

# 1 ITSec 1

## 1.1 Einführung

### 1.1.1 Definitionen

**Computer Security** Sammlung von Tools gegen Hacker und zum Schutz von Daten

**Network Security** Maßnahmen, um Daten während der Übertragung zu sichern

**Internet Security** Maßnahmen, um Daten während der Übertragung durchs Internet zu sichern

**Korrektheit** System hält sich an Spezifikationen

**Sicherheit** Bei einem Angriff ist die Ausgabe nicht desaströs

### 1.1.2 Aktuell

- Sicherheit hat nicht oberste Priorität
- Fehlerhafte Implementationen
- Offene Netzwerke
- Viele nicht-technische Angriffe

### 1.1.3 Services

- Daten-Vertrauenswürdigkeit
- Daten-Integrität
- Authentifizierung
- Zugangskontrolle
- Kein Leugnen

### 1.1.4 Mechanismen

- Verschlüsselung
- Datenintegrität
- Digitale Signaturen
- Authentisierungsaustausch
- Einigung auf Schlüssel
- Traffic-Padding (bogus data)
- Routing-Control
- Notarization (3rd Party)
- Access-Control

### 1.1.5 Security Goals

- Vertrauenswürdigkeit (Lauschen, Traffic-Analyse)
- Integrität (Modifications, Replay, Nichtanerkennung)
- Erreichbarkeit (DoS, Delay)

### 1.1.6 Policies

- Security-Policy: was ist erlaubt, was nicht
- Security-Mechanism: Methode, um Security-Policies durchzusetzen

## 1.2 Symmetrische Verschlüsselung

### 1.2.1 Kerckhoff

Ein System soll sicher sein, auch wenn alles über das System (außer dem Key) bekannt ist

### 1.2.2 Blockchiffren

- AES: 128 Bit Blocklänge
- DES: 64 Bit Blocklänge
- AES-Key: 128, 192, 256 Bit

### 1.2.3 Modi

**ECB** ElectronicCodeBook, unabhängige Blöcke

**CBC** CipherBlockChaining, XOR jeden nächsten Block mit der Ausgabe des letzten

**CFM** CipherFeedbackMode, Plaintext wird nach der Verschlüsselung (IV) ge-XOR-t, Ausgabe ist Eingabe des nächsten Blocks

**OFM** OutputFeedbackMode, wie CFM, nur nicht nach dem XOR an den nächsten Block weiterleiten, sondern davor

### 1.2.4 AES

- Rundenbasiert
- $x + 1$  Rundenschlüssel
- Plaintext als Matrix
- Erster Schlüssel wird mit dieser Matrix ge-XOR-t
- 10,12 oder 14 Runden:
  - Subtypes: S-Box-Ersetzung
  - ShiftRows: zyklisches Shiften
  - MixColumns: Matrixmultiplikation mit einer festen Matrix
  - Key-Addition: XOR mit dem Rundenschlüssel

### 1.2.5 Angriffe

- Ciphertext-only
- Known-Plaintext
- Chosen-Plaintext
- Chosen-Ciphertext

## 1.2.6 PRNG

- Pseudo-Random-Number-Generator
- Keine beweisbare Sicherheit
- Das jeweils folgende Bit ist mit einer Wahrscheinlichkeit von 50% eine 0 bzw. eine 1
- Die Generatoren sind trotzdem deterministisch, können also nachvollzogen werden

## 1.2.7 RC4

- Zwei Phasen:
  1. Erzeuge eine Permutation der Zahlenfolge 0..255, dies geschieht mit Hilfe eines Keys
  2. Gebe so viele Zahlen aus, wie benötigt werden
- In der ersten Phase werden alle 256 Stellen des Arrays mit einer mehr oder weniger zufälligen anderen Stelle des Arrays vertauscht
- In der zweiten Phase wird  $i$  immer inkrementiert, die  $i$ -te Stelle des Arrays wird mit einer anderen getauscht und ausgegeben wird der Wert, der an der Stelle der Summen der Werte der beiden getauschten Stellen steht

## 1.2.8 Schwächen von Stream-Chiffren

- Keine Integritätsprüfung (XOR mit weiteren Plaintexten möglich)
- Problem bei Key-Wiederholung ( $K \oplus K = 0$ )

## 1.3 Integrity Protection

### 1.3.1 Hash-Funktionen

- Beliebige Stringlänge wird auf eine feste abgebildet
- Einfach und schnell zu berechnen
- Keyed oder unkeyed

### 1.3.2 Eigenschaften

**Pre-image-resistant**  $y = h(x) \rightarrow x'$  mit  $h(x') = y$  ist schwer zu finden,  $0,69 \cdot 2^n$  Versuche

**Second pre-image-resistant**  $x, h(x) \rightarrow x \neq x'$  mit  $h(x) = h(x')$  ist schwer zu finden,  $0,69 \cdot 2^n + 1$  Versuche

**Collision resistant**  $x, x'$  mit  $h(x) = h(x')$  ist schwer zu finden,  $1,18 \cdot 2^{\frac{n}{2}}$  Versuche

### 1.3.3 Random Oracle Model

- Ein Orakel generiert den Hash eines Strings und gibt ihn aus und speichert ihn zusammen mit dem String zusätzlich in einer Datenbank
- Bei einer erneuten Anfrage, guckt das Orakel erst in der Datenbank nach und gibt den Hash im Erfolgsfall aus oder es generiert einen neuen Hash

### 1.3.4 Geburtstagsprobleme

- Wahrscheinlichkeiten der Gleichheit von Geburtstagen von Stunden in einem Hörsaal
- Hiermit lassen sich die Wahrscheinlichkeiten der Eigenschaften von oben erklären

### 1.3.5 MD5

- Die Nachricht wird auf  $x \cdot 512$  Bit gepadded ( $100..000 + 64$  Bit Länge des originalen Texts)
- Vier 32 Bit-Buffer werden mit Standardwerten belegt und ein 512 Bit-Block wird in 16 Blöcke mit je 32 Bit aufgeteilt
- Pro 512 Bit-Block gibt es dann vier Runden mit je 16 Durchläufen. In jeder Runde wird eine andere (festgelegte) Funktion benutzt und pro Durchlauf wird ein anderer 32 Bit-Block mit reingerechnet.
- Die Ausgabe sind wieder vier 32 Bit-Blöcke, die als Startbuffer für den nächsten 512 Bit-Block dienen

### 1.3.6 MAC

- Integritätsprüfung und Authentifikation der Nachricht
- Keyed Hash
- HMAC: XOR den Key mit opad (fester Wert), konkateniere das mit dem Hashwert von ((Key XOR ipad) konkateniert mit der Nachricht) und bilde darüber dann den Hashwert
- CMAC: Teile den String in Blöcke von 64 oder 128 Bit und padde eventuell den letzten Block. Der Ablauf ist dann wie bei CBC, nur dass der letzte Block mit  $k_1$  oder  $k_2$  maskiert wird

### 1.3.7 Replay

- Abfangen und später wieder senden
- Gegenmaßnahmen: Nonces, Timestamps, Counter

### 1.3.8 Terms

- One-Way-Hash ist pre-image-resistant
- Kryptographische Hash-Funktionen haben alle drei Eigenschaften

## 1.4 Asymmetrische Verschlüsselung

### 1.4.1 Grundlagen

- $k$  hat in  $\text{mod } n$  ein multiplikatives Inverses, wenn sich  $k$  und  $n$  keine Primfaktoren teilen
- $\phi(n)$  = Anzahl der Zahlen  $\text{mod } n$ , die ein multiplikatives Inverses besitzen
- Euklidischer Algorithmus:  $ggT$  berechnen,  $ggT = 1 \Rightarrow$  multiplikatives Inverses existiert und lässt sich auch effizient berechnen (erweiterter Euklidischer Algorithmus mit weiteren zwei Variablen)

### 1.4.2 RSA-Key-Generation

- $p, q$  zwei Primzahlen mit  $n = p \cdot q$
- Wähle ein  $e$  (invertierbar)  $\text{mod } \phi(n)$
- Öffentlicher Schlüssel:  $(e, n)$
- Berechne ein  $d$  mit  $d \cdot e \equiv 1 \text{ mod } \phi(n)$
- Privater Schlüssel:  $d$
- Sicherheit: wenn  $n$  faktorisiert werden kann, kann  $\phi(n)$  berechnet werden:  $\Rightarrow$  klar,  $\Leftarrow$  löse ein Gleichungssystem mit " $n = p \cdot q$  und  $\phi(n) = (p - 1)(q - 1)$ "

### 1.4.3 Faktorisierung

- Kein effizienter Algorithmus vorhanden
- Sub-exponentielle Laufzeit

### 1.4.4 Effiziente modulare Exponentiation

- Schreibe den Exponenten  $K$  als Binärzahl auf und betrachte seine  $k$  Bitstellen als  $K_i$
- $x^K = \prod_{i=0}^{k-1} x^{K_i \cdot 2^i}$
- Bsp:  $x^{37} = x \cdot x^{2^2} \cdot x^{2^5} = (((((x^2)^2)^2)x)^2)^2 x$

### 1.4.5 Public Key Cryptography Standard (PKCS)

- Definiert zwei Verschlüsselungsmethoden für RSA-Verschlüsselung: RSAES-PKCS1-v1.5 und RSAES-OAEP
- Beide Methoden beschreiben, wie man einen Bitstring padded und in einen Integer umwandelt

### 1.4.6 RSAES-OAEP

- Hinzufügen von Zufallselementen, ergibt ein probabilistisches Schema
- Nachricht  $m$  wird mit  $k_1$  0en gepadded,  $k_0$  ist ein Zufallsbitstring, des Weiteren werden noch zwei Hashfunktionen benötigt
- Heraus kommen zwei Strings, die recht zufällig sind, mit denen der Empfänger aber die Nachricht rekonstruieren kann

### 1.4.7 Signaturen

- Das Tupel  $(m, s)$  ist einfach zu berechnen, aber nicht für ein gegebenes  $m$
- $(m, s), (m', s') \Rightarrow (m \cdot m', s \cdot s')$  ist valide
- Gegenmaßnahme: benutze Hashfunktionen als Grundlage für Signaturen

### 1.4.8 Angriffe auf Signaturen

- Key-Only-Attack
- Known-Message-Attack:  $(m, s)$ -Paare sind bekannt und der Angreifer versucht daraus neue Paare zu generieren
- Chosen-Message-Attack: ein Angreifer bekommt zu einer Nachricht  $m$  eine valide Signatur  $s$  durch den Signierer ausgestellt

### 1.4.9 Digital Signature Standard (DSS)

- Benutzt des Digital Signature Algorithm und SHA-1
- Basiert auf dem Problem des diskreten Logarithmus
- Man benötigt zwei Primzahlen  $p$  und  $q$  und einen Generator  $g$
- Der Signierer wählt ein  $a$ , berechnet daraus  $A = g^a \pmod p$  und veröffentlicht  $(p, q, g, A)$  als seinen öffentlichen Schlüssel
- Zum Signieren wird nun ein  $k$  gewählt, mit dem aus der Nachricht zwei Werte  $r$  und  $s$  berechnet werden
- Diese Werte kann der Verifier nun mit der Nachricht und dem öffentlichen Schlüssel verifizieren

### 1.4.10 Diffie-Hellman

- Generator  $g$  und Primzahl  $p$
- Alice wählt ein  $a \in \{1, \dots, p - 2\}$  und Bob wählt ein  $b \in \{1, \dots, p - 2\}$
- Alice schickt  $g^a$  an Bob und der schickt  $g^b$  an Alice
- Beide berechnen  $g^{ab}$

## 1.5 Zertifikate und PK Infrastruktur

### 1.5.1 Trusted und Third Party

- Der Public-Key ist bekannt
- Signatur der CA unter dem Public-Key des Users ist ein Zertifikat
- Central Authority (CA)
- Alle Parteien müssen der CA vertrauen
- Public-Key der CA muss vorher sicher übertragen werden (wenn nicht schon vorhanden)

### 1.5.2 Zertifikat

beinhaltet:

- Public-Key des Users
- ID des Users
- Issuer
- Signatur des Hashs der drei Strings

### 1.5.3 Möglichkeiten

- Oligarchy (Browser)
- Cross-Certification
- CA-Hierarchie (bottom-up und top-down)
- Anarchie (PGP)

### 1.5.4 Revocation

- User teilt der CA mit, wenn ein Zertifikat ungültig ist
- CA gibt regelmäßig CRLs aus (signiert durch die CA)
- revoked sind die Zertifikate, die noch nicht abgelaufen aber ungültig sind

### 1.5.5 X.509

- Standard für PK-Zertifikate und CRLs
- Wird in SSL/TLS, IPSec, ... benutzt
- Inhalt: u.a. Version, Seriennummer, Issuer, Validity, Algorithmus, PubKey, ID von Issuer und User
- Signatur über den Inhalt

### 1.5.6 Inhalt von CRLs

- Timestamp
- Signatur-Algorithmus
- Issuer
- Revoked Zertifikate (Liste)
- Signatur unter das Ganze

### 1.5.7 OCSP

- Online Certificate Status Protocol
- Online-Abfrage des aktuellen Status eines Zertifikats

## 1.6 IPSec

### 1.6.1 Beispielanwendungen

- Sichere Verbindung zwischen zwei Teilen einer Firma oder von zu Hause zu einer Firma
- Sichere Verbindung zwischen zwei Hosts
- IPSec ist normalerweise in Router/Firewalls integriert

### 1.6.2 Modi von IPSec

**Transport-Mode** Der Payload eines IP-Pakets wird gesichert

**Tunnel-Mode** Das komplette IP-Paket wird inklusive IP-Header gesichert

### 1.6.3 Authentication-Header

- Authentifiziert den Quellhost gegenüber dem Zielhost und sichert die Integrität des Inhalts
- Es benutzt eine Keyed-Hashfunktion, um eine MAC zu generieren, die auch im AH liegt, der sich zwischen IP-Header und Payload bzw zwischen dem neuen und dem alten IP-Header liegt
- Im AH befinden sich außerdem noch Informationen über die für die MAC verwendeten Algorithmen und Schlüssel (Security Parameter Index (SPI), 32 Bit-Feld), eine 32 Bit Sequenznummer zur Identifikation und weitere Daten
- Die veränderlichen Teile des IP-Header, wie zB die TTL, fließen nicht mit in die Berechnung der MAC ein
- Im Transport-Mode sind also der IP-Header, der AH, der TCP/UDP-Header und der Payload authentifizierbar
- Im Tunnel-Mode sind der äußere IP-Header, der AH, der innere IP-Header, der TCP/UDP-Header und der Payload authentifizierbar

### 1.6.4 ESP-Protokoll

- Encapsulating Security Payload
- Bietet Authentifizierung der Quelle, Integrität und Geheimhaltung
- Es fügt einen Header, einen Trailer und eine MAC zu einem Paket hinzu
- Im Tunnel-Mode verschlüsselt ESP zusätzlich noch den inneren IP-Header
- Der Header beinhaltet hier wieder eine Sequenznummer und die benutzten Algorithmen und Schlüssel (SPI) und befindet sich da, wo der AH-Header auch saß

- Der Trailer ist nur zum padding und hat ein Feld, das die Art des Payloads angibt
- Die MAC wird nicht über den äußeren Header berechnet
- Verschlüsselt ist alles zwischen ESP-Header (exklusive) und ESP-Trailer (inklusive)
- Authentifiziert ist alles das, was auch verschlüsselt ist + der ESP-Header

### 1.6.5 AH vs ESP

- ESP hält den Payload geheim
- AH liefert Integrity-Protection für den äußeren IP-Header

### 1.6.6 Virtual Private Network

- ESP wird häufig dazu benutzt, um ein VPN aufzusetzen
- Pakete werden von einem Netzwerk zu einem Gateway geleitet, sie enthalten TCP/IP-Header eines anderen Netzwerks
- Das gesamte Paket wird durch Verschlüsselung geschützt
- Der Gateway entschlüsselt die Pakete und kann sie an das gewünschte Ziel weiterleiten (VPN-Tunnel)

### 1.6.7 Security Associations (SAs)

- Einseitige Verbindung zwischen einem Sender und einem Empfänger
- Jede SA wird identifiziert durch den SPI und ob die SA für AH oder ESP ist
- SAs werden in einer Datenbank gespeichert und sie besitzen noch weitere Werte, wie zB die Gültigkeitsdauer oder welcher Mode verwendet wird (oder Wildcard)
- Eine SA wird beim Sender dazu benutzt, um die korrekten Schlüssel und Algorithmen für einen bestimmten Empfänger zu wählen. Beim Empfänger wird sie dazu benutzt, um korrekt entschlüsseln zu können.

### 1.6.8 Internet Key Exchange (IKE)

- Zwei Phasen:
  1. In der ersten Phase werden die Schlüsselalgorithmen festgelegt (Nachricht 1+2), die DH-Werte ausgetauscht (Nachricht 3+4) und die Teilnehmer authentisiert (Nachricht 5+6, PSK oder Certs)
  2. In der zweiten Phase werden eine oder mehrere IPSec-SAs ausgehandelt, dh es wird eine SA ausgewählt (Alg) und Schlüssel generiert

## 1.7 Authentication und Key-Agreement

### 1.7.1 Definition

- Entity-Authentication ist ein Prozess, bei dem ein Teilnehmer sich sicher ist, dass der Partner der ist, für den er sich ausgibt und dass er wirklich an dem Protokoll teilnimmt
- Korrektheit: sind A und B ehrlich, so wird die Identität akzeptiert
- Transferability: Auths können nicht weitergeleitet werden
- Impersonation: klar

### 1.7.2 Ways to prove

- Wissen
- Position
- Eigenschaften
- Besitz

### 1.7.3 Lamport's One Time Passwords

Siehe Payword

### 1.7.4 Small-k-Attack

- Man-In-The-Middle fragt nach einem größeren Passwort
- Er kann dieses dann durch hashen auf das korrekte runterrechnen

### 1.7.5 Challenge Response

- Basiert auf einem bekannten und geteilten Secret-Key
- Auch für Authentication mit Timestamp, Nonces, Zufallszahl

### 1.7.6 Unilaterale Authentication

- Eine Person authentifiziert sich gegenüber einer anderen durch die Verschlüsselung eines Timestamps und einer ID oder eines zuvor bekommenen zufälligen Integers (gegen Replay)
- Der Besitz des Keys zur Verschlüsselung ist Beweis der Identität
- Auch mit einer Keyed-Hashfunktion möglich

### 1.7.7 Mutual Authentication

- Wie unilaterale Authentifizierung, nur dass beide authentifiziert werden
- A sendet eine Zufallszahl, B verschlüsselt diese Zahl zusammen mit einer von ihm gewählten Zufallszahl
- A entschlüsselt das und verschlüsselt die beiden Zahlen umgekehrt wieder
- Auch mit Keyed-Hashfunktion möglich

### 1.7.8 Passwort

Lauschen, Vergleich, Raten, Sniffer, Keylogger, John, Passwort gehashed, Dictionary-Attack, Salt, gebräuchliche Passwörter, zufällige Passwörter werden häufig aufgeschrieben, aussprechbare Passwörter (müssen länger sein), proaktiver Passwort-Check, Default-Passwörter, Social-Engineering, verbessern durch zusätzliche Maßnahmen (Biometrie), Graphical-Passwörter (zufällig genug?, Shoulder-Surfing), Reset über weiteres Geheimnis oder Email (Social-Engineering?), Fragen persönlich oder sensibel

### 1.7.9 Public-Key-Authentication

- A sendet  $h(R), ID_B, E_B(R, ID_B)$  an B
- B antwortet mit  $R$

### 1.7.10 Key-Transport

Master-Key zur Generierung von Sessions

### 1.7.11 Third-Party

Siehe Kerberos

### 1.7.12 SecurID

- RSA-Hardware-Token (von der Firma RSA), das aus der SN und der aktuellen Zeit und AES ein Token erstellt
- Zugang mit Token, Username und Passwort

### 1.7.13 Key-Establishment-Protocol

- Secret-Key für Sessions oder um Sessions wieder auszunehmen
- Entweder sucht sich einer den Key aus (3rd Party möglich) oder er wird gemeinsam berechnet
- Session-Keys zur Sicherheit und damit nicht so viele Passwörter gespeichert werden müssen
- Charakteristika:
  - Freshness
  - Kontrolle über die Wahl
  - Effizienz des Aufbaus
  - Anforderungen an die Third-Party
- Perfect forward secrecy: vergangene Keys haben keinen Einfluss auf zukünftige
- Eigenschaften:
  - Implicit Key Authentication: nur 1-2 andere haben Zugang zu dem Key
  - Key-Confirmation: ein zweiter kennt den Key
- Explicit-Key-Authentication: beide Eigenschaften

## 1.8 Kerberos

### 1.8.1 Generell

- Wird ab Windows2000 benutzt
- Für Netzwerk-Applications um ihre Peers zu authentifizieren

### 1.8.2 Entities

- KDC, der aus den *Authentication Server* (AS) und *Ticket Granting Server* (TGS) besteht
- Realms: verschiedene administrative Domains
- Principal: Ressource

### 1.8.3 Ablauf

1. Ticket-Granting-Ticket: Alice logged sich ein und fragt am AS an. Der erstellt einen Client/TGS-Session-Key, packt diesen in das TGT (zusammen mit anderen Daten) und verschlüsselt das Ticket mit dem Key des TGS. Dieses Ticket schickt er zusammen mit dem Client/TGS-Session-Key verschlüsselt an Alice. Alice kann dann den Session-Key entschlüsseln und bekommt das Ticket.
2. Alice schickt das Ticket (nicht zusätzlich verschlüsselt) und den Namen der Ressource zusammen mit dem aktuellen Timestamp mit dem Client/TGS-Session-Key verschlüsselt an den TGS. Der kann das Ticket entschlüsseln und damit auch das andere Paket, das Alice geschickt hat. Er vergleicht dann die Timestamps und Alice' Namen und generiert ein Ticket für die Ressource, das einen Client/Ressource-Session-Key enthält und so ziemlich das gleiche wie eben. Der TGS schickt das Ticket dann zusammen mit dem Client/Ressource-Session-Key verschlüsselt mit dem Client/TGS-Session-Key an Alice.

3. Alice entschlüsselt wieder und erhält den Client/Ressource-Session-Key, mit dem sie den aktuellen Timestamp verschlüsselt an die Ressource schickt. Zusätzlich schickt sie das Ticket für die Ressource mit. Diese kann das Ticket entschlüsseln, wieder alles vergleichen und den Timestamp entschlüsseln. Die Ressource antwortet dann mit dem verschlüsselten und um eins inkrementierten Timestamp.

#### 1.8.4 KDC

- Speichert alle Secret-Keys, die es mit den Principals teilt
- Authentifiziert User, speichert also deren Usernamen und Passwords
- Viele KDCs möglich, die sich die Login-Daten von einer zentralen DB holen

#### 1.8.5 Verschiedene Realms

- User@Realm1 will Ressource in Realm2 nutzen
- Der KDC in Realm2 wird dann zur Ressource in Realm1
- IDs: Name, Instance, Realm

#### 1.8.6 Kerberos 4

- DES + PCBC
- Nachteile: keine freie Auswahl der Algorithmen
- Ticket gilt nur knapp über 21 Stunden

#### 1.8.7 Kerberos 5

- Mehrere Algorithmen zur Verschlüsselung und zur Integrity-Protection zur Auswahl
- Principal-Name: Name, Realm
- Sequenznummer statt Timestamp möglich
- Ticket-Delegation möglich
- Anfang und Ende statt Dauer
- Cross-Realm-Authentication: Tree der KDC, reduziert die KDC-Keys, Tickets für alle KDC auf dem Weg

### 1.9 SSL und TLS

#### 1.9.1 Grundlagen

- Liegt zwischen Application- und TCP-Layer
- Server- oder Server/Client-Authentication
- Datengeheimhaltung und -integrität
- SSL = Secure Socket Layer
- TLS = Transport Layer Security
- Connection: flüchtig, assoziiert mit einer Session, zB HTTP-Request
- Session: Verbindung zwischen Client und Server, kann viele Connections haben, definiert Sicherheitsparameter

## 1.9.2 Session State

- SessionID
- Peer-Cert
- Kompressionsmethode
- Cipher-Spec (Verschlüsselungs- und Integritätsalgorithmus)
- Master-Secret
- IsResumable

## 1.9.3 Connection State

- Zufallszahl von Server und Client
- MAC-Key für Server und für Client
- Verschlüsselungsschlüssel für Server und für Client
- IV für CBC
- Sequenznummer

## 1.9.4 Handshake: KeyExchange

- Generierung des Master-Secrets (verschiedene KeyExchangeMethods):
  - RSA: Client generiert ein Geheimnis und verschlüsselt das mit dem Public-Key des Servers, der kann es dann mit seinem Private-Key entschlüsseln
  - Anonymous DH: DH ohne Signaturen
  - Ephemeral DH: DH mit Signaturen
  - Fixed DH: feste öffentliche DH-Werte (signiert durch CA)

So erhält man das pre-Master-Secret. Aus diesem lässt sich das Master-Secret berechnen:  $MS = \text{PRF}(pMS, \text{"master secret"}, RND1, RND2)$

## 1.9.5 Phase 1

- Client schickt:
  - Unterstützte Cipher-Suites
  - SessionID (wenn resumable)
  - Unterstützte Kompressionsmethoden
  - RND
- Server antwortet:
  - SessionID (eventuell neu)
  - Cipher-Suite (Verschlüsselungsalgorithmus, MAC-Alg., Verschlüsselungsschlüssel, MAC-Schlüssel)
  - Kompressionsmethode
  - RND

## 1.9.6 Phase 2

Der Server schickt abhängig von der KeyExchangeMethods sein Zertifikat und/oder seinen (signierten) Key an den Client

### 1.9.7 Phase 3

Der Client schickt abhängig von der KeyExchangeMethods sein Zertifikat und/oder seinen (Teil des) (signierten) Keys an den Server

### 1.9.8 Phase 4

Beide stimmen dem Protokoll zu (ChangeCipherSpec) und senden an den jeweils anderen einen Hash über die gesamte Kommunikation

### 1.9.9 Resumption

- Viel Overhead beim Generieren einer Session
- Nochmaliges Benutzen des Master-Secrets durch eine bekannte SessionID

### 1.9.10 Protokolle

- SSL-Alerts
- Record-Protocol: Text → Komprimiere(Text) → hänge MAC des komprimierten Texts dran → verschlüsseln → Record-Protocol-Header dran → wegschicken
- ChangeCipherSpec-Protokoll
- Handshake-Protokoll

### 1.9.11 Änderungen in TLS 1.2

- Variable PRF
- Neue Cipher-Suites, IDES und DES entfernt

### 1.9.12 Designing Applications

- Alle Zertifikate und Keys laden
- RND generieren
- Cipher-Suites laden
- Verifizieren der Certs (trusted Signatures, Daten, IDs, Revocations)

### 1.9.13 Access Control und Server Authentication

- Certs überprüfen, aber der User legt das Trust-Level für CAs fest
- Server-ID im Zertifikat
- Der User ist meist doof, es lässt sich für dumme User vieles faken (Adressbar, Lock, ...)
- Authentifizierung des Clients durch Zertifikat oder durch Username/Passwort

## 1.10 Secure Email

### 1.10.1 Grundlagen

- SMTP zum Senden und zur Kommunikation zwischen Servern
- IMAP/POP für Abrufe des Clients
- SMTP: Simple Mail Transfer Protocol
- IMAP: Internet Message Access Protocol
- POP: Post Office Protocol

### 1.10.2 MIME

- Ursprünglich waren Mails für 7 Bit Ascii
- MIME wandelt nicht-Ascii in Ascii um
- Es fügt einen neuen Header zum Email-Header hinzu:
  - Version
  - ContentType
  - Verschlüsselungstyp
  - ContentID
  - Content-Description

### 1.10.3 Ent-to-End

- Geheimhaltung
- Authentifizierung
- Integrität
- Nicht leugbar (scheiß Wort)
- Symmetrische Verschlüsselung, Schlüssel wird vom Sender gewählt und mit dem öffentlichen Schlüssel des Empfängers verschlüsselt, wird zusammen mit der symmetrisch verschlüsselten Nachricht verschickt
- Eventuell noch die Signatur unter den Hash der Nachricht reinpacken

### 1.10.4 S/MIME

- Secure Multipurpose Internet Mail Extension
- Fügt weitere Content-Types hinzu
- Standard-Algorithmen
- Benutzt X.509

### 1.10.5 Security

- End-to-End durch S/MIME oder PGP
- Hop-by-Hop durch STMPs, POP3s, IMAPs

### 1.10.6 PGP

- Zwei Keyrings (pubring, secring)
- Zertifikate können durch jeden User signiert werden
- User vergibt Trust-Level an User (lokal) und an Zertifikate (global)
- Key Legitimacy: Summe der Produkte zwischen Trust-Level des Zertifikats und des Users
- Verschiedene Paket-Typen: Session-Key, Signatur-Paket, Private-Key Paket, Literal Paket, UserID Paket, ...

## 1.11 SSH

### 1.11.1 Grundlagen

- SecureShell
- Unter anderem für sichere Remote-Logins
- Secure shell logins
- TCP port forwarding
- X11 forwarding

### 1.11.2 Komponenten

- SSH-Transport-Layer-Protocol: Aushandeln der Version, Client authentifiziert den Server durch PK, Aushandeln der Algorithmen, DH-Key-Exchange
- (User)-Auth-Protocol: Server authentifiziert den Client
- Connection-Protocol: Client sucht die Anwendung aus (Shell, Forwarding, X11)

### 1.11.3 Algorithmen

- Darauf wird sich während der Schlüsselgenerierung geeinigt
- Diverse symmetrische Algorithmen (Liste der möglichen wird übertragen)
- Verschiedene Algorithmen für den Hin- und Rückweg möglich
- RSA und DSS müssen für Signaturen unterstützt werden

### 1.11.4 Key Re-exchange

- Erwirkt einen neuen DH-Key
- Eventuell Wechsel des Algorithmusses

### 1.11.5 DH Schlüsselaustausch

- Der Client schickt seinen öffentlichen DH-Teil rüber. Der Server berechnet den DH-Key und hashed die vorherige Kommunikation.
- Der Server signiert den Hash und schickt seinen DH-Teil mit dem Hash an den Client

### 1.11.6 IVs

- Zur Generierung von Zufallszahlen
- Werden durch Hashen unterschiedlicher Nachrichten zwischen Client und Server generiert

### 1.11.7 Authentication-Methods

- Public-Key
- Passwort
- Host-basiert

## 1.12 Phishing

### 1.12.1 Varianten

- Spear-Phishing
- Whaling

### 1.12.2 Ablauf

- Links in Emails, IM, SMS mit falscher Login-Seite
- DNS-Cache-Poisoning
- XSS
- URLs mit Rechtschreibfehlern, Umleitungen oder als Subdomain

### 1.12.3 Webseiten

- Design abgeguckt
- Seite in Seite
- Werden auf gehackte Server gelegt oder Portweiterleitung auf andere Server

### 1.12.4 Gegenmaßnahmen

- Aufpassen
- Toolbars, Plugins
- User lassen sich jedoch wenig vorschreiben

### 1.12.5 Trend

- Phishing-based Key-Logger: spring nur bei bestimmten Seiten an
- Auch zusammen mit DNS-Cache-Poisoning

## 1.13 DNS und DNS-Security

### 1.13.1 Überblick

- DNS ist hierarchisch
- Jede Domain hat mindestens einen zuständigen DNS-Server
- Abfragen können rekursiv oder nicht rekursiv sein: Teilantwort vom DNS-Server oder komplette Auflösung der URL

### 1.13.2 Server

- Server sind die Verantwortlichen für den Teil des Baumes, den sie komplett kennen (Zone)
- Können zusätzliche Resource-Records cachen

### 1.13.3 Resolver

- Fragen nach Resource-Records
- Haben Zugang zu mindestens einem DNS-Server
- Bekommen eine komplette Antwort oder werden weiterverwiesen

#### 1.13.4 Cache

- Schneller Zugriff auf oft aufgerufene Seiten
- Schnelle Fehlerrückgabe bei häufig falschen URLs
- Timeout für einen Eintrag

#### 1.13.5 Angriffe

- Anfragen haben eine zufällige TransactionID. Die kann von Angreifern geraten werden und vor der DNS-Antwort beantwortet werden.
- Pharming (Cache-Poisoning)

#### 1.13.6 Motive

- Host unerreichbar machen
- Traffic umleiten
- Zensur

#### 1.13.7 Cache-Poisoning

- Einem Resolver eine gefakte Antwort unteschieben, damit er diese cached
- Angreifer stellt eine DNS-Anfrage für eine Ziel-URL und antwortet direkt darauf mit einer Evil-IP (Raten der TransactionID). Antworten anderer Server werden verworfen.
- Das Opfer bekommt dann beim Aufruf der URL die gecachte IP

#### 1.13.8 DNS-Rake Angriff

- Anfrage an x.domain.tld
- Direkte Antwort mit diversen QueryIDs und einem Address-Ressource-Record für www.domain.tld

#### 1.13.9 Gegenmaßnahmen

- Source-Port randomisieren
- IPSec + DNS, erzeugt aber zu viel Overhead
- DNSSec: signiere die Antwort mit dem privaten Schlüssel der Zone (wer signiert die Schlüssel?)

#### 1.13.10 Intrusion Detection

- Pakete überwachen
- Zweite Antwort abwarten
- ...

## 2 ITSec 2

### 2.1 Geld

#### 2.1.1 Kreditkartenzahlung

1. Karte ausgeben
2. Karte dem Händler geben
3. Autorisierung durch die Bank (H), VISA, Bank (K)
4. Beleg ausstellen
5. Beleg unterschreiben
6. Beleg verschicken (Bank (H))
7. Betrag ausgleichen zwischen Bank (H), VISA, Bank (K)
8. Kunde erhält Rechnung

#### 2.1.2 Kartentypen

**Debit Card** Ec-Karte, Kunde muss ein Konto haben, Transaktionen in Echtzeit

**Charge Card** Kunde zahl am Monatsende, Geld wird abgebucht, kein minimaler Betrag zu zahlen

**Credit Card** Kunde muss nicht direkt bezahlen, auch nicht am Monatsende

**Smart Card** Karte mit Chip, zB Geldkarte

#### 2.1.3 Unterscheidung nach...

- prepaid / postpaid / pay-now
- online / offline

#### 2.1.4 Fraud (Credit Card)

- Hotlists (lokal) und Limit
- Heute: Autorisierung durch Issuer und globale Hotlist
- Früher: Magnetstreifen konnte mit Kartenummer und Ablaufdatum erstellt werden
- Danach: MAC wurde mit privatem Schlüssel des Issuers erstellt (CVV) → skimming

#### 2.1.5 SSL / TLS beim Onlineshopping

- Ablauf wie gehabt
- Vorteile: Teil des Browsers, der User ist daran gewöhnt
- Nachteile: Vertrauen des Händlers nötig, Bank bekommt zu viel mit, keine User-Auth, Phishing-Probleme
- SSL setzt auf dem TCP-Layer (Transport-Layer) auf, unter dem Application-Layer

### 2.1.6 Anforderungen für e-payment

- Authorisierung (Betrag, Kunde, Bank)
- Datenvertraulichkeit und Authentizität
- Verfügbarkeit und Verlässlichkeit
- Atomicity
- Privacy (Kontrolle über die eigenen Daten)

### 2.1.7 CVV2

- auf die Karte gedruckt
- wird nicht beim Händler gespeichert

### 2.1.8 Static Data Authentication (SDA)

- Karte wird ins Terminal eingeführt
- Sie schickt dann das Zertifikat der Bank, Accountnummer, zusätzliche Daten und die Signatur an das Terminal
- Das Terminal verifiziert die Signatur und der Händler gibt den Betrag ein
- Das Terminal fragt nach der PIN, der Kunde gibt sie ein und das Terminal schickt sie an die Karte
- Die Karte überprüft die PIN und generiert eine MAC aus der Händler-ID, Betrag, Seriennummer und Nonces mit einem symmetrischen Schlüssel, den sie mit der Bank teilt
- Das Terminal geht optional online und lasst die MAC durch die Bank überprüfen

### 2.1.9 Dynamic Data Authentication (DDA)

- Jede Karte hat ein PK-Schlüsselpaar
- Das Terminal sendet eine zufällige Challenge an die Karte, diese signiert sie und schickt sie zurück
- Das Terminal sendet daraufhin alles mit dem Public-Key verschlüsselt an die Karte
- sonst wie SDA

### 2.1.10 Combined Data Authentication (CDA)

- MAC wird mit dem Private-Key der Karte signiert
- Sonst wie DDA

### 2.1.11 SET

duale Signatur: Nachricht in zwei Teile teilen, beide hashen und die konkatenierten Hashwerte nochmals hashen. Der letzte Hash wird zusätzlich zum normalen mitgeschickt (an die Nachricht + Hash gehängt). So kann jeder überprüfen, ob zwei Nachrichten zusammen gehören (durch den gemeinsamen Hashwert). Ablauf:

1. Kunde erstellt für die Bank ein eine Box mit den Zahlungsinfos und schickt diese zusammen mit den Bestellinfos (beide dual-signiert) verschlüsselt an den Händler
2. Dieser kann die Bestellinfos überprüfen und schickt den Betrag zusammen mit den Zahlungsinfos an die Bank (wahrscheinlich noch mit der dualen Signatur der Bestellinfos)
3. Die Bank öffnet die Bezahlinfos, vergleicht alles (Betrag, duale Signatur) und erstellt für den Händler ein signiertes, verschlüsseltes und wieder signiertes Token als Ergebnis (wenn der Vergleich erfolgreich war)

4. Der Händler bestätigt den Transfer dann mit seiner Signatur unter das verschlüsselte Token, er kann also die Ware raus schicken
5. Die Bank bestätigt daraufhin die erfolgreiche Transaktion

### 2.1.12 E-Cash

- Münze mit Signatur der Bank, Münzen haben Seriennummern
- Händler zahlt sie erst ein und gibt dann die Ware aus (Bank überprüft, ob die Münze schon vorhanden ist und damit dupliziert wurde)
- Bank muss viel speichern. Abhilfe: ein Ablaufdatum, die Datenbank wird so nicht unendlich groß → Traceability

### 2.1.13 DigiCash

- Bank hat einen RSA-Public-Key  $(e,n)$
- Kunde generiert eine Münze  $(SN, Exp, Val)$  und berechnet den Hash davon
- Er wählt eine Zahl  $r$  und schickt  $h \cdot r^e$  an die Bank
- Diese signiert es und schickt es zurück
- Der Kunde kennt ja das  $r$  und erhält durch Division die blind-signierte Münze

### 2.1.14 Cafe

- BlindSig mit Trusted-Center
- Der User ist nicht nachvollziehbar, solange er nicht betrügt

### 2.1.15 ATM

- Kontonummer PAN,  $C = E_{\text{priv.Bank}}(PAN)$ , Natural PIN sind die ersten 4 Zeichen von C
- Der Kunde kann eine PIN frei wählen, hierfür wird der Offset zur Natural-PIN bei der Bank gespeichert

### 2.1.16 Micropayment - PayWord

- Der Kunde gibt dem Broker seine Kontodaten und anderes, dieser erstellt ein Zertifikat für den Kunden als Beweis, dass er sein Kunde ist
- Der Kunde wählt für jeden Händler einen zufälligen Wert, hashed diesen  $n$  mal und schickt ihn signiert an den Broker und den letzten Hash an den Händler
- Müssen nun  $i$  Münzen bezahlt werden, so schickt der Kunde die folgenden  $i$  Hashwerte an den Händler. Dieser kann den Wert  $i$  Mal hashen und sollte dann auf den ihm bekannten Wert kommen. Wenn ja, verschicke Ware.
- Abends kann der Händler seine gesammelten Münzen an den Broker schicken, der überprüft diese noch einmal und schreibt dem Händler dann den Wert gut.

## 2.2 Biometrie

### 2.2.1 Biometrie zur

- Authentifizierung (positive Recognition)
- Identifizierung (negative Recognition)

## 2.2.2 Eigenschaften von biometrischen Daten

Können nicht...

- verloren werden
- vergessen werden
- gestohlen werden
- manipuliert werden
- geteilt werden

## 2.2.3 Biometrische Merkmale

Iris, Ohr, Gesicht, Handabdruck, Handgeometrie, Venenabdruck, Stimme, Gang, Thermogramm, Fingerabdruck, Tastenanschlag, Geruch, Unterschrift

## 2.2.4 Komponenten

- Sensormodul
- Feature-Extractions- und Qualitätsbewertungsmodul
- Matching- und Decisionmaking-Modul
- System-Datenbank

## 2.2.5 Ergebnisse nicht immer gleich

- Veränderungen im Alter
- Temperatur und Licht
- Teilweiser Fingerabdruck
- Krankheiten

## 2.2.6 Variationen

**Intra-Class-Variation** Variabilität zwischen Feature-Sets einer Person

**Inter-Class-Variation** Variabilität zwischen Feature-Sets mehrerer Personen

## 2.2.7 Scores

**Similarity-Score** Grad der Übereinstimmung zweier Datensets

**Match-Score** Grad der Übereinstimmung zweier Sets einer Person

**Impostor-Score** Grad der Übereinstimmung zweier Sets von verschiedenen Personen

## 2.2.8 Raten

**FAR** False Accept Rate

**FRR** False Reject Rate

- Videoüberwachung: hohe FAR, niedrige FRR
- High-Security-Apps: niedrige FAR, hohe FRR

### 2.2.9 Klassifizierungen

**Schaf** niedrige Intra-Class-Variabilität: geringe FAR und FRR

**Ziege** hohe Intra-Class-Variabilität: hohe FRR

**Lamm** niedrige Inter-Class-Variabilität: hohe FAR

**Wolf** User kann seine biometrischen Eigenschaften manipulieren

### 2.2.10 Biometrische Charakteristiken

- Universalität
- Eindeutigkeit
- Permanent vorhanden
- Messbarkeit
- Performanz
- Akzeptanz
- Schwierig zu umgehen

### 2.2.11 Verwundbarkeit von Systemen

- Umgehen der letzten beiden Module
- Verdeckte Datenaufnahmen und Replay
- (un)freiwillige Mitwirkung von autorisierten Personen
- Noisy Background
- Daten gestohlen (Fingerabdruck)

### 2.2.12 Biometrie

- Daten nicht geheim
- Können nicht neu generiert werden
- Können öfters benutzt werden → Tracing
- Private Daten (Krankheiten)

### 2.2.13 Angriffe

- Spoofing (ID verheimlichen oder andere annehmen)
- Angriffe auf Module
- Replay
- Daten verheimlichen
- Einfügen vieler Lämmer
- Templates sind Plaintext gespeichert

### 2.2.14 Gegenmaßnahmen

- Person lebendig?
- Puls
- Schweiß
- Irisreaktion auf Licht
- Bewegung

## 2.3 DRM

### 2.3.1 Anwendungsbereiche

Software, Audio, Video, E-Books, TV-Broadcasting

### 2.3.2 DRM Approaches

- Hardware (Dongle)
- Einzigartigkeiten auf dem PC, Hide Code, spezielle Formatierungen
- Eigenschaften des PCs

### 2.3.3 Heute

Lizenz-Server, Online-Registrierung, Whistleblower, Shareware, Support gegen Geld, Campus-Lizenzen

### 2.3.4 Broadcast

- Signal wird verschlüsselt gesendet
- Karte der Top-Set-Box hat den Key zur Entschlüsselung
- Dies personalisiert die Box
- Empfängt Nachrichten

### 2.3.5 CSS

- Player hat einen Key
- Er entschlüsselt den Disk-Key auf der Disk (408 vorhanden), einen für jeden Hersteller
- Player entschlüsselt die Title-Keys mit dem Disk-Key

### 2.3.6 deCSS

- 40 Bit Schlüssel
- Komplexität auf  $2^{25}$  reduzierbar
- $E_{kd}(\text{hash}(kd))$  ist auf der Disk gespeichert

### 2.3.7 AACS

- HDDVD und BluRay
- Ablauf:
  1. Auf dem Gerät gibt es mehrere geheime Device-Keys, mit deren Hilfe kann der Media-Key aus dem Media-KeyBlock generiert werden
  2. Mit der VolumeID der Disk und dem MediaKeyBlock kann auf den UniqueVolKey zugegriffen werden (AES-128)
  3. Mit dem UniqueVolKey kann der EncTitleKey entschlüsselt werden
  4. Mit diesem kann wiederum der verschlüsselte Inhalt der Disk entschlüsselt werden
- Zur Revocation von Keys werden diese mit einem Key eines höheren Baum-Levels verschlüsselt (oder Teilbäume werden verschlüsselt)

### 2.3.8 Angriffe auf AACS

- Device-Keys können durch eventuelle Schwachstellen (vor allem bei Software-Playern) ausgelesen und veröffentlicht werden
- Diese Keys können jedoch revoked werden

### 2.3.9 AACS bei Aufnahmen

- TitleKeys werden generiert, mit denen wird der Content verschlüsselt
- Die TitleKeys werden dann mit der DiskID und dem DiscKey (aus dem read-only MKB der Disc und einem RecorderKey generiert) verschlüsselt

### 2.3.10 Fairplay

- Audiostream wird mit MasterKey verschlüsselt
- Der MasterKey wird mit dem UserKey verschlüsselt in der Datei gespeichert
- Für jeden Track gibt es andere UserKeys, diese werden lokal und aufm Server gespeichert
- Über den Server kann man bis zu einer Grenze weitere Endgeräte hinzufügen oder diese wieder löschen

### 2.3.11 Watermarks

- Beweis, wer der Besitzer ist
- Wer hat das Copyright verletzt?
- Fingerprint: Sind zwei Kopien gleich?

### 2.3.12 Verstecken von Watermarks

- Least significant Bit → einfach zu erkennen
- kleine Änderungen am Film (Position des Untertitels, ...), an dem Musikstück (Echo, ...) oder am E-Book (Positionen des Texts, Zeichenabstand, Zeilenabstand, ...)

### 2.3.13 Probleme

- Reselling legal erworbener MP3s aus dem Internet
- Zurückgabe
- Gestohlene Player

## 2.4 Wahlen

### 2.4.1 Formen

- Web-Polls
- Elektronische Wahlmaschinen
- Online-Voting
- Vernetzte Wahlstationen
- Vernetzte Wahlmaschinen
- E-Counting

### 2.4.2 Fragen

- Wurde richtig gezählt?
- Wurde richtig gewertet?
- Kann das Ergebnis unabhängig nachvollzogen werden?
- Stimmverkauf möglich?
- Können Wähler zu einer bestimmten Stimme gezwungen werden?
- Ist die Wahl anonym?

### 2.4.3 Three Ballot

- Benutzung von so vielen Krypto-Eigenschaften wie möglich, ohne Kryptographie einzusetzen
- es gibt pro Wähler drei Wahlscheine
- Jeder Kandidat erhält eine Stimme, der zu wählende Kandidat erhält zwei Stimmen (Stimmen der drei Wahlzettel werden addiert)
- Wähler nimmt die Kopie eines beliebigen Wahlscheins mit
- Alle Wahlscheine werden nach der Wahl veröffentlicht
- Der Wähler überprüft, ob sein Wahlschein korrekt gewertet wurde (SN auf dem Schein), W'keit von  $\frac{1}{3}$ , dass ein Betrug auffällt
- Stimmen für einen Kandidaten = Stimmen auf den Wahlscheinen – Anzahl der Wähler
- Pro Wahl (alle drei Scheine zusammen) müssen  $p + 1$  Kreuze gemacht werden für  $p = \text{Anzahl der Kandidaten}$

### 2.4.4 Bingo Voting

- $n$  Wähler und  $p$  Kandidaten
- Für jeden Kandidaten werden im Vorfeld  $n$  Zufallszahlen generiert
- Für jede Zufallszahl eines Kandidaten wird ein Commitment  $C_N^P$  generiert (Lockable Box), diese werden veröffentlicht (mit Beweis, dass alles korrekt ist)
- Wahl: Wähler generiert eine Zufallszahl für den zu wählenden Kandidaten, für alle anderen Kandidaten werden Zufallszahlen aus der vorher generierten Liste benutzt
- Unbenutzte Dummy-Votes = Anzahl der Stimmen für den jeweiligen Kandidaten

- Veröffentliche dann die geöffneten ungenutzten Dummy-Votes (Commitments), die Wahlscheine, die Beweise der korrekten Rechnung und natürlich das Wahlergebnis
- Als Beweis der korrekten Erzeugung der Commitments kann dienen, dass im Vorfeld  $r$  Zufallszahlen und deren Commitments zusätzlich pro Kandidat gebildet wurden. Danach werden zufällig  $r$  der Commitments geöffnet und überprüft, ob sie korrekt zugeordnet wurden.
- Nach der Wahl muss noch bewiesen werden, dass pro Wahlschein nur eine Stimme enthält. Dies kann mit Pedersen-Commitments geschehen. Hierfür werden die Dummy-Votes eines Wahlscheins zweimal maskiert und entweder die Transition zwischen der Liste der Commitments und der Liste der ersten Maskierung wird offengelegt oder zwischen der Liste der ersten Maskierung und der der zweiten.
- Das System ist universell verifizierbar, das Problem wurde aber von der Maschine zum Protokoll verschoben (schwer verständlich?)

#### 2.4.5 Pedersen-Commitment

- Wähle zwei Generatoren  $g$  und  $h$
- Für  $m =$  Nachricht rechne  $c = g^m \cdot h^r$  mit  $r$  als Zufallszahl
- Maskierung durch  $c' = c \cdot h^s$

#### 2.4.6 Scantegrity

- Es gibt  $n$  ( $n$  Wähler) mit Seriennummer (als Dezimalzahl und als Barcode)
- Des Weiteren gibt es  $p$  Kandidaten, die ersten  $p$  Buchstaben des Alphabets werden auf jedem Wahlzettel zufällig den Kandidaten zugeordnet
- Der Wähler wählt nun einen Kandidaten schreibt sich den Buchstaben auf und reißt den SN-Zettel ab (trotzdem noch Barcode auf dem Schein vorhanden)
- Nach der Wahl werden die Seriennummern zusammen mit dem gewählten Buchstaben veröffentlicht
- Mit dem Switching-Board (wird im Vorfeld erzeugt) werden die Stimmen nun ausgezählt: Sortieren der Stimmen nach Kandidaten, aber geschuffelt. Als Beweis, dass das Board korrekt arbeitet, werden wieder mehr Stimmzettel erzeugt. Die Anzahl der überzähligen Zettel wird zufällig geöffnet und für die Wahl dann natürlich nicht benutzt
- Zum Beweis, dass korrekt gezählt wurde, wird ein zweites Switching-Board benutzt. Die Zwischenschritte werden veröffentlicht und für jede Stimme wird entweder die Transition von ersten zum zweiten oder vom zweiten zum dritten gezeigt.
- Bei einem Problemen wird der Wahlzettel in einen Umschlag gepackt und die SN wird verglichen. Dann kommt er in einen anderen Umschlag, so dass nur die Wahl sichtbar ist. Nun kommt der Umschlag mit anderen (selber Buchstabe, andere Kandidaten) in eine Box. Der Wähler sieht so den Buchstaben seiner Wahl aber nicht die Wahl selber.

### 2.5 Physical Protection

#### 2.5.1 Ziel

Jemanden an etwas hindern, einen Angriff erkennen, Alarm geben oder auf einen Angriff reagieren

#### 2.5.2 Models

**Derek** unskilled (Drogenbeschaffungskriminalität)

**Charlie** skilled (Kleinkrimineller)

**Bruno** skilled und intelligent (Gentleman-Dieb)

**Albert** geht in die Terrorist-Ecke

### 2.5.3 Arten von Master-Keys

**TPP** Total Position Progression Scheme, keine geteilten PINs

**RC** Rotating Constant Scheme, der Master-Key teilt sich immer eine bestimmte Anzahl an PINs mit dem normalen Key

### 2.5.4 Revocation

- Probleme, wenn ein Angestellter die Firma verlässt oder er einen Zweitschlüssel hat
- Revocation ist teuer → Schlösser mit Karte und PIN

### 2.5.5 Design

- Alle Bereiche abdecken
- Was soll abgesichert werden?
- Was möchte ich erreichen?
- Für welchen Typ Eindringling soll es ausgelegt werden?

### 2.5.6 Probleme

- Umgehen der Sensoren möglich?
- Fehlalarme
- Schwer, den korrekten unter vielen Fehlalarmen herauszufinden

### 2.5.7 Verfahren gegen Angriffe auf die Kommunikation

- Multiple Kabel
- Challenge-Response mit Kryptographie

## 2.6 Access Control

### 2.6.1 Level

- Application
- Middleware (zB Datenbank)
- Betriebssystem
- Hardware

### 2.6.2 OS-Access-Control

- Access-Control-Matrix mit Objekten (Dateien, Prozesse) in den Spalten und Subjekten (Prozesse, User) in den Zeilen, die Einträge sind dann die Rechte
- ACL: Rechte der Subjekte werden bezüglich eines Objekts bei diesem gespeichert, Gruppierungen als Abkürzung

### 2.6.3 Modify ACL

- Möglichkeit 1: Creator hat Rechte und kann diese ändern
- Möglichkeit 2: ein bestimmter User darf die ACL eines Objekts ändern

## 2.6.4 Unix

Hier bekommt der Owner, die Gruppe und jeder andere die Rechte bezüglich eines Objekts (read, write, execute)

## 2.6.5 UserIDs

- RealUserID
- EffectiveUserID (setuid)
- SavedUserID (zur Wiederherstellung der ID nach User-Wechsel)

## 2.6.6 Capability-List

- Rechte des Users bezüglich aller Objekte wird beim User gespeichert
- Im Gegensatz zu ACL schneller bei der Abfrage, auf welche Objekte ein User zugreifen darf
- Aber langsamer bei der Abfrage, welche User auf ein bestimmtes Objekt zugreifen dürfen

## 2.7 Firewalls

### 2.7.1 Components

- Stateful oder stateless Paketfilterung
- Proxy Gateways

### 2.7.2 Paketfilter

- Anfrage-Host + Port zu Ziel-Host + Port
- Erlauben bzw Verweigern von Zugriffen auf/von bestimmte Hosts/Ports

### 2.7.3 Fragmentation-Attack

- Senden von zwei ACK-Paketen, die dann auf dem Server wieder zusammengesetzt werden (ACKs gehen durch die Firewall)
- Hinter der Firewall ergeben sie ein SYN-Paket, das den Server "zwingt", ein ACK als Bestätigung der Verbindung zu senden

### 2.7.4 Stateless-Filter

- Alle Ports größer als 1023 werden für eingehende Verbindungen geöffnet
- Ein Paket auf einem dieser Ports könnte eine Antwort auf eine echte Verbindung sein
- Kann auch Mist sein
- Keep track of states

### 2.7.5 Stateful-Filter

Entscheidungen pro Paket aber im Kontext einer bestehenden Verbindung

### 2.7.6 Circuit-Level-Gateway

- Überprüft die Verbindung anhand der erlaubten Verbindungen
- Versteckt Host+Port vor der Öffentlichkeit
- Prüft bei bestehenden Verbindungen nicht die Pakete

### 2.7.7 Application-Level Gateway

- Audit und log alle Aktivitäten
- Kann Auth unterstützen
- Pro Application einen Proxy

### 2.7.8 Bastion Host

- Gehärtete Firewall mit Application-Level-Gateway hinter einem Paket-Filter
- Paket-Filter als Schwachstelle, vom Filter durchgelassene Pakete sollten erst durch den Bastion Host müssen um ins sichere Netz zu gelangen

### 2.7.9 Probleme

- Löst nicht die echten Probleme
  - Buffer Overflow
  - WEP

## 2.8 Malware

### 2.8.1 Arten

- Trojaner
- Viren
- Würmer

### 2.8.2 Trojaner

- Gibt vor, etwas zu machen und macht etwas anderes heimlich
- Repliziert sich nicht
- Beispiel: Keylogger, Shells, Proxies, DDoS, Spam

### 2.8.3 Rootkit

- Stealthness
- Wird meist nachgeladen
- Verstecktes Verzeichnis mit gehackten Binaries (ps, ln, du, ...), die eine IRC-Backdoor zu installieren (selber Hash wie Originale)
- Gegenmaßnahme: Installiere sauberes ps und such nach auffälligen Prozessen

### 2.8.4 Viren

- Insertion Phase und Execution Phase
- Payloads:
  - beschädige Files
  - lösche Files
  - stehle Files
  - ändere das BIOS

### 2.8.5 Typen

- Boot Sector Infectors
- Executables Infectors
- Multiparty-Viren
- TSR-Viren
- Stealth-Viren
- Encrypted-Viren
- Polymorphe Viren
- Makroviren

### 2.8.6 Conficker

- Trat im November 2008 auf
- Varianten von A bis E
- Wurm lässt HTTP-Server laufen und lädt dll bei Bedarf nach
- B und C verteilen sich auf Netzwerklaufrer und Wechseldatenträger
- Payload um Code nachzuladen bzw im zusätzlichen Code zu laden
- A: 250 Domains werden generiert und aufgerufen (5 TLDs)
- B: anderer Seed + drei weitere TLDs
- C: 50000 Domains werden generiert (110 TLDs) und davon werden 500 zufällig gewählt

### 2.8.7 Proof-Carrying Code

- User setzt Sicherheitspezifikationen
- Programmierer garantiert diese
- User validiert den Beweis

## 2.9 Buffer Overflow

### 2.9.1 Stack

- Wächst nach unten
- Kontrolliert durch den Compiler um lokale Funktionsvariablen und Kontext während eines Funktionsaufrufs zu speichern (zB auch EIP)
- LIFO, dynamisch

### 2.9.2 Heap

- Wächst nach oben
- Kontrolliert durch den Programmierer um Variablen zu speichern
- Dynamisch, allocated, deallocated

### 2.9.3 BSS

- Uninitialisierte globale und statische Variablen
- Werden mit 0 initialisiert (zB Counter-Variablen)
- Writable

### 2.9.4 Data

- Initialisierte globale und statische Variablen
- Read-only und read-write
- Writeable

### 2.9.5 Text

- Programmcode
- Read-only

### 2.9.6 Stack 2

- Informationen werden auf dem Stack gespeichert
- In sogenannten *Stack-Frames* (FILO)

### 2.9.7 Pointer

**ESP** Zeigt auf die Adresse des letzten Elements auf dem Stack. Jeder Stack-Frame enthält

- Parameter der Funktion
- Lokale Variablen
- SFP: Restore EBP
- RET: Restore EIP

**EBP** Zeigt auf das aktuelle Stack-Frame

**EIP** Zeigt auf die aktuelle Instruktion

### 2.9.8 Angriffe

- Biege einen jmp auf infizierten Code um
- Konfigurationen (Dateinamen, ...)
- User-Data
- Decision-Making-Values
- Return-to-libc: system, exec

### 2.9.9 Heap 2

- Ändern von Variablen
- Eventuell Schreiben in Dateien mit den Rechten des Programms

### 2.9.10 Vulnerable

strcpy, strcat, gets, scanf, printf

### 2.9.11 FormatStrings

- printf(buf) anstatt printf("%s", buf)
- % kann als Format-Symbol erkannt werden. Bei %n wird die Anzahl der Zeichen von buf geschrieben.
- Mit Argument wird die Anzahl in das Argument gespeichert, sonst wird es dahin geschrieben, wo der interne Stack-Pointer von printf hinzeigt
- %Mx schreibt M Bytes

### 2.9.12 Preventing

- Java nutzen
- Stack als non-executable markieren
- Randomisieren der Stack-Locations
- Verschlüsselte (XOR) Return-Adressen
- Statische Analyse
- Run-Time-Checking (Buffer, Arrays)
- Penetration-Test mit langen Strings

### 2.9.13 Statische Analyse

- Keine Analyse ist sowohl zuverlässig als auch komplett
- Allozierter Platz *ge* aktuelle Länge
- Viele false positives, aber auch neue Fehler bei bekannten Programmen gefunden

### 2.9.14 StackGuard

- Canaries vor den Pointern (random String)
- \0-terminiert
- Canary-Check vor jedem Jump → weniger performant

### 2.9.15 PointGuard

- Verschlüsselt alle Pointer im Speicher zur Laufzeit
- Generiert einen zufälligen Schlüssel und XOR-t die Pointer damit
- Nach dem Überschreiben zeigt der Pointer durch die Verschlüsselung auf eine zufällige Position

## 2.10 Intrusion Detection

### 2.10.1 Zu erkennen

- Versuchte und erfolgreiche Einbrüche
- Angriffe von legitimen Usern
- Malware
- DoS-Angriffe
- Bekannte und bestenfalls unbekannte Angriffe

### 2.10.2 Dennings Hypothese

- System nicht angegriffen:
  - Prozesse und User handeln mehr oder weniger vorhersagbar
  - Sie führen keine Kommandos aus, die gegen die Policy verstoßen
  - Aktionen von Prozessen gehorchen einer Menge an Spezifikationen
- Hypothese: angegriffene Systeme versagen bei mindestens einer der Regeln

### 2.10.3 Klassifikation IDS

- Anomaly Detection → bekannt ist gut, unbekannt ist schlecht
- Misuse Detection → schlecht ist bekannt, nicht schlecht ist gut
- Specification-based Detection → gut ist bekannt, nicht gut ist schlecht
- Alle genannten: statisch oder adaptiv (dynamisch)

### 2.10.4 Anomaly Detection

- Charakterisiert irgendwie das System und das Verhalten des Users
- Alarm, wenn aktuelle Werte abweichen:
  - Threshold
  - Statistik (Durchschnitt und Standardabweichung)
  - Markov

### 2.10.5 Markov-Modell

- System wird nach einer Aktion in einen neuen Zustand versetzt
- Mit der Zeit können über die Zustände Wahrscheinlichkeiten gebildet werden
- Anomalie, wenn ein unwahrscheinliches Ereignis eintritt
- Effektivität basiert auf den zugrunde liegenden Daten

### 2.10.6 Misuse-Matching

Kann nur bekannte Angriffe verhindern

### 2.10.7 IDS-Architektur

- *Agents* loggen Daten
- *Director* analysiert die Daten bezüglich seiner Regeln
- *Notifier* erhält das Ergebnis vom Director und handelt danach:
  - Rekonfigurieren der Agents/Director
  - Jemanden benachrichtigen
  - Aktivierung von Response-Mechanismen

### 2.10.8 Agent

- Host- oder Netzwerk-basiert
- Logs haben bei Host-based nur lokale Sicht, es ist also ein IDS pro System nötig
- Netzwerktraffic bei Netzwerk-basierten, Überwachung vieler Hosts, Verschlüsselung von Nachteil

### **2.10.9 Angreifen und Umgehen von NIDS**

- Überladen mit großen Datenströmen
- Verschlüsselung
- Splitten der böartigen Pakete
- TCP-Attacks: Insertion-Attack mit falscher Checksumme, TTL-Attacke mit kurzer TTL eines Pakets (kommt nicht bis zum IDS)

### **2.10.10 Netzwerkangelegenheiten**

- Netzwerkarchitektur diktiert die Positionierung der Agenten
- Fokus normalerweise auf Angriffe von außen

### **2.10.11 Incident-Prevention**

- Identifizierung des Angriffs vor dessen Beendigung
- Verhinderung, dass der Angreifer seinen Angriff abschließt
- Angreifer abschotten, ohne dass er es merkt

### **2.10.12 Intrusion Detection and Isolation Protocol**

- Boundary Controller als Grenzsysteme zum Abschotten eines Bereichs
- IDIP-Domain: Systeme, die miteinander kommunizieren können, ohne den Boundary-Controller zu passieren

### **2.10.13 Intrusion Handling**

- Bereitmachen für einen Angriff
- Identifizierung eines Angriffs
- Eindämmung des Angriffs (aktiv/passiv)
- Ausrottung des Angriffs
- Erholung vom Angriff
- Nachbearbeitung

### **2.10.14 Sammlung von Daten**

- Verschiedene Agenten haben unterschiedliche Sicht auf Events und produzieren so verschiedene Informationen
- Weder Host- noch Network-basierte Agenten reichen alleine aus

## **2.11 Anonymity Networks**

### **2.11.1 Begriffe**

- Anonymität
- Nicht verfolgbar
- Nicht verknüpfbar
- Nicht beobachtbar
- Verwendung von Pseudonymen

### 2.11.2 Anonyme Kommunikation

- Keine Sender-Empfänger-Beziehung feststellbar
- Unbekannte Identität von Sender/Empfänger
- Nicht messbare Menge und Art von Traffic zwischen zwei Peers

### 2.11.3 Anwendungen

- Privatsphäre beim emailen/surfen
- DigiCash zwischen Bank und Händler
- Anonymous e-voting
- Veröffentlichen ohne Zensur

### 2.11.4 Angriffe

- Passive Traffic-Analyse
- Aktive Traffic-Analyse
- Router kompromittieren

### 2.11.5 Broadcast

- An alle Senden, nur der gewollte Empfänger kann das Paket entschlüsseln
- viel Overhead

### 2.11.6 Mixes

- Ein Mix kann die Sender-Empfänger-Beziehung nachvollziehen → benutze mehrere
- Sender baut den Rückweg auf (Zuweisung von Public-Keys, Empfänger und symmetrischer Key für den Empfänger) und packt die Nachricht dazu
- Sender baut den Hinweg drumherum  $\text{Paket}_n = (\text{Paket}_{n-1}, \text{Empfänger})_{\text{PubKey}_n}$
- Der Empfänger packt aus, verschlüsselt seine Antwort mit dem symmetrischen Key und nutzt den Rückweg
- Die Mixer auf dem Rückweg verschlüsseln die Antwort mit ihrem zugewiesenen Public-Key
- Mixer senden nur, wenn das jeweils festgelegte der vier bekannten Ereignisse eintritt

### 2.11.7 Onion Routing

- Application-Layer, nicht Network-Layer
- Jeder Knoten auf einem Pfad kennt nur den Vorgänger und den Nachfolger
- Schicke pro Hop (den nächsten Empfänger, symmetrischer Key) $_{\text{PubKey}_E}$  und die mit dem symmetrischen Schlüssel verschlüsselte Nachricht an E
- Rückweg: die Router bauen eine Zwiebel um die Nachricht

### 2.11.8 Dining Cryptographers

- Selbst mit unendlicher Rechenpower nicht berechenbar
- Gruppe mit  $n$  Personen:  $n$  Zufallsbits um ein Bit zu senden
- Von jedem wird ein Zufallsbit generiert und an seinen linken Nachbarn geschickt
- Jeder (bis auf den Sender) veröffentlicht, ob seine zwei bekannten Bits gleich sind oder nicht (XOR)
- Der Sender veröffentlicht seine bekannten Bits XOR Nachricht
- XOR von allen Veröffentlichungen ergibt die Nachricht

### 2.11.9 TOR

- Zusätzliche Directory-Server: vertrauenswürdige Knoten
- Zusätzlicher Integritätscheck (gegenüber normalen Onion Routing)
- Jeder Router hat eine Verbindung zu jedem anderen Router
- Jeder Router hat ein longterm-PK-Paar zum Signieren und ein shortterm-PK-Paar für TLS-Handshakes
- Abfrage der Router über Directory-Server: wähle drei aus, tausche mit dem ersten einen symmetrischen Schlüssel aus und baue darüber eine sichere Verbindung zu ihm auf
- Der Client baut dann eine symmetrische Session zum zweiten auf (über den ersten getunnelt) und dann über den ersten und den zweiten zum dritten (exit-Node)
- Der Client verschlüsselt die Nachricht mit allen symmetrischen Schlüsseln, der Exit-Node sendet unverschlüsselt
- Viele Verbindungen und Apps über eine TOR-Kette möglich
- Es sind keine root-Privilegien für TOR-Router nötig

### 2.11.10 Location-Hidden Services

- Server, zu dem jeder eine Verbindung aufbauen kann, aber keiner seine Position kennt oder seinen Besitzer
- Gegen Zensur und Angriffe
- Wähle Introduction-Points und veröffentliche die auf den Directory-Servern
- Der Client wählt Rendezvous-Points und teilt diese dem Server über die Introduction-Points mit
- Alles läuft über TOR

## 2.12 Design Principles

### 2.12.1 Privilegien

- Ein Subjekt sollte nur die Privilegien bekommen, die es für die Ausführung seiner Aufgabe benötigt
- Viele Systeme vergeben ihre Rechte aber nicht granular genug
- Rechte sollten direkt nach der Beendigung der Aufgabe wieder entzogen werden

### 2.12.2 Economy

- Keep it simple stupid (KISS)
- Einfach heißt, dass wenig schief geht
- Geht doch was schief, ist es leicht wieder zu beheben

### 2.12.3 Complete Mediation

- Zugriffe auf alle Objekte sollten überprüft werden
- Problem bei anfänglicher Überprüfung der Rechte und nachträglicher Änderung der Rechte

### 2.12.4 Open Design

- Sicherheit sollte nicht auf der Geheimhaltung des Codes basieren
- Problem bei Firmen

### 2.12.5 Sonstiges

- Keine geteilten Kanäle (abhörbar)
- Wenn Sicherheitsvorkehrungen akzeptiert werden sollen, so muss die Bedienung nicht komplizierter sein als ohne Sicherheitsvorkehrungen
- Anpassungen beim Design, Analyse, Implementation

## 2.13 Evaluating Systems

### 2.13.1 Ziel

- Zeige, dass ein System bestimmte Bedingungen erfüllt, basiert auf Beweisen
- Formale Evaluierungsmethoden

### 2.13.2 Warum

- Unabhängige Evaluierung
- Andere Sicht auf die Dinge
- Aber: teuer

### 2.13.3 TCSEC

- Sechs verschiedene Evaluierungsklassen: C1, C2, B1, B2, B3, A1
- Sieben Trust-Level: C1-A1 + D

## 3 Data Communication and Internet Technology

### 3.1 Introduction

#### 3.1.1 Protokolle

- Datenkommunikation basiert auf Protokollen
- Protokolle zur Festlegung des Datenformats, der Kontrolle von Zugriffen, von Prioritäten, Routing, ...
- Zwei mögliche Implementationen: Protokoll für ein Programm festlegen oder umgekehrt. Zweite Möglichkeit wird häufig genutzt.

### 3.1.2 ISO-OSI

- Application (Standard-Interfaces)
- Presentation
- Session
- Transport (netzwerkunabhängig, End-to-End-Transfer)
- Network (Addressing und Routing)
- Data-Link (Flow-Control und Frame-Protection)
- Physical (Signalrepräsentation)

### 3.1.3 Layer 1

- Wird eine 1 als 1 erkannt?
- Wie lange muss eine Voltzahl anliegen?
- Welche Kabelarten?
- Welche Pin-Belegung?
- Datenrate

### 3.1.4 Layer 2

- Sichert die fehlerfreie Übertragung zu
- Daten werden in Frames segmentiert
- Empfänger überprüft ein Frame auf Korrektheit
- Flow-Control für Re-Transmission

### 3.1.5 Layer 3

- Verantwortlich für die Datenübertragung über große Distanzen zwischen heterogenen Subnetzwerken
- Uniform addressing of hosts
- Wählen des Wegs durchs Netz

### 3.1.6 Layer 4

- Managed die End-to-End-Kommunikation zwischen zwei Prozessen
- Garantiert die Vollständigkeit und die korrekte Reihenfolge der Pakete
- Addressing des einzelnen Kommunikationsprozesses

### 3.1.7 Layer 5

- Siehe Ende des Kapitels

### 3.1.8 Layer 6

- Siehe Ende des Kapitels

### 3.1.9 Layer 7

- Es werden standardisierte Interfaces angeboten (für oft genutzte Services)

### 3.1.10 Layers

- Layer  $n - 1$  setzt vor das Paket von Layer  $n$  noch einen Header (Layer 1 fügt noch einen Trailer hinzu)
- Unten liegende Layer bieten den darüber liegenden Funktionalitäten an
- PDU: Protocol Data Unit
- PDUs werden von Layer  $n$  zu Layer  $n - 1$  nicht zwingend 1:1 übersetzt, sie können auch gesplittet werden

### 3.1.11 Geräte und Infrastrukturen

- Hub/Repeater: kein gleichzeitiges Senden und Empfangen, alle teilen sich einen Broadcast-Kanal, jeder kann alles lesen
- Switch: das Gerät kennt die MAC der angeschlossenen Geräte und leitet die Pakete korrekt weiter, eventuell (Layer 4) auch zum Load-Balancing
- Router: können zusätzlich noch Daten über andere Router oder Switches ans Ziel leiten
- Backbone: Menge an Computern (meist Router), die Point-to-Point (direkt über ein Kabel verbunden) über große Distanzen verbunden sind

### 3.1.12 Broadcast-Netzwerk

- Hub: einer sendet, alle empfangen, nur der gewollte Empfänger löscht das Paket nicht
- Möglich durch viele Unicast-Pakete oder durch eine Broadcast-Adresse

### 3.1.13 Klassifikationen

PAN, LAN, MAN, WAN, Internet

### 3.1.14 Standards

- 802.3: Ethernet
- 802.11: Wlan
- 802.15: Bluetooth
- 802.16: WiMax
- 802.17: Resilient Packet Ring

### 3.1.15 TCP/IP

**Application-Layer** definiert gebräuchliche Kommunikationsservices (Telnet, ftp, SMTP, DNS, http, ...), beinhaltet bei Bedarf auch die ISO-OSI-Layer 5 und 6

**Transport-Layer** TCP und UDP

**Internet-Layer** IP-Protokoll

**Host-to-Network** Data-Link- und Physical-Layer

### **3.1.16 Cross-Layer-Protocol**

- Überspringen von Layern möglich
- Zum Beispiel beim Wlan ist es sinnvoll, die Layer 1 und 2 zu kontrollieren

## **3.2 Physical Layer**

### **3.2.1 LAN**

- Bis zu einem GBit
- 10m bis mehrere Kilometer
- Delay von ca 10ms
- Topologien: Bus, Stern, Ring, Tree, Mesh

### **3.2.2 MAN**

- Größere Distanzen
- Nur 1-2 Kabel
- Hauptunterschied: Timeslots

### **3.2.3 WAN**

- Verbindet LANs und MANs über größere Distanzen
- Die Topologie basiert auf den jeweiligen Anforderungen

### **3.2.4 Parameter**

- Physisches Transportmedium
- Pinbelegung
- Repräsentation der Roh-Bits
- Datenrate

### **3.2.5 Übertragungsmedien**

- Twisted-Pair
- Coaxial
- Glasfaser
- Satellit
- Funkverbindungen

### 3.2.6 Twisted-Pair

- Cat3: normal mit Isolierung
- Cat5: Teflon-Isolierung
- Cat6,7: Silberfolie pro Paar
- UTP: (unshielded) keine weitere Abschirmung
- STP: (shielded) jedes Paar wird einzeln abgeschirmt
- Analoge Signale
- Fehlerrate: ca.  $10^{-5}$

### 3.2.7 Koaxialkabel

- Höhere Datenraten über größere Distanzen: 1-2 GBit
- Bessere Abschirmung
- Fehlerrate: ca.  $10^{-9}$

### 3.2.8 Glasfaser

- Fast unendliche Datenraten möglich (bis zu 50000 GBit)
- Unempfindlich gegenüber elektromagnetischen Störungen
- Fehlerrate: ca.  $10^{-12}$
- Signal bleibt im Kern und wird von zweiten Medium reflektiert
- Probleme: Absorption und Dispersion
- LEDs und Laserdioden als Quelle, LEDs haben eine größere Wellenlänge (kleine Entfernung) und Laser haben eine kleine Wellenlänge (große Entfernung)

### 3.2.9 Übertragungsarten

- Baseband: digital
- Broadband: analog, wird auf ein Trägersignal moduliert

### 3.2.10 Cable-Codes

- +5V/-5V
- Synchronisierung durch Wechsel der Voltzahlen
- Vermeidung von Dauerstrom

### 3.2.11 NRZ

- Non Return to Zero
- 1: +1V
- 0: -1V
- Schlechte Clock-Sync und eventuell Dauerstrom

### 3.2.12 Differential NRZ

- 1: Level-Change
- 0: kein Spannungswechsel
- Ähnlich wie NRZ

### 3.2.13 Manchester-Code

- Zwei mögliche Spannungswechsel pro Clock-Pulse
- Der Wert wird in der Mitte übertragen
- 1: +1V zu -1V
- 0: -1V zu +1V
- Nur halbe Kapazität (1B/2B)
- Clock-Sync bei jedem Pulse

### 3.2.14 Differential MC

- 0: Wechsel zwischen den Pulsen
- 1: kein Wechsel zwischen den Pulsen
- In der Mitte eines Pulses wird immer gewechselt

### 3.2.15 4B/5B

- 4 Bit werden durch 5 Bit repräsentiert
- Vermeidung von vielen 1en und 0en

## 3.3 Data Link Layer

### 3.3.1 Grundlagen

- Unterteilung in zwei Teile: *Logical Link Control* (LLC) und *Medium Access Control* (MAC)
- LLC: Unterteilung in Frames, garantieren eine fehlerfreie Übertragung
- MAC: Zugangskontrolle zum Kommunikationskanal in Broadcast-Netzwerken

### 3.3.2 Frame Construction

- Header, Data, Trailer
- Header für Adresse, Framenummer, ...
- Trailer für Error-Check, Frame Checking Sequence (FCS)
- Adressen sind hier MAC-Adressen

### 3.3.3 Error Detection

- Einfachste Art: parity-Bit (gerade oder ungerade Anzahl der 1en, eine gerade Anzahl an Fehlern werden nicht entdeckt)
- Double Parity:  $4 \times 4$ -Matrix mit Parities für jede Zeile und jede Spalte

### 3.3.4 Hamming-Code

- Parity-Bit an der 1., 2., 4. und 8. Stelle
- Jede andere Stelle hat ihre Parity-Bits da stehen, wo ihre Binärrepräsentation eine 1 hat
- Je nachdem, welche Parity-Bits beim Empfänger falsch sind, kann das falsche Bit bestimmt werden
- Nur ein 1-Bit-Fehler kann korrigiert werden

### 3.3.5 CRC

- PDU (Bit-String) als Polynom sehen und dann mod 2 mit XOR durch ein gegebenes Polynom teilen
- Fülle den Divident mit 0en auf und dividiere, bis ein Rest bleibt
- Übertrage den originalen Divisor konkateniert mit dem Rest
- Der Empfänger kann das Polynom nun durch den gegebenen Dividenten teilen und sollte keinen Rest bekommen
- Nur Fehlerfinden, keine Korrektur (oder selten) möglich

### 3.3.6 Error-Protection

- FEC (Forward Error Corrective) korrigiert Fehler so gut es geht und wirft falsche Pakete einfach weg, für Video und Tonübertragung
- ARQ (Automatic Repeat Request) findet Fehler, kann diese aber nicht korrigieren, fehlerhafte Pakete werden neu geordert

### 3.3.7 Flow-Control

- Sender sendet und wartet auf ein ACK bzw einen Timeout
- Sliding-Window: Übertragungsfenster einer bestimmten Größe  $w$  und mit einem Modulus  $> w$ . Der Sender kann bis zu  $w$  Pakete senden. Sobald ein ACK kommt, kann der Sender neue Pakete senden.
- Go-back-n:  $ACK_j$  und  $REJ_j$  sagen dem Sender, dass bis zu einer bestimmten Position alles ok war.  $REJ_j$  sagt, dass Paket  $j$  fehlerhaft war, der Sender sendet dann alles von  $j$  an nochmal.
- Selective-Repeat: Nur ACKs werden gesendet. Bei einem Fehler speichert der Empfänger alle folgenden korrekten Pakete und wartet darauf, dass der Sender das kaputte nach einem Timeout erneut sendet (vom fehlerhaften an). Kommt dann ein ACK für die gespeicherten, so setzt der Sender beim letzten korrekt empfangenen Paket fort.
- Selective-Reject: Wie eben (speichern usw.), für ein fehlerhaftes Paket wird jedoch ein  $REJ_j$  geschickt, dass dann durch den Sender einzeln wiederholt wird.

### 3.3.8 SDLC

- Für Ring-Netzwerke mit einem Master und mehreren Slaves
- Zwei Phasen: Master sendet an alle Slaves, Slave kann seine Daten an den Master senden
- Phase 1: Master sendet. Jeder Slave prüft, ob er der Empfänger ist. Alle leiten weiter und der Master nimmt das Frame aus dem Netz.
- Phase 2: Master sendet Sequenz von 7 1en. Will ein Slave senden, ändert er die siebte 1 in eine 0, sendet, und schickt dann sieben 1en hinterher. Der Master erkennt an empfangenen sieben 1en, dass alle gesendet haben.

### 3.3.9 Bitstuffing

- Bei SDLC werden die Frames von 01111110 getrennt
- Diese Sequenz darf während der normalen Datenübertragung nicht vorkommen
- Der Sender fügt also nach fünf 1en eine 0 ein

### 3.3.10 HDLC

- Setzt SDLC für beliebige Netze um
- Frame mit Header, Data, Trailer
- CRC (FCS) im Trailer
- Adresse (und Sequenznummer + ACK-Nummer) im Header
- Verschiedene Frame-Typen im Control-Feld

### 3.3.11 PPP

- Point-to-Point-Protokoll
- Basiert auf bzw nutzt Struktur von HDLC
- Das Adressfeld beinhaltet nur 1en
- Unnummerierter Modus im Control

### 3.3.12 MAC

- TDMA (TimeDivisionMultipleAccess): der Benutzer bekommt für einen gewissen Zeitabschnitt die komplette Bandbreite (Baseband)
- FDMA (FrequencyDivisionMultipleAccess): Benutzer bekommen eine bestimmte Bandbreite zugewiesen (Broadband)

### 3.3.13 Reservation-Protocols

- Zwei Phasen: Reservierungsphase und Übertragungsphase
- Explizite oder implizite Reservierung
- Variante 1 (ohne Streitfälle): Jeder User bekommt einen Slot im Reservation-Frame zugewiesen und setzt diesen auf 1, wenn er senden will. Dann senden alle nacheinander.
- Variante 2 (mit Streitfällen): Der Reservierungsframe hat nur eine begrenzte Anzahl an Slots. Der User wählt nun einen aus und schreibt seine ID rein. User, die alleine in einem Slot sind, können senden.

### 3.3.14 Implicit Reservation

- Stations, die senden wollen, beobachten die Datenslots und markieren die, die nächste Runde sicher frei sind (aktuell mit Konflikt oder aktuell frei)
- In der nächsten Runde wählen die Stations dann einen Slot aus und senden, wenn kein anderer gleichzeitig auch auf diesem Slot senden möchte
- Im Konflikt-Fall muss die nächste Runde abgewartet werden

### 3.3.15 Token

- Im Ringnetzwerk
- Kein Master nötig
- Nur der Tokeninhaber darf senden

### 3.3.16 Aloha

- Dezentralisiert
- Sender senden auf der selben Frequenz
- Bei Kollisionen werden die Pakete zu einer zufälligen Zeit später wiederholt
- Doppelte Framelength ist gefährdet

### 3.3.17 Slotted Aloha

- Die Zeitachse wird in Slots unterteilt
- Weniger Kollisionen
- Problem der Synchronisierung

### 3.3.18 Dezentralisierte Protokolle

- Variante von Aloha
- Wenn keiner sendet, sende Signal + Data
- Kurze RTT nötig

## 3.4 Infrastructure

### 3.4.1 Hubs und Repeater

- Layer 1
- Aufnahme und Weiterleitung von Signalen
- Vergrößern des Netzwerkbereichs
- Kein gleichzeitiges Senden und Empfangen
- Broadcast-Kanal

### 3.4.2 Switch

- Point-to-Point, kein Broadcast
- Switch kennt die Adressen der angeschlossenen Geräte
- Gleichzeitiges Senden und Empfangen möglich
- Bis Layer 2,3 oder 4, L3 kann Routing übernehmen, L4 kann in den TCP-Header schauen und für Load-Balancing eingesetzt werden
- Für jeden Input-Port werden Buffer für den Output-Port bereitgestellt, nur ein Input-Port kann an den Ausgang verbunden werden

### 3.4.3 Switch Data Forwarding

- Jeder Switch hat eine Forwarding-Tabelle
- MAC, Port, Alter
- Ziel gefunden und es ist ungleich dem Sender: senden
- Ziel gefunden und es ist gleich dem Sender: verwerfen
- Ziel nicht gefunden: Broadcast
- Die ankommenden Pakete und deren Ziele füllen die Tabelle

### 3.4.4 Bridges

- Verbindung mehrerer LANs auf Layer 2
- Aufgabe ist das richtige Forwarding
  - Adaption verschiedener LAN-Typen
  - Nur notwendige Pakete werden weitergeleitet
  - Größere Netzwerke möglich
- Transparente Bridges leiten nur die Pakete weiter, die nicht für das Source-LAN bestimmt sind, sie besitzen Forwarding Tables (wie bei Switches)

### 3.4.5 Limitierung von Bridges

- Nur ein paar tausend Stations möglich
- Möglicherweise zu viele unnötige Broadcasts
- Bridges kommunizieren nicht mit Hosts, diese erhalten keine Infos

### 3.4.6 Spanning-Tree-Bridges

- Gegen Loops
- Master Bridge
- Bridge für jedes LAN
- Root-Ports auf jeder Bridge (bester Weg zum Root/Master Bridge)

### 3.4.7 Source Routing Bridge

- Quellen legen den Weg über die Bridges fest (müssen ihn also kennen)
- Wenn nicht, wird ein Route-Discovery-Frame gebroadcastet

### 3.4.8 Router

- Layer 3
- Eingehende Pakete werden auf dem besten Weg weitergeleitet
- Globaler Adressraum
- Keine Grenze bzgl. der Größe
- Lokale Administration des Netzwerks
- Broadcasts und ARPs werden nicht durchgelassen

### 3.4.9 LANs: Bus

- BNC mit Endwiderständen
- A schickt an B, andere Stationen ignorieren das Paket
- In der Größe beschränkt
- Ethernet

### 3.4.10 LANs: Star

- Fast-Ethernet
- Broadcast oder Point-to-Point
- Angreifbare zentrale Stelle

### 3.4.11 LANs: Tree

- Verbindung von Bussen oder Trees
- Router oder Repeater als Verbindungen

### 3.4.12 LANs: Ring

- Broadcast-Network
- Kette von Point-to-Point-Verbindungen
- Stations sind Repeater
- Variante: bidirektionaler Ring (zwei entgegengesetzte Ringe)

### 3.4.13 LANs: Meshed Network

- Point-to-Point-Verbindungen zwischen den Teilnehmern
- $\frac{N(N-1)}{2}$  Verbindungen bei  $N$  Knoten
- Hinzufügen neuer Knoten ist teuer
- Party Meshed: günstiger, aber Routing, Flow-Control und Blockierungskontrolle werden nötig

## 3.5 Ethernet

### 3.5.1 CSMA

- Carrier Sense Multiple Access (Collision Detection)
- Prinzip: Überprüfe, ob jemand auf dem Medium sendet, wenn nicht, sende
  - Vorteil: einfach, keine übergeordnete Koordination
  - Nachteil: kein gesicherter Zugang, möglicherweise langes Delay
- Problem, wenn Station A sendet und B denkt, das Medium sein frei, weil das Paket noch nicht bei ihm angekommen ist
- Sobald eine Kollision bemerkt wird, wird die Übertragung gestoppt
- Eine Station sollte die doppelte Zeit warten, die das Paket von einem Ende des Netzes zum anderen benötigt, um einen Konflikt zu bemerken
- Länge Ethernet: 2800m,  $25\mu\text{s}$  für eine Strecke
- Bei angenommenen 10 mBit und einer Paketgröße von 64 Byte muss ein Sender nur während des Sendens auf Konflikte achten

### 3.5.2 Nachrichtenformat eines Frames

- Präambel zum synchronisieren (zählt nicht zur Gesamtgröße des Pakets dazu)
- Start-Frame-Delimiter (SFD) markiert den Anfang des Frames (zählt auch nicht zur Gesamtgröße des Pakets dazu)
- Destination-Address (DA)
- Source-Address (SA)
- Länge von Data bzw Typ (L/T)
- Data: 0-1500 Byte
- Padding: 0-46 Byte
- FCS

### 3.5.3 Resolving Conflicts

- Non-Persistent (Aloha): warte eine zufällige Zeit nach einem Konflikt und sende erneut
- 1-Persistent: Sende nach einem Konflikt so schnell es geht noch einmal
- p-Persistent: bei freiem Kanal wird nur mit der Wahrscheinlichkeit  $p$  gesendet, nach einem Konflikt braucht eine Nachricht  $\frac{1}{p}$  Versuche

### 3.5.4 Binary Exponential Backoff

- Timeout:  $51,2\mu s$
- Nach der Item-Kollision wählt eine Station einen Wert aus  $[0, 2^i - 1]$  und nutzt diesen als Timeout, nachdem das Medium wieder frei ist
- Nach  $i = 10$  wird das Intervall nicht mehr vergrößert und bei  $i = 16$  wird die Übertragung abgebrochen
- Kleine Intervalle bei wenig Traffic und große Intervalle bei viel Traffic
- Evtl Benachteiligung einzelner Stations

### 3.5.5 Parameter

- Ethernet: 2800m, 10 mBit, 64 Byte minimale Framelength
- Fast Ethernet: 205m, 100 mBit, 64 Byte minimale Framelength
- Gigabit Ethernet: 200m, 1000 mBit, 520 Byte minimale Framelength

### 3.5.6 Benennung

- Kapazität
- Base- oder Broadband
- Maximale Segmentlänge oder Typ des Mediums

## 3.6 Ring Networks

### 3.6.1 Token Bus

- Nur der, der das Token hat, darf senden
- Baue in einem Bus-Netzwerk eine Art Ring durch Nachbarzuweisung anhand der Station-Adresse
- Token:  $T_{ID,NextID} := ID, NextID$
- Data:  $M_{ID} := ID, Data$
- Viel Overhead durch große Token-Nachrichten
- Problem bei neuen Knoten oder Knoten, die das Netzwerk verlassen

### 3.6.2 Token Ring

- Point-to-Point-Ring
- Garantierter Zugriff, ohne Kollisionen
- Fair, garantierte Antwortzeiten
- Mehrere Tokens möglich
- Differential Manchester-Code
- Stations fungieren als Repeater, beliebige Ausdehnung
- Ausgangslage: Daten werden alle weitergeleitet, Daten für die eigene Station werden kopiert
- Senden: der Ring wird beim Sender getrennt, eingehende Daten werden untersucht
- Token: 3 Byte
- Zwei Möglichkeiten: sendende Station wartet darauf, dass der gesendete Frame wieder ankommt und schickt dann das neue freie Token oder sie sendet das neue freie Token direkt nach dem Data-Frame
- Token als eine Art Header von Data-Frames oder als 3 Byte-Frame alleine
- Frame-Status (2 Bit, A+C) zeigt an, ob das Frame an der Zielstation angekommen ist und ob es verarbeitet wurde (Bits sind doppelt vorhanden)
- Token: StartDelimiter+AccessControl+EndDelimiter
- AccessControl: Monitor-Bit, Priority-Bits, Reservation-Bits (Priority für die nächste Runde)
- Monitor-Station, um den korrekten Ablauf zu überwachen, Station mit der höchsten ID wird MS, wenn diese ausfällt. Hält den Ring sauber und zusammen.

### 3.6.3 FDDI

- Token-Ring-LAN basierend auf Glasfaser
- 100 mBit, 200km, bis zu 1000 Stations
- Zwei Ringe mit entgegengesetzter Richtung
- Zweiter Ring als Backup, falls eine Station ausfällt
- Stations können an einen oder an beide Ringe angeschlossen werden
- Coding 4B/5B
- Lange Sync-Präambel, 0,005% Taktgleichheit vorausgesetzt

- Frames bis zu 4500 Byte
- Protokoll wie bei Token-Ring, nur mit mehreren Tokens
- Data-Frames ähnlich wie bei TR

### 3.7 Wide Area Networks

#### 3.7.1 MAN

- Ringförmiges Glasfasernetz
- 1-2 Kabel ohne Switching-Elemente
- Alle Computer werden an ein Broadcast-Medium angeschlossen
- Unterschied zu LAN: Clockpulse oder Puffertechniken

#### 3.7.2 Resilient Packet Ring (MAN)

- Wie FDDI (ähnlich, ein Ring für Datenübertragung, einer für die Kontrolle)
- Kein bestimmter Physical-Layer
- Einführung von Management-Funktionalitäten:
  - Funktion für Reaktion auf Breakdowns
  - Kurze Reaktionszeit auf Breakdowns
  - Reservierung von Datenraten
- Unterschiede zu FDDI:
  - Weiterzuleitende Pakete werden in die Transit-Queue geschrieben
  - Pakete für eine Station werden in den Receive-Buffer geschrieben und nicht weitergeleitet
  - Zu sendende Daten werden in den Transmit-Buffer geschrieben
- Flow-Control:
  - Fairness: Choke-Nachrichten über den zweiten Ring, wenn die Transit-Queue dauerhaft voll ist
  - Ausfallsicher: Nachrichten können auf den zweiten Ring geleitet werden

#### 3.7.3 WAN

- Große Distanzen
- Kein Broadband, nur Point-to-Point
- Möglichst hohe Datenraten sicher über lange Distanzen übertragen

#### 3.7.4 Transmission Technologies (WAN)

- Point-to-Point-Links am Beispiel von Telefonleitungen
- Circuit-Switching: Verbindung wird hergestellt, wenn sie benötigt wird, exklusive Ressourcenreservierung, Bsp ISDN
- Packet-Switching: Weiterentwicklung der beiden letztgenannten, eine physikalische Leitung wird mit vielen geteilt

### 3.7.5 ATM

- Telekommunikation: verbindungsorientiert, (Telefonie-) Performance-Garantie, kurze Verzögerung, gleichbleibende Verteilung der Ressourcen
- Datenkommunikation: verbindungslos, keine Performancegarantie, variables Delay
- ATM garantiert Kapazität, konstante Verzögerung und flexible und effiziente Übertragung
- ATM-Cells: mehrere virtuelle Verbindungen werden zu einem konstanten Stream zusammengefasst, Cell-Größe: 48+5 Byte

### 3.7.6 ATM-Netzwerk

- ATM-Switch: schickt die Cells durch das Netzwerk. Die Cell-Header von eingehenden Cells werden ausgelesen und die Daten werden an ihr Ziel geleitet.
- ATM-Endpoint: Beinhaltet ein ATM-Network-Interface-Adapter, mit dem verschiedene Netze mit dem ATM-Netz verbunden werden können

### 3.7.7 ATM-Verbindungserstellung

- Der Sender erfragt beim Switch eine Verbindung mit einer bestimmten Qualität zu einer ATM-Adresse
- Der Switch wählt die Route und erstellt eine virtuelle Verbindung zum Ziel (es bekommt dann eine virtuelle Verbindungs-ID), an das er die Anfrage weiterschickt
- Der Sender schickt ein ACK zurück über die erstellte Route
- Ab hier sind ATM-Adressen nicht mehr nötig, es werden nur noch die virtuellen Verbindungs-IDs genutzt

### 3.7.8 Serviceklassen

- Es gibt vier Serviceklassen mit verschiedenen Datenraten, mit oder ohne Synchronisierung und mit konstanter oder variabler Bitrate
- Die Klassen sind wie bei QoS für verschiedene Anwendungen

### 3.7.9 Synchronous Digital Hierarchy (SDH)

- ATM hatte Probleme bei der Einführung, andere Technologien sind im WAN deswegen wichtiger geworden
- SDH bietet höhere Datenraten als ATM
- Das Netz wird in einen supraregionalen, einen regionalen und einen lokalen Bereich unterteilt. Je nachdem, an welchem Bereich ein Endknoten hängt, hat er eine unterschiedliche Bandbreite zur Verfügung.

## 3.8 DSL

### 3.8.1 Letzte Meile

- Problem: keine neuen Kabel verlegen
- Nutzung alter Kabel
- Bsp: Modem, ISDN, DSL

### 3.8.2 Modem

- MODOulator-DEMODulator
- Digitale Daten werden in analoge umgewandelt und beim Empfänger zurück in digitale transformiert
- 56 kBit/s
- hohe Fehleranfälligkeit
- v.21 bis v.90
- Möglichkeiten der Modulation:
  - FSK** Frequency Shift Keying, braucht hohe Bandbreite
  - ASK** Amplitude Shift Keying, wenig Bandbreite aber fehleranfällig
  - PSK** Phase Shift Keying, komplexe Demodulation, robust, im Schnitt beste Wahl
  - QPSK** Quaternary Phase Shift Keying, vier Phasen, zwei Bits gleichzeitig codieren
  - QAM** ASK+QPSK, mehr Bits pro Zeiteinheit, aber auch fehleranfälliger
  - PAM** Pulse Amplitude Modulation, 128 verschiedene Amplituden, alle  $125\mu\text{s}$  eine andere

### 3.8.3 DSL

- Bis zu 1000 mBit/s
- Nutzung des gesamten Spektrums des Kupferkabels
- Datenrate hängt von der Entfernung zum Schaltkasten und von der Qualität des Kabels ab
- Automatische Datenratenanpassung
- Mehrere Carrier: 32 Channels mit je 4 kHz Bandbreite up und 256 Hz down
- Benutzung der optimalen Modulationsmethode, je nach Frequenz
- Geräte: Splitter mit Hochpass- (Data) und Tiefpassfilter (Sprache), Modem für die Modulation und eine TAE-Dose
- Im Switching-Center werden Sprache+Data wieder getrennt und die Daten werden auf eine schnelle Leitung gemuxt
- Verschiedene Varianten: HDSL, SDSL, ADSL, VDSL

## 3.9 IP

### 3.9.1 Network-Layer

- Layer 3
- Globale Adressierung
- Routing von Datenpaketen

### 3.9.2 Philosophien

**Verbindungslos** Pakete von variabler Länge werden von der Quelle an das angegebene Ziel gesendet. Spontan und ohne Reservierungen. Pakete können aber auch verschiedene Wege gehen und zu unterschiedlichen Zeiten ankommen.

**Verbindungsorientiert** Herstellen einer Verbindung durch Wahl des Kommunikationspartners. Überprüfung seiner Bereitschaft und Aufbau der Verbindung. Dann Datenübertragung und Beendigung der Verbindung.

### 3.9.3 Internet

- Arpanet als Vorgänger
- Verbindungslos
- Soll Katastrophen überstehen können
- Problem des schnell wachsenden Netzes: verschiedene Protokolle → TCP/IP

### 3.9.4 TCP/IP

- Nicht fehleranfällig, verlässlich und große Verfügbarkeit
- IP als Netzwerkprotokoll
- TCP als End-to-End-Protokoll (Transmission Control Protocol), verbindungsorientiert
- UDP (User Datagram Protocol), verbindungslos

### 3.9.5 Internet 2

- Verbindung: TCP/IP-Benutzung, Erreichbar über eine IP, Senden von IP-Paketen möglich
- Frühe Dienste: Email, Remote-Login, File-Transfer

### 3.9.6 Aufgaben des Internet-Layers

- Daten durch globale Netz schicken
- Wahl der Route in Sub-Knoten
- Kontrolle des Netzwerkstatus, Hilfsprotokolle für Adressübersetzung

### 3.9.7 IP

- IPv4: 32 Bit-Adressen
- Hierarchische Struktur
- Drei Netzwerkklassen
- Vier Adressformate (Multicast, Anycast, Unicast, Broadcast)
- Fragmentierung und Zusammensetzung der Pakete
- Maximal 64 kB, normal 1500 Byte

### 3.9.8 IP-Adressierung

- Klassen A bis E
- A: 1.0.0.0 - 126.255.255.255, erster Block fest
- B: 128.0.0.0 - 191.255.255.255, erster und zweiter Block fest
- C: 192.0.0.0 - 223.255.255.255, ersten drei Blöcke fest
- D: Multicast, 224.0.0.0 - 239.255.255.255
- E: Future-Use, 240.0.0.0 - 255.255.255.255
- Jeder Knoten hat (mindestens) eine einzigartige IP
- Router oder Gateways haben für jedes angeschlossene Netz eine IP

- 0.0.0.0: dieser Host
- 0.Host: Host in diesem Netzwerk
- 255.255.255.255: Broadcast ins eigene Netz
- 127.x.x.x: Loop

### 3.9.9 Routing

- Router haben Tabellen, anhand derer sie die ankommenden Pakete weiterleiten
- Anhand der IP kann direkt die Kasse bestimmt werden und wem das Netz gehört, sowie das Subnetz und das Terminal

### 3.9.10 IP-Adressen

- Sind rar
- Verschwendung vieler Adressen
- IPv6 mit 128 Bit

### 3.9.11 IP-Header

- Version
- Headerlänge
- Type of Service
- Total Length (65536 Byte bzw 64 kByte maximal)
- Identification
- TTL (max 255)
- DF, don't fragment
- MF, more fragments, weitere Fragmente folgen
- Fragment-Offset
- Protocol (UDP, TCP)
- Header-Checksum
- Source, Destination
- Options: hauptsächlich Routing, wird nicht unterstützt

### 3.9.12 IP-Subnets

- Router trennt zB ein Class-B-Netzwerk im dritten Block
- Die Subnetmask gibt an, wieviele Bits einer IP für das Netzwerk-Präfix sind
- Einfaches logisches AND von Subnetmask und IP

### 3.9.13 CIDR

- Classless-Inter-Domain-Routing
- Keine Klassen mehr
- Immer Angabe der Subnetmask

### 3.9.14 Routing-Tabellen

- Je nach Rang des Routers:, Atlantikrouter achten nur auf die ersten 13 Bit, ISP-Router nur auf die ersten 15 Bit, Router in einer Firma auf die ersten 25 Bit

### 3.9.15 NAT

- Private IP-Bereiche (10, 172, 192)
- NAT übersetzt private IP-Adressen in global nutzbare, wenn Traffic nach außen initiiert wird
- Basic NAT: Tabelle mit 1:1-Übersetzungen (genug externe IPs oder IP-Pool und dynamische Zuweisung)
- Hiding-NAT: wenig offizielle IP-Adressen, interne IPs werden auf meist eine externe IP gemapped, Angabe des Ports für die Antwort, Portforwarding

### 3.9.16 IPv6

- Bereinigte Header
- Eingebaut, was in IPv4 noch fehlte (Auth, Priority, ...)
- Header in IPv6 ist länger, aber nur, weil die Adressen länger sind
- IPv6-over-IPv4:
  - Header-Conversion: Router ersetzt IPv6-Header durch einen IPv4-Header
  - Tunneling: IPv4-Header wird vor das ganze IPv6-Paket gesetzt

## 3.10 Routing

### 3.10.1 Routing

- Jeder Router hat eine Tabelle, anhand derer er die ankommenden Pakete entsprechend weiterleitet
- Die Tabelle wird stetig angepasst
- Verbindungslos: jedes Paket muss einzeln geroutet werden und geht eventuell andere Wege
- Durch das Anpassen der Tabelle auf Grund von Responsezeiten, Durchsatz, ... können Pakete unterschiedliche Wege nehmen
- Ein optimales Routing hat einen möglichst kurzen Weg, hohen Durchsatz und kurze Responsezeiten
- Optimales Routing ist meist nicht möglich, da keine vollständigen Informationen über das Netzwerk existieren

### 3.10.2 Routing innerhalb und außerhalb autonomer Systeme

- innerhalb:
  - Routing Information Protocol (RIP)
  - Internet Gateway Routing Protocol (IGRP)
  - Open Shortest Path First (OSPF)
  - Intermediate System to Intermediate System (IS-IS)
- extern:
  - Border Gateway Protocol (BGP)
  - Exterior Gateway Protocol (EGP)
- Router Discovery Protocols:
  - ICMP Router Discovery Protocol (IRDP)
  - Hot Standby Router Protocol (HSRP)

### 3.10.3 Sub-Netzwerk

- Viele untereinander verbundene Router
- Teilweise redundante Verbindungen
- Optimierungen möglich
- Sink oder Sink-Tree für einen Router, Netzwerk ohne Loops mit dem aktuellen Router als Root

### 3.10.4 Statisches Routing

- Source-Routing: Quelle gibt den Weg an
- Internal Routing: Routing anhand der hier statischen Tabelle der Router
- Ein Ausfall ist hier katastrophal

### 3.10.5 Flooding

- Eher statisches Routing
- Router leiten ein Paket über alle Leitungen weiter, außer über die, durch die es gekommen ist
- Sehr robust, aber auch viel Overhead
- Mögliche Loops (TTL, Liste der bekannten Pakete)

### 3.10.6 NCC

- Network Control Center
- Adaptiv und zentralisiert
- Zentrale Station, die Informationen über das Netz sammelt und diese regelmäßig an die Router schickt
- Nur für kleine Netzwerke

### 3.10.7 Local Estimation Procedure

- Isolierte Router
- Routing anhand von Transmission Delays
- Jeder Router entscheidet selbst, sieht aber nur seine Nachbarn

### 3.10.8 Hot Potato

- Isolierte Router
- Schicke über den Pfad mit dem geringsten Load
- Liste des besuchten Pfades

### 3.10.9 Probalistisches Routing

- Statisch und isoliert
- Router entscheidet anhand von Messungen der Performance und durch die daraus folgende Zuweisung eines Proportionalitätsfaktors

### 3.10.10 Shortest Path

- Statisch:
  - Router hat eine Tabelle und entscheidet anhand von konstanten Werten (Kosten, Distanz, Kapazität)
  - Routing nach Dijkstra
- Adaptiv:
  - Dynamisch
  - Dynamische Werte (Delay, aktuelle Kapazität)
  - Regelmäßige Updates der Tabelle

### 3.10.11 Distance Vector Routing

- Adaptive Variante des Shortest Path
- Router kennt Distanzen zu seinen Nachbarn und propagiert diese zu ihnen
- Ändern sich nach Erhalt der Nachbartabelle(n) die minimalen Werte der eigenen Tabelle, schickt er die neuen Werte wieder an seine Nachbarn
- Unverlässliche Informationen
- Propagieren eines Fehlers nach Ausfall einer Leitung oder eines Routers
- Bouncing-Effekt: bei verschiedenen Wegkosten wird zwar das korrekte Ergebnis erreicht, dies geschieht jedoch nicht immer in wenigen Schritten, sondern kann sich auch langsam hochschaukeln (hängt von der Reihenfolge der Nachrichten ab)
- Count to Infinity bei einer Kette, in der ein Randknoten ausfällt

### 3.10.12 Split Horizon Algorithm

- Schicke den Distanzvektor über einen anderen Pfad an den Nachbarn als über den, der für Datenpakete genutzt wird
- In konstruierten Beispielen ist Count to Infinity trotzdem möglich

### 3.10.13 RIP

- Routing Information Protocol
- Distance-Vector alle 30 Sekunden per UDP (Anzahl der Hops als Metrik)
- Maximal 25 Einträge pro Nachricht
- Langsam, Count to Infinity, keine Subnetze
- IGRP (Cisco) mit anderen Metriken

### 3.10.14 Link State Routing

Finde Nachbarn mit HELLO

Link-Kosten durch ECHO, Router antworten sofort, Delay

Erstelle Link-State-Nachricht mit Kosten mit ID, Sequenznummer und TTL

Flooding des Netzes, Sequenznummer inkrementieren, TTL dekrementieren, abgelaufene Pakete fliegen raus, Router quittieren den Empfang

Sammele Daten und erstelle einen Pfadgraphen für das komplette bekannte Subnetz und route per Dijkstra

### 3.10.15 Hierarchisches Routing

- Fasse Router zu Regionen zusammen
- In die Tabelle kommen die Links zu den Routern der eigenen Region und Links zu kompletten fremden Regionen

### 3.10.16 OSPF

- Open Shortest Path First
- Mehrere Areas werden zu autonomen Systemen (AS) zusammengefasst
- Jedes autonome System hat ein Backbone, das alle Teile des AS verbindet
- Jede Area kann einen eigenen Algorithmus und eigene Daten einsetzen, um den kürzesten Weg zu finden
- Jeder Router im Backbone (beinhaltet alle Router, die in mehr als einer Area sind) braucht die Link-State-Datenbank beider Regionen

### 3.10.17 BGP

- Border Gateway Protocol (extern)
- Externe Router müssen nicht nur auf Effizienz achten, sondern auch auf Policies (politisch, wirtschaftlich, ...)
- Variante des Distance Vector Routing-Protokolls, bei dem nicht die Kosten der Übertragung überwacht werden, sondern die komplette Beschreibung von Pfaden
- Die Nachbarrouter senden den Weg, den sie gehen würden. So kann ein Router Loops von Anfang an vermeiden und leicht den besten Weg wählen.
- Routen, die gegen die Policy verstoßen, werden auf *inf* gesetzt

### 3.10.18 Choke-Messages

- Ein überladener Router kann Choke-Messages den Pfad zum Sender zurückschicken, damit dieser weniger sendet

## 3.11 Auxiliary Protocols

### 3.11.1 Helper-Protokolle

- Address Resolution Protocol (ARP)
- Reverse Address Resolution Protocol (RARP)
- Internet Control Message Protocol (ICMP)
- Internet Group Message Protocol (IGMP)

### 3.11.2 ARP

- Hardware-Adresse für eine logische Adresse (IP) herausfinden
- Wird gebroadcastet
- Zuordnungen werden gespeichert
- Optimierungen durch gelegentliches Mitteilen der eigenen IP

### 3.11.3 RARP

- IP eventuell (nach dem Booten) noch nicht fest zugewiesen
- Hardwareadresse wird gebroadcastet, der RARP-Server sucht dann die fest zugewiesene IP heraus

### 3.11.4 DHCP

- DHCP-Discover über einen DHCP-Relay-Agent per Unicast an den DHCP-Server
- Der Server muss nicht im selben Netz liegen
- Komplette Host-Konfiguration

### 3.11.5 ICMP

- Layer 3, baut auf IP auf
- Fehler und Kontrollnachrichten auf dem Netzwerklayer
- Beispiele: Echo, Ping, Traceroute

### 3.11.6 Multicast

- Zwischen Unicast und Broadcast
- Senden an mehr als eine Station aber nicht an alle
- Multicast-Adressen: 224.0.0.0 - 239.255.255.255
- Die Standard-IP-Funktionalitäten werden durch IGMP ergänzt
- Host sagt Router, dass er in eine Gruppe will
- Die Router tauschen die Informationen aus
- Sender sendet an den Router mit einer Gruppe als Empfänger

## 3.12 Quality of Service

### 3.12.1 Problem

Viele Dienste mit unterschiedlichen Anforderungen bzgl. Verlässlichkeit, Delay, Jitter, Durchsatz

### 3.12.2 Keeping QoS

- Kapazität der Router erhöhen
- Buffering
- Traffic-Shaping (Leaky Bucket, Token Bucket), Buffer auf der Senderseite
- Paket-Scheduling (gewichtet), FIFO, Priority Queue, Round Robin (Queues nacheinander), Weighted Fair Queuing (RR + Gewichtung)

### 3.12.3 RSVP

- Resource reSerVation Protocol
- Reserviert Netzwerkkapazität, Pufferkapazität und CPU-Time
- Kein Routing-Protokoll, aber ein Zusatz für ein solches
- Drei Level: Best Effort, Rate sensitive, Delay sensitive
- Der Sender schickt ein Paket, das auf dem Weg Informationen sammelt
- Der Empfänger schickt das Paket auf dem selben Weg wieder zurück und die Router reservieren die Ressourcen (die in dem Paket gefordert werden) oder geben einen Fehler zurück
- Timeout zum Löschen solcher Reservierungen

### 3.12.4 Integrated Services (IntServ)

- RSVP wird benutzt, um QoS für jeden einzelnen Datenfluss zu realisieren
- IP + Verbindungsorientierung
- Drei Klassen: Garantiert (Rate, Delay, Verlässlichkeit), Controlled Load (abgeschwächt, kurze Abweichungen möglich), Best Effort (Normalzustand)

### 3.12.5 Differentiated Services (DiffServ)

- Teile das Netz in Domains (diese unterstützen DiffServ)
- Interne Router fassen DiffServ-Daten zu Klassen zusammen
- ToS im IPv4-Header
- Drei Klassen: Default Forwarding (Best Effort), Expedited Forwarding (Pakete werden so transportiert, als ob kein anderer Traffic da ist, garantierte minimale Übertragungsrate), Assured Forwarding (vier Prioritätsklassen, die die Kapazität beim Router festlegen, niedrig priorisierte Pakete werden eventuell verworfen)

### 3.12.6 MPLS

- MultiProtocol Label Switching
- Das erste Paket gibt die Route vor, der die anderen folgen
- Weiterer Header vor dem IP-Header
- MPLS ist auch im Kernnetz möglich, ohne dass Sender und Empfänger etwas davon mitbekommen müssen

## 3.13 Layer 4

### 3.13.1 Aufgaben

- Verbindungsorientierter oder verbindungsloser Datentransfer
- Ansprechen von bestimmten Kommunikationsprozessen
- Fehlerkontrolle

### 3.13.2 Protokolle

- TCP: verlässlich und verbindungsorientiert
- UDP: verbindungslos, unverlässlich, schnell, Paketpermutationen möglich

### 3.13.3 TCP

- Fehlerfrei, bewahrt die Paketreihenfolge, keine Duplikate
- Errorhandling, ACKs, Flow-Control
- Prozesse werden über die Ports angesprochen
- TCP baut auf dem verbindungslosen IP auf, ist eigentlich auch verbindungslos, fügt aber Kontrollinformationen hinzu, sieht dann für Protokolle auf höheren Layern wie verbindungsorientiert aus
- TCP-Pakete werden auch durchnummeriert und eventuell neu übertragen

### 3.13.4 Socket

- IP+Port
- Zugangspunkt zu einem Server
- Mehrere Verbindungen sind möglich
- TPDU's heißen Segmente
- Segmente für den Verbindungsaufbau, Datenübertragung, Senden von ACKs und Beenden der Verbindung

### 3.13.5 TCP-Header

- Source-Port und Destination-Port
- 20 Byte Header + Options, davon 32 Bit Sequenznummer + 32 Bit ACK-Nummer
- Die Sequenznummer gibt die Bytezahl an, beginnt nicht zwingend mit 0
- Länge des Headers
- Fenstergröße
- Flags: URG, ACK, PSH, RST, SYN, FIN
- Checksum über pseudo-IP-Header

### 3.13.6 Herstellen einer Verbindung

- 3-Way-Handshake gegen das Verlieren, Duplizieren und Speichern von Paketen
- Connection-Request mit Sequenznummer  $x$ , ACK mit  $x + 1$  und mit Sequenznummer  $y$ , ACK mit  $y + 1$  und mit Sequenznummer  $x + 1$
- Bei bekannt gewordenen Fehlern, wird ein REJ statt einem ACK gesendet

### 3.13.7 (ir)regulärer Verbindungsaufbau

- Regulär: SYN, SYN+ACK, ACK
- Irregulär: zwei gleichzeitige SYNs, dann zwei SYN+ACK, es ist aber nur eine Verbindung möglich

### 3.13.8 Datenübertragung

- Sequenz, ACK und Data wird zum Server rübergeschickt
- ACK für die Sequenz (= Sequenz + Datenlänge), Sequenz (= das alte ACK) zum Client
- ACK auf Sequenz (= Sequenz + Datenlänge = Sequenz), Sequenz (= das alte ACK), Daten zum Server

### 3.13.9 Flow-Control

- Bytes statt Frames
- Buffer kann bei Out-of-Order-Segmenten dynamisch vergrößert werden
- Viele Verbindungen können sich einen Buffer teilen

### 3.13.10 Windows

- Dynamic Sliding Window: informiere den Sendenden regelmäßig über den Bufferstand
- Silly Window: um den Buffer nicht direkt wieder voll machen zu lassen, warte bis der Buffer leer genug ist und teile es dann dem Sendenden mit

### 3.13.11 Termination

- 4-Way Handshake
- FIN, ACK, FIN, ACK

### 3.13.12 Timer

- Sende und starte Timer
- Wenn der Timer triggert, sende das Paket neu oder beende die Verbindung
- Persistence-Timer: sende ein Testpaket, um einen eventuellen Verlust einer Buffer-Release-Nachricht zu erkennen
- Keep-Alive-Timer: Terminiere, wenn die Gegenstelle lange nicht reagiert
- Time-Wait-Timer: Timer für zu spät kommende Pakete bei der Terminierung einer Verbindung
- Retransmission Timer: zum erneuten Senden eines Segments

### 3.13.13 States während einer Verbindung

CLOSED, LISTEN, SYN RCVD, SYN SENT, ESTABLISHED, FIN WAIT 1, FIN WAIT 2, TIME WAIT, CLOSING, CLOSE WAIT, LAST ACK

### 3.13.14 Congestion Control

- Vermeidung der Überlastung des Netzwerks
- Wird mit der Maximum Segment Size (MSS) des Senders initialisiert
- Ohne Timeout wird das Fenster verdoppelt, bis zu einem Threshold, ab da linear
- Nach einem Timeout wird wieder bei den Anfangswerten angefangen

### 3.13.15 Fast-Retransmit und Fast-Recovery

- Beim Empfang eines Pakets außer der Reihe wird für dieses und die weiteren ein ACK für das letzte korrekte Paket gesendet. Der Sender wiederholt nach dem dritten ACK das fehlende oder falsche Paket.
- Nach dem empfang des dritten ACK braucht der Threshold(?) nicht auf 0 gesetzt zu werden sondern braucht nur halbiert werden, da weitere Pakete korrekt angekommen sind

### 3.13.16 UDP

- 8 Byte Header
- Keine ACKs, schnell
- Checksum (Standard)
- Kein großer Overhead
- DNS, BOOTP, TFTP, SNMP, RIP

### 3.13.17 Conclusion

- TCP für normale Applikationen
- UDP für Verbindungsaufbau oder für schnelle Datenübertragung (DNS, Streams)

## 3.14 Layer 5-7

### 3.14.1 Layer 5

- Session-Layer
- Kleinster der Application-orientierten Layer
- Kontrolliert Dialoge und setzt regelmäßig Synchronisierungspunkte

### 3.14.2 Layer 6

- Presentation-Layer
- Die Ausgangsdatenstruktur (zB ASCII) wird in eine eigene Datenstruktur übersetzt und beim Sender wieder in die Ausgangsdatenstruktur zurück umgewandelt

### 3.14.3 Huffman-Code

- Es existieren Wahrscheinlichkeiten für bestimmte Buchstaben. Baue anhand der Wahrscheinlichkeiten einen Binärbaum wobei die unwahrscheinlichen als erstes genommen werden. Die Blätter repräsentieren jeweils die Buchstaben.
- Beschrifte nun die Pfade von einem Vater zu seinen Söhnen mit 0 bzw 1
- Die Blätter werden nun durch die Bits vom Root aus dargestellt
- Wahrscheinlichere Buchstaben stehen weiter oben im Baum und haben deswegen eine kürzere Bit-Repräsentation als unwahrscheinlichere

## 3.15 DNS

### 3.15.1 Namespace

- Maximal 127 Level
- Maximal 63 Zeichen für Domainnamen
- Fully Qualified Domain-Name ist der absolute Domain-Name
- Relative Domain-Name ist nur mit einer Referenz auf einen FQDN gültig

### 3.15.2 Name-Server und Zonen

- Ein NS ist für eine Zone verantwortlich
- Ein Server kann aber auch mehrere Zonen verwalten
- Typen: Primary (Master) NS ist notwendig für eine Zone und mindestens ein Secondary (Slave) NS als eine Art Backup
- Secondary NS zur Teilung des Loads

### 3.15.3 Resource-Record

**SOA** gibt an, wer der Master-NS der Zone ist

**NS** gibt die Nameserver für eine Zone an

**A** Adresse des Hosts, mindestens eine muss vorhanden sein

**CNAME** Alias für Domain-Namen

**PTR** für Reverse-DNS-Lookups

**MX** x@Domain läuft über einen Relay, also x@Relay

## 4 Massively Distributed Systems

### 4.1 Motivation

#### 4.1.1 Heutige Probleme und gleichzeitig die wichtigsten Eigenschaften von P2P-Systemen

- Scalability
- Flexibility
- Security / Reliability

#### 4.1.2 Warum Peer-to-Peer

- Bandbreite, Rechenpower und Festplatten der heutigen Heimrechner werden immer größer
- Client-Server-Architektur hat nur limitierte Ressourcen (nicht skalierbar)
- Server als Single-Point-of-Failure
- Server können zensiert werden

#### 4.1.3 Probleme des heutigen Internets

- NAT
- Middleboxes (Proxies, ...)
- Dynamische Adressen

#### 4.1.4 Definition Peer-to-Peer

- P2P ist eine Klasse von Applikationen, die die Vorteile der Ressourcen der Endverbraucher nutzen. Diese dezentralisierten Ressourcen zu nutzen bedeutet, dass in einer Umgebung mit unsicherer Konnektivität und nicht vorhersehbaren IP-Adressen gearbeitet werden muss. Die P2P-Knoten müssen außerhalb des DNS-Systems arbeiten und sind komplett oder größtenteils nicht von zentralen Servern abhängig.
- Unterscheidung zwischen de-centralized resource usage und de-centralized information management

#### 4.1.5 Herausforderung für verteilte Systeme

- Wo sollen Data-Items gespeichert werden?
- Wie findet jemand diese Data-Items?
- Skalierbarkeit
- Robustheit

#### 4.1.6 Klassifizierung von P2P

- Filesharing (Napster, ...)
- Grid Computing (SETI)
- Instant Messenger (ICQ)
- Collaboration (Groove Workspace)

#### **4.1.7 ... oder durch die gescharte Ressource**

- Information (IM)
- Files
- Bandbreite
- Speicherplatz
- Prozessor-Cycles

#### **4.1.8 Overlay Network**

- Definition: Zwei Applikationen kommunizieren über einen anderen Pfad als den gegebenen Pfad des Netzwerk-Layers
- Ein Overlay-Netzwerk ist ein logisches Netzwerk über der darunter liegenden Topologie
- Ein Overlay kann seine eigene Topologie bilden

### **4.2 Unstructured Peer-to-Peer-Systems**

#### **4.2.1 Arten**

- Centralized P2P (Napster)
- Pure P2P (Gnutella 0.4)
- Hybrid P2P (Gnutella 0.6)

#### **4.2.2 Charakteristika**

- Peer: Knoten, der im Overlay aktiv ist
- Overlay: virtuelles Netzwerk, das TCP-Verbindungen zwischen Peers erstellt, komplett unabhängig von dem physischen Netz, eventuelle Hierarchien, eventuell zentrale Elemente, eventuell komplett randomisiert

#### **4.2.3 Bootstrapping**

- Meistens Teil des Protokolls
- Es muss mindestens ein Knoten des Overlays bekannt sein

### **4.3 Zentralisierte P2P-Netzwerke**

#### **4.3.1 Zentraler Server**

- Zum Verwalten der vorhandenen Daten
- Als Bootstrapping-Server
- Single Point of Failure (SPOF)
- Schnelles und vollständiges Lookup
- Server als Bottleneck

### 4.3.2 Napster

- Zentraler Server verwaltet die Daten
- Peers fragen die Daten ab, der Server antwortet mit den Ergebnissen
- Der Peer pingt die Ergebnisse an und lädt dann per http von diesen herunter
- Login, Login-ACK, viele Notify
- Search, mehrere Response, HTTP-GET

## 4.4 Pure-P2P-Systems

### 4.4.1 Charakteristika

- Kein SPOF, da es keine zentrale Entität gibt
- Längeres Lookup mit nicht garantierten Ergebnissen, auch wenn die Datei im Netz vorhanden ist
- Jeder Peer kann entfernt werden, ohne dass die Stabilität des Netzes gefährdet ist
- Verbindungen entstehen zufällig
- Bootstrap-Server nötig, wenn keine aktiven Peers bekannt sind
- Komplette dezentrales Routing

### 4.4.2 Gnutella 0.4

- Verbindung mit mindestens einem Peer
- Erforschung der Nachbarschaft (periodisch) durch Ping/Pong
- Suche anhand von Stichworten, die den Nachbarn geschickt werden (Query)
- Wähle das beste Ergebnis aus, werden durch Query-Hit übermittelt
- Verbinde dich mit dem Peer
- Sitzt der anbietende Peer hinter einer Firewall, kann der Peer durch eine Push-Nachricht dazu gebracht werden, die Datei zu senden (kann zu DDoS missbraucht werden)

### 4.4.3 Nachrichten in Gnutella 0.4

- Jede Nachricht hat eine TTL und die Anzahl der Hops, die sie schon gemacht hat
- PING: kein Payload
- PONG: Port, IP, Daten über die gescharten Daten
- Query: minimale Geschwindigkeit, Suchkriterien
- Query-Hit: Anzahl der Hits, Liste der Hits, Port, IP, Geschwindigkeit, NodeID

### 4.4.4 Flooding in Gnutella 0.4

- TTL von 7, TTL + Hops ist dann maximal 7
- Jeder Peer speichert die Nachrichten für eine bestimmte Zeit um Loops zu vermeiden
- Normales Broadcast der Nachrichten an alle Nachbarn (außer an den Sender)
- Großer Overhead durch das Flooden, Peers mit geringer Bandbreite haben Probleme
- Es kann mit einem Query nicht das komplette Netzwerk durchsucht werden
- Zickzack-Routen bzgl des physischen Netzwerks

## 4.5 Hybrid-P2P-Systeme

### 4.5.1 Charakteristika

- Einführung von Super-Peers mit einem höheren Kantengrad
- Super-Peers bilden eigenes Overlay
- Anfragen der Peers werden an die Super-Peers gestellt, die dann die Anfrage an die anderen Super-Peers weiterleiten

### 4.5.2 Gnutella 0.6

- Die normalen Peers teilen ihrem Super-Peer mit, was sie für Daten anbieten
- Wer Super-Peer wird, wird gemeinschaftlich anhand der Ressourcen der Peers entschieden (beim Verlassen eines Super-Peers oder wenn dieser zu viele Verbindungen hat)

### 4.5.3 Routing in Gnutella 0.6

- Die Leaf-Nodes senden die Anfrage an den Super-Peer
- Der guckt in seiner Tabelle nach, ob das Gesuchte in seinem Subnetz vorhanden ist
- Zusätzlich leitet er die Anfrage an seine Super-Peer-Nachbarn weiter, die die Anfrage so behandeln, wie er
- Die Super-Peers leiten dann nur an die Peers weiter, die die Dateien höchstwahrscheinlich anbieten. Ist das so, senden sie ein Query-Hit an den anfragenden Peer (über den Query-Weg) zurück
- Der Datenaustausch geschieht dann wieder über eine direkte http-Verbindung

### 4.5.4 Ergebnis von Gnutella 0.6

- Noch immer hoher Traffic
- Anfragen werden nicht an das komplette Netz gestellt
- Zickzack-Routen möglich (physisches Netz)
- Asymmetrische Verteilung der Last

## 4.6 Random Graphs, Small-Worlds, Scale-Free Networks

### 4.6.1 Small-World

- Komplexe natürliche Netzwerke entstehen nicht zufällig, sie haben *Small-World*-Charakteristika
- In sozialen Netzwerken kennt eine Person viele in ihrer nahen Umgebung und wenige weiter entfernt. Die Netze können sich schnell ändern, die grundlegenden Strukturen bleiben aber erhalten.
- Cluster-Koeffizient:  $C(v) = \frac{\text{Anzahl der Verbindungen zwischen den Nachbarn von } v}{\text{Anzahl der möglichen Verbindungen zwischen den Nachbarn von } v}$
- Small-World-Graphen haben in etwa einen Durchmesser von  $O(\sqrt{n})$

### 4.6.2 Zufällige Graphen

**Erdős-Renyi** Bilde alle Graphen mit  $n$  Knoten und  $m$  Kanten und wähle zufällig einen aus dieser Menge aus. Mit hoher Wahrscheinlichkeit ist der Graph zusammenhängend, wenn der Grad der Knoten im Schnitt  $O(\log n)$  ist. Des Weiteren ist der Durchmesser  $O(\log n)$ , wenn der Graph zusammenhängend ist.

**Gilbert** Den Kanten eines vollständig verbundenen Graphen mit  $n$  Knoten werden Wahrscheinlichkeiten zugeordnet. Liegt die Wahrscheinlichkeit größer als eine vorher festgelegt ist, wird die Kante gelöscht (oder halt umgekehrt, dass die Kante erstellt wird). Der Kantengrad ist  $(n - 1) \cdot p$ .

### 4.6.3 Milgram's Small-World-Experiment

- Briefe, die nur über Bekannte weitergegeben werden sollten
- Die Briefe, die ankamen (5%), brauchten im Schnitt 6 Hops

### 4.6.4 Watts-Strogatz-Models

- Konstruierung von Netzen mit Small-World-Eigenschaften
- Grid-like-Netzwerke haben eine lange durchschnittliche Pfadlänge, wenn sie einen hohen Cluster-Koeffizienten haben
- Vergleich von natürlich entwickelten Graphen und deren zufälliger Verteilung der Kanten: der Durchmesser ist vergleichbar, der Cluster-Koeffizient ist jedoch bei den natürlich entwickelten Graphen teilweise mehr als das Zehnfache größer, Small-World-Graphen haben also eine dichte lokale Struktur
- Erstelle einen Graphen, bei dem jeder Knoten mit seinen umliegenden  $k$  Nachbarn verbunden ist. Entscheide dann für jede Kante (mit einer gewissen Wahrscheinlichkeit), ob diese Kante gelöscht und durch eine andere, zufällige ersetzt wird. Mit höherer Wahrscheinlichkeit steigt die Zufälligkeit des Graphen.
- Bei 5000 Knoten und einer Wahrscheinlichkeit von 0.01 ist der Cluster-Koeffizient noch immer sehr hoch, die durchschnittliche Pfadlänge hat jedoch rapide abgenommen

### 4.6.5 Kleinberg-Model

- Jeder Knoten liegt in einem  $d$ -dimensionalen Grid und leitet jede Nachricht an den Nachbarn weiter, der den kürzesten Abstand zum Ziel hat
- Mit einer gewissen Wahrscheinlichkeit (abhängig von der Dimension) werden nun Shortcuts eingebaut (ungefähr so viele, wie Dimensionen existieren)
- Werden weniger Shortcuts eingebaut, so sind die Pfade zu lang, bei zu vielen Shortcuts wird die Entscheidung zu schwer

### 4.6.6 Barabasi-Albert

- Knoten mit vielen Kanten neigen dazu, mehr zusätzliche Kanten zu bekommen als Knoten, die wenig Kanten besitzen (the rich get richer)
- Power-Law: die Wahrscheinlichkeit, dass ein Knoten viele Kanten hat, ist gering, wobei die Wahrscheinlichkeit, dass ein Knoten wenig Kanten hat, größer ist
- Ein Grad, der dem Power-Law gehorcht, ist Scale-Free
- Barabasi-Albert-Model: Knoten mit hohem Kantengrad bekommen wahrscheinlicher neue Kanten dazu als Knoten mit geringem Kantengrad

### 4.6.7 Stabilität von Scale-Free-Netzwerken

- Wird zufällig ein Knoten aus einem Scale-Free Netzwerk ausgesucht und ausgeschaltet, so ist das mit hoher Wahrscheinlichkeit einer mit einem geringen Kantengrad. Solch ein Netzwerk ist also bzgl zufälliger Ausfälle stabiler als ein zufälliges Netzwerk.
- Wird jedoch ein Angriff auf einen bestimmten Knoten gefahren, so ist dies eher einer mit einem hohen Kantengrad, was das Netzwerk stark beeinträchtigt
- Gnutella folgt dem Power-Law (einige Nodes werden präferiert)

## 4.7 Structured P2P-Networks

### 4.7.1 Distributed Indexing im Kontext

- Zentralisierte P2P-Netzwerke haben eine Suchkomplexität von  $O(1)$  aber dafür eine Kantenverwaltung von  $O(n)$
- Pure-P2P- und Hybrid-P2P-Netzwerke haben eine Kantenverwaltung von  $O(1)$  aber dafür eine Suchkomplexität von  $O(n)$
- Netzwerke, die auf Distributed-Hash-Tables (DHT) basieren, liefern bei beiden ungefähr eine Komplexität von  $O(\log n)$
- Es gibt hier keine False Negative und das Netzwerk ist resistent gegen Angriffe und Ausfälle

### 4.7.2 Charakteristika

- Es gibt einen Adressraum, auf den sich jeder Node und jedes Data-Item abbilden lässt
- Jeder Knoten ist für die Data-Items verantwortlich, deren ID numerisch kleiner ist als seine ID aber größer als die ID seines Vorgängers
- Jeder Knoten verwaltet eine Liste von Nodes im Overlay und routet eine Anfrage für ein Data-Item an den Knoten aus der Tabelle, der numerisch am nächsten an der ID des Data-Items liegt, aber gleichzeitig kleiner als seine ID ist
- Als Data-Item kann je nach Anwendung entweder Port+IP des Nodes gespeichert werden, der die Datei anbietet oder die Datei selber

### 4.7.3 Chord

- Einfacher Algorithmus
- $O(\log n)$  States pro Node und durchschnittliche Pfadlänge
- $O(\log^2 n)$  beim Hinzufügen und Entfernen eines Knotens
- Jeder Node verwaltet eine Liste von anderen Nodes, die einer Formel folgt: eigene ID +  $2^{\text{Zeile des Eintrags}}$ . Diese ID wird geroutet und der Successor wird in die Liste (Finger-Table) gepackt. Die Liste hat also so viele Einträge, wie der Identifier Bits hat.
- Mit jedem Schritt kann also die Strecke vom aktuelle Node zum Zielnode ungefähr halbiert werden
- Bei Failures von Nodes während des Routings wird an den zweitnächsten Node geschickt, aber so, dass man nicht über das Ziel hinausschießt
- Jeder Node verwaltet zusätzlich noch eine Successor-Liste zu seinen  $k$  nächsten Nodes, so kann der Ring stabil bleiben
- Beim Hinzufügen eines neuen Knoten routet dieser zu seiner ID und kennt dadurch seinen Successor. Von diesem erhält er die Data-Items, für die er nun zuständig ist. Dann baut er seine Finger-Table auf und bekommt die Successor-Liste von seinem Successor. Nun müssen die anderen Nodes noch davon erfahren, dass der neue Knoten ins Netz gekommen ist. Dafür wird der Predecessor  $t_i$  der Schlüssel  $(s - 2^i)$  dazu gebracht, seine Finger-Table zu updaten.
- Entfernen von Knoten werden als Failures betrachtet. Der Successor übernimmt die Daten des entfernten Knoten und teilt dies den Nodes mit, die eventuell auf diesen gezeigt haben.
- Regelmäßig wird Fix-Fingers (ist ein Eintrag noch aktuell?) und Stabilize (ist mein Nachfolger noch mein Nachfolger?) ausgeführt
- Die Mitteilungen zur Änderung der Finger-Tables anderer Knoten kann auch lazy über Fix-Fingers geschehen

#### 4.7.4 Pastry

- Es gibt  $l$ -Bit Identifier, die als Identifier zur Basis  $2^b$ , normalerweise mit  $b = 4$ , interpretiert werden
- Jeder Knoten verwaltet nun eine Routing-Tabelle, ein Leaf-Set und ein Neighborhood-Set
- Beim Joinen eines Knotens (Bootstrap-Server) erstellt der Knoten eine ID und routet wieder nach dieser
- Auf dem Weg zum Ziel bekommt der Knoten bei jedem Hop eine Zeile aus deren Routing-Tabelle übermittelt. Von dem Ziel-Knoten bekommt der neue Knoten das Leaf-Set und das Neighborhood-Set.
- Die Routingtabelle ist nun so aufgebaut, dass die Einträge in der  $i$ -ten Zeile (von 0 beginnend) einen Präfix der Länge  $i$  mit dem aktuellen Knoten teilen
- Das Leaf-Set beinhaltet die  $k$  numerisch nächsten Knoten, das Neighborhood-Set beinhaltet die geographisch nächsten Nodes
- Beim Routing wird nun zuerst überprüft, ob ein Knoten des Leaf-Sets für die ID verantwortlich ist. Wenn nicht, wird versucht, über die Routing-Tabelle einen Schritt näher zu kommen (größerer Präfix zur Ziel-ID). Ist ein solcher Eintrag nicht vorhanden, wird die numerisch nächste ID aus allen drei Listen zusammen gewählt.
- Bei Pastry ist der Knoten für ein Daten-Item verantwortlich, der am numerisch nächsten zu diesem liegt
- Die Knoten des Leaf-Sets werden periodisch überprüft, Fehler in den anderen Listen werden beim Routing erkannt
- $O(\log n)$  bei den States pro Node, bei der durchschnittlichen Pfadlänge und beim Joinen von Nodes

#### 4.7.5 CAN

- Bei CAN gibt es einen  $d$ -dimensionalen Raum, der in Zonen eingeteilt wird, für die die Knoten zuständig sind
- Ein Data-Item und ein Knoten hat eine ID in diesem Raum, wobei die ID des Knotens nur beim Joinen wichtig ist
- Zwei Zonen sind benachbart, wenn sie  $d - 1$  Dimensionen miteinander teilen
- Das Routing zu einer Ziel-ID geschieht, indem ein Knoten die Anfrage an den Nachbarn weiterleitet, der näher an dem Ziel ist
- Will nun ein Node joinen, so generiert er seine ID und routet in die Zone, die dafür verantwortlich ist. Diese Zone wird nun zwischen dem aktuellen Verwalter und ihm hälftig aufgeteilt (Art des Schnitts abhängig von der letzten Teilung). Die Data-Items werden dann dem neuen Knoten übertragen. Als Optimierung kann ein Knoten sich auch vor der Teilung einer Zone die Nachbarzonen anschauen und dann entscheiden, welche Zone am besten geteilt wird.
- Fällt ein Knoten aus, so schicken seine Nachbarn nach dem Bekanntwerden sogenannte Takeover-Nachrichten und der schnellste übernimmt die Zone, versucht, diese mit seiner zu mergen und verwaltet im Falle des Misserfolg zwei Zonen
- $O(d)$  States pro Node,  $O(\frac{d}{4} \cdot n^{\frac{1}{d}})$  als durchschnittliche Pfadlänge und  $O(d \cdot n^{\frac{1}{d}})$  als Komplexität beim Joinen eines Nodes
- Als weitere Optimierung können mehrere Realitäten verwendet werden. Ich stell mir das so vor, dass eine andere Hashfunktion benutzt wird, die anderen Identifier generiert und damit alles durcheinanderwürfelt.
- Zusätzlich können noch mehrere Nodes für eine Zone verantwortlich sein oder die Nodes können ihre Nachbarn anpingen um nicht nur zum nächsten Nachbarn zu routen, sondern auch zum schnelleren von gleich weit entfernten Nachbarn

## 4.8 P2P-Applications

### 4.8.1 Internet Indirect Infrastructure

- In heutigen Internet konnten viele gute Ansätze nicht erfolgreich umgesetzt werden: Multicast, Anycast, Mobility, QoS
- Der normale Fall ist, dass ein Paket per Unicast von A nach B gesendet wird
- Ein Client meldet sich für ein Data-Item an, indem er unter der ID des Data-Items seine IP speichert (Setzen eines Triggers). Der Sender sendet die Daten dann an die ID und der i3-Node, der dafür verantwortlich ist, leitet die Daten weiter.
- Mobility: Der Client ändert die IP bei allen IDs, für die er sich angemeldet hat, sobald seine IP sich wechselt
- Multicast: Mehrere melden sich für eine ID an, verschiebt das Problem jedoch nur. Hier kann ein Distribution-Tree mit weiteren Triggern erstellt werden.
- Anycast: Teilnehmer der Anycast-Gruppe wählen einen Präfix und jeder einen Postfix. Die Trigger werden nun alle bei einer ID abgelegt (Präfix). Der Sender wählt nun einen eigenen Postfix (zB Vorwahl) und der i3-Node entscheidet anhand des längsten Präfixes der Trigger, an wen die Daten gesendet werden.
- Sender und Empfänger können eine beliebige Verknüpfung von Triggern setzen. So können Daten vom Sender oder vom Empfänger zu verschiedenen Services umgeleitet werden, um sie zB umcodieren zu lassen.
- Um geographisch große Umwege zu vermeiden, werden die Public-IDs nur für die Initiierung eines Kommunikationsprozesses benutzt. Danach können geographisch nahe Nodes zur Kommunikation genutzt werden.

## 4.9 Anonymity

### 4.9.1 Zensur

- Öffentliche Zensur oder heimliche Zensur
- Filtern anhand von Inhalt, Domain-Namen, IPs, Autoren
- Filtern durch Blacklists oder Whitelists
- Techniken: Proxies, DNS, Filtern von Ergebnissen in Suchmaschinen, Ausschluss aus dem Netzwerk, Löschen von Informationen
- Anonymität: wenn ein Element aus einer Gruppe nicht mehr von den anderen zu unterscheiden ist
- Identität: was auch immer etwas gleich oder unterschiedlich macht
- Identifier: ein Anhaltspunkt, durch den man jemanden identifizieren kann
- Pseudonym: künstlicher Identifier

### 4.9.2 FreeNet

- Jeder User besitzt eine GUID, die er sich nicht selber wählen kann (wird von zufällig gewählten Nodes vergeben)
- Daten werden so verschlüsselt und mit einer GUID versehen, dass keiner anhand der öffentlichen Daten auf den Inhalt schließen kann
- Die Daten werden nun an die ID geroutet (wenn diese nicht vergeben ist) und auf dem Weg dorthin von allen Nodes gespeichert, die die Datei weiterleiten. So kann eine Anfrage eventuell schneller beantwortet werden.
- Eine Anfrage für eine Datei verläuft ähnlich wie bei unstrukturierten Netzwerken: hat der Gefragte die Datei nicht, leitet er diese an den Nachbarn weiter, dessen ID besser zu dem Key passt

## 4.10 ePost

### 4.10.1 Email und P2P

- Normalerweise werden MTAs benutzt
- Email ist teilweise Peer-to-Peer, aber es basiert auf dem fast zentralisierten DNS und jeder User hat einen fest zugewiesenen Mailserver
- Eine Email-P2P-Variante sollte erlauben, dass die User die Emails von verschiedenen, vorher nicht festgelegten Maschinen abholen können

### 4.10.2 Ziele

- Erstellen einer allgemeinen, Server-freien Plattform für Email, IM, Kalender, ...
- Zeigen, dass viele Services von einer serverlosen Plattform angeboten werden können
- Zeigen, dass das Ganze mit Peer-to-Peer-Paradigmen zu schaffen ist

### 4.10.3 PAST

- Wie Pastry: Data-Items werden beim numerisch nächsten Node gespeichert
- Data-Items werden aber bei den  $k$  numerisch nächsten Nachbarn repliziert
- Mit  $k = 3$  sind Daten zu 99,99% verfügbar

### 4.10.4 Scribe

- Scribe ist eine Multicast-Infrastruktur, die auf Pastry aufsetzt
- Die Schnittstelle der Anwendung ist die folgenden: *create* (eine Gruppe mit einer groupID wird erstellt), *join* (der Gruppe groupID beitreten), *leave* (die Gruppe groupID verlassen), *multicast* (die angegebene nachricht an die Gruppe groupID senden)
- Die groupID ist der Hash des Titels der Gruppe, diese wird an dem Pastry-Node erstellt, der für die groupID zuständig ist (er ist dann der Rendezvous-Point)
- Will ein Knoten einer Multicast-Gruppe beitreten, so routet er eine join-Nachricht zum Rendezvous-Node. Kommt die Nachricht auf dem Weg dahin bei einem Node vorbei, der schon Mitglied der Multicast-Gruppe ist, so leitet dieser die Nachricht nicht weiter und fügt den Node einfach in seine Kindermenge bzgl dieses Multicast-Trees ein.
- Beim Senden einer Multicast-Nachricht an eine Gruppe wird diese an den Rendezvous-Node geschickt. Dieser speichert die IPs seiner Kinder, so dass die Nachricht nicht über Pastry geroutet werden muss.
- Will ein Node eine Gruppe verlassen, so überprüft er, ob er Kinder in seiner Tabelle hat. Wenn nicht, schickt er einfach eine leave-Nachricht. Wenn wohl oder wenn der Ausfall des Parents aufgefallen ist, routet der jetzt vaterlose Node einfach wieder zum Rendezvous-Node und bekommt so einen neuen Vater-Knoten. Fällt der Rendezvous-Knoten aus, schicken dessen Kinder eine Join-Nachricht los. Diese wird dann an den numerisch nächsten Node geleitet, der dann der neue Rendezvous-Knoten ist.

### 4.10.5 POST

- POST ist ein Layer über PAST/Scribe und bietet Sicherheit, Verfügbarkeit und Verlässlichkeit
- Jede Nachricht bzw alle Daten werden verschlüsselt
- Den Usern werden Schlüsselpaare, eine global eindeutige ID und ein verifizierbares Zertifikat zugewiesen
- Bei  $H(\text{Username})$  wird das Zertifikat und das Log des Users gespeichert
- POST benutzt PAST um die Daten sicher zu speichern

- Um eine Nachricht  $X$  zu speichern wird der Hash von  $X$  berechnet, damit wird  $X$  verschlüsselt und beim Hashwert der verschlüsselten Nachricht gespeichert
- User-spezifische Daten werden im Log gespeichert. Der Log-Head (bei  $H(\text{Username})$ ) zeigt auf den neuesten Eintrag, jeder Eintrag zeigt auf seinen Vorgänger. Mehrere Logs pro User möglich.
- Jeder User hat eine Scribe-Gruppe. Will nun jemand an einen anderen User eine Mail senden, so schickt er diese an einen zufälligen Knoten mit der Bitte, diese zuzustellen. Dieser Knoten tritt nun der Gruppe des Empfängers bei und wartet, bis dieser seiner Gruppe mitteilt, dass er online ist.

#### 4.10.6 ePost

- ePost basiert nun auf POST
- Eine Mail wird in ihre Einzelteile zerlegt (Headers, Body, Anhänge) und im System gespeichert. Im Log (anscheinend zu dieser Mail) steht nun sowas, wie "Mail gesendet", "Mail gelesen" oder "Mail gelöscht".
- Jeder teilnehmende Knoten lässt zusätzlich einen IMAP/POP3 und SMTP-Server laufen, damit Mails mit einem der diversen Clients abgerufen werden können
- Für die Zusammenarbeit zwischen ePost und der existierenden Email-Infrastruktur müssen Gateway-Knoten existieren