Fachhochschule Gelsenkirchen

Trusted Computing

→ Eine technologische Betrachtung

Prof. Dr. Norbert Pohlmann

Institut für Internet-Sicherheit
Fachhochschule Gelsenkirchen
https://www.internet-sicherheit.de

emSCB
European Multilaterally Secure Computing Base
www.emscb.org

Agenda

- Trusted Computing Grundlagen
- Trusted Plattform Module (TPM)
- Trusted Computing Funktionalitäten
- Virtualisierung
- Betriebssysteme / Microkernel
- Trusted Computing Base
- Sicherheitsplattform Turaya
- Anwendungsbeispiele

Agenda

- **Trusted Computing Grundlagen**
  - **Idee**
  - **Trusted Computing Platform Alliance**
  - **Trusted Computing Group**
  - **Status Quo**

Trusted Computing Grundlagen
→ Grundsätzliche Idee

- Durch ein Zusammenwirken von **vertrauenswürdigen Hardware-**, **Firmware-** und **Software-Sicherheitsmechanismen** sollen Rechnersysteme sicherer werden, so dass ihnen vertraut werden kann.
  - Integritätsprüfung des Betriebssystem
  - Authentisierung von Hard- und Software gegenüber dem Betriebssystem und externen Kommunikationspartnern
  - Verbesserter Datenschutz und verbesserte Sicherheit beim Aufbewahren und Übertragen von Daten
  - Schutz vor Malware (Virmen, Würmer, Trojaner, …)
  - Usw.

- **Verhält sich ein Rechnersystem für eine spezielle Aufgabe so, wie wir es erwarten, dann kann dem Rechnersystem vertraut werden.**

Trusted Computing Grundlagen
→ Trusted Computing Platform Alliance (TCPA)

- **Ziel**:
  - **Durch den Einsatz von spezieller Krypto-Hardware und darauf aufbauenden Betriebssystemen die Sicherheit verbessern**

- 1999 von Microsoft, Intel, IBM, Compaq und HP gegründetes Hersteller-Konsortium
- 180 Mitglieder (u.a. Infineon, Siemens, RSA, Nokia, Utimaco)
- Erste Veröffentlichung der Spezifikationen als Version 0.9 im August 2000

Trusted Computing Grundlagen
→ Trusted Computing Group (TCG) (1/3)

- **Ziel:**
  - **Entwicklung und Support von offenen Industriestandards für „Trusted Computing“ auf verschiedenen Plattformen (PC‘s, Server, Handys und PDA‘s)**

- Gegründet von AMD, HP, Intel und Microsoft
- Seit April 2003 Rechtsnachfolger der TCPA

- **Relevante veröffentliche Spezifikationen:**
  - Trusted Platform Module (TPM)
  - Trusted Software Stack Specification (TSS)
  - Trusted Network Connect Architecture (TNC)
  - Mobile Trusted Module Specification (MTM)

Trusted Computing Grundlagen
→ Trusted Computing Group (TCG) (2/3)

- **Aktueller Versionsstand der Spezifikationen:**
  - **TCG TPM Main Specification**
    - **Version 1.1b** → Februar 2002
    - **Version 1.2** → Oktober 2003
  - **TCG Software Stack Specification**
    - **Version 1.1** → August 2003
    - **Version 1.2** → Januar 2006
  - **TCG TNC Architecture**
    - **Version 1.0** → Mai 2005
    - **Version 1.1** → Mai 2006
  - **Mobile Trusted Module Specification**
    - **Version 0.9** → September 2006

Trusted Computing Grundlagen
→ Trusted Computing Group (TCG) (3/3)

- **Development Policies**
  - **Open Platform Development Model**
    - Plattformunabhängig
  - **Platform Owner and User Control**
    - Benutzer soll vollständige Kontrolle behalten
    - Alle Features deaktivierbar
  - **Privacy Effect of TCG Specifications**
    - Sicherheit der persönlichen Identifizierbarkeit

Trusted Computing Grundlagen
→ Status Quo (1/2)

- **TPM nach 1.1b und 1.2 Spezifikation erhältlich von**
  - Infineon, National Semiconductor, Atmel
- **Konforme Systeme werden ausgeliefert von**
  - Dell, Fujitsu Siemens, HP und IBM
- **Verfügbare Applikationen**
  - Turaya, RSA Secure ID, Checkpoint VPN, Verisign PTA, …
  - Software von IBM
    - Windows: verändertes Login, rudimentäre Verschlüsselungswerkzeuge
    - Linux: Testpaket (samt Quellen) mit Kernel-Modul, Bibliothek, API und Beispielprogrammen

Trusted Computing Grundlagen
→ Status Quo (2/2)

- **Forschungsprojekte in Rahmen von Trusted Computing**
  - **European Multilaterally Secure Computing Base (EMSCB)**
    - **EMSCB** strebt die Entwicklung einer vertrauenswürdigen, fairen und offenen Sicherheitsplattform mit offenen Standards an, die viele Sicherheitsprobleme herkömmlicher Rechnersystemplattformen löst.
  - **Open Trusted Computing (OTC)**
    - Beim **OTC** Konsortium handelt es sich um ein Forschungs und Entwicklungsprojekt, das sich zum Ziel gesetzt hat, vertrauenswürdige und sichere Computersysteme auf Basis von Open Source Software zu entwickeln.
    - Das Projekt fokussiert sich dabei sowohl auf herkömmliche Standard Computer Systeme als auch Embedded Systeme wie z.B. Mobiltelefone.

Agenda

- Trusted Plattform Module
  - Idee
  - Aufbau eines TMPs
  - Random Number Generator Unit
  - Hash Unit
  - Keyed Hashing for Message Authentication
  - RSA Unit
  - Endorsement Key
  - Attestation Identity Keys
  - Storage Root Key
  - Key-Slots
  - Platform Configuration Register
  - Taking TPM Ownership

Trusted Platform Module (TPM)
→ Idee

- Das TPM entspricht im Prinzip einer fest eingebauten SmartCard (**kleines Sicherheitsmodul**), die an ein Rechnersystem gebunden ist, wie z.B. PC, Notebook, PDA, Drucker, Router, Kühlschrank, usw.
- Kosten sollen kleiner als **ein** € sein!
- **Verbreitung der TPMs:**
  - 60 Millionen bis Ende 2006
  - 130 Millionen bis Ende 2007
  - 200 Millionen bis Ende 2008

TPM

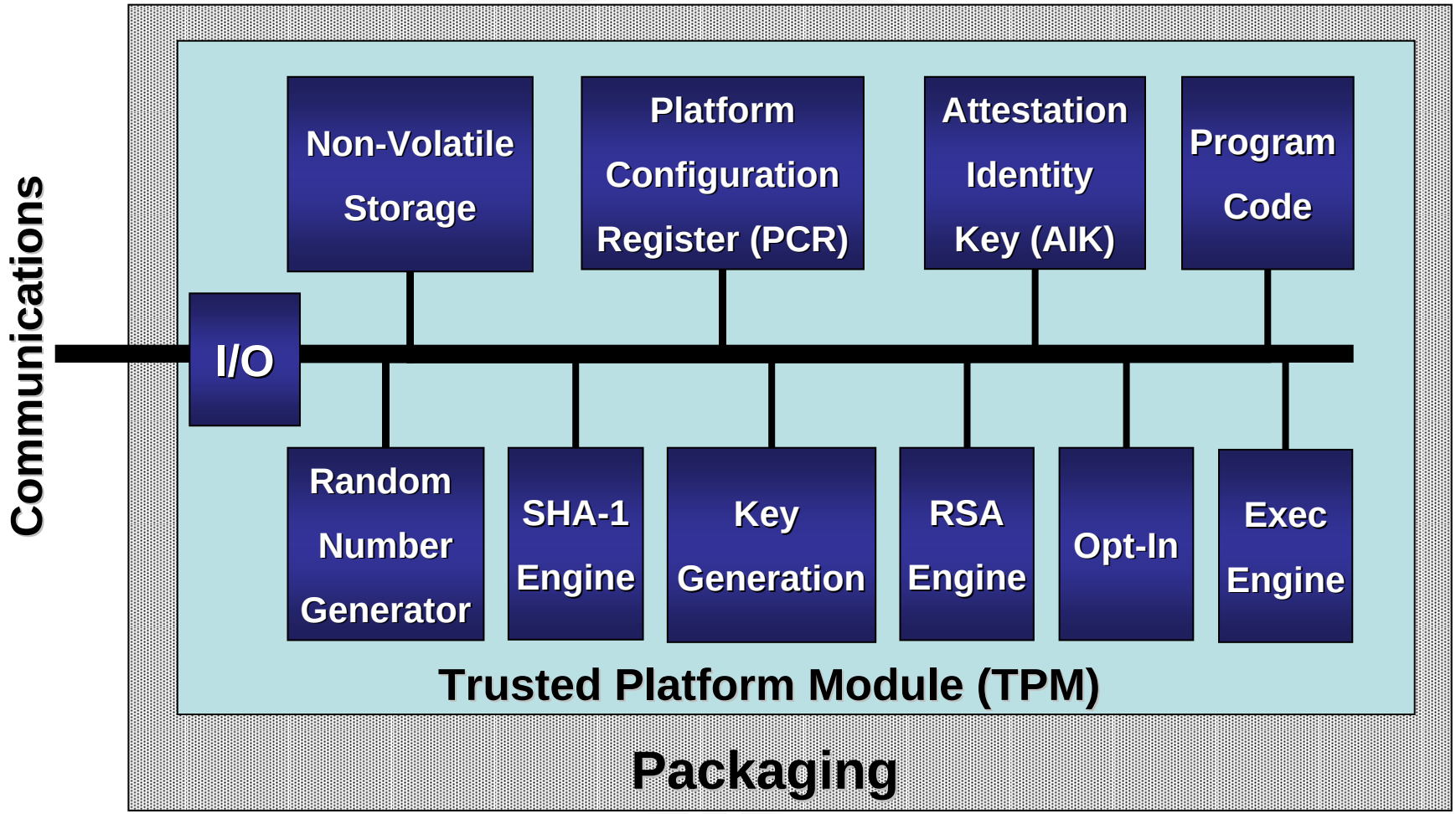
- Einheitliche Standard-Software im TPM.
- Die einzelnen Unternehmen machen dann ihre eigene Lösung.
  - Z.B. Microsoft: Next Generation Secure Computing Base (NGSCB)
  - EMSCB

Trusted Platform Module
→ Hauptfunktionen

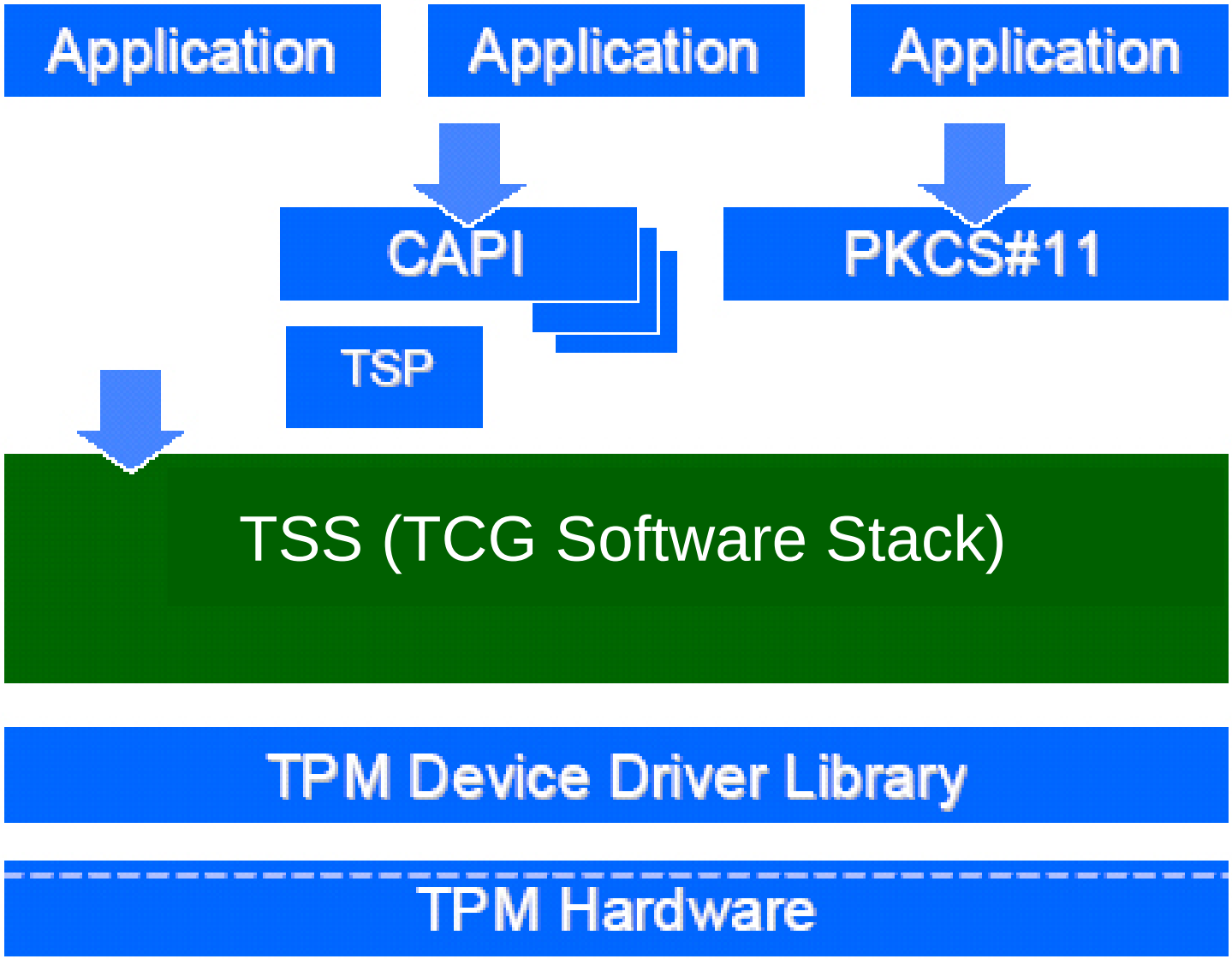
- **Schlüsselgenerierung, Ver- und Entschlüsselung**
  - Kann asymmetrische Schlüssel unter Verwendung eines sicheren Zufallszahlengenerators generieren
  - Mechanismus zur sicheren persistenten Datenspeicherung
  - Sichere Schlüsselspeicherung
  - Ver- und Entschlüsselung von aymmetrischen und symmetrischen Kryptographieverfahren

- **Platform Configuration Register (PCR)**
  - Speicherung der aktuellen Systemkonfiguration von Soft- und Hardware durch Hashwerte
  - Aufbau einer Chain of Trust beim Bootvorgang

Trusted Platform Module (TPM)
→ Basisfunktionen im TPM



Trusted Platform Module (TPM)
→ Software Stack (TSS)



Trusted Plattform Module
→ TPM 1.1b Funktionen (1/2)

- **Kryptographische Funktionseinheiten**
  - Random Number Generator (RNG)
  - Hash-Einheit (SHA-1)
  - DES, Triple-DES, optional AES192
  - Keyed Hashing for Message Authentication (HMAC)
  - RSA Engine
    - Generator für RSA-Schlüssel mit bis zu 2.048 Bit
    - Erzeugen von Signaturen
    - Ver- und Entschlüsseln

- **Nicht flüchtiger Speicher**
  - Endorsement Key (EK)
  - Attestation Identity Keys (AIK)
  - Storage Root Key (SRK)

Trusted Plattform Module
→ TPM 1.1b Funktionen (2/2)

- **Flüchtiger Speicher**
  - 10 Key-Slots für temporäre RSA Schlüssel
  - 16 Platform Configuration Register (PCR)
  - **Key Handles**
    Um temporär geladenen Schlüsseln Namen zur weiteren Bearbeitung zuzuweisen
  - **Authorization Session Handle**
    Wird genutzt um den Status der Autorisation für mehrere hintereinander abfolgende Befehle beizubehalten

- **Funktionalität**
  - Trusted (Authenticated) Boot (Secure Boot)
  - Sealing (Versiegelung)
  - Binding (Auslagerung von Schlüsseln)
  - (Remote-) Attestation (Beglaubigung der Systemkonfiguration)
  - Schutz kryptografischer Schlüssel

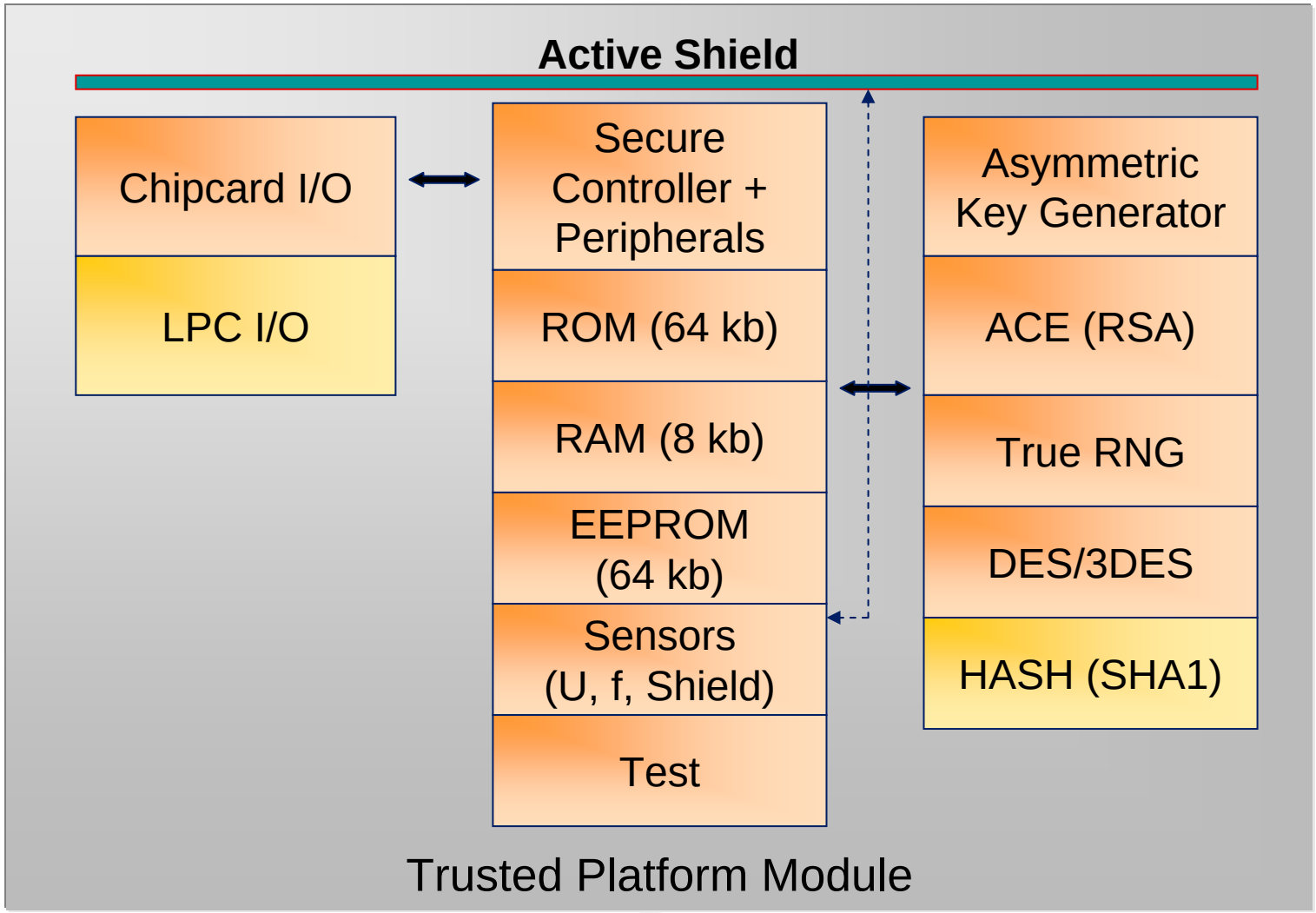
Trusted Plattform Module
→ TPM 1.2 Funktionen

- **Neue Features in TPM 1.2**
  - **Kryptographische Funktionseinheiten**
    - AES192, AES256, Triple-DES
    - Hash-Einheit:SHA-256
  - **Flüchtiger Speicher**
    - 24 Key-Slots für temporäre RSA Schlüssel
    - 32 Platform Configuration Register (PCR)

- **Neue Funktionalität**
  - Direct Anonymous Attestation
  - Removable Endorsement Key

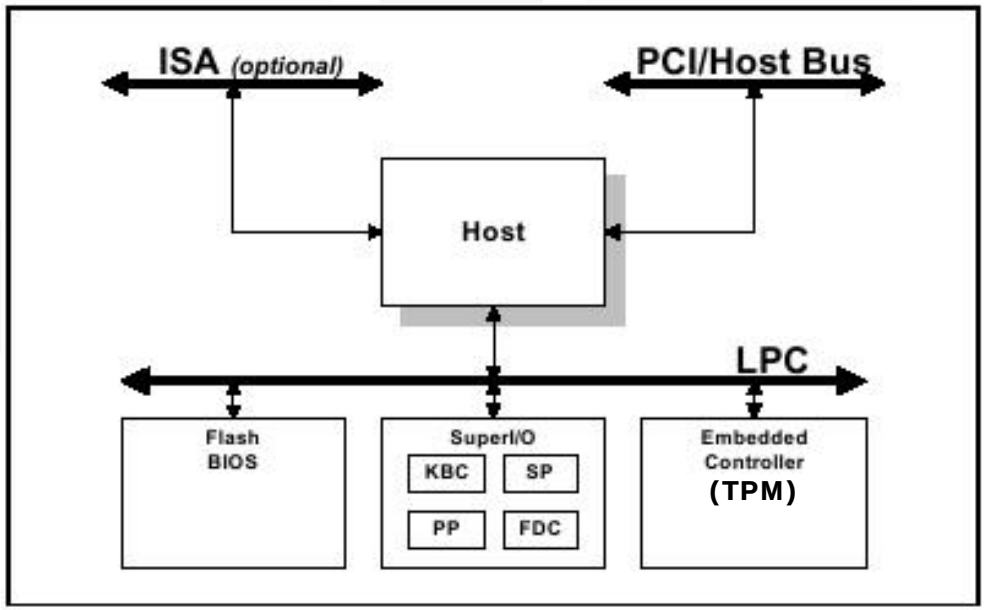
Trusted Plattform Module

→ TPM Aufbau Blockschaltbild (Version 1.1b)



Trusted Plattform Module
→ Low Pin Count - Bus

- Über den den LPC-Bus werden TPMs angebunden.
- Low Pin Count (LPC) ist ein Bus in PC Systemen.
- Der LPC-Bus wird wie ein ISA-Bus angesprochen.
- Beim LPC-Bus wird auch von serialisiertem ISA-Bus gesprochen.
- Hardwaremäßig ist der LPC-Bus jedoch ein serieller Bus, der keine Ähnlichkeiten zum ISA-Bus hat.



Trusted Plattform Module
→ Random Number Generator Unit (RNG)

- Nicht-deterministischer Zufallszahlengenerator **integriert im TPM**
  - Ein nicht-deterministischer Zufallszahlengenerator liefert bei **gleichen** Ausgangsbedingungen immer eine **unterschiedliche** Folge von Zufallszahlen
  - Ein nicht-deterministischer Zufallszahlengenerator verwendet **physikalische Vorgänge** zur Bestimmung der Zufallszahlen, er wird daher auch als physikalischer Zufallszahlengenerator bezeichnet.Mechanismen sind z.B.:
    - Laufzeiten von analogen Bauelementen (Schwingkreis)
    - Umgebungswärme (Wärmesensor)
    - Usw.
- Ein in Hardware realisierter **RNG** liefert hochwertige Zufallszahlen.
- Diese Zufallszahlen haben statistische Eigenschaften wie **gleichmäßige Häufigkeitsverteilung** und **geringe Korrelation**.

Trusted Plattform Module
→ Hash Unit (SHA-1, SHA-256)

- Eine **Hash Unit** stellt Funktionen zur Verfügung, die zu einer Eingabe aus einer üblicherweise **großen Quellmenge** eine Ausgabe aus einer im Allgemeinen **kleineren Zielmenge** (160 Bit) erzeugt.
- Diese Zielmenge wird als **Hash-Wert** bezeichnet.
- Eine Hash-Wert bildet einen Fingerabdrücken eines gesamt Quellmenge.
- Die im **TPM integrierte Hash-Unit** beschleunigt die Berechnung des Hash-Wertes und bildet eine vertrauenswürdige Instanz zur Erstellung.
- Die **Hash Unit** bildet eine wichtige Komponente für die Trusted Computing Funktionalitäten, da sie bei **fast jeder Funktion** verwendet wird.

Trusted Plattform Module
→ Keyed Hashing for Message Authentication (HMAC)

- Ein **HMAC** ist eine Art **Message Authentication Code** (MAC), der basierend auf einer kryptografischen Hash-Funktion berechnet wird.
- **HMACs** werden in vielen modernen Protokollen wie beispielsweise TLS oder **IPsec** verwendet.
- Die Sicherheit des **HMAC** erfordert nicht zwingend kollisionsresistente Hash-Funktionen, so dass der **HMAC** auch auf Basis des **MD5** berechnet werden kann.
- Der **HMAC** wird aus der Nachricht **N** und einem geheimen Schlüssel **K** mittels der Hash-Funktion **H** wie folgt berechnet.
- **K** wird auf die Blocklänge der Hash-Funktion (512 Bit für die meisten gängigen Hash-Funktionen) aufgefüllt.
- Falls die Länge von **K** größer als die Blocklänge der Hash-Funktion ist, wird **K** durch **H(K)** ersetzt.

$$HMAC_K(N) = H\Big((K \oplus opad) \,||\, H\big((K \oplus ipad) \,||\, N\big)\Big)$$

Die Werte *opad* und *ipad* sind dabei Konstanten, $\oplus$ steht für die bitweise XOR Operation und $||$ für die Verknüpfung durch einfaches Zusammensetzen.

Trusted Plattform Module
→ RSA Unit

- Die **RSA Unit** die im **TPM** integriert ist, beschleunigt die Rechenoperationen, die bei RSA-Algorithmus benötigt werden.
- RSA ist ein asymmetrisches Kryptosystem, das sowohl zur **Verschlüsselung** als auch zur **digitalen Signatur** verwendet werden kann.
- Es verwendet ein Schlüsselpaar bestehend aus einem **privaten Schlüssel**, der zum **Entschlüsseln** oder **Signieren** von Daten verwendet wird, und einem **öffentlichen Schlüssel**, mit dem man **verschlüsselt** oder **Signaturen prüft**.
- Der **private Schlüssel** wird **geheim** gehalten und kann nicht oder nur mit extrem hohen Aufwand aus dem **öffentlichen Schlüssel** berechnet werden.
- **RSA** ist nach seinen Erfindern Ronald L. Rivest, Adi Shamir und Leonard Adleman benannt.

Trusted Plattform Module
→ Schlüsseltypen

- **Non-Migratable Key**
  - Ein Schlüssel kann nicht kopiert und nicht in ein anderes TPM übertragen werden.
  - Beispiele:
    - EK: Endorsement Key (in jedem TPM)
    - SRK: Storage Root Key (in jedem TPM)
    - AIK: Attestation Identity Key

- **Migratable Key**
  - Diese Schlüssel werden vom Nutzer generiert und gespeichert.
  - Sie sind auf andere Speichermedien und insbesondere auch auf ein anderes TPM übertragbar.
  - Beispiele:
    - Alle vom Anwender gespeicherte Schlüssel

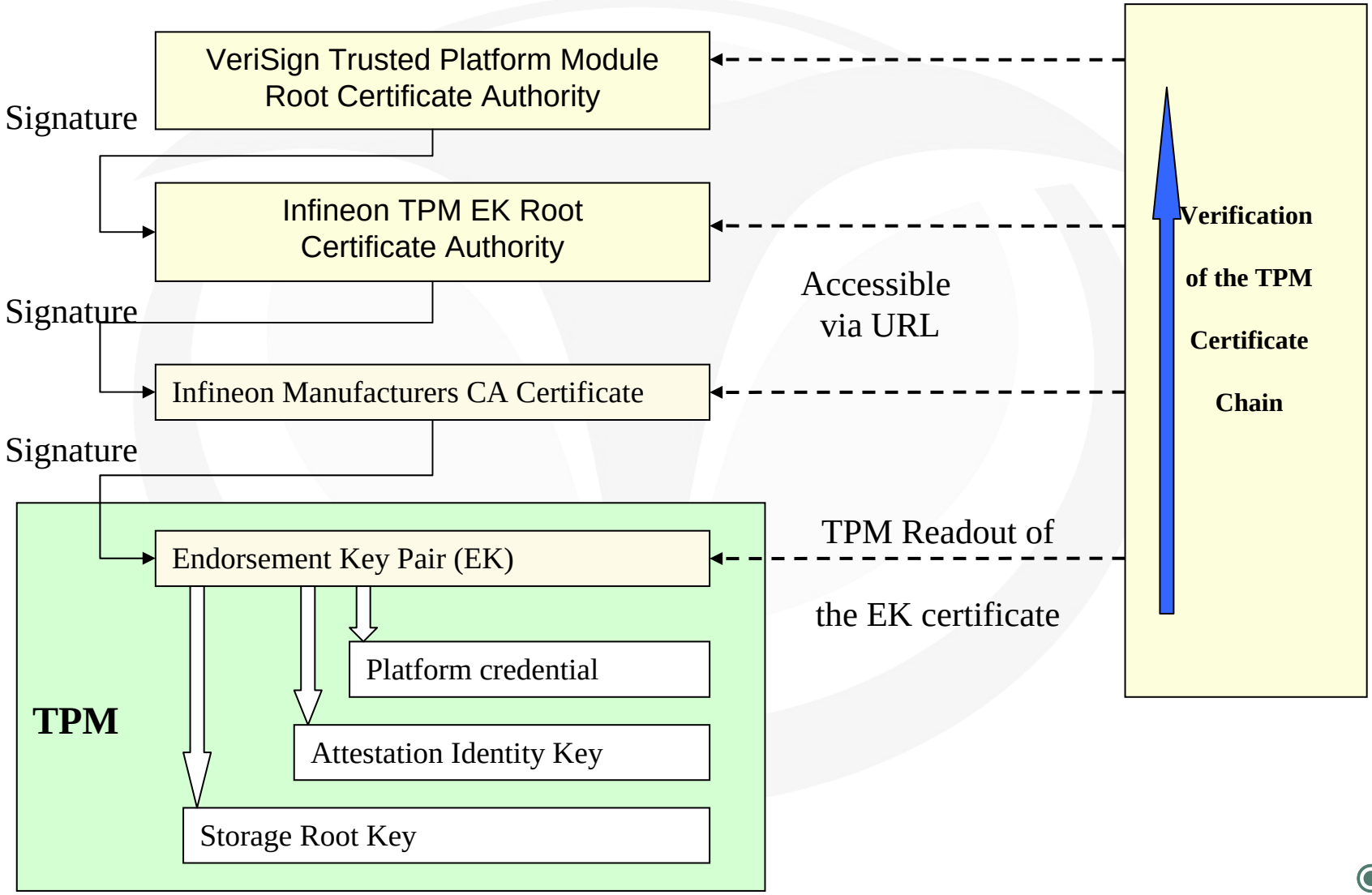
Trusted Plattform Module
→ Endorsement Key (EK) 1 / 3

- Der **Endorsement Key (EK)** ist ein 2048 Bit langer RSA-Schlüssel (Modulus n und d).
- Der Endorsement Key (EK) wird beim Herstellungsprozess des TPMs zufällig vom Hersteller generiert.
- Der Endorsement Key (EK) ist einem TPM eindeutig zugeordnet und ist einzigartig.
- Der private Teil des RSA-Schlüssels des EKs verlässt nie das TPM
- Der EK wird von einer Zertifizierungsstelle zertifiziert (Endorsement Key Zertifikat).
- Der EK ist Basis für weitere Schlüssel

Trusted Plattform Module
→ Endorsement Key (EK) 2 / 3

- Kann nicht gelöscht oder geändert werden
  - ab TCG **TPM** 1.2 ist dies möglich (Removable **EK**)
- **Privater Schlüssel** verlässt das **TPM niemals**
  - Der **EK** wird niemals direkt verwendet, es werden Unterschlüssel auf Basis des **EK** erzeugt, die für verschiedene Zwecke eingesetzt werden.
  - Der **EK** bildet den Root-Schlüssel für eine komplexe Schlüsselkette
- Da der **öffentliche Schlüssel** aus Sicht der Privatsphäre kritisch ist, kann er durch den Benutzer deaktiviert werden.

Trusted Plattform Module
→ Endorsement Key (EK) 3/3



Trusted Plattform Module
→ Attestation Identity Keys (AIK) – (1/2)

- Der **Attestation Identity Key (AIK)** ist ein von einem TPM erzeugtes Schlüsselpaar, das zur Beglaubigung der Vertrauenswürdigkeit gegenüber Dritten genutzt und dessen öffentlicher Teil an diese weitergegeben wird.
- Es sind keine Rückschlüsse auf den Endorsement Key möglich, aus dem es erzeugt wurde.
- 2048 Bit RSA Schlüsselpaar (öffentlich/privat) mit einem fixierten öffentlichen Exponenten ($e = 2^{16} + 1$)
- Jedem Besitzer des **TPMs** erzeugt den **AIK** neu
- Dieses Schlüsselpaar ist nicht migrierbar

Trusted Plattform Module
→ Attestation Identity Keys (AIK) – (2/2)

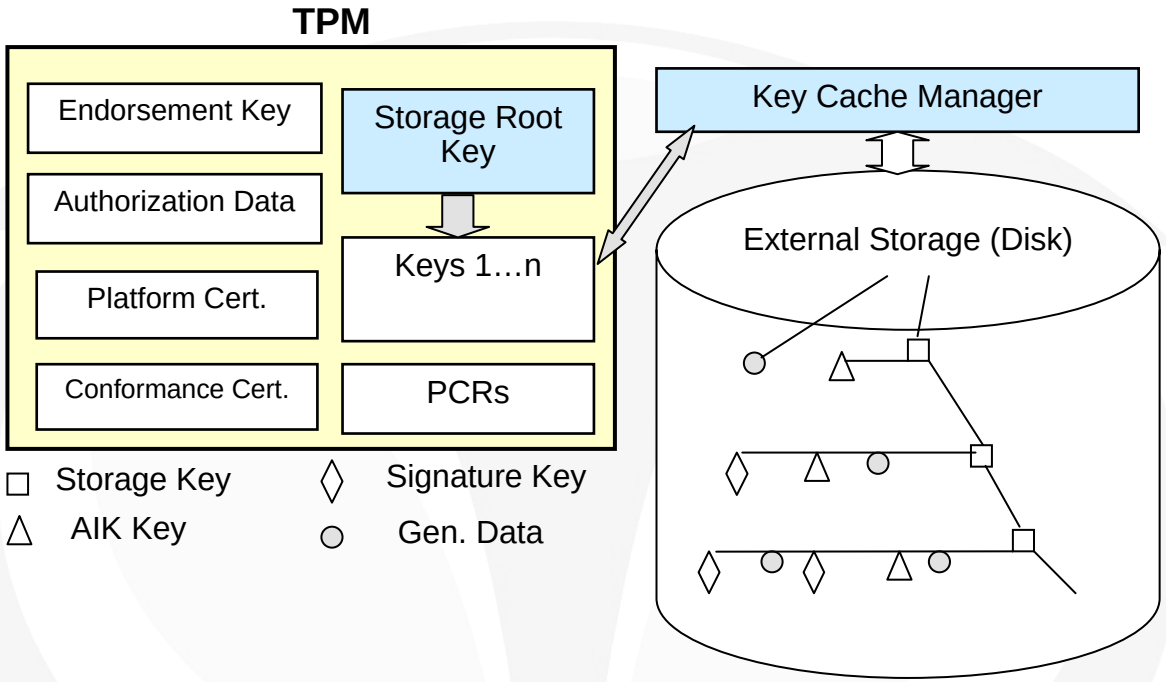
- Der **AIK** darf nur für die Signatur von **PCR**-Werten (Attestation) eingesetzt werden.

- Das Konzept des **AIKs** wurde eingeführt, weil der **EK** eines **TPM** nicht direkt für die Beglaubigung der Plattformintegrität (Attestation) eingesetzt werden sollte.
  - Da der **EK** immer eindeutig ist, wäre die Privatsphäre der Nutzer beeinträchtigt
  - Deshalb werden **AIKs** quasi als Pseudonym für den **EK** in solchen Beglaubigungsprozessen verwendet
  - Um dennoch sicherzustellen, dass nur konforme TC-Plattformen gültige **AIKs** erstellen, müssen die Schlüssel durch eine vertrauenswürdige Dritte Partei (Trusted Third Party) bestätigt werden, diese Bestätigung erfolgt in Form eines **AIK**-Zertifikats (Credential).

Trusted Plattform Module
→ Storage Root Key (SRK) 1/2

- Der **Storage Root Key (SRK)** ist ein RSA-Schlüssel innerhalb des TPM, auf dem die lokale Verschlüsselung von Daten aufbaut.
- Der Storage Root Key (SRK) ist ein 2048 Bit langer RSA-Schlüssel
- Der SRK wird beim TPM TakeOwnership Kommando generiert.
- Der öffentliche Schlüssel kann exportiert werden, der private Schlüssel bleibt im TPM.
- Der private Schlüssel verlässt das TPM nie, kann jedoch vom Benutzer gelöscht werden.
- Wechselt der Besitzer des Rechners, wird ein neuer **SRK** erzeugt.
- Dieses Schlüsselpaar ist migrierbar, und kann somit in einem Backupprozesse einbezogen werden
- Der **SRK** dient allein dem Zweck, weitere benutzte Schlüssel zu verschlüsseln und stellt somit die Wurzel des **TPM**-Schlüsselbaumes dar.

Trusted Plattform Module
→ Storage Root Key (SRK) 2/2



- Der Storage Root Key ist die Wurzel aller anderen Keys.
- Aber auch die verschlüsselten Daten hängen vom SRK ab.
- Mit Hilfe des TSS Core Services können beliebige Keys gespeichert werden.
- Er wird automatisch mit der “Take Ownership” Operation generiert.

Trusted Plattform Module
→ Zertifikate (1/2)

- Die Zertifikate dienen dem Nachweis der Vertrauenswürdigkeit des Rechnersystems im Auslieferungs- bzw. Herstellungszustand.

- **Endorsement Zertifikat**
  - Dieses Zertifikat bestätigt die Echtheit des TPM.
  - Genau genommen wird sichergestellt, dass das TPM von einem autorisierten Hersteller bereitgestellt wurde.

- **Platform Zertifikat**
  - Das Plattform-Zertifikat wird vom Hersteller der Plattform - also etwa eines PCs, Notebook oder Mobiltelefons - ausgestellt.
  - Es bestätigt, dass alle Plattform-Komponenten der TCG-Spezifikation genügen und dass die Plattform ein gültiges TPM enthält.
  - Es wird also bescheinigt, dass das aktuelle Rechnersystem eine vertrauenswürdige Plattform darstellt.

Trusted Plattform Module
→ Zertifikate (2/2)

- **Conformance Zertifikat**
  - Dieses Zertifikat bestätigt, dass das TPM-Design in der Plattform der TCG-Spezifikation genügt und das TPM korrekt implementiert ist.

- **Validation Zertifikat**
  - Dieses Zertifikat stellt für Komponenten oder Komponentengruppen die Übereinstimmung und Korrektheit der Implementierung gegenüber der TCG-Spezifikation sicher.

Trusted Plattform Module
→ Key-Slots

- **Key Slots** sind dynamische Schlüsselspeicher im **TPM**
- Die Anzahl der **Key Slots** ist begrenzt => Schlüssel müssen aus **TPM** ausgelagert werden
- Wenn Schlüssel aus **TPM** exportiert wird, dann nur in verschlüsseltem Zustand (mit **SRK** verschlüsselt).
  -> Binding Mechanismus

Trusted Plattform Module
→ Platform Configuration Register (PCR)

- Die **Platform Configuration Register (PCR)** sind Teil des flüchtigen Speichers und dienen zur vertrauenswürdigen Speicherung von 160-Bit Hashwerten, die die Systemkonfiguration widerspiegeln.
- **PCRs** dienen zur Speicherung von Zustandsabbildern der aktuellen Konfiguration von Soft- und Hardware (siehe Trusted Boot).
- Jedes **TPM** muss ein Minimum von 16 PCRs bereitstellen, wobei etwa die Hälfte davon für Integritätsprüfungen der Hardware belegt wird.
- Die Ergebnisse der Integritätsmessungen, die in den **PCRs** gespeichert werden, werden verwendet, um eine Aussage darüber zu treffen, ob sich ein Rechnersystem kompromittiert wurde.
- Wurde ein Rechnersystem von einem Virus oder Trojanischem Pferd befallen, verändert sich das Ergebnis der Integritätsmessungen.
- Mit Hilfe einer Trusted-Third-Party können Integritätsmessungen geprüft und mit gültigen Systemzuständen verglichen werden (Attestierung).

Trusted Plattform Module
→ Taking TPM Ownership

- Nach der Auslieferung eines Rechnersystems mit TPM kann das TPM mit Hilfe des „Take_Ownership“ Kommandos in Besitz genommen werden.
- Das „Take_Ownership“ Kommando benötigt als erstes Argument einen 160-Bit SHA-1 Hashwert eines Passwortes, das der Besitzer als Authentifizierungs-Passwort festlegt.
- Falls das TPM bereits im Besitz eines Benutzers ist, bleibt die Ausführung des „Take_Ownership“ Kommandos wirkungslos.
- Ein Besitzerwechsel ist nur dadurch möglich, dass eine spezielle Reset-Sequenz aktiviert wird.
- Nachdem das „Take_Ownership“ Kommando ausgeführt wurde, wird auch der Storage Root Key generiert.

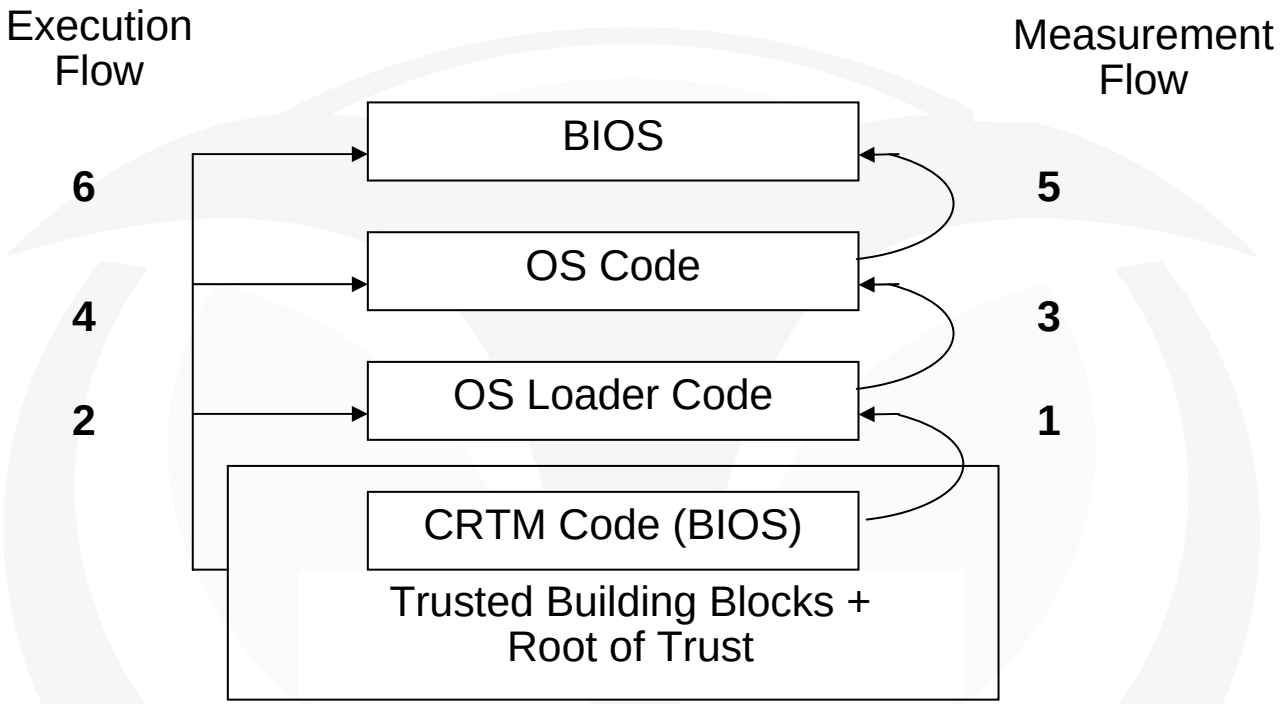
Agenda

- Trusted Computing Funktionalitäten
  - Trusted Boot
  - Sealing (Versiegelung)
  - Binding (Auslagerung von Schlüsseln)
  - Schutz kryptografischer Schlüssel
  - Remote Attestation (Beglaubigung der Systemkonfiguration)
  - Direct Anonymous Attestation
  - Removable Endorsement Key
  - Backup und Migrierbarkeit

Trusted Computing Funktionalitäten
→ Trusted Boot (1/3)

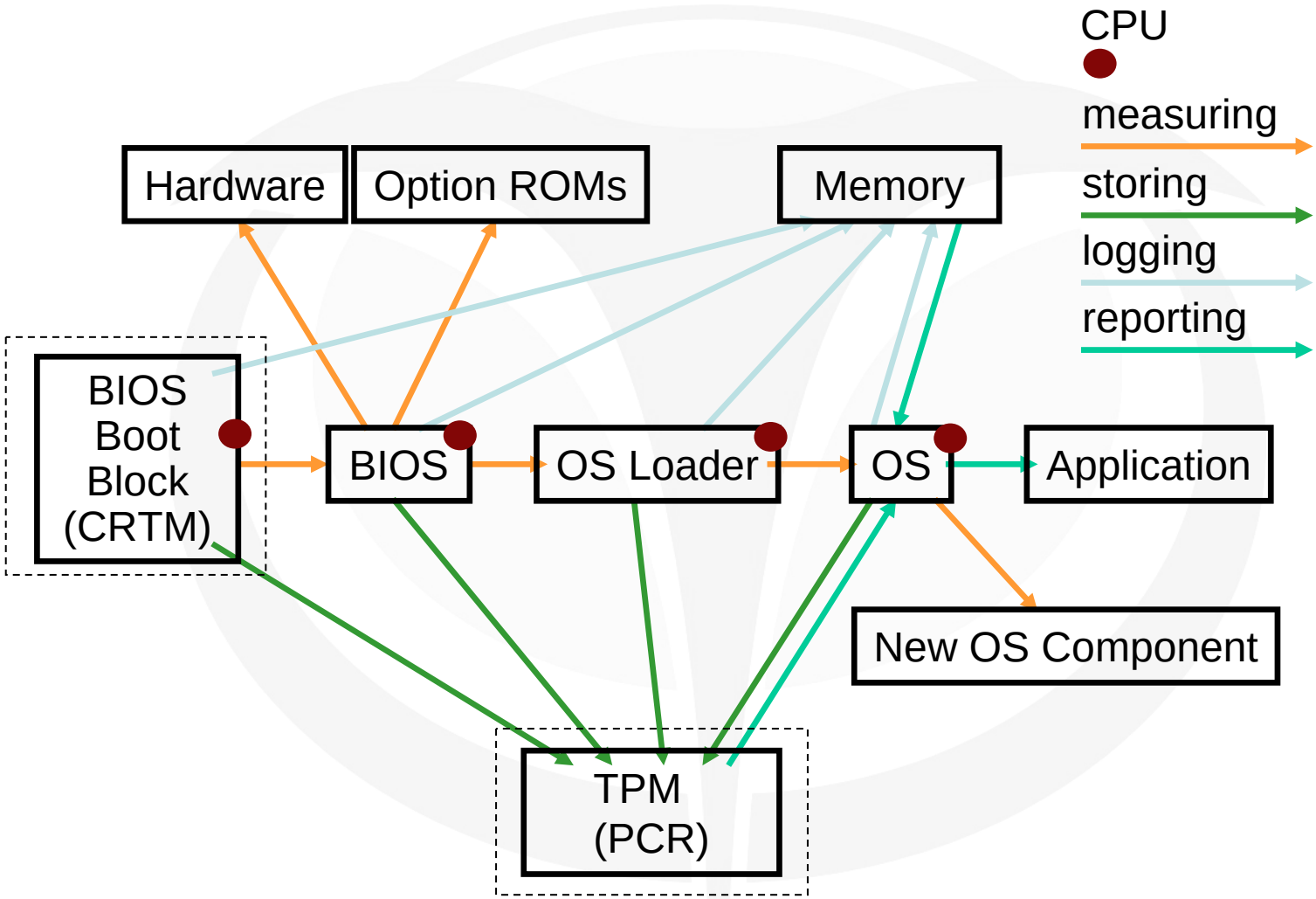
- Während des Booten werden die gemessenen System-konfigurationswerte in das TPM (Platform Configuration Register) geschrieben.
- Was kann gemessen werden?
  - BIOS, Hardwerekonfiguration, Loader, Trusted OS, Applications, usw.
- Die gespeicherten Werte können später für die Funktionen Sealing und Attestation verwendet werden.
- Das TPM misst nur die Laufzeitumgebung
  - Eine Kommunikationspartner kann anhand der Werte entscheiden, ob er der Laufzeitumgebung traut (Attestation)
  - Außerdem werden Daten an das Rechnersystem (TPM) gebunden (Sealing)

Trusted Computing Funktionalitäten
→ Trusted Boot (2/3)



- Beim Trusted Boot wird der Root of Trust gebildet, in dem sich die einzelnen Komponenten aufeinander aufbauend messen.
- CRTM (Core Root of Trust Measurement) muss „read only“ sein! (steht im EPROM oder im TPM)
- Jede Komponente wird zuerst gemessen bevor sie geladen wird.

Trusted Computing Funktionalitäten
→ Trusted Boot (3/3)



CRTM = Core Root of Trust Measurement, PCR = Platform Configuration Register

Trusted Computing Funktionalitäten
→ Sealing (Versiegelung)

- Bindung von Daten an die aktuelle Systemkonfiguration.
- Durch Bilden eines **Hash-Wertes** aus der Systemkonfiguration (Hard- und Software) können Daten an ein einziges TPM gebunden werden.
- Hierbei werden die entsprechenden Daten und die **Hash-Werte** aus den PCRs (Systemkonfiguration) zusammen verschlüsselt.
- Eine Entschlüsselung gelingt nur, wenn der gleiche **Hash-Wert** wieder ermittelt wird, was nur auf dem gleichen und unverändertem Rechnersystem mit dem entsprechenden TPM gelingen kann.
- Bei Defekt des TPM muss die Anwendung, die Sealing-Funktionen nutzt, dafür sorgen, dass die Daten nicht verloren sind.
- Auf diese Weise wird sichergestellt, dass auf versiegelte Daten nur wieder zugegriffen werden kann, wenn das Rechnersystem sich in einem bekannten Zustand (Systemkonfiguration) befindet.

Trusted Computing Funktionalitäten
→ Sealing (Berechnung - Beispiel)

- **Eingabe Parameter**
  - daten — {unverschlüsselte Daten}
- **Ausgabe Parameter**
  - cipher — {verschlüsselte Daten}
  - cryptedKEY — {verschlüsselter Schlüssel}
- **TPM Interne Funktionen und Daten**
  - encrypt (key, daten) — {AES, DES}
  - hash (daten) — {SHA-1, 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 // hash (daten // PCR-0 // … // PCR-x)))

cryptedKEY = encrypt (SRK, plainKEY // hash (plainKEY))

Trusted Computing Funktionalitäten
→ Unsealing (Berechnung)

- **Eingabe Parameter**
  - cipher {verschlüsselte Daten}
  - cryptedKEY {verschlüsselter Schlüssel}
- **Ausgabe Parameter**
  - daten {unverschlüsselte Daten}
- **TPM Interne Funktionen und Daten**
  - decrypt (key, daten) {AES, DES}
  - hash (daten) {SHA-1, SHA-256}
  - checkPCRs (Hash-Value) {vergleicht den Inhalt der PCRs mit den übergebenen Hash-Value}
  - SRK {Storage Root Key}
  - PCRs {PCR-0, PCR-1, …}

```
plainKEY = decrypt ( SRK, cryptedKEY )
daten // hash (…) = decrypt ( plainKEY, cipher )
if checkPCRs ( Hash-Value )
    return daten
else
    return ERROR
```

Trusted Computing Funktionalitäten
→ Binding (Auslagerung)

- Das **TPM** kann **Schlüssel auch außerhalb des Trust Storage (im TPM),** z.B. auf der Festplatte speichern.
- Diese werden ebenfalls in einem Schlüssel-Baum organisiert und deren Wurzel mit einem Key im **TPM** verschlüsselt.
- Somit ist die Anzahl der sicher gespeicherten Schlüssel nahezu unbegrenzt.

Trusted Computing Funktionalitäten
→ Schutz kryptografischer Schlüssel

- Schlüssel werden innerhalb des **TPMs** erzeugt, benutzt und sicher abgelegt.
- Sie müssen dieses also nie unverschlüsselt verlassen.
- Dadurch sind sie vor Software-Angriffen geschützt.
- Vor Hardware-Angriffen besteht ebenfalls ein relativ hoher Schutz (Sicherheit ist mit Smartcards vergleichbar).
  - Das **Active Shield** schützt die Hardware vor Angriffen
    - Sensoren (z.B. Temperatur)
    - Pseudo-Verbraucher
    - Random Waitstats
- Ein **TPM** kann durch das **Active Shield** physische Manipulation erkennen und die Daten unweigerlich Zerstören.

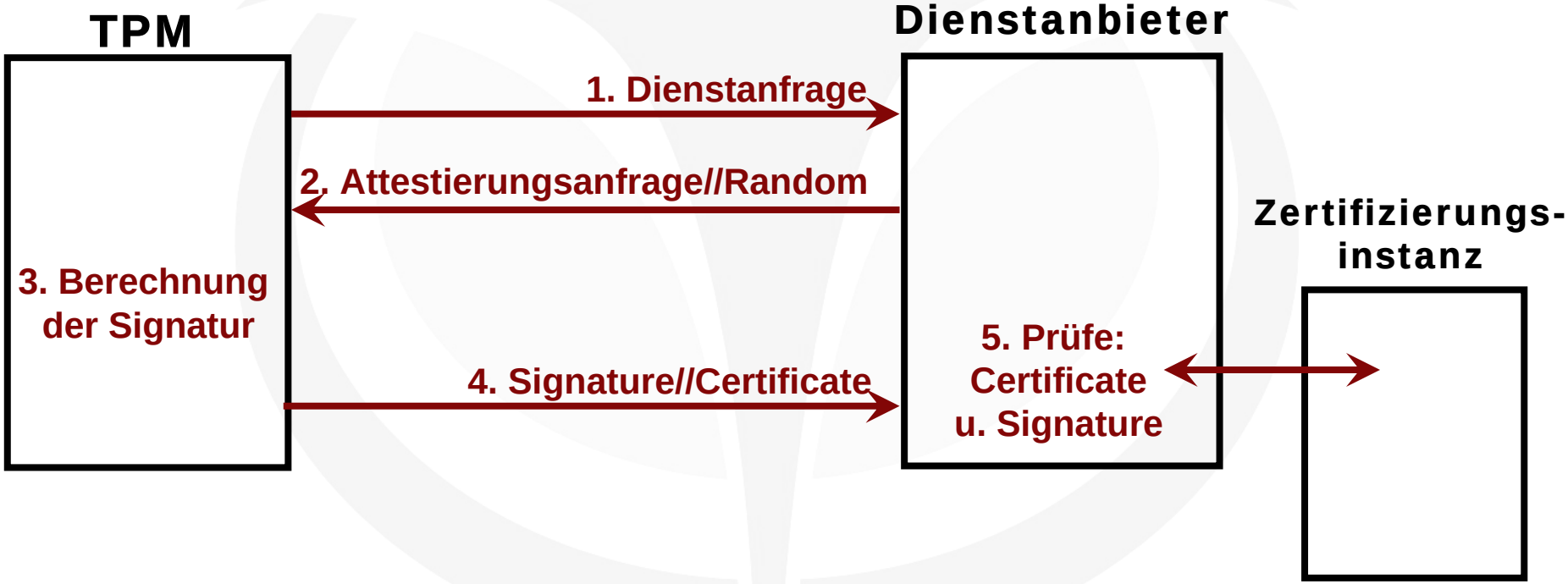
Trusted Computing Funktionalitäten
→ Remote Attestation (Beglaubigung) 1/2

- Remote Attestation ist die Betätigung der Integrität der Systemkonfiguration eines Rechnersystems gegenüber externen Kommunikationspartnern.
- Durch Attestation kann eine entfernte Partei davon überzeugt werden, dass die **Trusted Computing Plattform** über bestimmte Fähigkeiten verfügt und dass sich das Rechnersystem in einem wohldefinierten Zustand (entsprechende **PCR-Werte**) befindet.
- Diese **TPM-Funktionalität** hat erhebliche Auswirkungen auf die Privatsphäre eines Nutzers, weshalb zur Beglaubigung der Plattformkonformität (also Fähigkeiten und Zustand) niemals der **EK** direkt verwendet wird, sondern möglichst nur ein neu erzeugter **AIK**.
- Ferner sollte eine Attestation immer auch die explizite Zustimmung des **TPM-Eigentümers** erfordern .
- Zurzeit sind zwei verschiedene Attestationsverfahren vorgesehen:
  - Privacy CA (Trusted-Third-Party)
  - Direct Anonymous Attestation

Trusted Computing Funktionalitäten
→ Remote Attestation (Beglaubigung) 2/2

- Die ursprünglich vorgeschlagene Lösung (TPM-Spezifikation Version 1.1) benötigt eine vertrauenswürdige dritte Partei Zertifizierungsinstanz, PKI).
- Diese Privacy CA (Vertrauenswürdige Zertifizierungsinstanz) signiert alle neu erzeugten **AIKs,** sofern die „Plattform“ bestimmte festgelegte Richtlinien erfüllt, z.B. nachgewiesen durch gültige Zertifikate (EK Credential, TCPA Conformity Certificate, Platform Credential).
- Die Nachteile liegen in der notwendigen Verfügbarkeit der PKI und dem zentralen Angriffspunkt hinsichtlich der Privatsphäre der Nutzer.

Trusted Computing Funktionalitäten
→ Remote Attestation – Ablauf (1/2)



Trusted Computing Funktionalitäten
→ Attestierung: Berechnung - Beispiel

- **Eingabe Parameter**
  - **random** — **{Zufallszahl des Testes}**
- **Ausgabe Parameter**
  - **signature//certificate** — **{Signatur der Systemkonfiguration und Zertifikat des AIKs}**
- **TPM Interne Funktionen und Daten**
  - **sign (key, daten)** — **{RSA-Signatur}**
  - **hash (daten)** — **{SHA-1, SHA-256}**
  - **P-AIK** — **{geheimer AIK-RSA-Schlüssel}**
  - **AIK-certificate** — **{Zertifikat des AIKs}**
  - **PCRs** — **{PCR-0, PCR-1, …}** **z.B. aktuell abgespeicherte PCR-Werte**

signature = sign (P-AIK, hash (random // PCR-0 // … // PCR-x))

Trusted Computing Funktionalitäten
→ Attestierung: Verifikation - Beispiel

- **Eingabe Parameter**
  - signature//certificate {Signatur des Systemkonfiguration und Zertifikat des AIKs}
- **Ausgabe Parameter**
  - return value {Rückgabewert}
- **TPM Interne Funktionen und Daten**
  - very (key, daten) {RSA-Signatur-Verifikation}
  - hash (daten) {SHA-1, SHA-256}
  - Ö-AIK {öffentlicher AIK-RSA-Schlüssel}
  - checkPCRs (PCR-Values) {vergleicht den Inhalt der PCRs mit den gewünschten Werten}
  - PCRs {PCR-0, PCR-1, …}

```
if very ( Ö-AIK, signature ) and
if checkCERT ( certificate ) and if checkPCRs ( Hash-Value )
    return ok
else
    return ERROR
```

Trusted Computing Funktionalitäten
→ Direct Anonymous Attestation (DAA) 1/2

- Deshalb wurde mit der TPM-Spezifikation Version 1.2 eine als Direct Anonymous Attestation (**DAA**) bezeichnete Technik eingeführt.
- Durch ein komplexes kryptographisches Verfahren (spezielles Gruppensignaturschema) kann man die vertrauenswürdige dritte Partei einsparen und die Beglaubigung direkt zwischen den Beteiligten durchführen.
- Ein wichtiger Baustein dieser Technik bildet sogenannte Zero Knowledge Protokolle.
- Sie zeigen einem Diensteanbieter die Gültigkeit eines erzeugten **AIK**, ohne dass dabei Wissen über den korrespondierenden **EK** preisgegeben wird.

Trusted Computing Funktionalitäten
→ Direct Anonymous Attestation (DAA) 2/2

- Vergleichbar ist das Prinzip mit der Lösung eines Rubiks Würfels: Er geht davon aus, dass man einem Betrachter zunächst den ungeordneten und später den geordneten Würfel zeigt.
- So kann man einem Dritten jederzeit klarmachen, den Lösungsweg zu kennen, ohne diesen Weg erläutern zu müssen.
- Allerdings existieren auch bei **DAA** Einschränkungen bzgl. der gewährten **Anonymität**: Beispielsweise gibt es einen bestimmten Betriebsmodus (Named-Base Pseudonym, Rogue Tagging), der auf Wunsch des Diensteanbieter das Erkennen einer wiederholten bzw. missbräuchlichen Nutzung erlaubt.
- Damit ist eine Verkettung der durchgeführten Dienstanforderungen möglich, was natürlich die Anonymität einschränkt.

Trusted Computing Funktionalitäten
→ Removable Endorsement Key

- Mit der TPM-Spezifikation 1.2 hat es die **TCG** als Zugeständnis den **TPM-Herstellern** erlaubt, einen überschreibbaren **EK** in den Chip zu integrieren.
- Der Besitzer kann so den vorgegebenen Schlüssel löschen und für ungültig erklären.
- Dadurch verliert er allerdings unwiederbringlich den Zugang zum ursprünglichen Vertrauenssystem und muss ein eigenes System von Vertrauensstellungen aufbauen.

Agenda

- Virtualisierung
  - Definition
  - Softwarevirtualisierung
  - Hardwarevirtualisierung

Virtualisierung
→ Definition

- Virtualisierung bezeichnet Methoden, die es erlauben, Ressourcen eines Rechnersystems aufzuteilen.
  - Primäres Ziel ist es, eine Abstraktionsschicht zur Verfügung zu stellen, die von der eigentlichen Hardware (Rechenleistung, Speicherplatz) isoliert ist.
  - Eine logische Schicht wird zwischen Anwender und Ressource eingeführt, um die physischen Gegebenheiten der Hardware zu verstecken.
  - Mit dieser Technik werden Applikationen streng voneinander isoliert, sodass keine Applikation auf Ressourcen von andern Applikationen zugreifen kann.

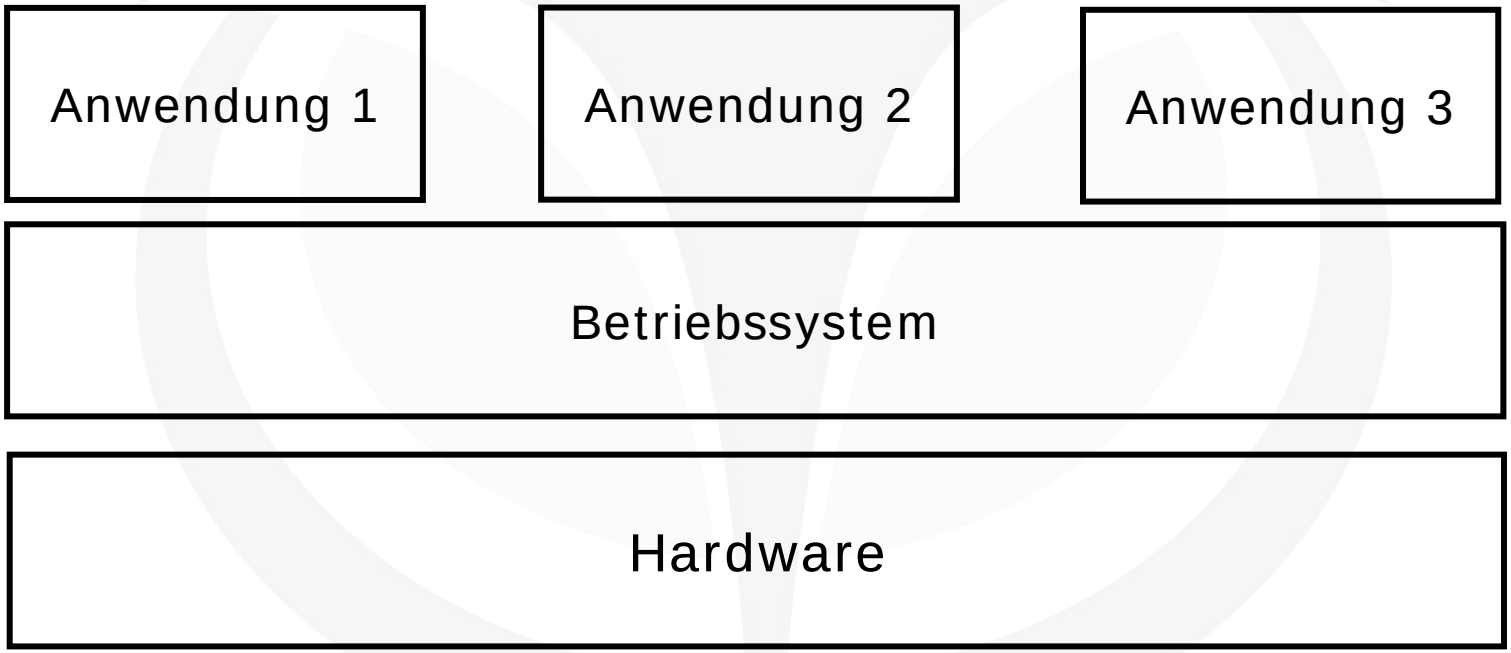
Virtualisierung
→ Softwarevirtualisierung (1/7)

- Die Softwarevirtualisierung kann für mehrere Zwecke eingesetzt werden.
  - Zur Simulation eines Betriebssystems
  - Zur Simulation einer Anwendung

- Es werden zwei Arten der Softwarevirtualisierung unterschieden:
  - Betriebssystemvirtualisierung mittels OS-Container
  - Systemvirtualisierung mittels **Virtual Machine Monitor (VMM)**

Virtualisierung
→ Softwarevirtualisierung

Herkömmliche Systeme ohne Virtualisierung



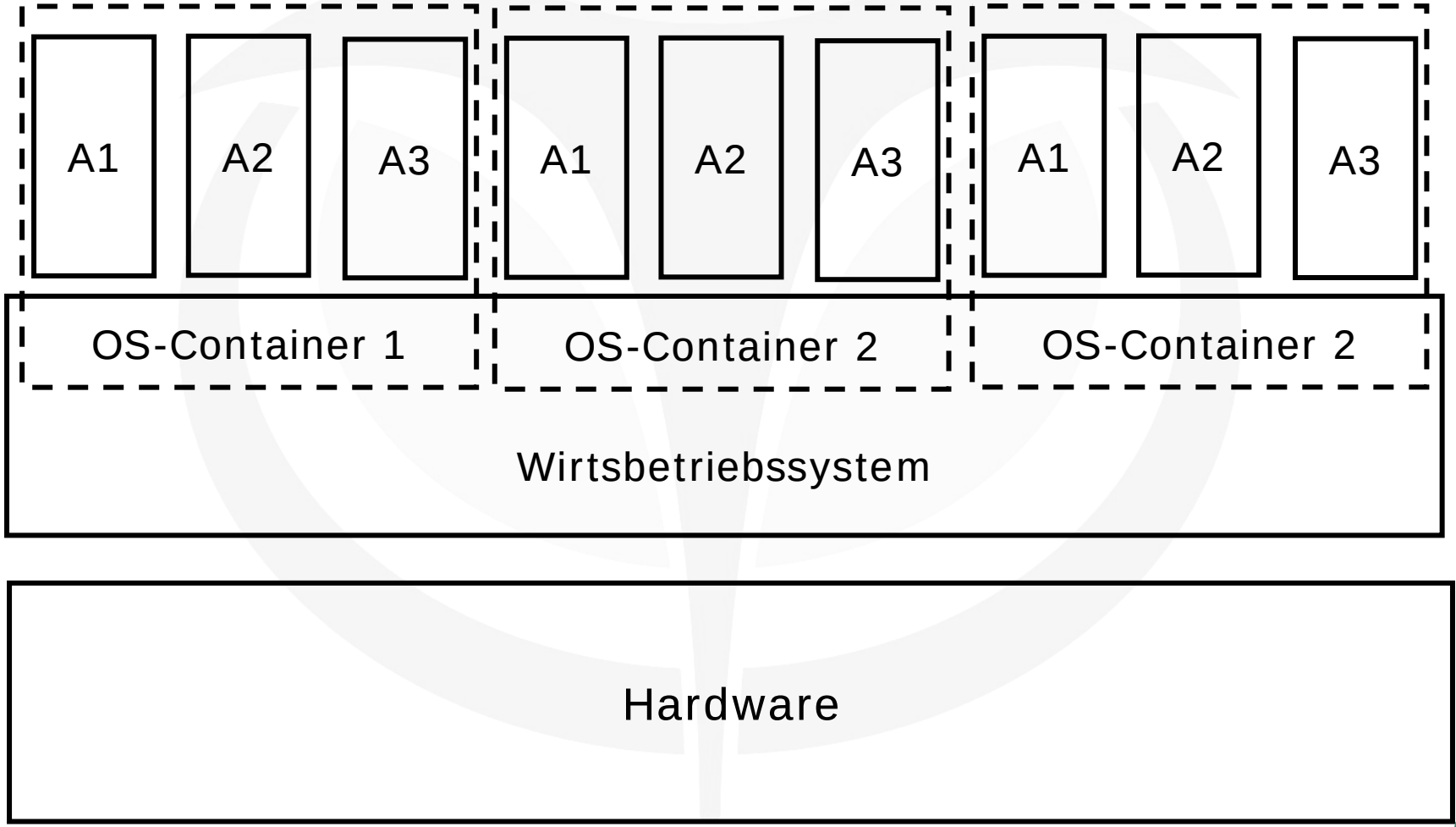
Virtualisierung
→ Softwarevirtualisierung (2/7)

Betriebssystemvirtualisierung mittels OS-Container

- Anwendungen bekommen eine komplette Laufzeitumgebung virtuell innerhalb eines geschlossenen Containers zur Verfügung gestellt.
- Es wird kein zusätzliches Betriebssystem gestartet.
- Die OS-Container stellen eine Teilmenge des Wirtbetriebssystems dar.
- Vorteil dieses Konzepts liegt in der guten Integration der Container in das Gastbetriebssystem.
- Der Nachteil dieses Konzepts liegt in den Containern, da aus den Containern heraus können keine Treiber bzw. andere Kernel geladen werden können.
- Bei der OS-Virtualisierung läuft immer nur ein Kernel.
- Es liegt dabei keine starke Isolation der einzelnen OS-Virtualisierungen vor.
- Es müssen Änderungen am Wirtbetriebssystems vorgenommen werden.

Virtualisierung
→ Softwarevirtualisierung

Betriebssystemvirtualisierung mittels OS-Container



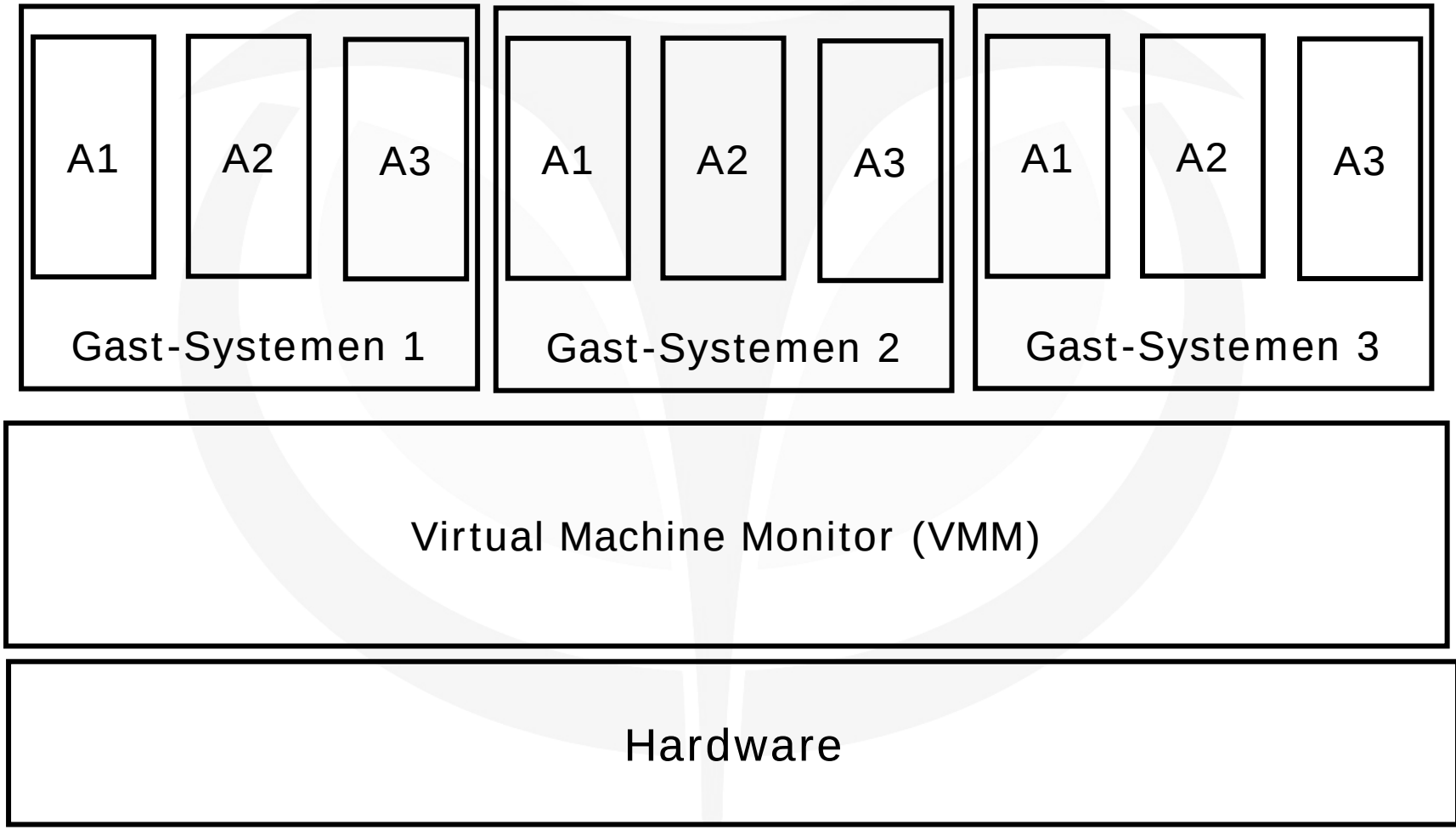
Virtualisierung
→ Softwarevirtualisierung (3/7)

Systemvirtualisierung mittels Virtual Machine Monitor (VMM)

- Bei Virtualisierung mittels eines VMM, werden die bereitstehenden nativen Ressourcen intelligent verteilt.
- Es werden drei mögliche Verfahren unterschieden:
  - **Hardware Emulation**
  - **Hardware Virtualisierung**
  - **Paravirtualisierung**
- Den einzelnen Gast-Systemen wird dabei jeweils ein eigener kompletter Rechner mit allen Hardware-Elementen vorgegaukelt. Wie zum Beispiel:
  - Prozessor
  - Laufwerke
  - Arbeitsspeicher
- Der Vorteil ist, dass an den Betriebssystemen selbst fast keine Änderungen erforderlich sind.
- Die Gast-Systeme laufen alle unter ihrem eigenen Kernel, was eine gewisse Flexibilität und Isolation im Ggs. zur Betriebssystemvirtualisierung mit sich bringt.

Virtualisierung
→ Softwarevirtualisierung

Betriebssystemvirtualisierung mittels OS-Container



Virtualisierung
→ Softwarevirtualisierung (4/7)

VMM → Hardware Emulation

- Die „Virtuelle Maschine“ simuliert die komplette Hardware und ermöglicht einem nichtmodifizierten Betriebssystem, das für eine andere CPU ausgelegt ist, den Betrieb.

- Beispielanwendungen:
  - Bochs ist ein freier x86-Emulator.
  - Viele Betriebssysteme, wie etwa Windows oder Linux, können unter Bochs betrieben werden.
  - Ebenso ist Bochs für viele verschiedene Betriebssysteme erhältlich.
  - Microsoft Virtual PC wird als Virtualisierungssoftware für Windows wie auch als x86-Emulator für Mac OS X angeboten.
  - Es ist Bestandteil des Produktes Microsoft Office Professional für Mac OS X.

Virtualisierung
→ Softwarevirtualisierung (5/7)

VMM → Hardware Virtualisierung

- Die Virtuelle Maschine stellt dem Gastbetriebssystem nur Teilbereiche der physischen Hardware in Form von virtueller Hardware zur Verfügung.
- Diese reicht jedoch aus, um ein unverändertes Betriebssystem darauf in einer isolierten Umgebung laufen zu lassen.
- Das Gast-System muss hierbei für den gleichen CPU-Typ ausgelegt sein wie das Host-System.

- Beispielanwendungen:
  - Mit VMware lassen sich mehrere Maschinen mit verschiedenen Betriebssystemen gleichzeitig virtualisieren.
  - Die virtualisierten Betriebssysteme sind in Abhängigkeit vom Speicherausbau etwas langsamer als vergleichbare Installationen auf identischer Hardware.

Virtualisierung
→ Softwarevirtualisierung (6 / 7)

VMM → Paravirtualisierung

- Bei Paravirtualisierung wird ein zusätzliches Betriebssystem virtuell neu gestartet.
- Es wird jedoch keine Hardware virtualisiert oder emuliert, sondern die virtuell gestarteten Betriebssysteme verwenden eine abstrakte Verwaltungsschicht, um auf gemeinsame Ressourcen zuzugreifen.
  - Netzanbindung
  - Festplattenspeicher
  - Benutzerein/-ausgaben)

Virtualisierung
→ Softwarevirtualisierung (7 / 7)

VMM → Paravirtualisierung Beispielanwendungen

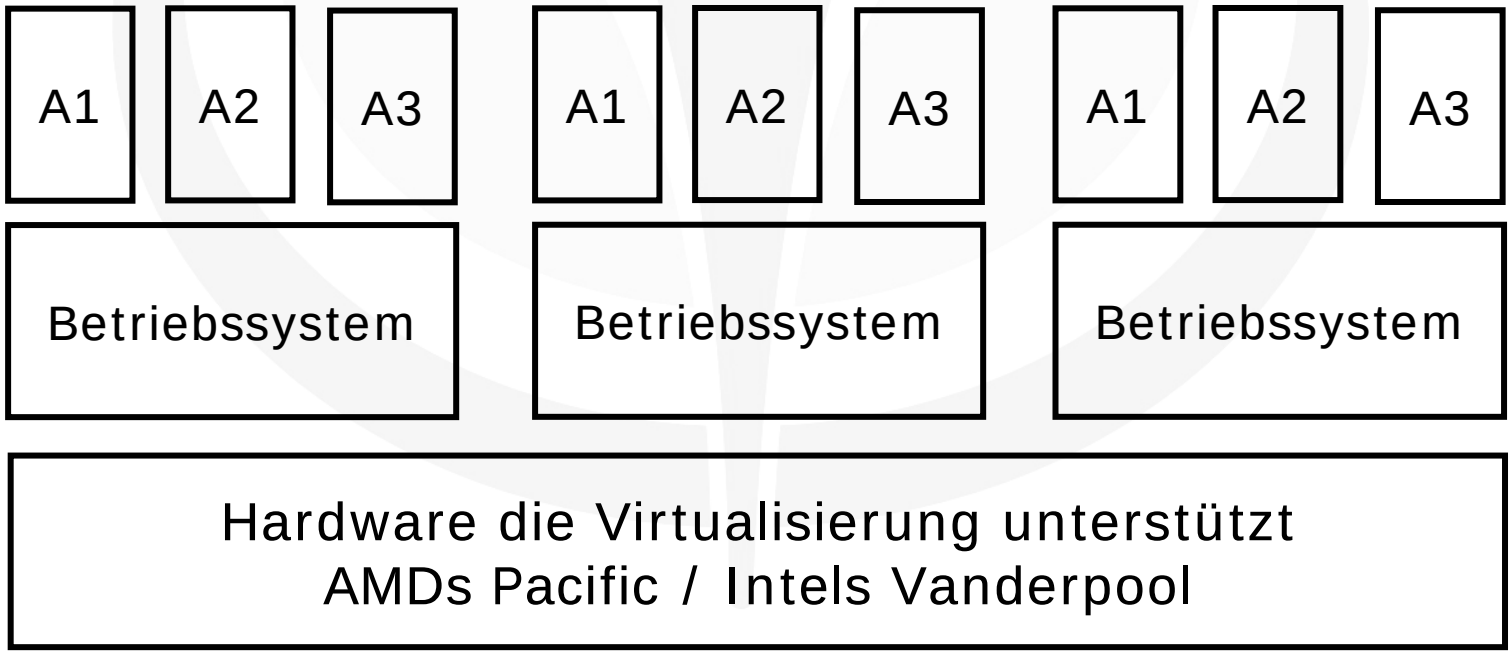
- **Turaya**
  - Vertrauenswürdige faire und offene Sicherheitsplattform
- **Xen**
  - Xen läuft direkt auf der x86-Hardware.
  - Die Hardware wird für die darauf laufenden Systeme (Domains) paravirtualisiert.
  - Es wird eine sehr hohe Performance erzielt, da die Hardware nicht emuliert wird, sondern diese den Gastsystemen mit einem sehr kleinen Overhead zur Verfügung gestellt wird.
  - Die Domains können unter anderem Linux, NetBSD und Windows sein.
  - Microsoft Windows wurde auch zum Laufen gebracht, jedoch darf die Modifizierung aus Lizenzgründen nicht freigegeben werden.

Virtualisierung
→ Hardwarevirtualisierung

- Für die Hardwarevirtualisierung wird eine neue Prozessorgeneration benötigt:
  - AMDs Pacific
  - Intels Vanderpool
- Diese neue Prozessorgeneration ist eine Implementierung einer Secure Virtual Machine (SVM)
  - Eine SVM ist eine besondere Virtuelle Maschine, die eine verbesserte Virtualisierung über Befehlssatzerweiterungen der Prozessorbefehle ermöglicht.
  - Die Erweiterungen von AMD und Intel sind nicht miteinander kompatibel, funktionieren jedoch nach dem gleichen Grundprinzip.
- Der Vorteil gegenüber Software-Lösungen ist die höhere Geschwindigkeit, die eine Hardware-Lösung bieten kann, da die Virtualisierung direkt in den I/O-Brücken der Computer eingebaut ist.
- Alle Betriebssysteme und somit auch die Anwendungen laufen vollkommen isoliert voneinander.

Virtualisierung
→ Hardwarevirtualisierung

- Es müssen keine Änderungen an den Betriebsystemen oder Anwendungen vorgenommen werden.
- Betriebssysteme und Anwendungen können unverändert parallel auf der Hardware, die Virtualisierung unterstützt, ausgeführt werden.
- Die Zugriffe auf die Hardware aus den Betriebssystemen heraus werden erkannt und automatisch abgefangen und entsprechend verarbeitet.



Agenda

- Betriebssysteme
  - Definition
  - Monolith
  - Microkern

Betriebssysteme
→ Definitionen 1/2

- **Betriebsmittel:**
  - CPUs, Speicher (physikalischer, virtueller)
- **Geräte:**
  - Eingabegeräte (Maus, Tastatur, . . .)
  - Ausgabegeräte (Bildschirm, Drucker, . . .)
  - Datenmedien (Festplatten, DVD, CD, Disketten, . . .)
  - Multimediageräte
  - Netzwerkgeräte
- API
- Tools

Betriebssysteme
→ Definitionen 2/2

- **Kern (engl. Kernel) stellt Funktionalität:**
  - monolithischer Ansatz
  - microkernscher Ansatz

- **Zugriff auf Funktionalität mittels Systemaufruf (Systemcall):**
  - Speicherverwaltung
  - Prozess/Task und Threads
  - Interprozesskommunikation (IPC)
  - Prozessumschaltung, Kontextwechsel
  - Benutzer- und System-Modus (User-, Kernel-Space)
  - Interrupt-Behandlung
  - Abstraktion und Virtualisierung

Betriebssysteme
→ Monolith 1/2

- **Ältere Monolithen**
  - Lange etabliert
  - Alle Treiber vereint im Kernel-Space
  - geringe Flexibilität
  - hohe Komplexität
  - geringe Portierbarkeit
  - wenig robust
  - schlechte Sicherheitsmechanismen
  - gute Performance

- **Beispiele:**
  - Ur-Systeme

Betriebssysteme
→ Monolith 2/2

- **neuere Monolithen**
  - Schichtprinzip
  - Alle Treiber vereint im Kernel-Space
  - mittlere Flexibilität
  - mittlere Komplexität
  - gut Portierbarkeit
  - weniger robust
  - schlechte Sicherheitsmechanismen
  - gute Performance

- **Beispiele:**
  - Linux
  - Solaris

Betriebssysteme
→ Microkern: Grundlage (1/2)

- Ende 80er
- Kapseln von Diensten in Servern (Flexibilität, Komplexität)
- Leichte Erweiterbarkeit (Hybride)
- Verteilte Systeme
- bessere Sicherheitsmerkmale (Verifizierung)
- Bereitstellung von Mechanismen (IPC)

Betriebssysteme
→ Microkern: Grundlage (2/2)

- Client-Server-Architektur bietet:
  - Single-Server
  - Multi-Server (horizontal, linear)
- Betriebssystemserver (OS personality, Hybrid)
- Plattformunabhängig
- User-Space (Prozess)
- IPC (syncron, asyncron)

Betriebssysteme
→ Microkern: erste Generation

- **Mach, 1989 Carnegie Mellon University (CMU)**
  - basiert auf 4.2BSD
  - 3. Release **Microkern**
  - viele Architekturen unterstützen
  - Netzwerktransparenz
  - Parallelismus bei Programmen und im System
  - größere Adressräume
  - Basis für die Entwicklung weiterer Betriebssysteme
  - langsam und zu groß

- **Besonderheiten:**
  - Port (Kommunikationskanal, Port Rights, Port Sets)
  - Nachrichten (auch indirekt per Zeiger)
  - Memory Objects (Speicher, Dateien, Pipes)
  - I/O System im Kern

Betriebssysteme
→ Microkern: zweite Generation

- **Der Microkern L4, Jochen Liedtke 1995:**
  - externer Pager
  - sehr kleiner Kern (< 12 KB)
  - 3 Abstraktionen
  - 7 Systemaufrufe
  - reintegrieren von Servern
  - leichte Zwischenprozesskommunikation (LIPC)
  - Clans und Chiefs (Kommunikationsgruppen, alt)
  - Flexpages

Betriebssysteme
→ Microkern: Erweiterungen

- Sigma0 Speicher-Dienst verwendet:
  - erster Prozess
  - alloziert gesamten Speicher
  - mapping 1:1
  - fungiert als Pager
- Omega0 Interrupt-Dienst
- L4Linux

Betriebssysteme
→ Microkern: Merkmale

- **Performance:**
  - Nachrichten (Overhead, langsam)
  - mehr Kontextwechsel
- **Konfiguration:**
  - Treiber
- **Flexibilität:**
  - Diensteintegration
- **Robustheit:**
  - Dienste neu starten
  - eigener Adressraum
- **Portierbarkeit:**
  - Server unabhängig
- **Sicherheit:**
  - pro Server

Betriebssysteme
→ Microkern: Zusammenfassung

- Auslagerung auf User-Space
- hohe Flexibilitat
- niedrige Komplexitat (mehr Vertrauenswürdigkeit)
- hohe Portierbarkeit der Dienste
- sehr robust
- verbesserte Sicherheitsmechanismen
- schlechte Performance (1. Generation)
- **mittlere bis gute Performance (2. Generation)**

Agenda

- Trusted Computing Base
  - Definition
  - Ziele
  - Messbarkeit
  - Dienste

Trusted Computing Base
→ Definition

- Unter der **Trusted Computing Base (TCB)** versteht man den Teil eines Hardware/ Software-Systems, auf dessen **korrektes**, d.h. spezifikationsgemäßes **Verhalten** man sich verlassen kann.
- Die **TCB** dient als **verlässliches Fundament**, um darauf weitere Komponenten aufzubauen.
- Die Idee stammt aus den siebziger Jahren und hat ihren Niederschlag auch in der Mikrokern-Architektur von Betriebssystemen gefunden.
- In den letzten Jahren lebte der Begriff wieder auf, da die Mikrokern-Architektur zusammen mit den Trusted Computing Funktionalitäten eine starke Synergie entwickeln, um **vertrauenswürdige Systeme** zu schaffen.

Trusted Computing Base
→ Definition

- Neben Microsoft existieren zurzeit viele Projekte, die auf Basis einer TCB eine vertrauenswürdige Sicherheitsplattform entwerfen.
  - **PERSEUS**
  - **OPENTC**
  - **EMSCB → Turaya**

- Das Thema TCB muss jedoch kontrovers betrachtet werden, da einerseits mit der TCB verlässliche Systeme geschaffen werden können, aber andererseits auch Einschränkung der Autonomie des Rechnersystemeigentümers verbunden sind.

Trusted Computing Base
→ Ziele

- Welche Ziele muss die **TCB** erfüllen, um vertrauenswürdige Systeme zu schaffen:
  - **Die TCB muss fehlerfrei sein.**
  - **Die TCB muss formal beweisbar sein.**

- Warum müssen diese Ziele verfolgt werden?
  - Existieren Fehler (z.B. Buffer Overflow) in der TCB kann dort Schadcode injiziert werden.
  - Somit verliert die TCB ihre Vertrauenswürdigkeit.
  - Mit Hilfe der formalen Beweisbarkeit wird eine Sicherheitsevaluation (Commen Criteria) auf hohem Niveau möglich.

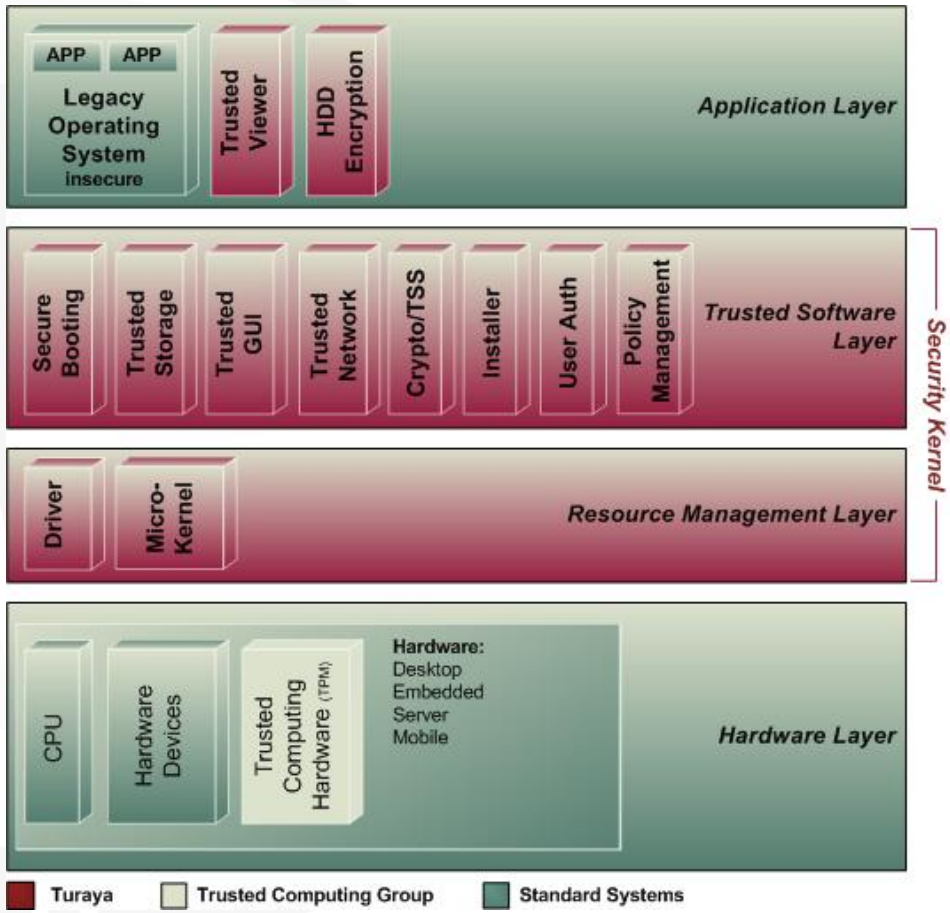
- Diese Ziele können erreicht werden, da die Komplexität der TCB klein gehalten wird.
  - Je weniger Lines of Code desto weniger Fehler

Trusted Computing Base
→ Messbarkeit

- Mit Hilfe der Trusted Computing Funktionalitäten ist man in der Lage, die **TCB** zu messen und eine Aussage darüber zu treffen, ob sich Teile der **TCB** verändert haben.
- Die Speicherung der Messwerte (Hash-Values) erfolgt in den **PCRs** eines **TPMs**.
- Die einzelne Teile der **TCB** werden auf die vorhandenen **PCRs** aufgeteilt, sodass einzelne Teile unabhängig bewertet werden.
- Die Messung der **TCB** kann beim Start des Systems erfolgen (Secure / Authenticate Boot) oder von einem Dienst (Loader) der durch die **TCB** bereitgestellt fortgesetzt werden.
- Dies wird benötigt, wenn nach dem Start des Systems, Teile der **TCB** nachgeladen werden.
- Nicht nur der Quellcode der **TCB** muss gemessen werden, sondern auch etwaige Konfigurationen der **TCB** müssen in die Messung eingehen, da die Konfiguration auch Einfluss auf die Vertrauenswürdigkeit des Systems haben.

Trusted Computing Base
→ Architektur im Detail

- ***Application Layer***
  - Herkömmliches Betriebssystem
  - Sichere Applikationen
- ***Trusted Software Layer***
  - Sicherheitsdienste
  - Applikationsmanagement
  - Sec. Policy Management
- ***Resource Management Layer***
  - Mikrokern / HW Sharing
  - Policy Enforcement
- ***Hardware Layer***
  - CPU
  - TC Technologie



Trusted Computing Base
→ Sichere Anwendungen

- **Trusted Viewer**
  - Stellt nach dem **What-you-see-is-what-you-get-Prinzip** einen vertrauenswürdigen Dokumenten-Viewer bereit.
  - Anwendungen können Dokumente so ablegen, dass nur der **Trusted Viewer** in der Lage ist, diese zu öffnen und darzustellen.
  - Ausgaben, die über den **Trusted Viewer** dargestellt werden, können nicht von anderen Anwendungen überlagert werden.

- **Device Encryption**
  - Mit der **Device Encryption** können blockorientierte Geräte (Festplatte, Memory Sticks, CD/DVD-Medien) verschlüsselt werden.
  - Die **Device Encryption** ist nach der Konfiguration transparent für den Anwender.

Trusted Computing Base
→ Dienste (1/3)

- **Trusted Storage (Manager)**
  - Der **Trusted Storage Manager** stellt einen vertrauenswürdigen Speicherbereich dar, an den sich Prozesse wenden können, um Daten sicher und integer abzulegen.
  - Es können Daten an die Konfiguration (**PCRs**), an einen User, oder an eine Applikation gebunden werden.
  - Der **Trusted Storage Manager** stellt außerdem die Eigenschaft „freshness“ bereit.
  - Das bedeutet, dass Replay-Angriffe erkannt und verhindert werden können.

- **Trusted GUI**
  - Verwaltet die Anwender-Input und -Output Geräte (Maus, Tastatur, Grafikkarte, …).
  - Stellt einen sicheren Pfad (**trusted path**) von der Tastatur-Eingabe bis in die Anwendung sicher, sodass keine Eingaben umgeleitet oder abgefangen werden können.

Trusted Computing Base
→ Dienste (2/3)

- **Trusted Network**
  - Dieser Dienst wird auch als **Trusted Network Connect (TNC)** bezeichnet.
  - Stellt ein vertrauenswürdiges Interface zur Verfügung, dass Netzwerkkomponenten prüft und gegebenenfalls die Verbindung verhindert.
- **Crypto/TSS**
  - Er bildet die zentrale Anlaufstelle für alle Anwendungen, die Funktionen aus der TSS benötigen.
- **Installer**
  - Stellt den Loader des Rechnersystems dar.
  - Er installiert und startet Dienste aus der TCB oder auch Anwendungen der User.
  - Verwaltet alle laufenden Prozesse und liefert eine vertrauenswürdige Instanz, um Prozesse zu identifizieren.

Trusted Computing Base
→ Dienste (3/3)

- **User Auth**
  - Bildet die Userverwaltung für die Anwender des Rechnersystems und stellt diesen Dienst anderen Anwendungen zur Verfügung.
  - Anwendungen können mit diesem Dienst eine User-Authentifikation anstoßen.
  - Es wird ermöglicht, dass Daten an einen User gebunden werden

- **Policy Management**
  - Dieser Dienst sorgt dafür, dass Regeln durchgesetzt werden.
  - Daten, die nur nach einer bestimmten Regel verarbeitet werden dürfen, sind an den **Policy Management** verschlüsselt gebunden.
  - Bevor diese Daten verarbeitet werden können, wird die Regel von **Policy Management** geprüft.

Konsortium EMSCB
→ Partner der deutschen Spitzenforschung

Konsortialführer

Ruhr-Universität-Bochum
eurobits

TECHNISCHE UNIVERSITÄT DRESDEN
Institut für Systemarchitektur

Fachhochschule Gelsenkirchen
Institut für Internet-Sicherheit

gefördert durch das
Bundesministerium für Wirtschaft und Technologie

Sirrix AG security technologies

escrypt Embedded Security

Die strategischen Industriepartner:

SAP

BLAUPUNKT Bosch Gruppe

infineon

Ziele
→ Technische Ziele

- ***Faires Policy Enforcement*** auf eigenen und entfernten Rechnersystemen
  - Enterprise Rights Management
- ***Faire Digital Rights Management*** Funktionalitäten schaffen
- ***Fünf Demonstratoren***
  - Turaya.Crypt → fertiggestellt
  - Turaya.VPN → fertiggestellt
  - Turaya (z.B. FairDRM) → Dezember 2006
  - Turaya.ERM → Ende 2007 - Partner SAP
  - Turaya (z.B. embsys) → Ende 2007 – Partner Bosch/Blaupunkt

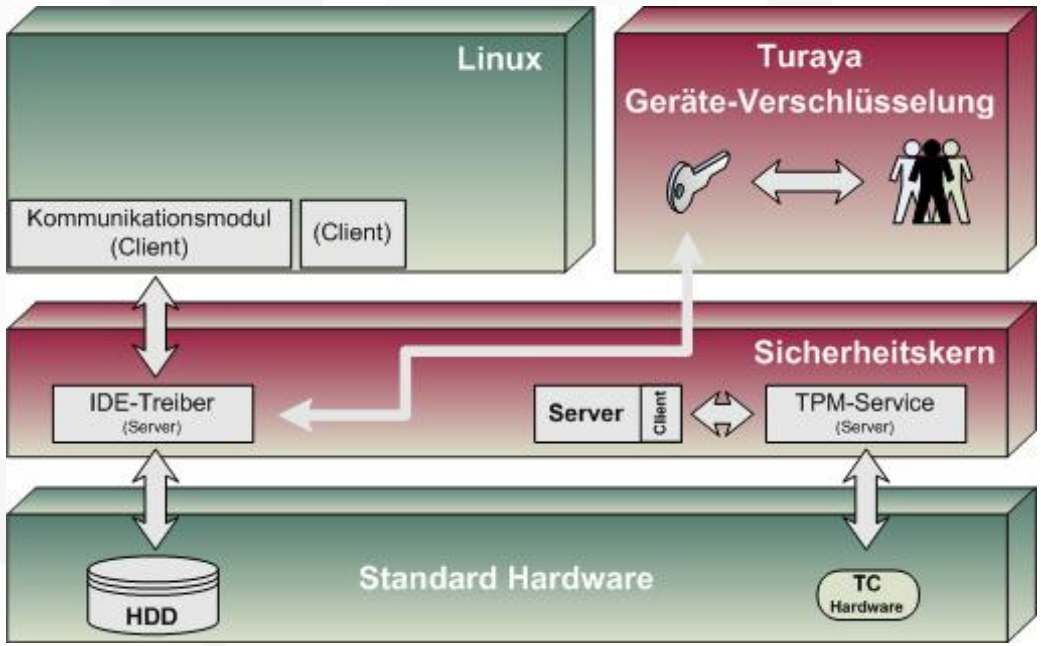
Agenda

- Anwendungsbeispiele
  - Turaya.Crypt
  - Turaya.VPN
  - Dokumentenmanagement
  - Verbesserungen von Homebanking

Architektur und Technologie
→ Turaya.Crypt

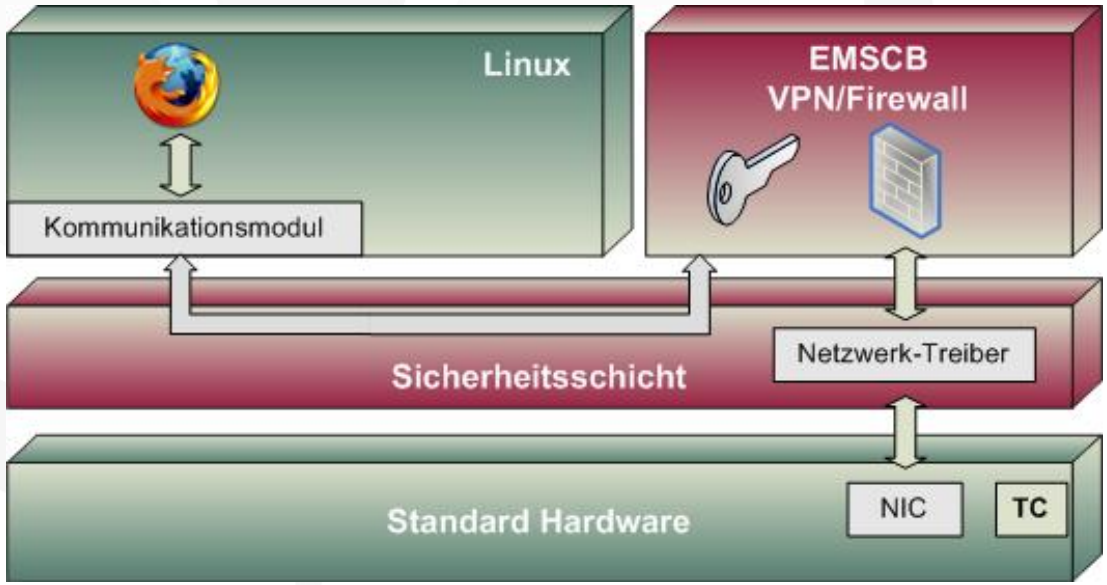
- ***Datentransfer*** zwischen Linux und **ausgelagertem IDE-Treiber**
- ***IDE-Treiber*** kommuniziert mit Verschlüsselungssystem
- ***Benutzerauthentifikation***, kryptographische Schlüssel und Operationen **isoliert von Linux**
- ***Verschlüsselung transparent*** für Nutzer und Standard-Betriebssystem

- ***Unterstützte Geräte***
  - Festplatten
  - USB Memory Sticks
  - CDR/DVD-Medien



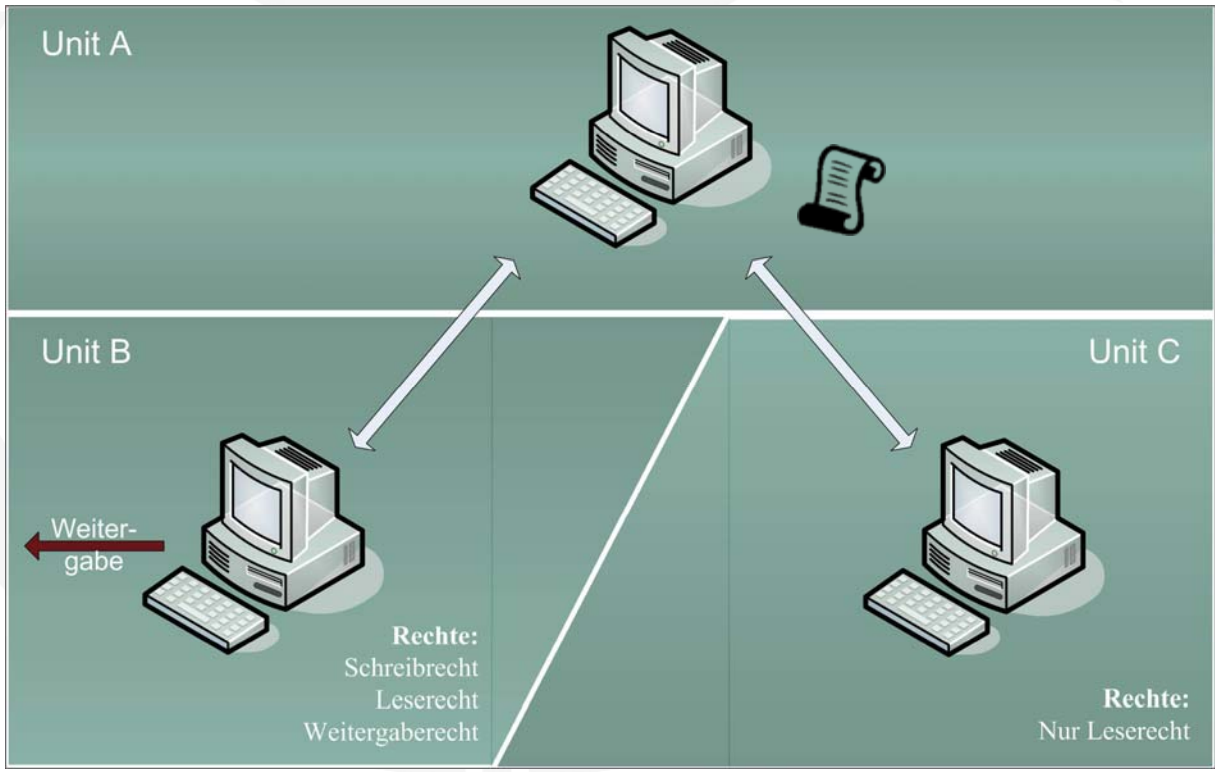
Architektur und Technologie
→ Turaya.VPN

- ***Isoliert vom Standard-Betriebssystem:***
  - Netzwerk-Treiber
  - VPN-Client und Zertifikate/Schlüssel
  - Firewall und Firewall-Policy
- ***Verschlüsselung transparent für Nutzer und Standard-Betriebssystem***



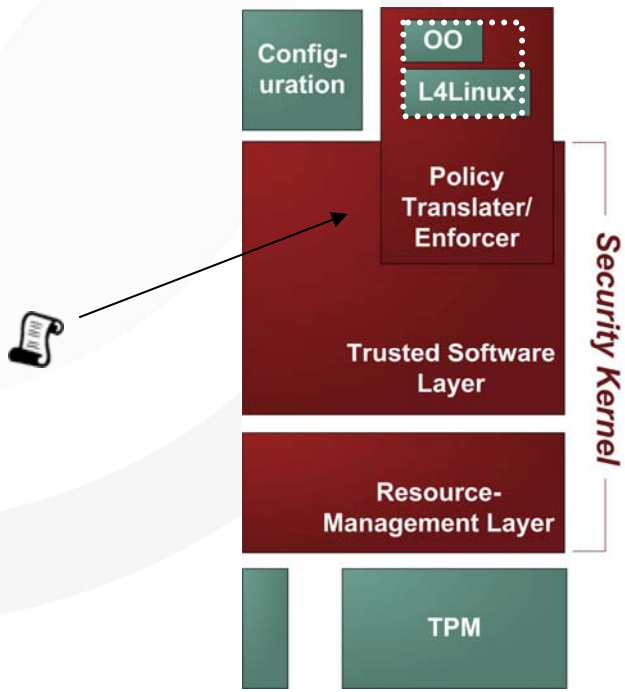
Beispielanwendung
→ Ablauf des Dokumentenmanagements (1/3)

- Ein Dokument wird mit entsprechenden Rechten verteilt.



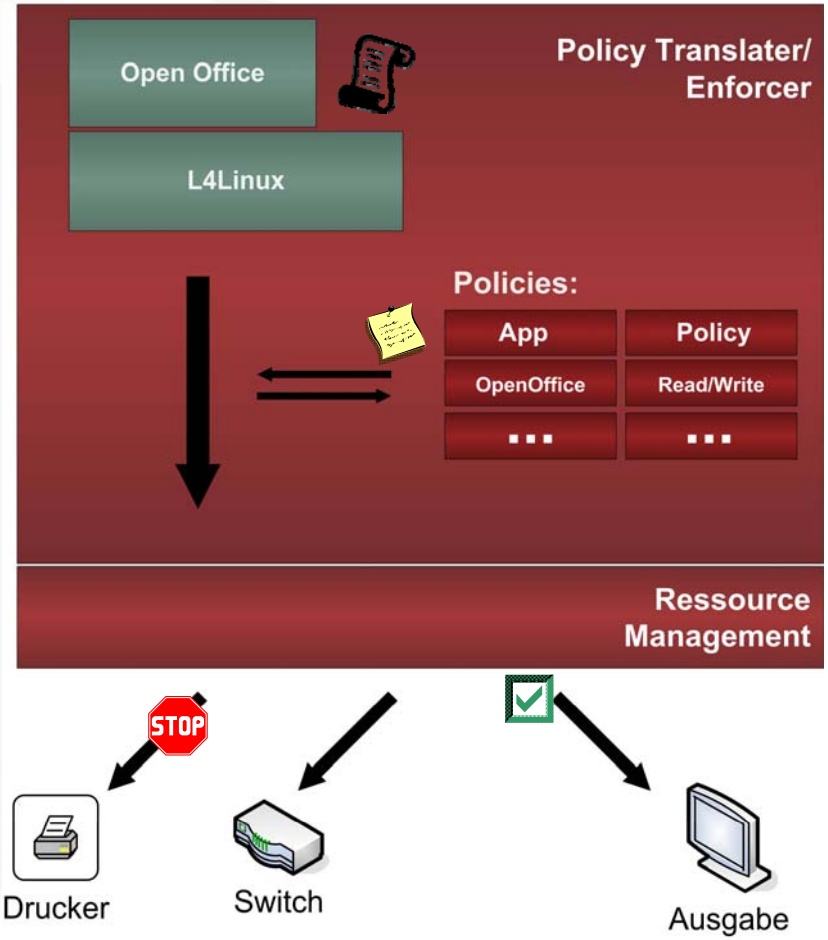
Beispielanwendung
→ Ablauf des Dokumentenmanagements (2/3)

1. Es wird festgestellt, dass ein sicheres Dokument angenommen werden soll.
2. Die OpenOffice Applikation auf L4L-Basis wird gestartet!
3. Das Dokument wird erst freigegeben, wenn Open Office gestartet ist.
4. Sind alle Kriterien erfüllt, kann das Angebot angezeigt werden.



Beispielanwendung
→ Ablauf des Dokumentenmanagements (3/3)

1. Nutzer hat Leserechte, jedoch kein Recht auf Weitergabe des Dokuments.
2. Der Policy Translator/Enforcer sorgt für die Einhaltung dieser Rechte → Kontrollstation.
3. Die Policys werden beispielsweise in einer Liste gehalten.

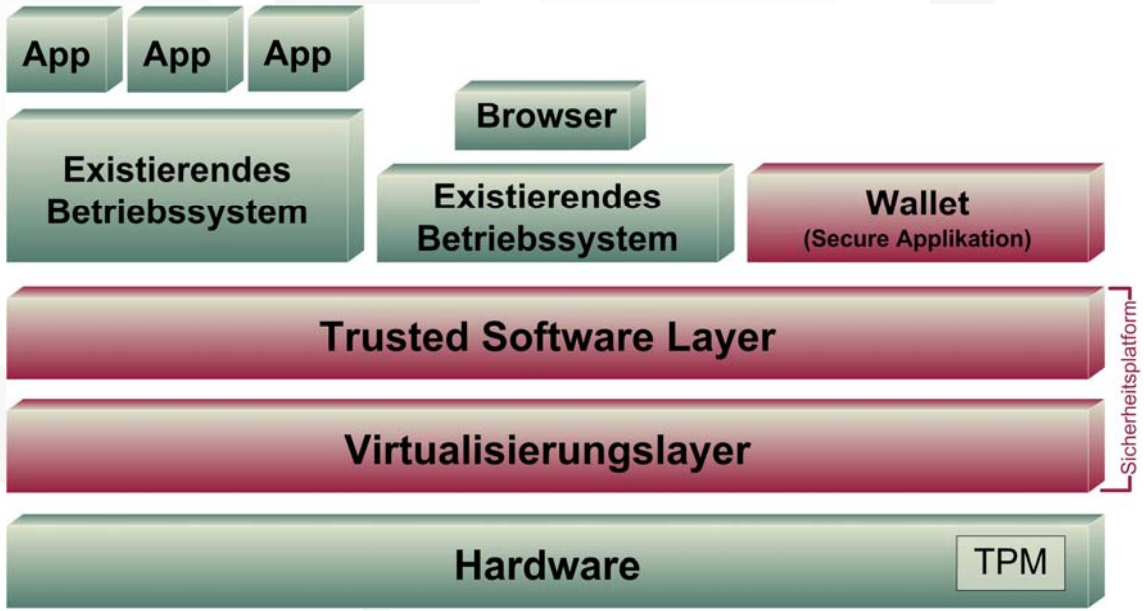


Verbesserungen von Homebanking
→ Basis Idee mit Turaya

- **Sichere Basis Turaya**
  - Schutz sensitiver Informationen
    - "Binden“ an vertrauenswürdige Systemkonfiguration
  - Isolation des Webbrowsers / der Banking-Software
    - Schutz vor Malware, Konfigurationsfehlern
  - Sicher durch den Einsatz von Trusted Computing Technologie
  - Externe Überprüfung der Anwenderplattform möglich (Attestation)
- **Wallet: Automatisierung kritischer Vorgänge**
  - Authentifizierung des Bankservers
  - Authentifizierung des Anwenders

Verbesserungen von Homebanking
→ Funktionsweise von Turaya

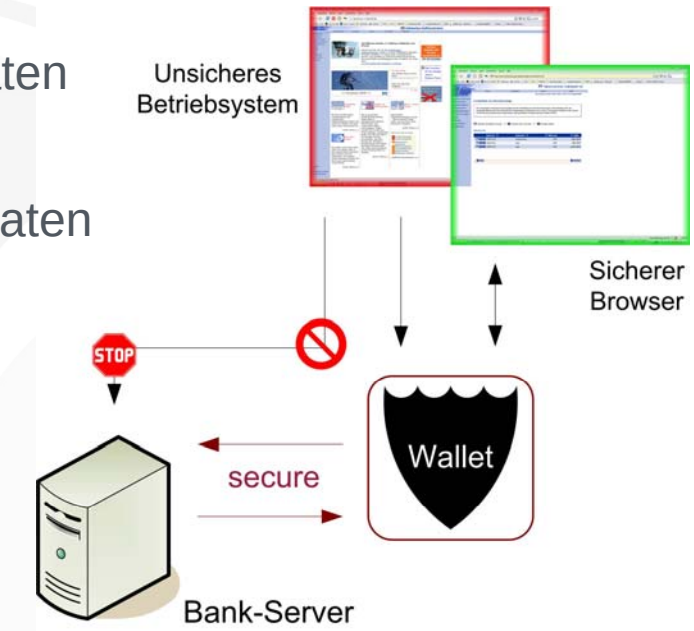
- **Sicherheitskern**
  - Authentifikation einzelner Compartments (lokal & remote)
  - Binden von Daten an einzelne Compartments
  - Trusted Path
  - Sicheres Benutzerinterface



Verbesserungen von Homebanking
→ Funktion Wallet

Wallet-Funktionen

- Einmalige Eingabe der Authentifizierungsdaten
  - durch Nutzer oder Bank
- Lokale Speicherung der Authentifizierungsdaten
- Authentifiziert den Bankserver
- sendet die Authentifizierungsdaten des Anwenders



Vorteile

- Benutzung herkömmlicher Webbrowser
- Webbrowser hat keinen Zugriff auf sensitive Daten
- Anwender muss den Bankserver nicht selbst authentifizieren
- Anwender muss sich die Authentifizierungsdaten nicht merken
- Schnell und kostengünstig umsetzbar

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?

Prof. Dr. Norbert Pohlmann

Institut für Internet-Sicherheit
Fachhochschule Gelsenkirchen
https://www.internet-sicherheit.de