

---

# Rechnernetze und Telekommunikation

**Netzwerksicherheit**

# Übersicht

---

## ◆ Grundlagen

- Bedrohungen, Netzwerk-Sicherheit im Sicherheits-Prozess

## ◆ Kryptographische Verfahren

- Symmetrische und asymmetrische Verschlüsselung, Secure Hashes

## ◆ Authentifizierung

- Kerberos, Zertifikate

## ◆ Sichere Netzwerk-Architekturen

- VPNs, Firewalls, Architektur für Web-Dienste

## ◆ Penetrations-Tests

# Was ist Sicherheit?

---

## ◆ Deutsch ist hier ungenau: Security vs. Safety

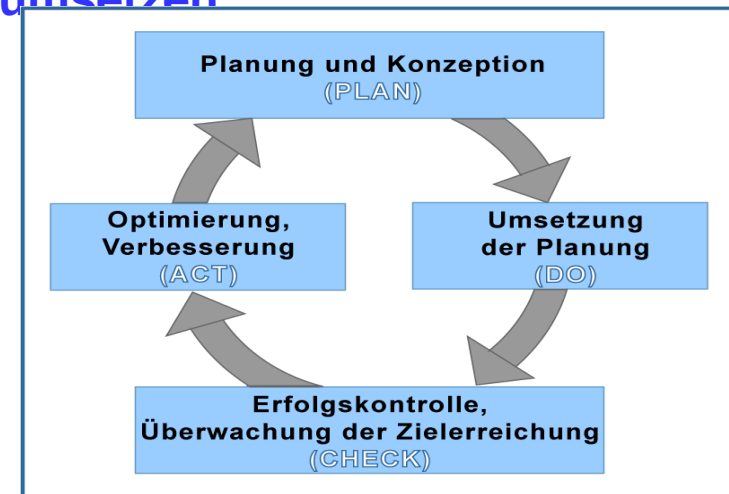
## ◆ IT-Security

- The **protection of information assets** through the use of technology, processes, and training. (Microsoft)
- Als Informationssicherheit bezeichnet man **Eigenschaften** von informationsverarbeitenden und -lagernden (technischen oder nicht-technischen) Systemen, die die **Schutzziele Vertraulichkeit, Verfügbarkeit und Integrität sicherstellen**. Informationssicherheit dient dem Schutz vor **Gefahren** bzw. **Bedrohungen**, der Vermeidung von wirtschaftlichen **Schäden** und der Minimierung von **Risiken**.. (Wikipedia)

## ◆ Begegnet Bedrohungen von IT-Systemen

# Sicherheit als Prozess

- ◆ **Sicherheit ist kein Zustand, sondern ein Prozess**
  - d.h. **Sicherheit unterliegt einer kontinuierlichen Dynamik**
    - (z. B. durch Änderungen im Bedrohungs- und Gefährdungsbild, in Gesetzen oder durch den technischen Fortschritt)
- ◆ **Sicherheit muss aktiv gemanagt, aufrecht erhalten und kontinuierlich verbessert werden**
  - **IT-Systemeinführung planen**
  - **IT-Sicherheitsmaßnahmen definieren und umsetzen**
  - **Erfolgskontrolle regelmäßig durchführen**
  - **Schwachpunkte oder Verbesserungsmöglichkeiten finden**
  - **Maßnahmen verbessern**
    - (Änderungen planen und umsetzen)
  - **IT-Sicherheitsaspekte bei Außerbetriebnahme berücksichtigen**



# ISMS - Information Security Management System

---

## ◆ Komponenten:

- Management-Prinzipien
- Ressourcen
- Mitarbeiter
- IT-Sicherheitsprozess
  - IT-Sicherheitsleitlinie  
(einschl. IT-Sicherheitsziele und -strategie)
  - IT-Sicherheitskonzept

## ◆ Standards

- ISO/IEC 27000
- BSI-Standard 100-1 (kompatibel zu ISO/IEC 27001)

## ◆ Netzwerk-Sicherheit ist nur ein kleiner Baustein!

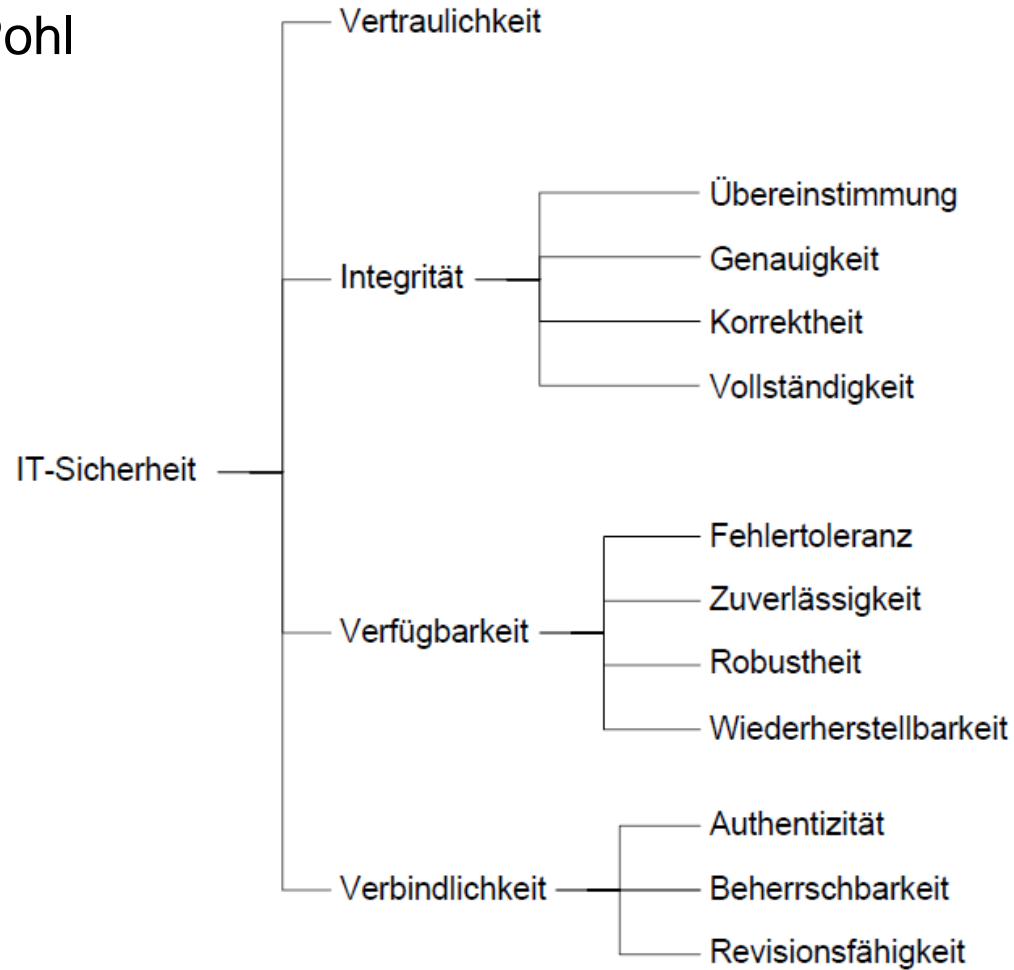
# Sicherheitsziele

---

- ◆ **Vertraulichkeit (privacy)**
  - Es können nur Berechtigte Daten lesen.
- ◆ **Integrität (integrity)**
  - Daten können nicht ohne Berechtigung verändert werden.
- ◆ **Verfügbarkeit (availability)**
  - Ist der Rechner/Service erreichbar?
- ◆ **Verbindlichkeit (non repudiation)**
  - Es kann nachgewiesen werden, wer was gesendet (getan) hat.
- ◆ **Authentizität (authenticity)**
  - Es ist klar, mit wem man kommuniziert
- ◆ **Zugriffskontrolle/Autorisierung (authorisation, access control)**
  - Darf derjenige das, was er tun will?
- ◆ **Potentiell unabhängige Anforderungen!**

# Weitere Klassifikation und abgeleitete Ziele

## ◆ Nach Prof. Hartmut Pohl



# Angriffe

---

- ◆ **Maskierung (Masquerade)**
  - Jemand gibt sich als ein anderer aus
- ◆ **Abhören (Eavesdropping)**
  - Jemand liest Informationen, die nicht für ihn bestimmt sind
- ◆ **Zugriffsverletzung (Authorization Violation)**
  - Jemand benutzt einen Dienst oder eine Resource, die nicht für ihn bestimmt ist
- ◆ **Verlust oder Veränderung (übertragener) Information**
  - Daten werden verändert oder zerstört
- ◆ **Verleugnung der Kommunikation**
  - Jemand behauptet (fälschlicherweise) nicht der Verursacher von Kommunikation zu sein
- ◆ **Fälschen von Information**
  - Jemand erzeugt (verändert) Nachrichten im Namen anderer
- ◆ **Sabotage**
  - Jede Aktion, die die Verfügbarkeit oder das korrekte Funktionieren der Dienste oder des Systems reduziert



# Angriffe auf Ziele

Sicherheitsziele	Bedrohungen						
	Mas- kierung	Abhören	Zugriffs- ver- letzung	Verlust oder Verän- derung (über- tragener) information	Verleug- nung der Kommuni- kation	Fäl- schen von Infor- mation	Sabotage (z.B. Überlast)
Vertraulichkeit	x	x	x				
Datenintegrität	x		x	x		x	
Verantwort- lichkeit	x		x		x	x	
Verfügbarkeit	x		x	x			x
Zugriffs- kontrolle	x		x			x	

## ◆ Überwiegend mit kryptographischen Mechanismen:

- Authentisierung
  - von Systemen/Benutzern (entity authentication)
  - von Datenpaketen (data origin authentication)
- Integritätssicherung (**integrity protection**)
- Verschlüsselung (**encryption**)
- Schlüsselmanagement (**key exchange**)
- ...

## ◆ Ohne kryptographische Mechanismen:

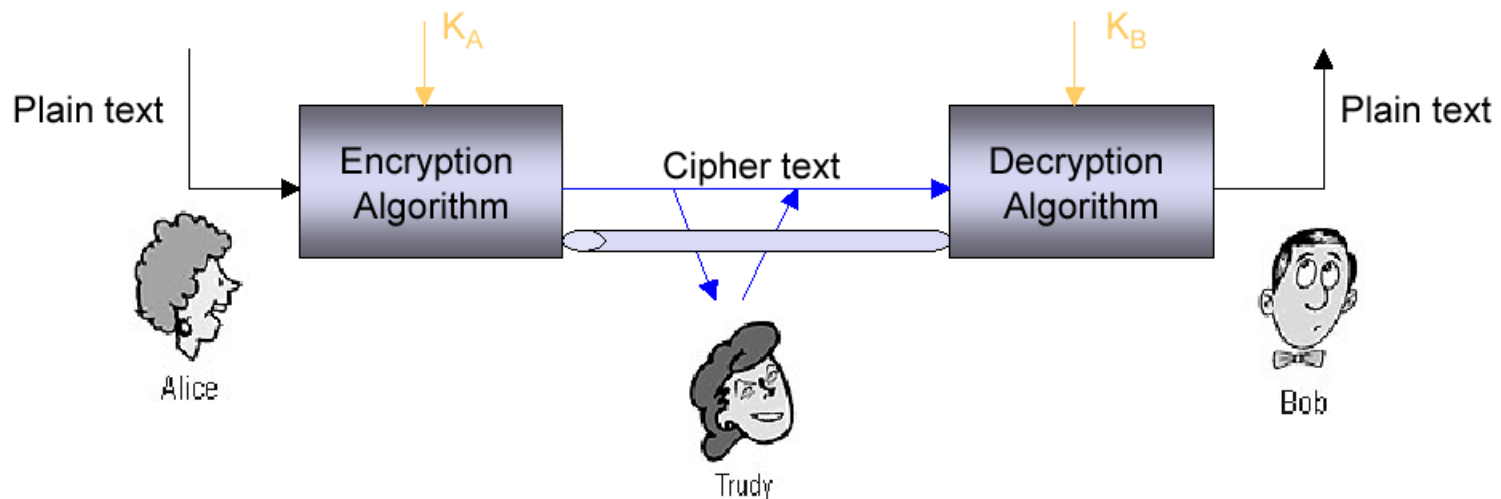
- Zugriffskontrolle (**access control**)
- Policy-Management
- Einbruchserkennung (**intrusion detection**)
- ...

# Prinzipien der Kryptographie

## ◆ Prinzip

- Sender verschlüsselt Daten so, dass ein Intruder die übertragene Information nicht erkennen kann.
- Empfänger ist in der Lage, die Daten zu lesen.

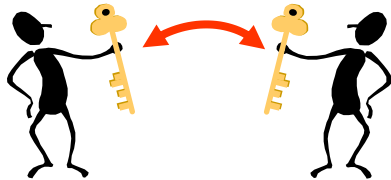
## ◆ Komponenten



# Kryptographie-Verfahren

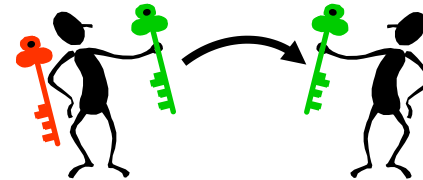
---

## Symmetrische Kryptographie



- ◆ Instanzen besitzen gemeinsamen geheimen Schlüssel.
- ◆ Vorteile:
  - geringer Rechenaufwand
  - kurze Schlüssel
- ◆ Nachteile:
  - Schlüsselaustausch schwierig
  - keine Verbindlichkeit

## Asymmetrische Kryptographie (Public-Key-Kryptographie)



- ◆ Schlüsselpaar aus privatem und öffentlichem Schlüssel
- ◆ Vorteile:
  - öffentliche Schlüssel sind relativ leicht verteilbar
  - Verbindlichkeit möglich
- ◆ Nachteile:
  - hoher Rechenaufwand
  - längere Schlüssel

# Beispiele – Symmetrische Verschlüsselung



ältere ENIGMA (ab 1918)



Vierwalzen-ENIGMA  
(Marineausführung, ab 1942)

# Voraussetzung

---

- ◆ **Notwendige Voraussetzung für sichere Verschlüsselung:**
  - **Durchprobieren der Schlüssel muss aussichtslos sein**
- ◆ **Beispiel: Klartextangriff mit Spezialrechner bei bekanntem symmetrischen Verfahren,  $10^{10}$  Schlüssel pro Sekunde**

<i>Schlüsselgröße</i>	<i>benötigte Zeit</i>	<i>Qualität</i>
40 Bits	100 Sekunden	schlecht
56 Bits	10 Tage	schwach
64 Bits	30 Jahre	mäßig
128 Bits	$10^{20}$ Jahre	gut
256 Bits	$10^{60}$ Jahre	sehr gut

# Kryptoanalyse (1)

---

## ♦ Ziel:

- Code knacken
- Schlüssel und Klartext herausfinden

## ♦ Ansätze:

- Entschlüsselungsangriff – wenn nur Geheimtext vorliegt
- Klartextangriff – wenn zusätzlich Teile des Klartextes vorliegen

## ♦ Notwendige Voraussetzung:

- Sprache der Nachricht muss bekannt sein!

# Kryptoanalyse (2) - Methoden

---

## ◆ Brute Force

- “einfach” alle Schlüssel ausprobieren

## ◆ Seitenkanalangriff

- Nutzt Details einer bestimmten physischen Implementierung eines Kryptosystems
- beobachtet Korrelationen zwischen charakteristischen Informationen der Implementierung um den verwendeten Schlüssel zu finden
  - z.B. Laufzeit des Algorithmus, des Energieverbrauchs des Prozessors oder elektromagnetische Ausstrahlung

## ◆ Time Memory Trade-Off (TMTO)

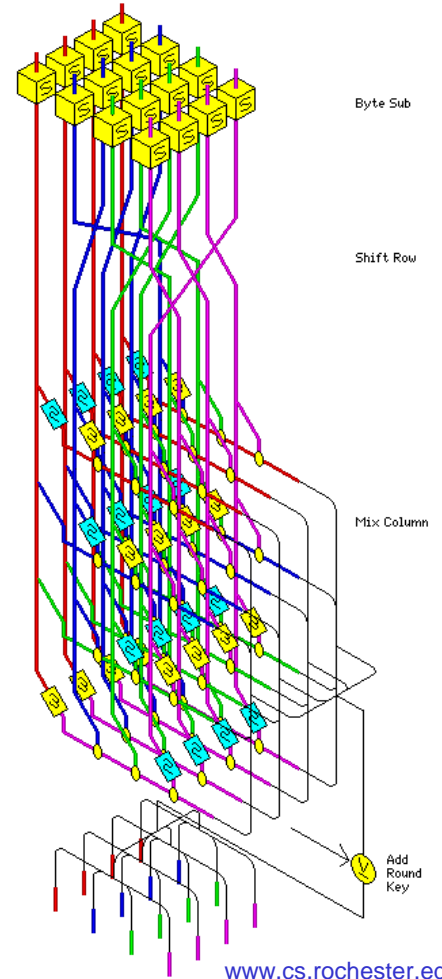
- Nutzen von (sehr viel (GB-TB)) vorberechneter Information, um Schlüssel zu finden
- z.B. “Rainbow-Tables”, die Informationen beinhalten, welcher Chiphertext bei bekanntem Plaintext zu einem bestimmten Schlüssel gehört (“Wörterbuch-Attacke”)



# AES - Advanced Encryption Standard

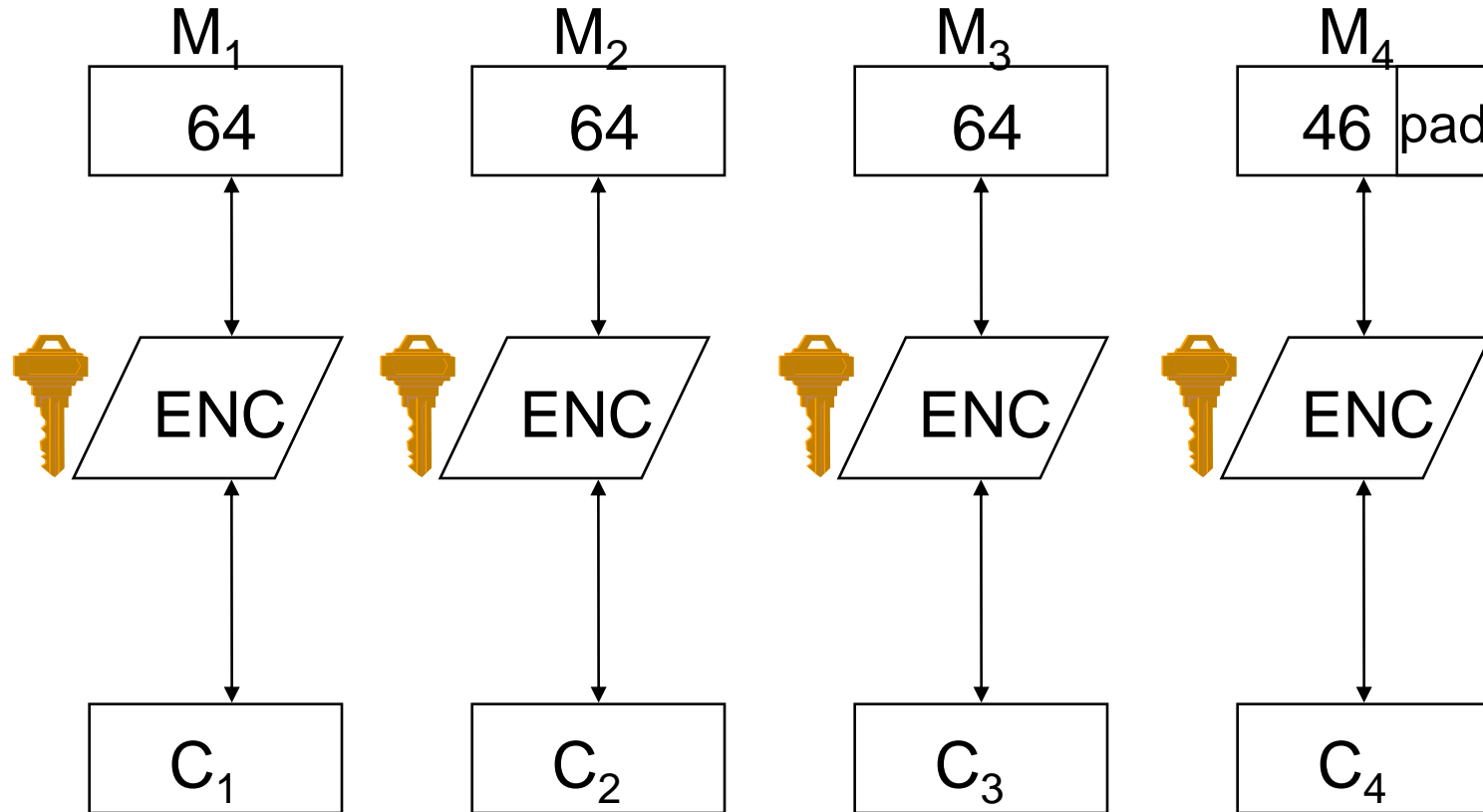
## ◆ Heute DAS symmetrische Verschlüsselungsverfahren

- Standardisiert seit 2001
- Das Verfahren ist bekannt, der Schlüssel ist geheim
- Schlüssellängen von 128, 192 und 256 Bit
- Blockchiffrierung: 64-bit-Blöcke
- Mehrstufiges Verfahren mit Transpositionen und Substitutionen
- Schnelle Realisierung auch in Software möglich
- Hardware-Realisierung ebenfalls möglich
- Weitere Informationen unter <http://csrc.nist.gov/encryption/aes/>



# Electronic Code Book (ECB)

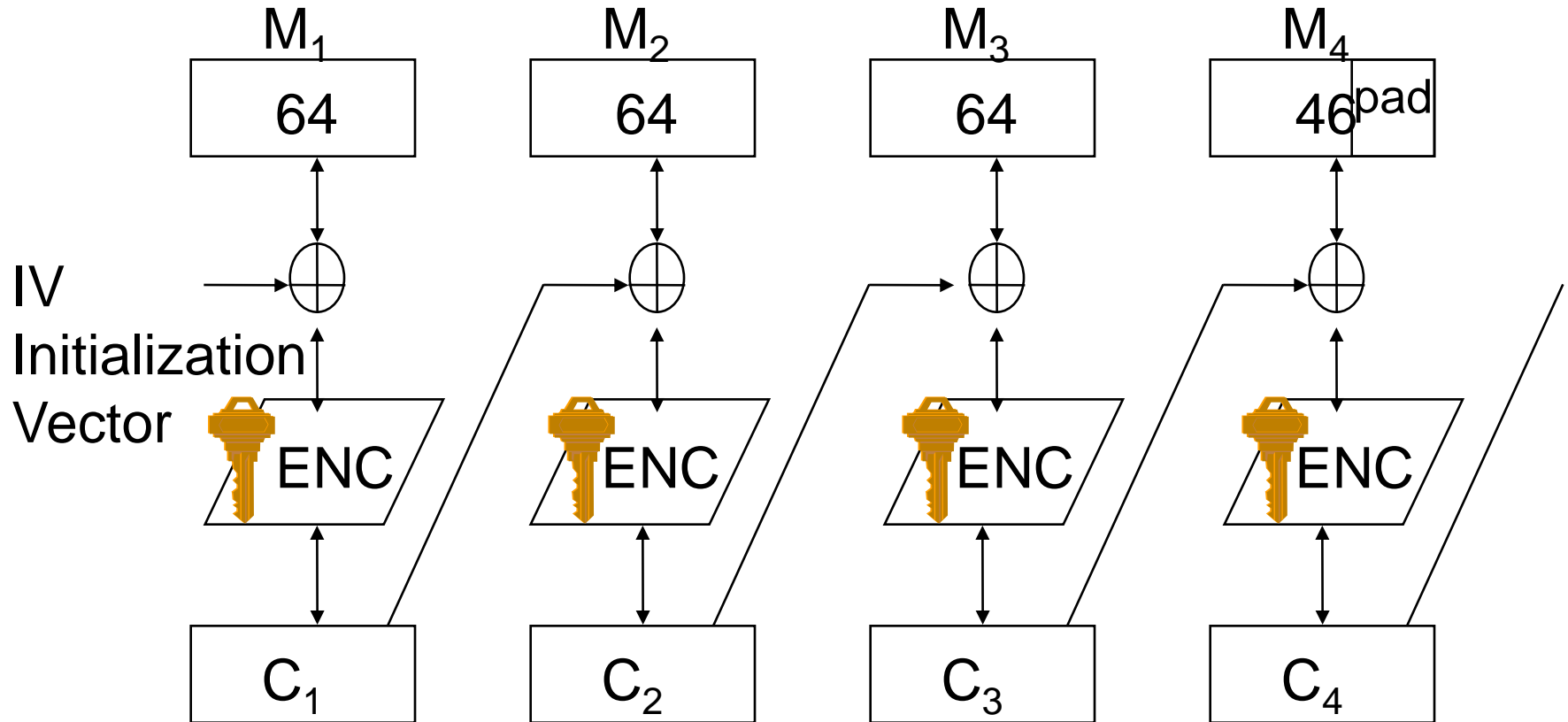
## Elementare Blockverschlüsselung



♦ **Zwei Nachteile:**

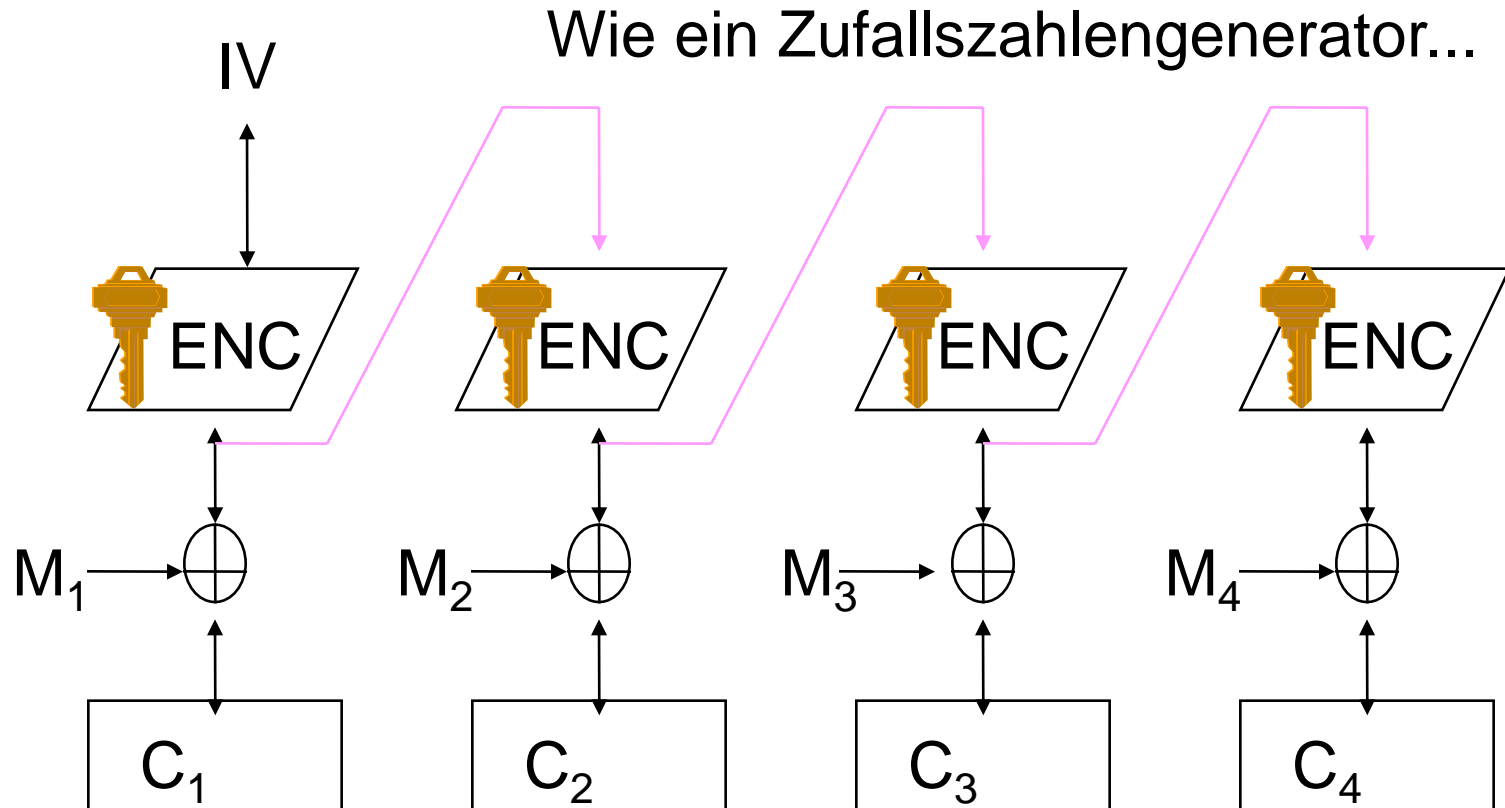
- Wiederholungen von Klartextblöcken im Geheimtext erkennbar
- Wiedereinspielen zuvor abgefangener Blöcke verletzt Authentizität

# Cipher Block Chaining (CBC)



- ◆ benutzt die Blockverschlüsselung für eine Stromverschlüsselung mit Rückkoppelung
  - ( $M_1 = M_3$ ) führt kaum zu ( $C_1 = C_3$ )

# Output Feedback Mode (OFB)



- ◆ Vorteile: Keine Fehlerpropagierung
- ◆ „One-time Pad“ kann im Voraus berechnet werden

# Vor- und Nachteile der symm. Verschlüsselung

---

## Pros:

- ◆ Effiziente Verschlüsselung
- ◆ Kurze Schlüssel
- ◆ Große Erfahrung mit den Algorithmen

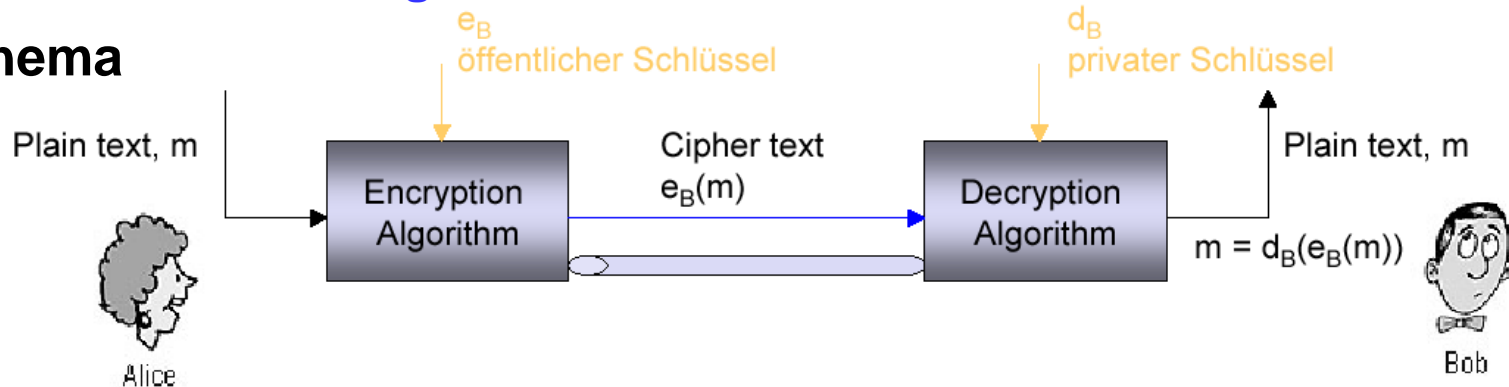
## Cons:

- ◆ Sichere Verteilung der Schlüssel.
- ◆ Viele Schlüsselpaare in einem großen Netzwerk
- ◆ Ggf. eine „Trusted Thrid Party“ TTP erforderlich

# Asymmetrische Kryptographie

- ◆ Kommunikationspartner können sicher kommunizieren, ohne einen gemeinsamen geheimen Schlüssel zu benötigen
  - Es gibt einen öffentlich bekannten Schlüssel und einen privaten Schlüssel
  - Grundlage: Die Berechnung des privaten Schlüssels auf Grundlagen des öffentlichen Schlüssels und des Verschlüsselungsalgorithmus ist praktisch nicht möglich.
- ◆ Vorteil
  - Es müssen keine geheimen Schlüssel verteilt werden

## ◆ Schema



# Der RSA-Algorithmus (1)

- ◆ Entworfen von Ron Rivest, Adi Shamir und Len Adleman



- ◆ Auswahl des privaten und öffentlichen Schlüssels:
  - Auswahl zweier großer Primzahlen  $p$  und  $q$ 
    - 768 bit für private Nutzung empfohlen von RSA Laboratories
    - 1024 bit für Nutzung innerhalb einer Firma
  - Berechne  $n = p * q$  und  $z = (p-1) * (q-1)$
  - Wähle eine Zahl  $e < z$ , die außer 1 keinen gemeinsamen Faktor mit  $z$  hat
  - Finde  $d$ , so dass  $ed-1$  durch  $z$  dividierbar ist
    - $d$  wird so gewählt, dass  $ed/z = 1$
    - Modulo-Operation
  - Öffentlicher Schlüssel:  $(n, e)$ , Privater Schlüssel:  $(n, d)$

# Vor- und Nachteile der asymm. Verschlüsselung

---

## Pros:

- ◆ Nur der private Schlüssel muss geheim gehalten werden
- ◆ Schlüsselmanagement erfordert nur Vertrauen in die Funktion der TTP (Trusted Third Party)
- ◆ Langlebige Schlüssel

## Cons:

- ◆ Geringer Durchsatz
  - Faktor 1000 und mehr gegenüber symm. Kryptographie
- ◆ Lange Schlüssel
- ◆ Sicherheit beruht auf wenigen mathematischen Prinzipien
- ◆ Beschränkte Erfahrung

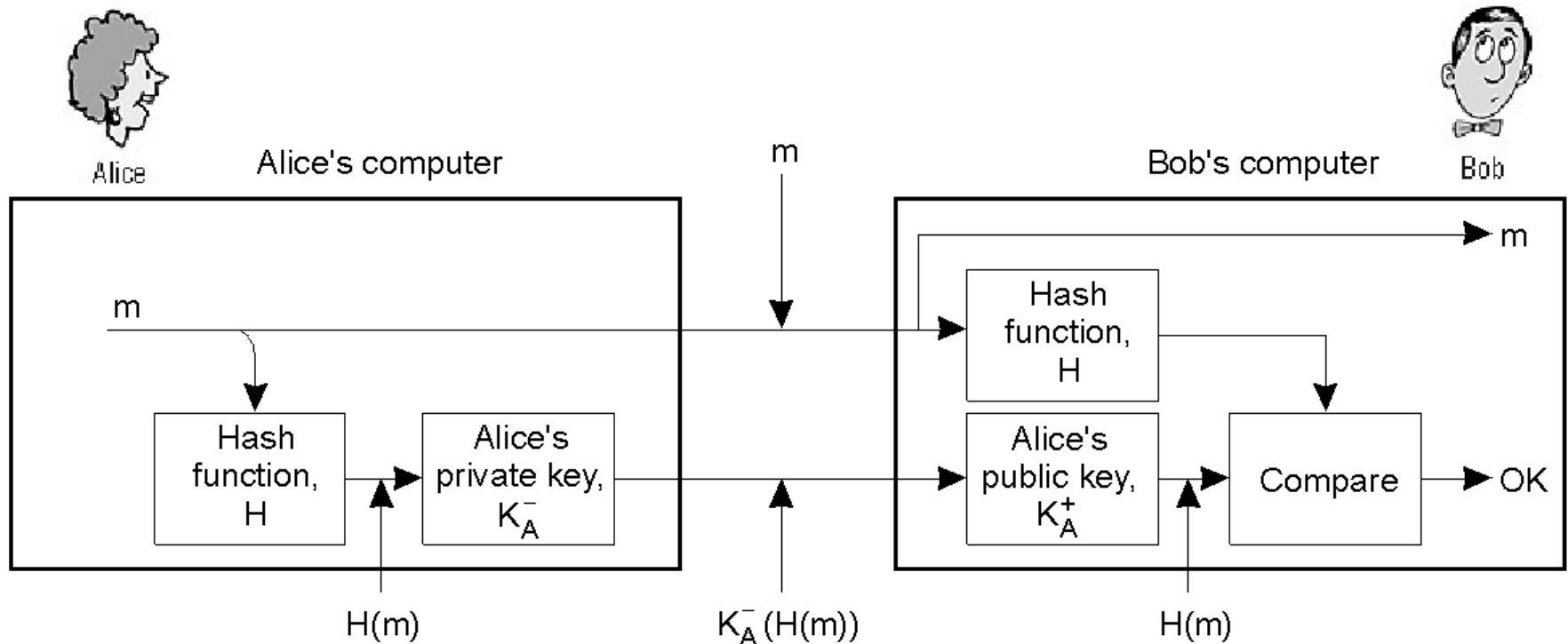


# Folgerung

---

- ◆ **Asymmetrische (Public Key) Verschlüsselung für**
    - **Schlüsselmanagement**
    - **Digitale Signaturen**
    - **Authentifizierung**
  - ◆ **Symmetrische (Shared Secret Key) Verschlüsselung für**
    - **Effiziente Verschlüsselung von großen Datenmengen**
- ➔ **Man benutzt asymmetrische Verfahren um einen Schlüssel für die anschließende symmetrische Verschlüsselung auszuhandeln**

# Digital Signaturen



- ◆ Digital Signatur mit einem Public-Key Verfahren und einer Hash-Funktion

# Message Digest

---

## ◆ Ziel

- Einfach zu berechnende digitale Signatur fester Länge (Fingerabdruck)

## ◆ Beispiel

- MD5 (128 Bit-Hashwert, 1991 Ron Rivest)
- SHA-1 (160 Bit-Hashwert, 1994 NIST)

## ◆ Vorgehensweise

- Anwenden der Hashfunktion  $H$  auf Nachricht  $m$ 
  - Message Digest:  $H(m)$

## ◆ Eigenschaften von Hashfunktionen

- Many-to-One
- Ergebnis fester Länge
- Bei gegebenem Message Digest  $x$  ist es praktisch unmöglich,  $H$  so zu ermitteln, dass  $H(m) = x$
- Es ist praktisch unmöglich, zwei Nachrichten  $m$  und  $m'$  zu finden, so dass  $H(m) = H(m')$  (*Kollision*, Verfahren für SHA-1 zz. bei  $2^{69}$  Versuche)

# Bsp: Schwächen von Message Digest-Algorithmen

---

## ◆ MD5

- Kollision bei MD5-Hashes
- Anwendung: gefälschtes PKI-Zertifikat
- 2008 auf 200 Playstation-3-Spielkonsolen berechnet

## ◆ SHA-1

- 2005: Komplexität zum Finden einer Kollision:  $2^{69}$  (statt  $2^{80}$ )
- 2009: Komplexität zum Finden einer Kollision:  $2^{52}$
- Folge: SHA-1 bald nicht mehr sicher

## ◆ Lösung: neue Algorithmen

- 2005: SHA-2 (SHA-224, SHA-256, SHA-512)
  - Gleicher Algorithmus wie SHA-1, nur längere Schlüssel
- 2012: Auswahl des Algorithmus “Keccak” durch das NIST als SHA-3

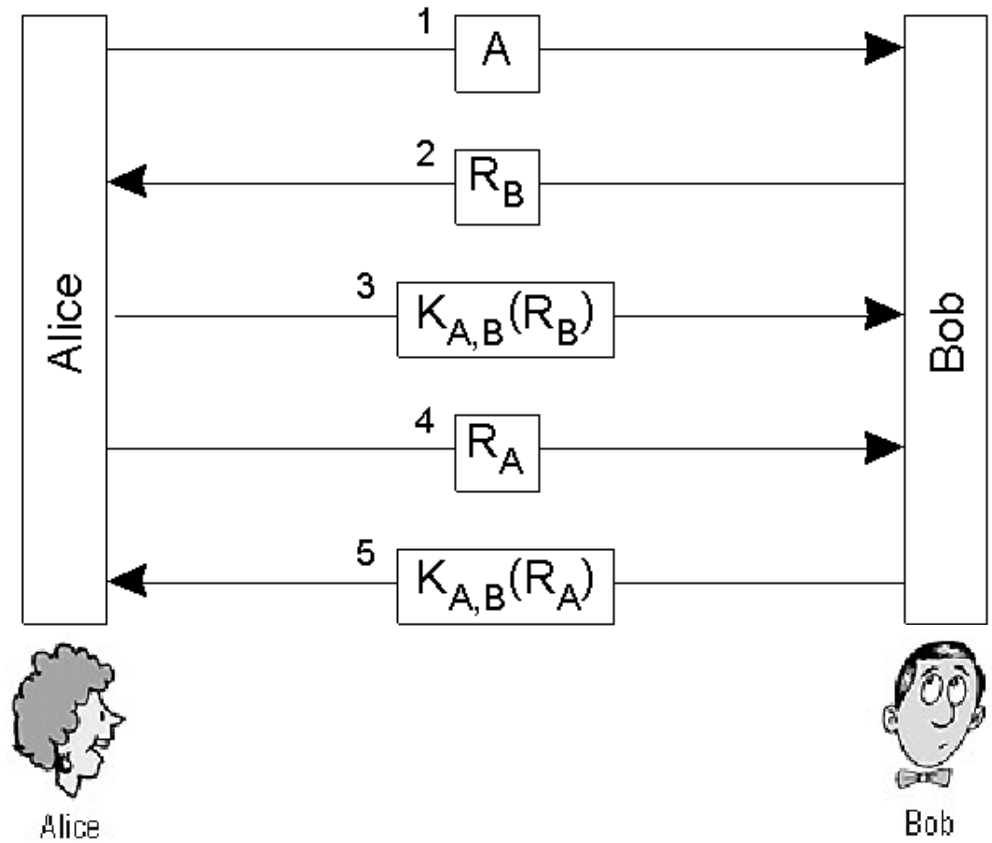
# Authentifizierung und Authentisierung

---

- ◆ **Authentifizierung (engl. authentication)**
  - Vorgang der Überprüfung der Identität eines Gegenübers
- ◆ **Authentisierung**
  - Vorgang des Nachweises der eigenen Identität. Zuweisung und Überprüfung von Zugriffsrechten auf Daten und Diensten
- ◆ **Zwischen Nutzern und/oder Maschinen**
  - Identität einer Maschine
    - IP-Adresse, Hostname, UID, ... ?
- ◆ **Generelles Problem: Identity-Management**
  - “User-Provisioning”
    - Verwalten von Nutzern und Accounts
  - Gewünscht: SSO (Single Sign-on) eine Authentifizierung für alle Dienste

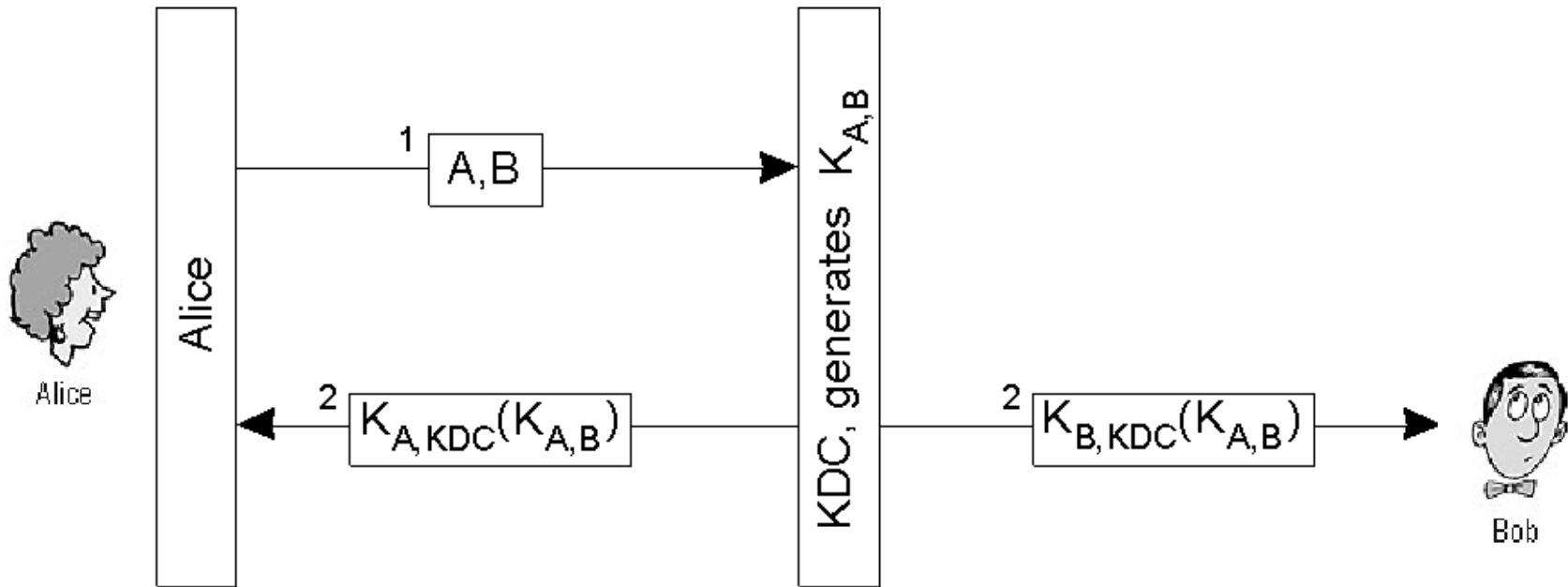
# Authentifizierung mit Secret Keys (1)

- Ziel
  - Bob möchte, dass Alice ihre Identität beweist
- Protokoll mit Shared Secret Key
  - Nonce: Zufallszahl ( $R$ ), die der Benutzer eines Protokolls nur einmal benutzt
  - Alice sagt „I am Alice“.
  - Bob sendet Nonce  $R$ , der von Alice verschlüsselt zurück gesendet wird.
  - Anschließend umgekehrt
- Challenge-Response
  - Häufig genutztes Verfahren



## Authentifizierung mit Secret Keys(2)

### ♦ Mit einem Key Distribution Center (KDC)



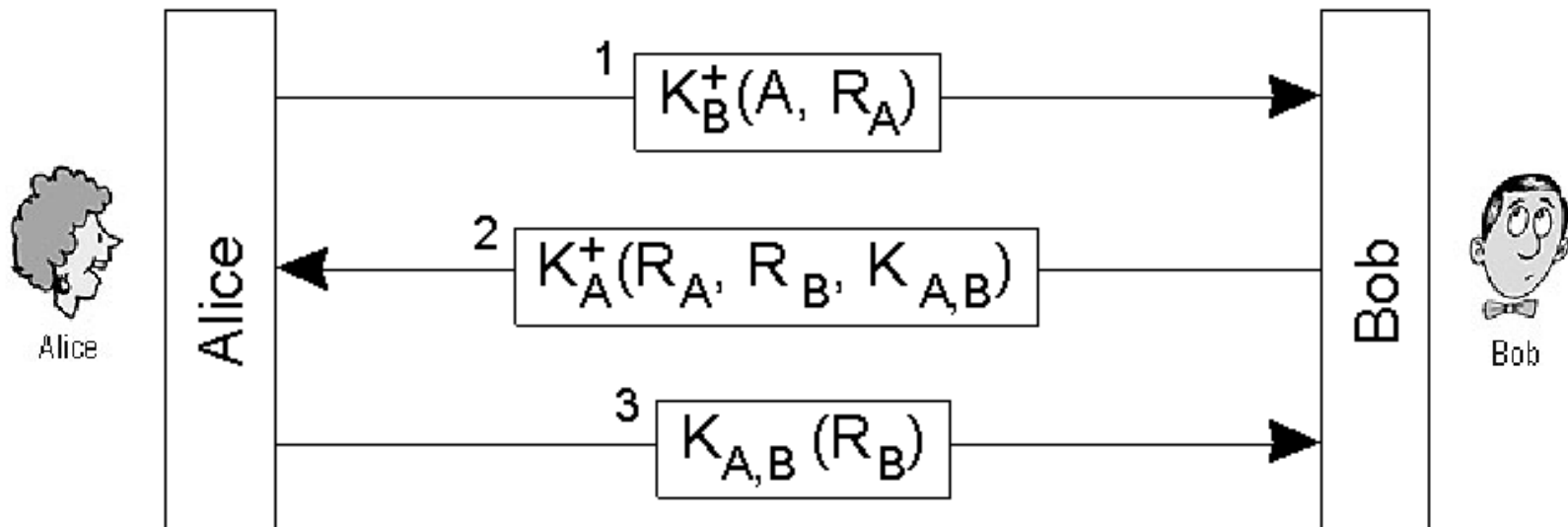
# Kerberos

---

- ◆ **Authentifizierungsdienst, der am MIT entwickelt wurde.**
  - Benutzt ein KDC
  - Authentifizierung von Benutzern, die auf Server im Netz zugreifen
  - Ursprünglich für den Einsatz in einer einzelnen Domäne (z. B. Uni) konzipiert
  - Ähnlich dem vorne vorgestellten Konzept
- ◆ **Einsatz**
  - OSF Distributed Computing Environment (DCE)
  - Microsoft Windows seit Windows 2000
- ◆ **Kerberos Server übernimmt die Rolle des KDC**
  - umfasst einen Authentication Service (AS) und
  - einen Ticket Granting Service (TGS)



# Authentifizierung mit Public Key Verfahren (1)



- ◆ Basierend auf  $R_A$  und  $R_B$  kann nun ein Session-Key für eine nachfolgende symmetrische Verschlüsselung bestimmt werden
- ◆ Angewendetes Verfahren bei
  - SSL, HTTPS, TLS, SSH, ...

# Authentifizierung mit Public Key Verfahren (2)

---

## ◆ Problem:

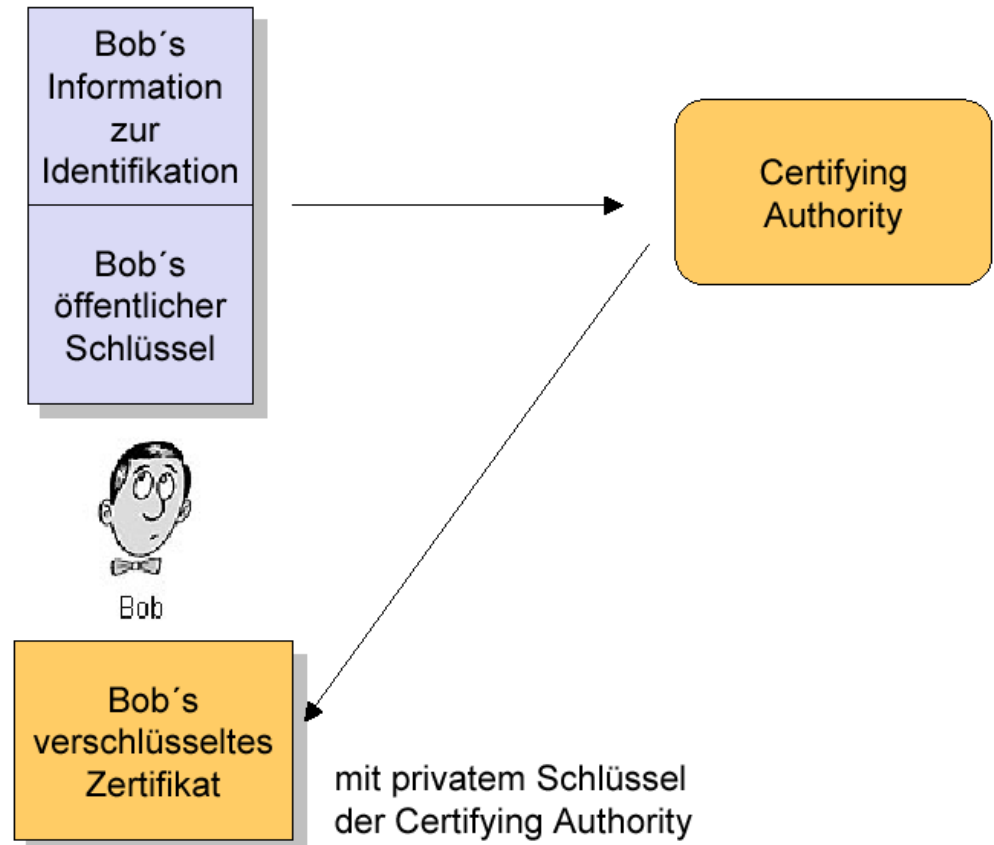
- Wie kann man sicher sein, dass man den richtigen Public Key kennt?
- Veröffentlichung (z.B. auf der Web-Site) ist ganz gut, aber nicht wirklich sicher

## ◆ 2 Ansätze

- Web-of-Trust: Nutzer bestätigen sich Peer-2-Peer die Gültigkeit von Schlüsseln
  - Beispiel: PGP
  - Probleme: Skalierung und benötigtes Verständnis beim Nutzer
- Public Key Infrastructure (PKI): Eine Hierarchie von trusted „Certification Authorities“ bestätigt zentral die Gültigkeit von Public Keys (Zertifikate)

# Public-Key Zertifizierung

- Vertrauenswürdige Instanz (Certifying Authority – CA) bestätigt den Zusammenhang zwischen Public-Key und einer Person/Institution
- Bestätigter Public-Key ist ein Zertifikat
- „Chain-of-Trust“ verschiedener CAs möglich
- Den Public-Keys der Root-CAs muss man vertrauen
  - z.B. in Browsern und Mail-Programmen eingebaut




# Standard für Zertifikate: X.509

---

## Zertifikat:

- Version
- Seriennummer
- Algorithmus/Parameter
- Aussteller
- Geltungsdauer
- Betreff
- Public Key
- Aussteller-Signatur

X.509 Zertifikat von X	
Public Key von X	
Seriennummer des Zertifikates	
Gültigkeitszeitraum	
Eindeutiger Name von X	
Eindeutiger Name der Zertifizierungsstelle	
Digitale Unterschrift der Zertifizierungsstelle	

# Mechanismen in Protokollen (1)

---

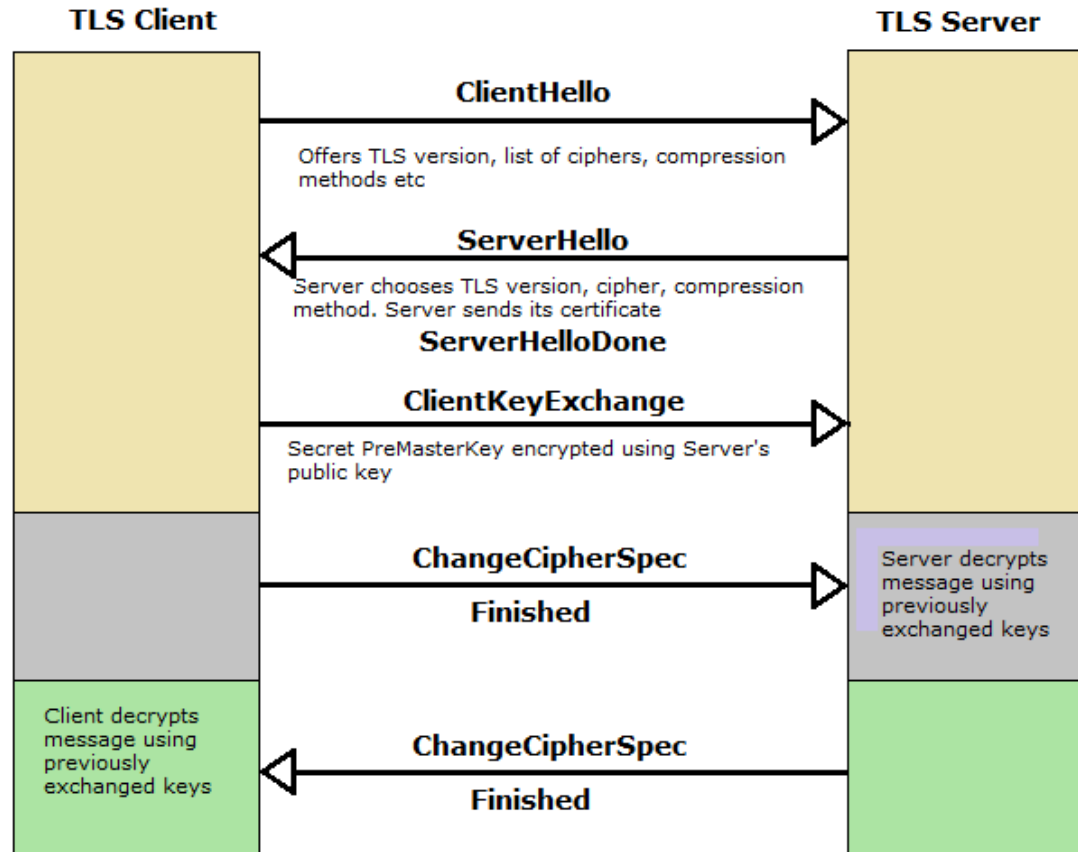
## ♦ HTTPS (HTTP secure): „sicherer“ Web-Zugriff

- Authentifizierung des Servers mittels Zertifikat
  - Meist authentifiziert sich der Client auf Anwendungsebene (mit Passw.)
  - Optional auch: Authentifizierung des Clients mittels Zertifikat
- Verschlüsselung der übertragenen Daten mittels sym. Verschlüsselung
- Übertragung über TCP-Port 443 (statt 80 für HTTP)
- Nutzt SSL/TLS, ähnlich z.B. WLAN mit PEAP

## ♦ S/MIME: signierte und/oder verschlüsselte Email

- Signierte Email mittels Zertifikat des Senders (s.o.)
- Verschlüsselte Email mittels Zertifikat des Empfängers
  - Verschlüsselt mit Public Key des Empfängers
  - Nur er kann das mit seinem Private Key wieder entschlüsseln
  - Daten werden wieder mit sym. Verschlüsselung verschlüsselt
    - Nur der sym. Schlüssel wird im „Envelope“ asym. verschlüsselt

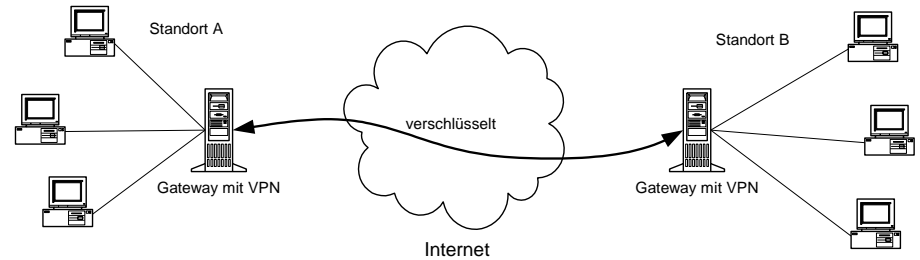
# Mechanismen in Protokollen (2): Ablauf TLS (Transport Layer Security)



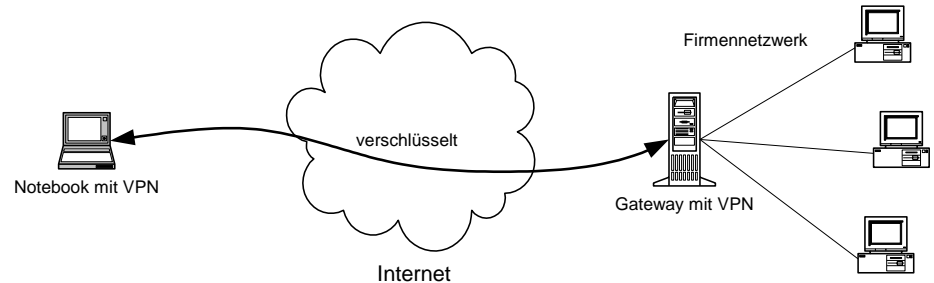
# Mechanismen in Protokollen (2)

## VPN – Virtual Private Network

- **Vollwertige LAN-LAN**  
Zusammenführung zweier räumlich getrennter IT-Netzwerke (private oder öffentliche Netze)
- **Vollwertiger Zugriff auf alle Ressourcen eines IT-Netzwerks von überall: Schaffung eines virtuellen lokalen Anschlusses**
- **Gewährleistet**
  - **Vertraulichkeit**
  - **Authentifizierung**
  - **Integrität**



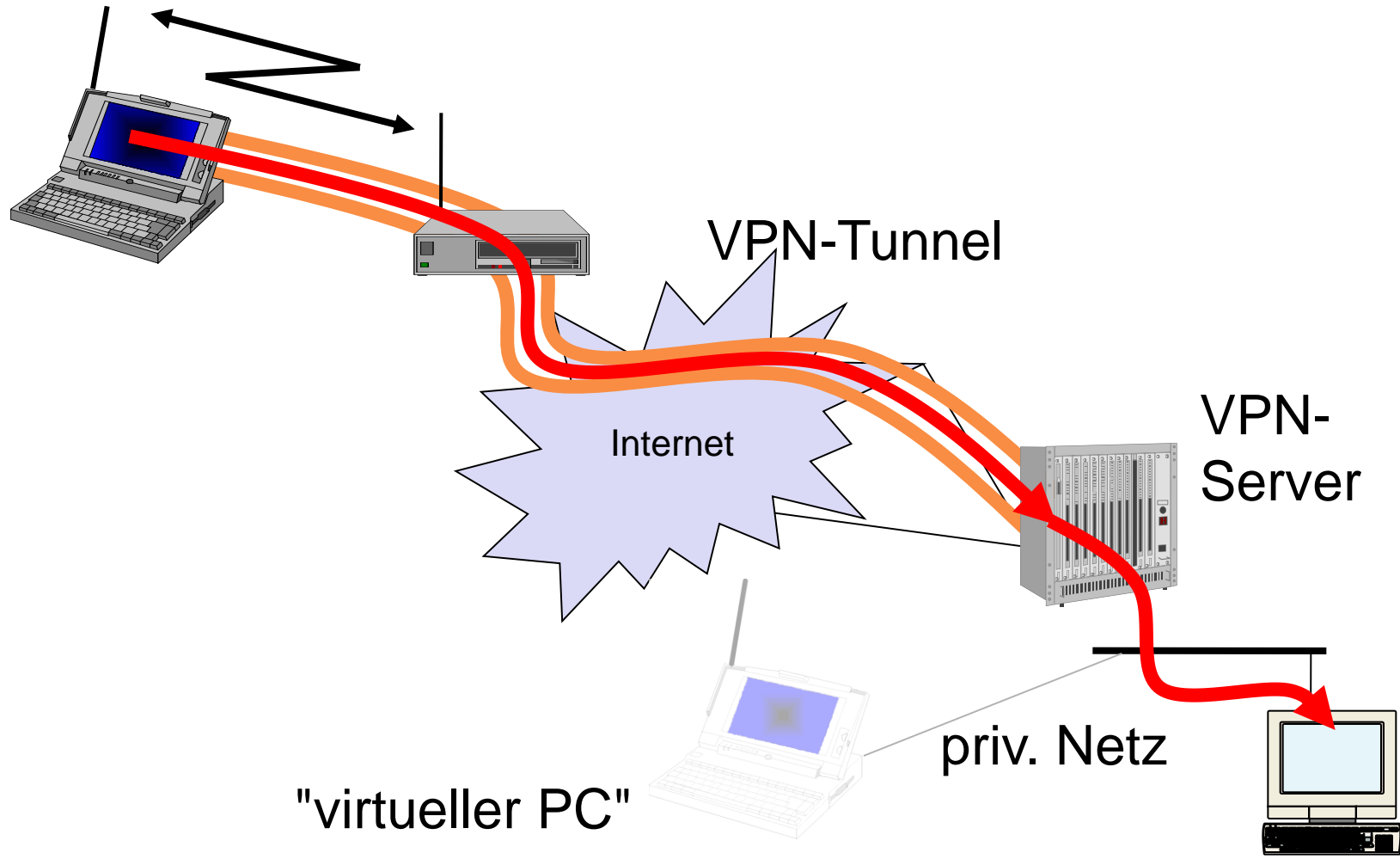
Site-to-Site



End-to-Site

# Mechanismen in Protokollen (3)

## Prinzip des VPN-Tunnels





# Mechanismen in Protokollen (4)

## VPN-Protokolle

---

- ◆ **IPSec VPN: remote Zugang zu einem Netz**
  - **Verschlüsselt werden IP-Pakete**
    - D.h. alles, was über IP versendet wird, ist verschlüsselt
  - **Authentifizierung des Clients und des Servers mittels Zertifikaten**
    - Es authentifizieren sich zunächst die Maschinen
    - Meist authentifiziert sich der Anwender danach zusätzlich mit Passw.
  - **Verschlüsselung der übertragenen Daten mittel sym. Verschlüsselung**
- ◆ **SSL VPN: remote Zugang zu einem Netz oder Diensten**
  - **Nutzt HTTPS**
  - **Kann IP-Pakete durch HTTPS *tunneln***
    - D.h. alles wird eingepackt und mit HTTPS in einer „Web-Session“ übertragen
  - **Vorteil gegenüber IPSec: funktioniert auch, wenn alle Dienste außer Web gesperrt sind (also fast immer!)**

# Sichere Netzwerk- und Systemarchitekturen

---

## ♦ Eine Kernkomponente: Firewall

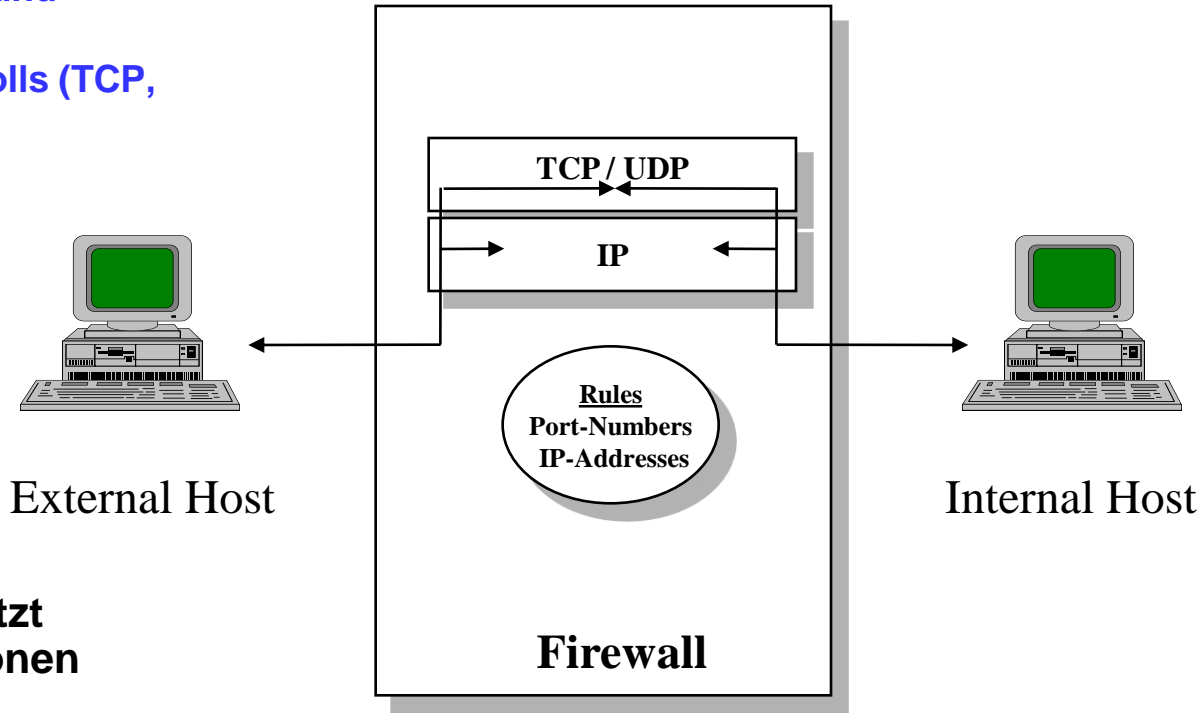
- Verbindung zwischen „sicherem“ und „unsicherem“ Netz
- Regelt und überwacht gesamten Datenverkehr
- Oder besser: zwischen verschiedenen Sicherheits-Domänen
  - Können auch innerhalb einer Organisation sein
  - z.B. zwischen WLAN und Festnetz, zwischen Produktion und Verwaltung, etc.

## ♦ Besteht meist aus mehreren Komponenten

- Packet Filter
- Application Gateways
- ggf. Intrusion Detection System (IDS)
- ggf. VPN-Gateway

# Packet Filter (1)

- Kontrolliert ankommende IP-Pakete anhand folgender Informationen:
  - IP Header, vor allem Quell- und Zieladresse
  - Art des „Transport“-protokolls (TCP, UDP, ICMP)
  - TCP/UDP Header, vor allem Quell- und Zielport
  - ICMP Nachrichtentyp
  - Interface, auf dem Paket ankommt
- Static Packet Filter: benutzt ausschließlich obige Quellen
- Dynamic Packet Filter: benutzt zusätzlich Kontextinformationen
- Löschen nicht-regelkonformer Pakete
  - Weiterleitung des Rests



## Packet Filter (2)

---

### ◆ Alternative Strategien zur Regelung

- **Permissive:** was nicht verboten ist, ist erlaubt
- **Prohibitiv:** was nicht erlaubt ist, ist verboten

### ◆ Beispiel (prohibitiv):

- **Reihenfolge ist entscheidend**

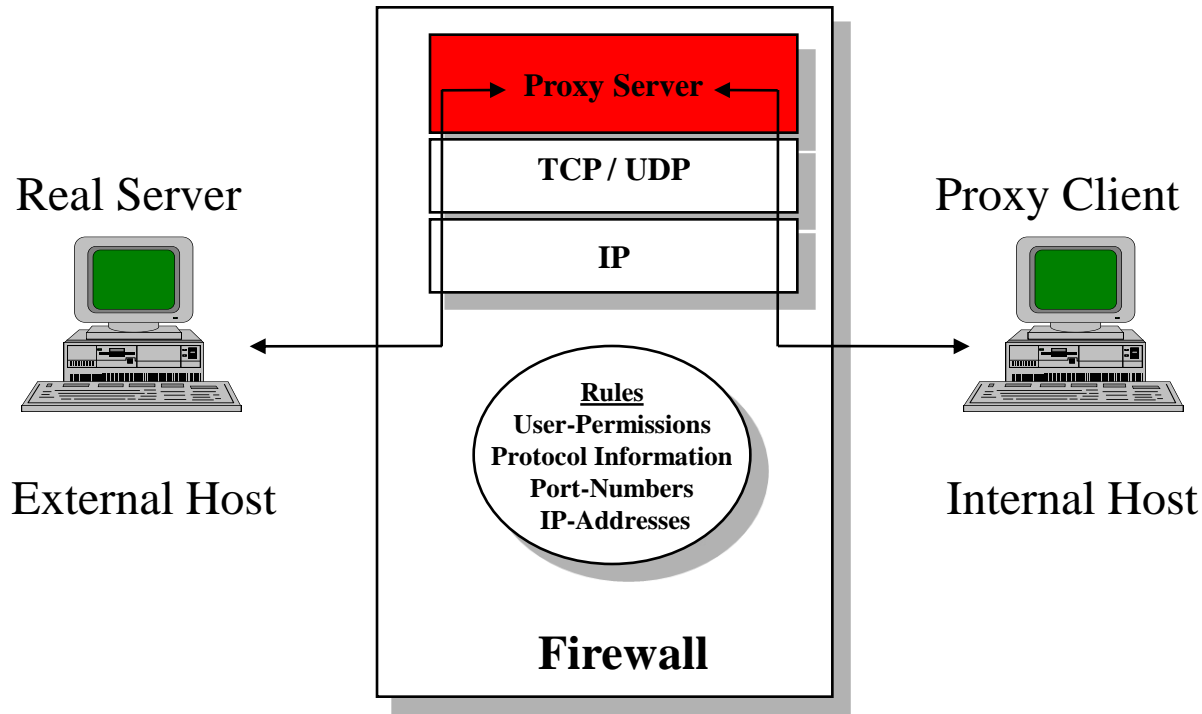
Zugang	Quelle	Port	Ziel	Port	Attribute
erlaubt	{inneres Netz}	*	*	25	*
erlaubt	*	25	*	*	ACK
...	...	...	...	...	...
blockiert	*	*	*	*	*

### ◆ Implementierungen

- **Linux:** IPChains, IPFilters, IPTables
- **Windows:** Internet Firewall
- **Router:** ACL (Access Control List)

# Application Gateways (Proxy Servers) (1)

- ◆ Proxy Server interpretieren Kommandos bzw. Daten und leiten diese abhängig von den Filter Regeln weiter
  - Spricht Protokolle der Anwendungsschicht
  - Spezialisierter Code für jede Anwendung



# Application Gateways (Proxy Servers) (2)

---

## ◆ Typische Proxies für:

- HTTP(S), SMTP/IMAP, Citrix (Terminalserver)

## ◆ Vorteile

- oft unveränderte Client-Software
- detaillierte Überwachung der übertragenen Daten
  - dadurch z.B. Virenschannung, Inhalt von Webseiten, ...
- umfassendes Logging möglich
- Lecks in inneren Diensten werden abgeschirmt

## ◆ Nachteile

- noch speziellere Programme pro Dienst nötig
- mehr Software (Unsicherheit) und Verwaltungsaufwand
- weniger Durchsatz als Transport Gateways

# Web Proxy Servers

---

- ◆ Für ausgehenden Web-Verkehr von internen Browsern
- ◆ Aufgaben eines Web-Proxies:
  - **Zwischenspeicher (Cache)**
    - gestellte Anfragen zu statischen Inhalten bzw. deren Ergebnis werden gespeichert
  - **Security-Scanner**
    - Scannen von Inhalten nach Schadcode
  - **Zensur/Zugriffssteuerung**
    - Sperren oder Protokollieren von bestimmten Webzugriffen
    - ggf. nutzerabhängig
    - Auch z.B. Ausfiltern von Werbung
  - **SSL-Terminierung**
    - Aufbrechen einer SSL-Verbindung (terminiert), um auch deren Inhalte auf Schadcode zu überprüfen
    - weitere Verschlüsselung zum Client (Browser) mit Proxy-Zertifikat, Problem: Benutzer sieht das Originalzertifikat nicht mehr

# Reverse Proxy Servers

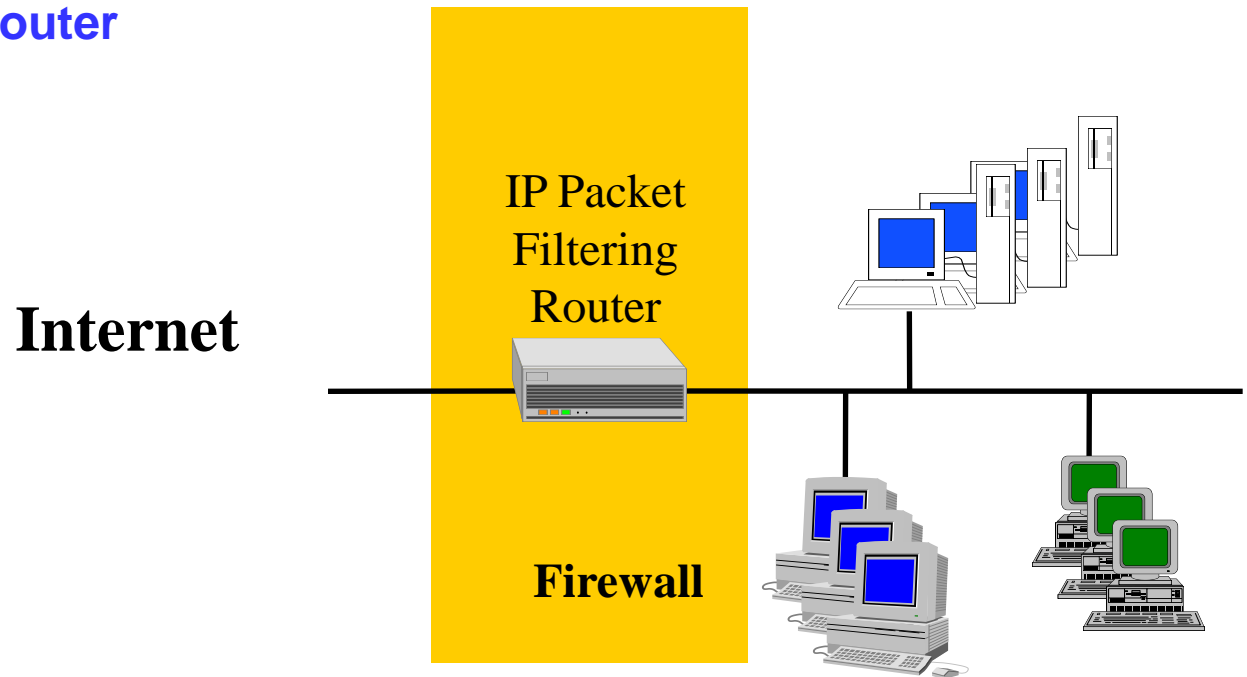
---

- ◆ Für eingehenden Web-Verkehr auf interne Server
- ◆ Aufgaben eines Reverse Web-Proxies:
  - Zwischenspeicher (Cache für statische Inhalte)
  - Lastverteiler (bei mehreren App-Servern im Backend)
  - Security-Scanner
    - Scannen von Inhalten nach Schadcode
    - Abweisen von unerwünschten Zugriffen (z.B. wg. IP-Adresse)
    - u.A. auch Bereinigung von URLs mit Script- oder SQL-Injection
  - Authentifizierung/Autorisierung
    - Prüft Credentials und verwaltet Sessions
    - Leitet nur authentifizierte Requests an Application-Server weiter (mit ID-Token)
    - Single-Sign-On!
  - Sicherer Zugangspunkt
    - Bietet einen HTTPS-Zugangspunkt für alle Dienste
  - Logging und Auditing (protokolliert Zugriffe)



# Firewall Architektur(1): Packet Filtering Router

- ◆ **Einfachste Lösung**
  - Nur Packet-Filtering
  - z.B. viele DSL-Router



# Firewall Architektur(2): DMZ (Demilitarized Zone)

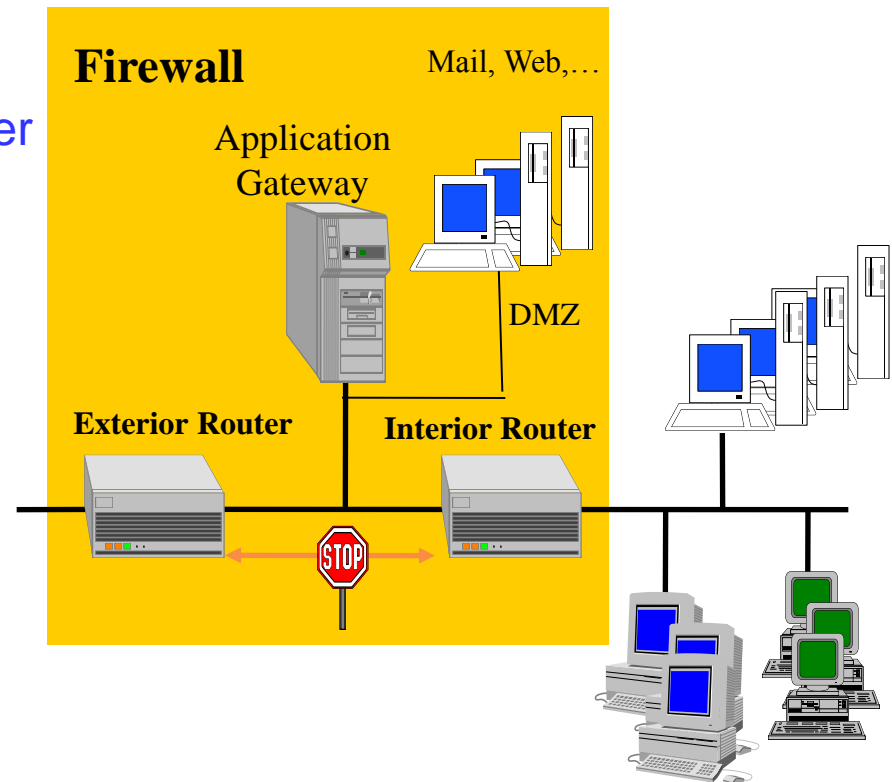
## ♦ Vorteile

- **Mehrere Hürden für Angreifer**
  - “Defense in Depth”
  - Ggf. Router/Packet-Filter von verschiedenen Herstellern
  - Innerer Router (bei Eroberung der DMZ einfach zu schließen)
- **DMZ kann heiklere Netzdienste anbieten**

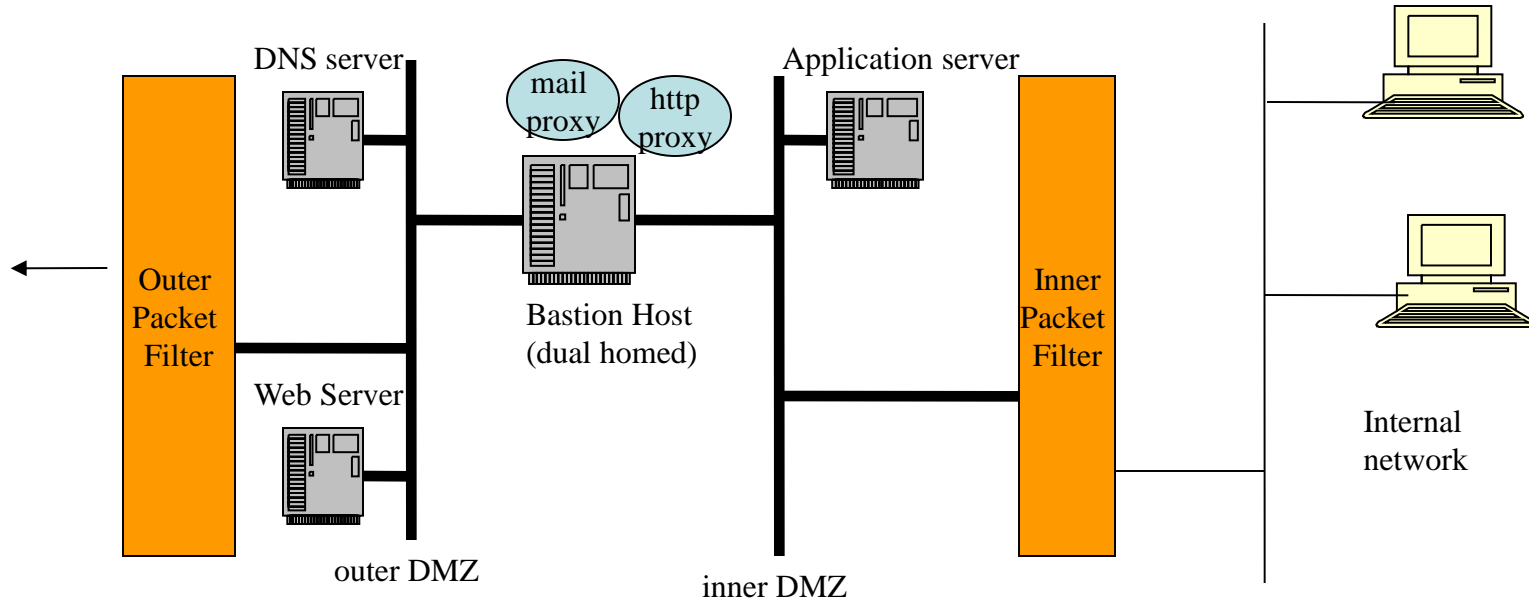
## ♦ Nachteile:

- **Weniger Durchsatz**
- **Komplexer und teurer**

Internet



# Mehrstufige Firewall



## ♦ Die Topologie und die Security Policies bestimmen:

- welche Art von Applikation in welcher Zone laufen darf
- welche Protokolle in welcher Zone zulässig sind
- ab wann Requests authentisiert sein müssen
- wer wie auf diese Zonen zugreifen darf, z.B. für Software-Wartung

# Firewalls

---

## ◆ Ungelöste Probleme

- Eingeschränkter Zugriff auf erwünschte Dienste
- Evtl. Durchsatzprobleme

## ◆ Firewalls helfen nicht gegen

- schlechte Passwörter
- Social Engineering
- physisches Eindringen
- 'Angriffe von Innen'
- 'Hintertüren' (z.B. Modems)
- Viren und Mail Bomben

## ◆ Auch im eigentlichen Einsatzgebiet kein vollkommener Schutz - Beispiele:

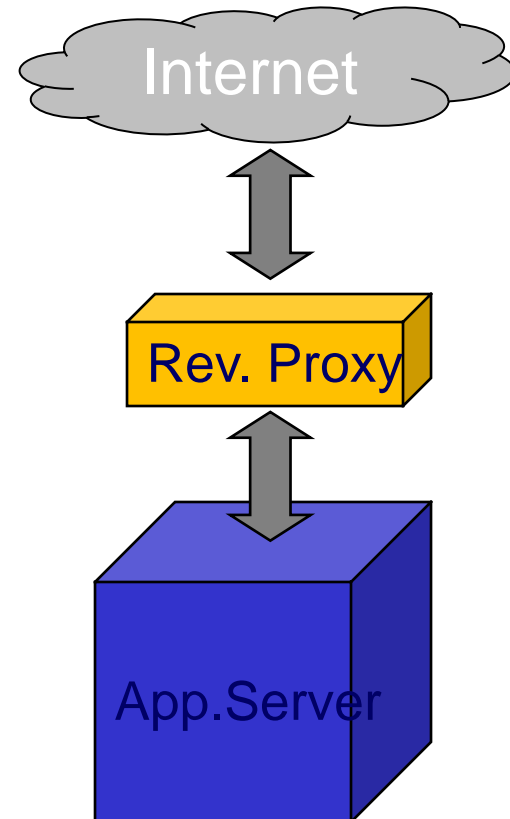
- Programm- und Konfigurationsfehler auf der Firewall
- Denial of Service Angriffe
- Schwierige, aber notwendige Anwendungsprotokolle

## ◆ Daher

- Abwägen von Schutz gegen Kosten

# Enterprise-Architektur für gesicherte Web-Services (1)

- **Application-Server**
  - z.B. xAMP
  - ggf. selbst verteilte Architektur
- **Angebunden über Reverse Proxy**
  - **App-Level Gateway**
    - Filtert Requests und Inhalte
  - **Stellt HTTPS-Zugang bereit**
  - **Überprüft Identitäten und Zugriffsrechte**
  - **Managet User-Sessions**
    - Session-Cookies für den Client
    - Tokens für den App.-Server
  - z.B. WebSEAL (IBM)



# Enterprise-Architektur für gesicherte Web-Services (2)

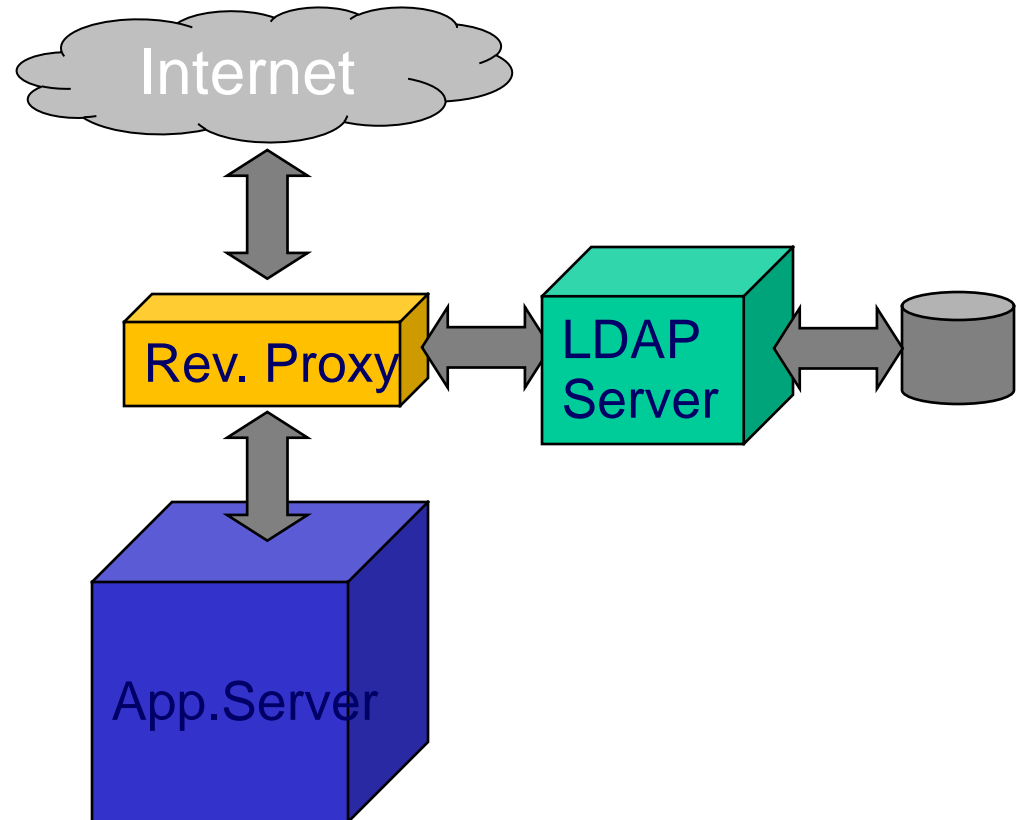
- **LDAP-Server zur Verwaltung**

- der Nutzer-Accounts
- der Nutzergruppen
- der Nutzer-Credentials
  - Passworte
  - Zertifikate

- **z.B. MS Active Directory, eDirectory oder OpenLDAP**

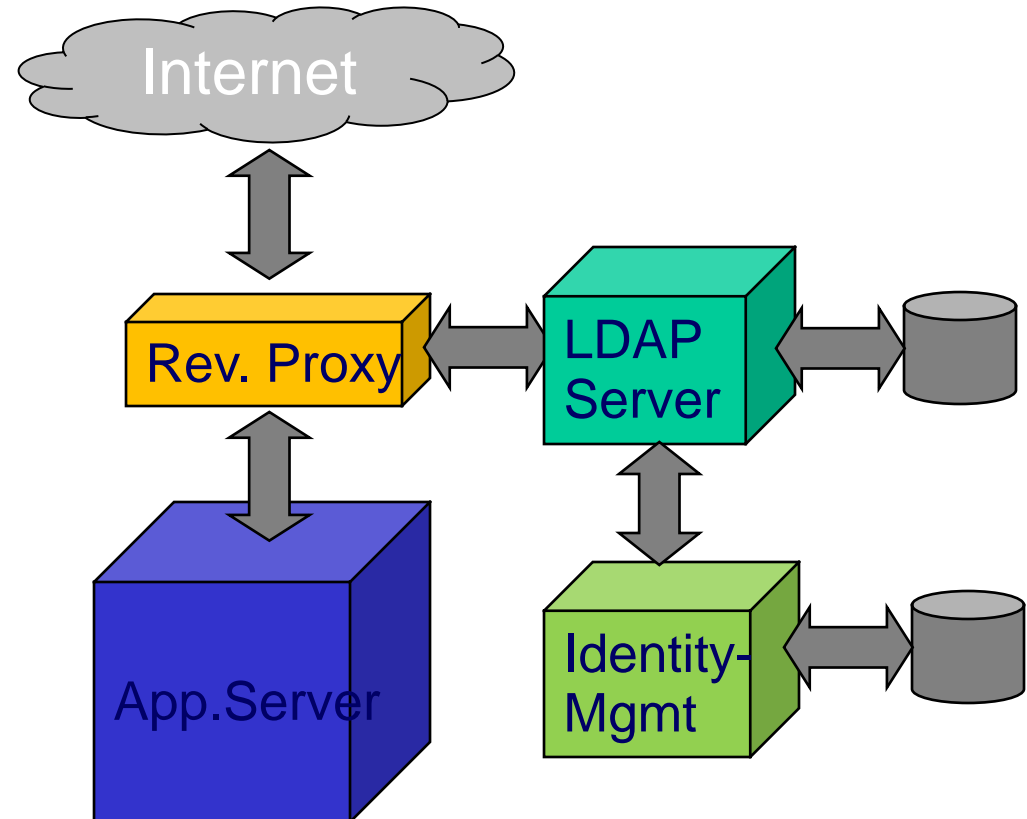
- **Ggf. weitere Server zur Überprüfung der Identitäten**

- **z.B. RADIUS oder zur Überprüfung von SecureID-Tokens**



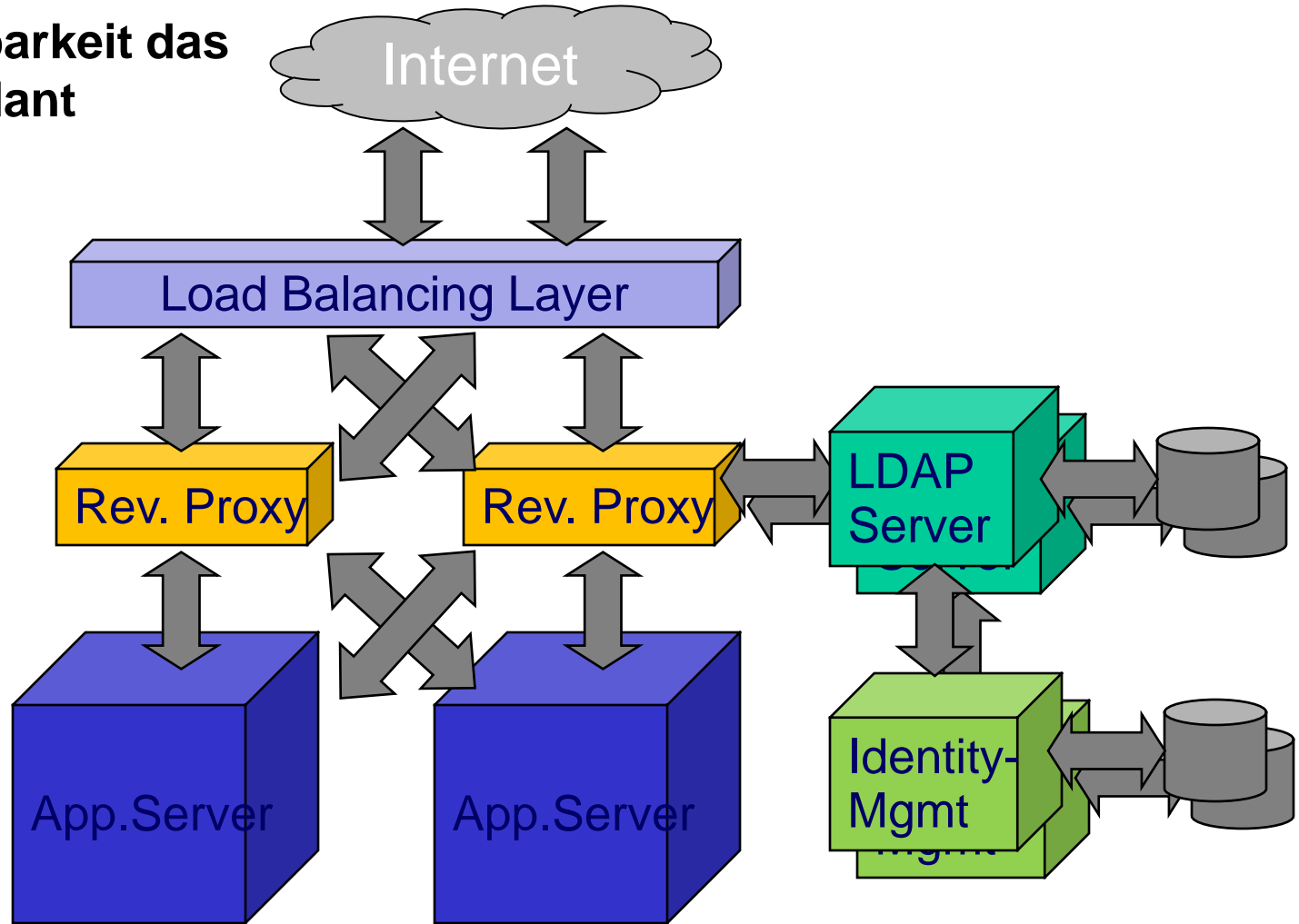
# Enterprise-Architektur für gesicherte Web-Services (3)

- **Identity-Management zur Verwaltung**
  - **der Nutzer-Identitäten**
    - Ein Nutzer kann verschiedene Accounts haben
    - Konsistentes Identity-Life-Cycle-Management
    - Erstellung der Credentials
  - **Der Nutzer-Rollen und ihrer Berechtigungen**
    - Role-Based-Access-Controll
- **z.B. IBM Tivoli oder MS Forefront Identity Manager**



## Enterprise-Architektur für gesicherte Web-Services (4)

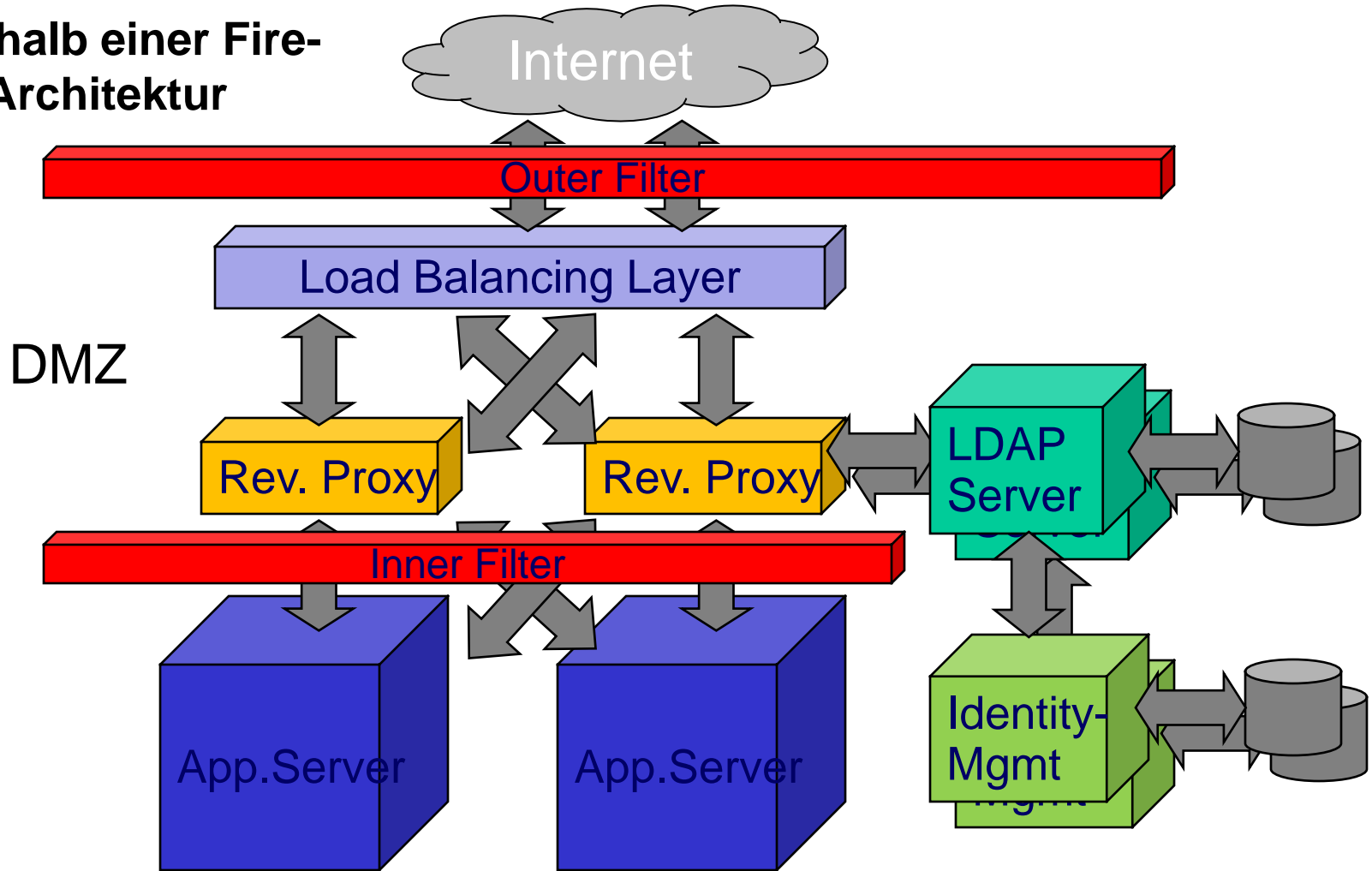
- Wg. Verfügbarkeit das alles redundant





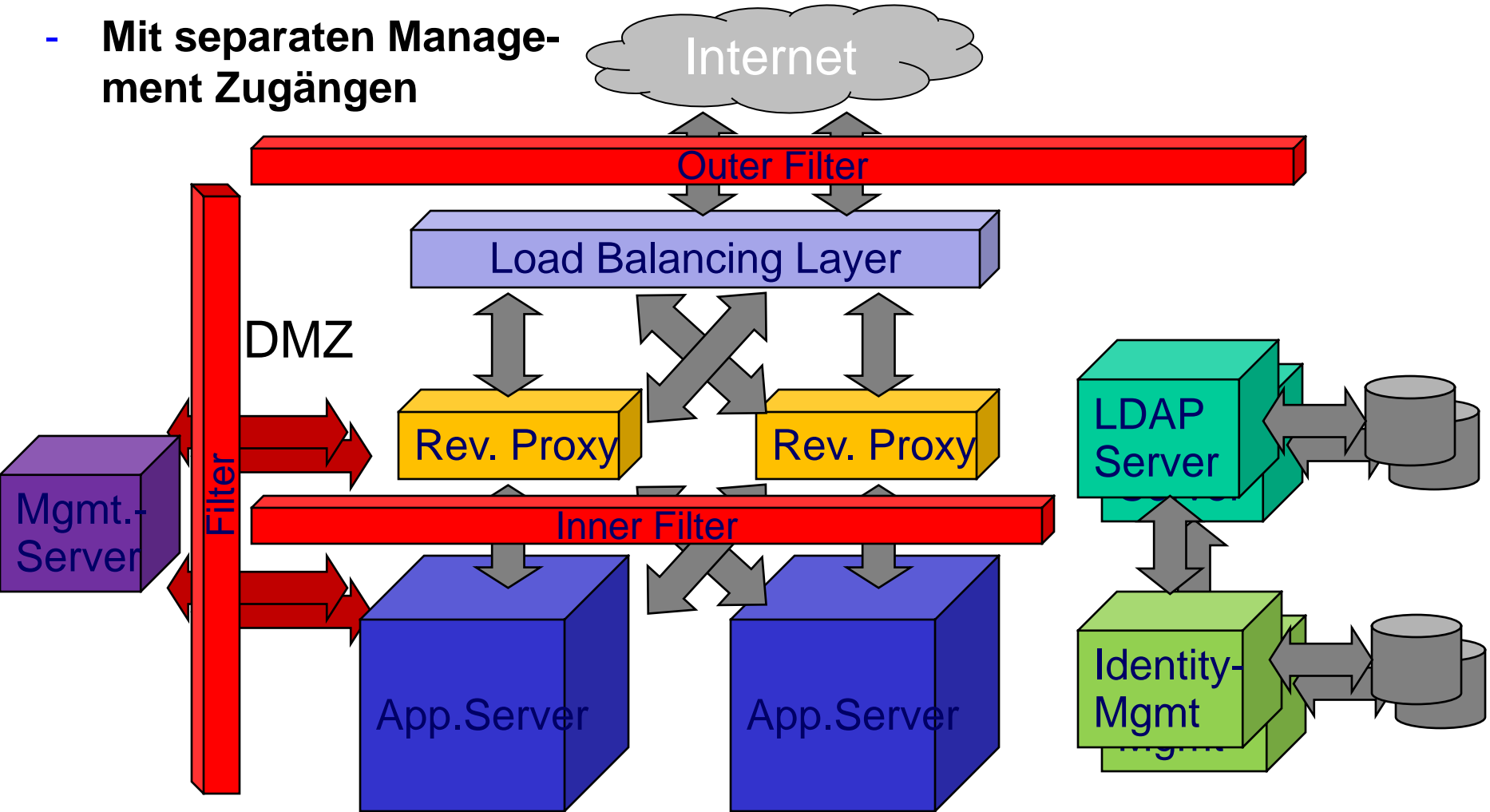
# Enterprise-Architektur für gesicherte Web-Services (5)

- Innerhalb einer Firewall-Architektur



# Enterprise-Architektur für gesicherte Web-Services (6)

- Mit separaten Management Zugängen



# Penetrationstest

---

## ◆ Umfassenden Sicherheitstest

- für einzelner Rechner oder ganze Netzwerke
- Nutzt Mittel und Methoden, die ein Angreifer anwenden würde
- ermittelt die Empfindlichkeit des zu testenden Systems gegen derartige Angriffe

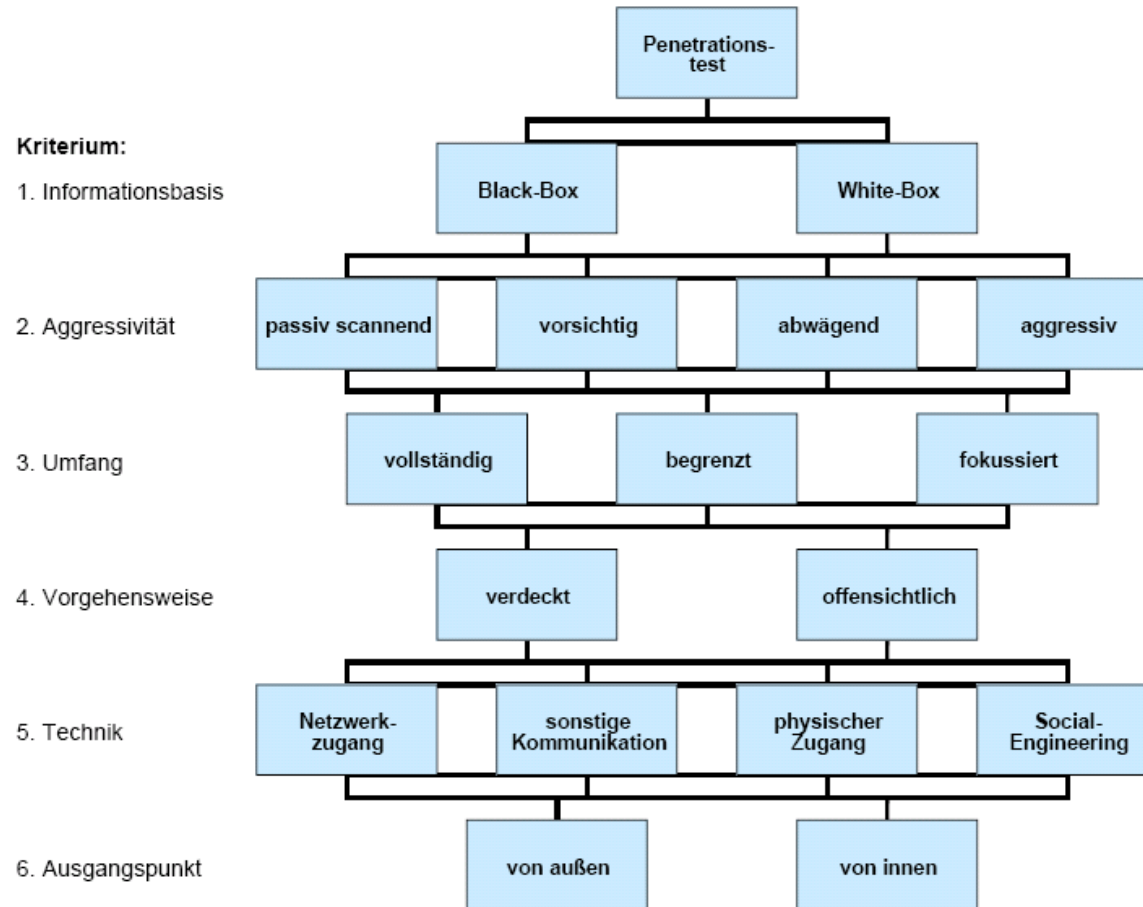
## ◆ Ziele eines Penetrationstests

- Identifikation von Schwachstellen
- Aufdecken potentieller Fehler, die sich aus der (fehlerhaften) Bedienung ergeben
- Erhöhung der Sicherheit auf technischer und organisatorischer Ebene
- Bestätigung der IT-Sicherheit durch einen externen Dritten

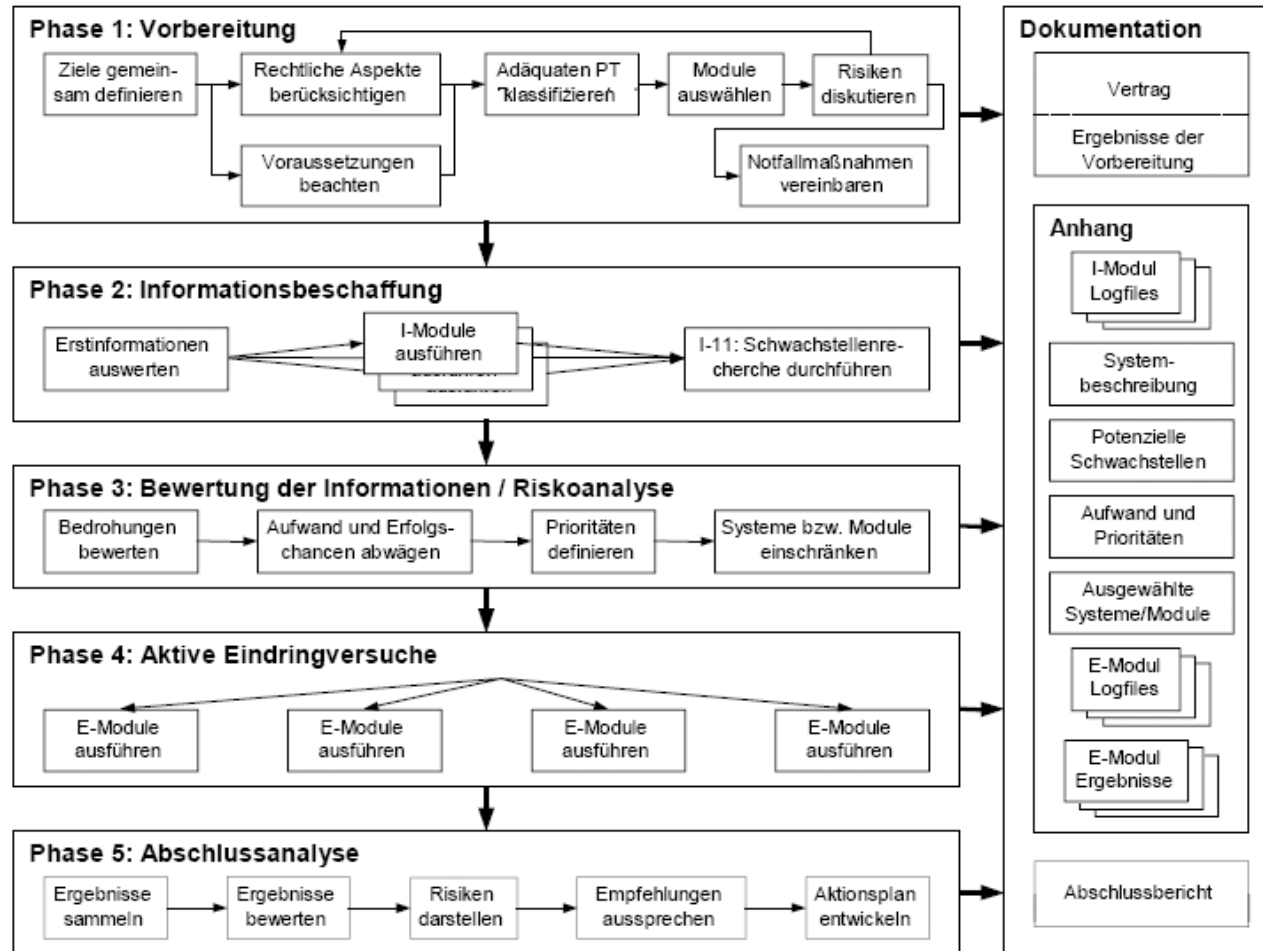
## ◆ Literatur

- A Penetration Testing Model, BSI 2014
  - NIST SP 800-42: Guideline on Network Security Testing
-

# Klassifikation von Pen.-Tests (nach BSI)



# Phasen eines Pen.-Tests (nach BSI)



# Mögliche Werkzeuge

---

## ◆ Phase 2:

- **Informationsbeschaffung**
  - Google (etc.)
  - nslookup (Adressinformationen)
- **Scanner**
  - nmap (Portscanner)
  - nessus (Schwachstellenscanner)
  - nikto (Webserver-Schwachstellenscanner)

## ◆ Phase 4:

- **Client-Web-Proxy**
  - Paros Proxy (zum Analysieren/Manipulieren von HTML-Requests)
- **Password-Cracker**
  - John the Ripper, Brutus (Brute Force und Defaults)

## ◆ Weitere Tools verfügbar in Linux-Distributionen BackTrack oder Knopix Security Tool Distribution (STD)

---