IT-Sicherheit

- Sicherheit vernetzter Systeme -

Priv.-Doz. Dr. Helmut Reiser
Dr. Wolfgang Hommel

Zeit: Montags, 15:15 – 17:45

Ort: Geschwister-Scholl-Platz 1,
Hörsaal A 014

Inhaltsübersicht

1. Einleitung
   - Internet Worm versus Slammer
2. Grundlagen
   - OSI Security Architecture und Sicherheitsmanagement
   - Begriffsbildung
   - Security versus Safety
3. Security Engineering
   - Vorgehensmodell: Bedrohungs-/ Risikoanalyse
   - Sicherheitsprobleme: Handelnde Personen, Notationen
   - Bedrohungen (Threats), Angriffe (Attacks), Schwächen (Vulnerabilities), z.B.:
     - Denial of Service
     - Malicious Code
     - Hoax, SPAM
     - Mobile Code
     - Buffer Overflow
     - Account / Password Cracking
     - Hintertüren / Falltüren
     - Rootkits
     - Sniffer
     - Port Scanner
   - Rechtliche Regelung: StGB
   - Top Cyber Security Risks
   - Sicherheitsanforderungen
4. Kryptologie, Grundlagen
   - Terminologie, Notationen
   - Steganographie
   - Kryptographie, Begriffe und Definitionen
   - Kryptoanalyse
5. Symmetrische Kryptosysteme
   - Data Encryption Standard (DES)
   - Advanced Encryption Std. (AES)

Innhaltsübersicht (2)

6. Asymmetrische und Hybride Kryptosysteme
   - RSA
   - Schlüssellängen und Schlüsselsicherheit
   - Hybride Systeme
   - Digitale Signatur
7. Kryptographische Hash Funktionen
   - Konstruktion von Hash-Fkt.
   - Angriffe auf Hash-Fkt.
   - MD4, MD5
   - Whirlpool Hashing
8. Sicherheitsmechanismen
   - Vertraulichkeit
   - Integrität
   - Identifikation
   - Authentisierung
   - Autorisierung und Zugriffskontrolle
9. Netz Sicherheit - Schicht 2: Data Link Layer
   - Point-to-Point Protocol (PPP)
   - Point-to-Point Tunneling Protocol (PPTP)
   - Layer 2 Tunneling Protocol (L2TP)
   - IEEE 802.1x
10. Schicht 2: WLAN Sicherheit
    - WEP
    - WPA
    - WPA2

Innhaltsübersicht (3)

11. Schicht 3: Network Layer
    - IP Gefahren und Schwächen
    - IPSec
    - Schlüsselverteilung mit IKE
12. Schicht 4 - Transport Layer
    - TCP / UDP
    - Secure Socket Layer / Transport Layer Security (SSL/TLS)
13. Schicht 7: Secure Shell (ssh)
    - SSH v1 versus SSH v2
    - Protokoll-Architektur
14. Firewalls und Intrusion Detection Systeme
    - Firewall-Klassen
    - Firewall-Architekturen
    - IDS-Arten
15. Beispiele aus der Praxis (LRZ)
15. Anti-Spam Maßnahmen
16. Beispiele aus der Praxis des LRZ
    - Struktur des MWN
    - Virtuelle Firewalls
    - Nat-O-Mat
    - NYX
17. Datenschutz
    - Persönlichkeitsrechte u. Datenschutz
    - Datenspuren und Schutzmaßnahmen

◉ Was ist nicht Gegenstand dieser Vorlesung
- Fortgeschrittenen krytographische Konzepte ⇒ Vorlesung Kryptologie
- Formale Sicherheitsmodelle und Sicherheitsbeweise

Einordnung der Vorlesung

- Bereich
  - Systemnahe und technische Informatik (ST), Anwendungen der Informatik (A)
- Hörerkreis (LMU)
  - Informatik Diplom
  - Informatik Master
  - Informatik Bachelor („Vertiefende Themen der Informatik für Bachelor“)
- Voraussetzungen
  - Grundlegende Kenntnisse der Informatik
  - Rechnernetze (wünschenswert und hilfreich)
- Relevanz für Hauptdiplomsprüfung
  - Vorlesung plus Übung: 3 + 2 SWS
  - Credits: 6 ECTS Punkte

Termine und Organisation

- Vorlesungstermine und Raum:
  - Montags von 15:15 – 17:45, Raum A 014 (Geschwister-Scholl Pl. 1)
- Übung; Beginn 04.11.2009
  - Mittwochs von 14:15 - 15:45 in E 006 (Geschwister-Scholl Pl. 1)
  - Übungsleitung:
    Stefan Metzger, metzger@lrz.de
    Tobias Lindinger, lindinge@nm.ifi.lmu.de
- Skript:
  - Kopien der Folien (pdf) zum Dowload
  - http://www.nm.ifi.lmu.de/itsec
- Kontakt:

Helmut Reiser	Wolfgang Hommel
reiser@lrz.de	hommel@lrz.de
LRZ, Raum I.2.070	LRZ, Raum I.1.107

- Sprechstunde:
  Montags 11:00 bis 12:00 im LRZ; nach der Vorlesung oder nach Vereinbarung

Schein

- Anmeldung zur **Übung** und Klausur
- Prüfung zum Erhalt des Scheins
- Notenbonus durch Hausaufgaben
  - Übungsblatt enthält Hausaufgabe
  - Hausaufgabe bei der Übung abgeben
  - Es werden 4 Blätter / Aufgaben gewählt und korrigiert

Anzahl korrekter Lösungen	Bonus	Beispiel
4	2 Stufen	Vorher: 3.0; Nachher: 2.3
2 oder 3	1 Stufe	Vorher: 3.0; Nachher: 2.7
1	0 Stufen	Vorher: 3.0; Nachher: 3.0

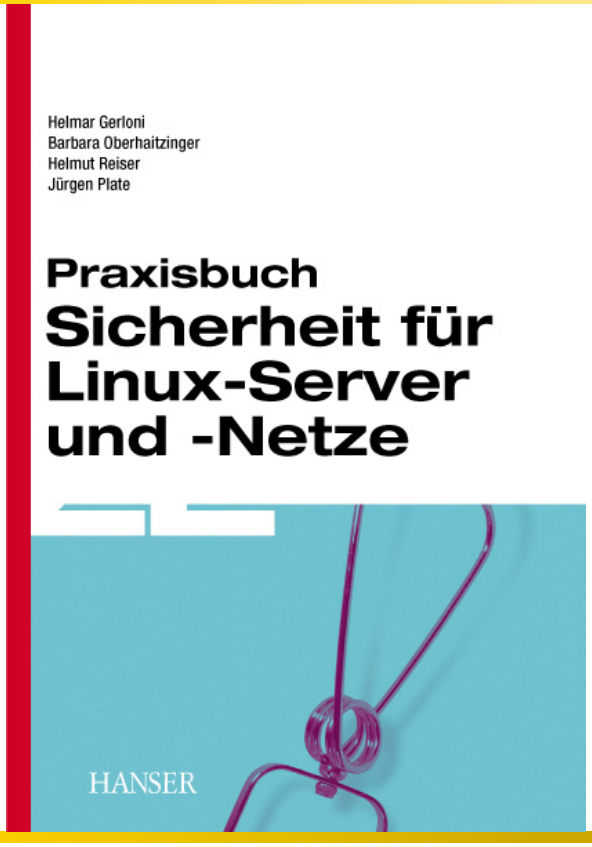
- Bonussystem nur wirksam bei **bestandener** Prüfung
- Beste Note 1.0

Literatur: IT-Sicherheit



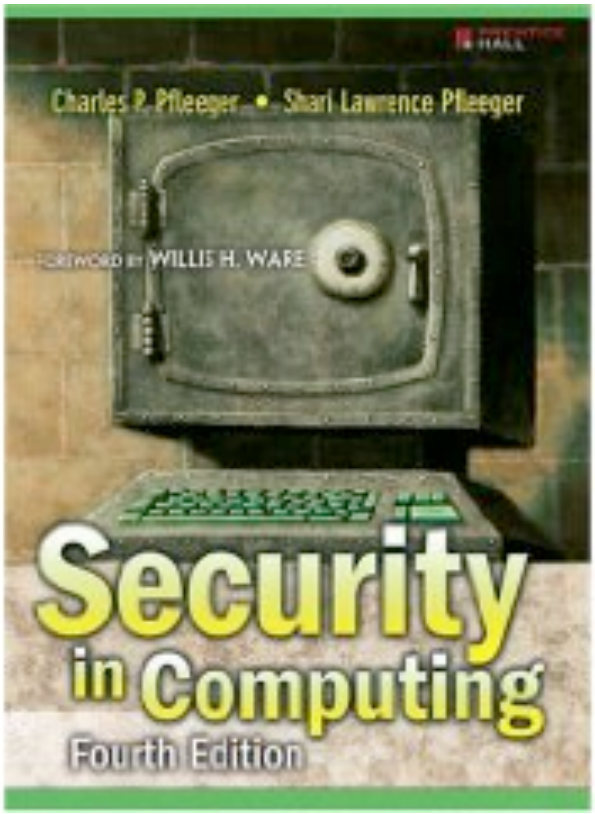
- Claudia Eckert
  IT-Sicherheit
  5. Auflage,
  Oldenbourg-Verlag, 2007
  ISBN 3486578510
  59,80 €

Literatur: IT-Sicherheit



- Helmar Gerloni, Barbara Oberhaitzinger, Helmut Reiser, Jürgen Plate
  Praxisbuch Sicherheit für Linux-Server und -Netze
  Hanser-Verlag, 2004
  ISBN 3-446-22626-5
  34,90 €

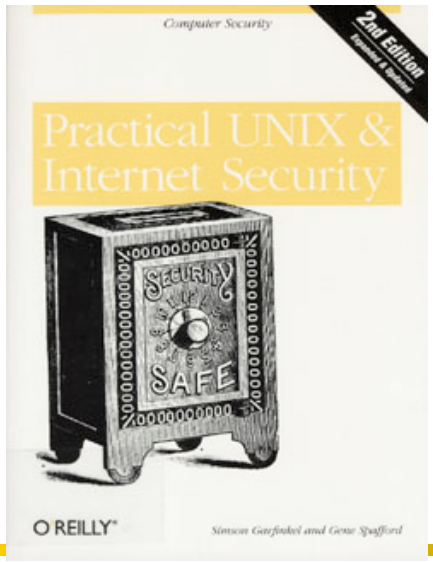
Literatur: IT-Sicherheit



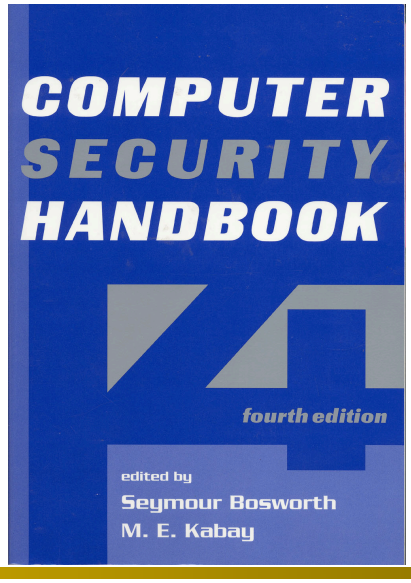
- Charles P. Pfleeger, Sharie L. Pfleeger
  Security in Computing
  4. Auflage,
  Pearson, 2006 / 2008
  ISBN 978-8120334151
  70 $

Literatur: IT-Sicherheit

- Simson Garfikel, Gene Spafford
  Practical Unix & Internet Security
  O'Reilly, 2003
  ISBN 0596003234
  ca. 50 €

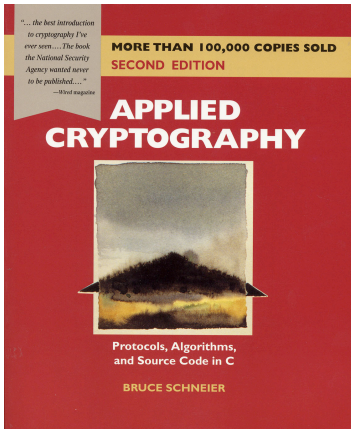


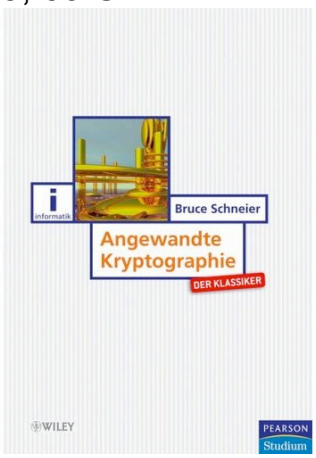
- Seymour Bosworth, M.E. Kabay
  Computer Security Handbook
  John Willey & Sons, 2003
  ISBN 0-471-41258-9
  ca. 90 – 100 €



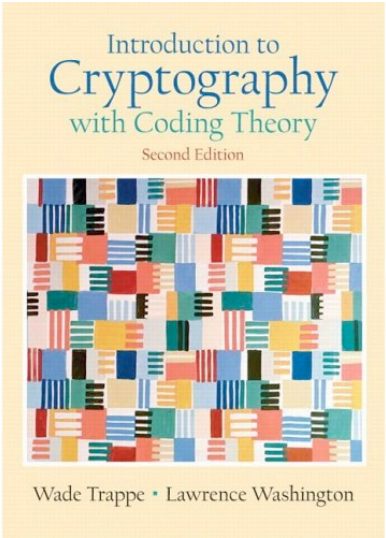
Literatur: Kryptologie

- Bruce Schneier
  Applied Cryptography
  John Willey & Sons, 1996
  ISBN 0-471-11709-9
  69 €
  Angewandte Kryptographie
  Pearson Studium, 2005
  ISBN 3827372283, 60 €



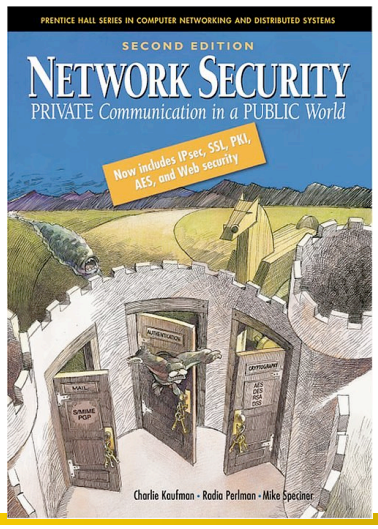


- Wade Trappe, Lawrence C. Washington
  Introduction to Cryptography with Coding Theory
  Prentice Hall, 2005
  ISBN 978-0131862395
  83 €

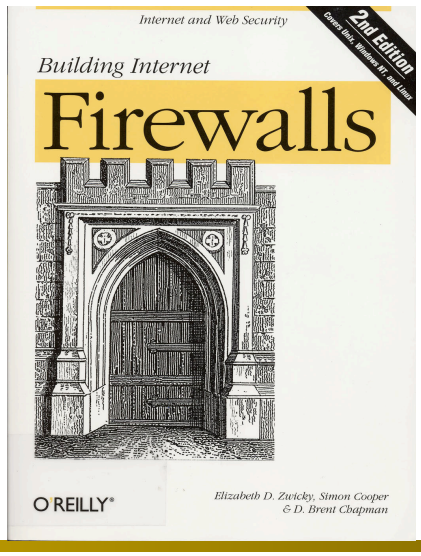


Literatur: Firewalls, Netzsicherheit

- Charly Kaufman, Radia Perlman, Mike Speciner
  Network Security, 2nd Ed.
  Prentice Hall, 2002
  ISBN 0-13-046019-2
  ca. 54 €



- Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman
  Building Internet Firewalls
  O'Reilly, 2002
  ISBN 1-56592-871-7
  ca. 50 €



Literaturliste

- Eine umfangreichere Literaturliste wird im Web zur Verfügung gestellt:

  www.nm.ifi.lmu.de/itsec

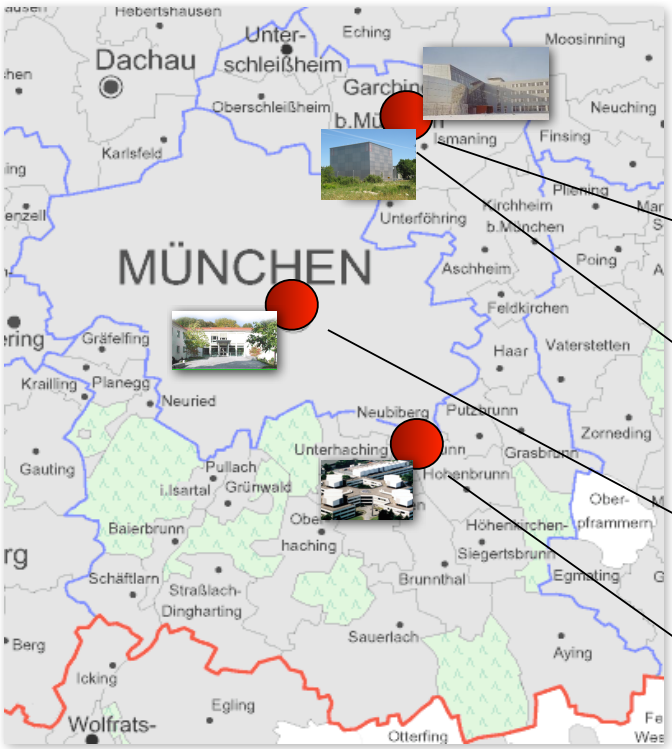
Weitere Veranstaltungen in diesem Semester

- Vorlesungen:
  - Grid Computing (Prof. Dr. Kranzlmüller, Dr. M. Schiffers)
    Mittwochs 9:00 – 12:00, LMU Hauptgebäude A017
    www.nm.ifi.lmu.de/grid/
  - IT-Management (Prof. Dr. Kranzlmüller, Dr. T. Schaaf)
    Donnerstags 14:00-17:00; 2. Semesterhälfte Dienstags 14:00-16:00
    www.nm.ifi.lmu.de/itmgmt/
- Praktika:
  - Rechnernetze (Prof. Dr. Kranzlmüller, Dr. V. Danciu, F. Liu, M. Metzker)
    www.nm.ifi.lmu.de/rnp
- Kompaktseminar:
  - Prozessorientiertes IT-Service Management (R. Kuhlig, Dr. M. Brenner, Dr. T. Schaaf, Prof. Dr. Kranzlmüller),
    www.nm.ifi.lmu.de/itsm

Weitere Veranstaltungen in diesem Semester

- Diplomarbeiten:
  www.nm.ifi.lmu.de/teaching/Auschreibungen/Diplomarbeiten
- Fortgeschrittenenpraktika, Systementwicklungsprojekte
  www.nm.ifi.lmu.de/teaching/Ausschreibungen/Fopras

Forschung: MNM Team



MNM TEAM
MUNICH NETWORK MANAGEMENT TEAM

der Bundeswehr Universität München

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 1: Einleitung

Kap. 1: Einleitung: Motivierendes Beispiel

1. Internet Worm
   - Historischer Rückblick
   - Funktionsweise
   - Lessons Learned
2. SQL Slammer Wurm
   - Historischer Rückblick
   - Funktionsweise
   - Lessons Learned
3. Vergleich von Internet Worm und Slammer

Historischer Rückblick I: 1988 Internet Worm

- Chronologie der Vorfälle an der University of Utah:
  - Mittwoch 2. November 1988
    - 17:01:59: Test oder Start des Wurms
    - 17:04: Maschine an der Cornell University „befallen“
    - 20:49: Wurm infiziert VAX 8600 an der Univ. Utah (cs.utah.edu)
    - 21:09: Wurm versucht von VAX aus andere Maschinen zu infizieren
    - 21:21: Load (Anzahl der rechenbereiten Prozesse) von 5
    - 21:41: Load von 7
    - 22:01: Load von 16
    - 22:06: Es können keine Prozesse mehr gestartet werden, Benutzer können sich nicht mehr anmelden
    - 22:20: Systemadministrator terminiert den Wurm Prozess
    - 22:41: Der Wurm ist zurück; Load 27
    - 22:49: System shutdown, reboot
    - 23:21: Der Wurm ist zurück; Load 37

Internet Wurm: Globale Sicht

- Mittwoch 2. Nov. 1988
  - 17:01:59: Wurm Test oder Start
  - 21:00: Stanford University; ca. 2500 Unix Maschinen infiziert
  - 21:30: MIT infiziert
  - 22:54: University of Maryland
  - 23:00: University of California, Berkeley
  - 24:00: SRI International
- Donnerstag, 3. Nov. 1988
  - 2:00: Lawrence Livermore National Laboratory
  - 2:28: E-mail Warnung; erreicht aber die meisten nicht vor Samst. 5. Nov. 5:00
    - Wurm infiziert SUN und VAX
    - Beinhaltet DES Tabelle
    - Nutzt `.rhosts` und `host.equiv`
    - Speichert `X*` Dateien in `/tmp`
  - 5:58: Bug fix posting aus Berkeley:
    - Sendmail's `debug` Kommando deaktivieren
    - C Compiler umbenennen
    - Linker umbenennen
  - 8:00: Berkely entdeckt `finger` Mechanismus
  - 10:30: TV Teams am MIT
    - Ca. 10 % Infektionsrate am MIT (2000 Maschinen)
  - 11:00: Titel-Story in den Nachrichten:
    - Mehr als 6000 hosts im Internet infiziert (10 %)

Internet Wurm: „How it Works“

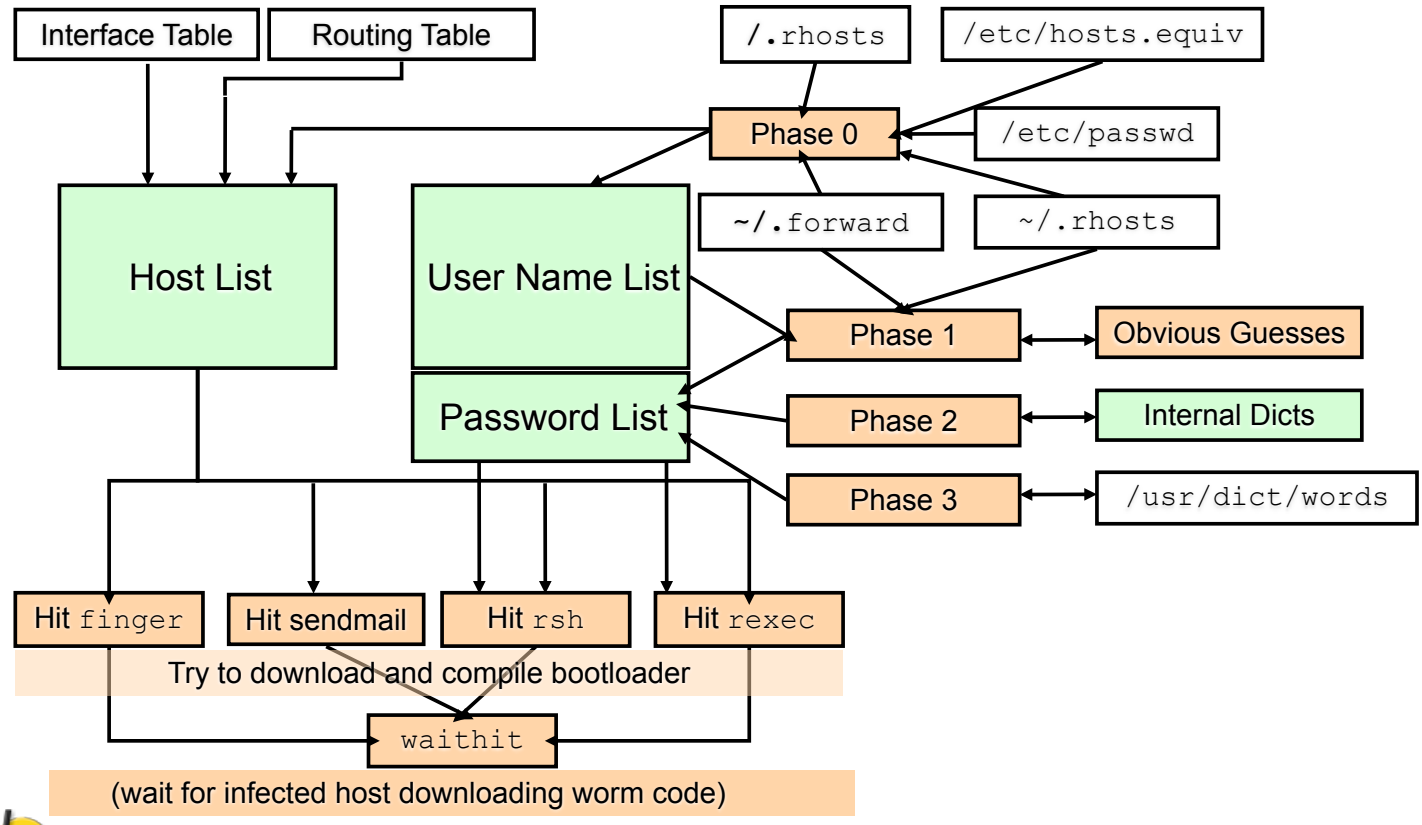
- Wie befällt er neue Maschinen?
  - `sendmail` Bug (seit langem bekannt)
  - `finger` Bug; Buffer Overflow (nur VAX werden befallen)
  - Remote execution (`rsh, rexec`)
- Welche Accounts werden angegr.
  - Offensichtliche Passwörter
    - Leeres Passwort
    - Benutzername
    - Benutzername+Benutzername
    - Infos aus GECOS-String
    - Nachname
    - Nachname rückwärts
  - Build-In Wörterbuch (432 Wörter)
  - `/usr/dict/words` (24’474 Wörter)
  - Trusted Host Beziehung (`.rhosts`)
- Welche hosts werden angegriffen?
  - Maschinen in `/.rhosts` und `/etc/host.equiv`
  - `.forward` Datei gebrochener Accounts
  - `.rhosts` Datei gebr. Accounts
  - Gateways aus der Routing-Tabelle
  - Endpunkte von Point to Point Verbindungen
  - Zufällig geratene Adressen
  - Nur Sun und VAX
- Was der Wurm NICHT tut:
  - Versuchen root access zu erhalten
  - Well-known Accounts angreifen
  - Daten zerstören
  - „Zeitbomben“ zurücklassen

Internet Wurm: Programm Struktur

`main` Routine

```
argv[0] := “sh”;      /* rename process */
Is there already a worm? /* faults here causes mass infection */
Initialize clock;
while (true) {
  cracksome(); /* attack accounts, try to find hosts */
  sleep(30);    /* hide the worm */
  Listen for other worms /* faults here causes mass infection */
  create a new process, kill the old  /* Camouflage */
  try to attack some machines;
  sleep(120); /* hide the worm */
  if (running > 12 hours)
          cleaning host List; /* reduce memory consumption */
  if (pleasequit && wordcheck > 10)
          exit
}
```

Internet Wurm: Attacking Engine

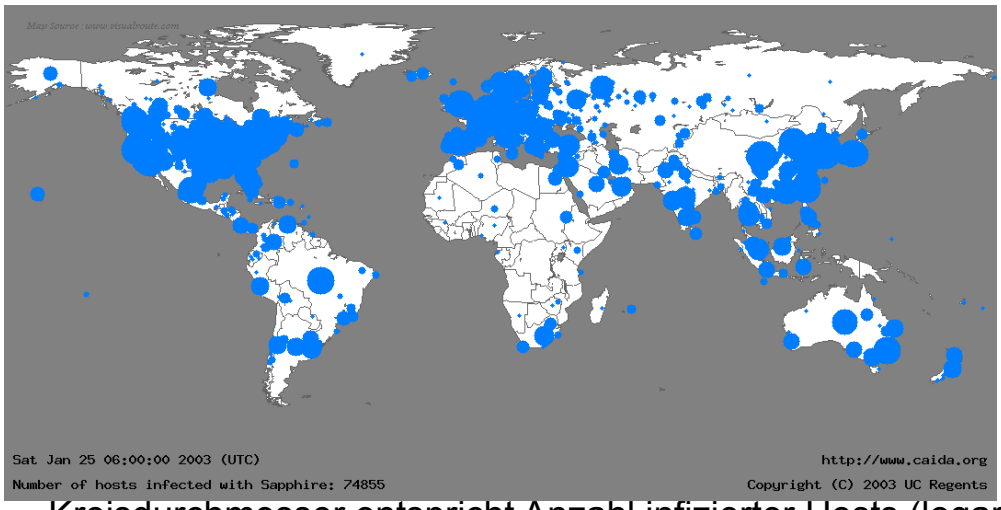


Internet Wurm: Lessons Learned

- Verursacher und rechtliche Folgen
  - Robert T. Morris, 23, Cornell Student (Sohn des NSA Chief Scientist)
  - Suspendierung von der Cornell University
  - Verurteilt zu $ 10.000 und 400 Stunden gemeinnütziger Arbeit
- Lessons Learned
  - (lange) bekannte Bugs fixen
  - Starke Passwörter benutzen
  - Least privilege Prinzip (sowenig Rechte wie nötig), strenge Zugriffskontrolle
  - Logging und Auditing
  - Keine reflexartigen Reaktionen
  - Kontinuierliche Information von sich und anderen
  - „Zentrales“ Security Repository
    - CERT (Computer Emergency Response Team) wurde gegründet www.cert.org

Historischer Rückblick II: 2003 Slammer Wurm

- Chronologie
  - Samstag, 25. Januar 2003: Kurz vor 5:30 Uhr (UTC), d.h. 6:30 Uhr (MEZ) taucht der Wurm auf
  - Verbreitung des Wurm um 6:00 Uhr (UTC):



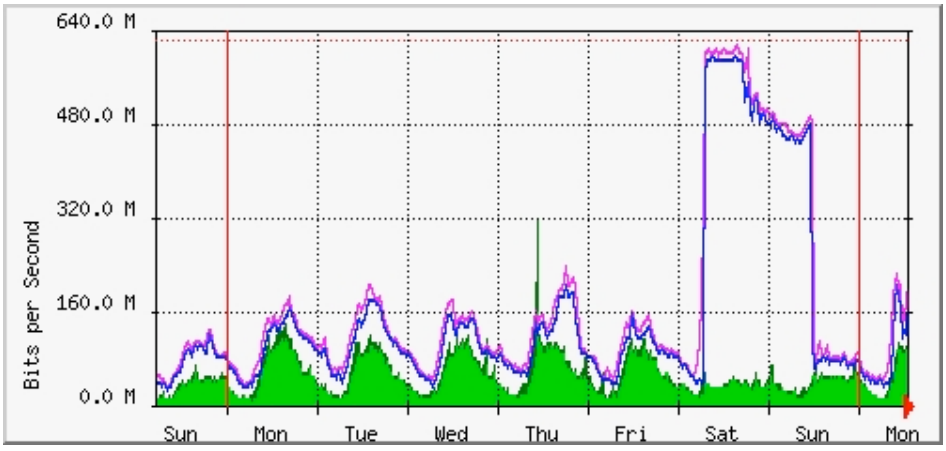
Quelle:

MPSS 03

Kreisdurchmesser entspricht Anzahl infizierter Hosts (logarithmische Darstellung)

SQL Slammer im Münchner Wissenschaftsnetz

- Münchner Wissenschaftsnetz (MWN), verbindet u.a. alle Standorte der Münchner Universitäten, der FH und der Bayerischen Akademie der Wissenschaften:
  Massive Störungen von Samstag 25.01.03 6:30 Uhr bis 26.01.03 11:30 Uhr
- Verkehrsstatistik am zentralen Router des MWN (1 Woche)



- Legende
  - Grün: eingehender Verkehr
  - Blau: ausgehender Verkehr
  - Dunkelgrün: Max. Peak im 5 Minuten Intervall (eingehend)
  - Magenta: Max. Peak im 5 Minuten Intervall (ausgehend)

Slammer Verbreitung und Folgen

- Schnellster Wurm in der Geschichte
  - 1. Minute: Verdopplung der Population alle 8,5 Sekunden (± 1 s)
  - > 3 Minuten: etwas verringerte Verbreitungsrate; Netzbandbreite wird zum beschränkenden Faktor
  - 10 Minuten: ca. 90 % aller anfälligen Hosts sind infiziert

- Folgen:
  - Große Teile des Internets nicht mehr erreichbar
  - Steuerungssysteme für die Stromversorgung gestört
  - Funktionsstörungen bei Geldautomaten
  - Steuerrechner von zwei Atomkraftwerken in den USA betroffen
  - .....

Slammer: Voraussetzungen

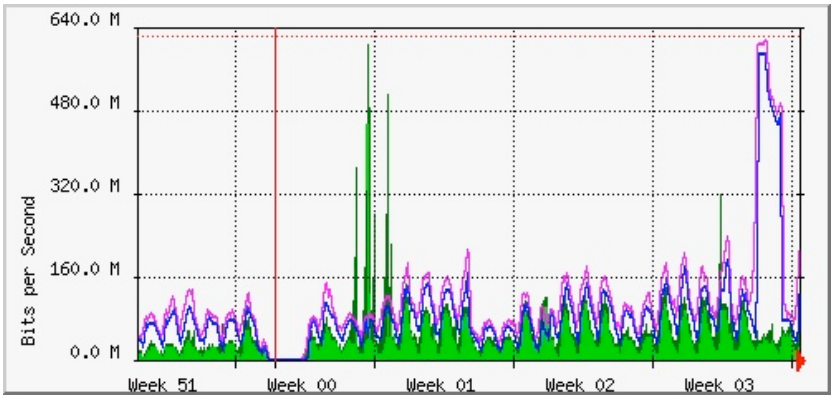
- SQL Server; Client Verbindungen über
  - NetBios (TCP Port 139/445)
  - Sockets (TCP Port 1433)
  - Monitor Port (UDP 1434) zur Ermittlung der Verbindungsart; Client schickt 0x02 an den Port; Server schickt Verbindungsinformationen
- Buffer Overflow Bug im SQL Server
  - Client setzt erstes Bit auf 0x04
    im Bsp. `\x04\x41\x41\x41\x41 (\x41` = „A“)
  - SQL Monitor nimmt Rest der Daten und öffnet damit Registry
    `HKLM\Software\Microsoft\Microsoft SQL Server\AAAA \MSSQLServer\CurrentVersion`
  - Über geeignet formatierte Daten kann hier ein Buffer Overflow herbeigeführt werden
- Problem:
  - SW von Drittanbietern beinhaltet SQL-Server
  - Dies ist nicht allgemein bekannt

Slammer: How it works

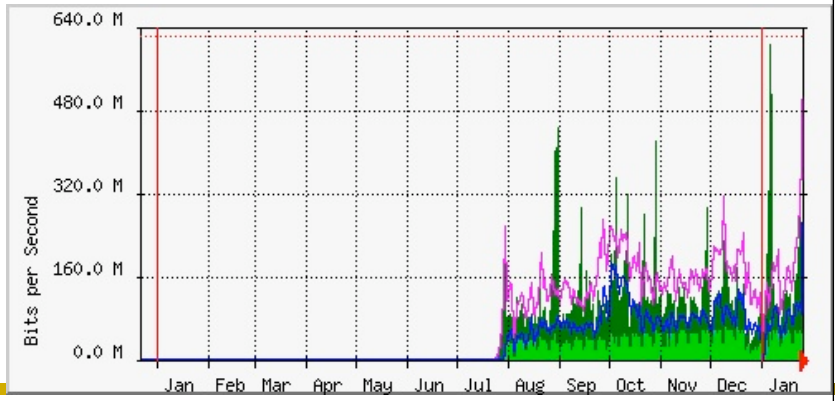
- Slammer passt in ein UDP Packet
  - 376 Byte groß, geschrieben in Assembler
  - Mit Header Informationen 404 Byte
- Slammer nutzt Buffer-Overflow an UDP Port 1434
- Nach Infektion:
  - „Raten“ zufälliger IP-Adressen
  - Angriff über UDP
- Keine Schadfunktionalität im eigentlichen Sinn
- Charakteristika:
  - UDP verbindungsloses Protokoll; wird nur durch Bandbreite beschränkt
  - Höchste beobachtete „Probing“-Rate: 26.000 Scans pro Sekunde
  - Aggressive Verbreitungsstrategie führt dazu, dass der Wurm mit anderen Würmern um Netzbandbreite konkurriert

Slammer im MWN (Forts.)

- Monatsstatistik



- Mind. 8 SQL-Server betroffen
- Maßnahmen:
  - Zugang zum MWN für diese Server gesperrt
  - Port 1434 gesperrt



Slammer: Lessons Learned

- Grundproblematik:
  Nicht behobenen Bugs in Anwendungen (kein Einspielen von Patches)
- Bundling von Software; Anwender weiß u.U. nichts von Sicherheitsproblemen und notwendigen Patches
- Angriffe über UDP können zu extrem schneller Verbreitung führen
- Gegenmaßnahmen:
  - Filtern des entsprechenden Verkehrs (UDP Port 1434) über Firewall
  - Fehler und Schwächen beheben
  - Nicht notwendige Dienste abschalten

Vergleich Internet Worm und ILOVEYOU

	Internet Worm	**Slammer**
Angegriffene Hosts/OS	SUN und VAX / UNIX	Microsoft Windows/SQL Server
Angriffsstrategie	Ziemlich komplex Nutzt eine Vielzahl von Bugs und fortschrittliche Strategien	Einfaches Assembler Programm nutzt Buffer Overflow
Schadfunktion	Verursacht große Load und viel Netzverkehr	Verursacht extremste Load und Netzverkehr
Verbreitung	~ 6.000 Systeme Ziemlich schnell	Extrem schnell 90 % aller verwundbaren Systeme nach 10 Minuten infiziert

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 2: Grundlagen

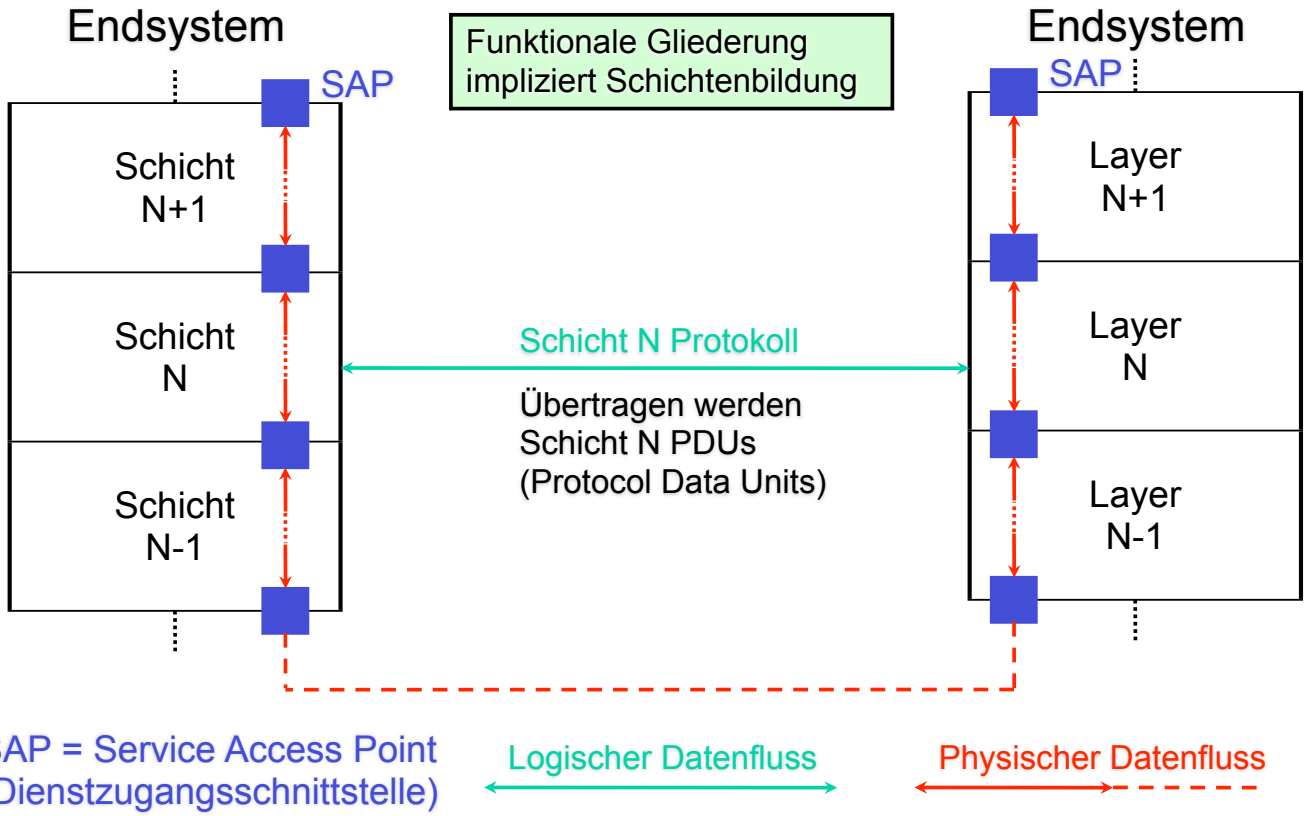
Kapitel 2: Inhalt

1. Überblick über die OSI-Sicherheitsarchitektur
2. ISO/OSI Referenzmodell
3. OSI Sicherheitsarchitektur
   1. Sicherheitsdienste
   2. Sicherheitsmechanismen
4. OSI Sicherheitsmanagement
5. Unterscheidung Security versus Safety

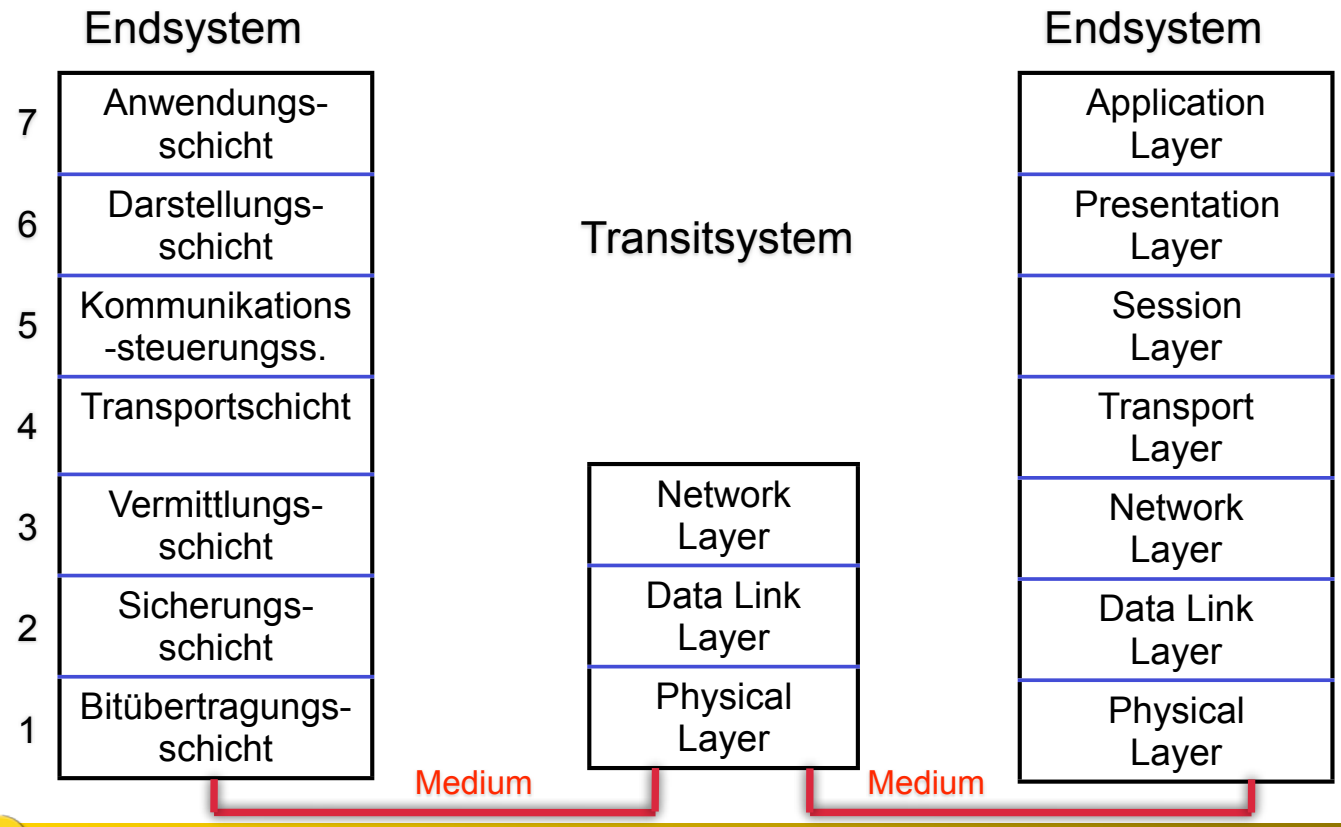
OSI Security Architecture: Überblick

- Standardisiert von der International Standardization Organization (ISO) 1988 und der International Telecommunication Union (ITU) 1991
- Dokumente:
  - ISO: ISO-7498-2; IS0-10181-1 bis –7 (Security Framework); ISO-11586-1 bis –6 (Upper Layer Security)
  - ITU: ITU-T X.800 – X.830
- Fokus liegt auf verteilten / vernetzten Systemen
- Beschreibung von Sicherheitsdiensten (Security Services), Sicherheitsmechanismen,.....
- Baut auf dem Open System Interconnection Reference Model (ISO/OSI-RM) auf

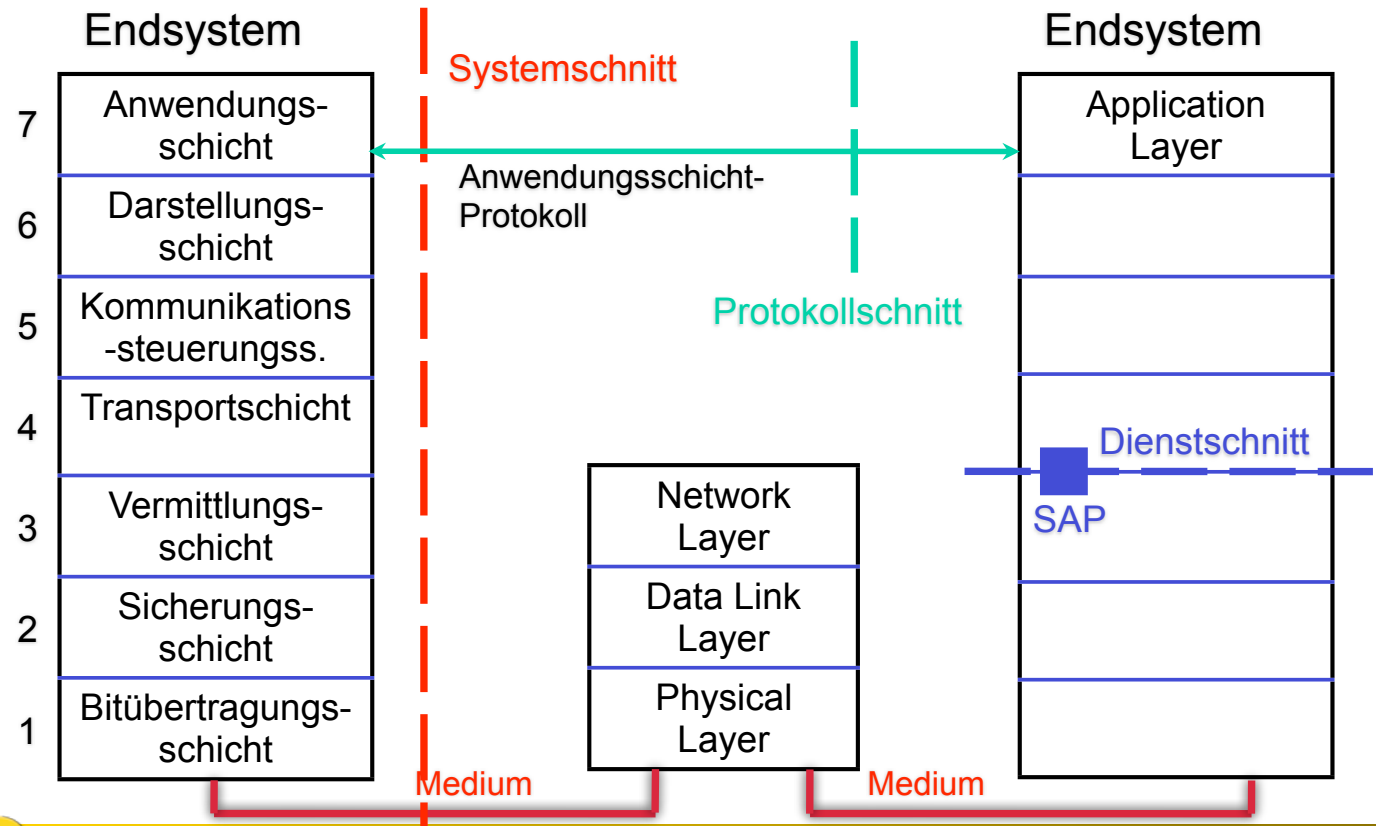
Prinzip des OSI-Referenzmodell



OSI Referenzmodell: Schichten



OSI Refenzmodell: Schnittbildung



OSI Security Architecture: Überblick

- Beschreibung von Sicherheitsdiensten (Security Services) und Sicherheitsmechanismen
- Beziehungen zwischen Services, Mechanismen und den Schichten
- Platzierung von Services und Mechanismen
- Security Management
- Hintergrundinformation
  - Bedrohungen und Angriffe
  - Security Policy
  - Grundlegende Mechanismen
- Fokus der Sicherheitsarchitektur
  - Sicherheitsbedürfnisse von verteilten / vernetzten Systemen
  - Betrachtet **keine** Host- oder Betriebssystem-Sicherheit

OSI Security Architecture: Dienste

- **Authentisierung (Authentication):**
  Jede Entität kann zweifelsfrei identifiziert werden
  - Peer Entity Authentication: Gegenseitige Authentisierung von zwei oder mehr Kommunikations-partnern
  - Data Origin Authentication: Identifikation des Senders bzw. des Autors einer Nachricht
- **Zugriffskontrolle (Access Control):**
  Schutz vor unberechtigter Nutzung von Ressourcen
- **Vertraulichkeit (Data Confidentiality):**
  Schutz der Daten vor unberechtigter Offenlegung
  - Connection confidentiality: Alle (N-) User Daten einer (N-) Verbindung
  - Connectionless confidentiality: Alle (N-) User Daten einer einzelnen (N-) SDU (Service Data Unit)
  - Selective field confidentiality: Bestimmte Felder der User Daten
  - Traffic flow confidentiality: Schutz vor Verkehrsflussanalyse. (Wer kommuniziert mit wem in welchem Umfang und zu welcher Zeit?)

OSI Security Architecture: Dienste (Forts.)

- **Datenintegrität (Data Integrity):**
  Erkennung von Modifikationen, Einfügungen, Löschungen, Umordnung, Duplikaten oder Wiedereinspielung von Daten
  - Connection Integrity with/without Recovery
  - Selective Field Connection Integrity
  - Connectionless Integrity
  - Selective Field Connectionless Integrity
- **Verbindlichkeit (Non-repudiation):**
  Niemand kann das Senden oder Empfangen der Daten leugnen
  - With proof of origin:
    Sender kann das Senden nicht leugnen; Empfänger kann beweisen welchen Ursprung die Daten haben
  - With proof of delivery:
    Empfänger kann Empfang nicht leugnen; Sender kann die Auslieferung beweisen

Fokus der Sicherheitsdienste

Authentication	Peer Entity
	Data Origin
Access Control	
Data Confidentiality	Connection
	Connectionless
	Selective field
	Traffic flow
Data Integrity	Connection
	Connectionless
	Selective field
	Recovery
Non-Repudiation	Proof of origin
	Proof of delivery

- Betrachtet werden **keine** Host- oder Betriebssystem-Sicherheit
- Fehlende bzw. nicht explizit spezifizierte Security Services:
  - **Identifikation (Identification);** Personalisierung: Zweifelsfreie Verbindung zwischen digitaler ID und Real World Entity (Person oder Organization)
  - **Autorisierung (Authorization):** Erteilung von Rechten an Entities
  - **Zurechenbarkeit (Accountability)**
  - **Anonymität (Anonymity)** (außer bei Traffic Flow Confidentiality)
  - (Verfügbarkeit (Availability))
  - **Ressourcenbeschränkung (Ressource constraints)**

OSI Security Architecture: Mechanismen

- Unterscheidung in:
  - spezifische (specific)
  - durchgängige (pervasive)

  Sicherheitsmechanismen

- Specific Security Mechanisms
  - Verschlüsselung (Vertraulichkeit)
    - Symmetrisch
    - Asymmetrisch
  - Digitale Signatur (Verbindlichkeit)
  - Zugriffskontrolle
    - Zugriffskontrolllisten, -matrizen
    - “Wissen und/oder Besitz”
    - Capabilities, Tickets
  - Prüfsummenverfahren (Integrität)
  - Notariatsfunktionen
  - Austausch von Authentisierungs-informationen; nutzt ggf.
    - Verschlüsselung und
    - Digitale Signatur
  - Traffic Padding, Anonymisierung (zur Verhinderung von Verkehrs-flussanalysen)
  - Kontrolle des Routing-Verfahrens

- Pervasive Security Mechanisms
  - Vertrauenswürdige Funktionen
  - Security Labels zur (Sensitivitäts-) Klassifikation der Daten
  - Eventmechanismen
  - Auditing und Logging
  - Recovery Mechanismen
    - Unmittelbar, kurz- und langfristig

Beziehung zwischen Service und Mechanismus

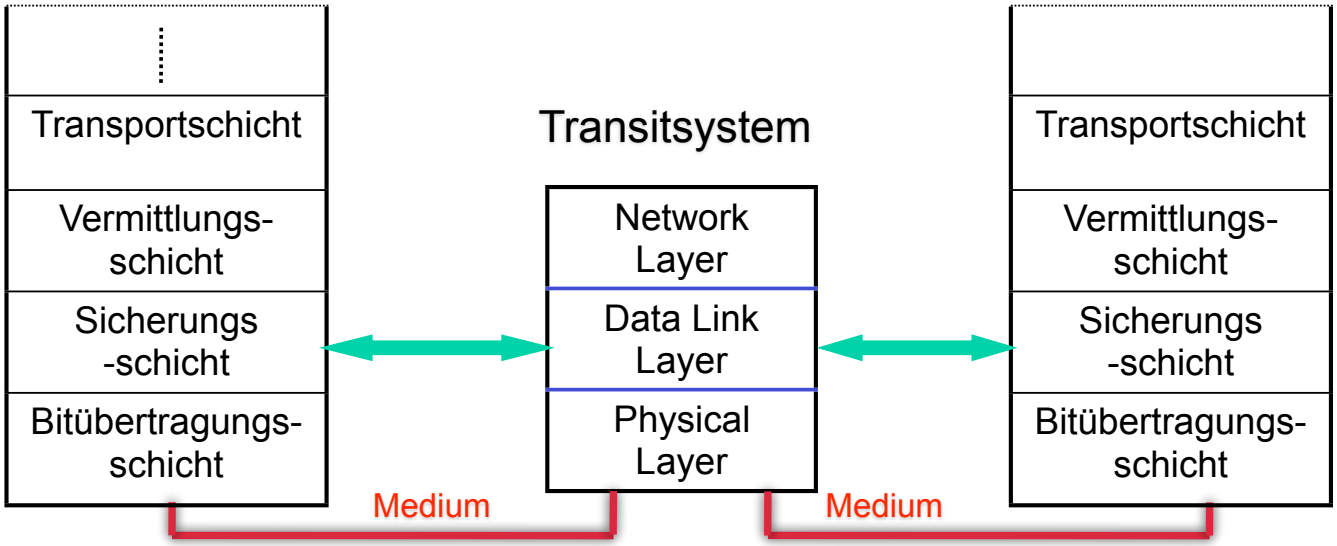
TABLE 1/X.80

Illustration of relationship of security services and mechanism

Mechanis / Servic	Enciphermen	Digita signatur	Acce contro	Dat integrit	Authenti catio exchange	Traffic paddin	Routin contro	Notari zatio
Peer entity authenticatio	Y	Y	·	·	Y	·	·	·
Data origi authenticatio	Y	Y	·	·	·	·	·	·
Access control servic	·	·	Y	·	·	·	·	·
Connection confidentiality	Y	·	·	·	·	·	Y	·
Connectionles confidentiality	Y	·	·	·	·	·	Y	·
Selective field confidentiality	Y	·	·	·	·	·	·	·
Traffic flow confidentiality	Y	·	·	·	·	Y	Y	·
Connection Integrity wit recover	Y	·	·	Y	·	·	·	·
Connection integrit without recover	Y	·	·	Y	·	·	·	·
Selective field connection integrit	Y	·	·	Y	·	·	·	·
Connectionless integrit	Y	Y	·	Y	·	·	·	·
Selective field connectionless integrit	Y	Y	·	Y	·	·	·	·
Non-repudiation. Origi	·	Y	·	Y	·	·	·	Y
Non-repudiation. Deliver	·	Y	·	Y	·	·	·	Y

Mechanismen auf unterschiedlichen Schichten

- Was ist die zu authentisierende/sichernde Entität?
- Wie weit reicht der Sicherheitsmechanismus?
  Bsp. Verschlüsselung auf Schicht 2 (Sicherungsschicht)
  d.h. jedes Transitsystem muss entschlüsseln



Vorschlag einer Zuordnung der Services auf Layer

TABLE 2/X.80

Illustration of the relationship of security services and layer

Servic	Laye						
	1	2	3	4	5	6	7
Peer entity authenticatio	·	·	Y	Y	·	·	Y
Data origen authenticatio	·	·	Y	Y	·	·	Y
Access control servic	·	·	Y	Y	·	·	Y
Connection confidentiality	Y	Y	Y	Y	·	Y	Y
Connectionless confidentiality	·	Y	Y	Y	·	Y	Y
Selective field confidentiality	·	·	·	·	·	Y	Y
Traffic flow confidentiality	Y	·	Y	·	·	·	Y
Connection Integrity with recover	·	·	·	Y	·	·	Y
Connection integrity without recover	·	·	Y	Y	·	·	Y
Selective field connection integrity	·	·	·	·	·	·	Y
Connectionless integrit	·	·	Y	Y	·	·	Y
Selective field connectionless integrity	·	·	·	·	·	·	Y
Non-repudiation Origi	·	·	·	·	·	·	Y
Non-repudiation. Deliver	·	·	·	·	·	·	Y

Y Yes, service should be incorporated in the standards for the layer as a provider option.

· Not provided.

* It should be noted, with respect to layer 7, that the application process may, itself, provid security services.

OSI Security Management

- OSI unterscheidet drei (Sicherheits-)Management-Kategorien:

1. **System Security Management**:
   - Management des gesamten verteilten (OSI-) Systems
   - **Nicht** das System im Sinne von Endsystem oder Betriebssystem

2. **Security Service Management:**
   - Management von dedizierten Sicherheitsdiensten

3. **Security Mechanism Management**:
   - Management von spezifischen Sicherheitsmechanismen

1. System Security Management Functions

- Policy Management
  - Aktualisierung
  - Konsistenzprüfung und Überwachung
  - Wartung
- Interaktion mit anderen OSI-Mgmt. Funktionen
- Event Handling
  - Reporting
  - Planung und Fortschreibung
  - Analyse
- Audit Management
  - Auswahl von zu loggenden Events
  - Aktivierung, Deaktivierung (entfernter) Logs
  - Sammlung und Auswertung
- Recovery Management
  - Erkennung und Report von Angriffen und Angriffsversuchen
  - (Reaktions-) Regeln für Administratoren

2. Security Service Management Functions

- Auswahl von Sicherheitsdiensten für jedes Ziel (Target) (Anforderungsanalyse)
- Auswahl, Wartung und Aktualisierung von Regeln / Policies für die gewählten Dienste
- Installation / Aktivierung entsprechender Mechanismen zur Realisierung der Dienste
- Aufruf der entsprechenden Funktionalität durch das Sicherheitsmanagement (-system)
- Interaktion mit anderen Management Funktionen

3. Security Mechanism Management Functions

- Schlüsselmanagement
  - Personalisierung
  - Schlüsselerzeugung
  - Schlüsselverteilung
  - Widerruf von Schlüsseln
- Kryptographie-Management
- Mgmt. der digitalen Signatur
- Zugriffskontrollmanagment
- Integritätsmanagement
- Authentisierungsmanagment

→
- Interaktion mit dem Schlüsselmanagment
- Einrichtung und Intialisierung der Verfahren
- Auswahl und Nutzung geeigneter Protokolle zwischen den Kommunikationspartnern (u. ggf. einer Trutsted Third Party (TTP) = Notar)
- Verteilung der Sicherheitsattribute und Informationen
- Wartung und Betrieb

3. Security Mechanism Mgmt. Functions (Forts.)

- Mgmt. der Notariatsfunktion
  - Verteilung von Information über Notare
  - Protokoll zur Kommunikation mit Notar
- Traffic Padding Management
  - Vorspezifizierte Datenraten
  - Zufällige Datenraten
  - Spezifikation der Nachrichtencharakteristika (z.B. Länge)
  - Variation der Spezifikation, z.B. in Abhängigkeit der Zeit
- Routing Control Management
  - Klassifikation der Links oder Subnetze nach Vertrauens-Level
  - Festlegung der Routing Verfahren entsprechend dieser Klassifikation

Unterscheidung von Security und Safety

- Beide Begriffe werden mit „Sicherheit“ übersetzt
- Security (Sicherheit)
  - Security Engineering
  - Security Policies
  - Sicherheitsanforderungen:
    Identifikation, Authentisierung, Autorisierung, Zugriffskontrolle,.....
  - Sicherheitsmechanismen realisieren Sicherheitsanforderungen
  - Auditing und Logging

  Verfügbarkeit (Availability) von Software und Hardware

- Safety (Betriebs-Sicherheit)
  - Verfügbarkeit (Availability) / Ausfallsicherheit (Reliability)
  - Betriebssicherheit für sicherheitskritische Programme, z.B., Steuerung und Überwachung von Flugzeugen oder (Atom-)Kraftwerken
  - Gesundheitliche Sicherheit / Ergonomie

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 3: Security Engineering

Inhalt (1)

1. Security Engineering – Vorgehensmodell
2. Sicherheitsprobleme: Handelnde Personen, Notation
3. Bedrohungen, Angriffe und Gefährdungen
   1. Denial of Service
   2. Distributed Denial of Service
   3. Malicious Code
      - Viren
      - Würmer
      - Trojanische Pferde
   4. Hoax und Spam
   5. Mobile Code
      - ActiveX
      - JavaScript
      - Java mit Ausblick auf die Java Sicherheitsarchitektur

Inhalt (2)

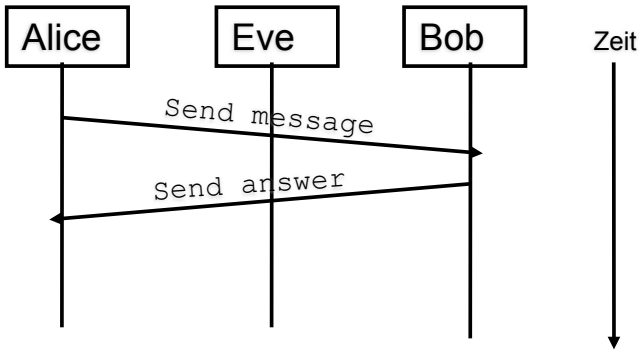
3. Bedrohungen, Angriffe und Gefährdungen (Forts.)
   6. Buffer Overflow
   7. Account / Password Cracking
   8. Hintertüren / Falltüren
   9. Rootkits
   10. Cross Site Scripting (XSS)
   11. Sniffer
   12. Port Scanner
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

Security Engineering: Vorgehensmodell

- Ziel: Sicheres System, sichere Infrastruktur
- Strukturiertes Vorgehen um das Ziel zu erreichen:
  1. **Bestandsaufnahme** der Ressourcen (Systeme, Dienste, Daten,.....)
  2. **Bedrohungsanalyse**: Welchen potentiellen Gefahren drohen diesen Ressourcen? Welche Angriffe x sind möglich?
  3. Bestimmung der Schadens-wahrscheinlichkeit E(x) für jeden möglichen Angriff
  4. Bestimmung der Schadenshöhe S (x)
  5. Errechnung des Risikos R(x) = E(x) * S(x)
  6. **Risiko-Priorisierung**
  7. Ableitung von **Sicherheitsanforderungen**
  8. Erstellung einer **Sicherheits-Policy**
  9. Auswahl von **Mechanismen** zur Durchsetzung der Sicherheitsanforderungen
  10. Implementierung und Betrieb
  11. Dokumentation
- Dieses Vorgehen ist **nicht** streng sequentiell
- Analog zu SW-Engineering Modellen sind Zyklen und Iterationen möglich
- Sec.Eng. ist fortwährender Prozess

Handelnde Personen

- Um Sicherheitsprobleme und -protokolle zu erläutern werden häufig die folgenden Personen verwendet
- Die „Guten“
  - **Alice (A)**
    Initiator eines Protokolls
  - **Bob (B)**
    antwortet auf Anfragen von Alice
  - **Carol (C) and Dave (D)**
    Teilnehmer in drei- oder vier-Wege Protokollen
  - **Trent (T)**
    Vertrauenswürdiger Dritter (Trusted arbitrator)
  - **Walter (W)**
    Wächter (Warden), bewacht A. und B.
- Die „Bösen“
  - **Eve (E)**
    Eavesdropper; Abhörender
  - **Mallory, Mallet (M)**
    Malicious attacker; Angreifer
- Bsp.: Abhören der Kommunikation zwischen A. und B. (UML Sequence Diagram)



Inhalt (1)

1. Security Engineering – Vorgehensmodell
2. Sicherheitsprobleme: Handelnde Personen, Notation
3. Bedrohungen, Angriffe und Gefährdungen
   1. Denial of Service
   2. Distributed Denial of Service
   3. Malicious Code
      - Viren
      - Würmer
      - Trojanische Pferde
   4. Hoax und Spam
   5. Mobile Code
      - ActiveX
      - JavaScript
      - Java mit Ausblick auf die Java Sicherheitsarchitektur

Bedrohungsanalyse: Bedrohungen und Angriffe

- IT Systeme sind **Bedrohungen (Threats)** ausgesetzt:
  - Unberechtigter Zugriff auf Daten
  - Diebstahl, Modifikation, Zerstörung
  - Störung der Verfügbarkeit, Ressourcen unbenutzbar machen
- **Angriff (Attack)**
  Unberechtigter Zugriff auf ein System (oder der Versuch)
  - **Passiver Angriff**
    Angreifer kann nur Informationen erlangen diese aber **nicht ändern**
  - **Aktiver Angriff**
    Angreifer kann Daten und/oder Systeme manipulieren
  - **Social Engineering**
    Angreifer erlangt Daten indem er „menschliche Schwäche“ ausnutzt
- Beispiele für Angriffe
  - Abhören (Eavesdropping, wiretapping, sniffing)
  - Maskerade (Masquerade)
    Mallet behauptet Allice zu sein
  - Brechen von Accounts (z.B. root)
  - Ressourcenmissbrauch; Diebstahl von Rechten
  - Kommunikationsangriffe: Änderung, Kopie, Wiedereinspielung, Fälschung, Umleitung, Verzögerung, Löschung von Nachrichten
  - Denial of service (DoS), Distributed DoS (DDoS)
    Berechtigte Nutzer können Dienste nicht mehr nutzen

Angriffe und Gefährdungen (Übersicht)

- Denial of Service (DoS), Distributed DoS (DDoS)
- Malicious Software (Malware)
  - Virus
  - Wurm
  - Trojanisches Pferd
  - “Mobile” Code
- Überwindung der Zugriffskontrolle
- Abhören und Sniffer
- Port Scanning
- Exploits
  - Buffer Overflows
  - Account/Password Cracking
  - Rootkits
  - Hintertüren (Backdoors)
- Social Engineering
  - Telefonanrufe
  - Täuschung
  - Videoüberwachung
  - „Shoulder surfing“
  - „Dumpster diving“ (Tauchen im Abfalleimer)
  - .....

Denial of Service (DoS) and DDoS

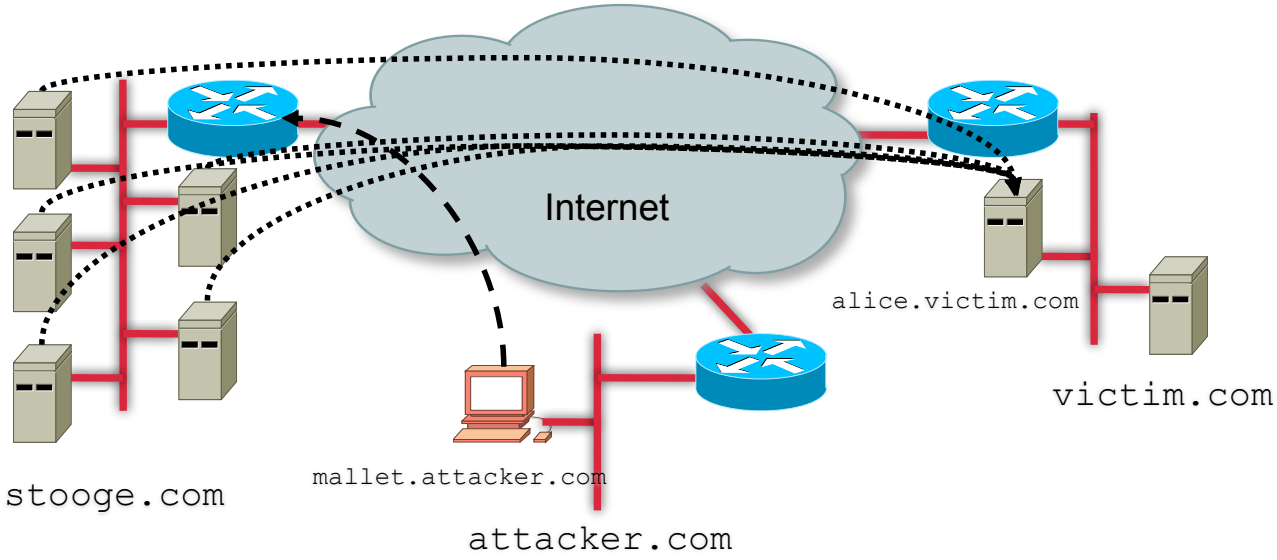
- Angriff versucht Zielsystem oder Netzwerk für berechtigte Anwender unbenutzbar zu machen, z.B. durch:
  - Überlastung
  - Herbeiführung von Fehlersituation
  - Ausnutzung von Programmierfehlern oder Protokollschwächen (Bugs)
- Arten von DoS
  - Anforderung bzw. Nutzung beschränkter oder unteilbarer Ressourcen des OS (z.B. CPU-Zeit, Plattenplatz, Bandbreite,….)
  - Zerstörung oder Veränderung der Konfiguration
  - Physikalische Zerstörung oder Beschädigung
- Beispiel:
  - Angestellter “konfiguriert” “Vacation” Mail mit cc: an interne Mailingliste Außerdem konfiguriert er automatische Bestätigung durch Empfänger
    ⟹ **Mailstorm**

DoS Beispiele

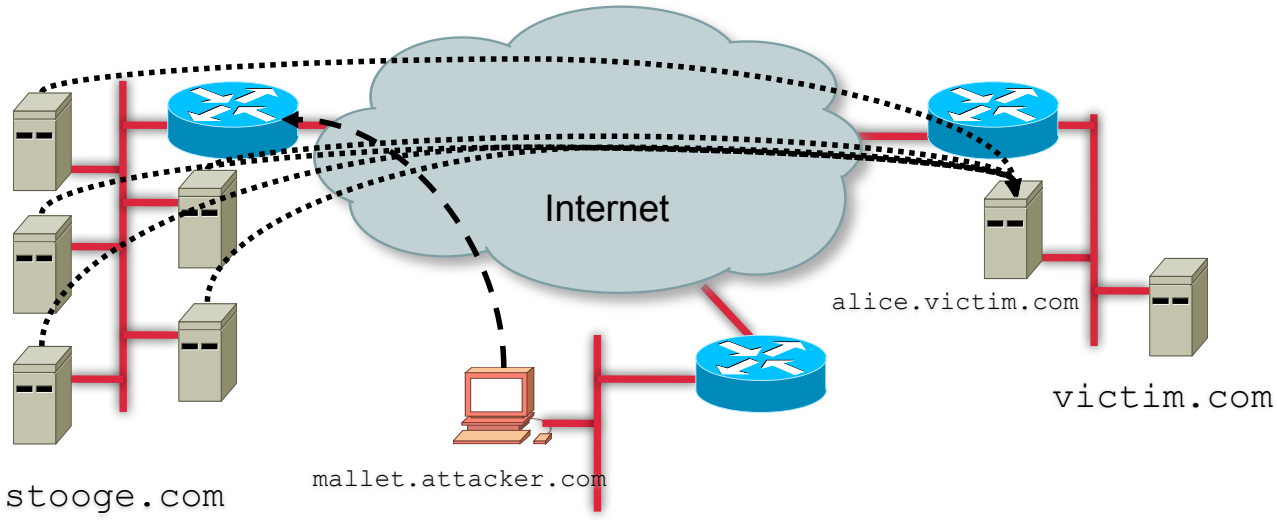
- **E-mail Bombing:** Überflutung der Inbox mit Mails
- **E-mail Subscription Bombing**: Opfer wird auf hunderten Mailinglisten registriert
- **Buffer Overflows; am Bsp. von Ping of Death**
  - IP-Paket größer als die max. erlaubten 2^16 Bytes
  - Übertragen in mehreren fragmentierten Paketen (Andernfalls würden die Router das Paket verwerfen.)
  - Reassemblieren am Zielsystem, führt zu Überlauf des internen Puffers im IP-Stack
  - Evtl. Absturz des Betriebssystems
- **Ausnutzung von Programmfehlern**
  - **Land:** gefälschtes IP Packet mit IP Source Adr. = IP Destination Adr. und Source Port = Dest. Port ⟹ 100 % CPU Last bei best. Implementierungen
  - **Teardrop:** Fragmentierte Pakete enthalten Feld `Fragment Offset` Hier Manipulation, so dass sich Fragmente “überlappen” ⟹ u.U. Absturz des Systems
- **“Konsum” von Bandbreite bzw. Betriebssystem-Ressourcen**
  - Fluten des Netzwerkes des Opfers (z.B. SMURF)
  - UDP `chargen/echo` Angriff
  - SYN-Flooding

DoS-Techniken: SMURF

- Angreifer sendet Strom von `ping` Packeten (ICMP) mit gefälschter Absender-Adresse (`alice.victim.com`) (Adressfälschung wird auch als IP-Spoofing bezeichnet) an IP-Broadcast Adresse von `stooge.com`
- Alle Rechner aus dem Netz von `stooge.com` antworten an `alice.victim.com`



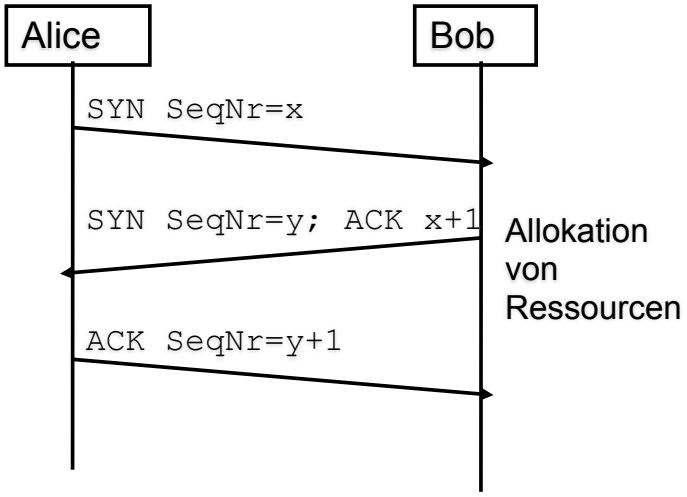
Gegenmaßnahmen?



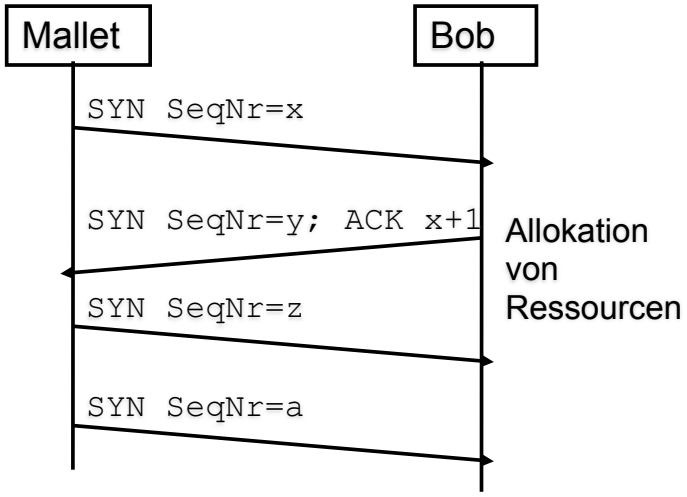
- ICMP deaktivieren (nur bedingt möglich und sinnvoll)
- IP-Broadcast am Router deaktivieren
  - Problem DDoS: nicht ein Angreifer sondern sehr viele Angreifer, die nicht Broadcast sondern versch. Rechner verwenden, oder `alice` direkt angreifen

DoS-Techniken: SYN Flooding

- TCP 3-Way Handshake zum Verbindungsaufbau



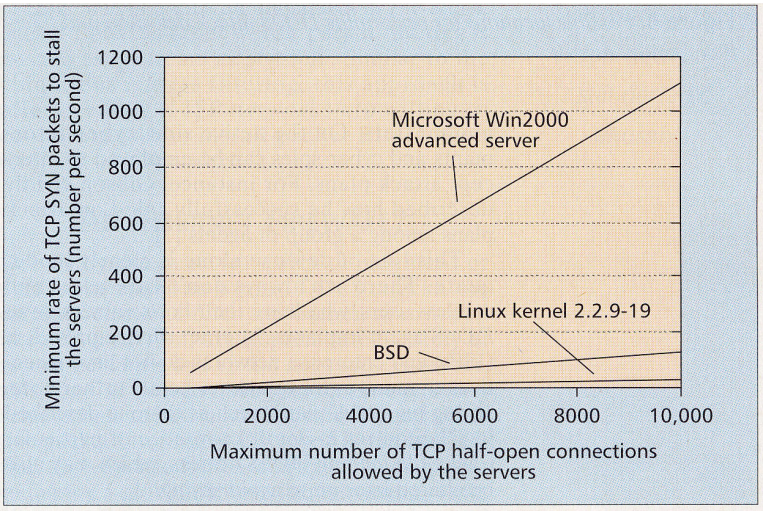
- SYN Flooding



  - Soviele „halboffene“ TCP-Verbindungen aufbauen bis Ressorucen von Bob erschöpft sind

SYN-Flood.: Verletzlichkeit der Betriebssysteme

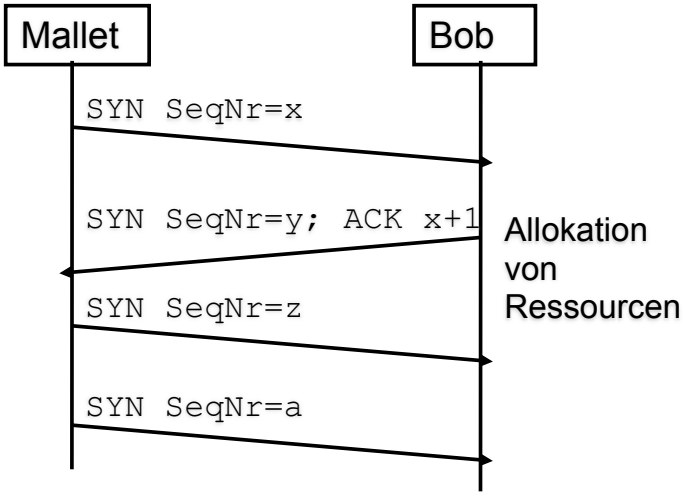
- Minimale Anzahl von SYN Paketen für erfolgreichen DoS
  Quelle: [Chang 02]



- Wiederholung von „verlorenen“ SYN Paketen:
  - Exponential Backoff zur Berechnung der Wartezeit
    - Linux und W2K
      (3s, 6s, 12s, 24s,....)
    - BSD
      (6s, 24s, 48s, ...)
  - Abbruch des Retransmit
    - W2K
      nach 2 Versuchen (d.h. nach 9 Sekunden)
    - Linux
      nach 7 Versuchen (d.h. nach 381 Sekunden)
    - BSD
      nach 75 Sekunden

SYN Flooding: Gegenmaßnahmen?

- Timer definieren:
  Falls `ACK` nicht innerhalb dieser Zeitspanne erfolgt, Ressourcen wieder freigeben
  - Nutzt nur bedingt
- Falls alle Ressourcen belegt zufällig eine halboffene Verbindung schliessen
  - Nutzt nur bedingt
- Maximale Anzahl gleichzeitig halboffener Verbindungen pro Quell-Adresse festlegen
  - Problem DDoS



- SYN Cookies:
  Sequ.Nr. y von Bob „kodiert“ Adressinfo von Mallet. Ressourcen werden erst reserviert wenn tatsächliches ACK y+1 von Mallet eingeht
  - Schützt nicht vor „Brute-Force“ Überflutung des Netzes

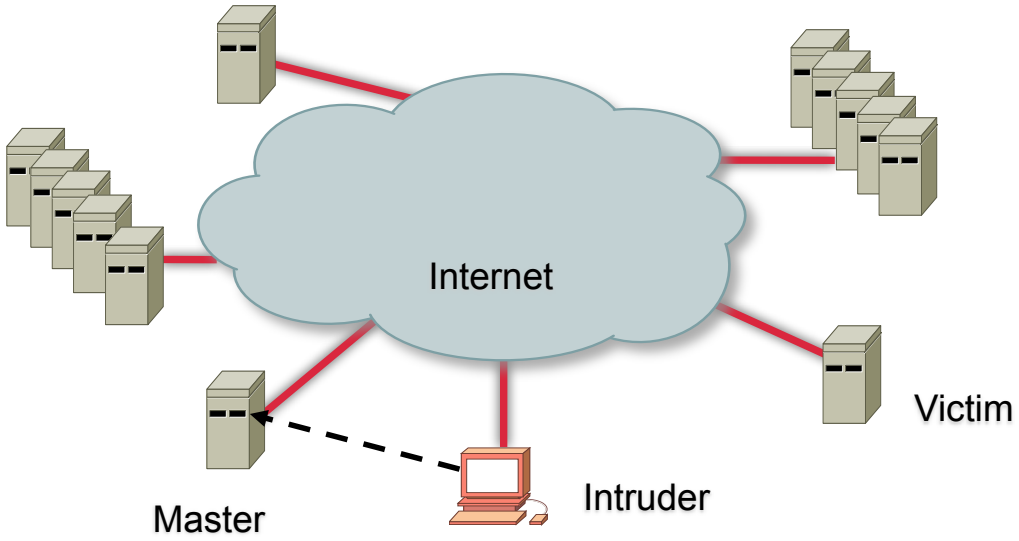
Distributed Denial of Service (DDoS)

- Historie:
  - Trinoo erstmals im Juli 99 aufgetaucht; Aug. 99: 227 Clients greifen eine Maschine der Uni Minnesota an (2 Tage Down-Zeit)
  - 7. Feb. 2000: Yahoo 3 Stunden Downzeit (Schaden ~ 500.000 $)
  - 8. Feb. 2000: Buy.com, CNN, eBay, Zdnet.com, Schwab.com, E*Trade.com und Amazon. (Bei Amazon 10 Stunden Downzeit und ~ 600.000 $ Schaden)
- Idee:
  DoS Angriffswerkzeuge werden auf mehrere Maschinen verteilt und führen auf Befehl eines Masters Angriff durch.
- Terminologie
  - Intruder oder Attacker: Angreifer (Person)
  - Master oder Handler: Koordinator des Angriffs (Software)
  - Daemon, Agent, Client, Zombie, Bot oder bcast Programm: Einzelkomponente die Teil des DDos durchführt (Software)
  - Victim oder Target: Ziel des Angriffs
- Beispiele:
  - Trinoo (Trin00)
  - Tribe Flood Network (TFN) und TFN2K
  - Stacheldraht

DDoS: Grundsätzlicher Ablauf

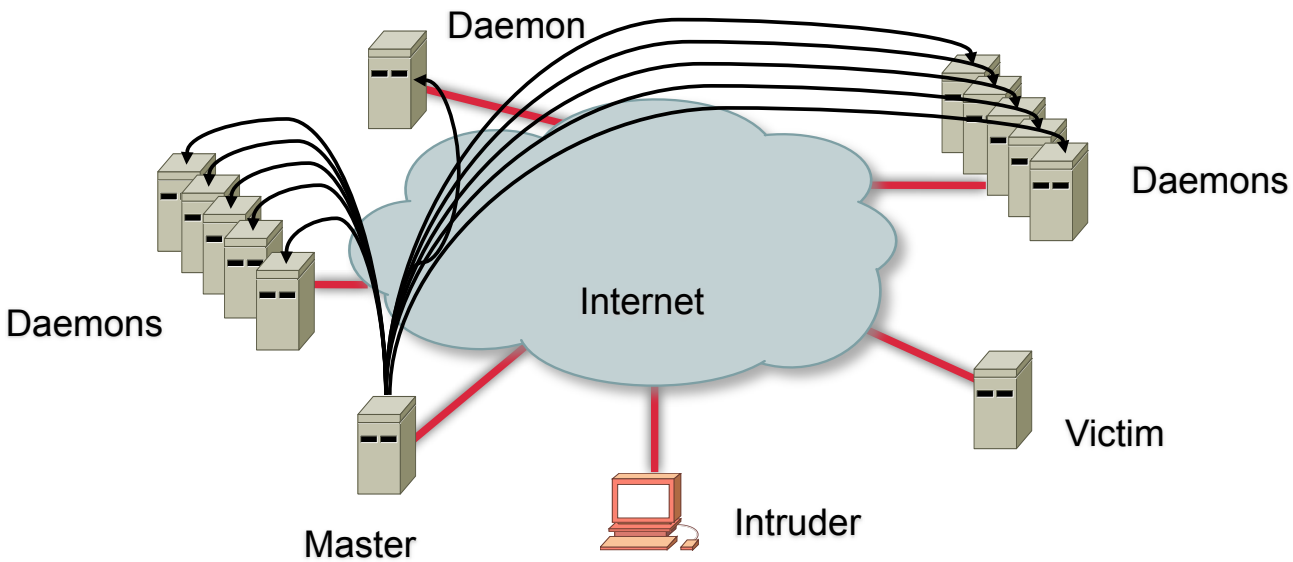
- Dreistufiges Verfahren:

1. Intruder findet Maschine(n) die kompromittiert werden können; Hacking Werkzeuge, Scanner, Rootkits, DoS/DDoS Tools werden installiert; ⇨ Maschine wird Master



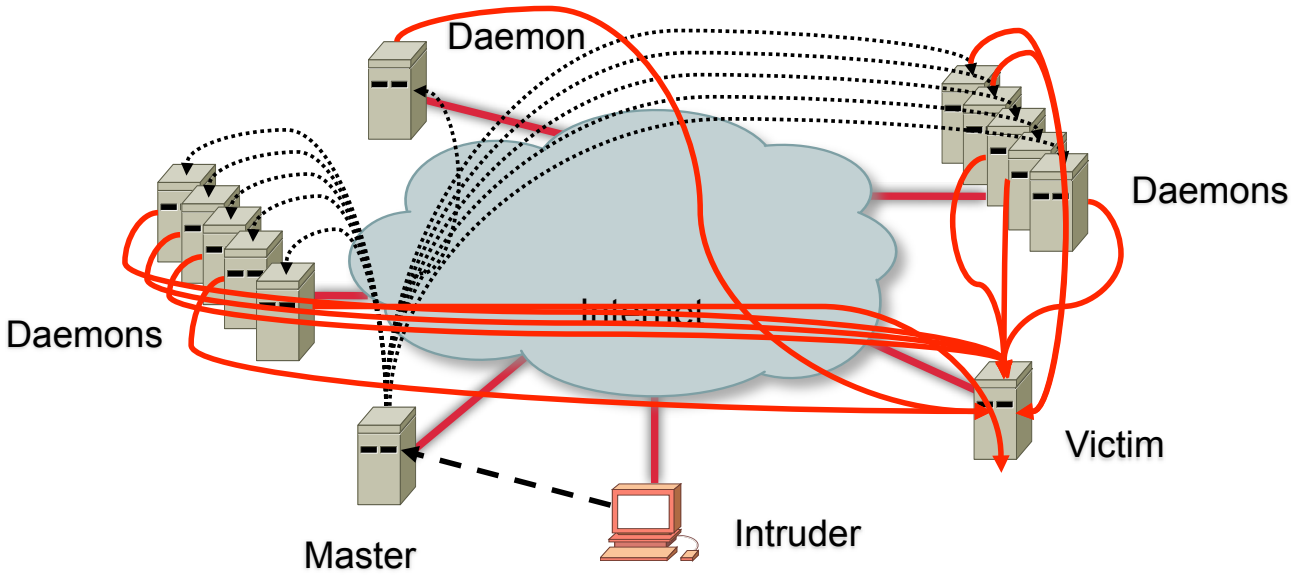
DDoS Phase 2: Initial-Mass Intrusion

2. Master versucht „automatisch“ weitere Maschinen zu kompromittieren um DDoS Daemon zu installieren



DDoS Phase 3: Angriff

3. Intruder startet Programm auf Master, das allen Daemonen mitteilt wann und gegen wen der Angriff zu starten ist.
   Zu vereinbartem Zeitpunkt startet jeder Daemon DoS Angriff



DDos Beispiele: TrinOO, TFN u. Stacheldraht

- Trinoo oder Trin00
  - Verteilter SYN Flooding Angriff
- Kommunikation:
  - Intruder -> Master: Master hört auf TCP Port 27665; Passwort „`betaalmostdone`"
  - Sonstiges Passwort: „`144adsl`"
  - Master -> Daemon: Daemon auf UDP Port 27444
  - Daemon -> Master: Master auf UDP Port 31335
    Beim Start `*HELLO*` Nachricht des Daemon
  - Kepp-Alive Kommunikation:
    Master -> Daemon: `png`
    Daemon -> Master: `PONG`
- Tribe Flood Network (TFN)
  - Master kompromittiert UNIX Systeme über RPC Buffer Overflow
  - SYN, ICMP, UDP Flooding
  - SMURF Angriff
- Kommunikation:
  - wird vollständig in `ICMP ECHO` und `ICMP REPLY` Nachrichten „versteckt":
    Kommando wird im Identifier Feld des ICMP Paketes kodiert; z.B.
    345 -> SYN Flooding;
    890 –> UDP Flooding;
  - Kein Passwort-Schutz
- Stacheldraht = Trinoo + TFN + verschlüsselte Kommunikation + Auto-Update des Agenten

DoS Schutz- und Gegenmaßnahmen

- Zusammenarbeit um Verbreitung der Tools und Durchführung der Angriffe zu verhindern
- Monitoring und Intrusion Detection
- Packetfilter auf Gateways und Routern (ingress und egress; vgl. Abschnitt über Firewalls)
- Überwachung von Volumen und Anzahl der Verbindungen pro Quell-Adresse
- Überwachung und Kontrolle der offenen Ports
- Blockieren von Broadcast
- Konfigurationen überprüfen
- Relevante Patches installieren
- Meldung an Provider aus dessen Netz DoS erfolgt
- Bug- und Sicherheitswarnungen (z.B. CERT) verfolgen
- Zusammenarbeit der Provider und der CERTs

Malicious Code: Virus

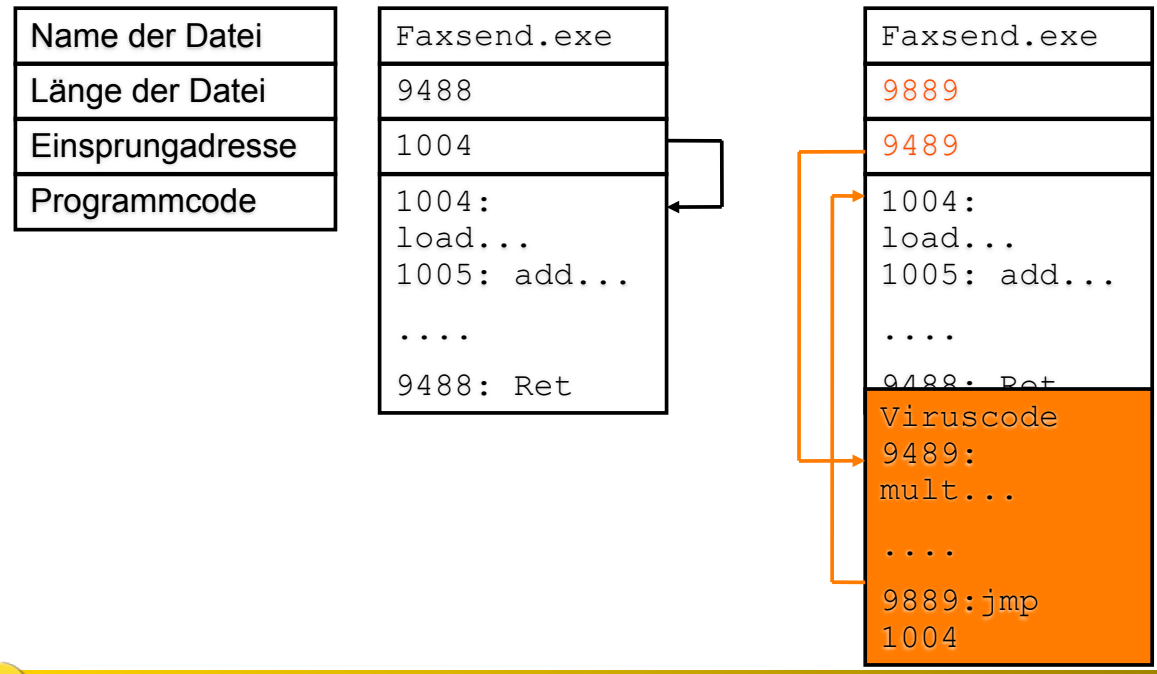
- Definition:
  - Befehlsfolge; benötigt Wirtsprogramm zur Ausführung
  - Kein selbstständig ablauffähiges Programm
  - Selbstreplikation (Infektion weiterer Wirte (Programme))
- Allgemeiner Aufbau:

Viruserkennung	void function virus { signature
Infektionsteil	suche Programm p ohne signature kopiere virus in p
Schadensteil ggf. mit Bedingung	if (tag == Freitag && monatstag == 13) { format c: d: e: f: g: h: i: j: k: l: m: }
Sprung	springe an den Anfang des Wirtsprogramm }

  - Daneben ggf. Tarnungsteil

Programm-Viren: Infektion

- Dateiformat vor der Infektion
- Datei nach der Infektion



Viren Klassifikation

- Klassifikation nach Infektionsziel:
  - **Programm-Virus (Link-Virus)**
    Infiziert ausführbare Dateien (.exe, .com, .sys)
  - **Bootsektor-Virus**
    Infiziert den Bootsektor von Festplatten oder Disketten
  - **Makro-, Daten-Virus**
    Infiziert „Daten-Dateien“ mit eingebetteten Makro-Sprachen (z.B. Visual Basic in MS Office, Postscript,...)
- Unterklassen:
  - **Multipartiter Virus**
    Infiziert mehr als ein Ziel
  - **Polymorpher Virus**
    Verschlüsselt den Viruscode; damit für Anti-Viren Software (AV) schwerer zu finden
  - **Retro-Virus**
    Greift aktiv AV an; versucht Scanner so zu verändern, dass er unentdeckt bleibt.
  - **Stealth Virus**
    Virus versucht sich vor AV zu verstecken. Sobald AV Dateien scannt, entfernt der Virus seinen Code aus den infizierten Dateien (Wiederherstellung des Orginalzustandes)
  - **Tunneling-Virus**
    AV versucht Systemaufrufe zum Schreiben in den Bootsektor zu unterbrechen. Virus ermittelt dabei direkte Speicheradresse des entsprechenden Systemaufrufs und „klinkt“ sich direkt an dieser Speicheradresse ein.

Einschub: DFN-CERT Alert 2009-1506 u. -1520

- Schwachstelle im Adobe Reader
- Betroffene Software: acroread, acroread_ja
- Betroffene Plattformen:
  - -1506: OpenSuSE 10.3, 11.0, 11.1
  - -1506: SLED 10 SP2 und SP3, SLED 11
  - -1520: SPARC Solaris 10
- Schwachstellen:
  - Denial of Service Schwachstellen (Anhängen von langen „#“ Zeichnketten an pdf-URL)
  - Cross Site Scripting (XSS); Ergänzung eines Links im pdf durch Java Script Code
- Lösung:
  - Patch einspielen

Malicious Code: Wurm

- Definition
  - Eigenständig ablauffähiges Programm
  - Benötigt keinen Wirt
  - Selbstreplikation
  - (besteht aus mehreren Programmteilen = Wurm-Segmente)
- Bsp.:
  - Internet Wurm, vgl. Kap. 1
  - ILOVEYOU
  - Melissa
  - SQL Slammer
  - Conficker
  - .....

Malicious Code: Trojanisches Pferd

- Definition:
  - Ein Programm dessen Ist-Funktionalität nicht mit der angegebenen Soll-Funktionalität übereinstimmt
    - Sinnvolle oder attraktive „Nutzfunktionalität“
    - Versteckte (Schad-) Funktionalität
    - Keine Vervielfältigung

- Beispiel: Unix Shell Script Trojan [Stoll 89]:

```
echo “WELCOME TO THE LBL UNIX-4 COMPUTER”
echo “LOGIN:”
read account_name
echo “PASSWORD:”
(stty -echo;\
 read password;\
 stty echo; echo “”;\
 echo $account_name $password >> /tmp/.pub)
echo “SORRY, TRY AGAIN.”
```


Trojanische Pferde, Beispiele

- Rundung bei der Zinsberechnung
  - Nutzfunktion: Zinsberechnung mit drei Stellen Genauigkeit
  - Versteckte Funktionalität: Abgerundete Beträge ab der 4. Stelle aufsummieren und auf definiertes Konto buchen.

- T-Online Power Tools (1998)
  - Nutzfunktion: Unterstützende Werkzeuge für den T-Online Decoder
  - Versteckte Funktionalität: Bei der Registrierung (Shareware) werden T-Online Zugangsdaten übermittelt

- FBI’s Magic Latern / D.I.R.T (Data Interception by Remote Transmission) (2001)
  - Integrierbar in (Nutzfunktion):
    - Word, Excel, Powerpoint
    - RTF (Rich Text Format)
    - Word Perfect
    - Autorun.bat on CDs
    - …. ??…..
  - Versteckte Funtkionalität:
    - Keyboard Logger
    - Auslesen entfernter Daten
    - Passphrase Logging (z.B., PGP Private Key Passphrase)
    - Übertragung des entfernten Bildschirminhalts
    - Übertragung v. entferntem Audio (falls Micro. vorhanden)
- „Bundestrojaner“

Malicious Code heute

- Grenzen zwischen Klassen verschwinden
- Heutige Schadsoftware umfasst i.d.R. mehrere Klassen, z.B.:
  - Virus mit Wurmfunktionalität
  - Wurm mit Trojaner und Backdoor
  - Schadsoftware mit DoS bzw. DDoS Funktionalität
  - Schadsoftware mit eigenem Mail-Server fürs Spamming
  - usw.

Malicious Code: Schutz- und Gegenmaßnahmen

- Keine Software aus „unsicheren“ Quellen installieren
- Anti-Viren Software (AV) installieren **UND** aktualisieren
- Firewall um Viren- und Mail-Scanner erweitern
- Ausführbare Dateien oder Dateisystem verschlüsseln
- Kryptographische Prüfsummen (vgl. Abschnitt Kryptographie)
- Integrity Checker installieren und regelmäßig testen (vgl. Abschnitt Tools)
- Schreibrechte sehr restriktiv vergeben (Need to know Prinzip)
- Automatische Makro Ausführung deaktivieren
- *Impfen: In die Programme wird bewusst der Erkennungscode X des Virus eingetragen.*
- *Weniger anfälliges OS wählen*

Mail: Falsche Virenwarnungen; Hoaxes

- GEZ-Gebührenerstattung:

Die öffentlich-rechtlichen Rundfunkanstalten ARD und ZDF haben im Frühjahr einen Gewinn von über 1 Mrd. DM erwirtschaftet. Dieses ist gemäß Bundesverfassungsgericht unzuläßig. Das OLG Augsburg hat am 10.01.1998 entschieden, daß an diesem Gewinn der Gebührenzahler zu beteiligen ist. Es müssen nach Urteil jedem Antragsteller rückwirkend für die Jahre 1997, 1998 und 1999 je Quartal ein Betrag von DM 9,59 (insgesamt 115,08 DM) erstattet werden. ACHTUNG! Dieses Urteil wurde vom BGH am 08.04.98 bestätigt.[....] Bitte möglichst viele Kopien an Verwandte, Freunde und Bekannte weiterleiten, damit die Gebühren auch ihnen erstattet werden.

- AIDS-Infektion im Kino:

Vor einigen Wochen hat sich in einem Kino eine Person auf etwas Spitzes gesetzt, das sich auf einem der Sitze befand. Als sie sich wieder aufgerichtet hat, um zu sehen, um was es sich handelte, da hat sie eine Nadel gefunden, die in den Sitz mit einer befestigten Notiz gestochen war: "Sie wurden soeben durch das HIV infiziert". Das Kontrollzentrum der Krankheiten berichtet über mehrere ähnliche Ereignisse, kürzlich vorgekommen in mehreren anderen Städten.

Alle getesteten Nadeln SIND HIV positiv. Das Zentrum berichtet, dass man auch Nadeln in den Geldrückgabe-Aussparungen von öffentlichen Automaten (Billette, Parking, etc.) gefunden hat. Sie bitten jeden, extrem vorsichtig zu sein in solchen Situationen. Alle öffentlichen Stühle müssen mit Wachsamkeit und Vorsicht vor Gebrauch untersucht werden. Eine peinlich genaue sichtliche Inspektion sollte ausreichen. Außerdem fordern sie jeden auf, allen Mitgliedern Ihrer Familie und Ihrer Freunde diese Nachricht zu übermitteln.

Dies ist sehr wichtig!!! Denk, dass Du ein Leben retten kannst, indem Du
diese Nachricht weiter verteilst.

Frank Richert
Polizeidirektion Hannover

Autobahnpolizei Garbsen

Hoax, mögliche Erkennungszeichen

- Warnung vor „extrem gefährlichen Virus“
- “Keine AV kann Virus erkennen”
- “Warnen Sie alle Bekannten und Freunde”
- Nicht plausible Bedrohung
  (z.B. physikalische Zerstörung des Rechners)
- Verweis auf namhafte Unternehmen oder Forschungseinrichtungen
- Kettenbriefe im klassischen Sinn:
  - Gewinnspiele oder Glücksbriefe
  - „Geld zurück“
  - E-Petitionen
  - Pyramidensysteme
  - „Tränendrüsenbriefe“
- Schutzmaßnahmen: Hoax Mail löschen und NICHT verbreiten
- Beispiele: http://www2.tu-berlin.de/www/software/hoaxlist.shtml

SPAM-Mail

- Unerwünschte Werbe Mail (unsolicited commercial e-mail)
- Begriff SPAM
  - SPAM eingetragenes Warenzeichen von Hormel Food
  - „Spam“-Sketch aus Monty Python‘s Flying Circus
- SPAM Aufkommen
  - LRZ Oktober 2008
  - Verarbeitete Mails: 14.556.0000 Mails
  - SPAM und Viren Mails 14.436.000 (~99,18 %)
    - Abgelehnte Mails 14.400.000 (~99 %);
    - Als SPAM markiert 35.000 (~0,24 %)
    - Vieren-Mails 1.000 (~0,01 %)
  - Ham Mails 120.000 (~0,82 %)
- Probleme:
  - Eingangs-Mailbox wird mit SPAM überflutet
  - Extrem störend
  - Zusätzlicher Aufwand (Speicherplatz, Arbeitszeit)
  - Zusätzliche Kosten (Infrastruktur, Übertragung, Personal,....)



Spam, klassische Gegenmaßnahmen: Spamfilter

- Prozess der eingehende Mails nach Spam durchsucht
- Arten von Spam-Filtern:
  1. Blacklist / Whitelist Ansatz:
     Sperren Mail Domänen die üblicherweise von Spammer benutzt werden
  2. Regelbasiert:
     Nachricht wird inhaltlich nach Spam-Merkmalen durchsucht;
     sowohl im Header wie auch im Body der Mail
  3. KI-basierter Ansatz:
     Neuronale- (NN-) oder Bayessche Netze versuchen Spam zu erkennen
- Vor- u. Nachteile:
  1. Effizient zu implementieren; Grobgranular; keine inhaltliche Prüfung
  2. Sehr hohe Erkennungsraten; Test aufwändiger
  3. Erkennungsrate abhängig vom Training (NN) oder Modellierung (Bayes); dafür große Datenmengen erforderlich

Spamfilter

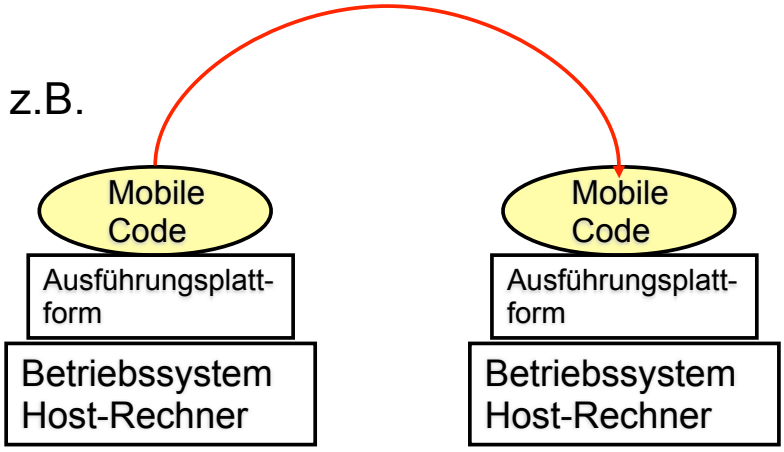
- Fehlerarten bei der Erkennung
  - Filter die „automatisch“ Entscheidungen treffen machen zwei Arten von (systematischen) Fehlern:
  - **Falsch positiv**: Mail wird als Spam erkannt obwohl keine Spam
  - **Falsch negativ**: Mail wird nicht als Spam erkannt obwohl Spam
- Welche Fehlerart ist problematischer?
- Policy für Spambehandlung:
  - Spam-Mail löschen und Empfänger ggf. benachrichtigen
  - Spam-Mail markieren und dann ausliefern
  - Welche Variante bevorzugen (unter Beachtung der Fehlerarten)?
- Bsp.: SpamAssassin (`www.spamassassin.org`)
  - Implementiert alle Filterarten (Blacklist, Regelbasis, Bayessches Netz)
  - Zentral und dezentral einsetzbar
  - Effizient und effektiv; Feingranular konfigurierbar;

Weitere Anti-SPAM Maßnahmen

- Eigenes Kapitel

Malicious Code: „Mobile Code“

- Abgrenzung zu Viren, Würmern und Trojaner fließend
- Hier - Mobile Code (aktiver Inhalt):
  - Code wird auf entferntem Rechner generiert
  - häufig in Web Seiten eingebettet und
  - auf lokalem Client-Rechner ausgeführt.
  - I.d.R. Ausführungsplattform oder Interpreter zur Ausführung erforderlich
- Verwendete Sprachen, z.B.
  - ActiveX
  - JavaScript
  - Java



Mobile Code: ActiveX

- Von Microsoft entwickelte Erweiterung von OLE (Object Linking and Embedding)
- ActiveX Control:
  - Wiederverwendbare Komponente
  - Binärformat
  - Standardisierte Schnittstelle
  - Beliebige Programmiersprache zur Entwicklung (z.B. C, Basic, C#,…)
  - Wird innerhalb des Browsers ausgeführt
- Probleme:
  - Keine Ausführungsbeschränkung
  - Voller Betriebssystemzugriff
  - Selbe Rechte wie ausführender Benutzerprozess
- Beispiele für ActiveX Malware:
- Internet Exploder (1996): “Signed” ActiveX Control das bei der Ausführung den Rechner herunterfährt.
- Chaos Computer Club (CCC) Demonstrator (27.1.97)
  - Control sucht nach Quicken
  - Erstellt Überweisung und trägt diese in die Liste offener Überweisungen in Quicken ein.
  - Quicken kann mit einer PIN TAN Kombination mehrere Überweisungen übertragen, d.h. unvorsichtiger User wird “gefälschte” Überweisung übertragen
  - `www.iks-jena.de/mitarb/lutz/security/activex.html`

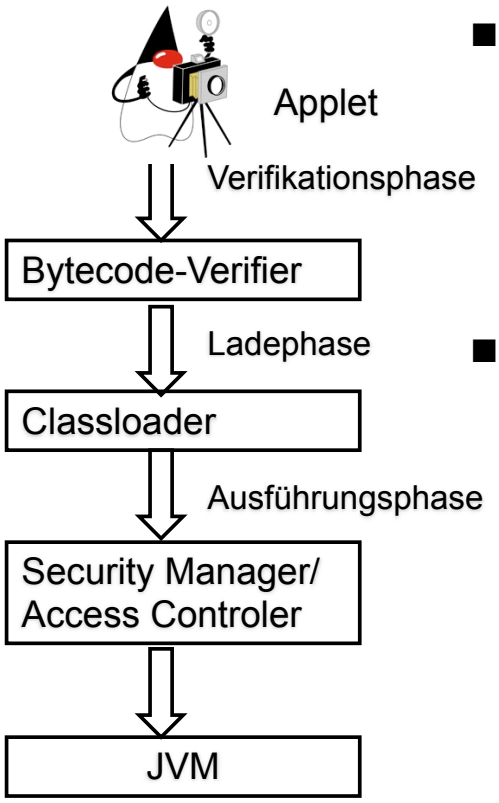
Mobile Code: JavaScript

- Entwickelt von Netscape
  - Scriptsprache; **syntaktisch** angelehnt an C,C++ u. Java
  - Einbettung aktiver Inhalte in Web-Seiten
  - Wird innerhalb des Browsers ausgeführt
- JavaScript Skript:
  - Kein Zugriff auf das Dateisystem (*außer* auf Cookies)
  - Keine Netzverbindungen (*außer* Download von URLs)
- Probleme
  - Kein explizites Sicherheitsmodell
  - Entwicklungsgrundsatz: „Identify (security holes) and patch approach“
- Umfangreiche Liste von Schwachstellen und Implementierungsfehlern
- Netscape 2.x
  - Auslesen der History
  - Lesender und schreibender Zugriff auf das Dateisystem
- Netscape 3.x
  - Versenden von Mail
- Netscape 4.x
  - Hidden frame mit eingebetteter `Post` Methode + Attachment sendet Files an böswilligen Web-Server
  - JavaScript eingebettet in Cookie; damit z.B. Lesen der Bookmarks oder HTML Dateien im Cache `www.peacefire.org/security/jscookies/`

Mobile Code: Java (Applets)

- Entwickelt von SUN Microsystems
- Applet
  - Byte-Code (vorkompiliert)
  - Benötigt JVM (Java Virtual Machine) zur Ausführung
- Mögliche Bedrohungen
  - Denial of Service
  - Implementierungsfehler in der JVM (z.B. Netscape 2.0 erlaubte Applet den Verbindungsaufbau zu beliebiger URL); Liste der Bugs: `www.cs.princeton.edu/sip/history/`
- Design der Sprache unterstützt Sicherheit und Robustheit
  - Keine Pointer
  - Kein direkter Speicherzugriff; Test ob Array-Grenzen eingehalten werden
  - Strenge Typisierung
- Explizite Sicherheitsarchitektur; im folgenden kurzer Einschub dazu

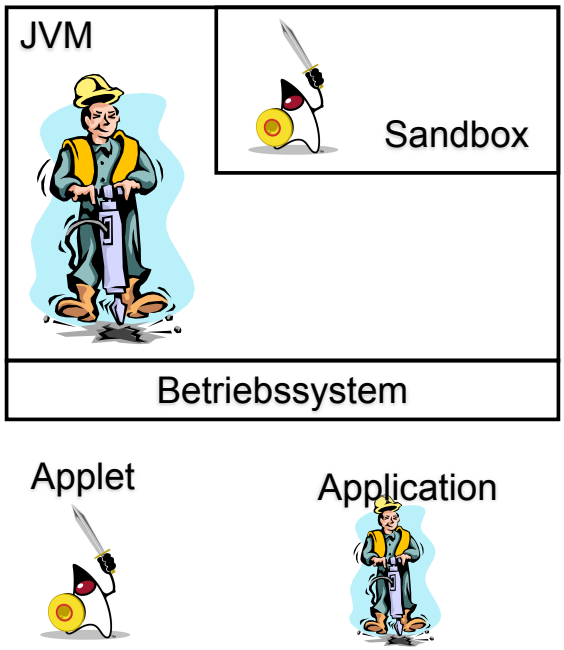
Einschub - Java Sicherheitsarchitektur: Komponenten



- **Bytecode-Verifier**: Es soll nur „korrekter“ Java Code ausgeführt werden!
  - Korrektes Format des `class` Files
  - Syntaktische Korrektheit
  - Einfache Datenflußanalyse (Simulation und Analyse des Programmablaufs); Kein Überlauf des Operandenstack
- **Classloader**: Laden und Instantiieren der Objekte
  - Schutz der Systemklassen vor Überschreiben
  - Sichtbarkeit (Alle Objekte die von einem Classloader geladen wurden bilden eigenen Namensraum)
  - Lade- und Bindestrategie: „lazy loading and early bound“
    Laden erst bei Verwendung des Objektes; bei weiterem Nachladeversuch wird bereits geladenes Objekt verwendet

Java Sicherheitsarchitektur: Sandbox

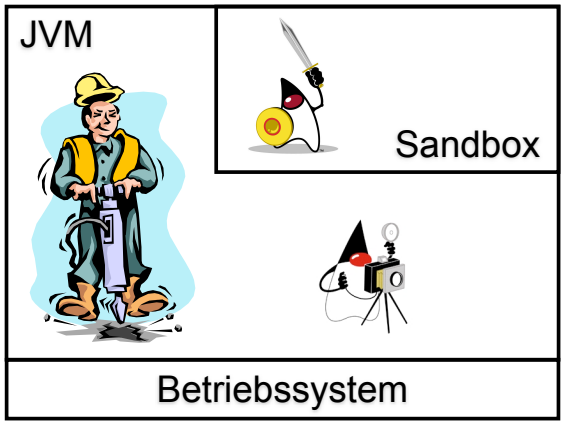
- Realisiert durch Security Manager
- Java 1.0.2 Sicherheitsmodell



- Beschränkung des Zugriffs auf Ressourcen des BS:
  - Dateisystem (Read, Write, Delete)
  - Netzwerkzugriffe
    (Zugriffe zu dem Rechner von dem das Applet geladen wurde, sind erlaubt)
  - Zugriff auf die JVM
    (z.B. Instantiieren von Classloader)
  - Verwaltung von Threads
  - Elemente der GUI
    (z.B. Clippboard, „fremde“ Fenster)
  - Zugriff auf Systemressourcen zum Starten von Prozessen
  - Eingeschränkter Zugriff auf Systemvariablen (Properties) der JVM
- Diese Ressourcen für Applets verboten

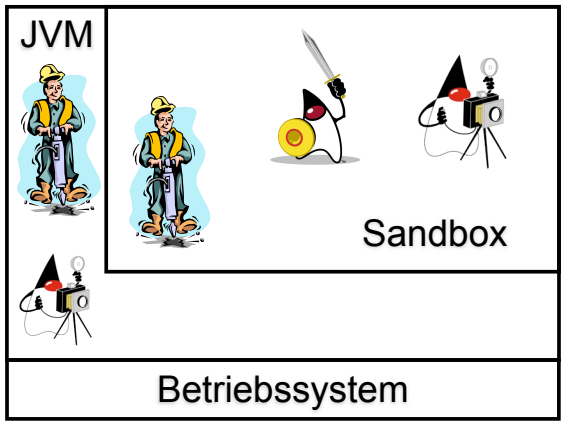
„Evolution“ der Sandbox

- Java 1.1
  - Trusted (d.h. signierte) Applets können Sandbox verlassen

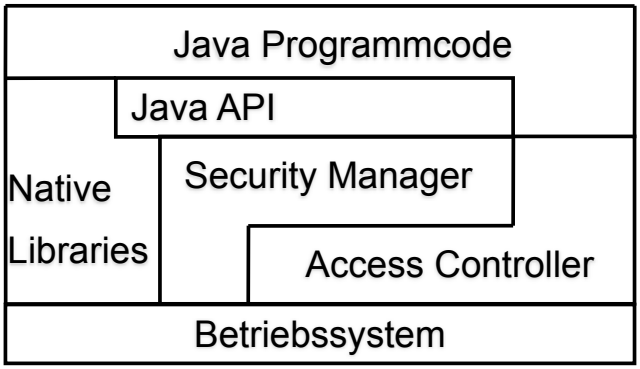



Signed Applet

- Java 1.2
  - Mechanismen der Sandbox zur Beschränkung der Rechte können auch feingranular auf Applikations angewandt werden



Java 1.2 Sicherheitsarchitektur



- Security Manager realisiert Sandbox; d.h. zentrale Instanz für Zugriffskontrolle
  - Relativ starres Konzept
  - Applets können Sandbox nicht verlassen, daher nutzt Sec.Mgr.
- Access Controller
  - parametrisierbar

- Zugriffskontrollentscheidung wird vom Access Controller getroffen; Grundlage der Entscheidung basiert auf:
  - **Code Source**
    URLs und Menge von Signaturen, d.h. öffentlichen Schlüsseln, kennzeichnen Java Objekte (Entitäten)
  - **Permission**
    kapselt Objekte mit bestimmten Operationen; über CodeSource einer Entität zugeordnet, stellt sie ein Recht dar
  - **Protection Domain**
    Gruppierungsobjekt, um alle Rechte einer CodeSource zusammenzufassen
  - **Policy Objekt**
    kapselt alle Permissions aller Entitäten

Java Security Manager / Access Controller

- Security Manager (vor Java 1.2)
  - Wird vor der Ausführung einer kritischen Operation mittels `check`-Methode aufgerufen
    Bsp.: Jeder lesende Dateizugriff ruft über `checkRead` den Sec.Mgr.
  - Falls Aufruf durch trusted Code, kehrt Sec.Mgr. zurück, andernfalls Security Exception
- Ab Version 1.2:
  - Sec.Mgr. ruft mit einem entspr. Permission Objekt den Access Controller über die Methode `checkPermission` auf
  - Damit Abwärtskompatibilität
  - Anwendbar auch für Applications
  - Erweiterbar um eigene Permission-Objekte
- Spezifikation einer Policy (Bsp.)

```
grant CodeBase
  „http://java.sun.com“,
SignedBy „gong“ {
permission
java.io.FilePermission,
  „read,write“, „/tmp/*“;
}
```

  - Alle Objekt von `java.sun.com`, die von `gong` signiert sind, bilden eine **Protection Domain**.
  - Allen Objekten aus dieser Domain wird die **Permission** erteilt, innerhalb von `/tmp` zu schreiben

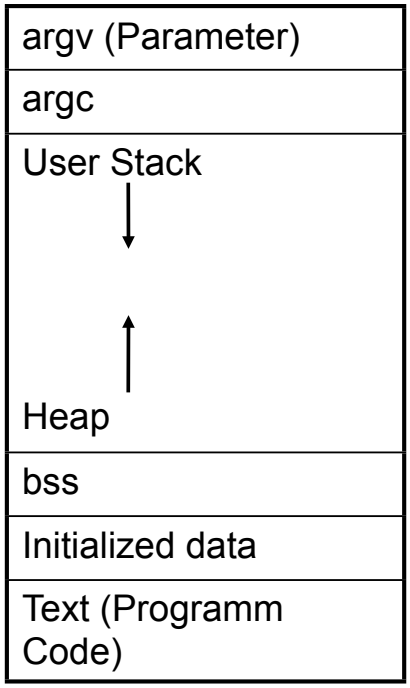
Inhalt (2)

3. Bedrohungen, Angriffe und Gefährdungen (Forts.)
   6. Buffer Overflow
   7. Account / Password Cracking
   8. Hintertüren / Falltüren
   9. Rootkits
   10. Cross Site Scripting (XSS)
   11. Sniffer
   12. Port Scanner
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

Exploits: Buffer Overflow

- Ziel: Ausführen von Code auf fremdem Rechner unter fremden Rechten (z.B. root)
- Vorgehen:
  - Auswahl eines Programmes, das z.B. mit SUID (Set User ID)-Bit, d.h. mit Rechten des Eigentümers, läuft
  - Überschreiben interner Programmpuffer, z.B. durch überlange Eingabe
  - Dabei Manipulation z.B. der Rücksprungadresse, dadurch
  - Ausführen von bestimmter Programmsequenz des Angreifers; z.B. Assembler- bzw. Maschinen-Code zum Starten einer Shell (shellcode)

- Speicherabbild eines Programmes (am Bsp. Unix)

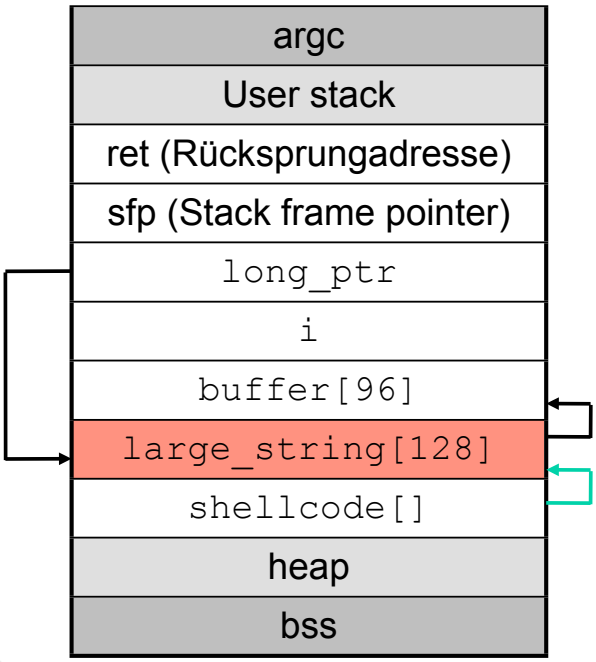


Beispiel-Code Buffer Overflow

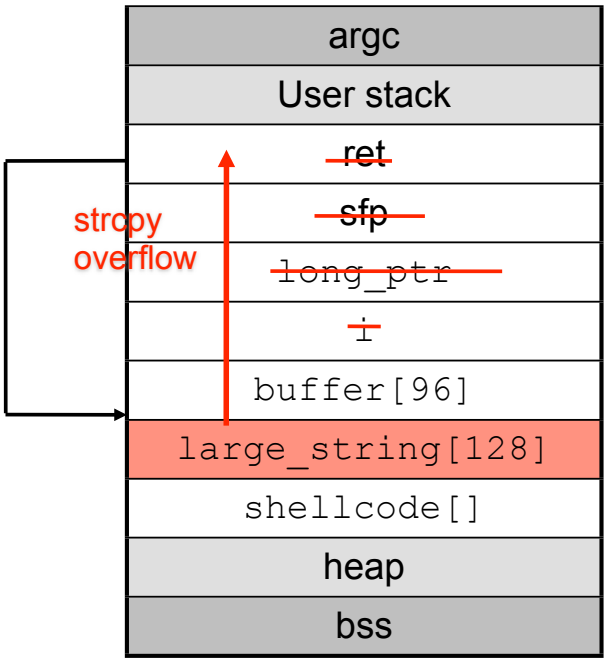
```
/* shellcode für Linux auf Intel Architektur */
char shellcode[]="\xeb\x1f\x5e
   \x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b
   \x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb
   \x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
char large_string[128];
void main() {
    char buffer[96];     /* char 1 Byte */
    int i;
    /*Zeiger auf large_string; long 4 Byte */
    long *long_ptr = (long *) large_string;
    /* large_string wird mit der Adresse von buffer belegt */
    for(i=0; i<32; i++)
        *(long_ptr + i)=(int) buffer;
    /*shellcode in large_string kopieren */
    for (i=0; i<strlen(shellcode); i++)
      large_string[i] = shellcode[i];
    /*Kopie von large_string in buffer; shellcode und  pointer; buffer overflow */
    strcpy(buffer,large_string);     }
```

Buffer Overflow (Bsp)

- Speicher vor dem Aufruf von `strcpy()`



- Nach Aufruf von strcpy() ist die Rücksprungadresse überschrieben; shellcode wird ausgeführt



Buffer Overflow Durchführung

- Im Bsp. shellcode fest ins Programm ein-compiliert
- Bei realem Buffer Overflow muss shellcode ins Laufzeitsystem “eingebracht” werde oder eine bestimmte Funktion einer Betriebssystem-Bibliothek ausgeführt werden.
- Dazu gezielte Manipulation von
  - Kommandozeilen-Argumenten
  - Shell-Umgebungsvariablen
  - Interaktiver Input
  - Nachrichten des Netzverkehrs
- Problem dabei ist das “Erraten” der Adresse des shellcode oder der Bibliotheksroutine im Speicher

Buffer Overflow Schutz

- Zentraler Ansatz; Modifikation von
  - SUID Programmen
  - Compiler:
    - Warnung vor “gefährlichen” Funktionen
    - Vermeidung dieser Fkt., d.h. nicht standardkonforme Änderung
    - Einbindung von Fkt. welche die Integrität des Stack vor der Rückgabe des Return-Wertes testen
  - Betriebssystem:
    - Stack Segmente werden als nicht ausführbar markiert
- Dezentraler Ansatz; Modifikation von
  - Einzelnen SUID Programmen
- “Sichere” Programmierung:
  - Verzicht auf Funktionen die Pointer auf ein Ergebnis im Speicher zurückliefern
  - Verzicht auf Funktionen zur String-Manipulation ohne Längenprüfung, z.B. `fgets()` statt `gets()` verwenden
  - Strenge Überprüfung der Eingabeparameter bezgl. Anzahl, Länge und Datentyp
  - Strenge Überprüfung verwendeter Umgebungsvariablen
- Weitere Info: [Smith 97]
- Adress Space Layout Randomization (ASLR): Speicherort für Heap, Stack, Executables und Libraries zufällig

Exploits: Account/Password Cracking

- Häufig Benutzername + Passwort zur Authentisierung
- “Erraten” von Benutzernamen und Passwort
- Brute-Force Angriff / Dictionary Attack bei bekanntem Benutzer-namen (alle möglichen “Passworte” durch-probieren)
- Brechen des Verschlüsselungsalgorithmus für das Passwort
- Social Engineering

- Password Cracking am Beispiel von UNIX:
  - Administrator (`root`) vergibt Benutzernamen
  - Eintrag in `/etc/passwd`
    - Datei für **alle** lesbar
    - Format des Eintrags

```
reiser:Ad9%y?SmW+zP&:23:17:HelmutReiser:/home/reiser:/bin/tcsh
Username:Password:UID:GID:Gecko-String:Home-Verzeichnis:Shell
```

Bsp. UNIX Authentisierung: User / Password

- Benutzer wählt Passwort
  - Passwort wird, mit sich selbst als Schlüssel, verschlüsselt und verschlüsselt gespeichert in `/etc/passwd`:
    `z.B. :Ad9%y?SmW+zP&:`
  - Auch `root` kennt Passwort **nicht**
- Authentisierung:
  - Eingegebenes Passwort wird mit sich selbst verschlüsselt u. mit dem in `/etc/passwd` verglichen
- Verschlüsselungsalgorithmus `crypt(pwd,salt)` bekannt
- Dictionary Attack:
  - Angreifer verschlüsselt Wörterbuch und vergleicht verschlüsselte Strings mit Einträgen in `/etc/passwd`
- Verhinderung der Dictionary Attack
  - Zus. Parameter `salt` in `crypt`
    - 12 Bit Zahl: 0 <= `salt` < 4096
    - Bei Initialisierung zufällig gewählt
    - Die ersten 2 Zeichen im Passwort String sind `salt`; im angegeb. Bsp. `Ad`
- Brute Force Dictionary Attack:
  - Angreifer muss Wörterbuch für **jeden** Benutzer mit dessen `salt` verschlüsseln und vergleichen
  - Bei heutiger Rechenleistung kein echtes Problem
- Verhinderung z.B. durch:
  - Shadow Password System (nur root kann verschl. Passwort lesen)
  - One-Time Passwords
  - ….

Exploits: Back Doors, Trap Doors

- Ziel: Angreifer will Zugang (Hintereingang) zu einer bereits kompromittierten Maschine
  - An der Betriebssystem-Authentisierung vorbei
  - Mit speziellen Rechten (z.B. root)
- Mechanismen, (z.B.):
  - Trojaner mit versteckter Funktionalität
  - Installation eines “versteckten” Netzdienstes, der zu bestimmten Zeiten einen Netzwerk-Port öffnet und auf Kommandos wartet
  - Eintrag in `.rhosts` Datei von root
  - Mail-Alias, dass bei Empfang einer Mail bestimmtes Programm startet
  - SUID/SGID Shell-Skript
- Schutzmechanismen:
- Integritäts-Checks:
  - Kryptographische Prüfsummen:
    - aller installierten Programme
    - Konfigurationsdateien
    - Sowie deren regelmäßige Überprüfung
  - Überprüfung der offenen Ports und der aktivierten Netzdienste
  - Suche nach ungewöhnlichen SUID/SGID Programmen

Exploits: Rootkits

- Werkzeugsammlung mit dem Ziel:
  - Verbergen dass Rechner kompromittiert wurde
  - Zugang zum Rechner erhalten
- Bsp. für Rootkit-Werkzeuge
  - Trojaner
  - Back Door Programme
  - Camouflage-Tools:
    - Tool zur Manipulation des System-Logs
    - Manipulierte Systemkommandos um bestimmte Dateien, Verzeichnisse, Prozesse und Netzverbindungen zu “verstecken”
  - Remote Control Tools
- Linux Rootkit IV als Beispiel
  - Trojaner und Backdoors: `chfn, chsh, inetd, login,passwd, rsh`: Passwort geschützte root-Shell
  - `bindshell`, Server als Backdoor
  - Camouflage:
    - Manipulierte Programme verbergen Rootkit-Information `du, find, ifconfig, killall, ls, netstat, pidof, ps, syslogd, tcpd, top`
    - Tools zur Manipulation der Logfiles `lastlog, utmp, wtmp`
  - Packetsniffer
  - Eigene `Crontab`

Beispiel: Back Orifice

- Remote Administration “Tool” für Windows
- Server muss auf der anzugreifenden Maschine installiert werden; ermöglicht vollen Zugriff auf alle Ressourcen
- Kann, wie ein Virus, an ausführbare Dateien “angehängt” werden
- Plug-In Architektur z.B. Plug-Ins zur Verschlüsselung der Kommunikation; Übertragung des Bildschirminhalts, Steuerung der Maus, Übertragung von Videos,.....
- Aktivierung entfernter Hardware, z.B. Kamera, Sound,....

Cross Site Scripting (XSS)

- Angriff gegen Web-Anwendungen
- Angreifer schleust Code in Web-Seiten oder Lin
- Betrachter der Seite führt Schad-Code aus
- XSS häufiger als Buffer Overflows:
  - 2007: 80% aller Angriffe basierten auf XSS
- Sprachen für XSS:
  - HTML
  - XHTML
  - JavaScript
  - JScript
  - ActiveX
  - Java
  - VBScript
  - Adobe Flash
  - RSS, Atom

http://
eval.symantec.com/
mktginfo/enterprise/
white_papers/b-

whitepaper_exec_summar
y_internet_security_thre

XSS: Klassifikation

- Drei Arten von XSS (Type 0 bis Type 2)

- DOM based (Type 0) bzw. local XSS:
  - Document Object Model (DOM): Objektmodell zur Repräsentation von HTML und XML Dokumenten
  - Schadcode wird direkt an Client-Skript übergeben; Server nicht beteiligt
  - z.B. Client-Skript liest Argument aus URL und fügt dieses ungeprüft in HTML-Dokument ein das lokal ausgeführt wird

- Nicht persistent (Typ 1) bzw. reflected XSS:
  - Benutzereingabe wird vom Server direkt an Nutzer zurückgeliefert
  - Schadcode wird vom Browser interpretiert
  - z.B. dynamisch generierte Web-Seiten; Ausnutzung von HTTP-Get (URL) oder HTTP-Put (Formular)
  - Nicht persistent, da Eingabe (Schadcode) nicht gespeichert wird.

XSS Klassifikation: Type 2

- Persistent (Type 2) bzw. stored oder second order:
  - Web-Anwendung ermöglicht Speicherung von Nutzereingaben (z.B. Wiki, Foren, Anwendungen für soziale Netzwerke)
  - Schadcode wird auf Web-Server gespeichert
  - und später beim Abruf ausgeführt
  - Mächtigste Art von XSS
  - Schadcode wird ggf. vielfach ausgeführt

XSS: Beispiele

- exemplarischer Angriffsvektor:
  `<script type="text/javascript">alert("Attack“);</script>`
  erzeugt Warnungsfenster mit „Attack“ (andere Skripte möglich)

- `http://www.goodsite.com/cgi/auth.cgi?<script type="text/javascript">alert("Atack“);</script>`

- Server verpackt Angriffsvektor in Antwortseite und Client führt diese aus

XSS: Anwendungsbeispiel

1. Alice besucht Online-Banking Seite von Bob
2. Mallet weiß von XSS Schwachstelle in Bob‘s Server
3. Mallet erzeugt URL die XSS Schachstelle nutzt
4. Mallet schickt gefälschte Mail (Absender Bob) mit dem Link an Alice
5. Alice nutzt Mallets URL um Bobs Seite zu nutzen, XSS wird wirksam, Schadskript wird in Alice Browser ausgeführt
6. Skript schickt Session Cookie an Mallet
7. Mallet nutzt Cookie, um Geld zu überweisen

Sniffer: Abhören des Netzverkehrs

- Local Area Network (LAN):
  - I.d.R. gemeinsam genutztes Medium (shared medium), z.B. Ethernet, Token Ring
  - Netzwerk-Karten können im Prinzip gesamten Verkehr mithören, aber
  - geben nur die an den Rechner adressierten Packete weiter
- Gefahr: “Promiscious Mode”:
  - Einstellung der Karte
  - Im Promiscious Mode werden **alle** Pakete gelesen
  - Gefahr wird durch „geswitchtes“ Netz relativiert
- Wide Area Network (WAN):
  - Jeder Vermittlungsrechner kann Nachrichten „mitlesen“
- Tools:
  - Übergang zwischen Werkzeug des System- sowie Netzadministrators und Cracker Tool sind fließend
  - `tcpdump` Standard-Werkzeug diverser Unix Varianten
  - `wireshark` Packet-Analyzer (Unix, Windows)
  - `snoop` Sun Solaris
  - .....

Port Scanner

- Suchen auf entferntem Rechner nach „offenen“ Ports
  - Versuch eines Verbindungsaufbau / pro Port
  - Falls erfolgreich = Port „offen“
- Damit Identifikation von Diensten
- Gezielte Suche nach Rechner die Dienste mit bekannten Schwächen anbieten
- Auch hier ist der Übergang zwischen nützlichem Werkzeug und Cracker Tool fließend
- WARNUNG: Port Scan auf fremde Netze/Maschinen werden als Angriff gewertet
- Beispiele:
  - `nmap` (Unix)
  - Knocker

Inhalt (2)

3. Bedrohungen, Angriffe und Gefährdungen (Forts.)
   6. Buffer Overflow
   7. Account / Password Cracking
   8. Hintertüren / Falltüren
   9. Rootkits
   10. Cross Site Scripting (XSS)
   11. Sniffer
   12. Port Scanner
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

Rechtliche Regelungen: Strafgesetzbuch

- Strafgesetzbuch (StGB) regelt Strafrecht
- Verletzungen der Normen werden im Strafverfahren verhandelt
- Antragsdelikt: Tat wird nur auf Antrag (Anzeige) i.d.R. durch den „Verletzten“ (§ 77 ff.) verfolgt (§ 202a, 202b, 303a, 303b)
- Offizialdelikt: Tat wird „von Amts wegen“ (Staatsanwaltschaft) verfolgt (§ 202c)

- § 202a: Ausspähen von Daten
- § 202b: Abfangen von Daten (seit 11.08.2007)
- § 202c: Vorbereiten des Ausspähens und Abfangens von Daten (seit 11.08.2007)
- § 303a: Datenveränderung
- § 303b: Computersabotage
- § 303c: Strafantrag

§ 202a StGB: Ausspähen von Daten

(1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.

§ 202b StGB: Abfangen von Daten

Wer unbefugt sich oder einem anderen unter Anwendung von technischen Mitteln nicht für ihn bestimmte Daten (§ 202a Abs. 2) aus einer nichtöffentlichen Datenübermittlung oder aus der elektromagnetischen Abstrahlung einer Datenverarbeitungsanlage verschafft, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft, wenn die Tat nicht in anderen Vorschriften mit schwererer Strafe bedroht ist.

§ 202c StGB: Vorbereitung des Abfangens oder Ausspähens von Daten („Hackerparagraph“)

(1) Wer eine Straftat nach § 202a oder § 202b vorbereitet, indem er

1. Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§ 202a Abs. 2) ermöglichen, oder
2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist,

herstellt, sich oder einem anderen verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft

(2) § 149 Abs. 2 und 3 gilt entsprechend.
(Vorbereitung der Fälschung von Geld und Wertzeichen)

- **Offizialdelikt**

§205 StGB: Strafantrag

(1) In den Fällen des § 201 Abs. 1 und 2 und der §§ 201a, 202, 203 und 204 wird die Tat nur auf Antrag verfolgt. Dies gilt auch in den Fällen der §§ 202a und 202b, es sei denn, dass die Strafverfolgungsbehörde wegen des besonderen öffentlichen Interesses an der Strafverfolgung ein Einschreiten von Amts wegen für geboten hält.

- § 202c fehlt in dieser Aufzählung; d.h. 202c ist Offizialdelikt

§ 303a StGB: Datenveränderung

(1) Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.

(2) Der Versuch ist strafbar.

(3) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

§ 303b StGB: Computersabotage

(1) Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er

1. eine Tat nach § 303a Abs. 1 begeht,
2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt oder
3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,

wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Handelt es sich um eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, ist die Strafe Freiheitsstrafe bis zu fünf Jahren oder Geldstrafe.

§ 303b StGB: Computersabotage (Forts.)

(3) Der Versuch ist strafbar.

(4) In besonders schweren Fällen des Absatzes 2 ist die Strafe Freiheitsstrafe von sechs Monaten bis zu zehn Jahren. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter

1. einen Vermögensverlust großen Ausmaßes herbeiführt,
2. gewerbsmäßig oder als Mitglied einer Bande handelt, die sich zur fortgesetzten Begehung von Computersabotage verbunden hat,
3. durch die Tat die Versorgung der Bevölkerung mit lebenswichtigen Gütern oder Dienstleistungen oder die Sicherheit der Bundesrepublik Deutschland beeinträchtigt.

(5) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

§ 303c StGB: Strafantrag

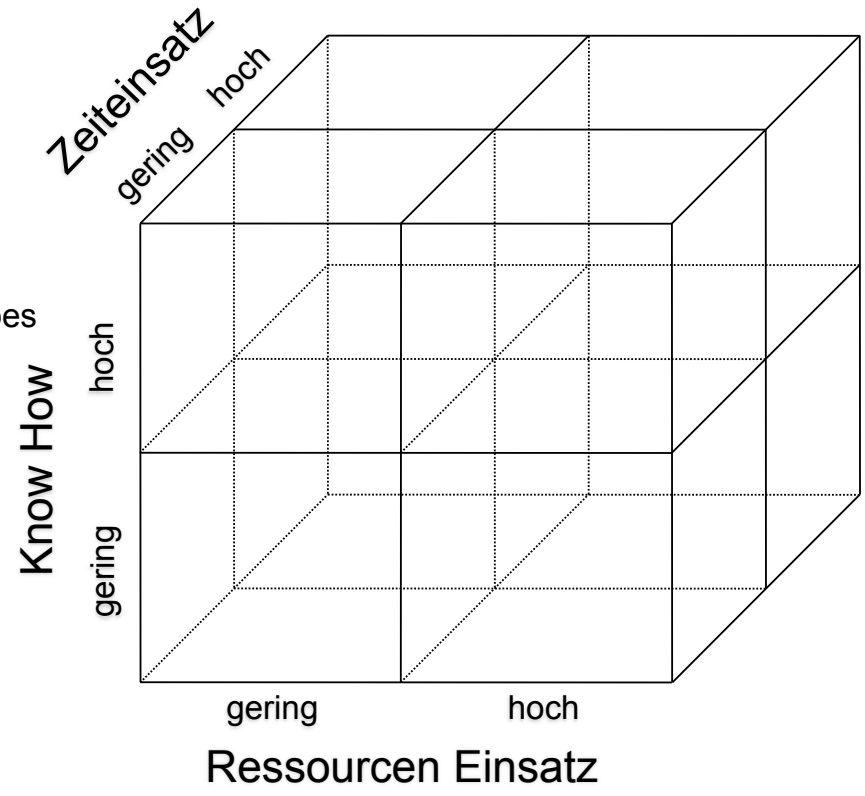
In den Fällen der §§ 303, 303a Abs. 1 und 2 sowie § 303b Abs. 1 bis 3 wird die Tat nur auf Antrag verfolgt, es sei denn, dass die Strafverfolgungsbehörde wegen des besonderen öffentlichen Interesses an der Strafverfolgung ein Einschreiten von Amts wegen für geboten hält.

Inhalt (2)

3. Bedrohungen, Angriffe und Gefährdungen (Forts.)
   6. Buffer Overflow
   7. Account / Password Cracking
   8. Hintertüren / Falltüren
   9. Rootkits
   10. Cross Site Scripting (XSS)
   11. Sniffer
   12. Port Scanner
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

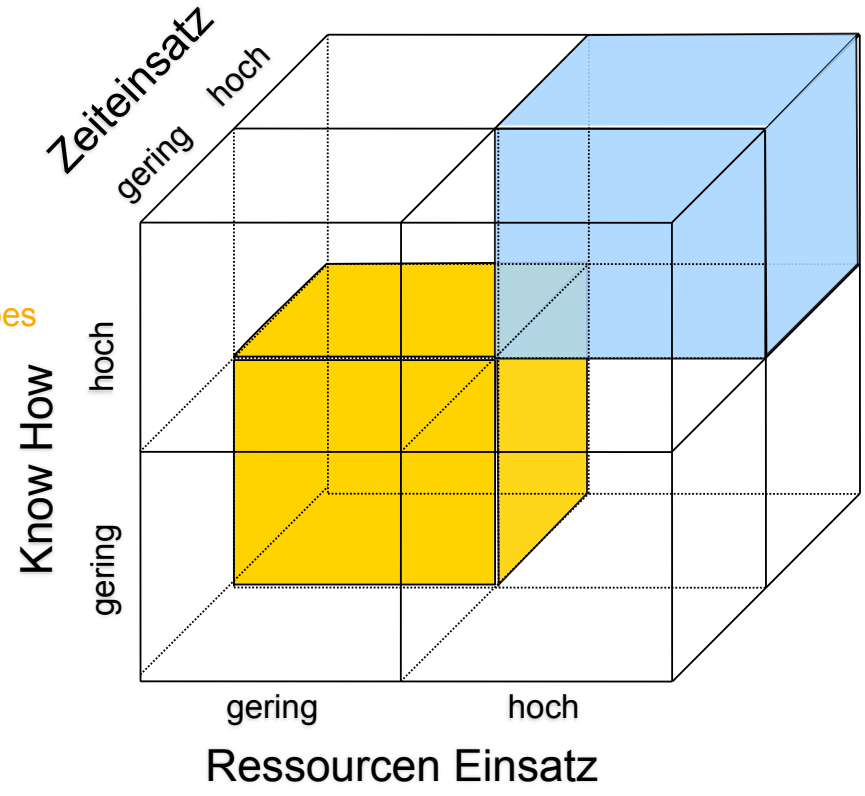
Klassifikation der Angreifer

- Einteilung nach:
  - Know How
  - Ressourcen Einsatz
  - Zeiteinsatz
- Beispiele:
  - Skript Kiddie oder Wannabes
  - Einzelperson (Cracker)
  - Wirtschaftsspionage
  - Nachrichtendienste



Klassifikation der Angreifer

- Einteilung nach:
  - Know How
  - Ressourcen Einsatz
  - Zeiteinsatz
- Beispiele:
  - Skript Kiddie oder Wannabes
  - Einzelperson (Cracker)
  - Wirtschaftsspionage
  - Nachrichtendienste



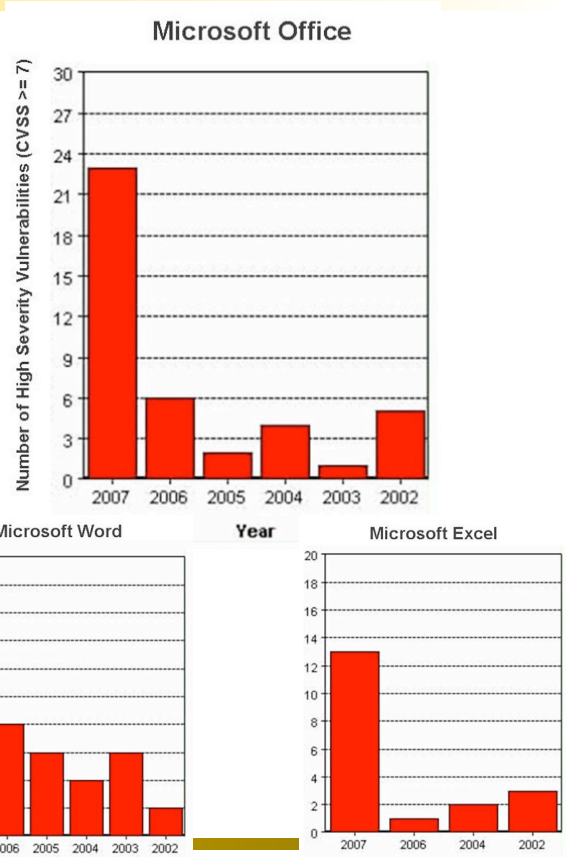
Twenty Most Vulnerabilities

- SANS Institute (System Administration, Audit, Networking and Security) SANS/FBI Top twenty list
  www.sans.org
- letzte Version 8.0; 28. Nov., 2007
  - Im folgenden die alten Folien zur Info
- Jetzt:
  The Top Cyber Security Risks
  (Stand: September 2009)
- Quellen:
  - Angriffsdaten von IPS Systemen aus 6.000 Organisationen
  - Vulnerability data from 9 Mio. Systems
  - Information aus Internet Strom Center des SANS

Twenty most vulnerabilites (V 8.0)

- SANS Institute (System Administration, Audit, Networking and Security)
  SANS/FBI Top twenty list
  www.sans.org
- Version 8.0; 28. Nov., 2007
- Client-side Vulnerabilities in

1. Web-Browsers
   - Bugs in Internet Explorer and Firefox
   - Hundreds of malicious Active X controls
   - Vulnerabilities in PlugIns
2. Office Software
   - Malformed Documents
   - Various Bugs



Twenty most vulnerabilites (V 8.0)

3. Email Clients
   - Distribution of Malware
   - Phishing, Spam
   - Social Engineering
   - DoS; Mail-bombing
4. Media Players
   - Vulnerabilities in most popular media player
   - Forgotten application

■ Server Side Vulnerabilities

1. Web Applications
   - CMs, Wikis, Bulletin Boards, Social Networks, ...
   - PHP Remote File Include
   - SQL Injection
   - Cross Site Scripting (XSS)
   - Cross-site request forgeries (CSRF)
2. Windows Services
   - Service Control Programm (SCP) Service Control Manager (SCM)
   - RPC and CIFS Vulnerabilities
   - Default enabled services
   - Built in local accounts (Local System, Local Service, Local Network)
3. Unix/MAC OS Services:
   - Brute force Attacks against Remote Services (ssh, ftp, telnet)
4. Backup Software:
   - Bugs in popular client server based backup software
   - CA BrightStore ARCServe
   - Symantec Veritas NetBackup
   - EMC Legato Networker

Twenty most vulnerabilites (V 8.0), cont.

5. Anti-Virus Software and Firewalls
   - Multiple remote code exec
   - Evasion Attacks; specially crafted files bypass sec. systems
6. Applications affected
   - Directory Servers (steal passwords)
   - Monitoring Systems; exploit Clients
   - Configuration and Patch Systems
7. Database Software
   - Default config with default passw.
   - SQL injection
   - Weak passwords
   - Buffer overflows

■ Security Policies and Personal

1. Excessive User Rights and Unauthorized Devices
   - Users can do unsafe things
   - Rogue wirelss access point
   - Personal laptop
   - Unmanaged Software
   - P2P Clients, ...
2. Phishing / Spear Phishing
   - Identity Theft
   - Information about staff or organization in phishing mail
   - Voice Phishing (call Phone Nr....)
3. Unenctypted Laptops and Removable Media
   - ID exposure
   - etc.

Twenty most vulnerabilites (V 8.0), cont.

- Application Abuse

1. Instant Messaging
   - Malware
   - Information theft
   - DoS
   - Vulnerabilities in IM
2. P2P and File Sharing
   - Malware
   - Information Manipulation
   - Attacking Techniques could be found in Cloud Computing and Grids

- Network Devices

1. VoIP Servers and Phones
   - Vulnerabilities in Cisco Call Manager and Asterisk
   - VoIP Phishing, eaversdropping, toll fraud, DoS

- Zero Day Attacks
  - Several Zero Day Attacks have been seen

Twenty most Vulnerabilities: How to protect?

- Close most dangerous holes first
- Install and make accessible only necessary software, services and ports
- Install latest security patches
- Establish and control fulfillment of security policy (all)
- Appropriate configuration
- Install a firewall
- Eliminate sample and unneccessary applications
- Filter HTTP requests
- Disable weak services
- Continious information about security weaknesses .....

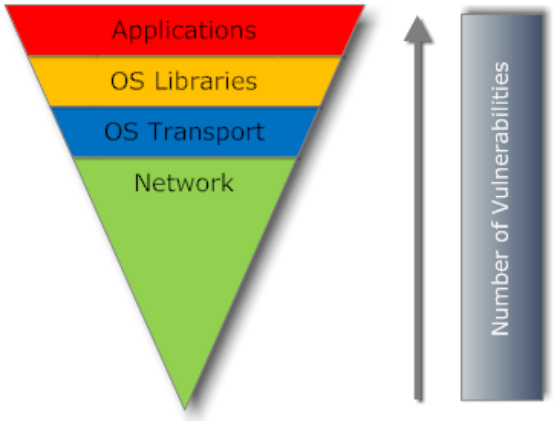
- Diese Hinweise stellen (mehr oder weniger) allgemeine (Sicherheits-)Grundsätze dar.
- Frage: Welche allgemeineren Schutzmechanismen (Sicherheitsanforderungen) – unabhängig von konkreten Angriffen – gibt es?

Top Cyber Security Risks: Executive Summary

- Priority One: Client-side software that remains unpatched
  - PDF Reader, QuickTime, Flash, Microsoft Office
- Priority Two: Internet-facing web sites that are vulnerable
  - More of 60% of the total attacks attempt
  - SQL injection, XSS,
- Operating systems continue to have fewer remotley-exploitable vulnerabilities that lead to massive Internet worms
  - Other than Conficker, no new worms for OSs
- Rising number of zero-day vulnerabilities

Top Cyber Security Risks: Vulnerability Trends (1)

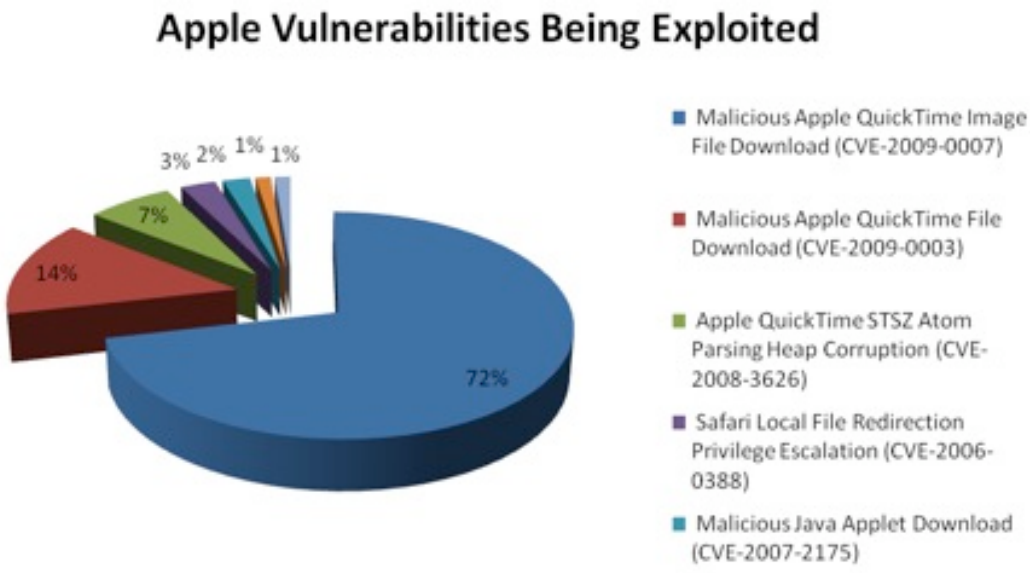
- Application Vulnerability Exceed OS Vulnerabilities



- Brute force password guessing and web application attacks
  - Microsoft SQL, FTP and SSH password guessing
  - SQL Injection, XSS and PHP File Include

Top Cyber Security Risks: Vulnerability Trends (2)

- Conficker/Downadup
  - 90 % Buffer overflow attack (Microsoft Security Bulletin MS08-067)
  - Sasser (2003) und Blaster (2004) continue to infect
- Apple: QuickTime and Six More



Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  1. WordPad and Office Text Converters Remote Code Execution Vulnerability (MS09-010)
  2. Sun Java Multiple Vulnerabilities (244988 and others)
  3. Sun Java Web Start Multiple Vulnerabilities May Allow Elevation of Privileges(238905)
  4. Java Runtime Environment Virtual Machine May Allow Elevation of Privileges (238967)
  5. Adobe Acrobat and Adobe Reader Buffer Overflow (APSA09-01)
  6. Microsoft SMB Remote Code Execution Vulnerability (MS09-001)
  7. Sun Java Runtime Environment GIF Images Buffer Overflow Vulnerability
  8. Microsoft Excel Remote Code Execution Vulnerability (MS09-009)
  9. Adobe Flash Player Update Available to Address Security Vulnerabilities (APSB09-01)
  10. Sun Java JDK JRE Multiple Vulnerabilities (254569

Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  11. Microsoft Windows Server Service Could Allow Remote Code Execution (MS08-067)
  12. Microsoft Office PowerPoint Could Allow Remote Code Execution (MS09-017)
  13. Microsoft XML Core Services Remote Code Execution Vulnerability (MS08-069)
  14. Microsoft Visual Basic Runtime Extended Files Remote Code Execution Vulnerability (MS08-070)
  15. Microsoft Excel Multiple Remote Code Execution Vulnerabilities (MS08-074)
  16. Vulnerabilities in Microsoft DirectShow Could Allow Remote Code Execution (MS09-028)
  17. Microsoft Word Multiple Remote Code Execution Vulnerabilities (MS08-072)
  18. Adobe Flash Player Multiple Vulnerabilities (APSB07-20)
  19. Adobe Flash Player Multiple Security Vulnerabilities (APSB08-20)

Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  20. Third Party CAPICOM.DLL Remote Code Execution Vulnerability
  21. Microsoft Windows Media Components Remote Code Execution Vulnerability (MS08-076)
  22. Adobe Flash Player Multiple Vulnerabilities (APSB07-12)
  23. Microsoft Office Remote Code Execution Vulnerability (MS08-055)
  24. Adobe Reader JavaScript Methods Memory Corruption Vulnerability (APSA09-02 and APSB09-06)
  25. Microsoft PowerPoint Could Allow Remote Code Execution (MS08-051)
  26. Processing Font Vulnerability in JRE May Allow Elevation of Privileges (238666)
  27. Microsoft Office Could Allow Remote Code Execution (MS08-016)
  28. Adobe Acrobat/Reader "util.printf()" Buffer Overflow Vulnerability (APSB08-19)
  29. Adobe Acrobat and Adobe Reader Multiple Vulnerabilities (APSB08-15)
  30. Windows Schannel Security Package Could Allow Spoofing Vulnerability (MS09-007)

Zero-Day Vulnerability Trends

- Code exploiting a flaw appears before a fix or patch is available
- First Choice: „File Format Vulnerabilities“:
  - Adobe Acrobat, Reader, and Flash Player Remote Code Execution Vulnerability (CVE-2009-1862)
  - Microsoft Office Web Components ActiveX Control Code Execution Vulnerability (CVE-2009-1136)
  - Microsoft Active Template Library Header Data Remote Code Execution Vulnerability (CVE-2008-0015)
  - Microsoft DirectX DirectShow QuickTime Video Remote Code Execution Vulnerability (CVE-2009-1537)
  - Adobe Reader Remote Code Execution Vulnerability (CVE-2009-1493)
  - Microsoft PowerPoint Remote Code Execution Vulnerability (CVE-2009-0556)

Twenty Critical Controls for Effective Cyber Defense

- Controls Subject to Automated Collection, Measurement and Validation:
  1. Inventory of Authorized and Unauthorized Devices
  2. Inventory of Authorized and Unauthorized Software
  3. Secure Configurations for Hardware and Software on Laptops, Workstations, and Servers
  4. Secure Configurations for Network Devices such as Firewalls, Routers, and Switches
  5. Boundary Defense
  6. Maintenance, Monitoring, and Analysis of Security Audit Logs
  7. Application Software Security
  8. Controlled Use of Administrative Privileges
  9. Controlled Access Based on Need to Know
  10. Continuous Vulnerability Assessment and Remediation
  11. Account Monitoring and Control
  12. Malware Defenses

Twenty Critical Controls for Effective Cyber Defense

- Controls Subject to Automated Collection, Measurement and Validation:
  13. Limitation and Control of Network Ports, Protocols, and Services
  14. Wireless Device Control
  15. Data Loss Prevention
- Controls not directly supported by automated collection:
  16. Secure Network Engineering
  17. Penetration Tests and Red Team Exercises
  18. Incident Response Capability
  19. Data Recovery Capability
  20. Security Skills Assessment and Appropriate Training to Fill Gaps

- Weitere Infos:
  http://www.sans.org/critical-security-controls/

Rückblick: OSI-Sicherheitsdienste

Authentication	Peer Entity
	Data Origin
Access Control	
Data Confidentiality	Connection
	Connectionless
	Selective field
	Traffic flow
Data Integrity	Connection
	Connectionless
	Selective field
	Recovery
Non-Repudiation	Proof of origin
	Proof of delivery

- **Identifikation (Identification);** Personalisierung: Zweifelsfreie Verbindung zwischen digitaler ID und Real World Entity (Person oder Organization)
- **Autorisierung (Authorization):** Erteilung von Rechten an Entities
- **Zurechenbarkeit (Accountability)**
- **Anonymität (Anonymity)**
- (Verfügbarkeit (Availability))
- **Ressourcenbeschränkung (Ressource constraints)**

- **Notwendigkeit einzelner Sicherheitsanforderungen muß in konkreter Umgebung untersucht werden.**

Zusammenfassung

- Security Engineering
  - Mehrstufiger, iterativer und kontinuierlicher Prozess
  - In diesem Abschnitt Fokus auf Bedrohungsanalyse: Überblick über mögliche Angriffe und Schwächen
  - Risiko-Priorisierung (wurde nicht vertieft) und dann Ableitung von (allgemeineren) Sicherheitsanforderungen

- In den nächsten Abschnitten:
  - Ableitung und Bewertung von Sicherheitsmechanismen zur Durchsetzung von Sicherheitsanforderungen
  - Zuerst grundlegende Techniken der Kryptologie

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 4: Grundlagen der Kryptologie

Inhalt

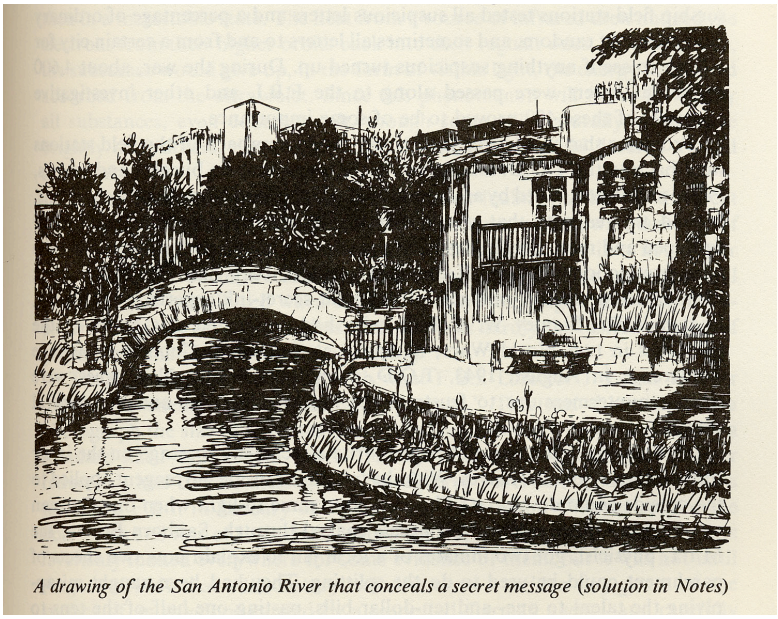
1. Kryptologie: Begriffe, Klassifikation
2. Steganographie
3. Kryptographie, Begriffe und Definitionen
   - Kryptosystem
   - Substitution
   - Permutation
   - Symmetrische versus asymmetrische Kryptosysteme
   - Kryptoanalyse
   - Abschätzung Brute-Force Angriff

Kryptologie: Begriffe, Klassifikation

- **Kryptographie**: Lehre von den Methoden zur Ver- und Entschlüsselung von Nachrichten
- **Kryptoanalyse, Kryptanalyse**: Wissenschaft von den Methoden zur Entschlüsselung ohne im Besitz den Schlüssels zu sein (Angriffe auf Kryptographische Verfahren)
- **Kryptologie** = Kryptographie + Kryptoanalyse
- **Kryptographische Protokolle**: Protokolle, die kryptographische Techniken verwenden, um z.B. Schlüssel auszutauschen, Kommunikationspartner zu authentisieren, ...
- **Steganographie** (verdecktes Schreiben): Methoden die bereits die Existenz der geheimen Nachricht verbergen (geheime Nachricht in anderen nicht geheimen „Nachrichten“ verbergen) Unterscheidung in **linguistische** und **technische** Steganographie

Linguistische Steganographie

- **Semagramme**: Nachrichten, die in **Details** von Schriften oder Bildern verborgen sind.
- Bsp. aus David Kahn: *The Codebreakers,* Scribner, 1996



A drawing of the San Antonio River that conceals a secret message (solution in Notes)

- Wo verbirgt sich die Nachricht?
- Wie lautet diese?

Linguistische Steganographie (Forts.)

- **Maskierung (Open Code):**
  Nachricht verborgen in offen übertragener, unverfänglicher Nachricht
  (z.B. Husten in „Wer wird Millionär“)
  - **Stichworte**: Begriff, Satzteil oder Satz mit vorher vereinbarter Bedeutung;
    z.B. *HIGASHI NO KAZE AME* („Ostwind, Regen“) im japanischen Wetterbericht - zwei mal wiederholt - sollte „Krieg mit USA“ bedeuten
- **Jargon, Millieu Code**:
  Sondersprachen oder Sonderzeichen beruflicher oder gesellschaftlicher Art
  - Bettler, Vagabunden und Gauner:
    Rotwelsch (Deutschland), Argot (Frankreich), ...
    z.B. „Schnee“ für Kokain; „Kies“ für Geld; „abstauben“ ,.....

  Für Zensoren durch „gestelzte“ Sprache relativ leicht erkennbar

Technische Steganographie

- Herodot (490 v.Chr.): Nachricht auf den rasierten Schädel eines Sklaven tätowiert
- Alle Arten von „Geheimtinten“
- Steganographie in digitalen Bildern; Beispiele mit `outguess`

Original



Steganographie



Steganographie in Bildern

- **Cover** = Bild in das die Nachricht eingebettet wird
- Finde redundante Bits im Cover
  - Least Significant Bits
  - „Rauschen“
  - Nahe zusammenliegende Farben
- Kodieren der Nachricht in diesen redundanten Bits

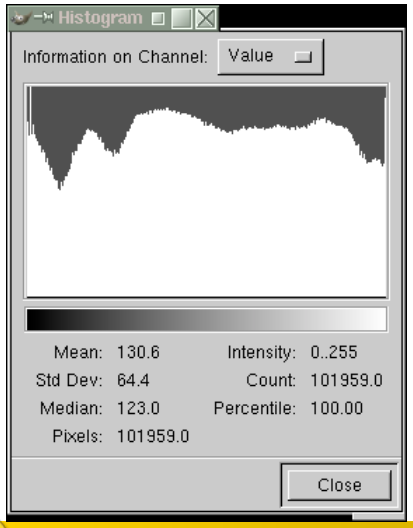
- Steganographie führt zu “sehr geringen Veränderungen” im Bild

Steganographie; Veränderungen im Bild

- Histogramm:

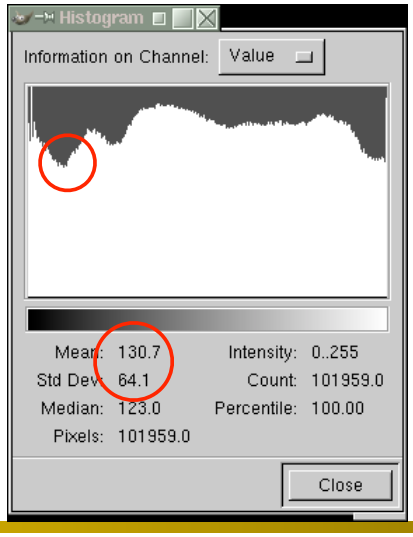
Orginal





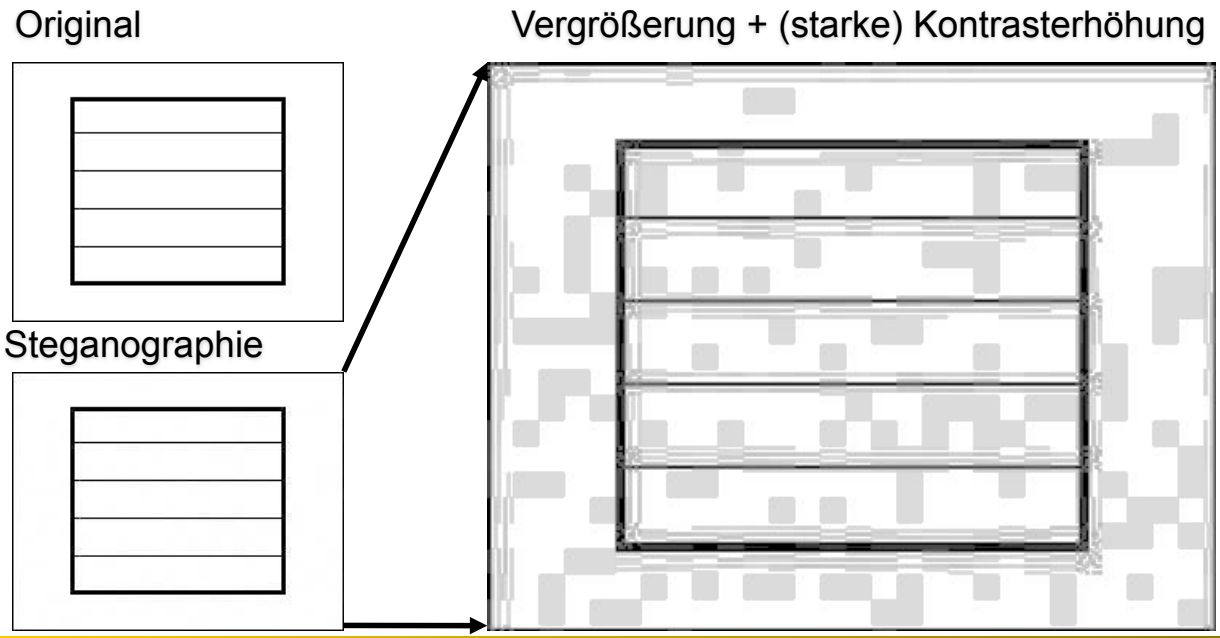
Steganographie





Steganographie; Merkmale

- Unterschiede bei “sehr strukturierten Bildern” mit hohem versteckten Datenvolumen evtl. erkennbar

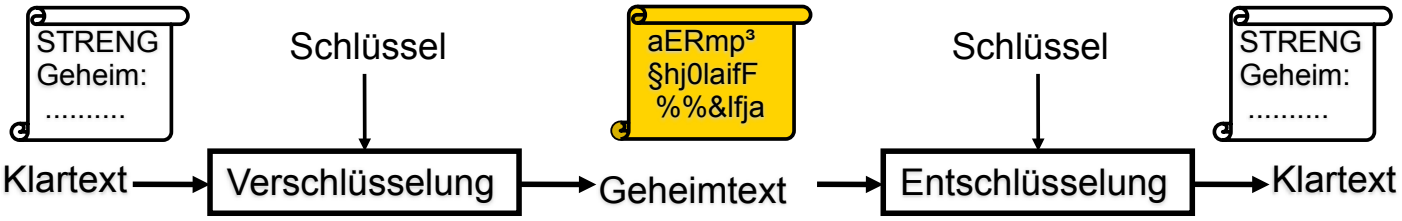


Inhalt

1. Kryptologie: Begriffe, Klassifikation
2. Steganographie
3. Kryptographie, Begriffe und Definitionen
   - Kryptosystem
   - Substitution
   - Permutation
   - Symmetrische versus asymmetrische Kryptosysteme
   - Kryptoanalyse
   - Abschätzung Brute-Force Angriff

Kryptographie, Begriffe

- **Klartext** (**Plaintext**): die zu verschlüsselnde Nachricht
- **Geheimtext (Ciphertext):** verschlüsselte Nachricht
- **Verschlüsselung, Chiffrierung (Encryption)**: Vorgang der Klar- in Geheimtext überführt
- **Entschlüsselung, Dechiffrierung (Decryption):** Überführung von Geheim- in Klartext
- **Chiffriersystem (Cryptographic Algorithm, Cipher):** Algorithmisches Verfahren zur Ver- bzw. Entschlüsselung
- Benötigen **Schlüssel (Key)**



Kryptographisches System (Def.)

- Geg. zwei endliche Zeichenvorräte (Alphabete) A_1 und A_2
- Ein Kryptosystem (KS) ist gegeben durch ein Tupel

$$KS = (M, C, EK, DK, E, D)$$

1. Nicht leere endliche Menge von Klartexten $M \subseteq A_1^*$ mit A_1^* Menge aller Worte über dem Alphabet A_1
2. nicht leere endliche Menge von Krypto- bzw. Chiffrentexten $C \subseteq A_2^*$
3. der nicht leeren Menge von Verschlüsselungsschlüsseln EK
4. der nicht leeren Menge von Entschlüsselungsschlüsseln DK sowie einer Bijektion $\mathrm{f}: \mathrm{EK} \rightarrow \mathrm{DK}$ Diese assoziiert zu jedem Verschlüsselungsschlüssel $K_E \in EK$ einen dazu passenden Entschlüsselungsschlüssel $K_D \in DK$, d.h. $f(K_E) = K_D$

Kryptographisches System (Def.); Forts.

- Kryptosystem (KS)

$$KS = (M, C, EK, DK, E, D)$$

5. Dem injektiven Verschlüsselungsverfahren

$$E : M \times EK \rightarrow C$$

6. Dem Entschlüsselungsverfahren

$$D : C \times DK \rightarrow M$$

mit der Eigenschaft, dass für zwei Schlüssel $K_E \in EK$ und $K_D \in DK$ mit $f(K_E) = K_D$ gilt:

$$\forall m \in M : D(E(m, K_e), K_d)) = m$$

D.h. ein bel. Klartext m der mit einem Verschlüsselungsschlüssel verschlüsselt wurde, kann mit dem passenden Entschlüsselungsschlüssel wieder entschlüsselt werden

Kryptosystem, Bsp.: Substitution

- **Substitution**: $f : A_1^n \rightarrow A_2^m$
- Alphabete: $A_1 = \{a, b, \ldots, z\} (= Z_{25}); A_2 = \{1,2,3,4,5\}$
- Verschlüsselungsverfahren: $E : A_1^1 \rightarrow A_2^2$
- Schlüssel $K_E = K_D$

	1	**2**	**3**	**4**	**5**
1	a	b	c	d	e
2	f	g	h	i	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

- Bsp.:
  42114213324354345 1 = iailhouse (Jailhouse)

Kryptosystem, Bsp.: Permutation

- **Permutation** als Spezialfall der Substitution: $f : A^n \rightarrow A^n$
  gleiche Wortlänge; gleiche Alphabete $A_1 = A_2 = \{a, b, ..., z\}$
- $K_E = K_D$ (NEWYORK,1) (+ Alg. zur Anwendung)
  (Zur besseren Lesbarkeit, werden Chiffrentexte trotzdem oft in Großbuchstaben dargestellt.)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
N	E	W	Y	O	R	K	A	B	C	D	F	G	H	I	J	L	M	P	Q	S	T	U	V	X	Z

  Zyklenschreibweise:
  (a,n,h) (b,e,o,i) (c,w,u,s,p,j) (d,y,x,v,t,q,l,f,r,m,g,k)
- Bsp.:
  QAOWI YOEMO NDOMP = thecodebreakers
  Chiffrentext wird in Blöcken übertragen
  Leer- und Satzzeichen werden nicht kodiert (Leerzeichen noch häufiger als „e“)

Kryptosystem: Symmetrische Verfahren

- Ver- und Entschlüsselungsschlüssel gleich, oder leicht voneinander ableitbar
- Kommunikationspartner teilen **gemeinsamen, geheimen Schlüssel (symmetrisch)**
- Setzt vorherige Verständigung (**Schlüsselaustausch**) voraus
- Protokoll:
  1. Alice und Bob vereinbaren („**out of band**“) gemeinsamen Schlüssel:
     $K_E = K_D = K_{A,B}$
  2. Alice verschlüsselt m: $C = E(m, K_{A,B})$ und sendet C an Bob
  3. Bob entschlüsselt C:

$$m = D(C, K_{A,B}) = D(E(m, K_{A,B}), K_{A,B})$$

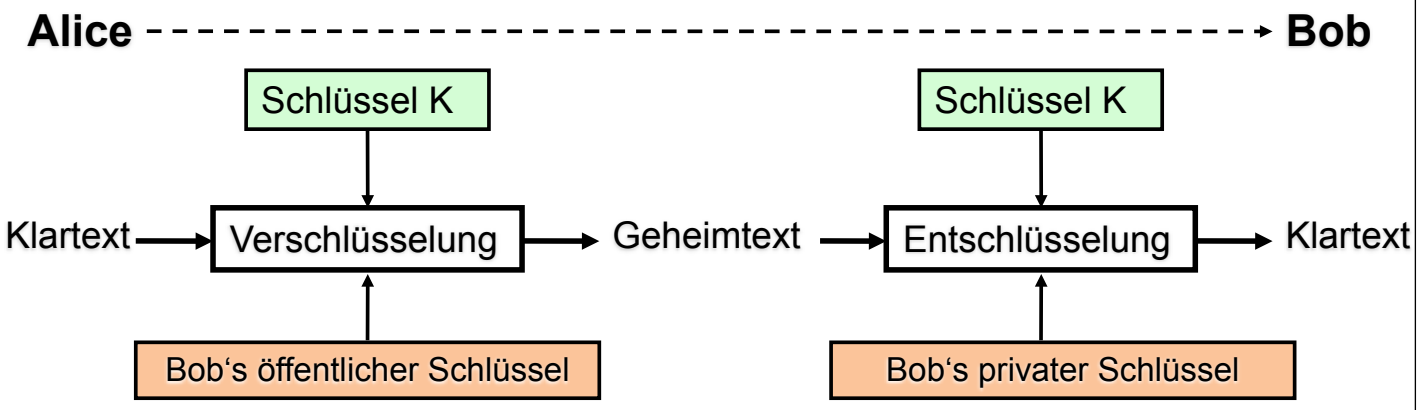
- Beispiele: DES, AES, IDEA,.....

Kryptosystem: Asymmetrische Verfahren

- Jeder Partner besitzt **Schlüsselpaar** aus
  - persönlichem, **geheim** zu haltendem **Schlüssel** (*private key*) (wird NIE übertragen)
  - und **öffentlich** bekannt zu gebendem **Schlüssel** (*public key*) (kann über unsichere und öffentliche Kanäle übertragen werden)
- Protokoll:
  1. Alice und Bob erzeugen sich Schlüsselpaare:
     $$(K_E^A, K_D^A); (K_E^B, K_D^B)$$
  2. Öffentliche Schlüssel werden öffentlich zugänglich gemacht
  3. Alice will m an Bob senden; dazu benutzt sie B‘s öffentlichen Schlüssel
     $$C = E(m, K_E^B)$$
  4. Bob entschlüsselt die Nachricht mit seinem privaten Schlüssel:
     $$m = D(C, K_D^B) = D(E(m, K_E^B), K_D^B)$$
- Bsp.: RSA, ElGamal,...

Vergleich Symmetrische / Asymmetrische Verfahren

	Symmetrisch	Asymmetrisch
Schlüsselaustausch	Sicherer Kanal erforderlich	öffentlich
Schlüssellänge	128 bis 256 Bit	1024 bis 4096 Bit
Geschwindigkeit		Faktor 100 bis 1000 langsamer



Kryptoanalyse

- Wissenschaft von Methoden zur Entschlüsselung **ohne** Schlüssel
- Klassen kryptographischer Angriffe:
  - **Brute force; exhaustive search:** vollständiges Durchsuchen des Schlüsselraumes
  - **Klartext Angriff (ciphertext-only):** Dem Analytiker stehen mehrere Chiffren zur Verfügung. Ziel: Schlüssel und/oder Klartext berechnen
  - **Bekannter Klartext (known-plaintext):** Analytiker kennt Klartext-/Chiffren-Kombinationen die mit selbem Schlüssel verschlüsselt wurden. Ziel: Schlüssel brechen oder Algorithmus der jede mit dem Schlüssel verschlüsselte Nachricht entschlüsseln kann
  - **Gewählter Klartext (chosen-plaintext):** Analytiker kann selber Klartexte wählen und diese verschlüsseln lassen.
  - **Gewählte Chiffre (known-ciphertext):** Angreifer kann sich zu ausgewählten Chiffren, den Klartext berechnen lassen
- Weitere Informationen: Vgl. F.L. Bauer: Entzifferte Geheimnisse

Einschub: Abschätzung Brute-Force Angriff

- Der Schlüssellänge sei 128 Bit
- Ihr Rechner ist in der Lage 50.000 Verschlüsselungs-operationen pro Sekunde durchzuführen
- Wie viele Jahre dauert ein Brute-Force-Angriff?
- Schlüsselraum $S = 2^{128} \approx 3,4 \cdot 10^{38}$
  340.282.366.920.938.463.463.374.607.431.768.211.456
- 1 Jahr hat rund 31.536.000 Sekunden
- (S / 50.000) / 31.536.000 Sekunden $\approx 2 \cdot 10^{27}$
  215.805.661.416.120.283.779.410.583,099802 Jahre
- Wieviele Schlüssel müssen Sie pro Sek. berechnen um „nur“ 100 Jahre zu brauchen?
- ~ 107.902.830.708.060.141.889.705.291.549,901
  $\approx 1,1 \cdot 10^{30}$

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 5: Symmetrische Kryptosysteme

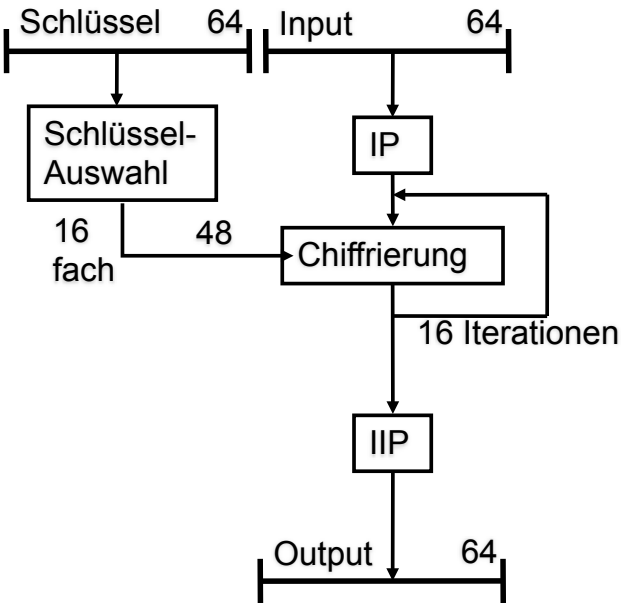
Inhalt

- Symmetrische Kryptosysteme
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)

DES (Data Encryption Standard)

- 1977 vom NBS (National Bureau of Standards) heute National Institute of Standards (NIST) in USA zum Standard erklärt
- 2002 durch AES (Advanced Encryption Standard) ersetzt
- DES entwickelt von IBM aus dem 128 Bit Verfahren LUCIFER
- Klassifikation:
  - Symmetrisches Verfahren
  - mit Permutation, Substitution und bitweiser Addition modulo 2
  - Blockchiffre mit 64 Bit großen Ein- und Ausgabeblöcken
  - Schlüssellänge 64 Bit, davon 8 Paritätsbits, d.h. effektive Schlüssellänge 56 Bit

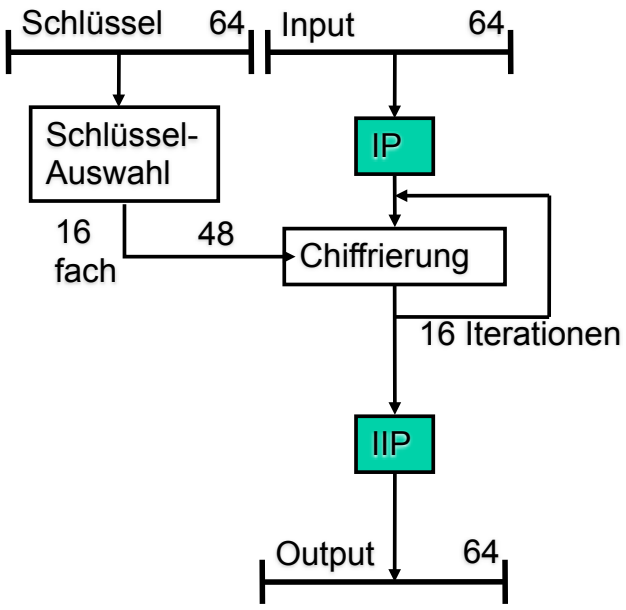
DES Funktion: Grundschema



- Ablauf der Verschlüsselung
1. Initialpermutation (IP) des Input-Blocks
2. 16 schlüsselabhängige Iterationen
   - 48 Bit lange Teilschlüssel
   - werden aus 64 Bit Schlüssel generiert
3. Inverse Initialpermutation (IIP) als Ausgangspermutation

- Entschlüsselung analog zur Verschlüsselung mit Schlüssel in umgekehrter Reihenfolge im Schritt 2.

DES Funktion: Grundschema



- Wie arbeiten Initialpermutation (IP) und Inverse Initialpermutation (IIP)?

DES: IP und IIP

- Initialpermutation IP

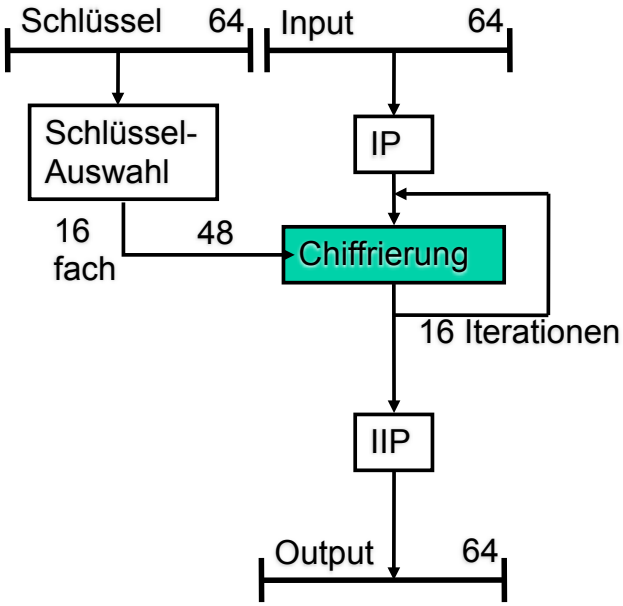
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- Inverse Initialpermutation IIP

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

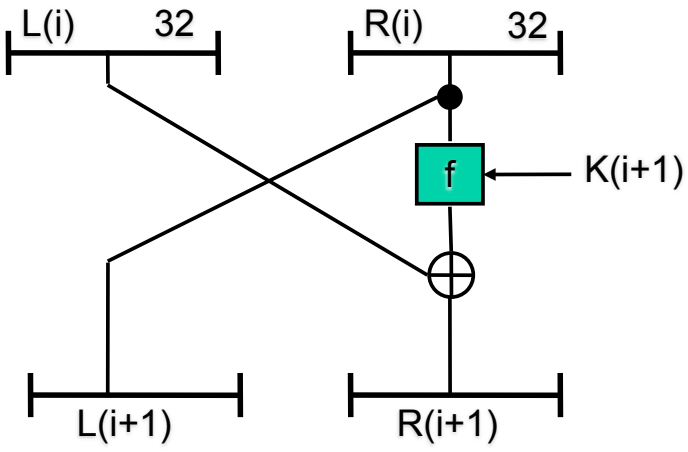
- D.h. aus Bit 58 des Input wird Bit 1, aus Bit 50 wird Bit 2,....., aus Bit 7 wird Bit 64

DES Funktion: Grundschema



DES Funktion: Verschlüsselungsiteration

- Ein Schritt (Runde) der Chiffrierung:

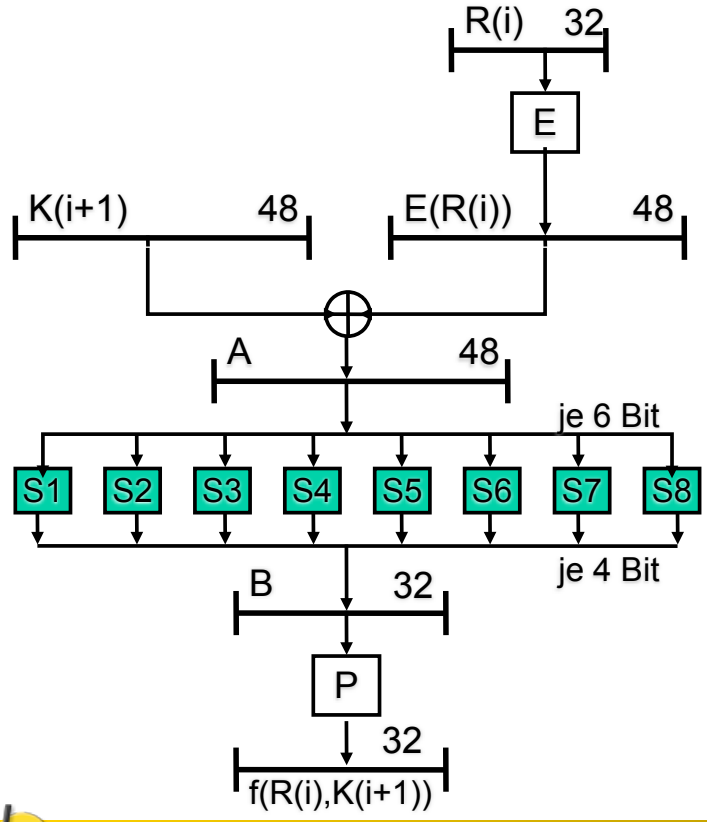


⊕ Addition modulo 2; entspr. XOR

- Verschlüsselungsblock (64 Bit) wird in linken (L) und rechten (R) Block a 32 Bit aufgeteilt
- Anwendung der Verschlüsselungs-iteration:
  L(0) = L und R(0) = R
  L(i+1) = R(i)
  R(i+1) = L(i) XOR f(R(i),K(i+1))

  für i=0,....,15
- Funktion f stellt Kern des Verfahrens dar.

DES Funktion f



- Rechter 32 Bit Input Block wird mittels Expansion E auf 48 Bit expandiert
- XOR Verknüpfung mit dem (Runden-) Schlüssel zum 48 Bit Block A
- A wird in 8 Blöcke zu 6 Bit aufgeteilt
- Jeder dieser Blöcke wird durch S-Box (Substitution) in 4 Bit Ausgabeblöcke (nichtlinear) abgebildet
- Konkatenation der acht 4 Bit Blöcke ergibt Block B der noch der Ausgangspermutation P unterworfen wird

Expansion E und Permutation P

- Expansion E:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

  - Bit 32 aus R(i) wird Bit 1 von E(R(i))

- Ausgangspermutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

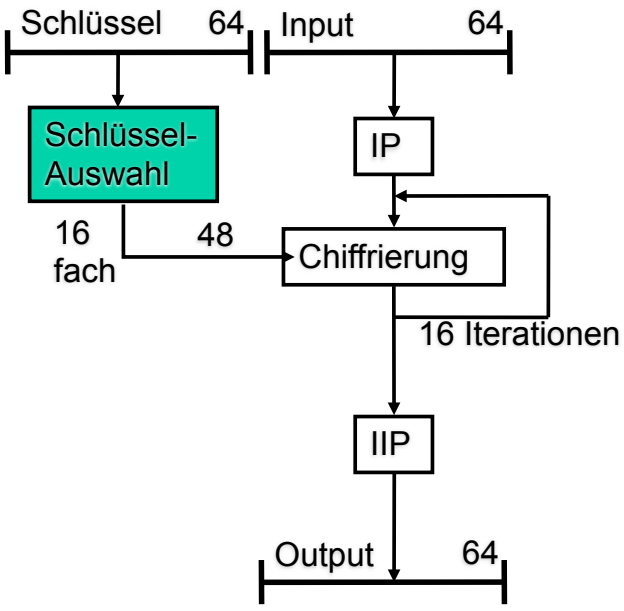
DES S-Boxen

- 6 Bit Input Block (i1,i2,i3,i4,i5,i6) wird auf 4 Bit Outputblock (o1,o2,o3,o4) abgebildet:
  - Redundante Bits (i1,i6) des Inputblocks bestimmten die Zeile der entspr. S-Box
  - Bits (i2,i3,i4,i5) bestimmen Spalte
  - Element in der Matrix bestimmt Outpublock
- Bsp. S-box S1:

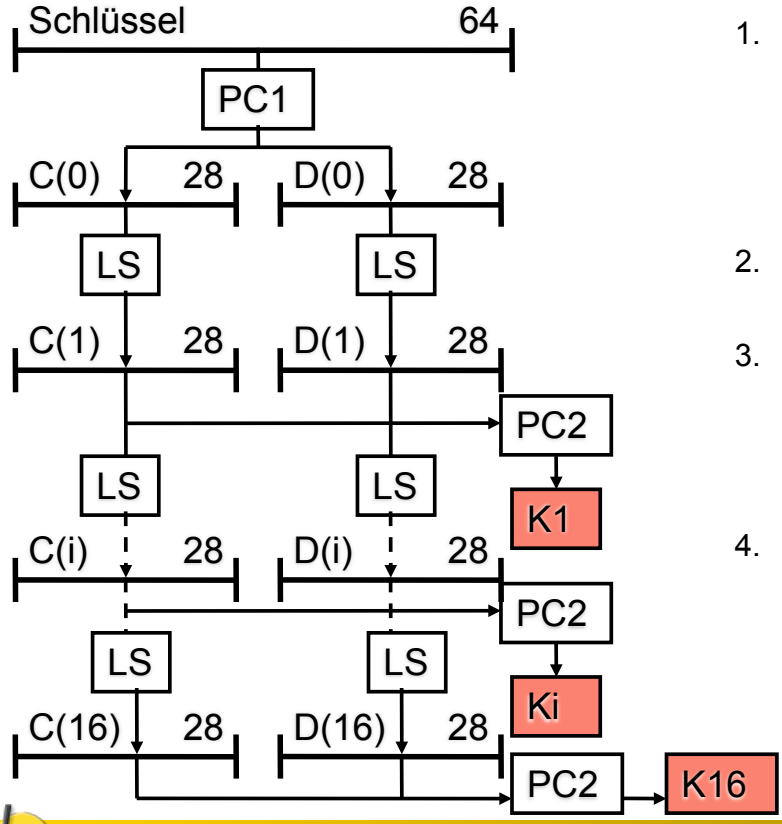
(i1,i6) \ (i2,i3,i4,i5)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- Beispiel
  - S-Box S1
  - Input (0,1,1,0,1,1)
  - Zeile (0,1) = 1
  - Spalte (1,1,0,1) = 13
  - Output (5) = (0,1,0,1)

DES Funktion: Grundschema

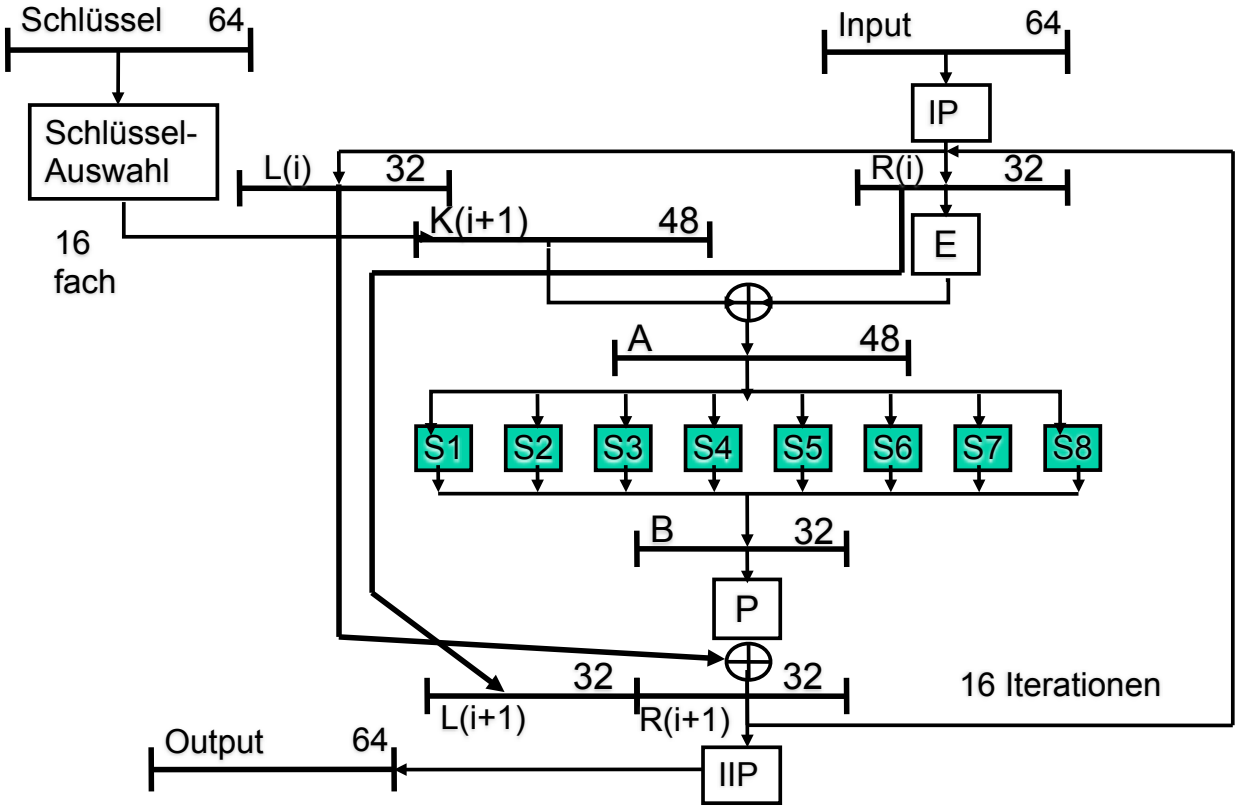


DES Schlüsselauswahl



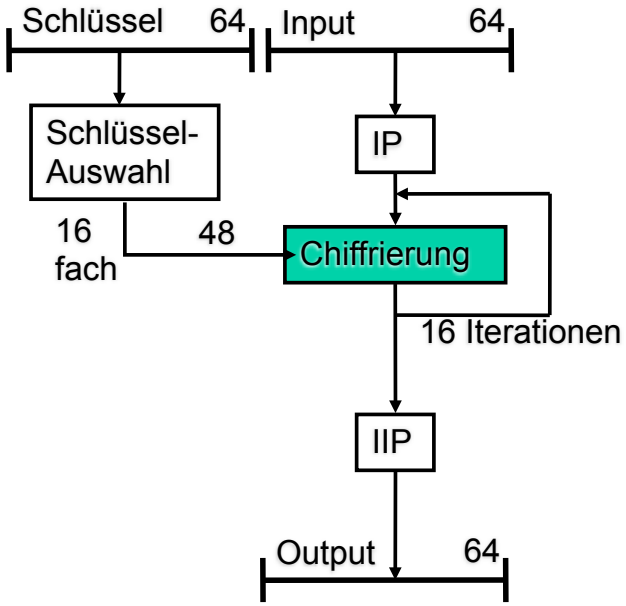
1. 64 Bit Schlüssel wird Permuted Choice 1 (PC1) unterworfen:
   - Key wird auf 56 relevante Bits gekürzt (jedes 8. Bit Parity)
   - Key wird permutiert
2. Schlüssel wird in zwei Teile a 28 Bit aufgeteilt
3. Blöcke werden zyklisch nach links geschiftet
   - In Runde 1,2,9 u. 16 um 1 Bit
   - 2 Bit sonst
4. Teilblöcke werden zusammen-gefasst und PC2 unterworfen:
   - Entfernen der Bits 9,18,22,25,35,38,43 u. 56
   - Permutation der verbleibenden 48 Bit

DES: Zusammenfassung



DES Entschlüsselung

- Es gilt D = E
- DES wird für Ver- und Entschlüsselung prinzipiell gleich verwendet, außer
- Umkehrung der Schlüsselreihenfolge
- In Runde i wird K(16-i) verwendet



DES Stärken und Schwächen

- Starker **Avalanche-Effekt** (Lawineneffekt; große Streuung) durch S-Boxen und Permutation P:
  Kleine Änderungen in der Eingabe, die nur eine S-Box betreffen breiten sich schnell aus.
  Eine Änderung eines Bits in der Eingabe verursacht eine Änderung von durchschnittlich 50% der Ausgabe
- 16 Iterationen:
  Known-plaintext Angriff auf DES mit < 16 Runden immer effizienter als Brute force
- Stark gegen analytische Angriffe
  Differentielle Kryptanalyse braucht 2^{47} gewählte Klartexte (chosen-pl.t.)

- (teilweise) geheimes Design
- Deutlich zu geringe Schlüssellänge
  Schlüsselraum der Größe 2^{56}
- 4 schwache Schlüssel mit:
  DES(DES(x,K),K) = x
- 6 semi-schwache Schlüsselpaare:
  DES(DES(x,K),K‘) = x
- Differentielle Kryptanalyse lässt sich in der Komplexität reduzieren auf 2^{37}

- Optimiert auf Implementierung in Hardware:
  Initialpermutation IP und inverse IP verbessern die Sicherheit nicht, sondern erhöhen nur den Aufwand für Software-Implementierungen

DES Varianten: Double und Tripple DES

- Double-DES:
  - DES(DES(m,K1),K2)
- Erwartete Komplexität:
  - bei Schlüssellänge n

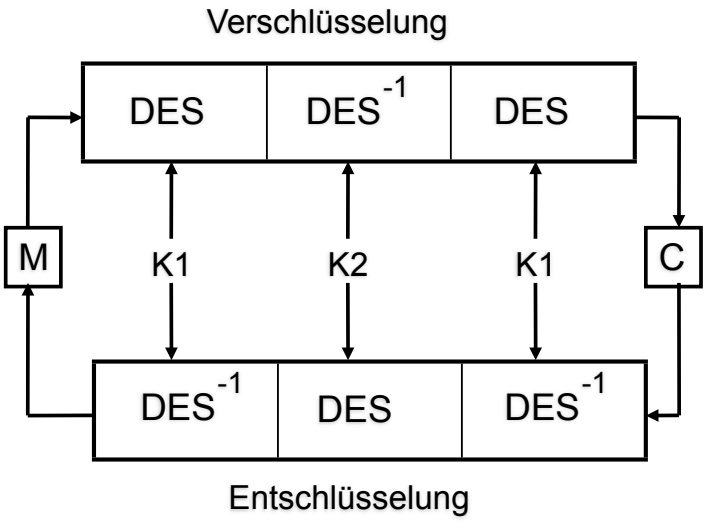
$$2^{2n}$$

- Merkle und Hellman haben gezeigt, dass ein Known-Plaintext Angriff möglich ist mit

$$2^{n+1}$$

  - Allerdings liegt der Speicherverbrauch bei 2^n Blöcken, bei DES 2^56 * 64 Bit, d.h. 10^17 Byte, d.h. im Bereich von Peta Byte
- D.h. doppelte Ausführung von DES bringt **KEINE** merkliche Steigerung der Sicherheit

- Triple-DES



- Komplexität

$$2^{2n}$$

Einschub: SSL/TLS Renegotiation Vulnerability

- Protokollschwäche im SSL/TLS Protokoll erlaubt Man-in-the-Middle Angriff
- Betroffene Software: Alle Systeme die auf SSL/TLS aufsetzen und Client-basierte Authentisierung per Zertifikat machen, z.B.: https, vpn-Software, smtps, ...
- Schwachstelle:
  - SSL/TLS erlaubt während der Sitzung verwendete Verfahren zu wechseln oder eine Re-Authentisierung zu verlangen (Renegotiation)
  - Protokollschwäche führt dazu, dass für Angreifer der Zugriff auf den Verkehr hat Man-in-the-Middle Angriff möglich wird
- Lösung:
  - derzeit keine
  - Workaround: Renegotiation deaktivern; Problem von Seiteneffekten
  - IETF arbeitet an einer Protokollaktualisierung
- weitere Infos: http://isc.sans.org/diary.html?storyid=7534

Vortragsreihe: Professioneller IT-Betrieb

- Vortrag 19.11.09; 14:15 - 16:00 Uhr; LRZ, Boltzmannstr. 1, Garching

- Abtrennung eines Teilkonzerns; Sascha Haupt (Fa. GeNUA)
  - Münchner Konzern verkauft Tochter an arabischen Investor
  - Netze müssen getrennt werden
  - Projekt: Aufbau einer Firewall Infrastruktur bei der Tochter
  - Migration bestehender FW-Regeln
  - Umzug von Diensten (ohne grössere Ausfallzeiten)
- Fa. GeNUA (Gesellschaft für Netzwerk- und Unix-Administration mbH)
  - IT Sicherheitsspezialist
  - Sitz in Kirchheim bei München
  - eigene Firewall-Produkte und zugehörige Dienstleistungen

Vortragsreihe: Professioneller IT-Betrieb

- 26.11.09:
  - Robert Koch (Uni der Bundeswehr):
    IDS Problemfelder und neue Ansätze;
  - Bernhard Schmidt (LRZ):
    IPv6 im Münchner Wissenschaftsnetz
- 03.12.09: Dr. Michael Brenner (LRZ):
  IT-Service-Management am LRZ: People, Process, Technology
- 10.12.09: Georg Mey (NetApp):
  Konzepte moderner Speichersysteme
- 07.01.10:
  - Dr. Ernst Boetsch (LRZ):
    Security-Bausteine für das Münchner Wissenschaftsnetz
  - Florian Gleixner (LRZ):
    Umfassende Serverüberwachung mit Nagios und SEC

Vortragsreihe: Professioneller IT-Betrieb

- 14.01.10: Christian Lotz (plan42 GmbH):
  ITIL V3 zwischen Anspruch und Realität
- 21.01.10: Marc Lindike (Flughafen München GmbH):
  ITIL-Erfahrungen und -Aktivitäten beim Münchner Flughafen
- 28.01.10:
  - Dr. Matthias Brehm (LRZ):
    Höchstleistungsrechner und ihre Anwendungen;
  - Dr. Bernd Reiner (LRZ):
    Das Archiv- und Backup-System des LRZ
- 04.02.10: Dr. Werner Degenhardt (LMU; Dept. Psychologie)
  Der menschliche Faktor der Informationssicherheit
- 11.02.10: Michael Storz (LRZ):
  Das Mailsystem des LRZ

Inhalt

- Symmetrische Kryptosysteme
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES); Historie

- 1997 öffentliche Ausschreibung des Dept. Of Commerce (Request for Candidate Algorithms for AES):
  - Algorithmus öffentlich und nicht klassifiziert,
  - Mindestblocklänge 128 Bit, Schlüssellänge 128, 192 und 256 Bit
  - Weltweit frei von Lizenzgebühren,
  - nutzbar für 30 Jahre, effizient sowohl in SW als auch versch. HW
- Dreistufiges (Vor-)Auswahlverfahren
  1. Pre-Round 1 (1/97 – 7/98)
     - Call for Candidates
  2. Round 1 (8/98 – 4/99)
     - Vorstellung, Analyse und Test
     - Auswahl der Kandidaten für Round 2
  3. Round 2 (8/99 – 5/2000)
     - Analyse und Tests
     - Auswahl der Finalisten
- Endgültige Auswahl durch NIST

AES Kandidaten

- Pre-Round 1: 21 Kandidaten, 6 aus formalen Gründen abgel.

Algo.	Land	Autor(en)	Algo.	Land	Autor(en)
CAST-256	Kanada	Entrust	MAGENTA	Deutschland	Deutsche Telekom
CRYPTON	Korea	Future Systems	MARS	USA	IBM
DEAL	Kanada	R. Outbridge, L. Knudsen	RC6	USA	RSA Laboratories
DFC	Frankreich	CNSR	RIJNDAEL	Belgien	J. Daeman, V. Rijmen
E2	Japan	NTT	SAFER+	USA	Cylink
FROG	Costa Rica	TecApro	SERPENT	UK, Norwegen, Israel	R. Anderson, E. Biham u.a.
HPC	USA	R.Schroeppel	TWOFISCH	USA	B. Schneier, J. Kelsey, u.a.
LOKI97	Australien	L. Brown, J. Pieprzyk u.a.			

AES: Round 2 Finalisten und Ergebnis

- Finalisten der Runde 2:

MARS	USA	IBM
RC6	USA	RSA Laboratories
RIJNDAEL	Belgien	J. Daeman, V. Rijmen
SERPENT	UK, Norwegen, Israel	R. Anderson, E. Biham, L. Knudsen
TWOFISCH	USA	B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Feruson

- 2. Oktober 2000: Rijndael wird gewählt
- 26. Nov. 2001: Veröffentlichung des FIPS-197 (Federal Information Processing Std.) durch NIST (National Institute for Standards and Technology)
- 26. Mai 2002: Inkrafttreten des Standards
- Informationen: `www.nist.gov/aes` mit Link auf AES-Homepage

AES

- Variable Blocklänge: 32*Nb Bits
- Variable Schlüssellänge: 32*Nk Bits
- Nb und Nk aus {4-8}; im Standard eingeschränkt auf 4,6 oder 8
- Variable Rundenanzahl Nr mit Nr = max(Nb, Nk) + 6
- Folgende Beispiele für Nb=Nk=4
- Rijndal arbeit auf sog. States:
  Input Bytes in0,in1,...,in15 werden in State kopiert:

input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

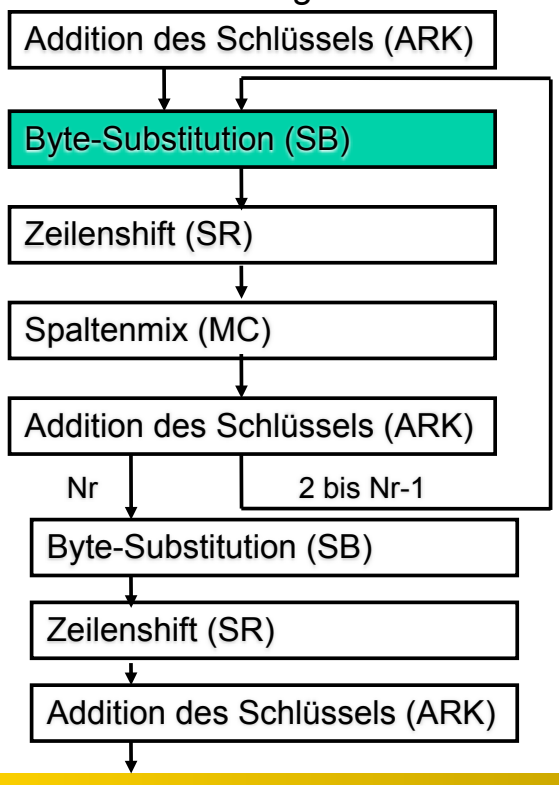
State array

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

- Runden arbeiten auf den States

AES: Ver- und Entschlüsselung

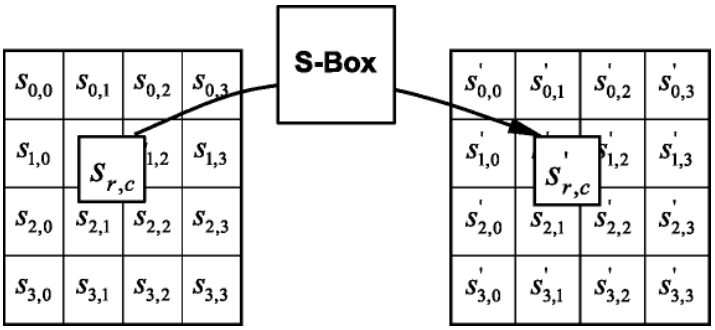
- Verschlüsselung



- Runden arbeiten auf sog. States
- Verschlüsselung
  - Ablauf der Runden 1 bis Nr-1:
    1. Byte-Substitution (`SubBytes`, SB)
    2. Zeilenshift (`ShiftRows`, SR)
    3. Spaltenmix (`MixColumns`, MC)
    4. Addition d. Rundenschlüssels (`AddRoundKey`, ARK)
- Entschlüsselung:
  - Runde 1 bis Nr-1:
    1. Addition d. Rundenschlüssels
    2. Inverser Spaltenmix
    3. Inverser Zeilenshift
    4. Inverse Byte-Substitution
- Letzte Runden Nr analog; aber **ohne** Spaltenmix

AES: Bytesubstitution (`SubBytes()`)

- Angewandt auf jedes Byte des State



- Byte S(r,c) wird als Kodierung eines Polynoms im Körper GF (2^8) aufgefasst
- Substitution:
  1. Bestimme multiplikatives Inverses von S(r,c) in GF(2^8)
  2. Affine Transformation des Inversen:
     - Matrixmultiplikation
     - XOR mit {63}

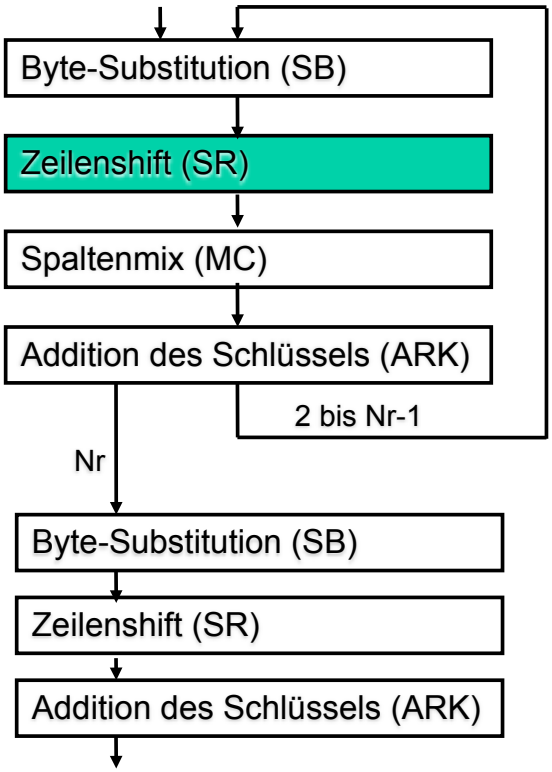
AES: Bytesubstitution Implementierung

- Vorausberechnung für alle 256 möglichen Polynome und Speicherung in S-Box (aus FIPS197)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

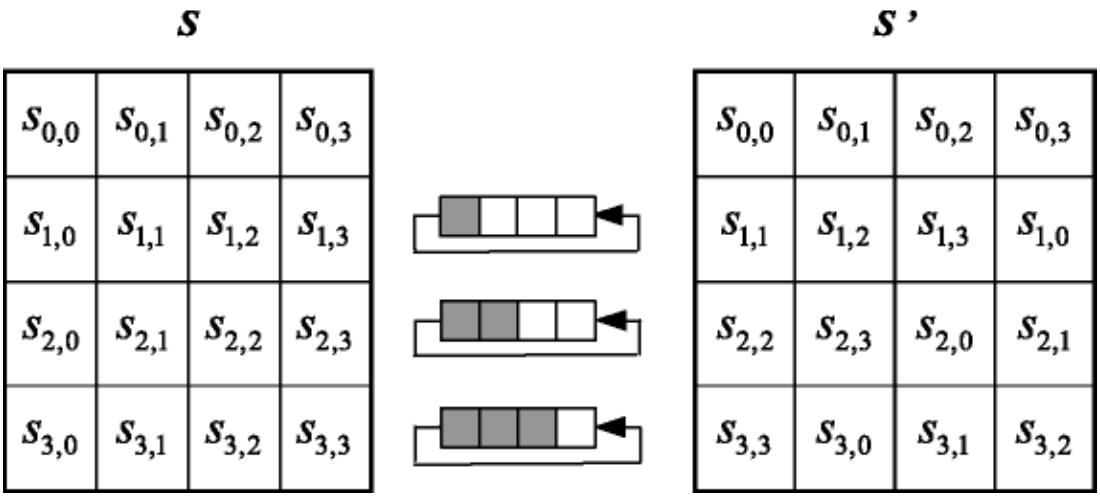
S-box: substitution values for the byte xy (in hexadecimal format).

AES: Verschlüsselung

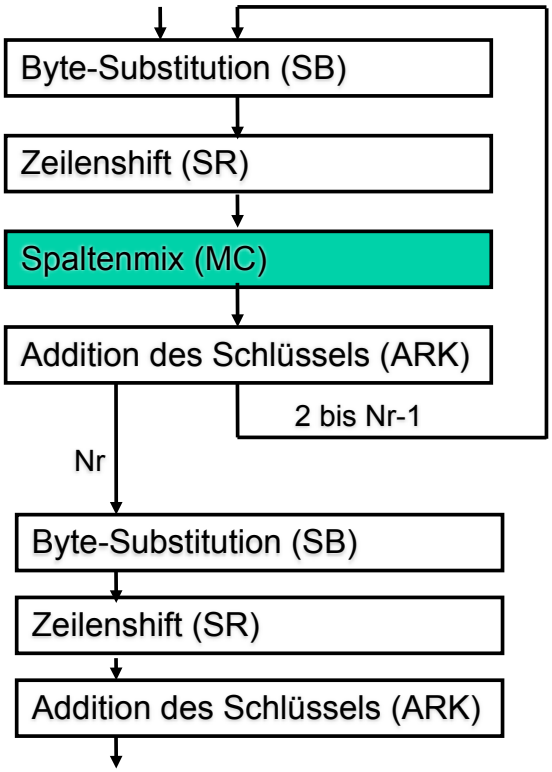


AES Zeilenshift (`ShiftRows()`)

- Zyklischer Shift der letzten drei Zeilen des State:Z
  - Zeile 1 bleibt unverändert
  - Zeile 2 um 1 Byte
  - Zeile 3 um 2 Byte
  - Zeile 4 um 3 Byte

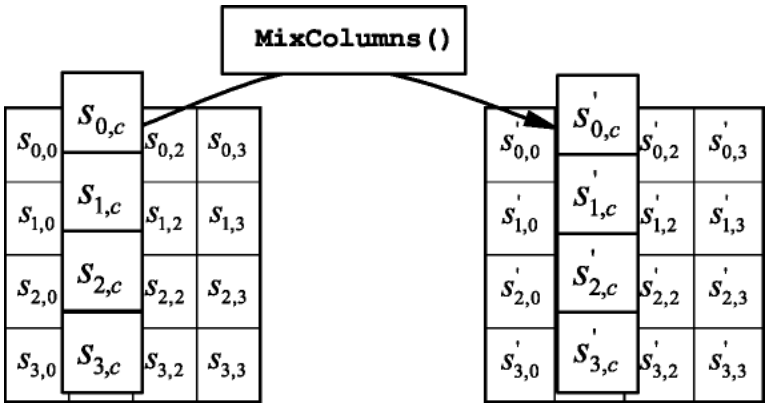


AES: Verschlüsselung



AES Spaltenmix (`MixColumns()`)

- Angewendet auf jede Spalte des State



- Jede Spalte wird als Polynom vom Grad 3 mit Koeffizienten aus GF(2^8) aufgefasst:
  - Multiplikation mit dem festen Polynom a(x) modulo x^4+1

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} .$$

AES Spaltenmix

- Darstellbar als Matrizenmultiplikation:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{für } 0 \le c < Nb.$$

Ausmultipliziert:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

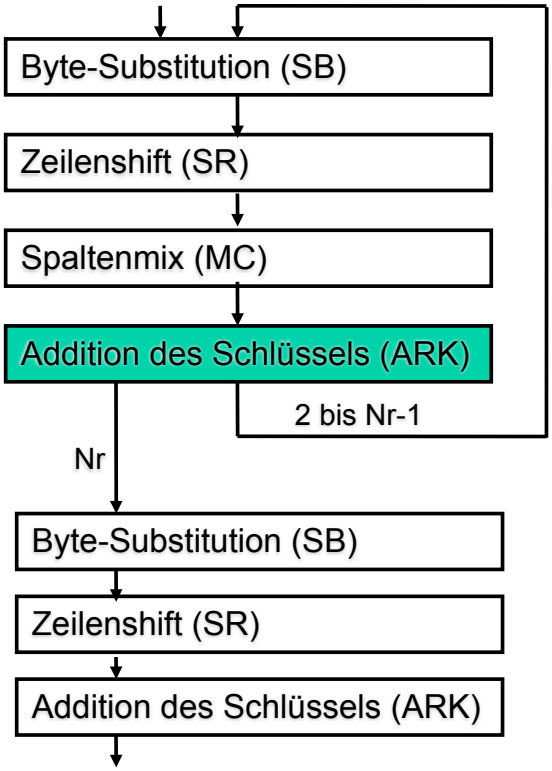
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

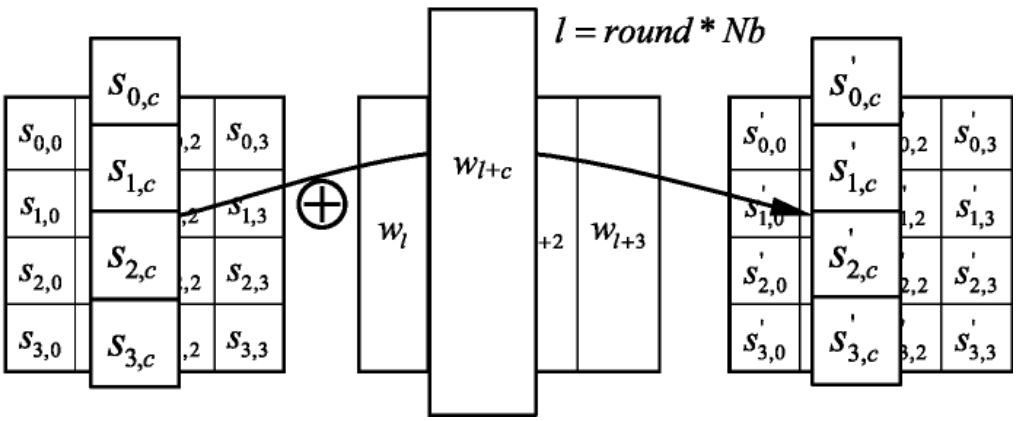
- $\bullet$ ist definiert in $GF(2^8)$ über irreduzibles Polynom $m(x) = x^8 + x^4 + x^3 + x + 1$

AES: Verschlüsselung



AES: Addition des Rundenschlüssels

- Funktion `AddRoundKey()`
- Jede Spalte des State wird mit einem „Wort“ des Rundenschlüssels XOR verknüpft

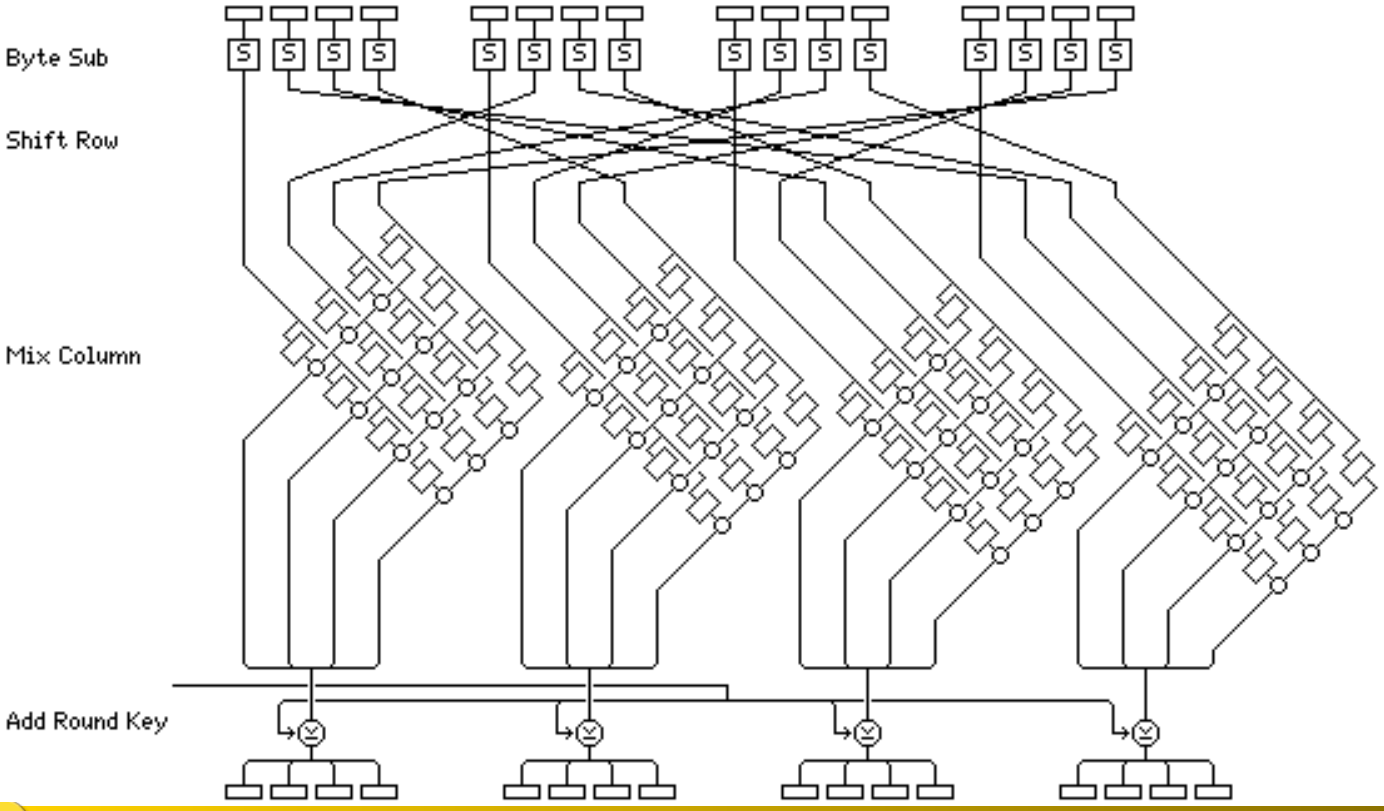


Schlüsselauswahl

- Schlüssel k besteht aus 32 * Nk Bits bzw. 4 * Nk Bytes
- Ein Wort W[i] besteht aus aus 4 Bytes
- W[0] sind die ersten 4 Byte des Schlüssels, W[1] die zweiten 4 Bytes, ..., W[Nk-1] die letzten 4 Bytes
- Insgesamt müssen Nb * (Nr + 1) Wörter berechnet werden
- Im folgenden sei Nk <= 6:
  für i = Nk bis Nb * (Nr +1) gilt:
  - W[i] = W[i-Nk] XOR `SubByte`(W[i-1])

  falls i mod Nk == 0 gilt
  - W[i] = W[i-Nk] XOR `SubByte`(`RotByte`(W[i-1]) XOR `Rcon`[i/Nk]
    mit `SubByte` = AES Byte-Substitution angew. auf Bytes eines Wortes
    mit `RotByte` = zyklischer Shift um 1 Byte (abcd wird bcda)
  - `Rcon[i]=(RC[i], 00, 00, 00)` (Rundenkonstante) mit
    RC[1] = {01}; RC[i] = {02} * RC[i-1]

AES Verschlüsselung

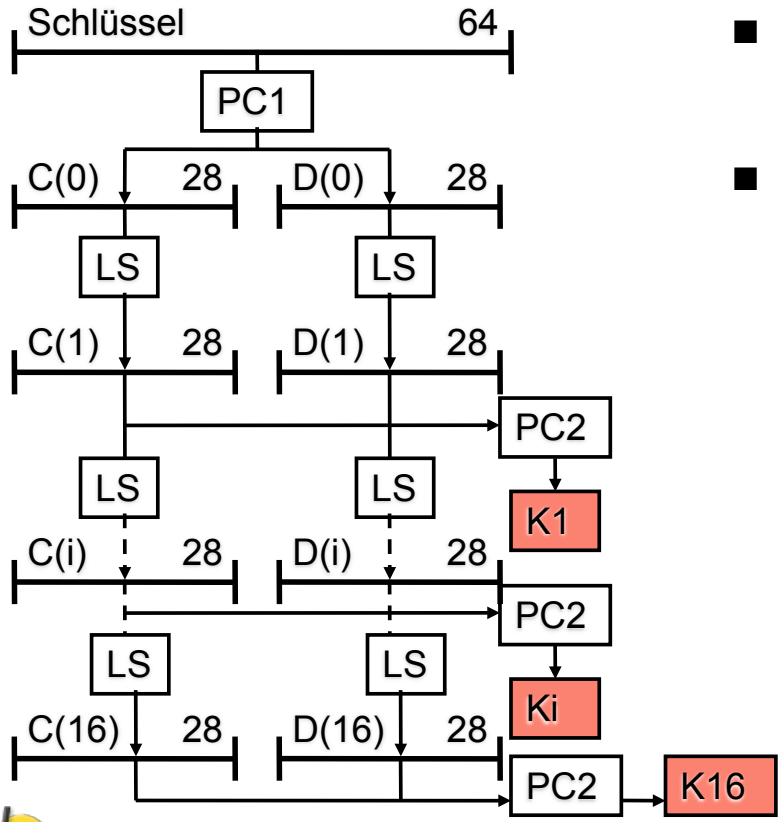


AES Design-Kriterien

- Design-Kriterien mussten offen gelegt werden
- Abschätzung und Stellungnahme zur Widerstandsfähigkeit gegen bekannte Angriffe

- Schlüsselauswahl mit nichtlinearer Durchmischung:
  wegen Verwendung der S-Box;
  damit widerstandsfähig gegen folgende Angriffe:
  - Kryptanalyst kennt Teile des Schlüssels und versucht den Rest zu berechnen
  - Zwei ähnliche Schlüssel haben **keine** große Zahl von gemeinsamen Rundenschlüsseln
  - Rundenkonstante verhindert Symmetrien im Verschlüsselungsprozess; jede Runde ist anders

Wdhlg.: DES Schlüsselauswahl



- Lediglich Permutation und Shift-Operationen

- Keine Substitution bzw. Anwendung von S-Boxen wie bei AES

AES Design-Kriterien (Forts.)

- Keine Feistel Chiffre, d.h. deutlich höhere Diffusion:
  nach 2 Runden hängt jedes Output Bit von jedem Input Bit ab
- Algebraische S-Box Konstruktion; Offengelegt; In hohem Maße nichtlinear
- Damit stabil gegen lineare und differentielle Kryptanalyse
- `ShiftRow` wurde eingefügt um zwei neue Angriffsarten zu verhindern (truncated differentials und Square attack)
- `MixColumn` für hohe Diffusion; Änderung in einem Input Byte verursacht Änderung in allen Output Bytes
- Auswahl von 10 Runden:
  Bei AES mit bis zu 7 Runden sind Angriffe bekannt die besser sind als Brute Force. Bei mehr als 7 Runden sind keine solchen Angriffe bekannt. D.h. 3 Runden „Reserve“ die sehr leicht erweitert werden können

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 6: Asymmetrische und Hybride Kryptosysteme

Inhalt

- Asymetrische Kryptosysteme
  - RSA
  - Sicherheit von RSA
- Schlüssellängen und Schlüsselsicherheit
- Hybride Kryptosysteme
- Digitale Signatur

Wdh.: Kryptographisches System (Def.)

- Geg. zwei endliche Zeichenvorräte (Alphabete) A_1 und A_2
- Ein Kryptosystem (KS) ist gegeben durch ein Tupel

$$KS = (M, C, EK, DK, E, D)$$

1. Nicht leere endliche Menge von Klartexten $M \subseteq A_1^*$
2. nicht leere endliche Menge von Krypto- bzw. Chiffretexten $C \subseteq A_2^*$
3. der nicht leeren Menge von Verschlüsselungsschlüsseln EK
4. der nicht leeren Menge von Entschlüsselungsschlüsseln DK sowie einer Bijektion $f : EK \rightarrow DK$ mit $f(K_E) = K_D$
5. Injektives Verschlüsselungsverfahren $E : M \times EK \rightarrow C$
6. Entschlüsselungsverfahren $D : C \times DK \rightarrow M$ mit $\forall m \in M : D(E(m, K_e), K_d)) = m$

Asymmetrisches Kryptosystem, Def.

- $KS = (M, C, EK, DK, E, D)$
- Schlüsselpaare müssen leicht (effizient) zu erzeugen sein, und es muss gelten:
  - $K_E = f(K_D), mit\ K_E \in EK,\ K_D \in DK$
  - $\forall m \in M: \quad D(E(m, K_E), K_D) = m$
  - K_E kann öffentlich bekannt gemacht werden
- Ver- und Entschlüsselungsfunktion E und D effizient zu berechnen
- K_D aus K_E nicht mit vertretbarem Aufwand berechenbar; Umkehrfunktion von f schwer zu berechnen; (Stichwort: **Einwegfunktionen**)
- Kryptosystem zur digitalen Signatur geeignet falls zus. gilt:
  - $\forall m \in M: \quad E(D(m, K_D), K_E) = D(E(m, K_E), K_D) = m$

RSA

- Benannt nach den Erfindern: Rivest, Shamir, Adleman (1978)
- Sicherheit basiert auf dem Faktorisierungsproblem:
  - Geg. zwei große Primzahlen p und q (z.B. 200 Dezimalstellen):
  - n=pq ist auch für große Zahlen einfach zu berechnen,
  - aber für gegebenes n ist dessen Primfaktorzerlegung sehr schwierig
- Erfüllt alle Anforderungen an asymmetrisches Kryptosystem
- In USA patentiert (Patent ist 2000 ausgelaufen)
- Große Verbreitung, verwendet in:
  - SSL (Secure Socket Layer)
  - PEM (Privacy Enhanced Mail)
  - PGP (Pretty Good Privacy)
  - GNUpg
  - ....

RSA Mathematische Grundlagen

- Restklassenarithmetik
- Restklassenring modulo m
- Multiplikatives Inverses
- Eulersche φ Funktion
- kleiner Fermatscher Satz
- Einweg-Funktion (One way function)
- Einweg-Funktion mit Falltür (Trapdoor one way function)

Vgl. Tafel

RSA Funktionsweise

1. Wähle große Primzahlen p und q und berechne $n = pq$
   n wird als RSA-Modul bezeichnet
2. Wähle $d \in [0, n-1]$ so dass d relativ prim ist zu $\varphi(n)$
   - D.h. $ggT(\varphi(n), d) = 1$
   - $\varphi(n) = (p-1)(q-1)$

   Wähle d so dass gilt: $\max(p,q) < d < \varphi(n) - 1$
3. Wähle $e \in [0, n-1]$ mit $e \times d \equiv 1 \bmod \varphi(n)$
   d.h. e ist das multiplikative Inverse modulo $\varphi(n)$ zu d
4. Öffentlicher Schlüssel (e, n)
5. Geheimer Schlüssel (d, n)
6. Verschlüsseln eines Klartextes $m \in [0, n-1]$

$$E(m) = m^e \bmod n = c$$

7. Entschlüsseln:

$$D(c) = c^d \bmod n$$

Erfüllt RSA Anforderungen an asym. Kryptosystem?

- Es sind zwei Sachverhalte zu zeigen:

1. Der Chiffrentext kann auch wieder entschlüsselt werden:

$$D(E(m)) = m$$

2. Das Verfahren eignet sich zur digitalen Signatur

$$D(E(m)) = E(D(m))$$

- Beweis vgl. Tafel

Einschub: Nomenklatur für kryptologische Verfahren

- Für Verschlüsselungsverfahren wird künftig die folgende Notation verwendet:

A_p	Öffentlicher (public) Schlüssel von A
A_s	Geheimer (secret) Schlüssel von A
A_p{m}	Verschlüsselung der Nachricht m mit dem öffentlichen Schlüssel von A
A_s{m} oder A{m}	Von A erstellte digitale Signatur von m
S[m]	Verschlüsselung von m mit dem **symmetrischen** Schlüssel S

RSA Sicherheit / mögliche Angriffe

1. **Brute force**: Testen aller möglichen Schlüssel
2. **Chosen-Cipertext Angriff**: (vgl. Tafel)
3. **Mathematische Angriffe**:
   - Faktorisierung von n;
   - direkte Bestimmung von $\varphi(n)$ ohne Faktorierung;
   - direkte Bestimmung von d ohne Bestimmung von $\varphi(n)$
4. **Timing Angriff:** Info: [Koch 96, Kali 96]
   - Überwachung der Laufzeit von Entschlüsselungsoperationen
   - Über Laufzeitunterschiede kann privater Schlüssel ermittelt werden
   - Analogie: Einbrecher ermittelt Kombination eines Tresors mit Hilfe der mitgehörten Stellzeiten am Zahlenschloß
5. **Angriffe auf Signaturen** (vgl. spätere Folien zur dig. Signatur)
   - Existentielle Fälschung
   - Multiplikativität von RSA

RSA: Chosen-Ciphertext Angriff

Vgl. Tafel

RSA Mathematische Angriffe

- Mathematische Angriffe lassen sich auf Faktorisierung zurückführen
- Schnellster bekannter Algorithmus General Number Field Sieve (GNFS), vgl. [Silv 01]
  - Laufzeitkomplexität: $L(N) = e^{(c+o(1))\bullet\sqrt[3]{\log(N)}\bullet\sqrt[3]{\log(\log(N))^2}}$
  - Speicherplatzkomplexität: $\sqrt{L(N)}$
- Faktorisierung wird einfacher falls:
  - Anzahl der Ziffern von p und q große Unterschiede aufweisen (z.B. |p| = 10 und |q| = 120)
  - Falls $d < 1/3 \bullet \sqrt[4]{n}$ kann d leicht berechnet werden
  - Die ersten m/4 Ziffern oder die letzten m/4 Ziffern von p oder q sind bekannt

Inhalt

- Asymetrische Kryptosysteme
  - RSA
  - Sicherheit von RSA
- Schlüssellängen und Schlüsselsicherheit
- Hybride Kryptosysteme
- Digitale Signatur

Wie lang muss ein sicherer Schlüssel sein? Einflussfaktoren

- Symmetrisches oder Asymmetrisches Verfahren ?
- Algorithmus
- PC / Software basierter Angriff
- Angriff mit integrierter Schaltung (ASIC, application specific integrated circuit)
- Angriff mit pogrammierbarer integrierten Schaltung (FPGA, field programmable gate array)
- GPGPU (General-purpose computing on graphics processing untis)
- Kosten und Ressourcenbedarf

Angriffe auf symmetrische Kryptosysteme

- Brute-Force Angriff
  - Durchsuchen des gesamten Schlüsselraums
  - Im Mittel ist halber Schlüsselraum zu durchsuchen
- Referenzzahlen; Größenordnungen (gerundet)

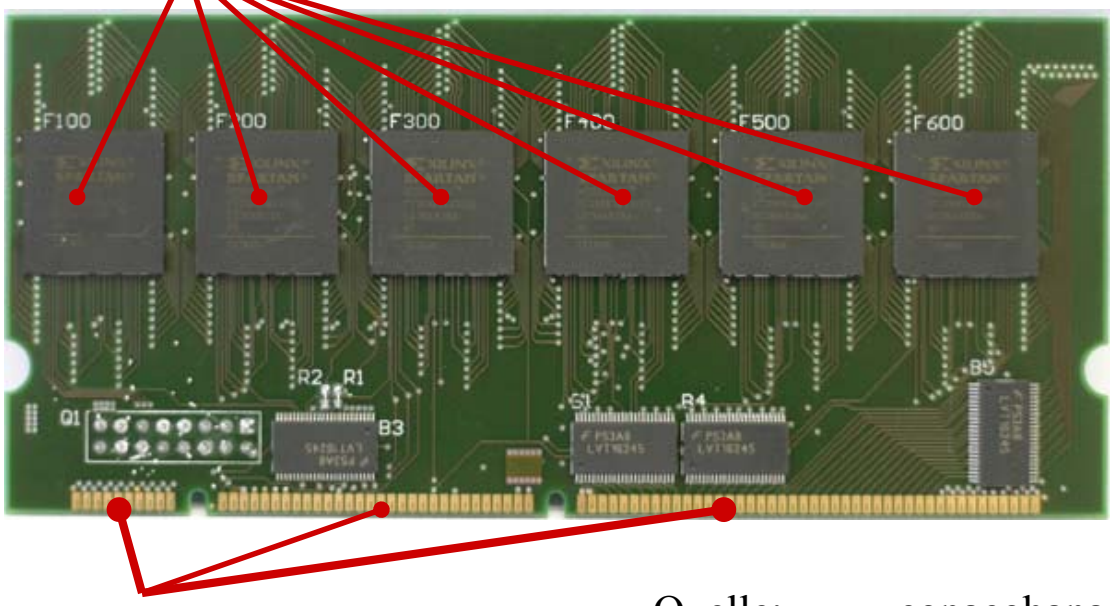
Tag: 86400
Jahr 31.536.000

	Größenordnu
Sekunden in einem Jahr	$3 * 10^7$
Alter des Universums in Sekunden	$4 * 10^{17}$
Schlüsselraum bei 64 Bit Schlüssellänge	$2 * 10^{19}$
Masse des Mondes [kg]	$7 * 10^{22}$
Masse der Erde [kg]	$6 * 10^{24}$
Masse der Sonne [kg]	$2 * 10^{30}$
Schlüsselraum bei 128 Bit Schlüssellänge	$3 * 10^{38}$
Anzahl Elektronen im Universum	10^{77} - 10^{79}

PC versus FPGA basierter Angriff

- FPGA Implementierung Copacobana (www.copacobana.org)

6x Spartan 3 FPGA (xc3s1000, FT256 packaging)



Connection to backplane (64-bit data bus)

Quelle: www.copacobana.org

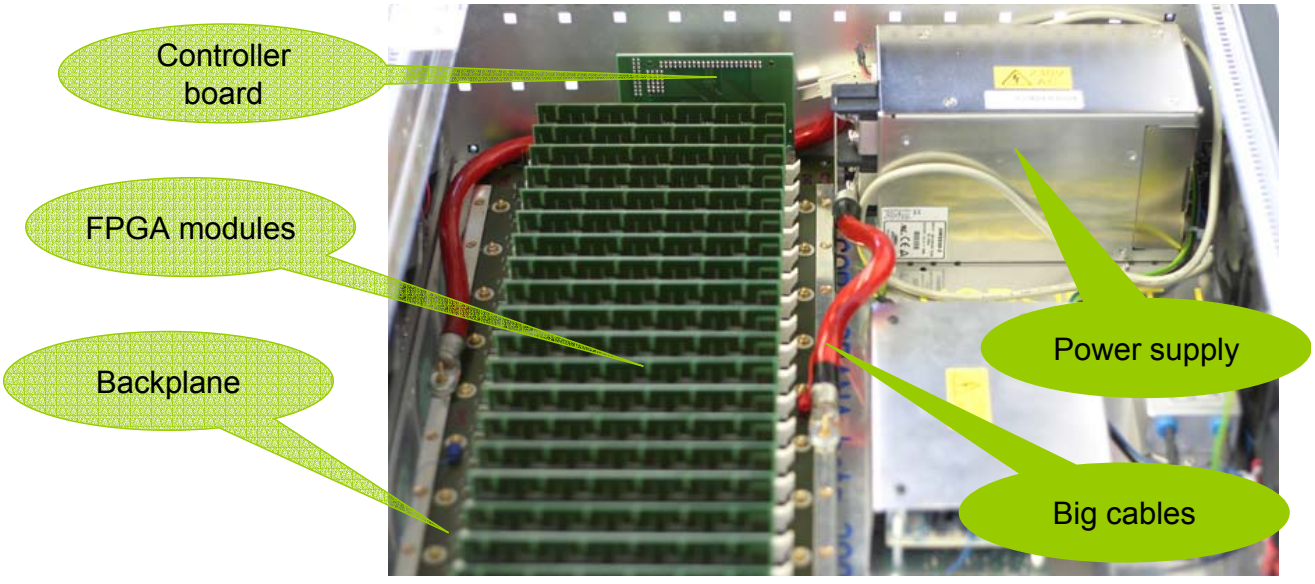
Copacobana



[Pelzl 2006]

Copacobana: Innenleben

- 20 Module pro Maschine mit 120 FPGAs



[Pelzl 2006]

DES: Brute Force Angriff

- Pentium 4; 3 GHz: ca. $2 * 10^6$ Schlüssel/s
- Copacabana;
  - 2006: $4{,}793 * 10^{10}$ Schlüssel/s
  - 2007: $6{,}415 * 10^{10}$ Schlüssel/s

Schlüssellänge [Bit]	#Schlüssel	durchschnittliche Zeit [s]		
		2006	2007	Pentium PC
40	$1{,}1 * 10^{12}$ ($5{,}5 * 10^{11}$)	11,5	8,6	274.878 3,18 d
56	$7{,}2 * 10^{16}$	751.680 8,7 d	561.600 6,5 d	$1{,}8 * 10^{10}$ 571 Jahre
128	$3{,}4 * 10^{38}$	$3{,}55 * 10^{27}$ $1{,}12 * 10^{20}$ J.	$2{,}65 * 10^{27}$ $8{,}4 * 10^{19}$ J	$8{,}5 * 10^{31}$ $2{,}6 * 10^{24}$ J.

DES Brute Force Angriff; Kosten

- Ziel: DES im Mittel in 8,7 Tagen brechen (Stand 2006)
- Dafür rd. 24.000 Pentium (a 150 €) erforderlich:
  3,6 Mio €
- Copacabana: **9.000 €**

AES: Brute Force

- Schlüssellänge 128 Bit
- FPGA Implementierung schaffen 22 Gb/s Durchsatz (2007)
- $22 * 2^{30} / 128 = 1{,}845 * 10^{8}$ Schlüssel/s

Schlüssellänge	Zeit [s]
128	$9{,}2 * 10^{29}$

RSA Schlüssellängen

- RSA Challange: Belohnung für das Brechen von RSA Schlüsseln, z.B. durch Faktorisierung (2007 eingestellt)

Dezimalstellen	Bits	Datum	Aufwand	Algorithmus
100	332	April 1991	7 Mips Jahre	Quadratisches Sieb
110	365	April 1992	75 Mips J.	
120	398	Juni 1993	830 Mips J.	
129	428	April 1994	5000 Mips J.	
130	431	April 1996	1000 Mips J.	General Number Field Sieve (GNFS)
140	465	Februar 1999	2000 Mips J.	
155	512	August 1999	8000 Mips J.	
160	530	April 2003	k.A.	GNFS(Lattice Sieve)
174	576	Dez. 2003	k.A.	GNFS(Lattice/Line Sieve)
193	640	Nov. 2005	30 2,2 GHz Opteron J.*	GNFS

* Halb so lange wie für RSA-200 Challange

RSA Schlüssellängen

- RSA Challange: Belohnung für das brechen von RSA Schlüsseln, z.B. durch Faktorisierung (2007 eingestellt)

Dezimalstellen	Bits	Datum	Aufwand	Algorithmus
100	332	April 1991	7 Mips Jahre	Quadratisches Sieb
110	365	April 1992	75 Mips J.	
120	398	Juni 1993	830 Mips J.	
129	428	April 1994	5000 Mips J.	
130	431	April 1996	1000 Mips J.	
140	465	Feb. 1999	2000 Mips J.	
155	512	August 1999	8000 Mips J.	
160	530	April 2003	k.A.	GNFS(Lattice Sieve)
174	576	Dez. 2003	k.A.	GNFS(Lattice/Line Sieve)
193	640	Nov. 2005	30 2,2 GHz Opteron J.*	GNFS

d.h. 512 Bit Schlüssel ist nicht mehr sicher!

* Halb so lange wie für RSA-200 Challange

Schlüsselsicherheit symmetrisch vs. asymmetrisch

- Verschiedene Institutionen geben Vergleiche heraus Bits of Security (äquiv. Schlüssellänge symmetrischer Verfahren)
  - NIST (National Institute of Standards and Technology) 2007:

Bits of Security	80	112	128	192	256
Modullänge (pq)	1024	2048	3072	7680	15360

  - NESSIE (New European Schemes for Signatures, Integrity and Encryption) (2003)

Bits of Security	56	64	80	112	128	160
Modullänge (pq)	512	768	1536	4096	6000	10000

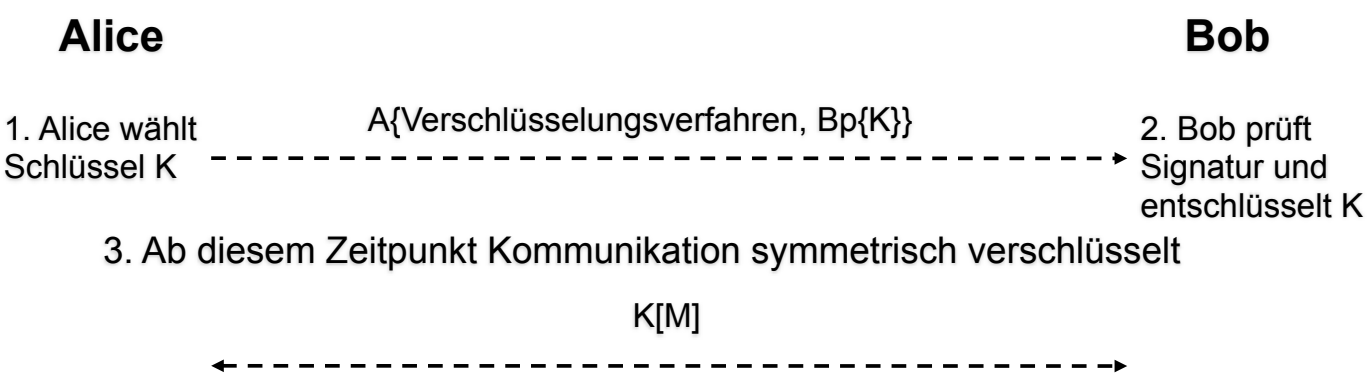
Empfohlene Schlüssellängen

- Quelle: ECRYPT 2006: Report on Algorithms and Keysizes
- Minimale Schlüssellängen:

Angreifer	Budget	Hardware	min. Anzahl Bits of Security
„Hacker“	0	PC	52
	< $ 400	PC(s)/FPGA	57
	0	„Malware“	60
Small organization	$ 10k	PC(s)/FPGA	62
Medium organization	$ 300k	FPGA/ASIC	67
Large organization	$ 10M	FPGA/ASIC	77
Intelligence agency	$ 300M	ASIC	88

Hybride Kryptosysteme

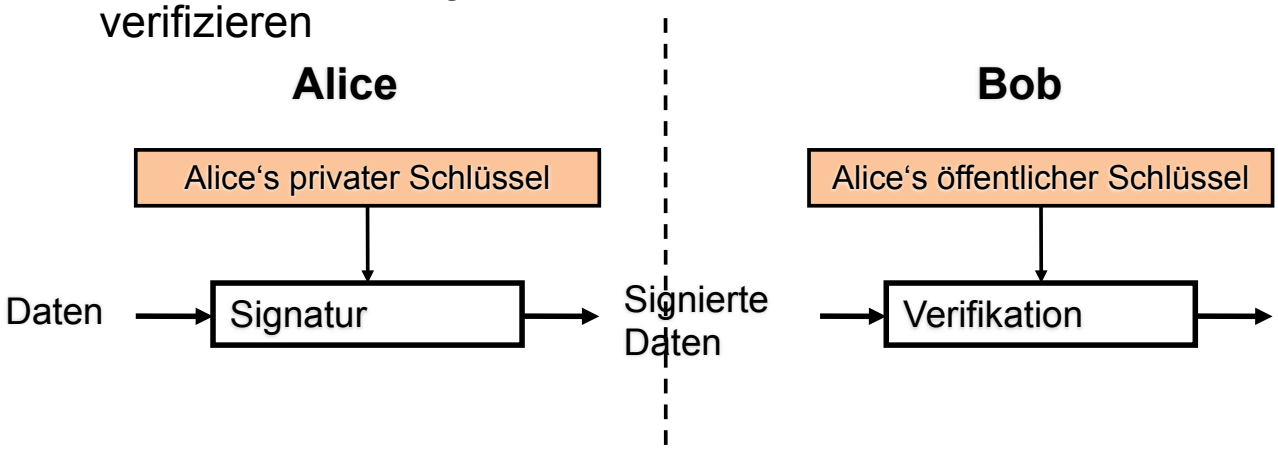
- Vereinen Vorteile von Symmetrischen und asymmetrischen Verfahren
- Asymmetrisches Verfahren zum Schlüsselaustausch
- Symmetrisches Verfahren zur Kommunikationsverschlüsselung



- Beispiele für hybride Verfahren: PGP, ssh,...

Digitale Signatur

- Alice „signiert“ Daten mit ihrem privaten Schlüssel
- Jeder kann die Signatur mit Alice‘ öffentlichem Schlüssel verifizieren



- Assymetrische Verfahren sind im Vergleich sehr langsam
- Daher i.d.R. nicht Signatur der gesamten Daten
- Lediglich kryptographischer Hash-Wert der Daten wird signiert (digitaler Fingerabdruck der Daten)

Digitale Signatur: Analogie zur Unterschrift

- Anforderungen an die (analoge) Unterschrift:

1. **Perpetuierungsfunktion:** Unterschrift kann nicht gefälscht werden und ist dauerhaft
2. **Echtheitsfunktion:** Die Unterschrift ist authentisch
3. Die Unterschrift kann **nicht wiederverwendet** werden.
4. **Abschlußfunktion:** Unterschrift kann später nicht verändert werden
5. **Beweisfunktion:** Unterzeichner kann seine Unterschrift später nicht leugnen

- Bei der Unterschrift auf Papier ist keine dieser Anforderungen vollständig erfüllt!
- Trotzdem wird die Unterschrift im Rechtsverkehr akzeptiert
- Ihre Funktion wird durch Rahmenbedingungen gesichert

Digitale Signatur: Erfüllung der Anforderungen?

1. **Perpetuierungsfunktion:** Fälschungssicher und dauerhaft
2. **Echtheitsfunktion:** authentisch
3. **Nicht wiederverwendbar**
4. **Abschlußfunktion:** nicht veränderbar
5. **Beweisfunktion:** Unterschrift nicht leugnen

- Lassen sich diese Funktionen bei der digitalen Signatur realisieren?

1. Solange privater Schlüssel geheim gehalten wird
2. Abhängig von zweifelsfreier Zuordnung des Schlüsselpaares zu einer Identität (Zertifizierung, CA)
3. Digitale Signatur „beinhaltet“ den Dateninhalt
4. vgl. 3.
5. Jeder kann Signatur bzw. Echtheit mit öffentlichem Schlüssel des Unterzeichners verifizieren

RSA Signaturangriffe

- Existentielle Fälschung:
  - Mallet wählt beliebige Zahl r mit 0 <= r < n
  - M behauptet r sei ein von Alice signiertes Dokument (z.B. abzuhebender Geldbetrag)
  - Empfänger verifiziert und falls m=A_p{r} sinnvollen Text ergibt, war der Angriff erfolgreich
- Multiplikativität von RSA
  - Mallet wählt zwei Nachrichten m1 und m2 und lässt Alice signieren
  - Mallet berechnet A{m1}*A{m2} mod n
  - Es gilt $A\{m1\} \bullet A\{m2\} \bmod n = (m1 \bullet m2)^{As} \bmod n$
  - D.h. M kann aus zwei signierten Dokumenten ein drittes, gültig signiertes Dokument erzeugen, ohne im Besitz des Schlüssels von Alice zu sein
- Gegenmaßnahme:
  Nicht das gesamte Dokument, sondern nur Hash-Wert signieren!

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 7: Kryptographische Hash Funktionen

Inhalt

- Def.: Kryptographische Hash-Verfahren
- Angriffe gegen One-Way Hash Funktionen
- Konstruktion von Hash-Funktionen
- Algorithmen:
  - MD4
  - MD5
  - Whirlpool

Kryptographische Hash-Funktionen: Grundlagen

- Hash-Funktionen bilden „Universum“ auf endlichen Bildbereich ab
- Hash-Funktion h sind **nicht** injektiv
- Bildbereich i.d.R. sehr viel kleiner als Universum
- Kollisionen möglich: $\exists x, y \in U : x \neq y \quad \wedge \quad h(x) = h(y)$

- Kryptographische Hash-Funktionen H:
  - Eingabe: beliebig langes Wort m aus dem Universum U
  - Ausgabe: Hash-Wert H(m) mit fester Länge
  - H soll möglichst kollisionsresistent sein

Einsatz kryptographischer Hash-Funktionen

- Integritätssicherung („Digitaler Fingerabdruck“):
  1. Alice erzeugt Nachricht m, berechnet H(m)=h und überträgt (m,h) an Bob (mindestens h muss gesichert werden, z.B. durch Verschlüsselung)
  2. Bob empfängt (m‘,h) und berechnet h‘=H(m‘)
  3. Falls h=h‘ kann davon ausgegangen werden, dass m=m‘, d.h. m wurde **nicht** verändert

- Digitale Signatur:
  - In der Praxis wird nicht die Nachricht m digital signiert
  - Stattdessen wird H(m) digital signiert {H(m)}
  - Übertragen wird dann (m,{H(m)})
  - Empfänger kann Quelle der Nachricht zweifelsfrei feststellen
  - Empfänger kann Integrität der Nachricht belegen

Def. Kryptographische Hashfunktion

- Schwache Hash-Funktion H:
  1. H besitzt die Eigenschaften einer Einwegfunktion
  2. Hash-Wert H(m) = h mit |h|=k ist bei gegebenem m einfach zu berechnen
  3. Bei gegebenem h = H(m) für $m \in A_1^*$ ist es praktisch unmöglich ein m‘ zu finden mit:

     $$m' \neq m,\ m' \in A_1^* \ \wedge\ H(m') = h$$

- Starke Hash-Funktion H:
  1. H ist eine schwache Hash-Funktion
  2. Es ist praktisch unmöglich eine Kollision zu finden, d.h. ein Paar verschiedene Eingabewerte m und m‘ mit:

     $$m' \neq m,\ m, m' \in A_1^* \ \wedge\ H(m) = H(m')$$

Birthday Attack auf One-Way Hash Funktionen

- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit p > 0,5 eine weitere Person mit Ihnen Geburtstag hat?
  - Antwort: 253
- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit p > 0,5 zwei Personen am selben Tag Geburtstag haben
  - Antwort: 23
- Wie können Sie dieses Wissen für Angriffe gegen Hash-Funktionen nutzen?
- Eine Kollision zu finden ist deutlich einfacher als zu einem gegebenen Hash-Wert einen passenden Text.

Birthday Attack: Vorgehensweise

1. Alice sichert mit einem m-Bit langen Hash eine Nachricht M.
2. Mallet erzeugt 2^(m/2) Variationen der Nachricht M

- Die Wahrscheinlichkeit für eine Kollision ist größer 0,5.

- Wie können 2^(m/2) Variationen erzeugt werden?
  - Z.B. Einfügen von „Space – Backspace – Space“ Zeichen zwischen Wörtern
  - Wörter durch „Synonyme“ ersetzen
  - .....

Beispiel für einen Brief mit 2^37 Variationen

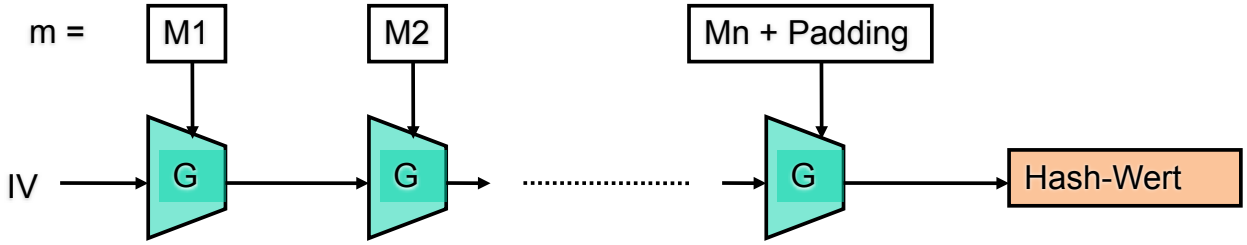
- [Stal 98]

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}
Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}
Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}
responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}
in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}
to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the
market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the
{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}
of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}
as proof of identity. An order with his signature, which is {appended / attached}
{authorizes / allows} you to charge the cost to this company at the {above / head office}
address. We {fully / --} expect that our {level / volume} of orders will increase in
the {following / next} year and {trust / hope} that the new appointment will {be / prove}
{advantageous / an advantage} to both our companies.

Konstruktion kryptographischer Hash Funktionen

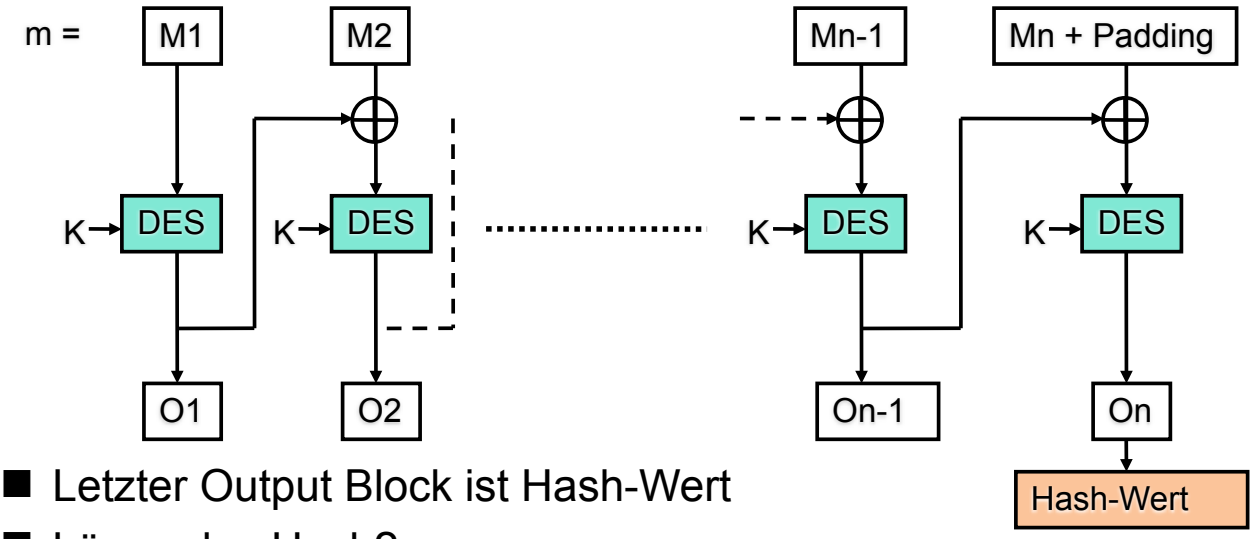
- Folge von Kompressionsfunktionen G
- Nachricht m wird in Blöcke Mi mit fester Länge zerlegt
- Hash Verfahren wird mit Initialisierungswert IV vorbelegt



- Letzter Block Mn muss ggf. auf vorgegebene Länge „aufgefüllt“ werden (Padding)
- Als Kompressionsfunktion G können verwendet werden:
  - Hashfunktionen auf der Basis symmetrischer Blockchiffren
  - Dedizierte Hash-Funktionen

DES als Kompressionsfunktion

- DES im Cipher Block Chaining (CBC) Mode



- Letzter Output Block ist Hash-Wert
- Länge des Hash?

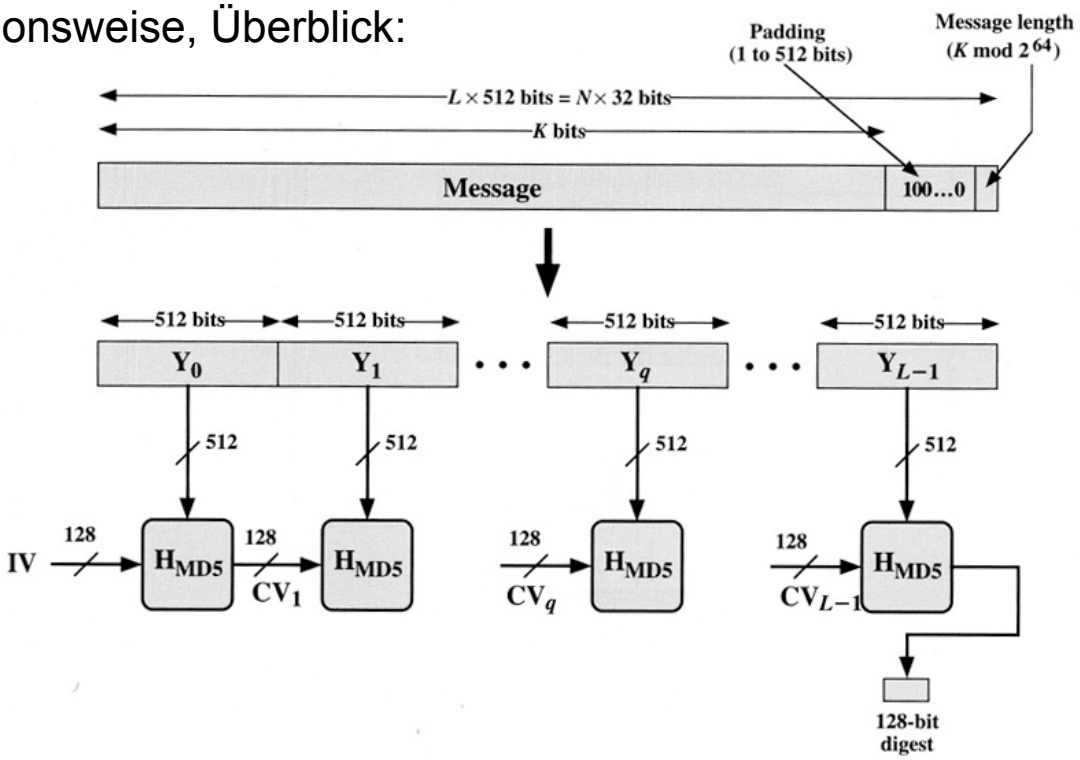
64 Bit

Hash-Funktionen: MD4

- Entwickelt von Ron Rivest: MD4 = Message Digest Nr. 4
- Design-Kriterien:
  - Kollisionsresistenz: Es gibt kein besseres Verfahren als Brute Force um zwei Nachrichten mit demselben MD4 Hash zu finden
  - Direkte Sicherheit: MD4 basiert auf keinerlei (Sicherheits-)Annahmen wie z.B. dem Faktorisierungsproblem
  - Geschwindigkeitsoptimiert für Software Implementierungen
  - Bevorzugt Little Endian 32 Bit Architekturen (Intel)
  - Einfach und Kompakt
- Erfolgreiche Angriffe
  - Boer und Bosselaers brechen die beiden letzten Runden der insges. drei
  - Merkle greift erfolgreich die ersten beiden Runden an
  - Angriff auf alle 3 Runden gelingt nicht
- Trotzdem: Rivest verbessert MD4; Ergebnis MD5

Hash-Funktionen: MD5

- Länge 128 Bit, arbeitet auf 512 Bit Blöcken
- Funktionsweise, Überblick:

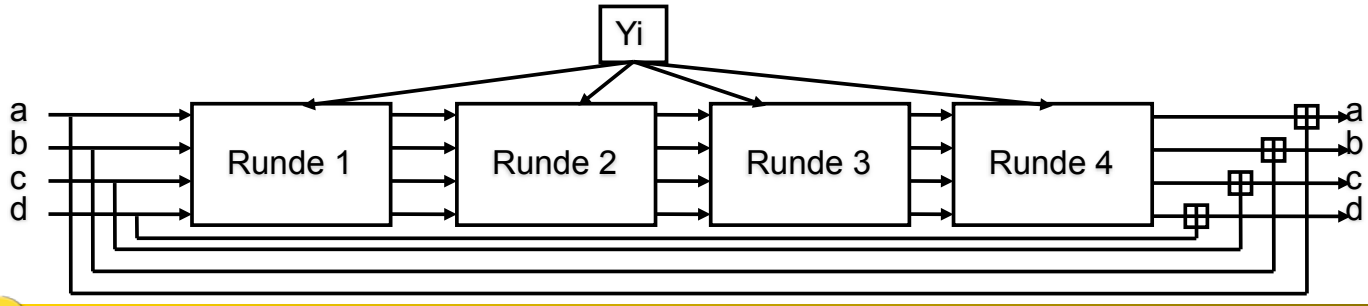


MD5 Ablauf

1. Padding Bits der Nachricht hinzufügen
2. Länge der Originalnachricht (mod 2^64) anfügen
3. Nachricht in 512 Bit Blöcke aufteilen
4. Initialisierung von 32 Bit-Variablen:

   A = 0x01234567 C = 0xFEDCBA98

   B = 0x89ABCDEF D = 0x76543210
5. Zuweisung a=A, b=B, c=C, d=D
6. Kompressionsfunktion H_{MD5} angewendet auf jeden (Teil-)Block



MD5 Kompressionsfunktion (1)

- 4 Runden mit je einer nichtlinearen Funktion

$$F(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$$

  - Funktionen so gewählt, dass korrespondierende Bits von X, Y, Z und dem Ergebnis unabhängig voneinander sind
- In jeder Runde wird die Funktion 16 mal auf einen 32 Bit Teilblock Mj von Yi wie folgt angewendet

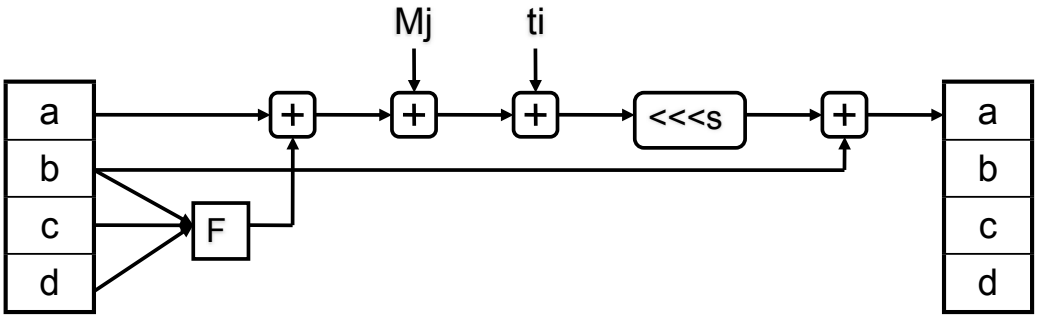
$$FF(a,b,c,d,M_j,s,t_i): \quad a = b + ((a + F(b,c,d) + M_j + t_i) <<< s)$$
$$GG(a,b,c,d,M_j,s,t_i): \quad a = b + ((a + G(b,c,d) + M_j + t_i) <<< s)$$
$$HH(a,b,c,d,M_j,s,t_i): \quad a = b + ((a + H(b,c,d) + M_j + t_i) <<< s)$$
$$II(a,b,c,d,M_j,s,t_i): \quad a = b + ((a + I(b,c,d) + M_j + t_i) <<< s)$$

MD5 Kompressionsfunktion (2)

- $FF(a,b,c,d,M_j,s,t_i): \quad a = b + ((a + F(b,c,d) + M_j + t_i) <<< s)$
  - + bezeichnet Addition modulo 2^32
  - $t_i = 2^{32} abs(\sin(i))$ mit i Grad im Bogenmaß; 0 <= i < 64 (i über 4 Runden)
  - <<< s bezeichnet zirkulären Shift um s Bits
  - Auswahl des Teilblocks Mj

Runde 1	Natürliche Ordnung	Runde 3	$(5+3i) \bmod 16$
Runde 2	$(1+5i) \bmod 16$	Runde 4	$7i \bmod 16$

- Beispiel: Elementarer Schritt in Runde 1



MD5: Rundenfunktion; 4 Runden mit 64 Schritten

- Runde 1:
  1. FF(a, b, c, d, M0, 7, 0xd76aa478)
  2. FF(d, a, b, c, M1, 12, 0xe8c7b756)
  3. FF(c, d, a, b, M2, 17, 0x242070db)
  4. FF(b, c, d, a, M3, 22, 0xc1bdceee)

- Runde 4:

  60. II(a, b, c, d, M4, 6, 0xf7537e82)
  61. II(d, a, b, c, M11, 10, 0xbd3af235)
  62. II(c, d, b, a, M2, 15, 0x2ad7d2bb)
  63. II(b, c, d, a, M9, 21, 0xeb86d391)

Sicherheit von MD5

- Differentielle Kryptanalyse auf MD5 mit nur einer Runde [Bers 92]:
  - Für jede der 4 Runden einzeln möglich
  - Angriff auf alle 4 Runden konnte nicht gezeigt werden
- Pseudokollision [BoBo 93]:
  - Zwei verschiedene Variablenbelegungen von a,b,c,d führen für verschiedene Inputblöcke zum gleichen Outputblock
  - Im Moment scheint eine Erweiterung des Ansatzes zu einem allgemeinen Angriff nicht möglich
- Erzeugung einer Kollision in der Kompressionsfunktion [Dobb 96]:
  - Zwei 512 Bit Blöcke produzieren den selben 128 Bit Output
  - Bis dahin gefährlichster bekannter Angriff
  - Bisher kein Mechanismus zur Generalisierung des Angriffs auf gesamten MD5 mit IV gefunden

Sicherheit von MD5 (Forts.)

- Kollision gefunden [Wang,Feng,Lai,Yu 2004]:
  - $MD5(M,N_i) = MD5(M',N_i')$
  - M und M' zu finden dauert ca. eine Stunde (IBM P690)
  - danach N_i und N_i' zu finden 15 Sek. bis 5 Minuten
  - funktioniert mit beliebigen Intialisierungsvektor IV
- In der Arbeit werden auch Kollisionen für MD4, HAVAL-128 und RIPEMD-128 angegeben
- Kollision in X.509 Zertifikat gefunden (Kollsion in den Schlüsseln) [de Weger 2005]
- Kollision in X.509 Zertifikat mit unterschiedlichen Identitäten [Stevens, Lenstra, de Wegener 2006/2007]

➡ MD5 (und SHA-1) nicht mehr verwenden!

➡ Algorithmen mit längeren Hash-Werten verwenden: z.B. SHA-224, SHA-256, SHA-384, SHA-512. Whirlpool, o.ä.

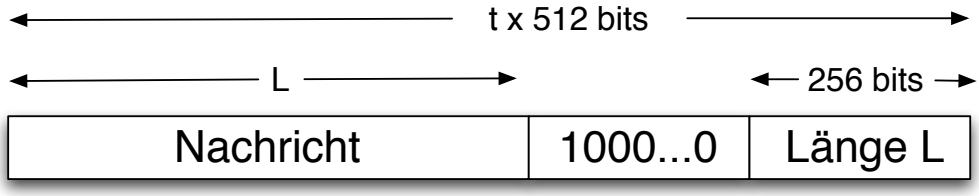
Whirlpool Hashing Funktion

- Entwickelt von P. Barreto und V. Rijmen
- im Rahmen des europäischen NESSI (New European Schemes for Signatures, Integrity, and Encryption) entwickelt
- Struktur sehr ähnlich zu AES, bzw. Rijndael
- 512 Bit lange Hashes bei max. Nachrichtenlänge v. 2^{256} Bits
- Design-Ziele:
  - Kollision zu finden benötigt $2^{n/2}$ Whirlpool Operationen
  - Zu gegebenen Hash h eine Nachricht x zu finden mit h = H(x) benötigt 2^n Whirlpool Operationen
  - Zu gegebenen Hash h und geg. Nachricht m eine Nachricht x zu finden benötigt 2^n

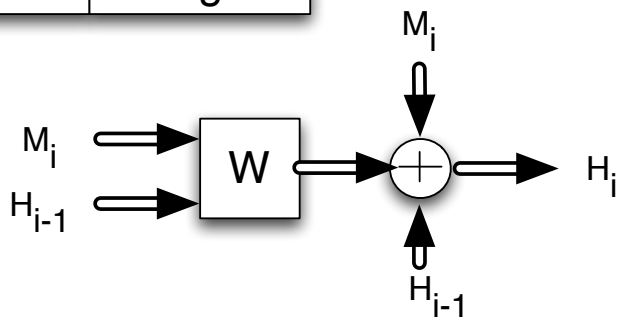
Whirlpool: Überblick

- Whirlpool (WP(m)) arbeitet auf 512 Bit langen Teilblöcken M_i
- Verwendet Block Chiffre W
- Arbeitet intern mit 8x8 Byte Matrix CState

1. Expansion von m auf ein Vielfaches von 512 Bit; Aufteilen in Nachrichtenblöcke M_0 bis M_{t-1}

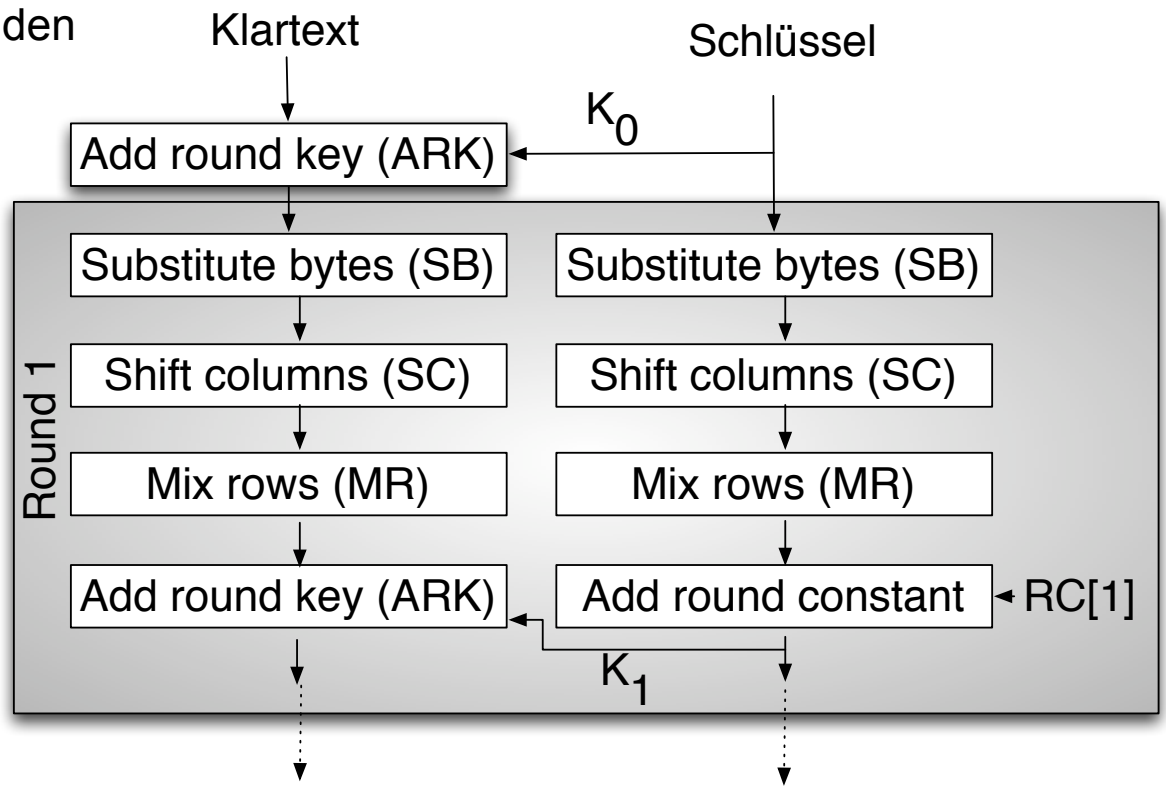


2. Initialisierung von $H_o = 0$
3. For $0 \le i \le t-1$
   $H_i = W_{Hi-1}(M_i) \oplus M_i \oplus H_{i-1}$
4. $WP(m) = H_t$

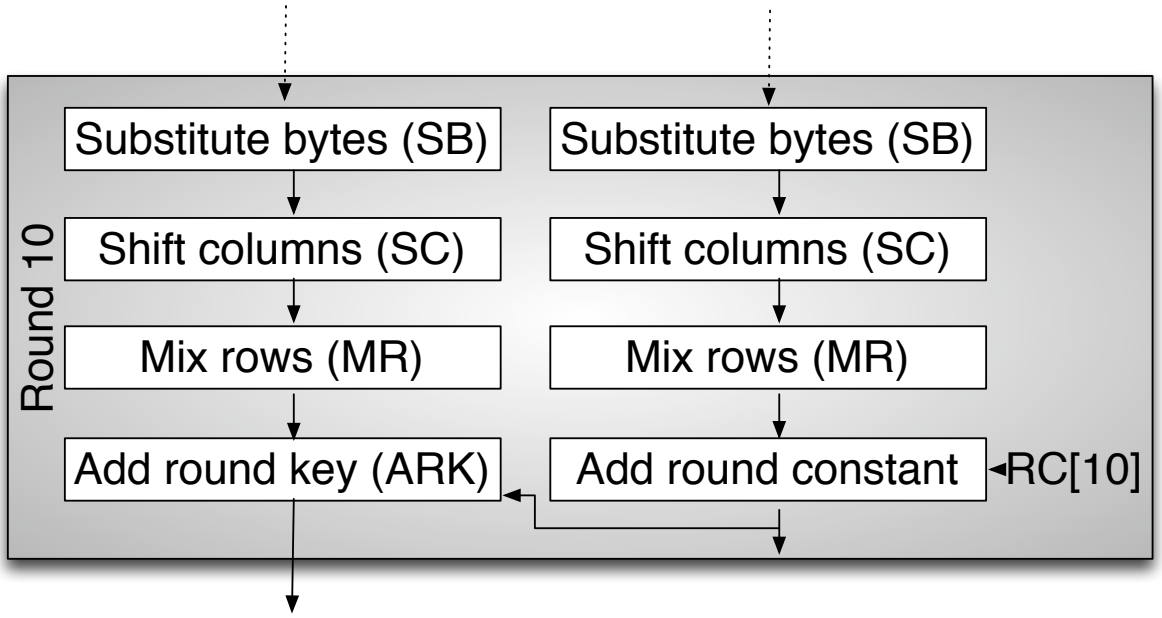


Whirlpool: Block Chiffre W

- 10 Runden



Whirlpool: Block Chiffre W; Fortsetzung

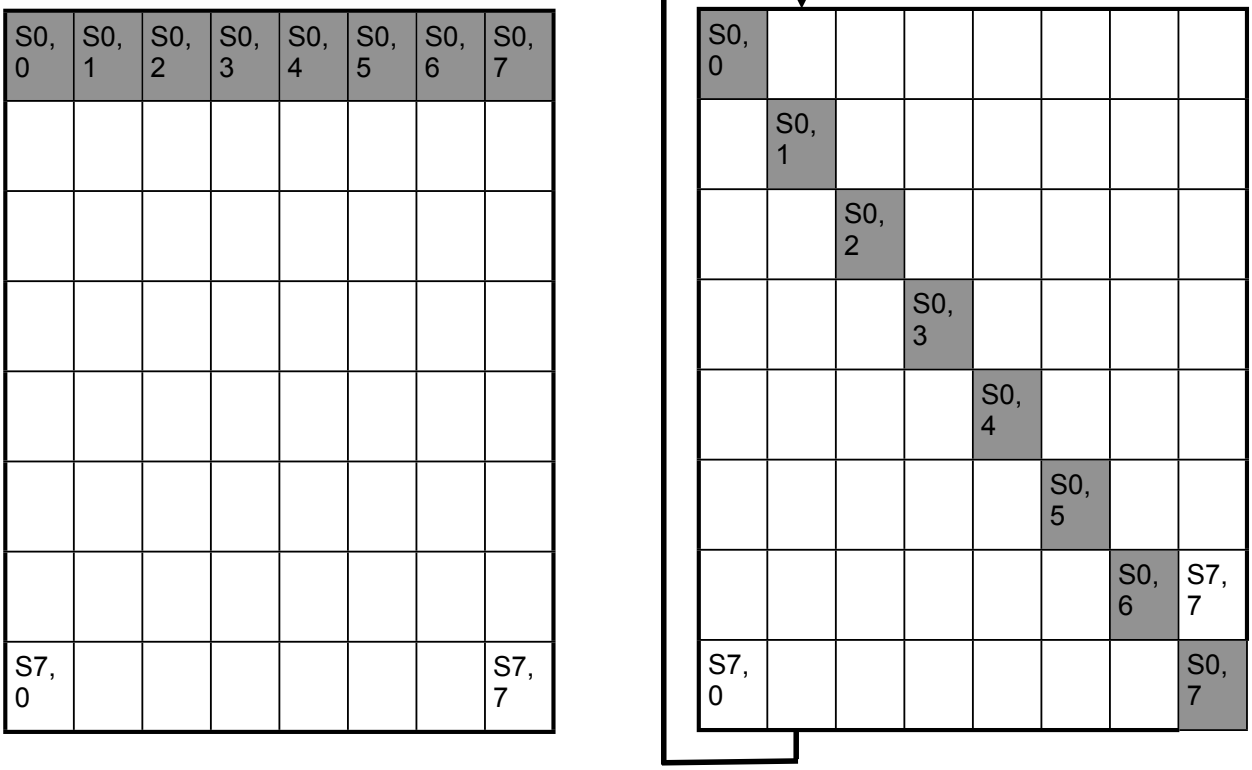


Whirlpool: Byte Substitution

- Mittels S-Box: auf Matrix-Elementen von CState
- 4 linken Bit bestimmen Spalte; 4 rechten Bit die Zeile

	00_x	01_x	02_x	03_x	04_x	05_x	06_x	07_x	08_x	09_x	$0A_x$	$0B_x$	$0c_x$	$0d_x$	$0E_x$	$0F_x$
00_x	18_x	23_x	$c6_x$	$E8_x$	87_x	$B8_x$	01_x	$4F_x$	36_x	$A6_x$	$d2_x$	$F5_x$	79_x	$6F_x$	91_x	52_x
10_x	60_x	Bc_x	$9B_x$	$8E_x$	$A3_x$	$0c_x$	$7B_x$	35_x	$1d_x$	$E0_x$	$d7_x$	$c2_x$	$2E_x$	$4B_x$	FE_x	57_x
20_x	15_x	77_x	37_x	$E5_x$	$9F_x$	$F0_x$	$4A_x$	dA_x	58_x	$c9_x$	29_x	$0A_x$	$B1_x$	$A0_x$	$6B_x$	85_x
30_x	Bd_x	$5d_x$	10_x	$F4_x$	cB_x	$3E_x$	05_x	67_x	$E4_x$	27_x	41_x	$8B_x$	$A7_x$	$7d_x$	95_x	$d8_x$
40_x	FB_x	EE_x	$7c_x$	66_x	dd_x	17_x	47_x	$9E_x$	cA_x	$2d_x$	BF_x	07_x	Ad_x	$5A_x$	83_x	33_x
50_x	63_x	02_x	AA_x	71_x	$c8_x$	19_x	49_x	$d9_x$	$F2_x$	$E3_x$	$5B_x$	88_x	$9A_x$	26_x	32_x	$B0_x$
60_x	$E9_x$	$0F_x$	$d5_x$	80_x	BE_x	cd_x	34_x	48_x	FF_x	$7A_x$	90_x	$5F_x$	20_x	68_x	$1A_x$	AE_x
70_x	$B4_x$	54_x	93_x	22_x	64_x	$F1_x$	73_x	12_x	40_x	08_x	$c3_x$	Ec_x	dB_x	$A1_x$	$8d_x$	$3d_x$
80_x	97_x	00_x	cF_x	$2B_x$	76_x	82_x	$d6_x$	$1B_x$	$B5_x$	AF_x	$6A_x$	50_x	45_x	$F3_x$	30_x	EF_x
90_x	$3F_x$	55_x	$A2_x$	EA_x	65_x	BA_x	$2F_x$	$c0_x$	dE_x	$1c_x$	Fd_x	$4d_x$	92_x	75_x	06_x	$8A_x$
$A0_x$	$B2_x$	$E6_x$	$0E_x$	$1F_x$	62_x	$d4_x$	$A8_x$	96_x	$F9_x$	$c5_x$	25_x	59_x	84_x	72_x	39_x	$4c_x$
$B0_x$	$5E_x$	78_x	38_x	$8c_x$	$d1_x$	$A5_x$	$E2_x$	61_x	$B3_x$	21_x	$9c_x$	$1E_x$	43_x	$c7_x$	Fc_x	04_x
$c0_x$	51_x	99_x	$6d_x$	$0d_x$	FA_x	dF_x	$7E_x$	24_x	$3B_x$	AB_x	cE_x	11_x	$8F_x$	$4E_x$	$B7_x$	EB_x
$d0_x$	$3c_x$	81_x	94_x	$F7_x$	$B9_x$	13_x	$2c_x$	$d3_x$	$E7_x$	$6E_x$	$c4_x$	03_x	56_x	44_x	$7F_x$	$A9_x$
$E0_x$	$2A_x$	BB_x	$c1_x$	53_x	dc_x	$0B_x$	$9d_x$	$6c_x$	31_x	74_x	$F6_x$	46_x	Ac_x	89_x	14_x	$E1_x$
$F0_x$	16_x	$3A_x$	69_x	09_x	70_x	$B6_x$	$d0_x$	Ed_x	cc_x	42_x	98_x	$A4_x$	28_x	$5c_x$	$F8_x$	86_x

Whirlpool: Shift Column



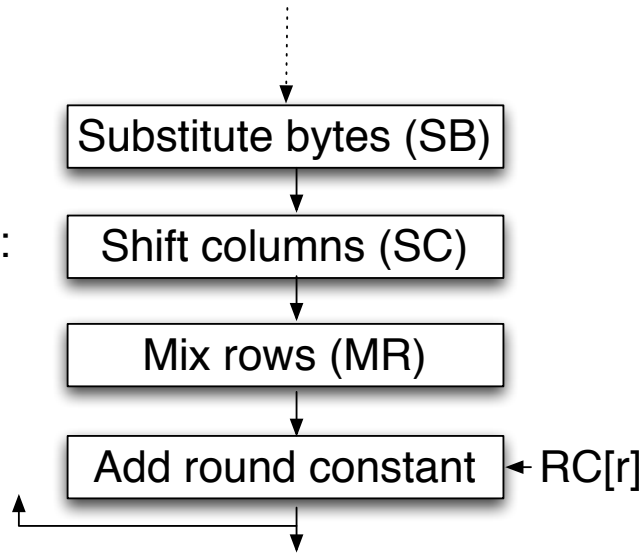
Whirlpool: Mix Row

- Matrixmultiplikation für jede Zeile von CState

$$\begin{bmatrix} s'_{i,0} \\ s'_{i,1} \\ s'_{i,2} \\ s'_{i,3} \\ s'_{i,4} \\ s'_{i,5} \\ s'_{i,6} \\ s'_{i,7} \end{bmatrix}^T = \begin{bmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \\ s_{i,4} \\ s_{i,5} \\ s_{i,6} \\ s_{i,7} \end{bmatrix}^T \cdot A = \begin{bmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \\ s_{i,4} \\ s_{i,5} \\ s_{i,6} \\ s_{i,7} \end{bmatrix}^T \cdot \begin{bmatrix} 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x \\ 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x \\ 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x \\ 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x \\ 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x \\ 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x \\ 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x \\ 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x \end{bmatrix}$$

Whirlpool: Add round key / constant

- CState wird mit K_i XOR verknüpft
- Berechnung von K_i



- Berechnung von RC[r] für Runde r:
  RC[0] = K
  for 1 ≤ r ≤ 10
    $RC[r]_{0,j}$ = Sbox[8(r-1)+j]
      für
      0≤j≤7 u. 1≤r≤10
    $RC[r]_{i,j}$ = 0 für
      0≤j≤7; 0≤j≤7 u.1≤r≤10

Vergleich Rijndal und W

	Rijndal	W
Blockgröße [bits]	128, 160, 192, 224 oder 256	immer 512
Rundenanzahl	10, 11, 12, 13 oder 14	immer 10
Schlüsselauswahl	ausgezeichneter Algorithmus (a priori)	Rundenfunktion von W
Reduktionspolynom	$x^8+x^4+x^3+x+1$ (0x11B)	$x^8+x^4+x^3+x^2+1$ (0x11d)

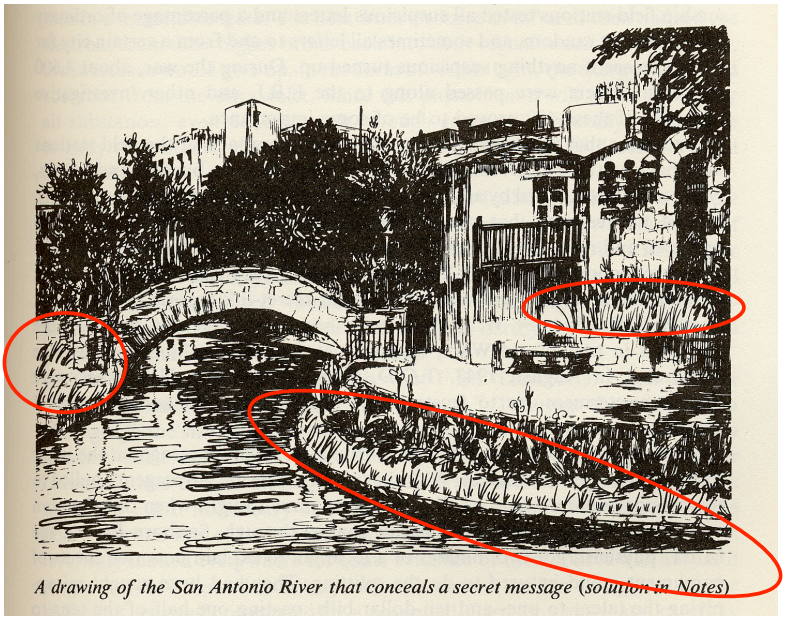
Sicherheits von Whirlpool

- Algorithmus wurde im Rahmen des NESSIE Projekts evaluiert
- Versteckte Schwächen wurden nicht gefunden
- Statistische Analyse: Whirlpool sollte stochastisch sein
- 512 Bit Länge bietet nur SHA-512
- Resistent gegenüber bekannten Angriffen
- Bisher kein erfolgreicher Angriff

- ABER: Algorithmus noch sehr jung (2003)
- bisher noch nicht so weit verbreitet

Linguistische Steganographie

- Bsp. aus David Kahn: *The Codebreakers,* Scribner, 1996



A drawing of the San Antonio River that conceals a secret message (solution in Notes)

- Nachricht als Morsezeichen kodiert.
- Kurze und lange Grashalme

- Nachricht lautet:

  Compliments of CPSA MA to our chief Col Harold R. Shaw on his visit to San Antonio May 11th 1945

IT-Sicherheit

- Sicherheit vernetzter Systeme -

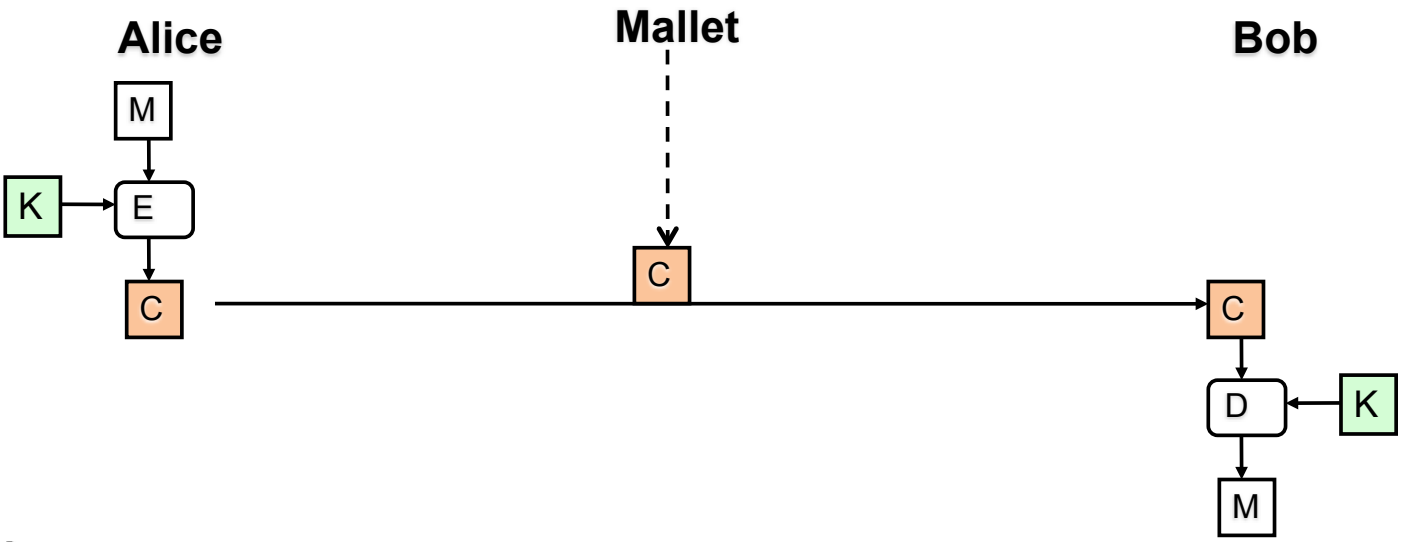
Kapitel 8: Sicherheitsmechanismen

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Vertraulichkeit (Confidentiality)

- Schutz der Daten vor unberechtigter Offenlegung
- Wie kann Vertraulichkeit realisiert werden?
  - Durch Verschlüsselung (Encryption)
  - Mallet kann Chiffrentext **nicht** nutzen



Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Integrität

- Erkennung von Modifikationen, Einfügungen, Löschungen, Umordnung, Duplikaten oder Wiedereinspielung von Daten
- Wie kann Integrität realisiert werden?
  - Modifikation, Einfügung, Löschung, Umordnung?
  - Kryptographischer Hash-Wert über die Daten
  - Duplikate, Wiedereinspielung von Daten?
  - Kryptographischer Hash-Wert + „gesicherte“ Sequenznummern und/oder Zeitstempel
- Verschlüsselung ein Mechanismus zur Integritätssicherung?
  - In Allgemeinheit: **NEIN, „**Blinde“ Modifikation des Chiffrentextes möglich
  - Abhängig vom Verschlüsselungsverfahren und den Daten kann es passieren, dass die Veränderung **nicht** automatisch erkannt wird
  - Auch mit semantischem Wissen kann Veränderung unbemerkt bleiben
  - Unwahrscheinliches aber mögliches Bsp.: Angreifer kippt Bit in verschlüsselter Überweisung; Entschlüsselung liefert 1000 statt 10 €

Angriff auf Mechanismen zur Integritätssicherung

- Angreifer verändert unbemerkt Daten **und** Hash-Wert
- Deshalb: Hash-Wert und ggf. Sequenznummern müssen vor Veränderungen geschützt werden
  - Sequenznummern oder Timestamp als Teil der geschützten Daten werden (automatisch) durch Hash geschützt
  - Sequenznummern im Protokoll-Header sind gesondert (durch Hash) zu schützen
  - Hash selbst wird z.B. durch Verschlüsselung geschützt
    - In diesem (Spezial-)Fall ist Verschlüsselung eine Möglichkeit zur Integritätssicherung
    - Bei verschlüsselten Hashes lassen sich „blinde“ Veränderungen am Chiffrentext automatisch erkennen
    - Übertragen wird < m, E(H(m)) >
    - Test beim Empfänger: Ist D(E(H(m))) gleich dem selbst berechneten Wert von H(m)

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Authentisierung: Arten

- Authentisierung wird unterschieden in:
  1. Authentisierung des Datenursprungs
  2. Benutzerauthentisierung
  3. Peer Entity Authentisierung
  - Weitere Unterteilung von 2. und 3.
    - Einseitig oder
    - Zwei- bzw. mehrseitige Authentisierung

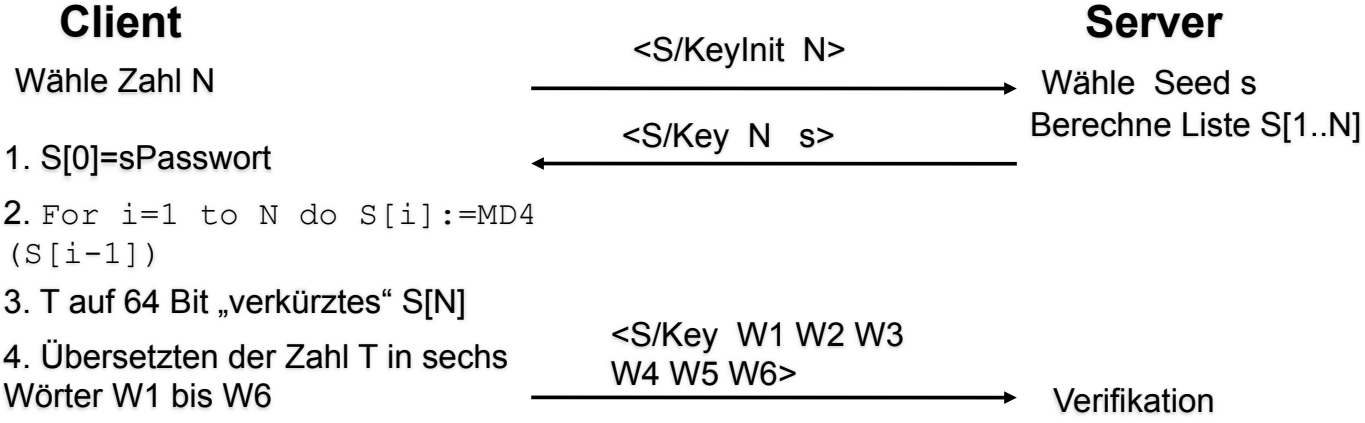
- Grundsätzliche Möglichkeiten zur Authentisierung:
  1. Wissen (Something you know)
  2. Besitz (Something you have)
  3. Persönliche Eigenschaft (Something you are)
  4. Kombinationen aus 1. – 3.

Benutzerauthentisierung

- Wissen
  - Passwort, Passphrase (Unix Passwort Verfahren, vgl. Kap. 3)
  - Einmal-Passwort
  - PIN
  - .....
- Besitz
  - Smart Card, Token, („physikalischer“) Schlüssel
  - Kryptographischer Schlüssel
- Eigenschaft
  - Biometrie:
    - Fingerabdruck
    - Stimmerkennung
    - Gesichtserkennung
    - Iris-Scan
    - Hand-Geometrie; Venenbild der Hand
    - Behavioral Biometrics, z.B.
      - Anschlags- oder Andruck-Charakteristik beim Schreiben
      - Lippenbewegungen

Einmal-Passwort Verfahren: S/Key

- Authentisierungsserver kennt Passwort des Benutzers



- Bei nächster Authentisierung wird S[N-1] verwendet, dann S [N-2], usw.
- Entwickelt von Bellcore [RFC 1760]

S/Key

- Verkürzungsfunktion
  - T := S[N] (128 Bit lang)
    T[0-31] := T[0-31] XOR T[64-95]
    T[32-63] := T[32-63] XOR T[96-127]
  - Weiter verwendet wird T[0-63]
- Eingabe einer 64 Bit Zahl ist fehleranfällig, daher
- Übersetzungsfunktion für T
  - Ergebnis 6 kurze (1 bis 4 Zeichen lange) englische Wörter
  - Wörterbuch mit 2048 Wörtern
  - Je 11 Bit von T liefern - als Zahl interpretiert - die Adresse des Wortes
  - Bsp. für einen solchen „Satz“: FORT HARD BIKE HIT SWING

OTP (One Time Password System)

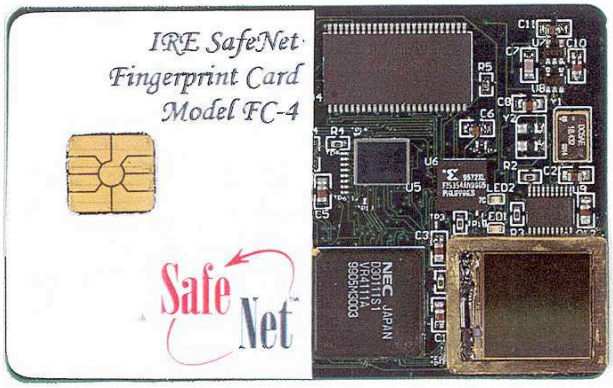
- Entwickelt von Bellcore [RFC 2289] als Nachfolger für S/Key
- Schutz vor Race Angriff:
  - S/Key erlaubt mehrere gleichzeitige Sessions mit einem Passwort
  - Angreifer kann abgehörtes Passwort für kurzen Zeitraum nutzen (Replay Angriff)
- Jede Anmeldung mit OTP braucht eigenes One-Time Passwort
- Sonst nur marginale Änderungen
- Unterstützt verschiedene Hash-Funktionen (MD4, MD5, SHA,..)
- Akzeptiert Passwort auch in Hex Notation
- Passwort muss mind. 10 und kann bis 64 Zeichen lang sein
- Verwendung von IPSec wird „empfohlen“

Angriffe auf S/Key und OTP

- Dictonary Attack:
  - Alle Nachrichten werden im Klartext übertragen, z.B.
    <S/Key 99 12745> <S/Key A GUY SWING GONE SO SIP>
  - Angreifer kann mit diesen Informationen versuchen Passwort des Benutzers zu brechen, z.B.:
    Wort 1: Automobile: BAD LOST CRUMB HIDE KNOT SIN
    Wort k: wireless-lan: A GUY SWING GONE SO SIP
  - Daher empfiehlt OTP die Verschlüsselung über IPSec
- Sicherheit hängt essentiell von der Sicherheit des gewählten Passwortes ab
- Spoofing Angriff:
  - Angreifer gibt sich als Authentisierungs-Server aus
  - Damit Man-in-the Middle Angriff möglich
  - Auch hier: OTP empfiehlt die Verwendung von IPSec zur Authentisierung des Servers

Authentisierung: Smart Cards

- Klassifikation und Abgrenzung:
  1. Embossing Karten (Prägung auf der Karte, z.B. Kreditkarte)
  2. Magnetstreifen-Karten; nur Speicherfunktion (alte EC-Karte)
  3. Smart Card (eingebettete Schaltung):
     - Speicherkarten
     - Prozessor-Karten
     - Kontaktlose Karten

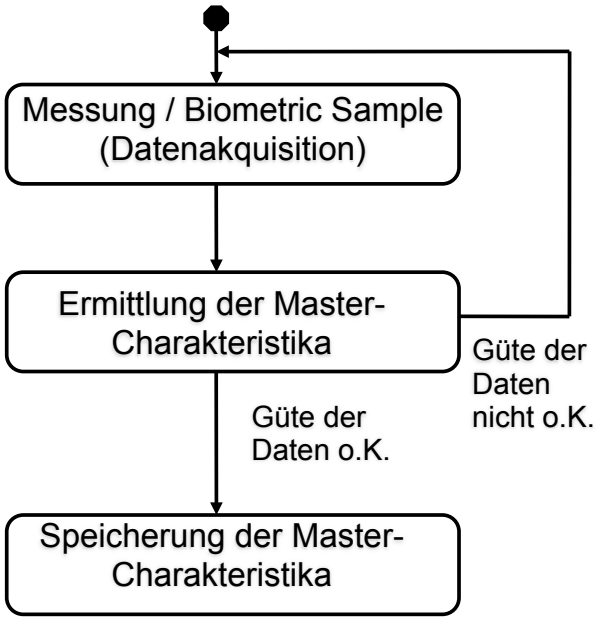


- Bsp.: Prozessor Karte mit Fingerabdruck-Sensor

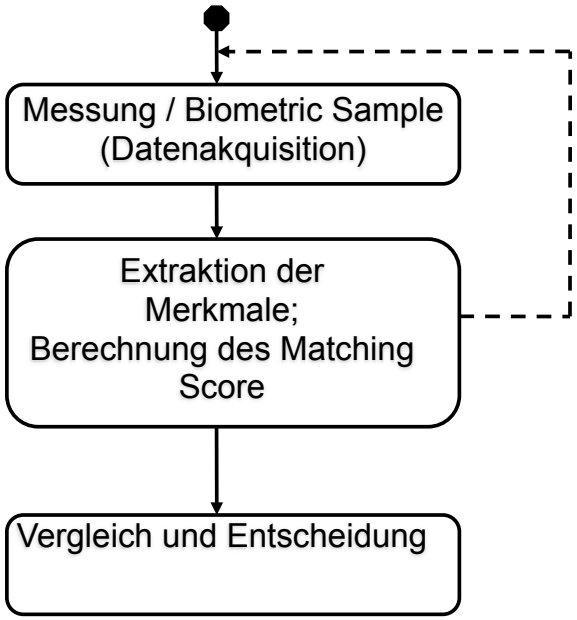
- Zugangsdaten werden auf Karte gespeichert oder erzeugt
  - Schutz der Daten ggf. durch Paßwort und/oder Verschlüsselung

Biometrie: allgemeines Vorgehen

- Initialisierung des Systems pro Nutzer
  - Viele Messungen möglich

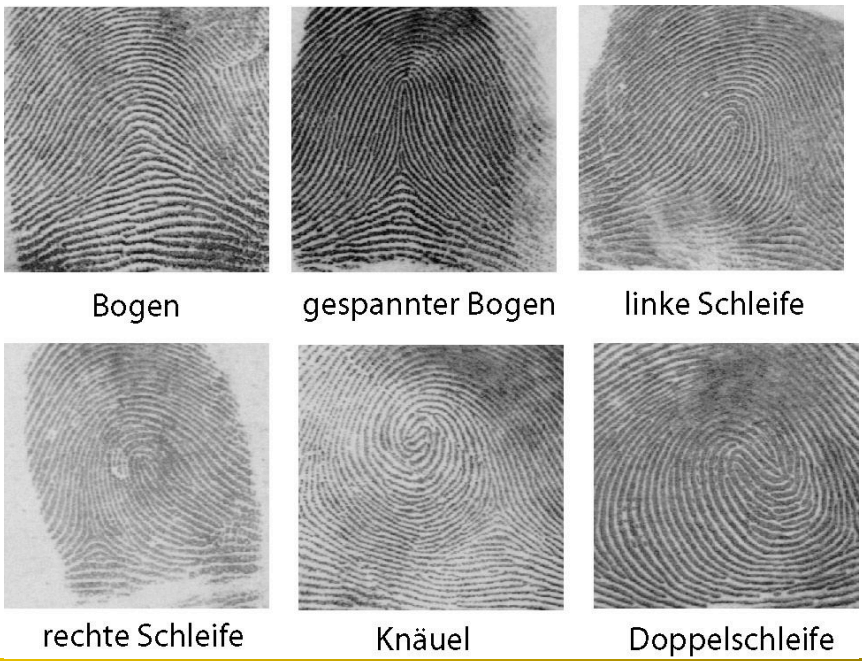


- Authentisierung
  - I.d.R. nur eine Messung möglich



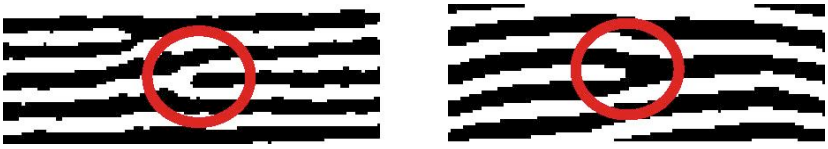
Biometrie am Bsp. Fingerabdruck

- Identifikation anhand des Fingerabdrucks hat lange Geschichte
- Merkmale von Fingerabdrücken sind gut klassifiziert
  Bsp. aus [KaJa96]



Fingerabdruck: Merkmalsextraktion

- Die vorgestellten Klassen lassen sich leicht unterscheiden
- Extraktion sogenannter Minuzien (Minutiae):
  - Repräsentation basierend auf charakteristischen Rillenstrukturen
  - Problem der Invarianz bei unterschiedlicher Belichtung oder unterschiedlichem Druck
    Folgende Beispiele sind äquivalent (entstanden durch untersch. Druck)

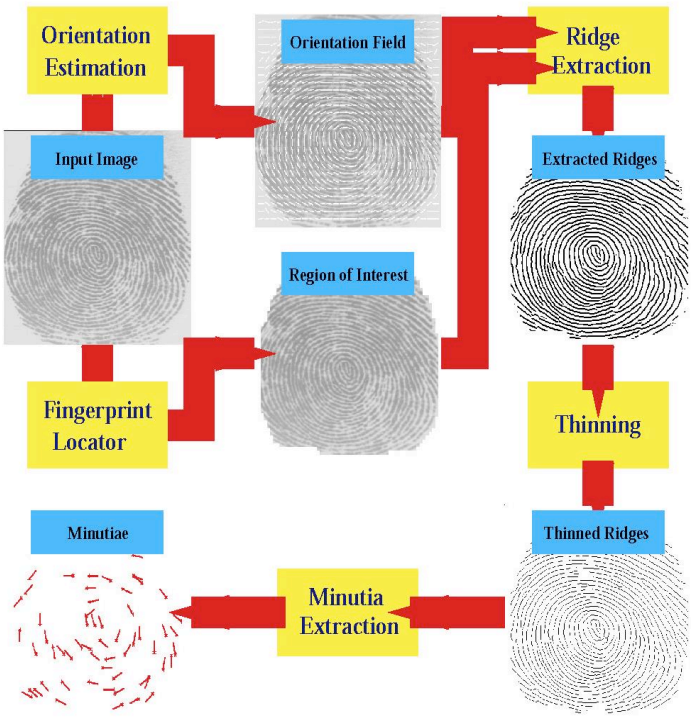


Rillen-Ende Rillen-Verzweigung

  - Solche äquivalente Rillenstrukturen werden zu einer Minuzie zusammen-gefasst
- Merkmale: Lage der Minuzien
  - Absolut bezüglich des Abdrucks, Relativ zueinander
  - Orientierung bzw. Richtung

Fingerabdruck: Minutiae Extraktion

- Algorithmus: Beispiel aus [JHPB 97]

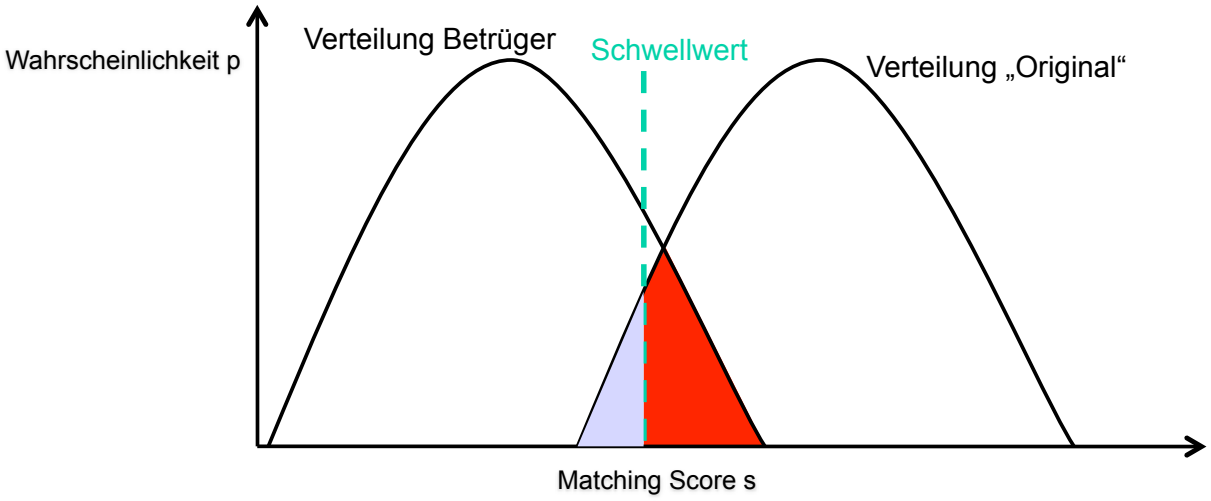


Fingerabdruck: Angriffe

- Sicherheit hängt auch von der Art des Sensors ab
  - Optische Sensoren (Lichtreflexion)
  - Kapazitive Sensoren (elektrische Leitfähigkeit, Kapazität)
  - Temperatur, Ultraschall,.....
- Optische Sensoren können einfach „betrogen“ werden [MaMa 02, Mats 02]
  - Finger-Form mit Hilfe von warmem Plastik abnehmen
  - Form mit Silikon oder Gummi ausgießen
  - Gummi-Finger verwenden
  - Akzeptanzrate bei vielen optischen Sensoren über 80 %
  - Finger-Form kann auch mit einem Fingerabdruck auf Glas erzeugt werden, d.h. der „Original-Finger“ ist **nicht** erforderlich
- Kapazitive Sensoren weisen Gummi Finger i.d.R. zurück
- Verbesserung durch kombinierte Sensoren

Biometrischen Authentisierung: Fehlerarten

- Biometrische Systeme sind fehlerbehaftet
- Fehlerarten:
  1. Falsch Positiv (Mallet wird als Alice authentisiert)
  2. Falsch Negativ (Alice wird nicht als Alice identifiziert)
- Fehler sind abhängig von Schwellwerteinstellungen



Biometrische Authentisierung: Fehlerraten

- Abschätzung der Fehlerraten:
  N: Anzahl der Identitäten
  FP: Falsch Positiv
  FN: Falsch Negativ
- Es gilt [PPK03]:

$$FN(N) \cong FN$$

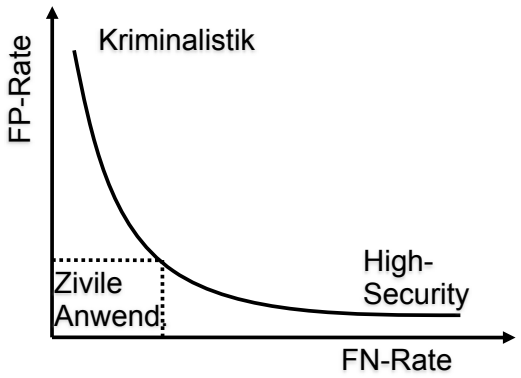
$$FP(N) \cong 1 - (1 - FP)^N \cong N \times FP$$

  falls

$$N \times FP < 0,1$$

- Anwendungsbeispiel:
  - N = 10.000
  - FP = 0,00001
  - Damit FP(N) = 0,1
  - D.h. Fehlerrate von 10 %; Angreifer probiert seine 10 Finger und hat nennenswerte Chance

- Fehlerraten, bzw. Einstellung der Schwellwerte abhängig vom Anwendungsszenario



- Platzierung von Anwendungen?
  - Hohe Sicherheitsanforderungen
  - Kriminalistische Anwendungen
  - "Zivile" Anwendungen

Benutzerauthentisierung: multimodale Systeme

- Sicherheit läßt sich durch multimodale Systeme deutlich erhöhen
- Multimodale Systeme kombinieren versch. Verfahren

	Wissen	Besitz	Biometrie
Wissen			
Besitz			
Biometrie			

- Auch verschiedene biometrische Verfahren lassen sich kombinieren:
  - Erhöhung der Sicherheit
  - Verringerung der Fehlerraten
  - Z.B. Verwendung von mehr als einem Finger

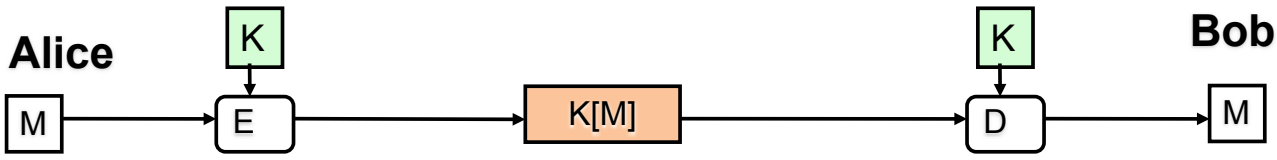
Authentisierung des Datenursprungs

- Möglichkeiten zur Authentisierung des Datenursprungs bzw. zur Peer-Entity-Authentication:

1. Verschlüsselung der Nachricht (Authentisierung erfolgt mittelbar durch Wissen, d.h. Kenntnis des Schlüssels)
2. Digitale Signatur
3. Message Authentication Code (MAC)
   MAC = Hashverfahren + gemeinsamer Schlüssel
4. Hashed Message Authentication Code (HMAC)

- Kombinationen der angegebenen Verfahren

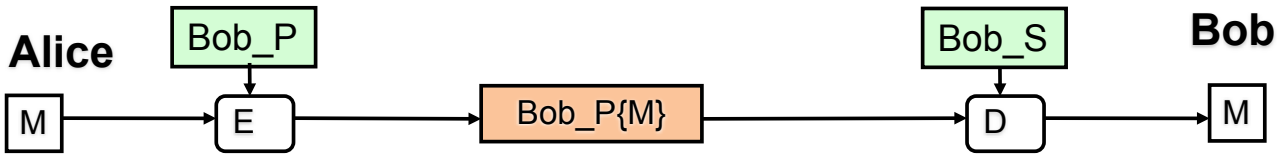
Authentisierung durch symm. Verschlüsselung



- Merkmale:
  - Authentisierung des Datenursprungs (Nachricht kann nur von Alice stammen)
  - Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten (nur Alice und Bob kennen K)

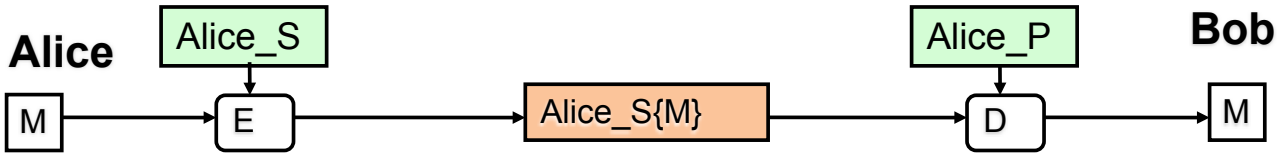
- „Nachteile“:
  - Sender kann die Sendung leugnen
  - Alice / Bob können Zugang / Empfang nicht beweisen

Authentisierung durch asym. Verschlüsselung



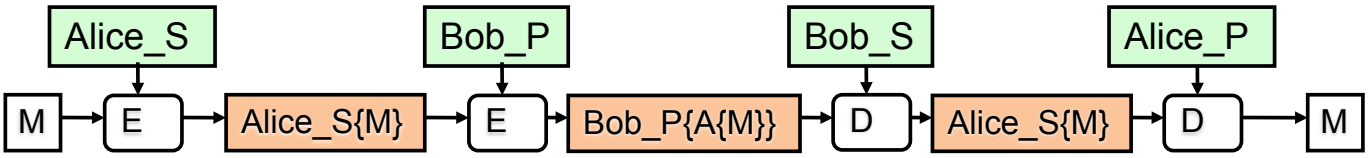
- Merkmale:
  - Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten (nur Bob kennt seinen privaten Schlüssel)
  - KEINE Authentisierung des Datenursprungs (Jeder kann senden)
  - Sender kann die Sendung leugnen
  - Alice / Bob können Zugang / Empfang nicht beweisen

Authentisierung: digitale Signatur



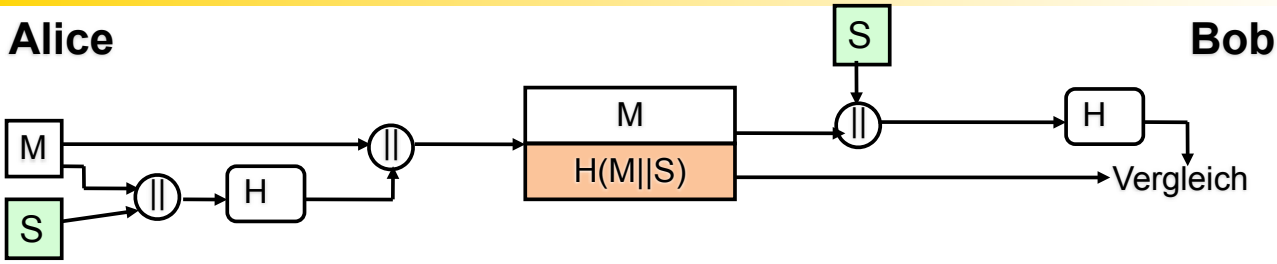
- Merkmale:
  - Authentisierung des Datenursprungs (Nachricht kann nur von Alice stammen, nur Alice kennt ihren geheimen Schlüssel)
  - Jeder kann Signatur verifizieren (auch ohne Mithilfe von Alice)
  - Alice kann Sendung nicht leugnen
  - Bob wird nicht authentisiert
  - Keine Vertraulichkeit (Jeder kann Nachricht lesen, jeder „kennt“ öffentlichen Schlüssel von Alice)
  - Alice kann Zugang nicht beweisen

Authentisierung: asym. Verschlüsselung + Signatur



- Merkmale:
  - Authentisierung des Datenursprungs
  - Nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten
  - Vertraulichkeit der Signatur
  - Alice kann Sendung nicht leugnen
- Operationen für Signatur und asymmetrische Verschlüsselung sind „teuer“
- Alice kann Zugang nicht beweisen
- Bei allen Verfahren bisher, **keine** Integritätssicherung

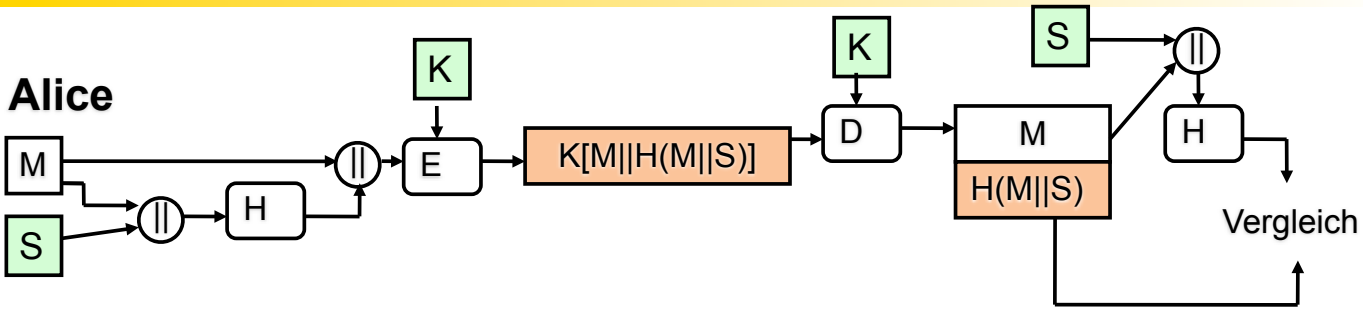
Verwendung von Hash-Fkt. zur Authentisierung



- Authentisierung des Datenursprungs (durch „Geheimnis“ S)
  - Nachricht wird mit S konkateniert und dann der Hash berechnet
- (Daten-) Integrität (durch Hash)

- Keine Vertraulichkeit, jeder kann M lesen
- Alice kann Sendung leugnen
- Alice/Bob können Zugang / Empfang nicht beweisen

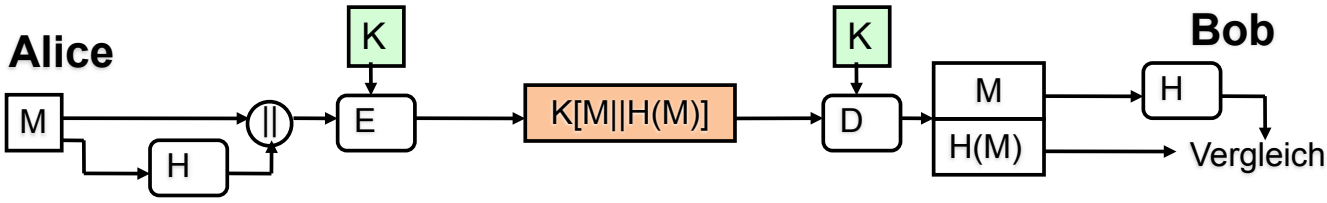
Verwendung von Hash-Fkt. zur Authentisierung



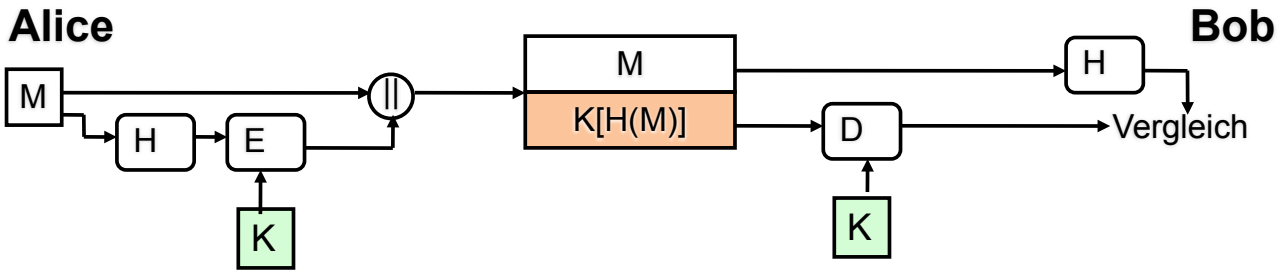
- Zusätzlich Vertraulichkeit durch Verschlüsselung

- Alice kann Sendung leugnen
- Alice/Bob können Zugang / Empfang nicht beweisen

Verwendung von Hash-Fkt. zur Authentisierung

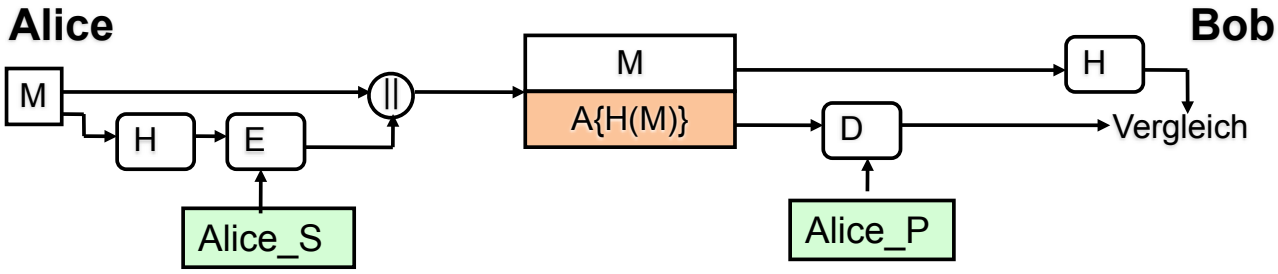


- Authentisierung des Datenursprungs (durch Schlüssel K)
- Vertraulichkeit
- Integrität



- Authentisierung und Integrität, keine Vertraulichkeit

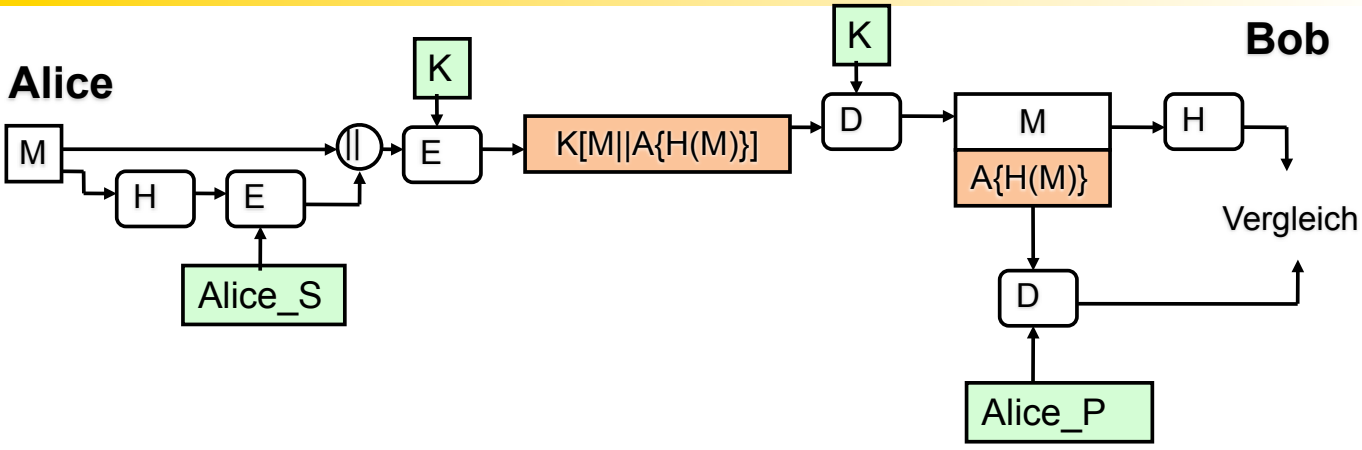
Verwendung von Hash-Fkt. zur Authentisierung



- Authentisierung des Datenursprungs durch digitale Signatur
  - Alice signiert Hash
- (Daten-) Integrität (durch Hash)

- Keine Vertraulichkeit, jeder kann M lesen
- Alice kann Zugang nicht beweisen

Verwendung von Hash-Fkt. zur Authentisierung

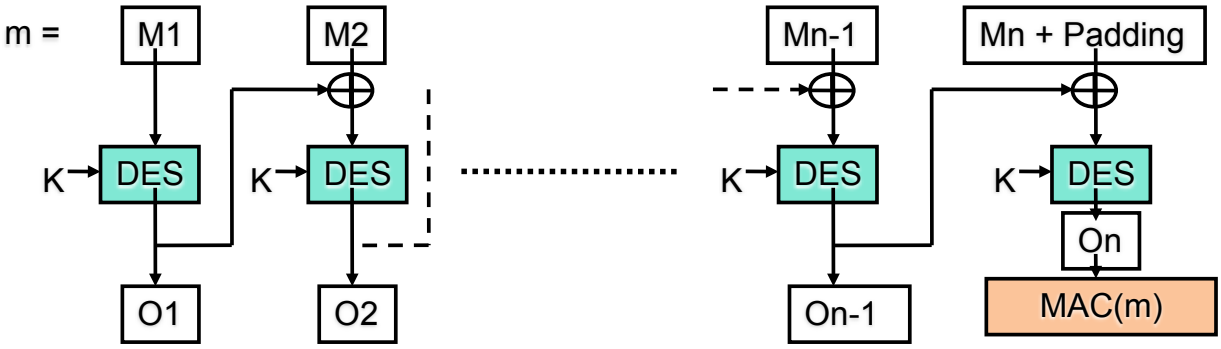


- Zusätzlich Vertraulichkeit durch (symmetrische) Verschlüsselung
- Am häufigsten verwendetes Verfahren

- Alice kann Zugang nicht beweisen

Authentisierung: MAC

- Message Authentication Code (MAC)
- Idee: Kryptographische Checksumme wird mit Algorithmus A berechnet, A benötigt einen Schlüssel
- MAC = A(M,K)
- Authentisierung über Schlüssel K (kennen nur Alice und Bob)
- Beispiel?



- DES im CBC Mode

Sicherheit von MACs

- Wie kann der MAC angegriffen werden?
- Brute force:
  - MAC ist n-Bit lang, Schlüssel K ist k Bit lang mit k > n
  - Angreifer kennt Klartext m und MAC(m,K)
  - Für alle Ki berechnet der Angreifer MAC(m,Ki) = MAC(m,K)
  - D.h. der Angreifer muss 2^k MACs erzeugen
  - Es existieren aber nur 2^n verschiedene MACs (2^n < 2^k)
  - D.h. mehrere Ki generieren den passenden MAC (2^(k-n) Schlüssel)
  - Angreifer muß den Angriff iterieren
    1. Runde liefert für 2^k Schlüssel ca. 2^(k-n) Treffer
    2. Runde liefert für 2^(k-n) Schlüssel 2^(k-2n) Treffer
    3. Runde liefert .... 2^(k-3n) Treffer
  - Falls k < n liefert die erste Runde bereits den korrekten Schlüssel

Hashed MAC (HMAC)

- Gesucht: MAC der **nicht** symm. Verschlüsselung sondern kryptographische Hash-Funktion zur Kompression verwendet
  - Hashes wie SHA sind deutlich schneller wie bspw. DES
- Problem: Hash-Funktionen verwenden keinen Schlüssel
- Lösung HMAC
  - Beliebige Hash-Funktion H verwendbar, die auf (Input) Blöcken arbeitet
  - Sei b die Blocklänge
  - Beliebige Schlüssellänge K mit |K| <= b verwendbar
  - Falls |K| < b:
    - Auffüllen mit 0 Bytes bis |K+| = b; d.h. K+ = K||0....0
  - Schlüssel wird mit Input- (`ipad`) bzw. Output-Pattern (`opad`) XOR verknüpft
    - `ipad` = 0x36 b mal wiederholt
    - `opad` = 0x5c b mal wiederholt

HMAC Algorithmus

$$HMAC(m) = H\Big[(K^+ \oplus opad)||H[(K^+ \oplus ipad)||m]\Big]$$

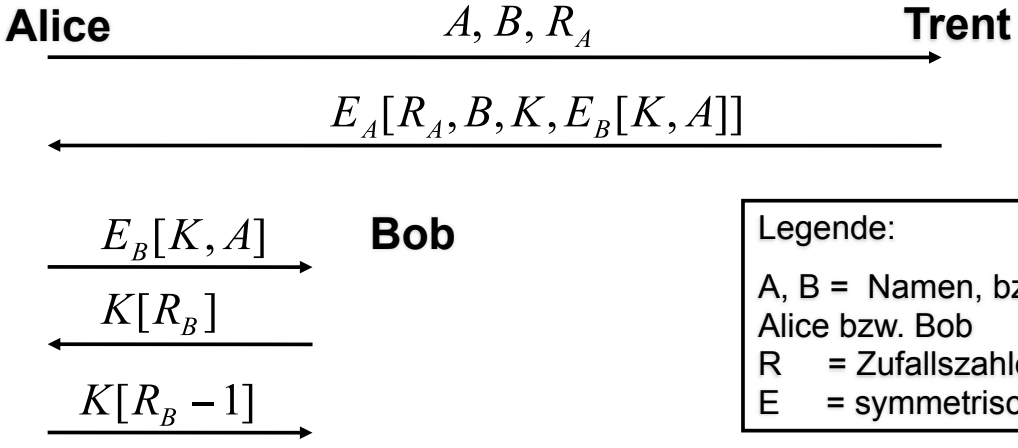
1. $K^+ :=$ Schlüssel K mit Nullen auffüllen bis dieser b Bits lang ist
2. b Bit Block $S_i := K^+ \text{ XOR } ipad$
3. Nachricht m mit dem Block S_i konkatenieren
4. Hash-Wert von $S_i||m$ berechnen
5. b Bit Block $S_o := K^+ \text{ XOR } opad$
6. S_o mit dem Ergebnis von 4. Konkatenieren
7. Hash-Wert über das Ergebnis von 6 berechnen

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Authentisierungsprotokolle: Needham Schröder

- Entwickelt von Roger Needham u. Michael Schroeder (1979)
- Verwendet vertrauenswürdigen Dritten Trent (Trusted Third Party, TTP)
- Optimiert zur Verhinderung von Replay-Angriffen
- Verwendet symmetrische Verschlüsselung
- Trent teilt mit jedem Kommunikationspartner eigenen Schlüssel

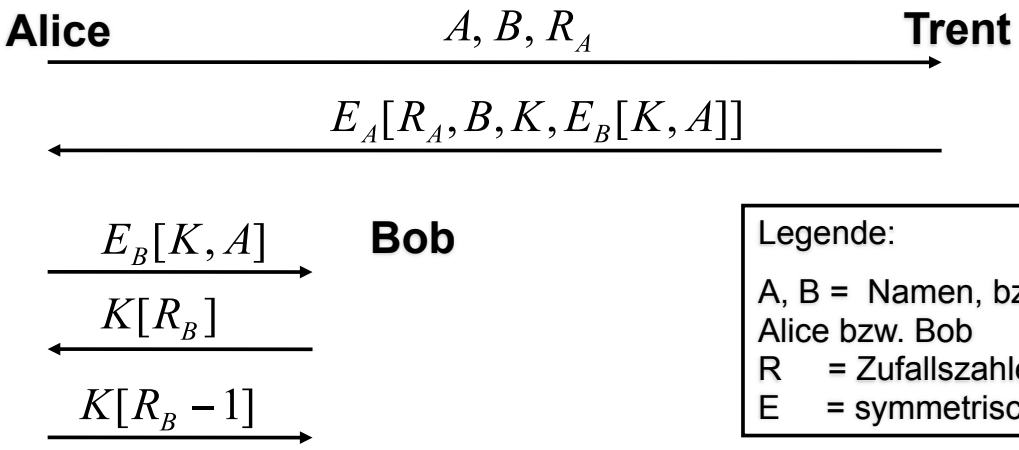


Legende:

A, B = Namen, bzw. Adressen von Alice bzw. Bob
R = Zufallszahlen
E = symmetrisches Verschl.Verf.

Needham Schröder: Diskussion

- Wodurch sollen Reply-Angriffe verhindert werden?
  - Zufallszahlen R_B und „Berechnung“ durch Alice (R_B - 1)
  - Angriff von Mallet die sich durch Reply als Bob ausgeben will werden dadurch verhindert (Protokoll kommt nicht zum Ende)
- Aber Protokollschwäche verhindert Reply-Protection

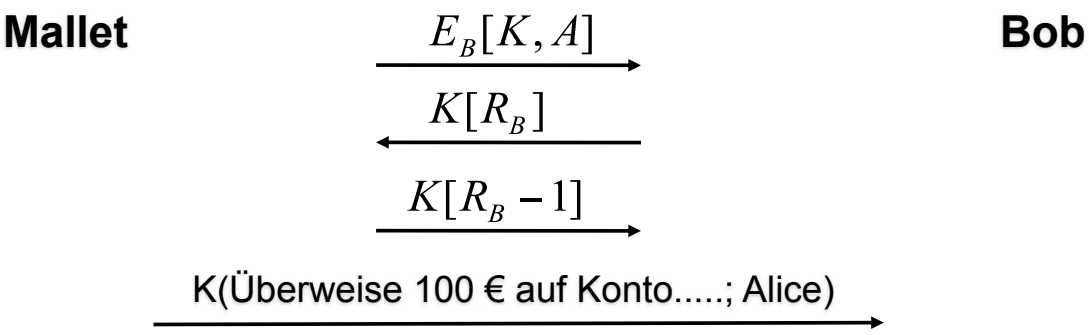


Legende:

A, B = Namen, bzw. Adressen von Alice bzw. Bob
R = Zufallszahlen
E = symmetrisches Verschl.Verf.

Needham Schröder Protokollschwäche

- Problem: Alte Sitzungsschlüssel K bleiben gültig
- Falls Mallet an alten Schlüssel gelangen und die 1. Nachricht von Alice an Bob mithören konnte, wird Maskerade möglich
- Mallet braucht keine geheimen Schlüssel von Trent ($K_{A,T}$, $K_{B,T}$)



- Lösungsidee:
  - Sequenznummer oder Timestamps einführen
  - Gültigkeitsdauer von Sitzungsschlüsseln festlegen

Authentisierungsprotokolle: Kerberos

- Trusted Third Party Authentisierungsprotokoll
- Entwickelt für TCP/IP Netze
  - Im Rahmen des MIT Athena Projektes (X Windows)
  - 1988 Version 4; 1993 Version 5
- Client (Person oder Software) kann sich über ein Netz bei Server(n) authentisieren
- Kerberos Server kennt Schlüssel **aller** Clients
- Basiert auf symmetrischer Verschlüsselung
- Abgeleitet vom Needham-Schröder Protokoll
- Hierarchie von Authentisierungsservern möglich; Jeder Server verwaltet einen bestimmten Bereich (sog. Realm)
- Über Kooperationsmechanismen der Kerberos Server kann Single-Sign-On realisiert werden

Kerberos Authentisierungsdaten

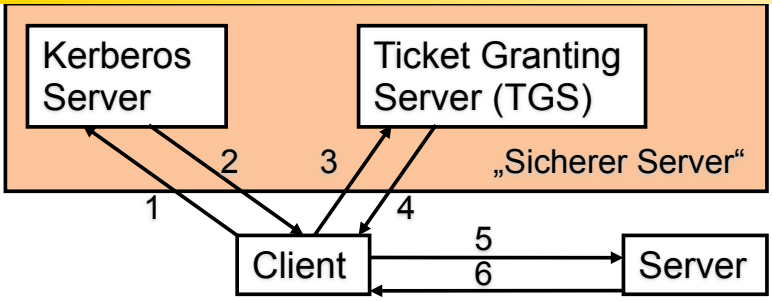
- Authentisierung basiert auf gemeinsamen (Sitzungs-)Schlüssel
- Kerberos arbeitet mit Credentials, unterschieden werden
  1. Ticket
  2. Authenticator
- **Ticket**
  - als „Ausweis“ für die Dienstnutzung; nur für einen Server gültig
  - wird vom Ticket Granting Server erstellt
  - **keine** Zugriffskontrolle über Ticket (nicht mit Capability verwechseln)

$$T_{c,s} = s, c, addr, timestamp, lifetime, K_{c,s}$$

- **Authenticator**
  - „Ausweis“ zur Authentisierung; damit Server ein Ticket verifizieren kann
  - vom Client selbst erzeugt
  - Wird zusammen mit dem Ticket verschickt

$$A_{c,s} = c, addr, timestamp$$

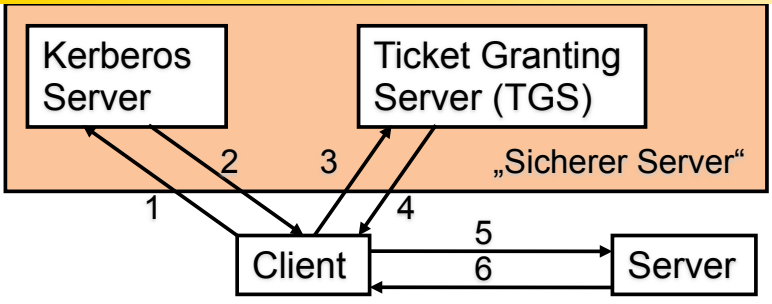
Kerberos Modell



1. Request für Ticket Granting Ticket
2. Ticket Granting Ticket
3. Request für Server Ticket
4. Server Ticket
5. Request für Service
6. Authentisierung des Servers (Optional)

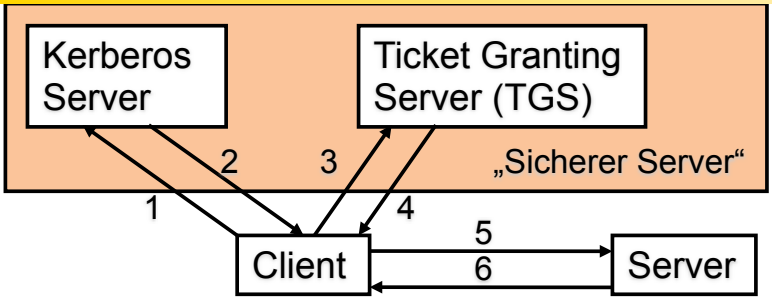
- Im folgenden Kerberos V.5 vereinfacht, d.h. ohne Realms und Optionenlisten; exaktes Protokoll [RFC 1510, Stal98, RFC 4120]

Kerberos: Initiales Ticket (ein Mal pro Sitzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Lebensdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

1. Request für Ticket Granting Ticket:
   c, tgs (Kerberos überprüft ob Client in Datenbank)
2. Ticket Granting Ticket:
   $K_c[K_{c,tgs}], K_{tgs}[T_{c,tgs}]$ mit $T_{c,tgs} = tgs, c, a, t, v, K_{c,tgs}$



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

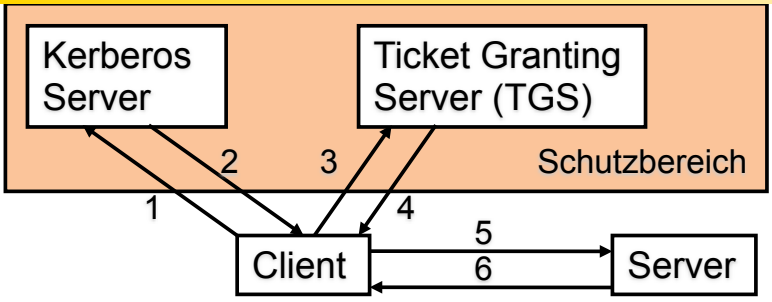
3. Request für Server Ticket:

   $s, K_{c,tgs}[A_{c,tgs}], K_{tgs}[T_{c,tgs}]$ mit $A_{c,tgs} = c, a, t$ $T_{c,tgs} = tgs, c, a, t, v, K_{c,tgs}$

4. Server Ticket:

   $K_{c,tgs}[K_{c,s}], K_s[T_{c,s}]$ mit $T_{c,s} = s, c, a, t, v, K_{c,s}$

Kerberos: Request for Service (pro Service-Nutzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

5. Request for Service:

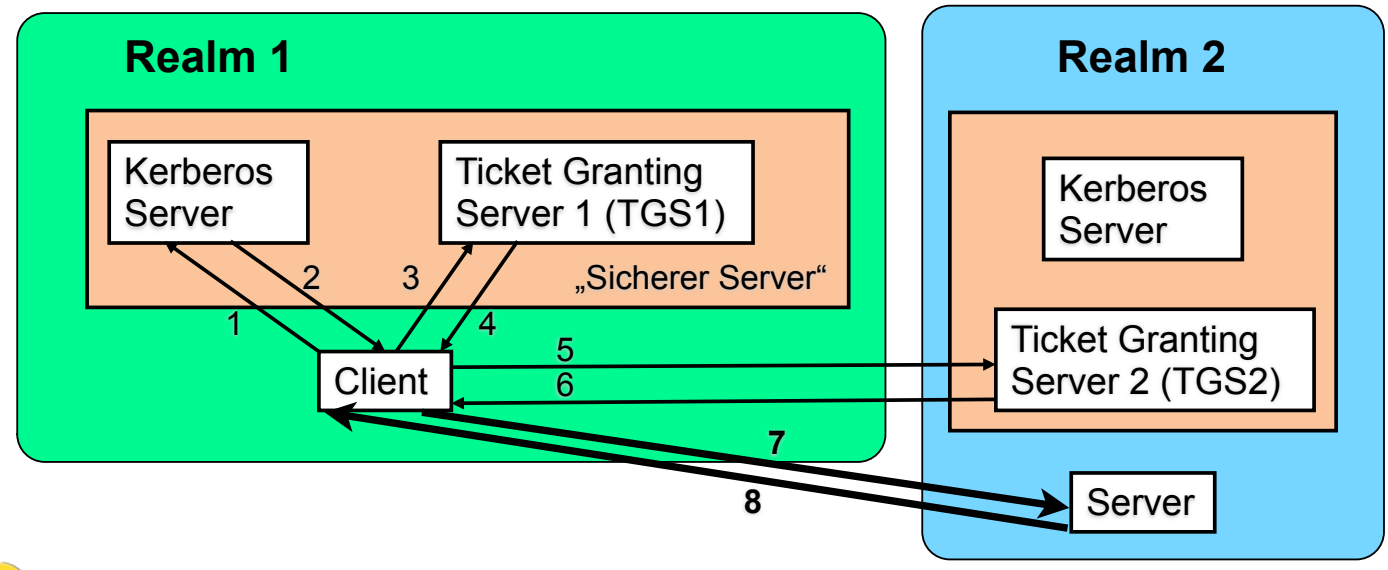
   $K_{c,s}[A_{c,s}], K_s[T_{c,s}]$ mit $A_{c,s} = c, a, t, key, seqNo$ $T_{c,s} = s, c, a, t, v, K_{c,s}$

6. Server Authentication:

   $K_{c,s}[t, key, seqNo]$

Multi-Domain-Kerberos

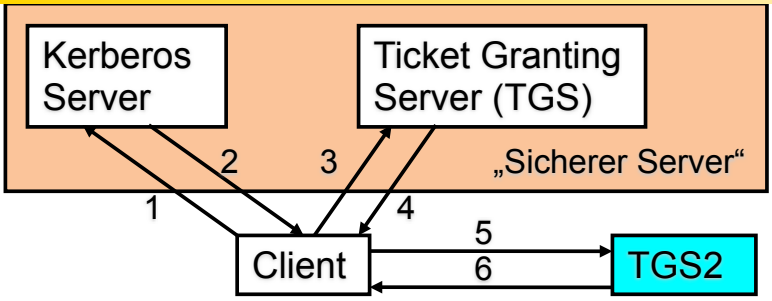
- Kerberos Server immer für eine Domäne (Realm) zuständig
- Frage: Domänenübergreifendes Kerberos (z.B. Kooperation von zwei unabhängigen Unternehmen)
- Idee: TGS der fremden Realm wird „normaler“ Server



Multi-Domain Kerberos

- Domänenübergreifende Authentisierung
- Erfordert Schlüsselaustausch zwischen TGS1 und TGS2: $K_{TGS1,TGS2}$
- Vertrauen (Trust) erforderlich:
  - Besuchende Domäne muss Authenticator und TGS der Heimat-Domäne vertrauen
  - Beide Domänen müssen sich auf „sichere“ Implementierung verlassen
- Skalierungsproblem:
  n Realms erfordern n * (n-1) / 2 Schlüssel, d.h. $O(n^2)$

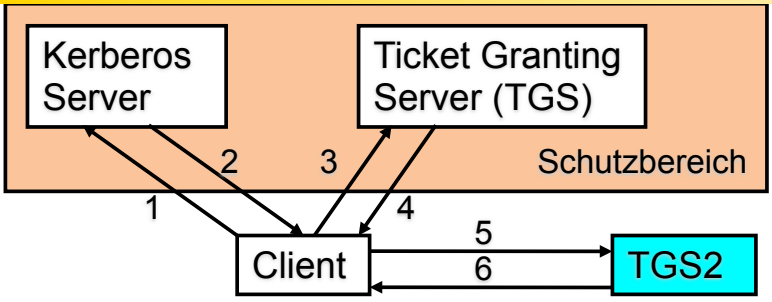
Multi-Domain Kerberos: Erweiterungen



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

3. Request für Server Ticket für fremden TGS (TGS2):
   $tgs2, K_{c,tgs1}[A_{c,tgs1}], K_{tgs1}[T_{c,tgs1}]$
   mit $A_{c,tgs1}=c,a,t$; $T_{c,tgs1}=tgs1,c,a,t,v,K_{c,tgs1}$
4. Server Ticket:
   $K_{c,tgs1}[K_{c,tgs2}], K_{tgs2}[T_{c,tgs2}]$ mit $T_{c,tgs2} = tgs2,c,a,t,v,K_{c,tgs2}$

Kerberos: Request for Service (pro Service-Nutzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

5. Request for Server Ticket beim TG2:
   $s, K_{c,tgs2}[A_{c,tgs2}], K_{tgs2}[T_{c,tgs2}]$
   mit $A_{c,tgs2} = c,a,t$ $T_{c,tgs2} = tgs2,c,a,t,v,K_{c,tgs2}$
6. Server Ticket:
   $K_{c,tgs2}[K_{c,s}], K_s[T_{c,s}]$
7. Weiterer Ablauf wie bei single Domain Kerberos

Kerberos Bewertung

- Sichere netzwerkweite Authentisierung auf Ebene der Dienste
- Authentisierung basiert auf IP-Adresse
  - IP Spoofing u.U. möglich
  - Challange Response Protokoll zur Verhinderung nur optional
- Sicherheit hängt von der Stärke der Passworte ab (aus dem Passwort wird der Kerberos Schlüssel abgeleitet)
- Lose gekoppelte globale Zeit erforderlich (Synchronisation)
- Kerberos Server und TGS müssen (auch physisch) gesichert werden
- Verlässt sich auf „vertrauenswürdige“ Software (Problem der Tojanisierung, vgl. CA-2002-29)

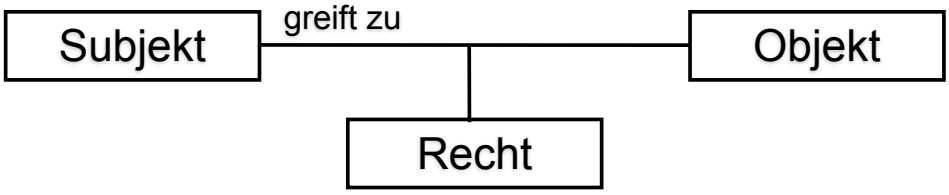
Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Autorisierung und Zugriffskontrolle

- Autorisierung: Vergabe / Spezifikation von Rechten
- Zugriffskontrolle: Durchsetzung dieser Rechte
- Häufig werden Autorisierung und Zugriffskontrolle zusammengefasst

- Handelnde werden als Subjekt bezeichnet
- Rechte werden an Subjekte erteilte
- Rechte gelten für Objekte
- Objekte sind die schützenswerten Einheiten im System



Zugriffskontrollstrategien: Klassifikation

- DAC (Discretionary Access Control)
  - Basieren auf dem Eigentümer Prinzip;
  - Eigentümer spezifiziert Rechte an seinen Objekten
  - Zugriffsrechte auf Basis der Objekte vergeben
- MAC (Mandatory Access Control)
  - Regelbasierte Festlegung der Rechte
  - Systemglobal
  - Z.B. Bell-LaPadula; Regeln werden über Sicherheitsklassen (unklassifiziert, vertraulich, geheim, streng geheim) spezifiziert
- RBAC (Role based Access Control)
  - Trennung von Subjekt und Aufgabe
  - Rechte werden nicht mehr an Subjekt sondern an bestimmte Aufgabe geknüpft
  - Subjekte erhalten Recht über Rollenmitgliedschaft(en)

Zugriffsmatrix

- Schutzzustand eines Systems zum Zeitpunkt t wird durch Matrix M(t) modelliert:
  - M(t) = S(t) x O(t); es gilt M(t): S(t) x O(t) ⟶ 2^R
  - R ist die Menge der Zugriffsrechte
  - Subjekte S bilden die Zeilen der Matrix
  - Objekte O bilden die Spalten
  - Ein Eintrag M(t,s,o) = {r1,r2,...,rn} beschreibt die Menge der Rechte des Subjekts s zum Zeitpunkt t am Objekt o

	Datei1	Datei2	Prozess 1
Prozess 1	read		
Prozess 2		read, write	signal
Prozess 3	read, write, owner		kill

  - Implementierung „spaltenweise“: Zugriffskontrolllisten (z.B. Unix)
  - Implementierung „zeilenweise“: Capabilities

Zugriffskontrolle: Referenzmonitor

- Zur Realisierung der Zugriffskontrolle ist „gesicherte“ Systemkomponente erforderlich
- Häufig als Referenzmonitor oder Access Control Monitor bezeichnet
- Erfüllt folgende Anforderungen:
  - Zugriff auf Objekte nur über den Monitor möglich
  - Monitor kann Aufrufenden (Subjekt) zweifelsfrei identifizieren (Authentisierung)
  - Monitor kann Objektzugriff unterbrechen bzw. verhindern

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Paßwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - Mandatory Access Control (MAC)
   - DAC
5. Identifizierung

Identifikation (Identification)

- Zweifelsfreie Verbindung (Verknüpfung) von digitaler ID und Real-World Entity (Person, System, Prozess,....)
- Ohne sichere Identifikation kann es keine Authentisierung geben
- Mindestens zweistufiger Prozess:
  1. **Personalisierung**:
     Zweifelsfreie Ermittlung der Real-World Identität (bei Personen, z.B. durch Personalausweis) und Vergabe einer digitalen ID (z.B. Benutzername)
  2. **Identifikation**:
     Verbindung von digitaler ID mit Informationen die nur die Entität nutzen / kennen kann (z.B. Passwort, Schlüsselpaar, bzw. öffentlicher Schlüssel)
- Problem: Falls Angreifer in der Lage ist seine Informationen mit fremder ID zu verbinden, kann er Maskerade Angriff durchführen

Identifikation durch digitale Signatur / Zertifikat

- Grundidee: Trusted Third Party (TTP) „unterschreibt“ für Identität einer Entität (vergleichbar mit einem Notar)
- Begriffe:
  - **Zertifikat**: Datenstruktur zur Verbindung von Identitätsinformation und öffentlichem Schlüssel der Entität; digital signiert von einer
  - **Certification Authority (CA) / Trust Center:** Trusted Third Party
  - **Realm**: Benutzerkreis der CA
    - Alle Benutzer in einer Realm „vertrauen“ der CA, d.h.
    - „Aussagen“ der CA werden von allen Benutzern als gültig, richtig und wahr angenommen
  - **(Local) Registration Authority (LRA):** Nimmt Anträge auf ein Zertifikat (**Certification Request**) entgegen; führt Personalisierung durch

Identifikation: Aufgabenspektrum einer CA

- **Generierung von Zertifikaten (Certfication Issue):** Erzeugung der Datenstrukturen und Signatur
- **Speicherung (Certification Repository):** Allgemein zugängliches Repository für Zertifikate
- **Wiederruf und Sperrung (Certification Revocation):** Falls geheimer Schlüssel des Zertifizierten komprimitiert wurde
- **Aktualisierung (Certifcation Update):** Erneuerung des Zertifikates nach Ablauf der Gültigkeit
- **Schlüsselerzeugung (Key Generation)**
- **Historienverwaltung (Certification History):** Speicherung nicht mehr gültiger Zertifik. (zur Beweissicherung)
- **Notardienst (Notarization):** CA signiert Vorgänge zwischen Benutzern (z.B. Verträge)
- **Zeitstempeldienst (Time Stamping):** CA bindet Info an Zeit
- **Realm-übergreifende Zertifizierung (Cross Certification):** Eigene CA zertifiziert fremde CAs
- **Attribut Zertifikate (Attribute Certificate):** Verbindung von Attributen an eine Identität (z.B. Rechte, Vollmachten,....)

Ablauf der Benutzerzertifizierung

1. Schlüsselgenerierung:
   - Zentral durch CA oder dezentral durch Benutzer
   - „Ausreichend sichere“ Schlüssel müssen erzeugt werden
   - Nur der Zertifizierte darf geheimen Schlüssel kennen
2. Personalisierung, Certification Request:
   - Benutzer beantragt ein Zertifikat (Certification Request)
   - Feststellung der Identität des Benutzers (z.B. durch pers. Erscheinen)
   - Benutzer muss belegen dass er im Besitz des passenden privaten Schlüssels ist (z.B. durch Challenge-Response-Protokoll)
3. Generierung der Datenstruktur für das Zertifikat:
   - Entsprechende Attribute werden aus dem Certification Request des Benutzers entnommen
   - Im folgenden X.509 Zertifikate als Beispiel
4. Digitale Signatur durch die CA

X.509v3 Zertifikat: Attribute

- X.509 internationaler ITU-T Standard als Teil der X.500 Serie:
  - Verzeichnisdienst
  - X.500 - X.530 wurde nie vollständig implementiert
- X.509 hat sich auf breiter Basis durchgesetzt
- Drei Versionen:
  - V1: 1988
  - V2: 1993
  - V3: 1995
- Definiert:
  - Datenformat für Zertifikat
  - Zertifikatshierarchie
  - Widerrufslisten (Certificate Revocation Lists, CRL)

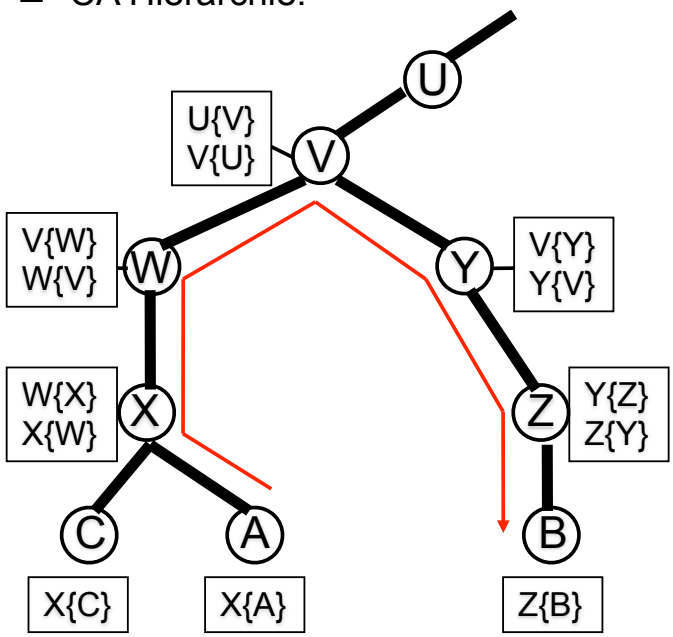
X.509v3 Zertifikat: Attribute

Version 3
Version 2
Version 1

Version	Versionsnummer (1,2,3); Default 1
SerialNumber	Pro CA eindeutige Nummer des Zertifikates
SignatureAlgorithm	Verw. Algorithmus für die digitale Signatur
Issuer	Distinguished Name (DN, vgl. X.500) der CA
Validity	Gültigkeitsdauer; Angegeben in notBefore und notAfter
Subject	„Gegenstand“ des Zert.; z.B. DN des Zertifizierten
SubjectPublicKey-Info	Öffentlicher Schlüssel, des Zertifizierten; Algorithmus für den Schlüssel; ggf. weitere Parameter
IssuerUnique-Identifier	Eindeutiger Bezeichner der CA (ab Version 2 optional); vgl. auch Issuer Feld
SubjectUnique-Identifier	Zusätzliche Info über Subject des Zertifikates (ab Version 2 optional)
Extensions	Ab V. 3: Einschränkungen, Bedingungen, Erweiterungen
Signature	digitale Signatur der gesamten Datenstruktur

Kopplung von Realms; Zertifizierungspfade

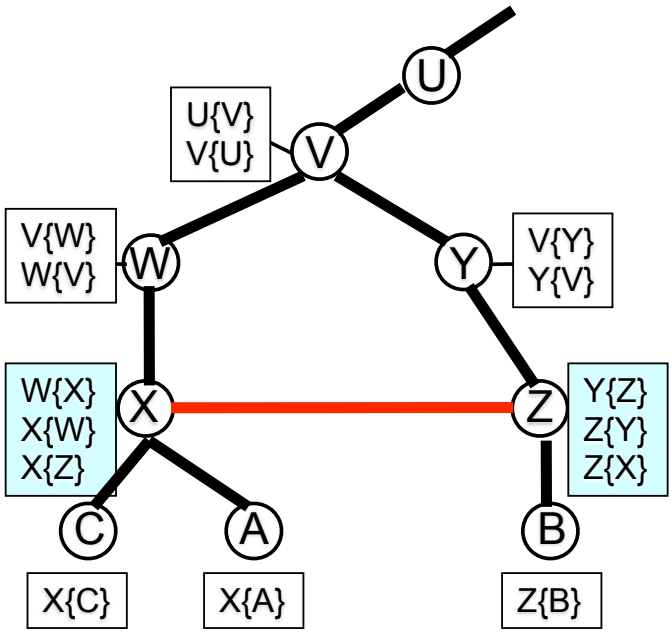
- Bisher wurde nur eine CA betrachtet, nun
- CA Hierarchie:



- Legende: X{A} = Zertifikat ausgestellt von X für A (X zertifiziert A)
- A kommuniziert mit B und möchte dessen Zertifikat verifizieren?
- Dazu Aufbau eines Zertifizierungs-pfades erforderlich:
  - A braucht folgende Zertifikate X{W},W{V},V{Y},Y{Z},Z{B}
  - **Alle** Zertifikate längs dieses Pfades müssen verifiziert werden
  - D.h. A braucht öffentliche Schlüssel von: X, W, V, Y und Z
- Im Bsp. eine streng hierarchische CA Infrastuktur
- Optimierung des Pfades?

Zertifizierungspfade; Cross Zertifizierung

- CA Hierarchie:
- Optimierung des Pfades?



- Cross-Zertifizierung nicht entlang der Hierarchieebenen
- Damit Aufgabe des hierarchischen Ansatzes
- Vermaschte bzw. vernetzte CA Infrastruktur
- Es entsteht ein „Web of Trust“ (vergleichbar PGP)
- Damit Pfade deutlich kürzer
- Pfadermittlung und Pfadverwaltung damit aber u.U. deutlich aufwändiger

Widerruf von Zertifikaten

- Falls Schlüssel kompromittiert wurde muss Zertifikat widerrufen werden
- Dazu Certificate Revocation Lists (CRLs):
  Liste aller Zertifikat-ID mit Datum der Ungültigkeit; digital signiert von CA
- Problem der Informationsverteilung:
  - Zeitnah, d.h. möglichst aktuell
  - Vollständig
  - Effiziente Verteilung
- Grundsätzliche Ansätze:
  - Push-Modell (regelmäßige Übersendung der CRL)
  - Pull Modell (Verifikator fragt bei Überprüfung aktuell nach ob Zertifikat noch gültig oder lädt sich CRL)
  - Vollständige CRL oder Delta Listen

Online Certificate Status Protocol (OCSP)

- Ermöglicht Clients die Abfrage des Zertifikatszustandes (zeitnah) bei einem Server (OCSP-Responder)
- OCSP-Responder i.d.R. betrieben von ausstellender CA
- Ablauf:
  - Client schickt Hash des zu verifizierenden Zertifikats
  - Responder prüft und antwortet mit einer der folgenden **signierten** Nachrichten:
    - „Good“ (Zertfikat ist gültig)
    - „Revoked“ (Zertifikat ist widerrufen, mit entsprechender Zeitangabe)
    - „Unknown“ (Responder kennt das Zertifikat nicht)
  - Reply Protection über optionale Zufallszahl (in Client Nachricht)
  - Client kann Positiv-Antwort fordern; Responder antwortet dann mit Hash des gültigen Zertifikates
- Kein eigenes Transportprotokoll; verwendet HTTP oder HTTPS
- Implementiert von den meisten Browsern

OSCP Diskussion

- Vorteile:
  - Geschwindigkeitsvorteil gegenüber CRL
  - Möglichkeit gesperrte von gefälschten Zertifikaten zu unterscheiden:
    - Responder darf „Good“ nur liefern, wenn Zertifikat gültig (Standard erlaubt Good auch bei Zertifikat nicht in Sperrliste)
  - Individuelle Abfrage für aktuell verwendetes Zertifikat
- Nachteile:
  - Aktualität hängt von Implementierung ab; es gibt Responder die CRL nutzen
  - Zertifikatskette muss vom Client geprüft werden (lässt sich ggf. über Server-based Certification Validation Protocol (SCVP) an den Server auslagern)

IT-Sicherheit

- Sicherheit vernetzter Systeme -

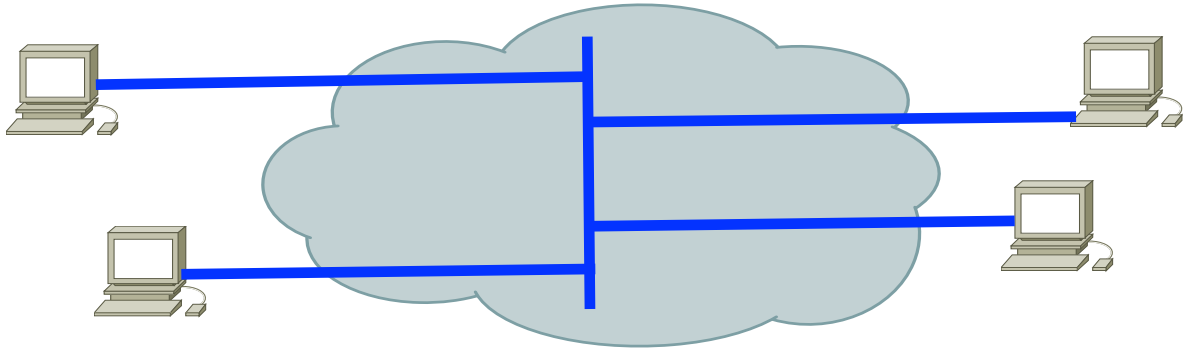
Kapitel 9: Netzsicherheit - Schicht 2: Data Link Layer

Inhalt

- Virtualisierung von Netzen
- Point-to-Point Protocol (PPP)
- Point-to-Point Tunneling Protocol (PPTP)
- Layer 2 Tunneling Protocol (L2TP)
- IEEE 802.1x

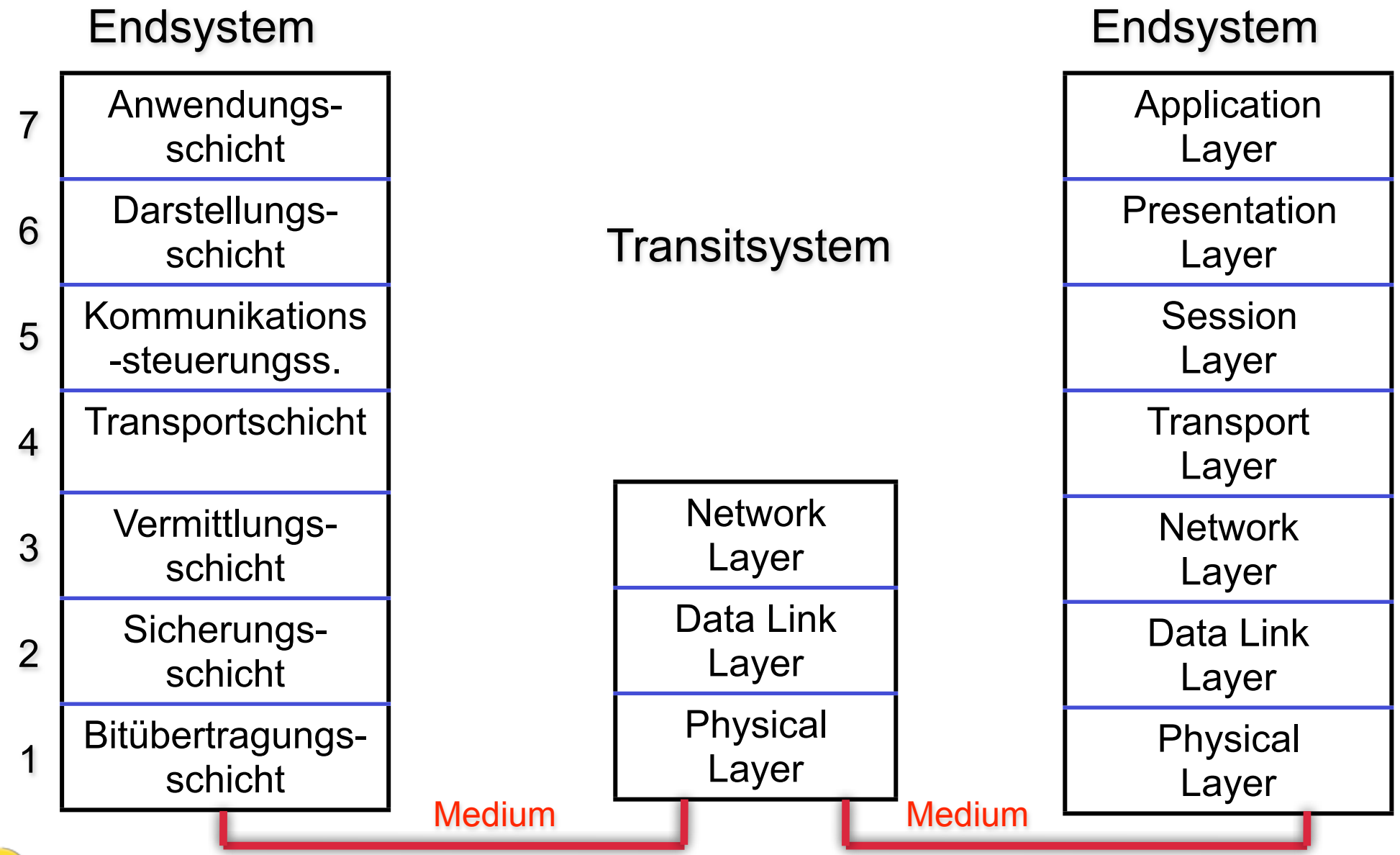
Virtual (Private) Network

- Grundidee:
  Nachbildung einer logischen Netzstruktur („Local Area Network“ oder eines „nicht öffentlichen“ Netzes) in beliebigen Topologien/ Technologien



- Virtualisierung auf jeder Schicht des OSI-Modells möglich

Rückblick: ISO/OSI Schichtenmodell (Kapitel 2)



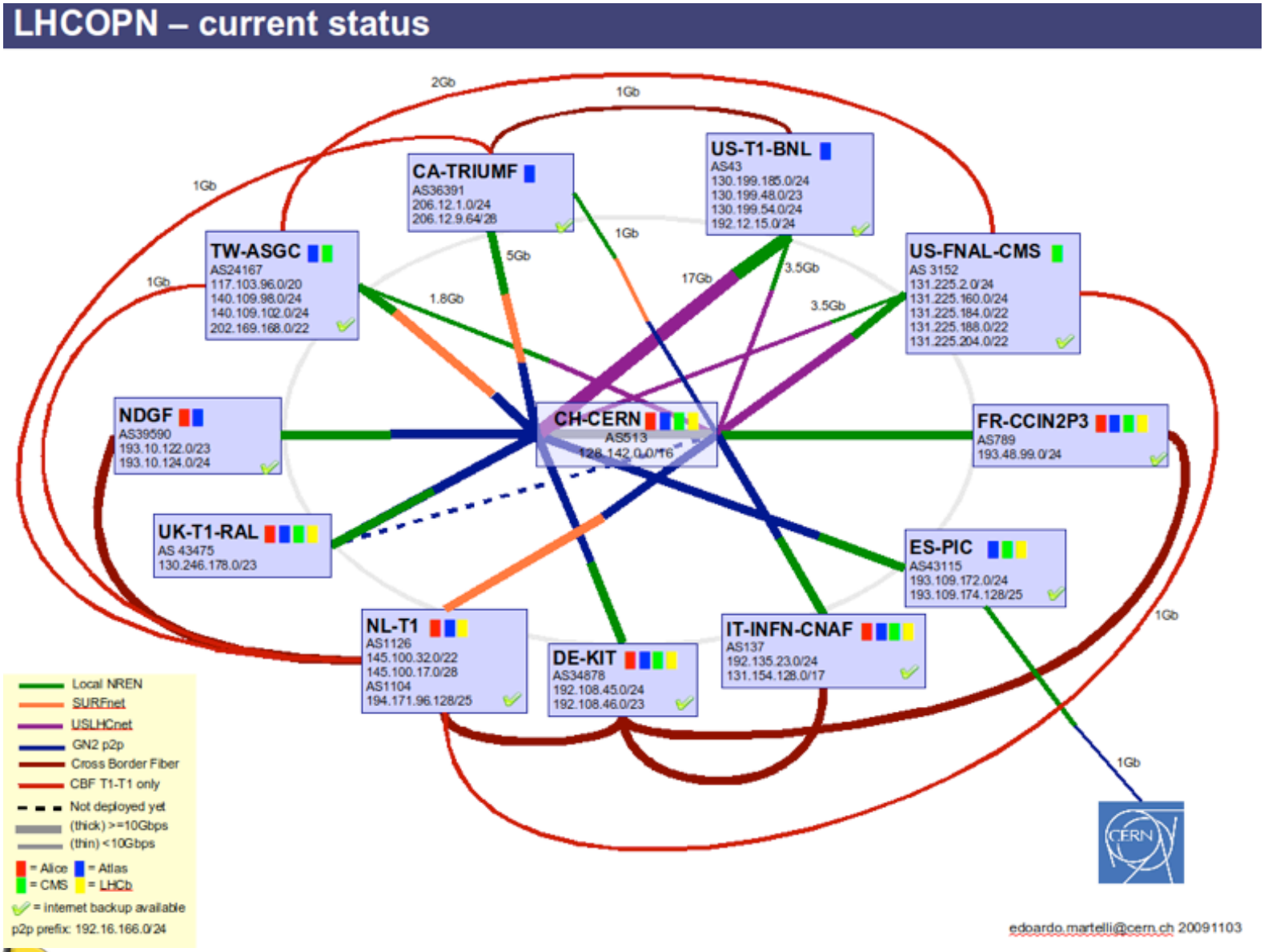
Virtual Network auf Schicht 1

- Virtual Private Wire Service (VPWS)
  - Provider bietet Punkt zu Punkt Verbindung
- Virtual Private Line Service (VPLS)
  - Provider bietet Punkt zu Multipunkt Verbindungen
- Beispiel:
  Optical Private Link oder Optical Private Network (OPN)
  - Provider betreibt Glasfaserinfrastruktur
  - Kunde erhält eine Wellenlänge (Farbe) in dieser Infrastruktur
  - Kunde kann diese nutzen wie einen dedizierten Schicht 1 Link
  - Kunde muss sich um Routing, Bridging, etc. selbst kümmern

Virtual Network auf Schicht 2/3/4

- Schicht 2:
  - Virtual LAN (VLAN)
    - Mehrere LAN Broadcast Domains über den selben physischen Link
    - Standard: VLAN Tagging (IEEE 802.1Q)
  - Virtual Private LAN Services (Achtung: Abkürzung auch VPLS)
    - Verbindet physisch getrennte (V)LANs miteinander
  - Point-to-Point Verbindungen
  - Layer2 Tunneling Protocol
  - ....
- Schicht 3 und höher:
  - IPSec
  - SSL / TLS
  - OpenVPN
  - ...

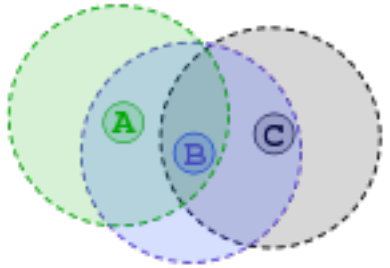
Beispiel: Large Hadron Collider



- Ca. 15 Petabytes an Rohdaten pro Jahr
- Auswertung verteilt auf über 100.000 CPUs
- Verteilung an ca. 6.000 Benutzer in über 450 Instituten

Aufgaben der Schicht 2

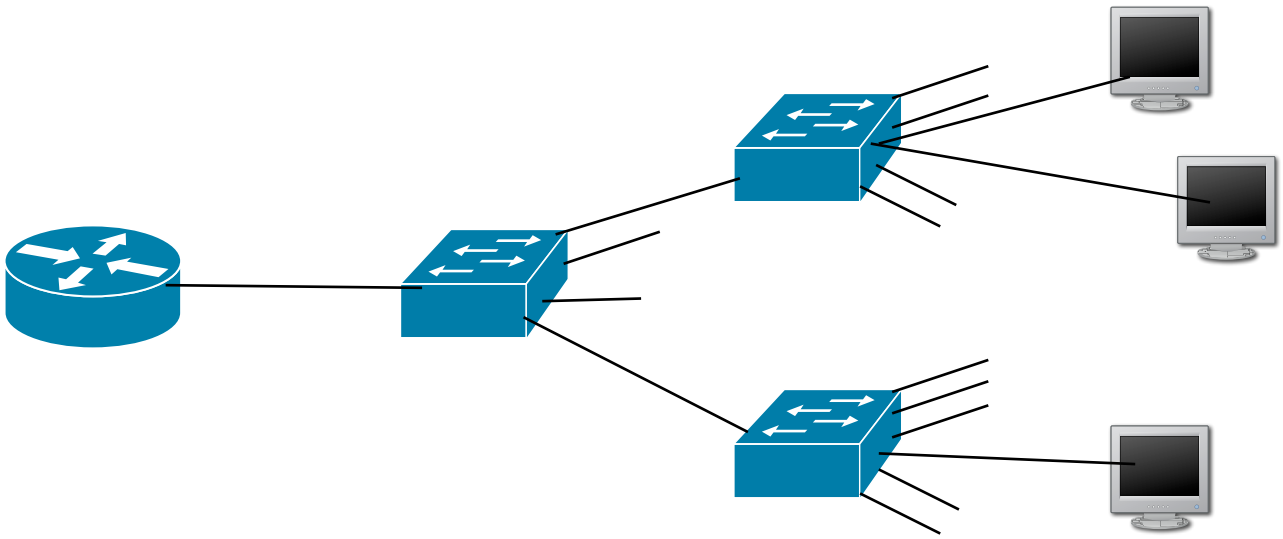
- Fehlerfreie Übertragung von Frames (Rahmen)
  - Aufteilung von Bitströmen in Frames
  - Fehlerkontrolle über Prüfsummen (z.B. Cyclic Redundancy Check, CRC)
  - Quittungs- und Wiederholungsmechanismen
- Flusskontrolle (Verhindert, dass der Empfänger mit Frames überflutet wird und diese verwerfen muss)
- Medienzugriffsverfahren für gemeinsam genutztes Übertragungsmedium
  - CSMA/CD bei Ethernet (IEEE 802.3)
  - CSMA/CA bei WLAN (IEEE 802.11)
  - ....



WLAN: Problem der „hidden stations“

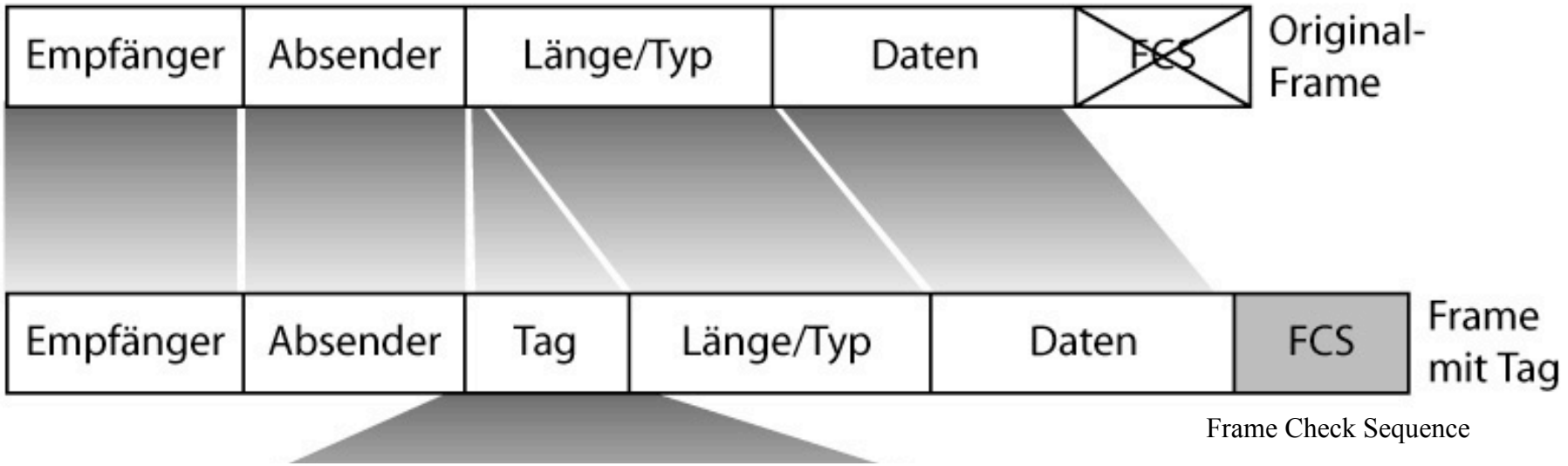
Virtual LAN (VLAN)

- LAN-Infrastruktur über mehrere Switches (Gebäude) hinweg
- Logisch verschiedene LANs auf einer Netzkomponenten
- Wunsch nach Verkehrsseparierung

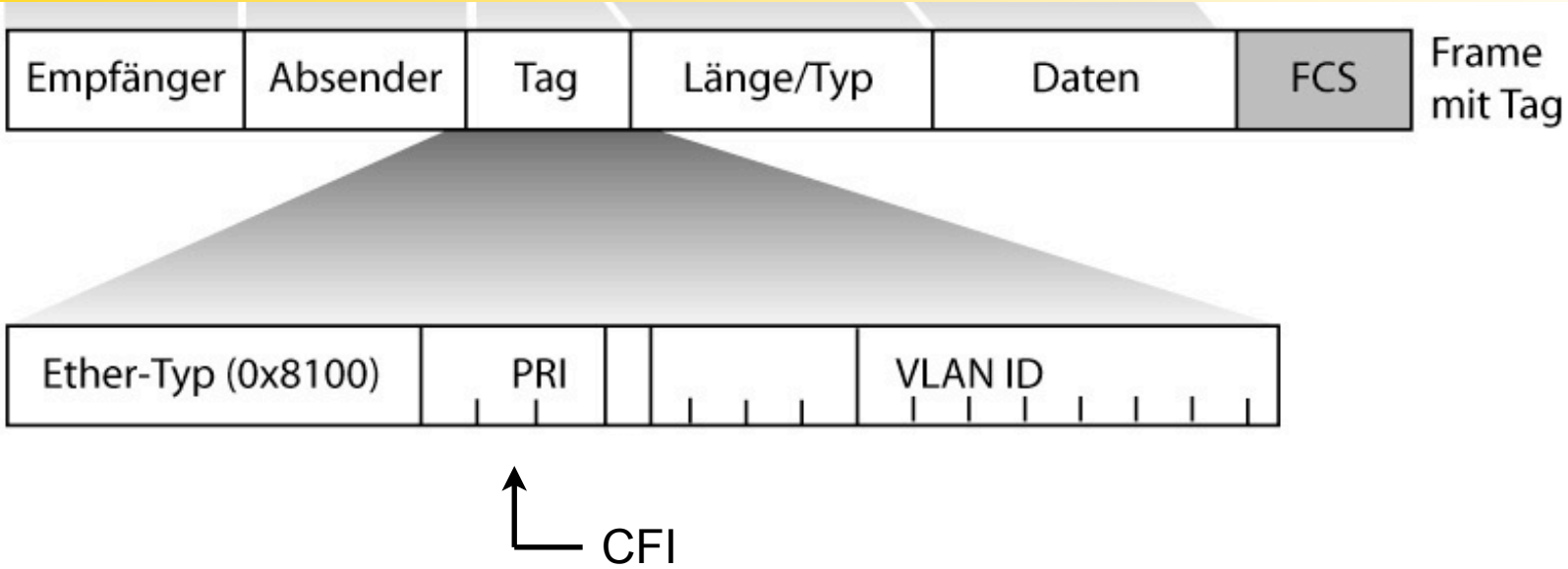


VLAN: Datenpakete

- Virtual Local Area Network (VLAN); IEEE 802.1Q
- VLAN definiert Broadcast Domäne
- Idee: Erweiterung des Ethernet-Frame um sog. Tag



VLAN: Tag Format



- Erweiterung des Ethernet-Frame um 32-bit Tag:
  - TPID (Tag Protocol Identifier): konstant 0x8100; d.h. 802.1Q Tag Information im Frame enthalten (2 Byte)
  - PRI (Priority): Priorisierung nach 802.1p (3 Bit)
  - CFI (Canonical Format Indicator): MAC Adressen in kanonischer Form (1 Bit); bei Ethernet 0; sonst (z.B. Token Ring) 1
  - VLAN-ID: Identifizierung des VLANs („VLAN NR.“) (12 Bit)
    - ID 0 = „kein VLAN“, ID 0xFFF ist reserviert
    - Somit 4094 verschiedene VLANs möglich

PPP: Überblick

- Punkt-zu-Punkt Protokoll; Entwickelt für Verbindungsaufbau über Wählleitungen
  - DSL, ISDN, Modem, Mobilfunk, Funk, serielle Leitungen,....
  - WAN-Verbindungen zwischen Routern
  - Angelehnt an HDLC (Highlevel Data Link Control); Schicht 2 Protokoll
- Spezifiziert in RFC 1661, 1662 und 2153
  - Frame Format mit Begrenzungssymbolen (Delimiter) und Prüfsumme
  - Link Control Protocol (LCP) für:
    - Verbindungsauf- und -abbau
    - Test
    - Aushandeln der Konfiguration (u.a. Nutzdatenlänge pro Frame)
  - Network Control Protocol (NCP) :
    - Aushandeln der Konfiguration der unterstützten Schicht 3 Protokolle (z.B. IP, IPX, Appletalk,...), verschiedene Schicht 3 Protokolle über einen PPP Link möglich
- Weitere Varianten: PPPoE (over Ethernet), PPPoA (over ATM)

PPP: Sicherheitsdienste

- Authentifizierung optional
- Im Rahmen der LCP-Aushandlung der Konfiguration kann jeder Partner eine Authentifizierung fordern
- Definierte Authentifizierungsprotokolle:
  - Password Authentication Protocol (PAP)
  - Challenge-Handshake Authentication Protocol (CHAP)
  - Extensible Authentication Protocol (EAP)

Password Authentication Protocol (PAP)

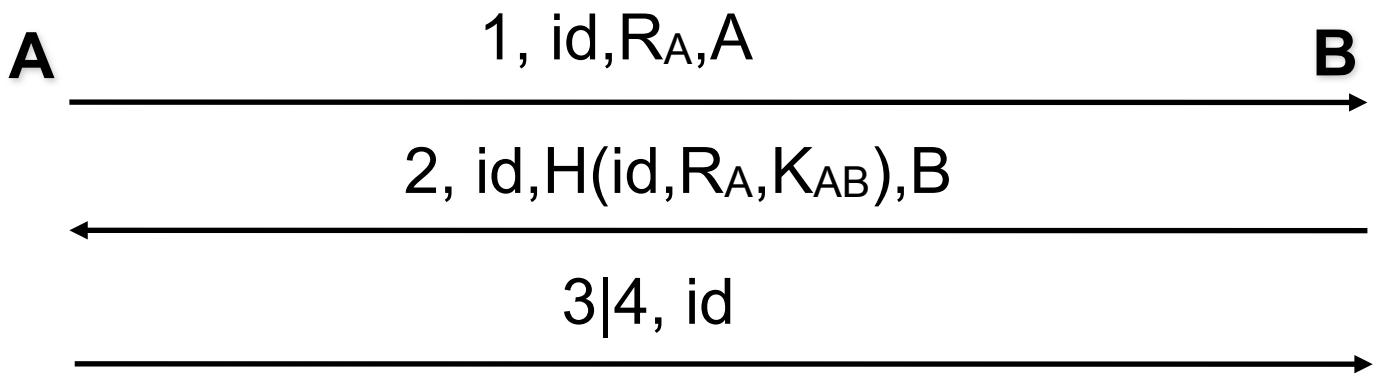
- Spezifiziert in RFC1334
- Authentisierende Entität kennt ID und Passwort aller Clients
- Client wird mit LCP zur Authentisierung via PAP aufgefordert
- Client schickt ID und Passwort **im Klartext**
- Server schickt im Erfolgsfall ACK

- Keine Verschlüsselung, Übertragung der Passwörter im Klartext

➡ Unsicheres Protokoll
RFC 1334: *„Any implementations which include a stronger authentication method (such as CHAP, described below) MUST offer to negotiate that method prior to PAP.“*

Challenge-Handshake Authentication Protocol: CHAP

- (Auch) RFC1334 und RFC1994
- Periodische Authentisierung durch 3-Way-Handshake Protokoll
- Basiert auf gemeinsamen Geheimnis (Passwort) K_{AB}
- A (Authenticator) fordert B zur Authentisierung auf:



  - id: 1 Byte Identifier („incrementally changing“) gegen Playback-Angriffe
  - R_A : Zufallszahl, H: Hash Verfahren, im Standard MD5
  - 3 = success; 4 = failure
- Auth-Request kann später beliebig neu geschickt werden

Sicherheitsrisiko PAP-Fallback

- Viele Clients unterstützen immer noch Server, die nur PAP anbieten
  - Für Client-Hersteller einfach zu implementieren
  - Abwärtskompatibilität vom Markt gewünscht
  - Die meisten Anwender kennen den Unterschied zwischen PAP, CHAP, etc. sowieso nicht: Hauptsache, es funktioniert!

- Man-in-the-middle-Angriff
  - Client kommuniziert nicht direkt mit Server, sondern über Angreifer
  - Angreifer gibt sich als „nur PAP“-Server aus
  - Angreifer erhält Klartext-Passwort vom Client
  - Somit kann der Angreifer u.a. als CHAP-fähiger Client gegenüber dem richtigen Server auftreten

Extensible Authentication Protocol (EAP)

- RFC3748 und RFC5247
- Authentisierungs-Framework, bietet gemeinsame Funktionen und Aushandlungsmechanismen für konkretes Verfahren (als Methode bezeichnet)
- Rund 40 Methoden werden unterstützt:
  - EAP-MD5; äquivalent zu CHAP
  - EAP-OTP (One Time Password); vgl. Kapitel 8
  - EAP-GTC (Generic Token Card)
  - EAP-TLS (Transport Layer Security) vgl. Abschnitt über SSL/TLS
  - EAP-SIM (Global System for Mobile Communications (GSM) Subscriber Identity Modules (SIM)
- Herstellerspezifische Methoden:
  - LEAP (Cisco) Lightwight Extensible Authentication Protocol
  - PEAP (Cisco, Microsoft, RSA) Protected Extensible Authentication Prot.
  - ....

EAP Grundlagen

- EAP kann Sequenz von Verfahren verwenden
- Verfahren muss aber vollständig abgeschlossen werden, bevor neues beginnt
- Request - Response Schema mit Success / Failure Antwort

- Beispiel: EAP-GTC (Generic Token Card, RFC3748)
  - Nutzbar für verschiedenste Autentisierungs-Token-Implementierungen
  - Request beinhaltet Nachricht, die dem Nutzer angezeigt wird
  - Nutzer gibt Token-Information ein
  - Server prüft und antwortet





Point to Point Tunneling Protocol (PPTP)

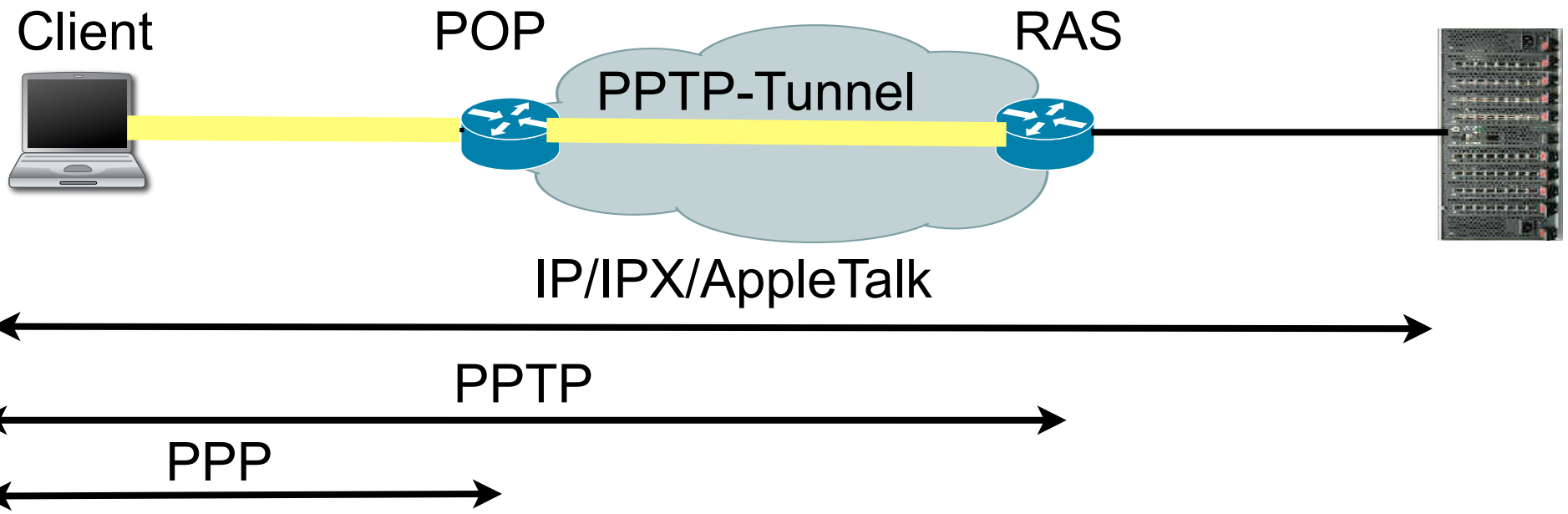
- PPP wurde für „direkt“ verbundene Systeme entwickelt
- Idee von PPTP (RFC2637, aber kein IETF-Standard):
  - Ausdehnung von PPP über Internet
  - PPTP realisiert Tunnel durch / über das Internet
  - Transport von PPP PDUs in IP-Paketen
  - Dazu werden PPP PDUs mit Generic Router Encapsulation Protocol (GRE) gekapselt
  - GRE ist ein Schicht 4 Protokoll

PPP Protocol Data Unit (PPP PDU)
GRE
IP
Sicherungsschicht
Bitübertragungsschicht (Physical Layer)

PPTP: Anwendungsfälle

- Eines der ersten einfach zu konfigurierenden VPN-Protokolle mit weiter Verbreitung seit Windows 95
- Verbindung eines Clients mit einem Remote Access Server (RAS)
  - Voluntary Tunneling
  - Client setzt PPTP aktiv ein
- Verbindung eines ISP Point of Presence (POP) mit einem PPTP Remote Access Server
  - Compulsory Tunneling
  - Client weiß nichts von PPTP
  - ISP POP handelt als Proxy (Stellvertreter) des Clients

PPTP: Voluntary Tunneling

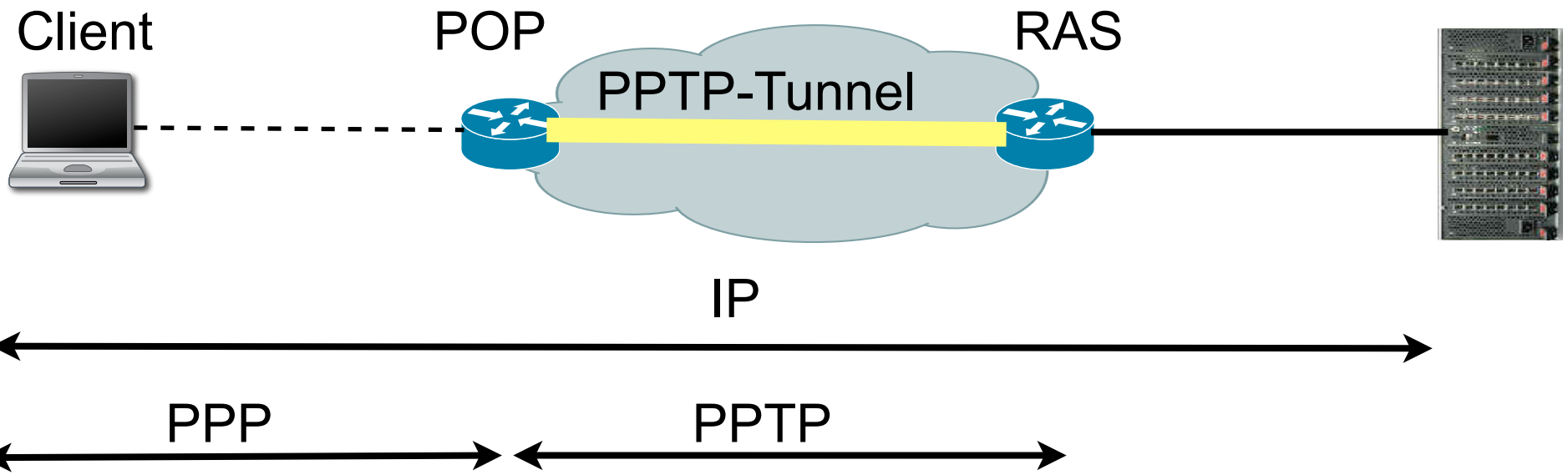


IP/IPX/AppleTalk
PPP
GRE
IP
Highlevel Data Link Ctrl.
Bitübertragungssch.

IP/IPX/AppleTalk
PPP
GRE
IP
Sicherungsschicht
Bitübertragungssch.

IP/IPX/AppleTalk
Sicherungsschicht
Bitübertragungssch.

PPTP: Compulsory Tunneling



IP
PPP
Highlevel Data Link Control
Bitübertragungssch.

IP
PPP
GRE
IP
Sicherungsschicht
Bitübertragungssch.

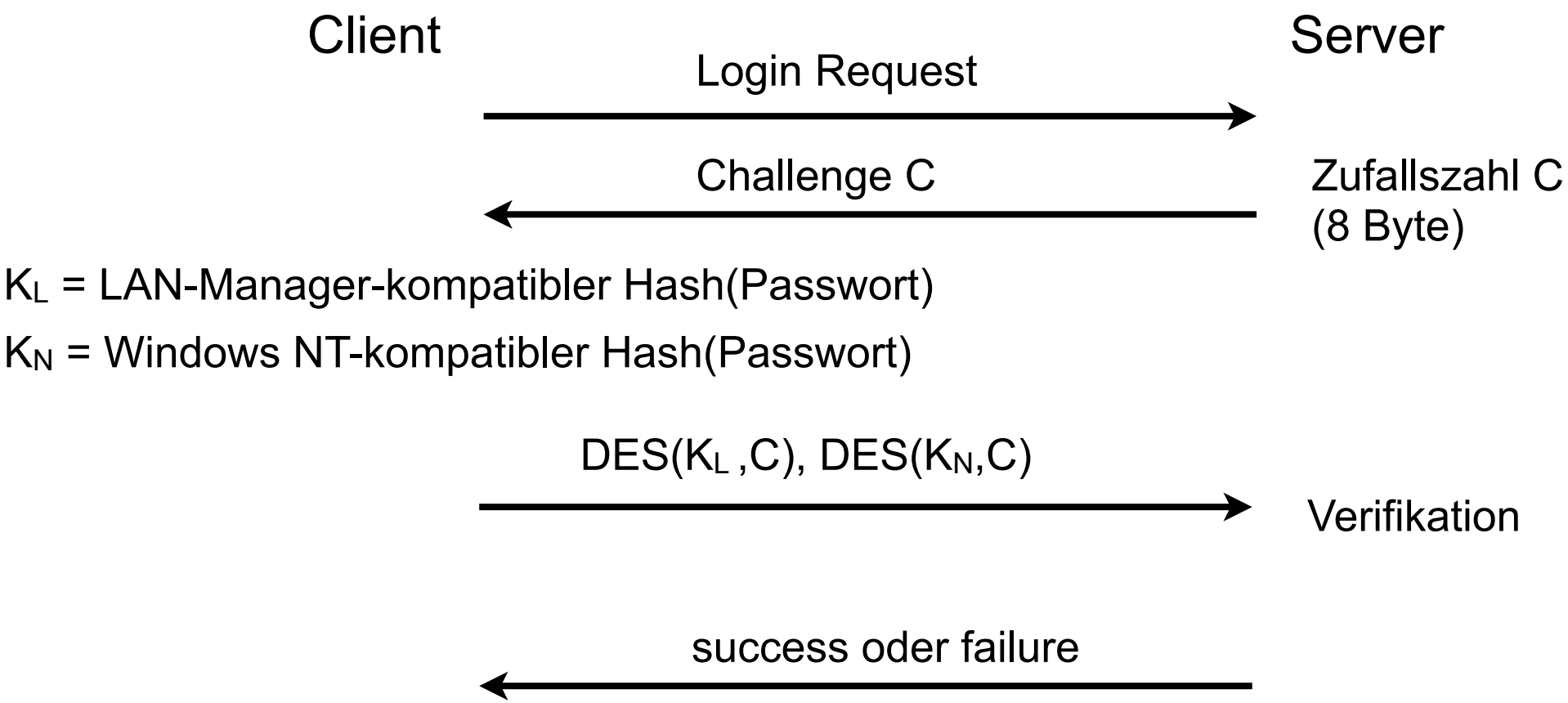
IP
Sicherungsschicht
Bitübertragungssch.

PPTP Sicherheit

- Von Microsoft entwickelt [RFC 2637] als Teil des Remote Access Service (RAS)
- Microsoft-eigene Erweiterungen:
  - Microsoft PPP CHAP (MS-CHAP) [RFC 2433]
  - Microsoft Point to Point Encryption Protocol (MPPE) [RFC 3078]
- Analyse von Bruce Schneier 1998; Fehler in
  - Password Hashing: schwacher Algorithmus erlaubt Eve, das Passwort zu ermitteln (Stichworte: LAN Manager Passwort und L0phtCrack)
  - Challenge/Response Protokoll erlaubt Maskerade-Angriff auf RAS Server (keine beidseitige Authentifizierung)
  - Verschlüsselung: Implementierungsfehler erlaubt Dekodierung
  - Verschlüsselung: Geratenes Passwort erlaubt Entschlüsselung
  - Kontrollkanal: Unautorisierte Nachrichten erlauben DoS (Crash des Servers)
  - Details: http://www.schneier.com/paper-pptp.pdf
- Microsoft besserte nach: PPTP v2 und MS-CHAPv2 [RFC 2759]

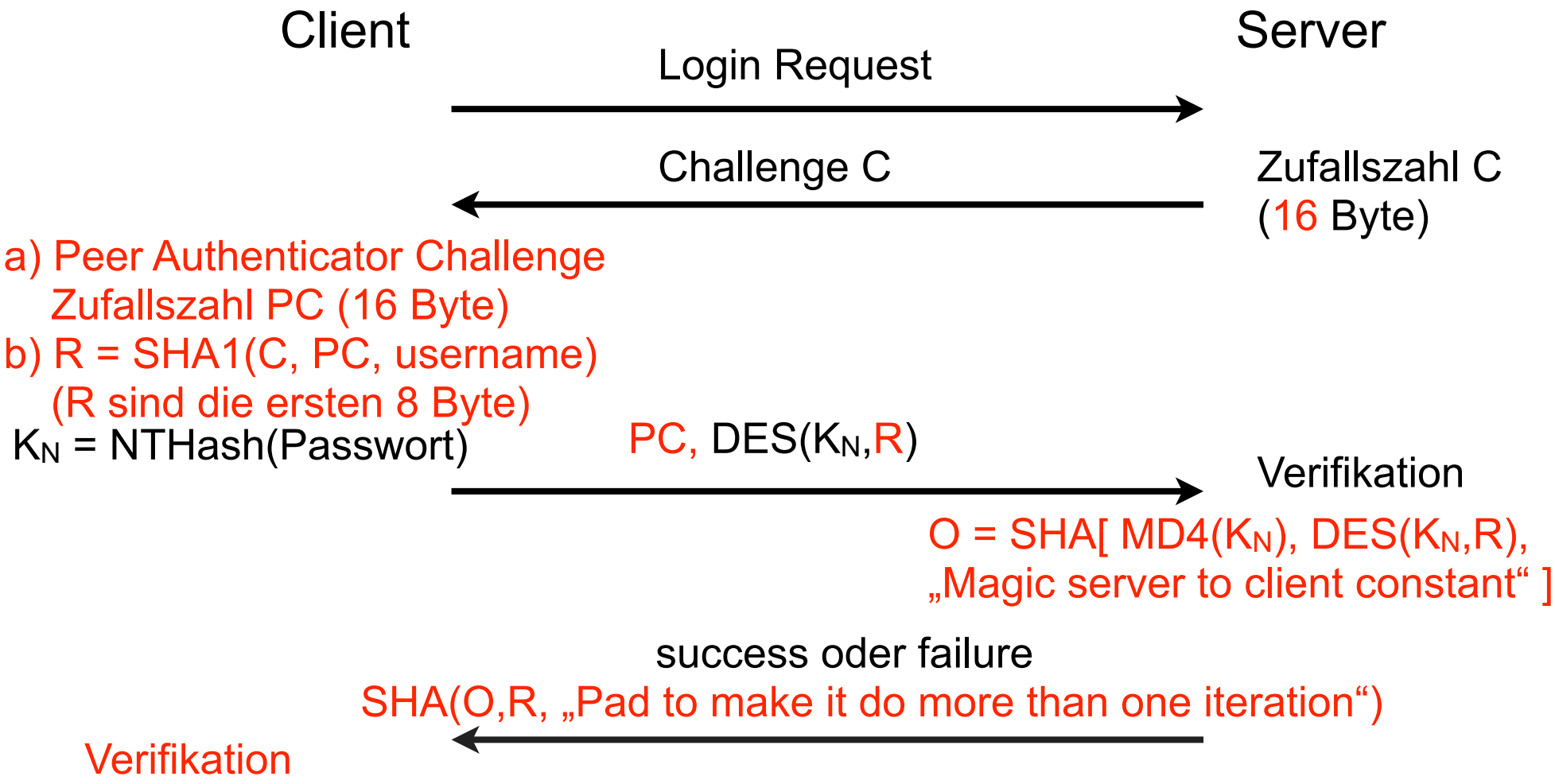
Vergleich MS-CHAP v1 und v2

- Version 1:



Vergleich MS-CHAP v1 und v2

- Änderungen in der Version 2



Sicherheit MS-CHAP v2

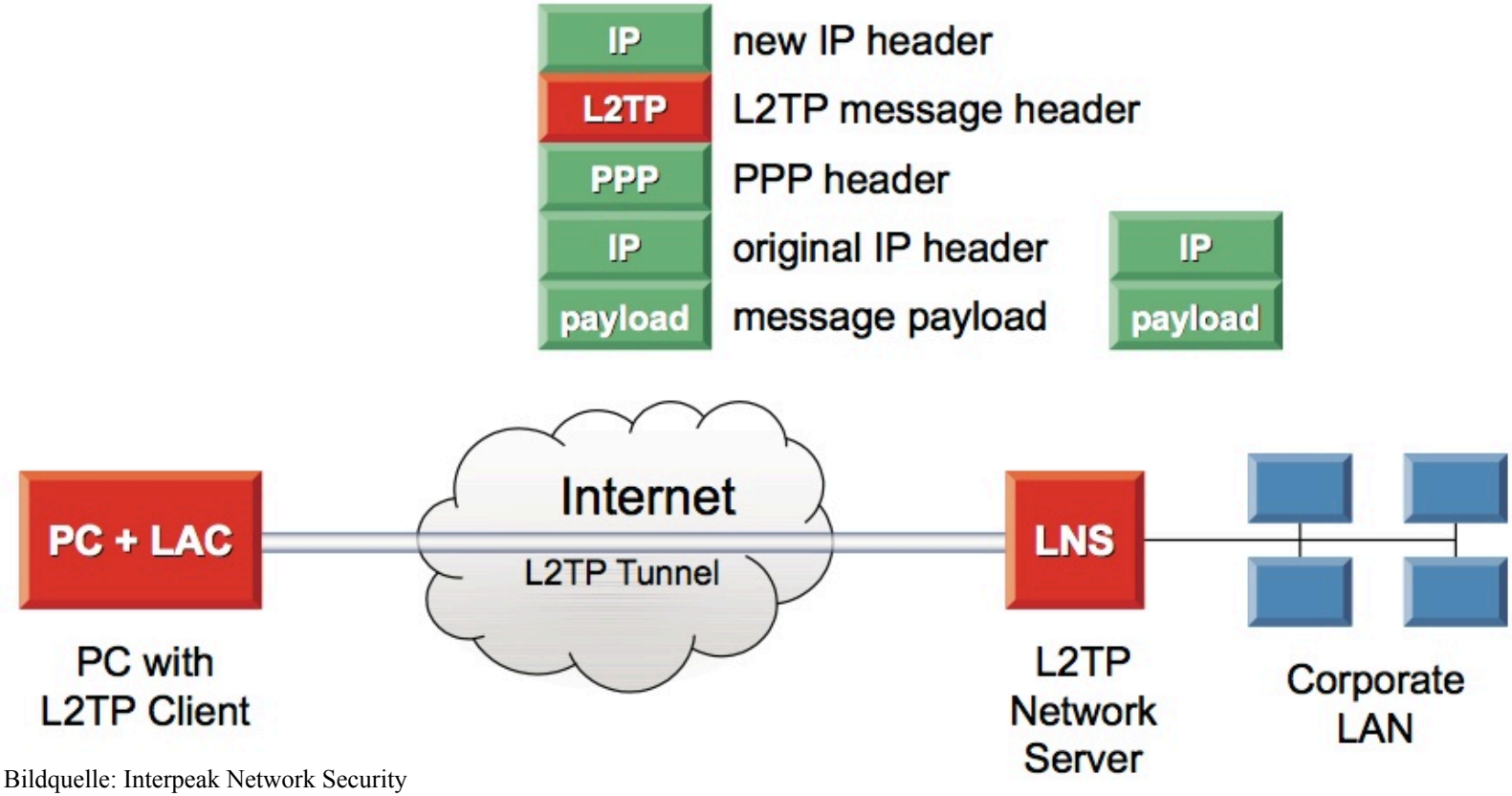
- Protokoll komplizierter als nötig
- Nutzen der „piggybacked“ Peer Authenticator Challenge PC fragwürdig
- Fazit von Bruce Schneier:
  - Auch MS-CHAP v2 hat keinen integrierten Schutz vor Angriffen
  - Starke Abhängigkeit von der Wahl eines „guten“ Benutzerpassworts
  - Bessere Verfahren (z.B. Encrypted Key Exchange und Varianten) waren bereits verfügbar, wurden von Microsoft aber nicht genutzt
- Version Rollback Attack möglich:
  Mallet „überzeugt“ Client und Server, MS-CHAP v1 zu verwenden

Layer 2 Tunneling Protocol (L2TP)

- L2TP [RFC 2661] entwickelt für Tunneling von PPP-Paketen
- Unterstützt verschiedene darunterliegende Protokolle:
  UDP, ATM, FrameRelay
- Tunnel-ID erlaubt Multiplexing von Verbindungen/Tunneln
- Authentisierung mittels CHAP oder PAP
- Vertraulichkeit über IPSec möglich [RFC 3193]
  (IPSec später in der Vorlesung)

L2TP Datenfluss

- L2TP Access Concentrator (LAC) verpackt Frames im lokalen Netz in eine L2TP-Nachricht
- L2TP-Nachricht wird über das Internet an den L2TP Network Server (LNS) geschickt und dort „ausgepackt“



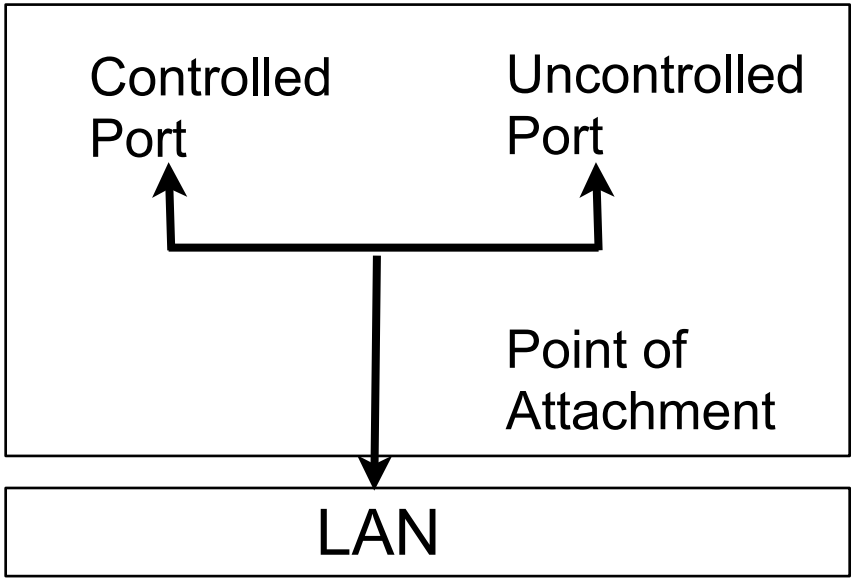
Bildquelle: Interpeak Network Security

IEEE 802.1X

- 802er Standards für Local Area Networks (LAN), insbesondere für Schicht 1 und 2, z.B.
  - 802.1Q Virtual Bridged LANs (VLAN)
  - 802.3 CSMA/CD (Ethernet)
  - 802.5 Token Ring
  - 802.6 Metropolitan Area Network
  - 802.11 Wireless LAN
  - 802.15 Wireless PAN (Personal Area Network)
  - 802.15.1 Bluetooth
- 802.1X Port Based Network Access Control
  - Authentisierung und Autorisierung in IEEE 802 Netzen
  - Häufig genutzt in WLANs und (V)LANs
  - Port-basierte Network Access Control

802.1X Grundlagen

- Rollen:
  - **Supplicant**: 802.1X Gerät, das sich authentisieren möchte
  - **Authenticator**: Gerät, an dem der Supplicant angebunden ist (z.B. Switch oder WLAN Access Point), erzwingt Authentisierung und beschränkt ggf. Konnektivität
  - **Authentication Server:** führt die eigentliche Authentisierung durch (z.B. RADIUS-Server mit LDAP-Backend)
  - **Port Access Entity (PAE)**: „Port“, an dem Supplicant angeschlossen ist
    - **Uncontrolled Port**: erlaubt Authentisierung des Gerätes
    - **Controlled Port**: erlaubt authentisiertem Gerät Kommunikation zum LAN

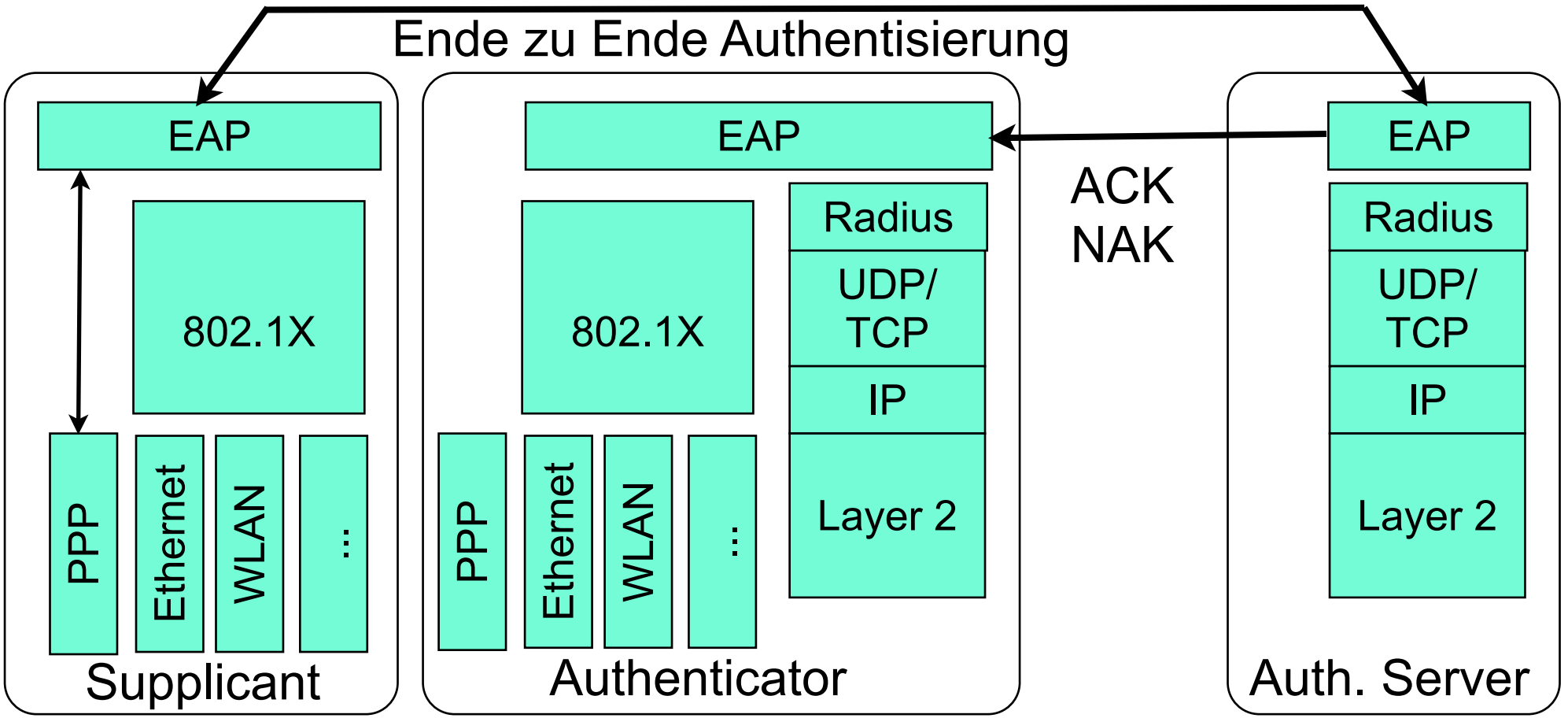


802.1X: Ablauf der Protokolle

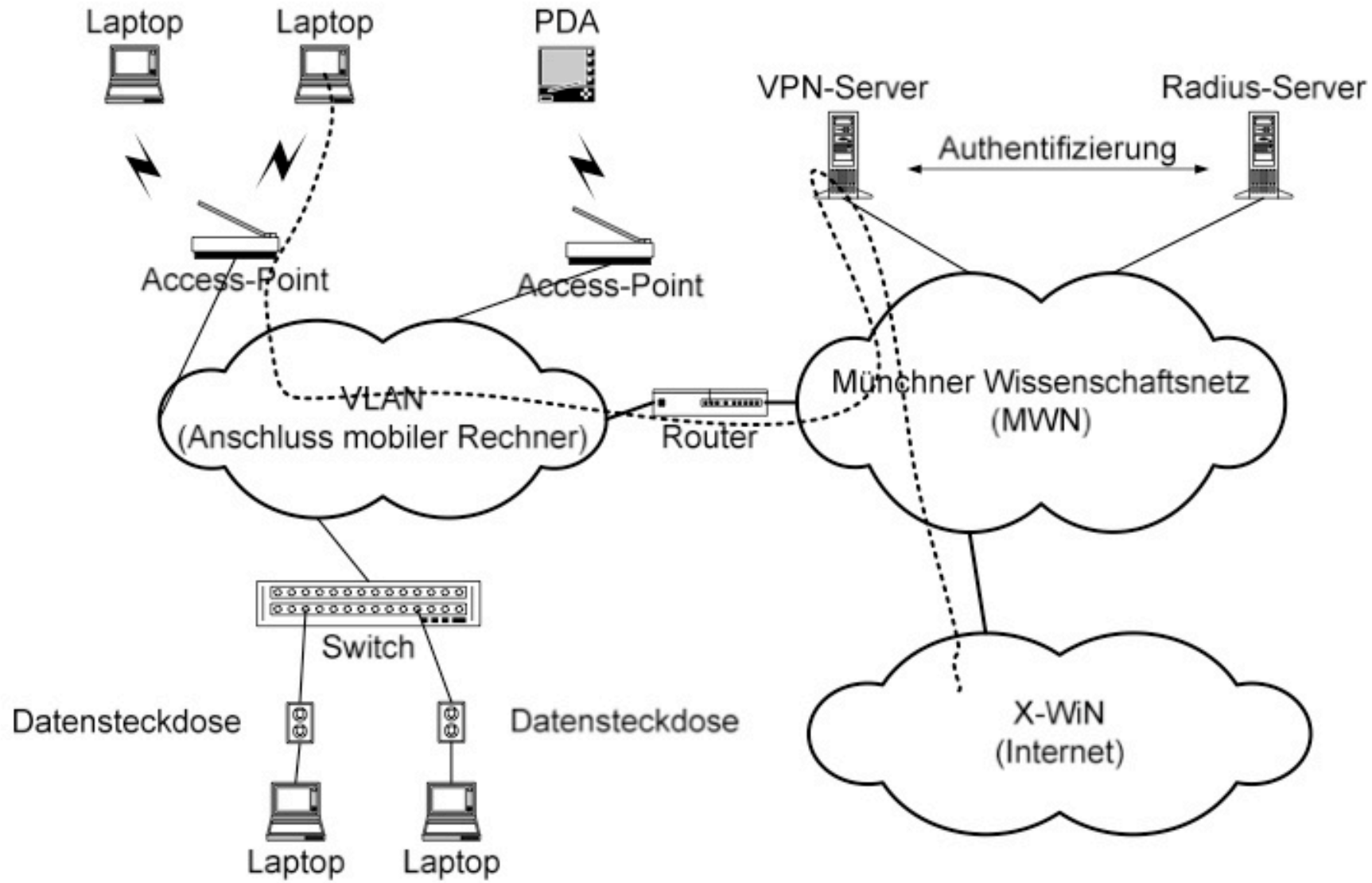
- Möglicher Ablauf:
  1. Supplicant fordert Controlled Port
  2. Authenticator fordert Authentisierung
  3. Nach erfolgreicher Authentisierung wird der Port freigeschaltet
- Supplicant oder Authenticator können Authentisierung initiieren
- 802.1X definiert keine eigenen Sicherheitsprotokolle, sondern nutzt bestehende:
  - Extensible Authentication Protocol (EAP) [RFC 3748] für Geräte-Authentisierung
  - EAP-TLS [RFC 5216] z.B. zur Aushandlung eines Session Key
  - RADIUS als AAA Protokoll (AAA = Authentisierung, Autorisierung und Accounting)

Extensible Authentication Protocol (EAP)

- Unterstützt verschiedene Auth.-Mechanismen
- Aushandlung erst während der Authentisierung mit Auth.-Server
- Authenticator ist nur Vermittler der Nachrichten

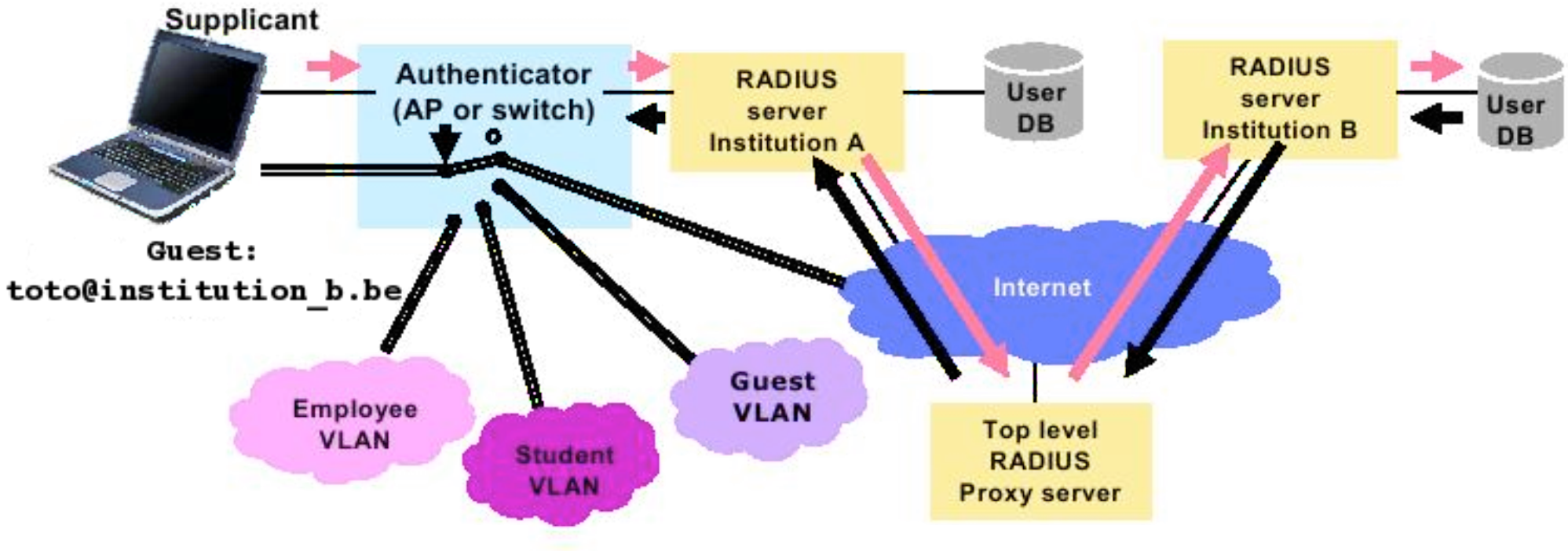


Beispiel: WLAN-Zugang im MWN



Beispiel: eduroam

- Weltweites Roaming in Hochschul-(WLAN-)Netzen
- 802.1X mit RADIUS-Authentifizierung an der jeweiligen Heimathochschule



Bildquelle: eduroam.be

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 10: Netzsicherheit - WLAN-Sicherheit (Schicht 2)

Inhalt

- WLAN: Eine kurze Einführung
- WLAN-Sicherheitsanforderungen und Mechanismen
- Wired Equivalent Privacy (WEP)
  - Authentisierung
  - Vertraulichkeit
  - Integrität
  - Autorisierung
  - Schwächen und Angriffe
- WiFi Protected Access (WPA)
  - Authentisierung mit 802.1X oder Preshared Keys (PSK)
  - Vertraulichkeit (TKIP)
  - TKIP-Schlüsselhierarchie
  - WPA- und TKIP-Sicherheit
- WPA 2

Wireless Local Area Network (WLAN)

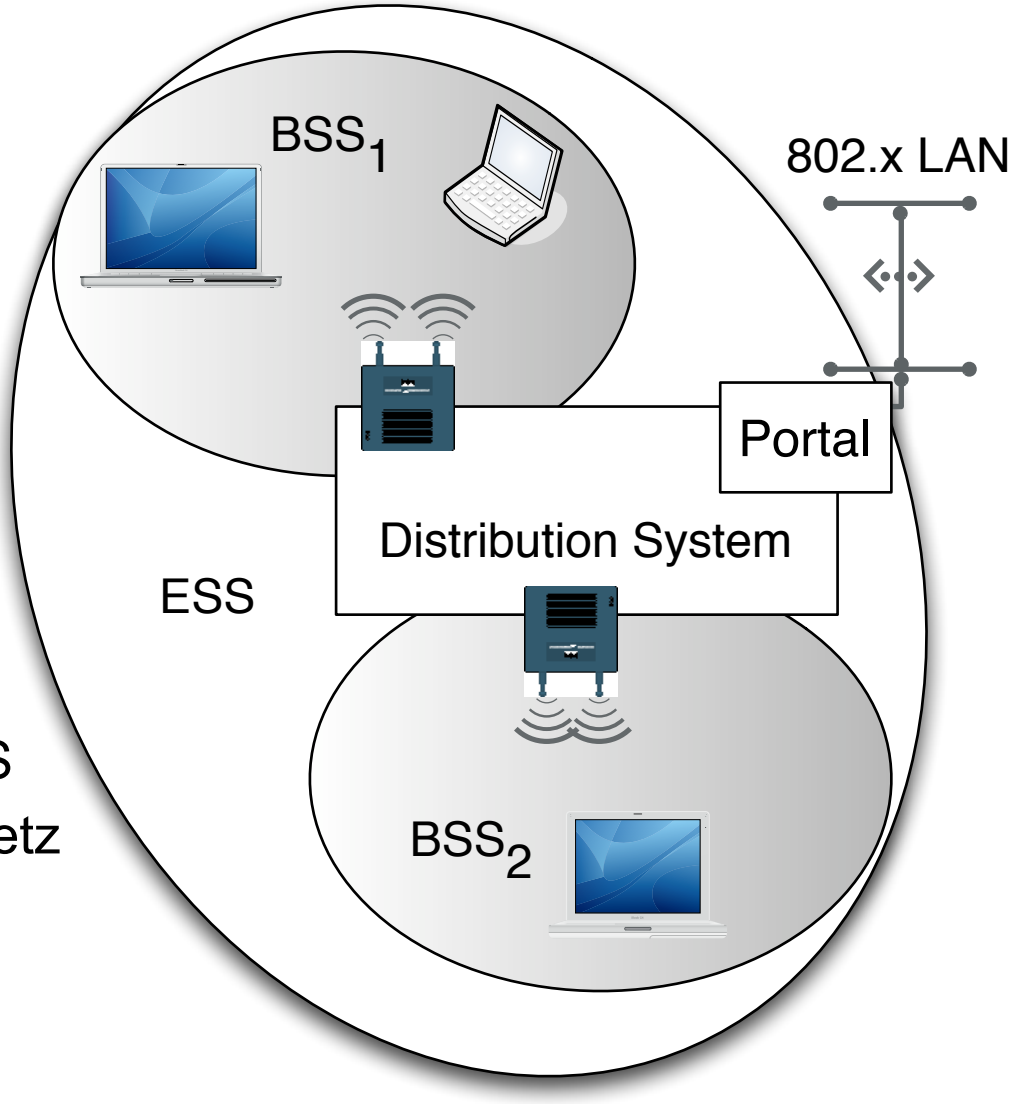
- WLAN standardisiert in IEEE 802.11x:

Standard	Frequenz [GHz]	maximaler Durchsatz [Mbit/s]
802.11	2,4	2
802.11a	5	54
802.11b	2,4	11
802.11g	2,4	54
802.11n (verabschiedet 09/2009)	2,4 5	600

- Alle Geräte teilen sich die Bandbreite
- Maximaler Durchsatz praktisch nicht erreichbar (netto wird i.d.R. weniger als die Hälfte erreicht, z.B. 120 Mbit/s bei 802.11n)

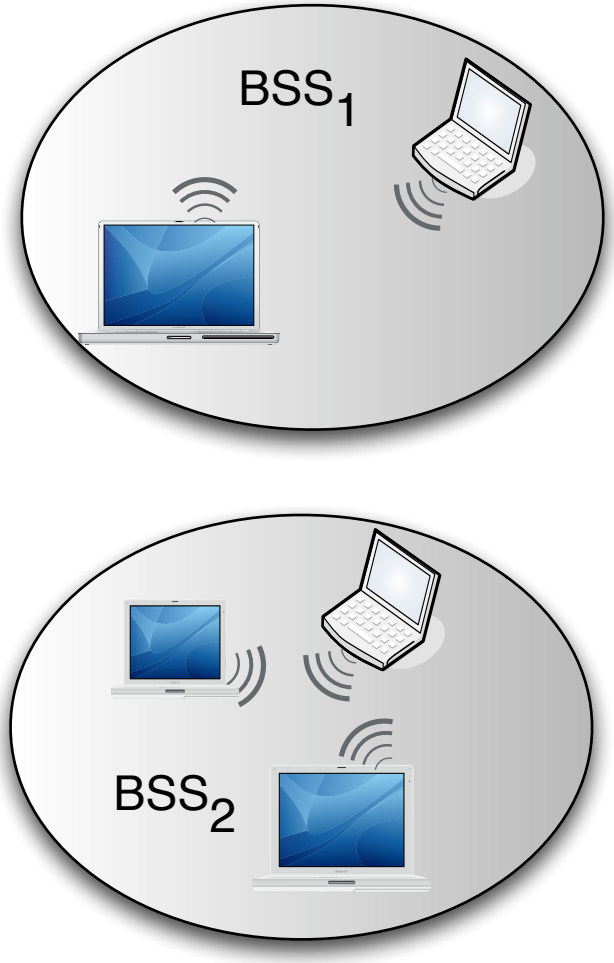
WLAN: Infrastruktur-Modus

- Access Point (AP): Zugangsknoten zum WLAN
- Station (STA)
  - Gerät mit WLAN-Ausstattung
  - (Intelligenter) Client
- Basic Service Set (BSS)
  - Gruppe von STAs, die selbe Frequenz nutzen
- Extended Service Set (ESS)
  - logisches Netz aus mehreren BSS
  - wird gebildet durch Verbindungsnetz (Distribution System (DSS))
  - ESS wird durch SSID identifiziert
- Portal: Verbindung zu anderen Netzen



WLAN: Ad-Hoc Modus

- Kein Access Point (AP) erforderlich
- Alle Stationen sind gleichberechtigt
- Basic Service Set (BSS)
  - Gruppe von STAs, die dieselbe Frequenz nutzen
  - Keine Kommunikation zwischen BSS möglich

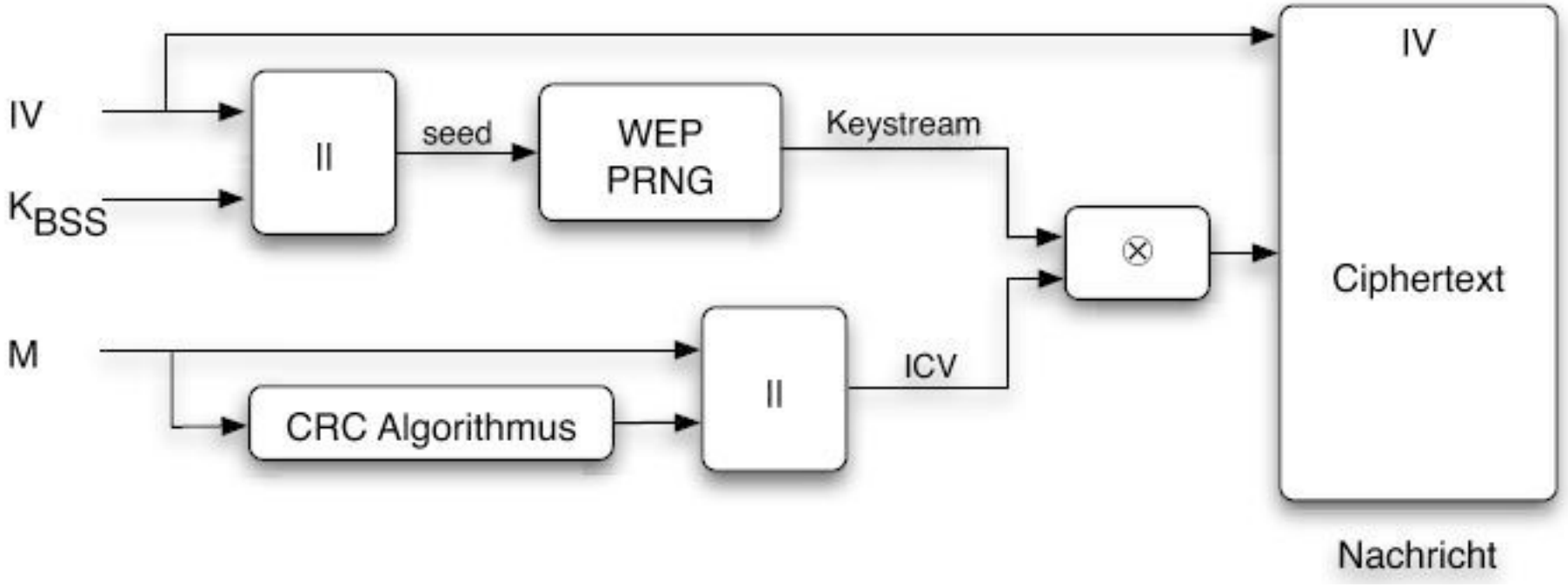


WLAN Sicherheitsmechanismen

- Mallet und Eve haben es im WLAN (wg. Funk) noch einfacher als in kabelgebundenen Netzen
- Sicherheitsanforderungen
  - Authentisierung
  - Zugangskontrolle zum Netz
  - Vertraulichkeit
  - Integrität
- Sicherheitsmechanismen
  - Wired Equivalent Privacy (WEP)
  - WiFi Protected Access (WPA)
  - WiFi Protected Access 2 (WPA2)
  - IEEE 802.11i (Standard, wegen Verspätung etablierte die Wi-Fi Alliance (Herstellerkonsortium) bereits WPA)
    - IEEE 802.11i D3.0 ist äquivalent zu WPA
    - IEEE 802.11i D9.0 ist äquivalent zu WPA2

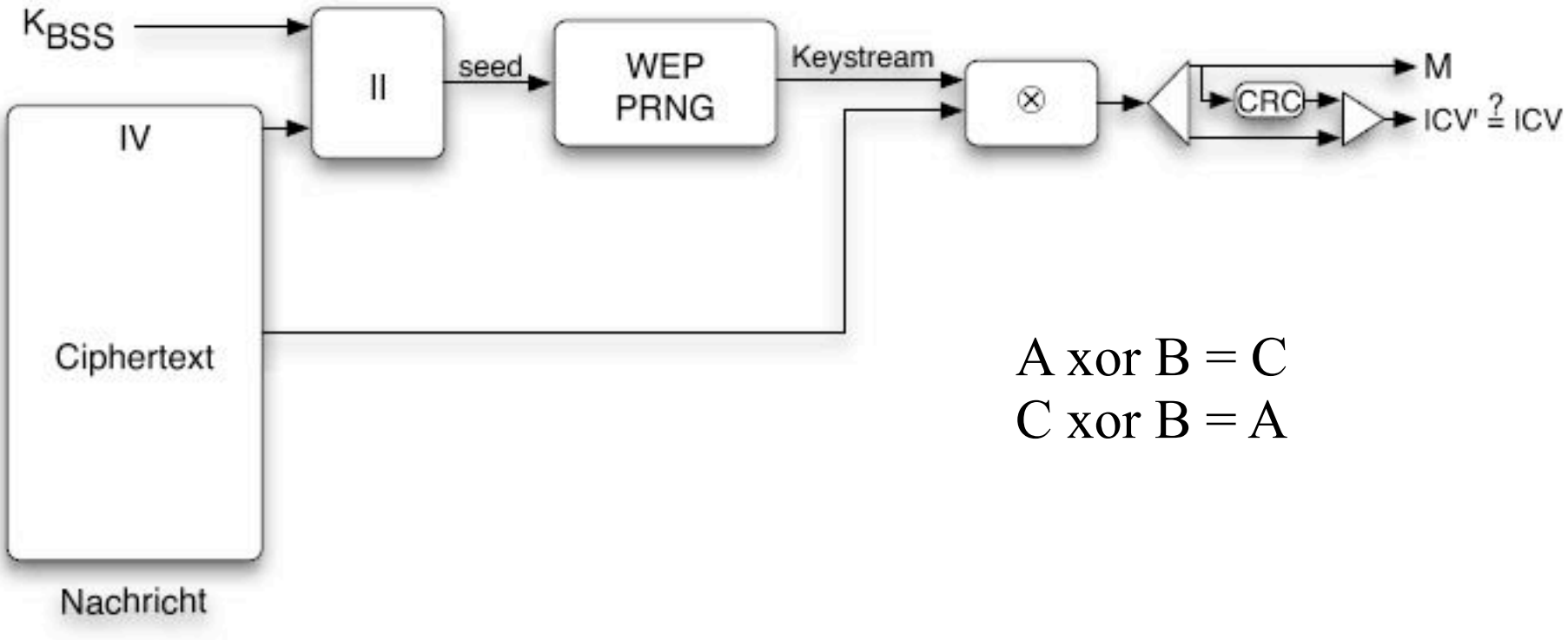
Vertraulichkeit: Wired Equivalent Privacy (WEP)

- Klartext wird mit Bitstrom XOR-verknüpft
- Bitstrom wird mit RC4 als Pseudozufallszahlengenerator (WEP PRNG) erzeugt
  - Für jede Nachricht 24-bit Initialisierungsvektor (IV) konkateniert mit 40-bit WEP-Schlüssel als 64-bit Seed für PRNG
  - Nachricht konkateniert mit CRC wird XOR verknüpft



WEP: Entschlüsselung

- IV wird im Klartext mit jedem Chiffretext übertragen
  - Jeder, der K_{BSS} kennt, kann Keystream erzeugen und Nachricht entschlüsseln
  - Selbstsynchronisierung von WEP
- Entschlüsselung ist inverser Vorgang zur Verschlüsselung



A xor B = C
C xor B = A

WEP: Integritätssicherung mit CRC-32

- Cyclic Redundancy Check (CRC) ist ein Fehlererkennungcode
- Entwickelt, um Übertragungsfehler in Netzen zu erkennen
- Mathematische Grundlagen:
  - Bit-String wird als Polynom mit Koeffizienten 0 und 1 aufgefasst
  - Nachricht M wird interpretiert als Polynom M(x)
  - Polynomberechnung modulo 2; d.h. Addition und Subtraktion identisch mit XOR
- Berechnung des CRC von M(x) zur Integritätssicherung:
  - Einigung auf Generatorpolynom G(x) (i.d.R. standardisiert)
  - Sei n der Grad von G(x), dann ist n+1 die Länge von G(x)
  - M(x) wird durch G(x) geteilt; M(x) mod G(x)
  - Teilungsrest ist CRC und wird mit M konkateniert
  - Empfänger berechnet Gesamtnachricht (M(x)|CRC) modulo G(x)
    = 0; Nachricht wurde nicht verändert (außer Änderung ist Vielfaches von G(x))
    ≠ 0; Nachricht wurde verändert

Anwendungen und Grenzen von CRC

- Einfach und billig in Hardware umzusetzen (32-bit Schieberegister)
- Gut geeignet für die Erkennung von „zufälligen“ Fehlern (z.B. bei Rauschen)
  - Ethernet
  - Festplatten-Datenübertragung
  - ...
- Aber: Keine kryptographische Hashfunktion!
  - Andere (sinnvolle) Nachrichten mit selbem CRC-Wert können einfach erzeugt werden
- Nur Fehlererkennung, keine Fehlerkorrektur möglich

WEP Authentisierung

- Open System Authentication
  - Entweder AP verschlüsselt nicht: Dann keine Authentifizierung, jeder kann AP nutzen
  - Oder bei aktivierter WEP-Verschlüsselung: Wer den Schlüssel kennt, kann Daten übertragen

- Shared Key Authentication
  - Challenge-Response-Protokoll
  - Basiert auf WEP-Verschlüsselung:

1. STA sendet Authentication Request an AP
2. AP sendet Challenge r im Klartext zurück
3. STA verschlüsselt r und sendet WEP(r) zurück
4. AP verifiziert

WEP Zugangskontrolle

- Bei Open System Authentication kann jeder senden
- Falls WEP aktiviert ist, kann nur senden, wer K_{BSS} kennt
- Viele APs bieten zusätzlich MAC-basierte Access Control Listen (ACLs)
  - Nur bekannte/freigeschaltete MAC Adressen dürfen senden
  - MAC kann einfach mitgelesen werden
  - MAC kann einfach gefälscht werden

WEP-Schwächen: Überblick

- WEP erfüllt KEINE der Sicherheitsanforderungen
- Vertraulichkeit:
  - Schlüsselmanagement und Schlüssel sind ein Problem
  - WEP ist einfach zu brechen
- Integrität
  - CRC kein geeignetes Verfahren zur Integritätssicherung bei absichtlicher Manipulation
- Authentisierung
  - basiert auf WEP
  - Fehler in der Umsetzung
- Zugriffskontrolle
  - Keine individuelle Authentifizierung, somit generell nur rudimentäre Zugriffskontrolle möglich

WEP Schwäche: Schlüsselmanagement

- Standard legt kein Schlüsselmanagement fest
- „Out-of-Band“ Schlüsselverteilung erforderlich
  - Manuelles Schlüsselmanagement oft fehlerbehaftet
  - Schlüssel werden sehr selten gewechselt
  - Oft wurde Open System Authentication ganz ohne Verschlüsselung aktiviert
- Schlüssellängen
  - WEP-40; 40 Bit Schlüssel (wegen Exportrestriktionen)
  - WEP-104; 104 Bit Schlüssel
    - Vom Benutzer z.B. in Form von 26 Hexziffern einzugeben
    - Somit mühsam/fehleranfällig und deshalb häufig sehr einfach gewählt
  - Aber selbst mit ausreichend langen Schlüsseln wäre WEP nicht sicher

WEP Schwäche: Verschlüsselung

- RC4 ist Stromchiffre, d.h. der selbe Schlüssel sollte nie wiederholt werden
  - IV soll dies verhindern
  - IV wird im Klartext übertragen
  - 24 Bit für den IV sind deutlich zu kurz
- Wiederverwendung des Keystream (bei gleichem IV)
  - Zwei Klartextnachrichten M_1 und M_2 mit $P_i = (M_i|CRC_i)$
  - $C_1 = P_1 \oplus RC4(IV_1, K_{BSS})$
  - $C_2 = P_2 \oplus RC4(IV_1, K_{BSS})$
  - dann gilt
  - $C_1 \oplus C_2 = (P_1 \oplus RC4(IV_1, K_{BSS})) \oplus (P_2 \oplus RC4(IV_1, K_{BSS})) = P_1 \oplus P_2$
  - d.h. falls Angreifer M_1 und C_1 kennt, kann er P_2 (somit M_2) aus dem mitgehörten C_2 berechnen, ohne K_{BSS} zu kennen (Known-Plaintext Angriff)
  - Known-Plaintext ist einfach zu erzeugen (Daten von außen schicken)

WEP Schwäche: Traffic Injection

- Known-Plaintext Angriff: Mallet kennt M und C:
  $C = RC4(IV,K_{BSS}) \oplus (M,CRC(M))$
- Damit kann Mallet den Key Stream berechnen:
  $RC4(IV,K_{BSS}) = C \oplus (M, CRC(M))$
- Absichtliche Wiederverwendung alter IVs möglich:
  Mallet berechnet
  $C' = RC4(IV, K_{BSS}) \oplus (M',CRC(M'))$
  und schickt (IV, C´) an Bob
- Bob hält dies für ein gültiges Paket

- Wissen über verwendete höherliegende Protokolle erleichtert rein passiven Known-Plaintext Angriff
  - Protokoll-Header, Adressen, Protokollprimitive sind Teile von M

WEP Schwäche: Integritätssicherung

- CRC und RC4 sind linear
- Mallet fängt Nachricht von Alice an Bob ab: (IV, C) mit
  $C = RC4(IV, K_{BSS}) \oplus (M, CRC(M))$
- Mallet verfälscht die Nachricht M zu Nachricht X
  - Mallet wählt beliebige Nachricht M´ mit derselben Länge
  - Mallet sendet Ciphertext $C' = C \oplus (M', CRC(M')) =$
    $RC4(IV, K_{BSS}) \oplus (M,CRC(M)) \oplus (M',CRC(M')) =$
    $RC4(IV, K_{BSS}) \oplus (M \oplus M', CRC(M) \oplus CRC(M')) =$
    $RC4(IV, K_{BSS}) \oplus (M \oplus M', CRC(M \oplus M')) =$
    $RC4(IV, K_{BSS}) \oplus (X, CRC(X))$
  - Mallet kennt Inhalt von X nicht, da er M nicht kennt
  - ABER: Eine „1“ an Position n in M´ führt zu gekipptem Bit an Position n in X; Mallet kann kontrollierte Änderungen in M durchführen. Beispiel: Zieladresse von IP-Paketen ändern

Weakness in Key Scheduling of RC4

- Papier von Fluhrer, Mantin und Shamir; 2001:
  - Grosse Zahl unsicherer Schlüssel wurden identifiziert, kleine Zahl von Bits reicht, um die meisten Output-Bits zu berechnen
  - Schwäche: IV wird mit K_{BSS} konkateniert; IV im Klartext übertragen
  - K_{BSS} bleibt relativ lange konstant, IV wechselt
  - Passive Ciphertext-Only Attack:
    - Eve muss 4 bis 6 Millionen Pakete mithören
    - Dies dauert nur wenige Minuten (ggf. Traffic stimulieren)
    - Abhängigkeit von der Schlüssellänge (40 oder 104 Bit) ist nur linear
- Klein zeigt 2005, dass es stärkere Korrelationen zwischen Keystream und Schlüssel gibt und verbessert den Angriff aus 2001

Breaking 104-bit WEP in less than 60 seconds

- Artikel von Tews, Weinmann, Pyshkin, Uni Darmstadt, 2007
- Aktiver Angriff
- Nutzt ARP-Request- und ARP-Reply-Pakete
  - Feste Länge der Pakete
  - Über Länge der Frames sind die verschlüsselten ARP Pakete erkennbar
  - Die ersten 16 Byte des ARP Paketes sind vorhersagbar
    - 8 Byte LLC Header (AA AA 03 00 00 00 08 06) gefolgt von
    - 8 Byte ARP Header:
      - 00 01 08 00 06 04 00 01 für ARP Request
      - 00 01 08 00 06 04 00 02 für ARP Response
  - XOR Verknüpfung abgehörter Pakete mit dieser Bytefolge liefert die ersten 16 Byte des Keystream
  - Wiedereinspielen abgehörter ARP Requests beschleunigt den Angriff
  - Erfolgsrate bei nur 40.000 Frames schon > 50 %
  - Erfolgsrate bei 85.000 Frames rund 95 %

Schlussfolgerung

- WEP ist **NICHT** sicher
- WEP sollte **NICHT** verwendet werden
- Das 2008er Update des Data Security Standards (DSS) der Payment Card Industry (PCI) verbietet die Nutzung von WEP im Rahmen jeglicher Kreditkarten-Datenverarbeitung ab Juli 2010

WiFi Protected Access

- WPA zur Verbesserung der Sicherheit eingeführt
- WEP-Hardware sollte weiter benutzbar bleiben
- Vertraulichkeit:
  - Temporal Key Integrity Protocol (TKIP)
  - Rekeying-Mechanismus zum automatischen Wechseln der Schlüssel
  - Hierarchie von Schlüsseln
- Integritätssicherung
  - TKIP Message Integrity - MIC (genannt „Michael“);
    zur Unterscheidung von MAC (Media Access Control)
  - Mit Schlüssel parametrisierte kryptographische Hash-Funktion
  - Verbessert ungeeigneten CRC-Mechanismus von WEP
- Authentisierung
  - Nach wie vor Möglichkeit für Pre-Shared Key (PSK)
  - Bietet aber auch 802.1X (insb. in großen IT-Infrastrukturen genutzt)

Temporal Key Integrity Protocol (TKIP)

- TKIP verwendet Schlüsselhierarchie, um kurzlebige Schlüssel zu erzeugen
- Drei Hierarchiestufen:

1. Temporäre Schlüssel (Temporal Key, TK)
   - In jede Richtung eigene Schlüssel
     - Zur Verschlüsselung (128 Bit)
     - Zur Integritätssicherung (64 Bit)
   - Erneuerung des Schlüsselmaterials durch `rekey key` Nachricht
   - `rekey key` Nachricht enthält Material, damit STA und AP neue Sitzungsschlüssel ableiten können; Nachricht verschlüsselt mit
2. Pairwise Transient Key (PTK)
   - Sichern die Übertragung temporärer Schlüssel
   - 1 Schlüssel zur Sicherung des Schlüsselmaterials
   - 1 Schlüssel zur Sicherung der `rekey key` Nachricht

TKIP Schlüsselhierarchie

3. Pairwise Master Key (PMK)
   - Höchster Schlüssel innerhalb der Hierarchie
   - Erzeugt vom 802.1X Authentication Server und vom AP an STA weitergereicht
   - Falls 802.1X Setup „zu komplex“; Preshared Keys möglich (d.h. in der Praxis: Passwörter)
   - Master Key wird zur Sicherung der key-encryption Keys genutzt
   - Damit Aufbau einer Sitzungsstruktur möglich; von der Authentisierung über 802.1X bis
     - Widerruf des Schlüssels
     - Ablauf des Schlüssels
     - STA verliert Kontakt zum AP
   - Klar: Kompromittierung des Master Key führt zur Kompromittierung der gesamten Hierarchie!

TKIP Schlüsselhierarchie Zusammenfassung

- Aus IEEE 802.11i-2004 (geht über reines TKIP hinaus)
- hier Verwendung von 802.1X

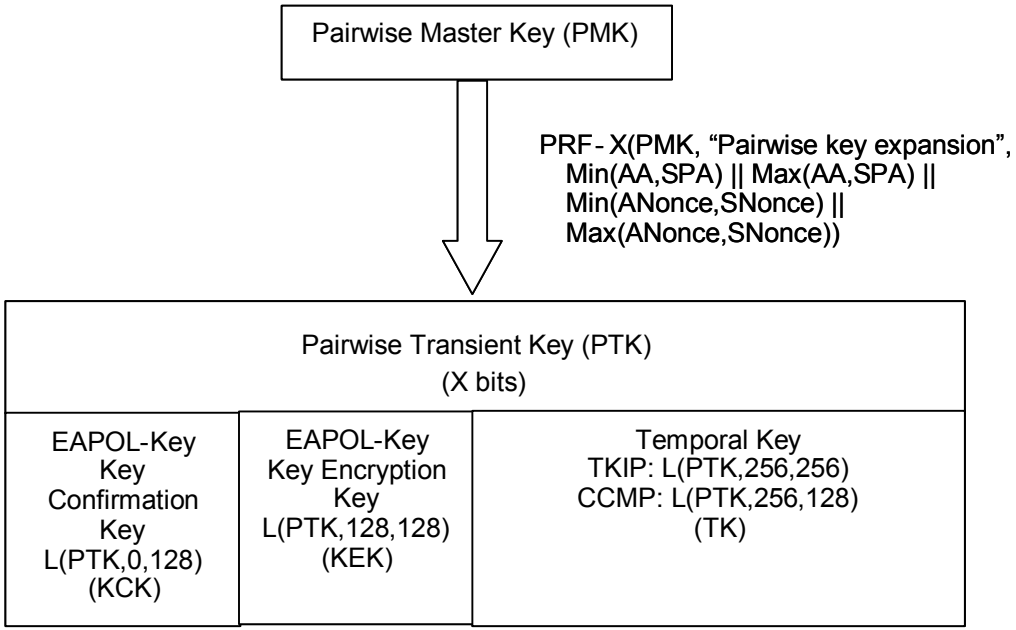


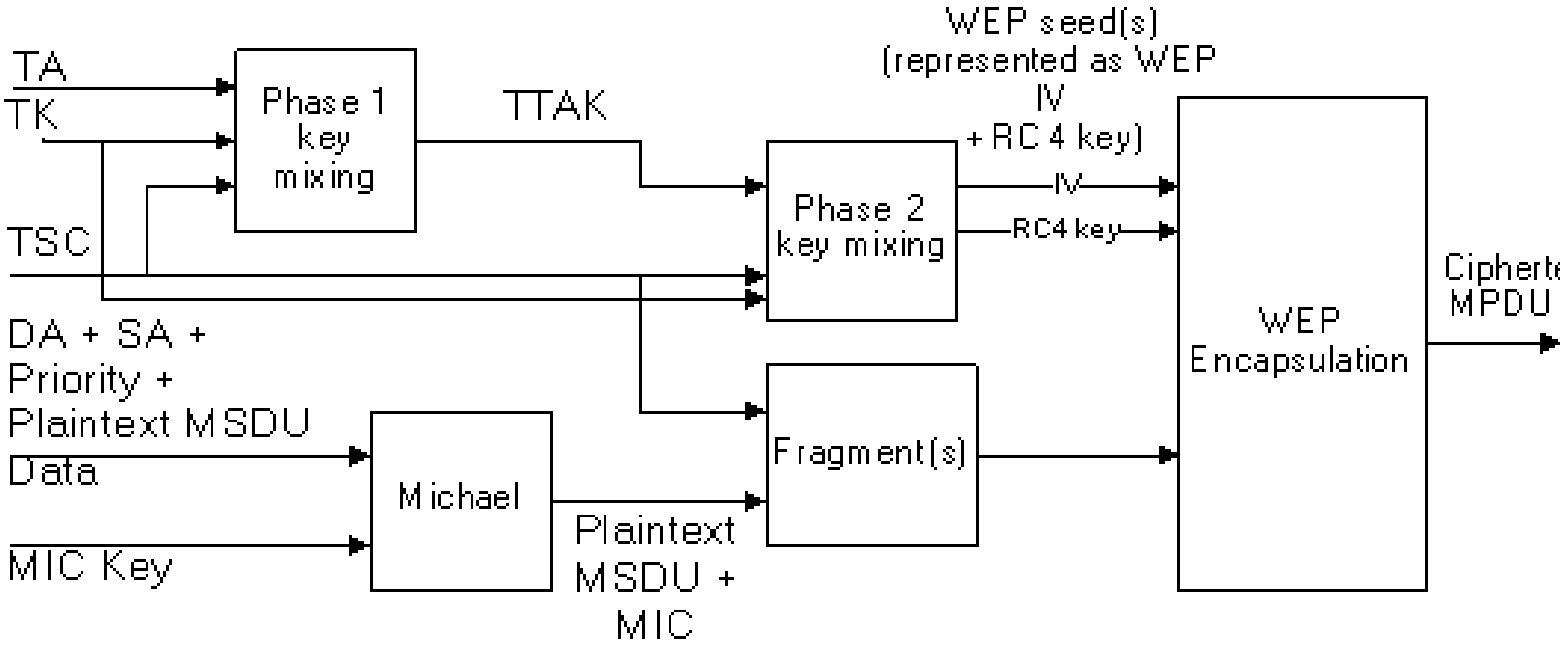
Figure 43s—Pairwise key hierarchy

- **PRF** Pseudo Random Function
- **AA** Authenticator Address
- **SPA** Supplicant Address
- **EAPOL** EAP over LAN
- **KCK** Key Confirmation Key (Integritätssicherung)
- **KEK** Key Encryption Key
- **L(x,0,128)** Teilstring ab Bit 0 mit Länge von 128

CCMP ist Bestandteil von WPA2 (später)

TKIP Verschlüsselung: Blockdiagramm

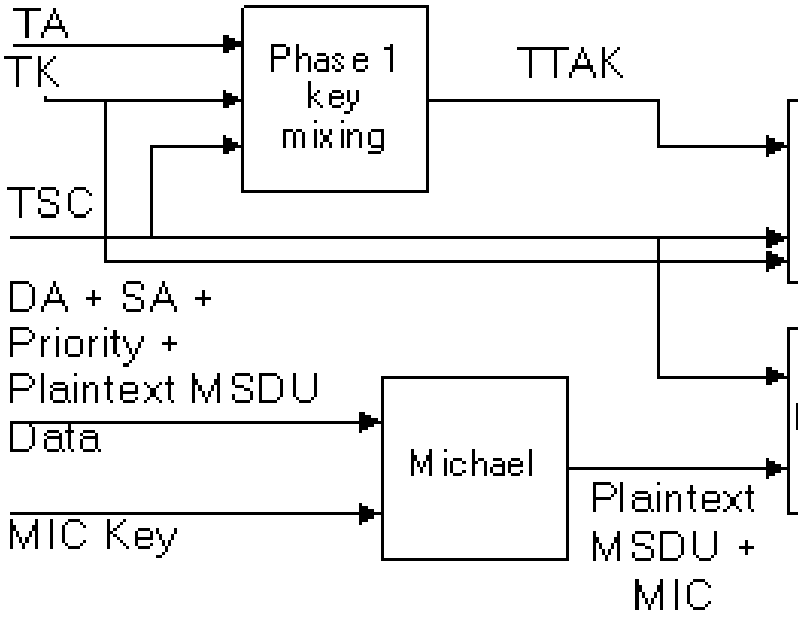
- Aus IEEE 802.1i-2004



- TA Transmitter Address
- TK Temporal Key
- TSC TKIP Sequence Counter
- DA Destination Address
- SA Source Address
- MSDU MAC Service Data Unit
- MPDU Message Protocol Data Unit
- TTAK TKIP Mixed Address and Key
- MIC Message Integrity Code

TKIP Verschlüsselung

- Aus IEEE 802.1i-2004

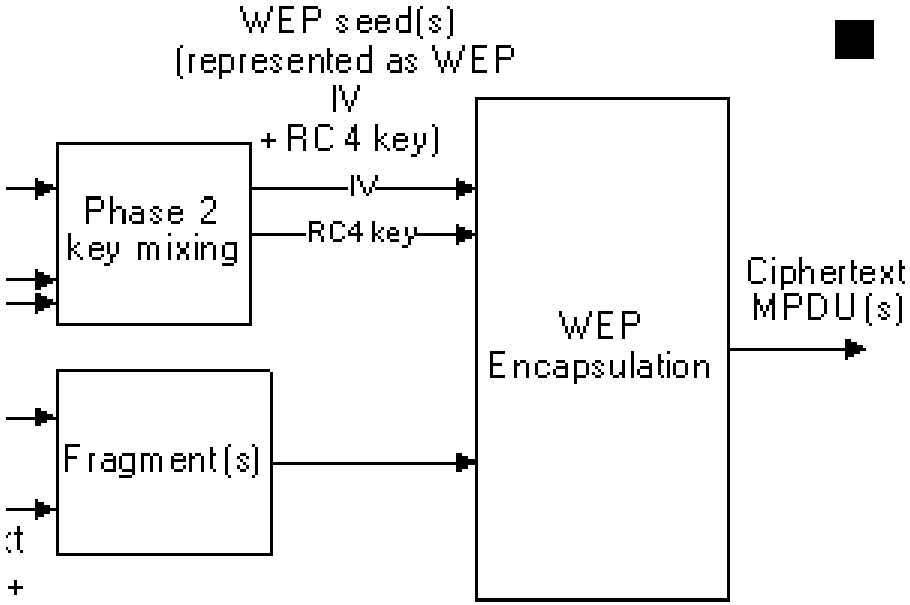


- TA Transmitter Address
- TK Temporal Key
- TSC TKIP Sequence Counter
- DA Destination Address
- SA Source Address

- Kein wirklich neues Verfahren; soll nur Schwächen beseitigen
- Phase 1 Key Mixing
  - TTAK = Phase1(TA,TK, TSC)
  - Phase1 ist nichtlineare Funktion mit XOR-Operationen, bitweiser UND-Operation sowie einer Verkürzungsfunktion
  - TA verhindert, dass zwei STAs denselben Schlüssel erhalten
  - TSC als Sequenznummer für MPDUs

TKIP Verschlüsselung: Phase 2

- Aus [IEEE 802.1i-2004]

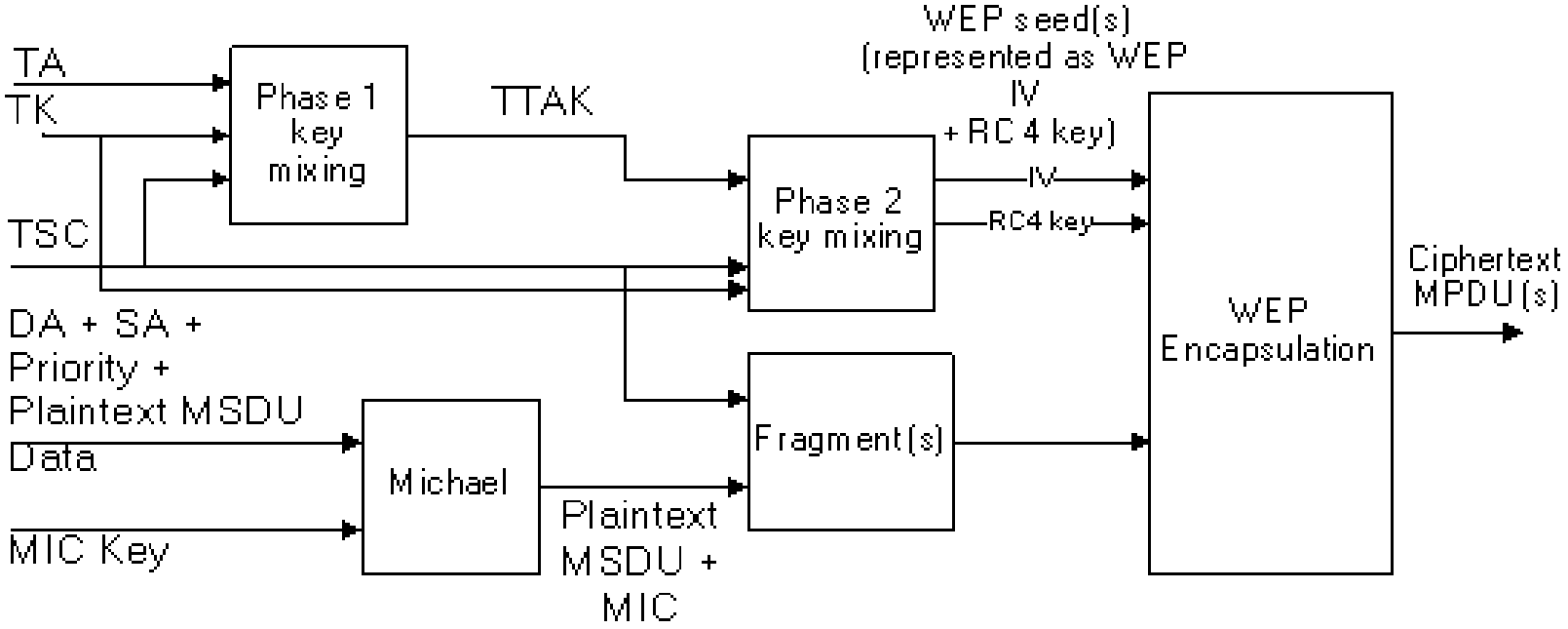


- **TTAK** TKIP Mixed Address and Key

- Phase 2 Key Mixing
  - TTAK = Phase1(TA, TK, TSC)
  - Phase2(TTAK, TK, TSC)
  - Phase2 ist Feistel-Chiffre:
  - Einfache Operationen für „schwache“ AP-Hardware
    - XOR, UND, ODER, >>
    - S-Box
  - Erzeugt 128 Bit WEP Schlüssel
    - 24 Bit Initialisierungsvektor
    - 104 Bit RC4 Schlüssel

TKIP Verschlüsselung: Zusammenfassung

- Aus IEEE 802.1i-2004



- Für jedes Frame (MSDU) wird eigener Schlüssel generiert
- Hardware-Abwärtskompatibilität; d.h. Verwendung von RC4 nach wie vor problematisch

WPA und TKIP: Sicherheit

- Bei Verwendung von Pre Shared Keys (PSK) hängt die Sicherheit stark von der Stärke des Passworts ab
- Angriff mit Rainbow-Tables (seit 2004)
- Angriff auf PRF Funktion der Schlüsselverteilung (August 2008)
  - nutzt GPUs (Graphics Processing Units) anstatt CPUs
  - Entwickelt auf NVIDIA-CUDA (Compute Unified Device Architecture)
    - Compiler und Entwicklungsumgebung
    - nativer Zugriff auf GPUs auf Grafikkarten
    - dadurch massive Parallelisierung möglich
    - damit Speedup von Faktor 30 und mehr möglich
    - Zeit für „Raten“ eines Passwortes reduziert sich auf 2-3 Tage
- Angriff auf TKIP Verschlüsselung (November 2008)
  - Entschlüsselung von Paketen mit teilweise bekanntem Inhalt ohne Kenntnis des Schlüssels möglich
  - Schlüssel ist damit nicht zu brechen

CERN School of Computing

Multi-core architectures – NVIDIA G80

- **128 stream processors**
- **330 GFlops (today's general purpose CPUs have ~10)**
- **150W**
- **Top of the line graphics hardware (along with the G92)**

damals

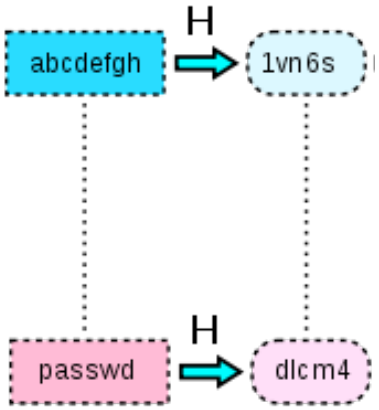


Einschub: Rainbow Tables

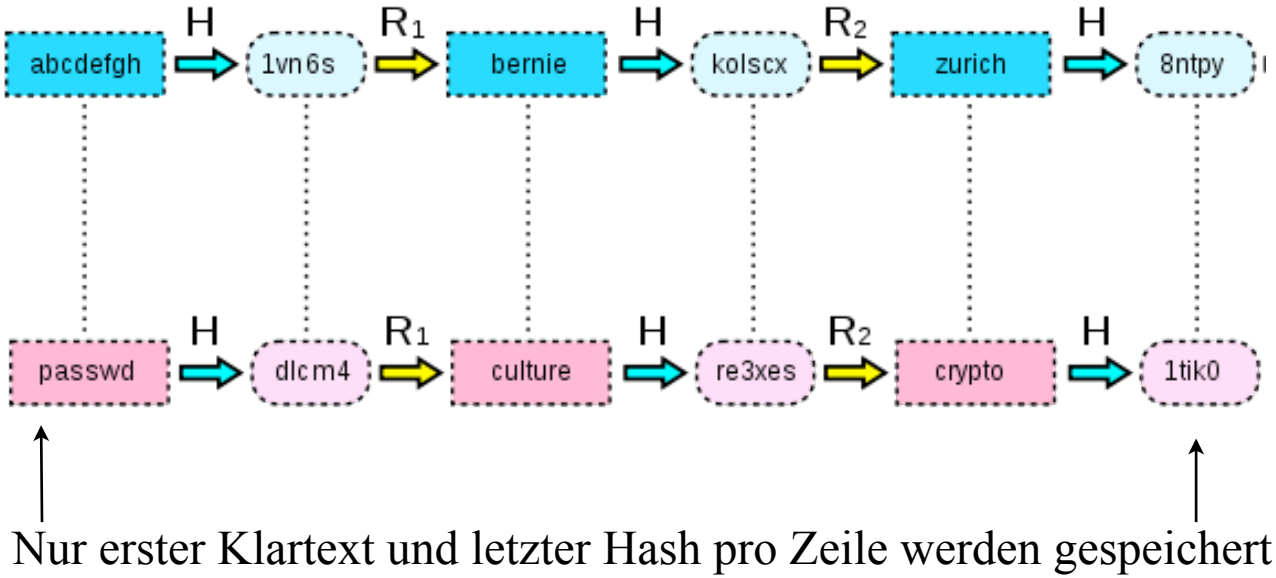
- Bei allen Krypto-Angriffen ist Rechenzeit- und Speicherplatzkomplexität zu betrachten
- Rainbow-Tables versuchen, optimalen time-memory tradeoff zu nutzen, um vollständigen Brute-Force-Angriff zu sparen
- Idee: Optimale Speicherung einer Klartext-zu-Hash Tabelle
- Kompakte Speicherung von sog. Chains (Ketten/PW-Sequenzen)
  - Kette startet mit initialem Klartext-Wort, dieses wird gehasht
  - resultierender Hash wird Reduktionsfunktion unterworfen
  - Reduktionsfunktion liefert weiteres potentielles Klartext-Wort
  - Dieser Vorgang wird n-mal wiederholt
  - relevant sind nur erstes Klartext-Wort und letzter Hash-Wert
  - Vorgang wird einmal für alle Wörter eines Wörterbuchs wiederholt
  - Kollisionen vermeiden: internes Klartext-Wort darf nicht Startwert einer anderen Kette sein
  - Rainbow Table speichert alle resultierenden Ketten (1. Klartext : letzter Hash)

Einschub: Rainbow Tables; Beispiele

- Trivialfall: Nur 1 Iteration
  - Speichert zu jedem Klartext seine Hashsumme
  - Rainbow-Tabelle wird sehr lang und damit zu groß



- 3 Iterationen:



Einschub: Rainbow Tables; Anwendung

- Rainbow-Tabelle mit w Einträgen und Ketten der Länge n
- MD5 Hash: `bca6a2aed3edc8e22f68ed65e39682c6` („IT-Sec“)
- Suche in Tabelle auf rechter Seite. Fallunterscheidung:

1. Hash-Wert gefunden, steht z.B. in Zeile 17
   - Kette aus Zeile 17 komplett durchlaufen Reduktionsfunktion n-1 liefert den gesuchten Klartext
2. Hash-Wert steht nicht in Rainbow-Table
   - Reduktion des Hashes (vereinfachtes Bsp. erste 6 Zeichen): `bca6a2`
   - MD5(`bca6a2`) liefert `3c41c8c8c5d27647d3f64937a801c90a`
   - Suche diesen Hash in Tabelle

- In der Praxis werden verschiedene Reduktionsfunktionen kombiniert
  - Ziel: Kollisionen / Wiederholungen vermeiden, um möglichst viele Klartexte abzudecken

Angriff auf TKIP Verschlüsselung

- Beck, TU-Dresden, Tewes, TU-Darmstadt; publ. 08.11.2008
- Erstes Verfahren, das keine Pre Shared Keys voraussetzt
- Basiert auf chop-chop Angriff (bekannt seit 2005)
- Funktionsweise:
  - Angreifer schneidet Verkehr mit, bis er verschlüsseltes ARP Packet findet (vgl. Folien „Breaking WEP in less than 60 seconds“)
  - letztes Byte wird entfernt
  - Annahme: Byte war 0; mit XOR-Verknüpfung mit bestimmten Wert wird versucht gültige Checksumme zu erzeugen
  - Paket wird an AP gesendet:
    - Inkorrekt: Paket wird verworfen
    - Korrekt: Client erzeugt MIC Failure Report Frame; Angreifer muss dann vor nächstem Versuch 60 Sekunden warten, sonst erzwungener Verbindungsabbau
  - Worst Case: 256 Tests für 1 Byte erforderlich. Praktisch: In 12 Minuten mindestens 12 Byte entschlüsselbar

Beck, Tewes Angriff (Forts.)

- Sicherheitsmaßnahmen von WPA
  - Anti-chopchop: zwei falsche MICs in 1 Minute ➪ Verbindungsabbau
  - TSC verhindert Wiedereinspielen
- Gegenmaßnahmen:
  - 60 Sekunden warten (vgl. Folie vorher)
  - Replay nicht an verwendeten, sondern an anderen Sendekanal
- Entschlüsselung des ARP Paketes ermöglicht:
  - Schlüsselstrom vom AP zu STA und MIC Code können ermittelt werden
  - Eigene verschlüsselte Pakete können an STA gesendet werden; z.B. zum Manipulieren von ARP-Paketen
- Grenzen des Angriffs
  - Rekeying Intervall muss ausreichend groß sein
  - QoS muss aktiviert sein, sonst stehen keine 8 Kanäle zur Verfügung
  - nur eine Richtung: AP zu STA

WPA 2

- Empfehlung: Verwendung von WPA 2 anstelle von WPA
- Änderungen:
  - AES ersetzt verpflichtend RC4
  - CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol) als Ersatz für TKIP
- Verfahren gilt derzeit als sicher
  - Verpflichtend für Geräte mit Wi-Fi Logo
- Aber: Verschlüsselung schützt nicht ewig
  - Mitgehörte Daten können evtl. später entschlüsselt werden

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 11: Netzsicherheit - Schicht 3: Network Layer - IPSec

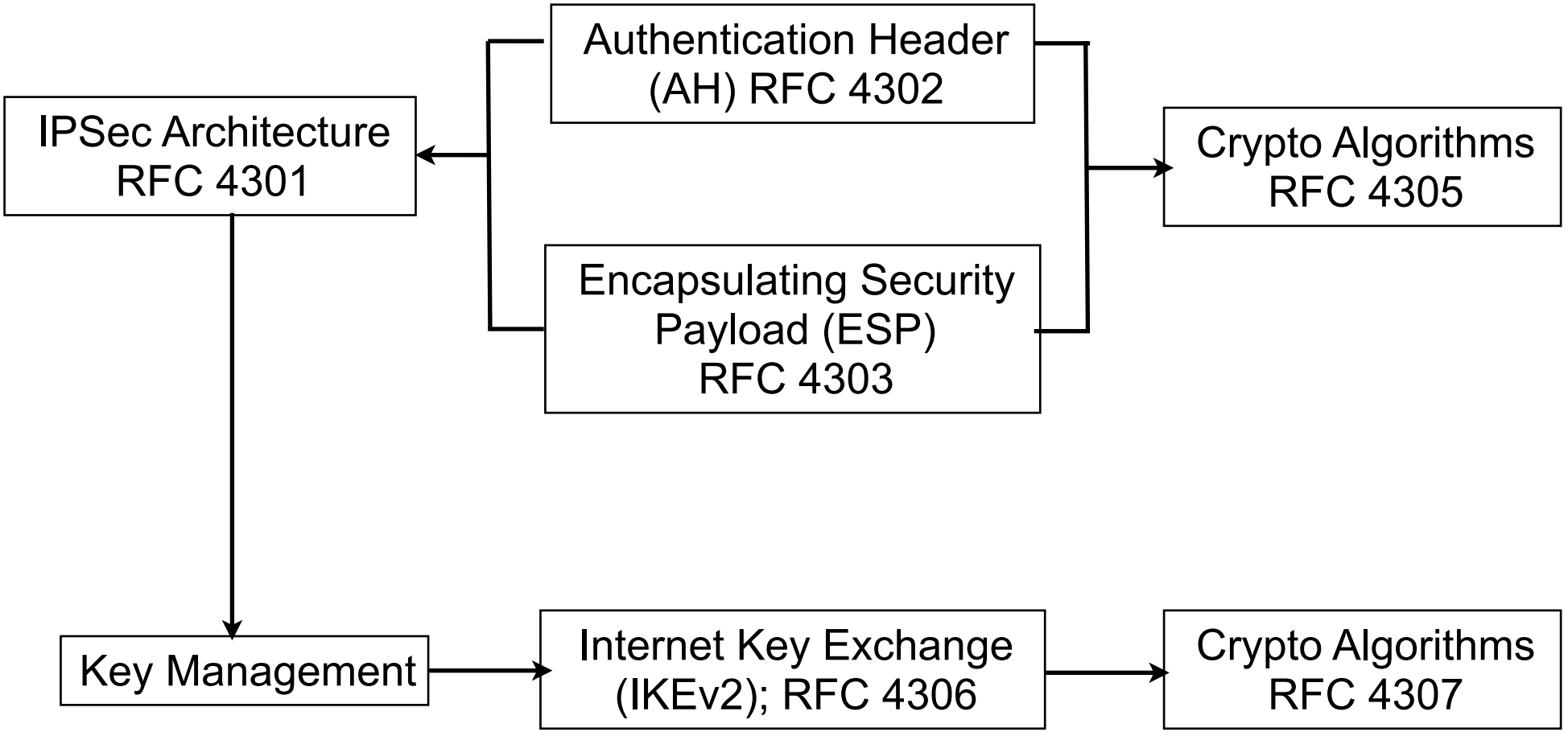
Inhalt

- Schwächen des Internet-Protokolls (IP)
- IPSec: Sicherheitserweiterung des IP-Protokolls
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)
  - Anwendungsbeispiele
- Schlüsselverteilung mit IKEv2 (Internet Key Exchange)
  - Aufbau einer IKE SA
  - Authentisierung der Partner
  - Aufbau der IPSec SA
  - Erzeugung von Schlüsselmaterial

IP: Gefahren und Schwächen

- Vertraulichkeit:
  - Mithören einfach möglich
  - Man-in-the-middle Attack
  - Verkehrsfluss-Analyse
- Integrität:
  - Veränderung der Daten
  - Session Hijacking
  - Replay Angriffe
- Authentisierung:
  - IP Spoofing
- Lösung: IPSec (Sicherheitserweiterungen für IP)
  - Integraler Bestandteil von IPv6
  - Als Erweiterungs-Header auch für IPv4 einsetzbar
  - Motivation: Erspart den Aufwand für entsprechende Gegenmaßnahmen in jeder einzelnen Anwendung (d.h. auf höheren Schichten)

IPSec-relevante Spezifikationen



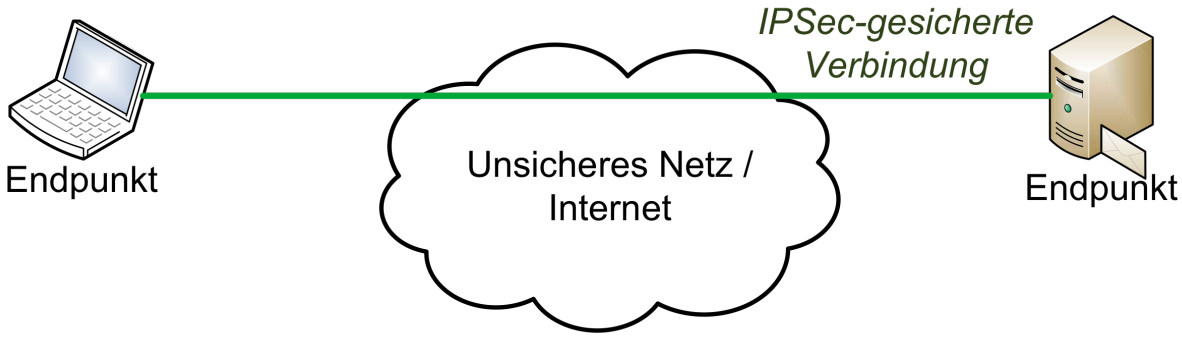
IPSec Überblick

- IP Authentication Header (AH)
  - Integrität des verbindungslosen Verkehrs
  - Authentisierung des Datenursprungs (genauer: des IP-Headers)
  - Optional: Anti-Replay Dienst
- IP Encapsulating Security Payload (ESP)
  - Vertraulichkeit (eingeschränkt auch für den Verkehrsfluss)
  - Integrität
  - Authentisierung (der sog. Security Association)
  - Anti-Replay Dienst
- Jeweils zwei verschiedene Betriebsmodi:
  - Transport Mode
  - Tunnel Mode

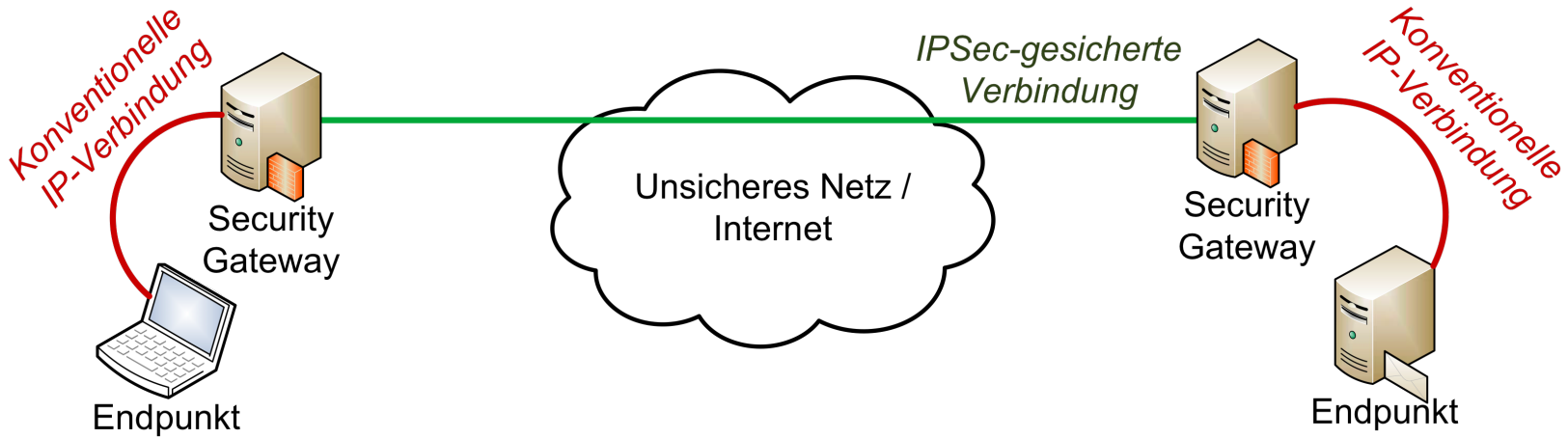
IPSec: Transport Mode / Tunnel Mode

- In beiden Modi können AH und/oder ESP eingesetzt werden

Transport Mode

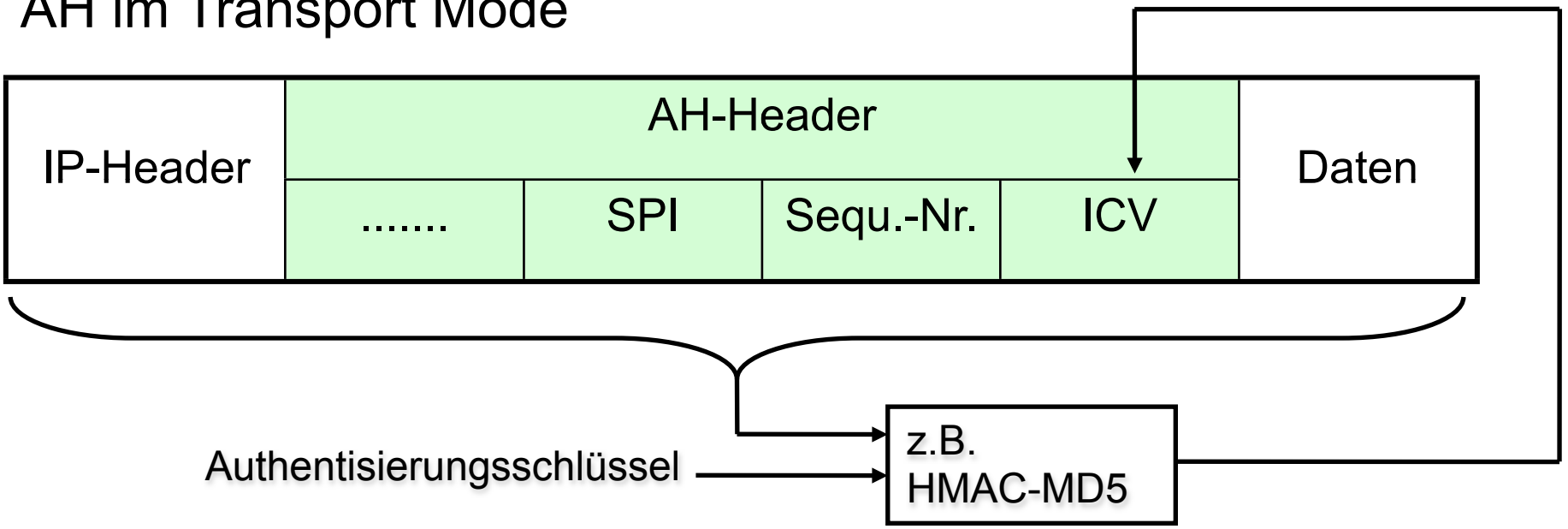


Tunnel Mode



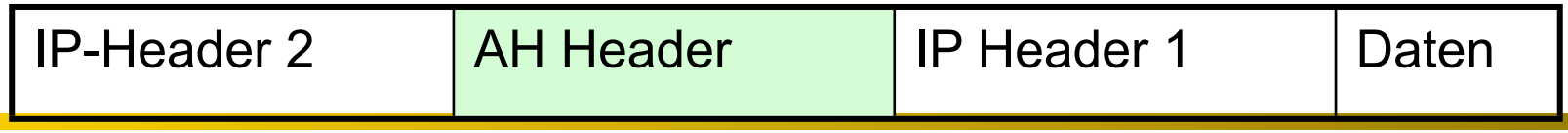
Authentication Header (AH) - Überblick

- AH im Transport Mode

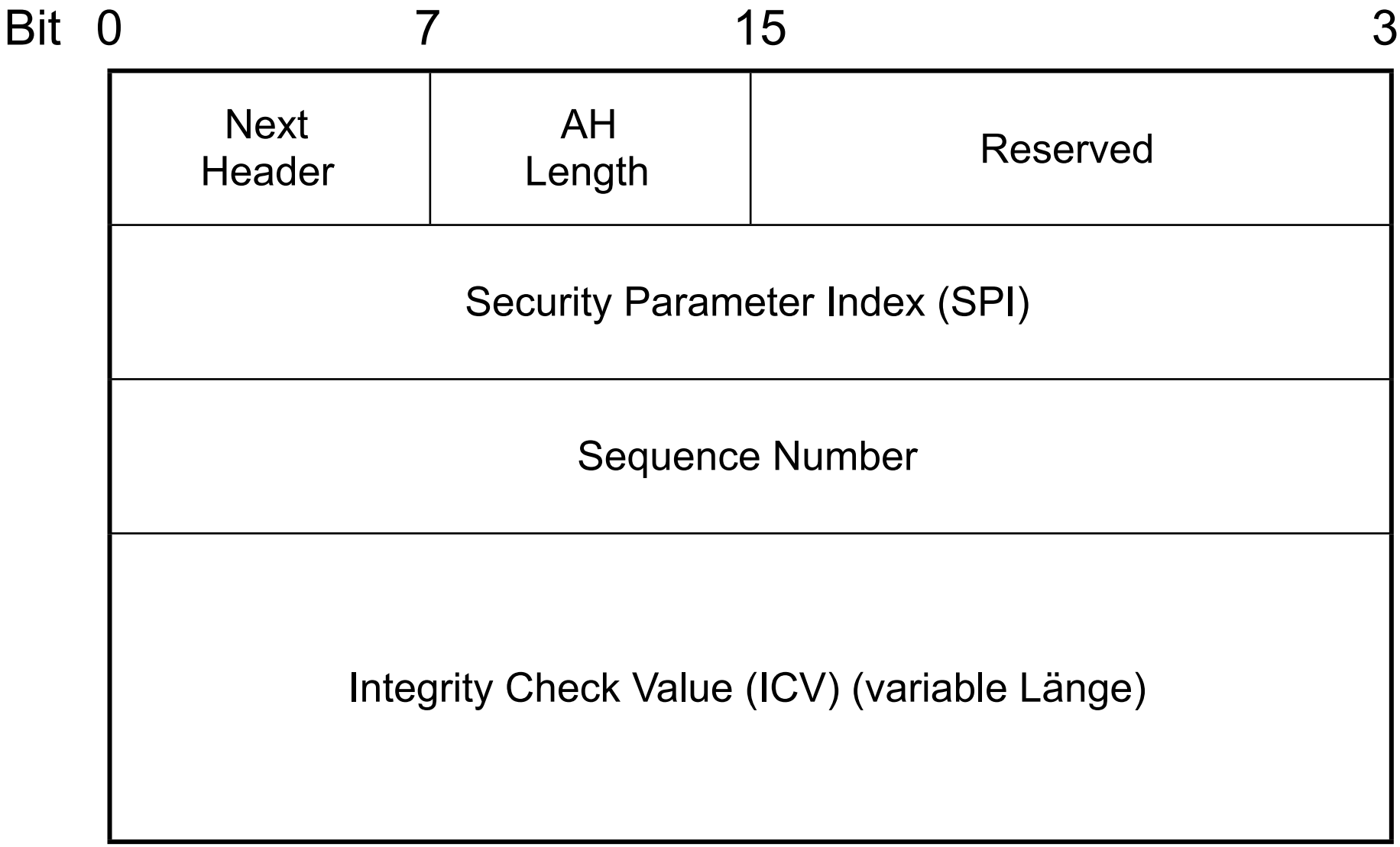


  - Integrität durch MAC
  - Authentisierung durch gemeinsamen Schlüssel
  - Anti-Replay durch gesicherte Sequenznummer

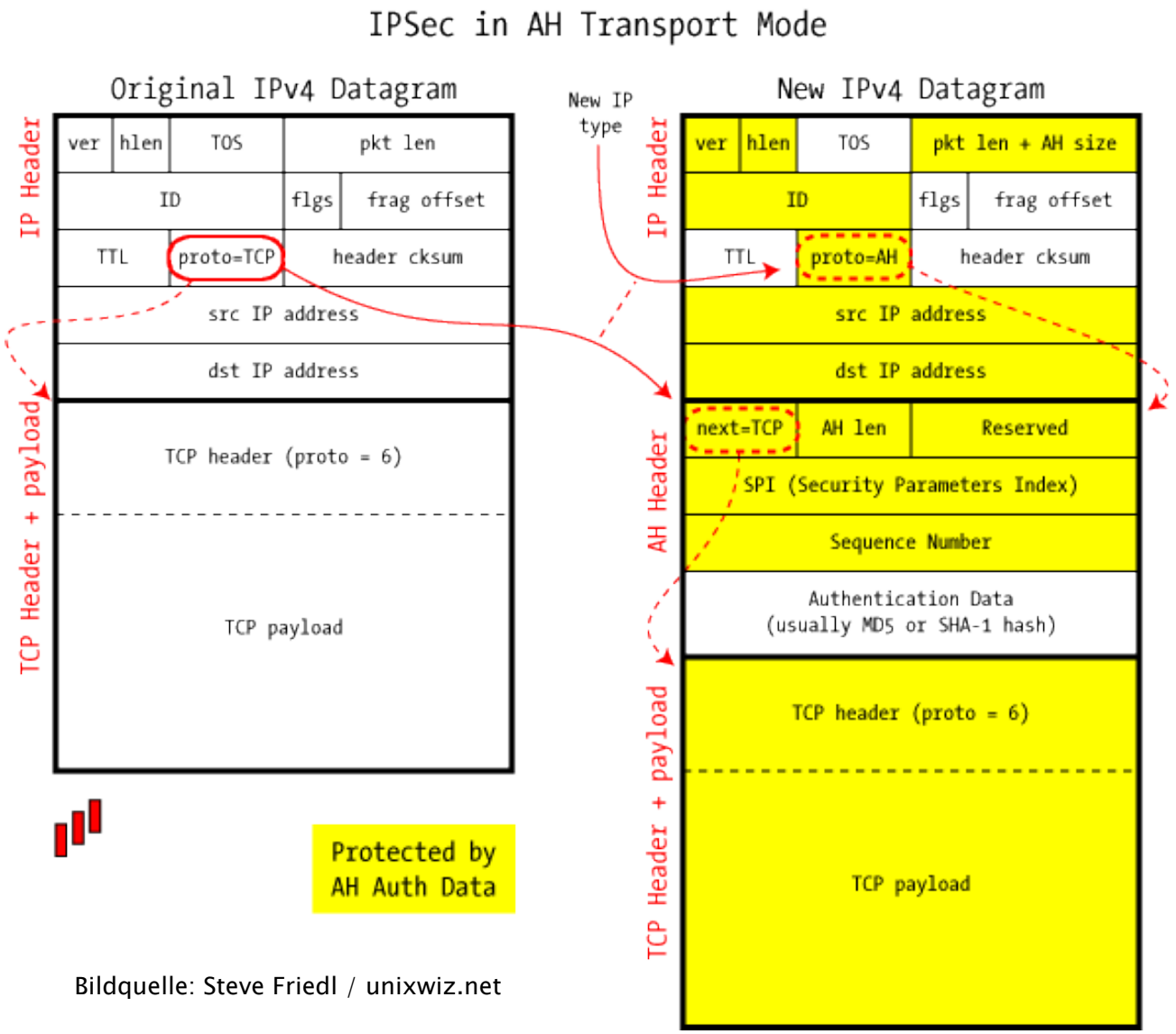
- AH im Tunnel Mode



AH Header im Detail

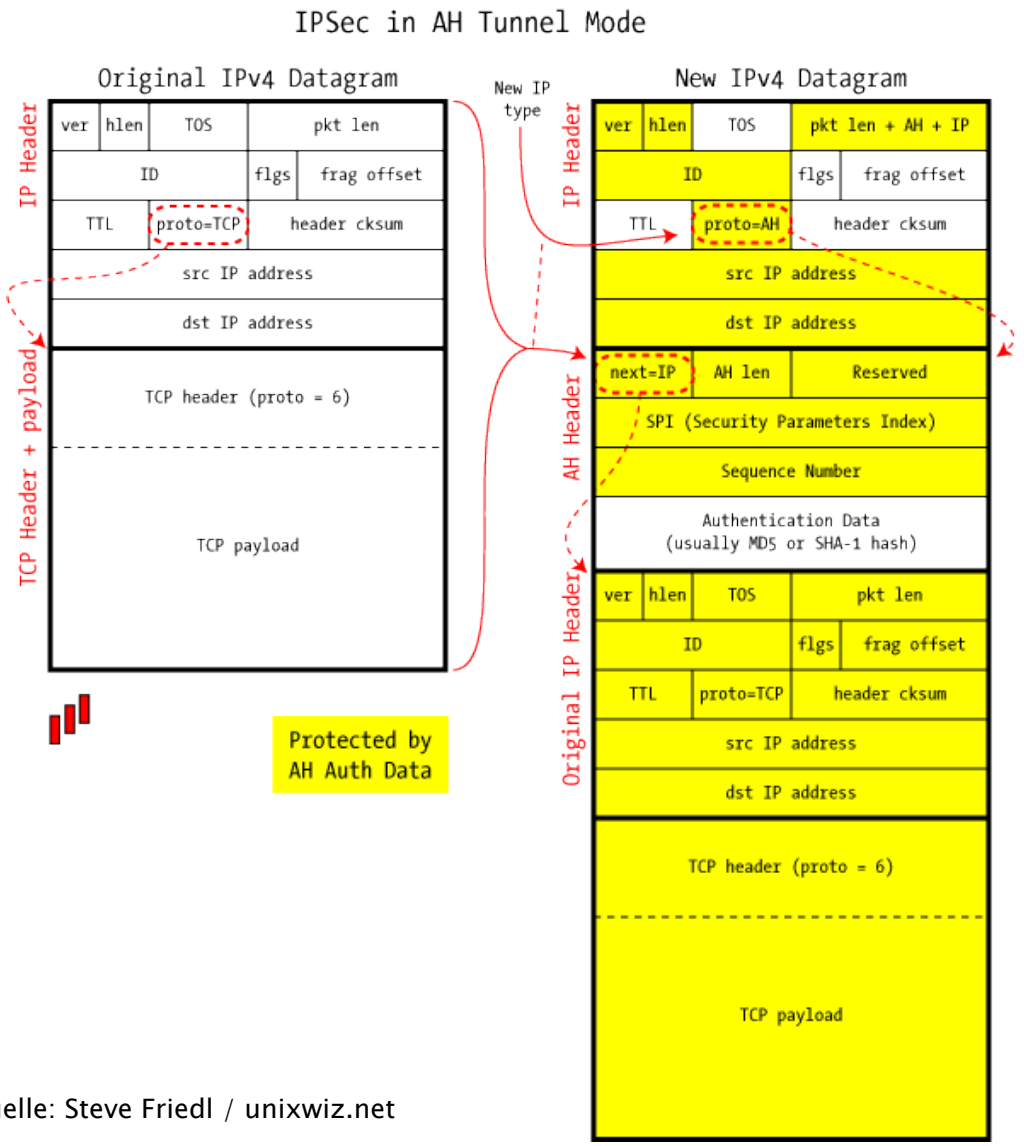


AH Transport Mode - Details



Bildquelle: Steve Friedl / unixwiz.net

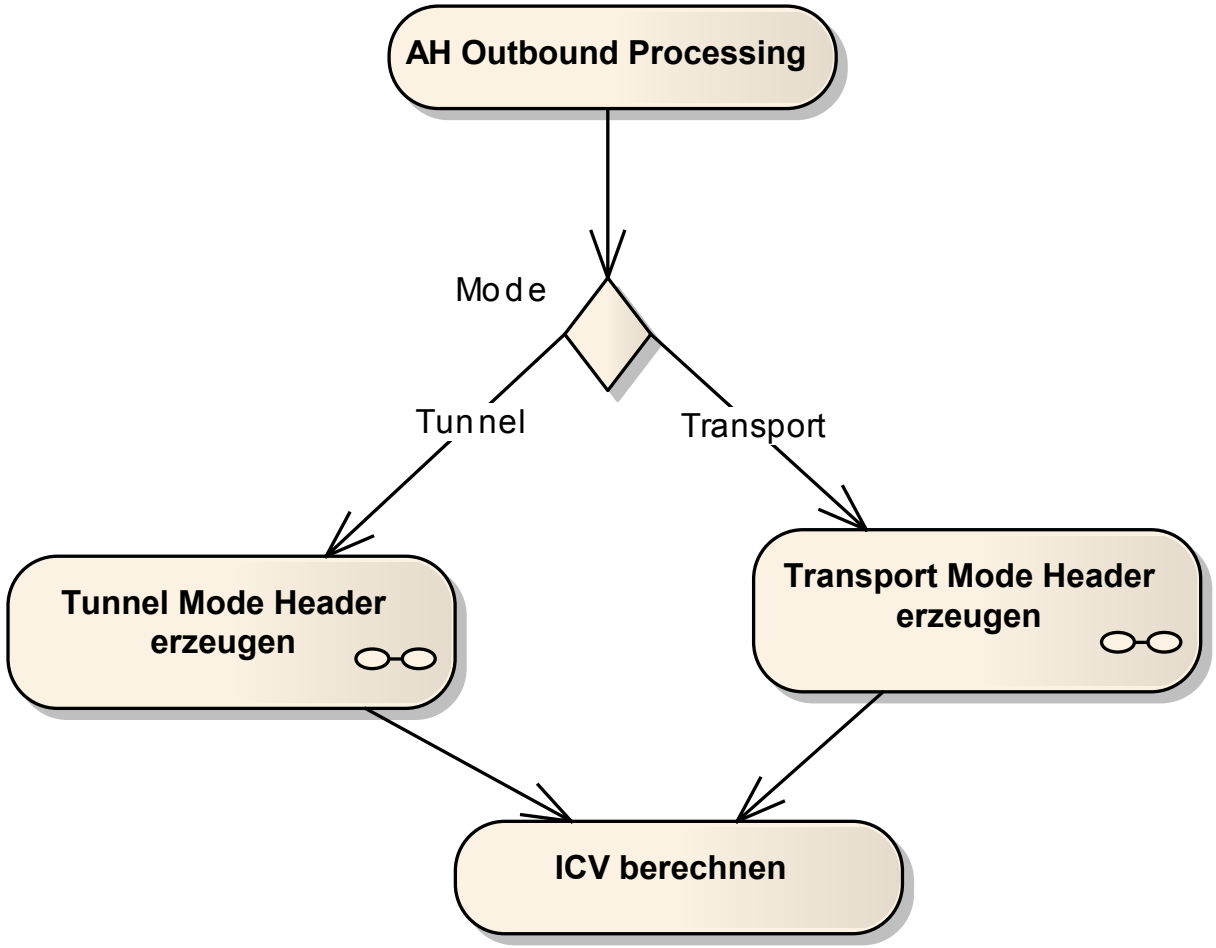
AH Tunnel Mode - Details



Bildquelle: Steve Friedl / unixwiz.net

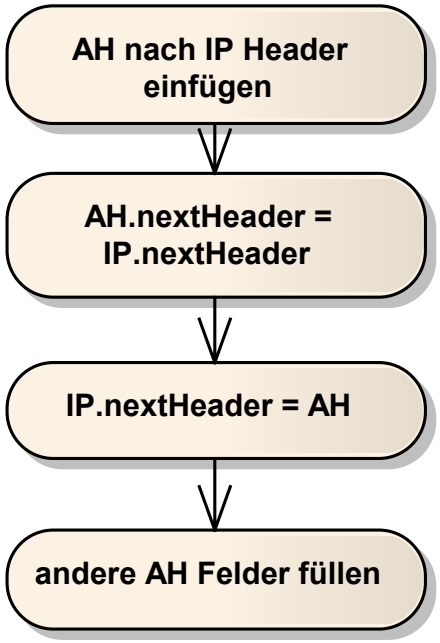
AH Outbound Processing

- IP-Stack hat ausgehendes Paket zu verarbeiten:

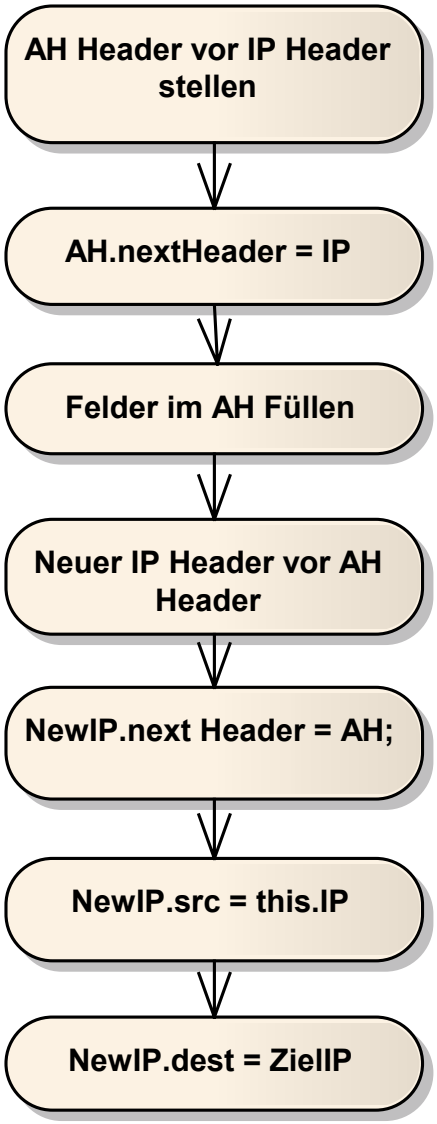


AH Outbound Processing 2

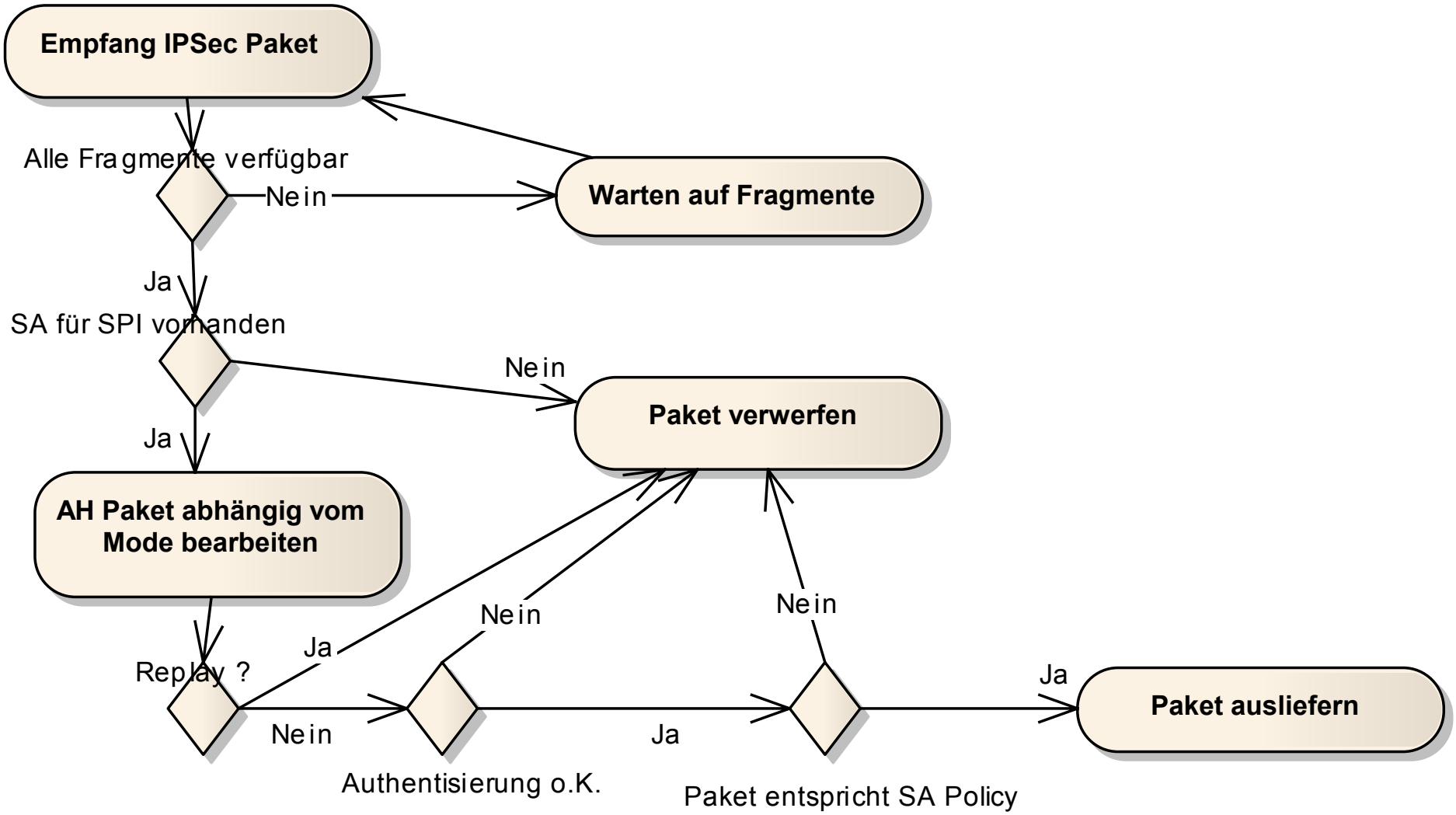
Transport Mode:



Tunnel Mode:

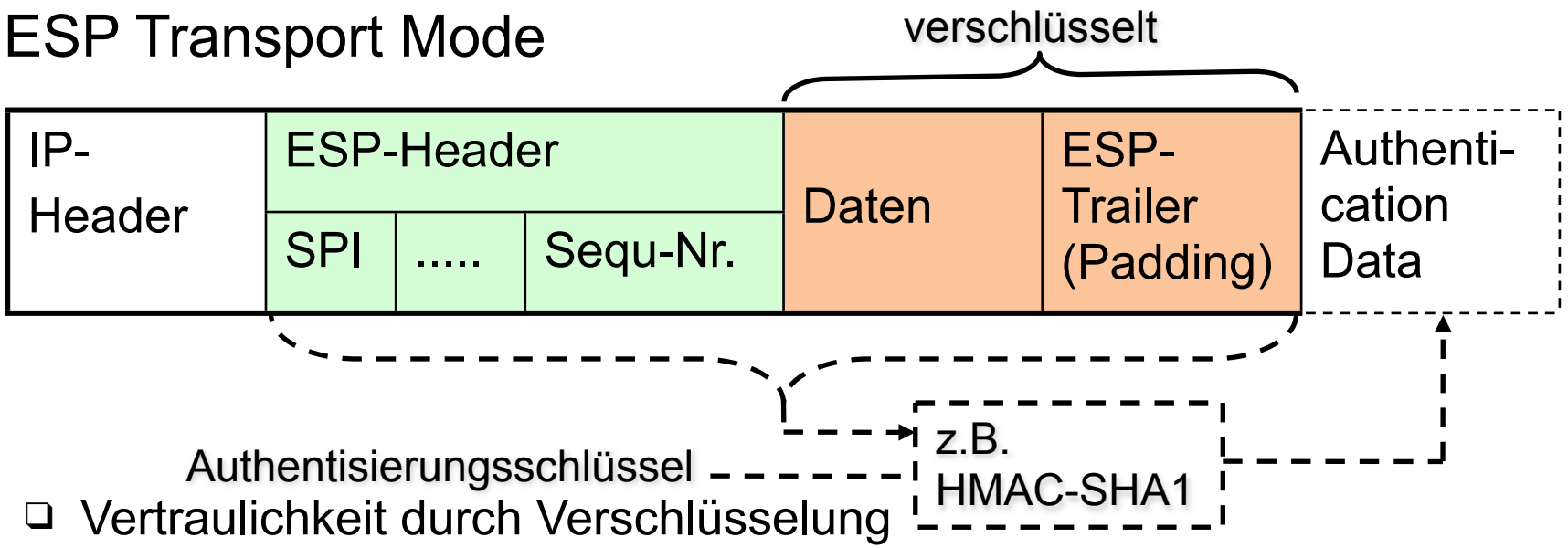


AH Inbound Processing



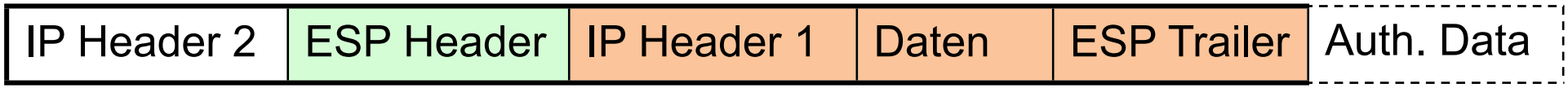
Encapsulating Security Payload (ESP) - Überblick

- ESP Transport Mode



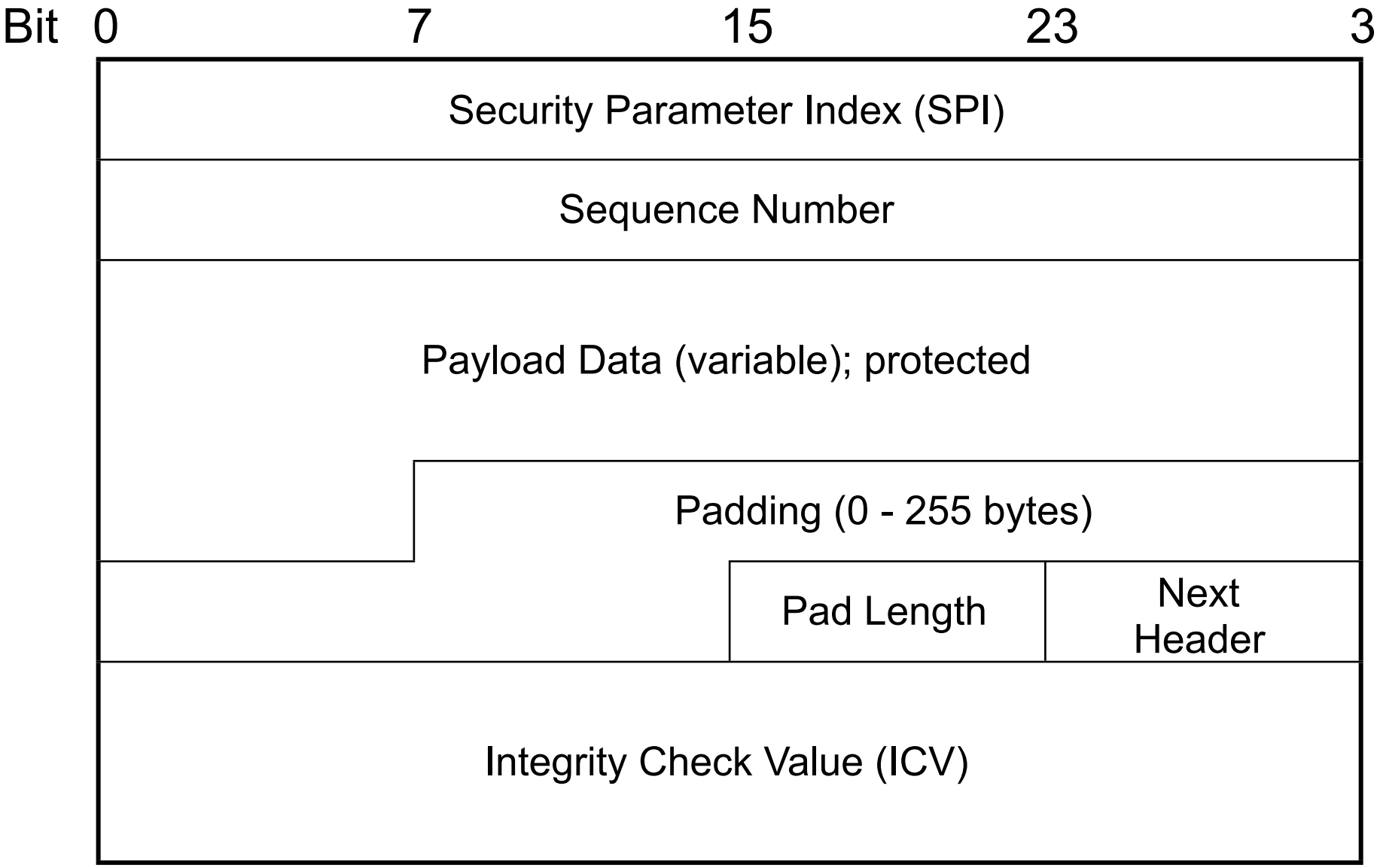
  - Vertraulichkeit durch Verschlüsselung
  - Integrität durch MAC (optional)
  - Authentisierung durch HMAC (optional)
  - Anti-Replay durch gesicherte Sequenznummer (optional)

- ESP Tunnel Mode

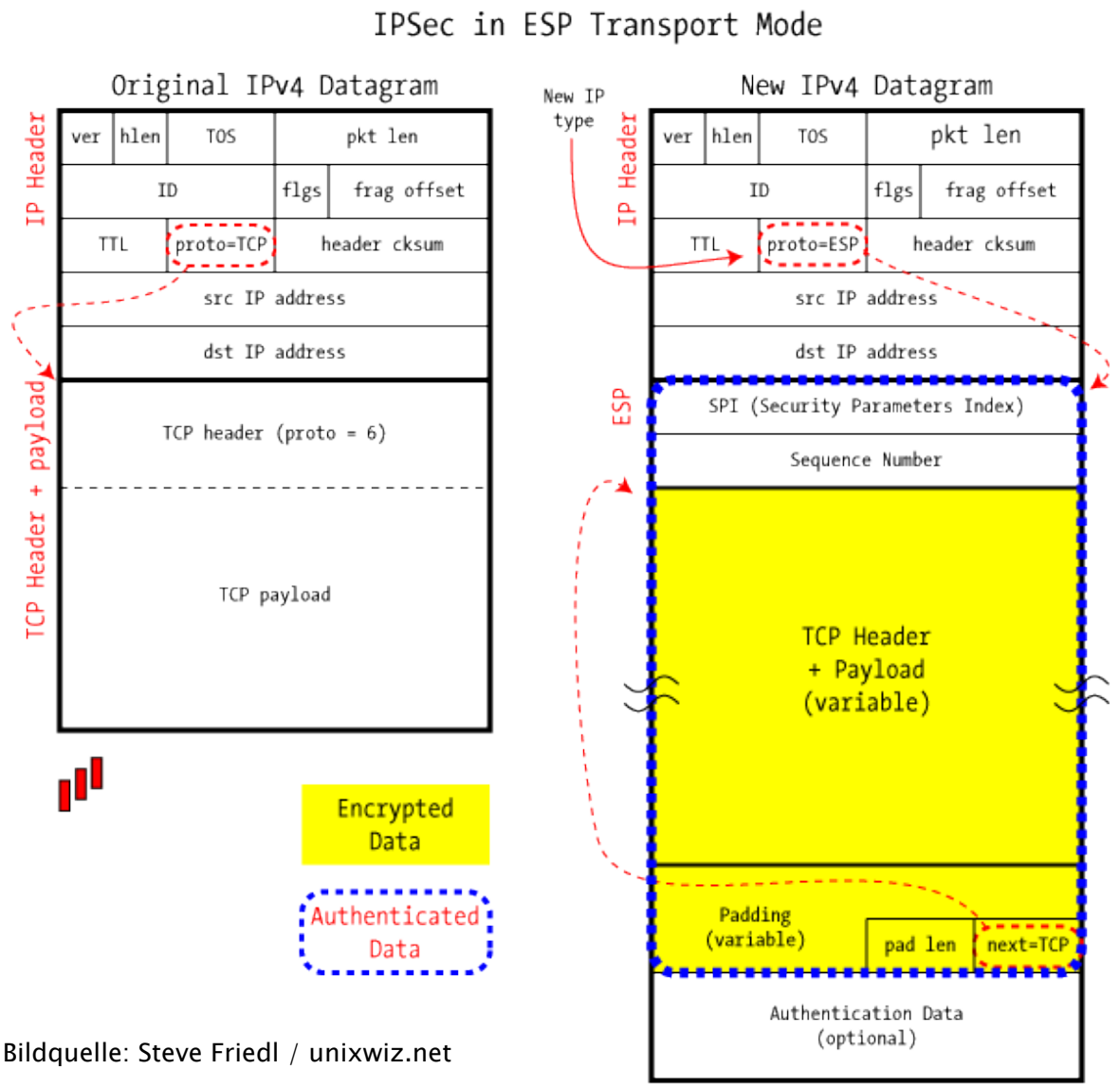


  - Anti-Traffic-Analysis durch verschlüsselten IP Header 1

ESP Header im Detail

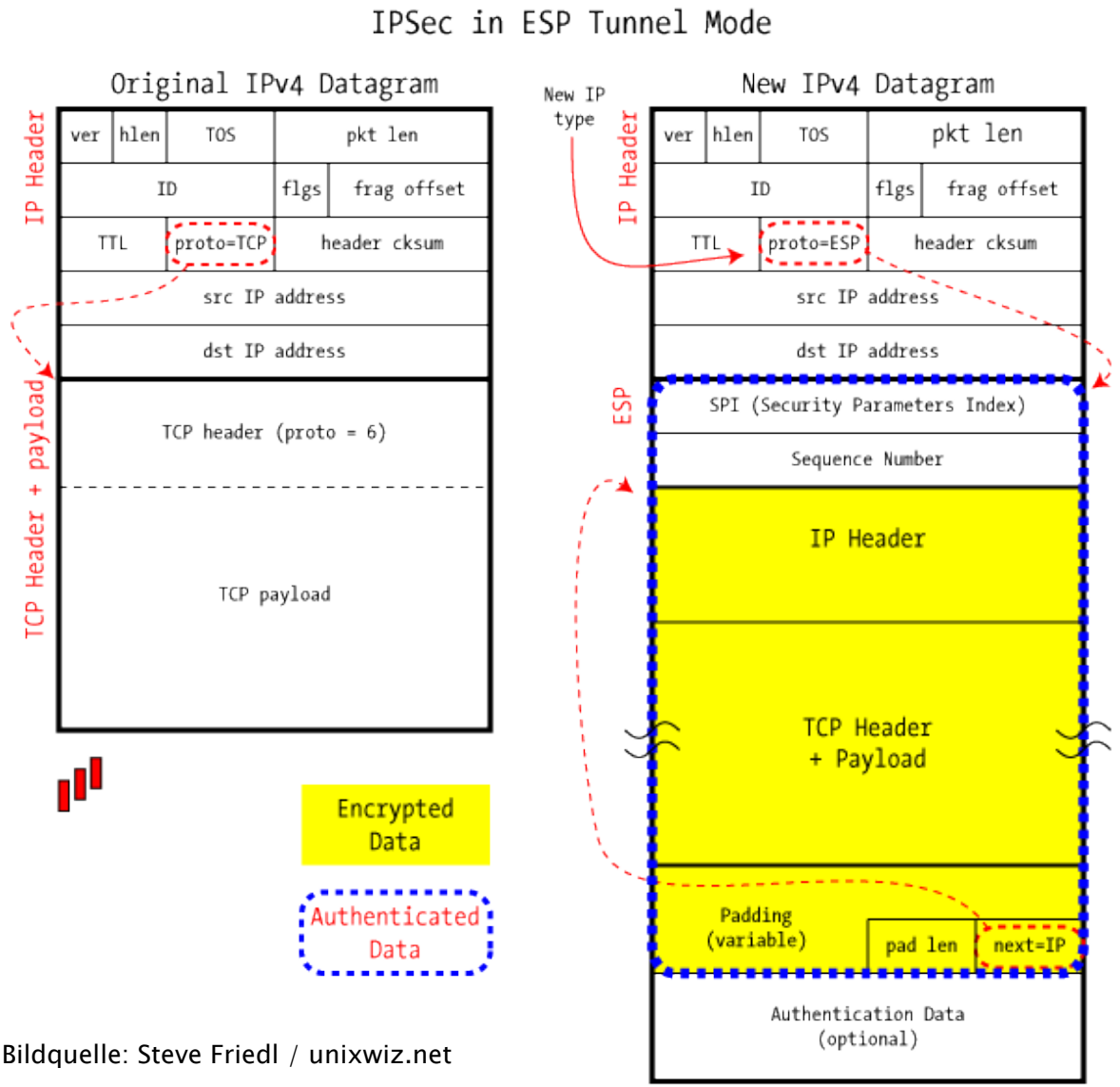


ESP Transport Mode - Details



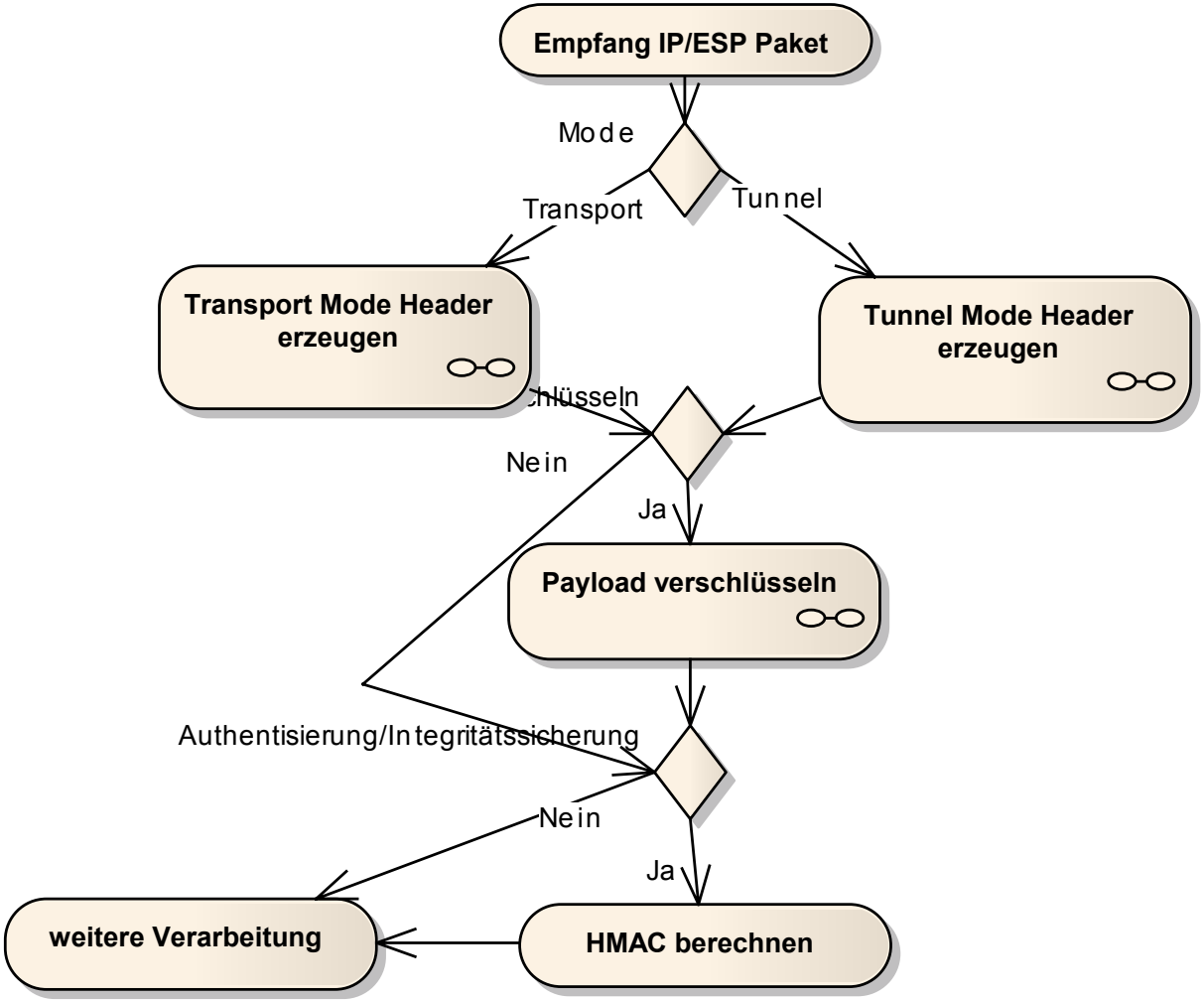
Bildquelle: Steve Friedl / unixwiz.net

ESP Tunnel Mode - Details

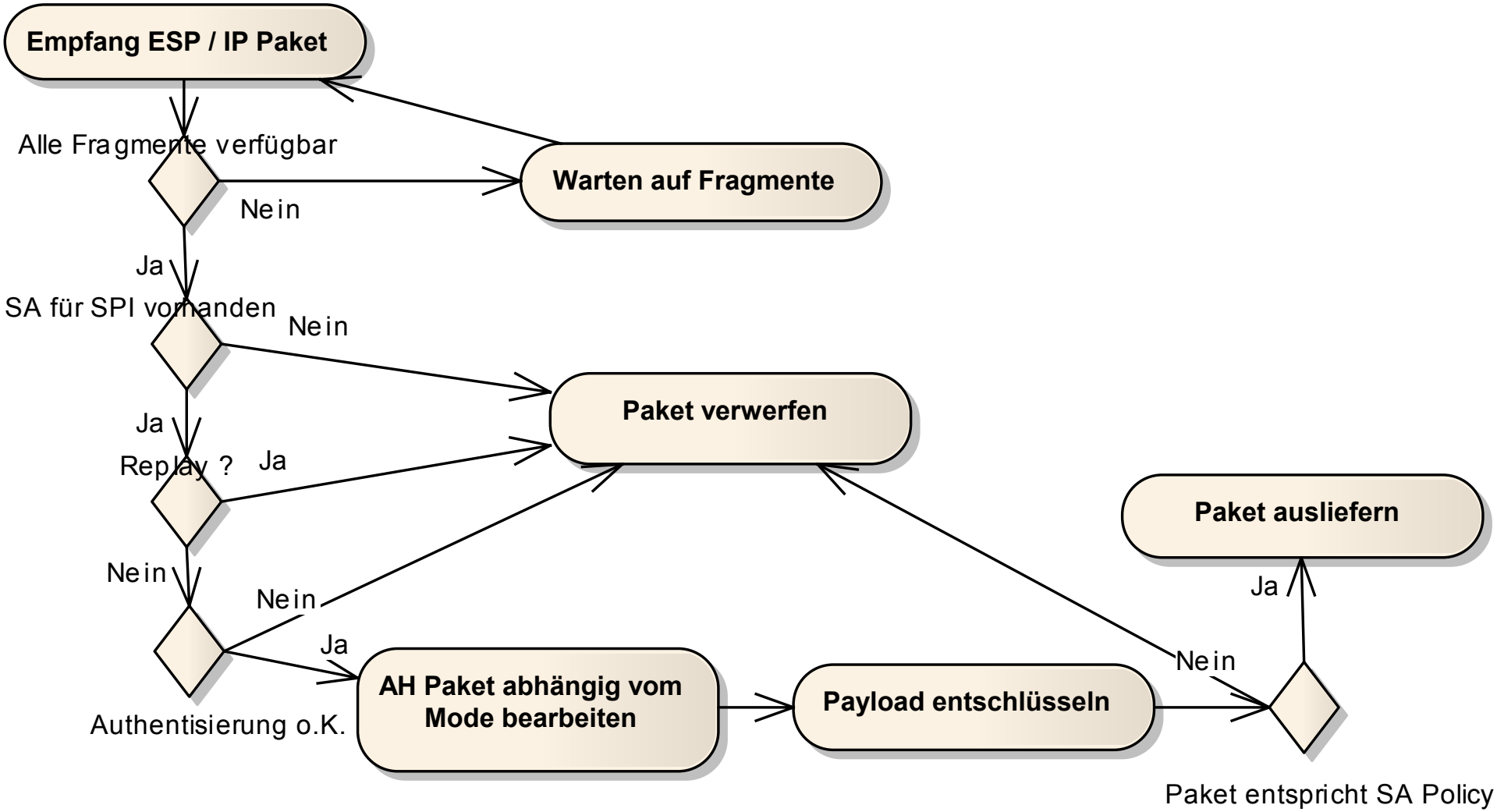


Bildquelle: Steve Friedl / unixwiz.net

ESP Outbound Processing



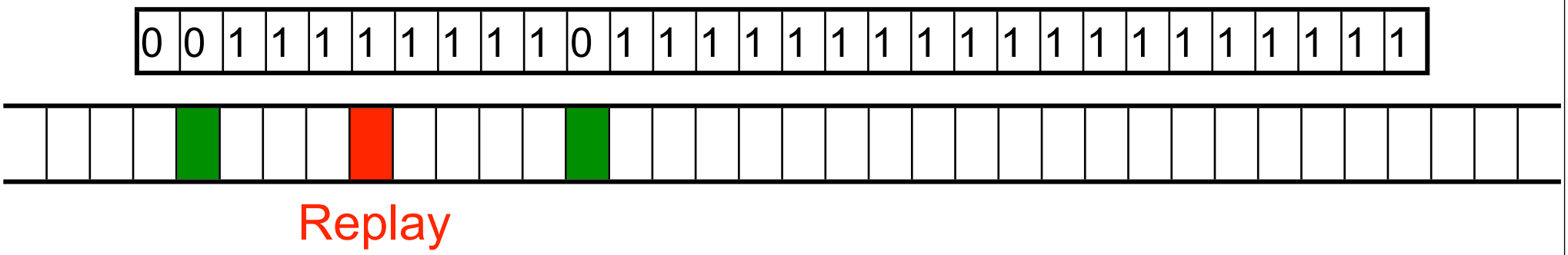
ESP Inbound Processing



IPSec Replay Protection

- Empfänger verwaltet Window für empfangene Pakete
  - Ursprünglich als Mechanismus, um Überfluten des Empfängers zu vermeiden
  - nicht größer als 32 Bit
- Grundprinzip:

Sliding Window empfangener Pakete



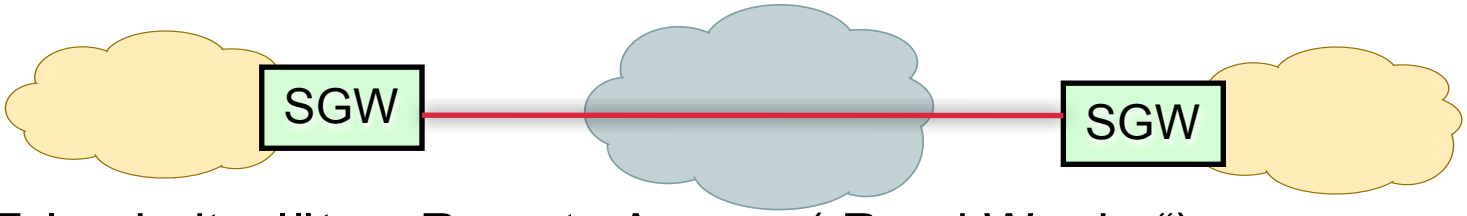
AH, ESP Algorithmen

- RFC 4305
- ESP Encryption
  - AES
  - 3DES
  - DES (Should Not)!
- ESP und AH Authentication
  - HMAC-SHA1-96
  - AES-XCBC-MAC-96
  - HMAC-MD5-96

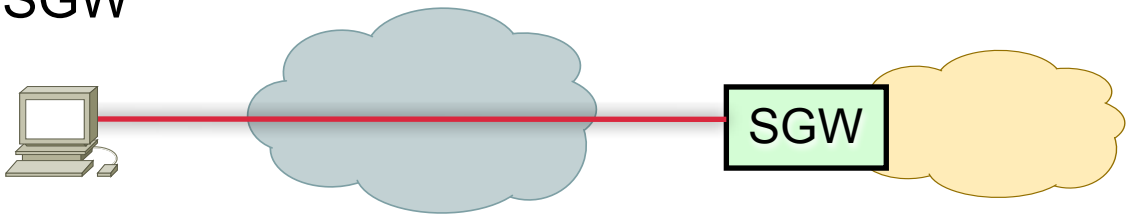
IPSec Anwendungsszenarien

- AH und ESP können kombiniert verwendet werden
- Auch Tunnel und Transport Mode können kombiniert werden

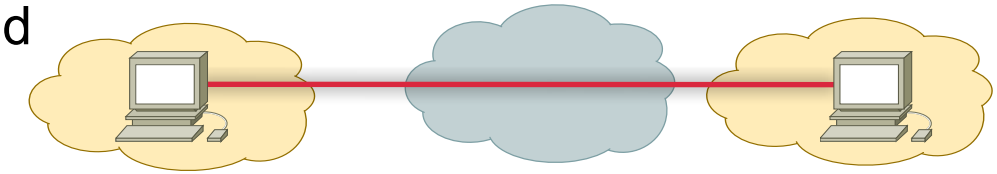
- Mögliche Einsatzszenarien
  - Kopplung von verschiedenen Unternehmensstandorten
    Verbindung von Security Gateway (SGW) zu Security Gateway



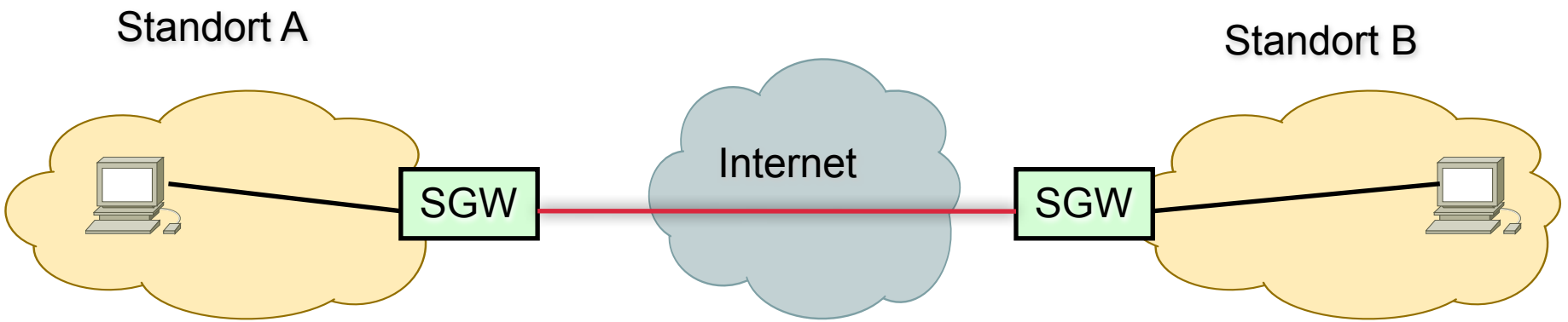
  - Telearbeitsplätze; Remote Access („Road Warrior“)
    Endsystem zu SGW



  - End-to-End



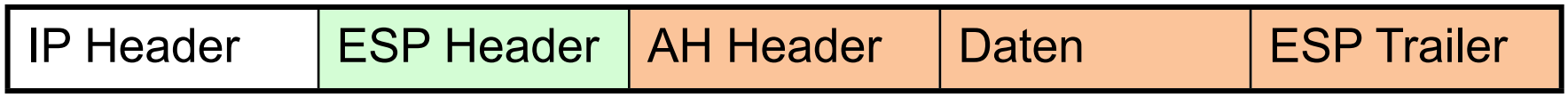
Szenario Standortvernetzung



- Mögliche Anforderungen:
  - Authentisierung SGW-to-SGW oder End-to-End
  - Integritätssicherung SGW-to-SGW oder End-to-End
  - Anti-Replay
  - Vertraulichkeit auch im (jeweils) internen Netz
  - SGW realisiert auch Firewall-Funktionen
  - Verwendung privater IP-Adressen in den Standorten
  - Verschattung interner Netzstrukturen

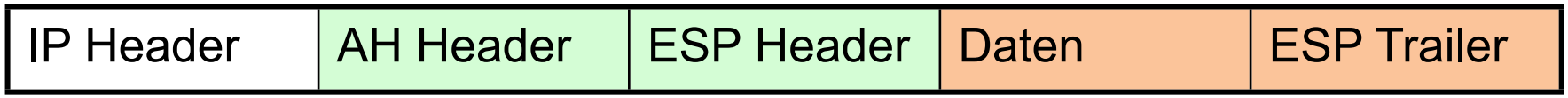
Protokollkombinationen

- AH Tunnel Mode am Security Gateway
  - Integritätssicherung
  - Authentisierung SGW to SGW
  - Private Adressen im internen Netz
- ESP Tunnel Mode am Security Gateway
  - Vertraulichkeit (auch der privaten Adressen)
- AH Transport am Endsystem / ESP Transport am SGW
  - Integritätssicherung
  - Authentisierung End to End
  - Vertraulichkeit ab SGW
  - Private Adressen nicht möglich
  - Nur theoretische Kombination; praktisch schwer realisierbar (Empfänger SGW nicht adressierbar)

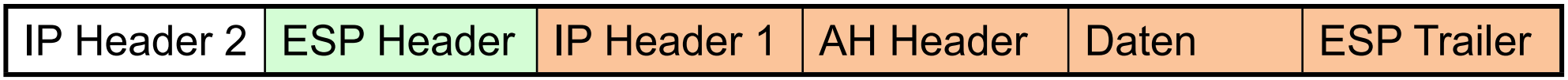


Protokollkombinationen (2)

- ESP Transport am Endsystem, AH Transport am SGW
  - Vertraulichkeit End to End
  - Authentisierung SGW to SGW
  - Private Adressen nicht möglich
  - SGW kann nicht mehr filtern (wegen Verschlüsselung)
  - Theoretisches Beispiel, in der Praxis schwer realisierbar, SGW nicht adressiert (transparentes SGW)



- AH Transport am Endsystem / ESP Tunnel am SGW
  - Integritätssicherung
  - Authentisierung End to End
  - Vertraulichkeit ab SGW
  - Private Adressen möglich



IPSec Security Association (SA)

- Inhalt einer SA
  - IPSec Protokoll Modus (Tunnel oder Transport)
  - Parameter (Algorithmen, Schlüssel, Zertifikat, Initialisierungsvektor,...)
  - Lebensdauer der SA
  - Sequenznummernzähler mit –overflow
  - Anti-Replay-Window
  - .....
- Identifikation einer SA per Kombination aus:
  - Security Parameter Index (SPI); 32 Bit Zahl
  - Ziel-Adresse
  - Verwendetes Protokoll (AH, ESP)
- D.h. in jede Richtung wird eine eigene SA vereinbart
- Jeder IPSec-Teilnehmer hält eine Security Policy Database (SPD) mit SAs

Inhalt

- Schwächen des Internet-Protokolls (IP)
- IPSec: Sicherheitserweiterung des IP-Protokolls
  - Authentication Header (AH)
  - Encapsulation Security Payload (ESP)
  - Anwendungsbeispiele
- Schlüsselverteilung mit IKEv2 (Internet Key Exchange)
  - Aufbau einer IKE SA
  - Authentisierung der Partner
  - Aufbau der IPSec SA
  - Erzeugung von Schlüsselmaterial

Einschub: Diffie-Hellman Schlüsselaustausch

- Ermöglicht den sicheren Austausch eines Schlüssels über einen unsicheren Kanal:
- Primzahl p und eine primitive Wurzel g (mod p) dürfen öffentlich bekannt gemacht werden
  (oft als Diffie-Hellman Group bezeichnet)

- Alice wählt ein x aus [1..p-2]
- Bob wählt ein y aus [1..p-2]
- Alice schickt $A = g^x \bmod p$ an Bob
- Bob schickt $B = g^y \bmod p$ an Alice

- Beide verwenden den folgenden Schlüssel:

$$Key = A^y = (g^x)^y = g^{xy} = (g^y)^x = B^x \pmod{p}$$

Einschub: Diffie-Hellman Beispiel

- Achtung: Üblicherweise Zahlen mit mehreren hundert Stellen!

- Alice und Bob einigen sich auf p=13 und g=2
- Alice wählt zufällig x=5, Bob wählt zufällig y=7
- Alice berechnet A = 2^5 mod 13 = 6, schickt dies an Bob
- Bob berechnet B = 2^7 mod 13 = 11, schickt dies an Alice
- Alice berechnet 11^5 mod 13 = 7
- Bob berechnet 6^7 mod 13 = 7
- Beide erhalten also das Ergebnis 7

- Angreifer kann die Zahlen 13, 2, 6 und 11 mithören, den Wert 7 aber nicht berechnen, da g^{xy} aufwendig zu berechnen ist, selbst wenn g, g^x und g^y bekannt sind.

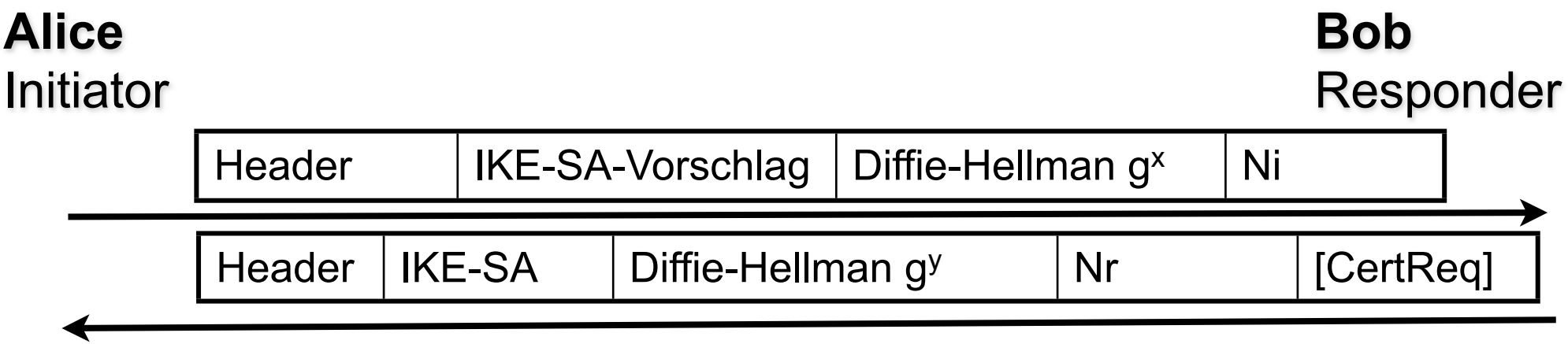
IPSec Schlüsselaustausch über IKEv2

- Protokollprimitive
  1. IKE_INIT
     - Aufbau einer bidirektionalen IKE SA
  2. IKE_AUTH
     - Authentisierung der Partner
     - Aufbau der ersten (und oft einzigen) bidirektionalen IPSec SA
  3. IKE_CHILD_SA
     - Aushandeln weiterer IPSec SAs
     - Re-Keying einer bestehenden SA

  - Ein durch IKE_AUTH etablierter Kanal kann für mehrere IKE_CHILD_SA Exchanges verwendet werden

- Erzeugung des Schlüsselmaterials
- Authentisierung in IKE (nicht erst in IPSec)

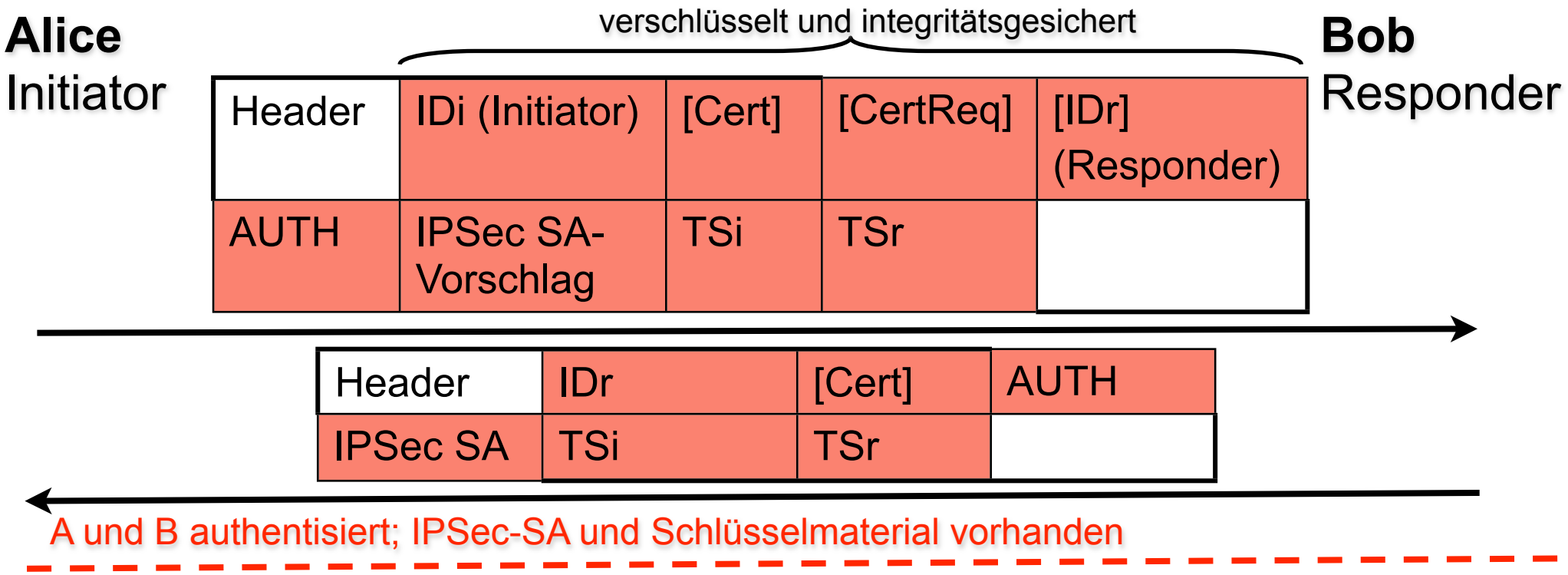
IKEv2: IKE_INIT



IKE-SA ausgehandelt, Schlüssel erzeugt, vertraulicher Kanal möglich; KEINE Authentisierung

- IKE-SA-Vorschlag:
  enthält die vom Initiator unterstützen Algorithmen
- Ni, Nr Zufallszahlen
- Diffie-Hellman Verfahren zur Berechnung von SKEYSEED
- Ableitung aus SKEYSEED (für jede Richtung separat)
  - SK_a: Authentisierungsschlüssel
  - SK_e: Schlüssel für Kryptoverfahren
- CertReq: Anforderung von Zertifikat(en); Optional

IKEv2: IKE_AUTH

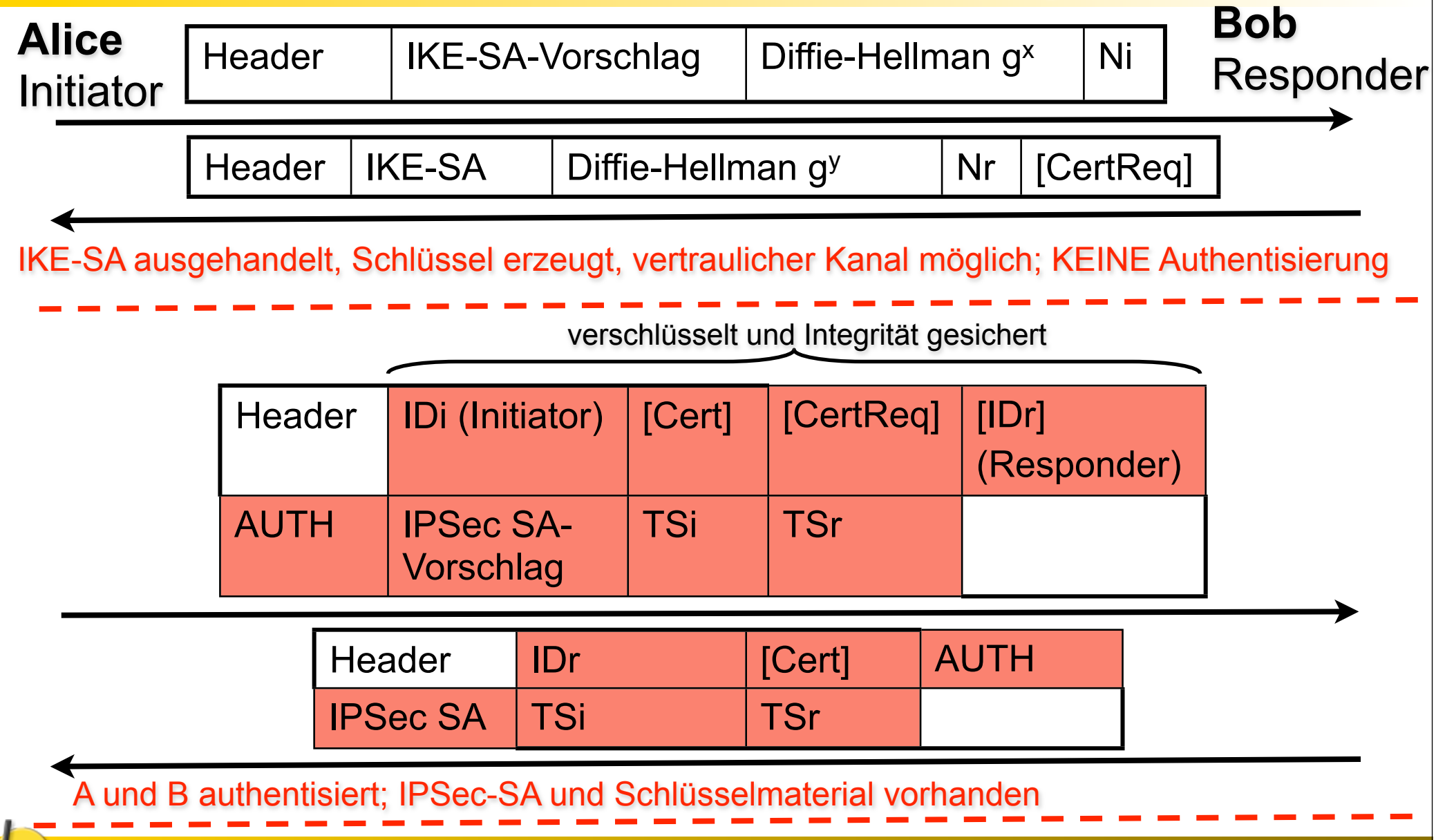


- Initiator und Responder können mehrere IDs haben; IDi und IDr bestimmen die jeweils gewählte ID
- Authentisierung über Public Key in AUTH
- Zertifikat und entsprechende Kette in Cert (Optional)
- TSx enthält Informationen aus lokaler Security Policy Database

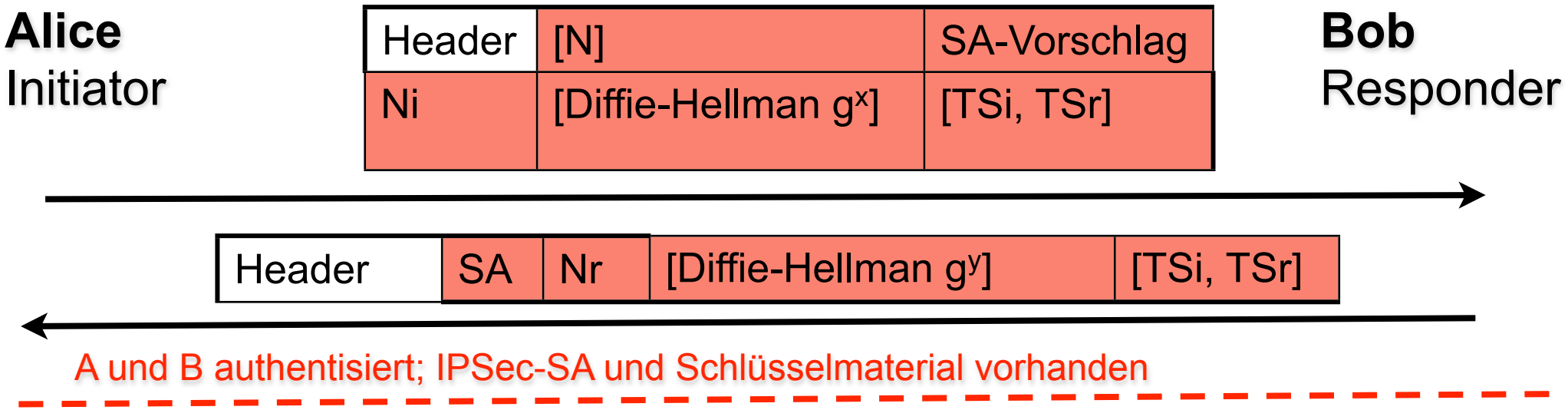
IKEv2: TSx

- Falls IP-Paket verarbeitet wird, für das „protect“ in der SPD gesetzt ist:
  - Paket muss verschlüsselt werden
  - Mögliches Problem: Es existiert keine SA
  - SPD-Verwaltung ist keine Aufgabe von IKE
  - Aber IKE dient zur Aushandlung von SAs
  - Informationen aus lokaler SPD können über TSx weitergegeben werden
  - Damit Wahrung der Konsistenz
- Bsp.: Bob ist Gateway für privates Subnetz
  - Alice will Verkehr ins Subnetz 10.11.12.* tunneln
  - TSi enthält Adress-Range: 10.11.12.0 - 10.11.12.255
  - Bob kann Adress-Range in TSr einschränken

IKEv2 : Zusammenfassung



IKEv2: CREATE_CHILD_SA



- Optional, da SA bereits mit IKE_AUTH ausgehandelt wird
- N enthält existierende SA, für die neues Schlüsselmaterial berechnet werden soll
- Optionaler Diffie-Hellman Key Exchange für Forward Security
- Nx Zufallszahlen

IKEv2: Schlüsselgenerierung

- IKE-SA legt fest:
  - Verschlüsselungsalgorithmus
  - Integritätssicherungsalgorithmus
  - Diffie-Hellman Group (p und g)
  - Zufallszahlenfunktion (Pseudo-random function, prf)

- prf wird zur Schlüsselerzeugung verwendet;
- Abhängig von der benötigten Schlüssellänge wird prf iteriert
  - prf+ = T1 | T2 | T3 | T4 | ... mit
  - T1 = prf(K, S | 0x01) K = Key, S = Seed
  - T2 = prf(K, S | 0x02)
  - .....
  - Tn = prf(K, S | 0x n)

IKEv2: IKE-Schlüsselmaterial

- IKE-SA Schlüsselmaterial:
  - SK_d verwendet zur Ableitung neuer Schlüssel für CHILD_SA
  - SK_{ai} Schlüssel für Integritätssicherung des Initiators
  - SK_{ar} Schlüssel für Integritätssicherung des Responders
  - SK_{ei} und SK_{er} Schlüssel für Verschlüsselung
  - SK_{ei} und SK_{pr} Erzeugung der AUTH Payload
- SKEYSEED = prf (Ni | Nr , g^{xy})
- IKE-SA Schlüsselmaterial:
  $\{SK_d \mid SK_{ai} \mid SK_{ar} \mid SK_{ei} \mid SK_{er} \mid SK_{pi} \mid SK_{pr}\} =$
  prf+ (SKEYSEED, Ni | Nr | SPI_i | SPI_r)
- CHILD_SA Schlüsselmaterial:
  - KEYMAT = prf+ (SK_d , Ni | Nr) bzw.
  - KEYMAT = prf+ (SK_d, g^{xy} | Ni | Nr)

IKEv2: Authentisierung

- mehrere Alternativen:
- Durch digitale Signatur eines vordefinierten Datenblocks
  - Verifikation durch Empfänger
  - Zertifikat (und evtl. entsprechende Kette) erforderlich
  - Optionale Anforderung und Übertragung: CertReq und Cert
  - Zertifikat kann auch schon bekannt sein
- Durch (keyed) HMAC des Datenblocks
- Verwendung des Extensible Authentication Protocol (EAP, vgl. Kap. 9)

IKEv2 Algorithmen

- Verschlüsselung:
  - DES, 3DES
  - RC5
  - IDEA, 3IDEA
  - CAST
  - Blowfish
  - AES
- Integritätssicherung:
  - HMAC_MD5_96
  - HMAC_SHA1_96
  - DES
  - AES
- Pseudo-Random Function (prf)
  - HMAC_MD5
  - HMAC_SHA1
  - HMAC_Tiger
  - HMAC_AES128

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Einschub:
Verizon 2009 Data Breach Investigations (Supplemental) Report

Inhalt

- Externe Aufklärung von erkannten IT-Sicherheitsvorfällen
- Überblick über die 15 häufigsten Bedrohungen 2009
- Ausgewählte Bedrohungen
  - Angriffsmethodik
  - Erkennen von Angriffen
  - Prophylaxe

Externe Aufklärung von IT-Sicherheitsvorfällen

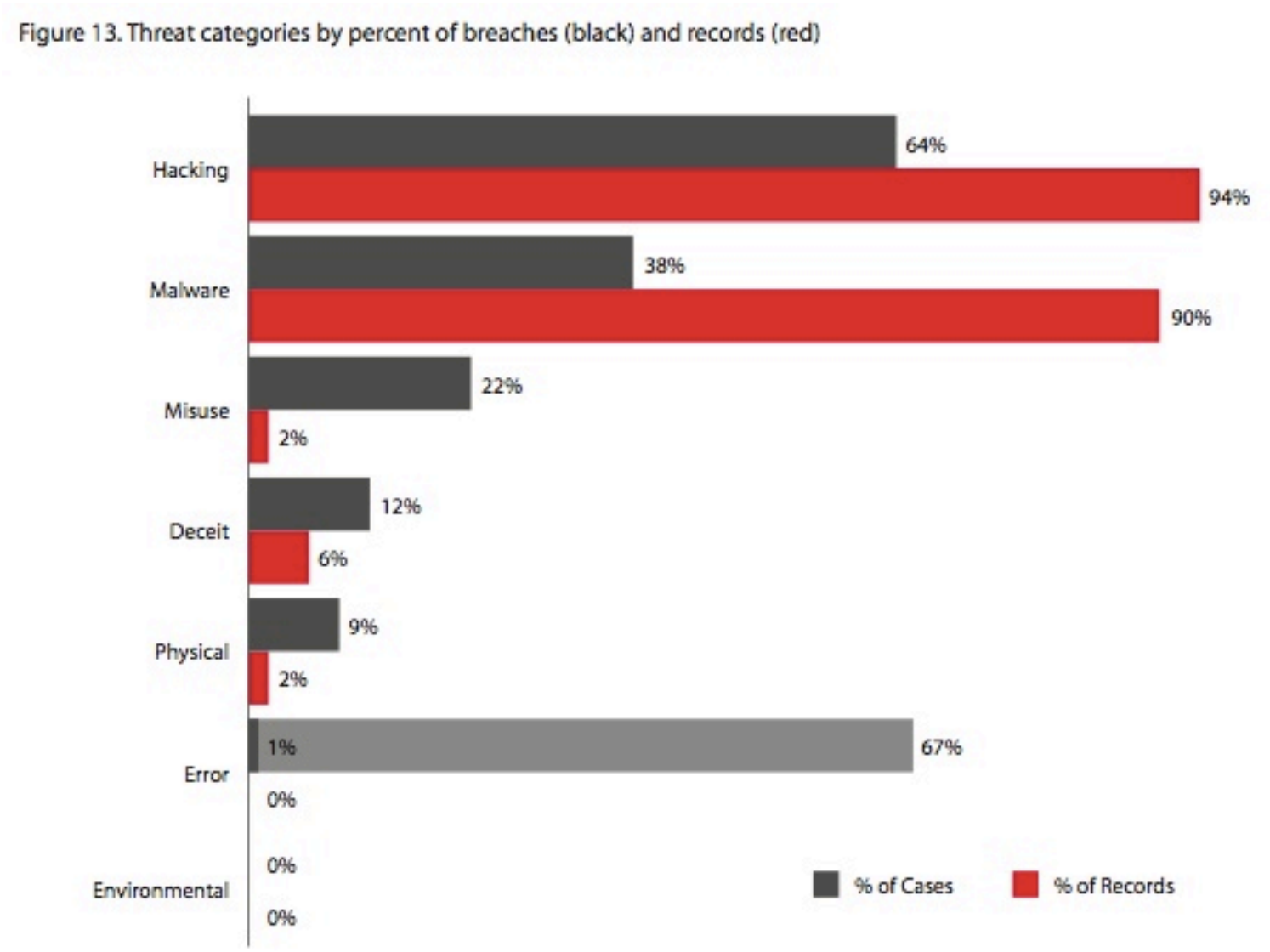
- Irgendwann wird das Problem bemerkt:
  - Systeme verhalten sich seltsam,
  - Kundendaten werden im Internet veröffentlicht,
  - Erpresser meldet sich, ...
- Vorfall muss möglichst genau aufgeklärt werden:
  - Wiederholungen verhindern
  - Juristische Aspekte (Haftung, Fahrlässigkeit, ...)
  - Sicherheit kontinuierlich verbessern
- Aufklärung häufig durch externe Dritte:
  - Oft kein firmeninternes IT-Forensik-Know-How vorhanden
  - Neutrale Analysen und Berichte als Beweismittel vor Gericht
  - Gefahr versuchter Vertuschung z.B. bei börsennotierten Unternehmen

Data Breach Report von Verizon

- Verizon Business
  - bietet IT-Forensik-Dienste für verschiedene Branchen
  - bekommt somit einen guten Überblick über erfolgreiche Einbrüche in IT-Systeme
  - fasst die Ergebnisse anonymisiert in jährlichen Berichten zusammen
- Dienstleisterberichte / White Papers sollten durchaus skeptisch betrachtet werden:
  - Behandeltes Themenspektrum nicht zwingend repräsentativ
  - Themenauswahl und -aufbereitung möglicherweise auch unter Marketing-Aspekten
  - Quellen können meist nicht überprüft werden
  - Keine externe Sicherung der fachlichen Qualität vor Veröffentlichung
- Nehmen als das, was es ist: Interessanter Überblick

Verizon Report: Überblick

- Verizon spezialisiert sich auf Angriffe, bei denen personen-bezogene Daten von externen Angreifern gestohlen wurden
- Interner Missbrauch (z.B. durch Mitarbeiter) kommt damit möglicher-weise zu kurz



Quelle: Verizon Business

Top 15 Bedrohungen lt. Verizon

Threat Category	Threat Action Type	Legend	% of Breaches	% of Records
Malware	Keyloggers and Spyware	KEYLOG	19%	82%
Malware	Backdoor or Command/Control	BACKDR	18%	79%
Hacking	SQL injection	SQLINJ	18%	79%
Misuse	Abuse of system access/privileges	ABUSE	17%	1%
Hacking	Unauthorized access via default credentials[2]	DFCRED	16%	53%
Misuse	Violation of Acceptable Use and other policies[3]	POLICY	12%	<1%
Hacking	Unauthorized access via weak or misconfigured ACLs	WKACL	10%	66%
Malware	Packet sniffer[4]	SNIFFER	9%	89%
Hacking	Unauthorized access via stolen credentials	STLCRED	8%	<1%
Deceit	Pretexting (Social Engineering)	SOCIAL	8%	2%
Hacking	Authentication bypass	BYPASS	6%	<1%
Physical	Physical theft of asset	THEFT	6%	2%
Hacking	Brute-force attack	BRUTE	4%	7%
Malware	RAM scraper[4]	RAMSCR	4%	<1%
Deceit	Phishing (and *ishing variations)	PHISH	4%	4%

Quelle: Verizon Business

Top 15 Bedrohungen (graphisch)



Quelle: Verizon Business

Bedrohung Nr. 1: Keylogger und Spyware

- Angriffsmethodik
  - U.a. Usernamen und Passwörter über Hardware, Software oder elektromagnetische / akustische Analyse abhören
  - Installation z.B. durch Benutzer (trojanisches Pferd), Lücke im Web-Browser oder bei physischem Zugang
  - Besonders häufig im Einzelhandel und in der Finanzbranche
  - Relevant in 19% der Fälle und bei 82% der gestohlenen Daten
- Erkennung durch Viren-/Malware-Scanner
  - Auffällige Prozesse laufen im Hintergrund
  - Dateien mit abgehörten Informationen im Filesystem
- Prophylaxe
  - Benutzer dürfen keine eigene Software installieren
  - Einmal-Passwörter
  - Ausgehenden Netzverkehr genau beobachten (Egress filtering)

Quelle: Verizon Business

Bedrohung Nr. 2: Backdoor or Command/Control

- Angriffsmethodik:
  - System (meist Server) wird über bekannte Sicherheitslücke kompromittiert
  - Angreifer installiert Software, um auch nach Schließen der Sicherheitslücke uneingeschränkten Systemzugang zu erhalten
  - Z.B. Austausch von Systemprogrammen und Serverdiensten (SSH, ...), Einfügen eigener Kernelmodule, ...
  - Relevant in 18% der Vorfälle und bei 79% der gestohlenen Daten
- Angriffserkennung:
  - Unerklärbare Systemaktivitäten / -last durch versteckte Hintergrundprozesse
  - Auffälliger Netzverkehr
- Prophylaxe:
  - Integritätssicherung durch Host Intrusion Detection Systeme
  - Zugriffseinschränkung durch Paketfilter-Firewalls
  - Zeitnahes Einspielen von Security-Patches für Serverdienste

Quelle: Verizon Business

Bedrohung Nr. 3: SQL Injection

- Angriffsmethodik:
  - Manipulation von Datenbankzugriffen bei mangelhafter Überprüfung der (bösartigen) Dateneingabe durch die Web-Anwendung
  - Ausmaß reicht vom Auslesen über das Manipulieren der gespeicherten Daten bis zur Kompromittierung des Datenbank-Servers
  - Relevant für 18% der Vorfälle und bei 79% der gestohlenen Daten
- Angriffserkennung:
  - Unübliche Einträge in Webserver- und Datenbank-Logfiles
- Prophylaxe:
  - Beim Implementieren auf Überprüfung von Benutzereingaben achten
  - Datenbank-Accounts mit minimal notwendigen Rechten anlegen
  - Datenbank-Fehlermeldungen nicht an Web-Client weiterleiten
  - Web Application Firewalls
  - Individuelle Penetrationstests

Quelle: Verizon Business

Bedrohung Nr. 4: Abuse of system access/privileges

- Angriffsmethodik:
  - Missbrauch der bewusst z.B. an Mitarbeiter oder Kunden/Anwender vergebenen Berechtigungen
  - besonders häufig z.B. bei gekündigten Administratoren
  - Relevant in 17% der Vorfälle, aber „nur“ bei 1% der gestohlenen Daten
  - Besonders häufig in der Finanzbranche und bei IT-Dienstleistern
- Angriffserkennung:
  - Überwachung aller privilegierten Accounts
  - Verhaltensanalyse (z.B. ungewöhnliche Login-Zeiten)
- Prophylaxe:
  - Auch administrative Accounts nur mit minimalen Rechten ausstatten
  - Separater Account pro Administrator
  - Erzwungene Protokollierung aller Eingaben (z.B. durch SSH-Daemon auf Server)
  - Rechte entziehen, bevor Missbrauch wahrscheinlicher wird (z.B. bereits vor Mitteilung der Kündigung)

Quelle: Verizon Business

Bedrohung Nr. 5: Default Credentials

- Angriffsmethodik:
  - Hard- und Softwaresysteme werden vorkonfiguriert ausgeliefert und z.B. mit den Default-Passwörtern vom Kunden in Betrieb genommen
  - Relevant in 16% der Vorfälle und bei 53% der gestohlenen Daten
  - Besonders häufig im Einzelhandel und der Lebensmittelindustrie
- Angriffserkennung:
  - Logins von unbekannten externen Adressen und/oder zu ungewöhnlichen Uhrzeiten
- Prophylaxe:
  - Keine Systeme im Auslieferungszustand in Produktivbetrieb nehmen
  - Default-Accounts löschen / Passwörter unbedingt ändern
  - Penetrationstests mit bekannten Default-Passwörtern

Quelle: Verizon Business

Bedrohung Nr. 6: Acceptable Use Policy violation

- Angriffsmethodik:
  - Mitarbeiter halten sich nicht an Einschränkungen z.B. bzgl. privater Internet-Nutzung mit Firmennotebooks
  - In der Folge Infektion mit Malware
  - Relevant in 12% der Vorfälle und bei < 1% der gestohlenen Daten
  - Häufig nicht Primärursache für Vorfälle, sondern vorbereitend oder additiv
- Angriffserkennung:
  - Auf Verstoß gegen Richtlinien prüfen (z.B. automatischer Scan von Browsercache und -history, Überprüfen von Proxy-Logfiles)
  - Virenscanner meldet Schadsoftware
- Prophylaxe:
  - Sicherheitsschulung und Sensibilisierung der Mitarbeiter
  - Benutzerrichtlinien einfach und klar verständlich halten
  - Angedrohte Sanktionen konsequent umsetzen

Quelle: Verizon Business

Bedrohung Nr. 7: Weak or misconfigured ACLs

- Angriffsmethodik:
  - Angreifer nutzt fehlende oder unzureichende Berechtigungseinschränkungen aus
  - Relevant in (nur!) 10% der Vorfälle und bei 66% der gestohlenen Datensätze
- Angriffserkennung:
  - Logfileüberprüfung
  - Verhaltensanalyse (z.B. Standort des Benutzers und Uhrzeit beim Login)
  - Intrusion Detection Systeme
- Prophylaxe:
  - Whitelist-Ansatz, d.h. alles verbieten, was nicht explizit erlaubt wurde
  - Minimale Berechtigungsvergabe an Benutzer und Administratoren
  - Zentrale Erfassung vergebener Berechtigungen

Quelle: Verizon Business

Bedrohung Nr. 8: Packet Sniffer

- Angriffsmethodik:
  - Angreifer nutzt kompromittierte Maschine oder physischen Netzzugang, um weitere Datenpakete abzuhören
  - Greift unverschlüsselte Datenpakete ab, selbst wenn Festplattenverschlüsselung genutzt wird
  - Ergebnisse werden häufig zunächst lokal gespeichert und nicht sofort per Internet übertragen
  - Relevant in 9% der Vorfälle und bei 89% der gestohlenen Daten
- Angriffserkennung:
  - Systeme mit Netzwerkkarten im „promiscuous mode“ suchen
  - Systeme mit großen Dateien / Schwankungen in der Plattenplatz-Belegung suchen
- Prophylaxe:
  - Separation des Netzdatenverkehrs
  - Verschlüsselte Datenübertragung auch im LAN
  - Host Intrusion Detection Systeme und Egress Filtering

Quelle: Verizon Business

Bedrohung Nr. 9: Stolen Credentials

- Angriffsmethodik:
  - Angreifer bringt Username/Passwort in Erfahrung oder gelangt in den Besitz von Smartcards, ...
  - „Insider“-Variante: Geschäftspartner/Kollege bestiehlt Mitarbeiter
  - Relevant in 8% der Vorfälle und bei < 1% der gestohlenen Daten
- Angriffserkennung:
  - Schwierig, da Angriff zunächst nach legitimem Zugriff aussieht
  - Logfile-Überwachung und Verhaltensanalyse
  - Anzeige der letzten Nutzungsdaten („last login“) beim Anmelden
- Prophylaxe:
  - Two-factor-authentication
  - Forcierte regelmäßige Passwortänderungen

Quelle: Verizon Business

Bedrohung Nr. 10: Social Engineering

- Angriffsmethodik:
  - Angreifer bringt Opfer dazu, sensible Informationen preiszugeben oder unscheinbare Aktionen im Rahmen eines Angriffs auszuführen
  - Vorgefunden in 8% der Vorfälle und bei 2% der gestohlenen Daten
  - Besonders häufig im Finanzwesen und bei IT-Dienstleistern - möglicherweise wegen besserer technischer Schutzmaßnahmen
- Angriffserkennung:
  - Bei guten Angriffen nur sehr schwer zeitnah zu erkennen
  - Auf ungewöhnliche Anfragen achten
- Prophylaxe:
  - Sensibilisierung der Mitarbeiter
  - Klar vorgegebene Richtlinien und Abläufe für die Auskunftserteilung
  - Ungewöhnliche Vorkommnisse melden lassen
  - Öffentlich zugängliche Informationen über Mitarbeiter auf einen sinnvollen Umfang einschränken

Quelle: Verizon Business

Bedrohung Nr. 11: Authentication bypass

- Angriffsmethodik:
  - Ausnutzung von (Software-)Fehlern, um ein System ohne Kenntnis des richtigen Passworts nutzen zu können
  - Relevant in 6% der Vorfälle und bei < 1% der gestohlenen Daten
- Angriffserkennung:
  - Logfile-Einträge auf Auffälligkeiten überprüfen
  - Intrusion Detection Systeme
- Prophylaxe:
  - Security-Aspekte bei der Software-Entwicklung von Anfang an beachten
  - Penetration Testing der eigenen Anwendungen
  - Vorgeschaltete Sicherheitsmaßnahmen, z.B. Web Application Firewalls

Quelle: Verizon Business

Bedrohung Nr. 12: Diebstahl

- Angriffsmethodik:
  - Angreifer bringt nicht nur Authentifizierungsinformationen in seinen Besitz, sondern ganze Geräte (Notebook, Smartphone, PC, ...)
  - Relevant für 6% der Vorfälle und bei 2% der gestohlenen Daten
- Angriffserkennung:
  - Fehlende Hardware
  - Videoüberwachung
  - Unbekannte Personen ohne Mitarbeiterausweis
- Prophylaxe:
  - Verschlüsselung der Daten mindestens auf mobilen Geräten
  - Datenträger vor der Verschrottung vollständig löschen
  - Ggf. Personenkontrolle am Ein-/Ausgang

Quelle: Verizon Business

Bedrohung Nr. 13: Brute-force attack

- Angriffsmethodik:
  - Z.B. Ausprobieren aller möglichen Passwörter, bis das richtige gefunden wurde
  - Besonders häufig im Einzelhandel und in der Gastronomie
  - Relevant in 4% der Vorfälle und bei 7% der gestohlenen Daten
- Angriffserkennung:
  - Logfileeinträge über gescheiterte Login-Versuche
  - Vermehrt Helpdesk-Anrufe bzgl. Freischalten automatisch gesperrter Kennungen
- Prophylaxe:
  - Passwort-Qualität über Passwort-Richtlinien sicherstellen
  - Inkrementelle Verzögerung von wiederholten Authentifizierungsversuchen desselben Benutzers
  - Two-factor-authentication

Quelle: Verizon Business

Bedrohung Nr. 14: RAM scraper

- Angriffsmethodik:
  - Schadsoftware durchsucht das RAM des befallenen Rechners nach „interessanten“ Daten
  - Dadurch Umgehung von Festplattenverschlüsselung
  - Relevant in 4% der Vorfälle und für 1% der gestohlenen Daten; Tendenz steigend
  - Bislang im Einzelhandel und in der Tourismusbranche am häufigsten
- Angriffserkennung:
  - Ähnlich wie bei anderer Schadsoftware, z.B. ungewöhnliches Systemverhalten, schlechte Performance, große Dateien, Virenscanner wird deaktiviert, ...
- Prophylaxe:
  - Installation neuer Software durch Rechteeinschränkung verhindern
  - Antiviren-Software nutzen
  - Ausgehenden Datenverkehr überwachen (Egress filtering)

Quelle: Verizon Business

Bedrohung Nr. 15: Phishing und Variationen

- Angriffsmethodik:
  - Benutzer wird z.B. über eine E-Mail mit gefälschter Absendeadresse auf den Webserver eines Angreifers gelockt
  - Die Webseite sieht wie die eines bekannten, vertrauenswürdigen Dienstes aus (z.B. Internet-Banking, Online-Shopping, ...), dient aber nur dazu, Benutzername/Passwort und ggf. andere Daten des Benutzers abzugreifen
  - Relevant in 4% der Vorfälle und für 4% (!) der gestohlenen Daten
- Angriffserkennung:
  - Seltsame Aufforderungen, z.B. zur Datenkorrektur bei einer Bank, per E-Mail; häufig mit vielen Grammatikfehlern
- Prophylaxe:
  - Sensibilisierung der Benutzer
  - Links in E-Mails genau prüfen, HTML-E-Mails als reinen Text anzeigen
  - Firmenweit keine ähnlichen Aufforderungen selbst per E-Mail schicken

Quelle: Verizon Business

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 12: Netzsicherheit - Schicht 4: Transport Layer SSL / TLS



Inhalt

- Transport Layer: Funktionen
- Secure Socket Layer (SSL) & Transport Layer Security (TLS) - Historie
- SSL / TLS Protokoll-Architektur
  - SSL / TLS Record Protocol
  - SSL / TLS Handshake Protocol
  - Schlüsselerzeugung
- SSL / TLS Anwendung

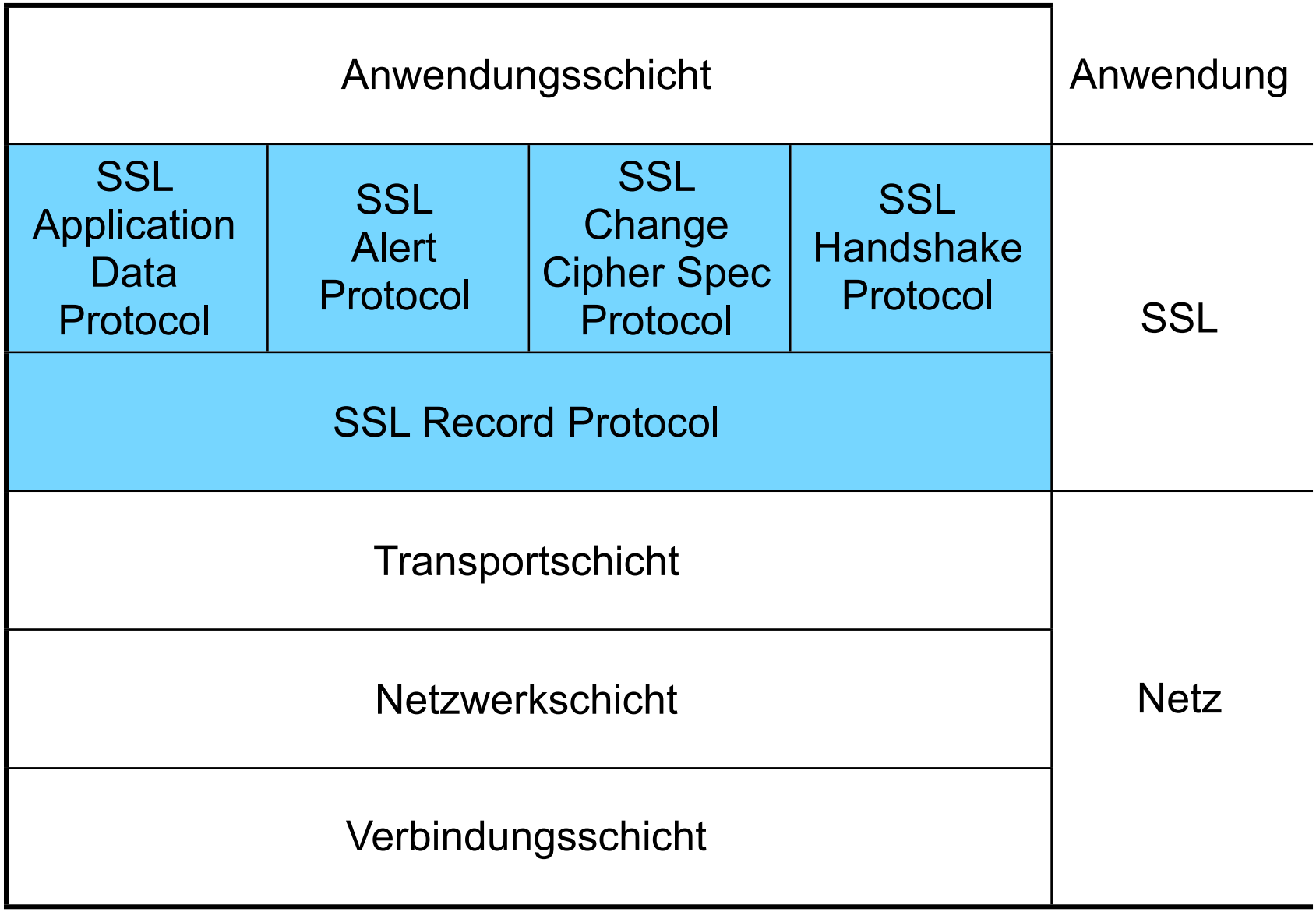
Transport Layer

- Nach OSI: Transportdienst zwischen Endsystemen; Ende-zu-Ende
  - Sicherer Transport von
  - Nachrichten der Endsysteme
- In der Internet-Welt: Ende zu Ende Verbindung zwischen Anwendungen
  - OSI-Schichten 5, 6 und 7 fallen in der Anwendungsschicht zusammen
  - Ports definieren die Prozesse (Dienste) der Anwendungsschicht
- Sicherungsprotokolle der Transportschicht
  - setzen auf TCP oder UDP auf
  - realisieren zum Teil die Funktionalität der Sitzungsschicht
  - liegen zwischen Transport Layer und Application Layer

Secure Socket Layer (SSL): Historie

- Ab 1994 ursprünglich entwickelt, um HTTP-Verkehr zu sichern (https); entwickelt von Netscape und ab SSL v2 in deren Browser integriert
- 1995 Internet Explorer mit PCT (Private Communication Technology)
- SSL v3: Protokollverbesserungen (aus PCT) und de-facto Standard
- Kann beliebige Anwendungen sichern (nicht nur HTTP)
- IETF entwickelt basierend auf SSL ab 1996 Transport Layer Security (TLS)
  - SSL gehört der Firma Netscape
  - IETF basierte freie Spezifikation
  - TLS 1.0 und SSL 3.0 sind nahezu identisch
  - SSL und TLS werden häufig synonym gebraucht
  - Aktuell: TLS v1.2 (August 2008)

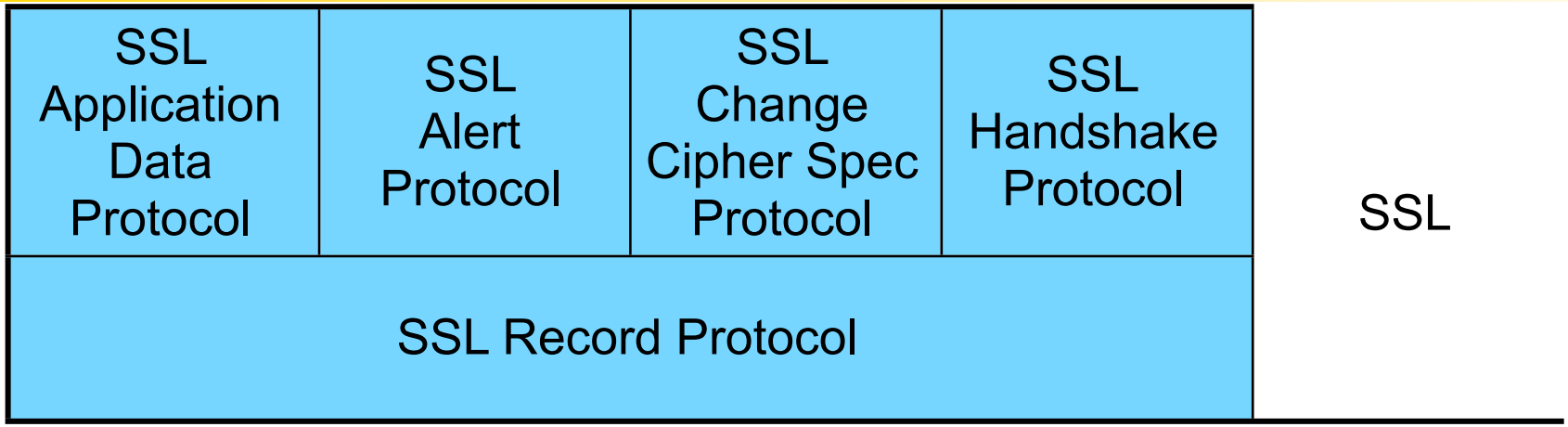
SSL/TLS Einordnung



SSL / TLS Überblick

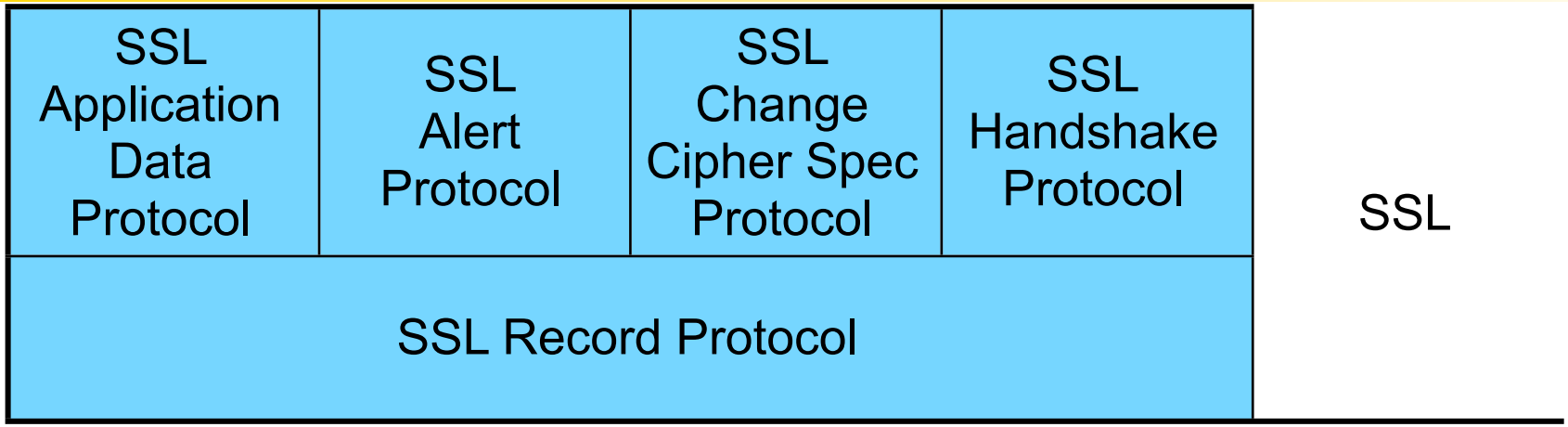
- Authentisierung
  - Vor der eigentlichen Kommunikation ist eine Authentisierung möglich
  - Einseitig oder auch zweiseitig:
    - Nur Client prüft Server (z.B. HTTPS bei Online-Banking)
    - Nur Server prüft Client (eher unüblich)
    - Client und Server prüfen sich gegenseitig (z.B. Intranet-Zugang mit Client-Zertifikat)
- Vertraulichkeit der Nutzdaten
  - Nur, falls während des Sitzungsaufbaus vereinbart
  - Verschiedene (symmetrische!) Verschlüsselungsverfahren: RC2, RC4, DES, 3DES, DES40, IDEA, AES
- Integrität der Nutzdaten
  - Kryptographischer Hash-Wert, parametrisiert mit Schlüssel: HMAC
  - Algorithmen: MD5, SHA

SSL/TLS Protokoll Architektur



- Application Data Protocol
  - Datenübermittlung zwischen Anwendung und SSL
  - Zugriff auf Record Protocol
- Alert Protocol:
  - Warn- und Fehlermeldungen
- Change Cipher Spec Protocol
  - Änderung der Krypto-Verfahren
  - Initialisierung und Einigung auf neu zu verwendende Verfahren

SSL/TLS Protokoll Architektur (Forts.)



- Handshake Protokol:
  - Authentisierung
  - Schlüsselaustausch
  - Vereinbarung der Parameter
- Record Protocol
  - Fragmentierung
  - Kompression der Klartext-Daten (optional)
  - Verschlüsselung (optional)
  - Integritätssicherung (optional)

SSL/TLS Record Protocol

7	15	23	31
Type	Major Version	Minor Version	Length
Length	Data		

- Type
  - Change Cipher Spec (20)
  - Alert (21)
  - Handshake (22)
  - Application Data (23)
- Major und Minor Version (z.B. 3, 2 für TLS 1.1)
- Length: Länge der Daten in Byte

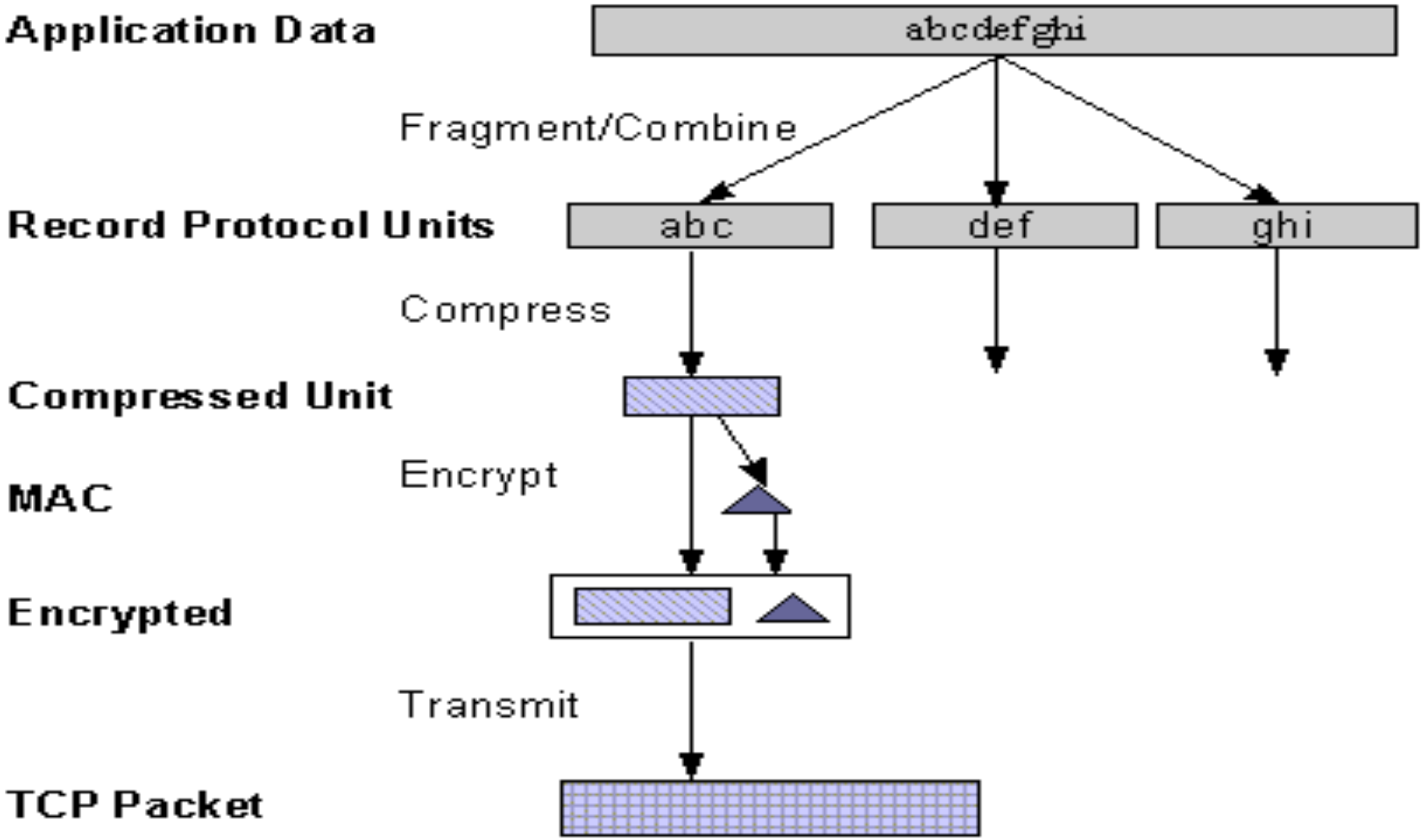
SSL/TLS Record Protocol

7	15	23	31
Type	Major Version	Minor Version	Length
Length	Data		

- Sender
  1. Fragmentierung der Nutz-Daten in max 2^{14} Byte
  2. Kompression der Daten (Default-Algorithmus *null*)
  3. Integritätssicherung mittels HMAC
  4. Verschlüsselung

- Empfänger:
  - Entschlüsselung; Integritäts-Check; Dekompression; Defragmentierung; Auslieferung an höhere Schicht

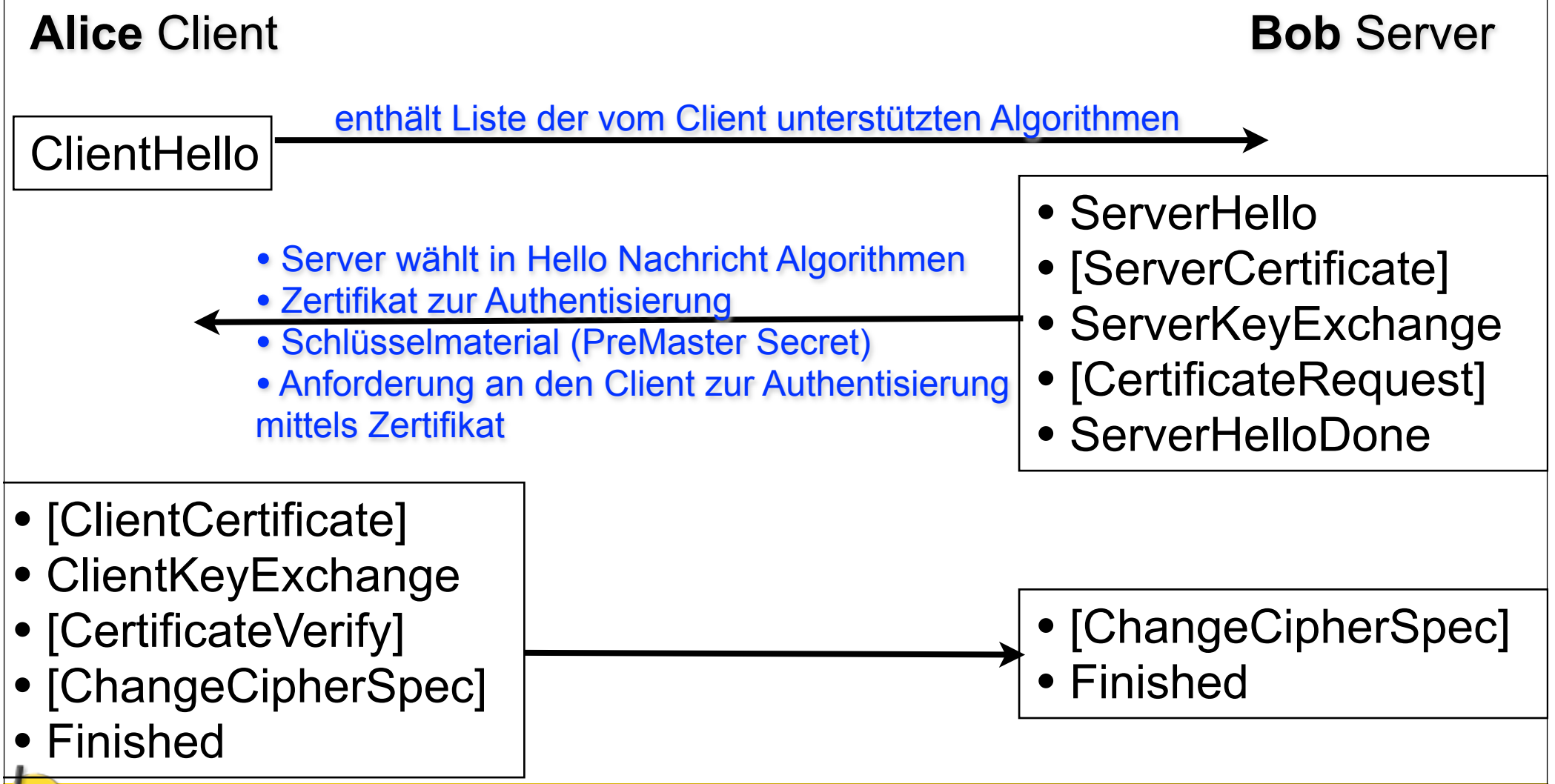
SSL/TLS Record Protocol - Ablauf



Bildquelle: Ralf S. Engelschall, Apache mod_ssl Dokumentation

SSL/TLS Handshake Protokoll

- Zweck: Authentisierung, Algorithmenauswahl, Schlüsselmaterial



SSL/TLS Handshake Protocol: Schlüsselerzeugung

- Schlüssel werden aus dem PreMasterSecret abgeleitet
- PreMasterSecret (variable Länge) wird erzeugt:
  - **RSA**: Zufallszahl; mit dem privaten Schlüssel des Servers verschlüsselt übertragen
  - **Diffie-Hellman**: Übertragung der Diffie-Hellman Gruppe unverschlüsselt; falls nicht schon in Zertifikat enthalten; Erzeugung des PreMasterSecret über Diffie-Hellman Verfahren
- MasterSecret (immer 48 Byte) wird aus PremasterSec. erzeugt
  - MasterSecret = PRF (PreMasterSecret, „Master Secret“, ClientHello.random + ServerHello.random)

SSL / TLS Schlüsselgenerierung

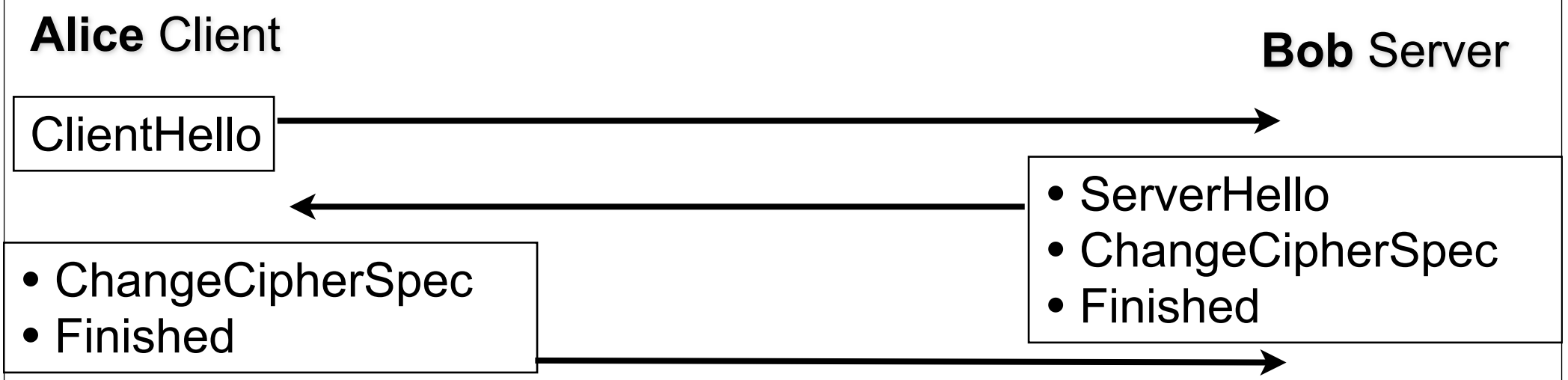
- KeyBlock = PRF (SecurityParameter.MasterSecret,
  „key expansion“,
  SecurityParameter.ServerRandom +
  SecurityParameter.ClientRandom)
- Der KeyBlock wird in folgende Teilblöcke zerlegt
  - client_write_MAC_secret [SecurityParameter.HashSize]
  - server_write_MAC_secret [SecurityParameter.HashSize]
  - client_write_key [SecurityParameter.KeyMaterialLength]
  - server_write_key [SecurityParameter.KeyMaterialLength]
- SSL erlaubt Schlüsselerzeugung auch ohne Authentisierung
  - In diesem Fall Man-in-the-Middle-Attack möglich und nicht erkennbar

SSL/TLS: Pseudo Random Function (PRF)

- Pseudo-Random Function (PRF); gebildet aus MD5 und SHA
- PRF soll Sicherheit bieten auch wenn MD5 oder SHA „gebrochen“ werden
- Expansionsfunktion P_hash(secret, seed)
  - Durch iterative Anwendung Schlüsselmaterial in beliebiger Länge
  - P_hash(secret,seed) = HMAC_hash(secret, A(1) | seed) |
    HMAC_hash(secret, A(2) | seed) |
    ...
    HMAC_hash(secret, A(n) | seed) mit
  - A(0) = seed
    A(i) = HMAC_hash(secret, A(i-1));
- PRF(secret, label, seed) = P_MD5(S1, label + seed) XOR
  P_SHA-1(S2, label + seed)
  - mit secret zerlegt in zwei Teilstrings S1 und S2

SSL Abbreviated Handshake

- Erlaubt Wiederverwendung und Duplizierung eines bestehenden Sicherheitskontextes
- HTTP 1.0; Jedes Item einer Webseite wird über eigene TCP Verbindung übertragen



- ClientHello enthält SessionID der zu duplizierenden Session
- Falls Server SessionID findet und duplizieren, will sendet er SessionID in ServerHello zurück

SSL Alert Protocol

- Rund 25 verschiedene Mitteilungen, z.B.
  - Beendigung der Session
  - Fehler in der Protokollsyntax (decryption failed, etc.)
  - Probleme mit der Gültigkeit von Zertifikaten
- Unterscheidet zwischen
  - Warnungen
  - Fehlern
- Fehler führen zu sofortigem Verbindungsende

SSL/TLS Anwendung

- Auswahl an SSL gesicherten Diensten

Port	gesicherter Dienst	Protokoll
443	HTTP	https
465, 587	SMTP (Mail)	ssmtp oder smtps
585,993	IMAP	imap4-ssl
636	LDAP	ldaps
989, 990	FTP	ftps
992	Telnet	telnets
995	POP3	pop3s

SSL/TLS und HTTP

- HTTPS ist aufgrund der weiten Verbreitung von HTTP das häufigste SSL-Einsatzgebiet
- Problem bei virtuellen Webservern (z.B. Apache VHost):
  - Pro IP-Adresse nur 1 Zertifikat möglich
  - Zum Zeitpunkt des SSL-Handshakes liegen noch keine im HTTP-Request enthaltenen Angaben zum gewünschen VHost vor
- Gelöst mit TLS v1.2: Server Name Indication übermittelt gewünschten Servernamen bereits beim Verbindungsaufbau
- Aber: Verzögerung zwischen Spezifikation (RFC 5246), Implementierung (z.B. OpenSSL) und Rollout (z.B. Integration in Linux-Distributionen)

Problematische Aspekte des SSL-Einsatzes

- Performance
  - Verbindungsaufbau ist rechenintensiv und damit langsamer
  - Hohe Belastung für Server mit vielen Clients
  - (Symmetrische) Verschlüsselung benötigt nur wenig Rechenzeit
    - Aber oft Verzicht auf Kompression (Entropiereduzierung!)
- Keine Ende-zu-Ende-Verschlüsselung
  - Bei Kommunikation über mehr als zwei Stationen erhält jede Zwischenstation den Klartext
- Usability in der Praxis suboptimal
  - Beispiel HTTPS-Zertifikatsprüfung in aktuellen Browsern: Häufig eher als lästig statt als hilfreich empfunden
- Immer wieder Implementierungsfehler in SSL-APIs

Einschub: OpenSSL Security Alert [07-Jan-2009]

- Incorrect checks for malformed signatures
- Systems affected:
  - OpenSSL prior to release 0.9.8.j
- Description / Impact:
  - Implementation error : malformed signature can be treated as a good signature rather than an error
- Impact:
  - Man in the middle attack
  - Bypassing validation
- Solution:
  - Upgrade to OpenSSL 0.9.8j
- Quelle:
  - `http://www.openssl.org/news/secadv_20090107.txt`

SSL-Angriffsmöglichkeiten

- Überwiegend implementierungsspezifische Angriffe
- Andere Varianten nur bedingt erfolgsversprechend:
  - Brute Force: Gesamten Schlüsselraum durchsuchen
  - Known Plaintext Attack: Viele Nachrichtenteile sind vorhersagbar, z.B. HTTP GET-Befehle
- Fehler im Protokoll-Konzept
  - Z.B. TLS & SSLv3 renegotiation vulnerability (November 2009)

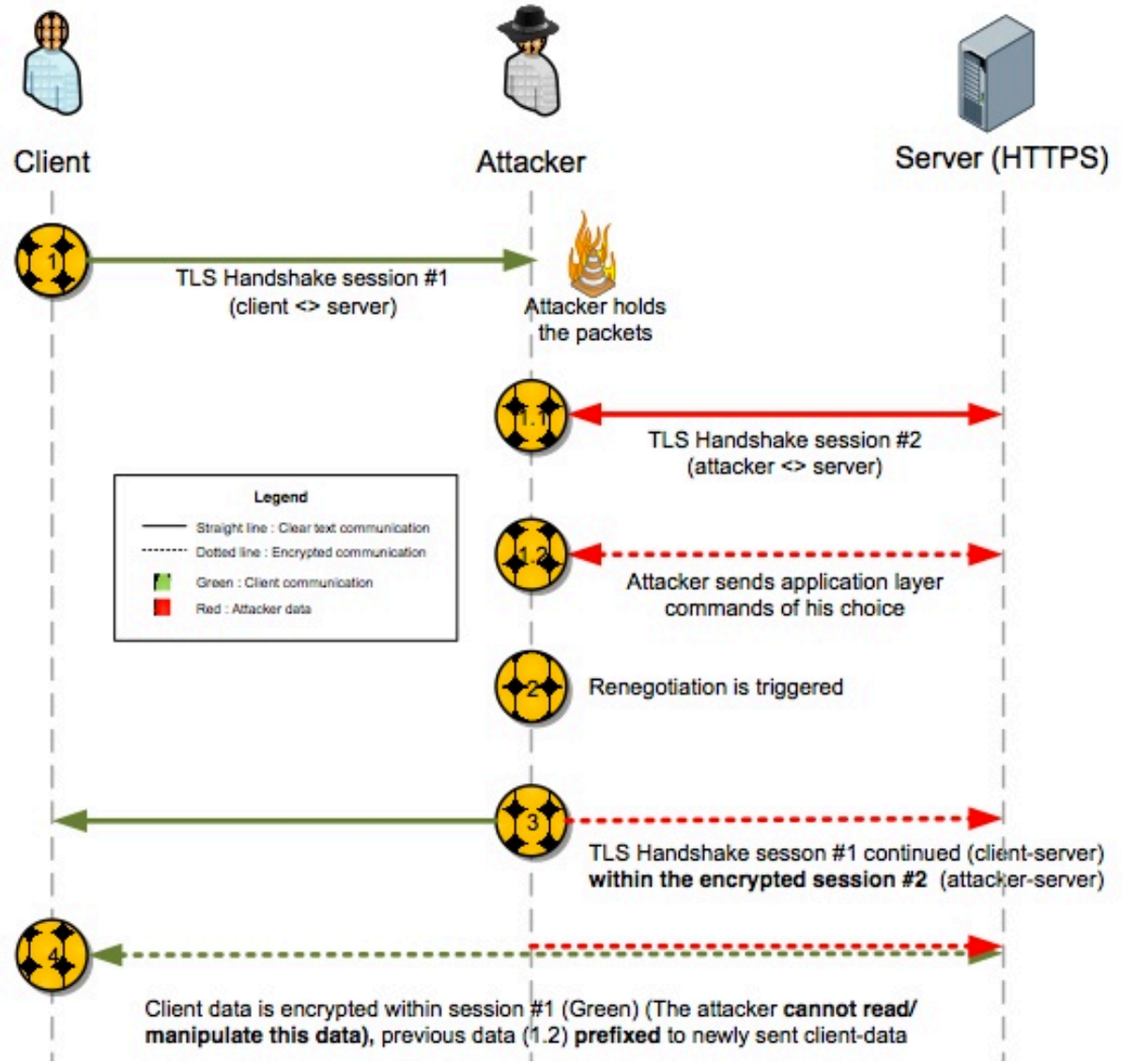
Einschub: TLS & SSLv3 renegotiation vulnerability

- Typische Ausgangssituation z.B. bei HTTPS:
  - Server kann erst nach Analyse der Client-Anfrage entscheiden, ob
    - Client-Zertifikat benötigt wird und/oder
    - stärkere kryptographische Verfahren eingesetzt werden sollen
  - Zu diesem Zeitpunkt besteht schon eine SSL-/TLS-gesicherte Verbindung ohne Client-Zertifikat bzw. mit zu schwachen Verfahren

- Server muss SSL-/TLS-Renegotiation anstoßen, um diesmal das Client-Zertifikat zu verlangen bzw. andere Algorithmen zu wählen.
- Nach erfolgreicher Renegotiation wird die ursprüngliche - nicht ausreichend abgesicherte - Client-Anfrage abgearbeitet.

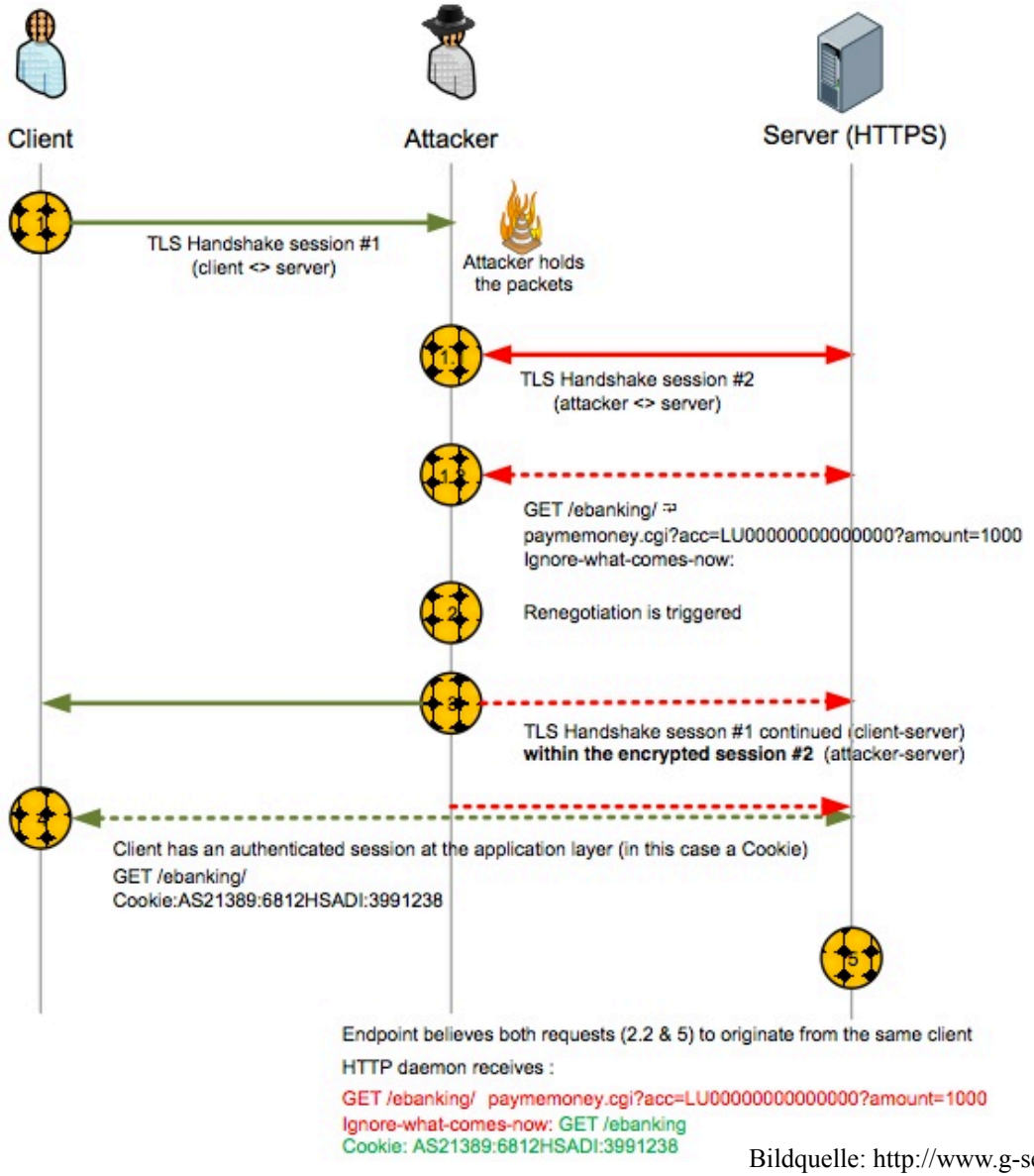
- „Man in the middle“ kann dies ausnutzen, um eigene Anfragen am Anfang der Verbindung einzuschleusen!

Einschub: Renegotiation attack - Allgemeiner Ablauf



Bildquelle: http://www.g-sec.lu/practicaltls.pdf

Einschub: Renegotiation attack - bei HTTPS



- Eigentliche Client-Anfrage wird durch vorangestelltes „X-Ignore-what-comes-now:“ neutralisiert.
- Dem Angreifer unbekannte Autorisierungsdaten (z.B. Cookies) werden vom Client geliefert.

Bildquelle: http://www.g-sec.lu/practicaltls.pdf

Einschub: Renegotiation vulnerability; Konsequenzen

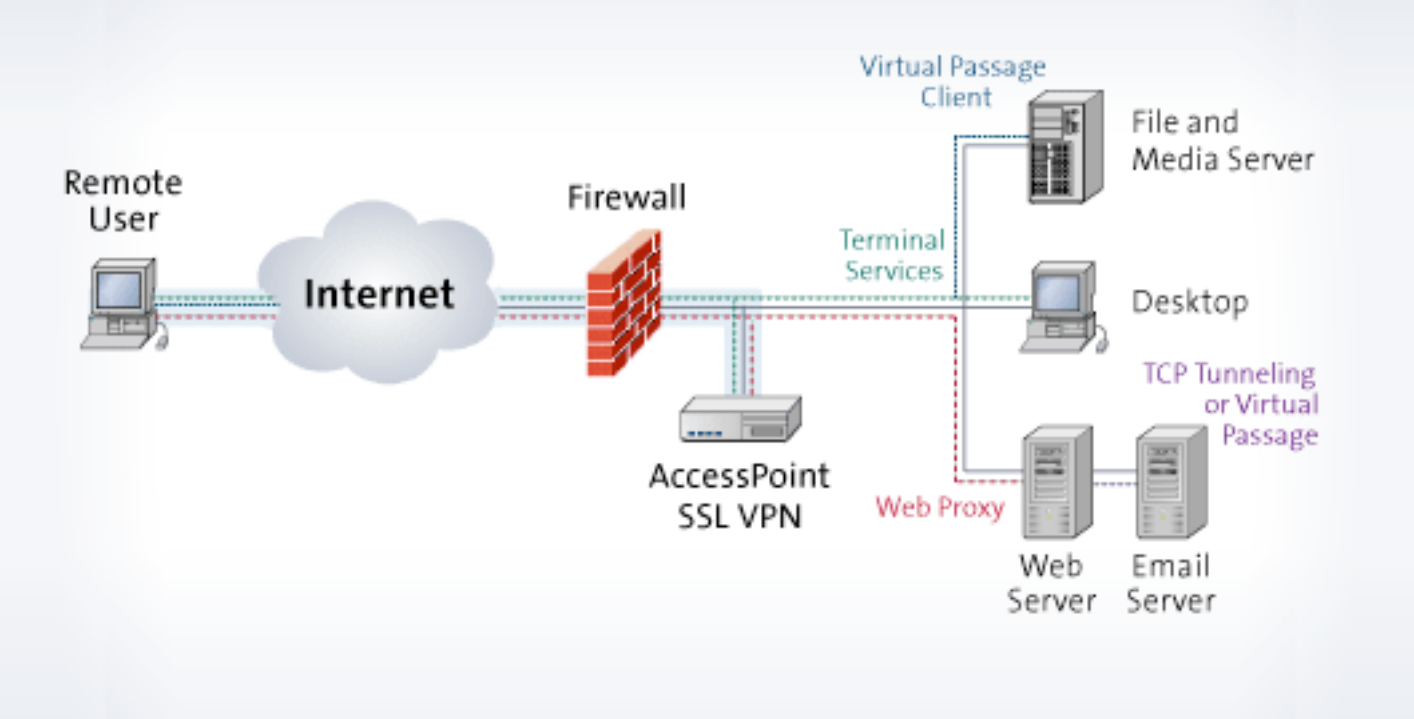
- Betrifft nicht nur HTTPS
  - Beispiel SMTP: Man-in-the-middle gibt Absender und Empfänger vor, erhält so z.B. die gesamte E-Mail samt Steuerkommandos vom Client
- Workaround:
  - Renegotiation per Konfiguration verbieten/deaktivieren
  - Problematisch, da inkompatibel zu den Anwendungen, die sie bislang nutzen
- IETF schlägt TLS Extension vor:
  - TLS-Sessions werden kryptographisch an Clients gebunden
  - Clients werden über Renegotiation informiert
  - In der initialen Aushandlung kann die Möglichkeit zur Renegotiation deaktiviert werden
  - Bereits in vielen Implementierungen kurzfristig umgesetzt

Anwendung: SSL-VPN

- Ziel wie z.B. bei PPTP und IPSec:
  Transport geschützter Daten über öffentliche Netze
- Wiederum Site-to-Site- und End-to-End-Einsatz möglich
- Häufige Eigenschaften in der Praxis:
  - Nutzung von TCP-Port 443 (offene Firewall-Ports)
  - „Schlanker“ Client, z.T. ohne manuelle Software-Installation
    - Wichtig insb. für mobile Geräte (private Notebooks, Smartphones, ...)
  - Früher z.T. nur zur Nutzung von Webanwendungen geeignet (kein „richtiges“ VPN)
  - Start der VPN-Verbindung per Login auf einer Webseite
  - Zusätzliche Funktionalität per Browser-Plugin
    - z.B. Virenscanner, Löschen des Cache bei Sitzungsende, ...

SSL-VPNs: Erste Generation

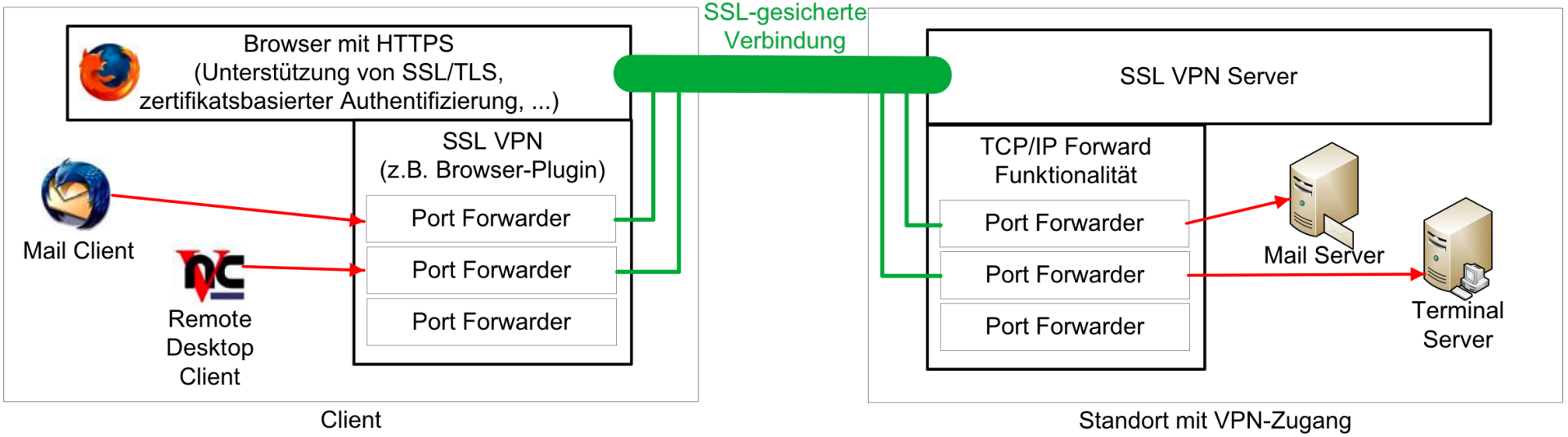
- Browserbasiert, einfache Inbetriebnahme
- Web Sessions für transparenten Client-IP-Adresswechsel (Roaming)



Bildquelle: www.menlologic.com

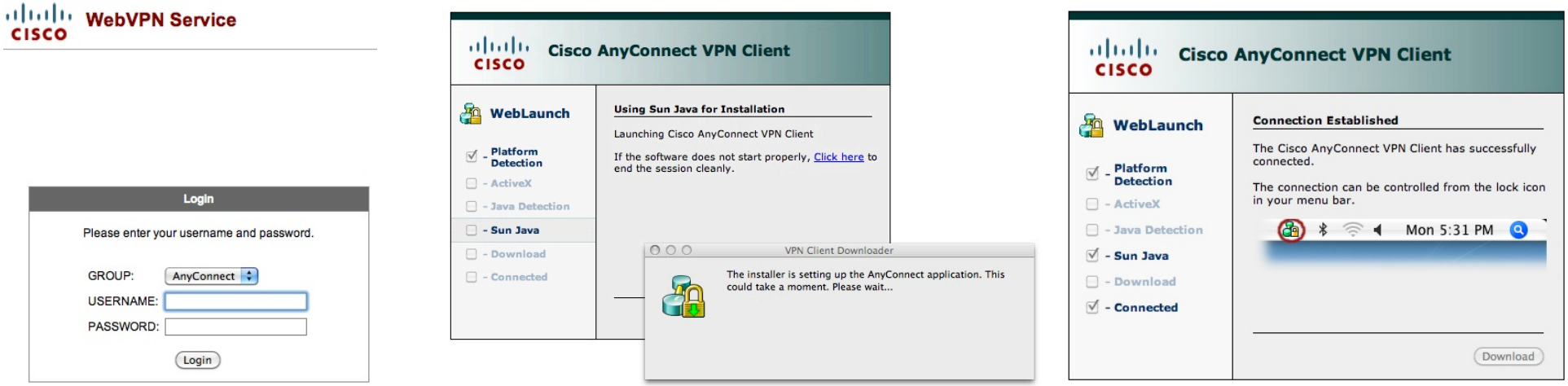
SSL-VPNs: Zweite Generation

- Web-basierte Installation eines „fat“ Clients
- Tunnel für TCP/IP-basierte Protokolle
- Z.T. Browser-Plugins für zusätzliche Funktionalität



SSL-VPNs: Aktuelle Trends

- IPSec zur Standort-Vernetzung
  - ausgereifte Produkte (Appliances)
  - zertifizierte, standardbasierte Lösungen (Compliance)
- SSL-VPN für mobile Anwender
  - schlanker Client
  - unkomplizierte Installation durch Anwender
- Beispiel Cisco AnyConnect SSL VPN Client (https://asa-cluster.lrz.de, nutzbar mit Campus^{LMU}-Kennung)



IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 13: Netzsicherheit - Schicht 7: Application Layer Secure Shell (SSH)

Inhalt

- SSH: Historie
- SSH v1 vs. SSH v2
- SSH-Protokollarchitektur
  - Transport Protocol
  - Authentication Protocol und Authentisierungsarten
    - None
    - Host based
    - Password
    - Public key
  - Connection Protocol
    - Interactive session
    - X11-Forwarding
    - Port-Forwarding
- Debian OpenSSL Debacle

Historie

- Entwickelt von Tatu Ylönen (TU Helsinki, Finland); 1995;
- Reaktion auf Passwort-Sniffing-Angriff an der TU Helsinki
- Ersatz für unsichere Unix-Tools: telnet, rlogin, rsh, rexec (ftp); (Übertragen Passworte im Klartext)
- Erste Version SSH-1 und Implementierung 1995, frei verfügbar
- Schnelle Verbreitung
- Kommerzielle Variante über SSH Communication Security; Gegründet Dez. 1995 von Ylönen
- Weiterhin freie Version (OpenSSH) verfügbar
- 1996 verbesserte Version SSH v2; inkompatibel zu SSH v1
- 2006 SSH v2 über IETF standardisiert [RFC 4251, RFC 4252, RFC 4253, RFC 4254, RFC 4255, RFC 4256]

SSH v1 vs. SSH v2

- Verbesserungen in SSH v2:
  - Diffie-Hellman Schlüsselaustausch
  - Verbesserte Integrität: Message Authentication Code (MAC) statt CRC32
  - Mehrere Shell-Sessions über eine SSH-Verbindung
- 2001: Mehrere Schwachstellen in SSH-1 gefunden:
  - RC4-verschlüsselte Passwörter können mitgelesen werden, falls beim Server leere Passwörter verboten sind
  - Replay-Angriff möglich
  - CRC32 erlaubt gezielte Modifikation der Nachrichten
  - CRC32 Attack detection code contains remote integer overflow; Angreifer kann Code mit Rechten des ssh-Damon (i.d.R. root) ausführen

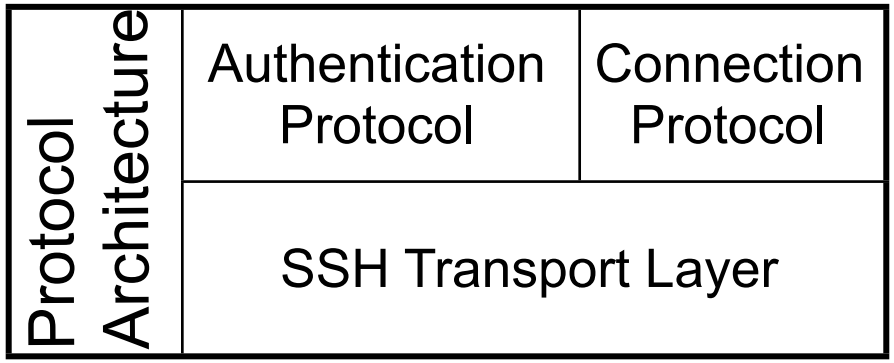
➡ SSH v1 ist unsicher, NICHT mehr verwenden

➡ Modus-Fallback auf SSH v1 server-seitig deaktivieren

➡ Im Folgenden nur SSH v2

SSH Architektur

- Geschichtete Architektur:



- SSH Protocol Architecture [RFC 4251]
- SSH Authentication Protocol [RFC 4252]
  - Authentisierung des Clients gegenüber dem Server („client-driven“)
- SSH Transport Layer Protocol [RFC 4253]
  - Authentisierung des Servers gegenüber dem Client
  - Vertraulichkeit, ggf. mit Kompression
  - Integrität
  - Perfect Forward Security, Re-Keying (oft nach 1 GB oder 1 h)
- SSH Connection Protocol [RFC 4254]
  - Multiplexing (mehrere logische Kanäle über eine SSH-Verbindung)

SSH Protocol Architecture

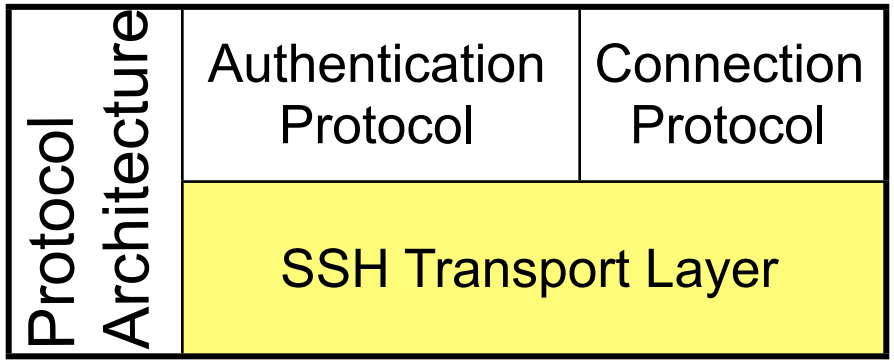
- Client (ssh) verbindet sich mit Host (Server, sshd)
- Jeder Host besitzt Schlüsselpaar
  - Zur Authentisierung gegenüber den Clients
  - Zwei Trust-Modelle:
    - Lokale Datenbasis mit Hostname - Public Key Paaren
    - Hostname und Public Key werden von CA zertifiziert, Client benötigt nur öffentlichen Schlüssel der CA
  - Client kann öffentlichen Schlüssel bei der ersten Verbindung zum Host in lokale Datenbasis übernehmen
    - ABER: Gefahr eines Man-in-the-Middle Angriffs
    - Sollte in der Praxis nicht verwendet werden, but „there is no widely deployed key infrastructure available on the Internet“
- Designprinzip: Erweiterbarkeit
  - Basisprotokoll so einfach wie möglich
  - Mindestsatz an zu unterstützenden Algorithmen (wg. Interoperabilität)
  - Erweiterungen v. Algorithmen, Formaten u. Protokoll möglich

Protocol Architecture: Sicherheit

- Algorithmen für jede Kommunikationsrichtung frei wählbar:
  - Verschlüsselung
  - Integritätssicherung
  - Schlüsselaustausch
  - Datenkompression
- Multiple Authentisierungsverfahren für jeden Client
- Server darf keine (eigene) Verbindung zum Client aufbauen
  - Ausnahme: Client fordert Forwarding bestimmter Dienste an
- Server darf keine Kommandos auf Client ausführen
- Perfect Forward Secrecy
  - Kompromittierung eines Session Key oder eines privaten Schlüssels darf nicht zur Kompromittierung einer früheren Session führen

SSH Architektur

- Geschichtete Architektur:

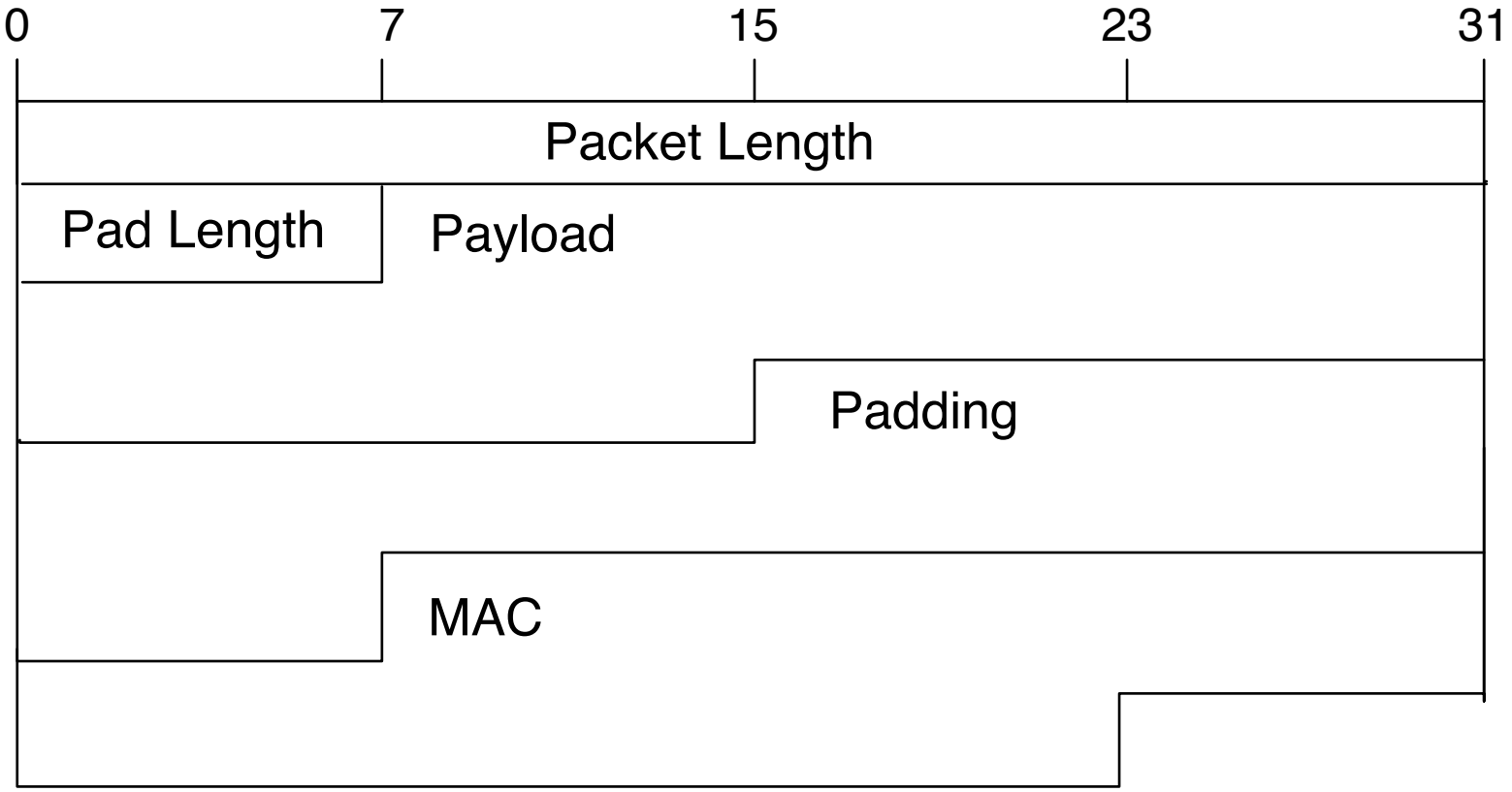


- SSH Authentication Protocol [RFC 4252]
  - Authentisierung des Clients gegenüber dem Server
- SSH Transport Layer Protocol [RFC 4253]
  - Authentisierung des Servers gegenüber dem Client
  - Vertraulichkeit
  - Integrität
  - Perfect Forward Security
- SSH Connection Protocol [RFC 4254]
  - Multiplexing

SSH Transport Protocol

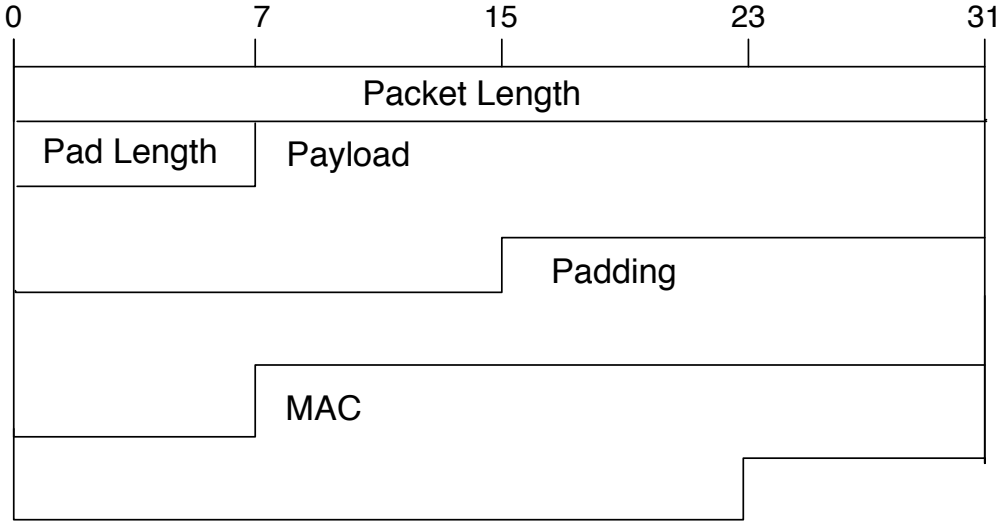
- Setzt auf verlässliches Schicht 4 Protokoll auf (i.d.R. TCP)
- Sicherheitsdienste:
  - Verschlüsselung der Nutzdaten
  - Integritätssicherung
  - Server-Authentifizierung (es wird (nur) der Host authentisiert)
  - Kompression der Nutzdaten
  - Aushandlung von Algorithmen
- Unterstützte Verschlüsselungsalgorithmen
  - 3DES, Blowfish, Twofish, AES, IDEA, CAST im CBC Modus
  - Arcfour (entspricht RC4, „has problems with weak keys“)
  - none („not recommended“)
- Hash-Algorithmen
  - HMAC-SHA1, HMAC-MD5, HMAC-SHA1-96, HMAC-MD5-96
  - HMAC-RIPEMD-160
- Asymmetrische Verfahren: RSA und DSA

Transport Protocol Packet Format



- Gesamtlänge von Packet Length inkl. Padding muss ein Vielfaches der Blocklänge oder ein Vielfaches von 8 ergeben
- Maximale Blocklänge: < 35.000 Byte

Transport Protocol Packet Format (Forts.)



- Packet Length = Pad Length + Payload + Padding
- Payload: Daten; falls Kompression aktiviert, wird dieses Feld mit zlib (deflate-Algorithmus, vgl. .gz-Dateien) komprimiert
- MAC über gesamtes Paket; berechnet vor Verschlüsselung
  - MAC = HMAC (gemeinsamer Schlüssel, SeqNr. | unencrypted Packet)
- Falls Verschlüsselung aktiviert, wird alles bis auf MAC verschlüsselt

Aushandlung der Algorithmen

- Schlüsselaushandlung erfordert explizite Authentisierung des Server-Hosts
  - Jeder Host besitzt mindestens 1 Schlüsselpaar in /etc/ssh ssh_host_dsa und ssh_host_dsa.pub oder ssh_host_rsa u. .._rsa.pub
- Jede Seite kann `kexinit` Nachricht schicken; enthält:
  - Cookie: Zufallszahl
  - jeweils komma-separierte Liste mit absteigender Priorität für:
    - kex_algorithm: Schlüsselaustausch
    - server_host_key_algorithm: Server listet Algorithmen, für die er Schlüsselpaare besitzt
    - encryption_client_to_server und server_to_client
    - mac_algorithm_client_to_server und server_to_client
    - compression_client_to_server und server_to_client
    - language_client_to_server und server_to_client: verwendete Sprache (Optional)
  - first_kex_packet_follows

Aushandlung der Algorithmen; Schlüsselaustausch

- Server analysiert die Client-Listen; gewählt wird erster Algorithmus, der auch vom Server unterstützt wird (!)
- Nach Abschluss der Aushandlung beginnt Schlüsselaustausch
- Optimierung: Jede Seite kann präferierten Algorithmus „raten“ und sofort das entsprechende key-exchange-Packet anhängen
  - richtig geraten: Folgepaket wird als key-exchange-Packet akzeptiert
  - falsch geraten: Paket wird verworfen

Schlüsselaustausch

- Schlüsselaustausch über Diffie-Hellmann mit SHA-1
  - Kombiniert mit digitaler Signatur durch Host -> Authentisierung
  - zwei definierte DH-Gruppen
- Ergebnis des Schlüsselaustausches: Secret K und Hash H

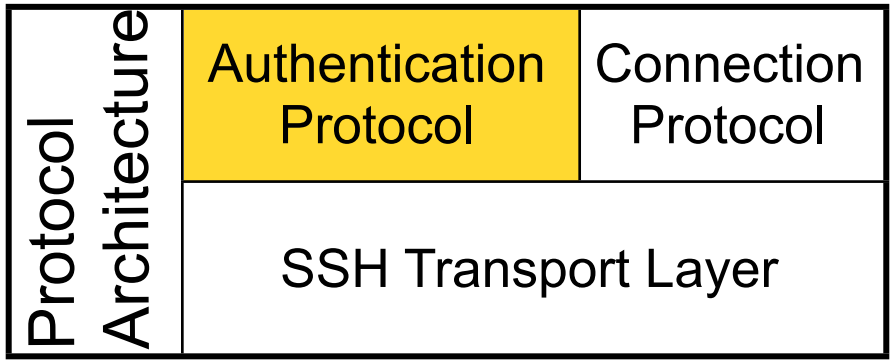
1. Client wählt x und sendet $e = g^x \bmod p$
2. Server wählt y und berechnet $f = g^y \bmod p$
   - 2.1. $K = e^y \bmod p$
   - 2.2. $H = \text{Hash}(V_C \mid V_S \mid I_C \mid I_S \mid K_S \mid e \mid f \mid K)$ mit
     V = Identität von C und S
     I = `kexinit` Nachrichten
     K_S = Public Host Key von S
   - 2.3. s = {H} (mit Private Host Key signiert)
   - 2.4. Übertragen wird (K_S | f | s)
3. Client verifiziert Signatur, berechnet K und H

Schlüsselmaterial

- Zum Ableiten des Keymaterials werden K und H verwendet:
  - IV Client -> Server = Hash (K | H | “A“ | session_id)
  - IV Server -> Client = Hash (K | H | “B“ | session_id)
  - Encryption Key Client ->Server = Hash (K | H | “C“ | session_id)
  - Encyption Key Server-> Client = Hash (K | H | “D“ | session_id)
  - Integrity Key Client -> Server = Hash (K | H | “E“ | session_id)
  - Integrity Key Server -> Client = Hash (K | H | “F“ | session_id)
- Daten werden immer vom Anfang des Hashes genommen
- Falls mehr Daten benötigt werden:
  K1 = Hash(K | H | x | session_id) mit x aus [“A“ ... “F“]
  K2 = Hash(K | H | K1)
  ...
  KX = K1 | K2 | ... | Kn
- Bis zu diesem Zeitpunkt weiß der Server nichts über den Client!

SSH Architektur

- Geschichtete Architektur:



- SSH Authentication Protocol [RFC 4252]
  - Authentisierung des Clients gegenüber dem Server
- SSH Transport Layer Security [RFC 4253]
  - Authentisierung des Servers gegenüber dem Client
  - Vertraulichkeit
  - Integrität
  - Perfect Forward Security
- SSH Connection Protocol [RFC 4254]
  - Multiplexing

SSH Authentication Protocol

- Zur Authentisierung des Nutzers
- Verwendet vertraulichen Kanal des SSH Transport Protocols
  - damit sichere Übertragung von Passwörtern o.ä. möglich

- Authentisierungsarten:
  - NONE:
  - Public-Key: muss von allen Implementierungen unterstützt werden. Private Key sollte passphrase-protected sein!
  - Password: Sollte unterstützt werden, überträgt ganzes Passwort
  - Hostbased: Optional
  - keyboard-interactive: Fragen-und-Antworten-Ablauf, z.B. für SecurID
  - GSS-API: Externe Authentifizierung, i.d.R. via Kerberos (single sign-on)

Authentication: Host Based

- Angelehnt an den .rhosts Mechanismus der r-Commands
- Vertrauenswürdige Hosts und Benutzernamen werden eingetragen in
  - /etc/hosts.equiv oder /etc/shosts.equiv
  - ~/.rhosts oder ~/.shosts
  - Bsp.: `testsrv.lrz.de reiser`
- Client-Host wird über Host-Key (vgl. Transport Protocol) authentisiert
- Damit Verhinderung von IP-, DNS-, oder routing-spoofing
- ABER:
  - Keine richtige Benutzerauthentisierung
  - Alle Nutzer auf der (Client-) Maschine können Server nutzen
- Sollte nicht verwendet werden

Authentisierung: Password

- Client fordert Nutzer zu Eingabe auf von
  - Benutzername
  - Passwort
- Validierung gegen (Server-) lokale Benutzerdaten
- Hinweis: Transport Layer bietet verschlüsselten Kanal
- Benutzername und Passwort dadurch vor Eve geschützt

Authentication: Public Key

- Nutzer muss Schlüsselpaar besitzen
  - ssh-keygen erzeugt Schlüsselpaar
  - Private Key mit Passphrase verschlüsselt
  - Private Key gespeichert in ~/.ssh/id_dsa oder id_rsa
  - Public Key in ~/.ssh/id_dsa.pub oder id_rsa.pub
  - Public Key auf Zielmaschine in ~/.ssh/authorized_keys kopieren
- Client signiert Nachricht
- Verifikation durch Server
- Sicherste Variante der Authentisierung
- ABER:
  - Leere Passphrase für Private Key erlaubt
  - Schutz ist dann einzig durch Dateisystemberechtigungen gegeben
  - Falls Maschine kompromittiert, kann der Angreifer in diesem Fall private Schlüssel zum Login verwenden
- KEINE leere Passphrase verwenden

Einschub: RSA-768 geknackt (01/2010)

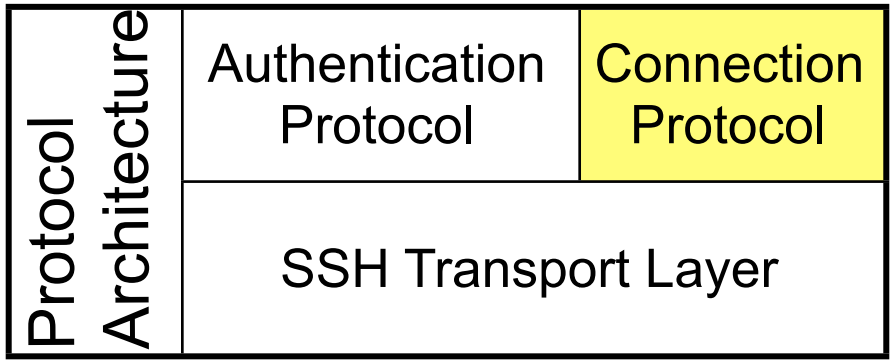
- Details unter http://eprint.iacr.org/2010/006.pdf
- 768 Bit (232 Dezimalstellen) lange Zahl in ihre Primfaktoren zerlegt
- Zahl wurde im Rahmen der RSA-Challenge als RSA-768 veröffentlicht
- RSA-Schlüssel mit „nur“ 768 gelten somit als geknackt

- Dauer: Ca. 2,5 Jahre
  - Erster Rechenschritt (Polynomial Selection) ca. 0,5 Jahre auf 80 PCs
  - Zweiter Rechenschritt (Sieving) ca. 2 Jahre auf Cluster mit mehreren hundert Rechnern
  - Eine aktuelle AMD Server CPU hätte alleine rund 1.500 Jahre benötigt

- Prognose: Genug Rechenleistung zum Knacken von RSA-1024 in 10 Jahren einfach verfügbar.

SSH Architektur

- Geschichtete Architektur:



- SSH Authentication Protocol [RFC 4252]
  - Authentisierung des Clients gegenüber dem Server
- SSH Transport Layer Security [RFC 4253]
  - Authentisierung des Servers gegenüber dem Client
  - Vertraulichkeit
  - Integrität
  - Perfect Forward Security
- SSH Connection Protocol [RFC 4254]
  - Multiplexing

SSH Connection Protocol

- Setzt auf SSH Transport Protocol auf
- Dienste:
  - Interaktive Login Session (`ssh reiser@testsrv.lrz.de`)
  - Entfernte Ausführung von Kommandos (`ssh testsrv.lrz.de ps`)
  - X11-Forwarding
  - Statisches Forwarding von einzelnen TCP/IP - Ports (lokal / remote)
  - Inzwischen auch dynamisches Forwarding auf SOCKS5-Basis
- Für jeden Dienst wird ein oder mehrere Channels aufgebaut
- Multiplexing aller Channels über einen SSH Transport Protocol Kanal
  - Damit Vertraulichkeit und
  - Integritätssicherung aller übertragenen Daten

SSH Connection Protocol: Aufbau v. Channels

- Jede Seite kann SSH_MSG_CHANNEL_OPEN schicken
- Nachrichtenformat
  - channel type: String der Art des Kanals („session“, „X11“, usw.)
  - sender channel: 32 Bit Integer als lokaler Identifikator des Channels
  - initial window size: 32 Bit Integer; maximale Anzahl Bytes, die an den Initiator gesendet werden können
  - maximum packet size: 32 Bit Integer; Maximale Paketgröße
  - ... weitere Parameter abhängig vom channel type
- Ablehnen des Channel Request mit SSH_MSG_CHANNEL_OPEN_FAILURE:
  - recipient channel: channel id des Senders
  - reason code: Grund der Ablehnung (z.B. adminstratively prohibited, connect failed, unknown channel, ...)
  - additional textual information
  - language tag

Aufbau von Channels (cont.)

- Akzeptieren des Channel Request mit SSH_MSG_CHANNEL_OPEN_CONFIRMATION:
  - recipient channel: channel id des Senders
  - sender channel: channel id des Responders
  - initial window size
  - maximum packet size
  - ... weitere Parameter abhängig vom channel type
- Falls Channel aufgebaut, sind folgende Aktionen möglich
  - Datentransfer (Empfänger muss wissen, was er mit den Daten tun soll, d.h. ggf. weitere Aushandlung erforderlich)
  - Channel-type spezifische Nachrichten
  - Schließen des Channels

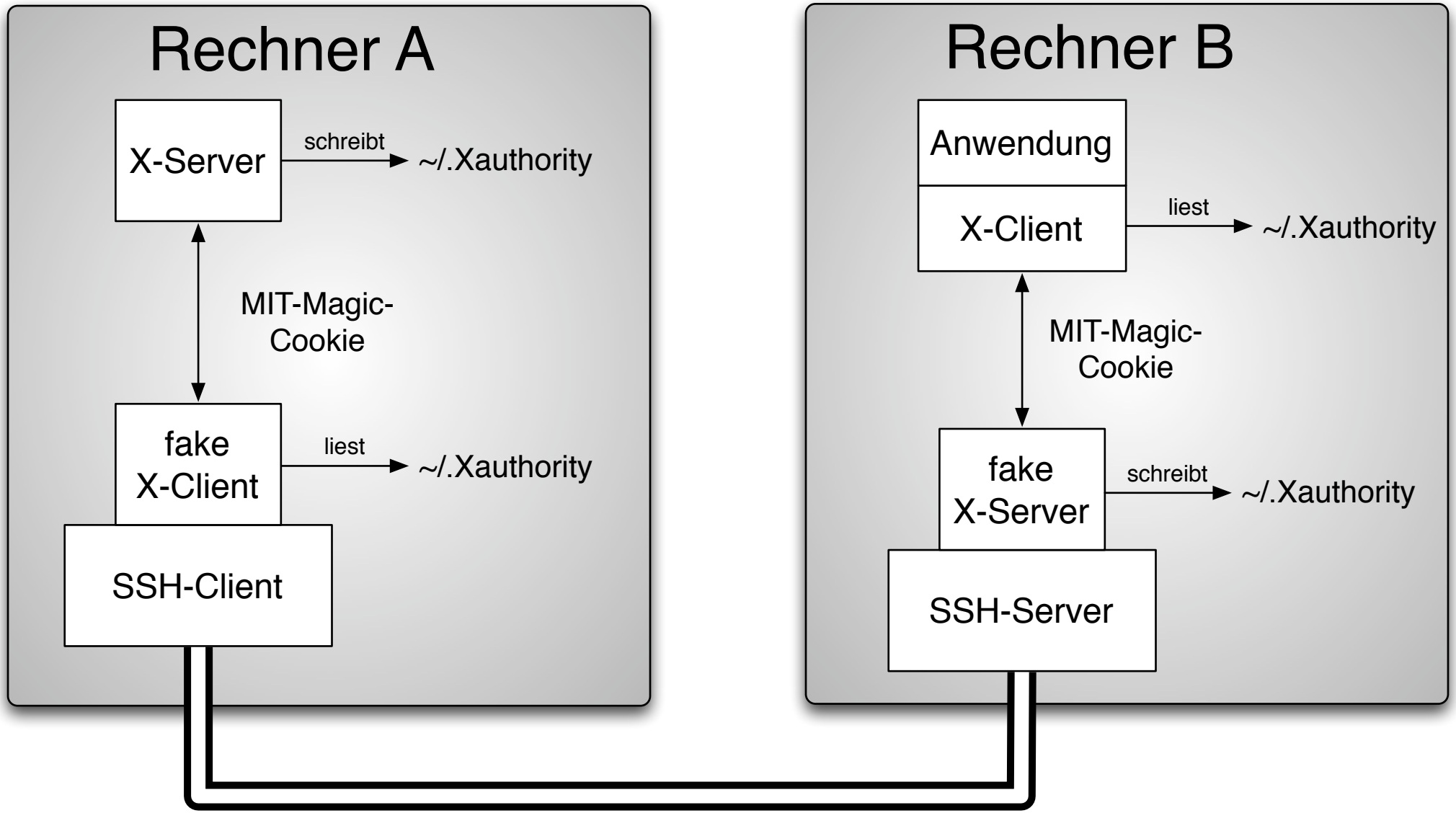
Interaktive Login Session

- SSH_MSG_CHANNEL_OPEN mit type „session“
- Nach Channel-Aufbau Anforderung eines Pseudo-Terminals; SSH_MSG_CHANNEL_REQUEST:
  - recipient channel
  - „pty-request“: Anforderung des Pseudo-Terminals (PTY)
  - want_reply: Boolsche Variable ob Antwort erforderlich
  - Term Environment Variable: Art des PTY (z.B. vt100, xterm,....)
  - Terminal width und height characters: Anzahl der Zeichen
  - Terminal width und height pixel: Anzahl der Pixel
  - Terminal modes: Betriebsarten des Terminals
- Umgebungsvariablen setzten mit SSH_MSG_CHANNEL_REQUEST mit type „env“
  - Paare mit Variablenname und Variablenbelegung
- Starten einer Shell mit SSH_MSG_CHANNEL_REQUEST mit type „shell“ (Default shell aus /etc/passwd wird gestartet)

X11

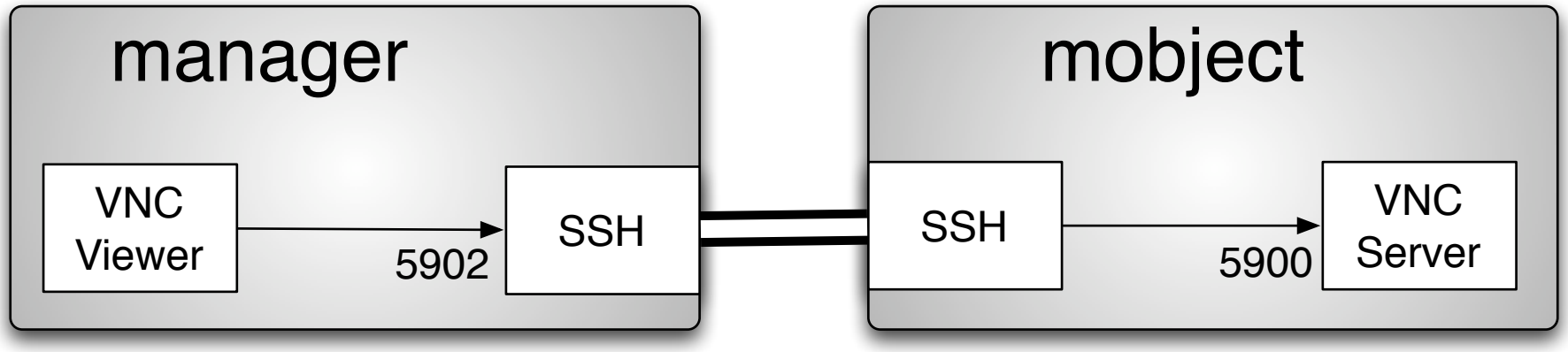
- X11 Window System im Rahmen von MIT Athena Projekt entwickelt
- X-Server läuft auf dem Rechner, bei dem die Grafikausgabe bzw. -eingabe erfolgen soll
- Programm, das Ausgabe macht, verbindet sich als X-Client
- X-Server (X-Display) und X-Clients (Anwendungen) können auf getrennten Maschinen laufen
- Nachrichten werden im Klartext übertragen, d.h. X11 hat selbst keine Mechanismen bzgl. Vertraulichkeit

SSH: X11-Forwarding



SSH: Port Forwarding

- Port auf einer lokalen Maschine wird zu einem Port auf entferntem System über SSH getunnelt
- Bsp.: Sichere Remote Administration über VNC und SSH
- VNC-Server wartet auf Port 5900
- VNCViewer erwartet Display-Nummer; Display Nummern n werden auf Port 5900 + n abgebildet
- ssh -L 5902:mobject:5900 reiser@mobject
- VNCViewer localhost:2



Debian OpenSSL Debacle

- OpenSSL: Freie Bibliothek zur Implementierung von SSL / TLS
- Wird genutzt von vielen Linux Derivaten, OpenSSH, Apache (mod_ssl), Onion Router (TOR), OpenVPN, etc.
- 2006: Code Analyse-Tools Valgrind und Purify erkennen potentielle Schwachstellen im Quellcode
  - Uninitialisierter Speicher
  - wird gelesen bevor er geschrieben wird
- Maintainer berichtigen diesen „Bug“ mit einem Patch
  - zwei Zeilen werden gelöscht `MD_Update(&m, buf, j)`
- Code-Abschnitt wurde verwendet, um die Entropie des Zufallszahlengenerators (PRNG) zu verbessern
- Patch bewirkt das Entropie nur noch von Prozess-ID (PID) abhängt
- In vielen Linux-Systemen 15 Bit, d.h. Entropie von 2^{15}

Debian OpenSSL Debacle Folgen

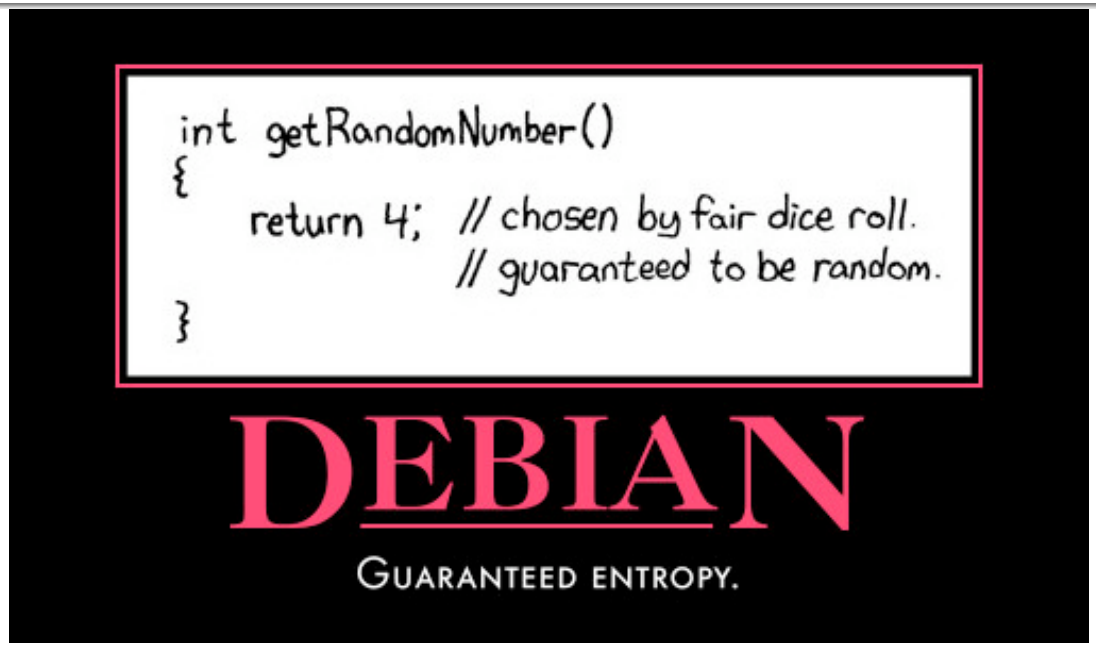
- Schlüsselraum von 32.767 Schlüsseln für jede Schlüssellänge und Schlüsseltyp
- 2008 von Luciano Belo (Argentinischer Forscher) entdeckt
- Erzeugung aller 1.024 und 2.048 Bit Schlüssel durch Moore:
  - Cluster mit 31 Xeon Cores
  - Zwei Stunden
  - Zusätzlich 6 Stunden für alle 4.096 Bit Schlüssel
- Betroffene Schlüssel:
  - SSH Host Keys
  - Benutzerschlüssel für Public Key Authentication in SSH
  - Sitzungsschlüssel
- SSH Perfect Forward Security (PFS)
  - Debian Bug verhindert PFS
  - Selbst wenn Maschine sicheren Zufallszahlengenerator besitzt

Debian OpenSSL Debacle Folgen

- Schwacher Schlüssel kann einfach genutzt werden
  - ssh -i weak-key4586 root@targetmachine
  - Zertifikate sind einfach zu fälschen
- Damit Spoofing
  - SSL-gesicherte Web-Seiten
  - SSH
  - Bsp.: AKAMAI Server, verteilte
    - Elster (Software für Steuererklärungen)
    - Treiber von ATI
- Entschlüsselung des Verkehrs

OpenSSL Debacle: Gegenmaßnahmen

- Alle Schlüsselpaare prüfen:
  - Skript (security.debian.org/project/extra/dowkd/dowkd.pl.gz)
- Firefox oder Internet Explorer Plugin zur Überprüfung
- Wiederruf von Zertifikaten mit schwachen Schlüsseln
  - Problem: CRLs oder OCSP wird oft nicht genutzt



Quelle: http://trailofbits.files.wordpress.com/2008/07/hope-08-openssl.pdf

IT-Sicherheit

- Sicherheit vernetzter Systeme -

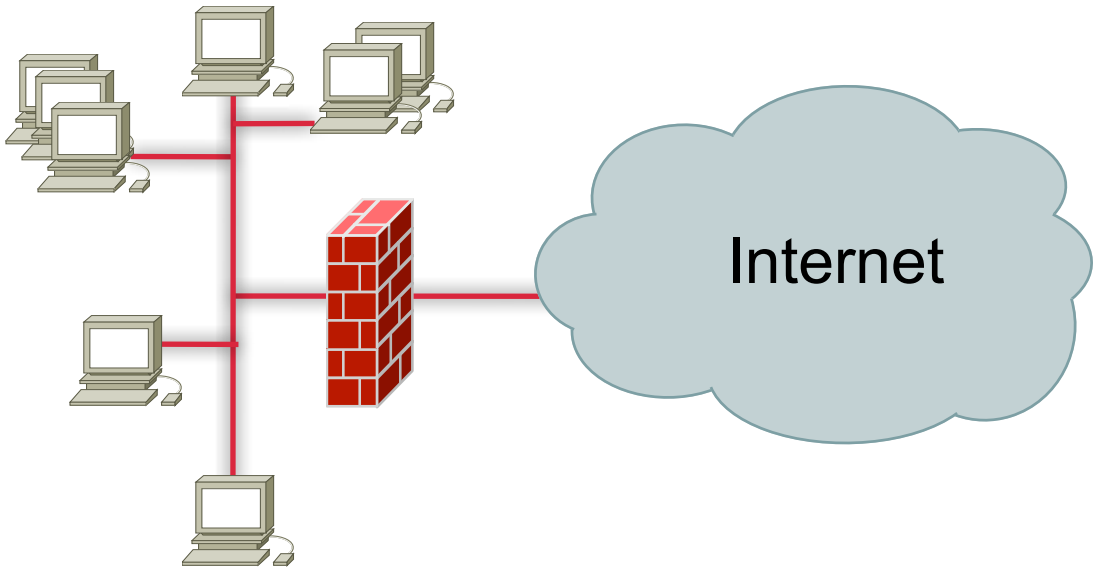
Kapitel 14: Firewalls und Intrusion Detection Systeme (IDS)

Inhalt

- Firewall-Klassen
  - Paketfilter
  - Application Level Gateway
  - Verbindungsgateway (Circuit Level Gateway)
- Firewall-Architekturen
  - Single Box
  - Screened Host
  - (Multiple) Screened Subnet(s)
- Intrusion Detection Systeme
  - Hostbasierte IDS (HIDS)
  - Netzbasierte IDS (NIDS)
  - Beispiele

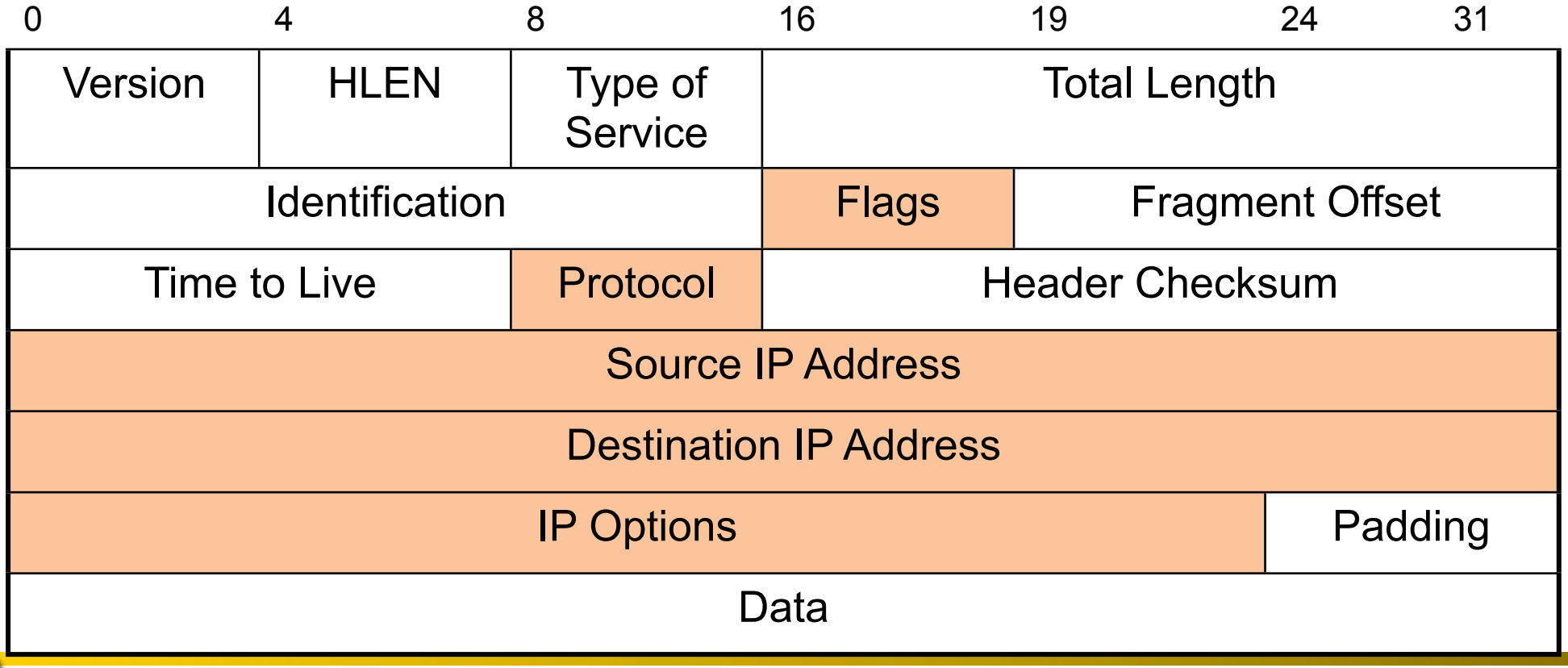
Firewall-Techniken

- Firewall:
  - Besteht aus einer oder mehreren Hard- und Softwarekomponenten
  - Koppelt (mindestens) zwei Netze als kontrollierter Netzübergang
  - Gesamter Verkehr zwischen den Netzen läuft über die Firewall (FW)
  - Realisiert Sicherheitspolicy bezüglich Zugriff, Authentisierung, Protokollierung, Auditing,....
  - Nur Pakete, die den Policies genügen, werden „durchgelassen“
- Klassen:
  - Paketfilter
  - Applikationsfilter (Application Level Gateway)
  - Verbindungsgateway (Circuit Level Gateway)
  - Kombinationen daraus



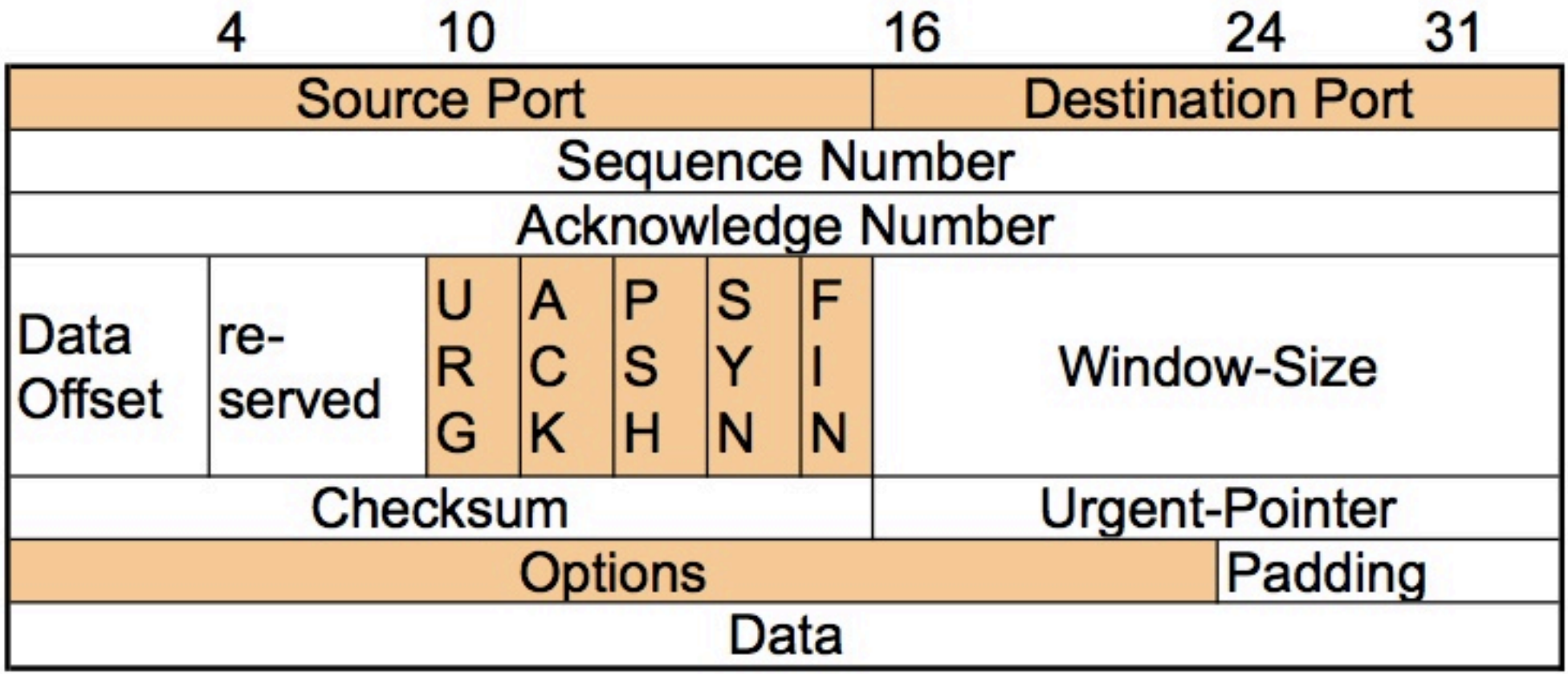
Firewall-Klasse „Paketfilter“

- Filtern auf OSI-Schichten 3 und 4
- Filterinformationen aus den Protokollpaketen
- Im Folgenden beispielhaft TCP/IP
- IP-Paket:



Paketfilter: TCP-Paketformat

- Bei Paketfilter-FW nur Regeln bzgl. Feldern der Paket-Header möglich!



FW für TCP/IP: Granularität der Filter

- Schicht 3: wesentliche Filter-Kriterien:
  - Quell-
  - Zieladresse
  - Protokoll der Transportschicht (z.B. TCP, ICMP, UDP,... vgl. `/etc/protocols`)
- FW auf IP-Basis kann damit Endsysteme filtern (erlauben, verbieten)
- IP-Spoofing u.U. erkennbar, falls:
  - Paket mit interner Quell-Adresse
  - kommt an externem FW-Interface an
- Keine Filterung auf Ebene der Dienste möglich

- Schicht 4: wesentliches Filter-kriterium:
  - Quell- sowie
  - Zielport
  - Flags
- Über Port-Nr. werden „well-known“ Services identifiziert;
  z.B. Port 22 = SSH
  (vgl. `/etc/services`)
- Allerdings nur Konvention; OS setzt diese nicht automatisch durch
- Weitere Konventionen:
  - privilegierte Ports (Ports < = 1023) nur für Systemdienste (unter UNIX)
  - Ports > 1023 für jeden nutzbar
- Flags zum Verbindungsaufbau und -abbau (vgl. Kap. 3 SYN Flooding)

Paketfilter: Filterregeln

- Grundsätzliche Ansätze:
  1. Alles, was nicht explizit verboten ist, wird erlaubt. (Blacklist)
  2. Alles, was nicht explizit erlaubt ist, wird verboten. (Whitelist)
- Reihenfolge der Regeln wichtig:
  - Regel, die zuerst zutrifft, wird ausgeführt („feuert“)
  - Daher ist im 2. Fall die letzte Regel immer: „alles verbieten“
- Statische Paketfilterung
  - Zustandslos
  - Pakete werden unabhängig voneinander gefiltert
- Dynamische Paketfilterung (stateful inspection)
  - Zustandsabhängig
  - FW filtert abhängig vom Zustand des Protokoll-Automaten
- Grundsatz: KISS
  Keep it Small and Simple

Paketfilter-Regeln: Beispiele

- Statischer Paketfilter:
  - Ausgehender Telnet-Verkehr erlaubt,
  - Eingehender Telnet-Verkehr verboten

Regel	Source	Destination	Protocol	Source Port	Dest. Port	Flags	Action
1	<intern>	<extern>	TCP	>1023	23	Any	Accept
2	<extern>	<intern>	TCP	23	>1023	!SYN	Accept
3	Any	Any	Any	Any	Any	Any	Drop, Log

- Dynamischer Paketfilter

Regel	Source	Destination	Protocol	Source Port	Dest. Port	Action
1	<intern>	<extern>	TCP	>1023	23	Accept
2	Any	Any	Any	Any	Any	Drop, Log

Bewertung: Paketfilter

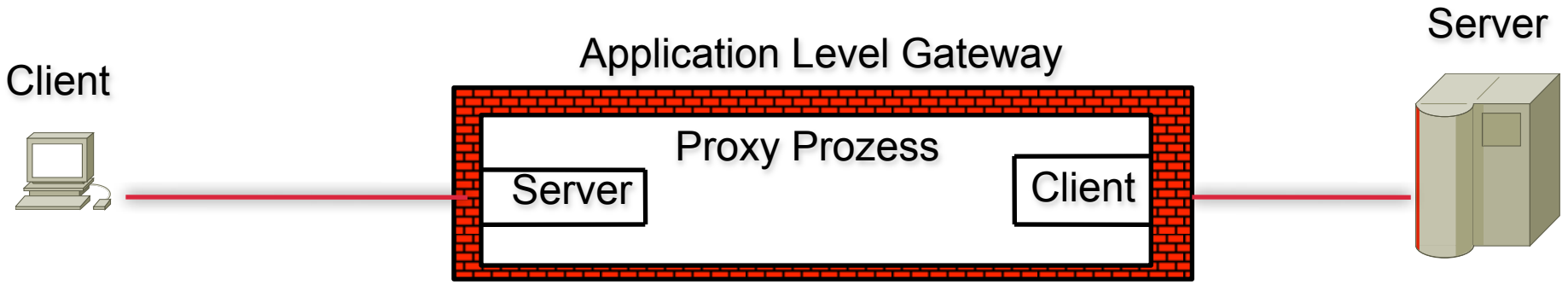
- Einfach und preiswert
- Effizient
- Gut mit Router-Funktionalität kombinierbar (Screening Router)
  - Integrität der Header-Informationen nicht gesichert; alle Felder können relativ einfach gefälscht werden
  - Grobgranulare Filterung
  - Keine inhaltliche Analyse bei freigegebenen Diensten
  - Statische Filterung hat Problem bei Diensten, die Ports dynamisch aushandeln (z.B. Videokonferenz-Dienst, FTP)
  - Abbildungsproblematik: Policy (Prosa) auf konkrete FW-Regeln
  - Erstellung einer Filtertabelle nicht triviale Aufgabe
    - Korrektheit ?
    - Vollständigkeit ?
    - Konsistenz ?

Weitere Firewall-Techniken auf Schicht 3/4

- Network Adress Translation (NAT)
  - Intern werden andere Adressen („private IP-Adr.“) oder Ports als extern verwendet
  - FW erledigt Adress/Port-Umsetzung
  - Statisch (SNAT) oder dynamisch (DNAT, Masquerading)
- Masquerading
  - Alle ausgehenden Pakete erhalten Adresse der FW
  - Gesamtes internes Netz wird verborgen
- Anti-Spoofing
  - Binden von FW-Regeln an bestimmte Interfaces (ingress, egress)
  - Wenn an externem Interface ein Paket mit interner Quell-Adresse ankommt, muss diese gefälscht sein

Applikationsfilter (Application Level Gateway)

- Filtern auf Schicht 7 (Anwendungsschicht)
- Analyse des Anwendungsschichtprotokolls u. d. Nutzdaten
- Für **jeden** Dienst, jedes Protokoll ist ein eigener Filterprozess (auch als **Proxy** bezeichnet) erforderlich
- Interner Client muss sich i.d.R. am Proxy authentisieren
- Proxy trennt Verbindung zwischen Client und Server
- Nach außen erscheint immer nur die Adresse des Application Level Gateways; völlige Entkoppelung von internem und externem Netz
- ALG kann Zustandsinformationen halten und nutzen



Ablauf einer Verbindung über ALG



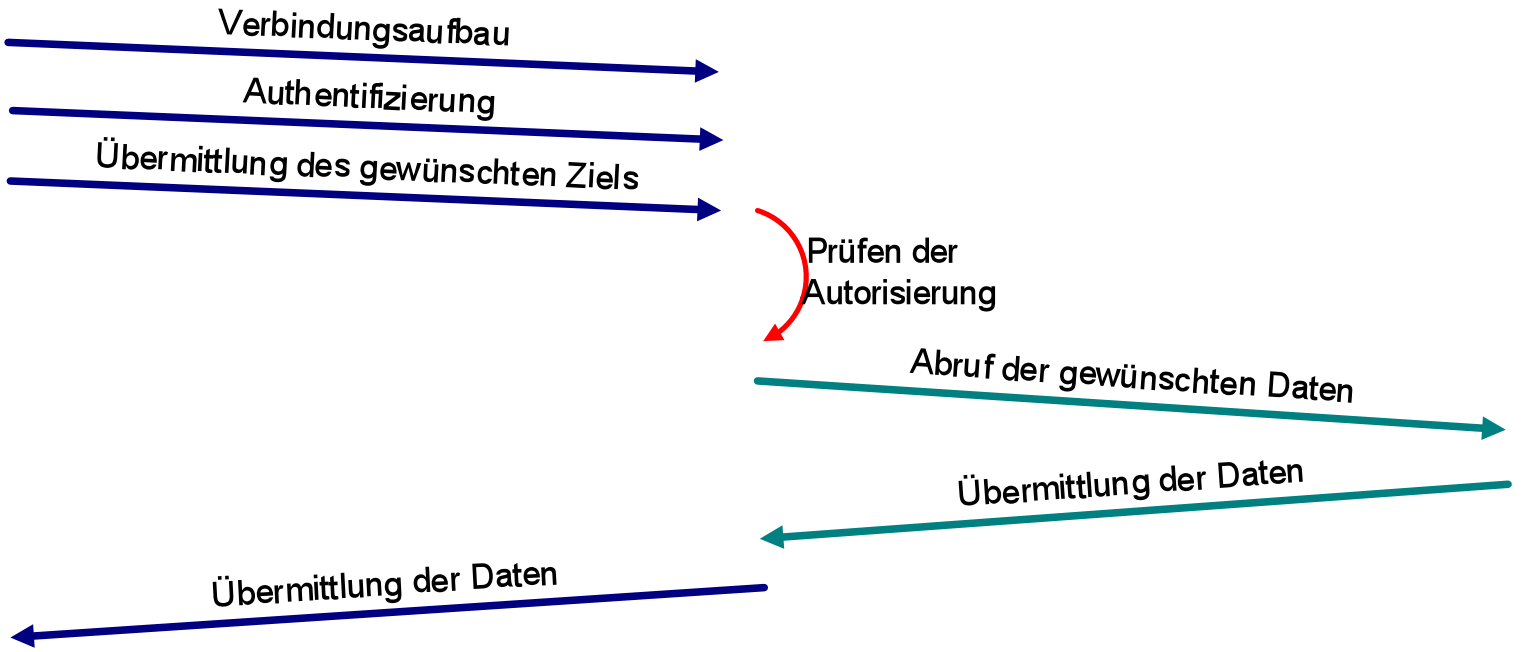
Client:
Browser



Application
Level Gateway
für HTTP



Webserver

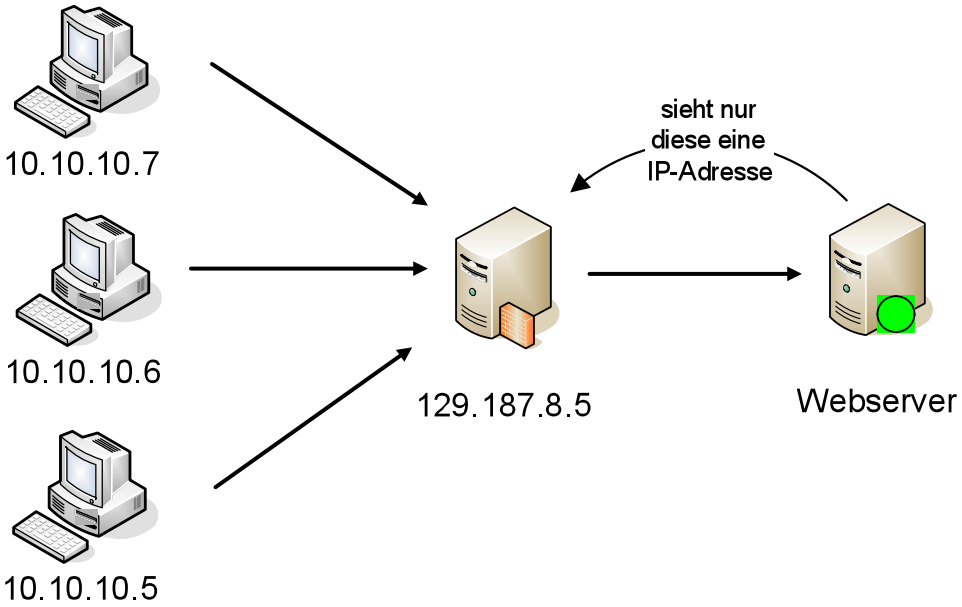


Proxies

- Für viele Standarddienste verfügbar (z.B. HTTP, Telnet, FTP,..)
- Oft problematisch für „proprietäre“ Protokolle (SAP R3, Lotus Notes,....)
- Beispiel eines HTTP-Proxys: Squid
  - Umfangreiche Access Control Listen (ACL) möglich:
    - Quell- / Zieladresse
    - Domains
    - Ports
    - Content (Nutzdaten-Analyse)
    - Protokoll-Primitive (z.B. FTP `put`, HTTP `POST`)
    - Benutzernamen, Datenvolumen, Uhrzeit, ...
  - Benutzerauthentisierung am Proxy
  - Zusätzlich Caching-Funktionalität
  - Beispiel:
    - `acl SSL_PORT port 443` (Definition von SSL Ports)
      `acl AUTH proxy_auth REQUIRED` (Benutzerauthentisierung)
      `http_access deny CONNECT !SSL_PORT` (Alle Verbindungen zu einem anderen Port außer SSL (HTTPS) verbieten)

Auswirkungen des ALG-Einsatzes

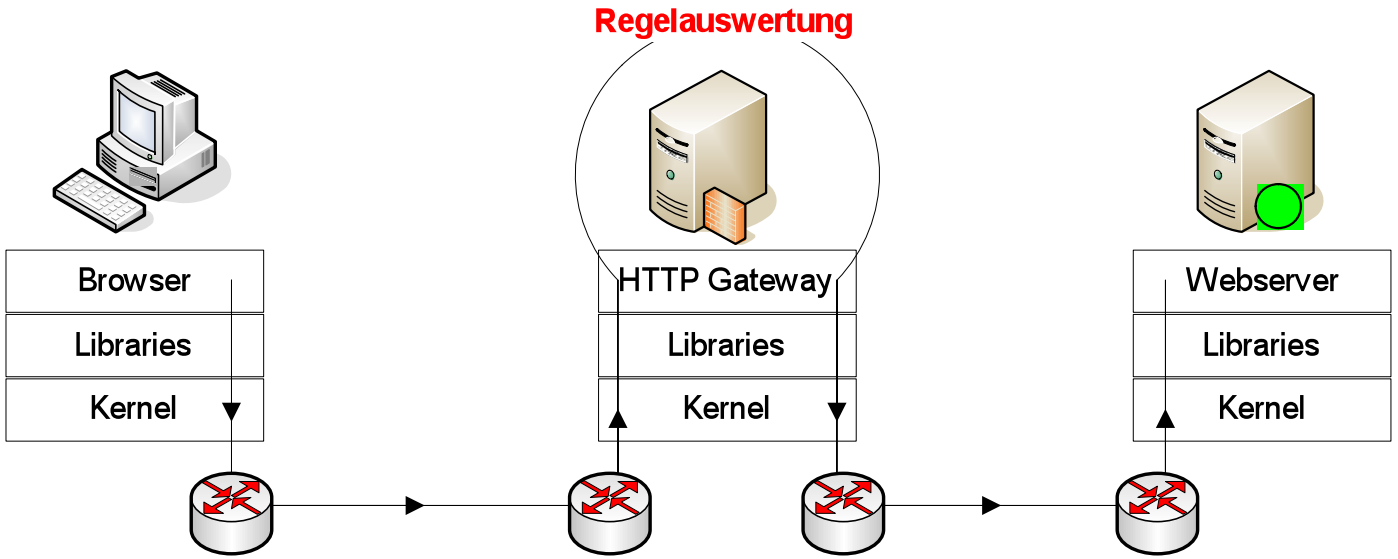
- Auf die Netz-Infrastruktur:
  - IP-Adressen der Clients werden hinter dem Gateway „versteckt“
  - Kein direktes Routing mehr (IP-Forwarding auf Gateways deaktivieren!)



- Auf die Clients:
  - Mindestens theoretisch schlechtere Performance als bei Paketfiltern
  - Client muss sich explizit an den Proxy wenden
    (Alternative: „Transparent Mode“ mit nur schwacher Authentifizierung)

Performance beim ALG-Einsatz





Automatische Konfiguration der Clients über PAC

- Web-Browser rufen Proxy-Einstellungen selbst ab:

```
function FindProxyForURL(url, host) {
    // Kein Proxy fuer LRZ-Seiten
    if (shExpMatch(url,"*.lrz.de/*")) { return "DIRECT"; }

    // URLs in einem bestimmten Netz laufen ueber einen dedizierten Proxy
    if (isInNet(host, „192.168.0.0", "255.255.255.0")) { return "PROXY squid.secp:3128" }

    // Alle anderen Anfragen gehen über Port 8080 von proxy.example.com.
    return "PROXY proxy.example.com:8080“;
}
```

- Nutzung wird (z.B. in Firmennetzen) oft erzwungen
  - Ports 80/443 nur für Proxy freigeschaltet
  - Alle anderen Verbindungen werden auf Informations-Webseite umgeleitet

Bewertung Applikationsfilter

- Feingranulare, dienstspezifische Filterung
- Umfangreiche Logging-Möglichkeit und damit Accounting
- Zustandsbehaftet
- Inhaltsanalyse (damit z.B. Filterung aktiver Inhalte möglich)
- Benutzerauthentisierung und benutzerabhängige Filterung
- Entkopplung von internem und externem Netz
- Möglichkeit der Erstellung von Nutzungsprofilen

- Möglichkeit der Erstellung von Nutzungsprofilen
- Jeder Dienst braucht eigenen Proxy
- Sicherheit der Proxy-Implementierung und -Konfiguration?
- Problem von Protokollschwächen bleibt weitgehend bestehen

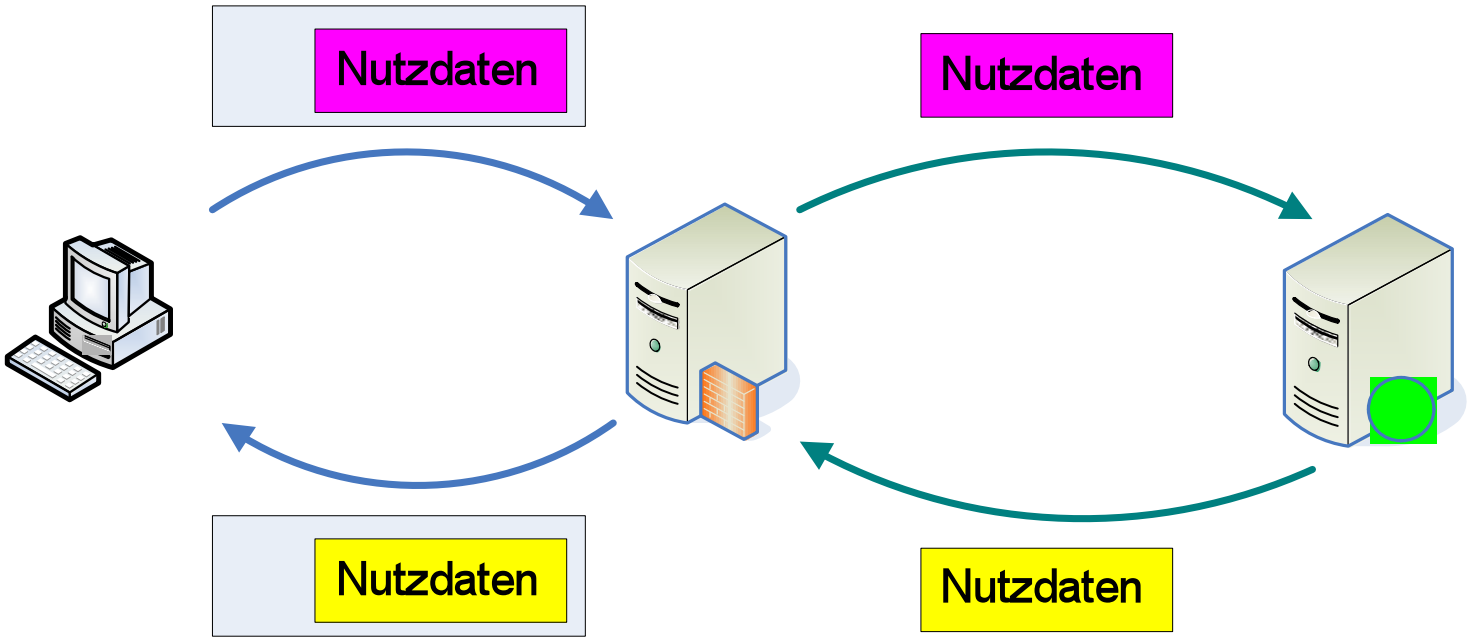
Content Filtering mittels ALG

- Proxy analysiert die Nutzdaten
- Unerwünschter Inhalt wird herausgeschnitten (Filtering) oder der gesamte Inhalt wird verworfen (Blocking)
- Beispiele:
  - FTP-Downloads erlauben, aber Uploads verbieten
  - Unerwünschte Webinhalte (HTML-Content via HTTP) unterdrücken
  - Häufige Tippfehler beim Mailversand (via SMTP) korrigieren
- Häufig Nutzung zentral gepflegter Rating-Systeme / Blacklists

Verbindungs-Gateway (Circuit Level Gateway)

- Filtert auf Schicht 3/4
- Circuit Level Gateway stellt generischen Proxy dar
  - CLG auch als „Multiprotokollproxy“ bezeichnet
- Nicht auf einzelne Dienste zugeschnitten, allgemeiner „Vermittler“ von IP-Verbindungen
- Trennt Verbindung zwischen Client und Server
- Benutzerauthentisierung am Gateway möglich
- Bsp. `SOCKS`:
  - SOCKS-Server filtert den TCP/IP Verkehr
  - Alle Verbindungen der Clients müssen über SOCKS-Server laufen
  - Daher Anpassung der Clients erforderlich („socksifying“)
  - Filtert nach: Quelle, Ziel, Art des Verbindungsaufbaus (z.B. Initiierung oder Antwort), Protokoll, Benutzer

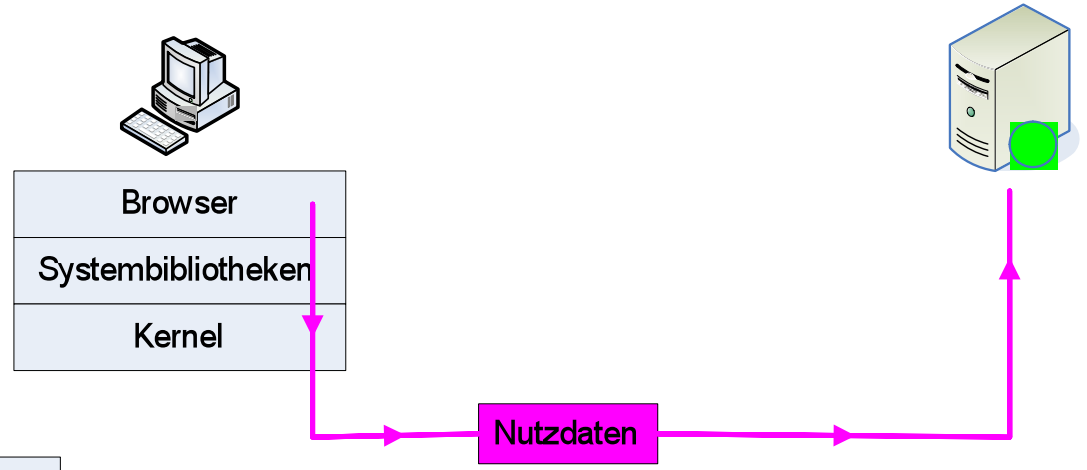
Prinzipieller Ablauf beim Einsatz von SOCKS



- Systenfunktionen wie connect() und bind() laufen über den SOCKS-Server
- Damit auch für Server, nicht nur für Clients geeignet

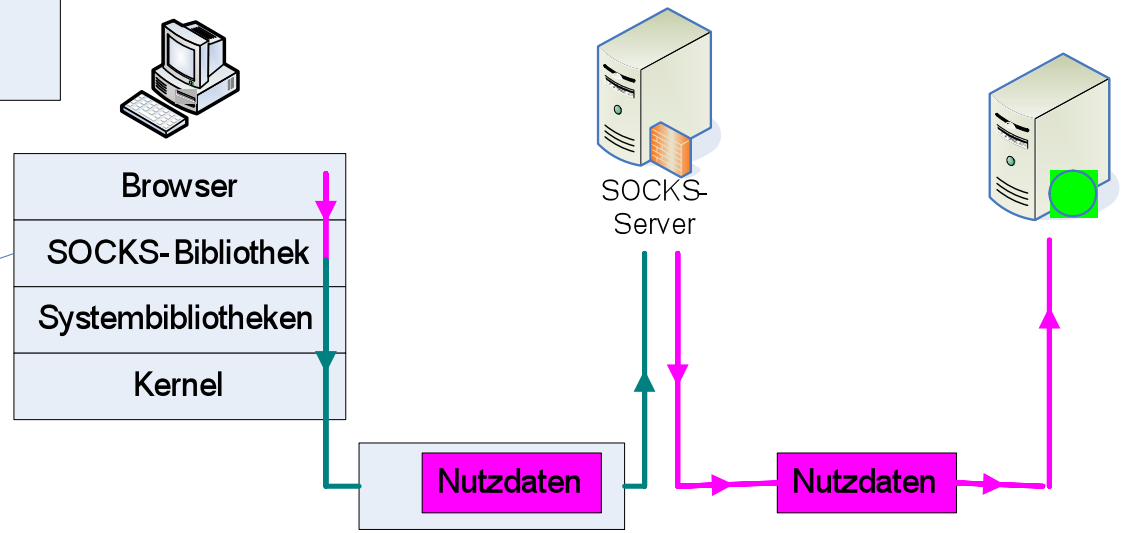
„Socksify“ mit LD_PRELOAD unter Linux

Ohne „Socksifizierung“:




```
export SOCKS_USERNAME="secp"
export SOCKS_PASSWORD="geheim"
export LD_PRELOAD="/usr/lib/libdsocks.so"
firefox
```

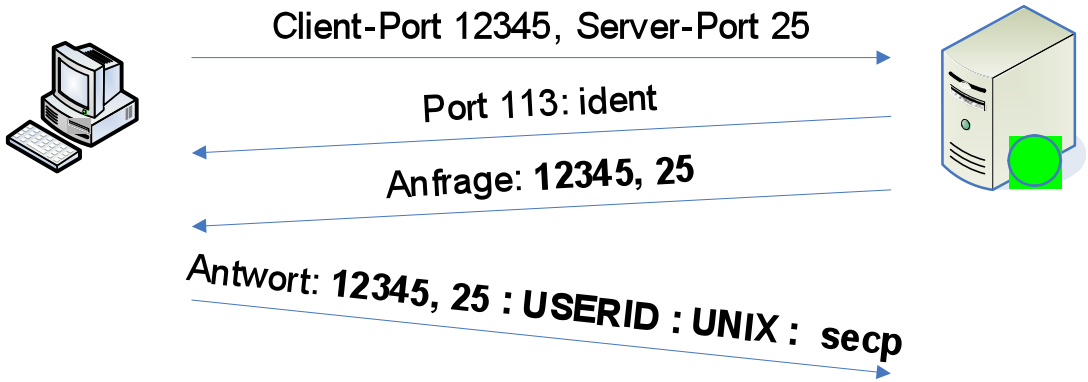
Mit „Socksifizierung“:



Funktioniert nicht bei „statisch“ gelinkten Binaries!

SOCKS v4 vs. v5

- SOCKS v4
  - Nur für TCP/IP
  - „Authentifizierung“ nur über ident-Protokoll
  - Client muss DNS-Auflösung selbst durchführen



- SOCKS v5
  - Vollwertige Client-Authentifizierung (GSS-API)
  - DNS-Auflösung über Server
  - Auch für UDP
  - Referenzimplementierung von NEC/Permeo, Open Source z.B. Dante
  - Einschränkungen bzgl. Zielrechner und -ports konfigurierbar

Einschub: GSS-API

- Generic Security Services API (Java, C)
- Generische Schnittstelle für Client-Server-Authentifizierung
- Häufig eingesetzt für Kerberos-Authentifizierung, da keine implementierungsübergreifend einheitliche Kerberos-API spezifiziert war
- Reduziert Implementierungsaufwand für Applikationen durch Entkopplung von Authentifizierungsmechanismen
- Unterstützt Nachrichten-Verschlüsselung („wrapping“)
- Aber: Recht komplex handzuhaben
  - Populäre Alternative: SASL (Simple Authentication and Security Layer)

Bewertung Verbindungs-Gateway

- Anwendungsunabhängige Filterung
- Ein Proxy für alle Dienste
- Umfangreiche Logging Möglichkeit und damit Accounting
- Zustandsbehaftet
- Benutzerauthentisierung und benutzerabhängige Filterung
- Entkopplung von internem und externem Netz
- Möglichkeit der Erstellung von Nutzungsprofilen

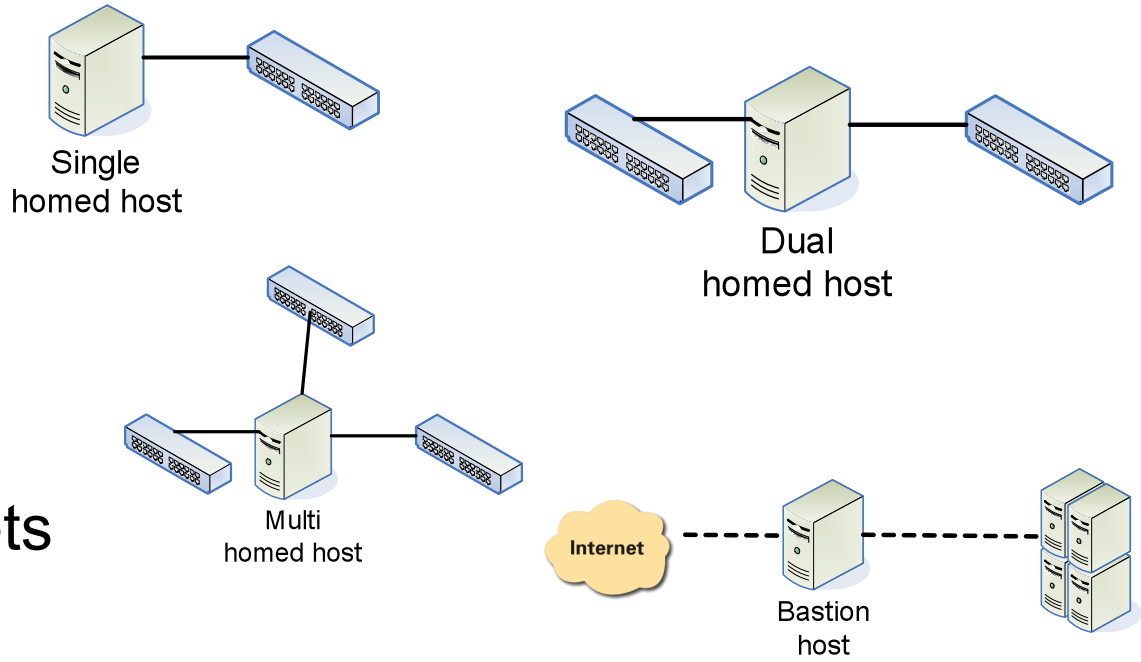
- Möglichkeit der Erstellung von Nutzungsprofilen
- I.d.R. keine Filterung nach Dienstprimitiven möglich
- Sicherheit der Proxy-Implementierung und -Konfiguration?
- Support durch oder Modifikation der Clients erforderlich

Firewall-Architekturen

- Kombinationen von FW-Komponenten und deren Anordnung wird als Firewall-Architektur bezeichnet

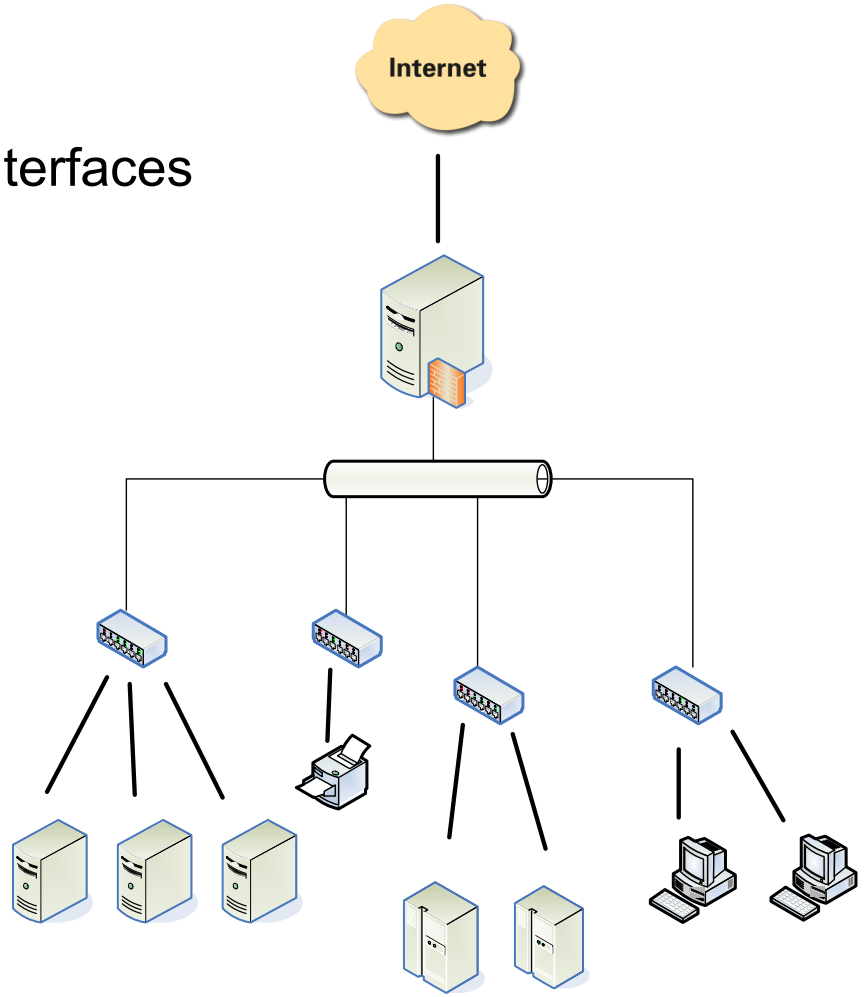
- Unterschiedlicher Schutzbedarf führt zur Bildung von Schutzzonen, z.B. öffentlich zugänglich, Mitarbeiternetz, interne Serversysteme, Verwaltungsnetz (Personaldaten), Testnetz, ...

- Single Box Architektur
  - Screening Router
  - Dual Homed Host
- Screened Host
- Screened Subnet
- Multiple Screened Subnets



Single Box Architektur

- FW als einziger Übergang ins interne Netz
  - Router (Screening Router) übernimmt FW-Funktionalität (i.d.R. Paketfilter)
  - „Normaler“ Rechner mit 2 Netzwerk-Interfaces (Dual Homed Host)

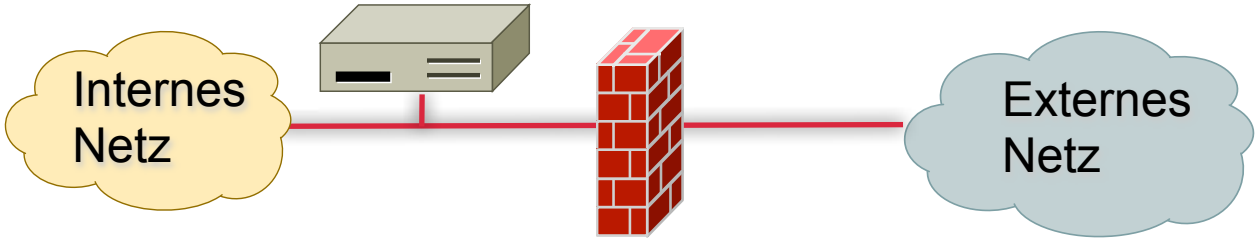


- Billige und einfache Lösung
- Single Point of Administration
- I.d.R. gute Performance (falls nur Paketfilter eingesetzt wird)

- ✶ Mangelnde Flexibilität
- ✶ Single Point of Failure

Screened Host

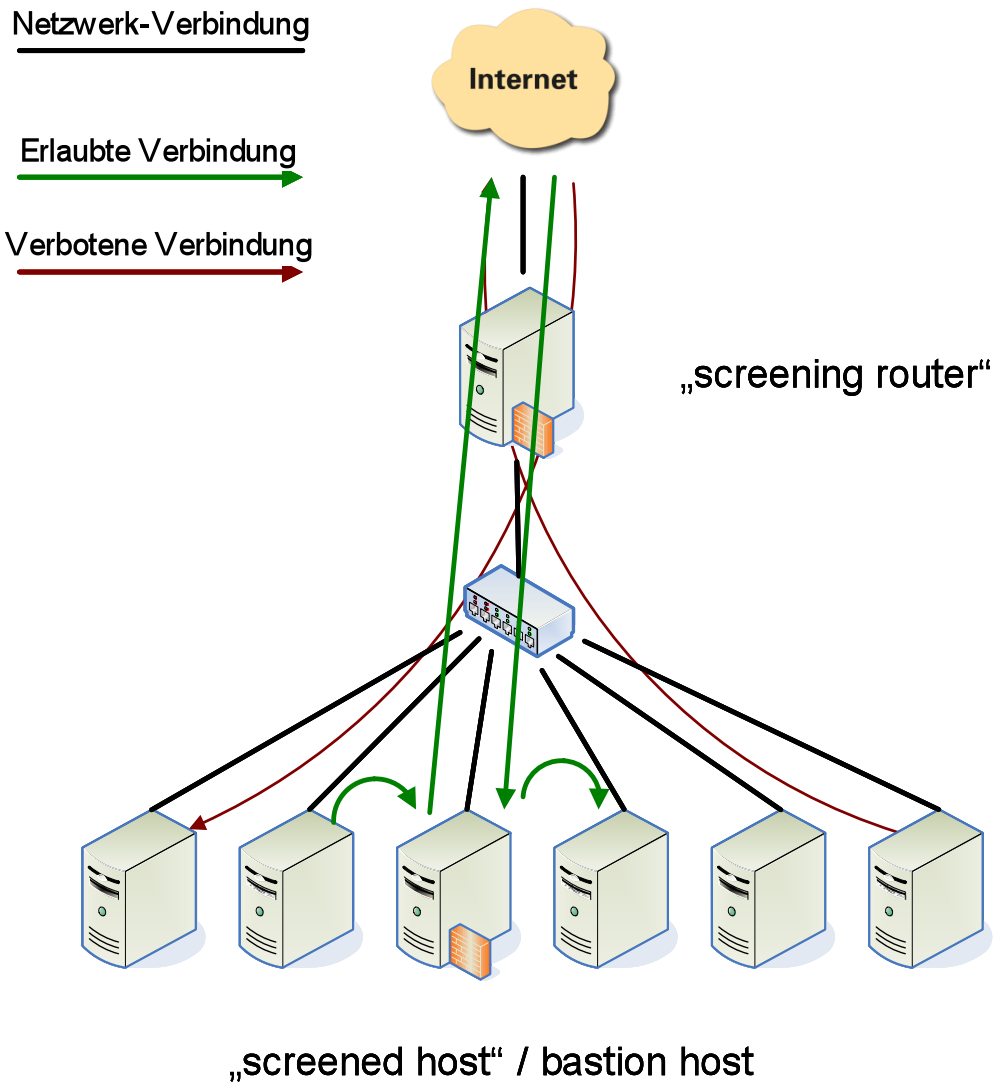
- FW (**Bastion Host**) liegt im internen Netz (nur 1 Interface)
- Verkehr von außen wird über Screening Router (vor-) gefiltert und i.d.R. zum Bastion Host geleitet
- Bastion Host kann Aplication Level Gateway oder Circuit Level Gateway realisieren



- Trennung von Paket- und Applikationsfilter
- Vorfilterung des externen Verkehrs
- Hohe Flexibilität

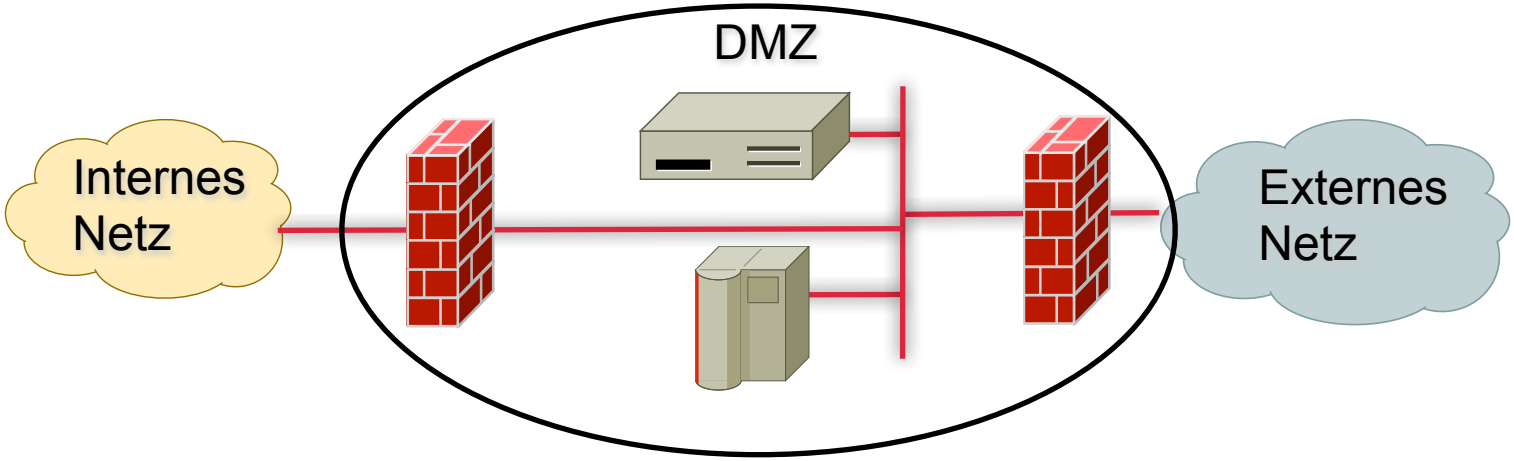
* Pakete können immer noch direkt in internes Netz gelangen

Screened Host: Verbindungen



Screened Subnet

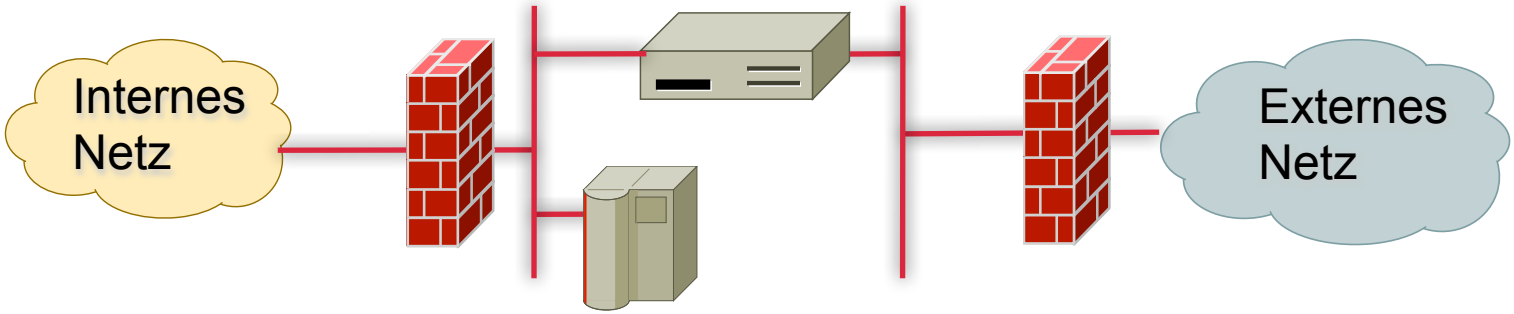
- FW Komponenten liegen in einem eigenen Subnetz (Perimeter Subnet), auch demilitarisierte Zone (DMZ) genannt
- Schutz der DMZ sowohl nach innen als nach außen durch Paketfilter
- Erweiterung der DMZ um dezidierte Server, z.B. HTTP/SMTP



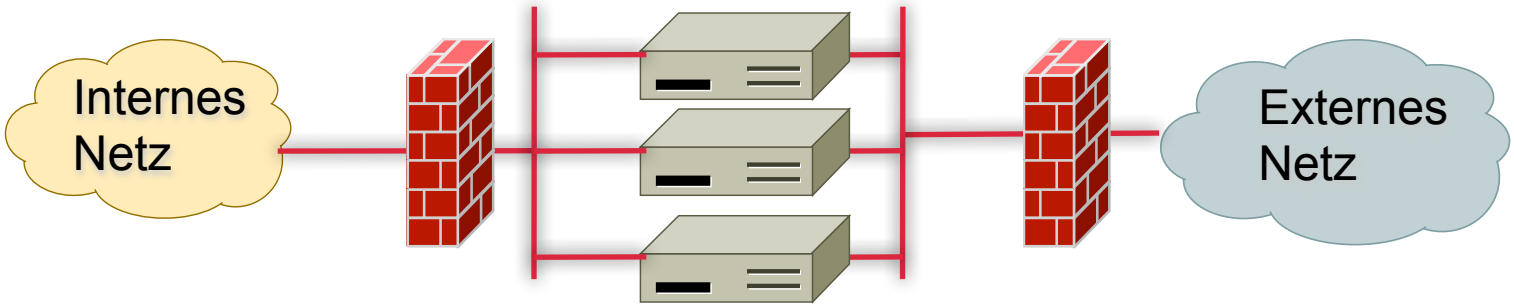
- Keine direkte Verbindung von außen nach innen mehr möglich
- Zusätzlicher Grad an Sicherheit
- Interner Router/FW schützt vor Internet und ggf. vor DMZ

Multiple Screened Subnet

- Verwendung zweier Perimeter-Subnets, getrennt durch Dual Homed Host



- Verwendung mehrerer Bastion Hosts (Redundanz)



Möglichkeiten und Grenzen von Firewall-Arch.

- Abgestufte Sicherheitskontrollen (vom Einfachen zum Komplexen)
- Möglichkeiten effizienter Protokollierung
- Möglichkeiten der Profilbildung

- Problem der Fehlkonfiguration
- Umfangreiche Kenntnisse erforderlich
- Trügerische Sicherheit
- Erheblicher Administrationsaufwand
- Tunnel-Problematik
  - Immer mehr Anwendungsprotokolle werden z.B. über HTTP getunnelt
  - Firewall kann dies nicht erkennen

Firewall-Management

- Herausforderungen:
  - Räumliche Verteilung und Heterogenität
  - Skalierbarkeit
  - Keine Konfiguration „von der Stange“ möglich
  - Individuelle Tests erforderlich
- Isolierter „Firewall Management Server“ zur Vorbereitung / Versionierung der Firewall-Konfigurationen
- Technische Aufgaben beim Firewall-Management:
  - Konfiguration der Regeln / Policies
  - Logfiles und gemeldete Events
  - Routing und interne Konfiguration (z.B. verwendete DNS-Server)
  - Backup / Restore
  - Software-Updates einspielen

Erstellen von Firewall-Konfigurationen unter Linux

- Evolution der Management-Tool:
  - Linux 2.0.x: ipfwadm
  - Linux 2.2.x: ipchains
  - Linux 2.4.x und Linux 2.6.x: iptables
  - Geplanter Nachfolger: Nftables (Name abgeleitet von „NetFilter“)
- Shell-Skripte mit zahlreichen iptables-Befehlen
  - Typischer Aufbau:
    Alle Regeln löschen; Default-Policies setzen; einzelne Regeln eintragen
  - Direkte Kontrolle über alle Abläufe
  - Aber suboptimale Benutzerfreundlichkeit und Skalierbarkeit
- Alternativen:
  - GUIs
  - Einfachere Konfigurationssprachen

Regelerstellung mit ferm („for easy rule making“)

- Beispiel Web-Server:

```
table filter {
    chain INPUT {
        policy DROP;

        # connection tracking
        mod state state INVALID DROP;
        mod state state (ESTABLISHED RELATED) ACCEPT;

        # allow local connections
        interface lo ACCEPT;

        # respond to ping
        proto icmp icmp-type echo-request ACCEPT;

        # our services to the world
        proto tcp dport (http https) ACCEPT;
    }

    # outgoing connections are not limited
    chain OUTPUT policy ACCEPT;

    # this is not a router
    chain FORWARD policy DROP;
}
```

Software und weitere Beispiele unter
http://ferm.foo-projects.org/

Grenzen von Firewalls

- Mit Tools zum „anonymen Surfen“ können Filterregeln umgangen werden
- Über HTTP können andere Protokolle getunnelt werden
  - z.B. PPP-Verbindungen über GNU httptunnel
- Starre Portzuordnung
  - Paketfilter lässt TCP-Port 443 (für HTTPS) zu
  - Mitarbeiter legt seinen SSH-Server zuhause auf Port 443 statt Port 22
  - oder: OpenVPN über Port 443
- Covert channels, z.B.
  - nslookup kommando1.kommando2.example.com

Intrusion Detection Systeme (IDS)

- Synonym: IT-Frühwarnsysteme

- Intrusion Detection Systeme (IDS)
  - Ziel: „Angriffe“ automatisch erkennen (und ggf. blockieren)
- Intrusion Prevention Systeme (IPS)
  - Ziel: Auf „Angriffe“ reagieren
    (Eskalation verhindern / Gegenmaßnahmen ergreifen)

- Methoden:
  - Passive wie auch aktive Überwachung
  - von Systemen und Netzen / Netzsegmenten

- Im Folgenden:
  - Hostbasierte und netzbasierte IDS
  - Angriffserkennung

Hostbasierte IDS (HIDS)

- Ausgeführt auf dem zu überwachenden System
  - Systemüberwachung
  - Applikationsüberwachung
  - Integritätsüberwachung (kryptographische Prüfsummen; Hashing)
- \+ Individuell an zu überwachendes System anpassbar
- \+ Sehr spezifische Aussagen über erkannte Angriffe möglich
- \- Benötigt Ressourcen des zu schützenden Systems
- \- HIDS selbst als Angriffsziel
- \- Anpassung an individuelles System erforderlich
- \- Fällt bei erfolgreichem Angriff mit angegriffenem System aus
- Bsp.:
  - Tripwire
  - AIDE (Advanced Intrusion Detection Environment)
  - Samhain

Netz-basierte IDS (NIDS)

- Eigener Sensor (kein Gastsystem)
- Überwachen den Netzverkehr (Mithören):
  - eines Rechners
  - eines (Sub-) Netzes
  - einer gesamten Domäne

\+ Ein Sensor für das gesamte Netz

\+ NIDS kann „unsichtbar“ installiert werden

\+ NIDS kann ausfallsicher installiert werden

\+ Verteilte Angriffe erkennbar

\- Betrieb am Mirror Port von Netzgeräten; Problem: Packet-Drop

\- Überlast des gesamten NIDS möglich

\- Verschlüsselte Daten und Kanäle

\- „nur“ Netzauffälligkeiten erkennbar

Angriffs- bzw. Mißbrauchserkennung

- Signaturbasiert:
  - Pattern Matching & Expertensysteme
  - „typische Angriffsmuster“
  - \+ Zuverlässigkeit
  - \- Nur bekannte Angriffe erkennbar (keine Zero Day Exploits)
  - Bsp.: Snort
- Anomalie-Erkennung:
  - Lernt „Normalverhalten“
  - Abweichung wird als Angriff gedeutet
  - \+ Flexibel bei neuen Angriffen
  - \- Adaptivität bei Netzänderungen
  - \- False Positives & False Negatives
- Integritätsprüfung
  - Berechnung kryptographischer Hashes
  - Speicherung auf WORM und Vergleich
  - \+ Schnell und Zuverlässig
  - \- Aufwand bei gewollter Datenänderung
  - Bsp.: Tripwire
- Kombination der Verfahren

Beispiel: Tripwire

- www.tripwire.org
- Host-basiertes IDS
- Überwacht anzugebende Dateien und Verzeichnisse
- Erstellt (verschlüsselte) Datenbank mit Prüfsummen:
  - MD5
  - SHA-1
  - HAVAL, ...
- Berechnet regelmäßig Hashes und vergleicht mit gespeicherten
- Probleme:
  - Auswahl schützenswerter Objekte
  - Berechtigte Änderungen (z.B. Software-Updates)
  - Initialstatus muss sicher sein

Beispiel: Snort



- www.snort.org
- Netzbasiertes IDS

- Signatur-basierte Analyse
  - Regeln in Dateien definiert (*.rules)
  - Alle Regeln mit logischem ODER verknüpft
  - Netzverkehr wird gegen diesen Regelsatz geprüft

- Konfiguration der Regeln
  - Manuell durch Administrator
  - Mitgelieferte Community Rules (ggf. Anpassungen erforderlich)
  - Kostenpflichtige, tagesaktuelle Vulnerability Research Team (VRT) Rules
    - z.B. Erkennen aktueller Exploits
    - fließen mit Verzögerung (30 Tage) in die Community Rules ein

Snort-Konfiguration

- Globale Variablen (snort.conf)
  - Netzstruktur, üblicherweise
    - HOME_NET (welche IP-Bereiche sind zu schützen?)
    - EXTERNAL_NET (!HOME_NET - alles andere)
  - Bekannte Server („noisy machines“ im HOME_NET), z.B.
    - DNS_SERVERS
    - HTTP_SERVERS
- Präprozessoren, z.B.
  - frag3 (zum Behandeln von in Fragmente zerlegte Angriffen)
  - stream5 (zielspezifische stateful inspection von Kommunikationsabläufen, z.B. Reihenfolge von TCP SYN/FIN/Reset, Nutzdaten in SYN-Paketen (für einige OS in Ordnung), ...)
- Output options, z.B.
  - Logfiles
  - Datenbanken (MySQL, PostgreSQL)

Snort - Detection Engine Konfiguration

- Generelles Vorgehen:
  - Netzverkehr wird abgehört (sniffing)
  - Pakete werden vorverarbeitet (preprocessing)
  - Vergleich der Pakete mit in Rules angegebenen Parametern
  - Entscheidung:
    - Verwerfen des Pakets (drop/reject; bei „inline“-mode)
    - Erzeugen eines Alarms (alert)
    - Paket protokollieren (log)
    - Ignorieren des Pakets (pass)
- Aufbau von Rules: Header und Body
- Header enthält:
  - Action
  - Protocol
  - Source IP / Source Port(s)
  - Operator
  - Destination IP / Destination Port(s)

Snort - Rules Body (1/2)

- Body einer Rule enthält allgemein
  - Event Message (Alarmmeldung, z.B. zum Finden in Logfiles)
  - Patterns: Wann trifft die Regel auf ein Paket zu?

- Patterns
  - content: Inhalt des Pakets
    - Als Sequenz von Hexzahlen oder ASCII (z.B. /bin/sh)
    - Groß-/Kleinschreibung wird ignoriert
    - depth (gibt Tiefe der Suche vor)
    - uricontent (Untersuchung des vom HTTP-Präprozessor extrahierten URI)
  - flag
    - TCP-Flags (SYN, FIN, RST, ACK, URG, PSH)
    - Operatoren (+, *, !) für Matching von Flag-Kombinationen

Snort - Rules Body (2/2)

- (Patterns)
  - flow
    - to_client, from_server
    - to_server, from_client
    - established
    - stateless
  - threshold
    - count n
    - seconds m
    - type
      - limit: Nur beim ersten Vorkommen von n Events pro Zeitintervall m,
      - threshold: Jedes n. Mal pro Zeitintervall m
      - both: Nur einmal pro Zeitintervall m nach n Events
    - track (by_src, by_dst)

Snort Rules - Beispiele

- Beispiel Threshold:
  Logge nur einen Event pro Minute, falls eigentlich mehr als 30 pro Minute auftreten

```
threshold: type both, track by_src, count 30, seconds 60
```

- Beispielregel:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"Internal SSH
attack 60/60sec"; flags:S; threshold: type both, track by_src,
count 60, seconds 60; sid:1000003;)
```

Header:
Action: alert
Quelle: $HOME_NET, beliebiger Port
Operator: ->
Ziel: $EXTERNAL_NET, Port 22

Body:
Event message: „Internal SSH...“
Flags: S = SYN
Logging: 1x pro Minute
falls mehr als 60 Events/Min
SID = Identifier der Rule

IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 15: Anti-Spam-Maßnahmen



Inhalt

1. Spam aus Betreibersicht
2. Spam-Statistik
3. Spam-Quellen
4. Abwehrmaßnahmen im Münchner Wissenschaftsnetz (MWN)
   - Mail aus dem MWN ins Internet
     - Bot-Net-„Infektionen“ verhindern
     - Bot-Net-Überwachung; Identifikation von Clients im MWN
     - Statistische Verkehrsanalysen
     - Authentifizierung der Sender
     - Kennzeichnung von Netzen, aus denen keine Mail verschickt werden sollte
   - Mail aus dem Internet ins MWN
     - Phase I: „(Spam-) Mails zurückweisen“
     - Phase II: Inhaltliche Bewertung und Markierung

Spam aus Sicht der Betreiber von Mail-Servern

- (seit ca. 2003/2004) Viren verschicken sich selbst und Spam:
  - Problem: Mailadressen werden z.T. generiert
  - Spam-/Viren-Mail nicht zustellbar, wenn Adresse ungültig
  - Mailserver antwortet mit Mitteilung an den Absender (ebenfalls gefälscht)
  - Mailserver versucht über längeren Zeitraum, diese Mitteilung zuzustellen
  - Folge: Hohe Last auf den Mail-Servern
- Abwehrmaßnahmen, Grundidee:
  - Formale Verfahren (z.B. protokollkonformes Verhalten)
  - Statistische Analysen
  - Überprüfungen des sendenden Mailservers
  - Frühzeitige Verifizierung der Empfängeradressen
  - Keine (aufwendige!) inhaltliche Analyse (Mail-Body)
- Ziel: Ressourcenschonende Abwehr von Spam

Spam-Statistiken

- Gesicherte Aussagen schwierig
- Unterschiedlichste Arten von Statistiken:
  - Unternehmensstatistiken
  - Statistiken von Herstellern von Sicherheitslösungen
  - Statistiken von Blacklist-Betreibern
- Unterschiedlichster Fokus
  - Regional
  - Bot-Net
- Gute Sammlung unterschiedlichster Statistiken:
  - http://spamlinks.net/stats.htm

Spam-Quellen: Länder

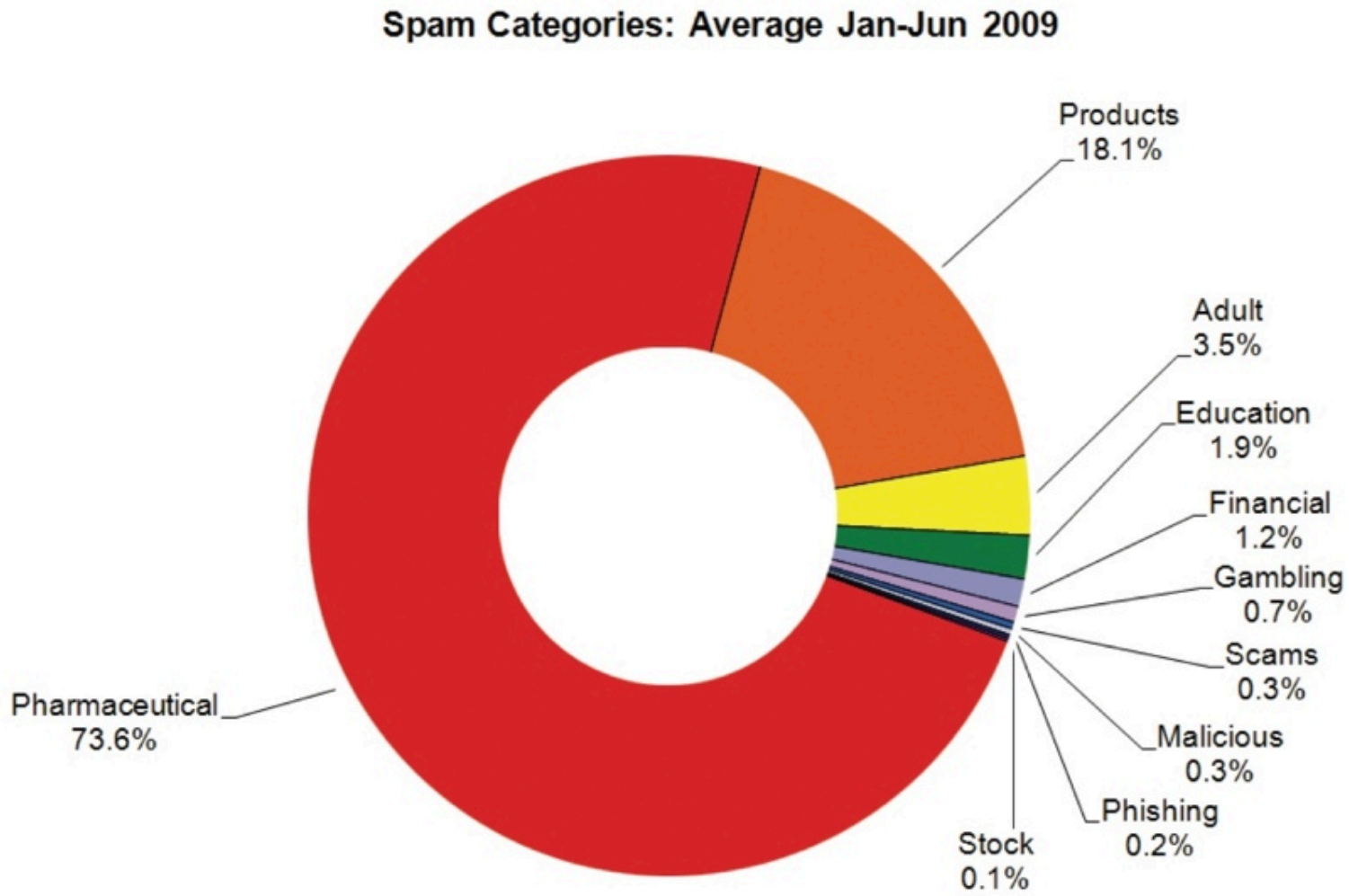
The 10 Worst Spam Origin Countries		As at 04 January 2010
Rank	Country	Number of Current Known Spam Issues
1	United States	2388
2	China	552
3	Russian Federation	428
4	United Kingdom	290
5	Spain	240
6	Argentina	231
7	Italy	183
8	Brazil	183
9	France	179
10	Germany	177

Quelle:

www.spamhaus.org

Spam-Statistik: Inhaltliche Klassifikation

- Quelle: www.marshal.com



Top Spammer (www.spamhaus.org)

1		**Canadian Pharmacy** A long time running pharmacy spam operation. Uses botnet spam techniques to send tens-of-millions of spams per day. Probably uses many affiliates all over the world to spam but is probably based in Eastern Europe and hosts sites on botnets and on Chines	**United States**
2		**Leo Kuvayev / BadCow** Russian/American spammer. Does "OEM CD" pirated software spam, copy-cat pharmaceuticals, porn spam, porn payment collection, etc. Spams using virus-created botnets and seems to be involved in virus distribution. Partnered with Vlad - aka "Mr. Green"	**Russian Federation**
3		**HerbalKing** Massive affilitiate spam program for snakeoil Body Part Enhancement scams. Also does replica luxury goods, pharma and porn. Spams via botnets, bulletproof hosting offshore and even sometimes uses fast flux hosting.	**India**
4		**Vincent Chan / yoric.net** Vincent Chan and his Chinese partners have been sending spam for years. They mainly do pharmacy, and are able to send out huge amounts daily. The	**Hong Kong**

Stand: 01/2010

Wichtigste Spam-Quellen

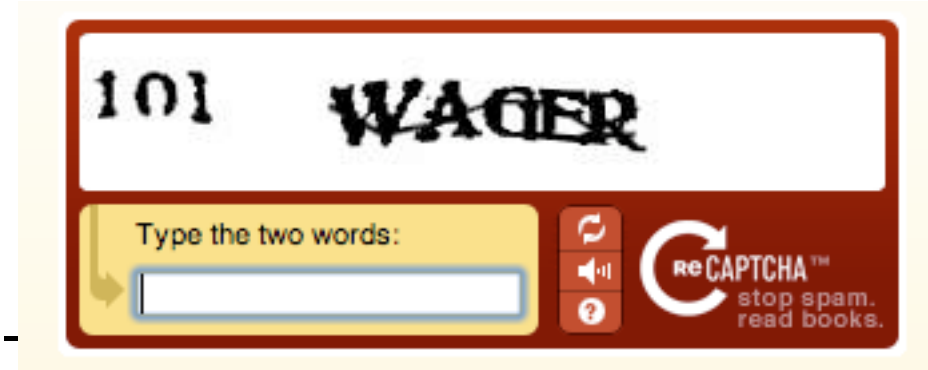
- Bot-Netze / Viren
- Wegwerf-Accounts bei Freemailern
- Weitergeleiteter Spam
  - Nutzer legt in nicht gesicherter Domäne ein .forward an
  - Gesicherte Domäne wird mit weitergeleiteter Spam belastet
- Backscatter-Spam

Spam-Quellen: Bot-Netze / „Viren“

- Start 2003: Wurmfamilien wie Sobig bauen Botnet auf
- Frühe (Spam-) Botnetze:
  - Optimiert, um möglichst viele Mails in kurzer Zeit zu generieren
  - keine vollständige SMTP-Engine implementiert
  - z.T. zentrale Komponenten erforderlich (Bot-Server)
  - Fire-and-forget-Prinzip (nutzbar im Greylisting; später in diesem Kapitel)
- Neuere template-basierte Botnetze
  - Implementieren z.T. vollständige SMTP-Engine
  - Spam-Templates
  - Liste von Adressen
  - arbeitet völlig autonom; keine zentralen Komponenten erforderlich

Spam-Quellen: Wegwerf-Accounts

- Freemailer bieten Möglichkeit, über Web-Inferface Mail zu versenden
- Automatisiertes Anlegen von Mail-Accounts war möglich
- Spammer legt große Zahl von Accounts an und sendet Spam
- Gegenmaßnahme: CAPTCHA (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part)
  - Turing-Test: kann Mensch unterscheiden, ob er mit Mensch oder Computer kommuniziert
  - CAPTCHA: Computer unterscheidet Computer von Mensch
  - Beispiel (recapchta.net)
- Problem: erste Bots mit CAPTCHA-Erkennungsroutinen
  - 2006: phpBB-Bot registriert sich bei CAPTCHA-gesichertem Bulletin-Board



Spam-Quelle: Backscatter

- Indirekter Spam durch „Rückstreuung“
- Spammer verwendet gültige Adressen als Sender-Adresse
- Automatismen generieren automatische Antwort
  - Unzustellbarkeitsnachricht (Empfänger existiert nicht)
  - Vacation-Mail
  - Empfänger erzeugt automatisiert Empfangsbestätigung
  - Weiterleitung; aber Ziel-Server nimmt diese nicht an
  - Mailing-Listen, für die keine Schreibberechtigung besteht
  - Anti-Spam-System berichtet über Blockade der Mail
  - Virenscanner findet Virus und informiert Sender
- Antwort (ggf. mit Spam-Inhalt) geht an unbeteiligten Dritten

Inhalt

1. Spam aus Betreibersicht
2. Spam-Statistik
3. Spam-Quellen
4. Abwehrmaßnahmen im Münchner Wissenschaftsnetz (MWN)
   - Mail aus dem MWN ins Internet
     - Bot-Net-„Infektionen“ verhindern
     - Bot-Net-Überwachung; Identifikation von Clients im MWN
     - Statistische Verkehrsanalysen
     - Authentifizierung der Sender
     - Kennzeichnung von Netzen, aus denen keine Mail verschickt werden sollte
   - Mail aus dem Internet ins MWN
     - Phase I: „(Spam-) Mails zurückweisen“
     - Phase II: Inhaltliche Bewertung und Markierung

Spam aus dem MWN ins Internet

- LRZ verantwortlich für den Betrieb der Netzinfrastruktur bis zur „Datendose“
- Verantwortung über Endsysteme liegt bei Instituten
- Vorgaben über Betriebssysteme o.ä. sind nicht möglich
- Große Heterogenität bei den betriebenen Systemen
- Viele Gäste im Netz
- Deutlich andere Struktur als in „normalen“ Unternehmen

➡ Bestimmter Anteil infizierter Systeme lässt sich nicht vermeiden

➡ Damit auch potentielle Quellen für Spam im MWN

- Ziel: Spam soweit wie möglich verhindern

Potentielle eigene Spam-Quellen: Schutzmaßnahmen

- Meldung oder Beschwerde von Extern
  - Kommt sehr selten vor; Prozess zur Abuse-Bearbeitung
- Schutz vor Infektionen mit Viren oder Bot-Net-Clients
  - LRZ betreibt eigenen Windows Update Server (WSUS)
  - Bayernweite Lizenz für Viren-Scanner; kostenlos nutzbar für:
    - Wissenschaftler
    - Mitarbeiter der Universitäten und Forschungseinrichtungen
    - Studenten
    - Nutzung für **private** Zwecke explizit erlaubt
  - Betrieb eines eigenen Update-Servers für Signaturen
  - Awareness-Kampagnen und Information
- Bot-Net-Überwachung
  - Detektion von Bot-Net-Clients
  - Sperrung entsprechender Rechner
  - Information an Nutzer oder Netzverantwortliche

Eigene Spam-Quellen blocken: Verkehrsanalysen

- Statistische Analyse des TCP/IP-Verkehrs
- Unterschiedliche Netzbereiche und Mechanismen
  - Private Netze, Studentenwohnheime, etc.
    - dynamische Verkehrsbeschränkung (Strafpunkte)
    - ggf. automatische Sperre der Rechner
    - ➡ Nat-O-Mat
  - Zentraler Internet-Übergang
    - Internetanschluss: 10 Gbit/s
    - Übergang ins deutsche Forschungsnetz (betrieben vom DFN Verein)
    - Accounting-Mechanismen zur Bestimmung der Anzahl von Mail-Verbindungen
    - Verschiedene Schwellwerte (vgl. folgende Folien)
    - Derzeit keine automatischen Reaktionen
    - Alarmierung, (menschliche) Überprüfung und Reaktion / Eskalation
    - Ausnahmelisten für bekannte Mail-Server im MWN

Verkehrsanalyse: Schwellwerte

- Monitoring-Intervalle: 5 Minuten und 1 Stunde

- Schwellwerte und Reaktion:
  - Statistik Log: 5 Verb. / 5 Min. 30 Verb. / 1 h
  - Soft Limit: 20 Verb. / 5 Min. 80 Verb. / 1 h (Mail an Benutzer)
  - Hard Limit: 300 Verb. / 5 Min 1000 Verb. / 1h (Sperre und Mail)

- Gründe für hohes Mail-Aufkommen
  1. Großer Mailserver
  2. Legitimer Rechner generiert viele Mails (z.B. Monitoring, Stau von Nachrichten, Software läuft Amok)
  3. Versand von Rundbriefen oder Newslettern
  4. Infektion mit Malware und / oder Kompromittierung des Rechners

- ➡ Whitelisting, um False Positives zu vermeiden

Mail-Monitoring: Zahlen

- Schwellwerte (Wdh.)
  - Statistik Log: 5 Verb. / 5 Min. 30 Verb. / 1 h
  - Soft Limit: 20 Verb. / 5 Min. 80 Verb. / 1 h
  - Hard Limit: 300 Verb. / 5 Min 1000 Verb. / 1 h
- Durchschnittliches Mailaufkommen über alle Rechner
  - 1,91 Mails / 5 Minuten
  - 4,96 Mails / 1 h
- Mailaufkommen großer Server
  - bis zu 2.000 Mails / 5 Min
  - bis zu 10.000 Mails / 1 h
- Welches Mailaufkommen schafft ein infizierter Commodity-PC (= Rechner der „Aldi-Klasse“)?
  - bis zu 3.500 Mails / 5 Min.
  - bis zu 18.000 Mails / 1 h

Eigene Spamquellen: Gegenmaßnahmen

- Nutzer-Authentifizierung beim Mail-Versand
  - Nutzer muss sich vor Mail-Versand beim Server authentisieren
    - z.B. mit Benutzername und Passwort
  - RFC 2476 („Message Submission“; 1998)
  - Port 587 anstatt Port 25
  - Kommunikation z.B. über SSL geschützt
- Markierung eigener Netze, aus denen keine Mail kommen sollte
  - Typischerweise gedacht für Dial-Up Netze
  - Eintrag in Blacklisten; z.B. PBL (Policy Block List) von Spamhaus
- Information für Betroffene: Ordentliche Pflege der RIR DB
  - Regional Internet Registry (RIR), z.B. Réseaux IP Européens Network Coordination Centre (RIPE NCC) zuständig für Europa
  - Datenbank mit Informationen über Netz und Betreiber
  - Damit Zuordnung IP-Adresse zu ISP
  - Abfrage mit: whois -h whois.ripe.net <IP-Adresse>

Sender Policy Framework (SPF)

- Erschwert das Fälschen der Absenderadresse
- Schützt damit vor Backscatter-Spam
- Zusätzlicher DNS Resource Record TXT für SPF
  - Enthält Adressen aller Systeme der Domain, die Mail versenden dürfen (Sender Policy)
  - Empfangsserver kann prüfen, ob sendender Rechner berechtigt ist
  - Absender-Fälscher müsste über berechtigten Mail-Server versenden
- Beispiel:
  - Abfrage nach RR vom Typ TXT für die Google Mail Domain:
    `host -t txt gmail.com`
  - `v=spf1 ip4:216.239.32.0/19 ip4:64.233.160.0/19 ip4:66.249.80.0/20 ip4:72.14.192.0/18 .....`

Probleme mit SPF

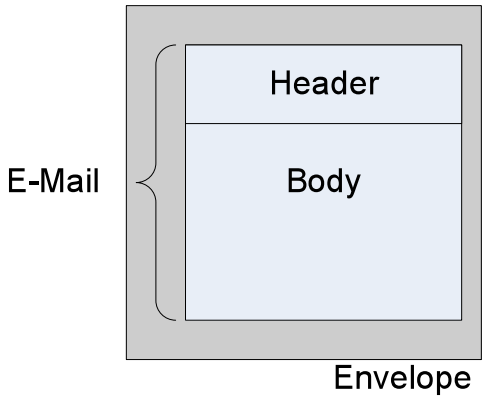
- Spammer verwenden Domains mit korrekter SPF-Konfiguration
  - „Einwegdomains“
  - Der größte Teil der Domains mit SPF-Konfiguration gehört Spammern
- Bot-Net-Client aus korrekt konfigurierter Domain versendet Spam
- Zum Teil schlechte Usability:
  - Weiterleitung von E-Mails mit Beibehalten der ursprünglichen Absenderadresse erfordert Whitelists
  - Bei mehreren E-Mail-Konten Nutzung verschiedener Mailserver zum Mailversand notwendig, aber aus Sicherheitsgründen oft eingeschränkt

Inhalt

4. Abwehrmaßnahmen im Münchner Wissenschaftsnetz (MWN)
   - Mail aus dem MWN ins Internet
     - .....
   - Überblick über das Simple Mail Transfer Protokoll
   - Mail Infrastruktur im LRZ
   - Mail aus dem Internet ins MWN
     - Phase I: „(Spam) Mail zurückweisen“
     - Phase II: Inhaltliche Bewertung und Markierung
   - Betriebserfahrungen und Statistiken

Aufbau von E-Mails

- Header:
  Meta-Informationen wie Absender, Empfänger, ...
- Body:
  Eigentlicher Inhalt, bestehend aus „Body Parts“
- Beim Versand:




```
Message-Id: <87488y4@1nannyplace.com>
Date: Thu,  5 Aug 2004 15:49:10 -0400
Subject: Cheap software - Save up to 60%
From: "Maricela" <wtdxc@1nannyplace.com>
To: Wolfgang.Hommel@lrz-muenchen.de

Hello,                          Leerzeile!

We offer all the software you can imagine!
     Best prices on the net!

Examples:

$60 Ahead NERO 6.3 POWERPACK
$140 Adobe Premiere Pro 7.0
$40 Quicken 2004 Premier Home & Biz
$60 Ahead NERO 6.3 POWERPACK
$20 McAFEE Personal Firewall Plus 2004
```

Verwendung von MIME in E-Mails

Multipurpose Internet Mail Extension

```
Date: Wed, 28 Jun 2006 15:35:16 +0200
MIME-Version: 1.0
To: Wolfgang Hommel <Wolfgang.Hommel@lrz-muenchen.de>
Subject: Word-Dokument
Content-Type: multipart/mixed;
 boundary="------------030407050703000207030304"

This is a multi-part message in MIME format.
--------------030407050703000207030304
Content-Type: text/plain; charset=ISO-8859-15;
format=flowed
Content-Transfer-Encoding: 8bit


Anbei die gewünschte Word-Datei…

--------------030407050703000207030304
Content-Type: application/msword;
 name="blabla.doc"
Content-Transfer-Encoding: base64
Content-Disposition: inline;
 filename="blabla.doc"

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP 7/
CQAGAAAAAAAAAAAAAAABAAAAKwAAAAAA
AAAAEAAALQAAAAEAAAD+////
AAAAACoAAAD//////////////////////////////////
…..
AAAAAAAA
--------------030407050703000207030304--
```

Base64-Kodierung für Binärdaten

MIME-Parts werden durch Boundaries voneinander getrennt

Simple Mail Transfer Protocol: Überblick

- Textbasiertes Protokoll; Protokollablauf

Client	Server	Erklärung
Verbindungsaufbau Port 25		
	220 mail.domain.de	Begrüßung
HELO client.domain.de		Client meldet sich an
	250 Hello client.domain.de	Server bestätigt
MAIL FROM: <user@test.de>		Absenderadresse (Envelope)
	250 Sender OK	
RCPT TO: <empfang@bla.de>		Empfängeradresse (Envelope)
	250 Recipient OK	
DATA		Client möchte Mail senden
	354 Enter mail, end with "." on a line by itself	Server akzeptiert

Simple Mail Transfer Protocol: Überblick

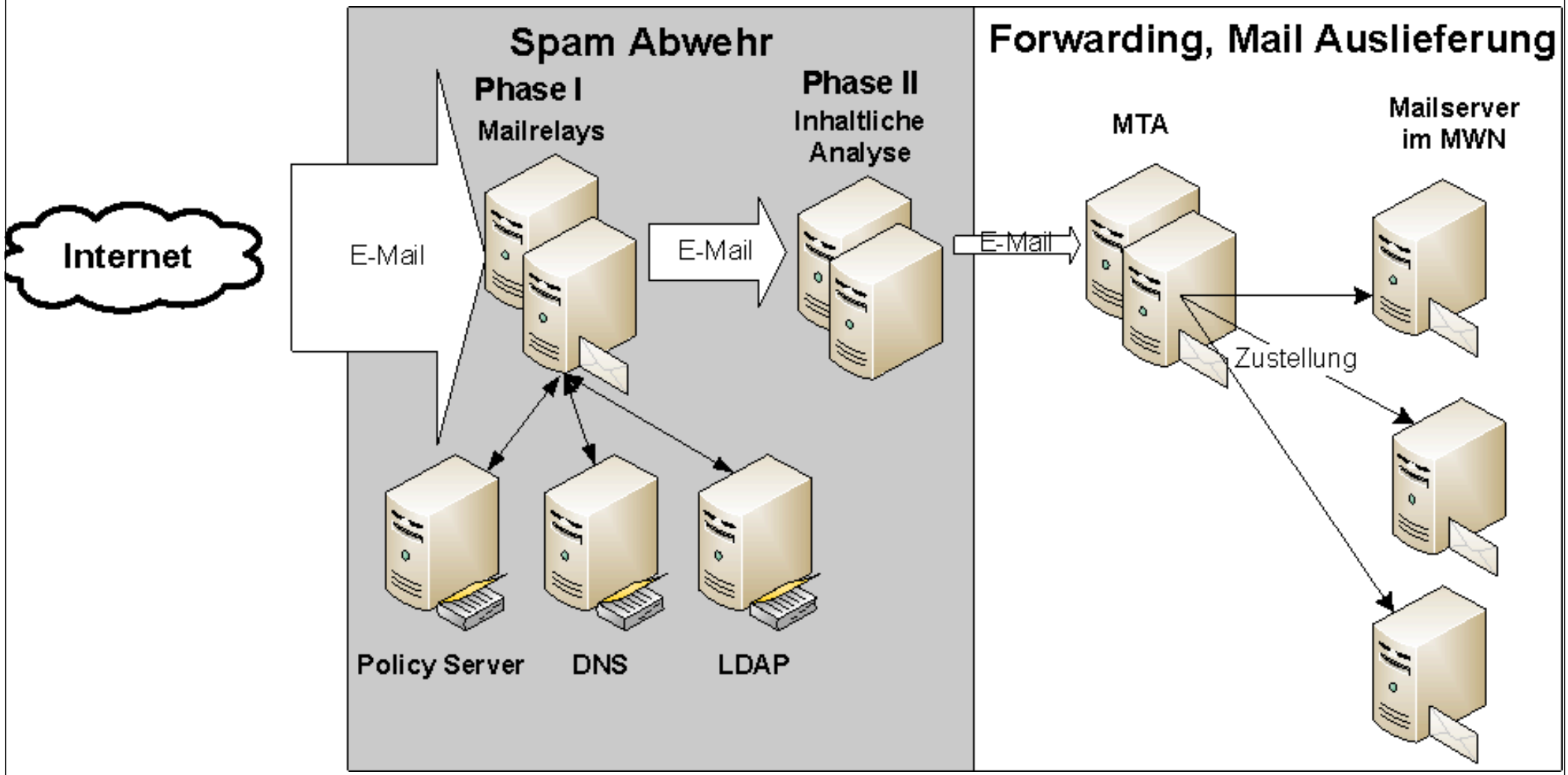
- Protokollablauf Fortsetzung

Client	Server	Erklärung
FROM: fake@fake.de TO: <gast@gast.de> Subject: Testmail Und hier kommen dann die Daten. ;-) .		Client gibt Maildaten an; Hinweis: Envelope-Adressen müssen nicht mit Adressen in der Mail übereinstimmen
	250 Message accepted for delivery	
quit		Client meldet sich ab
	221 Connection closed	Server bestätigt

SMTP: ungesichertes Protokoll

- SMTP-Informationen sind nicht integritätsgeschützt
  - Adressen lassen sich leicht fälschen
- Vertraulichkeit des E-Mail-Inhalts ist nicht sichergestellt
- Verschlüsselung des E-Mail-Body schützt nicht vor Verkehrsflussanalyse

Mail-Infrastruktur im MWN



- Phase I: Entscheidung, ob Mail angenommen oder abgelehnt wird; erst in Phase II wird Protokollprimitiv „DATA“ akzeptiert

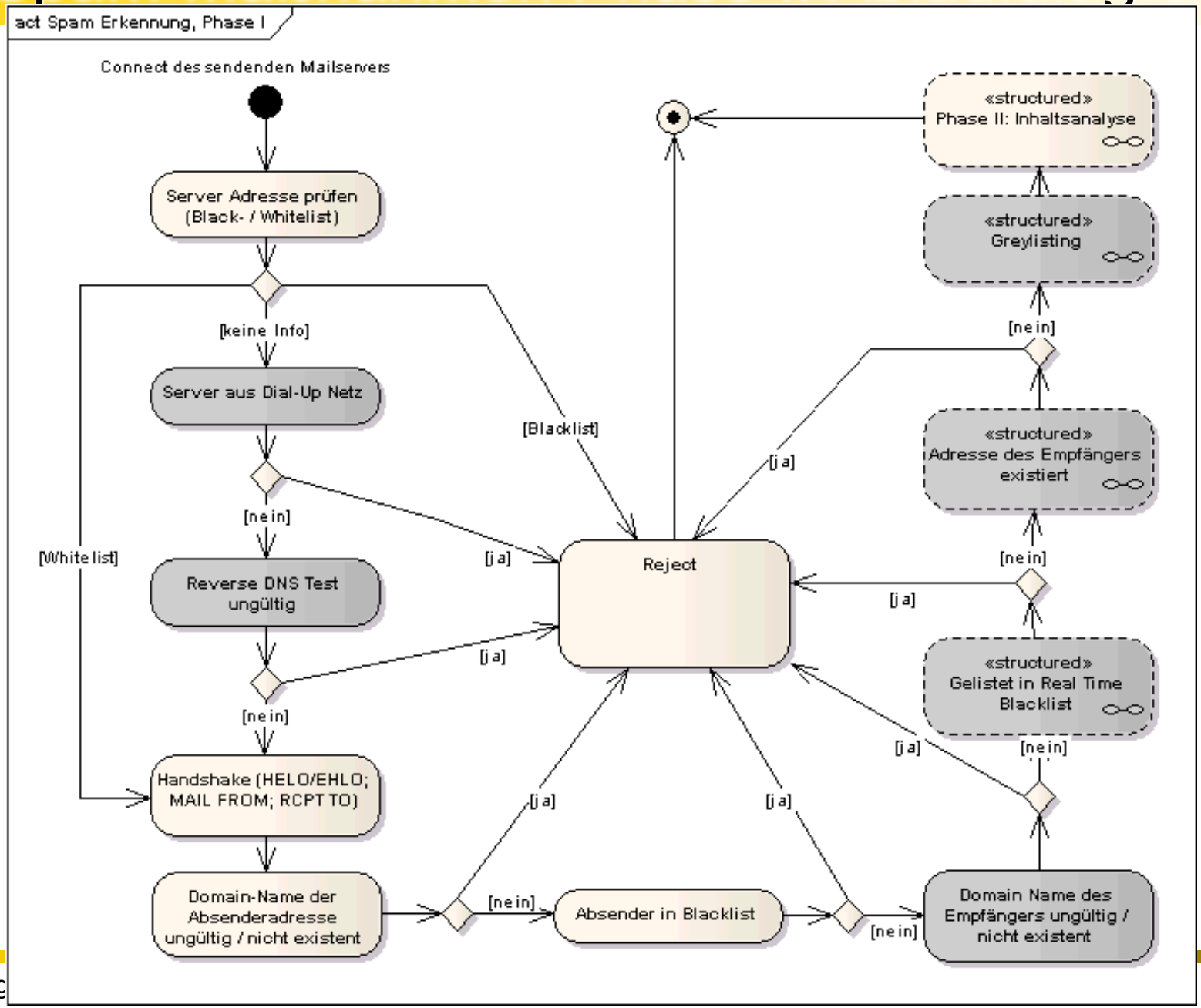
Spam Abwehr Grundlagen

- LRZ: Spam-Anteil 95 - 99,5 %
- Ressourcenschonende Verfahren unbedingt erforderlich
- Grundidee:
  - Annahme nicht regelkonformer Mails ablehnen
  - März 2008: ca. 6 Mio. (99,5%) Mails werden täglich abgelehnt
  - Spitzen: 20 Mio. täglich
  - So wenig inhaltliche Analyse wie möglich
  - Billige Aktionen am Anfang, teure am Ende

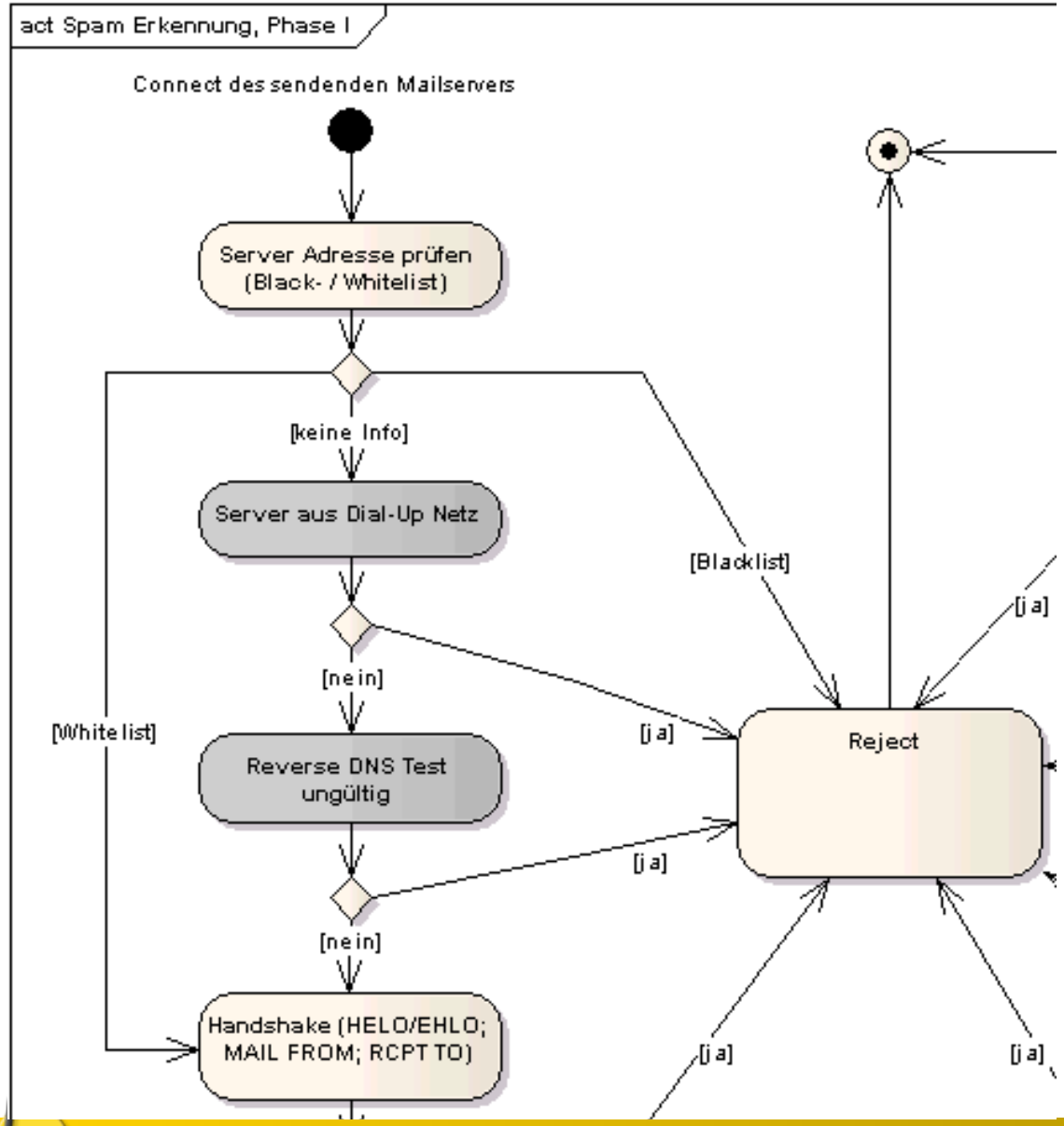
Spam-Abwehr: Phase I

- Völliger Verzicht auf inhaltliche Analyse
- Kriterien zur Ablehnung abgeleitet aus Protokolllogik
- Hierfür nutzbare Daten:
  - IP-Adresse des sendenden Mail Transfer Agent (MTA)
  - Domain aus Protokollelementen HELO bzw. EHLO (vergleichbar mit HELO + zus. Info über Server-Features; ESMTP)
  - Mail-Adresse aus Envelope
  - Mail-Adresse der Empfänger aus Envelope
- Formale Kriterien finden, die Spammer (z.B. Bot) von regulärem MTA unterscheiden; dann
  - Mail sehr früh und ohne weiteren Ressourceneinsatz ablehnen
  - Regulärer MTA
    - korrekt implementiert
    - korrekt konfiguriert
    - gut administriert

Spam-Maßnahmen Phase I: Ablaufdiagramm



Spam-Maßnahmen Phase I: Ablaufdiagramm

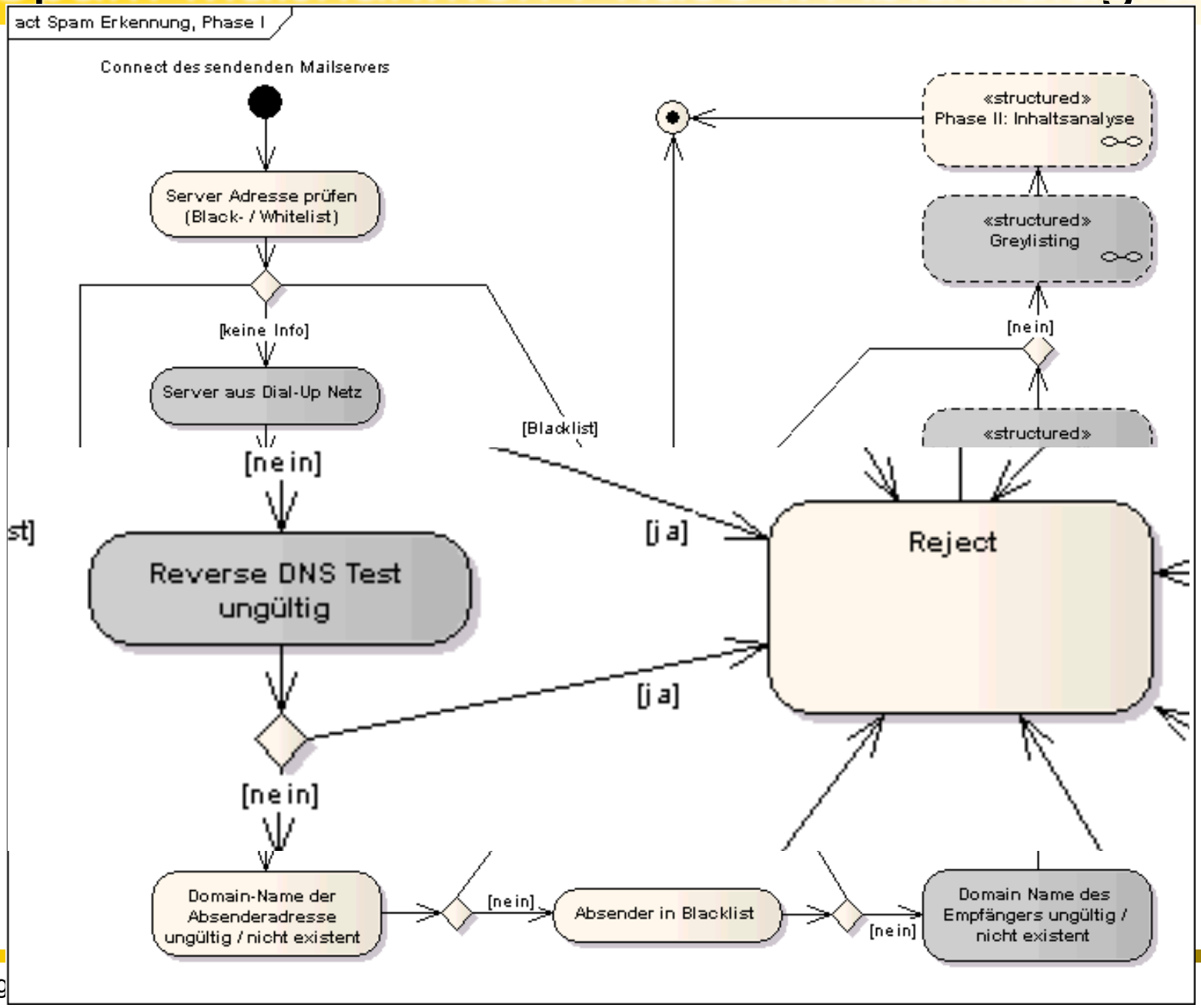


- Black-/Whitelist
- Blacklist
  - Adressen und Domainnamen bekannter Spammer
  - Vom LRZ selbst erstellt und gepflegt
  - ➡ Mail wird abgelehnt
- Whitelist
  - Ausnahmeliste von Servern, die als valide Mailquellen betrachtet werden
  - Bsp.: Prof. erhält Mail aus bekannter Spam-Domain
  - ➡ Handshake wird ausgeführt

Phase I: Server aus Dial-Up Netzen?

- Bots finden größte Verbreitung auf PCs von Heimanwendern
  - Leider oft schlecht administriert; keine Sicherheitsadministration
  - ISPs führen oft keine Missbrauchserkennung durch
  - Mit recht hoher Bandbreite (DSL/Kabel) ans Internet angebunden
  - Charakteristika: erhalten bei Verbindungsaufbau dynamische IP Adresse
- Wie sind Dial-Up Netze erkennbar
  - Namensgebung (z.B. *.t-dialin.net)
  - Idealfall: alle ISPs weltweit dokumentieren Art der Nutzung
  - Eintrag als Kommentar in der RIR-Datenbank (z.B. bei RIPE o. DENIC)
  - Markierung in DNS-basierten Blacklisten (z.B. Spamhaus PBL; vgl. später in Vorlesung)
- Können auch „reguläre“ MTAs ausgeschlossen werden?
  - Ja! Aber:
    - Statische statt dynamischer Adressen verwenden
    - tritt selten auf
    - Eintrag in Whitelist möglich

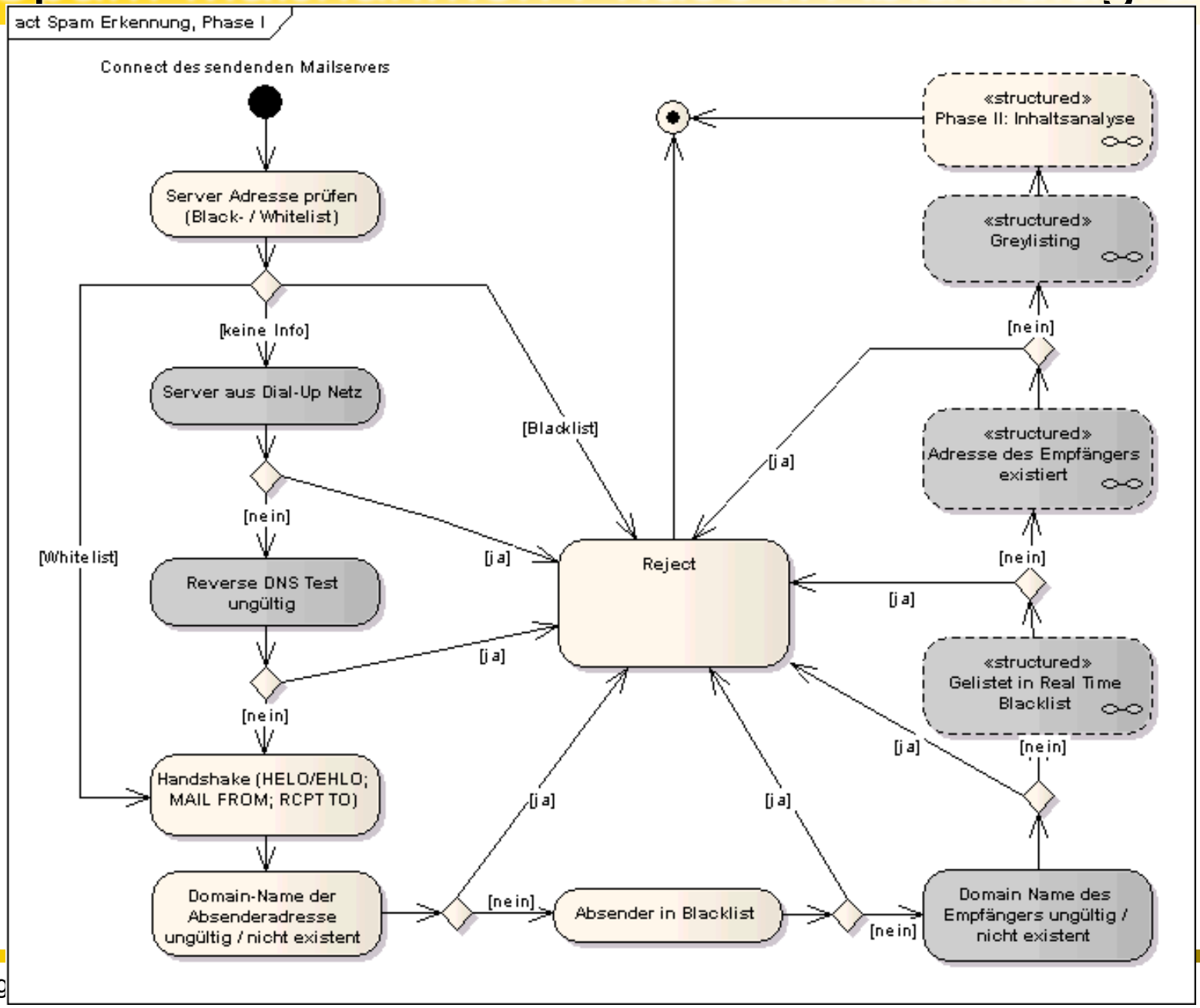
Spam-Maßnahmen Phase I: Ablaufdiagramm



Reverse DNS-Test

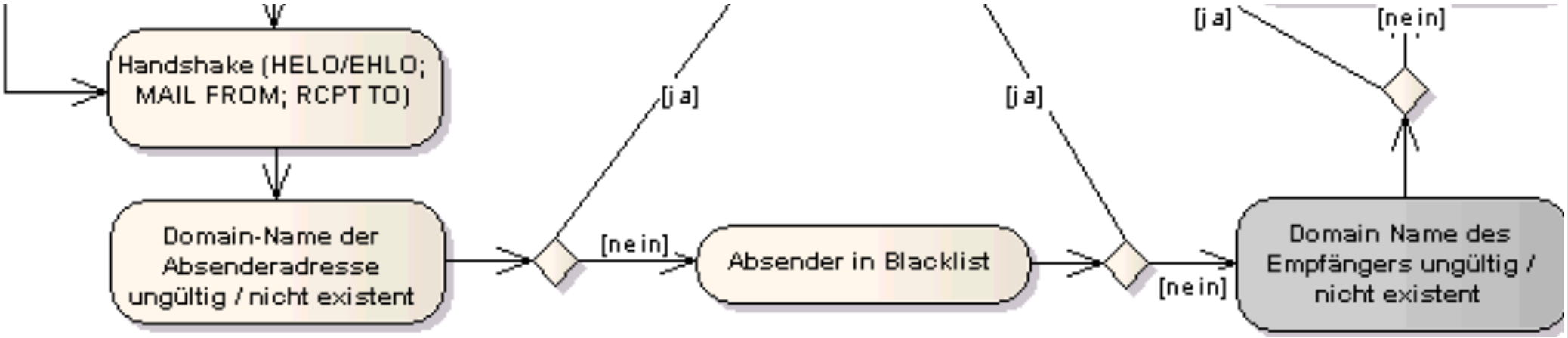
- Konsistenzprüfung bezüglich DNS-Konfiguration
- Dreistufiges Verfahren:
  1. Existiert zur aufrufenden IP-Adresse der zugehörige Name? (PTR Record im DNS)
  2. Gibt es zu jedem Namen eine zugehörige IP-Adresse? (A Record im DNS)
  3. Stimmen aufrufende IP-Adresse und IP-Adresse aus dem A Record überein?
- Viele Provider wenden nur Test (1) an
- Auch für diesen Check ist wieder eine Whitelist erforderlich

Spam-Maßnahmen Phase I: Ablaufdiagramm

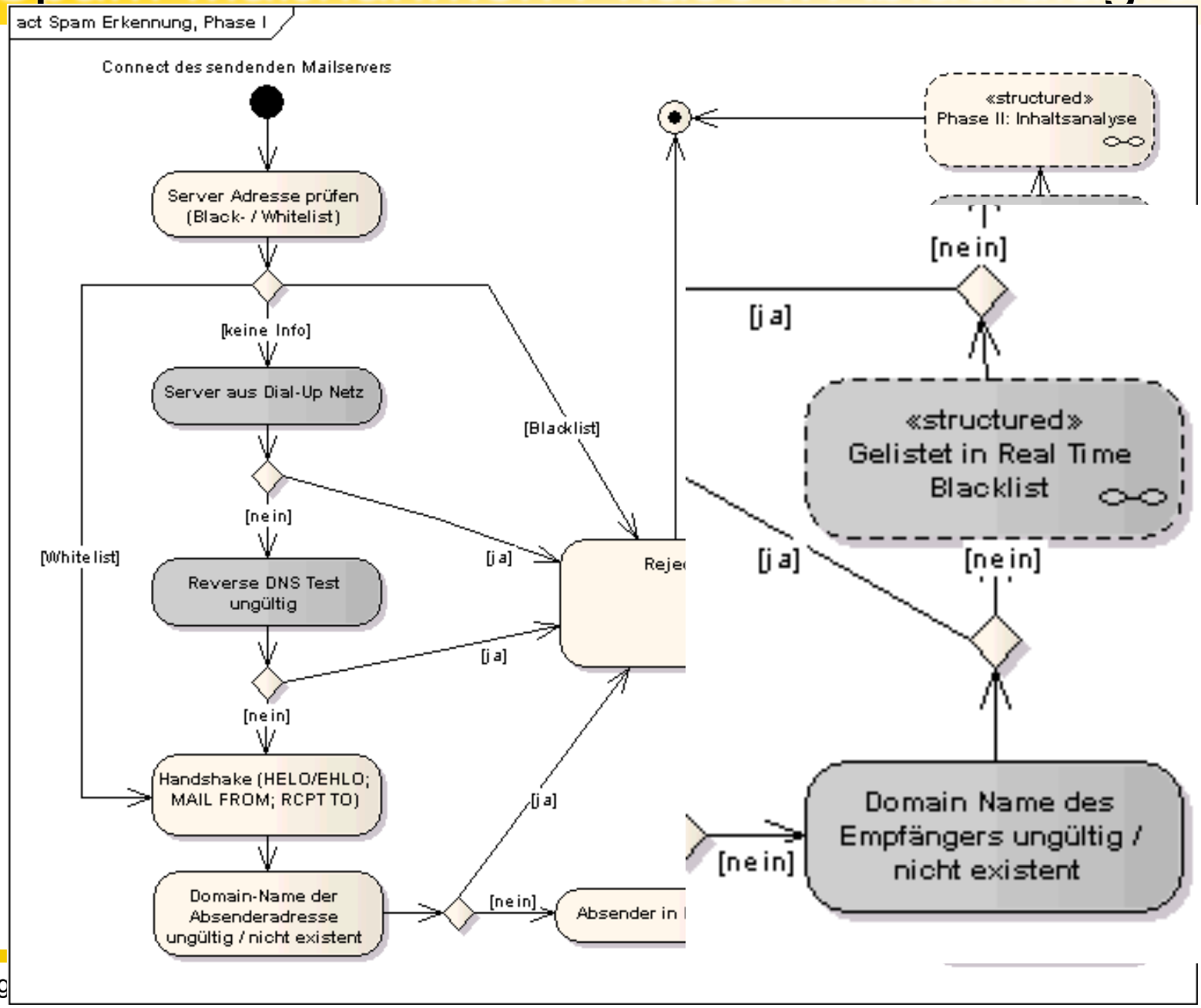


Spam-Maßnahmen Phase I: Ablaufdiagramm

- Erst jetzt wird der Handshake durchgeführt (HELO/EHLO)
- Test auf Gültigkeit/Existenz des Domain-Namens aus Handshake
- Test auf Absender in Blacklist (MAIL FROM:)
- Test auf Gültigkeit/Existenz der Empfänger-Domain (RCPT TO:)



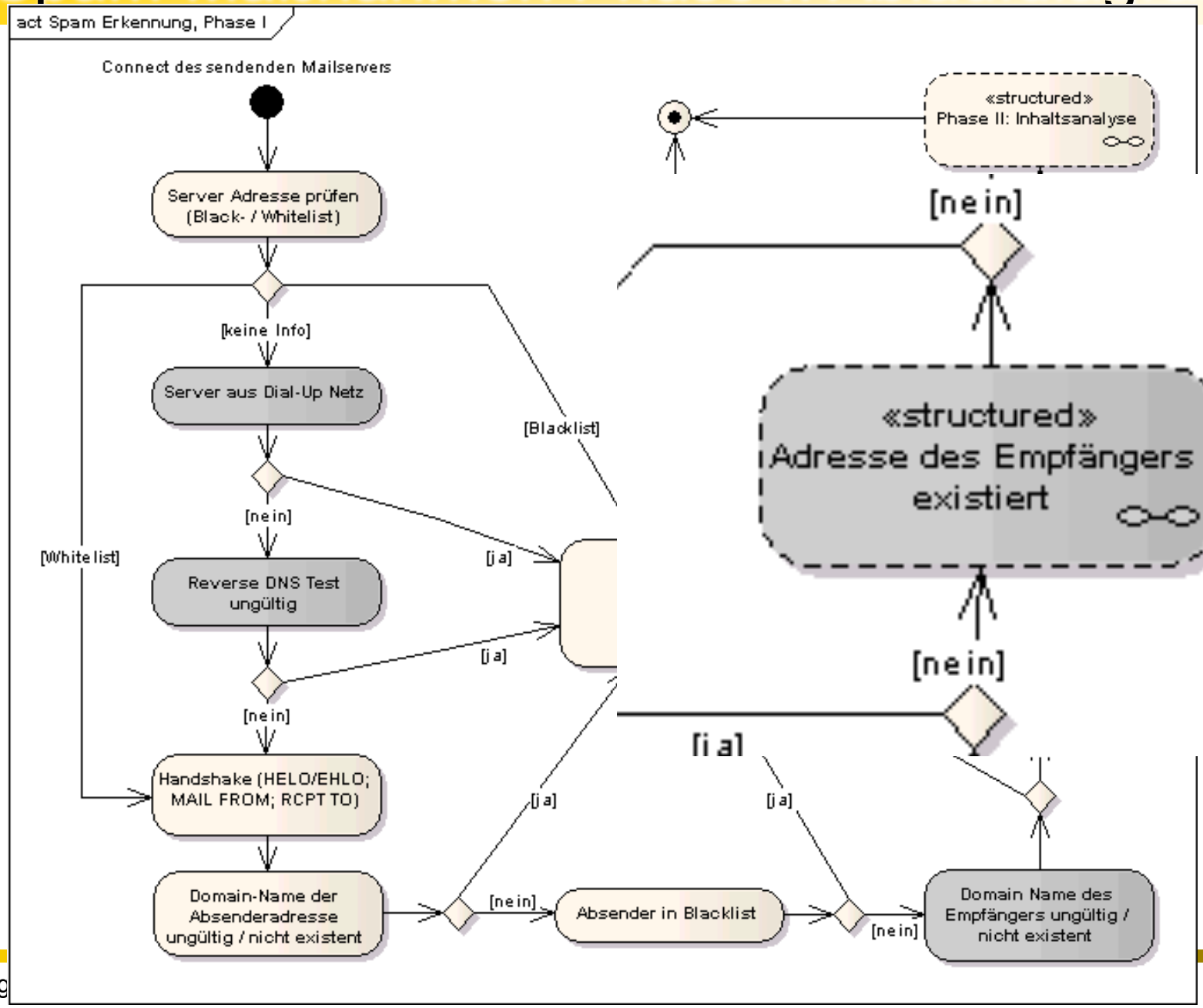
Spam-Maßnahmen Phase I: Ablaufdiagramm



Realtime Blacklisting (RTB)

- Bekannte Spammer bzw. deren IP-Adressen werden in spezieller DNS-Zone gespeichert
- Für jede Mail: Reverse DNS-Lookup in dieser Zone nach IP des Absenders
- Bei (spezieller) Antwort:
  - Absender ist gelistet
  - D.h. Spammer
- Bsp.: Mail von mailout.lrz-muenchen.de (129.187.254.112)
  - dig, host oder nslookup 112.254.187.129.pbl.spamhaus.org
    keine Antwort
- Bsp.: Mail von 217.227.25.81 (xxxxxxx.dip.t-dialin.net)
  - dig, host oder nslookup auf 81.25.227.217.pbl.spamhaus.org
    Antwort: Address: 127.0.0.11

Spam-Maßnahmen Phase I: Ablaufdiagramm



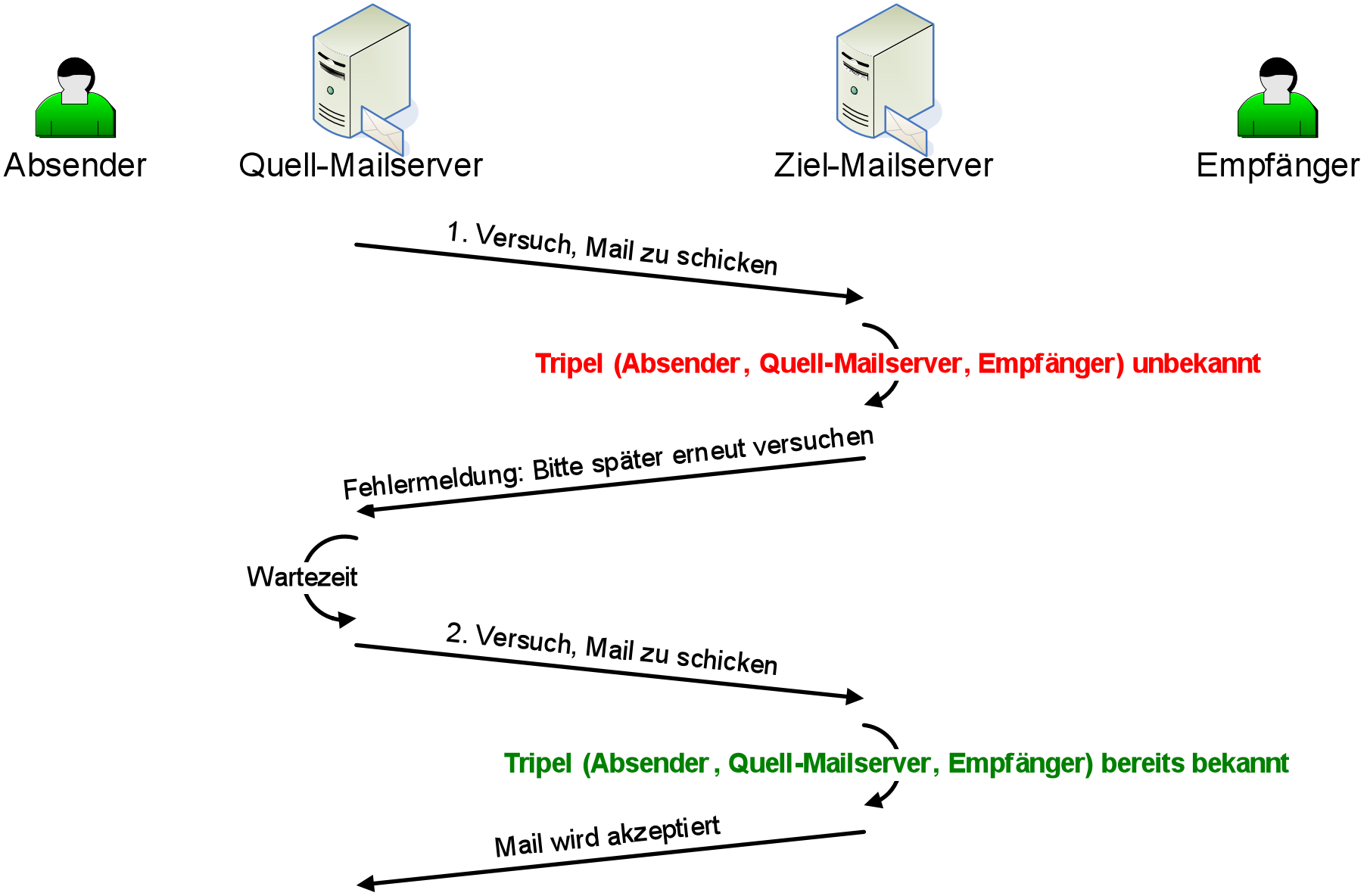
Test: Empfängeradresse existiert?

- Früher wurden alle Mails angenommen
  - Relays konnten nicht prüfen, ob Adresse gültig; diese Info ist nur bei Endsystemen (MTA) der Institute bekannt
  - Problem: evtl. Backscatter Spam bei nicht existenter Adresse
  - LRZ könnte auf Blacklist landen
- Heute: Adressprüfung
- Benutzerverwaltung basiert auf LDAP
- Mailadressen über LDAP im MWN abfragbar
- Falls LDAP nicht unterstützt:
  - SMTP Callout (Rückfrage beim Mail-Server der Empfänger Domain)
  - Teuer; deshalb nur Ausnahmefall

Greylisting

- Ursprüngliches Ziel: Last für Server Betreiber reduzieren
- Ausnutzung des „fire and forget“-Prinzips vieler Spammer
  - Spam wird nur einmal verschickt
  - Mailserver, der Mail nicht zustellen kann, versucht Zustellung mehrmals
- Idee: 1. Versuch der Zustellung wird abgelehnt
- Daten zur Erkennung einer „Mail-Relationship“:
  - IP-Adresse des sendenden Mail-Servers
  - Absenderadresse
  - Empfängeradresse
- Realisierung: Blocking Time
  - Mail-Relation erst nach Ablauf der Blocking Time akzeptieren
  - danach jede weitere Sendung in der Relation sofort akzeptieren
  - Häufig verwendete Werte: 50 Sek. bis 5 Minuten
  - LRZ: anfangs 15 Minuten, heute 29 Minuten

Greylisting - Ablaufdiagramm



Greylisting

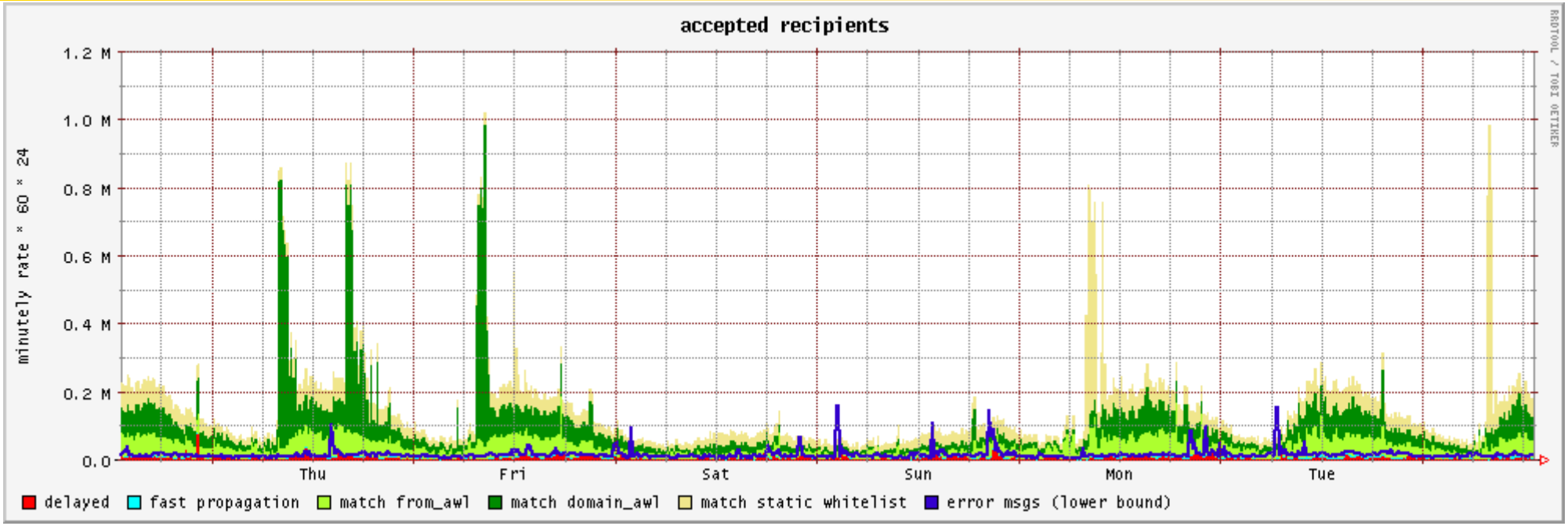
- Nachteil: Verzögerung der 1. Mail einer Relation (Blocking Time)
- Vorteil: In der Vergangenheit extrem wirkungsvoll (> 90 %)

- Probleme:
  - Phisher mit systematischen Retry-Versuchen (seit September 2006)
  - 4 Versuche mit 5 Minuten Abstand (daher die neue Grenze mit 29 Min)
  - Erste Spammer mit „langen“ Retry-Versuchen

⇨ Greylisting wird (früher oder später) seine Wirksamkeit verlieren

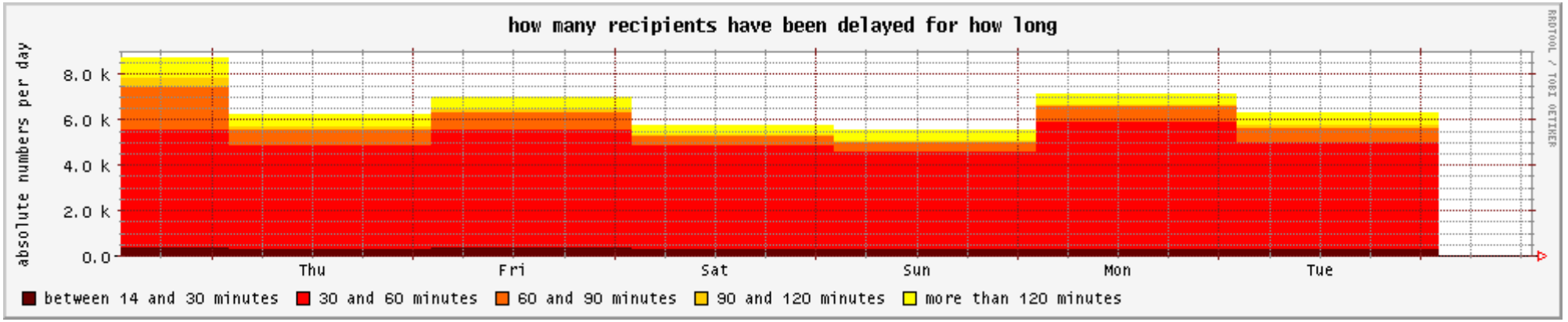
- Greylisting wurde durch die anderen (vorgestellten) Maßnahmen „nach hinten“ verschoben (vgl. Ablaufdiagramm)

Greylisting am LRZ: Akzeptierte Verbindungen



- Zeitraum: Mi. 18.10.06 bis Mittwoch 25.10.06
- awl = Automatic White List
  (LRZ verwendet IP-Adresse des Servers und Absenderadresse;
  Empfängeradresse wird nicht berücksichtigt)
- from_awl: awl der Absenderadressen
- domain_awl: awl der Domainnamen
- error msgs: Fremde Mailserver antworten mit Fehlermeldungen

Greylisting am LRZ: Verzögerung



- Zeitraum: Mi. 18.10.06 bis Mittwoch 25.10.06
- Im Normalfall liegt die Verzögerung bei 30 Minuten
  (für die erste Verbindung – eines Servers - der noch nicht in der awl ist)

Phase II: Inhaltliche Analyse

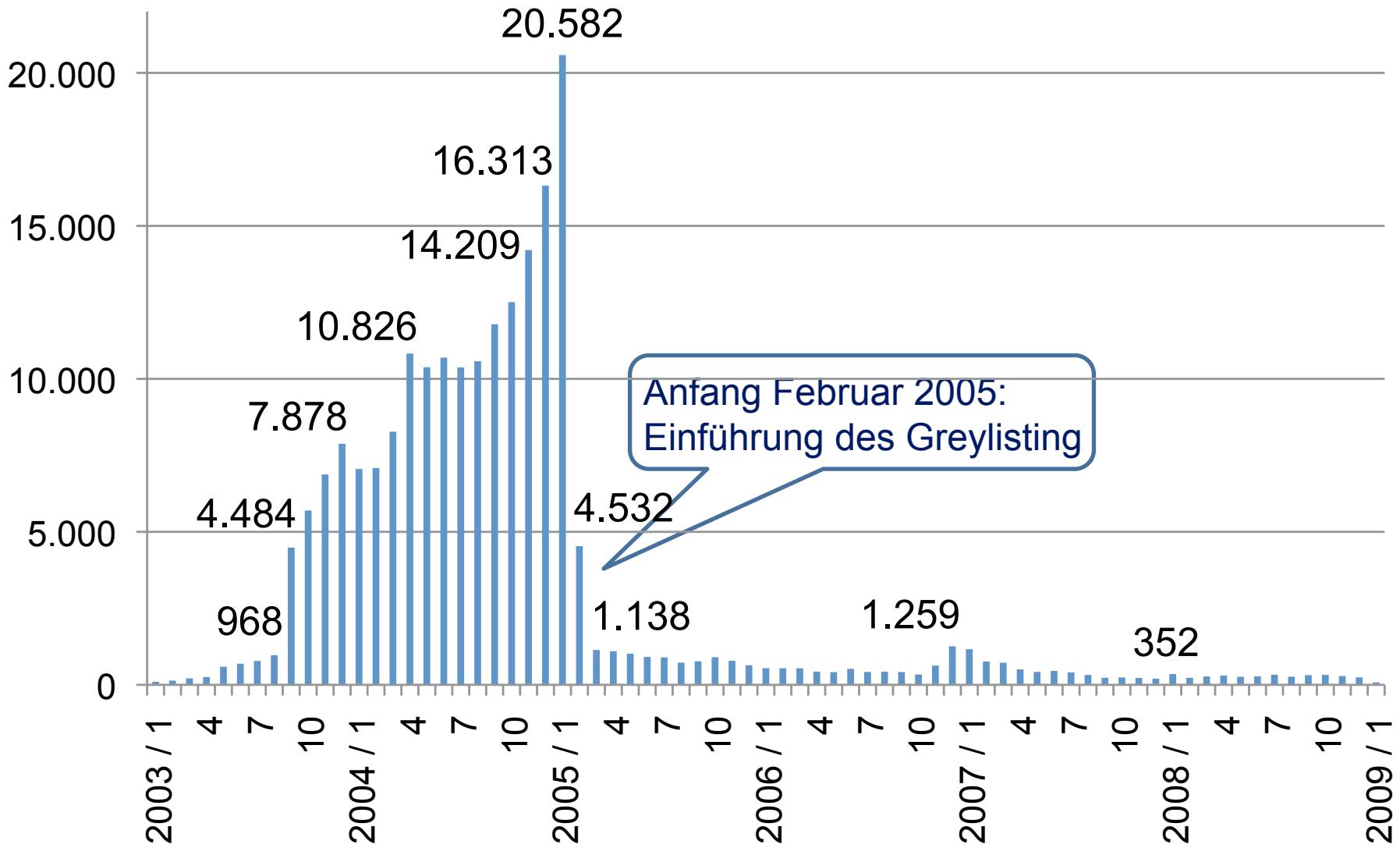
- Erst nach Abschluss der Phase I werden die eigentlichen Mail-Daten (DATA) angenommen
- Damit werden zusätzliche Ressourcen benötigt

- Phase II: Inhaltliche Analyse
- Spam-Filter: Spam-Assassin markiert potentielle Spam-Mails
- Viren-Filterung: Quarantäne mit Benachrichtigung

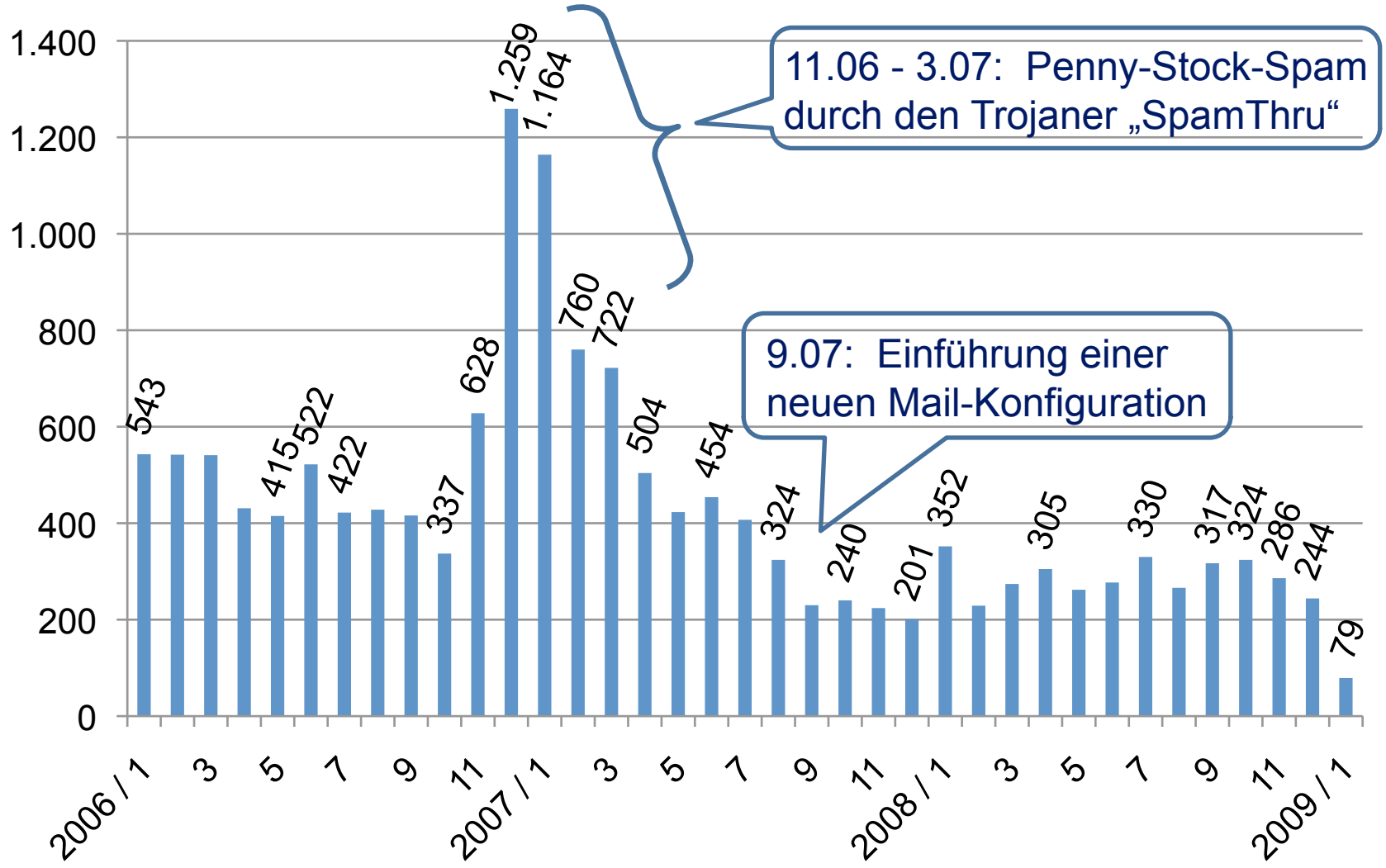
Inhalt (1)

4. Abwehrmaßnahmen im Münchner Wissenschaftsnetz (MWN)
   - Mail aus dem MWN ins Internet
     - .....
   - Überblick über das Simple Mail Transfer Protokoll
   - Mail Infrastruktur im LRZ
   - Mail aus dem Internet ins MWN
     - Phase I: „(Spam) Mail zurückweisen“
     - Phase II: Inhaltliche Bewertung und Markierung
   - Betriebserfahrungen und Statistiken

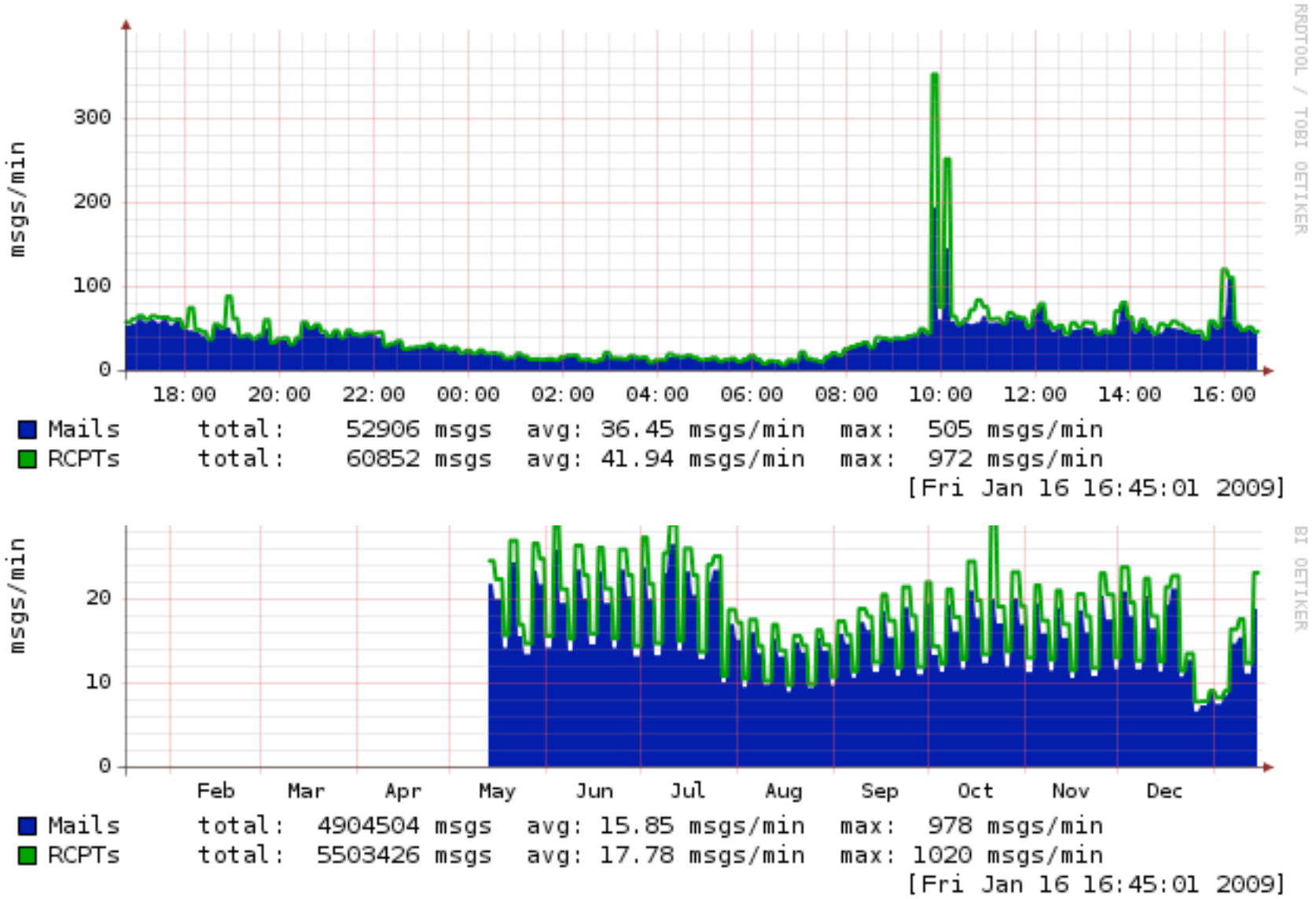
Persönliche Spam-Statistik eines LRZ-Mitarbeiters



Persönliche Spam-Statistik eines LRZ-Mitarbeiters



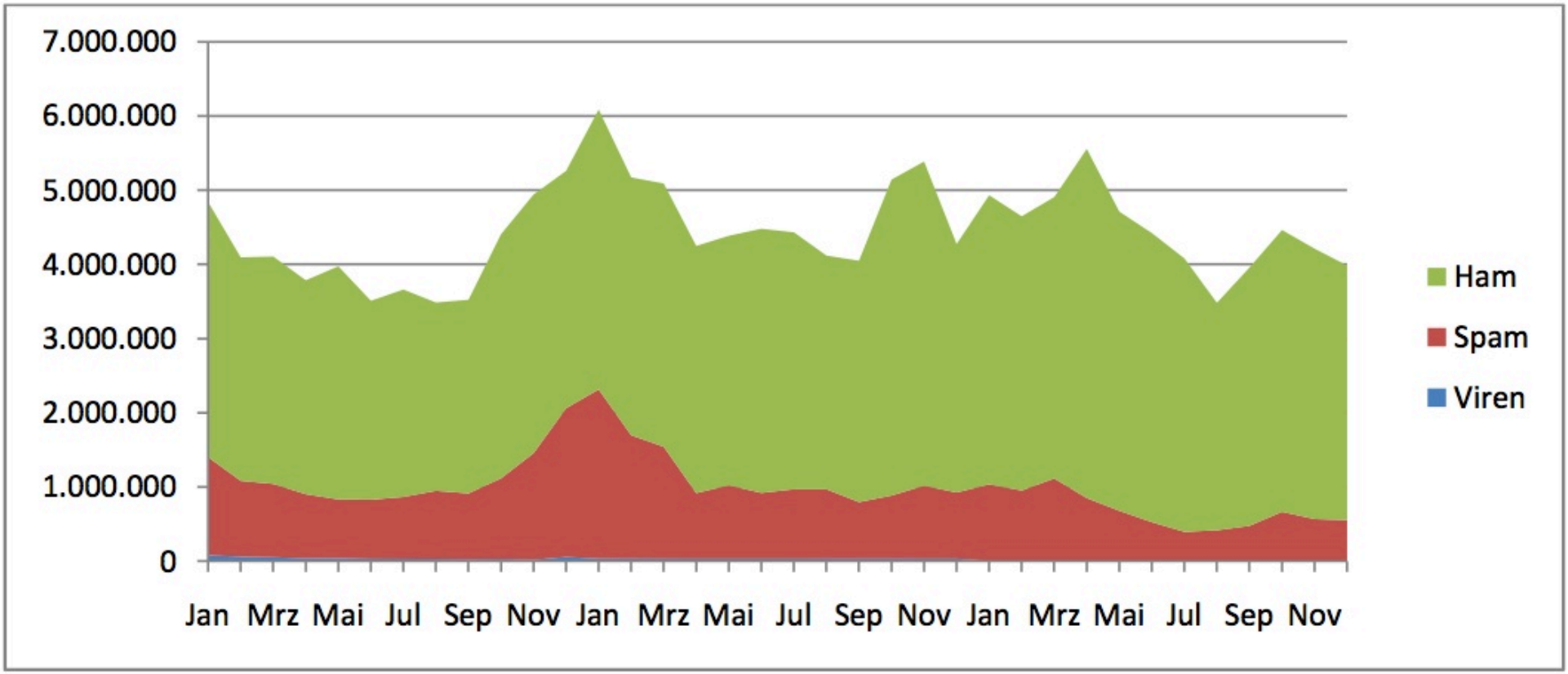
Betriebserfahrungen: Maildurchsatz



Mailnutzung im MWN

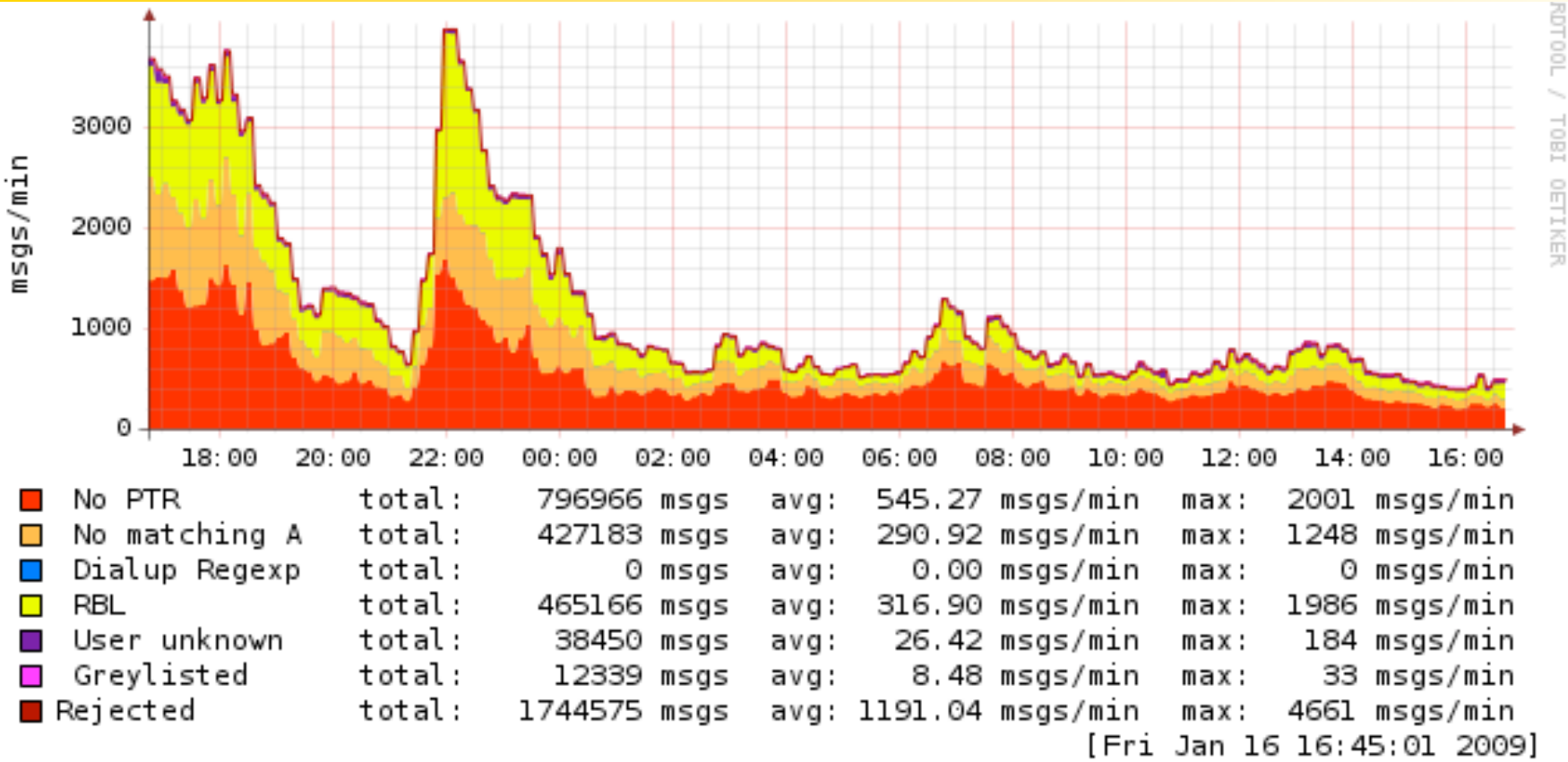
POP/IMAP-Server für …		Anzahl Benutzer
… Mitarbeiter der vom LRZ bedienten Einrichtungen (Mailserver „mailin“):		
Ludwig-Maximilians-Universität München	6.870	
Technische Universität München	5.347	
Bayer. Akademie der Wissenschaften (inklusive LRZ)	603	
Fachhochschule München	190	
andere bayerische Hochschulen	182	
andere wissenschaftliche Einrichtungen	1.862	15.054
… die Fakultät Physik der Technischen Universität München		2.434
… das myTUM-Portal der Technischen Universität München		33.746
… Studenten der Ludwig-Maximilians-Universität München (Campus^{LMU})		47.023
… Studenten anderer Münchner Hochschulen		1.657
Gesamt		99.914

Betriebserfahrung: Ham, Spam und Viren



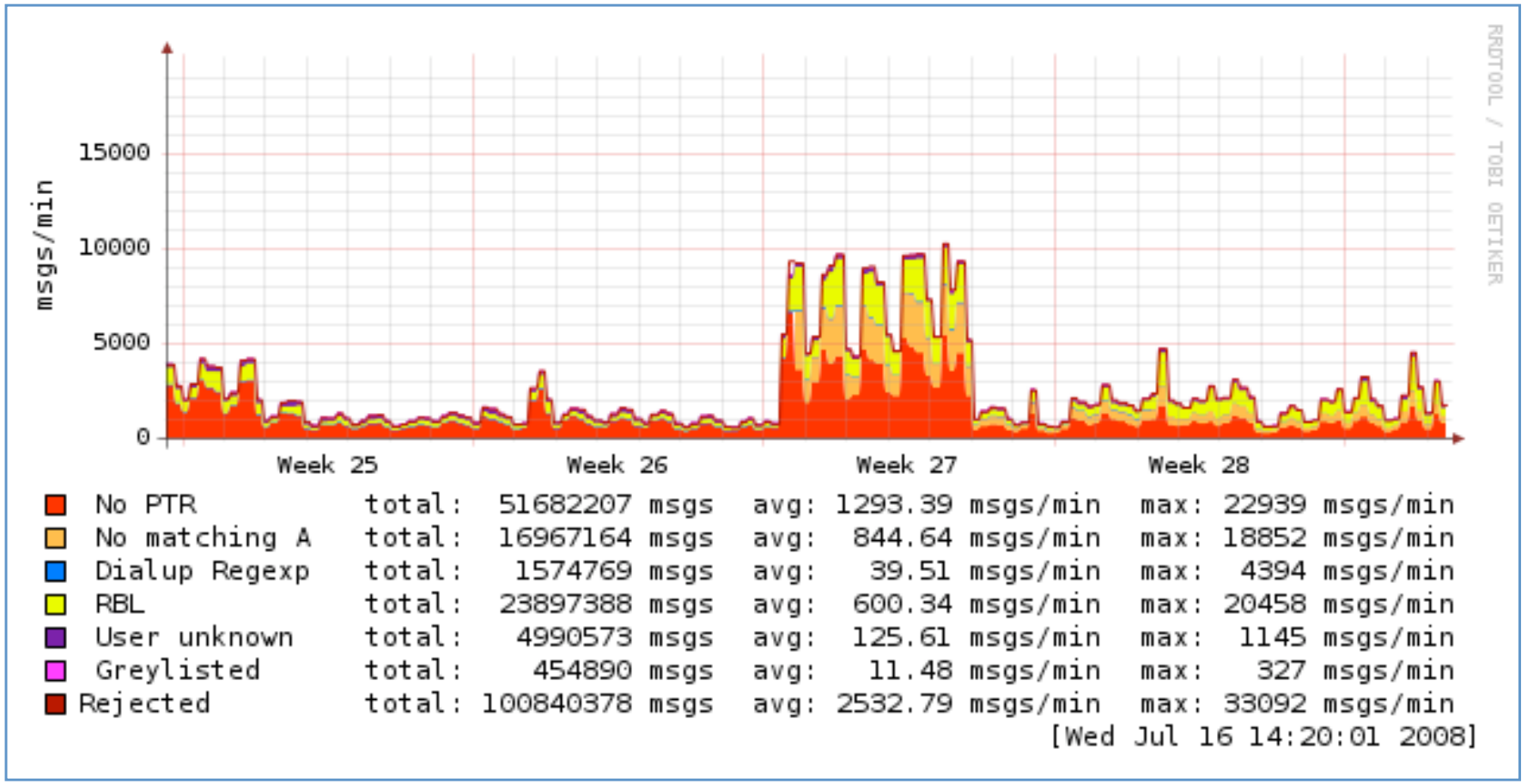
Ham-, Spam- und Virenaufkommen pro Monat in den Jahren 2006 bis 2008

Betriebserfahrungen: Anti-Spam Mechanismen



- No PTR = IP Adresse zu Name
- No matching A = keine Adresse zum Namen
- Dialup Regexp = eigene Regeln für Dialup-Erkennung
- RBL = Real Time Black Lists

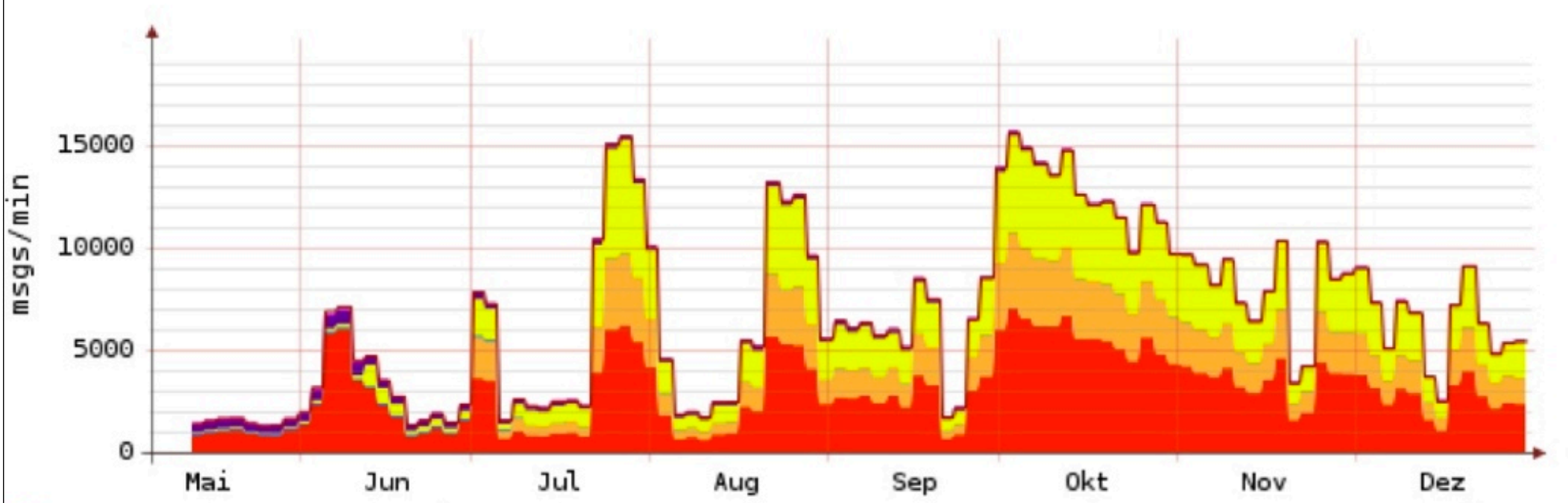
Betriebserfahrung: Spam-Wellen



- Mo. 30.06.08 12:00 bis Fr. 04.07.08 24:00
- Spitzenwert: 33.092 Mails / Min

Betriebserfahrung: Spam-Wellen

- Spam-Wellen



No PTR	total:	576929104	msgs	avg: 3005	msgs/min	max: 23952	msgs/min
No matching A	total:	271482782	msgs	avg: 1790	msgs/min	max: 19461	msgs/min
Dialup Regexp	total:	3862795	msgs	avg: 20	msgs/min	max: 5132	msgs/min
RBL	total:	383370020	msgs	avg: 1997	msgs/min	max: 22902	msgs/min
User unknown	total:	31307670	msgs	avg: 163	msgs/min	max: 17454	msgs/min
Greylisted	total:	3708989	msgs	avg: 19	msgs/min	max: 570	msgs/min
Rejected	total:	1275599318	msgs	avg: 6644	msgs/min	max: 55077	msgs/min

[Mon Feb 23 14:14:36 2009]

Betriebserfahrungen: Wirksamkeit der Mechanismen

