht gegenüber der Transportschicht besteht darin, dass **Zustandsinformationen über einen längeren Zeitraum** und über verschiedene Einzelverbindungen hinweg gespeichert und für die Verwaltung genutzt werden können.
- Für das zustandslose HTTP-Protokoll, das für jeden Zugriff auf eine Webseite eine neue TCP-Verbindung aufbauen kann, bedeutet das, dass mehrere solcher Verbindungen zu einer **TLS/SSL-Sitzung** gebündelt und damit **effizienter** als die jeweiligen Einzelverbindungen **verwaltet** werden können.
- Die TLS/SSL-Protokolle sind erweiterbar und flexibel, um Zukunftssicherheit vor allem bei den Verschlüsselungsalgorithmen zu gewährleisten.
- **TLS/SSL arbeitet transparent**, so dass es leicht eingesetzt werden kann, um Anwendungsprotokollen/-diensten, ohne eigene Cyber-Sicherheitsmechanismen, **vertrauenswürdige Verbindungen** zur Verfügung zu stellen.

## → Aufgaben

- Das Record Layer Protokoll **leitet die Klartext-Anwendungsdaten aus der Anwendungsschicht verschlüsselt an die Transportschicht weiter.**
- Das Record Layer-Protokoll bietet zwei verschiedene Cyber-Sicherheitsdienste, die zusammen oder einzeln genutzt werden können:
  1. **Client-to-Server-Verschlüsselung** zwischen den beiden Transport-Endpunkten
  2. Sicherung der Nachrichten-Integrität und -Authentizität
- Zu den Aufgaben des Record-Protokolls gehören
  - die **Fragmentierung** der Klartext-Anwendungsdaten der Anwendungsschicht
  - **Kompression** der resultierenden Fragmente
  - **Berechnung von HMACs** über die (komprimierten) Fragmente
  - **Verschlüsselung** der komprimierte Fragment mit HMAC



# TLS/SSL – Record Layer Protokoll

## → Aufbau

- Gesamtgröße: 5 Byte

0	1	2	3	5
Type	Version Major	Version Minor	Length	

- **Type** (1 Byte) : Nummer des “höheren” TLS/SSL-Protokolls
  - Change Cipher Spec: 20
  - Alert: 21
  - Handshake: 22
  - Application Data: 23
- **Version Major** (1 Byte) : Hauptnummer der Version
- **Version Minor** (1 Byte) : Unterversion
- **Length** (2 Byte) : Länge der Nutzdaten in Byte.

Maximalwert darf nicht grösser sein als  $2^{14} + 2.048$  (16.384 + 2.048 Byte).





# TLS/SSL – Record Layer Protocol

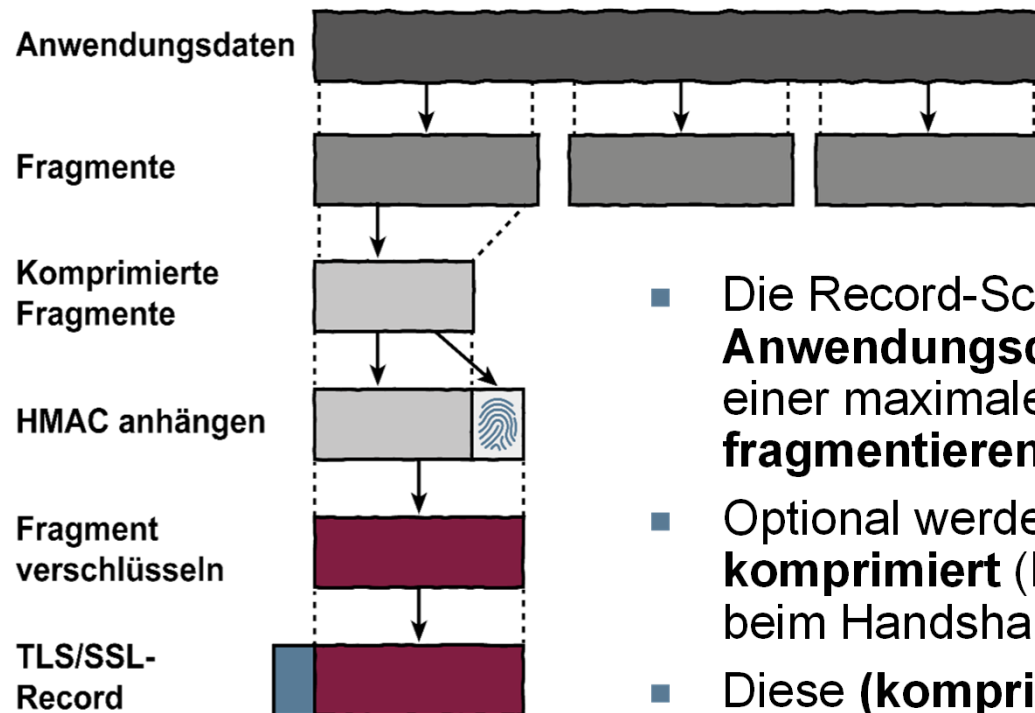
## → HMAC Berechnung

**HMAC = KH (** HMAC-Key,  
SeqNum || Compressed.type || Compressed.version ||  
Compressed.length || Compressed.fragment **)**

KH = “Keyed-Hashing for Message Authentication”-Verfahren; HMAC-Verfahren

# TLS/SSL – Record Layer Protocol

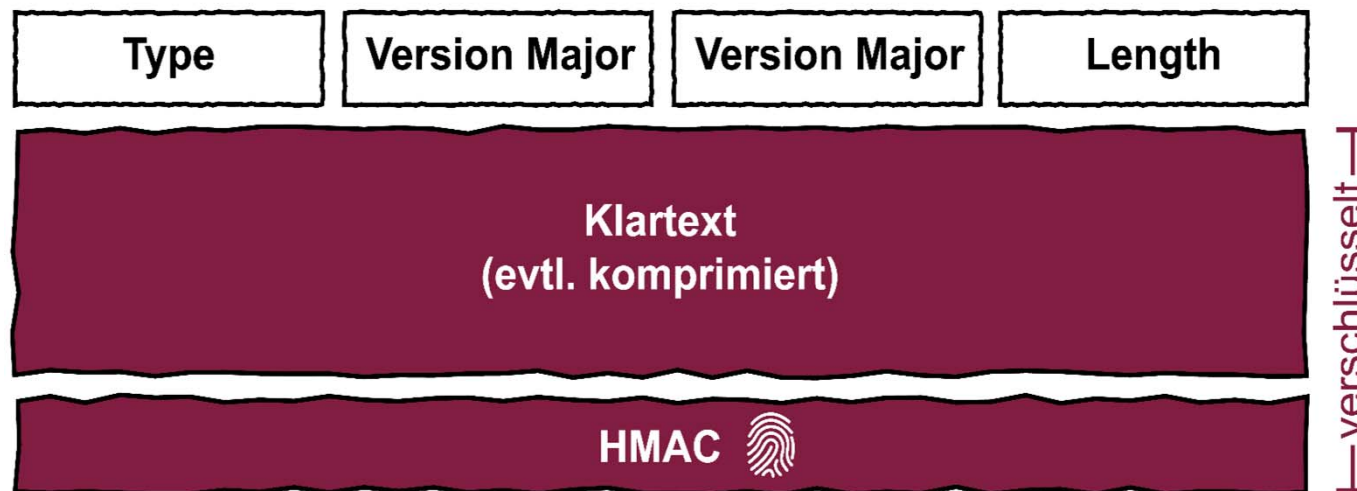
## → Ablauf



- Die Record-Schicht hat die Aufgabe, **Anwendungsdaten** in TLS/SSL-Records mit einer maximalen Größe von 214 Byte zu **fragmentieren** (Fragmente).
- Optional werden die **Fragmente** dann **komprimiert** (Komprimierungsverfahren wird beim Handshake ausgehandelt).
- Diese **(komprimierten)Fragmente** + Sequenznummer werden dann mit einer HMAC-Funktion **gehasht**.
- Die **Fragmente** und der **HMAC** werden dann **verschlüsselt**.
- Aus dem verschlüsselten Fragment wird ein **TLS/SSL-Record** gebildet.

# TLS/SSL – Record Layer Protocol

## → Verschlüsselte Daten



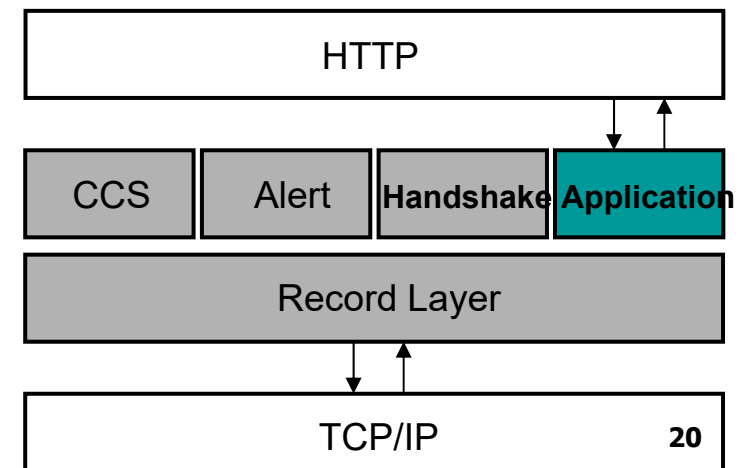
- Die eigentlichen Nutzdaten der Protokolle werden bei TLS/SSL verschlüsselt über das Application Data Protokoll übertragen.
- Der komplette Record-Layer-Header ist ungeschützt und somit lesbar.
- Zudem kann beim Handshake-Protokoll die Art der Handshake-Nachricht ausgelesen werden.



# TLS/SSL – Application Data Protokoll

## → Aufgaben

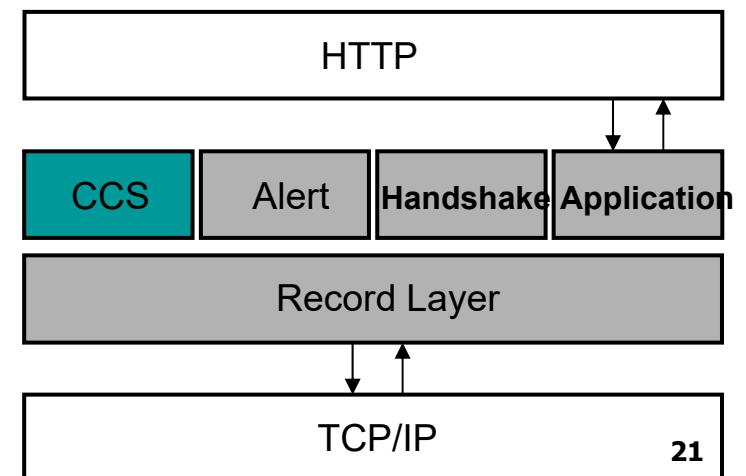
- Das **Application Data Protokoll** ist definiert, um die Anwendungsdaten transparent, d.h. ohne Betrachtung des Inhalts durchreichen zu können.
- Die anderen Protokolle oberhalb des Record Protokolls werden auf ihren Inhalt hin untersucht, dies ist bei den Daten des Application Data Protokoll nicht der Fall.
- Die Daten werden entsprechend der Sicherheitsparameter (aus dem Handshake Protokoll) fragmentiert, komprimiert, gegen Verfälschung geschützt und verschlüsselt.
- Gesamtgröße: Variabel



# TLS/SSL – ChangeCipherSpec (CCS)

## → Aufgaben

- Das **ChangeCipherSpec** Protokoll wird bei **Änderung des kryptografischen Algorithmus bzw. Parametern** genutzt .
- Es enthält nur eine Meldung bestehend aus einem Byte mit dem Wert 1.
- CCS bewirkt, dass der Empfänger die während des Handshakes ausgehandelten Parameter für die aktive Sitzung übernimmt.
- Wird eine vorhandene Sitzung reaktiviert, erfolgt die Meldung ChangeCipherSpec nach der Meldung ServerHelloDone.





# TLS/SSL – Alert Protokoll

## → Aufgaben

- Das Alert Protokoll dient der **Signalisierung von besonderen Zuständen bzw. Problemen** wie Fehler oder Verbindungsabbruch.

- Protokollnachricht enthält nur 2 Felder: Sicherheitslevel und Fehlercode.

- Gesamtgröße: 2 Byte

0	1	2
Type	Error	

- Das erste Byte kann den Wert **warning**(1) oder **fatal**(2) haben und gibt den Grad der Fehlermeldung an.
- Wird der Wert “**fatal**” übertragen, **beendet TLS/SSL** sofort die Verbindung.
- Andere Verbindungen aus der Session bleiben bestehen, aber es kann keine neue Verbindung erzeugt werden.
- Das zweite Byte beschreibt den Fehler genauer.



## → Fatale Alertmeldungen:

Name	Inhalt
Bad_record_mac(20)	Falscher HMAC für Record
Record_overflow(22)	Record Länge ist größer als $2^{14} + 2.048$ Byte
Decompression_failure(30)	Dekomprimierungsfunktion erhält falschen Input
Handshake_failure(40)	Sender akzeptiert die Sicherheitsparameter nicht
Illegal_parameter(47)	Feld im Handshake war inkonsistent
unknown_ca(48)	Root-Zertifikat konnte nicht gefunden werden oder ist nicht unter den vertrauenswürdigen Zertifizierungsinstanzen
access_denied(49)	Zertifikat gültig, aber die Zugangskontrolle hat entschieden, keinen Zugriff zu gewähren
decode_error(50)	Länge der Nachricht falsch oder Nachricht konnte nicht decodiert werden, da ein Feld nicht korrekt ist
decrypt_error(51)	Beim Handshake ist die kryptografische Operation fehlgeschlagen; Sender war nicht fähig, die Signatur oder das Ende der Nachricht zu verifizieren
protocol_version(70)	Protokollversion wird nicht unterstützt
insufficient_security(71)	Server erwartet höhere Cipher Suite als der Client unterstützt
internal_error(80)	Interner Fehler unabhängig von Nutzern

# TLS/SSL – Alert Protokoll

## → Warnungen, Alert-Meldungen:

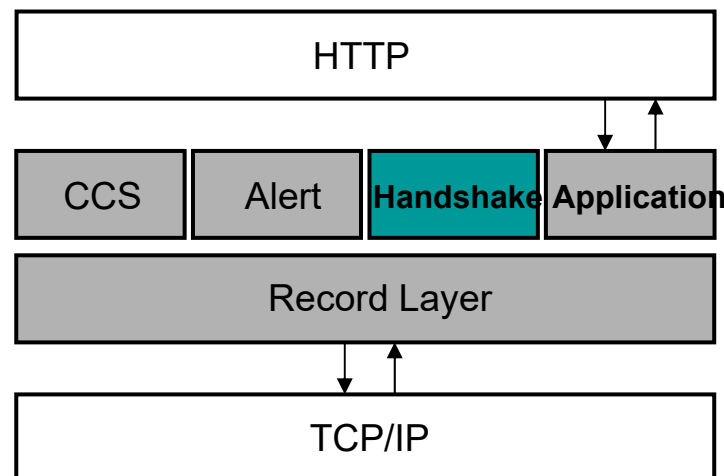
Name	Inhalt
bad_certificate(42)	Zertifikat konnte nicht erfolgreich verifiziert werden
unsupported_certificate(43)	Zertifikat eines nicht unterstützten Types
certificate_revoked(44)	Zertifikat wurde vom „Signer“ gesperrt
certificate_expired(45)	Zertifikat ist abgelaufen
certificate_unknown(46)	Sonstige Zertifikatsprobleme
user_canceled(90)	Handshake von Nutzer abgebrochen
no_renegotiation(100)	Wenn nach dem ClientHello oder Hello die Parameter neu ausgehandelt werden sollen
unsupported_extension(110)	Versendet von Clients, die vom Server eine Erweiterung bekommen, die sie nicht verlangt haben





## → Aufgaben

- Die Handshake-Nachrichten ermöglichen den **Aufbau einer TLS/SSL-Verbindung**.
- Dient u.a. zur
  - **Identifikation und Authentifizierung** der Kommunikationspartner, sowie
  - zum **Aushandeln kryptographischer Algorithmen, Schlüssel und Parameter**,die u.a. im TLS/SSL Record Layer Protokoll verwendet werden und zum Austausch benötigter geheimer Informationen.



# TLS/SSL – Handshake Protokoll

## → Aufbau

- Gesamtgröße: 5 Byte

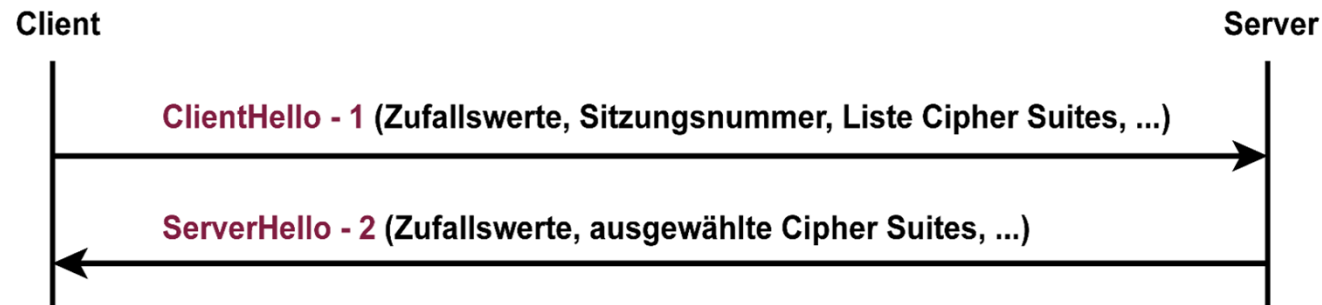
0	1	4	5
Type	Length	Content	t

- Type (1 Byte): Zeigt eine von 10 möglichen Nachrichten an (siehe Abb.)
- Length (3 Byte): Länge der Nachricht in Bytes
- Content (1 B.): Parameter, welche mit dieser Nachricht assoziiert sind

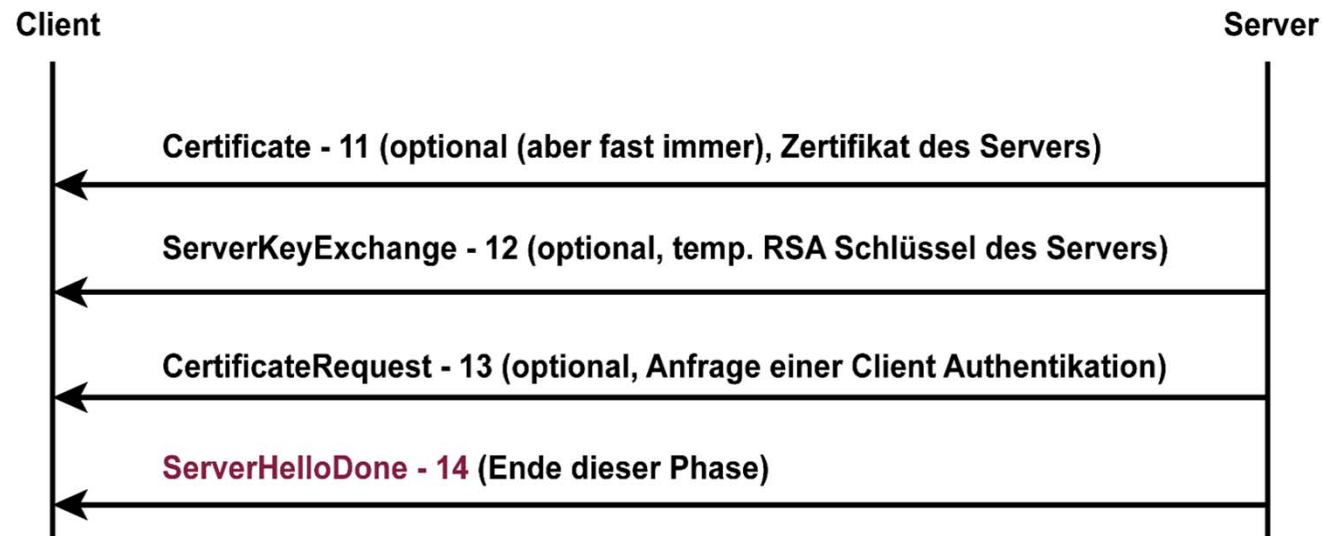


## → Nachrichten-Typen (1/2)

### Phase 1 (Aushandlung der Sicherheitsparameter)



### Phase 2 (Serverauthentifizierung und Schlüsselaustausch)

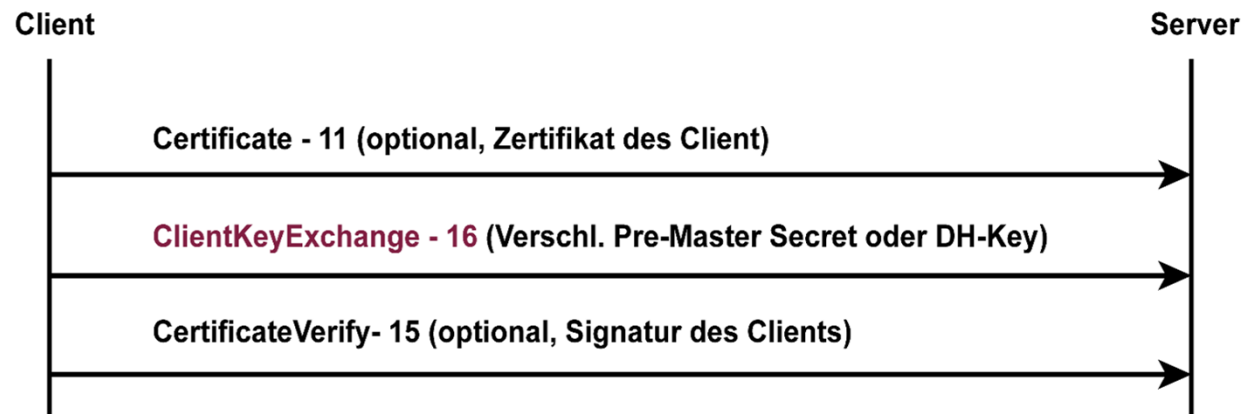




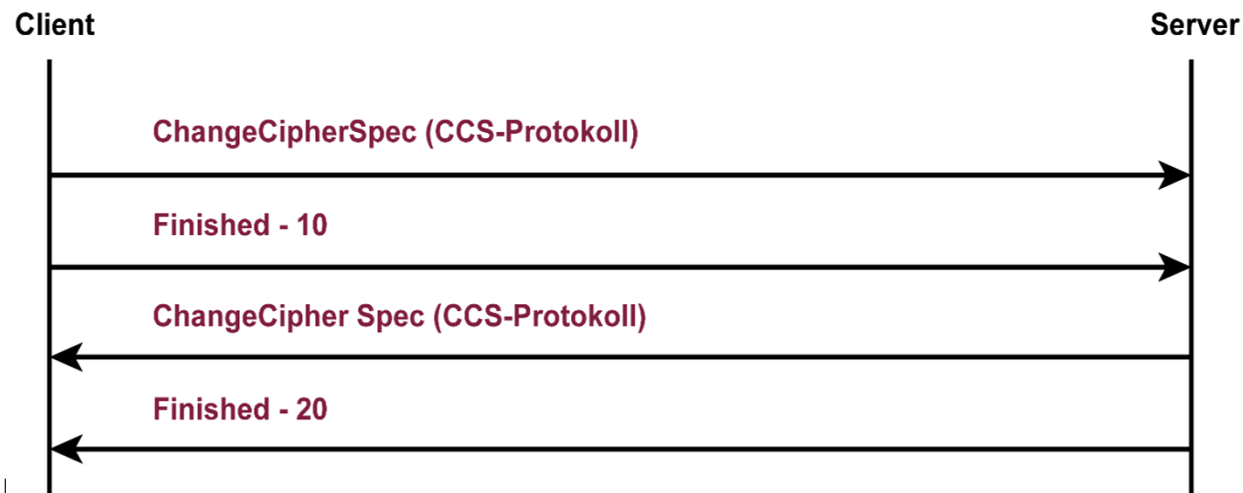
# TLS/SSL – Handshake Protokoll

## → Nachrichten-Typen (2/2)

### Phase 3 (Clientauthentifizierung und Schlüsselaustausch)



### Phase 4 (Cipher Suite wird aktiviert und der Handshake beendet)





## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- **Protokollablauf: Prinzipien, Schritte und Phasen**
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- Zusammenfassung



# Protokollablauf

## → Informationsaustausch (1/2)

---

- TLS/SSL ist ein **zustandsbehaftetes Cyber-Sicherheitsprotokoll**, mit dem **Sitzungen** zwischen Kommunikationspartnern **etabliert werden können**.
- Ein Client kann zu einem Zeitpunkt **mehrere** solche **Sitzungen** zum **gleichen oder zu verschiedenen Servern** unterhalten.
- TLS/SSL benutzt eine Session, um **Zustandsinformationen** über einen **längeren Zeitraum** zu speichern und nutzen zu können.
- Wie bei IPsec nutzt TLS/SSL für die **bidirektionale Verbindung** zwischen Client/Server zwei unterschiedliche Sitzungsschlüssel.
- Kryptographische Verfahren und Hashfunktion werden pro Sitzung ausgehandelt.
- TLS/SSL versucht, möglichst wenig geheime Informationen über das nicht vertrauenswürdige Transportsystem Internet zu übertragen.
- Daher werden lediglich Basisinformationen ausgetauscht, womit die beteiligten Kommunikationspartner (Client und Server) dezentral ihre Geheimnisse, wie die gemeinsamen Session Keys und HMAC-Schlüssel, berechnen.



# Protokollablauf

## → Informationsaustausch (2/2)

- Eine TLS/SSL-Session ist eine „**Security-Association**“ zwischen einem Client und einem Server.
- Sessions werden über das Handshake-Protokoll aufgebaut.
- Eine Session definiert eine **Menge von kryptographischen Sicherheitsparametern**, die gemeinsam über mehrere Verbindungen genutzt werden können.
- Der Sinn der Session besteht darin, nicht jedes Mal eine neue zeitaufwendige Verhandlung der Sicherheitsparameter auszuführen.



# Sitzungs- und Verbindungskonzepte

## → Session Zustände

Ein Session-Zustand ist definiert über folgende Parameter:

- **Session-Identifizier** Zufällige Bytesequenz, die vom Server erzeugt wird, um eine aktive oder wiederherstellbare Session zu identifizieren.
- **Peer-Certificate** X509.v3 Zertifikat des Clients (falls vorhanden).
- **Compression-Method** Kompressionsalgorithmus.
- **Cipher-Spec** Definiert den Verschlüsselungsalgorithmus sowie die One-Way-Hash-funktion für den HMAC. Es werden noch weitere Attribute, etwa die HMAC-Länge, festgelegt.
- **Master-Secret** 48-Byte, die vom Client und Server benutzt werden, um die geheimen Schlüssel daraus abzuleiten.
- **Is-Resumable** Mit diesem Flag wird angezeigt, ob diese Session für neue Verbindungen genutzt werden kann.





# Sitzungs- und Verbindungskonzepte

## → TLS/SSL-Connection

- Die **TLS/SSL-Connection** ist ein **logischer Transportkanal** zwischen Client und Server.
- Es handelt sich bei TLS/SSL um eine Peer-to-Peer-Verbindung, die nur temporär genutzt wird.
- Die **TLS/SSL-Connection** ist immer mit einer **Session** assoziiert.
- Damit können **aus einer Session mehrere TLS/SSL-Connections** aufgebaut und parallel betrieben werden.



# Sitzungs- und Verbindungskonzepte

## → Zustände einer TLS/SSL-Connection

- **Server-/Client-Random** Zufallszahlen, die von Server und Client für jede Verbindung neu erzeugt werden.
- **Server-Write-MAC-Secret** Geheimer Schlüssel für die HMAC-Operationen on Daten, die zum Server übertragen werden.
- **Client-Write-MAC-Secret** Geheimer Schlüssel für die HMAC-Operationen von Daten, die zum Client übertragen werden.
- **Server-Write-Key** Symmetrischer Schlüssel für die Verschlüsselung vom Server und für die Entschlüsselung vom Client.
- **Client-Write-Key** Symmetrischer Schlüssel für die Verschlüsselung vom Client und für die Entschlüsselung vom Server.
- **Initialization-Vectors** Wird genutzt, wenn ein entsprechender Mode of Operation mit Initialization Vector (IV) verwendet wird.
- **Sequence-Numbers** Sowohl Client als auch Server nutzen Sequenznummern für die gesendeten und empfangenen Daten jeder Verbindung.  
Falls der Client oder der Server ein Change Cipher Spec verschickt oder erhält, wird die Sequenznummer wieder auf Zero gesetzt. Sequenznummern können nicht größer als  $2^{64} - 1$  werden.



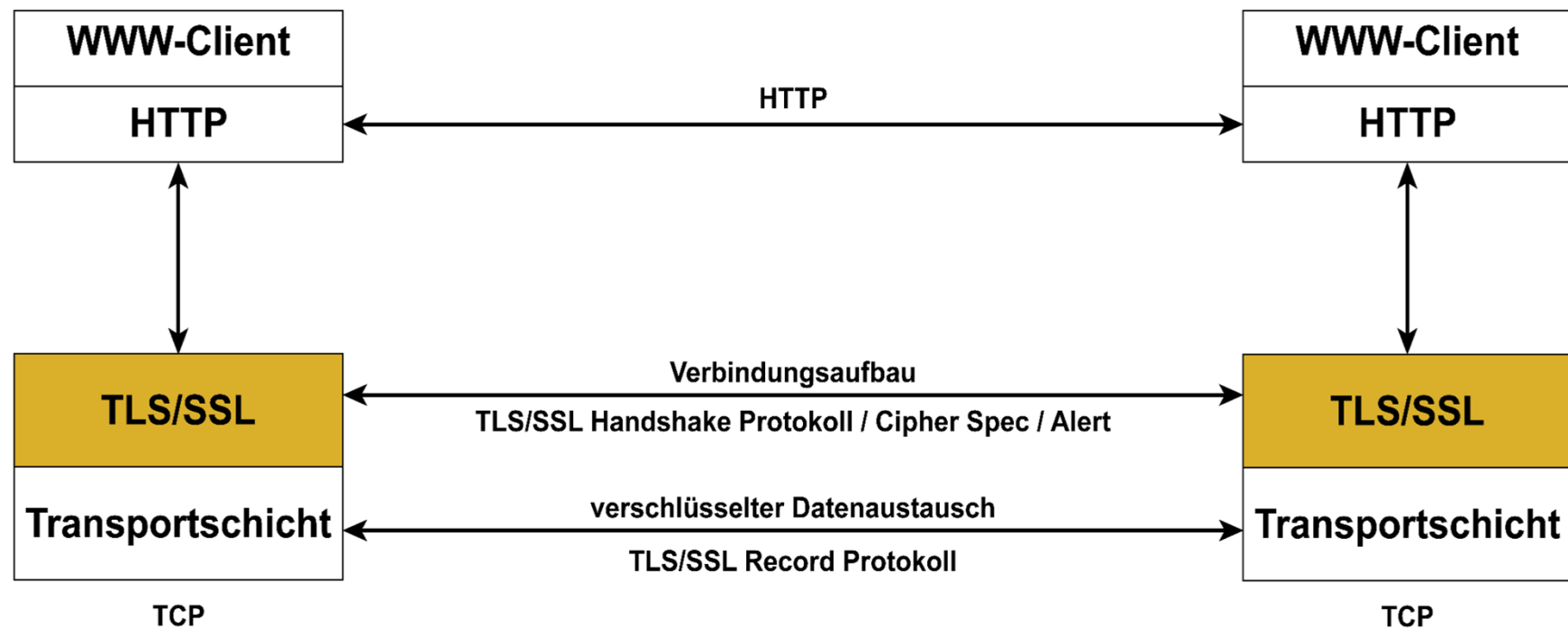
# Sitzungs- und Verbindungskonzepte

## → Session/Connection

- An den Zuständen der Verbindungen wird deutlich, dass die verwendeten Schlüssel jeweils nur für eine Verbindung gelten.
- Für alle Verbindungen einer Sitzung werden die gleichen Verfahren genutzt.
- D.H. bei der erneuten Verwendung einer bereits bestehenden Verbindung kann auf das Aushandeln der Verfahren im Handshake verzichtet werden.

# Übersicht des Protokollablaufs

## → TLS/SSL





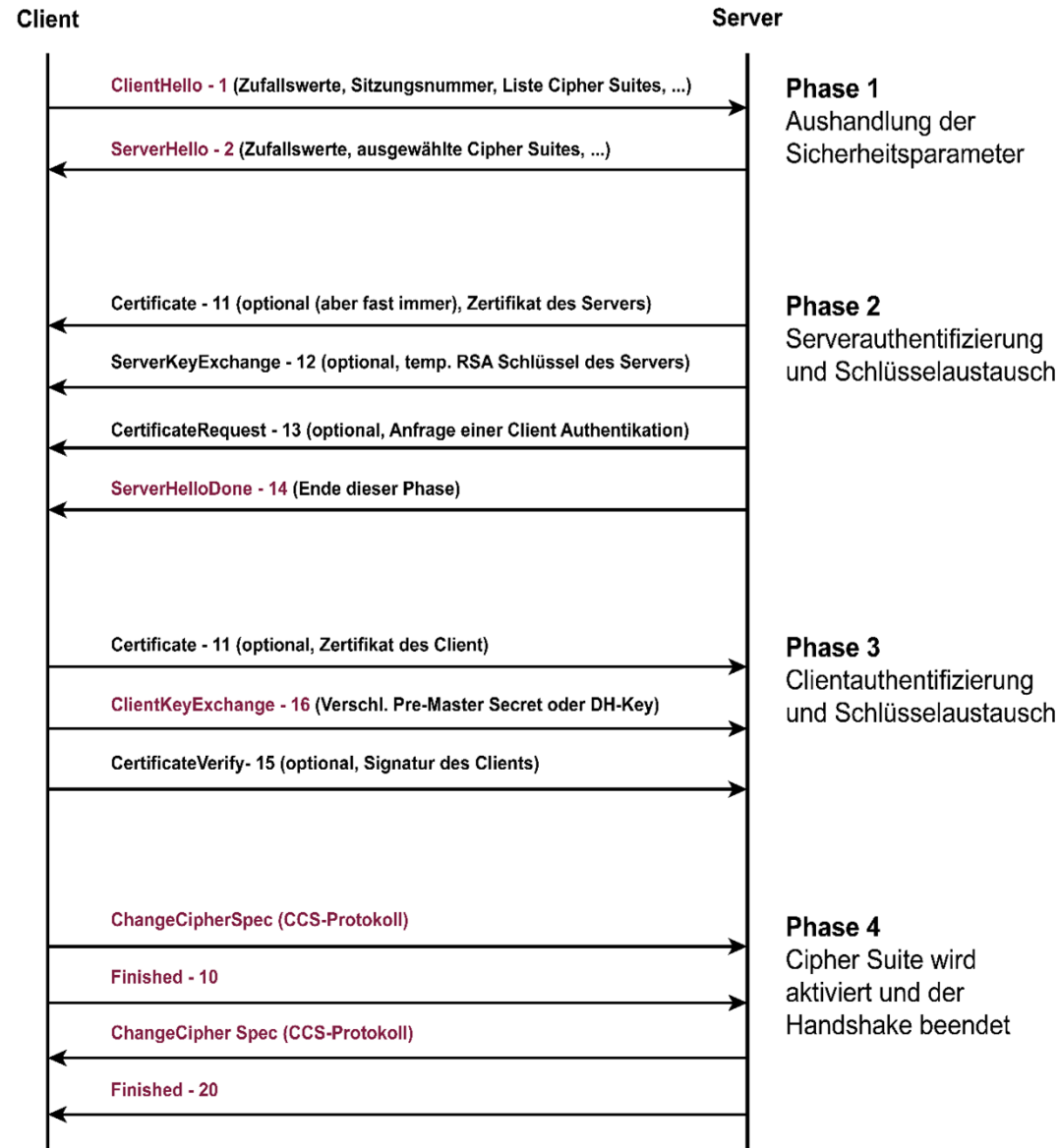
# Protokollablauf

## → Schritte und Phasen

- Der Protokollablauf bei TLS/SSL erfolgt in **zwei Schritten**.
- **1. Schritt: Verbindungsaufbau**, unterteilt in **vier Phasen**:
  - *1. Phase*: Aushandlung der Sicherheitsparameter.
  - *2. Phase*: Serverauthentisierung (Optional) und Schlüsselaustausch
  - *3. Phase*: Clientauthentisierung (Optional) und Schlüsselaustausch
  - *4. Phase*: Beendigung des Handshakes
- **2. Schritt: Transfer-Mode**  
*verschlüsselte und integritätsgesicherte Datenübertragung*



## → Verbindungsaufbau





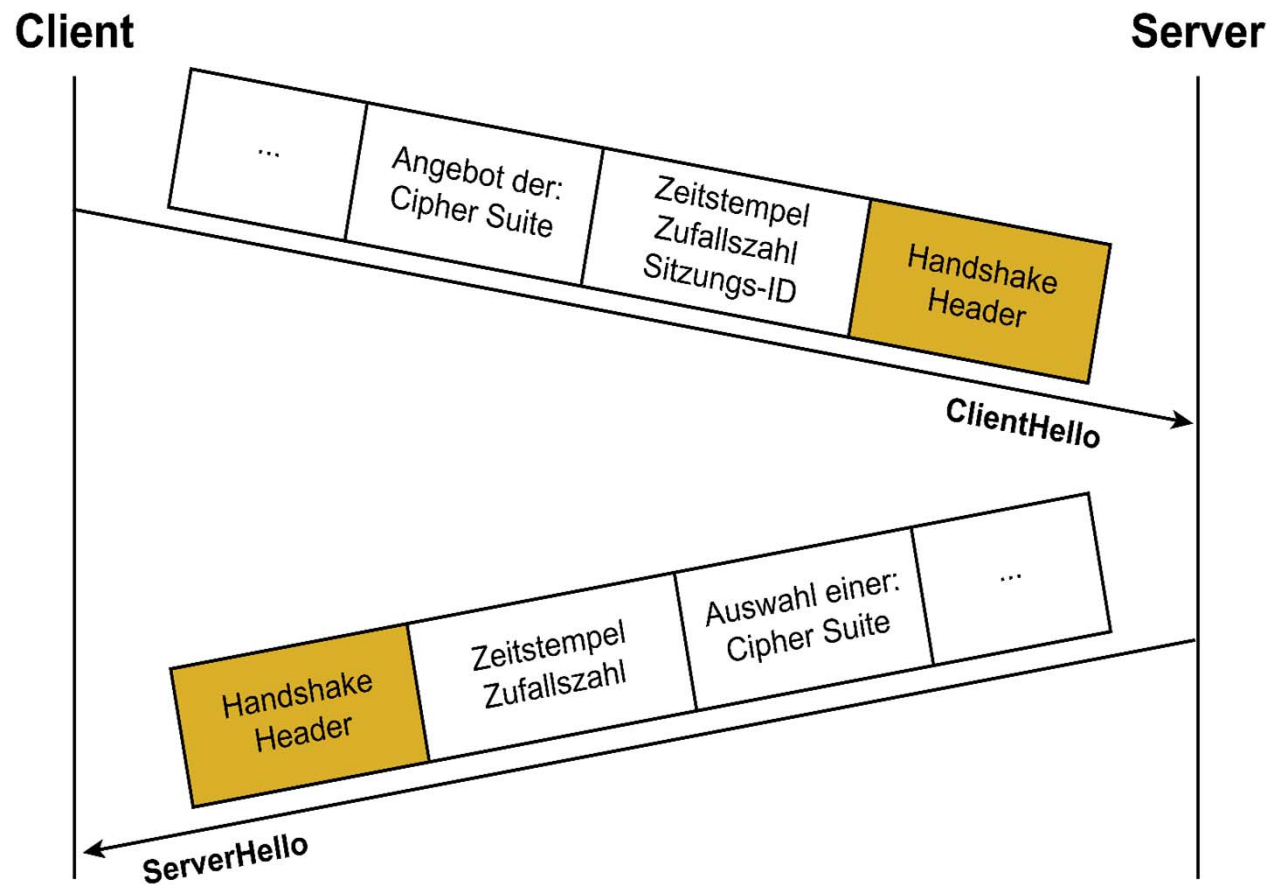
## → Phase 0: HelloRequest

- Der Server **kann** ein “HelloRequest (0)” immer zum Client senden.
- Diese Nachricht hat keine Parameter.
- Sie wird vom Server geschickt, um den Client zu einem „ClientHello“ zu veranlassen.
  - Client antwortet nur, wenn er sich nicht in einem Handshake befindet.
- Soll nicht zum Aufbau einer Verbindung genutzt werden(!), dies ist die Aufgabe des Clients mit ClientHello (nächste Folie).
- Erhält der Server auf ein HelloRequest keine Antwort, **kann** die Verbindung geschlossen werden.



# TLS/SSL - Verbindungsaufbau

## → Phase 1: Übersicht Aushandlung der Sicherheitsparameter



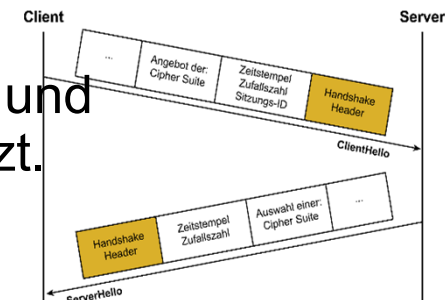




# TLS/SSL - Verbindungsaufbau

## → Phase 1: ClientHello

- In der ersten Phase werden die Sicherheitsparameter festgelegt.
- Mit dem Nachrichten-Typ „ClientHello (1)“ startet der Client den Aufbau der TLS/SSL-Verbindung.
- In einer bereits aktiven TLS/SSL-Verbindung führt diese Nachricht zu einer Neuverhandlung der Sicherheitsparameter.
- In dieser Klartextnachricht sind bereits wichtige Informationen zur Erzeugung des gemeinsamen geheimen Schlüssels enthalten:
  - Random Client - RC (4 Byte Zeitstempel + 28 Byte Zufallszahl)
  - Sitzungs-ID,
  - Prioritätenliste der Cipher Suites (Kryptographie- und Kompressionsverfahren), die der Client unterstützt.

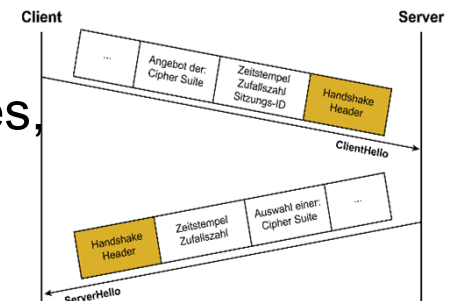


- Der Client sollte **nur** kryptographische Verfahren und Funktionen sowie Schlüssellängen für die **Cipher Suite anbieten**, die sein gewünschtes **Cyber-Sicherheitsniveau erfüllen**.



## → Phase 1: ServerHello

- Mit dem Nachrichten-Typ „ServerHello (2)“ antwortet der Server auf das „ClientHello“ des Clients.
- Die Nachricht enthält die gleichen Parameter wie der Nachrichten-Typ „ClientHello“, teilweise allerdings mit leicht abgewandelter Bedeutung.
  - Random Server - RS (4 Byte Zeitstempel + 28 Byte Zufallszahl)
- Zudem enthält dieser Nachrichten-Typ die ausgewählte Cipher Suite vom Server, die Client und Server nachfolgend nutzen.
- Hat der Client eine Sitzungs-ID vorgeschlagen, überprüft der Server diese Sitzungs-ID und übernimmt sie, sofern diese Sitzung noch nicht zu lange zurückliegt.
- Gibt der Server keine Sitzungs-ID an, so bedeutet dies, dass die gegenwärtige Sitzung später nicht erneut aufgenommen wird.
- Der Server sollte **nur kryptographische Verfahren und Funktionen** sowie Schlüssellängen auswählen, die sein gewünschtes **Cyber-Sicherheitsniveau erfüllen**.





## → Cipher Suites

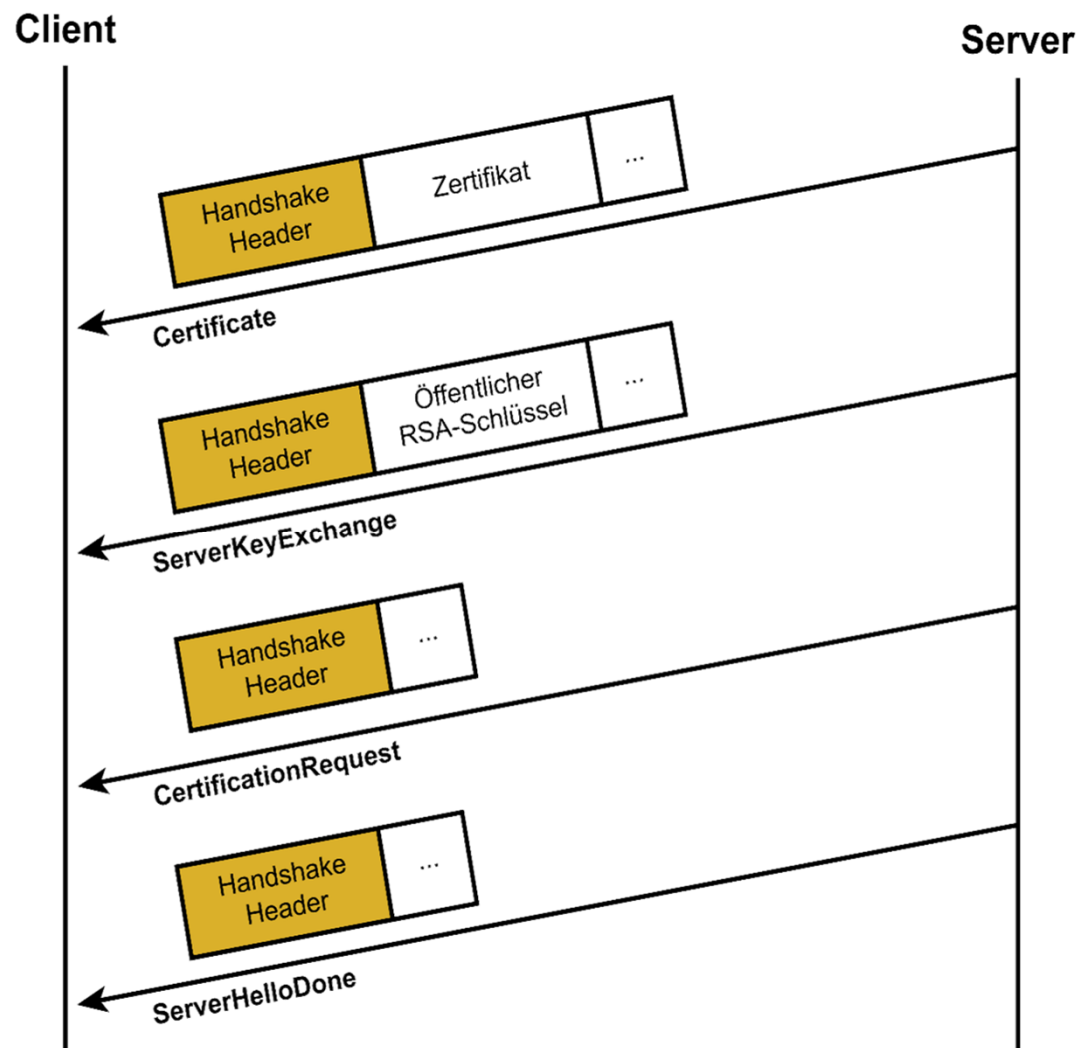
- Beim “**Server/Client Hello**” werden die sogenannten **Cipher Suites** ausgewählt.
- Dabei handelt es sich um eine Kombination aus Schlüsselaustauschverfahren, Verschlüsselungsverfahren mit Schlüssellänge und ein Verfahren zum Integritätscheck.
- Jede Cipher Suite definiert mögliche Kombinationen aus Schlüsselaustauschverfahren (DHE\_RSA), Verschlüsselungsalgorithmus (AES)/Schlüssellänge (256)/Mode of Operation (GCM) und die One-Way-Hashfunktion (SHA384) für den HMAC.





## → Phase 2: Übersicht

### Serverauthentifizierung und Schlüsselaustausch

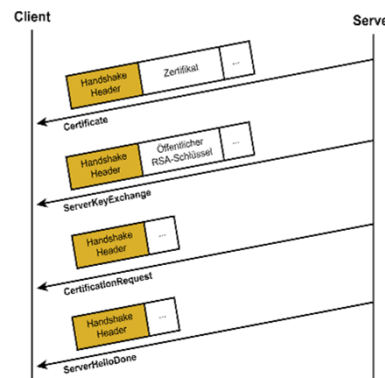




# TLS/SSL - Verbindungsaufbau

## → Phase 2: Certificate

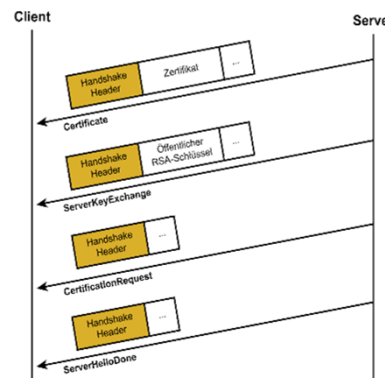
- In der zweiten Phase finden die implizierte **Serverauthentifizierung** und der **Schlüsselaustausch** statt.
- Soll der Server authentifiziert werden, so muss er dem Client im Nachrichten-Typ “Certificate (11)” sein Zertifikat zukommen lassen.
- Beim Server handelt es sich im Normalfall um ein X.509v3-Zertifikat, das für eine Domäne einer Organisation (zum Beispiel internet-sicherheit.de) von einer Zertifizierungsinstanz ausgegeben wurde.
- Dabei muss das Certificate zu der Cipher Suite passen!
- Wird RSA verwendet, muss das Zertifikat einen RSA-Schlüssel enthalten etc.



# TLS/SSL - Verbindungsaufbau

## → Phase 2: ServerKeyExchange

- Diese Meldung wird nicht gesendet, wenn der Server ein Zertifikat mit festen Diffie-Hellman Parametern oder mit einem RSA-Schlüssel besitzt.
- Versendet der Server kein Zertifikat, so lässt er dem Client im Nachrichten-Typ “ServerKeyExchange (12)” einen temporären öffentlichen RSA-Schlüssel zukommen.

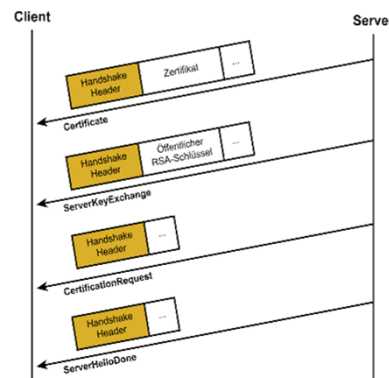




# TLS/SSL - Verbindungsaufbau

## → Phase 2: CertificationRequest

- Falls der Server auch eine Authentifikation des Client fordert, sendet er den Nachrichten-Typ „CertificationRequest (13)“.

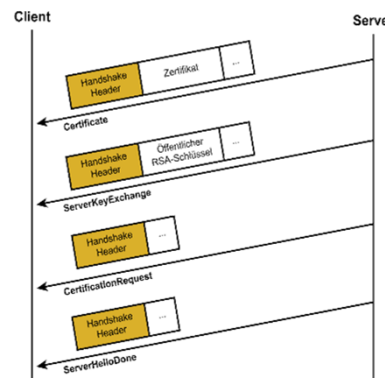




# TLS/SSL - Verbindungsaufbau

## → Phase 2: ServerHelloDone

- Der Server beendet seine Übertragung dieser Phase mit dem Nachrichten-Typ „ServerHelloDone (14)“.
- Dieser Nachrichten-Typ ist notwendig, da die Nachrichten-Typen „Certificate“, „ServerKeyExchange“ und „CertificateRequest“ nicht notwendigerweise geschickt werden.
- So erkennt der Client das Ende der Phase 2.



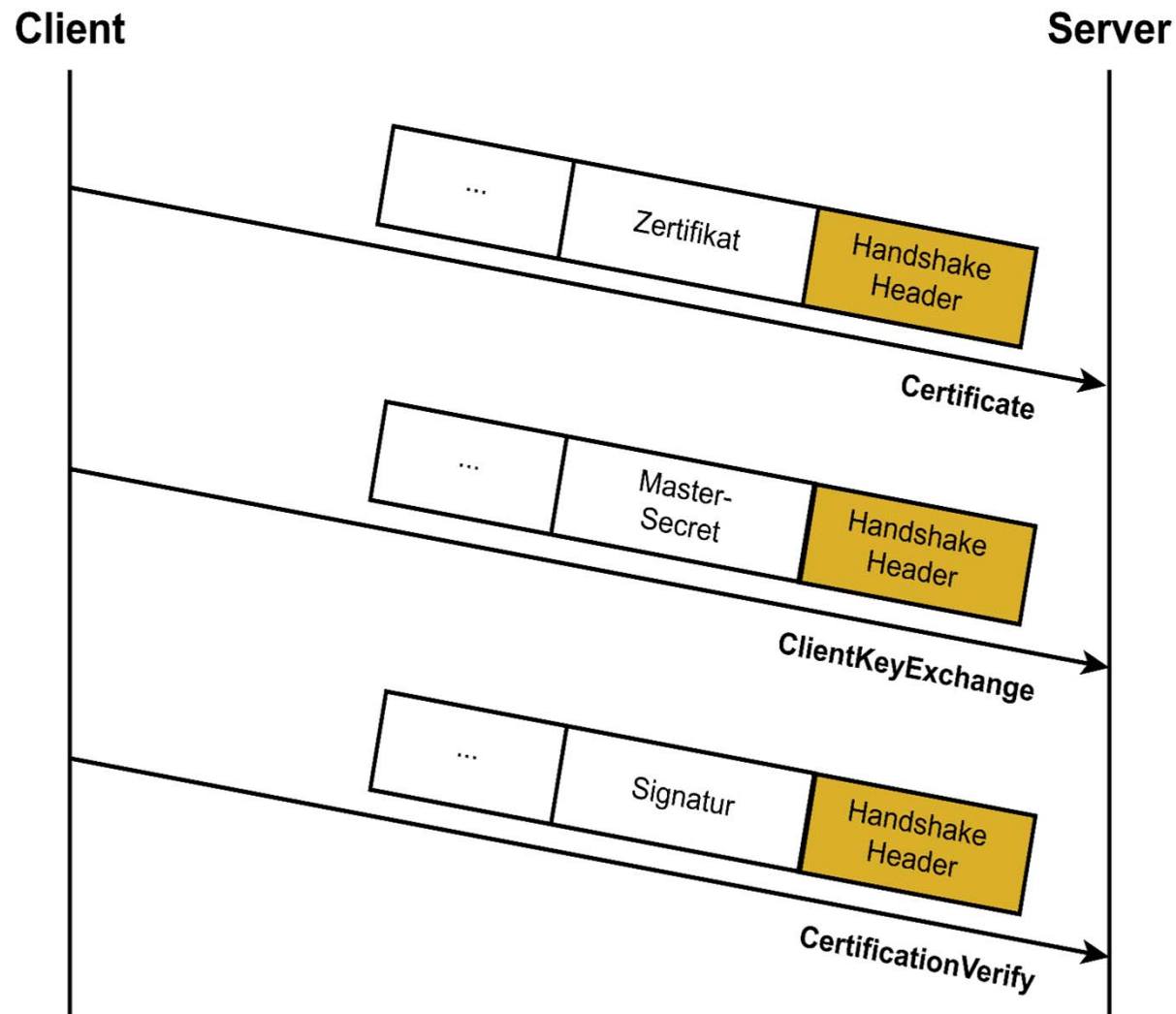




# TLS/SSL - Verbindungsaufbau

## → Phase 3: Übersicht

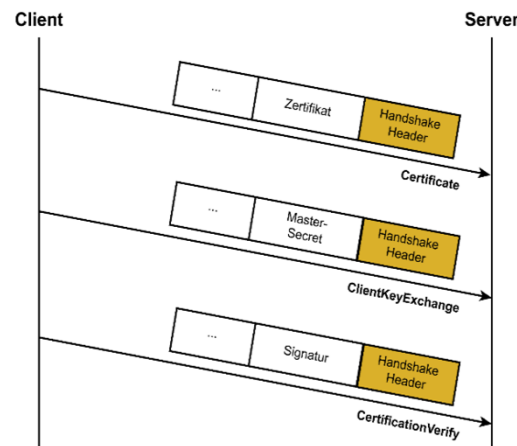
### Clientauthentifizierung und Schlüsselaustausch



# TLS/SSL - Verbindungsaufbau

## → Phase 3: Certificate

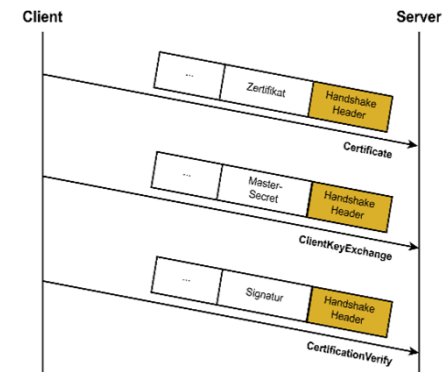
- In der Phase 3 kann der Client vom Server authentisiert werden und der Schlüsselaustausch wird fortgeführt.
- Soll der Client authentifiziert werden, so muss er dem Server sein Zertifikat im Nachrichten-Typ „Certificate (11)“ zukommen lassen.



# TLS/SSL - Verbindungsaufbau

## → Phase 3: ClientKeyExchange

- Der Client prüft zunächst die Gültigkeit des Server-Zertifikats.
- Anschließend übermittelt er dem Server mit dem Nachrichten-Typ „ClientKeyExchange (16)“ seine geheime Basisinformation mithilfe des Kryptoverfahrens aus der ausgewählten Cipher Suite, das 48 Byte Pre-Master-Secret (Pre).
- Haben sich Client/Server auf das RSA-Verfahren geeinigt, so verschlüsselt der Client das Pre-Master-Secret mit dem öffentlichen Schlüssel des Servers aus dem „Certificate“ (Server-Zertifikat) oder aus dem „ClientKeyExchange“.
- Beim Diffie-Hellman-Schlüsselaustauschverfahren sendet der Client nur seinen öffentlichen Schlüssel an den Server zurück.
- Das Diffie-Hellman-Verfahren wird hier nicht zur Berechnung des geheimen Schlüssels genutzt, sondern zur dezentralen Berechnung des Pre-Master-Secrets (Pre).

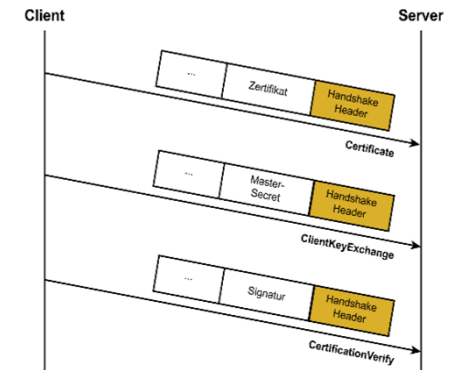




# TLS/SSL - Verbindungsaufbau

## → Phase 3: CertificateVerify

- Mit dem Nachricht-Typ „CertificateVerify (15)“ signiert der Client Informationen, sofern er sie in dieser Phase gesendet hat.
- Client „beweist“ dem Server, dass er im Besitz des geheimen Schlüssels für das Zertifikat ist.



- **Ablauf:**
  - Client berechnet einen Hashwert über die Bytes aller bisher ausgetauschten Handshake-Nachrichten, von ClientHello bis ClientKeyExchange.
  - Client signiert diesen Hashwert mit seinem geheimen Schlüssel.
  - Der Server berechnet einen Hashwert über die gleichen Handshake-Nachrichten und verifiziert die Signatur mit dem öffentlichen Schlüssel aus dem Zertifikat.



# TLS/SSL - Verbindungsaufbau

## → Master Secret Berechnung

Das Pre-Master-Secret und die ausgetauschten Zufallszahlen werden nun von Client und Server genutzt, um das 48 Byte Master-Secret zu berechnen, aus dem die benötigten geheimen Schlüssel (Session Keys, Key zur HMAC-Berechnung, ...) abgeleitet werden.

***Berechnung des Master-Secrets :***

**Master-Secret** = **KH** ( Pre-Master-Secret, ClientHello.random ||  
ServerHello.random )

**KH** = „Keyed-Hashing for Message Authentication-Verfahren; HMAC-Verfahren



# TLS/SSL - Verbindungsaufbau

## → Schlüsselerzeugung

- Alle Schlüssel (Session Keys, Key zur HMAC-Berechnung, ...) werden nach dem einen gleichartigen Schema sowohl vom Client als auch vom Server aus dem Master-Secret abgeleitet.
- Dazu generiert das Protokoll so lange eine Folge von Schlüsselblöcken „Key-Block“, bis alle Schlüssel damit konstruiert sind.

### *Berechnung des Key-Blocks (Schlüsselblock)*

**Key-Block** = **KH** (      Master-Secret, ServerHello.random ||  
   ClientHello.random      )

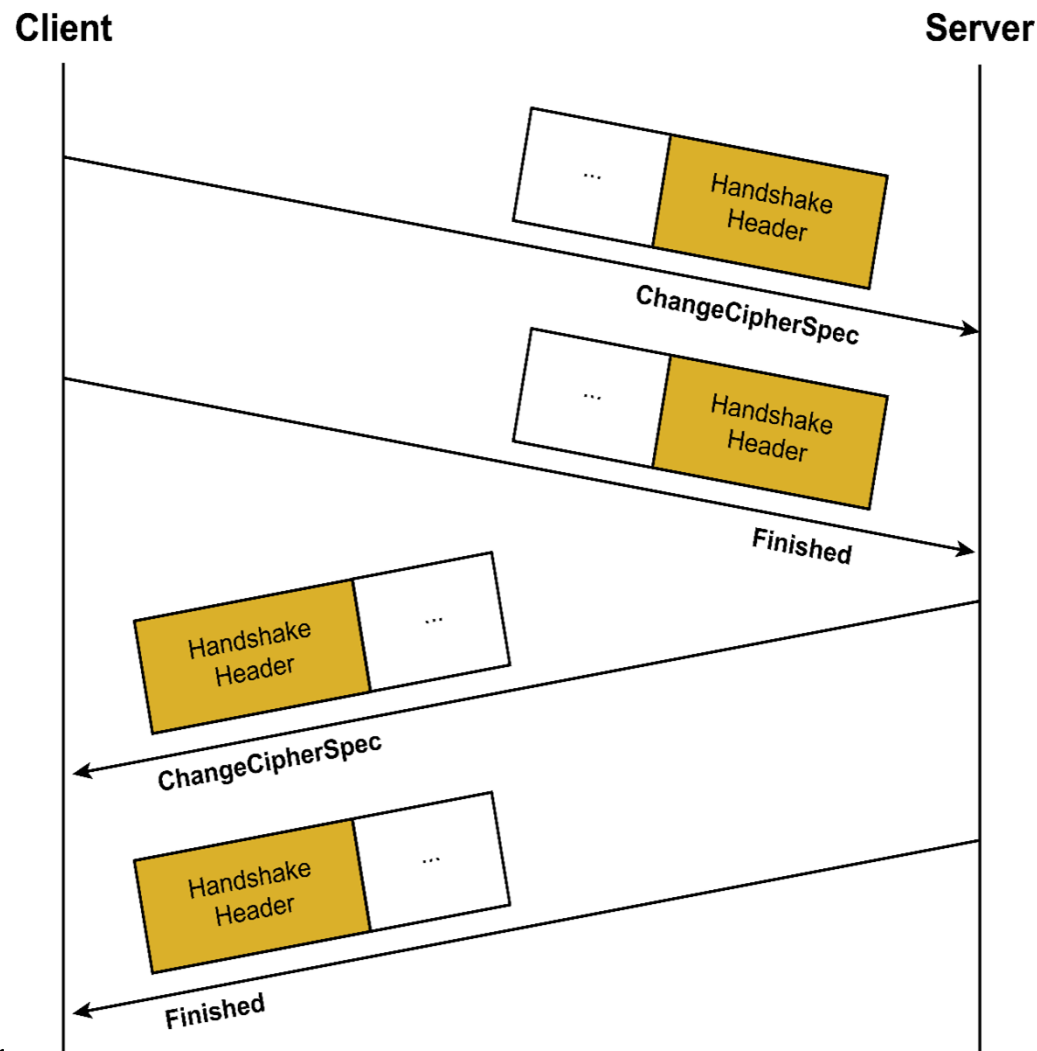
**KH** = „Keyed-Hashing for Message Authentication“-Verfahren; HMAC-Verfahren



# TLS/SSL - Verbindungsaufbau

## → Phase 4: Übersicht

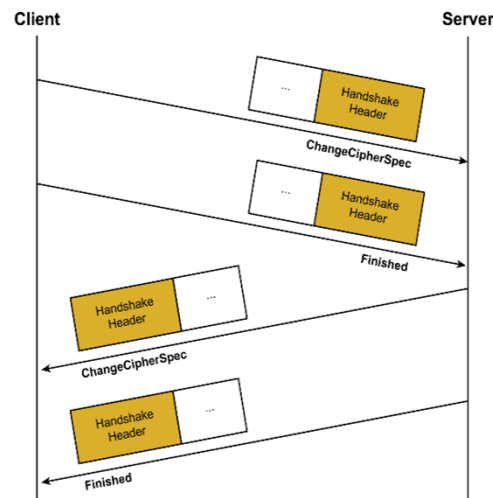
Cipher Suite wird aktiviert und der Handshake beendet



# TLS/SSL - Verbindungsaufbau

## → Phase 4: ChangeCipherSec + Finished

- Die Phase 4 beendet den Handshake.
- Mit der „ChangeCipherSpec“-Nachricht (CCS-Protokoll) zeigen Client und Server, dass sie ab jetzt die ausgehandelten Verfahren/Parameter nutzen.
- Mit der „Finished (20)“-Nachricht symbolisieren beide, dass ihr Teil des Verbindungsaufbaus abgeschlossen ist.
- Diese Meldung ist die erste Nachricht, die bereits mit den neuen Sicherheitsparametern für das Record Protokoll behandelt wird.







# TLS/SSL-Technologie

Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- **Domänen-Zertifikaten**
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- Zusammenfassung

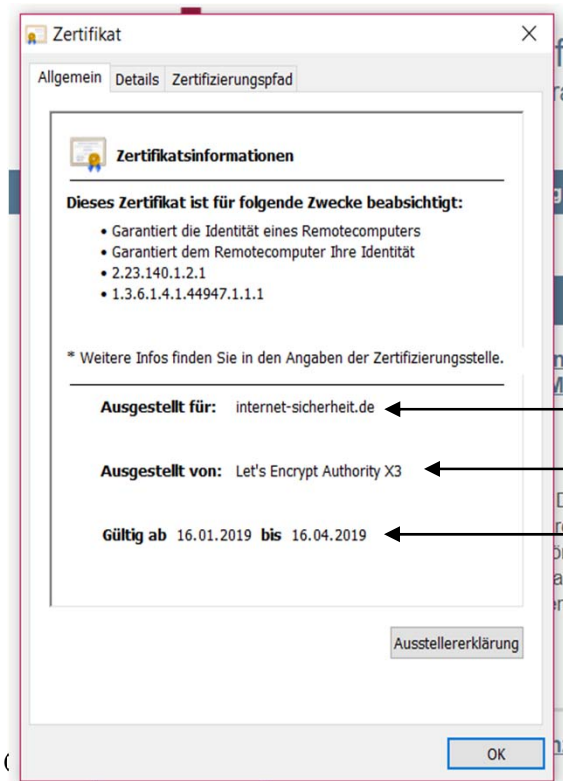
## → Domänen-Zertifikate

---

- **Domänen-Zertifikate helfen, Attribute von Domänen zu überprüfen.**
- **Wesentliche Inhalte eines Domänen-Zertifikates** sind:
  - Name der Organisation, deren Authentizität durch dieses Domänen-Zertifikat bestätigt wird
  - Public-Key der Organisation (Domäne)
  - Name der ausstellenden Zertifizierungsstelle
  - Gültigkeit des Domänen-Zertifikats
- **Aufgabe eines Domänen-Zertifikats:**
  - Eindeutige Zuordnung eines Public-Key's zu einer Organisation.
  - Für die Korrektheit dieser Zuordnung und dass die Organisation, die den passenden geheimen Schlüssel besitzt, auch existiert, verbürgt sich die ausstellende Zertifizierungsstelle.
  - Diese signiert das Domänen-Zertifikat, wodurch es für Dritte ohne die Kenntnis des geheimen Schlüssels der Zertifizierungsstelle unmöglich wird, das Domänen-Zertifikat zu verändern.

# TLS/SSL – Domänen-Zertifikate

## → Zertifikatsinformationen – if(is)



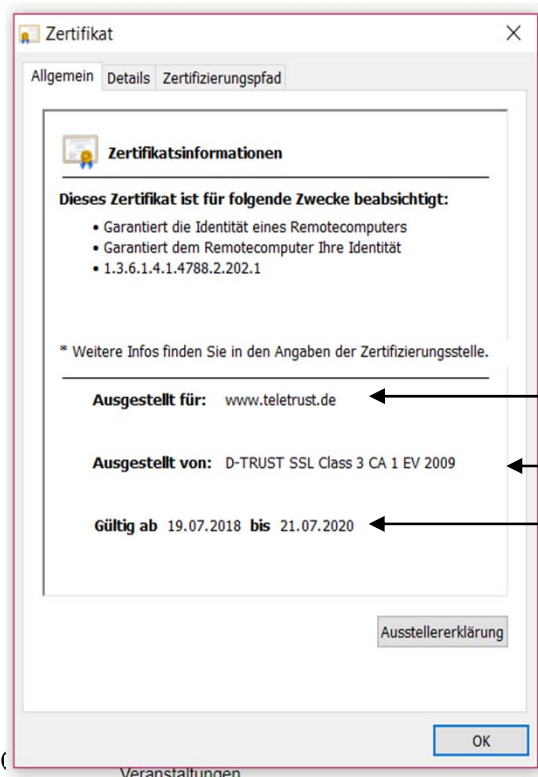
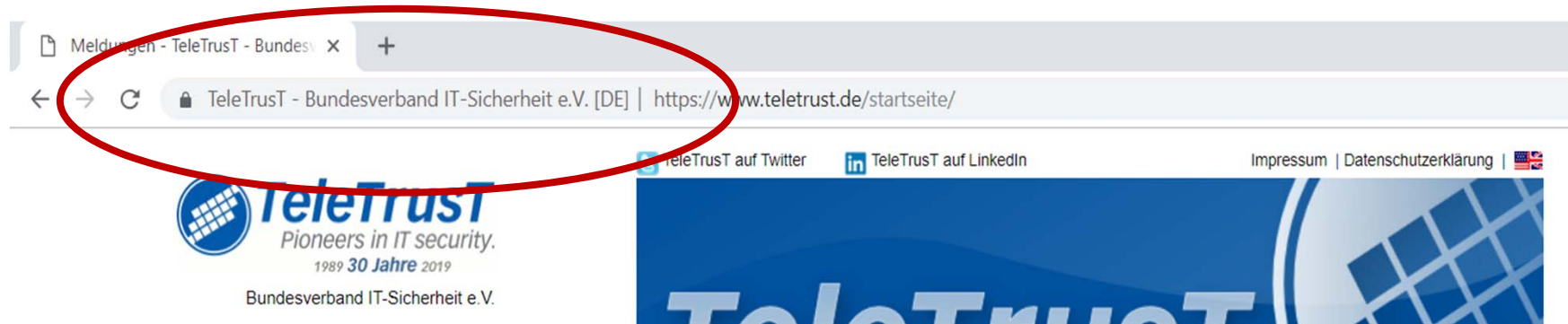
*Domänen-Namen*

*Herausgeber des Zertifikates*

*Gültigkeitszeitraum*

# TLS/SSL – Domänen-Zertifikate

## → Zertifikatsinformationen – if(is)



*Domänen-Namen*

*Herausgeber des Zertifikates*

*Gültigkeitszeitraum*



# TLS/SSL – Domänen-Zertifikate

## → Wichtige Aspekte (1/2)

---

- Eine Domäne muss schon **registriert sein, bevor** ein TLS/SSL-Zertifikat **beantragt wird**, weil Zertifizierungsstellen, die ein Zertifikat ausstellen, die Domain-Inhaberschaft überprüfen müssen.
- Bezüglich der Überprüfung der Domain-Inhaberschaft gibt es verschiedene Vertrauensstufen:
  - Domain Validated (DV)
  - Organisation Validated (OV)
  - Extended Validation (EV)
- **Domain Validated (DV)-Zertifikate:**
  - DV-Zertifikate sind die **einfachste Art** von TLS/SSL-Zertifikaten.
  - DV steht für Domain Validation (Validierung/Überprüfung der Domain).
  - Eine Zertifizierungsstelle bestätigt damit, dass zum Beispiel der Website-Inhaber die **administrative Kontrolle über die Domain** nachweisen kann.
  - Sie enthalten wenige Identitätsinformationen im Zertifikat, zum Beispiel enthalten sie keinerlei Unternehmensinformationen.



# TLS/SSL – Domänen-Zertifikate

## → Wichtige Aspekte (2/2)

---

### ■ Organization Validated (OV)-Zertifikate:

- OV-Zertifikate enthalten eine Unternehmensverifizierung.
- Das heißt, es sind Informationen zum jeweiligen Unternehmen enthalten.
- Im Gegensatz zu EV-Zertifikaten werden diese Informationen allerdings nicht so prominent angezeigt.
- Um die Identitätsdaten eines Unternehmens zu erkennen, müssen sich die Besucher der Webseite die Zertifikatdetails ansehen.

### ■ Extended Validation (EV)-Zertifikate:

- EV-Zertifikate enthalten die meisten Unternehmensdaten, und Firmen müssen bei dieser Variante die strengsten Anforderungen erfüllen, bevor sie ein solches TLS/SSL-Zertifikat erhalten.
- Hier steht die verifizierte Identität eines Unternehmens im Mittelpunkt und verleiht so der Webseite **das höchste Maß an Glaubwürdigkeit**: der Name des betreffenden Unternehmens wird deutlich in der grünen Adresszeile eines Browsers angezeigt.



# TLS/SSL – Domänen-Zertifikate

## → Root-Zertifikat

- Ein Root-Zertifikat ist das oberste Zertifikat im Verzeichnisbaum.
- Damit ein Domain-Zertifikat vom Browser geprüft werden kann, muss dem Browser das verwendete Root-Zertifikat der Zertifizierungsstelle bekannt sein.
- Dazu können Root-Zertifikate zusätzlich zu den im Browser schon vorhandenen installiert werden.
- Die gängigsten Browser haben heutzutage schon die wichtigsten Root-Zertifikate der größten Zertifizierungsstellen vorinstalliert.
- Einmal akzeptierte Zertifikate können natürlich auch im Browser verwaltet, also aufgelistet und gegebenenfalls gelöscht werden.
- Besucht man eine Seite, deren Zertifikat bzw. Zertifizierungsstelle dem Browser unbekannt ist, so erhält der Nutzer einen Hinweis, mit der Möglichkeit, das Zertifikat zu prüfen, ihm einmalig oder dauerhaft zu vertrauen, oder das Zertifikat in den Browser zu laden, wodurch zukünftig an dieser Stelle kein Hinweis mehr erscheinen würde.



# TLS/SSL-Zertifikate

## → TLS/SSL Authentikation: Bewertung

---

- **Die Qualität der TLS/SSL-Client-Authentikation hängt ab von:**
  - Der Vertrauenswürdigkeit der Zertifizierungsinstanz
  - Dem Level, auf dem die Zertifizierungsinstanz die Authentizität des Teilnehmers überprüft
  - Der Bereitstellung einer CRL (Certificate Revocation List) durch die Zertifizierungsinstanz
- **Verschiedene Zertifizierungs-Anbieter bieten unterschiedliche Klassen von Zertifikaten an:**
  - Ein Zertifikat der Klasse 1 bekommt der Teilnehmer schon nach der Zusendung des Namens und der E-Mail-Adresse, ohne dass seine Identität näher geprüft wird.
  - Für ein Zertifikat der Klasse 2 wird z.B. die Zusendung einer Kopie des Ausweises oder des Führerscheins verlangt.
  - Bei höheren Klassen muss der Teilnehmer sich persönlich (z.B. mit Hilfe des Ausweises) bei einer Registration Authority (RA) authentisieren.
  - Hierbei stellt sich die Frage, wie vertrauenswürdig eine solche Registration Authority (RA) ist.





# TLS/SSL-Technologie

Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- **TLS/SSL**  
**Authentifikationsmethoden**
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt



# Authentifikationsmethoden

## → Einführung

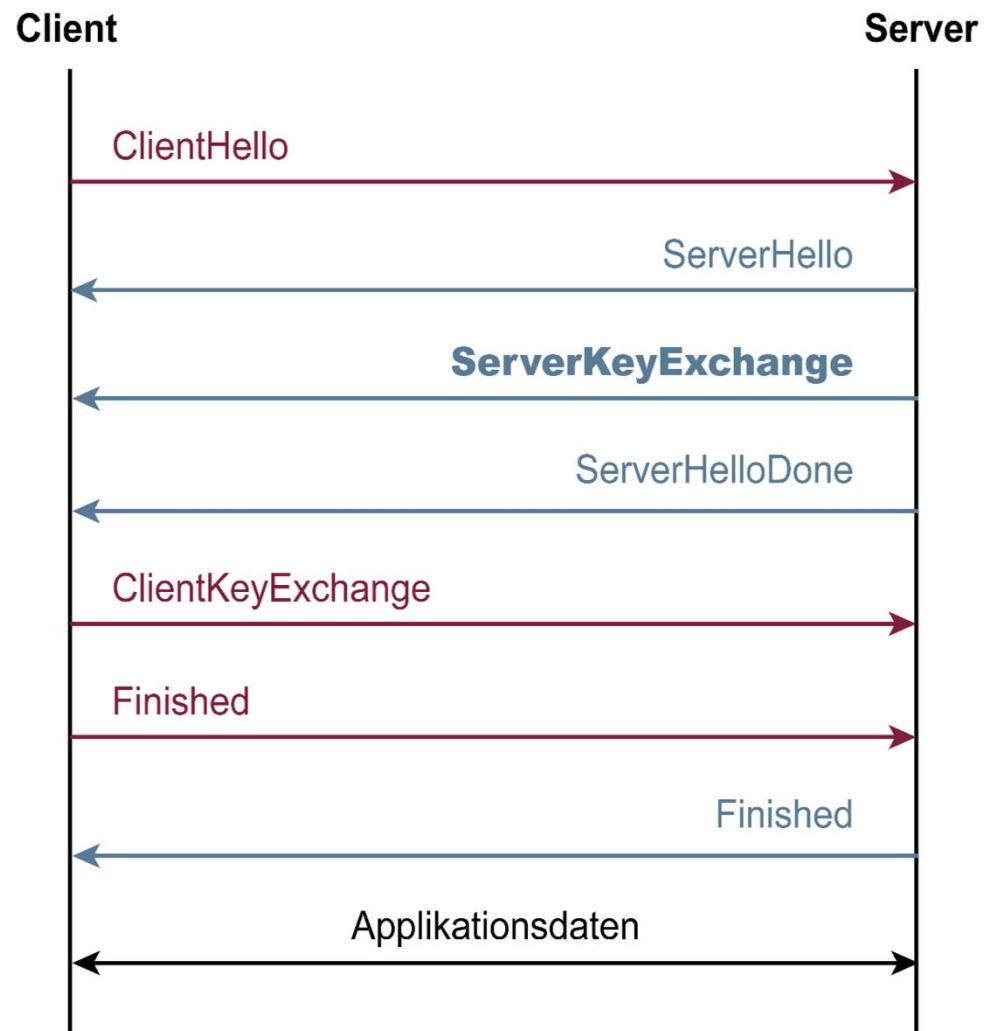
- TLS/SSL benutzt Public Key Zertifikate für die Authentifizierung von Server und Client.
- **Es werden drei 3 Verbindungsarten unterschieden:**
  - 1. Server und Client ohne Authentifizierung
  - 2. Server authentifiziert, Client anonym (gängigste Art)
  - 3. Server und Client authentifiziert



# Authentifikationsmethoden

## → Server und Client ohne Authentifizierung

- Bei dieser Verbindungsart verlangt weder der Server noch der Client eine Authentifizierung seines Kommunikationspartners durch ein Zertifikat.
- Nach Beendigung des Verbindungsaufbaus können Applikationsdaten ausgetauscht werden.
- Die **beiderseits anonyme Verbindung** ist nicht sicher gegenüber aktiven Man-In-The-Middle-Angriffen.
- **Diese Verbindungsart sollte daher vermieden werden!**

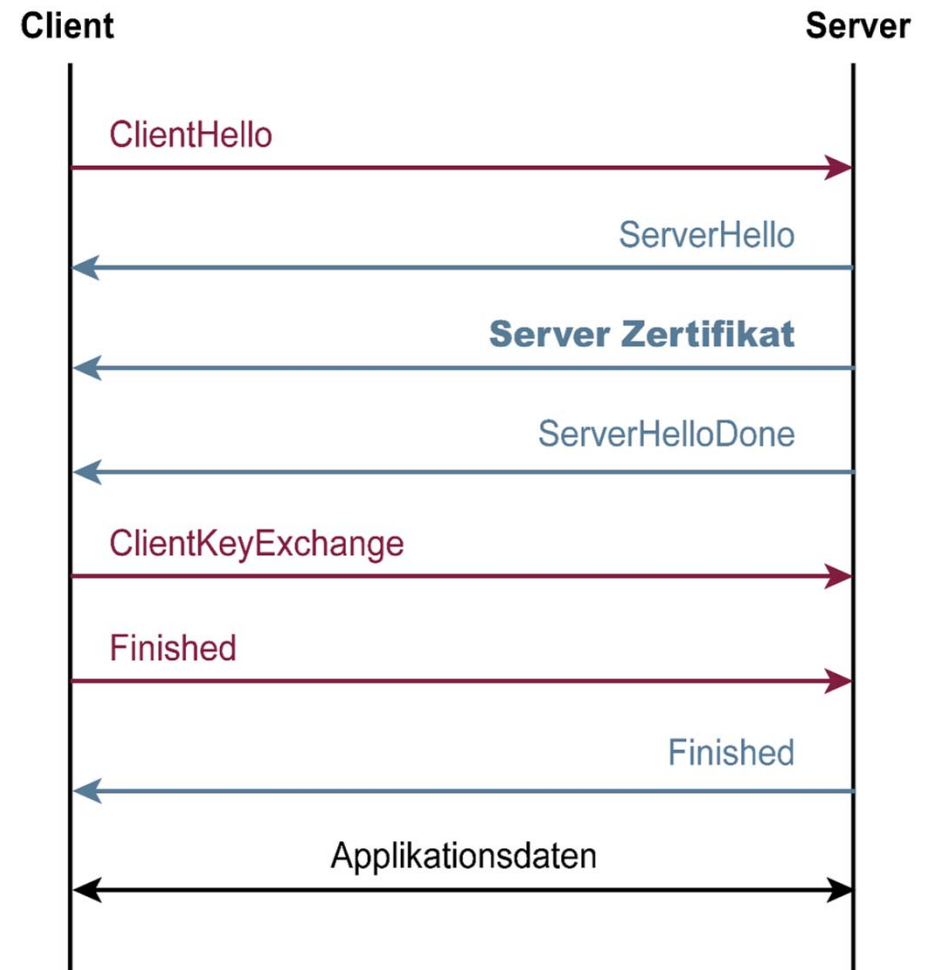




# Authentifikationsmethoden

## → Server authentifiziert, Client anonym

- Hier teilt der **Server** seinen öffentlichen Schlüssel dem Client nicht durch eine ServerKeyExchange-Nachricht mit, sondern durch die **Übermittlung seines Zertifikates**.
- Kann der Client das **Zertifikat verifizieren**, so kann sich dieser **sicher sein**, dass nach einem erfolgreichen Verbindungsaufbau eine TLS/SSL-Verbindung zu genau jenem Server zustande gekommen ist, dessen Zertifikat empfangen wurde.
- Hier kann sich ein Man-In-The-Middle in Richtung des Clients nun nicht mehr als falscher Server ausgeben, womit durch die Server-Authentifizierung aktive Angriffe verhindert werden.





# Authentifikationsmethoden

## → Server Authentikation im Detail

---

### Ablauf:

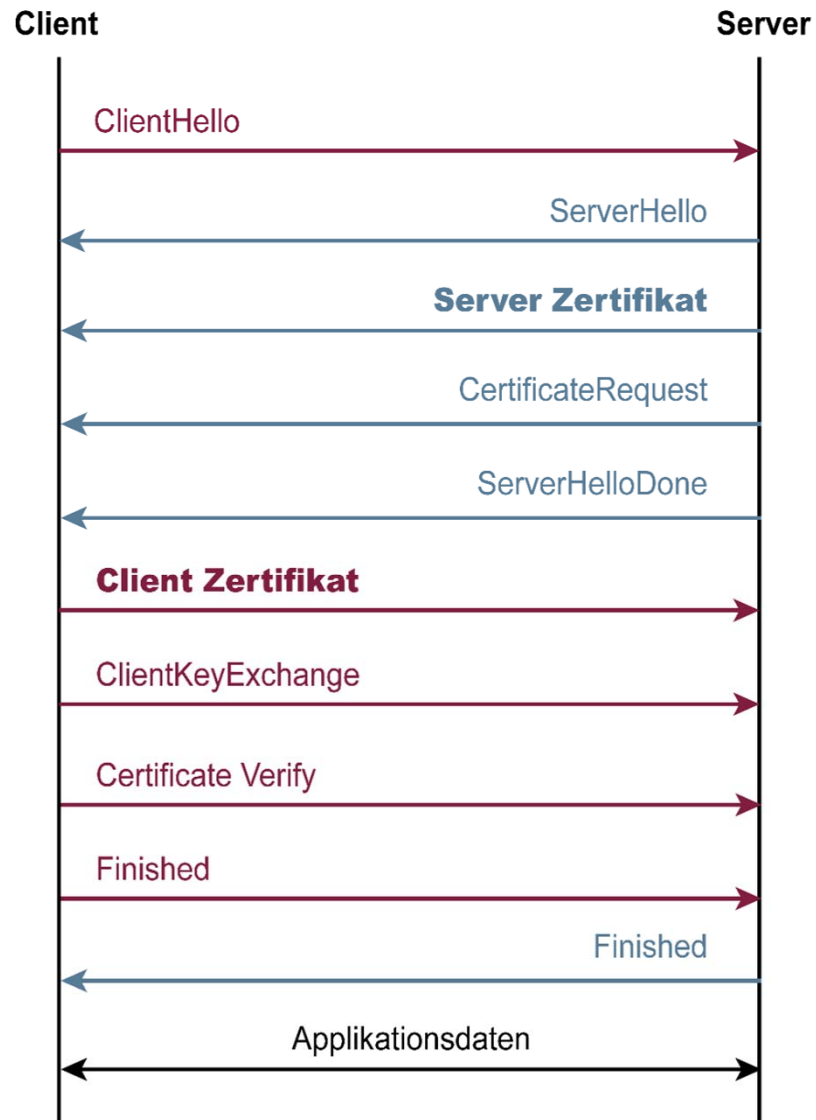
1. Client wählt Internetseite an
2. Server übermittelt dem Client sein öffentliches Domänen-Zertifikat  
mit dem öffentlichen RSA-Schlüssel
3. Client prüft das Domänen-Zertifikat auf Gültigkeit
4. Client generiert Zufallszahl (Pre-Master Secret), verschlüsselt diese für den Server mit dem öffentlichen RSA aus dem Domänen-Zertifikat
5. Server entschlüsselt die Zufallszahl (Pre-Master Secret), generiert die Session Keys und verschlüsselt die Daten nun mit dem Sessionkey für den Client.
6. Verschlüsselter Datenaustausch zwischen Server und Client, und damit die Gewissheit, dass der Server der echte ist.



# Authentifikationsmethoden

## → Server und Client authentifiziert

- Der Server kann mit der CertificateRequest-Nachricht auch eine Client-Authentifikation über ein Zertifikat des Clients beantragen.
- Dieser stellt das Zertifikat mit der ClientCertificate-Nachricht zu und beweist mit der CertificateVerify-Nachricht, dass er auch tatsächlich jener ist, dessen Name im Zertifikat erscheint.
- Falls der Client kein Zertifikat besitzt, welches in der vom Server bekannt gegebenen Liste auftaucht, wird der Verbindungsaufbau abgebrochen.





# TLS/SSL-Technologie

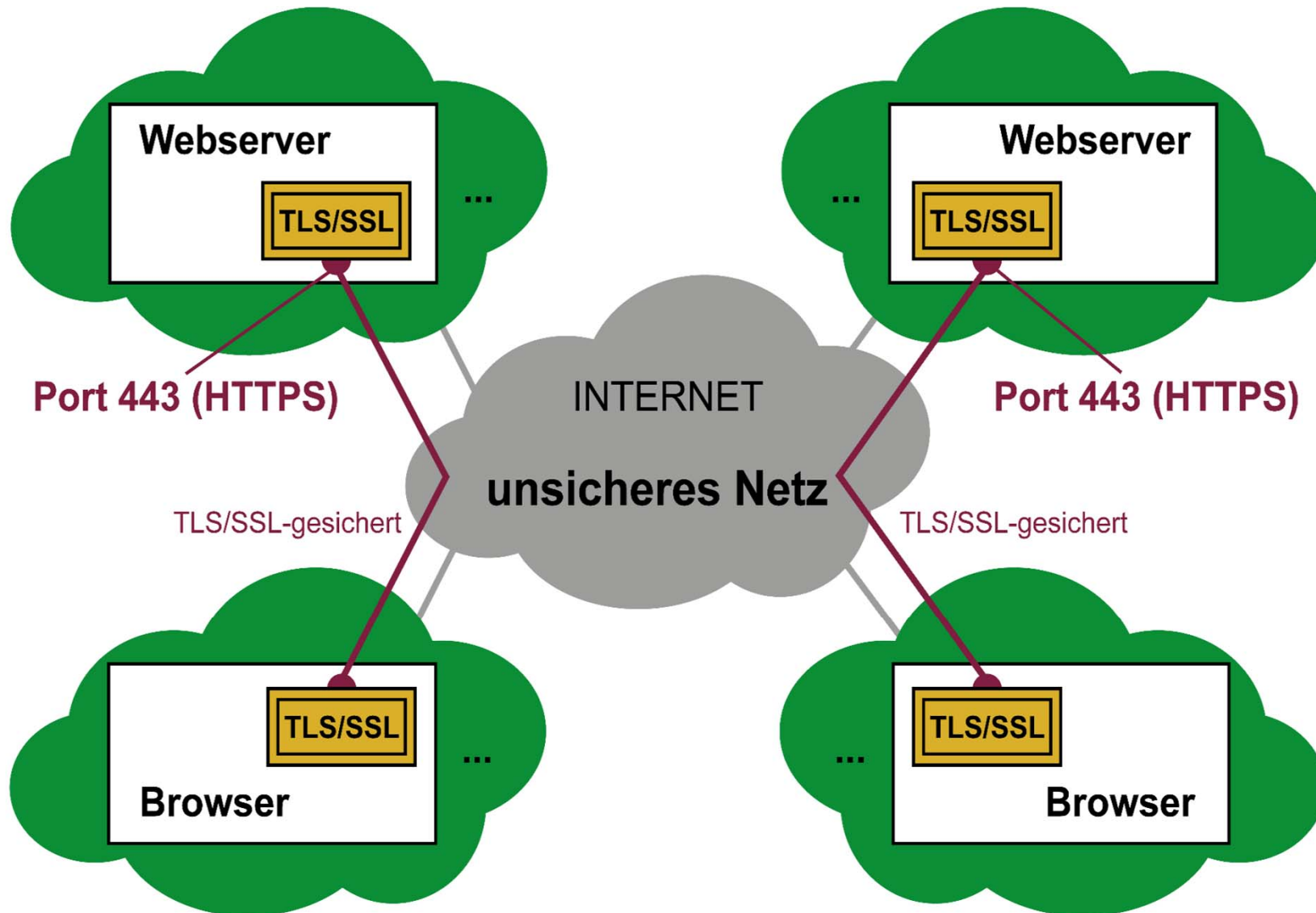
→ Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- **TLS/SSL Anwendungsformen**
- TLS/SSL Protokollmitschnitt
- Zusammenfassung



## → Gesicherte Web-Kommunikation

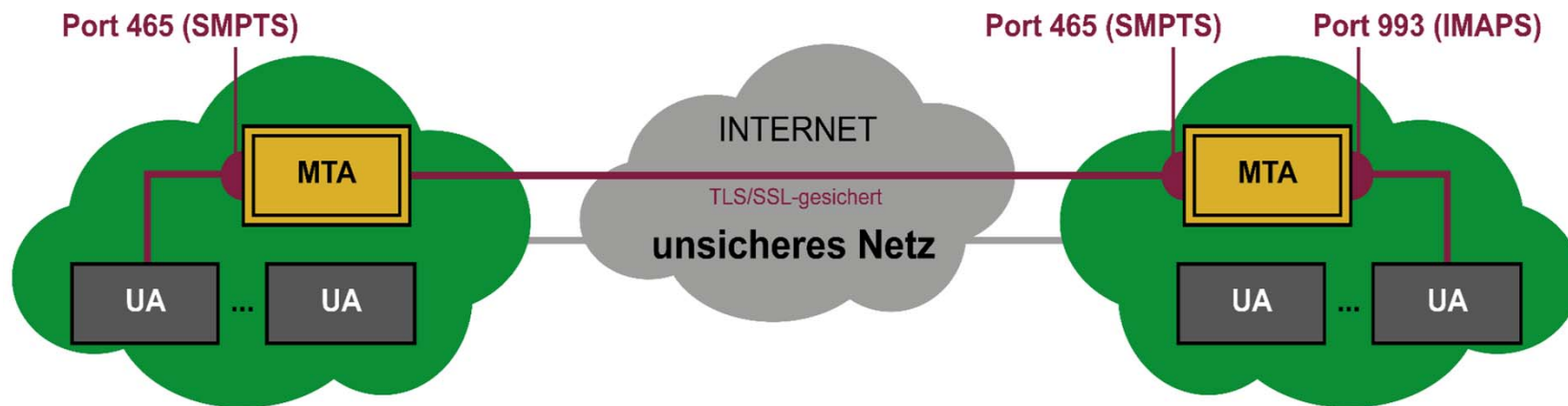






# Anwendungsformen

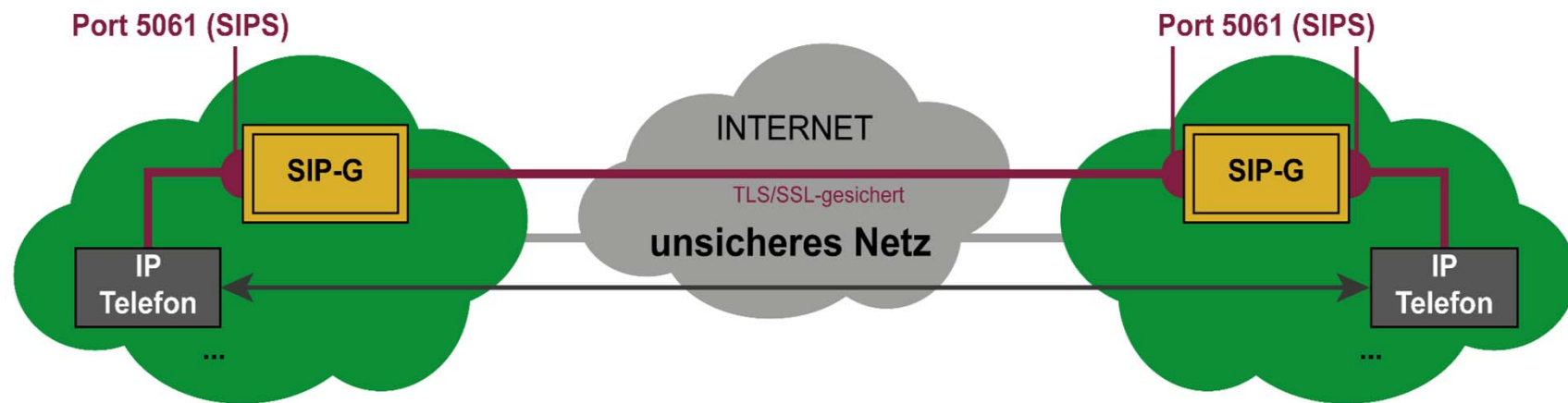
## → Gesicherte Mail-Kommunikation





# Anwendungsformen

## → Gesicherte SIP-Kommunikation





# TLS/SSL-Technologie

Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

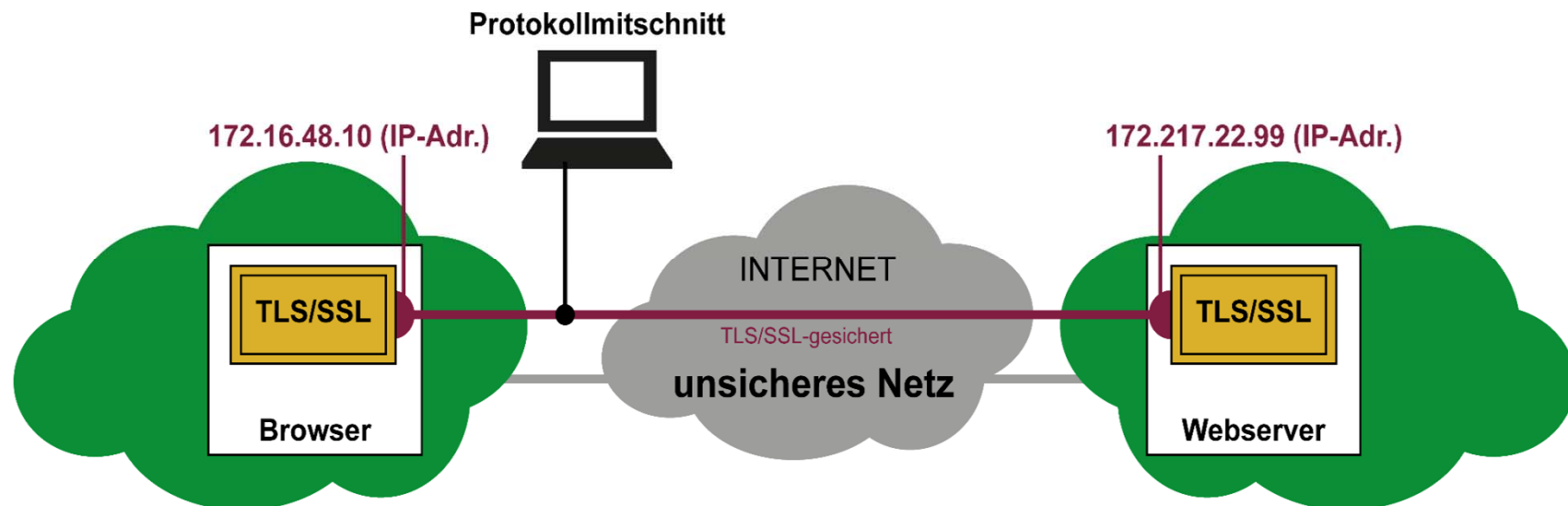
## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- **TLS/SSL Protokollmitschnitt**
- Zusammenfassung

# Protokollmitschnitt

## → TLS/SSL-Protokoll - Übersicht





## TCP-Verbindungsaufbau

**Frame 1:** 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0      **C >>> S**  
Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 0, Len: 0  
**Flags: 0x002 (SYN)**

**Frame 2:** 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0      **C <<< S**  
Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)  
Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 0, Ack: 1, Len: 0  
**Flags: 0x012 (SYN, ACK)**

**Frame 3:** 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0      **C >>> S**  
Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 1, Ack: 1, Len: 0  
**Flags: 0x010 (ACK)**

## → TLS/SSL-Protokoll – Teil (2/11)

### Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch

**Frame 4:** 236 bytes on wire (1888 bits), 236 bytes captured (1888 bits) on interface 0      **C >>> S**  
 Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
 Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 1, Ack: 1, Len: 170

#### Secure Sockets Layer

TLSv1.2 Record Layer: Handshake Protocol: Client Hello

#### Handshake Protocol: Client Hello

##### Random

gmt\_unix\_time: Jan 23, 2025 15:01:26.000000000 CET

random\_bytes: 3f823c9ec03c535a245b6a5e78212356556bcf188d4b3a9f...

##### Cipher Suites (11 suites)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02b)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA (0xc00a)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0xc014)

Cipher Suite: TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0039)

Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA (0xc009)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0xc013)

Cipher Suite: TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x0033)

Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x002f)

Cipher Suite: TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0x000a)

Compression Methods (1 method)

Compression Method: null (0) ...



# Protokollmitschnitt

## → TLS/SSL-Protokoll – Teil (3/11)

*Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch*

### **Frame 5**

#### **TCP- Bestätigung für Frame 4**

**Frame 5:** 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0

**C <<< S**

Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)

Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 1, Ack: 171, Len: 0

**Flags: 0x010 (ACK)**



## → TLS/SSL-Protokoll – Teil (4/11)

### *Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch*

**Frame 6:** 2780 bytes on wire (22240 bits), 2780 bytes captured (22240 bits) on interface 0

**C <<< S**

Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)

Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 1, Ack: 171, Len: 2714

Flags: 0x018 (PSH, ACK)

#### **Secure Sockets Layer**

TLSv1.2 Record Layer: Handshake Protocol: Server Hello

#### **Handshake Protocol: Server Hello**

##### **Random**

**gmt\_unix\_time:** Aug 28, 2018 11:12:01.000000000 CEST

**random\_bytes:** 32b5bbb6291f31b56d071ceaae14e1e7be0863ac6c7c3a06...

Session ID Length: 0

**Cipher Suite:** TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)

Compression Method: null (0)

Extensions Length: 24

Extension: renegotiation\_info

Type: renegotiation\_info (0xff01)

Length: 1

Renegotiation Info extension

Renegotiation info extension length: 0



**→ TLS/SSL-Protokoll – Teil (5/11)**

---

**Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch**

... (Fame 6)

**TLSv1.2 Record Layer: Handshake Protocol: Certificate**

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 2289

**Handshake Protocol: Certificate**

Certificates Length: 2282

Certificates (2282 bytes)

Certificate Length: 1156

**Certificate (id-at-commonName=www.google.de,id-at-organizationName=Google LLC,id-at-localityName=Mountain View,id-at-stateOrProvinceName=California,id-at-countryName=US)**

**signedCertificate**

**version: v3 (2)**

**serialNumber: 25340433**

**signature (sha256WithRSAEncryption)**

**Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)**

**issuer: rdnSequence (0)**

**rdnSequence: 3 items (id-at-commonName=Google Internet Authority G3,id-at-organizationName=Google Trust Services,id-at-countryName=US)**



## → TLS/SSL-Protokoll – Teil (6/11)

---

### Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch

... (Frame 6)

#### **TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange**

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 333

Handshake Protocol: Server Key Exchange

Length: 329

EC Diffie-Hellman Server Params

curve\_type: named\_curve (0x03)

named\_curve: secp256r1 (0x0017)

Pubkey Length: 65

**pubkey: 04fa540dea1e37e332080d8d9ecafec2764a50ecb689b78d...**

Signature Hash Algorithm: 0x0401

Signature Hash Algorithm Hash: SHA256 (4)

Signature Hash Algorithm Signature: RSA (1)

Signature Length: 256

signature: 14ed6843402163b44f1d5699c329732ad3856c43b4792361...

#### **TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done**

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 4

Handshake Protocol: Server Hello Done



# Protokollmitschnitt

## → TLS/SSL-Protokoll – Teil (7/11)

*Aushandlung der Sicherheitsparameter, Serverauthentifizierung und Schlüsselaustausch*

### **TCP- Bestätigung für Frame 6**

**Frame 7:** 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0      **C >>> S**  
Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 171, Ack:  
2715, Len 0  
    **Flags: 0x010 (ACK)**



## → TLS/SSL-Protokoll – Teil (8/11)

### Schlüsselaustausch, CCS, ...

**Frame 8:** 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits) on interface 0 **C >>> S**  
 Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
 Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 171, Ack: 2715,  
 Len: 126

Flags: 0x018 (PSH, ACK)

#### Secure Sockets Layer

TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange

##### Handshake Protocol: Client Key Exchange

Length: 66

EC Diffie-Hellman Client Params

Pubkey Length: 65

**pubkey: 0418395fd6c8855b0e43de39ad413466f3a4c513e8e58c2e...**

TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 40

Handshake Protocol: Hello Request

Handshake Type: Hello Request (0)



## → TLS/SSL-Protokoll – Teil (9/11)

CCS, ...

**Frame 9:** 345 bytes on wire (2760 bits), 345 bytes captured (2760 bits) on interface 0      C <<< S  
Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)  
Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 2715, Ack: 297,  
Len: 279

Flags: 0x018 (PSH, ACK)

### Secure Sockets Layer

TLSv1.2 Record Layer: Handshake Protocol: New Session Ticket

#### Handshake Protocol: New Session Ticket

Handshake Type: New Session Ticket (4)

Length: 219

TLS Session Ticket

Session Ticket Lifetime Hint: 100800

Session Ticket Length: 213

**Session Ticket:** 00f7fe033d2ec55ea45919d7b87195bf3247ae08e9462ea4...

TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 40

Handshake Protocol: Hello Request



# Protokollmitschnitt

## → TLS/SSL-Protokoll – Teil (10/11)

### Verschlüsselte und integritätsgesicherte Datenübertragung

**Frame 10:** 229 bytes on wire (1832 bits), 229 bytes captured (1832 bits) on interface 0 **C >>> S**  
 Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
 Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 297, Ack: 2994,  
 Len: 163

#### Secure Sockets Layer

##### TLSv1.2 Record Layer: Application Data Protocol: http

Content Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 158

**Encrypted Application Data:** 000000000000000019758b52e6af38f292ada9eafc1118770...

**Frame 11:** 454 bytes on wire (3632 bits), 454 bytes captured (3632 bits) on interface 0 **C >>> S**  
 Internet Protocol Version 4, Src: 172.16.48.10 (172.16.48.10), Dst: 172.217.22.99 (172.217.22.99)  
 Transmission Control Protocol, Src Port: 36102 (36102), Dst Port: https (443), Seq: 460, Ack: 2994,  
 Len: 388

#### Secure Sockets Layer

##### TLSv1.2 Record Layer: Application Data Protocol: http

Content Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 383

**Encrypted Application Data:** 000000000000000025d9a6a016536db15bbcfa2b4e37eb1a3...



## → TLS/SSL-Protokoll – Teil (11/11)

---

### *Verschlüsselte und integritätsgesicherte Datenübertragung*

**Frame 12:** 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 0      **C <<<**  
 Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)  
 Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 2994, Ack: 460,  
 Len: 69

#### **Secure Sockets Layer**

##### **TLSv1.2 Record Layer: Application Data Protocol: http**

Content Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 64

**Encrypted Application Data:** 0000000000000000131f990c1519fe80b83ab43a6ec15eb65...

**Frame 13:** 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0      **C <<< S**  
 Internet Protocol Version 4, Src: 172.217.22.99 (172.217.22.99), Dst: 172.16.48.10 (172.16.48.10)  
 Transmission Control Protocol, Src Port: https (443), Dst Port: 36102 (36102), Seq: 3063, Ack: 460,  
 Len: 38

#### **Secure Sockets Layer**

##### **TLSv1.2 Record Layer: Application Data Protocol: http**

Content Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 33

**Encrypted Application Data:** 0000000000000000211199f018207c918fd98d8e0a3e7a5ca...



# TLS/SSL-Technologie

Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## → Inhalt

---

- Ziele und Ergebnisse der Vorlesung
- Einleitung
- Architektur und Protokolle
- Protokollablauf: Prinzipien, Schritte und Phasen
- Domänen-Zertifikaten
- TLS/SSL Authentifikationsmethoden
- TLS/SSL Anwendungsformen
- TLS/SSL Protokollmitschnitt
- **Zusammenfassung**





# TLS/SSL-Technologie

## → Zusammenfassung (1/3)

---

- **TLS (Transport Layer Security) / SSL (Secure Socket Layer)** ist ein Cyber-Sicherheitsprotokoll.
- TLS/SSL ist **applikationsunabhängig** und setzt logisch auf einem Transportprotokoll auf.
- TLS/SSL bündelt 3 Sicherheitsdienste:
  - **Authentifizierung** von Server und Client
  - **Verschlüsselung** der Daten
  - **Integritäts- und Authentizitätsüberprüfung** der Daten
- Gängigste Protokolle die TLS/SSL nutzen: https, ftps, imaps, pop3s, smtps, telnet, sips, ...
- Die eigentlichen Protokolldaten werden anstatt im eigenen Anwendungsprotokoll über das **Application Data- / Record Layer-Protokoll** von TLS/SSL verschlüsselt und integritätsgesichert übertragen (z.B. HTTP <-> HTTPS).



# TLS/SSL-Technologie

## → Zusammenfassung (2/3)

---

- Anhand der **Portnummer** lässt sich erkennen, um welches ursprüngliche Protokoll es sich bei den übertragenen Daten handelt.
- TLS/SSL besteht aus **zwei Schichten**, nämlich dem Record Layer-Protokoll und den vier TLS/SSL-Teilprotokollen:
  - ChangeCipherSpec
  - Alert
  - Handshake
  - Application Data
- Der Protokollablauf bei TLS/SSL erfolgt in zwei Schritten:
  - 1. Schritt:** Verbindungsaufbau, unterteilt in 4 Phasen:
    - 1. Phase: Aushandlung der Sicherheitsparameter.
    - 2. Phase: Serverauthentisierung (Optional) und Schlüsselaustausch
    - 3. Phase: Clientauthentisierung (Optional) und Schlüsselaustausch
    - 4. Phase: Beendigung des Handshakes
  - 2. Schritt:** Transfer-Mode

*Verschlüsselte und integritätsgesicherte Datenübertragung*



# TLS/SSL-Technologie

## → Zusammenfassung (3/3)

- TLS/SSL beinhaltet 2 wichtige Instanzenkonzepte:
  - TLS/SSL-Session
  - TLS/SSL-Connection
- **Cipher Suites** sind eine Kombination aus **Schlüsselaustauschverfahren**, **Verschlüsselungsverfahren** mit **Schlüssellänge** und ein **Verfahren zum Integritätscheck**.
- Aufgabe eines Domänen-Zertifikats:  
Eindeutige Zuordnung eines Public-Key's zu einer Organisation.
- TLS/SSL bietet 3 Verbindungsarten:
  1. Server und Client ohne Authentifizierung
  - 2. Server authentifiziert, Client anonym**
  3. Server und Client authentifiziert



# IT-Sicherheit

## 9 QUIC-Protokoll http 1.3



**Gerrit Kalkbrenner**  
**Gerrit.Kalkbrenner@hwr-berlin.de**

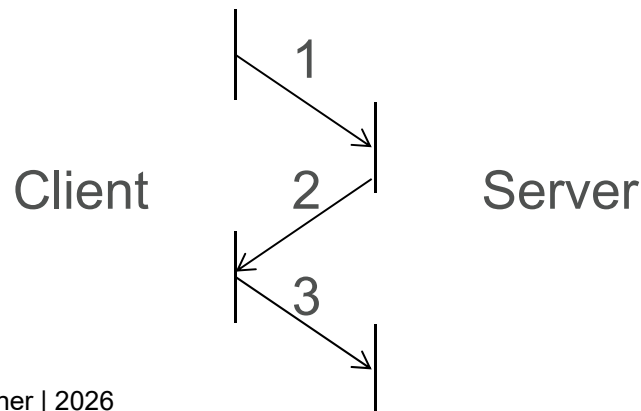
## QUIC - ein neues Transport-Protokoll

- Im Internet bahnt sich eine umwälzende Änderung an, die zu schnelleren Übertragungen führen wird
- von Google angestoßen
- Ablösung von TCP
- Beschleunigter Aufbau von Web-Seiten
- QUIC = Quick UDP Internet Connection



# Entwicklungen

- Rasante Geschwindigkeitssteigerung des Internet
- Selbst Privatanwender haben Gigabit-Anschlüsse
- TCP schöpft Geschwindigkeit nicht aus
- Üblich: 3 Wege-Handshake





# Zugriff auf Web-Seiten

Aufbau einer Verbindung:

1. 3 Wege Handshake für TCP
  2. Handshake für TLS, Aushandeln der Sicherheit
  3. Handshake HTTP
- Klingt nach wenig, summiert sich aber für aus etlichen Teilen bestehenden Web-Seiten
  - Vorgehen so weit wie möglich verkürzen!



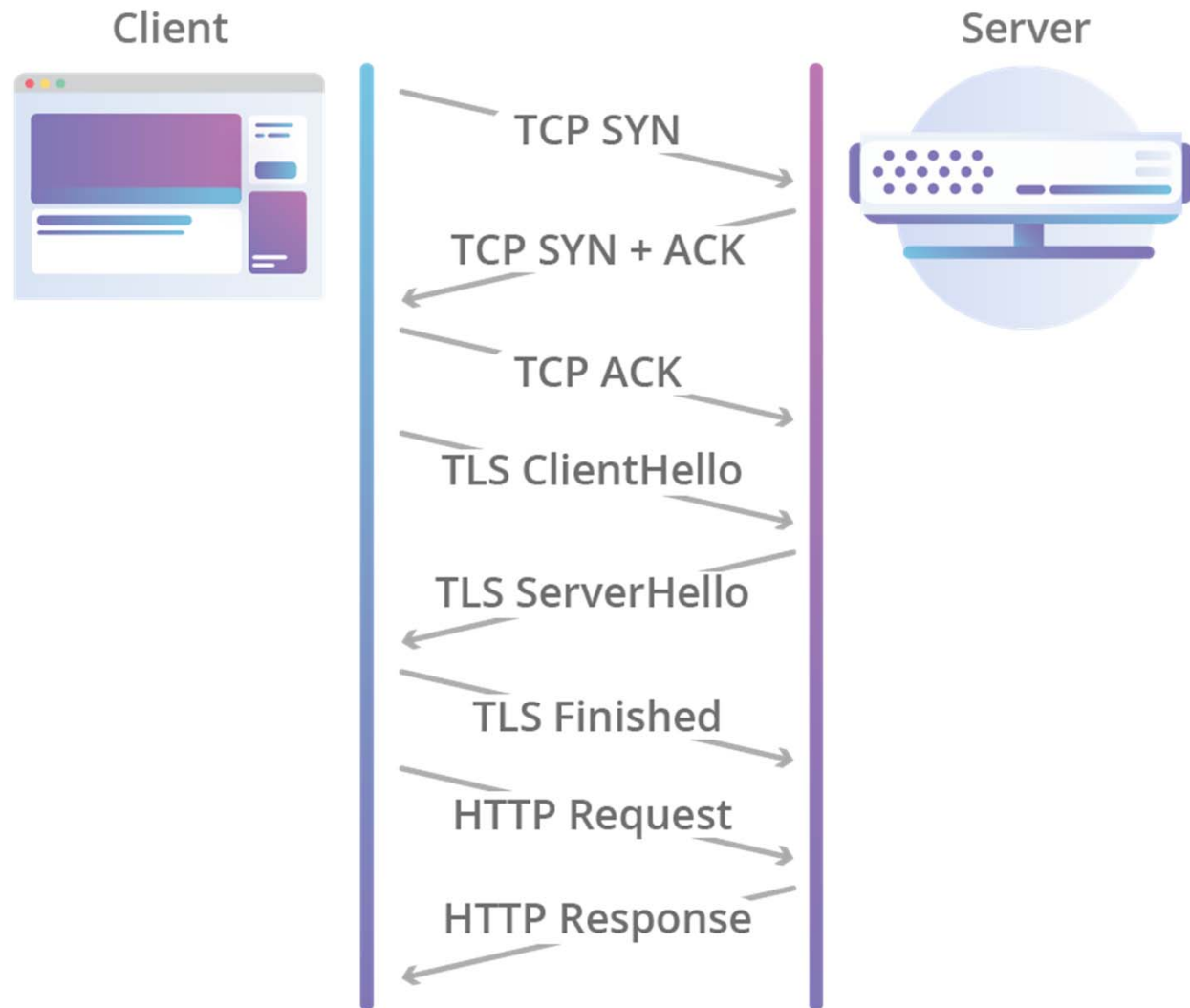
## QUIC - Quick UDP Internet Connections

- 2012 bei Google begonnen
- Verschachtelung der Handshake-Wartezeiten
- Sehr Erfolgsversprechend, daher 2017 Übergabe an die IETF
- Neu dafür: HTTP/3, TLS 1.3
- Inzwischen die Hälfte des Verkehrs zwischen Google Servern und Chrome mit QUIC
- Facebook komplett, beta in Edge, Firefox, Safari



# TCP+ TLS

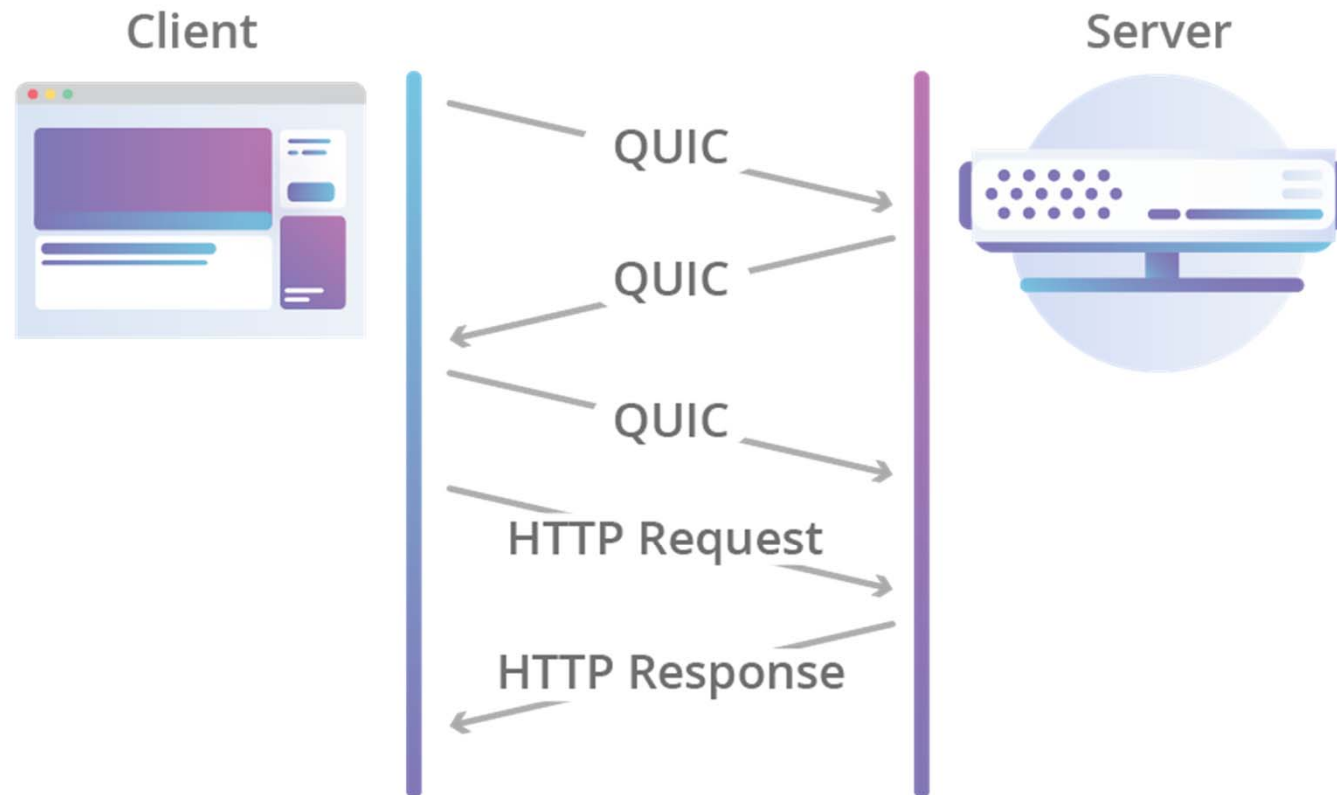
## HTTP Request Over TCP + TLS



Quelle: Alessandro Ghedini

# QUIC

## HTTP Request Over QUIC





## Potenzial

- Bei Round Trip von 200 mS
- HTTPS: oft bis zu 2 Sekunden für den Aufbau
- Komplexes Protokoll: 3 Schritte auf einem mal
- Was passiert, wenn ein Schritt scheitert?
- QUIC bietet ausschließlich Verschlüsselung
- Rechenzentren und NSA wollen gerne mithören



# **IT-Sicherheit**

## **07 Signatur, Zertifikate und PKI**

**Gerrit.Kalkbrenner@hwr-berlin.de**



# Signatur, Zertifikate und PKI

## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- **Digitale Signaturen und Zertifikate**
- **Public-Key-Infrastrukturen**
- **Gesetzlicher Hintergrund**
- **PKI-enabled Application**
- **Zusammenfassung**



# Signatur, Zertifikate und PKI

## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- Digitale Signaturen und Zertifikate
- Public-Key-Infrastrukturen
- Gesetzlicher Hintergrund
- PKI-enabled Application
- Zusammenfassung



# Ziele und Ergebnisse der Vorlesung

## → Signatur, Zertifikate und PKI

- Gutes Verständnis für die Cyber-Sicherheitsprinzipien und Cyber-Sicherheitsmechanismen: **Digitale Signatur** und **elektronische Zertifikate**.
- Erlangen der Kenntnisse über **Public-Key-Infrastrukturen (PKI)** und **Vertrauensmodelle**.
- Gewinnen von praktischen Erfahrungen durch die Betrachtung von Beispielen zu **PKI-enabled Application**.



# Signatur, Zertifikate und PKI

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- **Digitale Signaturen und Zertifikate**
- Public-Key-Infrastrukturen
- Gesetzlicher Hintergrund
- PKI-enabled Application
- Zusammenfassung





# Digitale Signaturen und Zertifikate

## → Eigenhändige Unterschrift (1/2)

- Fragen, die bei einer eigenhändigen Unterschrift gestellt werden, sind:
  - Welchen Wert hat eine eigenhändige Unterschrift?
  - Wer hat etwas davon?
  - Welche Bedeutung hat die eigenhändige Unterschrift?
  - Welche Bedingungen müssen erfüllt sein, damit die Unterschrift einen Vorteil hat?

Dr. Gerd Müller  
Sonnenallee 100a  
1000 Wohlfühlstadt

02.01.2018

Fachgroßhandel für Waschmaschinen  
Aachener Str. 70  
50674 Köln

Sehr geehrter Herr Maier,

hiermit bestelle ich, auf der Grundlage Ihres Angebotes (Nr.345/11/17) vom 13.11.2017, bei Ihnen eine Waschmaschine im Wert von 650 Euro.

Mit freundlichen Grüßen

Gerd Müller



# Digitale Signaturen und Zertifikate

## → Eigenhändige Unterschrift (2/2)

- Funktionen einer eigenhändigen Unterschrift:
  - **Abschlussfunktion**  
Vollendung einer Erklärung - hebt sich vom Entwurf ab
  - **Identitätsfunktion**  
Unterschrift macht die Identität des Ausstellers kenntlich
  - **Echtheitsfunktion**  
Dokument stammt vom Aussteller
  - **Warnfunktion**  
Schutz des Unterzeichners vor Übereilung
  - **Beweisfunktion (Urkundenbeweis)**  
Erleichtert die Beweisführung im Streitfall



# Digitale Signaturen und Zertifikate

## → Digitale Signatur (1/3)

- Signaturerstellung zu einer Nachricht m:

$$s = S ( m, GSA )$$

S: Signaturfunktion, z.B. RSA-Verfahren

m: Nachricht, die signiert werden soll

s: Signatur, z.B. 3.000 Bit Zeichenkette

GSA: geheimer Schlüssel des Nutzers A,  
der die Nachricht signiert

- Verifikation der Signatur der Nachricht m:

$$V ( m, s, ÖSA ) = \text{true} \quad ?$$

V: Verifikationsfunktion

ÖSA: öffentlicher Schlüssel des Nutzers A,  
der die Nachricht signiert hat



# Digitale Signaturen und Zertifikate

## → Digitale Signatur (2/3)

- Public-Key-Verfahren haben eine relativ **hohe Verarbeitungszeit** für eine Operation.
  - Zu signierende Nachricht  $m$  muss kleiner als die verwendete Schlüsselgröße sein.
  - 100 MB = 800.000.000 Bit
    - ca. 400.000 Operationen bei 2.048 Bit Schlüssellänge
    - ca. 11 h bei 100 ms pro Operation
- Die **Zusammengehörigkeit** von Einzelsignaturen ist nicht gegeben.



# Digitale Signaturen und Zertifikate

## → Digitale Signatur (3/3)

- Lösung: **One-Way-Hashfunktion**

$$AV ( h_m = H ( m ), s, \text{ÖSA} ) = \text{true}$$

$h_m$ : Hashwert der Nachricht  $m$

$H$ : One-Way-Hashfunktion

$AV$ : Angepasste Verifikationsfunktion

- **Vorteile:**

- Signierung von **beliebig langen** Nachrichten möglich.
- Geringere **Zeit** für die Erstellung der Signatur nötig.
- Gewährleistung der **Integrität** - Jedes Bit der Nachricht ist in die digitale Signatur eingeschlossen!



# Digitale Signaturen und Zertifikate

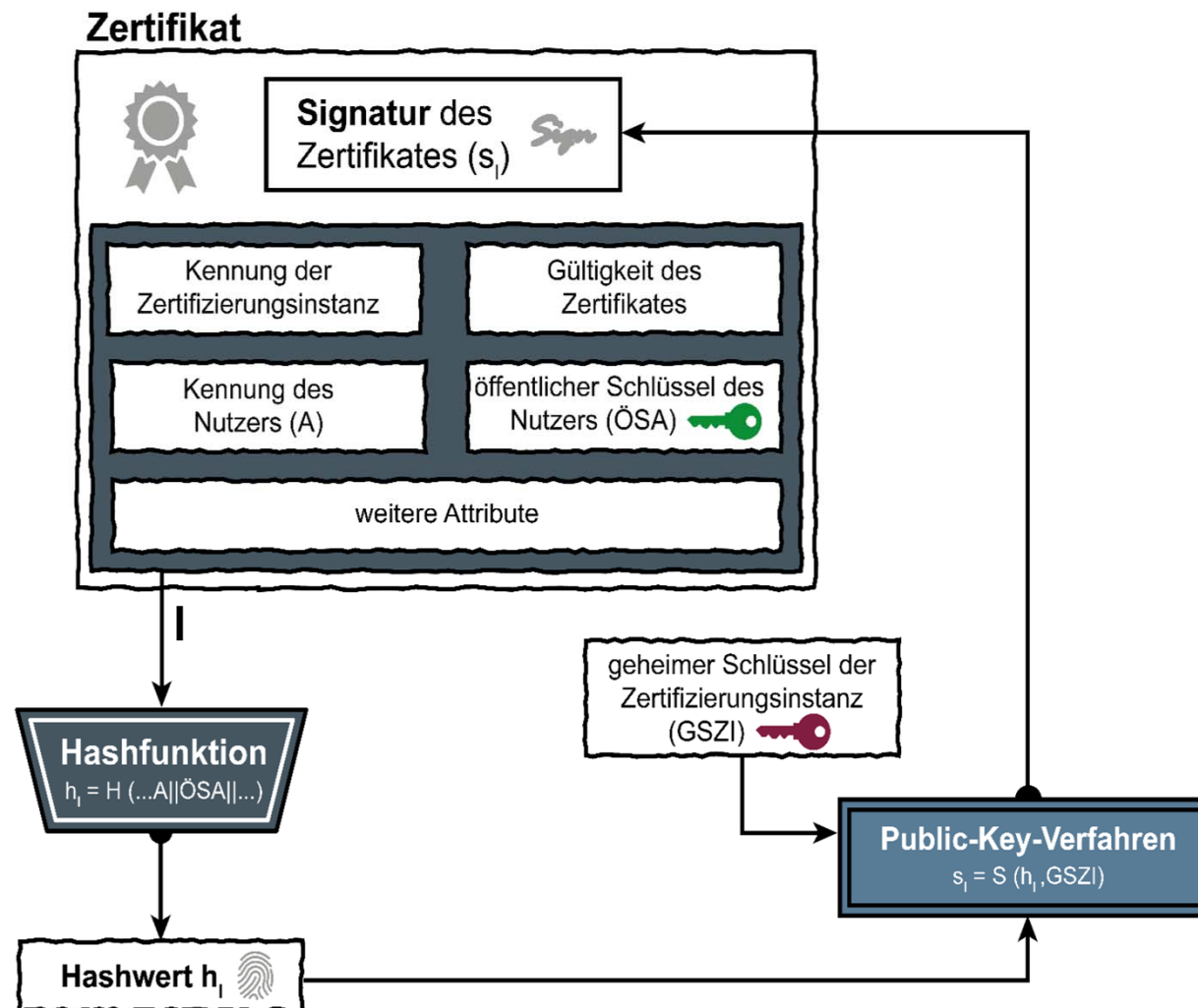
## → Digitale Zertifikate

- Mit Hilfe von elektronischen Zertifikaten können:
  - **Attribute** von Nutzern überprüfbar nachgewiesen werden,
  - die **Authentizität** von öffentlichen Schlüsseln nachgewiesen werden.
- Ausgestellt werden elektronische Zertifikate durch eine **Zertifizierungsinstanz**:
  - Spezialisierte allgemeine Anbieter für Vertrauensdienste, Berufsverbände (z.B. Notare, Steuerberater/ Wirtschaftsprüfer, Ärzte/Krankenschwestern/Hebammen), Personal und IT-Abteilungen von Unternehmen, Behörden.
  - Wirksame Prüfung aller relevanten Daten (z.B. Identität, Ausweis, Urkunden und weitere Attribute).



# Digitale Signaturen und Zertifikate

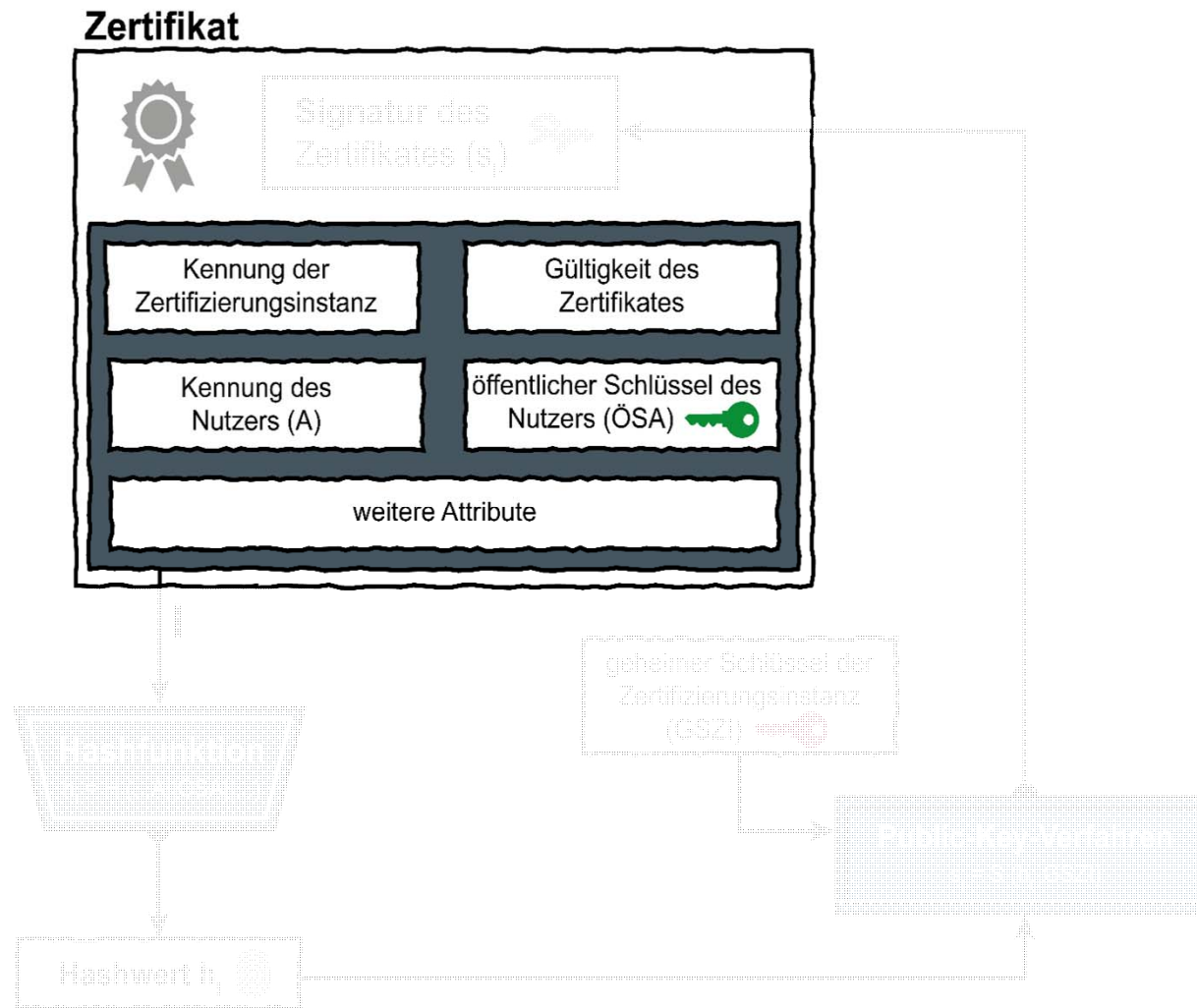
## → Erstellung eines digitalen Zertifikates





# Digitale Signaturen und Zertifikate

## → Erstellung eines digitalen Zertifikates

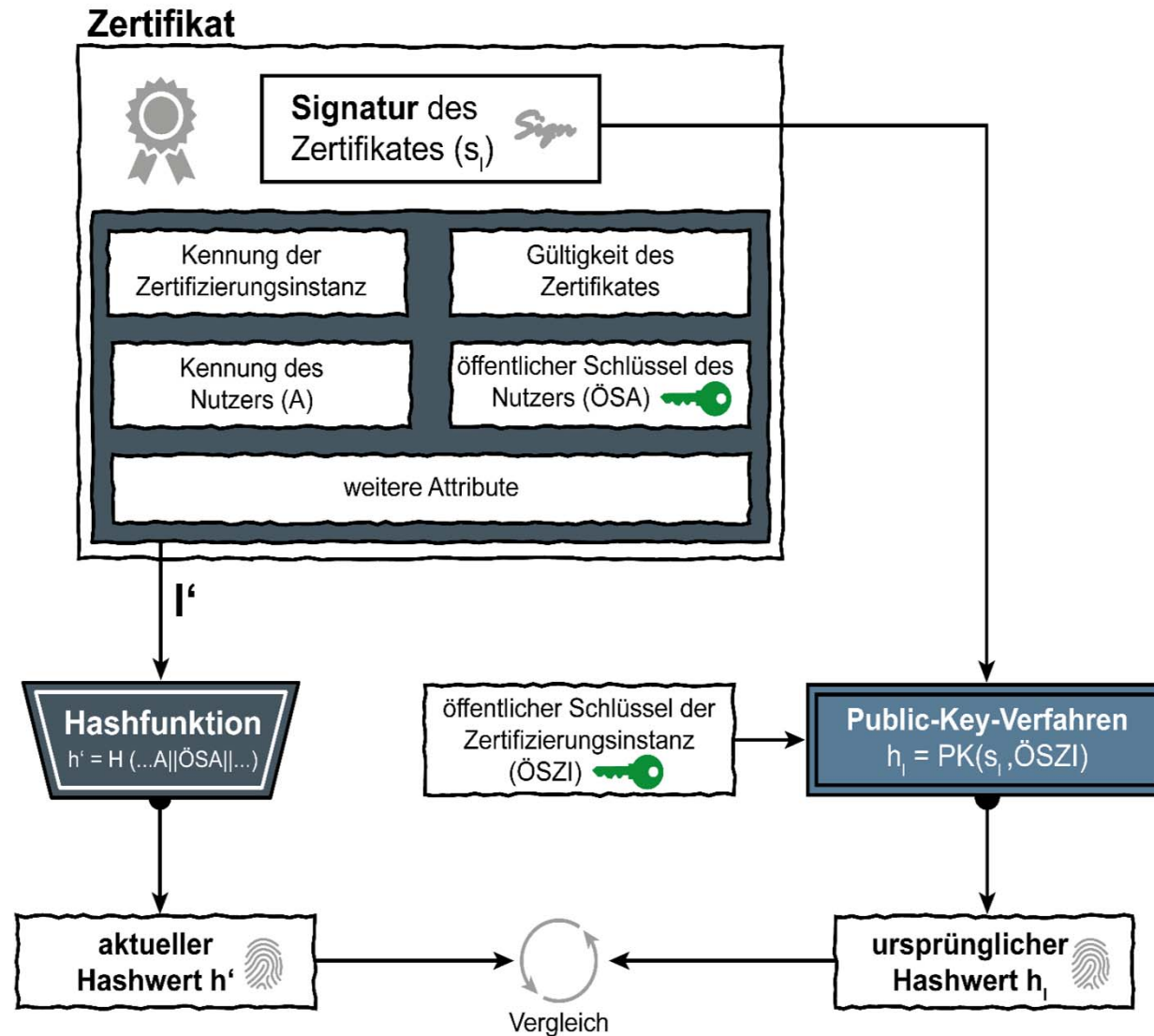






# Digitale Signaturen und Zertifikate

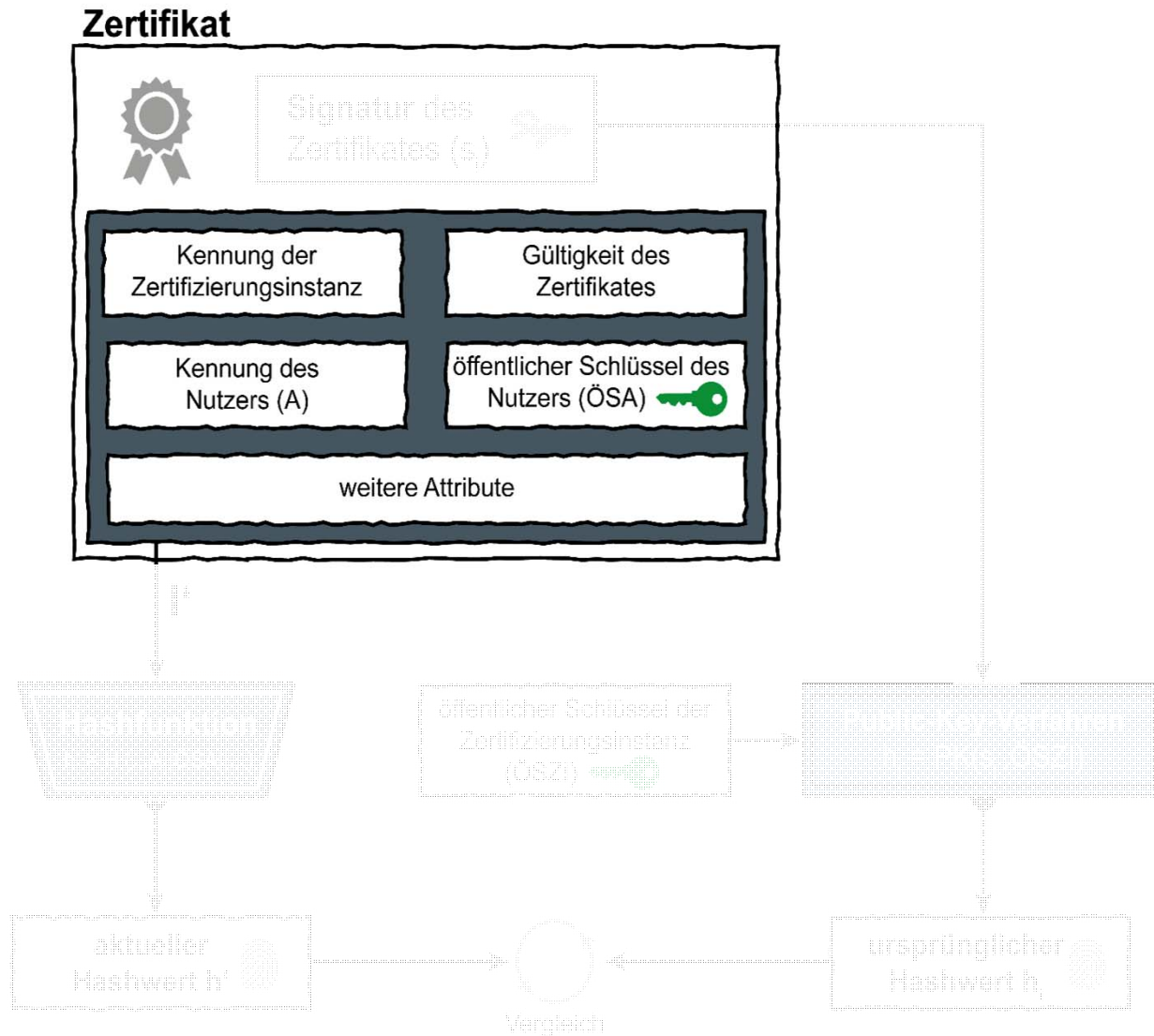
## → Verifikation eines digitalen Zertifikates





# Digitale Signaturen und Zertifikate

## → Verifikation eines digitalen Zertifikates



# Signatur, Zertifikate und PKI

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Digitale Signaturen und Zertifikate
- **Public-Key-Infrastrukturen**
- Gesetzlicher Hintergrund
- PKI-enabled Application
- Zusammenfassung



# Public-Key-Infrastrukturen

## → Idee und Definition (1/2)

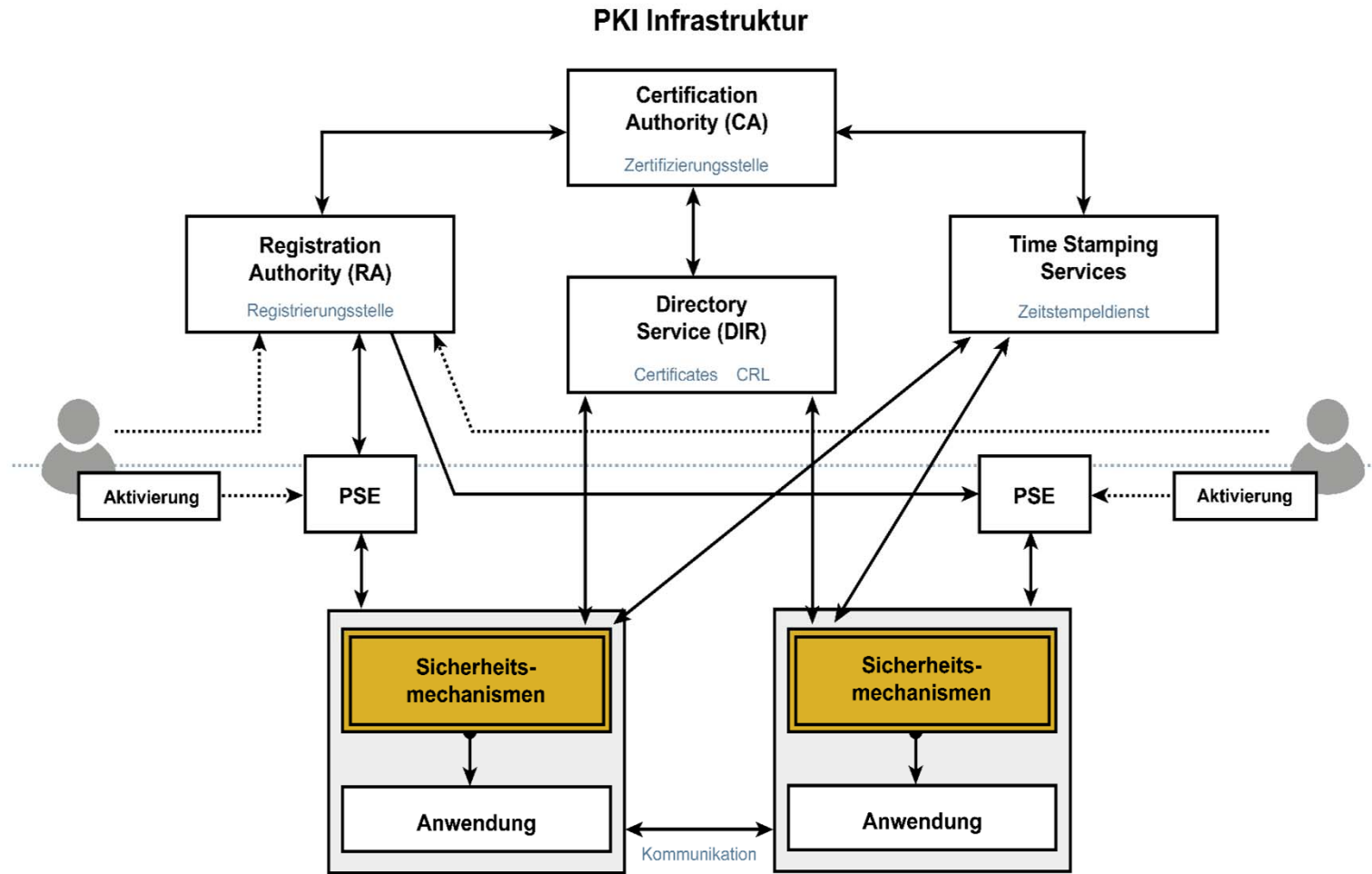
- **Verwalten** von Zertifikaten mit öffentlichen Schlüsseln und weiteren Attributen über deren gesamten **Lebenszyklus**:
  - Erstellung, Aufbewahrung, Verwendung, Löschung.
  - Wichtig: Verifizierung der ursprünglichen Identität der Inhaber.
- **Analogie**: Standesamt und Einwohnermeldeamt.
- **Bestandteile einer PKI**:
  - Hardware
  - Software
  - Regelwerk

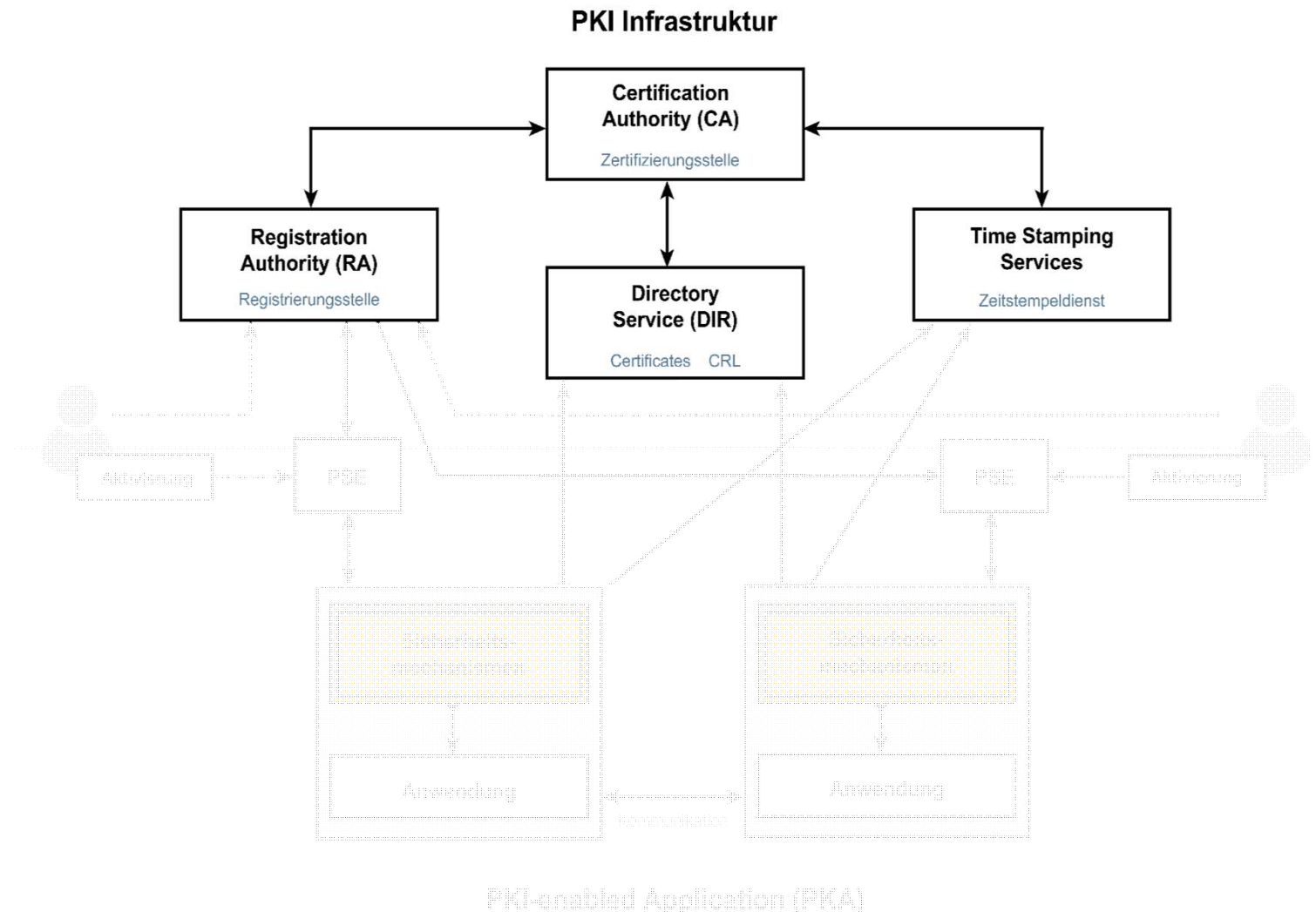


# Public-Key-Infrastrukturen

## → Idee und Definition (2/2)

Cyber-Sicherheitsbedürfnisse		Cyber-Sicherheitsmechanismen
Authentizität	→	Signatur
Integrität	→	Signatur
Verbindlichkeit	→	Signatur
Einmaligkeit	→	TimeStamp
Vertraulichkeit	→	Verschlüsselung







# Public-Key-Infrastrukturen

## → Aufbau und Funktionsweise (2/4)

---

- **Registration Authority (RA):**
  - Schnittstelle zwischen dem PKI-Nutzer und der Certification Authority (CA).
  - Private oder öffentliche Einrichtung (z.B. Berufsverbände, Unternehmen, Behörden und öffentliche Dienstleister).
  - Anträge auf Zertifizierung erfassen.
  - Identität der Antragsteller gemäß des Regelwerks prüfen.
- **Certification Authority (CA):**
  - Vergabe von eindeutigen digitalen Identitäten.
  - Erzeugung von Zertifikaten.
  - Verwaltung von Schlüsselpaaren pro Nutzer.





# Public-Key-Infrastrukturen

## → Aufbau und Funktionsweise (3/4)

---

- **Directory Service (DIR):**
  - Verzeichnisdienst zur Verwaltung der Zertifikate.
  - Öffentlich zugreifbar.
- **Certificate Revocation List (CRL):**
  - Sperrliste für zurückgezogene oder kompromittierte Schlüssel/Zertifikate.
  - Vor jeder Verifikation sollte ein Abgleich mit der Sperrliste erfolgen.
- **Time Stamping Service:**
  - Erstellung von gesicherten Zeitsignaturen gemäß des Regelwerks.



## → Aufbau und Funktionsweise (4/4)

---

- **Personal Security Environment (PSE):**
  - Sammlung aller sicherheitsrelevanten Daten eines Teilnehmers.
    - Geheimer Schlüssel des Nutzers,
    - öffentlicher Schlüssel der Zertifizierungsinstanz,
    - ggf. Zertifikate seiner Kommunikationspartner.
  - Mögliche Formen: Software, Smartcards, USB-Token, allgemeine Sicherheits-Module, SIM-Karte im Smartphones, TPM, usw.
- **PKI-enabled Application (PKA):**
  - Anwendungen, die auf Basis der Sicherheitsmechanismen einer PKI umgesetzt werden (z.B. Dokumentenverschlüsselung, Zahlungssysteme, IPSec-Kommunikation, ...).



# Public-Key-Infrastrukturen

## → Wirksamkeit (1/2)

- Sichere und vertrauenswürdig umgesetzte PKIs haben eine hohe Wirksamkeit bezüglich der Cybersicherheit.
- Unsicher gespeicherte Schlüssel sind ein Sicherheitsrisiko.
- Aus diesem Grund werden in der Regel Smartcards oder USB-Sicherheitstoken verwendet.
- Restrisiko durch Malware kann mit externen Lesegeräten und Tastaturen minimiert werden.



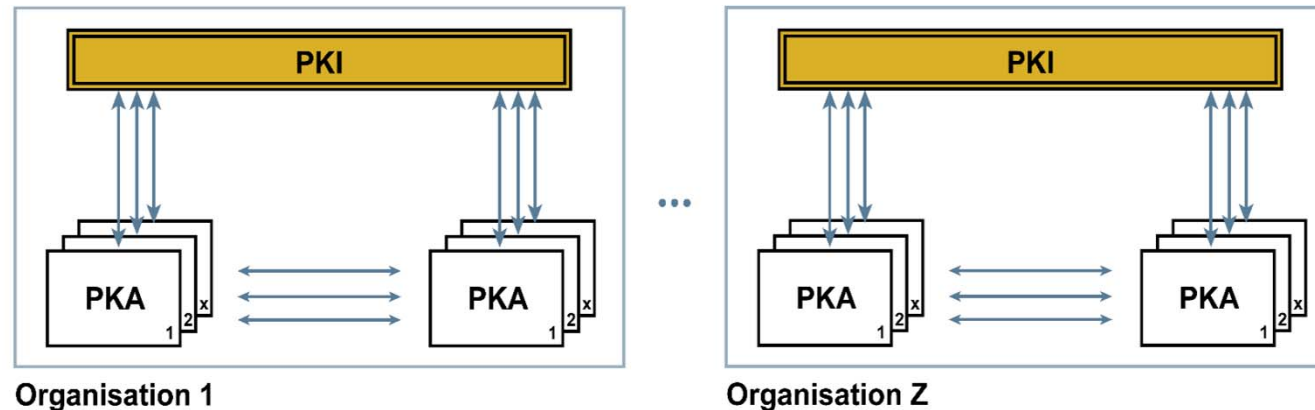
## → Wirksamkeit (2/2)

---

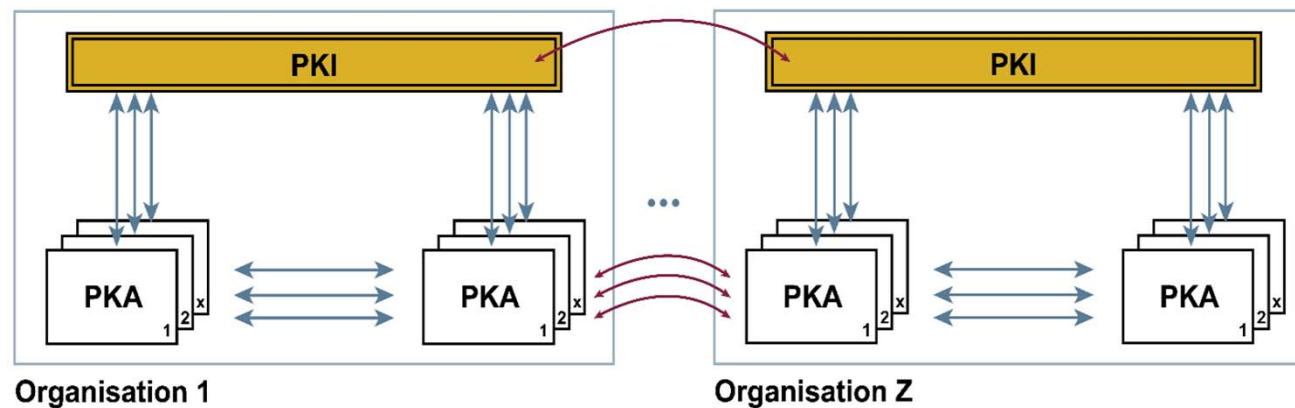
- Es können drei Kategorien an Lesegeräten unterschieden werden, die einen unterschiedlichen Level an Sicherheit und damit an Wirkung gegen Angriffe zur Verfügung stellen:
- **1. Basisleser:**
  - Anzeige und Eingabe über das selbe IT-System des Nutzers.
  - Hohes Restrisiko durch Malware.
- **2. Standardleser**
  - Anzeige und Eingabe über externes IT-System.
- **3. Komfortleser**
  - Zertifizierter Standardleser.



- Geschlossene und dezentrale PKI-Systeme:

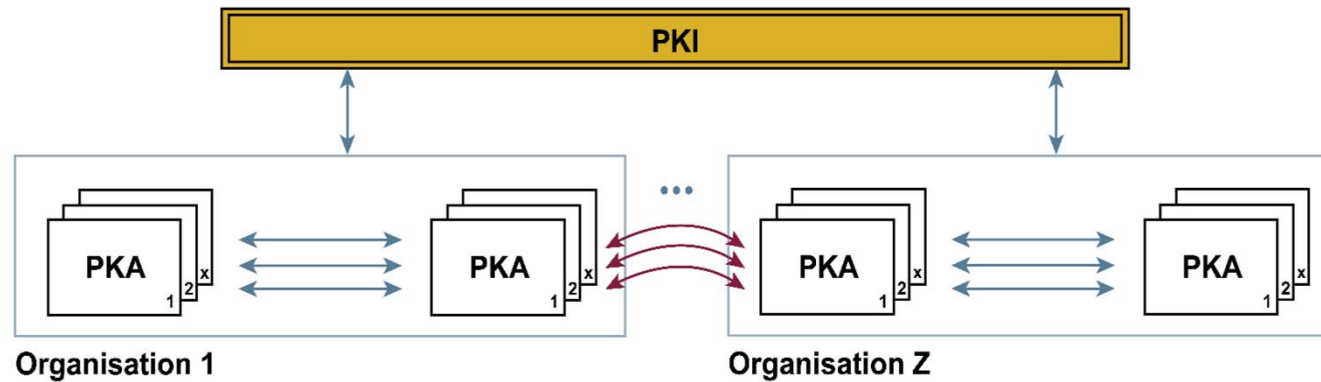


- Offene und dezentrale PKI-Systeme:





- Offene, zentrale PKI-Systeme:





## → Probleme in der Praxis (1/3)

---

- **Probleme bei geschlossenen PKI-Systemen:**
  - PKI-Dienstleistungen können nur innerhalb einer Organisation verwendet und nicht für die Kommunikation nach außen genutzt werden.
  - In der Praxis existieren jedoch viele organisationsübergreifende Prozesse.
- **Probleme bei offenen PKI-Systemen:**
  - Abgleich der verschiedenen organisationsspezifischen Regelwerke nötig.
  - Ziel ist die Schaffung einer gemeinsamen, verbindlichen Vertrauensbasis (Level of Trust).
  - Viele unterschiedliche, teilweise sehr komplexe Standards.



## → Probleme in der Praxis (2/3)

---

- **Unterschiedliche Verantwortung für PKIs und PKAs in Unternehmen:**
  - Verständigung über verschiedene Abteilungen hinweg nötig.
  - Aufwand in großen Unternehmen höher.
- **Henne-Ei-Problem:**
  - PKIs benötigen konsequenten Einsatz der bestehenden Technologien und die Umsetzung der Security Regelwerke.
  - Realität: Organisationen können sich nur schwer auf den Abgleich ihrer individuellen Cyber-Sicherheitskonzepte einigen.
  - Dadurch gestaltet sich der Aufbau eines gemeinsamen „Level of Trust“ langwierig, und längst fällige Entscheidungen werden nicht getroffen.





## → Probleme in der Praxis (3/3)

---

- **Hoher personeller und organisatorischer Aufwand:**
  - Sensibilisierung der Nutzer für die Cyber-Sicherheit.
  - Schulung der Nutzer für die Produkte und die Planung.
  - Durchführung des Roll-Out.
- **Key-Recovery bei der Verschlüsselung:**
  - Verfahren zur Entschlüsselung benötigt, falls:
    - Technischen Defekte auftreten,
    - PSE verloren gehen,
    - Mitarbeiter aus dem Unternehmen ausscheiden.



- Die folgenden vier Kernsätze können als Grundlage für die erfolgreiche Realisierung eines PKI-Systems gelten:
  - Verschiedene Anwendungen haben unterschiedliche Cyber-Sicherheitsbedürfnisse.
  - Unterschiedliche Cyber-Sicherheitsbedürfnisse lassen sich isoliert einfacher verwirklichen.
  - Isolierte Lösungen haben einen klaren Fokus.
  - Ein klarer Fokus verringert die auftretenden Probleme und ermöglicht eine schnellere, einfachere und kostengünstigere Umsetzung.



## → Umsetzungskonzepte (2/4)

---

- **Umsetzungskonzept „TLS/SSL“:**
  - Vertrauliche Kommunikation zwischen Client und Server.
  - TLS/SSL Infrastruktur ist gegeben:
    - Web Server,
    - Clients,
    - mehrere hundert PKIs,
    - Open Source Bibliotheken,
    - TLS/SSL-Accelerator-Lösungen.
  - 2018 wurden schon mehr als 80 % der IP-Pakete mit TLS/SSL im Internet verschlüsselt.



- **Umsetzungskonzept „E-Mail-Sicherheit“:**
  - Zu schützende Unternehmensdaten sollen personenorientiert und vertraulich ausgetauscht werden.
  - Das gegenseitige Wissen um die Identität der Kommunikationspartner ist von zentraler Bedeutung.
  - Insbesondere wenn via E-Mail Prozesse mit nachfolgenden Kosten (Bestellungen, Wareneinkauf, ...) ausgelöst werden, liegt die Verbindlichkeit im Interesse der Unternehmen.
  - Mailprogramme sind den Nutzern bekannt und vertraut.
  - Sicherheitsrelevante Funktionen müssen so eingepasst werden, dass der Nutzer sich nur einem Mindestmaß an neuen Funktionalitäten gegenüberübersieht und jederzeit Klarheit über die nötigen Arbeitsschritte und ihre Folgen hat.



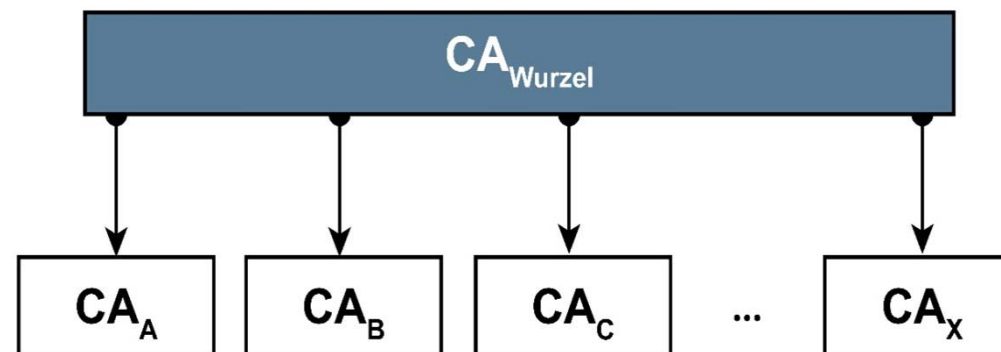
## → Umsetzungskonzepte (4/4)

---

- **Umsetzungskonzept „Verbindlicher Austausch von Transaktionsdaten“:**
  - Transaktionsdaten sind 'besondere' Kommunikationsdaten, da sich aus ihnen in der Regel kostenrelevante Aktionen ableiten.
  - Für den Empfänger wie für den Sender steht die Verbindlichkeit im Mittelpunkt.
  - Diese Anwendungen sind meist firmen- bzw. gerätebezogen und basieren auf geschlossenen Systemen (z.B. der Austausch von Rechnungsdaten zwischen Telekommunikationsanbietern und ihren Partnerunternehmen).
  - Die Bandbreite reicht von kleinen Datenmengen pro Monat bis hin zu einer hohen Anzahl von Transaktionen pro Sekunde.
  - Der Fokus liegt auf der möglichst nahtlosen Integration in bestehende Workflows.

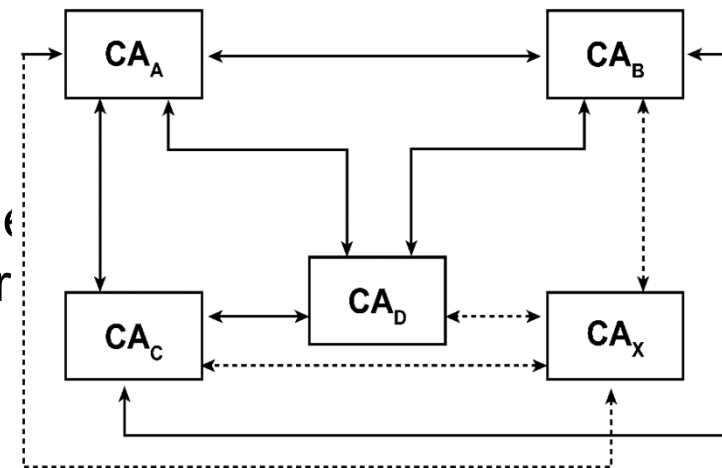


- **Übergeordnete CA (Wurzel-CA, Root CA):**
  - Wurzel-CA generiert Zertifikate der öffentlichen Schlüssel der untergeordneten CAs.
  - Der öffentliche Schlüssel der Wurzel-CA ist im PSE untergebracht oder wird als Zertifikat zum Abrufen angeboten.
  - In den meisten Fällen akzeptieren Unternehmen, Organisationen oder Länder keine derartige Unterordnung.
  - Nur in großen, geschlossenen PKI-Systemen etabliert.



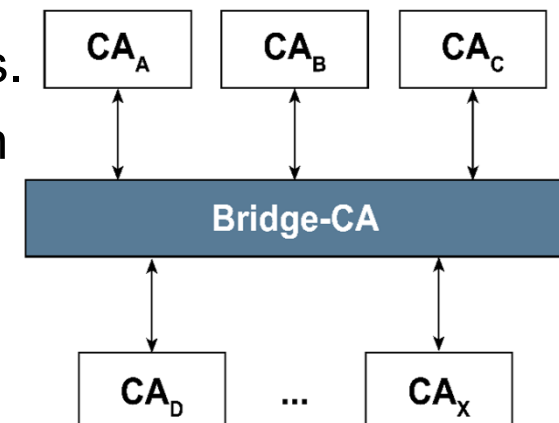
- **n:n-Cross-Zertifizierung:**

- Jede CA tauscht ihre öffentlichen Schlüssel selbstständig mit jeder anderen CA aus.
- Authentischer Austausch der öffentlichen Schlüssel aufwendig.
- Multiple Vertragsverhandlungen nötig.
- Abweichende Verträge und Vereinbarungen zwischen den beteiligten Betreibern möglich.
- Nur bei kleinen Gruppenunabhängige PKI-Betreiber etabliert, und auch dort nur in abgegrenzten Geschäftsprozessen.





- **1:n Cross-Zertifizierung (Bridge CA):**
  - Geringer Verwaltungsaufwand, da es für jede CA nur einen Vertragspartner gibt.
  - Entscheidungsfreiheit über die passende Vertrauenskette.
  - Bridge CA fungiert als zentrale Vermittlungsinstanz zwischen den beteiligten Organisationen → Geeignete Policy benötigt.
  - CAs übergeben authentisch ihre öffentlichen Schlüssel an die Bridge CA.
  - Bridge CA signiert eine Tabelle der öffentlichen Schlüssel aller beteiligten CAs.
  - Die eigene CA stellt dann all ihren Nutzern den öffentlichen Schlüssel der Bridge CA als Zertifikat zur Verfügung.







# Signatur, Zertifikate und PKI

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Digitale Signaturen und Zertifikate
- Public-Key-Infrastrukturen
- **Gesetzlicher Hintergrund**
- PKI-enabled Application
- Zusammenfassung



# Gesetzlicher Hintergrund

- eIDAS (EU-Verordnung)
- **e**lectronic **I**dentification, **A**uthentication and **T**rust **S**ervice



## Gesetzlicher Hintergrund → eIDAS (1/7)

## electronic Identification, Authentication and Trust Service

- EU-Verordnung 910/2014 über elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt (eIDAS).
  - Gleichstellung, Interoperabilität, gegenseitige Anerkennung der **Vertrauensdienste** der Mitgliedsstaaten.
- Für alle in der EU niedergelassenen Vertrauensdiensteanbieter (VDA).
  - Ausgenommen: Vertrauensdienste innerhalb geschlossener Nutzergruppen, wie z.B. interne Unternehmenslösungen.
- In Deutschland umgesetzt durch:
  - Signaturgesetz (SigG)
  - Signaturverordnung (SigV)

# Gesetzlicher Hintergrund

## → eIDAS (2/7)

- Das Vertrauen durch eIDAS ist untrennbar mit der Rechtssicherheit verbunden.
  - Ein elektronisches Dokument soll in der EU den gleichen Stellenwert haben wie ein analoges.
- Unterteilung in qualifizierte und nicht-qualifizierte VDA.
  - Freiwillige Akkreditierung alle drei Jahre möglich (Konformitätsbewertung).
- Art. 13: Zusätzliches Vertrauen durch freiwilliges EU-Vertrauenssiegel (Analogie: „IT Security made in Germany“ Qualitätssiegel der TeleTrustT).

# Gesetzlicher Hintergrund

## → eIDAS (3/7)

- Art. 28: Suspendierung von qualifizierten Zertifikaten möglich.
- Art. 35-40: Elektronische Siegel als Pendant zu elektronischen Signaturen.
  - Signaturen → natürliche Personen
  - Siegel → Organisationen
- **Elektronische Fernsignaturen:**
  - Sichere Signaturerstellungseinheit (SSEE) befindet sich beim qualifizierten VDA.
  - Auslöser der Signatur wird mit technischen Mitteln verlängert.
  - VDA muss geeignete Cybersicherheitsmechanismen realisieren.

# Gesetzlicher Hintergrund

## → eIDAS (4/7)

- Art. 11, 13: Haftung und Beweislast:
  - Qualifizierte VDA sind in der Nachweispflicht.
  - Bei nicht-qualifizierten VDA liegt die Nachweispflicht hingegen beim Kunden.
  - Der VDA haftet, wenn er die in der eIDAS-Verordnung genannten Pflichten nicht eingehalten hat (z.B. wenn die Dienste nicht dem neusten Stand der Technik entsprechen).
  - VDA kann Haftung im Vorfeld beschränken.

# Gesetzlicher Hintergrund

## → eIDAS (5/7)

- Art. 43-44: Elektronisches Einschreiben - Anforderungen:
  - Identifizierung des Absenders mit hohem Maß an Vertrauenswürdigkeit.
  - Identifizierung des Empfängers vor Zustellung der Daten.
  - Absenden und Empfang ist durch fortgeschrittene elektronische Signatur oder ein fortgeschrittenes elektronisches Siegel eines qualifizierten VDA vor Veränderung geschützt.
  - Jede Veränderung von Daten wird deutlich angezeigt.
  - Zeit und Datum von Versand, Empfang oder Änderung der Daten wird durch qualifizierte elektronische Zeitstempel angezeigt.

# Gesetzlicher Hintergrund

## → eIDAS (6/7)

- De-Mail:
  - Es versieht immer der akkreditierte Anbieter selbst die Nachrichten mit einer qualifizierten Signatur (Fernsignatur).
  - Überall, wo in eIDAS qualifizierte Zeitstempel vorgesehen sind, benutzt De-Mail Prüfsummen und qualifizierte Signaturen.
  - De-Mail schreibt zwingend eine Transportverschlüsselung zwischen den Anbietern vor.
  - De-Mail überlässt es den Anbietern, eine sichere Dokumentenablage anzubieten.
  - De-Mail ist aktuell also nicht vollständig eIDAS-konform.





# Gesetzlicher Hintergrund

## → eIDAS (7/7)

- Art. 19: Sicherheitsanforderungen an Vertrauensdiensteanbieter:
  - Alle qualifizierten und nicht-qualifizierten VDA müssen Cybersicherheitsmechanismen gemäß dem Stand der Technik implementieren.
  - Sicherheitsvorfall müssen innerhalb von 24 Stunden an die zuständige nationale Stelle bzw. Datenschutzbehörde sowie die betroffenen natürlichen oder juristischen Personen gemeldet werden.
  - Betreffen Cybersicherheitsvorfälle mehrere Mitgliedsstaaten, so müssen die Aufsichtsstellen der betroffenen Mitgliedsstaaten und die ENISA davon in Kenntnis gesetzt werden.
  - Aufsichtsstelle entscheidet über Veröffentlichung des Vorfalls.



# Signatur, Zertifikate und PKI

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Digitale Signaturen und Zertifikate
- Public-Key-Infrastrukturen
- Gesetzlicher Hintergrund
- **PKI-enabled Application**
- Zusammenfassung



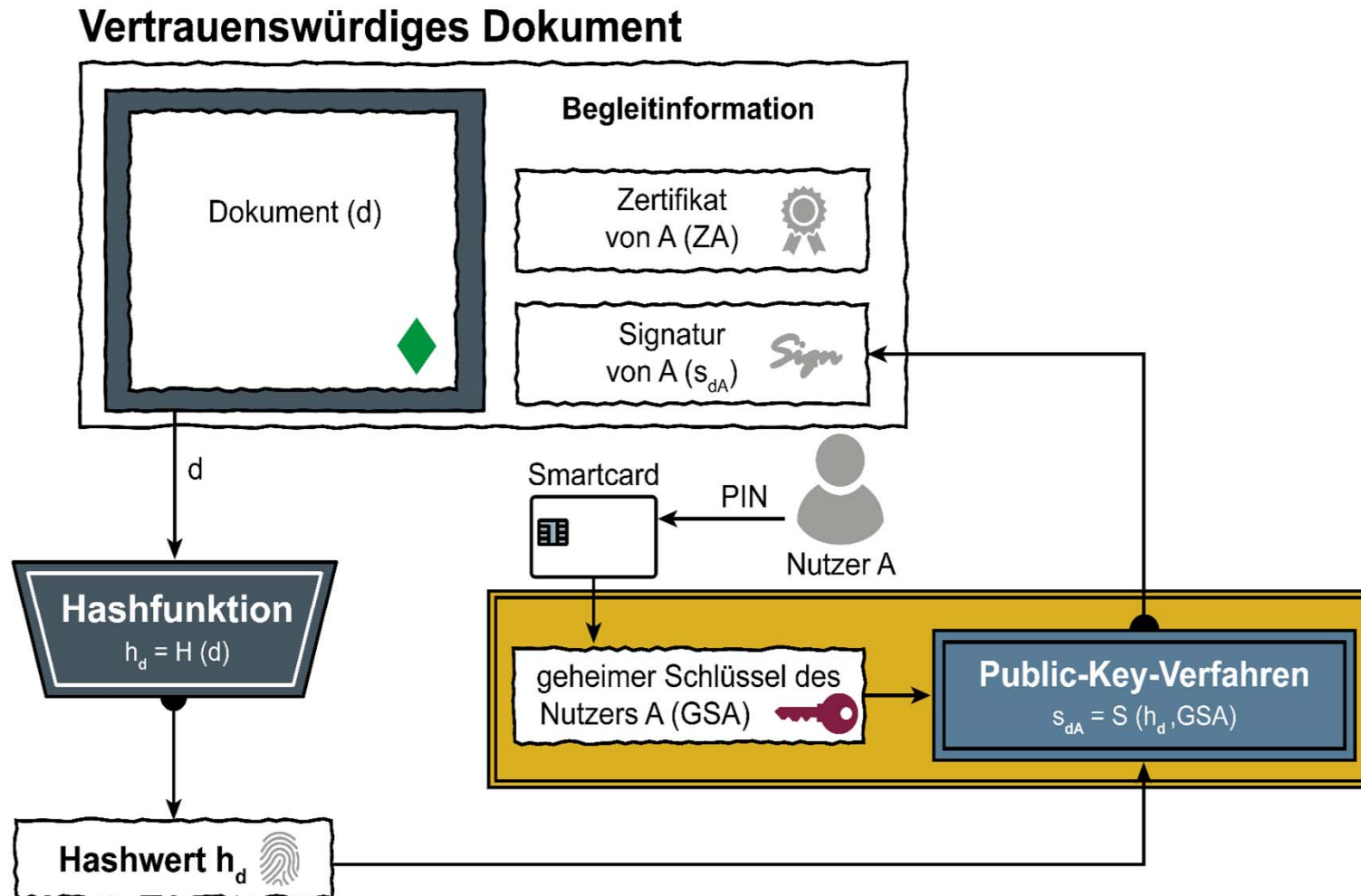
# PKI-enabled Application

## → E-Mail Sicherheit (1/8)

- Eine Information wurde früher entweder auf einer Schreibmaschine getippt oder mithilfe eines Textverarbeitungssystems in ein IT-System eingegeben und anschließend ausgedruckt.
- Der Ausdruck wurde unterschrieben, in einen Briefumschlag gesteckt und vertraulich an den gewünschten Empfänger gesendet.
- Der Empfänger erkannte an der Unversehrtheit des Umschlags, dass die Information vertraulich übermittelt worden war.
- Nach dem Öffnen des Briefes konnte der Empfänger an der eigenhändigen Unterschrift die Echtheit des Absenders oder des Autors überprüfen.
- Die eigenhändige Unterschrift ist zudem eine rechtsgültige Unterschrift.
- E-Mail-Sicherheit bedeutet, die gleiche Sicherheit bei Mails zu haben, die auch für Briefe gilt.

# PKI-enabled Application

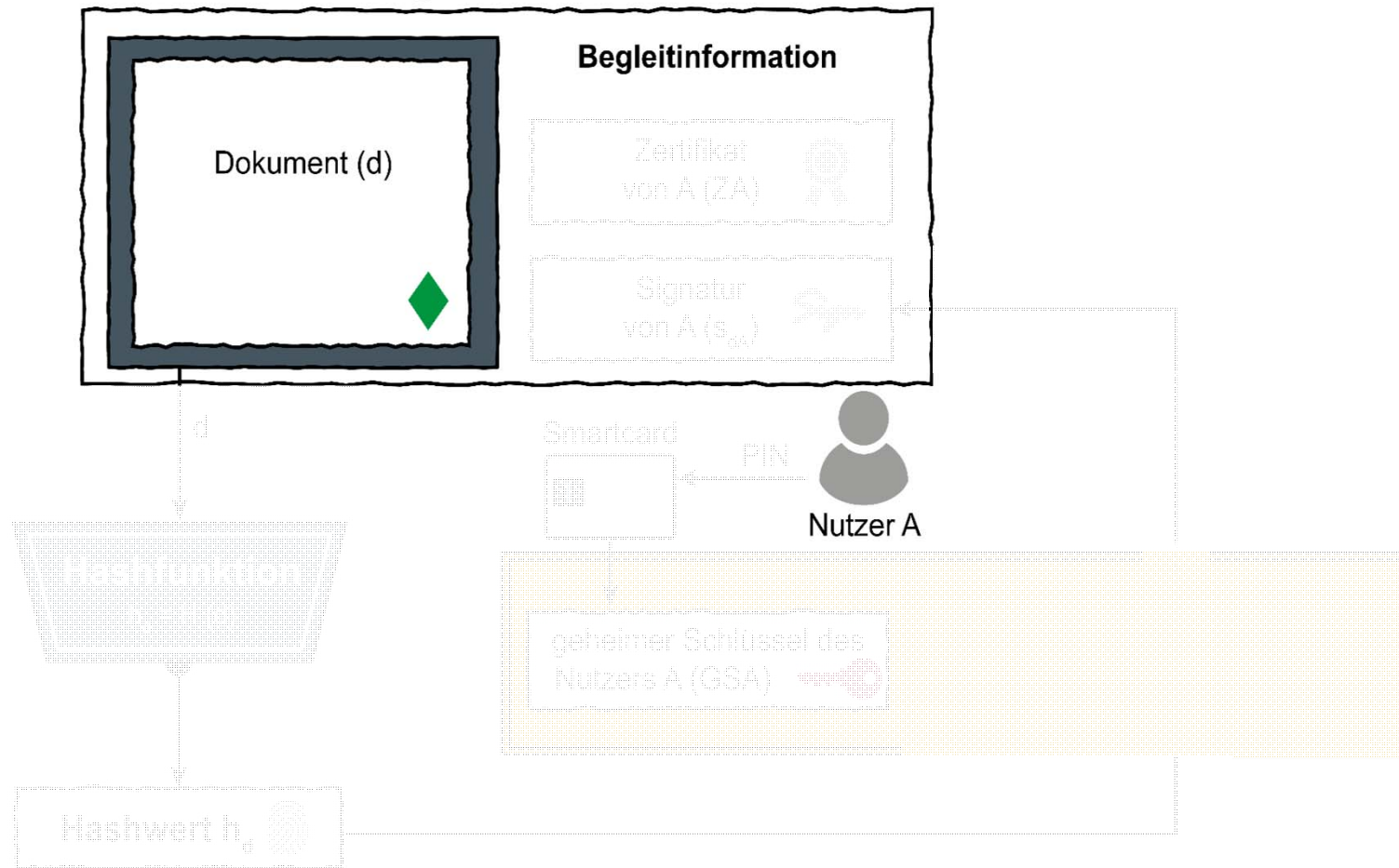
## → E-Mail Sicherheit (2/8)



# PKI-enabled Application

## → E-Mail Sicherheit (2/8)

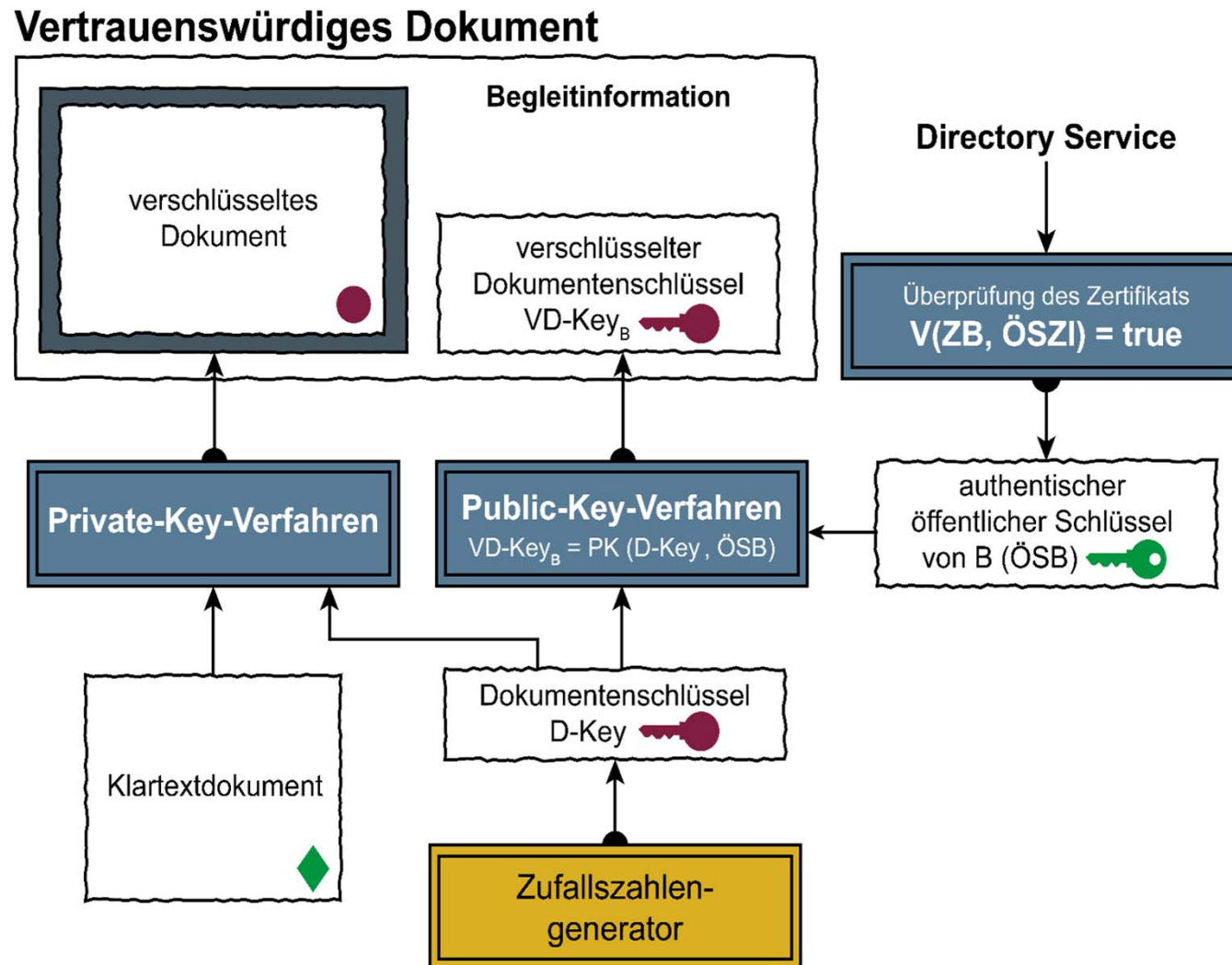
### Vertrauenswürdiges Dokument



# PKI-enabled Application

## → E-Mail Sicherheit (3/8)

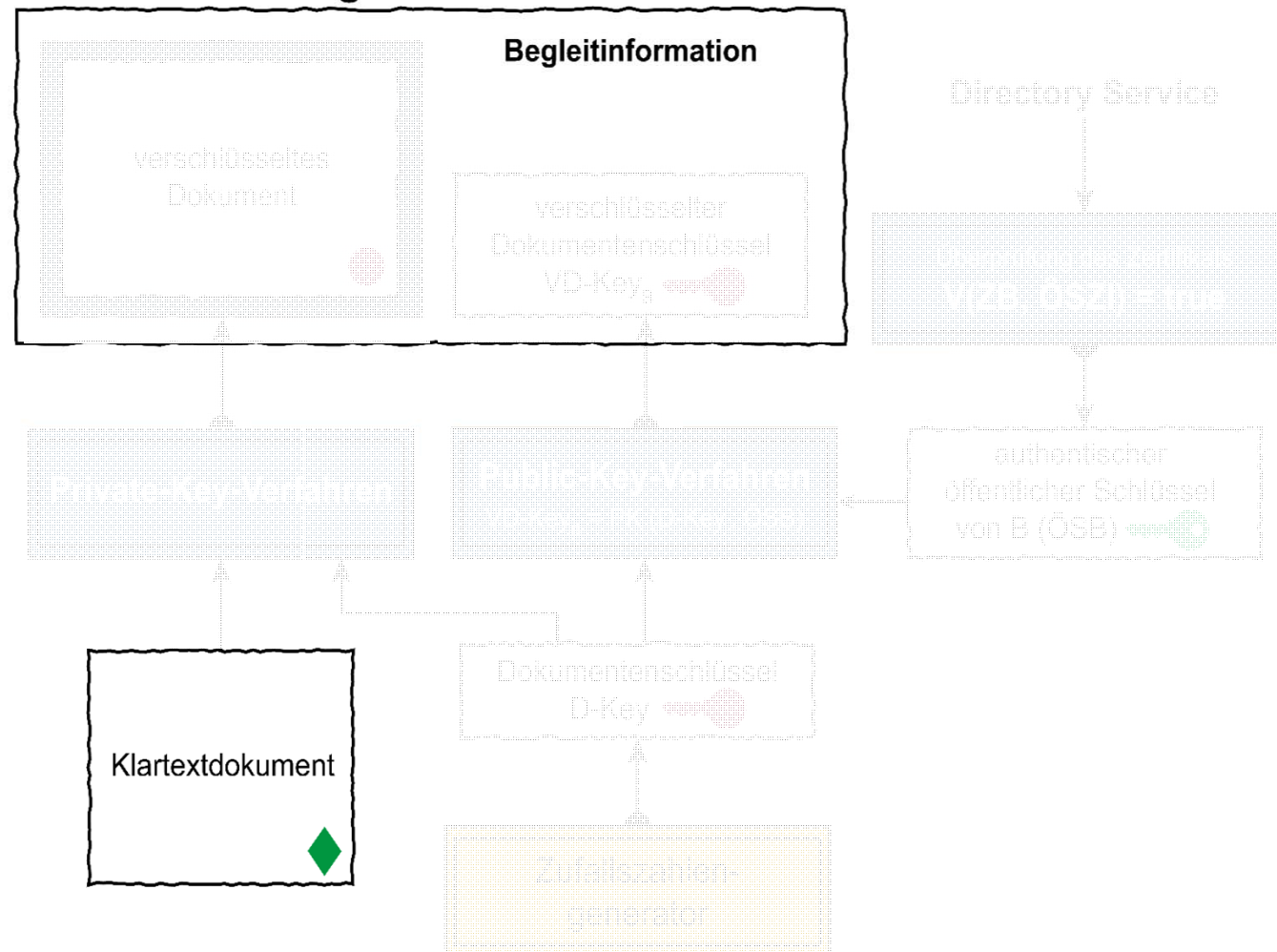
- Digitaler Zeitstempel:
  - Die Beweiskraft eines elektronischen Dokuments hängt häufig zusätzlich auch vom Zeitpunkt seiner Erstellung ab.
  - Da die Uhren und Zeitfunktionen sich in handelsüblichen IT-Systemen und Betriebssystemen ohne Probleme verstellen lassen, kann die Urzeit nicht für eine vertrauenswürdige Zeitangabe bei der digitalen Signatur angegeben werden.
  - Ein Zeitstempel benötigt eine digitale Bescheinigung einer Zertifizierungsstelle.
  - Sie bestätigt, dass das Dokument zum angegebenen Zeitpunkt ihr vorgelegen hat, indem sie den Zeitstempel digital signiert.



# PKI-enabled Application

## → E-Mail Sicherheit (4/8)

### Vertrauenswürdiges Dokument



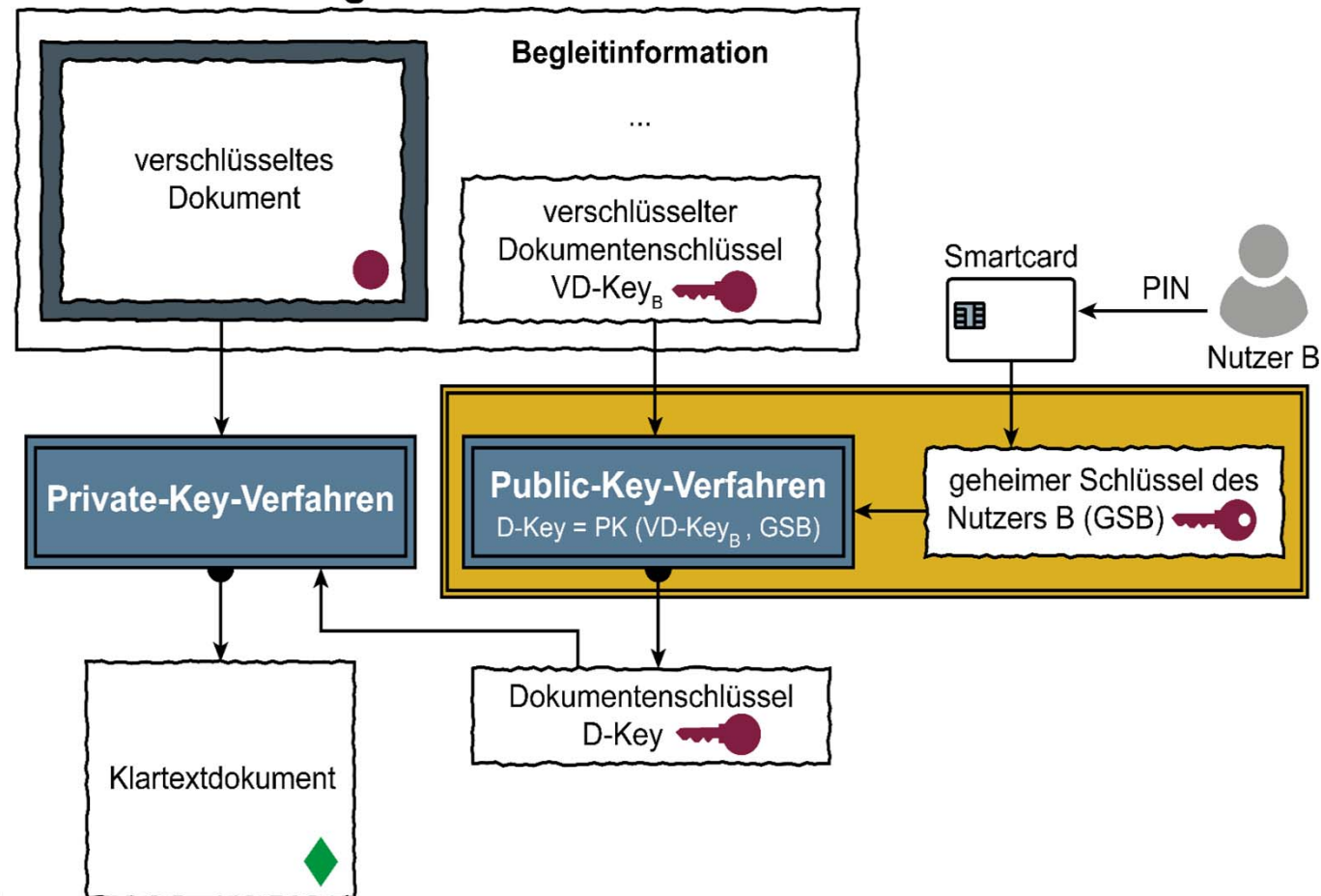




# PKI-enabled Application

## → E-Mail Sicherheit (5/8)

### Vertrauenswürdiges Dokument

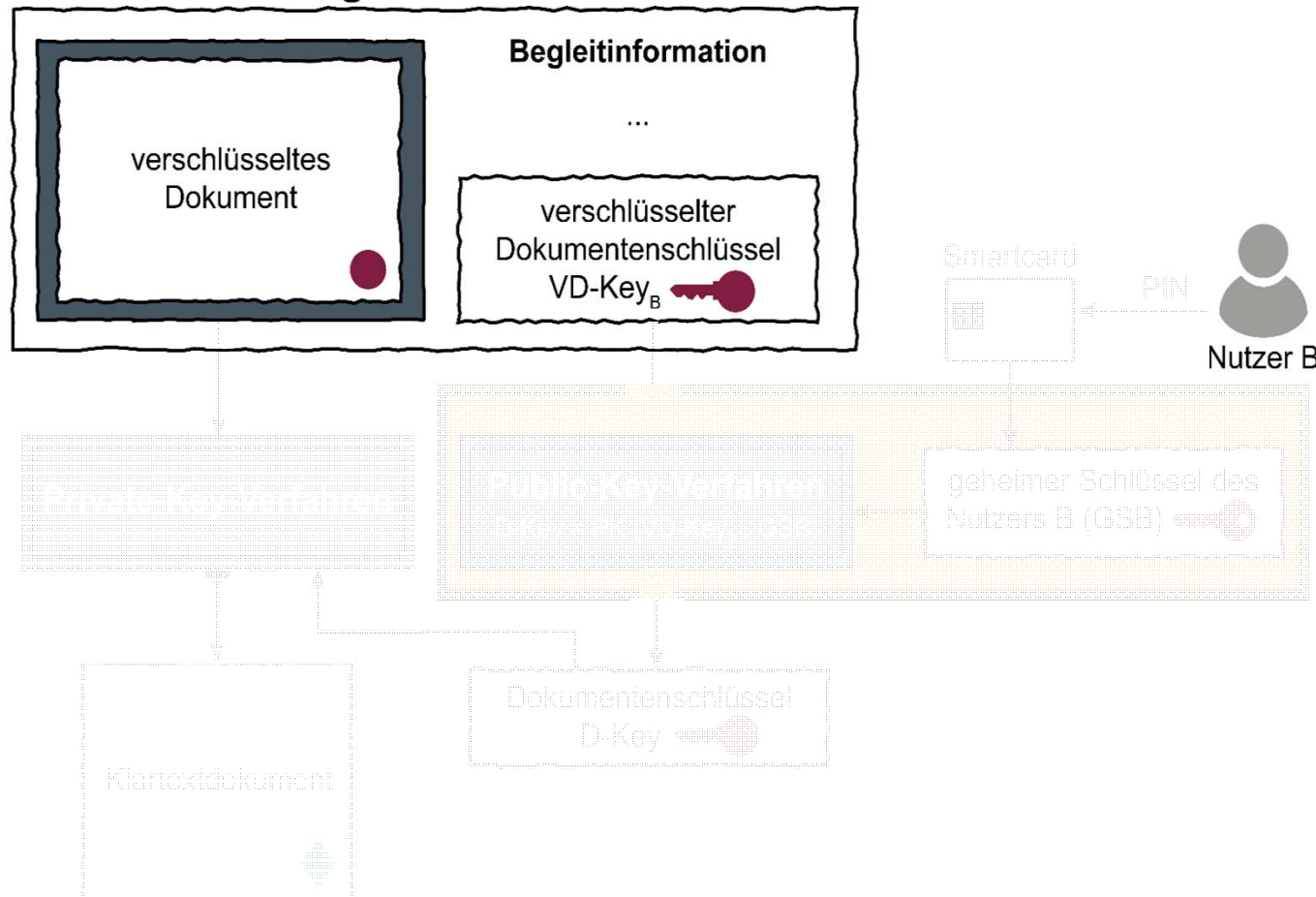




# PKI-enabled Application

## → E-Mail Sicherheit (5/8)

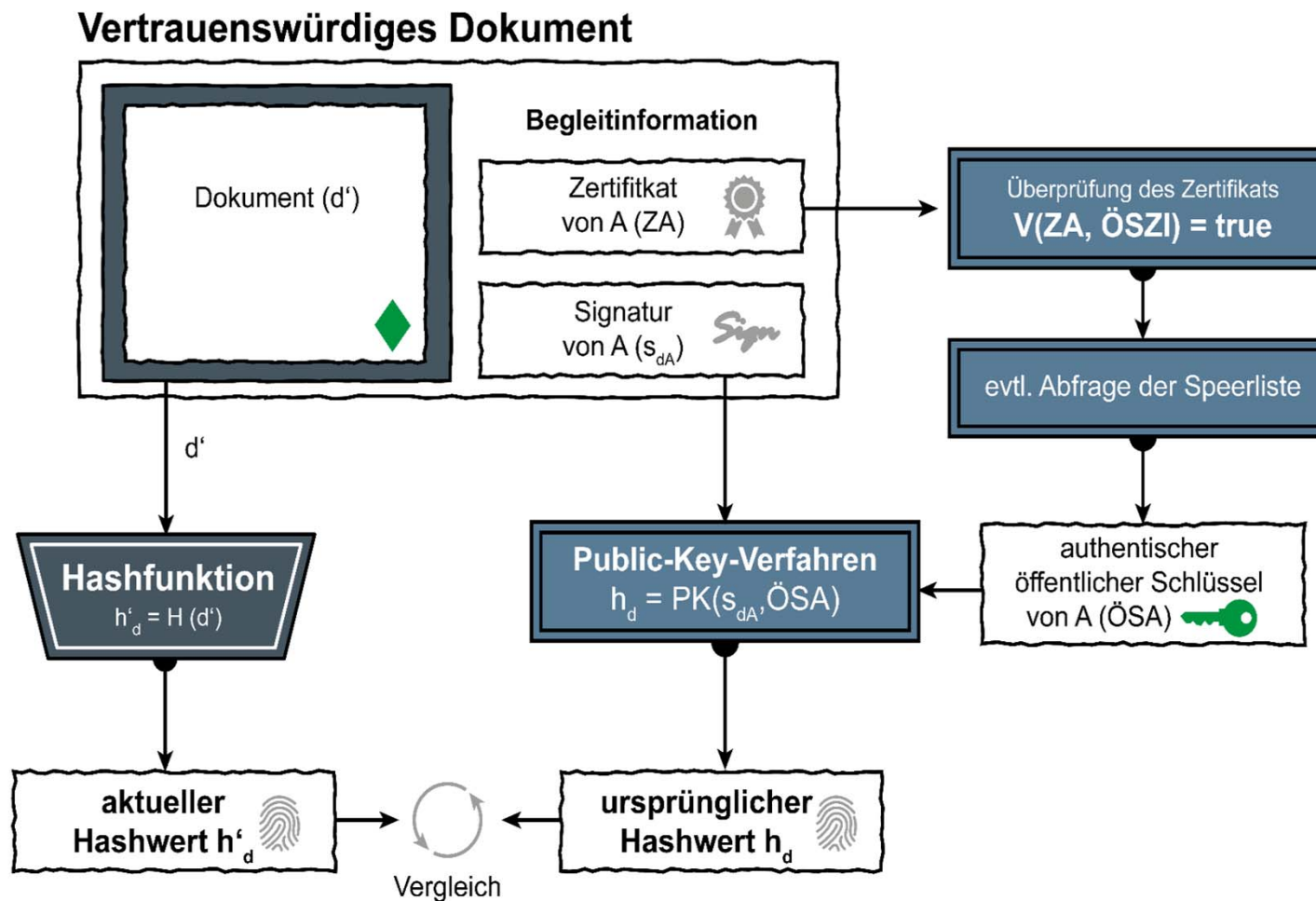
### Vertrauenswürdiges Dokument





# PKI-enabled Application

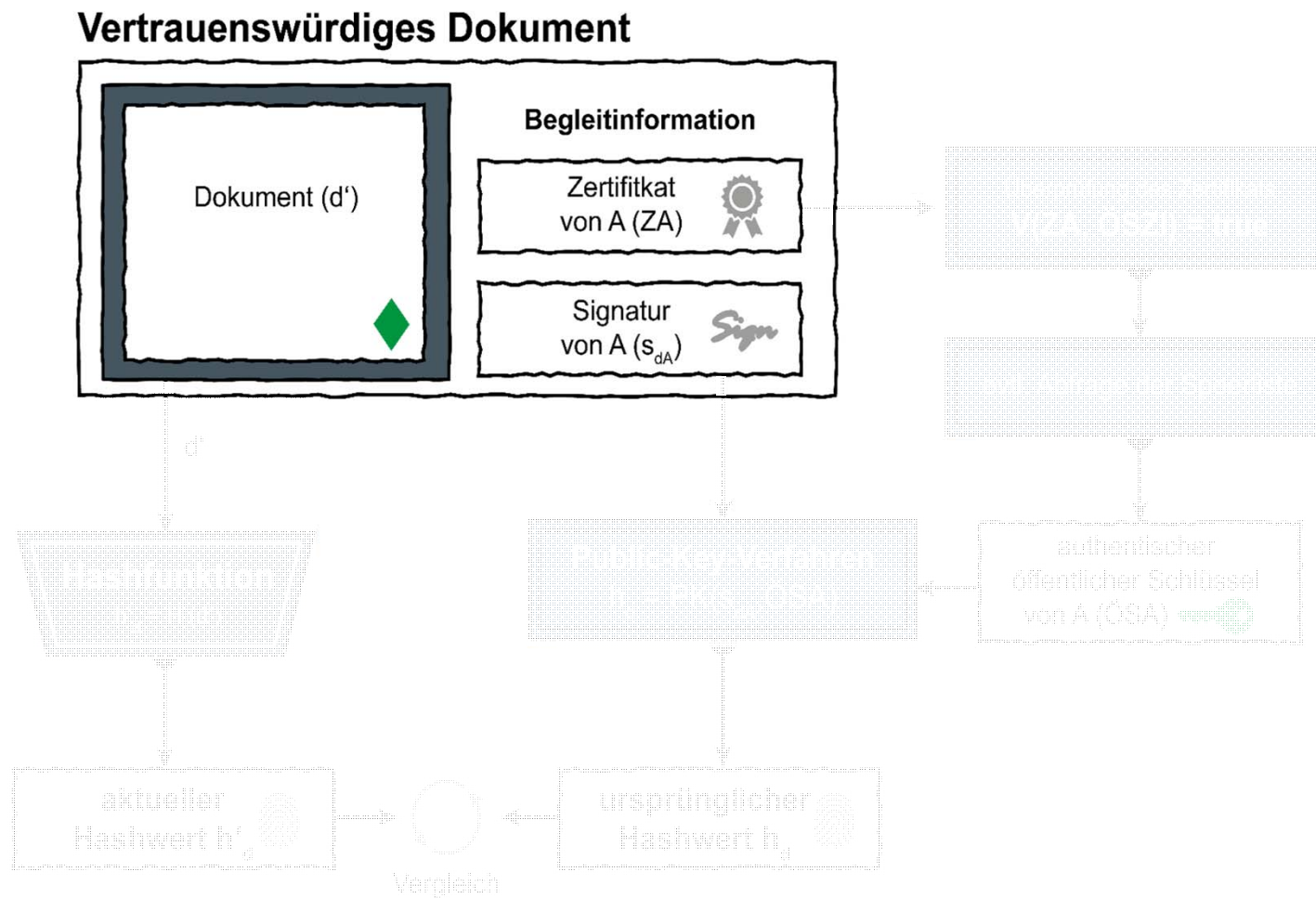
## → E-Mail Sicherheit (6/8)





# PKI-enabled Application

## → E-Mail Sicherheit (6/8)





- Nach erfolgreicher Überprüfung der Signatur ist sichergestellt, dass das Dokument **unversehrt** übertragen worden ist. Das bedeutet:
  - Niemand hat das Dokument manipuliert (Gewährleistung der **Datenunversehrtheit**).
  - Die angegebene Zeit der Signatur wurde nicht geändert.
  - Nur die Nutzer, die in den Begleitinformationen angegeben sind, konnten die entsprechende Signatur durchführen.
- Diese Funktionen machen die digitale Signatur zum elektronischen Äquivalent der eigenhändigen Unterschrift.
- Als zusätzliche Sicherheitsfunktion steht die Verschlüsselung des Dokuments zur Verfügung, die seine Vertraulichkeit garantiert.



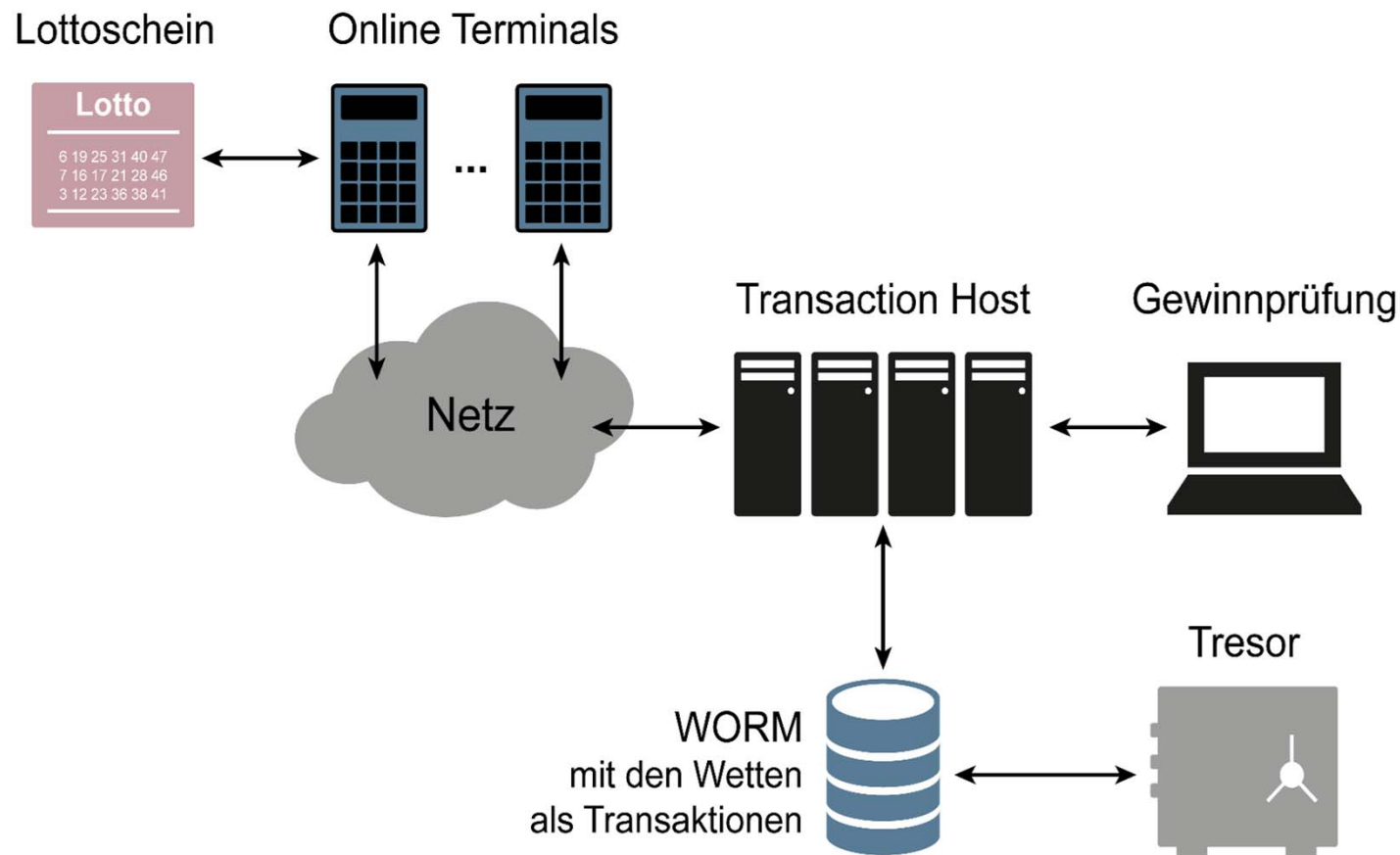
- E-Mail-Sicherheit aus Sicht des Nutzers?
- Die Sicherheitsfunktionen werden in die Anwendungen integriert (z.B. Mailsoftware und Browser).
- Der Nutzer kann dabei durch einfachen Mausklick die Sicherheitsfunktionen aufrufen.
- Je einfacher die Nutzbarkeit der Sicherheitsfunktionen ist, desto höher wird die Akzeptanz der Nutzer sein.
- E-Mail-Gateways können eingehende E-Mails zentral entschlüsseln und intern verteilen, oder ausgehende E-Mails signieren.
  - Empfänger kann sicher sein, dass diese Mails wirklich von der entsprechenden Organisation stammen.



# PKI-enabled Application

## → Lotto-Online-Glückspiel (1/4)

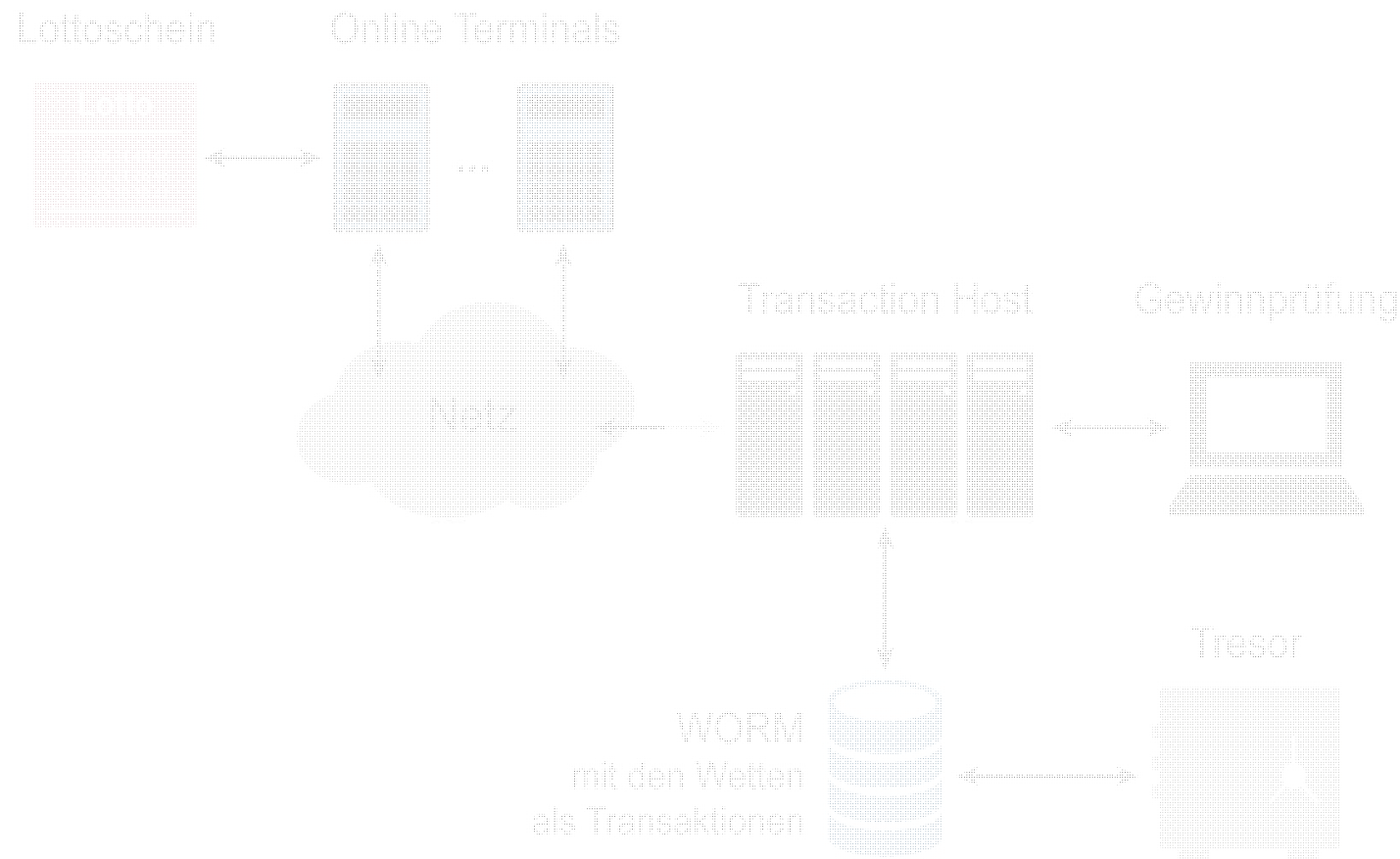
- Altes Verfahren der Manipulationssicherung von Wetten:



# PKI-enabled Application

## → Lotto–Online–Glückspiel (1/4)

- Altes Verfahren der Manipulationssicherung von Wetten:







# PKI-enabled Application

## → Lotto–Online–Glückspiel (2/4)

- Neue Herausforderungen: Gewinnspiele (z.B: Sportwetten) mit sehr viel schnelleren Lebenszyklen.
- Ablösung des WORM durch Nutzung eines Zeitstempeldienstes für Manipulationssicherung von Transaktionsdaten.

$$h = H(\text{Transaction, Datum, Uhrzeit})$$

- Anschließend wird dann dieser Hashwert mit dem geheimen Schlüssel von Lotto ( $GS_{\text{Lotto}}$ ) digital signiert.

$$s = S(h, GS_{\text{Lotto}})$$



# PKI-enabled Application

## → Lotto–Online–Glückspiel (3/4)

- Wenn die Ziehung vorbei ist, kann die Gewinnprüfung umgesetzt und jede Transaktion verifiziert werden.

**$V(H(\text{Transaktion, Datum, Uhrzeit}), s, OS_{\text{Lotto}}) = \text{true?}$**

V: Verifikationsfunktion

H: Hashfunktion

S: Signaturfunktion

s: Signatur der Transaktion

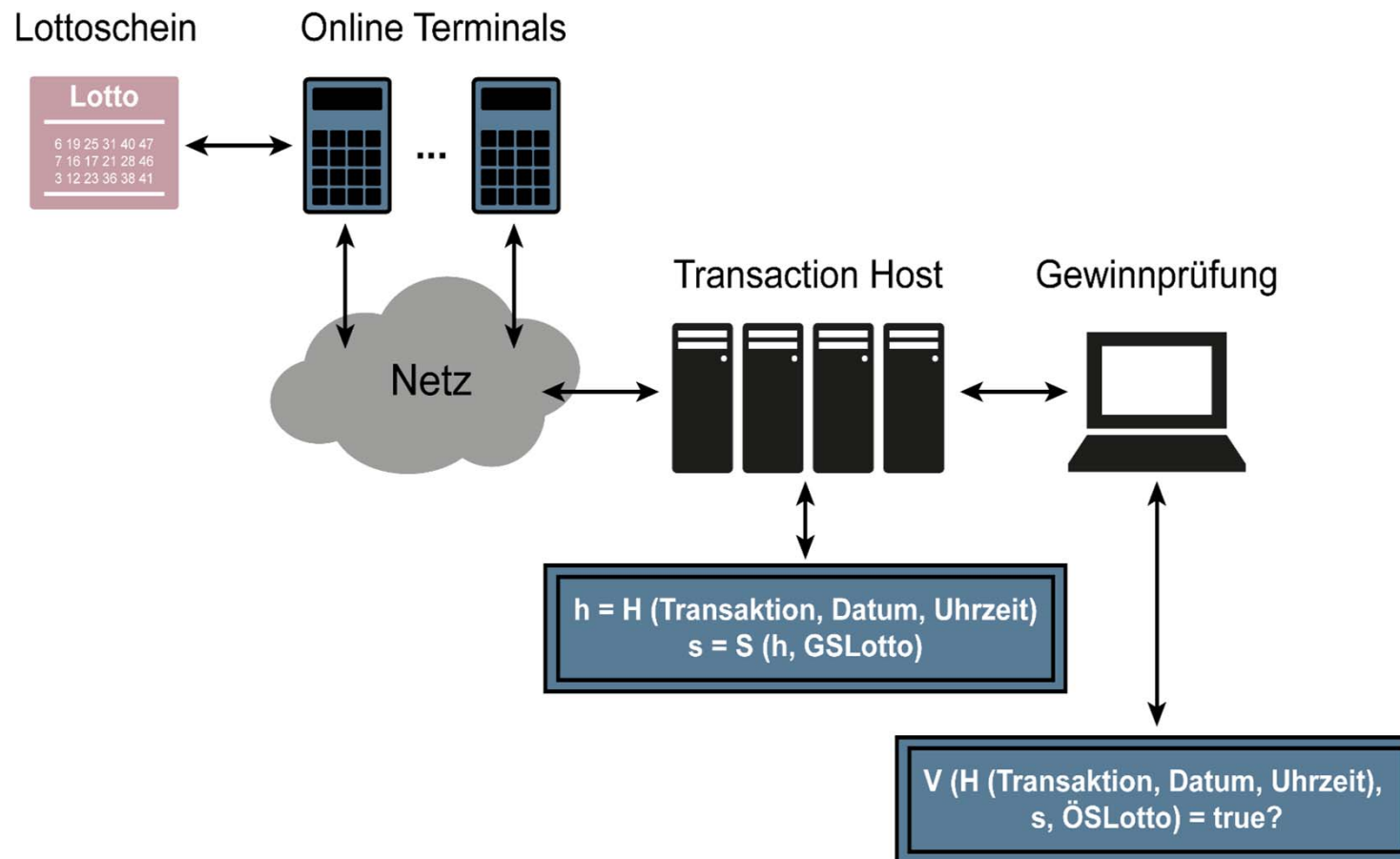
$OS_{\text{Lotto}}$  : Öffentlicher Schlüssel von Lotto



# PKI-enabled Application

## → Lotto-Online-Glückspiel (4/4)

- Neus Verfahren der Manipulationssicherung von Wetten:



## → Lotto-Online-Glückspiel (4/4)

- Neues Verfahren der Manipulationssicherung von Wetten:



# Signatur, Zertifikate und PKI

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Digitale Signaturen und Zertifikate
- Public-Key-Infrastrukturen
- Gesetzlicher Hintergrund
- PKI-enabled Application
- **Zusammenfassung**



# Signatur, Zertifikate und PKI

## → Zusammenfassung

- Digitale Signatur, elektronische Zertifikate und Public-Key-Infrastrukturen sind wichtige Cyber-Sicherheit-Prinzipien, -Mechanismen und -Konzepte für die Realisierung von Cyber-Sicherheitslösungen und -Diensten.
- Public-Key-Infrastrukturen stellen die Basis für organisationsübergreifende Cyber-Sicherheitssysteme dar und haben über eIDAS in Europa eine rechtliche Grundlage.
- Die Beispiele der PKI-enabled Application zeigen auf, was mit den Vertrauensdiensten einer PKI an sicherheitsrelevante Anwendungen umgesetzt werden können.



# **IT-Sicherheit**

## **09 Trusted Computing**

**Gerrit Kalkbrenner**  
**Teile: Norbert Pohlmann**  
**[Gerrit.Kalkbrenner@hwr-berlin.de](mailto:Gerrit.Kalkbrenner@hwr-berlin.de)**



# Trusted Computing

## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- **Kernfunktionalitäten**
- **Sicherheitsarchitektur**
- **Trusted Network Connect**
- **Zusammenfassung**





# Trusted Computing

## → Inhalt

- **Ziele und Ergebnisse der Vorlesung**
- Kernfunktionalitäten
- Sicherheitsarchitektur
- Trusted Network Connect
- Zusammenfassung



# Ziele und Ergebnisse der Vorlesung

## → Trusted Computing

- Gutes Verständnis über die **Sicherheitsarchitektur** und **Sicherheitsprinzipien** von Trusted Computing erlangen.
- Erlangen der Kenntnisse über die TPM Schlüsselhierarchie, Authenticated Boot, Attestation, Binding, Sealing und Trusted Network Connect.
- Gutes Verständnis zu verschiedenen **Kernelarchitekturen** erlangen.
- Gutes Verständnis über praktische Anwendungen durch Betrachtung von **Turaya** als Beispielanwendung.



# Trusted Computing

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- **Kernfunktionalitäten**
- Sicherheitsarchitektur
- Trusted Network Connect
- Zusammenfassung



# Kernfunktionalitäten → Herausforderungen

- Reaktive Cyber-Sicherheitssysteme
  - „Airbag-Methode“
- Proaktive Cyber-Sicherheitssysteme
  - „Vorausschauend“
- Das Software-Problem



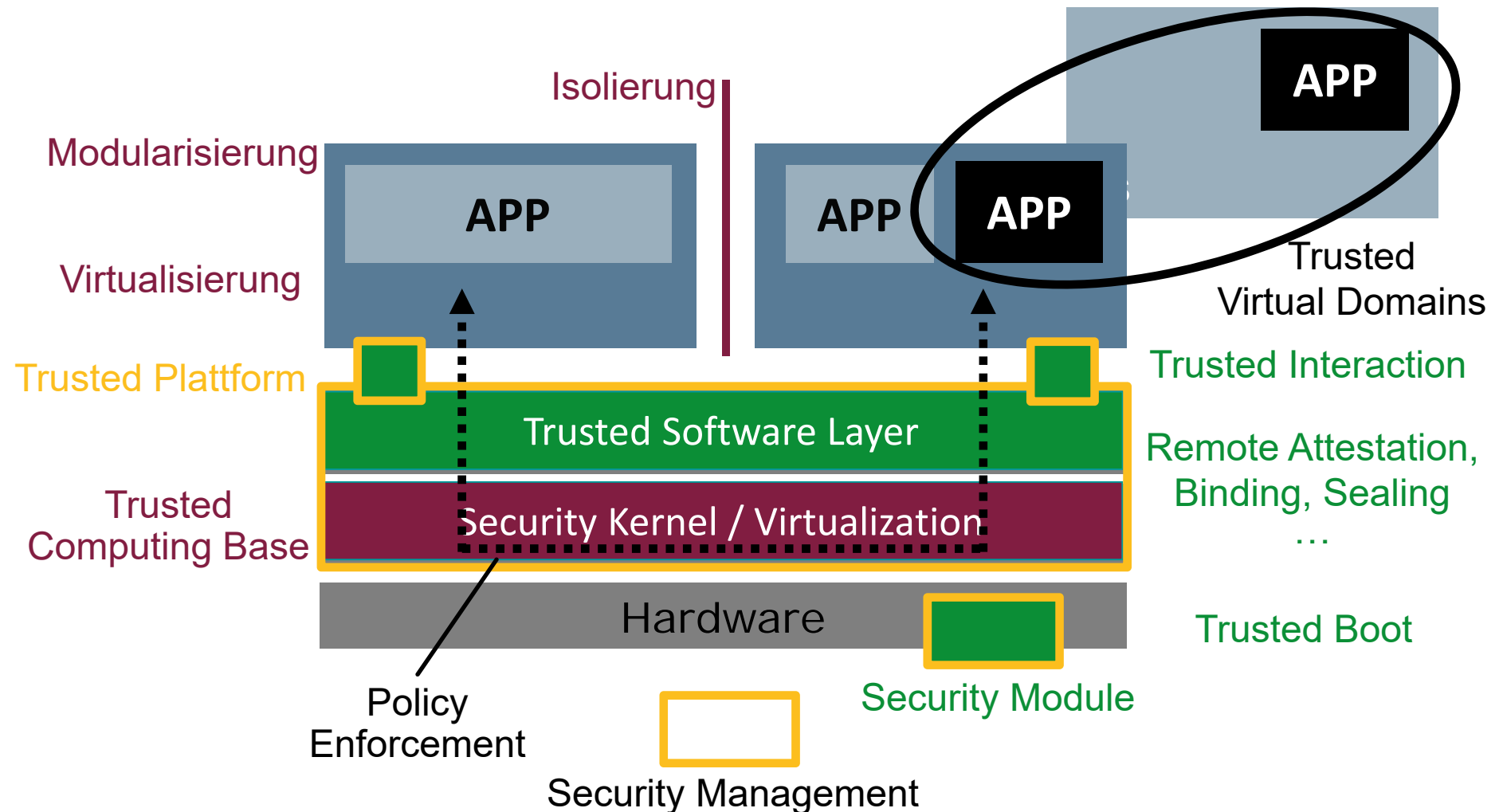


# Kernfunktionalitäten → Sicherheitsprinzipien

## Robustness/Modularity

Trusted Process

## Integritätsprüfung





# Trusted Computing

## → Inhalt

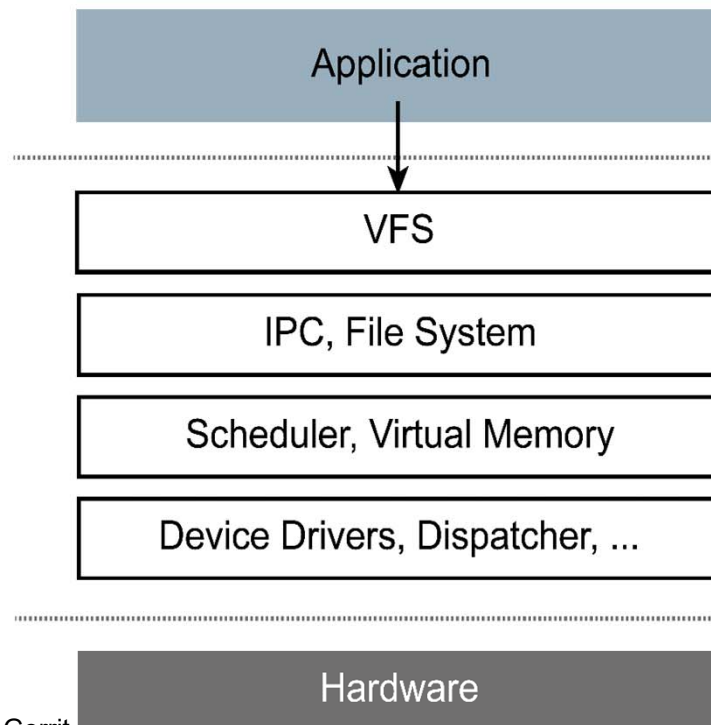
- Ziele und Ergebnisse der Vorlesung
- Kernfunktionalitäten
- **Sicherheitsarchitektur**
- Trusted Network Connect
- Zusammenfassung



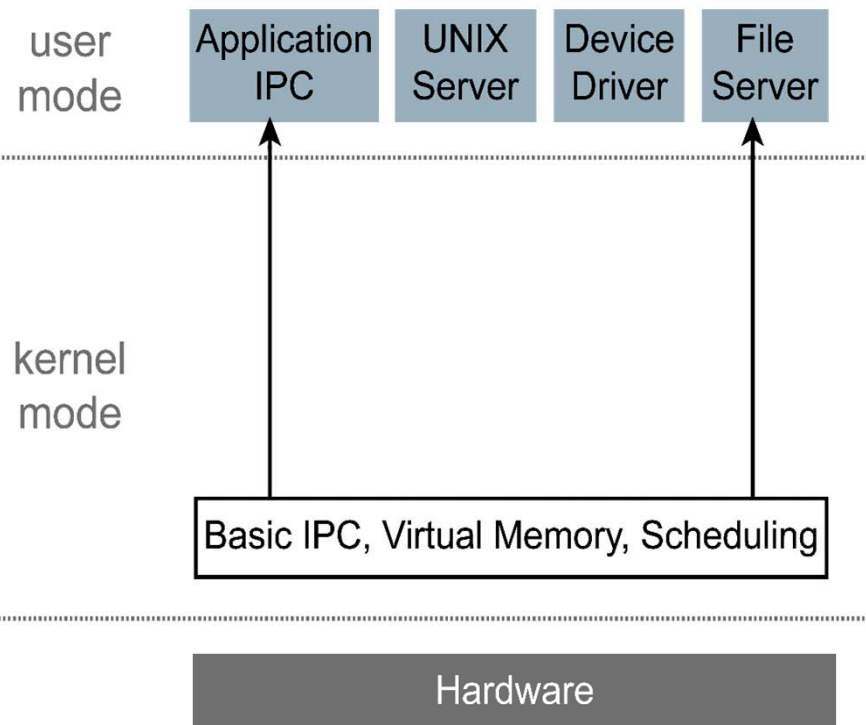
# Sicherheitsarchitektur

## → Kernelarchitekturen (1/3)

### Monolithic Kernel based Operating System



### Microkernel based Operating System





# Sicherheitsarchitektur

## → Kernelarchitekturen (2/3)

- **Vorteile** eines monolithischen Kernels:
  - Lange etabliert
  - Gute Performance
- **Nachteile** eines monolithischen Kernels:
  - Alle Treiber vereint im Kernel-Space
  - Geringere Flexibilität
  - Höhere Komplexität
  - Wenig robust
  - Schlechte Sicherheitsmechanismen





# Sicherheitsarchitektur

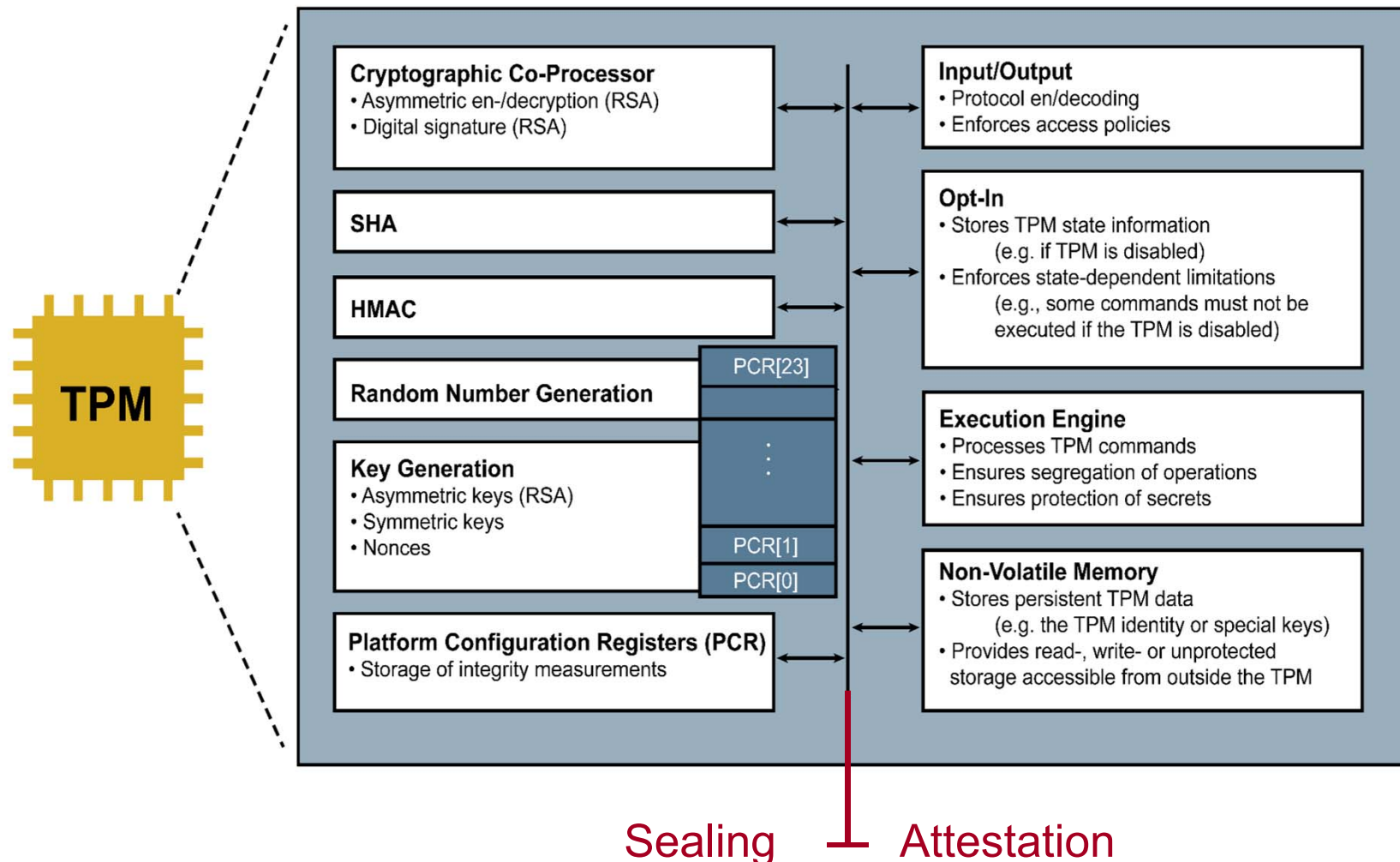
## → Kernelarchitekturen (3/3)

- **Vorteile** eines Mikrokernels:
  - Höhere Robustheit
  - Höhere Modularität
  - Höhere Flexibilität
  - Höhere IT-Sicherheit (kontrollierbare Interprozesskommunikation)
  - Weniger benötigter Speicherplatz
- **Nachteile** eines Mikrokernels:
  - Geringere Performanz, da erhöhte Kommunikation zwischen den Prozessen



# HSM: Trusted Platform Module

## → Hardware-Sicherheitsanker



# Sicherheitsarchitektur

## → Core Root of Trust for Measurement (CRTM)

- Messvorgang über einzelne **Systemzustände** (Hard- und Software).
  - Speicherung der Messungen in den **PCRs**.

### ■ Authenticated Boot:

- Systemzustände messen.
- Speicherung in den PCRs.
- Überprüfung der Integrität.

### ■ Secure Boot:

- Systemzustände messen.
- Überprüfung der Integrität.
- Ggf. Bootvorgang stoppen.

Entity  $E_0$

**"Transitives Vertrauen"**



# Sicherheitsarchitektur

## → Identitäten (1/2)

- **Endorsement Key (EK):**
  - Eindeutige TPM-Identität (nicht migrierbar).
  - RSA-Schlüsselpaar (im Herstellungsprozess erzeugt).
  - Geheimer Schlüssel im TPM gespeichert.
  - Öffentlicher Schlüssel ist datenschutzsensitiv.
  - TPM-Hersteller verwaltet PKI.
- **Endorsement Credential (EC):**
  - Elektronisches Zertifikat vom TPM-Hersteller.
  - Bestätigt ordnungsgemäße Erstellung und Einbettung des EK.
  - Bestandteile: TPM-Herstellernamen, TPM-Modellnummer, TPM-Version, Öffentlicher Schlüssel des EK.

## → Identitäten (2/2)

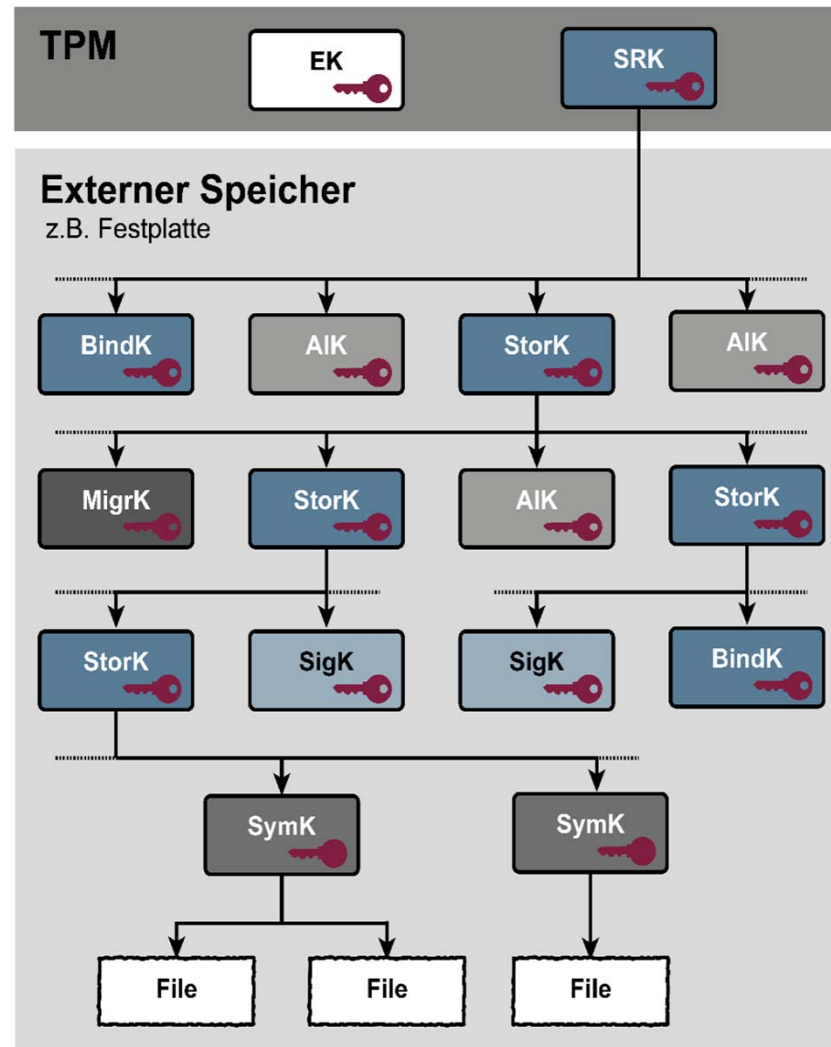
- **Platform Identität (PI):**

- Entspricht der TPM-Identität (EK).
- Physikalische oder logische Bindung des TPMs an die Plattform (z.B. mittels anlöten an das Motherboard oder Kryptographie).
- Plattform  $\hat{=}$  Motherboard/IT-System.
- Plattform muss konform zur Evaluierungsrichtlinien der TCG sein  
→ Conformance Credential (CC)

- **Platform Credential:**

- Elektronisches Zertifikat vom Plattform-Hersteller.
- Bestätigt gültige Verbindung zwischen TPM und Plattform  
→ Trusted Plattform.
- Bestandteile: Name des Plattformherstellers, Plattformmodell und Versionsnummer, Verweise auf die EC und CC.

## → Schlüssel und deren Eigenschaften (1)



# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (2)

- **Migratable Keys** → Auf andere Plattformen übertragbar.
- **Non-Migratable Keys** → An die Plattform gebunden.
- **Storage Root Key (SRK):**
  - Wurzel der Schlüsselhierarchie.
  - Während der Installation des TPM-Eigentümers generiert.
  - Löschung des TPM-Eigentümers → Löschung des SRK → Kein Zugriff auf die Schlüsselhierarchie mehr.
  - **Eigenschaften:**
    - Steht im nicht flüchtigen Speicher des TPMs.
    - Ist nicht migrierbar.

# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (3)

- **Attestation Identity Keys (AIK):**
  - Verwendet für die Trusted Computing Funktion „**Attestation**“:
    - Authentische Bestätigung der Integrität einer Plattformkonfiguration (z.B. aktuelle Hard- und Softwareumgebung).
  - Nötig, da EK datenschutzsensibel ist.
  - AIKs werden vom TPM-Besitzer generiert.
  - TPM/Plattform kann mehrere AIKs besitzen (z.B. für Online-Banking, E-Mail, ...)
  - **Eigenschaften:**
    - Stehen im nicht flüchtigen Speicher des TPMs.
    - Nicht migrierbar.



# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (4)

- **Storage Keys (StorK):**
  - Verschlüsselung von weiteren Schlüsseln und Daten außerhalb des TPMs.
  - Verwendet für die Trusted Computing Funktion „**Sealing**“:
    - Zustand der Plattform wird Teil der Verschlüsselung.
    - Entschlüsselung nur im vorher definierten Zustand möglich.
  - **Eigenschaften:**
    - RSA-Schlüsselpaar.
    - Im Allgemeinen darf die Migration zu anderen TPMs erfolgen.



# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (5)

- **Binding Keys (BindK):**
  - Verschlüsselung von beliebigen Daten außerhalb des TPMs.
  - Entspricht der asymmetrischen Verschlüsselung.
  - **Eigenschaften:**
    - RSA-Schlüsselpaar (es können auch andere Algorithmen vom TPM unterstützt werden).
    - Im Allgemeinen darf die Migration zu anderen TPMs erfolgen.
    - Kann nur mit Binding-Befehlen verwendet werden.



# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (6)

- **Signing Keys (SigK):**
  - Nachweis der Authentizität und Integrität von beliebigen Daten /Protokollnachrichten innerhalb und außerhalb des TPMs.
  - **Eigenschaften:**
    - RSA-Schlüsselpaar (es können auch andere Algorithmen vom TPM unterstützt werden).
    - Im Allgemeinen darf die Migration zu anderen TPMs erfolgen.

# Sicherheitsarchitektur

## → Schlüssel und deren Eigenschaften (7)

**TPM Key Object** 

**General Information**

Key Type

Algorithm

Authorization Secret

**Specific Information**

Key Length

Key Data

**Key Properties**

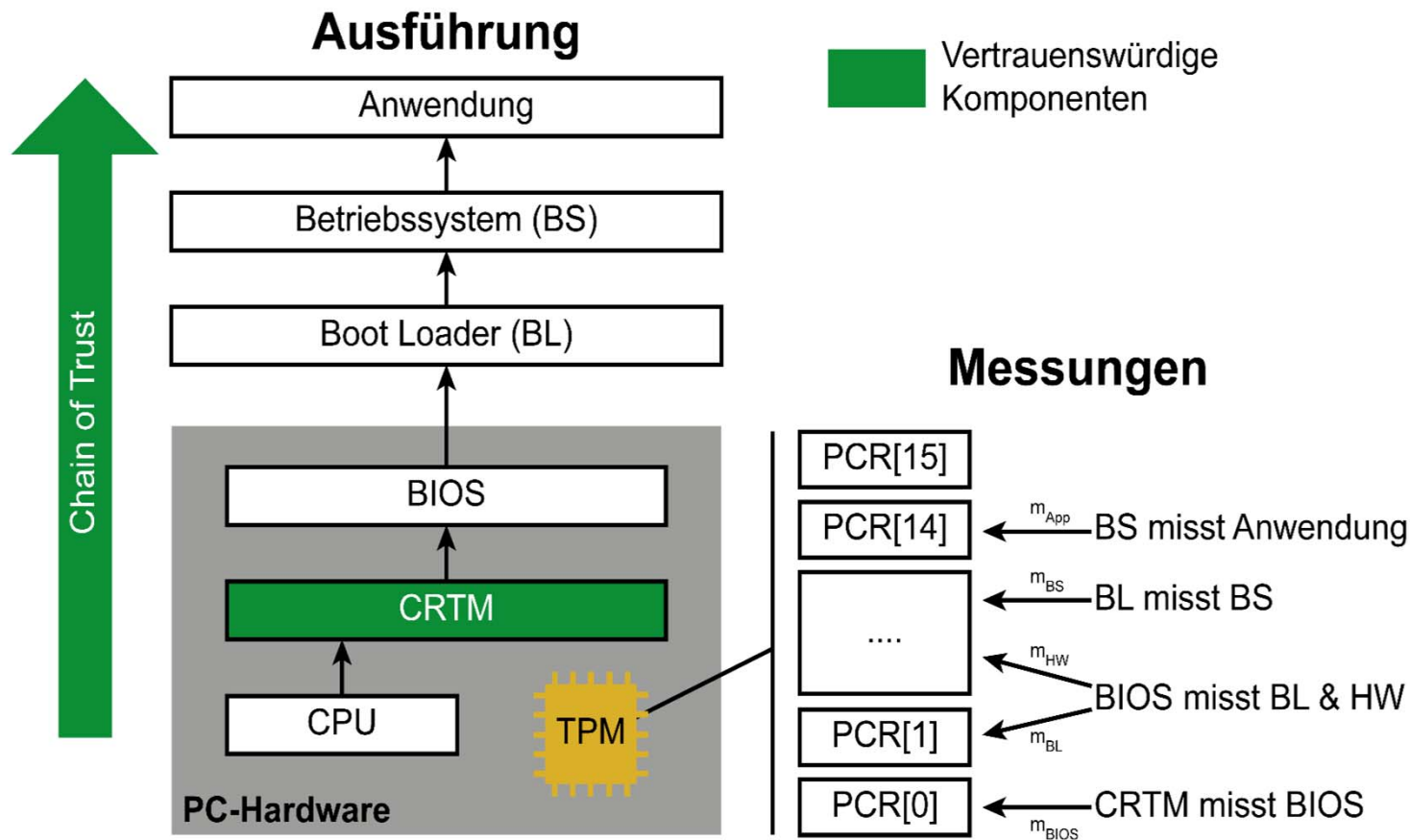
Migration

PCR Values



# Sicherheitsarchitektur

## → Authenticated Boot





# Sicherheitsarchitektur

## → Sealing Funktionen und Parameter

### *Eingabe Parameter*

*daten* {unverschlüsselte Daten}

### *Ausgabe Parameter*

*cipher* {verschlüsselte Daten}

*cryptedKEY* {verschlüsselter Schlüssel}

### *TPM Interne Funktionen und Daten*

*encrypt ( key, daten )* {symmetrischer Verschlüsselungsalgorithmus „AES“}

*H ( daten )* {One-Way-Hashfunktion „SHA-256“}

*genKey()* {Schlüsselerzeugung}

*SRK* {Storage Root Key}

*PCRs* {PCR-0, PCR-1, ...} z.B. aktuell abgespeicherte PCR-Werte

*plainKEY = genKEY ()*

*cipher = encrypt ( plainKEY, ( daten // H ( daten // PCR-0 // ... // PCR-x ) )*

*cryptedKEY = encrypt ( SRK, plainKEY // H ( plainKEY ) )*



# Sicherheitsarchitektur

## → Un-Sealing Funktionen und Parameter

### *Eingabe Parameter*

<i>cipher</i>	<i>{verschlüsselte Daten}</i>
<i>crypteKEY</i>	<i>{verschlüsselter Schlüssel}</i>

### *Ausgabe Parameter*

<i>daten</i>	<i>{unverschlüsselte Daten}</i>
--------------	---------------------------------

### *TPM Interne Funktionen und Daten*

<i>decrypt ( key, daten )</i>	<i>{symmetrischer Verschlüsselungsalgorithmus „AES“}</i>
<i>H ( daten )</i>	<i>{One-Way-Hashfunktion „SHA-256“}</i>
<i>checkPCRs ( Hash-Value )</i>	<i>{vergleicht PCRs-Inhalte mit Hash-Value}</i>
<i>SRK</i>	<i>{Storage Root Key}</i>
<i>PCRs</i>	<i>{PCR-0, PCR-1, ...}</i>

*plainKEY = decrypt ( SRK, crypteKEY )*

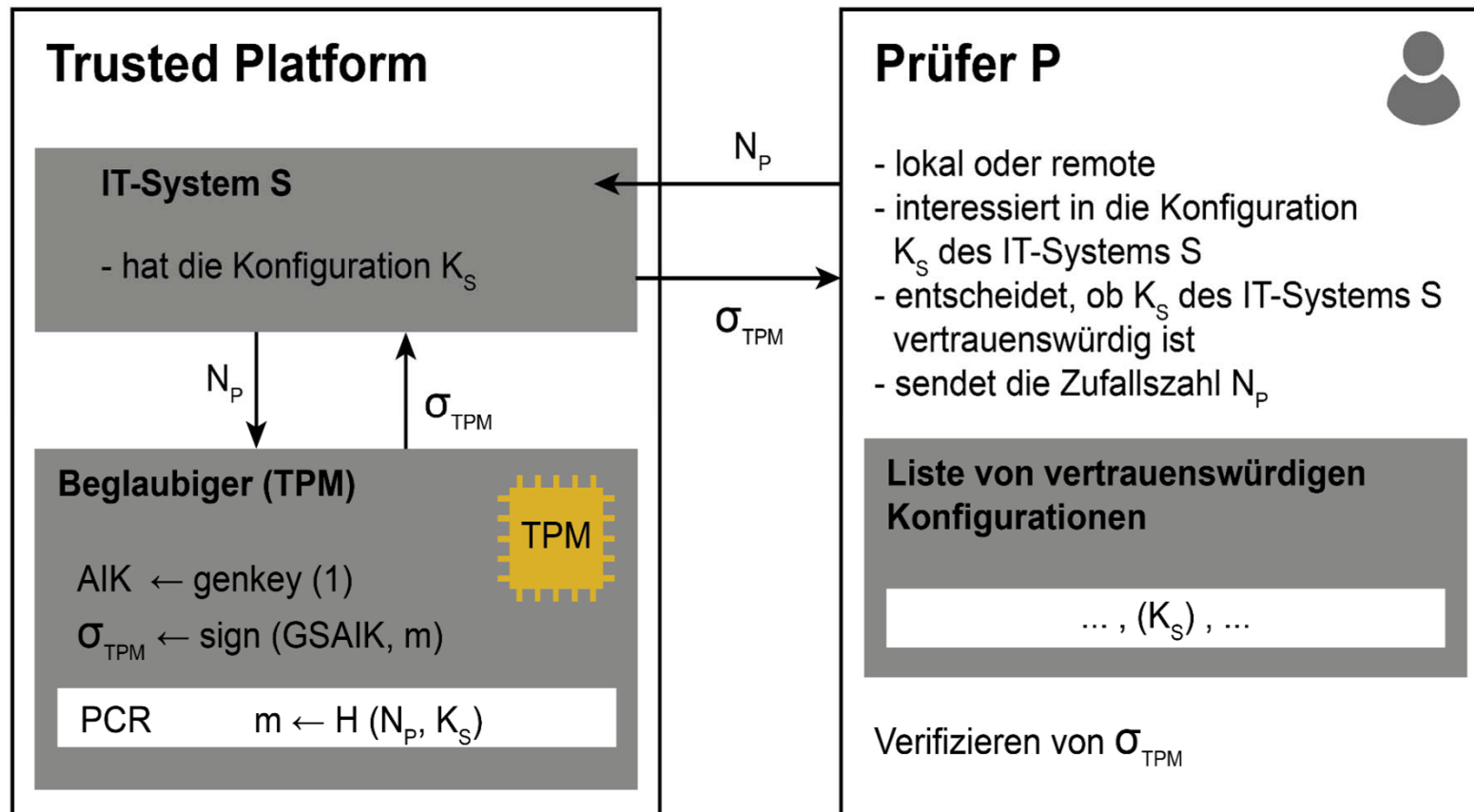
*daten // H ( daten // PCR-0 // ... // PCR-x ) = decrypt ( plainKEY, cipher )*

*if ( checkPCRs ( Hash-Value ) )*

*return daten*

*else*

*return ERROR*







# Sicherheitsarchitektur

## → Signaturfunktion für die Attestierung

### *Eingabe Parameter*

*random*

*{Zufallszahl des Prüfers  $P - N_P$ }*

### *Ausgabe Parameter*

*signature//certificate*

*{Signatur der aktuellen Systemkonfiguration des AIKs}*

### *TPM Interne Funktionen und Daten*

*sign ( key, daten )*

*{RSA-Signatur}*

*H ( daten )*

*{One-Way-Hashfunktion "SHA-256"}*

*GSAIK*

*{geheimer AIK-RSA-Schlüssel}*

*AIK-certificate*

*{elektronisches Zertifikat des AIKs}*

*PCRs*

*{PCR-0, PCR-1, ...} z.B. aktuell abgespeicherte PCR-Werte*

$$\sigma_{TPM} = \text{sign} ( \text{GSAIK}, H ( \text{random} // \text{PCR-0} // \dots // \text{PCR-x} ) )$$



# Sicherheitsarchitektur

## → Verifikationsfunktion für die Attestierung

### *Eingabe Parameter*

*signature//certificate*

*{Signatur der Systemkonfiguration des AIKs}*

### *Ausgabe Parameter*

*return value*

*{Rückgabewert}*

### *TPM Interne Funktionen und Daten*

*very ( key, daten )*

*{RSA-Signatur-Verifikation}*

*H ( daten )*

*{One-Way-Hashfunktion "SHA-256"}*

*ÖSAIK*

*{öffentlicher AIK-RSA-Schlüssel}*

*checkPCRs ( PCR-Values )*

*{vergleicht den Inhalt der PCRs mit den gewünschten Werten}*

*PCRs*

*{PCR-0, PCR-1, ...}*

*if ( very ( ÖSAIK,  $\sigma_{TPM}$  ) ) and*

*if ( checkCERT ( certificate ) ) and*

*if ( checkPCRs ( Hash-Value ) )*

*return ok*

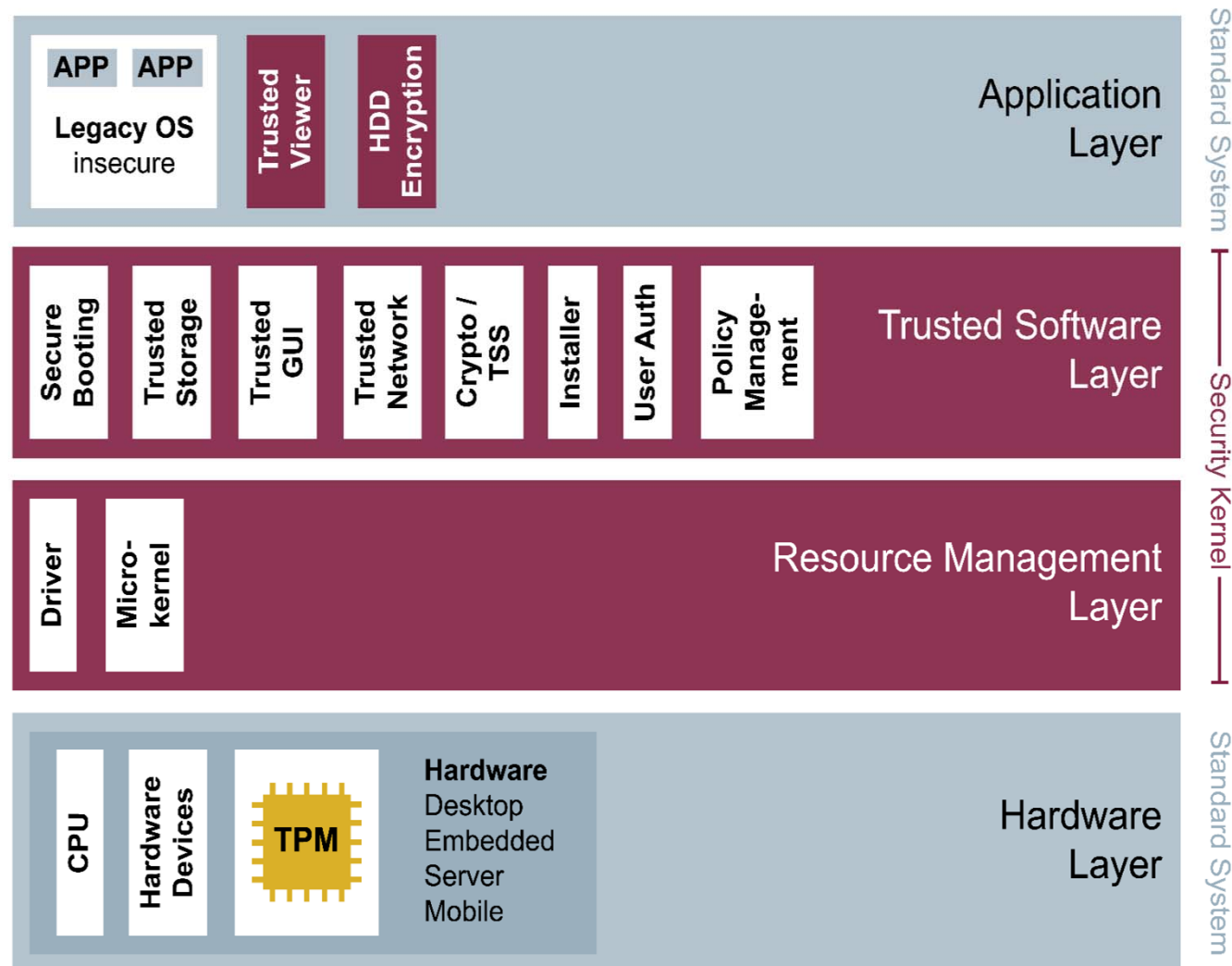
*else*

*return ERROR*



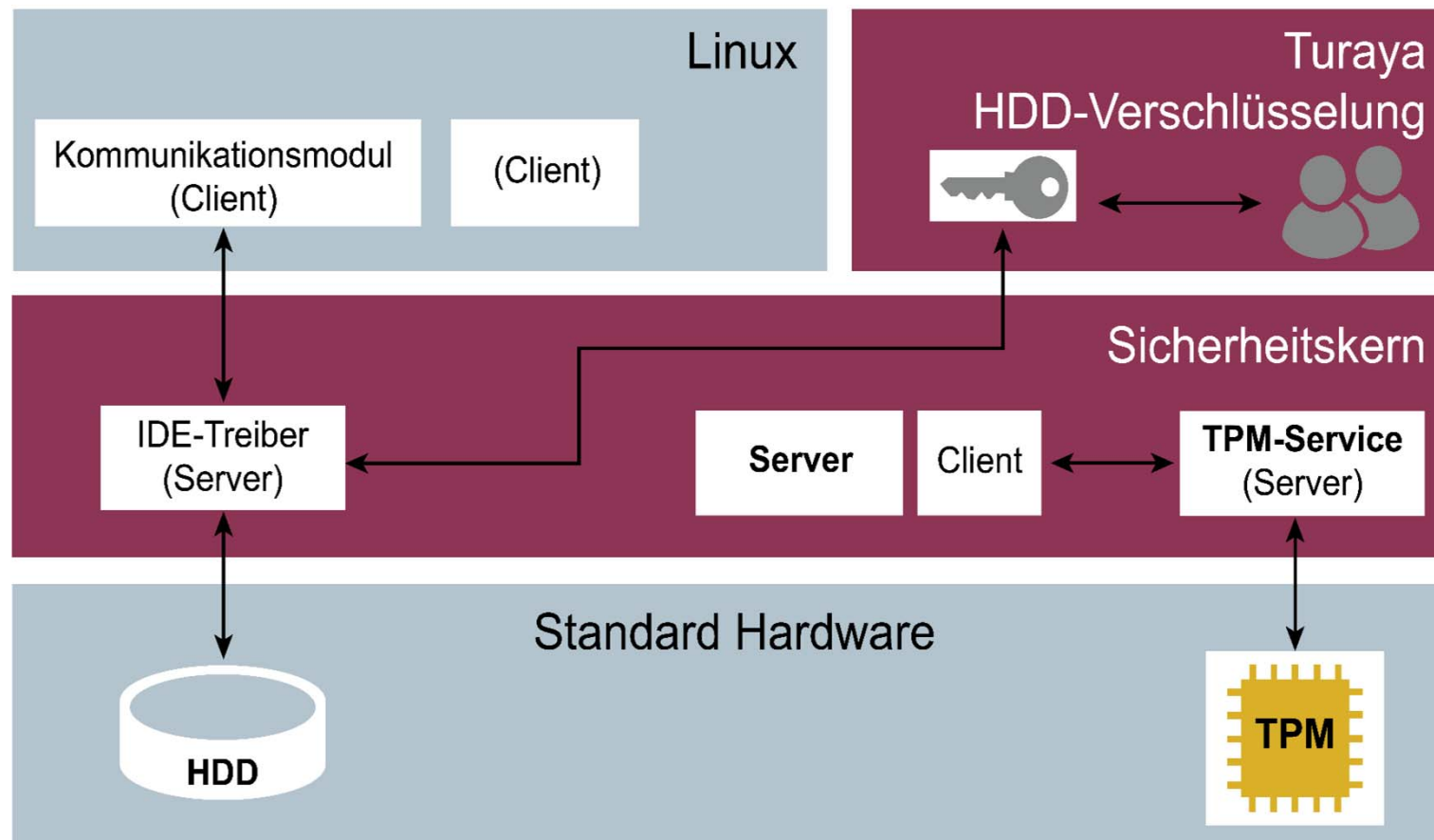
# Sicherheitsarchitektur

## → Trusted Plattform



# Sicherheitsarchitektur

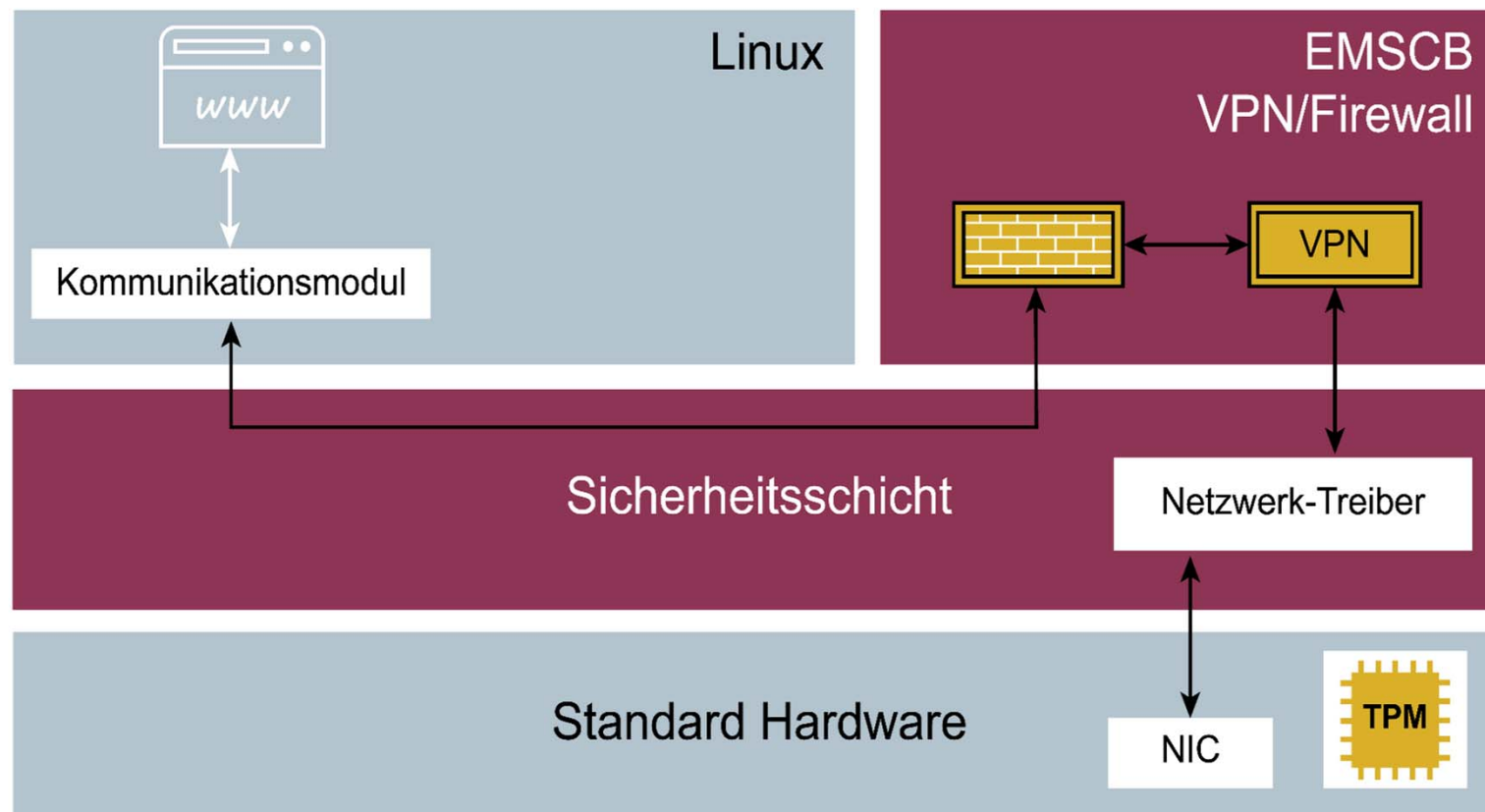
## → Beispielanwendung: Turaya.Crypt





# Sicherheitsarchitektur

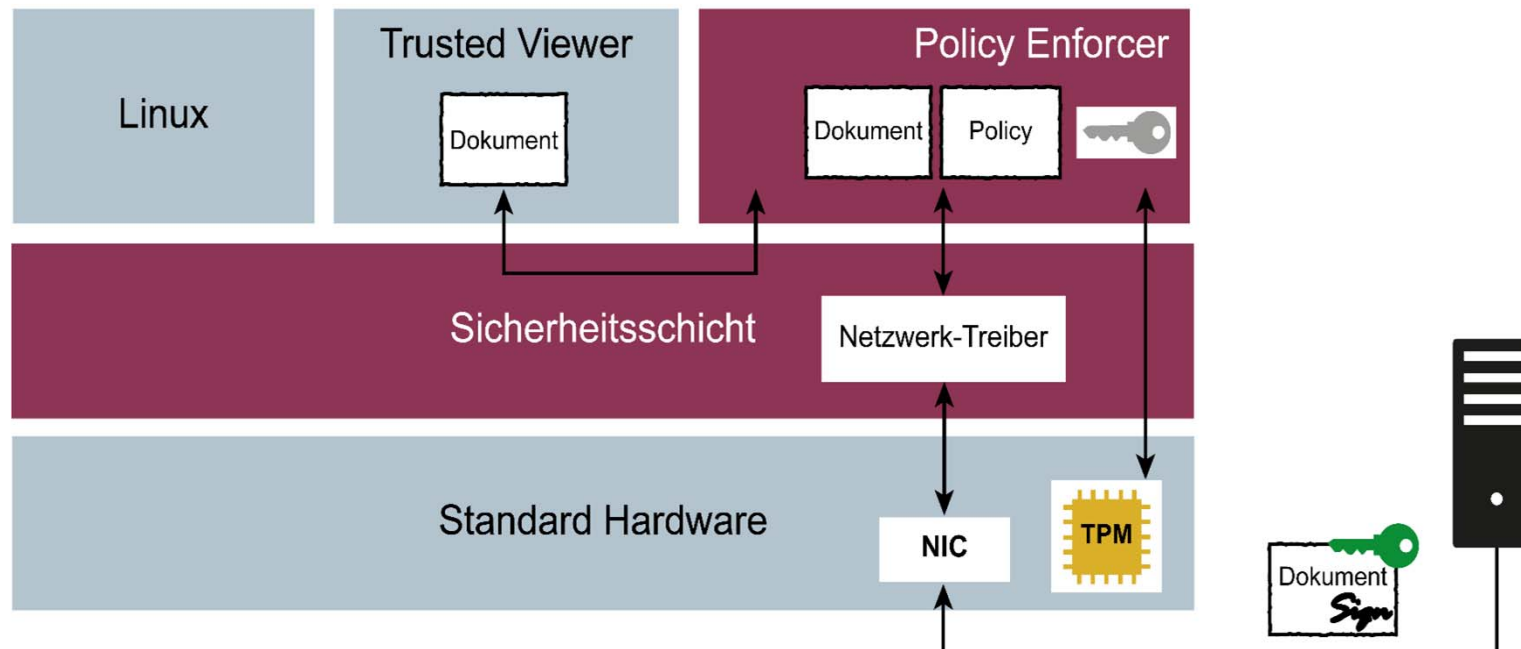
## → Beispielanwendung: Turaya.VPN





# Sicherheitsarchitektur

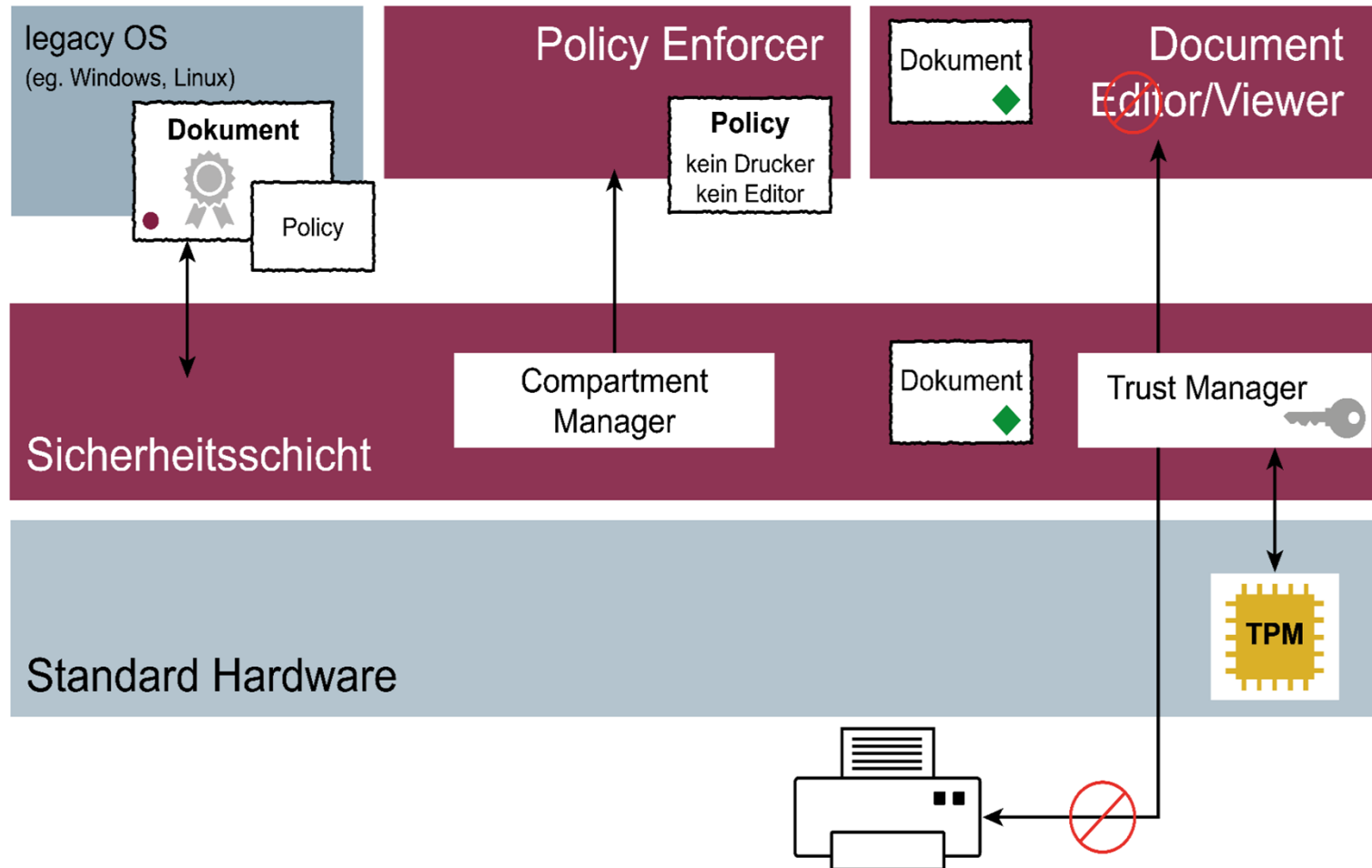
## → Beispielanwendung: Turaya.FairDRM





# Sicherheitsarchitektur

## → Beispielanwendung: Turaya.FairDRM





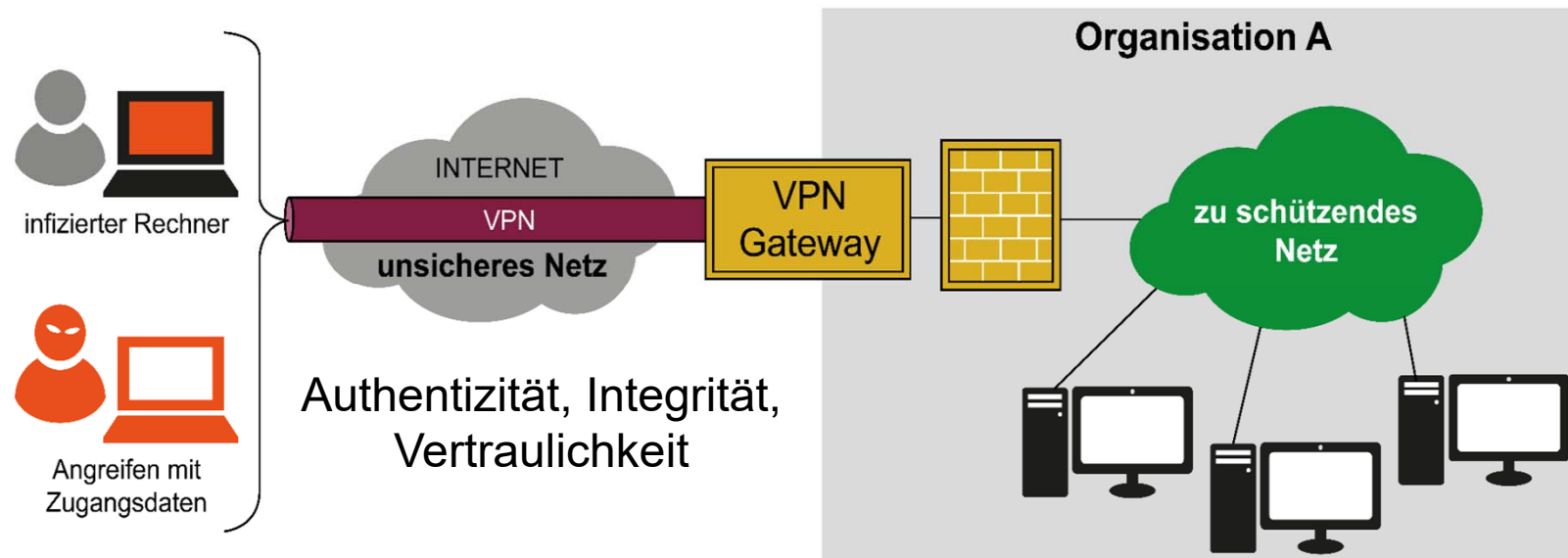
# Trusted Computing

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Kernfunktionalitäten
- Sicherheitsarchitektur
- **Trusted Network Connect**
- Zusammenfassung



# Trusted Network Connect → Problemstellung



## Lösungsansätze:

Microsoft NAP,

Cisco NAC,

Trusted Computing Group (TCG) → **Trusted Network Connect (TNC)**



# Trusted Network Connect

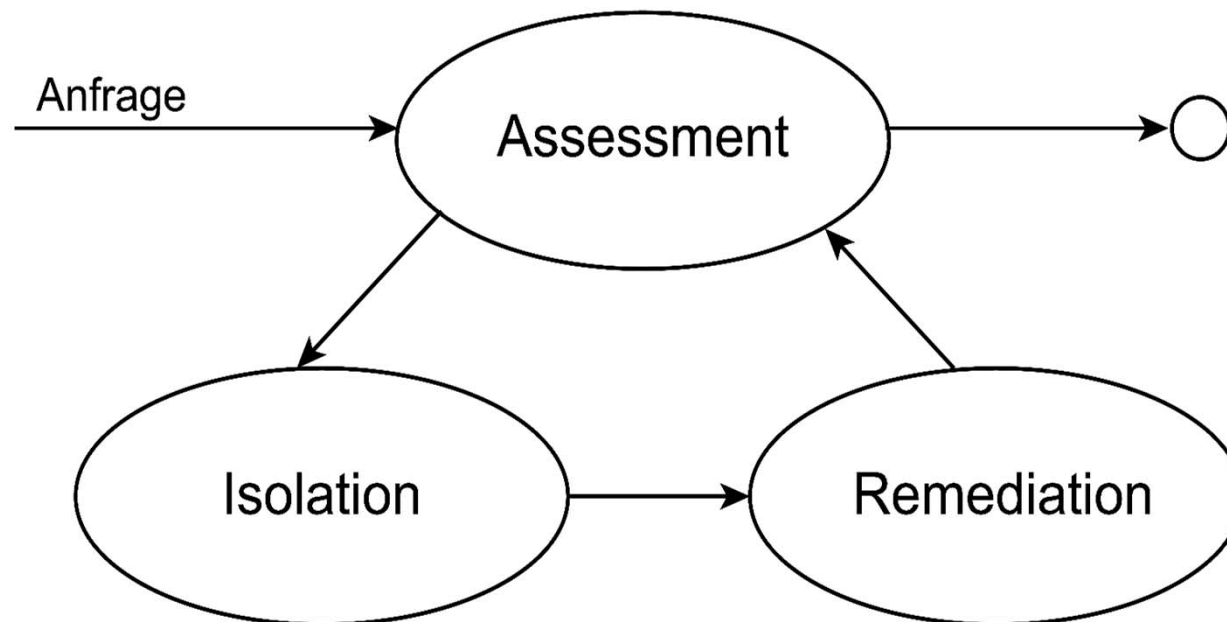
## → Vertrauenswürdige Netzwerkverbindungen

- Vertrauenswürdigkeit abhängig von der **Integrität**.
- **Alle** beteiligten **Kommunikationspartner** betrachten.
- **Kommunikationsrichtung** getrennt betrachten.
- Alle IT-Systeme, Infrastrukturelemente und das Umfeld der Kommunikation bewerten.
- Anforderungen in **Policies** definieren (z.B. erlaubte Konfigurationen, Betriebssysteme, Hard- und Software, ...).
- Überprüfung **vor dem Zugriff (Prävention)**.



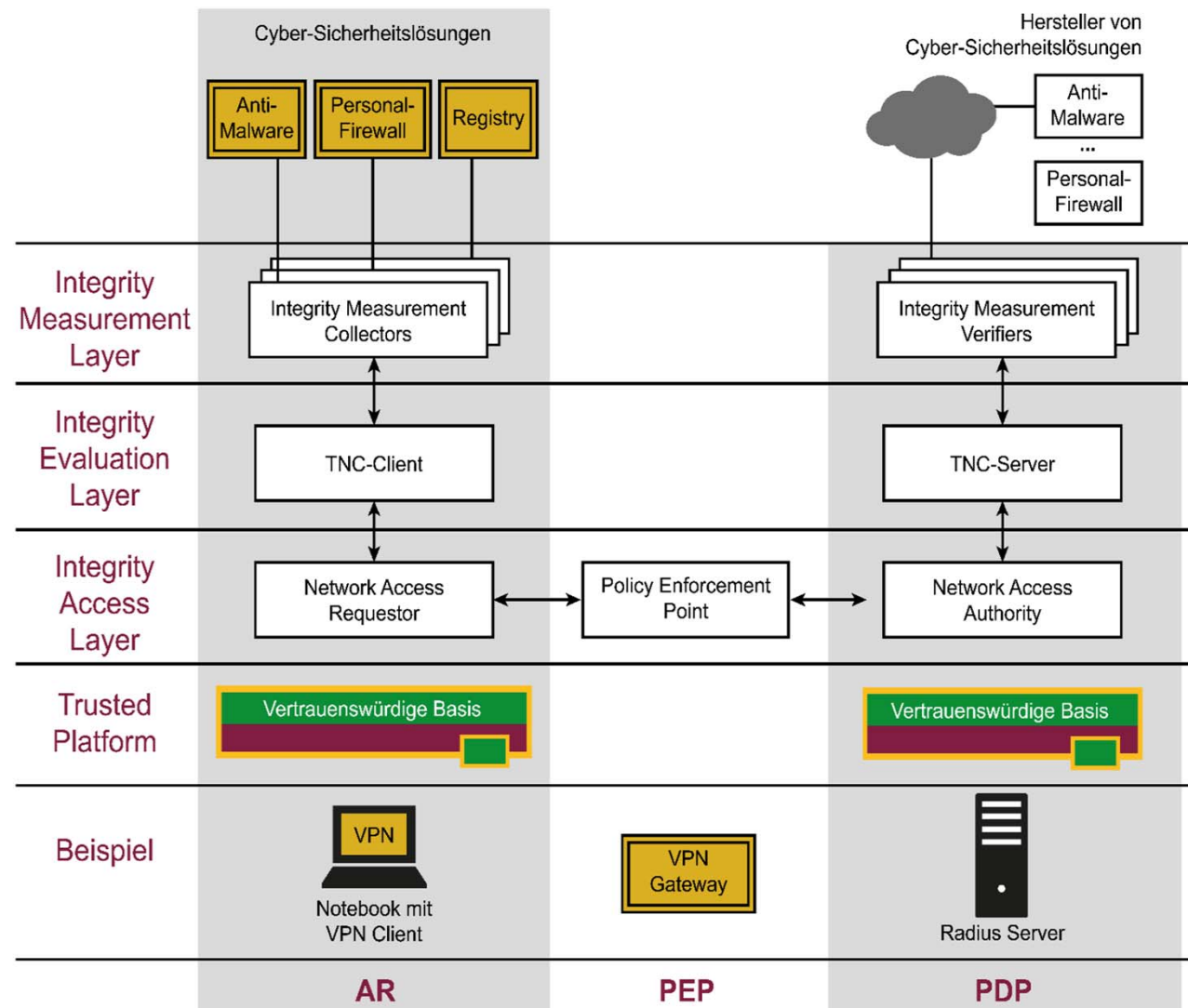
# Trusted Network Connect

## → Phasen



# Trusted Network Connect

## → Struktur





# Trusted Computing

## → Inhalt

- Ziele und Ergebnisse der Vorlesung
- Kernfunktionalitäten
- Sicherheitsarchitektur
- Trusted Network Connect
- **Zusammenfassung**

# Trusted Computing

## → Zusammenfassung (1/2)

- Die Kernfunktionalitäten von Trusted Computing sind:
  - **Robustheit und Modularität**
  - **Integritätsüberprüfung**
  - **Trusted Process**
  - **Trusted Plattform**
- Vor- und Nachteile der verschiedenen **Kernelarchitekturen** müssen miteinander abgewogen werden.
- **CRTM** ist die Vertrauensbasis. Das Vertrauen ist **transitiv**.
- Die **TPM Schlüsselhierarchie** ermöglicht eine sichere Speicherung von Daten, auch auf externen Speichermedien.



# Trusted Computing

## → Zusammenfassung (2/2)

- Wichtige Trusted Computing Funktionen sind:
  - **Authenticated Boot**
  - **Binding**
  - **Sealing**
  - **Attestation**
- Vertrauenswürdige Netzwerkverbindungen können durch **Trusted Network Connect (TNC)** realisiert werden.
- Die Festlegung einer sicheren und vertrauenswürdigen **Systemkonfiguration** ist mit zahlreichen Schwierigkeiten (technisch und politisch) verbunden.