

## Zusammenfassung - Netzwerke II

### Drahtlose Netzwerke

#### 1.1.1 Grundlagen

**Wireless Host (Drahtloser Teilnehmer):** Endsystem auf dem die Applikation läuft (stationär oder mobile), z.B. Smartphone, PC

**Wireless Link (Drahtlose Verbindung):** Verbindet Teilnehmer direkt oder per Basisstation (Abdeckung, Datenrate)

**Basisstation (Base Station):** Überträgt Datenpakete zwischen drahtgebundenem zu drahtlosem Netzwerk, meist mit drahtgebundenem Netzwerk verbunden (WLAN Access Point, UMTS Basisstation)

**Drahtloses Infrastruktur Netzwerk:** Netzwerkteilnehmer sind über Basisstation mit dem Netz verbunden

**Drahtloses Ad-Hoc Netzwerk:** Keine Infrastruktur (Basisstationen), Teilnehmer bilden das Netz selbst.

Nachteile: passive Teilnehmer haben trotzdem Stromverbrauch, eigene Daten landen auf fremden Mobiltelefonen und höhere Latenz

**Single-Hop:** Genau ein wireless Link

**Multi-Hop:** Übertragung geht über mehrere wireless Links in Folge

Übliche Datenraten		Beispiele für Single und Multi-Hop	
		Single Hop	Multiple Hops
<b>GSM (2G)</b>	0.56 Mb/s	Infrastruktur	Host verbindet sich mit Basisstation (Wifi, zellulare Netzwerke) und diese dann mit dem Internet
<b>UMTS (3G)</b>	4 Mb/s		
<b>LTE (4G) und 802.11b</b>	5 - 11 Mb/s		
<b>802.11ag</b>	54 Mb/s	Keine Infrastruktur	Keine Basisstation und auch keine Verbindung zu weiterem Internet. Muss durch mehrere drahtlose Geräte: <i>MANET</i> , <i>VANET</i>
<b>802.11n</b>	200 Mb/s		

#### Herausforderungen bei drahtloser Übertragung

- Teilnehmer zeitweise nicht erreichbar (Funkloch)
- IP-Adresse ändert sich
- Höhere Anzahl an Übertragungsfehlern durch Interferenz (Störung durch andere Teilnehmer) oder Dämpfung → Bessere Fehlerbehandlung
- Kurzer Paketverlust führt bei TCP zu angeblicher Netzüberlastung (obwohl nur kurzzeitige Störung)
- Medium kann abgehört werden
- Mehrwege-Ausbreitung: Signale werden an unterschiedlichsten Oberflächen reflektiert → Am Empfänger sowohl konstruktive als auch destruktive Überlagerung möglich

⇒ Funkkanal ist zeit- und ortsvariant!

**Modulationsarten:** Frequenz-, Amplituden- & Phasenmodulation, Quadraturamplitudenmodulation (QAM) ⇒ Kombination von Amplituden- und Phasenmodulation (*QAM-8*: 3 Bit pro Symbol, *QAM-1024*: 10 Bit pro Symbol).

Höhere Modulationsarten bieten höhere Übertragungsrate sind aber fehleranfälliger. Bei größerem Signal-Rausch-Abstand (SNR - Stärke des Nutzsymbols bezogen auf Störung) kann höhere Modulation eingesetzt werden da Kanal anscheinend nicht so stark gestört (*QAM-16* = 4Mbps, *QAM-256* = 8Mbps)

**Bit-Error-Rate (BER):** Wahrscheinlichkeit, dass ein fehlerhaftes Bit übertragen wird.

**Hidden Terminal Problem:** Teilnehmer A, B & C. A und B hören sich, B und C hören sich aber A und C hören sich nicht → Bei Übertragung  $A \rightarrow B$  und  $C \rightarrow B$  stören sie sich unbewusst gegenseitig.

TODO: BEHEBUNG / VERMINDERUNG DURCH?

#### Aufteilen eines Mediums:

- TDMA (Time Division Multiple Access)
  1. synchron: Jeder Teilnehmer hat festen Zeitslot, nur in diesem kann er senden
  2. asynchron: keine festen Zeitslot, jeder nutzt aktuellen Zeitslot wenn er Daten hat - Absender wird in Header geschrieben

- **FDMA** (Frequency Division Multiple Access)
  1. Teilnehmer nutzen unterschiedliche Frequenzen
- **CDMA** (Code Division Multiple Access)
  1. Teilnehmer nutzen unterschiedliche Spreizcodes, **Vorteil:** Störungsunempfindlicher, **Nachteil:** Mehr Datenübertragung
  2. Zu übertragende Daten werden vom Sender mit Spreizcode multipliziert, Ergebnisbits  $\Leftrightarrow$  **Chips**
  3. Empfänger multipliziert empfangende Daten mit Spreizcode des Senders
  4. Teilnehmer senden zur gleichen Zeit im gleichen Band, Daten werden beim Empfänger durch bitweise Multiplikation mit Code zurückgewonnen
  5. Andere Teilnehmer wirken als zusätzliches Rauschen ( $\Rightarrow$  Umso mehr Teilnehmer umso geringerer SNR  $\Rightarrow$  Sendeleistung erhöhen)

### CDMA - Beispiel zur Kodierung

Sender hat Spreizcode  $(1, 1, 1, -1, 1, -1, -1, -1)$  und versendet Daten  $d_1 = -1, d_0 = 1$ . Nach Multiplikation der Daten mit Spreizcode:

$Z_{1,m} = (-1, -1, -1, 1, -1, 1, 1, 1)$ ,  $Z_{0,m} = (1, 1, 1, -1, 1, -1, -1, -1)$ . Der Empfänger dekodiert folgendermaßen:  $\frac{\sum_{m=1}^M Z_{i,m} \cdot c_m}{M}$ .

Dabei ist  $M$  Länge des Spreizcodes (in diesem Beispiel 8),  $c_m$  Spreizfaktor an der Stelle  $m$ ,  $Z_{i,m}$  die gespreizten Daten, d.h:

$$d_1 = \frac{(-1) \cdot 1 + (-1) \cdot 1 + (-1) \cdot 1 + 1 \cdot (-1) + (-1) \cdot 1 + 1 \cdot (-1) + 1 \cdot (-1) + 1 \cdot (-1)}{8} = \frac{-8}{8} = -1$$

$$d_0 = \frac{1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + (-1) \cdot (-1) + 1 \cdot 1 + (-1) \cdot (-1) + (-1) \cdot (-1) + (-1) \cdot (-1)}{8} = \frac{8}{8} = 1$$

Bei mehreren Sendern multipliziert der Empfänger das überlagerte Signal mit dem jeweiligen Spreizcode, da diese orthogonal zueinander sind, kommen die richtigen Daten des jeweiligen Senders wieder raus.

### 1.1.2 Wireless Local Area Networks

#### Protokollstack:



**802.11:** '97, FHSS/DSSS, 1-2MBit/s, 2.4 GHz

**802.11b:** '99, DSSS, 1 - 11MBit/s, 2.4 GHz

**802.11n:** '09, OFDM/MIMO, 6 - 600MBit/s, 2.4 oder 5 GHz

**802.11ac:** '14, MU-MIMO, bis zu 6.93 GBit/s, 5 GHz

**802.11ay:** '19, 20 - 40 GBit/s, 60GHz

#### Begriffe:

- **Basic Service Set (BSS):** Stationen die auf dem gleichen Übertragungskanal Daten austauschen
- **Extended Service Set (ESS):** Zusammenschluss mehrerer BSS zu Kommunikationsnetz, Roaming zwischen den BSS
- **Service Service Set ID (SSID):** Name des Netzwerkes
- **Ad-Hoc Mode / Independent BSS (IBSS):** Alle Stationen gleichberechtigt, kein Access Point
- **Infrastructure BSS:** Geräte kommunizieren über AP (=Übergang zu drahtgebundenem Netz)

#### Kanäle:



- Bis zu 13 Kanäle mit je 5 MHz zwischen 2410 MHz - 2483 MHz.
- Bei **DSSS** Kanalbreite = 22MHz, bei **OFDM** = 20 MHz / 40 MHz (ohne / mit Kanalbündelung). Störungsfreier Betrieb nur bei passendem Abstand (5 Kanäle bei DSSS).

**Hierzu bitte auch das erste Übungsblatt vom Praktikum durchlesen!**

Accesspoint sendet regelmäßig **Beacon Frames** mit SSID und MAC-Adresse. Wireless Stations scannen Kanäle nach diesen Frames, wählen verfügbaren AP (Einstellungen & Signalstärke) aus, führt Authentifizierung durch und erhält anschließend IP-Adresse per DHCP.

**Passives Scannen**

PC ist passiv, hört Kanal ab

APs senden *Beacons*

PC sendet Association Request an ausgewählten AP

AP sendet Association Response an PC

**Aktives Scannen**

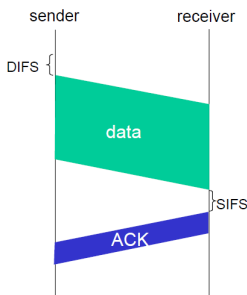
PC sendet Probe Request

APs senden Probe Response

PC sendet Association Request an ausgewählten AP

AP sendet Association Response an PC

Keine Kollisionserkennung (Collision Detection (CD)) möglich  $\Rightarrow$  ACKs auf Schicht 2

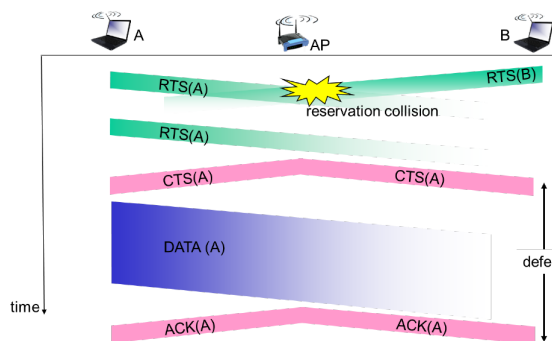


- Sender wartet Zeitspanne *DIFS* (Distributed Coordination Function Interframe Spacing) während der Medium frei sein muss
- Sender überträgt Daten (kein Collision Detection)
- Empfänger prüft CRC
- Empfänger sendet ACK falls CRC korrekt nach Wartezeit *SIFS* (Short Interframe Spacing) ( $SIFS < DIFS$ ) um Senden eines normalen Frames dazwischen zu verhindern außerdem Umschalten von *Empfangen* auf *Senden*

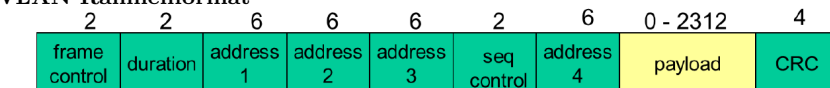
**Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA):**

- Sender lauscht (*Carrier Sense*) ob Medium frei (Dauer: **1 DIFS**)
- Falls frei: *random backoff* im aktuellen *Contention Window* würfeln wenn mehrere Stationen gleiches DIFS abgewartet haben und es sonst zu Kollision käme. Nach Ablauf des *random backoffs* Daten übertragen
- Falls nicht frei: Warten bis Kanal frei, wenn Kanal länger als 1 DIFS frei, dann *random backoff* verringern & anschließend Daten senden
- Falls kein ACK erfolgt: Contention Window verdoppeln und Daten erneut senden
- Empfänger sendet ACK nach Ablauf eines **SIFS** bei korrekt erhaltenen Daten

**Nachteile CSMA/CA:** Senden dauert lange, Kollision wird nicht erkannt - ist sehr zeitaufwändig und sollte daher vermieden werden  
 Besser eine Kollision bei kurzen Kontrollpaketen als bei langen Datenpaketen  $\Rightarrow$  **RTS/CTS**-Verfahren

**Ablauf:**

- Sender reserviert Kanal mit kurzem **Request-To-Send (RTS)**-Paket, Kollisionen möglich aber weniger schlimm da nur kleines Paket welches günstig erneut gesendet werden kann
  - Empfänger antwortet mit **Clear-To-Send (CTS)**
  - Sender sendet Datenpaket
- $\Rightarrow$  Andere Stationen empfangen RTS & CTS und berücksichtigen belegten Kanal  
 $\Rightarrow$  Funktioniert auch bei *Hidden Terminal* da CTS empfangen wird

**WLAN Rahmenformat**

Funktion	ToDS	FromDS	Add. 1	Add. 2	Add. 3	Add. 4
IBSS	0	0	destination	source	BSSID	unused
To AP	1	0	BSSID	source	destination	unused
From AP	0	1	destination	BSSID	source	Unused
WDS (bridge)	1	1	receiver	transmitter	destination	source

- Max. 2313 Bytes an Nutzdaten
- 3. Adresse erlaubt Umsetzung auf Ethernet-Rahmen
- WLAN-Reichweitenvergrößerung durch überlappende BSSs, IP-Adresse bleibt gleich da identisches Subnetz - nur AP ändert sich
- $\Rightarrow$  Switch ändert Port $\leftrightarrow$ IP-Zuordnung wenn sich Teilnehmer vom neuen AP meldet

Unterschied CSMA/CA $\leftrightarrow$ CSMA/CD: CSMA/CD bei Ethernet sendet *JAM*-Signal, CSMA/CA erkennt keine Kollision (versucht nur zu verhindern)

### 1.1.3 Personal Area Networks (PAN)

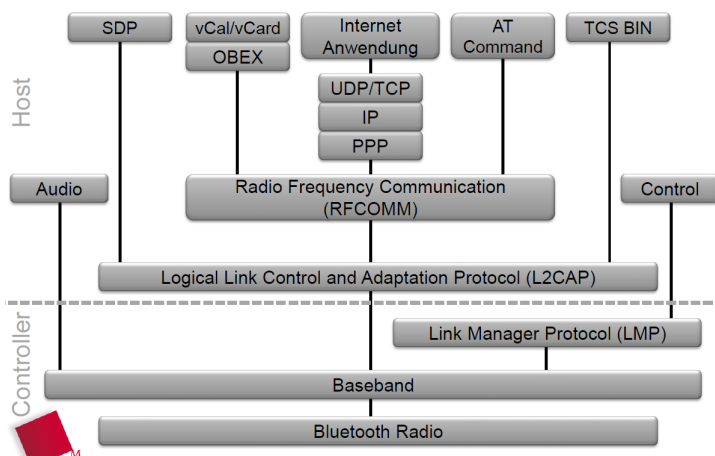
Drahtlos oder drahtgebundenes (Ad Hoc) Netzwerk von Kleingeräten, oft nur wenige Meter Reichweite.

#### Bluetooth

- Mehrfachzugriff mit TDMA, Frequenzsprungverfahren (Kanalwechsel nach jedem Zeitslot  $\Rightarrow$  Robustheit gegen Störer)
- Class 1: 100mW - 100m Reichweite, ◦ Class 2: 2.5mW - 10m Reichweite, ◦ Class 3: 1mW - 1m Reichweite
- '99: Einführung, 732,2kbit/s ◦ '04: v2.0 mit bis 2.1Mbit/s ◦ '16: v5.0 mit IoT Erweiterungen
- Ad Hoc Netzwerk (keine Infrastruktur nötig) ◦  $\leq 8$  aktive &  $\leq 255$  geparkte Geräte ◦ Master gibt Zeit vor, gewährt Slaves, aktiviert geparkte

#### Bluetooth Profile

Profil spezifiziert Anwendung von Bluetooth für bestimmten Zweck



- **AT Kommando:** Kommando zur Modemsteuerung
- **Baseband:** Basisband, Paketformate
- **L2CAP:** Logical Link Control and Adaptation Protocol: Bietet verbindungsorientierte- und lose Dienste zwischen Baseband und höheren Schichten
- **MCAP:** Multi-Channel Adaptation Protocol: Stellt Kontrollkanal *MCL* und Datenkanäle *MDL* bereit
- **RFComm:** Virtuelle, serielle Verbindungen, Emulation serieller Ports
- Geräte im HealthCare Bereich ursprünglich per RFCOMM angebunden  $\Rightarrow$  '08 Verabschiedung von standardisiertem *Health Device Profile*
- Zwei Rollen: **Source** = Datenquelle, **Sink** = Empfänger (Smartphone)
- Verbindungsauf- und abbau, Wiederaufbau abgebrochener Verbindungen

Sensoren (z.B. Pulsmesser, Thermometer) sollen lange Laufzeit ( $\Rightarrow$  geringer Stromverbrauch) aufweisen. Bluetooth 4.0 beinhaltet *Bluetooth Smart* (Low-Energy Profil auf Basis von einem *Generic Attribute Profile* (GATT)).

#### ZigBee

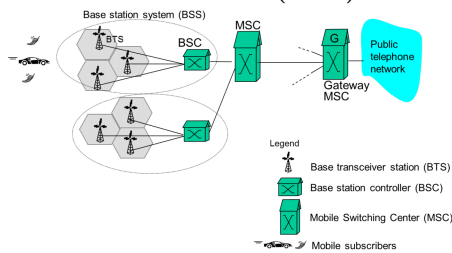
- Ziel: Drahtlose Übertragung bei geringem Stromverbrauch (geringe Datenraten: 20 - 250kbit/s, selten aktiv (*low duty-cycle*)).
- Übertragen von Sensordaten, Heim- und Gebäudeautomatisierung
- **Endgerät:** Reduced Function Device *RFD* - nur Teil des ZigBee Protokolls implementiert (geringere Kosten)
- **Router:** Full Function Device *FDD*, kann Daten weiterleiten ◦ **Koordinator:** Gibt zusätzliche Parameter vor, koordiniert das PAN

### 1.1.4 Zellulare Netzwerke

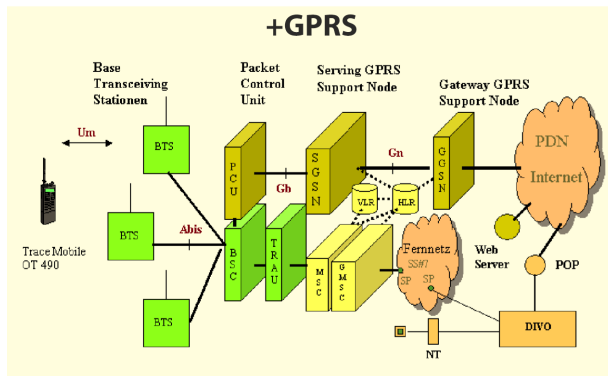
- **1G:** Analog (A/B/C-Netz) ◦ **2G:** GSM ab '92, 2.5G = GPRS, 2.75G = EDGE ◦ **3G:** UMTS ab '03 ◦ **4G:** ab '14 LTE ◦ **5G:** ab '21, Latent  $< 1\text{ms}$
- **Mobilfunkzelle:** Von Base Transceiver Station (BTS) abgedeckter Bereich ◦ **Air-Interface:** Untere 2 Netzwerkschichten Mobile Station  $\Leftrightarrow$  BTS

#### Ressourcenzuteilung in der Zelle:

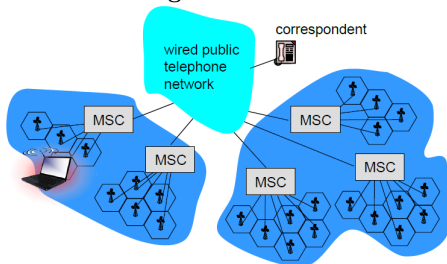
- **GSM:** Kombination aus FDMA & TDMA, Spektrum wird in einzelne Frequenzkanäle, jeder Kanal wiederum in Zeitschlitz aufgeteilt
- **UMTS:** CDMA Verfahren - Unterschiedlicher Code für unterschiedliche Nutzer

**2G Netzwerkkonstruktion (GSM):**

- **Base Station Controller (BSC):** übernimmt Ressourcenzuweisung und Mobilitätsmanagement in einem Base Station Subsystem (BSS)
- **Mobile Switching Center (MSC):** Anrufauf- und Abbau, Verbindung ins Festnetz, Mobilitätsmanagement
- **Gateway-MSC:** Vermittlungsfunktionen, Verbindung zu anderen Netzen bzw. Festnetz
- **PSTN:** Public Switched Telephone Network



- Mit GPRS-Komponenten erstmals paketvermittelte Datendienste
- verwendet bereits bestehende BTS mit
- **Packet Control Unit (PCU):** Kommuniziert über den BSC mit Endgerät und auch mit der SGSN, überwacht und verwaltet Datenpakete, Ressourcenverteilung
- **Serving GPRS Support Node (SGSN):** Übernimmt Vermittlung der Datenpakete und die Funktion des VLR
- **Gateway GPRS Support Node (GGSN):** Ist der Router, der das Mobilfunknetz mit dem Internet verbindet und die IP-Adresse zur Verfügung stellt
- **Home Location Register (HLR):** Datenbank mit Informationen zum Nutzer, enthält Rufnummer und zuletzt bekannten Aufenthaltsort
- **Visitor Location Register (VLR):** Datenbank mit Informationen zu allen Nutzern, die sich im vom MSC bedienten Bereich befinden
- Sprachdaten laufen über  $BTS \Leftrightarrow BSC \Leftrightarrow MSC \Leftrightarrow G - MSC \Leftrightarrow Festnetz$
- Paketdaten laufen über  $BTS \Leftrightarrow BSC \Leftrightarrow PCU \Leftrightarrow SGSN \Leftrightarrow GGSN \Leftrightarrow Internet$

**Mobilitätsmanagement:**

- Nutzer kann sich zwischen Zellen verschiedener MSCs, auch von anderen Providern, bewegen
- Mobilgerät prüft im eingeschalteten Zustand, ob sich aktuelle Location Area ändert
- ⇒ sendet Location Update mit neuer Area ⇒ Home Location Register (HLR) wird aktualisiert
- Eingehender Anruf: Befragung von HLR nach aktuellem Ort des Angerufenen über dessen temporäre *Roaming-Nummer*, anschließend Verbindungsaufbau über das VLR des MSC
- ⇒ Falls gerade keine Verbindung besteht: Broadcast-Nachricht über alle Basisstationen des jeweiligen MSC (*Paging*), Mobile meldet sich ggf., damit genaue Basisstation bekannt

**GSM Handover (= MSC/Inter-BSC Handover)**

Mobilgerät wechselt bei bestehender Verbindung von einer Basisstation zur anderen.

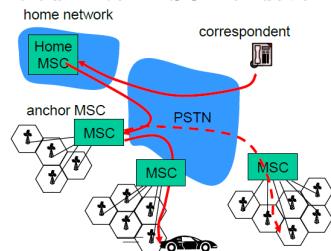
**Ursachen:** Bewegung des Nutzers (stärkeres Signal eines anderen BSS), aktuelle Zelle überlastet

**Ablauf Inter-BSC Handover:**

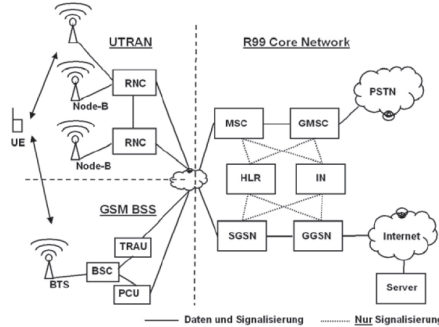
- Altes BSS informiert MSC über anstehendes Handover
- MSC reserviert Ressourcen zu neuer BSS
- Neue BSS reserviert Zeitslot (TDMA)
- Neue BSS signalisiert an MSC und alte BSS Bereitschaft zum Handover
- Alte BSS weist Mobilgerät an, Handover zu neuer BSS durchzuführen
- Mobilgerät aktiviert Kanal in neuer BSS
- Mobilgerät bestätigt Handover an MSC, diese leitet Daten um
- MSC weist alte BSS an, Ressourcen des Mobilgeräts freizugeben

**Arten von Handover:**

- **Intra BSC:** Aktuelle und neue Zelle gehörten zum selben BSC
- **Inter BSC:** Aktuelle und neue Zelle gehören zu unterschiedlichen BSC aber gleichen MSC
- **Inter MSC:** Aktuelle und neue Zelle gehören zu unterschiedlichen MSC
- **Subsequent Inter MSC:** Teilnehmer wechselt nach *Inter MSC* in Zelle eines dritten MSC
- **Subsequent Handback:** Teilnehmer wechselt nach *Inter MSC* zurück in Gebiet des ersten MSC

**Ablauf Inter-MSC Handover:**

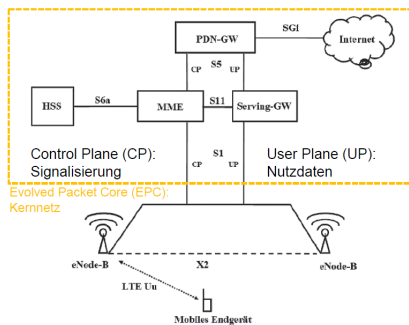
- Anker-MSC = erste MSC während eines Anrufs
- Daten bzw. der Anruf wird zunächst an Anker-MSC geleitet
- Dann Weiterleitung zu aktueller MSC

**3G Netzarchitektur:**

- Mit UMTS '99 neues Air-Interface *Universal Terrestrial Radio Access Network (UTRAN)*
- basierend auf Wideband-CDMA
- Ab UMTS v4 Umstellung auf IP basierte Kommunikation (Sprache & Daten)

**LTE & LTE Advanced (4G):**

Umstellung auf *Orthogonal Frequency Devision Multiplexing (OFDM)*: Übertragungstechnik mit flexibler Bandbreite zwischen 1.25 bis 20 MHz  
 LTE-Endgerät muss Mehrantennenverfahren unterstützen (*Multiple Input, Multiple Output MIMO*), LTE-Netz rein paketbasiert (Sprache per VoIP)

**1.1.5 Long Term Evolution (LTE, LTE-A)**

- **LTE + 2 Releases:** *LTE Advanced* - 3 Gbit/s DL bzw. 1.5 Gbit/s UL, Bündelung bis zu 5 Carrier
- **LTE + 3/4 Releases:** *LTE Advanced Pro* - Bündelung von bis zu 32 Carriern, QAM-256, LTE Narrow Band IoT, **5G** Standardisierung läuft
- **Packet Data Network Gateway:** Internetschnittstelle
- **Mobility Management Entity:** Mobilitätsmanagement
- **Serving-Gateway:** Weiterleitung von Nutzdaten ins Kernnetz
- **Home Subscriber Service:** entspricht HLR bei GSM
- **evolved Node-B (eNode-B):** Basisstation

**1.1.6 5G**

**Enhanced Mobile Broadband (eMBB):** Hohe Datenraten, **Massive Machine Type Communications (mMTC):** IoT Anwendungen

**Ultra-Reliable and Low-Latency Communication (uRLLC):** z.B. drahtlose Vernetzung in der Produktion

**5G New Radio:** Nutzung mehrerer Antennen (*MIMO*), mmWave-Bänder (24 - 30GHz), flexible und kürzere Slotzeiten (<1ms)

**Ziele:** E2E Latenz < 1ms, 1000x höheres Datenvolumen, 10-100x mehr Geräte, 10-100x mehr typische Nutzerdatenraten

**Umsetzung:** 5G Zellen mit LTE-Kernnetz (non-standalone), 2-3Gbit/s  $\Rightarrow$  5G mit 5G-Kernnetz (standalone), mmWave Bänder

**1.1.7 Zusammenfassung Drahtlose Netzwerke**

Ersetzen der unteren 2 Schichten (Link- & Physical-Layer) durch drahtlose Varianten  $\Rightarrow$  Auswirkungen auf obere Schichten minimal, aber:

$\Rightarrow$  Fehlinterpretation von Paketverlusten auf drahtlosem Link von TCP führt zu Congestion Window Verringerung  $\Rightarrow$  Datenrate sinkt

$\Rightarrow$  Verzögerung durch Link-Layer Retransmission (Auswirkungen auf Echtzeitanwendungen), Drahtloser Link meist geringere Datenrate

## Security

### 1.2.1 Grundlagen

- **Vertraulichkeit:** Nur Sender und rechtmäßige Empfänger sollen die Nachricht verstehen können
- **Integrität:** Sicherstellen, dass Nachricht unverändert ist (ob durch Übertragungsfehler oder Angriff)
- **Authentisierung:** Nachweis, dass eine Person diejenige ist, die sie vorgibt zu sein
- **Authentifizierung:** Sicherstellen, dass Kommunikationspartner derjenige ist, für den er sich ausgibt (Prüfung der *Authentisierung*)
- **Authorisierung:** Nachweis von speziellen Rechten
- **Betriebssicherheit:** Absicherung des Firmennetzes gegen Eingriffe von außen

### 1.2.2 Grundlagen der Kryptographie

#### Arten von Verschlüsselung

$K_A$ : Schlüssel für Verschlüsselung,  $K_B$ : Schlüssel für Entschlüsselung,  $m$ : Klartext,  $K_A(m)$ : Ciphertext - Klartext verschlüsselt mit  $K_A$   
 $m = K_B(K_A(m))$

- **Symmetrisch:**  $K_A$  identisch zu  $K_B$ , Verfahren z.B. *AES*
- **Public Key:** Paar unterschiedlicher Schlüssel:  $K_A = K_B^+$  und  $K_B^-$ , ein Schlüssel ist beiden bekannt (*public key* -  $K_B^+$ ), der andere nur Empfänger (*private key* -  $K_B^-$ ), Verfahren z.B. *RSA*

#### Arten von Angriffen

- **Cipher-Text only:** Angreifer hat nur Geheimtext, entweder alle möglichen Schlüssel ausprobieren (*brute force*) oder statistische Analyse
- **Known-Plaintext:** Angreifer kennt Klar- und zugehörigen Geheimtext, kann Rückschlüsse auf Schlüssel ziehen
- **Chosen-Plaintext:** Angreifer kann Geheimtext zu selbstgewählten Klartext bekommen, Verschlüsselungsalgorithmus ggf. ausnutzbar

#### Einfache symmetrische Verschlüsselungen

- **Cesar-Chiffre:** Verschiebung des Alphabets als Schlüssel, Abbildung des Klartextes auf verschobenes Alphabet ergibt Ciphertext
- **Substitutions-Chiffre:** Schlüssel ist eine Abbildungsvorschrift (Buchstabe  $b$  wird abgebildet auf  $b'$ )
- **Blockchiffre:** Verarbeitung des Klartextes in  $k$ -Bit großen Blöcken (d.h. Abbildung eines  $k$ -Bit Blocks auf  $k$ -Bit Geheimtext)

z.B.  $k = 3$ :  $000 \Rightarrow 110$ ,  $001 \Rightarrow 111$ ,  $010 \Rightarrow 101$ ,  $011 \Rightarrow 100$ ,  $100 \Rightarrow 011$ ,  $101 \Rightarrow 010$ ,  $110 \Rightarrow 000$ ,  $111 \Rightarrow 001$

Der Schlüssel ist die Abbildungstabelle, Anzahl möglicher Abbildungen:  $(2^k)!$ , d.h. für  $k = 3$  gibt es  $8!$ , also 40320, Abbildungsmöglichkeiten  
 $\Rightarrow$  Heutige Blockchiffren (*DES*, *3DES*, *AES*) verwenden Funktionen um Abbildungen zu erzeugen da bereits kleine  $k$  riesige Tabellen erzeugen

#### Cipher Block Chaining

- Identischer Klartext produziert identischen Geheimtext  
 $\Rightarrow$  Rückschlüsse auf Schlüssel möglich, daher bitweise *XOR*-Operation mit zufälligem Bitmuster auf Klartext
- Identischer Klartext erzeugt nun anderen Geheimtext, Empfänger benötigt zum Entschlüsseln das zufällige Bitmuster
- Um nicht doppelt so viele Daten (Geheimtext + Zufallsmuster) versenden zu müssen wird *Cipher Block Chaining* angewandt  
 $\Rightarrow$  Nur das erste zufällige Bitmuster (*Initialization Vector VI*) wird unverschlüsselt an Empfänger gesendet  
 $\Rightarrow$  Danach ist vorhergehender Geheimtext das zufällige Bitmuster für den nächsten Block Klartext
- Für  $c$  = Ausgegebener Geheimtext (*Cipher*)  $K$  = Schlüssel,  $m$  = Klartext ist der Verlauf dann wie folgt:  
 $c(0)$  = Initialisierungsvektor  
 $c(1) = K(m_1 \text{ XOR } c(0))$   
 $c(2) = K(m_2 \text{ XOR } c(1))$

#### Data Encryption Standard (DES):

56-Bit symmetrischer Schlüssel, Verarbeitung von 64-Bit Blöcken mit *Cipher Block Chaining*, per *Brute Force* knackbar, *3DES* etwas sicherer

#### Advanced Encryption Standard (AES):

Nachfolger von *DES*, 128/192 oder 256-Bit symmetrischer Schlüssel mit 128-Bit Blöcken, AES-256 kann nicht geknackt werden

#### Public Key (asymmetrische) Verschlüsselung:

- Öffentlicher Schlüssel  $K_B^+$  wird zur Verschlüsselung verwendet, Privater Schlüssel  $K_B^-$  zur Entschlüsselung
- Öffentlicher Schlüssel sowohl Empfänger als auch Sender bekannt, privater Schlüssel ist aber nur Empfänger bekannt
- Sender überträgt  $K_B^+(m)$ , Empfänger entschlüsselt mit  $K_B^-(K_B^+(m))$ , d.h. es muss gelten  $K_B^-(K_B^+(m)) = m$
- Man darf nicht vom öffentlichen auf den privaten Schlüssel schließen können

**RSA Algorithmus:**

- Text wird als Bitmuster angesehen, die Verschlüsselung ergibt wieder ein Bitmuster - den Geheimtext
- RSA nutzt modulare Arithmetik und die Tatsache, dass Teilerbestimmung einer gegebenen Zahl sehr rechenaufwändig sind
- Verschlüsselung mit public key ist allerdings auch sehr rechenaufwändig, da z.B.  $m^e$  berechnet wird ( $e$  = öffentlicher Schlüssel)
- Symmetrische Verschlüsselung wesentlich schneller  $\Rightarrow$  Asymmetrische Verschlüsselung zum Aufbau einer sicheren Verbindung
  - $\rightarrow$  danach mit einem zweiten Schlüssel symmetrische Verschlüsselungen austauschen
  - $\rightarrow$  Teilnehmer A & B verwenden RSA um symmetrischen Schlüssel  $K_S$  auszutauschen, danach symmetrische Verschlüsselung (*AES*) mit  $K_S$

**Integrität einer Nachricht sicherstellen**

- Berechnen einer Prüfsumme (*Hash*) über Nachricht + gemeinsames Geheimnis (ansonsten könnte Angreifer die Nachricht verändern und dann erneut eine gültige Prüfsumme über die veränderte Nachricht berechnen), Empfänger berechnet Hash selbst und überprüft ihn mit übertragenem Hash

**Kryptographische Hashfunktionen**

- Berechnung eines Strings  $H(m)$  fester Größe aus Nachricht  $m$ , vom Rechenaufwand her nicht möglich, Kollision zu erzeugen, so dass  $H(x) = H(y)$
- MD5 mit 128-Bit Hash (unsicher) ◦ SHA-1 mit 160-Bit Hash (inzwischen auch kritisch) ◦ SHA-2 mit 224/256/384/512-Bit Hash (empfohlen)

**Message Authentication Code (MAC)**

- Gemeinsames Geheimnis besteht aus *Authentication Code*  $s$  und Nachricht  $m$ , Sender hängt  $H(m + s)$  (=MAC) an Nachricht  $m$  an
- Übertragung von Nachricht + MAC an Empfänger, dieser berechnet ebenfalls Hash aus  $m + s$  und prüft, ob gesendeter MAC passt

**Digitale Signaturen**

- Unterschrift als Bestätigung der Urheberschaft eines Dokuments, MAC mit *shared key*  $s$  ungeeignet, da  $s$  sowohl Unterzeichner als auch Prüfer bekannt sein muss  $\Rightarrow$  Prüfer kann Unterschrift fälschen

**Lösung:**

Berechne  $H(m)$ ,  $m$  := zu unterschreibende Nachricht

Unterzeichner nutzt private key zur Berechnung von  $K_B^-(H(m))$ , wird zusammen mit  $m$  versendet

Prüfer prüft ob  $H(m) = K_B^+(K_B^-(H(m)))$  (entschlüsseln mit öffentlichem Schlüssel), falls ja: Unterschrift gültig

**Zertifizierung von öffentlichen Schlüsseln**

- Angreifer kann behaupten, dass dessen public key gleich dem vom Unterzeichner ist  $\Rightarrow$  Öffentlicher Schlüssel muss Person zugeordnet werden
- *Certification Authority* (**CA**) prüft Identität (z.B. Personalausweis) - erstellt Zertifikat, dass public key zur Person gehört (Angabe der Domain) und unterschreibt das Zertifikat per digitale Signatur
- Unterzeichner sendet eigenen public key und Zertifikat an Empfänger, dieser prüft mit öffentlichem **CA**-Schlüssel, ob Zertifikat gültig (d.h. public key richtig) ist

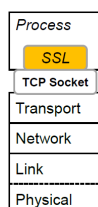
**Authentifizierungsprotokoll mit Shared Secret**

- Kommunikationsteilnehmer prüft, ob ein anderer derjenige ist, für den er sich ausgibt
- IP-Adresse kann gefälscht sein oder Angreifer führt Playback-Angriff durch (Kommunikation aufzeichnen & wieder abspielen)

Lösungen: *shared secret* oder Verwendung einer nur einmal verwendeten, zufälligen Zahl (*Nonce*) die mit symmetrischem Schlüssel verschlüsselt wird

- Sender sendet eigenen Namen an Empfänger
- Empfänger sendet Nonce  $R$  zurück
- Sender sendet mit eigenem privaten Schlüssel verschlüsseltes  $R$
- Empfänger fordert public key an
- Sender sendet public key (ggf. mit Zertifikat einer CA) an Empfänger
- Empfänger prüft ob  $K_A^+(K_A^-(R)) = R$

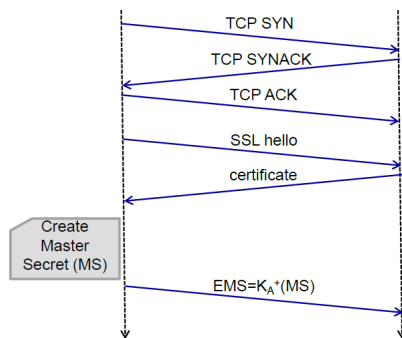
## 1.2.3 Secure Sockets Layer (SSL)



- Viele TCP-basierte Anwendungen benötigen **Vertraulichkeit, Integrität, Authentifizierung**
- Keines der Internet-Transportprotokolle unterstützt Verschlüsselung übertragener Informationen
- Auf Anwendungsschicht implementiert (z.B. *java.net.ssl* oder *OpenSSL*)
- SSL stellt diese Funktionalitäten durch Verschlüsselung, Message Authentication Code und mit Zertifikaten zur Verfügung
- SSL v3.1 von *IETF* als **Transport Layer Security TLS 1.0** standardisiert
- Drei Phasen: Handshake, Key Derivation (Herleitung von Schlüsseln), Data Transfer (eigentliche Datenübertragung)



**SSL Handshake:** Webbrowser als Client, Server besitzt public & private key + Zertifikat für public key zur Identitätsbestätigung (Domainname)



- TCP-Verbindung aufbauen
- Client sendet Liste an unterstützten *SSL Cipher Suites* + Nonce
- Server sendet Wahl der *Cipher Suite*, CA-Zertifikat und Server-Nonce
- Client:
  - prüft Server-Zertifikat
  - generiert *Pre-Master Secret (PMS)*
  - sendet mit Public Key des Servers verschlüsseltes PMS an den Server
- Client & Server leiten mit *KDF* aus PMS und Nonces das *Master Secret (MS)* her
  - Empfänger verschlüsselt *MS* mit public key des Servers = **EMS**
- Aus *MS* werden die Schlüssel  $E_A, E_B, M_A, M_B$  hergeleitet
  - ⇒ **Nun alle Nachrichten verschlüsselt + MAC authentifiziert**
- Client sendet *MAC* aller Handshake-Nachrichten, die vom Server überprüft werden
- Server sendet *MAC* aller Handshake-Nachrichten, die vom Client überprüft werden
  - ⇒ **Verhinderung von z.B. Wahl einer schwächeren Cipher Suite**

### SSL Key Derivation

- Statt *Master Secret* direkt als Schlüssel zu verwenden werden aus Sicherheitsgründen vier unterschiedliche Schlüssel aus einer *Key Derivation Function KDF* mit Eingabeparametern Master Secret und Zufallsdaten, abgeleitet (nur für eine Sitzung gültig):

$E_B$ : Verschlüsselung Client → Server

$M_B$ : MAC zur Integritätsprüfung der Daten von Client → Server

$E_A$ : Verschlüsselung Server → Client

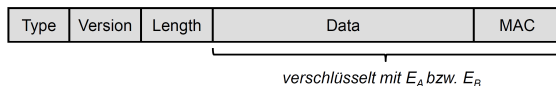
$M_A$ : MAC zur Integritätsprüfung der Daten von Server → Client

### SSL Datenübertragung



- Aufteilung des TCP-Byte-Stroms in separate Abschnitte (*SSL Records*)
- Jeder Record enthält zusätzliche Sequenznummer (beginnend bei 0) um Reordering von Records verhindern
- *MAC* wird berechnet über Daten + MAC-Schlüssel ( $M_A$  oder  $M_B$ ) + Sequenznummer
- Nutzung von *Nonces* verhindert Replay-Angriff (Wiederabspielen einer Datenverbindung)

### SSL Record



- **Type:** Unterscheidung von Handshake/Nutzdaten/Beenden
- **Version:** SSL-Versionsnummer
- **Length:** Länge der Daten (ohne Header) - Extraktion aus dem TCP-Byte Strom

### SSL Cipher Suites

- Enthält Algorithmus für: *Public Key* Verschlüsselung, *symmetrische* Verschlüsselung und für *Message Authentication Code*
- Client bietet Reihe von *SSL Cipher Suites* an (vom Client favorisierte ganz oben), Server sucht eine aus (unabhängig von Clientpriorität)

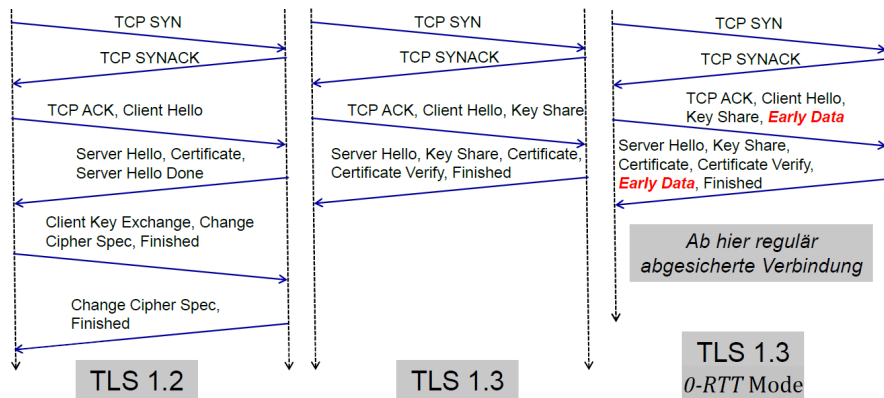
Symmetrisch	Public Key	MAC
DES, 3DES, AES	RSA, DH (Diffie-Hellmann), ECDH (Elliptic-Curve DH)	SHA-1, SHA-2, MD5

### Vorzeitiges Schließen einer SSL-Verbindung

- Wird durch zusätzlichen *SSL-Record* zum ordnungsgemäßen Beenden verhindert (Authentifizierung erfolgt über *MAC*)

### 1.2.4 TLS v1.3

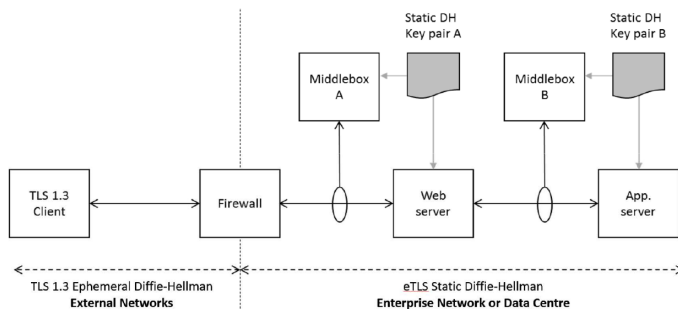
- Ciphersuite-Konzept geändert: MAC berechnen & Verschlüsselung erfolgt nun in einem Schritt
- Verbindungsaufbau (*full handshake*) von 2 RTT auf 1 RTT verringert, *0-RTT*-Mode: Daten bereits in erster Nachricht enthalten
- Verhindern von Downgrades durch Signierung der Liste an Verschlüsselungsverfahren, unsichere kryptographische Verfahren entfallen
- **Perfect Forward Secrecy:** Kommunikationspartner generieren nur für diese Verbindung gültige Kurzzeitschlüssel, die nach Verbindungsende verworfen werden und nicht aus Langzeitgeheimnis rekonstruiert werden können



#### 0-RTT Mode:

Client & Server teilen *Pre-shared Key* z.B. aus vorheriger Verbindung, und können diesen gleich zur Authentifizierung und Verschlüsselung der *Early Data* verwenden  
**allerdings:** Kein *Forward Secrecy*, anfälliger ggü. Replay-Attacken

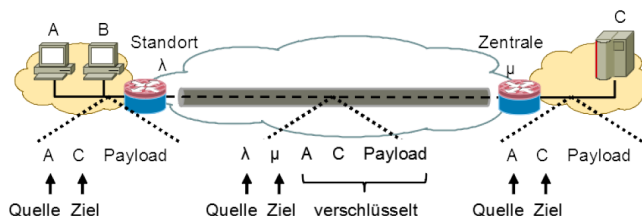
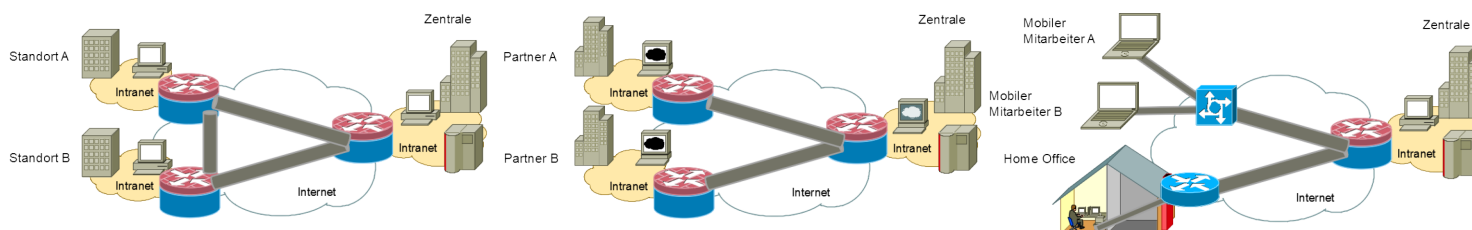
### 1.2.5 Enterprise Transport Security (ETS)



- Unternehmensnetze setzen viele Middleboxes zur Angriffserkennung (*Intrusion Detection*) oder als Application Layer Firewall ein, Datenverkehr kann aufgrund von *Perfect Forward Secrecy* nicht entschlüsselt werden → keine Überwachung durch Middleboxen möglich  
 ⇒ Verzicht auf *Perfect Forward Secrecy*, TLS 1.3 wird um statisches Diffie-Hellman Schlüsselpaar zwischen Middlebox und Unternehmensserver erweitert

### 1.2.6 Virtual Private Networks (VPN)

- Stellt über ein öffentliches, ggf. unsicheres, Netzwerk eine gesicherte, private Vernetzung zwischen Endgeräten her
- Zur Anbindung von auswärtigen Standorten, externen Partnern oder Mitarbeitern (*Remote Access*)
- Implementierung auf verschiedenen Schichten möglich (Transportschicht: *Cisco-Client*, Netzwer-Schicht (IP): *IPSec*)



- Tunneling:** Weiterleitung beliebiger Datenpakete über unsicheres Transitnetz
- VPN nutzt eigenen Adressraum
- $A, B, C$  := VPN-Adressraum,  $\lambda, \mu$  := Internet-Adressraum
- Authentisierung:**  
*Password Authentication Protocol PAP* - unsicher da Klartextaustausch  
*Challenge Handshake Authentication Protocol CHAP* - Challenge-Response
- Software:**  
 Server (VPN-Gateway) per VPN-Server Software  
 Client per VPN-Client (Cisco) / VPN-Adapter
- Protokolle:**  
*Point-to-Point Tunnelling Protocol PPTP*  
*Layer Two Tunnelling Protocol L2TP* (*PPTP*-Nachfolger)  
 - Tunnelt beliebige Pakete über UDP

### 1.2.7 IPSec

o Dominierendes Protokoll für VPNs auf Netzwerkschicht, verschiedene Verfahren:

- **Authentication Header (AH)**

- Garantiert Integrität der übertragenen Daten, Authentifizierung der Quelle eines Paketes (**keine** Verschlüsselung)

**Header:**

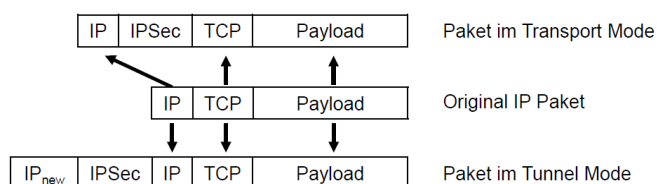
- *Integrity Check Value (ICV)*: HMAC (Hash über Nutzdaten + Geheimnis) über IP-Paket
- *Security Parameter Index (SPI)*: Index auf eine *Security Association* (kommt später)
- *Sequence Number (SN)*: Steigende Nummer zum Schutz vor *Replay*-Attacken

- **Encapsulated Security Payload ESP**

- Vertraulichkeit der Daten durch symmetrische Verschlüsselung + zusätzliche Authentifizierung der Quelle eines Paketes

**Header:**

- Ebenfalls *ICV* (hier wird äußerer IP-Header **nicht** mit einbezogen), *SPI* und *SN*
- *Initialization Vector (IV)*: Initialisierungsvektor für symmetrische Verschlüsselungsverfahren
- *Encrypted Payload (Encrypted Payload)*: Verschlüsselte Nutzdaten



o **Transport Mode:** Einfügen des *IPSec*-Headers in original IP-Paket

⇒ Gesicherte IP-Verbindung zwischen zwei Hosts (*End-to-End*)

o **Tunnel Mode:** Kapselung des Original IP-Paketes (inkl. Header) in neues IP-Paket mit *IPSec*-Header

⇒ Gesicherte Verbindung zwischen Host/Netz und Netz/Netz

⇒ Im neuen IP-Header neue Src- bzw. Dest-Adressen (die Tunnelendpunkte)

**Security Association (SA):**

- (unidirektionale) Festlegung der für *IPSec* Kommunikation zwischen zwei *IPSec*-Hosts verwendeten Parameter: Protokoll, Modus, Verschlüsselung
- legen fest, **was** in **welcher Art** verschlüsselt wird
- Jede *SA* hat eindeutigen *Security Parameter Index (SPI)*
- *IPSec* Einheit erhält die *SPIs* in *Security Association Database (SPD)*

**Problem mit Network Address Translation (NAT):**

o *NAT* ändert Port und IP-Adresse (klappt nicht, wenn die im *Tunnel-Mode* verschlüsselt sind)

⇒ *IPSec NAT Traversal*: UDP Datagram zum Übertragen der *IPSec* Pakete

**Internet Key Exchange (IKE):**

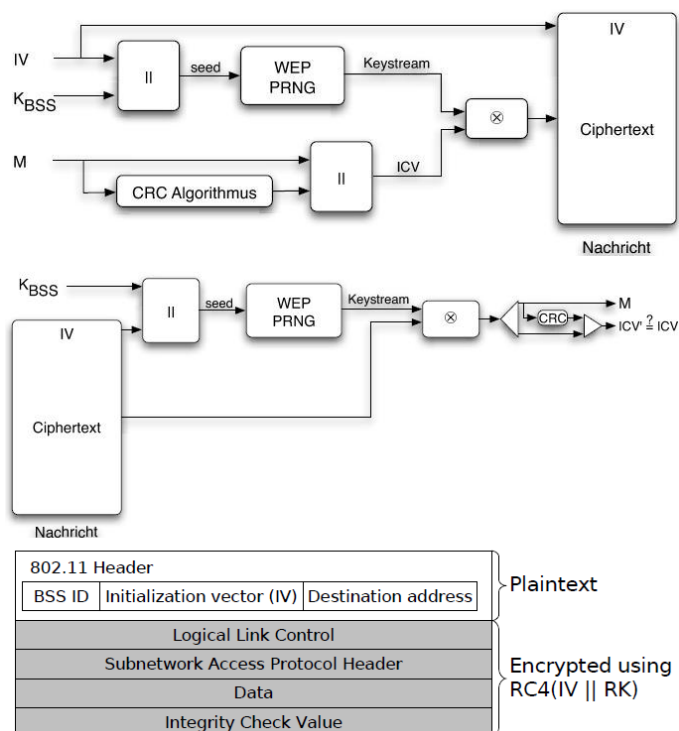
- Automatisiertes Aushandeln der *SAs* (mit Schlüssel)
- Basiert auf *Security Association Key Management Protocol (ISAKMP)*
- Variante 1: *Pre-Shared Key (PSK)*
  - Beide Seiten sind mit gemeinsamen Geheimnis vorkonfiguriert
  - *IKE* wird genutzt um *SAs* aufzusetzen
- Variante 2: *Public Key Infrastruktur (PKI)*
  - Beiden Seiten haben Public- und Private Key + **Zertifikat**
  - *IKE* zur Authentifizierung und Aufsetzen der *SAs* (ähnlich *SSL Handshake*)
- Phase 1: Aufbau einer bi-direktionalen *IKE SA* (**keine** *IPSec SA*)
  - Verwendet *ISAKMP*, entweder im *Aggressive Mode* (1.5 RTT) oder *Main Mode* (3 RTT)
- Phase 2: Aushandlung von *IPSec SA* (ebenfalls mit *ISAKMP*), Teilnehmer signieren ihre Nachrichten

### 1.2.8 Alternative VPN Protokolle

- o **OpenVPN:** per *OpenSSL* verschlüsselte *TLS*-Verbindung (TCP/UDP), *Bridge-Mode*: Layer 2 Tunnel, *Routing-Mode*: IP-Tunnel
- o **Port Forwarding / Tunnelling mit SSH:** Tunneln von TCP-Verbindungen über *SSH* (Anwendungsschicht) - kein echtes *VPN*

## 1.2.9 Sichere WLANs

## 802.11: Wired Equivalent Privacy (WEP)



- Schlüsselverteilung nicht Teil von WEP, Aufgaben lediglich Authentifizierung und Verschlüsselung (mit Stromchiffre RC4)
- **Schlüssel:**
  - 24-Bit Initialisierungsvektor (IV), wechselt von Rahmen zu Rahmen, wird im Klartext übertragen
  - 40-Bit (später 104-Bit) symmetrischer Schlüssel (Root Key)
- $m(i) :=$  Nachricht-Byte;  $ks(i) :=$  Schlüsselstrom-Byte;  $c(i) :=$  Cipher-Byte
- Sender:  $c(i) = ks(i) \text{ XOR } m(i)$ ; Empfänger:  $m(i) = ks(i) \text{ XOR } c(i)$
- Integrity Check Value (ICV):
  - CRC32 über zu verschlüsselnde Daten
  - Root Key und IV zur Verschlüsselung von Daten und ICV
  - schützt vor zufälligen Fehlern, nicht vor Angreifer (kein geeigneter MAC)
- **Authentifizierung:**
  - **Open System:** Alle Authentifizierungsnachrichten w/o Prüfung zulassen  
⇒ Client kann ohne Schlüssel keine korrekten Nachrichten senden
  - **Shared Key:** Client sendet Request an AccessPoint  
AP sendet 128-Byte Nonce, Client verschlüsselt Nonce mit shared key  
AP entschlüsselt Ciphertext mit IV und shared key (Ergebnis = Nonce)

**Probleme mit Shared Key Authentication bei WEP:**

- Angreifer kann IV, Nonce und Antwort beobachten, daraus kann er gültigen Schlüsselstrom und IV erzeugen:  
 $\text{Nonce XOR (Nonce XOR keystream)} = \text{keystream} \Rightarrow$  Angreifer kann gültige Authentifizierungsnachrichten für beliebige Nonce erstellen  
⇒ **Unsicherer als Open System**
- **Problem 1:** Nur  $2^{24}$  verschiedene Schlüssel (IV-Länge)  $\Rightarrow$  IV wird aufgrund heutiger Datenraten in weniger als einem Tag erneut verwendet  
Angreifer kann selbst Datenverkehr generieren (z.B. ARP-Requests durch mitschneiden da ARP bekannte Länge bzw. Plaintext fix)
- **Problem 2:** IV wird im Klartext übertragen  $\Rightarrow$  Angreifer kann Wiederholung leicht erkennen

**802.11i: Temporal Key Integrity Protocol (TKIP)**

- Ersatz für WEP, aber mit selber Hardware funktionsfähig, IV erweitert auf 48-Bit, neue Hash-Funktion, Personal-Mode mit pre-shared key

**802.11i: Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP)**

- Verwendet AES mit 128-Bit Schlüssel und 48Bit IV (anstatt RC4)
- Teil des WPA2-Verfahrens, WPA2 würde alternativ auch TKIP erlauben (**empfohlen ist allerdings WPA2 + CCMP**)

**802.11i: WPA3**

- 128-Bit (personal mode) bzw. 192-Bit (enterprise mode) + Forward Secrecy

**Extensible Authentication Protocol (EAP)**

- Protokoll für sichere Ende-zu-Ende Authentifizierung zwischen mobilen Clients und Authentifizierungsserver über einen Access Point
- EAP-TLS: Nutzung von Zertifikaten; EAP-PEAP: Verschlüsselter Tunnel für Authentifizierung

**802.1X**

- Erweitert 802.11 durch EAP um Sicherheitsfunktionen wie: Schlüsselmanagement, Nutzeridentifikation, Tokens...

## 1.2.10 DNS Security Extensions (DNSSEC)

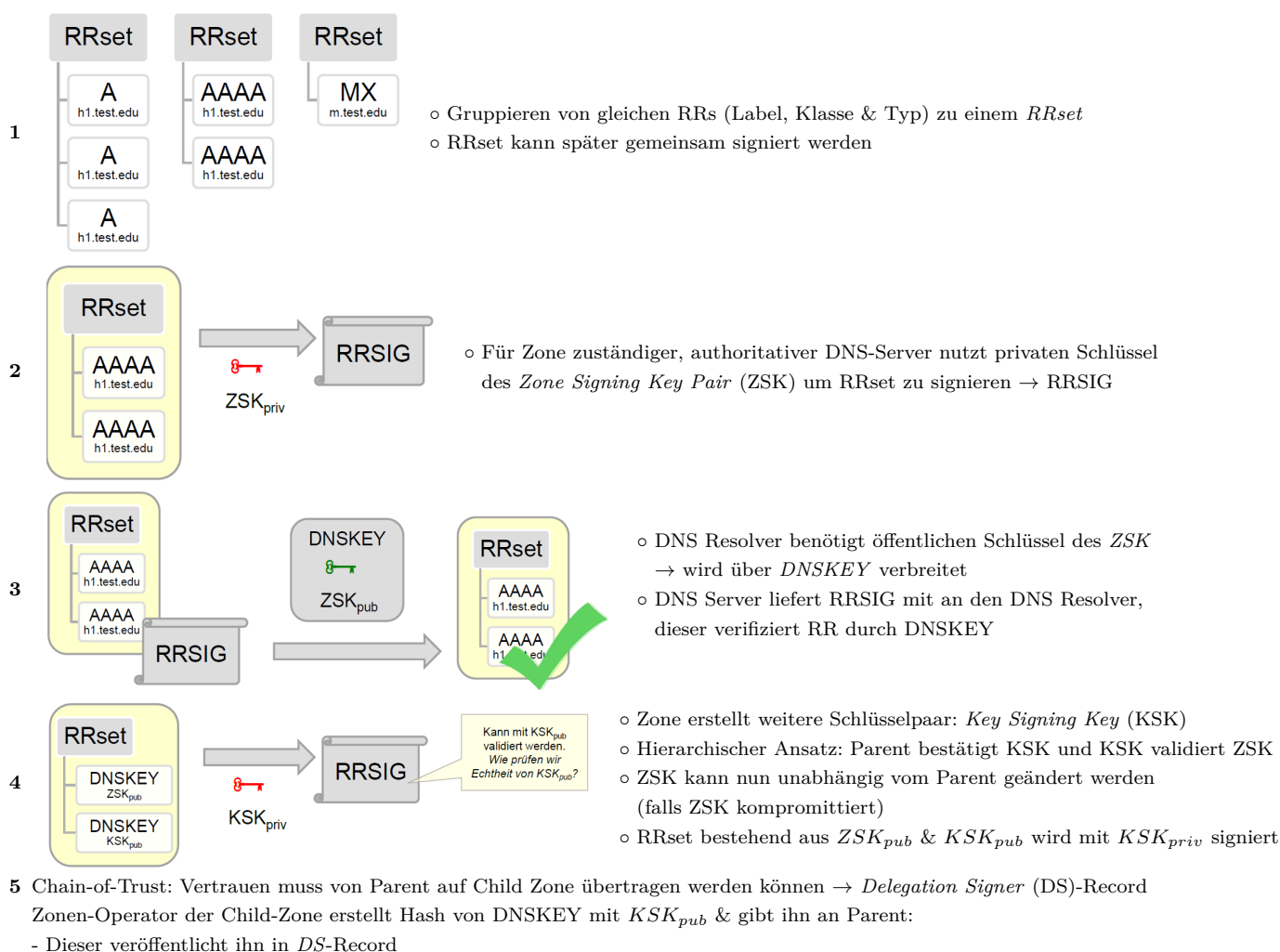
o Absicherung von Authentizität und Integrität der DNS-Informationen (**nicht** Vertraulichkeit)

⇒ Verwendung von Signaturen, umgesetzt mit asymmetrischen Kryptosystem (public & private Key):

- *Resource Record* (RR) wird mit geheimen Schlüssel unterschrieben
- DNS Client prüft mit öffentlichem Schlüssel die Authentizität und Integrität des empfangenen *RRs*
- *Chain-of-Trust*: Höhere Zone im DNS-Baum bestätigt durch Signierung die öffentlichen Schlüssel der unteren Zonen,  
→ der Anfragende muss nur obersten public key kennen (*Key Signing Key*)

o Realisierung über neue Resource Record Typen:

RR Typ	Erläuterung	
RRSIG	Digitale Signatur eines <i>Resource Record Sets</i> (RRset)	
DNSKEY	Public Signing Key, wird vom Resolver zur Prüfung digitaler Signaturen verwendet	
DS	Delegation Signer: Hash des <i>DNSKEY</i> um dessen Authentizität sicherzustellen, delegiert Vertrauen eine Ebene weiter in Chain-of-Trust	
CNSKEY, CDS	Child Zone informiert Parent: Key-Signing-Key ändern, DNSSEC an/aus	



## 1.2.11 DNS over TLS

o Absicherung von Vertraulichkeit

1. DNS Client (DNS Stub Resolver) baut TCP-Verbindung zu Resolver auf
2. TLS Handshake wird durchgeführt
  - Resolver sendet TLS Zertifikat
  - DNS Stub Resolver prüft Zertifikat mit Hash des öffentlichen Schlüssels des Resolvers

3. Ab jetzt sendet DNS Stub Resolver alle DNS Anfragen über mit TLS geschützte TCP-Verbindung
  - Übertragung einer Nachricht: 2-Byte Längenangabe + Nachricht  
beim Zusammensetzen muss man wissen, wie lange die ursprüngliche Nachricht war, wenn sie in einzelne TCP-Pakete zerstückelt wurde
4. Verbindung bleibt offen bis DNS Stub Resolver sie schließt
5. Umsetzung: Als Betriebssystem-Dienst (z.B. *systemd-resolved* unter Linux) oder lokaler Forwarder der als localhost-DNS-Server im Betriebssystem eingetragen wird und Anfragen über TLS weiterleitet

### 1.2.12 DNS over HTTPS

#### Anfrage über HTTP Methode GET

```
GET /dns-query?dns=AAABAAABAAAAAAAAA3d3dwd1e6FtcGx1A2NvbQAAQAB HTTP/2
Accept: application/dns-message
```

#### Anfrage über HTTP Methode POST

```
POST /dns-query HTTP/2
Accept: application/dns-message
Content-Type: application/dns-message
Content-Length: 33
```

```
<33 bytes represented by the following hex encoding>
00 00 01 00 00 01 00 00 00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 00 01 00
01
```

base64 kodierte DNS Query

DNS Query (33 Byte)

- Absicherung von Vertraulichkeit
- Probleme: Genutzter DNS-Server ist Anwendungsspezifisch (z.B. Browser), firmeneigene DNS-Server werden umgangen
- Internet Web-/Maidienste funktionieren nicht mehr (werden nur vom internen DNS aufgelöst)
- DNS-Filter gegen Malware unwirksam
- Nur wenige Firmen (Cloudflare, Google) bieten DNS-over-HTTPS  
→ Konzentration auf wenige Anbieter, Privatsphäre Gefährdung
- Viel Overhead (Anwendungsschicht: DNS, HTTP, TLS  
Transportschicht: TCP statt UDP)

DNS-over-TLS (DoT)	DNS-over-HTTPS (DoH)
DNS Nachrichten direkt über TLS-gesicherte TCP Verbindung ausgetauscht: Overhead gering (2 Byte Längenangabe)	DNS Nachrichten über HTTP Methode GET oder POST geschickt, höherer Overhead durch HTTP Nachricht und Header
DNS (Stub) Resolver vom Betriebssystem → durch Admin konfiguriert	DNS Resolver in Anwendung (Browser) konfiguriert – sofern kein Discovery-Ansatz verwendet wird
Üblicherweise TCP Verbindung über Port 853 – leicht zu in Firewall zu blockieren	Nutzung von Port 443 (wie reguläre https Anfrage) – schwer in Firewall zu blockieren