Anatol **BADACH**
Erwin **HOFFMANN**

Technik der IP-NETZE

5. Auflage

GRUNDLAGEN DER IPv4- UND IPv6-KOMMUNIKATION

Mit über 700 Abbildungen

HANSER

Badach/Hoffmann

Technik der IP-Netze

Anatol Badach
Erwin Hoffmann

Technik der IP-Netze

Grundlagen der IPv4- und IPv6-Kommunikation

5., überarbeitete Auflage

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Jürgen Dubau, Freiburg/Elbe
Layout: Erwin Hoffmann mit LaTeX
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Max Kostopoulos
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

Print-ISBN: 978-3-446-47371-3

E-Book-ISBN: 978-3-446-47426-0

Inhaltsverzeichnis

Vorwort

Begriff: IP-Netze

Das Internet ist inzwischen zum unabdingbaren Kommunikationsmedium geworden, über das jeder zu jeder Zeit Information über fast alles abrufen sowie Nachrichten senden und empfangen kann. Unsere heutige Gesellschaft kann man sich ohne Internet kaum noch vorstellen. Voraussetzung zur Kommunikation zwischen Rechnern sind bestimmte Regeln, die vor allem die Datenformate und die Prinzipien der Datenübermittlung festlegen. Diese Regeln werden als Kommunikationsprotokolle bezeichnet. TCP/IP (*Transmission Control Protocol / Internet Protocol*) stellt eine derartige Protokollfamilie dar, sie wird im weltweiten Internet, in privaten Intranets und in anderen Netzen verwendet. Netze, die auf dieser Protokollfamilie aufbauen, bezeichnet man als *IP-Netze*.

Komplexität und Weiterentwicklung

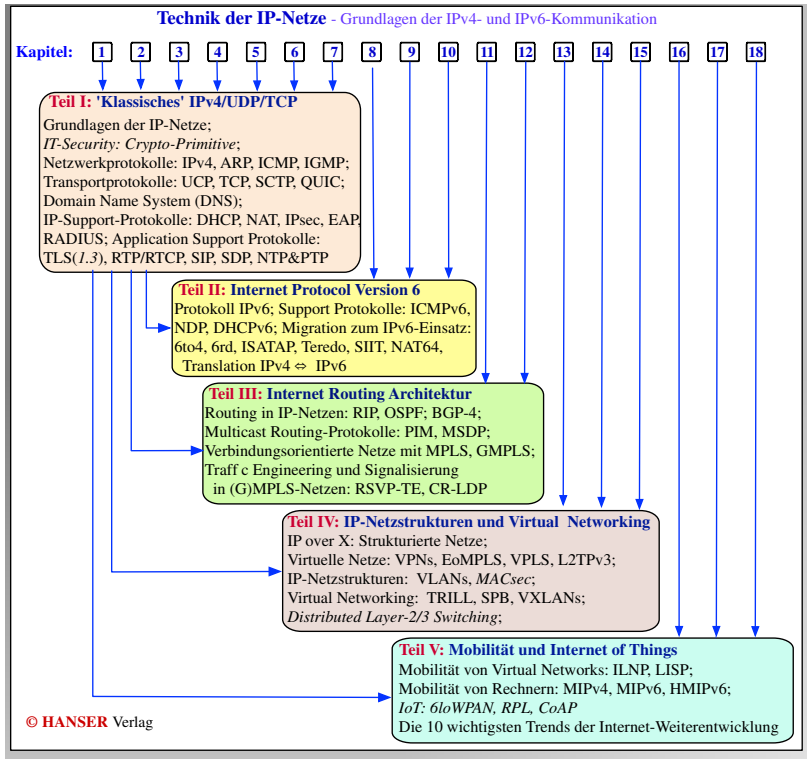
Ein IP-Netz – und insbesondere das Internet – besteht nicht nur aus mehreren Rechnern und IP/TCP dazwischen, sondern dahinter verbergen sich sehr komplexe Vorgänge. Das Internet stellt einen weltweiten Dienst zur Übermittlung nicht nur von Daten, sondern auch von audiovisuellen Informationen, also von Audio und Video, in Form von IP-Paketen dar. Vergleicht man diesen Dienst mit dem Briefdienst der Post, so entspricht ein IP-Paket einem Brief und die sogenannte IP-Adresse einer postalischen Adresse. Das massive Wachstum des Internet und die dabei entstehenden Probleme und neuen Anforderungen haben die Entwicklung sowohl eines neuen Internetprotokolls, des IPv6, als auch von Techniken MPLS und GMPLS für die Übermittlung der IP-Pakete über Hochgeschwindigkeitsnetze, insbesondere über optische Netze, vorangetrieben. Noch in der ersten Dekade dieses Jahrhunderts hat man von *Next Generation IP Networks* gesprochen und sie sind bereits Realität geworden.

Ziel des Buches

Dieses Buch gibt eine fundierte Darstellung zentraler Komponenten der TCP/IP-Protokollfamilie, wie z.B. IP, TCP, UDP, DNS und DHCP, sowie von Routing sowohl beim klassischen IP, IPv4 genannt, als auch beim IPv6. Das Buch erläutert die Strategien für die Migration zum Einsatz von IPv6, präsentiert die Konzepte zum Aufbau der IP-Netze auf Basis verschiedener Netztechnologien, wie LANs, WLANs, SDH und WDM, und geht auch auf die IP-Weitverkehrsnetze mit (G)MPLS ein. Die Themen wie die Realisierung von VPNs, *Virtual Networking* in LANs durch die Bildung von VLANs und VXLANs, Konzepte und Einsatz von TRILL und *Shortest Path Bridging* werden ebenso präsentiert. Die Darstellung der Protokolle MIPv4, MIPv6 und HMIPv6 zur Unterstützung der Mobilität von Rechnern wie auch der Protokolle ILNP und LISP, mit denen man die Mobilität virtueller Netzwerke erreichen kann, rundet den Inhalt dieses Buches ab.

An wen richtet sich das Buch?

Das Buch ist so aufgebaut, dass sowohl die notwendigen technischen Grundlagen fundiert dargestellt als auch verschiedene Aspekte bei der Planung und Verwaltung der IP-Netze diskutiert werden. Damit eignet es sich nicht nur als Lehrbuch für Studenten und Neueinsteiger, sondern auch als Nachschlagewerk für alle Interessenten, die für die



Planung, Realisierung, Verwaltung und Nutzung des Internet, von privaten Intranets und anderen IP-Netzen verantwortlich sind.

Aufbau des Buches

Zurzeit ist kein Buch verfügbar, in dem die Technik der IP-Netze so breit dargestellt wäre. Daher kann dieses Buch als ein Handbuch für alle Netzwerk-Verantwortlichen dienen. Durch die fundierte und praxisorientierte Darstellung der Inhalte eignet sich gut dieses Buch auch für alle 'Internet-Fans' zum Selbststudium. Dieses Buch präsentiert in 17 Kapiteln, die auf fünf Teile verteilt sind, alle wichtigen Aspekte der IP-Netze und kann nicht wie ein spannender Roman in einem Schlag durchgelesen werden. Das vorliegende Bild zeigt dessen logische Struktur und Abhängigkeiten zwischen Inhalten einzelner Teile, um den Lesern eine Orientierung zu geben, aus welchen Teilen man Kenntnisse benötigt, um beim Lesen verschiedene, voneinander abhängige Themenbereiche besser zu verstehen.

Inhalte der Kapitel

Betrachtet man die einzelnen Kapitel dieses Buches etwas detaillierter, so lassen sie sich wie folgt kurz charakterisieren:

Kapitel 1

Kapitel 1 präsentiert die Entwicklung des Internet sowie die notwendigen Grundlagen der Rechnerkommunikation und der *Kommunikationsprotokolle* und geht u.a. daher auf die folgenden Probleme ein: Welche Funktionen liegen den Kommunikationsprotokollen zugrunde und wie kann die Kommunikation in IP-Netzen mittels eines Schichtenmodells anschaulich dargestellt werden? Wie können die verbindungslo-

se und die verbindungsorientierte Kommunikation in IP-Netzen interpretiert werden und welche Bedeutung hat die Transportschicht? Welche Sicherheitsziele werden in IP-Netzen verfolgt und wie können diese technisch umgesetzt werden? Wie koordiniert die IETF die technologische Entwicklung des Internet und wie können wir diese verfolgen?

Kapitel 2 führt den Leser in die Problematik der IT-Security ein: Netze sind *Vermittlungsnetze*, die Nachrichten *zuverlässig* und *unveränderlich* von *A* nach *B* transportieren sollen. Heute besteht der Bedarf aber auch darin, dass der Nachrichtentransport *vertraulich* erfolgt. Wie das zu bewerkstelligen ist, versucht Kapitel 2 zu vermitteln, indem vier Krypto-Primitive eingeführt und erklärt werden. Moderne Kryptographie (Pre-Quantum) kommt nicht ohne Elliptische Kurven aus, deren Bedeutung und Einsatz hier erläutert wird. Auch die Frage der Benutzer- und Rechnerauthentifizierung als 'Digitale Identitäten' mit und ohne *X.509 Zertifikate* wird beleuchtet. Kapitel 2

Kapitel 3 stellt sowohl IPv4 als auch dessen Hilfsprotokolle ARP und ICMP umfassend dar und erläutert u.a. folgende Fragestellungen: Wie sind IPv4-Pakete aufgebaut und welche Steuerungsangaben kann der Header eines IPv4-Pakets enthalten? Welche Arten von IPv4-Adressen gibt es und wie werden sie aufgebaut? Wie erfolgt die *Adressierung* in IP-Netzen und wie werden Subnetze gebildet? Welche Bedeutung haben die Protokolle ARP und ICMP und wie funktionieren sie? Wie realisiert man *Multicasting* in IP-Netzen mit dem Protokoll IGMP? Kapitel 3

Von großer Bedeutung in IP-Netzen ist die sogenannte *Transportschicht* mit den klassischen Protokollen TCP und UDP; hierzu kommen noch die neuen Protokolle SCTP und UDP-Lite. Weil das IP keine zuverlässige Übermittlung der Pakete garantiert, verwendet man hauptsächlich das TCP und in einigen Fällen das SCTP, um die zuverlässige Übermittlung der IP-Pakete zu gewährleisten. Kapitel 4 präsentiert die Aufgaben der Transportschicht. Diese können einfach gestaltet sein, wie bei UDP, das verbindungslos fungiert; aber auch komplex, wie bei TCP. Hier werden *Sessions* aufgebaut und unterhalten und sich um den Datenfluss gekümmert. Aufbauend hierauf wurde das Protokoll SCTP entwickelt. Einen anderen Ansatz stellt QUIC dar, das auf UDP aufsetzt, intern aber über umfangreiche Mechanismen zur Steuerung und vor allen Datenverschlüsselung entsprechend TLS verfügt. Deren prinzipielle Arbeitsweise wird hier dargestellt und speziell betrachtet: Wie funktioniert der gemeinsame Datenfluss über ggf. mehrere Übertragungsmedien? Kapitel 4

Die Rechner in IP-Netzen werden zwar durch ihre IP-Adressen lokalisiert, aber es ist sinnvoll, statt einer IP-Adresse einen Rechner über seinen Namen anzusprechen – wie es auch unter Menschen üblich ist. Dies ist mit dem *Domain Name System* (DNS) möglich. Kapitel 5 liefert eine fundierte Darstellung von DNS, geht auf verschiedene Möglichkeiten des DNS-Einsatzes ein und erörtert u.a. die folgenden Probleme: Wie funktioniert DNS und welche Aufgaben kann DNS wahrnehmen? Wie erfolgt die Ermittlung der IP-Adresse aufgrund des Hostnamens und umgekehrt? Welche Informationen als Resource Records enthält DNS und wie werden diese strukturiert? Welche Ziele werden mit ENUM, DynDNS und DNSSEC und DNSCurve verfolgt? Kapitel 5

Kapitel 6 stellt, als *IP-Support-Protokolle* bezeichnete, ergänzende Lösungen für das IPv4-Protokoll dar und erläutert hierbei u.a. die folgenden Aspekte: Wie können sich Kapitel 6

Rechner mittels DHCP automatisch eine gültige IPv4-Adresse zuweisen? Welche Lösungen für die Nutzung von privaten IPv4-Adressen mithilfe von NAT (*Network Address Translation*) gibt es und welche Probleme entstehen dabei – insbesondere bei der audiovisuellen Kommunikation? Wie kann *IPsec* zum verschlüsselten und authentisierten Austausch von IP-Paketen genutzt werden? Welche Probleme verursacht NAT bei IPSec und wie können diese bewältigt werden? Welche Probleme ergeben sich bei der Überprüfung von 'Digitalen Identitäten' unter Einsatz von EAP, RADIUS und LDAP?

Kapitel 7 Mehrere ergänzende Lösungen sind nicht nur für das IPv4 nötig, sondern in Form spezieller Protokolle auch für Applikationen, sodass wir von *Application Support Protokollen* sprechen. Kapitel 7 präsentiert diese, erläutert deren Aufgaben und geht u.a. auf folgende Fragestellungen ein: Wie kann der Datentransport zwischen Applikationen mittels TLS (1.3) gesichert realisiert werden und welche Voraussetzungen sind hierfür nötig? Welche Protokolle zur Realisierung der Echtzeitkommunikation in IP-Netzen benötigt werden, welche Aufgaben haben sie und wie werden sie konzipiert? Gerade diese Protokolle benötigen eine Zeitsynchronisation zwischen den Teilnehmern. Wie wird dies mittels NTP und PTP ermöglicht, und welche Zeitformate und -referenzen werden hierbei genutzt?

Kapitel 8 Um den steigenden Anforderungen an IP-Netze gerecht zu werden, wurde das IPv6 als '*IP der nächsten Generation*' entwickelt und die Ära von IPv6 hat bereits begonnen. IPv6 bringt neue Möglichkeiten und diese reichen von Sicherheitsfunktionen über mehr Flexibilität bis hin zur Unterstützung von neuartigen Anwendungen. Das IPv6 ermöglicht die automatische Konfiguration von Rechnern, sodass man sogar von *Plug&Play-Konfiguration* spricht. Kapitel 8 stellt das IPv6 ausführlich dar und geht u.a. auf die folgenden Probleme ein: Welche Ziele wurden bei der Entwicklung von IPv6 verfolgt? Welche neuen Funktionen bringt IPv6 mit sich? Welche Arten von IPv6-Adressen gibt es und wie können sie den Rechnern zugewiesen werden?

Kapitel 9 Ein wichtiges Ziel bei der Entwicklung von IPv6 war die Unterstützung der automatischen Konfiguration von Rechnern. Hierfür stehen die Protokolle ICMPv6, NDP und DHCPv6 zur Verfügung. Diese *IPv6 Support Protokolle* stellt Kapitel 9 dar und geht hierbei u.a. auf folgende Aspekte ein: Wie wurde ICMPv6 konzipiert und welche Aufgaben hat es? Welche Funktionen liefert NDP, um die automatische Konfiguration von Rechnern mit IPv6 zu unterstützen? Wie bekommt ein IPv6-Rechner seine Netzkonfiguration automatisch zugewiesen?

Kapitel 10 Da die Umstellung von allen Rechnern, in denen das klassische IPv4 verwendet wird, auf das IPv6 nicht auf einen Schlag geschehen kann, benötigt man geeignete Systemlösungen für die Migration zum IPv6-Einsatz. Kapitel 10 präsentiert verschiedene Ansätze und Systemlösungen für die Koexistenz von IPv4 und IPv6 – vor allem die Konzepte *IPv6 over IPv4* und *IPv4 over IPv6*. Die Integration der IPv4- und der IPv6-Netze dank der Translation *IPv4* $\Leftrightarrow$ *IPv6* wird ebenso präsentiert. Hier werden u.a. folgende Probleme erörtert: Wie kann man sich die Koexistenz von IPv4 und IPv6 in einem Rechner vorstellen, wann ist diese Koexistenz möglich und welche Bedeutung hat sie? Wie kann die IPv6-Kommunikation über IPv4-Netze erfolgen? Wie lassen sich IPv6-Netzsegmente über IPv4-Netze verbinden? Wie können die Rechner aus

IPv6-Netzen auf das IPv4-Internet zugreifen? Wie erfolgt die Translation IPv4 ⇔ IPv6 und was ermöglicht sie?

Kapitel 11

Router fungieren in IP-Netzen als Knoten, ermitteln optimale Übermittlungswege, die sogenannten *Routen*, für die empfangenen IP-Pakete und leiten sie weiter. Kapitel 11 vermittelt eine kompakte Darstellung von Routing-Grundlagen und -Protokollen. Es werden hier die *Routing-Protokolle* RIP-1, RIP-2 und OSPF sowie BGP-4 erläutert. Dieses Kapitel zeigt auch, wie eine redundante Router-Auslegung mithilfe der Protokolle HSRP und VRRP erfolgen kann, und stellt die Protokolle PIM-SM und MSDP für das *Multicast-Routing* dar. Hierbei werden u.a. folgende Fragen beantwortet: Welche Aufgabe haben die Router und wie funktionieren sie? Welche Prinzipien liegen den Routing-Protokollen zugrunde? Wie verlaufen die Routing-Protokolle RIP und OSPF? Welche Erweiterungen dieser Protokolle sind für IPv6 notwendig? Wie funktioniert BGP-4, für welche Zwecke und wie kann es eingesetzt werden? Wie können die Router am Internetzugang redundant ausgelegt werden? Wie realisiert man Multicast-Routing in IP-Netzen?

Kapitel 12

Die IP-Netze im Weitverkehrsbereich basieren überwiegend auf dem MPLS-Konzept und auf der, als GMPLS (*Generalized MPLS*) bezeichneten, dessen Erweiterung. Die Techniken MPLS und GMPLS ermöglichen die Konvergenz von Ethernet u.a. mit SDH- und WDM-Netzen. Dank dieser Konvergenz können Ethernet heutzutage nicht nur als LAN eingerichtet werden, sondern *Ethernet-Services* können sogar weltweit verfügbar gemacht werden. Kapitel 12 stellt die Konzepte und Protokolle zum Aufbau der IP-Netze mit dem MPLS und dem GMPLS vor und geht u.a. auf die folgenden Probleme ein: Worin bestehen die Konzepte MPLS und GMPLS und welche Möglichkeiten entstehen durch deren Einsatz? Welche Services werden durch *Traffic Engineering* in IP-Netzen erbracht? Wie werden (G)MPLS-Netze aufgebaut und wie wird die IP-Kommunikation über sie realisiert? Wie erfolgt die IP-Kommunikation über optische Netze? Wie können Datenpfade über (G)MPLS-Netze dynamisch eingerichtet werden?

Kapitel 13

In vielen Unternehmen können gleichzeitig unterschiedliche Netztechnologien eingesetzt und entsprechend integriert werden. Sie lassen sich mithilfe von *Tunneling-Techniken* so einsetzen, dass virtuelle Standleitungen für den Transport von Daten über öffentliche IP-Netze aufgebaut werden können. Diese Idee hat zur Entstehung von VPNs geführt. Somit erläutert Kapitel 13 einerseits die Konzepte für den IP-Einsatz in Netzen mit klassischen LANs (Ethernet), Punkt-zu-Punkt-Verbindungen (z.B. physikalischen Standleitungen, Satellitenverbindungen) und WLANs. Andererseits präsentiert dieses Kapitel auch die Lösungen und Protokolle für den Aufbau von VPNs auf Basis sowohl klassischer IP-Netze mittels des IPsec als auch der IP-Netze mit den Techniken MPLS bzw. GMPLS, die auch als *Provider Provisioned VPNs* bezeichnet werden. Hierbei geht dieses Kapitel u.a. auf folgende Fragestellungen ein: Wie kann man sich ein logisches LAN-Modell vorstellen und wie kann die *Multiprotokollfähigkeit in LANs* erreicht werden? Welche Ideen liegen den WLANs nach IEEE 802.11 zugrunde und wie werden WLANs mit einem Ethernet gekoppelt? Welche Typen von virtuellen Netzen gibt es, wie werden sie aufgebaut und wie können sie genutzt werden? Wie lassen sich sichere, virtuelle IP-Netze aufbauen?

Kapitel 14

In den letzten Jahren haben sich einige Megatrends auf der 'Netzwerkwelt' herauskristallisiert. In privaten Netzwerken spricht man heute von *Layer-3-Switching* und von VLANs (*Virtual LANs*). Dabei stellt die Virtualisierung von Rechnern neue Anforderungen an IP-Netze. Die Unterstützung der Mobilität virtueller Rechner und virtueller Netzwerke sowie der Wunsch nach flexibler Möglichkeit, einen Rechner bzw. ein Netzwerk an das Internet parallel anbinden zu können, sind nur die wichtigsten von ihnen. Diese Probleme erläutert Kapitel 14 und präsentiert neue Konzepte und Protokolle hierfür, um diesen Anforderungen gerecht zu werden. Hervorgehoben sei hier *Shortest Path Bridging* (SPB), TRILL, VXLANs, ILNP und LISP. Dieses Kapitel geht u.a. auf folgende Fragen ein: Wie werden moderne IP-Netzwerke physikalisch und logisch strukturiert? Wie funktionieren Layer-2- und Layer-3-Switches, wo und wie werden sie eingesetzt? Wie können komplexe, auch virtuelle Rechner enthaltene VLANs gebildet werden und welche Bedeutung dabei hat VLAN Tagging? Worin bestehen die Ideen von TRILL, SPB, VXLAN, ILNP und LISP? Welche Möglichkeiten der Integration von IPv4 und IPv6 liefert LISP?

Kapitel 15

Der Gedanke der Virtuellen Netze kann aber von einer IP-basierten Infrastruktur gelöst werden und sehr effizient auf dem Layer 2 umgesetzt werden. Hiermit ergeben sich Konvergenzen von Layer 2 und Layer 3 Verfahren, die unter dem Stichwort 'Distributed Layer-2/3 Switching in Kapitel 15 vorgestellt werden, die bis zu auf Ethernet basierten virtuellen Netzen reicht, sowie so wie sie heute im Provider-Umfeld angeboten werden.

Kapitel 16

Um die Mobilität in IP-Netzen zu ermöglichen, wurden die Protokolle MIP (*Mobile IP*), MIPv6 (*Mobile IPv6*) und HMIPv6 (*Hierarchical MIPv6*) entwickelt. Kapitel 16 zeigt, wie diese Protokolle funktionieren und was gemacht werden muss, damit ein mobiler Rechner während bestehender Verbindungen ein Subnetz verlassen und in ein neues hinein bewegen kann, ohne die bestehenden Verbindungen abbrechen zu müssen. Auch die Integration von Hotspots mit dem Internet und die Möglichkeiten von Roaming zwischen Hotspots werden präsentiert. Darüber hinaus werden u.a. die folgenden Aspekte erörtert: Welche Ansätze und Protokolle zur Unterstützung der Mobilität in IP-Netzen gibt es? Wie kann *Roaming* zwischen Hotspots realisiert werden? Wie verläuft die Kommunikation beim Einsatz von MIP bzw. von MIPv6?

Kapitel 17

Ein in der Tat neues 'Kapitel' haben wir mit der Diskussion um das 'Internet of Things' IoT Kapitel 17 aufgeschlagen. Hier stellen wir die wesentlichen Konzepte de 'Internet der Dings', die Adaption von IPv6 für IoT – 6LoWAPN – die nun notwendigen Routingvefahren, sowie das Applikationsprotokoll CoAP vor.

Kapitel 18

Der Zukunft des Internet und seine möglichen Anwendungen und Perspektiven ist Kapitel 18 gewidmet. Wir wissen ja: "Schwer zu sehen, in ständiger Bewegung die Zukunft ist." (Meister Joda). Trotzdem ist hier der Versuch gewagt, die beherrschenden Tendenzen in systematischer Weise zu betrachten und die aktuelle Diskussion in den Internet-Gremien zu referenzieren.

Verständnisfragen

Zu jedem einzelnen Kapitel gibt es ergänzend 'Verständnisfragen', durch die der geneigte Leser sein Wissen um zentrale Punkte der einzelnen Kapiteln überprüfen

und ggf. vertiefen kann. Die Antworten auf diesn Fragen finden sich Online auf der Webseite des Buchs.

Vorwort zur vierten Auflage

Geschuldet der schnellen Entwicklung der Internet-Technologien und der guten Akzeptanz unserer 'Technik der IP-Netze' haben wir unsern 'siebenjährigen' Updatezyklus (erste, zweite und dritte Auflage) etwas beschleunigt und stellen nun mit der vierten Auflage eine aktualisierte Version zur Verfügung. Neben obligatorischen Verbesserungen und Richtigstellungen in einigen Details, sind folgende Aspekte neu hinzugekommen:

- Die heutzutage als unentbehrliche IT-Security bei der Internetnutzung wird auf dem aktuellen Stand umfangreich diskutiert und in Kapitel 2 an zentraler Stelle untergebracht. Wir führen hier die vier zentralen *kryptographischen Primitiven* vor, die eine hervorgehobene Rolle einnehmen.
- Die Switching-Technologien für IP-Netze umfassen nun Layer-2 und Layer-3 Eigenschaften, deren Würdigung in Kapitel 16 zu finden ist.
- Ein wichtiger neuer Aspekt stellt das *Internet of Things* (IoT) dar. Dies würdigen wir in den Kapiteln 17 und 18. Während zunächst Lösungen für Komponenten mit Ressourcen-beschränkter und im Besonderen geringer elektrischer Leistung (6LoWAPN) und die hieraus erwachsenden Konsequenzen, die sich für das Routing und für Applikationen, wie dem *Constrained Application Protocol* (CoAP) ergeben, beschrieben werden, spannen wir in Kapitel 18 dieses neue Umfeld mit seinen vielfältigen Facetten in Gänze auf.

Die vierte Auflage wurde auch typographisch aufgefrischt:

Layout-Anpassungen und Beispiele

- Das Layout wurde über die vielen Kapitel vereinheitlicht und die Tabellen mit einem modernen Design versehen.
- Beispiele im Buch werden – wie in diesem Fall – mit einem links-seitigen Balken hervorgehoben.

Vorwort zur fünften Auflage

Dies Auflage beinhaltet im wesentlichen redaktionelle Änderungen und Anpassungen, die der Entwicklung des Internets und seiner Protokolle geschuldet sind. Allerdings verweist diese Auflage bereits auf das nächste Projekt der Autoren, das virtuellen Netzstrukturen gewidmet sein wird.

Redaktionelles

- *Redaktionelle* Änderungen betreffen die Verlinkung der Inhalte und die Verbesserung der Nutzung des Buchs als PDF-Dokument. Somit kann man im PDF nun zwischen Kapiteln und Abschnitten navigieren. Alle Links zu den IETF-Quellen wurden von *http* auf *https* umgestellt; das trägt zum Komfort bei, da dort keine 'Umleitung' gesetzt wurde. Wir wurden gelegentlich gefragt: *Warum gibt es das Buch nicht als eBook?* Die Antwort ist einfach: 'Technik der IP-Netze' wurde im

aktuellen Format mittels LaTeX gesetzt. Darstellung, Seitenumbruch und Lesbarkeit sind darauf abgestimmt. Bei einem eBook-Format ginge dies verloren.
Bei den Abbildungen, wo wir Bitstrukturen erklären, halten wir es nun mit *Edsger Dijkstra* (siehe Vorspann zum Thema 'Internet-Routing-Architektur') und fangen bei '0' zu zählen an: Das erste Bit wird mit '0' nummeriert. Das ist auch die übliche Schreibweise in der Literatur.
Wurde das Manuskript bislang unter MacOS erstellt, findet nun Linux Anwendung. Die neuen Abbildungen wurden mit *InkScape* erstellt, an dessen Nutzung sich der Autor noch gewöhnen muss ;-)

Inhaltliches

- *Inhaltliche* Änderungen sind überschaubar, und es mussten lediglich einige wenige Fehler korrigiert werden. In Kapitel 2 wurde die ECC-Kryptographie endlich entsprechend ihrer Bedeutung nach eingeführt. Trotzdem ist 'Technik der IP-Netze' weder ein Lehrbuch der Kryptographie noch der 'IT-Security'. Allerdings ist die Literatur zu diesen Themen relativ überschaubar, sodass wir einen aktuellen Abriss hierzu als notwendig betrachtet haben.
  Das Kapitel 4 integriert nun endlich auch Multipath-TCP an der richtigen Stelle, und es wurde dem QUIC-Protokoll ein eigener Abschnitt gewidmet, der versucht, dieses doch recht komplexe Protokoll zumindest in seinen Grundzügen zu erklären.
  In verteilten System spielt die Zeitsynchronisation von IT-Systemen eine wichtige Rolle (auch weil man Eindringlinge erkennen will). Daher haben wir in Kapitel 7 sowohl die Protokolle NTP als auch PTP beschrieben; nicht immer zur Zufriedenheit des Autors.

Die fünfte Auflage ist dem Internet-Aktivisten *Sven Guckes* gewidmet.

Technik der IP-Netze – Homepage

Die Homepage des Buches ist unter https://www.fehcom.de/pub/tipn.html erreichbar, wo sich auch Korrekturen einfinden werden. Ihre Kritik, Verbesserungsvorschläge und eventuell Ihre Korrekturen sind willkommen und wir nehmen sie gerne entgegen. Für Lehr- und Ausbildungszwecke stellen wir die Abbildungen auf Anfrage zur Verfügung.

Danksagung

Ein so umfangreiches Buch kann ohne Anregungen von außen und einen entsprechenden Erfahrungsaustausch nicht geschrieben werden.

Ein besonderer Dank gilt Herrn Dipl. math. Jürgen Müller (Darmstadt) und Herrn Dirk Müller (Koblenz) für die notwendige Sorgfalt, die dritte Auflage des Buches intensiv durchzuarbeiten und uns Korrekturen vorzuschlagen. Ebenso möchten wir Frau Hasselbach als primäre Ansprechpartnerin sowie Frau Irene Weihart und Frau Kristin Rothe und nicht zu vergessen Herrn Jürgen Dubau für das sorgfältige Lektorat auch in dieser Auflage danken. Bei einem so umfangreichen Projekt ergeben sich immer Inkonsistenzen, auf die die aufmerksame Leserschaft stößt.

Die Autoren

Prof. Dr.-Ing. Anatol Badach

Über 30 Jahre war er auf den Gebieten Informatik und Telekommunikation beruflich tätig; Promotion (1975), Habilitation (1983). Von Dezember 1985 bis August 2012 war er Professor im Fachbereich Angewandte Informatik an der Hochschule Fulda. Seine Schwerpunkte in Lehre und Forschung waren: Rechnerkommunikation, Netzwerktechnologien und Multiservice Networking. Er hat u.a. auf den Gebieten: Netzwerktechnologien und Protokolle, VoIP und Next Generation Networking geforscht und verfolgt mit Engagement einige wichtige Entwicklungen weiter.

Prof. Badach ist Autor zahlreicher Veröffentlichungen und u.a. zahlreicher anderer Fachbücher, darunter *Voice over IP – Die Technik*, *Netzwerkprojekte* (Mitautor), *Web-Technologien* (Mitautor), *Integrierte Unternehmensnetze*, *Datenkommunikation mit ISDN*, *High Speed Internetworking* (Mitautor), *ISDN im Einsatz*. Seine Erfahrung vermittelt er weiter als Leiter/Referent bei Fachkongressen und -seminaren, Berater bei innovativen Projekten und Entwicklungen, Autor von Fachbeiträgen.

https://www.competence-site.de/Anatol-Badach

Prof. Dr. Erwin Hoffmann

Jahrgang 1958, Studium der Physik und Astrophysik an der Universität Bonn und 1989 Promotion an der TU München (Max-Planck-Institut für Physik und Astrophysik). Durch seine Tätigkeit in der experimentellen Teilchenphysik am CERN und Fermilab verschaffte er sich Kenntnisse über unterschiedlichste Rechnerbetriebssysteme. Beruflich war er zunächst im Bereich Hochgeschwindigkeitsnetze (FDDI) engagiert sowie mit der Implementierung von TCP/IP auf IBM-Großrechnern.

Seit 1998 ist er an der Weiterentwicklung der Software von D.J. Bernstein involviert und veröffentlicht diese als *Public Domain*. Heute ist er Professor an der Frankfurt University of Applied Sciences mit den Schwerpunkten Rechnernetze, Betriebssysteme, IT-Security, Software Engineering sowie Verteilten Systeme.

https://www.fehcom.de

Dieses Buch möchten wir all jenen widmen, die dank ihrer technischen Schöpfungen zur Entstehung des Internet beigetragen haben, und ebenso denen, die sich dafür engagieren das Internet weiterzuentwickeln, es offen und aufrecht zu erhalten.

Prof. Anatol Badach (Fulda/Petersberg)
Prof. Erwin Hoffmann (Höhn-Schönberg) – im September 2022

Teil I

'Klassisches' IPv4/UDP/TCP

Wilkomen in CSNET!

Michael,

This is your official welcome to CSNET.
We are glad to have you aboard.

From: Laura Breeden
breeden%csnet-sh.arpa@csnet-relay.csnet
To: rotert%germany@csnet-relay.csnet
Via: csnet-relay; 3 Aug 84 10:44-MET

1 Grundlagen der IP-Netze

Internet als IP-Netz

Die heutige Gesellschaft kann man sich ohne Internet kaum noch vorstellen. Das *Internet* ist ein weltweites Rechnernetz, in dem nicht nur die Daten, sondern auch alle digitalisierten Echtzeitmedien wie Sprache, Audio und Video mit dem *Internet Protocol* (IP) übermittelt werden. Das Internet und alle anderen Netze auf Grundlage von IP nennt man *IP-Netze*. Die Kommunikation zwischen zwei Rechnern über ein IP-Netz bedeutet aber nicht nur zwei Rechner und IP dazwischen, sondern dahinter verbergen sich sehr komplexe Kommunikationsregeln, die in Form von *Kommunikationsprotokollen* spezifiziert werden.

Protokollfamilie TCP/IP

In IP-Netzen bilden alle Kommunikationsprotokolle eine Protokollfamilie, die sogenannte Protokollfamilie TCP/IP. Diese Familie, die sich seit mehr als 40 Jahren entwickelt hat, enthält außer IP und TCP (*Transmission Control Protocol*) eine Vielzahl weiterer Protokolle. Um diese Protokolle systematisch erläutern zu können, ist ein anschauliches Modell sehr hilfreich. Es basiert auf dem *OSI-Referenzmodell* (*Open System Interconnection*), das bereits Ende der 70er Jahre eingeführt wurde.

Überblick über das Kapitel

Dieses Kapitel schildert in Abschnitt 1.1 kurz die bisherige und zukünftige Entwicklung des Internet und beschreibt in komprimierter Form die Hauptkomponenten des WWW (*World Wide Web*). Abschnitt 1.2 erläutert die grundlegenden Funktionen der *Kommunikationsprotokolle* und geht dabei insbesondere auf die Ideen der Fehlerkontrolle, Flusskontrolle und Überlastkontrolle. Dem Schichtenmodell für die Darstellung von Prinzipien der Rechnerkommunikation widmet sich Abschnitt 1.3. Allgemeine Prinzipien der Kommunikation in IP-Netzen erläutert Abschnitt 1.4. Die wichtigsten Komponenten der Protokollfamilie TCP/IP präsentiert kurz Abschnitt 1.5. Abschnitt 1.6 geht auf den Aufbau der Organisation IETF (*Internet Engineering Task Force*) und die Internet-Standards ein. Schlussbemerkungen in Abschnitt 1.8 runden dieses Kapitel ab.

Ziel dieses Kapitels

In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Wie sah die bisherige Entwicklung des Internet aus und welche aktuellen Trends gibt es?
- Welche Funktionen liegen den Kommunikationsprotokollen zugrunde und wie kann die Kommunikation in IP-Netzen mittels eines Schichtenmodells anschaulich dargestellt werden?
- Wie können die verbindungslose und die verbindungsorientierte Kommunikation in IP-Netzen interpretiert werden und welche Bedeutung hat die *Transportschicht* in IP-Netzen mit den Protokollen TCP, UDP sowie SCTP und QUIC?
- Wie koordiniert die IETF die technologische Internet-Weiterentwicklung und wie können wir diese verfolgen?

1.1 Entwicklung des Internet

Es begann in den 60er Jahren

Die ersten Spuren, die in indirekter Form zur Entstehung des Internet beigetragen haben, führen zurück in die 60er Jahre. In dieser Zeit wurde zum ersten Mal für die amerikanische Regierung eine Kommunikationsform für den Fall eines nuklearen Krieges erforscht. Die damaligen Überlegungen beinhalten bereits die noch heute geltenden Grundprinzipien der paketvermittelnden Kommunikation. Die Entwicklung des Internet lässt sich grob in folgende Phasen einteilen:

- Das Internet vor der Nutzung des WWW (*World Wide Web*): Aufbau- und Experimentierphase als ARPANET und Verbreitung des Internet vor allem als Forschungs- und Wissenschaftsnetz.
- Die *Schaffung des WWW*.
- Das *Internet nach der Etablierung des WWW* als weltweite Kommunikationsinfrastruktur für wissenschaftliche, private und kommerzielle Nutzung.

1.1.1 Internet vor der Nutzung des WWW

ARPANET als Vorläufer des Internet

Die Geschichte des Internet ist eng mit der Entstehung des ersten Rechnernetzes im Jahr 1969 verbunden. Die Entwicklung dieses Rechnernetzes wurde vom *US Defense Advanced Research Project Agency* (DARPA), einer *Organisation des Department of Defense* (DoD), initiiert und es trug den Namen ARPANET (*Advanced Research Project Agency Network*). Abb. 1.1-1 illustriert den Aufbau des ARPANET.

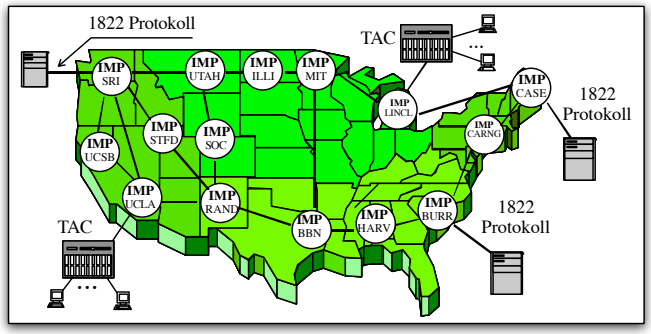


Abb. 1.1-1: Allgemeiner Aufbau von ARPANET – IMP dienen als Knoten
TAC: Terminal Access Controller, IMP: Internet Message Processor

Geburt von ARPANET

DARPA wollte zunächst digitale Telekommunikation auf Basis einer 'packet switching'-Methode über unterschiedliche Netze bereitstellen. Als erster Schritt hierzu wurde am 2. September 1969 am *University College of Los Angeles* (UCLA) ein Computer an einen *Internet Message Processor* (IMP) angeschlossen. Der IMP war auf der Basis eines Honeywell 516 Rechners der Firma *Bolt, Beranek & Newman* (BBN) gebaut worden.

70er Jahre

Anfang der 70er Jahre wurden die mittlerweile 15 zusammengeschalteten IMPs unter dem Namen *ARPANET* gehandelt. Das Kommunikationsprotokoll der IMPs trug die

Bezeichnung BBN 1822 und kann als Vorläufer von IP gelten. Um ARPANET mit anderen Paketnetzen koppeln zu können, wurden 1974 ein Internetwork-Protokoll sowie Gateways entwickelt.

ICCB und NCP

Die weitere technische Entwicklung der zunächst NCP (*Network Control Program*) genannten Protokolle wurde vom DARPA entkoppelt und in die Obhut des *Internet Configuration Control Board* (ICCB) gegeben. Mit der 1983 von der *Defense Communication Agency* (DCA) vorgenommenen Trennung des militärisch genutzten Teils des Netzes *MILNET* vom ARPANET war ein weiterer wichtiger Schritt für die breite öffentliche Entwicklung des Internet gemacht.

BSD und Sockets

Diese Trennung hatte auch entscheidenden Einfluss auf das Betriebssystem UNIX, das von der Firma AT&T 1969/1970 entwickelt wurde. Wiederum am UCLA wurde in dieses Betriebssystem (genauer: unter UNIX System III) eine Netzwerk-Programmierschnittstelle *Sockets* implementiert, die es erlaubte, eine direkte Rechnerkommunikation mit dem ARPANET aufzunehmen. Dieses UNIX wurde als *Berkeley Software Distribution* (BSD) gegen eine geringe Gebühr abgegeben und fand daher schnellen Einzug in Lehre und Forschung. Die weitere Verbreitung von UNIX und Internet sowie ihre technische Fortführung waren die Folge. Nach der ersten Version BSD 4.0 folgte 4.2 und anschließend 4.3, wobei die spätere kommerzielle Weiterentwicklung durch die Firma Sun Microsystems als Betriebssystem Sun OS und später Solaris erfolgte.

IAB und TCP/IP

Eine 1983 stattfindende Reorganisierung des ICCB führte nicht nur zur Konstituierung des *Internet Activity Board* (IAB) anstelle des ICCB, sondern auch zur Festlegung der als Standard geltenden, nun TCP/IP genannten Protokollfamilie. Mit der weiteren Entwicklung wurde auch dieser organisatorische Rahmen zu eng. Das IAB wurde zum *Internet Architecture Board* umfirmiert und u.a. um folgende Gremien ergänzt:

IETF *Internet Engineering Task Force* als offenes Gremium von Netzwerk-Architekten und -Designern, vor allem aus interessierten Firmen und Einzelpersonen gebildet, um die Entwicklung des Internet zu koordinieren `https://www.ietf.org`. Auf die Organisation der IETF geht Abschnitt 1.7 näher ein.

IESG *Internet Engineering Steering Group* mit der Aufgabe, die Tagesaufgaben der IETF zu managen und eine erste technische Stellungnahme zu neuen Internet-Standards zu beziehen `https://www.ietf.org/iesg.html`.

IRTF *Internet Research Task Force* als Gremium zur Grundlagendiskussion langfristiger Internet-Strategien und -Aufgaben `https://www.irtf.org`.

IEPG *Internet Engineering and Planning Group*, eine offene Arbeitsgruppe von Internet- Systemadministratoren, die dem Ziel verpflichtet sind, einen koordinierten Internet-Betrieb zu gewährleisten `https://www.iepg.org`.

ICANN *Internet Corporation for Assigned Names and Numbers* mit der Aufgabe, die Verwendung und die Konsistenz der im Internet benutzten Namen, Optionen, Codes und Typen zu regeln und zu koordinieren (`https://www.icann.org`).

NSFNet

Nach der Trennung des militärischen vom zivilen Teil des ARPANET wurde dieses zunächst zum Austausch wissenschaftlicher Informationen genutzt und von der *National Science Foundation* (NSF) betreut. Diese baute 1986 den zivilen Teil als nationales Backbone-Netz aus, das als NSFNet bekannt geworden ist. Drei Jahre später (1989) waren ca. 100 000 Rechner, die sich an Universitäten und Forschungslabors, in der US-Regierung und in Unternehmen befanden, am NSFNet angeschlossen. In nur einem Jahr (1990) hat sich die Anzahl der angeschlossenen Rechner verdoppelt, wobei das NSFNet ca. 3 000 lokale Netze umfasste. Dies war auch der Zeitpunkt, an dem das *Domain Name System* (DNS) eingeführt wurde.

EARN

Auch in Europa wurden die ersten Ansätze zur Vernetzung der Forschungsinstitute durch EARN (*European Academic Research Network*) durchgeführt, um die bislang nationalen Netze wie z.B. BitNet in England und das vom DFN-Verein (*Deutsches Forschungsnetz*) getragene WiN (*Wissenschafts-Netz*), miteinander zu koppeln.

USENET

Neben dem direkten, d.h. festgeschalteten und teuren Anschluss ans Internet, wie er bei Universitäten und Forschungseinrichtungen sowie auch bei Firmen üblich ist, wurde bald ein loser Verbund von Systemen – vor allem auf UNIX-Rechnern basierend – aufgebaut, die über Telefonleitungen und Modems gekoppelt waren: das USENET. Hier wurden die Rechner über das Protokoll UUCP (*UNIX to UNIX Copy*) miteinander verbunden und Nachrichten ausgetauscht. Hauptzweck des USENET war die Verbreitung von E-Mail sowie vor allem von *NetNews*, die in *Newsgroups* themenstrukturierte, virtuelle Nachrichtenbretter darstellen, in denen zunächst technische Fragen zu Rechnern, Programmiersprachen und dem Internet behandelt wurden. USENET war zeitweise so populär, dass es mit dem Internet selbst identifiziert wurde.

Cyberspace

Die 'kopernikanische Wende' des Internet vollzog sich mit der Schaffung des WWW [Abschnitt 1.1] durch Tim Berners-Lee. Damit wurde die Möglichkeit geschaffen, mittels eines einfachen 'Browsers' grafisch auf öffentlich verfügbare Internet-Ressourcen über Webserver zugreifen zu können.

'dot-com' Internet

Sehr schnell fand die 'kopernikanische Wende' Ergänzung in einer 'keplerschen Wende': Mit Realisierung einer allgemeinen nutzbaren Verschlüsselung des Datenverkehrs zunächst auf Grundlage des *SSLeay*-Protokolls, entwickelt durch die Firma Netscape Anfang der 90er Jahre, konnte nun das Internet auch für den kommerziellen Einsatz genutzt werden. Das Internet explodierte, was sowohl die Anzahl der Teilnehmer und der Anwender, als auch die Server und die Datenmenge betraf. Das Internet mutierte vom Wissenschaftsnetz zum multimedialen *Cyberspace* und zum kommerziellen, immer geöffneten Einkaufsparadies, der 'dot-com'-Ökonomie.

1.1.2 Die Schaffung des WWW

Der Aufschwung und die umfassende Verbreitung des Internet ist einer Errungenschaft des europäischen Labors für Elementarteilchenforschung CERN (*Conseil Européen pour la Recherche Nucléaire*) in Genf zu verdanken. Mit dem raschen Wachsen und der Internationalisierung der Forschergruppen stellte sich heraus, dass die bisherige Infrastruktur des Internet, das maßgeblich zum Austausch der Forschungsergebnisse genutzt wurde, nicht mehr adäquat war. So wurde nach einem Verfahren gesucht, mit dem die Informationsquellen mittels *Hyperlinks* untereinander direkt verknüpft

werden konnten. Der CERN-Mitarbeiter Tim Berners-Lee hatte 1989/1990 [GC02] die Idee,

- die Dokumente in einer speziellen Seitenbeschreibungssprache HTML (*Hypertext Markup Language*) aufzubereiten und diese untereinander durch Hyperlinks zu verbinden, wobei — HTML
- die Dokumenten-Referenzen über einheitliche Adressen URL (*Uniform Resource Locator*) erfolgen sollten und — URL
- die Verknüpfung über ein neues, einfaches Protokoll HTTP (*Hypertext Transport Protocol*) abgewickelt werden sollte. — HTTP

Diese Idee brachte den Vorteil, dass nun nicht mehr der Systemadministrator des Servers, sondern der Dokumenten-Eigentümer für die Verknüpfung der Informationen verantwortlich war [Abb. 1.1-2]. Das nach dieser Idee weltweit verteilte System stellt heute unter dem Namen *World Wide Web* (WWW) – auch kurz *Web* genannt – die wichtigste Informationsquelle dar. WWW bildet ein weltweites Geflecht (Web) von Rechnern, die als *Webserver* fungieren und verschiedene Informationen enthalten.

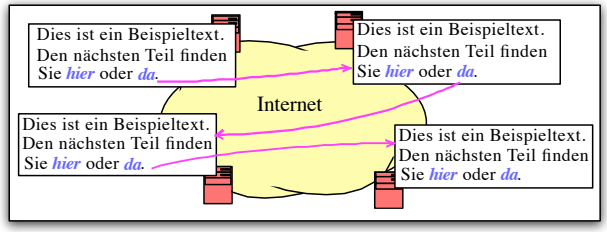


Abb. 1.1-2: Verknüpfung von Dokumenten auf unterschiedlichen Servern mittels Hyperlinks

Zusammen mit seinem Kollegen Robert Cailliau schrieb Tim Berners-Lee den ersten graphischen *Webbrowser* (als Software zur Darstellung der Web-Inhalte) sowie den ersten Webserver. Neben der graphischen Version wurde auch bald eine zeichenorientierte Browser-Version entwickelt, die weitgehend plattformunabhängig war. Mit der Verbreitung von Webbrowsern war der Siegeszug des WWW nicht mehr aufzuhalten. Heute spricht man in Bezug auf den Transport der verschiedenen Informationen im WWW vom *Web-Dienst*. — WWW als Web-Dienst

Der Web-Dienst stellt einen Internetdienst auf grafischer Basis dar, der hauptsächlich zur Informationsabfrage verwendet wird. Die für die Realisierung des Webdienstes erforderlichen Komponenten zeigt Abb. 1.1-3. — Hauptkomponenten des Webdienstes

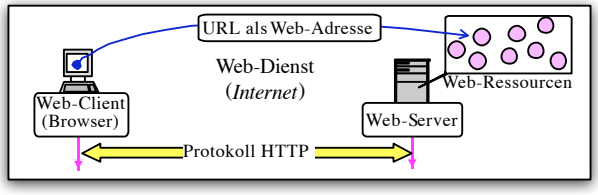


Abb. 1.1-3: Hauptkomponenten des Webdienstes – URL dient als Adresse

Die Grundkomponenten des Webdienstes sind:

Webbrowser
- Eine Software für die Darstellung von Web-Inhalten in Form von *Webseiten* (*Web-Pages*) auf dem Bildschirm des Rechners. Diese Software stellt einen *Web-Client* dar und man bezeichnet sie als *(Web)-Browser*. Ein Browser zeigt die angeforderte Webseite an und bietet zahlreiche Funktionen für die Navigation im Web-Dienst.

URLs als Web-Adressen
- Einheitliche *Web-Adressen* zur Angabe der Lokation von Web-Inhalten, die man auch Web-Ressourcen nennt. Eine Web-Ressource stellt oft eine Datei in beliebigem Format (wie z.B. HTML, JPEG oder GIF) dar. Als einheitliche Web-Adresse wird eine *URL* (*Uniform Resource Locator*) verwendet. `https://www.hs-fulda.de/fb/ai` ist ein Beispiel hierfür. Die Adressierung von Web-Ressourcen wird noch näher erläutert.

Webserver
- Webserver mit *Web-Inhalten* (Web-Ressourcen), auf die über das Internet zugegriffen werden kann. Die Web-Inhalte werden auch *Web-Content* genannt. Auf einem Webserver können auch herkömmliche Programme abgespeichert und an den Web-Dienst über eine Software-Schnittstelle, beispielsweise CGI (*Common Gateway Interface*), angebunden werden. Diese Programme können über das Internet aufgerufen werden.

HTML
- Eine abstrakte Sprache für die Beschreibung von *Webseiten*. Eine Webseite besteht in der Regel aus mehreren Web-Objekten und wird als Hypertext dargestellt. Für die Darstellung von Webseiten verwendet man die Seitenbeschreibungssprache HTML (*Hypertext Markup Language*), die in den Jahren 1989/1990 entwickelt wurde. HTML wird beständig weiterentwickelt und modifiziert, sodass es bereits mehrere HTML-Varianten (derzeit HTML 5) gibt. Ergänzt wird dies durch eine Formatierungssprache, *Cascading Style Sheets* (CSS) (aktuelle Version 3), durch die eine Trennung zwischen Inhalt und Format möglich wird.

Protokoll HTTP
- Ein Protokoll für den Transport von Web-Inhalten zwischen Browsern und Webservern. Hierfür dient das HTTP (*Hypertext Transport Protocol*). Hat ein Benutzer eine Webseite angefordert (z.B. indem er einen Hyperlink auf dem Bildschirm angeklickt hat), sendet sein Browser die Anforderung (d.h. einen HTTP-Request) an den durch die URL angegebenen Webserver. Dieser empfängt diese Anforderung und sendet eine Antwort (d.h. einen HTTP-Response), in der sich der angeforderte Web-Inhalt befindet, an den Browser zurück.

HTTP nutzt TCP

Für die Übertragung der Web-Inhalte zwischen Webserver und -browser nutzt HTTP das verbindungsorientierte Transportprotokoll TCP (*Transmission Control Protocol*). Dies bedeutet, dass eine TCP-Verbindung für die Übermittlung von Web-Inhalten zwischen Web-Client und -Server aufgebaut werden muss. Das Protokoll TCP wird in Abschnitt 3.3 detailliert beschrieben.

Adressierung von Web-Ressourcen

Um die Lokation einer gewünschten Web-Ressource im Internet anzugeben, braucht

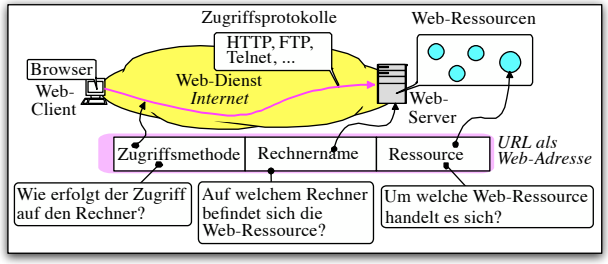


Abb. 1.1-4: Prinzip der Adressierung beim Web-Dienst

man die *Web-Adresse*. Was muss aber eine Web-Adresse angeben und wie sieht sie aus? Abb. 1.1-4 zeigt, was man beim Web-Dienst zu tun hat.

Was muss eine Web-Adresse enthalten?

Beim Zugriff auf eine Ressource muss folgendes angegeben werden [BRS03]:

- Die Art und Weise wie der Zugriff auf den Webserver erfolgt, also die Zugriffsmethode, d.h. welches Protokoll (HTTP, FTP, ...) verwendet wird.
- Der Rechner, auf dem sich die gewünschte Ressource befindet. Man muss auf den Rechner verweisen, um ihn eindeutig zu lokalisieren.
- Die Ressource, um die es sich handelt.

1.1.3 Internet nach der Etablierung des WWW

Internet-Strukturierung

Das Internet ist nach der Geburt des WWW ein so komplexes weltweites Rechnernetz geworden, dass es nicht möglich ist, hier die Struktur seiner physikalischen Vernetzung zu zeigen. Sie ist unbekannt und wächst ständig. Das Internet ist aber nach einem hierarchischen Prinzip aufgebaut. Wie Abb. 1.1-5 illustriert, stellt das Internet eine Vernetzung von Rechnern dar, in der man mehrere Schichten unterscheiden kann.

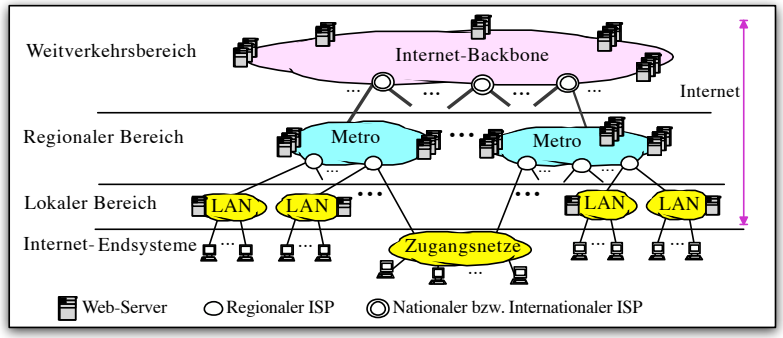


Abb. 1.1-5: Allgemeine Internet-Strukturierung – Aufbau als baumartige Struktur
ISP: Internet Service Provider; LAN: Local Area Network; Metro: Metro(politan)-Netz

Die untere Schicht bilden lokale Netzwerke (LANs) mit den Webservern, die den privaten Firmen, öffentlichen Institutionen, Hochschulen und anderen Organisationen gehören; sie können als *lokaler Internet-Bereich* angesehen werden. Die mittlere Schicht

bilden regionale Netze mit regionalen Internetdienstanbietern, die sogenannten ISPs (*Internet Service Provider*). Diese Schicht stellt den regionalen Internet-Bereich dar. Bei den regionalen Netzen handelt es sich in der Regel um Hochgeschwindigkeitsnetze innerhalb von Großstädten, weshalb man sie als *Metro-Netze* bzw. *City-Netze* bezeichnet, die heute von häufig lokalen Netz-Providern zur Verfügung gestellt werden. Die obere Schicht, die den Internet-Weitverkehrsbereich darstellt, bilden nationale und internationale Hochgeschwindigkeitsnetze mit nationalen bzw. internationalen ISPs. Die nationalen und internationalen Hochgeschwindigkeitsnetze werden miteinander gekoppelt und bilden das *Internet-Backbone*, auch *Internet-Core* genannt.

Jeder ISP stellt einen Internetzugangspunkt dar, der auch als Einwahlknoten bzw. als POP (*Point of Presence*) bezeichnet wird.

1.1.4 Meilensteine der Internet-Entwicklung und Trends

Das Internet hat sich unmittelbar nach der Etablierung des WWW rasant entwickelt und adaptiert sich mit einem hohen Tempo an den Bedarf der Nutzer stetig weiter. Dies möchten wir jetzt in kurzer Form näher zum Ausdruck bringen. Hierfür zeigt Abb. 1.1-6 wesentliche Meilensteine bisheriger Internet-Entwicklung seit 1990 sowie wichtige Entwicklungstrends.

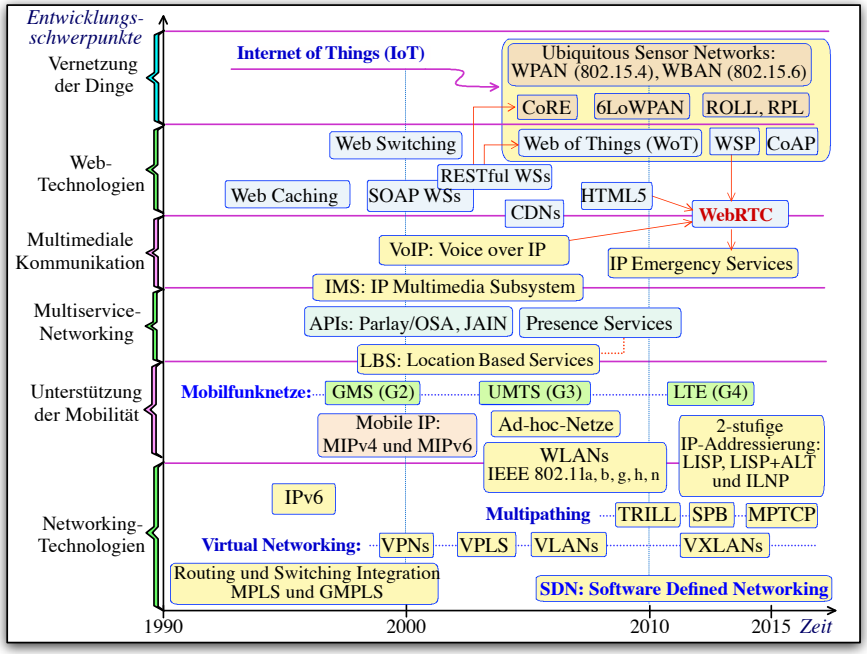


Abb. 1.1-6: Internet und IP-Netze; Meilensteile der bisherigen Entwicklung und Trends
CoRE: Constrained RESTful Environment, G: Generation, ROLL: Routing over Low power and Lossy neworks, RPL: Routing Protocol for Low power and Lossy networks, WPAN: Wireless Personal Area Network, WBAN: Wireless Body Area Network, WS: Web Services, WSP: WebSocket Protocol

Meilensteine bei Networking-Technologien

Integration von Routing und Switching

In lokalen Netzwerken wurden bereits zu Beginn der 90er Jahre sowohl Router als auch Switches eingesetzt und man hat damals von Routing und *Switching Integration* gesprochen. Diese Integration hat auch im Backbone des Internet und in großen privaten IP-Netzen stattgefunden. Folglich hat man versucht, die beiden Techniken Routing und Switching in einer Netzwerkkomponente (als Multi-Layer-Switch bezeichnet) zu integrieren. Ein besondere Art der Integration von Routing und Switching liegt dem Konzept MPLS (*Multi-Protocol Label Switching*) zugrunde [Abb. 1.4-4]. MPLS wird in Abschnitt 12.2 beschrieben.

MPLS, GMPLS

Bei MPLS werden die IP-Pakete über ein Netz quasi im Gänsemarsch übermittelt [Abb. 11.1-1]. Dadurch entsteht die Möglichkeit, die Dienstgüte (*Quality of Service*) auf einem geforderten Level zu garantieren. Dies ist für die multimediale Kommunikation von enormer Bedeutung. Um das MPLS-Konzept in optischen Netzen, in denen man WDM (*Wavelength Division Multiplexing*) verwendet, einsetzen zu können, wurde GMPLS (*Generalized MPLS*) entwickelt [Abschnitt 12.3]. In den 90er Jahren hat man bei IP-Netzen mit (G)MPLS von *Next Generation IP Networks* gesprochen.

IPv6

Als Meilenstein in der Internet-Entwicklung kann das in 1994 spezifizierte Konzept von IPv6 angesehen werden. Es sei hervorgehoben, dass damals die Knappheit von offiziellen IPv4-Adressen die treibende Kraft der Entwicklung von IPv6 war. Mitte der 90er Jahre wurde aber die als NAT (*Network Address Translation*) bezeichnete Möglichkeit 'entdeckt', die privaten IPv4-Adressen nutzen zu können [Abschnitt 3.3]. Das NAT-Konzept, insbesondere dessen Variante PAT (*Port Address Translation*), hat dazu beigetragen, dass man IPv6 in der Tat damals (und sogar bis Ende des ersten Jahrzehnts dies Jahrhunderts) noch nicht unbedingt gebraucht hat. Die Ära von IPv6 hat erst 'richtig' nach 2010 begonnen; unmittelbar nachdem die letzten offiziellen IPv4-Adressen vergeben wurden.

Virtual Networking

Schon in der zweiten Hälfte der 90er Jahre hat man mit *Virtual Networking* begonnen. Physikalische Leitungen im WAN-Bereich wurden durch virtuelle Verbindungen ersetzt, und es entstanden *Virtual Private Networks* (VPNs [Abschnitt 13.1]). Bereits zu Anfang dieses Jahrhunderts konnte man schon mehrere Ethernet-Segmente dank des (G)MPLS-Einsatzes über virtuelle Leitungen an einen zentralen Ethernet-Switch (Layer-2-Switch [Abb. 15.2-2]) anbinden und auf diese Weise ein standortübergreifendes, verteiltes Ethernet einrichten; damit wurde VPLS (*Virtual Private LAN Service* [Abschnitt 13.4) geboren. Etwa zur gleichen Zeit bildete man IP-Subnetze in lokalen Netzwerken als beliebige Gruppen von Rechnern und bezeichnete diese Gruppen als VLANs (*Virtual LANs*). Durch die Virtualisierung von Rechnern besteht heute – theoretisch gesehen – dank dem Konzept VXLAN (*Virtual Extensible LAN*) die Möglichkeit, aus virtuellen Rechnern (*Virtual Machines*) bestehende VLANs sogar weltweit zu bilden.

Multipathing

Um parallele, über mehrere Datenpfade verlaufende Kommunikation zwischen Rechnern, insbesondere in Datacentern, zu ermöglichen, wurden hierfür die Konzepte TRILL (*TRansparent Interconnection of Lots of Links*), SPB (*Shortest Path Bridging*) und MPTCP (*Multipath TCP* [Abschnitt 6.5]) entwickelt. Mit TRILL und SPB können standortübergreifende VLANs gebildet werden. Mittels SPB kann auch ein verteilter,

virtueller Ethernet-Switch auf Basis eines Ethernet-basierten Netzwerks – sogar eines standortübergreifenden Netzwerks – eingerichtet werden und die an diesem virtuellen Ethernet-Switch angeschlossenen Rechner können auch zu verschiedenen VLANs zugeordnet werden.

Technologien zur Unterstützung der Mobilität

Der Einsatz tragbarer Rechner (insbesondere Laptops und Smartphones) hat dazu geführt, dass von der IEEE mehrere Standards für WLANs (*Wireless LANs*) im ersten Jahrzehnt dieses Jahrhunderts spezifiziert wurden. Ein WLAN ermöglicht aber nur eine räumlich beschränkte Mobilität von tragbaren Rechnern. Heutzutage werden jedoch oft in Wirt-Servern mit zahlreichen virtuellen Rechnern mehrere, aus den in ihnen eingerichteten virtualisierten Rechnern bestehende, virtuelle Netzwerke gebildet; sie werden oft als *Clouds* bezeichnet. Hierfür sind Konzepte nötig, um eine aus mehreren virtuellen Rechnern enthaltene Cloud weltweit transferieren und an verschiedenen Standorten einsetzen zu können, ohne dass die IP-Adressen von Rechnern in der Cloud geändert werden müssen. Um diese Traumvorstellung zu verwirklichen, ist eine neue zweistufige, flexible IP-Adressierung notwendig. Wie diese zu realisieren und zu nutzen ist, beschreiben die in Abschnitt 14.7 präsentierten Konzepte LISP (*Locator/ID Separation Protocol*), LISP+ALT (*LISP Alternative Logical Topology*) und ILNP (*Identifier-Locator Network Protocol*).

Software Defined Networking

Die Virtualisierung von Rechnern und der zunehmende Bedarf an flexiblen, spontanen und an Geschäftsprozesse angepassten IT-Diensten verlangen neue Ideen zur variablen und raschen Bereitstellung von Netzwerkdiensten. *Software Defined Networking* (SDN) stellt eine solche Idee dar und ist als enorm wichtiger Entwicklungstrend im Internet und in Netzwerken mit IP zu betrachten. SDN ermöglicht die Bereitstellung universeller und programmierbarer Netzwerkknoten zur Weiterleitung von Daten. Diese Netzwerkknoten können fast alle denkbaren Netzwerkfunktionen erbringen – und dies sogar parallel für die beiden Internetprotokolle IPv4 und IPv6. Dadurch können beim SDN verschiedene programmierbare Netzwerkdienste (*Programmable Network Services*) realisiert werden. Folglich kann man beim SDN sogar von Netzwerkprogrammierbarkeit (*Network Programmability*) sprechen.

Unterstützung der Mobilität

MIPv4 und MIPv6

Die Unterstützung der Mobilität im Internet und in IP-Netzen ist ein bedeutendes Thema schon seit Beginn der Internet-Ära. Bereits Mitte 90er Jahre wurden die beiden Protokolle MIPv4 (*Mobile IPv4*) und MIPv6 (*Mobile IPv6*) konzipiert, um die Mobilität von Rechnern zwischen IP-Subnetzen zu ermöglichen. Kapitel 15 widmet sich der Unterstützung der Mobilität in IP-Netzen mit MIPv4 und MIPv6.

UMTS und LTE und WLANs

Erst die Mobilfunknetze der 3-ten und 4-ten Generation (G3 und G4), d.h. UMTS (*Universal Mobile Telecommunications System*) als G3 und LTE (*Long Term Evolution*) als G4, haben dazu beigetragen, dass verschiedene Arten von Smartphones heute als multifunktionelle Endgeräte am Internet dienen. Durch die breite Einführung von WLANs und die flächendeckende Verfügbarkeit von UMTS- bzw. von LTE-Diensten wurde das Problem 'Internet unterwegs mit Laptops, Tablets und Smartphones' gelöst. Als offenes Problem gilt aber noch die Mobilität kleiner Netzwerke und zwar so, dass die Adressen von Rechnern in diesen Netzwerken an jedem neuen Internetzugang

nicht geändert werden müssen. Dieses Problem soll mit 2-stufiger IP-Adressierung gelöst werden [Abschnitt 14.7].

Es werden auch Konzepte entwickelt, um *Ad-Hoc-Netzwerke*, in denen sowohl die Knoten als auch die Endsysteme mobil sind, verwirklichen zu können. Diese Netzwerke haben eine große Bedeutung, da sie es spontan ermöglichen, fahrende Autos bis zu einer bestimmten Entfernung untereinander zu vernetzen: *Car-to-Car-Networks* (*C2C Networks*). Ad-Hoc-Netzwerke

Multiservice-Networking

Als wichtiger Trend bei IP-Netzen am Ende der 90er Jahre war das *Multiservice-Networking*, der die *Integration* (*Konvergenz*) der Netze aufgegriffen hat. Da verschiedene TK-Netze (wie PSTN, ISDN, GSM, UMTS) schon damals zu konvergieren begannen, Stand der Wunsch im Raum, alle TK-Netze seitens des Internet als ein heterogenes TK-Netz zu nutzen, intelligente Netzdienste auf Basis des Protokolls IP zu entwickeln und diese den Teilnehmern an allen TK-Netzen über das Internet zugänglich zu machen. Konvergenz der Netze

Um intelligente Netzdienste auf Basis eines TK-Netzes zu entwickeln, muss man jedoch auf bestimmte Software-Schnittstellen, APIs (*Application Programming Interface*), im Netzkern zugreifen. Da diese noch in den 90er Jahren nur für den Netzbetreiber zugänglich waren, war es damals nicht möglich, dass die Netzdienste durch Dritte, also durch die Nicht-Netzbetreiber, konzipiert und entwickelt werden konnten. Um dies zu ändern, wurde zu mit Beginn dieses Jahrhunderts ein vom Netz unabhängiges API entwickelt, über das man auf die Dienste wichtiger TK-Netze zugreifen kann (siehe Abschnitt 1.4 in [Bad22]). Es handelt sich um Parlay/OSA (*Open Service Architecture*).

Idee von Parlay/OSA: Die grundlegende Idee von Parlay/OSA besteht darin, dass man verschiedene TK-Netze als Kernnetz betrachtet. Die Dienste dieses Kernnetzes sind über Parlay/OSA API für die Nicht-Netzanbieter zugänglich. Somit können sie Netzanwendungen entwickeln und auf speziellen Application-Servern installieren, sodass man auf diese über das Internet zugreifen kann. Parlay/OSA

Ein ähnliches Konzept wie bei Parlay/OSA wurde auch bei JAIN (*Java API for Integrated Networks*) von der Firma Sun Microsystems (jetzt Oracle) verfolgt. JAIN

Bei der Nutzung von Parlay/OSA kann man beliebige Netzdienste – z.B. auf Basis von UMTS bzw. von LTE – entwickeln und sie über das Internet zugänglich machen. Da die Lokation von mobilen Benutzern in Mobilfunknetzen UMTS und LTE mit einer bestimmten Genauigkeit bekannt ist, sind *Location Based Services* (LBS) realisierbar; und diese bilden die Grundlage für Presence Services. Die Einsatzmöglichkeiten von *Presence Services* sind sehr breit und haben in sozialen Netzwerken (z.B. Facebook) eine wichtige Funktion; sie ist aber nicht immer vorteilhaft. LBS und Presence Services

Bei LBS und Presence Services spielt das IMS (*IP Multimedia Subsystem*) eine wichtige Rolle. IMS ermöglicht es, die Server der den Nicht-Netzanbieter, am UMTS bzw. am LTE zu installieren und verschiedene Multimedia Services (Spiele, Filme, ...) zum Abruf per Internet anzubieten. IMS

Multimediale Kommunikation

VoIP ist heute MMoIP

Mit Multiservice-Networking hängt auch die Realisierung der multimedialen Kommunikation zusammen. Die Entwickler haben seit geraumer Zeit davon geträumt, über ein Netz zu verfügen, über welches man alle Informationsarten (Audio, Video und Daten) übermitteln könnte. Die Konzepte und Protokolle für VoIP (*Voice over IP*) sind ein wichtiger Schritt in diese Richtung. In der Wirklichkeit ist VoIP mit dem Signalisierungsprotokoll SIP (*Session Initiation Protocol*) nicht nur VoIP, sondern *Multi-Media over IP* (MMoIP [Abb. 6.3-4]).

IP-Radio und IP-Fernsehen mit CDNs

Durch die Einführung der geeigneten Protokolle wie z.B. IP-Multicasting [Abb. 10.6-1] sind Dienste wie IP-Radio und IP-Fernsehen realisierbar. Bereits seit 2005 spricht man von *Triple Play*. Darunter versteht man das gebündelte Angebot der drei Dienste *Internet*, *IP-Telefonie* (VoIP) und *Fernsehen* für private Haushalte. Bei zeitversetztem Abruf der Echtzeitsendungen (wie Radio- bzw. Fernsehsendungen) spielen Webbasierte *Content Delivery Networks* (CDNs) mit zahlreichen Lieferungsservern eine Schlüsselrolle (siehe Kapitel 10 in [BRS03]). Und zwar bestimmt ein *Redirect-Router* – nach der Lokation des die Echtzeitsendung abrufenden Rechners – den Lieferungsserver, aus welchem (möglichst nicht weit gelegen vom abrufenden Rechner) die gewünschte Echtzeitsendung ausgeliefert werden soll, damit man eine gute Qualität gewährleisten kann.

IP Emergency Services

Eine wichtige Funktion herkömmlicher, öffentlicher Netze für die Sprachkommunikation ist die von ihnen angebotene Möglichkeit, in einem Notfall einen Notruf (*Emergency Call*) abzusetzen. Die Netze für die Sprachkommunikation bieten daher die Notrufdienste an; z.B. unter den Notrufnummern 110 für die Polizei und 112 für die Feuerwehr und die Rettungsdienste. Diese Dienste – und auch zahlreiche ähnliche – müssen zukünftig auch in öffentlichen VoIP-Systemen, also in der Tat im Internet, angeboten werden; hierbei spricht man von *IP Emergency Services* bzw. von *VoIP Emergency Services*.

LoST als 'Bruder' von DNS

Verschiedene Organisationen und Standardisierungsgremien sind in dieser Hinsicht aktiv. So wurde bei der IETF beispielsweise eine Arbeitsgruppe namens *Emergency Context Resolution with Internet Technologies* (ECRIT) ins Leben gerufen, um die Konzepte und Protokolle für die Realisierung von IP Emergency Services zu spezifizieren. Eine wichtige Funktion in Notrufsystemen im Internet besteht in der Ermittlung der IP-Adresse der richtigen Notrufleitstelle – und zwar aufgrund des in Form von URN (*Uniform Resource Name*) dargestellten Geschehens, d.h. was ist passiert, und des Standorts des Geschehens. Um diese Funktion sicher zu realisieren, wurde das Konzept LoST (*Location-to-Service Translation*) entwickelt [RFC 5222]. LoST stellt eine hierarchische, baumartige Vernetzung von Rechnern dar (vergleichbar dem DNS) und kann folglich als jüngster Bruder vom DNS betrachten werden.

Web-Technologien

Web-Caching

In der ersten Phase der Web-Ära hat man hauptsächlich auf Webserver zugegriffen, um verschiedene Webinhalte in Form von Websites herunterzuladen. Um die Webinhalte, welche in der Zeit unverändert bleiben, nicht erneut über lange Strecken übermitteln zu müssen und schneller liefern zu können, hat man das in Rechnern gut bewährte Caching-Prinzip für das Internet übernommen – und so wurde Web-Caching 'geboren',

siehe hierzu Kap. 8 in [BRS03]. Für das Web-Caching werden in der Regel spezieller Rechner als *Web-Cache-Server* eingesetzt. Große Internet Service Provider setzen mehrere Web-Cache-Server ein, sodass ein vernetztes Web-Caching-System entsteht. Für die Kommunikation zwischen Web-Caches wurde ICP (*Internet Cache Protocol*) entwickelt.

Web-Switching

Stark gefragter Webcontent wird schon lange nicht mehr nur auf einem Webserver abgespeichert, sondern auf mehreren Webservern, die sogar weltweit verteilt sein können. Diese Webserver bilden eine Gruppe von Servern, die unter einer IP-Adresse erreichbar sein muss und als *verteilter Webserver* angesehen werden kann. Als technische Basis dafür dient das Ende der 90er Jahre entwickelte *Web-Switching* und darunter versteht man die Verteilung von aus dem Internet kommenden Anfragen gemäß dem gefragten *Webcontent* auf mehrere Webserver. Web-Switching bildet heute die Grundlage für E-Commerce-Geschäfte; für Näheres darüber siehe Kap. 7 in [BRS03].

Web Services

Das Internet wurde zuerst hauptsächlich dafür benutzt, um per Browser den Zugriff auf verschiedene Informationen und Applikationen zu ermöglichen. Ende 90er Jahre hat man aber erkannt, das Internet sich auch als universelle Plattform für die Kommunikation zwischen verteilten Anwendungen eignet und die Idee, webbasierte Dienste durch die Vernetzung von verteilten Anwendungen zu realisieren, hat zur Entstehung von *Web Services* geführt. Ein Web Service ist ein Dienst auf Basis des Internet und des Protokolls HTTP, der durch die Vernetzung von verteilten Anwendungen und den Einsatz von XML (*eXtensible Markup Language*) zur Bildung von 'Nachrichten' – genauer von XML-Nachrichten – mit den zwischen Anwendungen zu übertragenden Daten erbracht wird. Jeder Web Service kann über einen Verzeichnisdienst veröffentlicht werden, um ihn bekannt, auffindbar und damit aufrufbar zu machen (vgl. Kapitel 11 in [BRS03]).

SOAP WS und RESTful WS

Es sei angemerkt, dass man zuerst bei *Web Services* (WS) das Protokoll SOAP (*Simple Object Access Protocol*) verwendet hat, um XML-Nachrichten in HTTP-Requests und -Responses zu übermitteln. Der Einsatz von SOAP bringt allerdings einen Nachteil mit: Die Web-Ressourcen (Objekte) können nicht direkt adressiert werden. Somit wurde später das Transferprinzip REST (*REpresentational State Transfer*) bei Web-Services eingesetzt, so dass Web-Ressourcen direkt mit URLs adressierbar sind. Vergleichbar, wie es ursprünglich bei Einführung des WWW vorgesehen war. Aus diesem Grund unterscheidet man zwischen SOAP-basiertem WS (*SOAP WS*) und REST-basiertem WS (*RESTful WS*).

CDNs, Request-, Content-Routing

Das klassische Modell der Webdienste, bei dem der Webcontent von einem Ursprungs-Server aus weltweit auf jede Webanfrage hin an den Benutzer geschickt wird, ist in einigen Situationen nicht mehr praktikabel, was besonders zeitkritischen Content, also beim Streaming-Media (Video, Internet-TV, -Spiele etc.) betrifft. Damit man Streaming-Media in guter Qualität den Benutzern liefern konnte, ist Anfang dieses Jahrhunderts die Idee für *Content Delivery Networks* (CDNs) entstanden. Die Webserver mit dem gleichen zeitkritischen Content, sind jetzt nicht immer an einem Standort, sondern werden weltweit verteilt. In diesem Falle muss der 'günstigste' Webserver ausgewählt und die Anfrage an ihn gerichtet werden. Diesen Vorgang nennt man

Request-Routing oder auch *Content-Routing*. Näheres über CDN findet sich in Kapitel 10 von [BRS03].

WebRTC – eine richtungsweisende Idee

Mit dem zweiten Jahrzehnt in diesem Jahrhundert versucht man eine richtungsweisende Idee zu verwirklichen, die sogenannten WebRTC (*Web Real-Time Communication*), eine Art *Web Video Telephony*. Diese Idee besteht darin, multimediale Echtzeitkommunikation mithilfe von HTML5-fähigen Webbrowsern einfach zu realisieren, ohne dafür zusätzliche Softwaremodule installieren zu müssen. Zur Unterstützung von WebRTC können bei Bedarf von einem Webserver verschiedene RTC-spezifische Funktionsmodule heruntergeladen werden und im Webbrowser diese (quasi automatisch) einzubauen. Bei WebRTC kann ein spezieller Server um RTC-Funktionen erweitert werden (hierzu eignet sich jeder Webserver) und als Manager von multimedialen Verbindungen zwischen Browsern dienen. Es ist zu erwarten, dass eine große Akzeptanz von WebRTC in Smartphones, Tablets und in Smart-TV in der Zukunft zu großen Veränderungen der heutigen Kommunikationslandschaft führen wird und die Videotelefonie per Smart-TV nur eine Frage der Zeit ist. Ferner besitzt WebRTC einen bedeutenden Einfluss auf die Realisierung von *IP Emergency Services*.

WebSockets

Für die Realisierung von WebRTC wurde das *WebSocket Protocol* (WSP) entwickelt [RFC 6455]. WSP wird auch beim Einsatz von Webtechnologien zur Vernetzung verschiedener 'Dinge' mit dem Internet eingesetzt; man spricht hierbei von *Web of Things* (WoT).

Vernetzung der Dinge – Internet of Things

USNs, IoT und WoT

Das Streben insbesondere nach mehr Energieeffizienz, mehr Lebensqualität und besserer Umweltüberwachung führt dazu, dass verschiedene Systeme zur drahtlosen Vernetzung von Sensoren und Aktoren – in der Tat zur Vernetzungen aller möglichen 'Dinge' — ständig und immer mehr an Bedeutung und an Verbreitung gewinnen; folglich entstehen *Sensornetze/Sensornetzwerke*. Weil Sensornetze überall und jederzeit zum Einsatz – z.B. zur Industrie-/Gebäude-/Heimautomation, Gesundheits-/Umweltüberwachung – kommen können, werden sie als *ubiquitäre Sensornetze* bzw. kurz als USNs (*Ubiquitous Sensor Networks*) bezeichnet[1]. USNs sind sehr stark ressourcenbeschränkt, insbesondere *energiearm* und *verlustbehaftet*; demzufolge spricht man von *Constrained Networks* bzw. von LLNs (*Low power and Lossy Networks*).

Internet of Things, Web of Things

Die Anbindung von USNs an das 'heutige' Internet führt zur Entstehung eines neuen Internetteils, welcher als *Internet of Things* (IoT) – also *Internet der Dinge* – bezeichnet wird [ITU-T Y.2060]. Im IoT kommen auch einige Web-Technologien zum Einsatz, sie müssen an die Besonderheiten von Sensornetzen angepasst werden: *Web of Things* (WoT); siehe hierzu ITU-T Y.2063.

Sensornetze werden oft nach IEEE-Standards IEEE 802.15.4 (WPAN, *Wireless Personal Area Network*) und `IEEE 802.15.6` (WBAN, *Wireless Body Area Network*) aufgebaut und dabei wird IPv6 eingesetzt. Dadurch kann eine global eindeutige IPv6-Adresse jedem Sensor/Aktor zugewiesen werden und er ist dann über das Internet zugreifbar.

[1] Da in Sensornetzen sowohl (passive) Sensoren wie auch als Ausführungsorgane aktiv fungierende Aktoren vorhanden sind, sprechen wir allgemein von *Sensor-Aktor-Netzen/Netzwerken*.

6LoWPAN

Um IPv6 in Sensornetzen auf Basis von WPANs nach `IEEE 802.15.4` einsetzen zu können, wurde das Konzept 6LoWPAN (*IPv6 over Low-power PAN*) bei der IETF in RFC 4944 spezifiziert. 6LoWPAN gilt bereits als ein effizientes Konzept für den Einsatz von IPv6 in Sensornetzen.

CoRE und CoAP

Im IoT sollen REST-basierte Web Services, auch als *RESTful WSs* bezeichnet, zum Einsatz kommen. Diese müssen jedoch an ressourcenbeschränkte Umgebungen (*constrained environments*) in Sensornetzen adaptiert werden. Die Umsetzung dieser Anforderung hat sich die IETF Working Group CoRE (*Constrained RESTful Environments*) vorgenommen. Da das normale Webprotokoll HTTP in Sensornetzen praktisch nicht einsetzbar ist, wird dieses in Sensornetzen durch das neue, von der Working Group CoRE entwickelte Webprotokoll CoAP (*Constrained Application Protocol*) ersetzt.

ROLL und RPL

Sensornetze als LLNs können auch beliebig verteilte Strukturen mit intelligenten Knoten bilden, daher ist auch ein Routing-Protokoll in diesen Netzen nötig. Auf dem Gebiet *Routing over LLNs* ist die IETF Working Group ROLL (*Routing Over Low power und Lossy networks*) aktiv, und das Ergebnis ist u.a. das Routing-Protokoll RPL (*IPv6 Routing Protocol for Low-Power und Lossy Networks*).

Zukünftige Entwicklungen in Kapitel 18

Die weitere Entwicklung der Internet-Technologien kann der geneigte Leser Kapitel 18 entnehmen, wo die Themen in einzelnen Abschnitten unter Verweis auf über 600 Online-Quellen nachverfolgt werden können.

1.2 Funktionen der Kommunikationsprotokolle

Fehlerursachen

In einem Netz können die zu übertragenden Daten verfälscht werden. Die Ursachen dafür sind meist auf die schlechte Qualität des Übertragungsmediums zurückzuführen. Eine Verfälschung der Daten kann auch durch äußere Einflüsse wie etwa starke elektromagnetische Felder in der Umgebung oder durch das *Nebensprechen* entstehen. Übertragungsstörungen führen nicht nur zu einer Datenverfälschung, sondern sogar zu einem Datenverlust. Um dies zu vermeiden, müssen entsprechende Funktionen in den Kommunikationsprotokollen enthalten sein. Diese Funktionen lassen sich in drei Gruppen aufteilen:

- **Fehlerkontrolle** (*Fault Control*),
- **Flusskontrolle** (*Flow Control*) und
- **Überlastkontrolle** (*Congestion Control*).

Datenverfälschungen und -verluste

Die *Fehlerkontrolle* umfasst alle Maßnahmen in einem Kommunikationsprotokoll, mit denen Datenverfälschungen und -verluste während der Übertragung entdeckt und beseitigt werden können. Die *Flusskontrolle* bedeutet eine gegenseitige Anpassung der Sende- und der Empfangsseite in Bezug auf die übertragene Datenmenge. Die *Überlastkontrolle* betrifft alle Vorkehrungen, die dazu dienen, ein Netz nicht zu überlasten. Bei der Überlastung eines Netzes müssen die übertragenen Datenblöcke oft

verworfen werden und die Verweilzeit von Datenblöcken im Netz durch 'Staus' in Knoten nimmt stark zu. Im Folgenden werden diese Funktionen näher erläutert.

1.2.1 Prinzipien der Fehlerkontrolle

Die Fehlerkontrolle hat die Aufgabe, jede fehlerhafte Situation während der Datenübertragung zu entdecken und zu beseitigen. Sie ist Bestandteil jedes Kommunikationsprotokolls und wird beim Empfänger mittels festgelegter Quittungen (Bestätigungen) und beim Sender durch die Zeitüberwachung realisiert. Im Weiteren werden alle möglichen Fehlersituationen dargestellt und notwendige Maßnahmen zu ihrer Beseitigung aufgezeigt.

Erste 'eiserne Regel'

Allen Kommunikationsprotokollen liegen zwei 'eiserne Regeln' zugrunde:

> Datenblöcke können während der Übertragung verfälscht werden. Deshalb muss nach dem Absenden jedes Datenblocks dieser im Speicher der Quellstation für den Fall gehalten werden, falls eine wiederholte Übermittlung notwendig ist.

Negative Auswirkungen infolge der Verfälschung von übertragenen Datenblöcken können durch die Umsetzung dieser Regel und durch eine wiederholte Übermittlung ausgeglichen werden. Abb. 1.2-1a zeigt die fehlerlose Übermittlung eines Datenblocks. Diese wird von der Empfangsseite positiv quittiert (bestätigt) und eine Kopie des Datenblocks in der Quellstation gelöscht. Auch eine Quittung stellt einen kurzen, vom Protokoll festgelegten Datenblock dar.

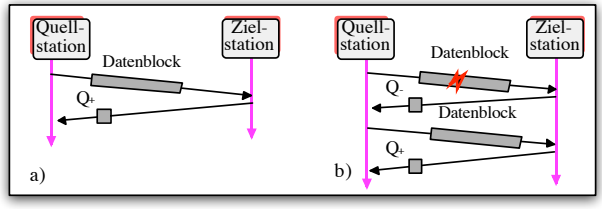


Abb. 1.2-1: Übermittlung eines Datenblocks: a) fehlerlos, b) fehlerhaft
Q+: positive Quittung, Q–: negative Quittung

Negative Quittung bei Störungen

In Abb. 1.2-1b tritt bei der Übermittlung des Datenblocks eine Störung auf, was eine negative Quittierung zur Folge hat. Der gestörte Datenblock wird durch die Zielstation einfach verworfen. Da in der Quellstation eine Kopie des betreffenden Datenblocks gehalten wird, sendet die Quellstation den gleichen Datenblock noch einmal – diesmal fehlerfrei – zu der Zielstation, die ihn positiv quittiert. Die Kopie des übertragenen Datenblocks kann nun in der Quellstation gelöscht werden.

Verfälschung von Quittungen

Eine besondere Situation entsteht dadurch, dass nicht nur die Datenblöcke während der Übertragung verfälscht werden können, sondern auch die Quittungen. Wird eine positive Quittung so verfälscht, dass die Quellstation sie als negative Quittung interpretiert, führt dies zu einer unnötigen wiederholten Übermittlungen des betreffenden Datenblocks und zur Verdoppelung von Daten am Ziel.

Der schlimmste Fall (*worst case*) bei der Übermittlung eines Datenblocks entsteht dann, wenn sowohl der übertragene Datenblock als auch dessen negative Quittung verfälscht werden. Wie Abb. 1.2-2 zeigt, empfängt die Quellstation in diesem Fall eine positive Quittung und könnte deshalb die Kopie des Datenblocks löschen. Dies würde aber zum Verlust des Datenblocks führen. Um einen solchen Fall zu bewältigen, müssen die Kommunikationsprotokolle zwei Stufen der Fehlerkontrolle realisieren. Die hier angesprochene Fehlerkontrolle bezieht sich nur auf die Übermittlung einzelner Datenblöcke, die oft aufgrund der Segmentierung von zu übertragenden Dateien entstehen. Die Fehlerkontrolle muss auch auf Dateiniveau realisiert werden. Die einzelnen Datenblöcke, die zu einer Datei gehören, werden in der Zielstation zu einer Datei zusammengesetzt. Ist ein Datenblock der Datei in der Zielstation nicht vorhanden, sendet sie eine negative Quittung, die sich auf diese Datei bezieht. Die Quellstation muss dann entweder den verloren gegangenen Datenblock oder sogar die ganze Datei nachsenden.

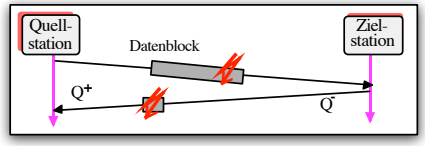


Abb. 1.2-2: Worst case einer Datenmittlung: Daten und Quittung werden verfälscht

Die zweite 'eiserne Regel', die bei allen Kommunikationsprotokollen realisiert werden muss, um Datenverluste während der Übertragung zu erkennen, lautet: Zweite 'eiserne Regel'

> Datenblöcke können bei der Übertragung verloren gehen, sodass man nur eine begrenzte Zeit auf eine positive oder negative Quittung für einen Datenblock warten soll.

Dies muss über eine Zeitüberwachung realisiert werden, um Verluste von Datenblöcken während der Übertragung zu erkennen. Dazu ist im Protokoll eine maximale Wartezeit auf eine Quittung festzulegen. Eine solche Wartezeit wird auch als *Timeout* bezeichnet und kann als 'Geduldzeit' interpretiert werden. Nach dem Absenden eines Datenblocks muss die Überwachung der maximalen Wartezeit auf die Quittung aktiviert werden. Es stellt sich die Frage, wann die Datenblöcke während der Übermittlung eigentlich verloren gehen. Dass ein übertragener Datenblock bei einem plötzlichen Bruch der Leitung verloren geht, ist selbstverständlich, doch das ist selten der Fall. Die häufigste Ursache für den Verlust eines Datenblocks ist eine Verfälschung in seinem Header oder Trailer, sodass er auf der Leitung nicht vollständig erkannt und damit in der Zielstation nicht aufgenommen werden kann.

Abb. 1.2-3 illustriert die fehlerhafte Situation, in der ein Datenblock verloren gegangen ist. Nach dem Absenden des Datenblocks wird die 'Geduldzeit' überwacht. Kommt innerhalb dieser Zeit keine Quittung an, interpretiert dies die Quellstation als verloren gegangenen Datenblock und wiederholt die Übermittlung. Nach dem wiederholten Absenden kommt eine positive Quittung noch während der 'Geduldzeit' an und die Kopie des Datenblocks kann dann gelöscht werden. Geduldzeit

Nummerierung von Datenblöcken

Auch eine Quittung kann verloren gehen. Wie Abb. 1.2-3b zeigt, wird dies ebenfalls mit Hilfe der Zeitüberwachung erkannt. In einem solchen Fall kann ein Datenblock in der Zielstation doppelt vorhanden sein. Deswegen muss für die Zielstation klar werden, dass es sich nicht um einen neuen Datenblock handelt, sondern um eine wiederholte Übermittlung. Werden die transferierten Datenblöcke nicht nummeriert, kann das zur Verdopplung von Daten am Ziel führen. Derartige Datenverdopplungen lassen sich mit der Nummerierung von Datenblöcken ausschließen. Aus diesem Grund werden bei allen Kommunikationsprotokollen die übertragenen Datenblöcke nummeriert.

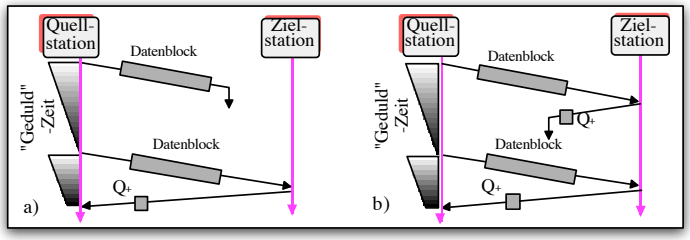


Abb. 1.2-3: Fehlerhafte Übermittlung: a) Datenblockverlust, b) Quittungsverlust

Modulo-Verfahren

Anmerkung: Bei einer fortlaufenden Nummerierung der übertragenen Datenblöcke kann die Zielstation erkennen, ob es sich um eine wiederholte Übermittlung handelt und somit einen doppelt vorhandener Datenblock entdecken. Eine Nummerierung der übertragenen Datenblöcke besteht darin, dass jedem Datenblock eine bestimmte Sequenznummer zugeteilt wird. Diese Nummerierung kann aber nicht beliebig fortgesetzt werden. Ursache hierfür ist die begrenzte Anzahl von Bit für die Nummernabspeicherung im Header des Datenblocks und deshalb werden die Datenblöcke nach dem Modulo-Verfahren nummeriert. In den meisten Fällen wird die Nummerierung nach dem Modulo 8 oder 128 realisiert. Bei Modulo 8 werden die einzelnen Datenblöcke von 0 bis 7 gekennzeichnet und verschickt. Ist die 7 als letzte Nummer vergeben worden, wird der Zähler zurückgesetzt und die Nummerierung startet bei 0. Äquivalent dazu funktioniert die Nummerierung nach dem Modulo-128-Verfahren, bei dem die Nummern bis 127 vergeben werden.

Bei der Nummerierung von Datenblöcken kann eine Gruppe von empfangenen Blöcken gleichzeitig durch die Zielstation quittiert werden, um damit die Verkehrslast im Netz durch eine geringere Anzahl von Quittungen zu reduzieren.

Nummerierungsfenster (Window)

Beim Aufbau einer Verbindung muss sichergestellt sein, dass die Quellstation den Datenblöcken jene Sequenznummern zuteilt, die auch von der Zielstation erwartet werden. Aus diesem Grund ist zu vereinbaren, welchen Zahlenwert das Nummerierungsfenster hat und mit welcher Sequenznummer bei der Übermittlung der Datenblöcke begonnen wird.

1.2.2 Realisierung der Flusskontrolle

Bedeutung der Flusskontrolle

Bei der Datenkommunikation tritt häufig der Fall ein, dass die Daten beim Sender rascher 'produziert' werden, als der Empfänger sie 'konsumieren' kann. So ist eine Situation vorstellbar, in der ein Großrechner im Netz eine große Menge von Daten an einen entfernten kleinen Drucker übermittelt. Der Großrechner muss, um ein Überfließen des Druckerspeichers zu verhindern, die Menge der zu übertragenden Daten

der Aufnahmefähigkeit des Druckers anpassen. Die Anpassung muss durch entsprechende Kommandos vom Drucker gesteuert werden. Dieses einfache Beispiel weist auf die Bedeutung der gegenseitigen Abstimmung zwischen Quell- und Zielstation in Bezug auf die Menge der zu übertragenden Daten hin.

Ziel der Flusskontrolle

Unter *Flusskontrolle* versteht man alle Maßnahmen, die zur Anpassung der gesendeten Datenmenge der Quellstation an die Aufnahmekapazität der Zielstation führen. Die Flusskontrolle kann realisiert werden

- mittels der Meldungen *Halt* und *Weitersenden*,
- über einen *Kreditmechanismus* (Kredite), oder
- vermöge eines *Fenstermechanismus* (Window).

Meldungen: Halt, Weitersenden

Die einfache Flusskontrolle mit Hilfe der Meldungen *Halt* und *Weitersenden* verläuft wie folgt: Stellt der Empfänger fest, dass er nicht mehr in der Lage ist, die empfangenen Daten aufzunehmen, schickt er dem Sender die Meldung *Halt*. Der Sender ist nach dem Empfang von Halt verpflichtet, das Senden von Daten einzustellen, bis der Empfänger die Meldung *Weitersenden* übermittelt und damit den *Halt*-Zustand aufhebt. Ein Nachteil dieses einfachen Verfahrens besteht darin, dass eine Verfälschung der Meldungen *Halt* oder *Weitersenden* besondere Konsequenzen hat: Wird *Halt* während der Übertragung verfälscht und vom Sender als *Weitersenden* empfangen, so sendet er die Daten weiter. Kommt *Weitersenden* beim Sender als *Halt* an, wird der Sendeprozess auf Dauer gestoppt.

Flusskontrolle mittels Krediten

Bei einer Flusskontrolle mit Hilfe von Krediten erteilt der Empfänger dem Sender einige Kredite für die Übermittlung von Datenblöcken. Sind diese Kredite aufgebraucht, muss der Sender die Übermittlung einstellen. Ein Kredit definiert eine Anzahl von Datenblöcken, d.h. deren Sequenznummer, die der Sender abschicken darf, ohne auf eine Quittung vom Empfänger warten zu müssen. Hierbei ist die maximale Länge der Datenblöcke festgelegt. Im Normalfall werden die Kredite laufend erteilt, sodass ein ununterbrochener Datenverkehr aufrechterhalten werden kann. Die Übermittlung von Krediten muss vor Störungen geschützt werden. Bei der Störung einer Kreditmeldung könnte der Sender ohne weitere Kredite bleiben und der Empfänger auf weitere Datenblöcke warten. Damit wäre die Datenübermittlung blockiert. Es muss sichergestellt sein, dass eine Kreditmeldung nicht verdoppelt wird. Wäre dies nicht der Fall, könnte der Sender weitere Datenblöcke senden, die vom Empfänger nicht aufgenommen werden könnten.

Flusskontrolle über Fenstermechanismus

Die Flusskontrolle über einen Fenstermechanismus stützt sich auf eine Sequenznummer der übertragenen Datenblöcken. Vor der Datenübermittlung sprechen sich Quell- und Zielstation über ein *Fenster* innerhalb des Wertebereiches der Sequenznummern ab. Die Fenstergröße `W` bedeutet:

> Die Quellstation darf maximal `W` Datenblöcke absenden, ohne auf eine Quittung von der Zielstation warten zu müssen, d.h. `W` ist bei der Quellstation als Anzahl der Kredite zu interpretieren.

Bei der Zielstation stellt `W` die Kapazität des Empfangspuffers für die ankommenden Datenblöcke dar.

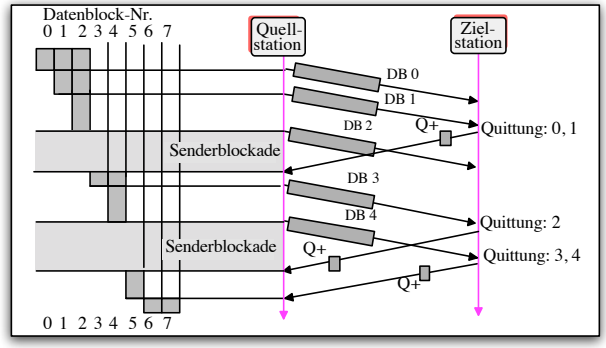


Abb. 1.2-4: Veranschaulichung der Flusskontrolle über den Fenstermechanismus

Flusskontrolle per Fenstermechanismus

Abb. 1.2-4 zeigt den Fall, in dem die Fenstergröße `W = 3` und die Datenblöcke nach dem Modulo-8-Verfahren nummeriert werden. Bei `W = 3` darf die Quellstation drei Datenblöcke absenden, ohne auf eine Quittung warten zu müssen. Abb. 1.2-4 zeigt gleichzeitig die freie Sequenznummer, die der Sender für die Nummerierung verwenden darf. Da `W = 3`, sind maximal drei Nummern zu vergeben. Wie hier ersichtlich, ist während der Übermittlung der ersten drei Datenblöcke keine Quittung angekommen, also muss die Quellstation den Sendeprozess unterbrechen. Dies führt zu einer Senderblockade. Nach der ersten positiven Quittung, mit der die Datenblöcke mit den Nummern 0 und 1 positiv quittiert wurden, darf sie zwei weitere Datenblöcke senden. Dieses Beispiel zeigt, welche Auswirkungen die Fenstergröße auf die Auslastung des Übertragungsmediums hat. Insbesondere im Fall `W = 1` muss man nach dem Absenden jedes Datenblocks den Sendeprozess stoppen. Dies führt selbstverständlich zu einer schlechten Ausnutzung des Übertragungsmediums.

Die Fenstergröße kann als Kredit für die Vergabe von Nummern für die abzusendenden Datenblöcke interpretiert werden. Die meisten Kommunikationsprotokolle realisieren die Flusskontrolle nach dem Fenstermechanismus.

1.2.3 Überlastkontrolle

Ein Netz hat eine bestimmte Aufnahmekapazität, d.h. zu jedem Zeitpunkt kann sich darin nur eine begrenzte Anzahl von Datenblöcken befinden. Wird diese Anzahl überschritten, hat dies die folgenden negativen Auswirkungen:

- *Die Aufnahmepuffer im Netz (in Knoten) sind voll*; dies führt dazu, dass die im Netz eintreffenden Datenblöcke verworfen werden müssen.
- *Es bilden sich Warteschlangen von Datenblöcken vor den Übertragungsleitungen*; durch die so verursachten großen Verweilzeiten der Datenblöcke im Netz entstehen große Verzögerungen der übertragenen Datenblöcke.

Congestion Control

Die Maßnahmen, mit denen eine Überlastung des Netzes verhindert wird, bezeichnet man als *Überlastkontrolle* (Congestion Control). Die wichtigsten Kriterien für die Beurteilung der Überlastung von Netzen sind:

- Durchsatz (*Throughput*) und
- Datenverweilzeit (*Latenzzeit*, *Delay*) im Netz.

Durchsatz

Unter dem *Durchsatz eines Netzes* versteht man den Anteil des Datenverkehrs, der von dem Netz akzeptiert wird. Den Verlauf des Durchsatzes in Abhängigkeit vom Gesamtdatenverkehr zeigt Abb. 1.2-5a. Ist der Datenverkehr im Netz klein (kleine Belastung), werden alle ankommenden Daten durch das Netz aufgenommen; dabei müssen normalerweise keine Vorkehrungen gegen die Überlast ergriffen werden. Bei hoher Netzbelastung dagegen müssen bestimmte Maßnahmen getroffen werden, um eine Überlastung zu vermeiden, und sie führen zur Einschränkung der Datenmenge, die ins Netz gesendet wird.

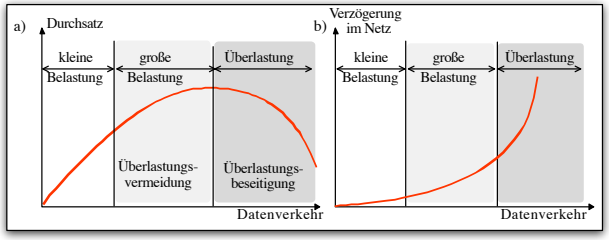


Abb. 1.2-5: Auswirkungen der Netzüberlastung: a) auf den Durchsatz, b) auf die Datenverweilzeit im Netz

Ist der Datenverkehr im Netz so groß, dass das Netz überlastet ist, müssen andere Aktionen eingeleitet werden, um die bestehende Überlastung zu beseitigen. Wie in Abb. 1.2-5a ersichtlich, nimmt der Durchsatz in der Überlastsituation mit zunehmendem Datenverkehr sehr stark ab.

Verweilzeit im Netz

Abb. 1.2-5b veranschaulicht, welche Auswirkungen die Netzbelastung auf das Verhalten der Datenverweilzeit (Latenzzeit) im Netz hat. In einer Überlastsituation muss also mit großen Verzögerungen für die Datenübertragung im Netz gerechnet werden. Die wichtigste Maßnahme für die Vermeidung von Überlasten besteht in der Einschränkung der Datenströme, die ins Netz fließen. Welche Maßnahmen gegen die Überlastung in einzelnen Netzen und Kommunikationsprotokollen ergriffen werden, hängt auch von der Realisierung der Flusskontrolle ab. Komplexere Verfahren der Flusskontrolle können z.B. mittels *Explicit Congestion Control* (ECN) realisiert werden, wie dies in Abschnitt 4.5 besprochen wird.

1.3 Schichtenmodell der Kommunikation

Was ist OSI?

Als man Mitte der 70er Jahre versuchte, die Rechner unterschiedlicher Hersteller miteinander zu vernetzen, hat sich folgendes Problem ergeben: Es sind dringend Kommunikationsregeln nötig, damit ein Rechner des Herstellers *X* mit einem Rechner des Herstellers *Y* kommunizieren kann. Es sollte möglich sein, dass jeder Rechner für die Kommunikation mit allen anderen Rechnern *offen* (bereit) ist. In diesem Zusammenhang wurde bereits damals von der Vernetzung offener Systeme – also von *Open System Interconnection* (OSI) – gesprochen und nach einem Modell für ihre Verwirklichung gesucht.

OSI-Referenzmodell

Daher wurde ein Schichtenmodell eingeführt, dass die Prinzipien der Kommunikation zwischen verschiedenen Systemen beschreibt und die OSI-Vorstellung ermöglicht. Es

wird deshalb *OSI-Referenzmodell* genannt. Standardisiert wurde es von ISO (*International Organization for Standardization*) und es wird auch als *ISO/OSI-Referenzmodell* bzw. kurz als *ISO/OSI-Modell* bezeichnet.

1.3.1 Konzept des OSI-Referenzmodells

Idee von OSI

Die Idee von OSI illustriert Abb. 1.3-1. Gemäß OSI wird ein Rechner als *offenes System* angesehen. Diese Systeme werden durch Übertragungsmedien untereinander verbunden und enthalten entsprechende *Kommunikationsprotokolle*, nach denen *logische Verbindungen zwischen Applikation* in den einzelnen Systemen nach Bedarf aufgebaut und *Nachrichten* bzw. allgemein *Daten* übertragen werden können.

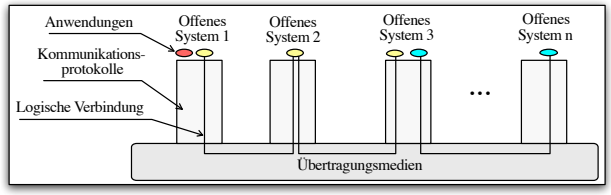


Abb. 1.3-1: Idee von OSI: Jedes System soll mit jedem anderen kommunizieren können

Um die Kommunikationsprotokolle für die Verwirklichung der Zielvorstellung von OSI zu entwickeln, wurde die komplexe Aufgabe der Kommunikation zwischen verschiedenen Systemen so auf sieben Teilaufgaben verteilt, dass diese den einzelnen Schichten, die in einer Hierarchie zueinander stehen, zugeordnet werden. Dadurch ist ein OSI-Referenzmodell mit sieben Schichten entstanden; man spricht hier auch vom *OSI-Schichtenmodell*.

Allgemeines OSI-Schichtenmodel

Ein Rechnernetz enthält aber nicht nur die Rechner als Endsysteme, sondern auch die Netzknoten (Router, Switches) als *Zwischensysteme*. Die Aufgabe der Kommunikation in Zwischensystemen kann aber zu drei untereinander liegenden Schichten zusammengefasst werden. Daher enthalten die Zwischensysteme nur die ersten drei Schichten. Abb. 1.3-2 zeigt die allgemeine Struktur des OSI-Referenzmodells. Die unterste Schicht 1 repräsentiert die physikalische Netzanbindung, also die Übertragungstechnik. Die Schichten von 2 bis 6 repräsentieren bestimmte Funktionen der Kommunikation. Schicht 7 enthält die Anwendungen (*Applikationen*).

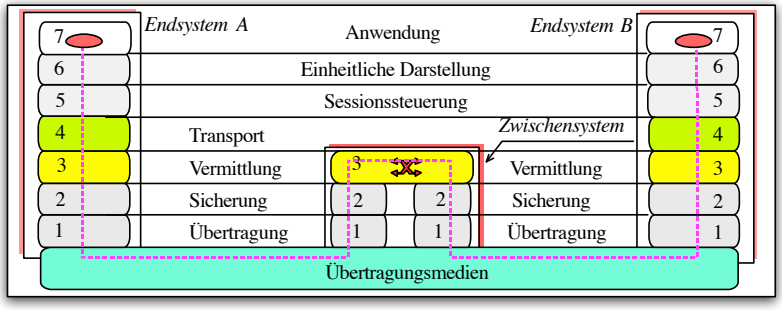


Abb. 1.3-2: OSI-Referenzmodell: Zerlegung der Kommunikationsaufgabe in 7 Schichten

Die einzelnen Schichten im OSI-Referenzmodell sind:

Funktionen der Schichten

1. **Physikalische Schicht** (Übertragungsschicht, *Physical Layer*)
   Sie definiert die mechanischen und elektrischen Eigenschaften sowie die Funktionen und die Abläufe bei der Bitübertragung.
2. **Sicherungsschicht** (*Data-Link Layer*)
   Diese Schicht garantiert eine sichere Übertragung zwischen zwei direkt benachbarten Stationen (Knoten). Dazu werden die übertragenen Bit in *Frames* (Rahmen) zusammengefasst und am Ende mit einer Prüfsumme versehen. Dadurch ist eine Fehlererkennung möglich. In LANs wird die Schicht 2 in zwei Teilschichten aufgeteilt: *Schicht 2a* als *MAC-Schicht* (*Media Access Control*), die den Zugriff auf das Übertragungsmedium regelt, und *Schicht 2b* als *LLC-Schicht* (*Logical Link Control*) [Abb. 12.1-1], die eine Sicherungsschicht darstellt.
3. **Netzwerkschicht/Vermittlungsschicht** (*Network Layer*)
   Diese Schicht hat die Aufgabe, die Daten blockweise zwischen Endsystemen zu übermitteln. Die innerhalb dieser Schicht übertragenen Datenblöcke werden oft *Pakete* genannt. Schicht 3 stellt eine *Paketvermittlungsschicht* dar.
4. **Transportschicht** (*Transport Layer*)
   Die Transportschicht hat u.a. die Aufgabe, eine gesicherte *virtuelle Ende-zu-Ende-Verbindung* für den Transport von Daten zwischen den Endsystemen bereitzustellen. Die Aufgaben der Transportschicht bestehen vor allem in der Korrektur der Übermittlungsfehler und sind von den Protokollen der Schicht 2 und 3 sehr stark abhängig.
5. **Sitzungsschicht** (*Session Layer*)
   Sie ist die unterste anwendungsorientierte Schicht und regelt den Auf- und Abbau von Kommunikationsbeziehungen (Sitzungen, Sessions) sowie deren Wiederherstellung nach Störungen im Transportsystem. Hier findet die *Synchronisation* und somit der *geregelte Dialogablauf* zwischen zwei Kommunikationsprozessen statt.
6. **Darstellungsschicht** (Präsentationsschicht, Presentation Layer)
   Die Umsetzung verschiedener Darstellungen der Information (z.B. die Zeichensätze ASCII und EBCDIC) auf ein einheitliches Format auf der Senderseite ist die Aufgabe der Darstellungsschicht. Diese Schicht kann auch Funktionen enthalten, mit denen Daten komprimiert, konvertiert und verschlüsselt werden können. Vor der Web-Ära war das ASN.1-Konzept (*Abstract Syntax Notation*) für diese Schicht von großer Bedeutung. Inzwischen wurde die Aufgabe von ASN.1 durch XML (*eXtensible Markup Language*) übernommen.
7. **Anwendungsschicht** (*Application Layer*)
   In dieser Schicht sind die sogenannten OSI-Anwendungsprogramme angesiedelt. Zu den wichtigsten OSI-Standardanwendungen gehörten in den 90er Jahren E-Mail (X.400) und verteilter Verzeichnisdienst (X.500) und Filetransfer (FTAM).

Im Allgemeinen kann die Schicht `n-1` im OSI-Referenzmodell als Erbringer bestimmter Kommunikationsdienste für die Schicht `n` angesehen werden. Die mit der Kommunikation verbundenen Aufgaben in den Endsystemen können bestimmten Klassen von Aufgaben zugeordnet werden. Abb. 1.3-3 bringt dies deutlicher zum Ausdruck.

Übermittlungsdienste

Die ersten drei Schichten realisieren die *Übermittlungsdienste*. Schicht 1 realisiert die Übermittlung von Daten bitweise zwischen zwei direkt verbundenen Stationen (d.h. zwischen einem Endsystem und seinem Netzknoten bzw. zwischen zwei benachbarten Netzknoten). Die ersten zwei Schichten realisieren die gesicherte Übermittlung von

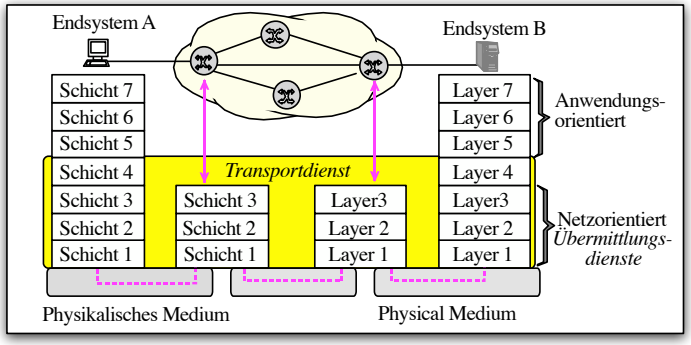


Abb. 1.3-3: Klassen von Aufgaben im OSI-Referenzmodell

Daten in Form von Frames zwischen zwei direkt verbundenen Stationen [Abb. 1.3-5]. Die ersten drei Schichten realisieren in der Regel eine ungesicherte Übermittlung von Datenpaketen zwischen zwei Endsystemen, also z.B. über mehrere Zwischensysteme.

Schicht 4 und Schicht 2

Eine besondere Rolle hat die Schicht 4 (Transportschicht). Sie hat insbesondere die Aufgabe, die unzuverlässige Übermittlung von Datenpaketen zwischen zwei Endsystemen – also den Dienst der ersten drei Schichten – zuverlässig (fehlerfrei) zu machen. Die Aufgabe von Schicht 4 ist somit mit der Aufgabe von Schicht 2 vergleichbar. Diese beiden realisieren die Sicherung der Datenübermittlung. Schicht 2 kümmert sich um die Datenübermittlung über eine 'Leitung' und Schicht 4 kümmert sich um die Übermittlung von Daten zwischen zwei Endsystemen, die in der Regel nicht direkt (physikalisch) verbunden sind.

Transportdienst

Die ersten vier Schichten können daher als *Transportdienst* angesehen werden, der einen gesicherten Datenaustausch zwischen zwei Endsystemen garantiert. Diesen Dienst nutzen die Schichten 5, 6 und 7, die anwendungsorientiert sind.

Dienst der Schichten: Paketierung

PDU und SDU vs. Frame, Paket, Segment, Nachricht

Ein zentrales Konzept der Rechnerkommunikation ist die *Paketierung* der Daten: Die Nutzdaten der Schicht $n + 1$ werden um die Kontrollinformationen der Schicht n angereichert, bzw. den auf Schicht n vorliegenden Datenpakete wird der *Payload* entnommen und dieser der Schicht $n + 1$ übergeben. Wie in Abb. 1.3-5 gezeigt, wird dieses Konzept auf allen Schichten realisiert:

- Die zu übertragenen Daten werden in 'Pakete' variabler Länge geschnürt und mit einem *Protokollkopf* (*Header*) versehen, der das Ziel (*Destination*) und die Herkunft (*Source*) sowie die Verwendung (*Protocol-Identifier*) angibt.
- Das Gesamtpaket, also die *Nutzlast* (*Payload*) und der Protokollkopf, wird als *Protocol Data Unit* (PDU) bezeichnet, die Nutzlast als *Service Data Unit* (SDU).
- Wird zusätzlich das Gesamtpaket durch einen *Trailer* ergänzt, der Informationen zum Schutz vor Datenverfälschung enthält, wird die Bezeichnung *Frame* genutzt.

Während der Begriff *PDU* für alle Schichten des Kommunikationsmodells verwendet werden kann, deutet die Bezeichnung *Paket* an, dass Bezug auf die Netzwerkschicht

genommen wird; *Frames* sind vor allem auf der Sicherungsschicht (Schicht 2) im Einsatz; die TCP-Pakete werden *Segmente* genannt und die Applikationsdaten häufig einfach *Nachrichten*. Gelegentlich wird (besonders bei TLS) der Begriff *Records* genutzt, der aber Synonym zu *Frames* zu verstehen ist, sich allerdings nun nicht mehr auf die Sicherungsschicht bezieht.

Verbindungslose vs. verbindungsorientierte Kommunikation

Verbindungs-orientiert, Verbindungslos

Protokollieren zwei Kommunikationsinstanzen auf der jeweiligen Schicht *Zustandsinformationen* über ihren Partner, und tauschen beide diese Kontrollinformation gegenseitig aus, sprechen wir von *verbindungsorientierter Kommunikation*; im anderen Falle von *verbindungsloser Kommunikation*. In Bezug auf die Schichten des Kommunikationsmodells haben sich folgende Bezeichnungen eingebürgert:

- Eine *verbindungsorientierte* Kommunikation auf den anwendungsorientierten Schichten (7 bis einschließlich 5) wird in der Regel als *Sitzung* (*Session*) bezeichnet, insbesondere dann, wenn dies die Applikation selbst betrifft.
- Auf den Paket- bzw. Frame-übertragenden Schichten 4, 3 und 2 sprechen wird in der Regel bei einer *verbindungsorientierten Übermittlung* schlichtweg von einer *Verbindung* und
- auf der Schicht 1 wird eine kontinuierlich bestehende und überwachte, physikalische Signalverbindung kurz als *Link* bezeichnet.

1.3.2 Schichtenmodell der Protokollfamilie TCP/IP

Auch die Protokollfamilie TCP/IP kann in einem Schichtenmodell dargestellt werden. Dieses Modell ist eine vereinfachte Variante des OSI-Referenzmodells, in der die anwendungsorientierten Schichten 5, 6 und 7 aus dem OSI-Referenzmodell [Abb. 1.3-3] zu einer Schicht zusammengefasst sind. Abb. 1.3-4 zeigt das Schichtenmodell der Protokollfamilie TCP/IP.

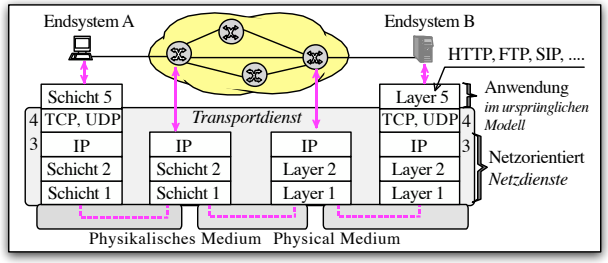


Abb. 1.3-4: Ursprüngliches Schichtenmodell der Protokollfamilie TCP/IP

Im Allgemeinen entsprechen die Funktionen der Schichten 1, 2, 3 und 4 im Schichtenmodell der Protokollfamilie TCP/IP den Funktionen der gleichen Schichten im OSI-Referenzmodell. Die Protokolle der Schichten 1 und 2 in den beiden Schichtenmodellen – d.h. von OSI und von TCP/IP – können auch identisch sein. Innerhalb der Schicht 3 im Modell von TCP/IP wird das Protokoll IP (*Internet Protocol*) angesiedelt. Innerhalb der Schicht 4 werden zwei, in der Regel die Transportprotokolle TCP (*Trans-*

mission Control Protocol) und UDP (*User Datagram Protocol*) eingesetzt. TCP ist ein verbindungsorientiertes; UDP hingegen ein verbindungsloses Transportprotokoll. Als weitere Transportprotokolle fungieren SCTP (*Stream Control Transmisson Protocol*) [Abb. 3.6-1] und DCCP (*Datagram Congestion Control Protocol*). Auf die Unterschiede zwischen TCP und UDP geht Abschnitt 1.4 näher ein.

Schicht 5 als ursprüngliche Anwendungsschicht

Vergleicht man die Schichtenmodelle von OSI mit dem ursprünglichen Ansatz von TCP/IP, d.h. Abb. 1.3-3 und Abb. 1.3-4, stellt man fest, dass die oberen anwendungsorientierten Schichten 5, 6 und 7 aus dem OSI-Referenzmodell beim Schichtenmodell für TCP/IP zu einer Anwendungsschicht zusammengefasst sind.

Support-Protokolle

Mit dem heutigen *Internet der Dinge* (*Internet of Things*), dem *ubiquitous Computing*, den Anforderungen an die Echtzeitkommunikation, der Notwendigkeit für Verschlüsselung und den hiermit einhergehenden sehr unterschiedlichen Anwendungen ergibt sich als Anforderung für die Internetprotokolle eine ergänzende Unterstützung in Form von *Application-Support-Protokollen*, die Bedarfsweise eingesetzt werden und quasi eine zusätzliche Kommunikationsschicht darstellen. Zu diesen Protokollen zählen speziell die Protokolle TLS (*Transport Layer Security*) und DTLS (*Datagram Transport Layer Security*).

Abb. 1.3-5 zeigt der Aufbau von Daten, die zwischen den kommunizierenden Instanzen innerhalb einzelner Schichten übermittelt werden. Die Funktion der *(Application-)Support-Protokolle* lässt sich durchaus mit den Eigenschaften identifizieren, die im OSI-Modell für die *Präsentations-Schicht* vorgesehen war und die in Konsequenz zu einer Erweiterung des ursprünglichen TCP/IP-Schichtenmodells geführt hat.

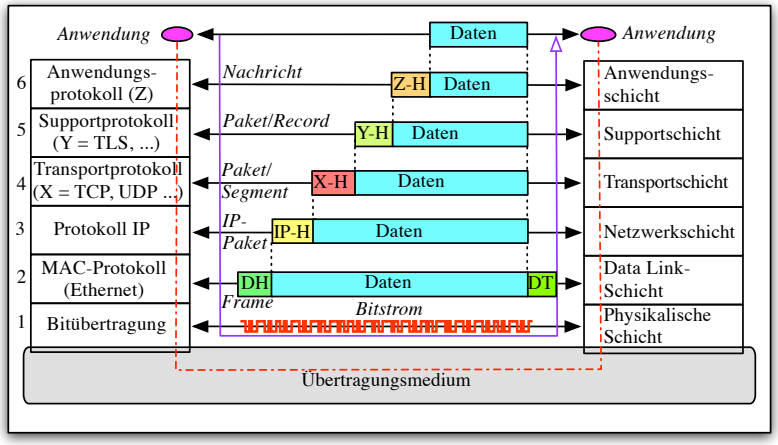


Abb. 1.3-5: Erweitertes Schichtenmodell der Protokollfamilie TCP/IP – Strukturen von zwischen den kommunizierenden Instanzen übermittelten Daten DH: Data-Link Header, DT: Data-Link Trailer

Strukturierung der übermittelten Daten

Vereinfacht kann man sich die Übermittlung von Daten zwischen zwei Anwendungen folgendermaßen vorstellen: Dem zu sendenden Datenblock `Daten` wird ein Header `Z-H` mit bestimmten Angaben des Anwendungsprotokolls `Z` (z.B. `Z` = HTTP) im Quellrechner vorangestellt. Dies stellt sicher, dass das Paar [`Z-H`, `Daten`] immer an

die gleiche Instanz des Anwendungsprotokolls Z – nun aber im Zielrechner – übergeben wird. Bei Bedarf wird ein Protokoll der Application-Support-Schicht Y (Y = TLS oder DTLS) angefordert, um die Nutzdatenübertragung zu sichern. Hierdurch weiß der Zielrechner, wie er dieses Paket zu verarbeiten und ggf. zu entschlüsseln hat. Das resultierende Paket [Y-H, [Z-H, Daten]] muss nun an die Instanz des gleichen Supportprotokolls Y im Zielrechner übermittelt werden. Hierfür wird es an das Transportprotokoll X (X = TCP bzw. UDP) übergeben. Nun wird ein Header X-H des Transportprotokolls X vorangestellt, sodass eine Dateneinheit [X-H[Y-H[Z-H,Daten]]] des Transportprotokolls entsteht. Diese Dateneinheit wird nun an die IP-Instanz übergeben, wo ihr ein IP-Header (IP-H) hinzugefügt wird. So entsteht ein *IP-Paket*, das als Payload in einen *Data-Link Frame* (*DL-Frame*) eingebettet und durch den Data-Link-Header (DLH) und den Data-Link-Trailer (DLT) ergänzt wird. Dieses DL-Frame wird nun zum Zielrechner übertragen. Dort müssen die empfangenen Daten aus Schicht 1 an die Anwendung (Schicht 6) übergeben werden.

Abb. 1.3-5 zeigt den zusammengefassten Übermittlungsvorgang:

Übermittlungsvorgang

1. Quellrechner: *Vorbereitung von Daten zum Senden*

Daten		⇒
Anwendungsprotokolleinheit	[Z-H,Daten]	⇒
Supportprotokolleinheit	[Y-H[Z-H,Daten]]	⇒
Transportprotokolleinheit	[X-H[Y-H[Z-H,Daten]]	⇒
IP-Paket	[IP-H[X-H[Y-H[Z-H,Daten]]]]	⇒
DL-Frame	[DLH[IP-H[X-H[Y-H[Z-H,Daten]]]]DLT].	

2. DL-Frame wird *bitweise* übertragen.
3. Zielrechner: *Übergabe von Daten an die Anwendung*

DL-Frame	[DLH[IP-H[X-H[Y-H[Z-H,Daten]]]]DLT]	⇒
IP-Paket	[IP-H[X-H[Y-H[Z-H,Daten]]]]	⇒
Transportprotokolleinheit	[X-H[Y-H[Z-H,Daten]]]	⇒
Supportprotokolleinheit	[Y-H[Z-H,Daten]]	⇒
Anwendungsprotokolleinheit	[Z-H,Daten]	⇒Daten.

Bemerkung: Abb. 1.4-2 illustriert eine vereinfachte Situation, bei der die zu sendenden Datenmenge so groß ist, dass man sie nicht in einem IP-Paket übermitteln kann. Hier kommt TCP zum Einsatz, und die Daten werden auf mehrere IP-Pakete aufgeteilt. Man spricht hierbei von *Segmentierung der Daten*. Ein IP-Paket enthält damit ein Datensegment.

1.4 Allgemeine Prinzipien der IP-Kommunikation

Die wichtigen Prinzipien der Kommunikation in IP-Netzen können weitgehend aus dem in Anschnitt 1.3.2 dargestellten Schichtenmodell abgeleitet werden. Hierbei spielen die Schichten *Netzwerkschicht* mit dem Protokoll IP und *Transportschicht* mit den Protokollen TCP und UDP eine dominierende Rolle. Bevor auf diese beiden Schichten eingegangen wird, wird zunächst die Bildung von IP-Paketen kurz vorgestellt.

1.4.1 Bildung von IP-Paketen

Nutzung von UDP

Bei der Bildung von IP-Paketen ist zu unterscheiden, ob TCP oder UDP als Transportprotokoll eingesetzt wird. Beim Einsatz des verbindungslosen Transportprotokolls UDP werden die Daten bzw. eine Nachricht einer Anwendung – als Nutzlast – um den UDP-Header ergänzt, sodass eine UDP-Dateneinheit entsteht. Wie Abb. 1.4-1 zeigt, wird aus jeder UDP-Dateneinheit durch das Voranstellen eines IP-Header ein *IP-Paket* gebildet. Da die IP-Pakete keine Angaben zur Synchronisation enthalten, um sie auf der Leitung zu 'markieren', müssen sie in *Data-Link Frames* (*DL-Frames*) eingebettet werden.

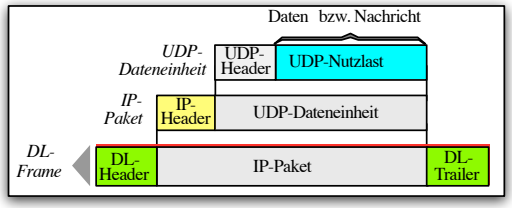


Abb. 1.4-1: Kapselung der Nutzlast beim UDP-Einsatz

MAC-Frames in LANs

In LANs bildet die sogenannte MAC-Funktion (*Media Access Control*) den Kern der Data-Link-Schicht. Wird ein IP-Paket in einem LAN übermittelt, wird es in einen MAC-Frame eingebettet. Bei der Übermittlung der IP-Pakete über eine Leitung bzw. über eine Punkt-zu-Punkt-Verbindung wird innerhalb der Schicht 2 häufig das Protokoll PPP (*Point-to-Point Protocol*) verwendet. In diesem Fall stellen die DL-Frames *PPP-Frames* dar (siehe Abschnitt 13.2).

Bedeutung von DL-Frames

Jedes zu übertragende IP-Paket muss immer in einen DL-Frame eingebettet werden. Dies bedeutet, dass jedem IP-Paket ein DL-Header vorangestellt wird und nach dem Ende des IP-Pakets folgt ein DL-Trailer. Diese beiden enthalten bestimmte *Synchronisationsangaben* (oft die Bitfolge `01111110`), um den Beginn und das Ende des DL-Frames auf einer Leitung zu erkennen. Abb. 1.4-2 illustriert, wie die IP-Pakete aus den Daten bzw. aus der langen Nachricht eines Anwendungsprotokolls bei der Nutzung des verbindungsorientierten Transportprotokolls TCP gebildet werden.

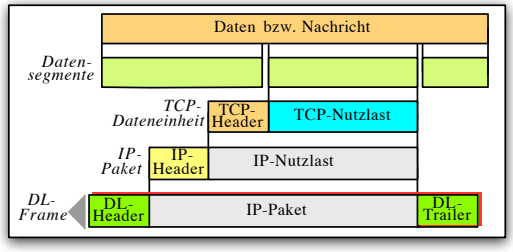


Abb. 1.4-2: Verkapselung der Nutzlast beim TCP-Einsatz

Anders als bei UDP entstehen aus den zu übermittelnden Daten bei TCP mehrere *Datensegmente*. Jedes Datensegment wird dann um einen TCP-Header erweitert,

sodass eine TCP-Dateneinheit entsteht. Aus jeder TCP-Dateneinheit wird im nächsten Schritt ein IP-Paket gebildet. Zum Senden wird das IP-Paket in einen DL-Frame eingekapselt.

Wie aus Abb. 1.4-1 und Abb. 1.4-2 ersichtlich ist, werden die IP-Pakete zum Senden immer in entsprechende DL-Frames der zweiten Schicht eingekapselt, die vom Übermittlungsnetz abhängig sind. Erst in einem DL-Frame kann ein IP-Paket über ein physikalisches Netz gesendet werden.

1.4.2 Netzwerkschicht in IP-Netzen

Arten der Netzwerkschicht

Die Netzwerkschicht in IP-Netzen hat die Aufgabe, die Daten in Form von IP-Paketen zwischen Endsystemen zu übermitteln. Hierbei unterscheidet man zwischen der *verbindungslosen* und der *verbindungsorientierten* Netzwerkschicht:

Verbindungslos

- Wird *keine* Route über das Netz für einen Strom der von einem Quellrechner zu einem Zielrechner zu übermittelnden IP-Pakete festgelegt, sondern jedes einzelne Paket aus diesem Strom nach einem eigenen Weg über das Netz zum Zielrechner übermittelt, handelt es sich um die *verbindungslose Netzwerkschicht*.

Verbindungs-orientiert

- Wird *eine* Route über das Netz für einen Strom der von einem Quellrechner zu einem Zielrechner zu übermittelnden IP-Pakete festgelegt und werden alle Pakete aus diesem Strom nach dem gleichen Weg über das Netz, der eine logische Verbindung darstellt, zum Zielrechner übermittelt, handelt es sich um die *verbindungsorientierte Netzwerkschicht*.

Verbindungslose Netzwerkschicht

Verbindungslose Netzwerkschicht

Die verbindungslose Netzwerkschicht bedeutet, dass die Vermittlungsnetzknoten im IP-Netz die Router darstellen und die einzelnen IP-Pakete als *Datagrams* voneinander unabhängig über das Netz übermittelt werden. Diese Übermittlungsart entspricht dem Versand von Briefen bei der Post. Jedes IP-Paket kann daher mit einem Brief verglichen werden. Der Router würde einer Briefverteilungsstelle entsprechen. Abb. 1.4-3 illustriert die Struktur der verbindungslosen Netzwerkschicht in IP-Netzen.

Interpretation der IP-Adresse

Die ersten drei unten liegenden Schichten realisieren also beim Einsatz von Routern einen *verbindungslosen Übermittlungsdienst*. Dieser entspricht dem Briefpostdienst und eine IP-Adresse ist mit einer postalischen Adresse vergleichbar. Die IP-Adresse stellt auch einen Zugangspunkt zum Dienst für die Übermittlung der IP-Pakete dar und ist oberhalb der Schicht 3 – also an der Grenze zu Schicht 4 – anzusiedeln.

Die IP-Instanz kann als ein IP-Multiplexer angesehen werden. Zwischen den IP-Instanzen werden die IP-Pakete übermittelt. Jedes IP-Paket setzt sich aus einem IP-Header `IP` und einer Transportprotokolldateneinheit `TP-D` zusammen, d.h, es hat die Struktur `[IP-H,TP-D]`.

Die Ports des IP-Multiplexers repräsentieren die Nummern der Protokolle von Schicht 4, die auf die Übermittlungsdienste direkt zugreifen können (vgl. Abb. 1.4-7 und Abb. 1.4-8). Die Protokollnummer wird im IP-Header übermittelt [Abb. 2.2-1] und informiert, von welchem Protokoll die Dateneinheit im IP-Paket stammt. Jedem Pro-

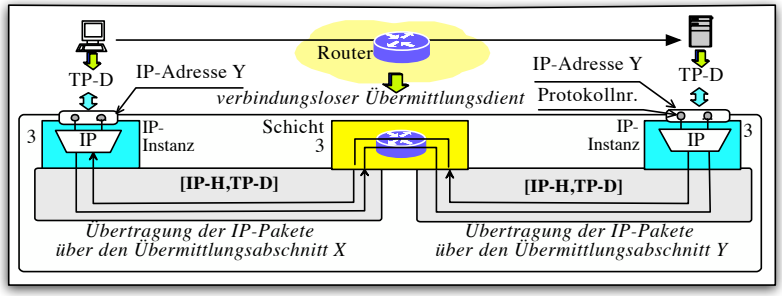


Abb. 1.4-3: Struktur der verbindungslosen Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

tokoll der Schicht 4 wird daher von der IANA (*Internet Assigned Numbers Authority*) eine feste und weltweit eindeutige Nummer zugewiesen.

Verbindungsorientierte Netzwerkschicht

Verbindungs-orientierte Netzwerkschicht

Der Einsatz von MPLS (*Multi-Protocol Label Switching*) bzw. von GMPLS (*Generalized MPLS*) führt zur verbindungsorientierten Netzwerkschicht in IP-Netzen [Kapitel 11]. In diesem Fall fungieren die *(G)MPLS-Switches* als Vermittlungsnetzknoten. Bei der verbindungsorientierten Netzwerkschicht wird zuerst eine Route über das Netz für die Übermittlung eines Stroms der IP- Pakete festgelegt und danach werden alle IP-Pakete aus diesem Strom im 'Gänsemarsch' über das Netz vom Quellrechner zum Zielrechner übermittelt. Diese Übermittlungsart wird heute hauptsächlich in IP-Netzen von großen Netzdienstanbietern realisiert.

Abb. 1.4-4 zeigt die Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen beim MPLS-Einsatz.

MPLS-Multiplexer

Die ersten drei unten liegenden Schichten realisieren beim MPLS-Einsatz einen verbindungsorientierten Übermittlungsdienst. Die IP-Adresse stellt einen Zugangspunkt zu diesem Dienst dar. Die IP-Instanz enthält hier – im Vergleich zur IP-Instanz in Abb. 1.4-3 – zusätzlich einen *MPLS-Multiplexer*.

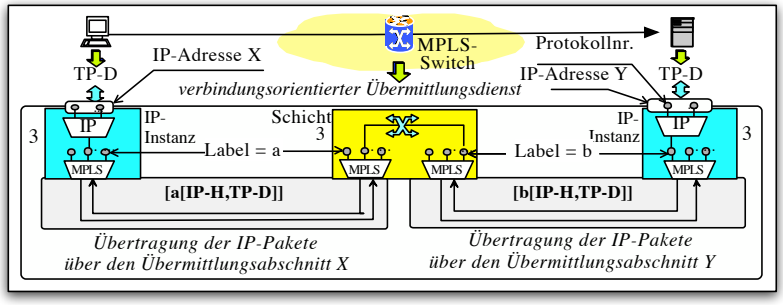


Abb. 1.4-4: Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

Virtuelle Verbindung

Einem Strom von IP-Paketen wird ein Port im MPLS-Multiplexer zugeordnet. Somit können mehrere Datenströme parallel übermittelt werden. Die Portnmmern des MPLS-Multiplexers stellen die *Labels* dar. Ein Label wird immer den zu übermittelnden IP-Paketen eines Stroms vorangestellt. Abb. 1.4-4 bringt dies zum Ausdruck. Ein Label informiert, von welchem Port im MPLS-Multiplex ein IP-Paket stammt bzw. welchem Port es übergeben werden muss. Ein MPLS-Switch leitet – im Allgemeinen – ein empfangenes IP-Paket nach einer Switching-Tabelle von einem Port zu einem anderen weiter. Daher kann ein anderes Label den IP-Paketen eines Stroms auf einem anderen Übermittlungsabschnitt vorangestellt werden. Zwischen den Ports im MPLS-Multiplexer entsteht entsprechend im Quell- und im Zielrechner eine logische Verknüpfung, die als *virtuelle (logische) Verbindung* interpretiert wird.

1.4.3 Verbindungslose IP-Kommunikation im Internet

Nachbildung des Briefdienstes

Das Internet stellt eine weltweite Kopplung von physikalischen Netzen dar, in denen das Protokoll IP eingesetzt wird. Somit kann das Internet als heterogenes IP-Netz angesehen werden. Als IP-Netz setzt sich das Internet aus einer Vielzahl von IP-Subnetzen zusammen, die mit Hilfe von Routern miteinander vernetzt sind. Daher ist die Netzwerkschicht im heutigen Internet verbindungslos [Abb. 1.4-3]. Ein Router leitet jedes empfangene IP-Paket unabhängig von der aktuellen Lage im Netz und von anderen Paketen weiter.

Abb. 1.4-5 illustriert das Prinzip der Kommunikation im Internet an einem Beispiel, in dem eine Folge von TCP-Dateneinheiten gesendet wird. Jede dieser Dateneinheiten wird als ein IP-Paket gesendet. Im Zielrechner setzt TCP die in den IP-Paketen empfangenen Daten wieder zusammen. Gehen einige TCP-Dateneinheiten bei der Übertragung verloren bzw. werden sie verfälscht, so fordert TCP im Zielrechner vom Quellrechner eine wiederholte Übertragung an [Abschnitt 4.3].

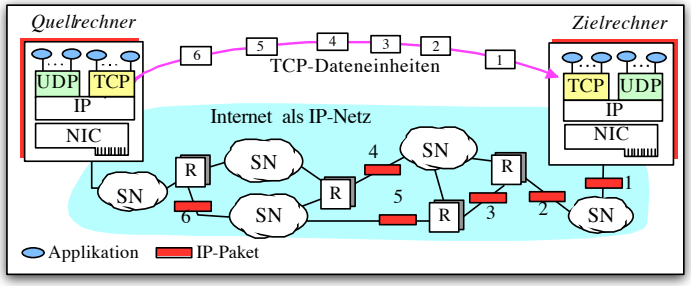


Abb. 1.4-5: Prinzip der Kommunikation im Internet – Datagramm-Prinzip
R: Router, SN: IP-Subnetz, NIC: Network Interface Card (Controller)

IP-Pakete wie Briefe

Beim Einsatz von Routern werden die IP-Pakete als *Datagrams* (also wie Briefe) unabhängig voneinander zum Zielrechner gesendet. Die wichtigsten Angaben in IP-Paketen sind die IP-Adressen von Quell- und Zielrechner. Da die einzelnen IP-Pakete unabhängig voneinander abgeschickt werden, können sie am Ziel in einer anderen Reihenfolge ankommen, als sie abgeschickt wurden. Für die Wiederherstellung von Daten aus so empfangenen IP-Paketen ist TCP verantwortlich.

Bedeutung von TTL

Da die IP-Pakete im Netz zirkulieren können, ist es nötig, ihre Verweilzeit im Netz zu kontrollieren. Der Quellrechner gibt als TTL-Angabe (*Time To Live*) im IP-Header [Abb. 2.2-1] an, wie lange das IP-Paket im Netz verweilen darf. Weil der TTL-Wert in jedem Router um 1 verringert wird, ist er identisch mit der maximalen Anzahl von Routern, die ein IP-Paket durchlaufen darf. Fällt der TTL-Wert auf 0, wird das IP-Paket im Router verworfen. Der Quellrechner wird dann mit einer Meldung des Protokolls ICMP (*Internet Control Message Protocol*) darüber informiert.

1.4.4 Transportschicht in IP-Netzen

Interpretation der IP-Adresse

Um die Bedeutung der Transportschicht in IP-Netzen näher zu erläutern, zeigt Abb. 1.4-6 die vereinfachte Struktur von Rechnern am IP-Netz. Die IP-Adresse eines Rechners kann einem Kommunikationspuffer zugeordnet werden, der einen Zugangsport zum Protokoll IP darstellt. Dieser Kommunikationspuffer befindet sich an der Grenze zwischen der Schicht 3 mit dem Protokoll IP und der Schicht 4 mit den Transportprotokollen TCP und UDP.

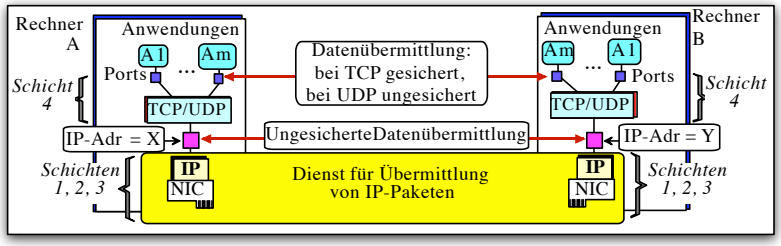


Abb. 1.4-6: Vereinfachte Struktur von Rechnern am IP-Netz
A: Applikation, Adr: Adresse, NIC: Network Interface Controller

Die drei Schichten 1, 2 und 3 stellen einen Dienst für die Übermittlung der IP-Pakete zwischen den Rechnern zur Verfügung. Es handelt sich hier um eine ungesicherte Übermittlung von IP-Paketen zwischen IP-Adressen. Eine IP-Adresse stellt einen Zugangspunkt zu diesem Übermittlungsdienst für die Protokolle TCP und UDP der Transportschicht (Schicht 4) dar.

Arten der Kommunikation

Die Transportschicht regelt den Verlauf der Datenübermittlung zwischen Anwendungen – genauer gesagt zwischen Ports dieser Anwendungen – in verschiedenen Rechnern. Hierbei sind zwei Arten der Kommunikation zu unterscheiden:

- *verbindungslose* Kommunikation beim UDP-Einsatz,
- *verbindungsorientierte* Kommunikation beim TCP-Einsatz

UDP-Multiplexer

Abb. 1.4-7 zeigt die Transportschicht mit UDP. Eine UDP-Instanz kann als UDP-Multiplexer angesehen werden. Die Eingangsports zu diesem Multiplexer stellen die Kommunikationspuffer einzelner UDP-Anwendungen dar, die kurz als *Ports* bezeichnet werden. Der Ausgangsport des UDP-Multiplexers führt zu einer IP-Adresse. Da-

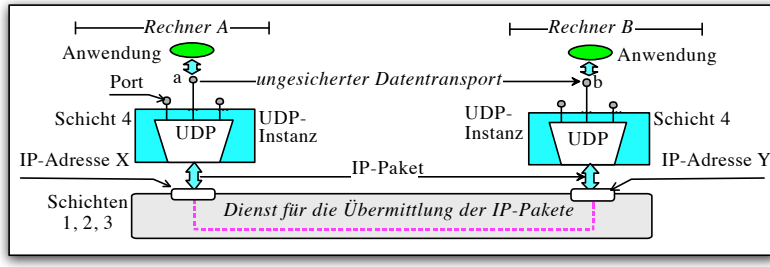


Abb. 1.4-7: Transportschicht mit UDP; ungesicherter Datentransport

mit können mehrere UDP-Anwendungen parallel auf den Dienst für die Übermittlung der IP-Pakete zugreifen.

Verbindungslose Kommunikation

Beim UDP-Einsatz ist die Kommunikation zwischen zwei Anwendungen verbindungslos, d.h, es wird keine Vereinbarung über den Verlauf der Kommunikation zwischen ihnen getroffen. Der Quellrechner als Initiator der Kommunikation übermittelt ein UDP-Paket an den Zielrechner, ohne ihn zu 'fragen', ob er in der Lage ist, dieses Paket zu empfangen. Bei derartiger Kommunikation findet daher keine Fehler- und Flusskontrolle statt [Abschnitt 1.2].

Verbindungsorientierte Kommunikation

Bei der verbindungsorientierten Kommunikation zwischen zwei Anwendungen beim TCP-Einsatz vereinbaren die beiden kommunizierenden Rechner zuerst, wie die Kommunikation zwischen ihnen verlaufen soll, d.h. wie die zu übertragenden Daten zu nummerieren sind und wie die Fehler- und die Flusskontrolle ablaufen sollen. Eine Vereinbarung zwischen zwei Rechnern in Bezug auf den Verlauf der Kommunikation zwischen ihnen wird als TCP-Verbindung bezeichnet [Abb. 1.4-8].

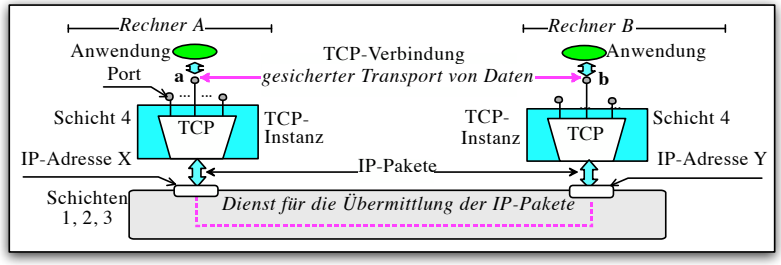


Abb. 1.4-8: Transportschicht mit TCP; gesicherter Datentransport

TCP-Multiplexer

Eine TCP-Instanz ist auch ein TCP-Multiplexer. Die Eingangsports zu diesem Multiplexer stellen die Ports einzelner TCP-Anwendungen dar. Der Ausgangsport des TCP-Multiplexers führt wie bei UDP zu einer IP-Adresse, sodass mehrere TCP-Anwendungen parallel auf den Übermittlungsdienst für IP-Pakete zugreifen können.

Well-known Ports

Die TCP- und UDP-Anwendungen wie z.B. HTTP, FTP bzw. SIP sind feste Standardanwendungen, die unter den allgemein bekannten und weltweit eindeutigen Portnummern (in Zielrechnern!) erreichbar sind. Eine derartige Nummer wird in der TCP/IP-Welt als *Well-known Port* bezeichnet. Eine Zusammenstellung von

Standardanwendungen und deren Portnummern kann in UNIX-Rechnern in der Datei `/etc/services` eingesehen werden. Unter der Adresse `https://www.iana.org/assignments/port-numbers` befindet sich die Auflistung aller Well-known Ports.

Lokation von Anwendungen

Um eine TCP- und eine UDP-Anwendung eindeutig weltweit zu lokalisieren, muss man Folgendes angeben:

- auf welchem Rechner die Anwendung läuft; das bestimmt eindeutig die IP-Adresse des Rechners.
- auf welchen Port im UDP- bzw. TCP-Multiplexer die Anwendung zugreift; das bestimmt die UTP- bzw. TCP-Portnummer.

Bedeutung von Socket

Eine TCP- und UDP-Anwendung lokalisiert man daher durch die Angabe (IP-Adresse, Port). Dieses Paar hat eine fundamentale Bedeutung bei der Rechnerkommunikation und wird als *Socket* bezeichnet. Die Rechnerkommunikation bei TCP/IP kann mit Hilfe von Sockets sehr anschaulich dargestellt werden. Abb. 1.4-9 illustriert dies.

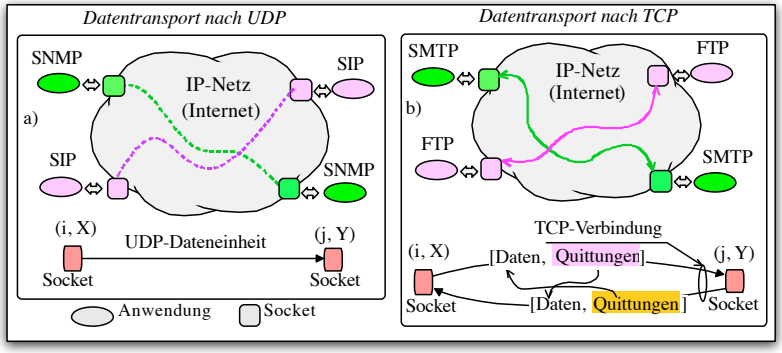


Abb. 1.4-9: Datentransport zwischen Anwendungen: a) beim UDP-Einsatz, b) beim TCP-Einsatz

Socket als Software-Steckdose

Sockets dienen somit als Zugangspunkte zu einer Wolke, die ein IP-Netz bzw. das ganze Internet repräsentiert. Ein Socket kann auch als 'Software-Steckdose' für den Anschluss einer Anwendung an das IP-Netz angesehen werden. Jedem Socket steht im Rechner ein reservierter Speicherplatz als Kommunikationspuffer zur Verfügung. Die zu übertragenden und zu empfangenden Daten einer Anwendung werden jeweils in dem für das Socket reservierten Kommunikationspuffer abgelegt. Sockets sind somit auf die Zeitdauer der Verbindung beschränkt.

Wie Abb. 1.4-9a zeigt, wird bei UDP keine Verknüpfung von Sockets hergestellt, sondern eine UDP-Dateneinheit direkt an den Zielrechner gesendet und ihr Empfang vom Zielrechner nicht bestätigt.

TCP-Verbindung

Bei TCP hingegen [Abb. 1.4-9b] vereinbaren die zwei Rechner, wie der Verlauf des Datentransports zwischen den Sockets geregelt werden soll. Damit wird zwischen beiden Sockets eine *logische Verknüpfung* hergestellt, die eine *TCP-Verbindung* darstellt. Ein Socket bei TCP ist auch ein Endpunkt einer TCP-Verbindung. Eine TCP-Verbindung ist *vollduplex* und setzt sich aus zwei entgegen gerichteten, unidirektionalen Verbindungen zusammen. Eine TCP-Verbindung kann somit als 'zweispurige virtuelle Stra-

ße' über ein IP-Netz verstanden werden, über die ein gesicherter Datentransport erfolgt indem die empfangenen Daten quittiert werden [Abschnitt 4.3].

1.4.5 Multiplexmodell der Protokollfamilie TCP/IP

Nach der Beschreibung der einzelnen Schichten im Schichtenmodell für TCP/IP soll jetzt die Adressierung in IP-Netzen näher dargestellt werden. Abb. 1.4-10 zeigt ein Multiplexmodell der Protokollfamilie TCP/IP, falls ein IP-Netz auf LAN-Basis, z.B. auf Ethernet-Basis, aufgebaut wird.

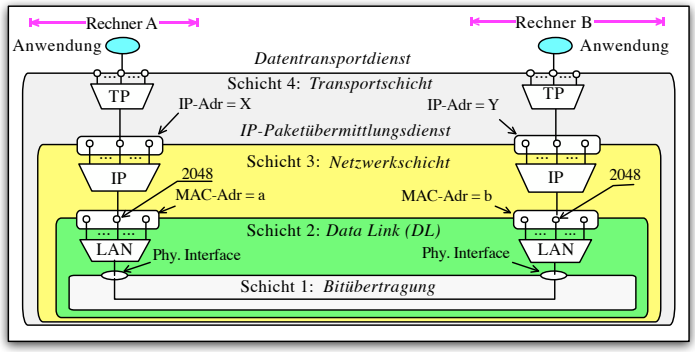


Abb. 1.4-10: Multiplexmodell der Protokollfamilie TCP/IP beim IP-Netz auf LAN-Basis
TP: UDP bzw. TCP

Hier soll u.a. gezeigt werden, dass alle Schichten von `1` bis `n-1` einen Übermittlungsdienst für die Schicht `n` zur Verfügung stellen. Die Schicht `1` stellt einen Dienst für die Übermittlung der Bitströme zur Verfügung. Der Zugang zu diesem Dienst erfolgt über physikalische Interfaces.

Interpretation der MAC-Adresse

Ein Rechner am LAN enthält normalerweise eine LAN-Adapterkarte, die zusammen mit einem Treiber u.a. die Funktion eines Multiplexers realisiert [Abb. 1.4-10]. Die Ports in diesem *LAN-Multiplexer* repräsentieren die Nummern der Protokolle von Schicht 3 [Abschnitt 1.3]. Die Nummer von IP ist beispielsweise `2048` (dezimal), bzw. `0x800` hexademizmal[2] Jeder Rechner am LAN ist unter einer *MAC-Adresse* erreichbar. Sie ist an der Grenze zwischen Schicht 2 und 3 anzusiedeln und kann auch als Zugangspunkt zum Dienst der Schicht 2 interpretiert werden. Über eine MAC-Adresse können daher verschiedene Protokolle der Schicht 3 auf diesen Dienst – also auf den LAN-Dienst – zugreifen.

IP-Multiplexer

Logisch gesehen wird die IP-Protokollinstanz aus der Schicht 3, die als *IP-Multiplexer* interpretiert werden kann (vgl. Abb. 1.4-3 und Abb. 1.4-4), an den Port 2048 im LAN-Multiplexer angebunden. Ein Port im IP-Multiplexer repräsentiert die Nummer eines Protokolls der Transportschicht. Schicht 3 stellt einen Dienst für die Übermittlung der IP-Pakete zwischen entfernten Rechnern bereit. Eine IP-Adresse kann als Zugangspunkt zu diesem Dienst betrachtet werden, und über sie können mehrere Protokolle der

[2] Bei Ethernet-Frames erfolgt die Angabe dieses *EtherType* [`https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml`] im MAC-Header unmittelbar vor dem eigentlichen Payload.

Transportschicht diesen Dienst nutzen. Die Instanzen der Transportprotokolle TCP bzw. UDP realisieren ebenfalls die Multiplexfunktion [Abb. 1.4-7 und Abb. 1.4-8]. Daher können mehrere TCP- bzw. UDP-Anwendungen über eine IP-Adresse auf die Dienste für die Übermittlung der IP-Pakete zugreifen.

1.5 Komponenten der Protokollfamilie TCP/IP

Nach der Darstellung der Kommunikationsprinzipien bei TCP/IP anhand des Schichtenmodells soll nun gezeigt werden, welche Protokolle den einzelnen Schichten zuzuordnen sind und wie sie kooperieren. Abb. 1.5-1 zeigt die Protokollfamilie TCP/IP beim klassischen Protokoll IP, d.h. IP in Version 4 (IPv4); wobei sich eine vergleichbare Darstellung für IPv6 in Abb. 10.1-1 findet, die auch das neue Protokoll QUIC einschließt.

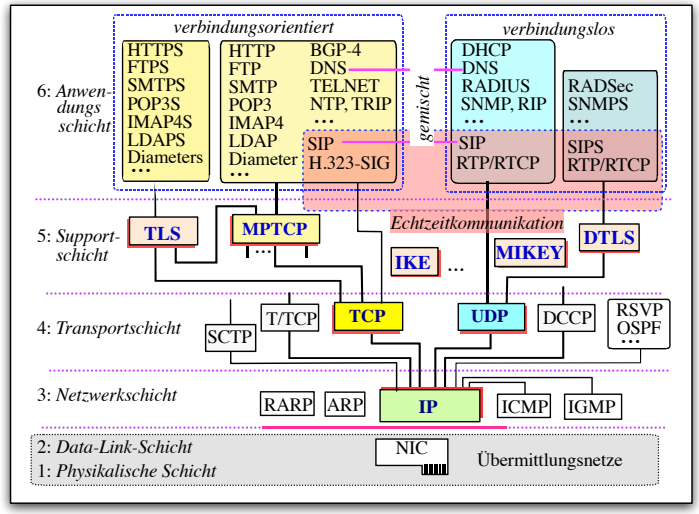


Abb. 1.5-1: Protokolle der Familie TCP/IPv4 im Schichtenmodell
NIC: Network Interface Card (Adapterkarte)

Wie hier gezeigt wurde, besteht die Protokollfamilie TCP/IP nicht nur aus den Protokollen TCP und IP, sondern enthält eine Reihe weiterer Protokolle, die den Schichten *Netzwerkschicht*, *Transportschicht*, *Supportschicht* und *Anwendungsschicht* im erweiterten Schichtenmodell für TCP/IP [Abb. 1.3-5] zugeordnet werden können.

1.5.1 Protokolle der Netzwerkschicht

Die Netzwerkschicht im Schichtenmodell für TCP/IP beschreibt u.a., wie die IP-Netze logisch auf *IP-Subnetze* aufgeteilt werden können und wie die Daten in Form von IP-Paketen in einzelnen IP-Subnetzen und zwischen ihnen übermittelt werden. Die Protokolle der Netzwerkschicht sind:

- **IP**: *Internet Protocol* liegt sowohl in der alten Version 4 (IPv4) als auch in der neuen Version 6 (IPv6) vor. IPv4 und IPv6 sind unterschiedliche Implementierungen auf der Netzwerkschicht und nutzen getrennte Adressräume bzw. Adressierungsverfahren. Im Detail wird IPv4 in Kapitel 2 und IPv6 in Kapitel 7 dargestellt.
- **ARP**: *Address Resolution Protocol* nutzt einen Broadcast-Dienst innerhalb der Schicht 2 zur dynamischen Ermittlung einer MAC-Adresse eines Rechners im LAN, falls seine IP-Adresse bekannt ist [Abschnitt 3.6].
- **RARP**: *Reverse Address Resolution Protocol* unterstützt ebenfalls die Adressierung und stellt das Gegenstück zu ARP dar. Es hat die Aufgabe, für eine MAC-Adresse eine IP-Adresse zu bestimmen [Abschnitt 3.6].
- **ICMP**: *Internet Control Message Protocol* wird für die Übermittlung von Fehlermeldungen und anderen Kontrollangaben verwendet [Abschnitt 3.7].
- **IGMP**: *Internet Group Management Protocol* gilt als Erweiterung von ICMP und dient vornehmlich dazu, das Management von *Multicast-Gruppen* in IP-Subnetzen zu unterstützen [Abschnitt 3.8].

  Bemerkung: Die Protokolle ICMP und IGMP werden üblicherweise der Schicht 3 im TCP/IP-Schichtenmodell zugeordnet. Da die Nachrichten dieser Protokolle in IP-Paketen übermittelt werden, könnte man ICMP und IGMP nach den im OSI-Referenzmodell geltenden Prinzipien zwar der Schicht 4 zuordnen, aber ICMP und IGMP sind keine Transportprotokolle.

1.5.2 Protokolle der Transportschicht

In der Transportschicht befinden sich die Protokolle für die Unterstützung der verbindungsorientierten und der verbindungslosen Kommunikation sowie andere spezielle Protokolle. Die wichtigsten sind:

- **TCP**: *Transmission Control Protocol* ermöglicht die verbindungsorientierte Kommunikation zwischen Rechnern. Hierbei wird zwischen ihnen eine virtuelle Verbindung aufgebaut, die als TCP-Verbindung bezeichnet wird. Eine TCP-Verbindung kann als 'Straße' mit zwei entgegen gerichteten Spuren angesehen werden [Abb. 1.4-9b]. Somit ist TCP ein *verbindungsorientiertes Transportprotokoll*. Durch die Realisierung der Fehler- und der Flusskontrolle garantiert TCP einen zuverlässigen Datentransport. Da bei TCP die zu übertragenden Byte nummeriert werden, ist TCP ein *bytestream-orientiertes Protokoll*. TCP wird in Abschnitt 4.3 beschrieben.
- **MPTCP**: *Multipath TCP* stellt für Anwendungen mehrere TCP-Verbindungen bereit, die über unterschiedliche IP-Adressen und über mehrere parallel verlaufende Datenpfade (*Multipath*) geführt werden können [RFC 6824].
- **UDP**: *User Datagram Protocol* erlaubt lediglich eine verbindungslose Kommunikation zwischen Rechnern, bei der keine virtuelle Verbindung aufgebaut wird. Somit ist UDP ein *verbindungsloses Transportprotokoll*. Bei UDP erfolgt keine Fehler- bzw. Flusskontrolle, sodass UDP im Gegensatz zu TCP keinen zuverlässigen Datentransport garantiert. Auf UDP geht Abschnitt 4.2 ein.
- **T/TCP**: *Transaction TCP* ist eine Ergänzung von TCP im Hinblick auf die Unterstützung sogenannten Transaktionen. Unter einer Transaktion versteht man einen

Kommunikationsvorgang, der aus mehreren Phasen besteht, die alle korrekt durchgeführt werden müssen.

- **SCTP**: *Stream Control Transmission Protocol* ermöglicht genau wie TCP die verbindungsorientierte Kommunikation zwischen Rechnern. Bei SCTP wird eine virtuelle Verbindung, die sogenannte *SCTP-Assoziation*, aufgebaut [RFC 4960]. Eine SCTP-Assoziation kann als 'Autobahn' mit einer beliebigen Anzahl von entgegen gerichteten Spuren angesehen werden. Daher ist SCTP ein verbindungsorientiertes Transportprotokoll. Auf SCTP geht Abschnitt 4.6 näher ein.
- **RSVP**: *ReSource ReserVation Protocol* ist kein Transportprotokoll, sondern ein Protokoll für die Reservierung von bestimmten Netzressourcen, wie z.B. der Bandbreite in Leitungen, um die Anforderungen der Echtzeitkommunikation zu erfüllen [RFC 2205]. Diese Anforderungen sind unter dem Begriff *Quality of Service* (QoS) bekannt. RSVP wird erweitert und als Signalisierungsprotokoll in (G)MPLS-Netzen verwendet. Dies wird in Abschnitt 12.5 näher dargestellt.
- **OSPF**: *Open Shortest Path First* ist ein Routing-Protokoll, das vor allem bei Internet-Routern Verwendung findet [RFC 5340]. OSPF wird in Abschnitt 11.3 ausführlich besprochen.
- **QUIC**: *Quick UDP Internet Connections* stellt eine Entwicklung von Google dar und ist ein Zwitter verbindungsloser Kommunikation über UDP mit einem dedizierten Verbindungsmanagement und in der Regel mandatorischen Verschlüsselung. Mittlerweile in RFC 9000 standardisiert, erläutern wir die Grundlagen von QUIC in Abschnitt 4.8.

  Bemerkung: Die *Routing-Protokolle* [Kapitel 11 und Kapitel 12] werden in der Literatur der Netzwerkschicht (Schicht 3) zugeordnet, also der Schicht, in der IP angesiedelt ist. Da die OSPF-Nachrichten direkt in IP-Paketen übermittelt werden, lässt sich OSPF nach den im TCP/IP-Schichtenmodell geltenden Prinzipien nicht der Netzwerkschicht zuordnen, sondern der Transportschicht. Bei der Übermittlung von Nachrichten des Routing-Protokolls RIP (*Routing Information Protocol*) wird UDP verwendet; somit ist RIP der Anwendungsschicht zuzuordnen. Das Routing-Protokoll BGP (*Border Gateway Protocol*) nutzt dagegen TCP und ist daher ebenfalls der Anwendungsschicht zuzuordnen.

1.5.3 Protokolle der Supportschicht und für Echtzeitkommunikation

Mit der wachsenden Durchdringung des Internet und der Substitution der klassischen Telefondienste durch VoIP stellte sich die Notwendigkeit, die Transportdienste der Internetprotokolle stärker auf die Eigenschaften der Anwendungen abzustimmen. In diesem Zusammenhang können die Protokolle wie TLS, DTLS und NTP je nach Sichtweise als *Application-Support-Protokolle* oder als *Transport-Support-Protokolle* betrachtet werden, und die wichtigsten von ihnen sind:

- **TLS**: *Transport Layer Security* ist der Nachfolger der *Secure Socket Layer* (SSL), die von der Firma Netscape entwickelt wurde, um die *Webtransaktionen* zu sichern. Bei TLS [RFC 5246] werden die Daten verschlüsselt, deren Integrität gesichert

und die beiden Kommunikationspartner können sich gegenseitig authentisieren [Abschnitt 7.2].

- **DTLS**: *Datagram TLS* ist die UDP-nutzende Variante von TLS [RFC 6347].
- **IKE**: Das *Internet Key Exchange Protokoll* [RFC 5996] ist ein Supportprotokoll und wird bei IPsec (*IP Security*) hierzu verwendet, damit die IP-Instanzen in zwei kommunizierenden Rechnern vereinbaren können, auf welche die Art die Kommunikation zwischen ihnen geschützt werden soll. Dies wird als Sicherheitsvereinbarung *Security Association* (SA) bezeichnet.
- Die weiteren Protokolle der Supportschicht sind **SOCKetS**, genauer SOCKSv5 gemäß RFC 1928, das als Authentisierungsprotokoll zwischen einem Client und einem Proxy-Server dient [Abschnitt 7.1], sowie das Protokoll **NBoT** (*Network Basic Input Output System (NetBIOS) over TCP*), das als Programmschnittstelle (API) für CIFS, also das *Common Internet File System*, bzw. SAMBA von Windows-Unix-Endsystemen genutzt wird und gemäß RFC 1001 und 1002 Transportdienste zur Übertragung von NetBIOS über IP-Netze bereitstellt.

Echtzeit-kommunikation

Die *Echtzeitkommunikation* fasst eine besondere Klasse von Anwendungen zusammen, deren Anforderungen nicht unmittelbar auf der Transportschicht umgesetzt werden können: Diese verlangen spezielle Implementierungen, die mit dem *Real-time Transport Protocol* (**RTP**) [RFC 3550] und dem 'zuarbeitenden' *Real-time Transport Control Protocol* (**RTCP**) [RFC 3605] sowie dem *Real Time Streaming Protocol* (**RTSP**) [RFC 2326] realisiert wurden. Bei der Echtzeitkommunikation dient SIP (*Session Initiation Protocol*) als *Signalisierungsprotokoll* dazu, Verbindungen auf- und wieder abzubauen.

Abgesicherte Echtzeit-kommunikation

Abgesicherte Echtzeitkommunikation kann in Ergänzung zu den unverschlüsselten Echtzeitprotokollen mittels der Pendants **SRTP** (*Secure RTP*) [RFC 3711] und **SRTCP** (*Secure RTCP*) [RFC 3711] erzielt werden. Hierbei wird das Supportprotokoll MIKEY (*Multimedia Internet KEYing*) verwendet, damit kommunizierende Einrichtungen untereinander vereinbaren können, wie die Kommunikation zwischen ihnen geschützt werden soll.

Zeit-synchronisation

In verteilten Systemen – wie dem Internet – spielt die Zeitsynchronisation der interagierenden Komponenten mitunter eine wichtige Rolle. Hierzu muss die Zeitinformation ausgehend von einer Referenzzeit, den Systemen mitgeteilt werden, was Aufgabe der folgenden Protokolle ist:

- *Das Network Time Protocol* (**NTP**) – aktuell in der Version 4 in RFC 5905 spezifiziert – liefert im Internet hierfür die Basis, steht aber seit einiger Zeit in der Kritik, diese Aufgabe nicht angemessen durchzuführen.
- Das *Precision Time Protocol* (**PTP**) entsprechend dem *IEEE-Standard 1588* stellt hierzu eine Alternative dar, die wir ebenfalls in Abschnitt 7.4 diskutieren.

1.5.4 Komponenten der Anwendungsschicht

In der Anwendungsschicht sind, neben den bereits erwähnten Protokollen für die Echtzeitkommunikation, verschiedene Funktionskomponenten angesiedelt. Diese lassen sich in die folgenden vier Gruppen aufteilen:

- **Anwendungsprotokolle** werden im Weiteren als Protokolle wie z.B. FTP und HTTP verstanden, mit dem sich eine bestimmte Anwendung realisieren lässt.
- **Netzdienstprotokolle** bezeichnen Protokolle (z.B. DHCP), mit dem ein bestimmter Netzdienst erbracht wird. Beispielsweise können mit DHCP-Hilfe die IP-Adressen dem Rechner nach Bedarf dynamisch zugeteilt werden. Dies stellt einen Netzdienst dar. Auch Routing-Protokolle, wie z.B RIP, können als Netzdienstprotokolle betrachtet werden. Ein weiteres, in seiner Bedeutung nicht zu unterschätzendes Netzdienstprotokoll ist der *Zeitstempeldienst*, der z.B. in Form des *Network Time Protocols* (NTP) vorliegt [RFC 5905] und zur Synchronisation von Rechnern und Netzknoten (Router) dient.
- **Benutzerdienstprotokolle** sind spezielle Kommandos unter UNIX und LINUX mit denen (entfernte) Netzdienste in Anspruch genommen werden können. Allgemein werden diese als `r`-*Kommandos* bezeichnet, wobei sowohl verbindungslose Anwendungen wie `rwho`, `rexec` und `rsh`, als auch die verbindungsorientierten Kommandos `rlogin`, `rcp`, `rexec` genutzt werden können.

Je nachdem, ob ein Protokoll der Anwendungsschicht das verbindungsorientierte Transportprotokoll TCP oder das verbindungslose UDP verwendet, lassen sich die Protokolle der Anwendungsschicht als *verbindungsorientiert*, *verbindungslos* bzw. *gemischt* klassifizieren [Abb. 1.5-1].

Abb. 1.5-2 benennt die wichtigsten Funktionskomponenten der Anwendungsschicht.

Verbindungsorientierte Anwendungsprotokolle

Verbindungsorientierte Anwendungsprotokolle sind u.a.:

- **HTTP**: *Hypertext Transport Protocol* ist neben SMTP das wichtigste Anwendungsprotokoll im Internet. HTTP sorgt für die Datenübermittlung zwischen Webbrowser und Webserver. *HTTP over TLS* wird als HTTPS bezeichnet.
- **SMTP**: *Simple Mail Transport Protocol* ermöglicht die Übermittlung von E-Mails im Internet. Heute wird in der Regel das *Extended SMTP* (ESMTP) eingesetzt, das eine 8-Bit-transparente Übermittlung ermöglicht.
- **TELNET** ist ein Protokoll, mit dem sich der Anwender in einer interaktiven Sitzung auf einem entfernten Computer einloggen kann und gilt als Urvater der anwendungsbezogenen TCP/IP- Protokolle.
- **FTP**: *File Transfer Protocol* dient zur Übermittlung von Dateien zwischen zwei über ein IP-Netz verbundenen Rechnern. Es ist bewusst einfach und robust aufgebaut, sodass die Datenübertragung auch über in der Qualität schlechte Verbindungen (z.B. Satellitenkommunikation) möglich ist. FTP kann auch die TLS-Funktion nutzen. Man spricht dann von *FTPS*.

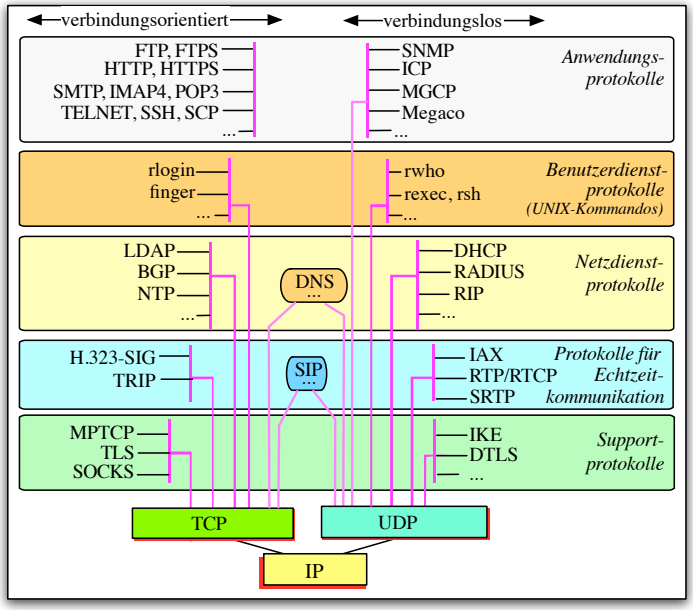


Abb. 1.5-2: Einordnung der Protokolle entsprechend ihrer Kommunikationsschicht

Verbindungslose Anwendungsprotokolle

Verbindungslose Anwendungsprotokolle sind u.a.:

- **SNMP**: *Simple Network Management Protocol* ermöglicht die Abfrage der Zustände von Netzwerkkomponenten und liegt dem Netzwerkmanagement zugrunde.
- **ICP**: *Internet Cache Protocol* ist ein Protokoll, nach dem Web-Caching-Systeme im Internet kooperieren [BRS03].
- **MGCP**: *Media Gateway Control Protocol* dient zwischen den *VoIP-Gateways* für die Anbindung herkömmlicher Komponenten an VoIP-Systeme [Bad22]. Das Protokoll Megaco entspricht der Funktion nach dem MGCP.

Verbindungsorientierte Netzdienstprotokolle

Verbindungsorientierte Netzdienstprotokolle sind u.a.:

- **NTP**: *Network Time Protocol* realisiert die Zeitsynchronisation der Rechner und Netzkomponenten im Internet [RFC 5905]. Einige Client/Server-Anwendungen verlangen die gleichen 'Uhrzeit' zu ihrem Funktionieren.
- **BGP**: *Border Gateway Protocol* dient der Übermittlung von Routing-Informationen zwischen autonomen Systemen (AS) [Abschnitt 11.4].
- **LDAP**: *Lightweight Directory Access Protocol* verwendet man bei der Realisierung verteilter Verzeichnisdienste und wird vor allem als Backend für die Benutzerauthentisierung genutzt [Abschnitt 15.3].

Verbindungslose Netzdienstprotokolle

Verbindungslose Netzdienstprotokolle sind u.a.:

- **DHCP**: *Dynamic Host Configuration Protocol* kann die dynamische Vergabe von IP-Adressen und weiterer Netzparameter übernehmen. DHCP wird im Abschnitt 6.2 beschrieben.

- **RADIUS**: *Remote Dial-In User Service* wird in Abschnitt 6.6 besprochen und ermöglicht die Berechtigungsprüfung von Benutzern, die auf Netzressourcen zugreifen wollen. Dies kann beim Provider-Zugang ins Internet, beim Anmelden im WLAN aber bereits auch am Anschluss an einen Ethernet-Switch der Fall sein. Das Nachfolge-Protokoll *Diameter* [RFC 6733] wird hauptsächlich im IMS (*IP Multimedia Subsystem*) eingesetzt [Bad22].
- **RIP**: *Routing Information Protocol* dient als internes Routing-Protokoll vornehmlich in kleineren IP-Netzen.

Das wohl wichtigste Protokoll im Internet ist **DNS** (*Domain Name System*), das sowohl TCP als auch UDP nutzt [Kapitel 4]. DNS ist ein gemischtes Netzdienstprotokoll.

Protokolle für die Echtzeit-kommunikation

Protokolle zur Unterstützung der Echtzeitkommunikation sind u.a.:

- **RTP**: *Real-time Transport Protocol* hat die Aufgabe, zeitkritische Anwendungen wie Audio- und Videokommunikation über ein IP-Netz zu unterstützen. Ihm steht RTCP (*RTP Control Protocol*) zur Seite. RTP ist die Grundlage für VoIP und für WebRTC. Eine erweiterte RTP-Version zur sicheren Audio- und Videokommunikation trägt die Bezeichnung SRTP (*Secure RTP*) [Bad22].
- **SIP**: Das *Session Initiation Protocol* dient als sogenanntes Signalisierungsprotokoll bei der Echtzeitkommunikation und wird hauptsächlich über UDP eingesetzt; es kann aber auch TCP nutzen.
- **IAX**: *Inter-Asterisk eXchange* ist ein kombiniertes Protokoll für die Signalisierung (z.B. bei VoIP) und für den Transport von Echtzeitdaten (Audio, Video) über IP-Netze. Die Version 2 von IAX beschreibt das IETF-Dokument [RFC 5456]. IAX2 nutzt UDP für den Transport seiner Nachrichten. Bei IAX2 unterscheidet man zwischen *zuverlässigen* und *unzuverlässigen* Nachrichten. Die zuverlässigen Nachrichten transportieren die *Signalisierungsangaben* und werden von der Empfangsseite bestätigt. Die unzuverlässigen Nachrichten transportieren Echtzeitdaten und werden nicht bestätigt. IAX2 hat viel gemeinsam mit dem Protokoll SCTP.
- **TRIP**: *Telephony Routing over IP* wurde der Übermittlung von Routing-Informationen zwischen autonomen Systemen für die VoIP-Unterstützung vorgesehen und daher gilt TRIP als Bruder von BGP [Bad22].

Signalisierungs-protokolle

Bei der Echtzeitkommunikation wischen kommunizierenden Endeinrichtungen müssen für Verbindungen auf- und abgebaut werden. Folglich benötigt man spezielle Protokolle, die dies zwischen IP-Telefonen bei VoIP, wie auch bei Webbrowsern unter Nutzung des WebRTC-Dienstes realisieren. Diese Protokolle werden als *Signalisierungsprotokolle* bezeichnet. Neben dem Protokoll SIP gehört hierzu die Signalisierung nach dem ITU-T-Standard H.323 (kurz H.323-SIG), die über TCP abgewickelt wird. Für weitere Informationen sei auf [Bad22] verwiesen.

1.6 IETF und Internet-Standards

IETF

Um die Weiterentwicklung des Internet und seine Anwendungen voranzutreiben, wurde die Organisation *Internet Engineering Task Force* (IETF) gegründet. Zu ihren Aufgaben gehört die Koordination sämtlicher Aktivitäten, die mit der technologischen

Weiterentwicklung und der Standardisierung der Internetdienste und -protokolle zusammenhängen. Die IETF-Dokumente werden als RFC (*Request for Comments*) im Internet veröffentlicht.

RFC als Internet-Standards

Ein Schlüssel zur raschen Entwicklung des Internet und der IP-Netze ist vor allem der offene Zugang zu den als RFC im Internet veröffentlichten IETF-Dokumenten, die als *Internet-Standards* dienen. Außerdem kann jeder einen neuen RFC vorschlagen, wobei die Vorgehensweise RFC 5000 festlegt.

Verzeichnis der RFC

RFC reichen bis ins Jahr 1969 zum Vorläufer des Internet zurück. In Oktober 2018 liegen bereits über 8500 veröffentlichte RFC vor. Alle RFC sind auf mehreren Rechnern im Internet abgespeichert und kostenlos für jeden Nutzer verfügbar. Ein Verzeichnis aller RFC, die vom *RFC Editor* verwaltet wird, ist unter der Adresse `https://www.rfc-editor.org/rfcsearch.html` zu finden. Die Suche in dieser Datenbank kann durch die Angabe der Nummer des gesuchten RFC oder durch die Angabe eines Suchkriteriums (z.B. Name eines Protokolls wie IP, TCP, OSPF, ...) erfolgen oder alternativ über `https://www.rfc-editor.org/rfc-index2.html`.

Organisation der IETF

Der Erfolg des Internet ist teilweise der gut durchdachten Organisation der Zusammenarbeit zwischen der IETF und den anderen Institutionen zu verdanken. Welche Institutionen an der Entstehung von Internet-Standards beteiligt sind und wie sie zueinander stehen, zeigt Abb. 1.6-1.

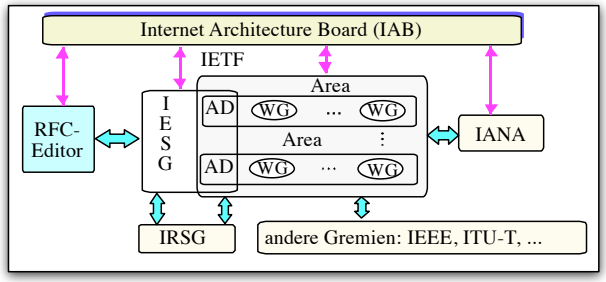


Abb. 1.6-1: Organisation der IETF und die Zusammenarbeit mit anderen Internet-Gremien
AD: Area Director, IANA: Internet Assigned Numbers Authority, IESG: Internet Engineering Steering Group, IRSG: Internet Research Steering Group, WG: Working Group

IAB und RFC Editor

Die Entwicklung des Internet wird vom *Internet Architecture Board* (IAB) in Zusammenarbeit mit der *Internet Research Task Force* (IRTF) [`https://ietf.org`] koordiniert. Dem Vorsitzenden des IAB wurde der Titel *Internet Architect* verliehen. Außerdem wurde im IAB der Posten des RFC Editor eingerichtet, der jeden RFC prüfen und zur Veröffentlichung vorbereiten soll.

Area als Themenbereich

Da die Palette von Entwicklungen um das Internet und deren Anwendungsaspekte herum sehr breit ist, werden bei der IETF bestimmte Themenbereiche definiert. Ein Themenbereich wird als *Area* bezeichnet. In jeder Area wird ein *Area Director* (AD) benannt, der die Aktivitäten innerhalb der Area koordiniert. Es existieren u.a. folgende Areas: *Applications Area*, *Internet Area*, *Routing Area*, *Security Area*, *Transport Area*.

Working Groups

Für die Entwicklung von Standards zu den einzelnen Themen in jeder Area werden mehrere *Working Groups* (WGs) gebildet. Eine WG übernimmt die Verantwortung

für die Entwicklung von Standards, die in der Regel ein Thema (z.B. ein Protokoll oder eine Applikation) betreffen. Eine Auflistung von WGs findet man unter: https://www.ietf.org/html.charters/wg-dir.html

Hervorzuheben sind u.a. folgende aktive WGs (Stand Oktober 2013):

- Internet Area: dhc (*Dynamic Host Configuration*), 6man (*IPv6 Maintenane 6*), dmm (*Distributed Mobility Management*), mip4 (*Mobility for IPv4*)
- Routing Area: ccamp (*Common Control and Measurement Plane*), mpls (*Multiprotocol Label Switching*), ospf (*Open Shortest Path First*), pim (*Protocol Independent Multicast*), pwe3 (*Pseudowire Emulation Edge to Edge*), rtgwg (*Routing Area Working Group*), sidr (*Secure Inter-Domain Routing*)
- Transport Area: aqm (*Active Queue Management and Packet Scheduling*), ippm (*IP Performance Metrics*), mptcp (*Multipath TCP*), tcpm (*TCP Maintenance and Minor Extensions*), tsvwg (*Transport Area Working Group*)
- Real-Time Applications and Infrastructure Area: avtcore (*Audio/Video Transport COre Maintenance*), rtcweb (*Real-Time Communication in WEB-browsers*), sipcore (*Session Initiation Protocol Core*)

Um die Entwicklung der Internet-Standards zu verfolgen und eine gut strukturierte Übersicht über die Internet-Drafts zu erhalten, verweisen wir auf die Seite https://www.potaroo.net/ietf/html/xids-all.html. Für einen schnellen und übersichtlichen Zugriff auf alle IETF Working-Groups und deren Dokumente, ist die Adresse https://www.in2eps.com/x0/tk-ietf-wg-lists.html zu empfehlen.

IESG

Für die technische Verwaltung von IETF-Aktivitäten ist die *Internet Engineering Steering Group* IESG verantwortlich. Zur IESG gehören die Direktoren der einzelnen Areas, die ADs. Der Entwurf jedes Internet-Standards, den man als *Internet Draft* bezeichnet, wird vor seiner Spezifikation als RFC innerhalb der IESG diskutiert. Ein Internet Draft wird nur mit der Zustimmung der IESG als Internet-Standard veröffentlicht. Die IESG arbeitet mit dem RFC-Editor zusammen, der für die Veröffentlichung der RFC zuständig ist.

IANA

Eine besondere Rolle unter den Internet-Gremien spielt die *Internet Assigned Numbers Authority* IANA. Sie dient als zentrale Stelle für die Registrierung von Internet-Adressen, -Namen, Protokollnummern und anderen Parametern, die weltweit eindeutig sein müssen [https://www.iana.org/numbers.html].

1.7 Schlussbemerkungen

In diesem Kapitel wurden in komprimierter Form vor allem die notwendigen Grundlagen dargestellt, die für die Beschreibung von Ideen, Kommunikationsprotokollen und System- und Sicherheitslösungen für IP-Netze in den weiteren Kapiteln hilfreich sind. Abschließend sei noch auf Folgendes hingewiesen:

Web-Technologien

- Das Internet verdankt die heutige Popularität hauptsächlich dem *Web* mit dem Protokoll HTTP. Der Webdienst bedeutet heute nicht nur TCP/IP und HTTP. Für seine effiziente Realisierung werden verschiedene Technologien eingesetzt, sodass man von *Webtechnologien* spricht. Zu ihnen gehören u.a. die Konzepte und Protokolle

für *Web-Switching*, *Web-Caching*, *Content Delivery Networks* sowie für verschiedene Arten von Web-Services. Diese Aspekte werden in diesem Buch außer Acht gelassen; für Näheres sei auf [BRS03] verwiesen.

- Ein wichtiger Trend bei der Weiterentwicklung des Internet ist die Unterstützung der Echtzeitkommunikation, und hierfür waren verschiedene Konzepte und Protokolle zur Übermittlung von Audio und Video über IP-Netze notwendig. Bei der im Internet zunehmenden Echtzeitkommunikation handelt es sich um audiovisuelle Kommunikation, und man spricht in diesem Zusammenhang auch von VoIP (*Voice over IP*) bzw. auch von MMoIP (*Multi-Media over IP*). Für den Transport audiovisueller 'Daten' in IP-Netzen wurde RTP (*Real-time Transport Protocol*) entwickelt [Abschnitt 7.3]. Die audiovisuelle Kommunikation stellt eine Art Videotelefonie dar. Hierbei benötigt man ein *Signalisierungsprotokoll*, um u.a. virtuelle Verbindungen (*Sessions*) auf- und abzubauen. SIP (*Session Initiation Protocol*) ist ein derartiges Protokoll [Abschnitt 7.4]. Zur Realisierung der audiovisuellen Kommunikation im Internet kann auch der Webdienst mit dem Protokoll HTTP eingesetzt werden, was unter dem Begriff WebRTC (*Web Real-Time Communication*) geführt wird. Bei WebRTC dienen Webbrowser zusätzlich als Soft-Videotelefone, und sie werden entsprechend an Webserver virtuell angebunden, sodass die Webserver als quasi Vermittlungsknoten beim Auf- und Abbau von virtuellen Verbindungen zwischen Soft-Videotelefonen fungieren. Wegen Platzmangel wurde aber hier auf die Darstellung von WebRTC verzichtet; für weitere Informationen über WebRTC sei auf das Wissensportal [Bad14] verwiesen.

  Echtzeitkommunikation: VoIP, MMoIP

- Zur Übertragung der IP-Pakete auf das Medium wird zusätzlich zur physikalischen Anbindung die Datensicherungsschicht [Abb. 1.3-2] benötigt. Im Zuge der TCP/IP-Einführung hatte diese zunächst an Bedeutung verloren, erfährt aber derzeit eine Renaissance bei den *Software-Defined Networks* (SDN): *Virtuelle Interfaces* ermöglichen die Steuerung der Kommunikation bei Cloud-Services und bilden somit einen wichtigen Bestandteil des Sicherheitskonzepts.

  Programmierbare MAC-Schicht

- Die Integration verschiedener, in der Regel drahtloser Sensor-Aktor-Netze in das herkömmliche Internet führt zur Entstehung des *Internet of Things* (IoT) bzw. auf Deutsch des *Internet der Dinge*. Das IoT ist eine funktionelle Erweiterung des Internet mit dem Ziel, Alltagseinrichtungen (Geräte, Sensoren) unterschiedlicher Art und mit unterschiedlichen Fähigkeiten – also verschiedene smarte Dinge – sowohl untereinander als auch mit Rechnern am Internet so zu vernetzen, dass sie alle möglichen Internetdienste nutzen können, um dadurch die Erbringung einer breiten Palette neuer innovativer Services überall und jederzeit zu ermöglichen. Dank des IoT werden bald alle technischen Dinge, insbesondere die unseres alltäglichen Lebens, den Menschen überall und jederzeit zugänglich und somit nutzbar sein.

  Internet of Things – als funktionelle Erweiterung des Internet

- Die Komplexität des zukünftigen Internet und der Weiterentwicklung seiner Anwendungen kann an dieser Stelle auch nicht annähernd dargestellt. Daher greifen wir die Diskussion nach Vorstellung der bestehenden Grundlagen in den folgenden 17 Kapitel erneut im abschließenden Kapitel 18 auf. Leser des Ebooks können hier auf die zugrunde liegenden Quellen unmittelbar über das Internet zugreifen.

  Weiterentwicklung des Internet

1.8 Verständnisfragen

- Rekapitulieren Sie den Aufbau des OSI-Referenzmodells.
- Welche Schicht hat welche Funktion?
- Wie sieht das TCP/IP-Modell aus?
- Was ist verbindungslose versus verbindungsorientierte Kommunikation?
- Was versteht man unter Fehler- und Flusskontrolle?
- Was ist der Unterschied zwischen *Service Datagram Unit* (SDU) und *Protocol Datagram Unit* (PDU)?
- Was ist ein Frame und was ein 'Service Access Point' (SAP)?
- Was ist ein 'Payload'?
- Welche Aufgabe muss im Schichtenmodell beim Übergang einer Informationseinheit von Schicht $N-1 \rightarrow N$ und umgekehrt von Schicht $N \rightarrow N+1$ realisiert werden?
- Welche Dienste stellt IP bereit?
- Was unterscheidet TCP von UDP?
- Wie ist es möglich, eine TCP/IP-Verbindung zwischen zwei Endknoten über verschiedene Netze, z.B. WLAN und ein Mobilfunknetz, durchzuführen?
- Welche Ideen vereint das von Google ins Leben gerufene QUIC-Protokoll?
- Welche beiden Protokolle können zur Zeitsynchronisation zwischen Komponenten in Netzen genutzt werden?
- Für welche Internet-Dokumente ist das IETF zuständig, und wie wird hier eine Abstimmung erzielt?

2 Sicherheit in der IP-Kommunikation

Die Architektur der TCP/IP-Protokollfamilie sah zunächst nur technische *Sicherungsmaßnahmen* für die Kommunikation vor: War zunächst TCP/IP als quasi geschlossenes System im Rahmen des ARPANet vorgesehen, so war die in den 70er und 80er Jahren des letzten Jahrhunderts vorherrschende Nutzung des Internet im wissenschaftlichen Betrieb zu sehen. Die Qualifizierung an seiner Teilnahme erfolgte ausschließlich durch die vorhandenen technischen Möglichkeiten seiner Teilnehmer.

Alice, Bob und Eve

Der erste Schritt bestand natürlich darin, das Internet überhaupt für die Anwender bereitzustellen, also die *Verfügbarkeit* (*Availability*) zu sichern. Die nächste zentrale Anforderung bestand daran, dass die Daten vollständig und unverfälscht von *A* nach *B* (über *C*) gelangen und zudem die *Integrität* der Nachrichten (= *Datenpakete*) zu gewährleisten. Mit der kommerziellen und quasi privaten Nutzung des Internet gewann die vertrauliche Übertragung von Nachrichten eine bedeutende Rolle: Aus *A* und *B* wurden *Alice* und *Bob*, die ihre Internet-Konversation gegenüber dem *Eavesdropper Eve* schützen müssen:

Überblick über das Kapitel

Entsprechend der aktuellen Diskussion, haben wird dieses Kapitel wie folgt gestaltet:

Daten im System

- Zunächst wollen wir eine Replik auf die technische Entwicklung der IT-Sicherheit und ihrer Wurzeln vornehmen, die Rolle der Daten und der mit ihnen umgehenden Systeme, d.h. der Akteure, in einem Modell beschreiben, was in der Darstellung der heute genutzten vier Primitive der IT-Security mündet [Abschnitt 2.1].

Krypto-Primitive

- Der Sicherung der Datenübertragung und -speicherung ist der folgende Abschnitt gewidmet, wobei hier die Ideen der vier Krypto-Primitive entwickelt werden, von denen anschließend umfangreich Gebrauch gemacht wird [Abschnitt 2.2].

Klassische Verschlüsselung und Hashes

- Die klassische symmetrische Verschlüsselung findet sich im nächsten Abschnitt wieder; wobei wir auf Strom- und Blockchiffren sowie auf die Verschränkung der verschlüsselten Datenblöcke im Betriebsmode eingehen. Zudem werden auch Hashfunktionen und ihre heutige Nutzung ausführlich behandelt.

Public-Key-Kryptographie

- Die *Public-Key-Kryptographie* liefert uns die beiden Krypto-Primitive *Schlüsseltausch* und *Signierung*, wobei der Schlüsseltausch speziell auf Grundlage des RSA- als auch der Diffie-Hellman-Algorithmen und die Nutzung digitaler Signaturen mittels X.509-Zertifikaten vorgestellt wird [Abschnitt 2.5].

ECC-Kryptographie

- Neu in diesem Abschnitt ist die moderne Kryptographie mittels Elliptischer Kurven, die sich sowohl für den Schlüsseltausch, als auch für Signaturoperationen in idealer Weise eignen [Abschnitt 2.6].

Digitale Identitäten

- In den IT-Systemen wird nicht anonym agiert, sondern sowohl die Benutzer als auch die Systeme besitzen eine *digitale Identität* und sind somit authentisierbar. Wie dies umgesetzt werden kann, wollen wir ebenfalls beleuchten [Abschnitt 2.7].
- Abschließend wollen wir uns in Abschnitt 2.8 der sicheren und vertraulichen Kommunikation über IP-Netze widmen.

2.1 Grundlagen und Entwicklung der IT-Sicherheit

Mit den *Snowden*-Veröffentlichungen und den vermeintlichen 'russischen Hackerangriffen' sind IT-Sicherheit und Cybersecurity in aller Munde. Spätestens mit dem Inkrafttreten der Datenschutzgrundverordnung (DSGV) im Jahr 2018 ist auch Datenschutz aus der Ecke des 'Datenschutzbeauftragten' (DSB) in die öffentliche Wahrnehmung gerückt.

Der Gegenstand und das Zusammenspiel von IT-Sicherheit und Datenschutz wird häufig bis hinauf in kompetente Stellen wie dem Bundesamt für Sicherheit im Informationswesen (BSI) nicht klar differenziert, sondern gemeinsam als *Informationssicherheit* zusammengefasst. In klassischer Lesart möchten wir aber unterscheiden in Bezug auf

- den *Datenschutz* – vor allem in Form digital vorliegender Informationen – was ihre Vertraulichkeit betrifft,
- der *Kryptographie* – die das mathematische Handwerkzeug liefert, den Datenschutz auf allen Ebenen zu gewährleisten, sowie
- die *IT-Security* – die sich auf die IT-Systeme bezieht und wie die Schutzmaßnahmen umgesetzt und sichergestellt werden können.

Letzteres umreißt das bekannte 'magische IT-Security-Dreieck' (CIA), das aus der militärischen Nomenklatur entsprungen ist und die *Schutzziele* der IT-Security beinhaltet:

Confidentiality
- Schutz vor *Spionage* durch Sicherstellung der Vertraulichkeit,

Integrity
- Schutz vor Verfälschungen der Daten im Hinblick auf *Korrumpierung* und *Manipulation*,

Availability
- Schutz vor Datenverlusten und Ausfall der IT-Infrastruktur durch technische Fehler und *Sabotage*.

Obwohl das Internet dem ARPANet [Abb. 1.1-1] entsprungen ist, haben bei der Entwicklung der Netzwerkkommunikation diese militärischen Ziele nicht unmittelbar im Vordergrund gestanden und wurden nur da beachtet, wo es unerlässlich ist, wurde doch das Internet als prinzipiell offenes System [Abb. 1.3-4] verstanden.

Im Grunde kann man das Internet als 'Informationsmarkt' verstehen, der über nahezu beliebige und kostengünstige Ressourcen verfügt und zudem – in seiner heutigen Ausprägung – für praktisch jedermann unbeschränkt nutzbar ist. Wie auf jedem offenen, ungeregelten Markt gibt es Akteure mit ganz unterschiedlichen Interessen: Von der Verbreitung von Wissen und Know-how über kostenpflichtige Dienste bis zu den bekannten 'Fake News'-Verbreitern. Es ist wichtig, dieses Zusammenspiel zu verstehen und die daraus erwachsenden Konsequenzen zu berücksichtigen.

2.1.1 Daten und ihre Nutzung

Abb. 2.1-1 zeigt ein einfaches Modell der Datennutzung, was aber bereits die wesentlichen Merkmale in Form einer Grammatik beschreibt:

- *Subjekte* sind IT-Systeme und Netze, aber auch Menschen,
- die *Operationen* (Verarbeiten, Speichern, Transportieren)
- von *Datenobjekten* in verschiedenen Ausprägungen vornehmen.

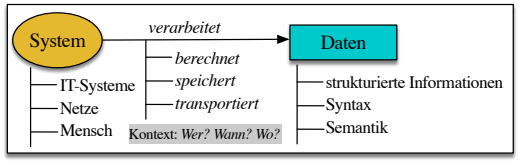


Abb. 2.1-1: Modell der Datenverarbeitung

Daten begreifen wir als strukturierte Informationen, die sowohl *Syntax* als auch *Semantik* und zudem einen *Kontext* aufweisen.

Daten im Netz → Datenpakete

Bei der Datenübertragung über Netze werden die Daten in Nachrichten-Container [Abb. 1.5-1] variabler Größe gesteckt und üblicherweise als *Pakete* aufgefasst, die einen Paket-Header besitzen und in Form von *Frames* ebenso einen Paket-Trailer. Dieses Konzept hat sich als überaus erfolgreich erwiesen und alle anderen Technologien wie z.B. die verbindungsorientierte Kommunikation in Gänze verdrängt.

Daten in Verarbeitung → Datenworte

Bei der Verarbeitung der Daten spielt die Länge eines *Datenworts* ein hervorgehobene Rolle. Daten, Instruktionen und Hauptspeicheradressen werden in dieser Struktur uniform beschrieben. Während die vorige Rechnergeneration das Datenwort auf 32 Bit begrenzte, arbeiten heute Rechnerarchitekturen mit einer internen 64-Bit-Darstellung, sei es in *Least-Significant Bit* (first) LSB oder *Most-Significant Bit* (first) MSB Organisation. Mittels der 64-Bit-Repräsentierung können $2^{64} - 1$ Byte adressiert werden, was einer Datenmenge von mehr als 18,446 Trillionen Byte entspricht. Die Umstellung der Daten in die verschiedenen Formate wird quasi 'en passant' während des entsprechenden Verarbeitungsvorgangs vorgenommen und ist für den Benutzer heute nicht mehr von Bedeutung.

Der Anwender ist ausschließlich an den Dateninhalten interessiert, wobei die Menge der zu verarbeitenden Daten immer noch eine begrenzende Rolle spielt – speziell dann, wenn die Syntax nicht optimal gewählt ist und andauernd Umrechnungen in verschiedene Datenformate erforderlich sind, die sich stark auf die Performance auswirken.

Data in flight

Aufgabe der Technik der Netze ist es natürlich, die Daten zu transportieren, und zwar dergestalt, dass diese *verfügbar*, *unverfälscht* und ggf. *vertraulich* zwischen den Teilnehmern ausgetauscht werden können. Diese Aspekte werden wir in den folgenden Abschnitten im Detail erläutern.

Eine weiteres wichtiges Kriterium besteht darin, die *Authentizität* der Datenquelle (Sender) und der Datensenke (Empfänger) sicherzustellen, sodass Daten mit der notwendigen *Autorisierung* und *Berechtigung* übertragen werden können.

Data at rest

Die langfristige Verfügbarkeit von Daten wird durch ihre persistente Speicherung ermöglicht, die in der Regel auf dem physikalischen Medium blockweise erfolgt. Diese Datenblöcke lassen sich sowohl im Hinblick auf ihre Integrität durch Checkhsummen sichern, wobei dieses auf dem Datenträger selbst und ergänzend vom Dateisystem (wie z.B. ZFS[1]) vorgenommen wird.

Dem Kriterium der Vertraulichkeit kann durch blockweise Verschlüsselung entsprochen werden; die Verfügbarkeit wird durch Redundanzen (z.B. RAID-1[2]) oder mittels Backup/Restore realisiert.

Data in computation

Die Verarbeitung der Daten erfolgt im heutigen Verständnis unverschlüsselt. Daten im Hauptspeicher des Rechners, aber auch in den CPU-Caches werden durch ergänzende Paritätsinformationen gegen Verfälschungen gesichert, sind aber – als transiente Daten – nur solange verfügbar, solange die CPU und der Hauptspeicher mit elektrischer Energie versorgt werden. Die Sicherstellung der Datenintegrität auch in oder nach einem irregulären Betriebszustand muss von der Software-Architektur gewährleistet werden.

Shared IT → Data Leakage

Die Rechnerressourcen, aber auch die Netze werden von mehreren (quasi) zeitgleich laufenden Programmen gemeinsam genutzt, wobei der *Kernel* des Betriebssystems die Aufgabe hat, diese gegeneinander abzugrenzen und somit eine unbeabsichtigte Datenweitergabe (*data leakage*) zu unterbinden. Dass dies nicht so einfach ist, haben die *Spectre* genannten Fehler im Design vor allem der Intel-CPUs gezeigt. Dies gilt im besonderen, da heutige Systeme nicht *single-use*, sondern durch die Virtualisierung [vgl. Kapitel 14] im Rechenzentrum oder in der 'Cloud' [vgl. Kapitel 17] von ganz unterschiedlichen Benutzern und Benutzergruppen geteilt werden.

Erweiterte Datenattribute: Metadaten

Entsprechend Abb. 2.1-1 sind Daten komplexe Objekte mit Syntax und Semantik. Neben diesen inhärenten Eigenschaften besitzen Daten auch *kontextuelle Attribute*, die wir auch als erweiterte Datenattribute bzw. Metadaten bezeichnen und bei der Datenverarbeitung entweder automatisch anfallen oder von prinzipiellem Belang sind:

Gültigkeit
- Wann wurden die Daten erzeugt und wie lange sind die hierin enthaltenen Informationen gültig bzw. von Belang?

Herkunft
- Wer ist der Erzeugende der Daten (die Datenquelle)?

Vertraulichkeit
- Welche Weiterverarbeitungs- bzw. Weitergaberechte sind mit den Daten verknüpft? Im einfachsten Falle lässt sich dies durch die Klassifikation der Daten in 'vertraulich', 'privat' und 'öffentlich' beschreiben.

Metadaten und Datamining

Diese kontextuellen Datenattribute sind nun ihrerseits *Metadaten*, für die im Grunde die gleichen Bedingungen wie für die eigentlichen Daten gelten. Die Datennutzung findet in der Regel *systemisch* statt, d.h. unter Einbeziehung der Metadaten, wie das

[1] Solaris Zeta File System ZFS
[2] RAID = Redundant Array of Inexpensive Drives

für *Datamining* heute typisch ist. Das Verständnis der Daten wird dadurch nicht nur von ihrer Semantik, d.h. dem Inhalt bestimmt, sondern ganz wesentlich durch die ergänzenden Metadaten[3].

Die Interpretation von Daten bezieht immer auch den bestehenden Kontext, d.h. die Metadaten, mit ein, durch die die Semantik angereichert wird. Kontextdaten können unterschiedlichere Attribute als die eigentlichen Nutzdaten aufweisen.

2.1.2 Rolle der IT-Security

IT-Systeme, die naturgemäß Daten verarbeiten, müssen aufgrund ihrer Architektur und dem definierten Workflow auf diese Attribute Rücksicht nehmen, was besonders auch der Gesetzgeber im Rahmen der 'Datenschutzgrundverordnung' (DSGVO) fordert.

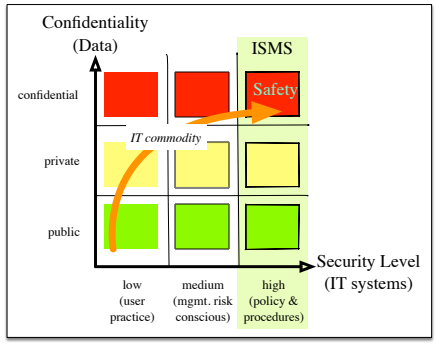


Abb. 2.1-2: Zusammenspiel von Datensicherheit und IT-Security
ISMS: IT Security Management System, mgmt.: Management

Während die Kryptographie – wie wir in den nachfolgenden Abschnitt sehen werden –die Möglichkeit bietet, die avisierten (Daten-)Schutzziele zu erreichen, kann die Rolle der IT-Security wie folgt charakterisiert werden:

Aufgabe der IT-Security

Aufgabe der IT-Security ist es,

- 'Best Practice' Verfahren für den Datenschutz und den Sicherheitsstand der IT-System und der IT-Infrastruktur zu benennen und ggf. einzufordern und
- die Konsequenzen mangelhafter Kryptographie und unzureichendem Systemschutz und nicht-adäquater IT-Infrastruktur im Falle eines 'Incidents' (Sicherheitsvorfalls) zu *mitigieren* und damit seine Auswirkungen unter Kontrolle zu halten bzw. zu minimieren.

IT-Security hat somit nicht die intrinsische Aufgaben, die prinzipiellen Schwachstellen in den Systemen und somit (potentiellen) Ursachen für einen Incident zu eliminieren – sprich die IT-Systeme 'besser' und 'sicherer' zu machen – sondern Sicherheitsvorfälle

[3] Dies lässt sich durch folgende Anekdote trefflich illustrieren: https://www.wired.de/article/das-us-militaer-verbietet-fitness-tracking-mit-gps

weniger wahrscheinlich eintreten zu lassen und diese somit vor allem 'managebar' zu gestalten[4]. Hierzu dienen dann ergänzende IT-Systeme wie Firewalls und Virenscanner, die allerdings selbst wiederum ein Sicherheitsrisiko in sich tragen.

Zertifizierungen und Change Management

Ergänzend lässt sich sagen, dass dies auch die Rolle des BSI und anderer 'IT-Security' Dienstleister ist. Zertifizierungen im IT-Security Umfeld dienen in erster Linie dazu, potentielle Risiken und Schwachstellen zu erkennen und diese im Rahmen eines *Incident-* und *Change Management-Process* mit anderen Firmenprozessen kompatibel zu gestalten.

Data Safety

Data Safety

Dieser Zusammenhang wird durch den Terminus *Data Safety* verdeutlicht. Abb. 2.1-2 zeigt das Zusammenspiel vom Vertraulichkeitsattribut und der Qualifikation von IT-Systemen: Systeme auf niedriger IT-Security-Stufe, z.B. auch IoT-Devices [vgl. Kapitel 17], sollten nur öffentliche und Kontext-irrelevante Daten verarbeiten, während vertrauliche Daten eines hohen IT-Security-Niveaus bedürfen. Im FinTec-Bereich und speziell in Banken wird dem durch die Einführung eines expliziten *IT Security Management Systems* (ISMS) Rechnung getragen.

IT-Commodity-Systeme

Faktisch unterscheiden sich die kryptographischen Algorithmen nicht zwischen sogenannten IT-Commodity-Systemen wie IoT-Devices, Tablets und PCs von denen Systemen in der FinTec-Welt. Allerdings wird bei den letzteren mit speziellen Werkzeugen die Handhabung vertraulicher Daten gemonitort.

Nicht vergessen dürfen wir, dass IT-Systeme heute auch umfangreich in Geräten des täglichen Alltags, Autos, Landmaschinen und natürlich auf Flugzeugen eingesetzt werden. Treten hier Störungen auf, können diese lebensgefährliche Auswirkungen aufweisen, speziell dann, wenn diese Störungen durch Einflüsse von Außen (sprich: Hacker) herbeigeführt werden können.

Fehlerhafte Kryptographie

Während die Kryptographie hier eine berechenbare Größe darstellt, die mit mathematischen Modellen arbeitet, ist es die Implementierung und Nutzung von kryptographischen Systemen mitunter nicht. Hierbei können wir folgende Muster identifizieren:

- Der kryptographische Algorithmus wurde *jeopardized*; also absichtlich so entworfen bzw. standardisiert, dass hierdurch nicht das gewünschte Sicherheitsniveau erreicht wird. Beispiele hierfür ist die Standardisierung des DES-Algorithmus[5](mit nur 56 statt 64 Bit Stärke) und die Bereitstellung des Dual EC Zufallszahlengenerators durch die NIST[6].
- Die Implementierung ist *falsch* und entspricht nicht der 'Best Practice'. Hierzu zählt z.B. die Nutzung der RC4-Verschlüsselung ohne *Nonce* bei frühen TLS-Versionen; oder aber auch der Einsatz von AES mit *Cipherblock Chaining* (CBC).
- Die Implementierung ist *fehlerhaft* und ermöglicht z.B. *Side-Channel-Attacken*. Dies kann mitunter sehr subtil sein; aber auch große Konsequenzen aufweisen. In diese Kategorie fällt z.B. der Heartbleed-Fehler bei DTLS.

[4] Im Englischen als *Vulnerabilities Equities Process* bezeichnet; vgl. https://en.wikipedia.org/wiki/Vulnerabilities_Equities_Process.
[5] auch wenn dies 'offiziell' verneint wurde: https://cryptome.org/nsa-inman-1979.pdf
[6] siehe: https://projectbullrun.org/dual-ec/documents/dual-ec-20150731.pdf

Auf einige dieser Probleme wollen wir in den betreffenden Abschnitten eingehen.

2.1.3 Akteure und Identitäten bei der Datenverarbeitung

Jeglicher Datenaustausch bedingt eine *Datenquelle* und eine *Datensenke*. Ist die Datenquelle die Erzeugende, ist sie somit auch der natürliche Dateneigner, und diese stehen quasi zur freien Verfügung. In allen anderen Fällen müssen die Fragen gestellt werden:

- Dürfen die Daten, die von Dritten stammen, weitergegeben werden?
- Ist es zulässig, die Daten an bestimmte Dritte zu übermitteln?

Unabhängig davon, welches Vertraulichkeitsattribut die Daten aufweisen, so ist doch Herkunft und Ziel der Daten zu bestimmen. Technisch gesehen kann das bei Internetgestützten Systemen durch die Angabe der IP-Adressen [siehe Kapitel 3] von Sender und Empfänger realisiert werden. Wie aus Abb. 2.1-1 hervorgeht, ist dies aber nicht ausreichend:

- Es ist vielmehr immer ein *Prozess*, der die Datenverarbeitung vornimmt;
- Netze besitzen hier nur eine Vermittlerrolle, sind aber für die vertrauliche und korrekte Weiterreichung der Daten verantwortlich.
- Falls notwendig, muss zusätzlich die Möglichkeit bestehen, den ausführenden *Menschen* zu identifizieren.

Identitäten

Menschen und Prozessen muss somit eine *Identität* zugewiesen werden, der sie unterscheidbar macht. Im einfachsten Fall ist dies eine fortlaufende Nummer, wie dies z.B. bei Prozessen[7] üblich ist. Bei rechnergestützten Prozessen ergänzt man dies um den Hostnamen, z.B. in der Form *pid@host*. Die pid ist nun aber lediglich die Instanz eines Prozesses, die aktuell ausgeführt wird, der Prozess selbst bekommt in der Regel einen 'sprechenden' Namen wie z.B. *daemon*, der für alle Instanzen benutzt wird, also beispielsweise als *daemon@host*.

Name = Öffentliche Identität

Ebenso wird zur Identifikation von Menschen der *Username* verwandt, der aber natürlich nicht eindeutig sind. Um Personen, aber auch Prozesse ansprechen zu können, müssen diese bekannt sein, wie dies beispielsweise bei Webanwendungen über die URL [Abb. 1.1-4] gemacht wird. Namen kennzeichnen somit eine *öffentliche Identität*, wobei der Grad der Öffentlichkeit auch beschränkt sein kann, z.B. auf eine Benutzergruppe, die Teilnehmer nur über deren *Aliasnamen* bzw. *Pseudonyme* kennt.

Authentisierung

Identitäten müssen gegenseitig und gegenüber Dritten *beglaubigt* werden und eindeutig sein, d.h. immer auf die gleiche Person bzw. den gleichen Prozess verweisen. Die Beglaubigung lässt sich organisatorisch (wie z.B. beim Personalausweis) oder aber kryptographisch lösen und verlangt einen eigenen *Authentisierungsprozess*, der zwei Endzustände kennt:

1. Die Person oder der Prozess kann aufgrund der vorliegenden Informationen nachvollziehbar einer Identität – und nur genau dieser – zugewiesen werden

[7] die Process Identification `pid`

2. Die Informationen widersprechen sich oder sind nicht aussagekräftig genug, um die Identität zweifelsfrei feststellen zu können.

PSK-Verfahren

Das Authentisierungsproblem lässt sich notwendig aber nicht hinreichend lösen, falls zusätzlich zum öffentlichen Teil der Identität ein geheimer, *privater* Teil hinzukommt, der nur den beteiligten Parteien bekannt ist und vertraulich ausgetauscht wurde: das *Passwort*. Daher sprechen wir hier auch von *Pre-Shared Key* PSK-Verfahren.

1-Faktor-Authentisierung

Da die Identität öffentlich bekannt ist, ist das lediglich das Passwort der (kryptographisch) entscheidende Teil, und wir sprechen hierbei von einer *1-Faktor-Authentisierung* (1FA). Neben der Tatsache, dass es anderen nicht bekannt sein darf, ist die Qualität des Passworts im Hinblick auf seine Länge und seine *Entropie*[8] maßgeblich, da es häufig über einen längeren Zeitraum unverändert genutzt wird.

Einmal-Passwörter

'Sichere' Passwörter sollten aber nur ein einziges Mal Einsatz finden, wofür es eine Reihe von technischen Lösungen und Anwendungen gibt:

TLS → PRF oder HKDF

- Bei der IP-Kommunikation zwischen Prozessen werden insbesondere unter Nutzung der *Transport Layer Security* [Abschnitt 7.2] einmalig generierte Passwörter zur Sicherstellung der Authentizität der (verschlüsselten) Datenpakete für jede TLS-Sitzung benutzt. Diese werden mittels einer *Pseudo-Random Function* PRF oder einer *HMAC-based Key Derivation Function* HKDF generiert.

One-Time Password

- Bei einigen *One-Time Password* OTP-Verfahren wird das einmalige Passwort von einem statischen Passwort und dem aktuellen Zeitfenster abhängig gemacht. Hierbei müssen die Komponenten *zeitsynchron* arbeiten.

Gegenbeispiel mit statischen Schlüsseln

> Statisch vergebene Passwörter zur Authentisierung der kommunizierenden Instanzen finden nur bei wenigen Protokollen Einsatz, so wie dem in Abschnitt 15.3 beschriebenen *RADIUS*-Verfahren[9], wo das Passwort zur Verschlüsslung der RADIUS-Nachrichten verwendet wird. Auch der Datenaustausch von Internet-Routen mittels des *Border Gateway Protocols* (BGP, Abschnitt 11.4) erfolgt verschlüsselt über die ausgetauschten Pre-Shared Keys zwischen den Peers.

2FA und MFA

Eine 2-Faktor- (2FA) oder Mehr-Faktor-Authentisierung (MFA) liegt dann vor, wenn zusätzlich zum Wissen (= des Passworts) ein personalisierter Besitzgegenstand im Authentisierungsprozess vorhanden sein muss, z.B. in Form einer Chipkarte (die auf die Identität ausgestellt sein muss). *Biometrische Informationen* wie z.B. Fingerabdruck- oder Retina-Scan können ebenfalls ergänzend für die Authentisierung von Personen herangezogen werden (Mehr-Faktor-Authentisierung). In allen diesen Fällen sind die Zugangsdaten zunächst zu registrieren. Biometrische Daten müssen geeignet durch mathematische Verfahren abgebildet werden, sodass deren Korrektheit und Signifikanz gewährleistet ist.

Biometrie

MCA = Multi-Channel Authentication

Stand der Technik ist heute eine *Mehr-Kanal-Authentisierung* bei der ergänzende Informationen (*Token*) über einen zusätzlichen Übertragungskanal (also Email, SMS, Web) angefordert bzw. mitzuteilen ist. Dreh- und Angelpunkt für die MCA ist ein *registriertes Device*, das in Form eines Mobiltelefons (über die bekannte Rufnummer), eine Applikation des *Identity Service Providers* (IdP), oder die spezifischen Merk-

[8] wir können an dieser Stelle 'Entropie' mit Zufälligkeit gleichsetzen
[9] RADIUS: Remote Dial-In User Service

male z.B. eines Webbrowsers (über dessen gesendete Metadaten). In allen diesen Fällen, wird die Authentisierung an dieses Device gebunden. Dies gilt speziell für die Betriebssysteme iOS und Android, die diesen Service bereits integriert haben.

Autorisierung

In vielen praktischen Anwendungsfällen ist die Identität einer Person oder eines Prozesses von keinem besonderen Belang (und muss auch gar nicht bekannt sein), sondern vielmehr, ob eine Berechtigung vorliegt, beispielsweise einen Service zu nutzen [Abb. 2.1-3].

Shibboleth an Hochschulen

> Die Ausleihe von Fachartikeln und Büchern in digitaler Form an den Hochschulen ist gestattet, wenn der Student eingeschriebenes Mitglied an einer Hochschule ist. Der Verlag – oder die Datenquelle –, von der aus das Material bezogen wird, nutzt dann den sogenannten *Shibboleth*-Dienst[10] der Hochschule: Der Student muss sich dann gegenüber der *eigenen* Hochschule mittels seiner Kennung (häufig Matrikelnummer) und Passwort authentisieren. Nur das Ergebnis der Überprüfung wird an den Verlag weitergereicht.

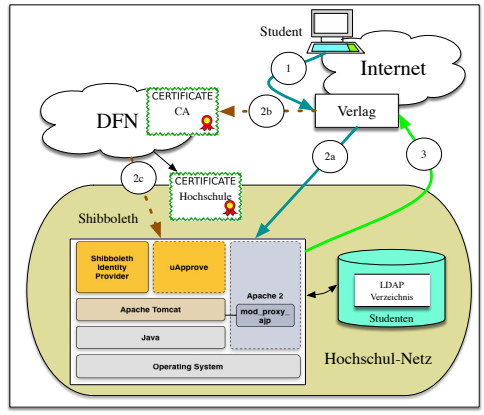


Abb. 2.1-3: Shibboleth als IdP an Hochschulen
(1) Student will über Verlag Fachartikel einsehen, (2a) Anfrage geht an den Shibboleth-Server der HS und Student meldet sich dort an, (3) erfolgreiche Anmeldung geht an Verlag; alternativ: (2b) Anfrage geht über den DFN, der (2c) die Anmeldung an die HS weiterleitet.

Kennzeichnend ist hierbei Folgendes:

- Die Autorisierung ist ein *interpersonelles* Attribut, das von einer Gruppe von Nutzern geteilt werden kann.
- Es liegt eine *transitive* Authentisierung vor, wobei ein spezieller Authentisierungsdienst mittels eines *Identity Service Providers* IdP zum Einsatz kommt

Identity Service Provider

Digitale Identitäten

Will man Authentisierung unabhängig von einer oder mehreren Benutzergruppen machen – wie sie heute beispielsweise auf Grundlage von privaten Anbietern wie *Facebook*, *LinkedIn* und *Google* vorliegen – ist ein System zur Verwaltung digitaler Identitäten zu schaffen. Mittels der modernen *Publik Key*-Kryptographie und dem Aufbau von öffentlichen und privaten Zertifizierungsstellen, den *Certificate Autorities*

[10] siehe: https://www.shibboleth.net

Public Key Infrastructure

CA oder auch *Trust Center* TC, ist dies technisch seit etwa 1995 möglich: *Public Key Infrastructure* PKI. Der zu betreibende Infrastrukturaufwand ist aber insbesondere seitens staatlicher Seite beachtlich, und so hat es bislang nur Estland als relativ kleine Nation geschafft, dies auch in allen Aspekten, die die öffentliche Verwaltung betreffen, zu realisieren.

X.509-Zertifikate

Anders sieht die Situation in Unternehmen aus: Anwendungen werden *X.509-Zertifikate → Digitale Identitäten* ausgestellt, die mit sogenannten *key files* komplettiert werden. Hierdurch erhalten Anwendungen *digitale Identitäten*, die mit bestimmten Rollen und Rechten verknüpft sein können. Auf die technischen Grundlagen hierzu gehen wir in Abschnitt 2.7 noch genauer ein.

2.1.4 Entwicklung der Internet-Kryptographie

Wie bereits Abb. 2.1-1 darstellt, geht es bei der IT-Sicherheit um die Sicherheit, d.h. Vertraulichkeit, Integrität und Verfügbarkeit der von IT-Systemen verarbeiteten Daten. Mit der Entwicklung von IT-Systemen, sprich Computern, seit Endes des 2. Weltkrieges nimmt die IT immer mehr Einzug in das tägliche Leben: Wirtschaft, Kultur und Politik sind ohne Computer nicht mehr vorstellbar. Die Abhängigkeit vom Funktionieren der IT-Systeme ergibt sich insbesondere auch für die IT-gesteuerte Infrastruktur wie Kommunikationssysteme und elektrische Energieversorgung: Die 'analoge' Welt wird durch die digitale abgelöst. Wir benötigen daher nicht nur Datenschutz und IT Security, sondern sowohl die IT-Systeme als auch die Daten müssen zuverlässig zur Verfügung stehen: *IT Safety*.

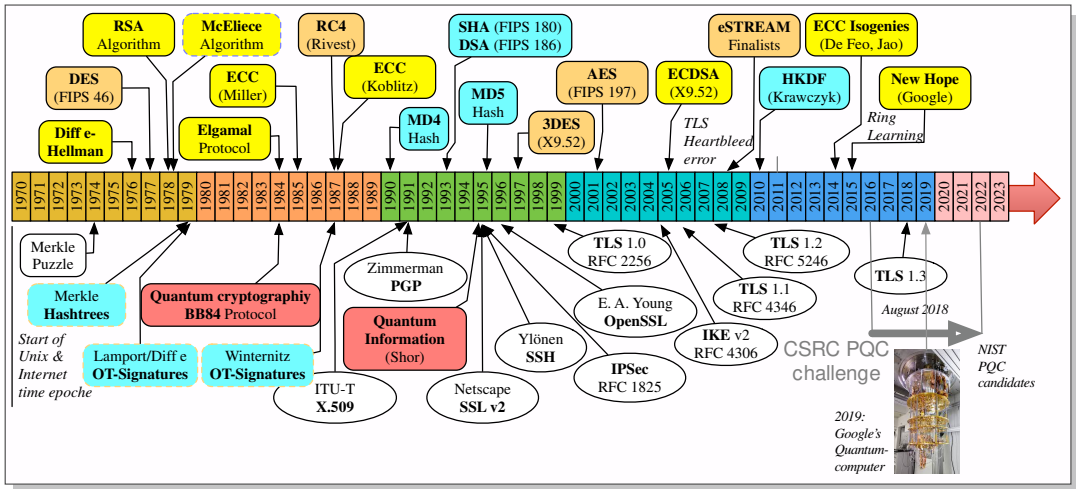


Abb. 2.1-4: Entwicklung und Meilensteine der wichtigsten kryptographischen Errungenschaften der Internet-Zeit.
Oberhalb der Zeitleiste: Wichtige Algorithmen und Standards; unterhalb: Implementierungen; Hashtrees und Quantenkryptographie werden im Buch nicht weiter besprochen; Gelb: Schlüsseltausch- und Signaturprotokolle, orange: symmetrische Verschlüsselung; türkis: Hashverfahren

1.1.1970: Beginn des Internet Zeitalters

Einschneidende Entwicklungen sind seit Beginn der 70er Jahre des letzten Jahrhunderts zu verzeichnen: die Geburt der Computernetze und die Entwicklung leistungsfä-

higer und preisgünstiger Betriebssysteme, insbesondere Unix, das die IP-Technologie von Anfang an beinhaltete. Deshalb kann auch der 1.1.1970 als das Geburtsdatum des Internet-Zeitalters bezeichnet werden. Die Uhren der Betriebssysteme führen dieses Datum als Beginn der 'Neuzeit' und ihrer Zeitrechnung.

1970 - 1980: Lösung des Schlüsseltausch-Problems

Mit der Nutzung der Internet-Technologie [vgl. Abschnitt 1.1] ergab sich sehr früh die Notwendigkeit, die Kommunikation vertraulich ablaufen zu lassen. Die kryptographischen Grundlagen der symmetrischen Verschlüsselung waren hinreichend bekannt, verlangten aber einen effizienten, Computer-gestützten Algorithmus – und stellten unmittelbar die Frage des Schlüsseltausches zwischen den Teilnehmern. Letzteres wurde – nachdem die zentrale Frage durch *Ralph Merkle* [Mer78] einmal gestellt war – durch *Diffie* und *Hellman* [DH76] und später durch den RSA-Algorithmus von *Rivest*, *Shamir* und *Adleman* – durch die bahnbrechende Entdeckung der *Public-Key-Kryptographie* gelöst. Mit der Verfügbarkeit des *Data Encryption Standards* DES als Blockchiffre mit nachprüfbarer Güte waren bis Ende der 70er Jahre die wesentlichen Grundlagen geschaffen worden.

1980 - 1990: Ergänzung der Grundlagen

Kryptographie entstammt aus dem Mathematikgebiet der Zahlentheorie. Die Größenordnung der natürlichen Zahlen, von denen hier Gebrauch gemacht wird, ist schwindelerregend: Quadrillionen ($10^{24} \sim 2^{80}$) sind hier als 'kleine' Zahlen zu verstehen. In der 80er Jahren gab es erste Vorstöße, den Zahlenraum der natürliche Zahlen zu verlassen und sich algebraischen Strukturen wie *Galois-Feldern* und damit den Lösungen polynomialer Gleichungen und hier speziell den *elliptischen Kurven* zuzuwenden [Mil85; KKM85]. Zudem gab es erste Ansätze, quantenmechanische Effekte für die Datenübertragung zu nutzen [BB84; Sho95].

1990 - 1995: Hashes und Public Key Infrastructure

Anfang der 90er Jahre lag der Fokus aber zunächst auf der praktischen Umsetzung kryptographischer Verfahren: Nachdem zusätzlich die *Stromchiffre* RC4 entwickelt wurde, war als nächster Durchbruch die Entwicklung von Hashfunktionen zu verzeichnen. Zwar waren Hashes auf Grundlage von DES durch einen Trick bereits im Einsatz [Abb. 2.3-2], die explizite Verfügbarkeit von Hashfunktionen (MD5 und SHA) gestattete es jedoch, beliebig große Datenmengen in kurzer Zeit mit einer Hashsumme zu versehen. Das war der Türöffner für digitale Signaturen: Das Konzept der *X.509 Zertifikate* und der *Public Key Infrastructure* war geboren.

1995 - 2000: Rapide Implementierungen

Damit waren im Grunde alle Bausteine für die moderne Kryptographie vorhanden, und es entwickelten sich rapide die Lösungen *Pretty Good Privacy* PGP für den E-Mail- und Datenaustausch, der führende Webbrowser-Hersteller Netscape brachte das Protokoll *Secure Socket Layer* SSL hervor, die *Secure Shell* SSH für interaktive Anwendungen fand umfangreichen Einsatz und ersetzte die unverschlüsselte Datenübertragung per TELNET und FTP. Im Zuge der Entwicklung der IPv6-Protokolls wurde auch die Erweiterung IPSec vorgenommen, die auf dem *Internet Key Exchange* IKE aufsetzt.

2000 - 2010: Konsolidierung und umfangreiche Verbreitung

Ab dem Jahr 2000 kann von einer Konsolidierungsphase gesprochen werden. Die diversen Implementierungen, die teilweise unter US-Exportrestriktionen litten (und diese wiederum umschifften), wurden in Internet-RFCs öffentlich gemacht und somit offiziell sanktioniert. Mit dem *Advanced Encryption Standard* AES betrat ein stärkerer und flexibler Nachfolger für DES für die Blockverschlüsselung den Ring,

und es entwickelte sich die *Elliptic Curve Cryptography* ECC als Ergänzung zum Diffie-Hellman-Schlüsseltausch mittels diskreter Logarithmen. Im Rahmen des europäischen eSTREAM-Projekts [Bab+08] wurden zudem neue Stromchiffren entwickelt, die die bisherigen ablösen sollen. Auch das auf der INDOCRYPT 2004 [MV04] vorgestellte Verfahren des *Galois Counter Mode* sollte sich als wegweisend herausstellen und beendet die lange geführte Debatte 'MAC-then-crypt' durch einen komplett neuen, einfacheren und sichereren Ansatz. Ergänzend hierzu wurde von *Hugo Krywczyk* aus den IBM Watson Labs ein neuer Ansatz vorgestellt, wie kryptographisch *sichere* Passwörter mit hoher Entropie mittels einer *HMAC-based Key Derivation Function* [Kra10] erzeugt werden können.

2010 - 2020: Ende des Dornröschenschlafs und Post-Quantum-Kryptographie

Mit der Ruhe war es dann etwa ab dem Jahr 2010 vorbei: Im TLS-Protokoll, das für den verschlüsselten Internet-Datenverkehr das Arbeitspferd ist, wurden viele Implementierungslücken aufgedeckt, und mit dem *Heartbleed*-Fehler bei TLS zeigte sich, wie verletzlich die Internet-Kommunikation geworden ist. Zudem wurde offenkundig, dass die Public Key Infrastructure durch nachlässigen Umgang mit der Technik und Root-Zertifikaten sowie durch Geheimdienste kompromittiert war und damit eher ein Problem als eine universelle Lösung darstellt. Neue Ansätze, die auch die vorwiegend theoretische Diskussion um die *Post Quantum Cryptography* PQC[11] mit einschließen, werden z.Zt. entwickelt. Hier sei auf den *New Hope*-Algorithmus verwiesen, den Google in seinen Chrome-Browser einbaut. Das stark überarbeitete TLS-Protokoll fand im August 2018 in Version 1.3 endgültig Niederschlag in RFC 8446.

2.1.5 Schichtenspezifische IT-Security-Protokolle

PKI

Mit der Erfindung der asymmetrischen Verschlüsselung Mitte der 70er Jahre und der avisierten Kommerzialisierung des Internet Mitte der 90er Jahre ergab sich die Möglichkeit und zugleich Notwendigkeit, den Internet-Datenverkehr zu verschlüsseln. In den Jahren zuvor war das Konzept der *Public Key Infrastructure* (PKI) entwickelt worden, und mit den nun verfügbaren Rechnerressourcen fand dies mittels der *Secure Socket Layer* (*SSLeay*) zunächst Einzug in den damals dominierenden Netscape Webbrowser sowie dem Apache Webserver.

Abb. 2.1-5 illustriert die heute gängigen Verfahren zur Absicherung der (Internet-)-Kommunikation (mittels Verschlüsselung) auf den unterschiedlichen Kommunikationsschichten:

Dark Fiber

- Bei der physikalischen Übermittlung der Signale kann eine *Dark Fiber* genutzt werden. Eine Dark Fiber muss nicht im eigentlichen Sinne ein Glasfaserkabel sein; jedoch beinhaltet der Terminus, dass immer eine *eigene physikalische Übertragungsstrecke* vorhanden ist, die nicht mit anderen Teilnehmern geteilt wird, sondern in der die Bits sozusagen 'privat' transportiert[12] werden.

  Die Übertragung per 'Dark Fiber' bedeutet nicht unbedingt, dass die Signale zusätzlich verschlüsselt sind. Wie Abb. 2.1-5 zeigt, kann die Anbindung per 'Dark Fiber' nur Punkt-zu-Punkt, d.h. zwischen den Netzknoten erfolgen. Zudem ist die

[11] siehe: https://pqcrypto.org, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

[12] *Google* verwendet 'Dark Fiber' bei den Providern, um seine Rechenzentren zu koppeln.

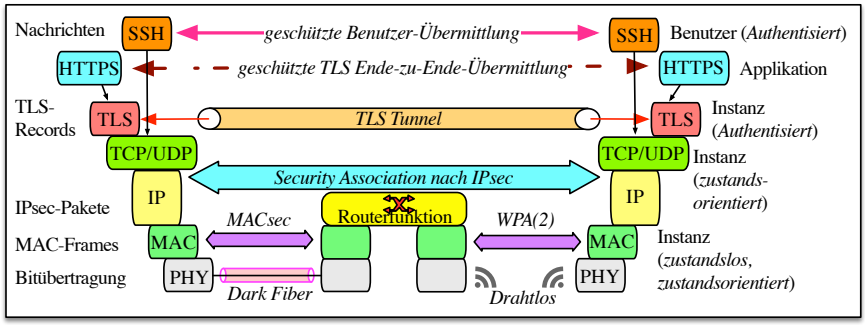


Abb. 2.1-5: Gesicherte IP-Kommunikation auf den verschiedenen Schichten
PHY: Physikalische Schicht, MAC: Media Access Control Schicht, IP: Internet Protocol, IPsec: IP Security, TCP: Transmission Control Protocol, TLS: Transport Layer Security, HTTPS: Hypertext Transport Protocol (Secure), SSH: Secure Shell

Übertragung *zustandslos*, da weder das Verfahren ausgehandelt werden muss noch die darauf aufbauenden Anwendungen hierüber informiert sind.

- Link-Level-Verschlüsselung findet üblicherweise im WLAN statt, wobei hierfür das weit verbreitete Protokoll WPA2 [Abschnitt 13.3] eingesetzt wird. Die Netzwerkschlüssel werden hierbei entweder statisch aufgrund eines *Pre-Shared Keys* PSK oder dynamisch, d.h. ephemeral mittels des *Extensible Authentication Protocols* EAP zur Verfügung gestellt [Abschnitt 15.1]. Dieses auf IEEE 801.1X basierende Verfahren wurde auf das Ethernet-LAN übertragen und wird als MACsec geführt [Abschnitt 14.5]. Interessant ist MACsec für Internetprovider, die eine Kopplung ihrer Netze (Cross-Connect) vornehmen und dabei die Daten-Frames quasi über fremdes Terrain übertragen.

  Link-Level-Verschlüsselung

- Auf der Netzwerkschicht kann als Sicherheitsprotokoll *IP Security* (IPsec) [Abschnitt 6.4] genutzt werden. Alternativ kann das von *Dan Bernstein* erfundene *CurveCP-Verfahren* [`https://curvecp.org/`] als quasi *Transportverschlüsselung* Verwendung finden. Im Gegensatz zu IPsec wird hier keine PKI vorausgesetzt.

  IPsec

  Bei beiden Verfahren kann die Verschlüsselung entweder zum nächsten IP-Peer-Knoten oder zwischen Quelle und Ziel aufgesetzt werden. Auch hier gilt, dass die Verschlüsselung für die Anwendung nicht 'sichtbar', also transparent ist, wenn auch eine Verhandlung zum Aufbau der notwendigen Sicherheitsverfahren gefordert ist; bei IPsec als *Security Association* bezeichnet.

- Das auf TCP aufbauende Verfahren der *Transport Layer Security* (TLS) stellt den Nachfolger des *Secure Socket Layer* (SSL)-Protokolls dar und bildet das heutige Rückgrat des *Internet E-Commerce* [Abschnitt 7.2]. TLS ist immer an eine Anwendung (*Applikation*) gebunden und muss von dieser explizit angefordert werden.

  TLS

  TLS setzt im Grunde eine funktionierende PKI voraus, wobei einige Verschlüsselungsverfahren auch ohne diese auskommen können. TLS ist ein *zustandsorientiertes* Sicherheitssystem, auch wenn es zunächst als transparente Anwendungs-Supportschicht gedacht war. Gebräuchlich sind neben HTTPS vor allem die E-Mail-Protokolle SMTPS, POP3S und IMAPS.

SSH, PGP

- Einige verbreitete Protokolle der Anwendungsschicht besitzen intrinsische Mechanismen, um die zu übertragenden *Nachrichteninhalte* automatisch zu verschlüsseln. Zunächst sei hier das Protokoll *Secure Shell* SSH mit seinen 'Ablegern' *Secure Copy* SCP, *Secure FTP* SFTP und *rsync* genannt. Auch das E-Mail-Verschlüsselungsprotokoll *Pretty Good Privacy* PGP gehört in diese Kategorie.

Neben der Frage der *Verschlüsselung* kommt auch immer die der *Integrität*, d.h. Unverfälschtheit der übertragenen Daten hinzu sowie inwiefern die *Authentizität* der Kommunikationspartner – im Sinne von 'der Richtige' – gewährleistet ist. Vertraulichkeit durch Verschlüsselung und die Integrität der übertragenen Informationen können durch technische Maßnahmen garantiert werden, Autorisierung des *Kommunikationspartners* allerdings nur im Rahmen eines Vertrauensmodells.

2.2 Prinzipien und Primitive der Kryptographie

Das klassische C-I-A-Dreieck der IT-Security verlangt in Konsequenz, dass die *Availability*, d.h. die Verfügbarkeit von Daten und IT-Systemen, die vorrangige Stelle einnimmt: Ohne Verfügbarkeit spielt die Vertraulichkeit und Integrität von Daten keine Rolle.

Redundanzen

Die Verfügbarkeit von IT-Systemen hängt mit den Eigenschaften der Hard- und Software zusammen, deren Zusammenspiel von einer Zwischenkomponente, der Firmware (manchmal auch als Treiber bezeichnet), geregelt wird. In Bezug auf die Hardware kann die Verfügbarkeit durch Redundanzen, d.h. doppelter Auslegung im Rahmen eines *Cold-* oder *Hot-Standby* sowie *Loadsharing* verbessert werden. Die Robustheit von IT-Systemen ist somit nicht nur eine intrinsische Eigenschaft, sondern auch eine Frage der Architektur.

Software-Architektur

Auch bei Software ist eine an die Aufgabenstellung angepasste Architektur maßgeblich, was in Rahmen des *Software Engineerings* SWE behandelt wird. Die SW-Architektur stellt das Bindeglied zwischen den fachlichen und den nicht-fachlichen Anforderungen[13] dar. Zu den IT-Security-relevanten NFRs zählen:

- **Locality Principle**: Daten sollen dort verarbeitet werden, wo sie entstehen.
- **Least Privileges**: Prozesse, die Daten verarbeiten, dürfen nur mit geringsten Systemrechten ausgestattet sein und den 'Security-Kontext' des Anwenders nicht negativ beeinflussen.
- **Segregation of Duties**: Ein Prozess sollte nur wenige Verarbeitungskompetenzen besitzen, d.h. sollte 'single-use' sein. Weitere Verarbeitungsschritte werden von dedizierten Folgeprozessen vorgenommen[14].

Diese Grundsätze der Datenverkapselung und -verarbeitung müssen in der Praxis durch eine fachgerechte Implementierung und deren Tests ergänzt werden. Die Tests beinhalten Modul-, Integrations- und Systemprüfungen. Zusätzlich können auch Performance- bzw. Regressionstests vorgenommen werden [Bla09].

[13] engl.: Functional and None-functional Requirements FR/NFR
[14] siehe: https://cr.yp.to/qmail/qmailsec-20071101.pdf

IT-Systeme, die diese Qualifikation erfüllen, sind auch zur Verarbeitung vertraulicher Daten geeignet, sofern von den folgenden vier Primitive der Kryptographie Gebrauch gemacht wird, die zudem die Integrität und Herkunft der Daten verifizierbar machen.

2.2.1 Verschlüsselungs-Primitiv $\mathbb{C}$

Die Sicherung der Vertraulichkeit (engl. *non-disclosure*) von Daten und Informationen ist eine Notwendigkeit, die im zivilen Leben die Privatsphäre sichert, im militärischen aber dazu dient, den 'Feind' über die eigenen Mitteln und Vorgehensweisen im Unklaren zu lassen.

Secret-Key-Kryptographie

Bei der klassischen Verschlüsselung von Daten wird ein Schlüssel genutzt, der den Anwendern bekannt sein muss, aber ansonsten von keinem Dritten; der Schlüssel ist also geheim: *Secret Key Cryptography*. Bezeichnen wir die Klartextdaten mit x und den Schlüssel mit k, ergeben sich zwei Operationen:

- *Verschlüsselungsoperation*: $x' = \mathbb{C}(x, k)$
- *Entschlüsselungsoperation*: $x = \mathbb{C}^{-1}(x', k)$

Die verschlüsselten Daten wurden als x' eingeführt und die Schlüsseloperation bzw. Chiffrefunktion mit $\mathbb{C}$. Der geheime Schlüssel k dient sowohl zum Ver- als auch zum Entschlüsseln, was auch die Wortwahl 'symmetrische Verschlüsselung' begründet. Bei aktuell implementierten Verschlüsselungsfunktionen wie AES gilt zudem: $\mathbb{C} = \mathbb{C}^{-1}$, d.h. Ver- und Entschlüsselung findet über den gleichen Algorithmus statt, wodurch die Implementierung deutlich vereinfacht wird.

Zustandslose und zustandsbehaftete Schlüssel

Somit können Operationen wie Festplattenverschlüsselung oder das Anlegen und Auslesen des Passwort-Safes trivial gelöst werden. Hierbei bleibt der geheime Schlüssel über einen längeren Zeitraum unverändert; er ist *zustandslos*. Sind die Daten aber mit einem Partner, z.B. über das Netzwerk auszutauschen, muss dem Gegenüber der Schlüssel (und natürlich auch der Algorithmus) bekannt sein, damit die empfangenen, verschlüsselten Daten wieder entschlüsselt und somit im Klartext gelesen werden können. Hierbei empfiehlt es sich nicht, immer den gleichen Schlüssel zu verwenden, sondern es sollte bei jeder Transaktion der Schlüssel gewechselt werden. Die Transaktions- bzw. Sitzungsschlüssel sind daher nur temporär gültig bzw. ephemeral und somit *zustandsbehaftet*.

Bevor wir in Abschnitt 2.4 die technische Implementierungen von $\mathbb{C}$ besprechen, müssen wir zunächst klären, wie unser Gegenüber zu seinem Sitzungsschlüssel kommt.

2.2.2 Schlüsseltausch-Primitiv κ

Das Schlüsseltauschproblem ist eine asymmetrische kryptographische Operation, daher auch häufig die Bezeichnung 'asymmetrische Verschlüsselung'. Tatsächlich werden hier sogenannte *Trapdoor*-Funktionen eingesetzt, deren Zweck darin besteht, die Berechnung eines Ergebnisses für einen potentiellen Angreifer schwierig zu machen, für den Nutzer hingegen einfach.

In der klassischen 'asymmetrischen' Kryptographie, die die Mathematik über Ganzzahlen und endliche Felder (*Finite Fields*) benutzt, kommen historisch zwei unterschiedliche Lösungsansätze in den Einsatz:

KEM
- Die *Key Encryption Method*, bei der der geheime Schlüssel 'verschlüsselt' übertragen wird. Dies wir durch das *RSA-Verfahren* ermöglicht.

KEX
- Die *Key Exchange Method* nach dem *Diffie-Hellmann-Verfahren*. Hierbei verständigen sich beide Partner auf einen gemeinsamen geheimen Schlüssel.

Das RSA-Verfahren lässt sich anschaulich sehr einfach darstellen, was häufig mit der Einführung zweier Akteure – *Alice* und *Bob* – und ggf. *Charlie* und *Eve* einhergeht:

Schlüsseltausch von Alice und Bob

> Wir stellen uns vor, dass *Alice* und *Bob* mit dem Verschlüsselungs-Primitiv ihre vertraulichen Daten gegen Dritte schützen möchten. *Bob* besitzt eine hinreichend sichere Schachtel mit einem Schnappschloss, das nur er mit seinem privaten Schlüssel wieder öffnen kann. Die Schachtel wird nun von *Alice* angefragt und *Bob* händigt *Alice* diese in geöffnetem Zustand aus. Die Schachtel an sich ist wertlos und enthält ... nichts weiter, muss aber *Bob* zugewiesen sein. Nun generiert Alice *einen* Sitzungsschlüssel, legt ihn in die Schachtel und lässt diese zuschnappen. *Alice* bittet nun *Charlie*, die Schachtel mit dem Schlüssel an *Bob* zu übergeben. Da die Schachtel verschlossen ist und nur *Bob* sie öffnen kann, bleibt der Sitzungsschlüssel für *Charlie* im Verborgenen. *Bob* öffnet letztlich mit seinem privaten Schlüssel die Schachtel und entnimmt den geheimen Sitzungsschlüssel.

Public-Key-Kryptographie

Damit der Schlüsseltausch erfolgen kann, wird ein öffentlicher und privater Schlüssel von demjenigen benötigt, mit dem eine verschlüsselte Konversation vorgenommen werden soll: der Empfänger bzw. Receiver r. Seinen *public key* wollen wir als p_r bezeichnen und seinen *private key* mit $\bar{p}_r$.

Die Schlüsseltauschoperation für den gemeinsamen Schlüssel s läuft nun in zwei Schritten ab:

- Initiierung des Schlüsseltausches (vom Sender aus): $s' = \kappa(s, p_r)$
- Vollendung des Schlüsseltausches durch den Receiver: $s = \bar{\kappa}(s', \bar{p}_r)$

Wir sehen, der *Sender* muss über den *public key* p_r vom *Receiver* verfügen, und natürlich nutzen auch beide Partner den gleichen Algorithmus für den Schlüsseltausch, damit der 'verschlüsselte' Sitzungsschlüssel aus e korrekt entnommen werden kann. Die Schlüsseltauschfunktionen (*key encryption*) κ und $\bar{\kappa}$ sind hierbei unterschiedlich – und auch unterschiedlich schnell! Wichtig ist auch, dass es eine ein-eindeutige Zuordnung von *private key* und *public key* gibt: $p_r \Leftrightarrow \bar{p}_r$.

Der Schlüsseltausch erfolgt hierbei *anonym*, was eine fatale Konsequenz aufweisen kann:

Man-in-the-Middle-Angriff

> Wie wir sehen, hängt der Schlüsselaustausch von der Qualität der Schachtel und des Schlosses ab. *Eve* muss sich anstrengen, diese unbemerkt zu öffnen, um den Sitzungsschlüssel zu entnehmen und um damit die von ihm abgefangenen vertraulichen Nachrichten zu entschlüsseln. Wird *Eve* auch zur Übergabe der leeren Schachtel bemüht, kann sie aber cleverer vorgehen: Sie nimmt die Schachtel von *Bob* entgegen, tauscht sie durch seine eigene Schachtel aus und reicht diese an *Alice* weiter. Auf dem Rückweg öffnet sie nun (ihre) geschlossene Schachtel, entnimmt den Sitzungsschlüssel und

> legt einen neuen (ihren) eigenen in *Bobs* Schachtel, um diese an *Bob* zu übergeben. Somit kann *Eve* nun – als Beteiligter im Datenaustausch – komplett unbemerkt und ohne Aufwand die verschlüsselte Konversation zwischen *Bob* und *Alice* verfolgen: ein *Man-in-the-Middle*-Angriff (MitM) wurde durchgeführt!

Anonymer Schlüsseltausch

Ein anonymer Schlüsseltausch ist nur für Szenarien ratsam, wo ergänzende Nachrichtenauthentisierung vorgenommen werden kann. In allen anderen Fällen ist die Authentizität zumindest des Empfängers (Receivers) sicherzustellen, was glücklicherweise auch über die 'asymmetrische' Kryptographie möglich ist; nun aber mit umgekehrten Rollen für den *private* und *public key*. Zuvor benötigen wir aber noch eine weitere wichtige Kryptofunktion, das *Hash-Primitiv*.

2.2.3 Hash-Primitiv h

Ein zentrales Element der IT-Security ist die Möglichkeit, eine Nachricht so zu kennzeichnen, dass diese vom Empfänger als *nicht-verändert* verifiziert werden kann. Hierbei gehen wir davon aus, dass sowohl die Speicherung, als auch die Übertragung von Nachrichten immer mit Fehlern verknüpft ist, weil externe Einflüsse die Nutzdaten überlagern und modifizieren können.

Hashsumme ≠ Checksumme

Technische Lösungen hierfür bieten Paritäts-Bits und Checksummen, also Informationen, die der Nachricht hinzugefügt werden. Dies kann entweder blockweise oder aber nachrichtenweise, d.h. mit variabler Datenlänge, durchgeführt werden. Interessanterweise waren Hashfunktionen schon früh bekannt und wurden beispielsweise bei der Sicherung von Unix-Passwörtern genutzt; es brauchte aber bis Anfang der 90er Jahre ehe Ron Rivest [Abb. 2.1-4] mit MD4 (*Message Digest 4*) die mathematischen Grundlagen für die nicht-injektiven Hashfunktionen legte.

Merkmale von Hashfunktionen

Hashfunktionen besitzen folgende Eigenschaften [Abb. 2.2-1]:

- Hashfunktionen sind *Einwegfunktionen*, bei denen kein mathematisches Verfahren bekannt ist, vom Hashwert h auf den Ursprungswert x zurürck zurechnen (*nicht-reversible*).
- Eine Hashfunktion liefert für eine beliebige Nachricht x einen Hashwert h mit *konstanter Wertelänge* L: $h = \hbar(x)$.
- Es gibt *keine mathematische Umkehrfunktion* für $\hbar$: $x \neq \hbar^{-1}(h) \Rightarrow \not\hbar^{-1}$.
- Hashfunktionen nutzen den *Avalanche-Effekt*; d.h. wenn sich der Input-Wert nur um ein Bit ändert, ist der Hashwert ein komplett anderer und lässt sich nicht aus beiden Teilen ableiten: $x + \delta \Rightarrow \hbar(x + \delta) \neq \hbar(x) + \hbar(\delta)$.
- Hashfunktionen müssen *kollisionsresistent* sein, d.h. es darf nicht möglich sein, einen Wert z mathematisch zu bestimmen, für den gilt: $\hbar(x) = \hbar(z)$. Dies hängt natürlich von der Größe der Bildmenge ab; wobei die Urbildmenge als prinzipiell unbeschränkt gilt.

Der Wert h einer Hashfunktion weist einige interessante Eigenschaften auf, die natürlich von der konkreten Hashfunktion abhängig sind:

- Der Hashwert h ist *zustandslos*; d.h. für eine gegebene Hashfunktion $\hbar$ und bei gleichem Eingabewert x ist er immer identisch.

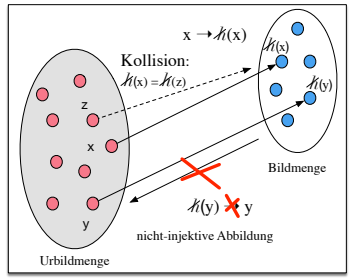


Abb. 2.2-1: Zur Definition von Hashfunktionen;
Die Urbildmenge ist unbeschränkt, während die Anzahl der Elemente in der Bildmenge durch die Forderungen einer konstanten Hashlänge beschränkt ist; bei SHA-256 besitzt die Bildmenge also $2^{256} \approx 10^{77}$ Elemente.

- Der Hashwert h besitzt eine *maximale Entropie* und verhält sich wie ein Zufallswert, und zwar (weitgehend) unabhängig von x. Diese Eigenschaft wird auch häufig als *Pseudo-Zufallswert* beschrieben.

Rainbow-Tabellen

In der Praxis spielt die schnelle Berechenbarkeit von Hashwerten eine große Rolle. Übliche Hashfunktionen wie MD5 und SHA arbeiten schnell, auch bei bei großen Datenmengen für den Eingangswert x. Dies ist dann wichtig, wenn Hashes zur Sicherstellung der Datenintegrität genutzt werden. Die schnelle Berechenbarkeit führt aber zu dem Phänomen, dass Hashwerte für lexikalische Ausdrücke generiert und 'online' gestellt werden können: *Rainbow-Tabellen*.

Werden Hashes zur Verschleierung von Passwörtern genutzt, sind Hashfunktionen sinnvoll, die viele Systemressourcen (z.B. Hauptspeicher) 'verbrauchen'. Hierzu zählen vor allem *bcrypt* und *scrypt*, die gerne auch als Grundlage für *Password-Based Key Derivation Functions* genutzt werden.

Hashwert = Index

Der große Nutzen von Hashwerten besteht aber darin, dass man Daten hiermit *indizieren* kann, wobei der Index eine konstante Länge aufweist. Der Index ist somit eindeutiger Repräsentant der Daten und mit diesem über die Hashfunktion h verknüpft. Dies wird speziell für *Digitale Signaturen* ausgenutzt, denen wir uns nun zuwenden wollen.

Eine weitere populäre Nutzung von Hashfunktionen liegt in Form der *Blockchain* und hier speziell der Erzeugung von Krypotwährungen wie Bitcoin vor, wo deren Eigenschaft das mathematische Fundament des Mining darstellt:

Hashwerte bei Bitcoins

> Die Nicht-Umkehrbarkeit von Hashfunktionen spielt aktuell bei der Berechnung von Bitcoins ₿ eine zentrale Rolle: Bitcoins werden 'gemined', indem gefordert wird, dass der SHA-256-Hashwert immer weiter gegen Null tendiert: '`000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f`'. Der hier hexadezimal dargestellte Wert ist im Genesis-Block der Bitcoin-Blockchain vorhanden. Wenn alle 256 Bit des Hashwerts auf Null gesetzt sind bzw. wenn der Inputwert y bekannt ist, für den diese Bedingung gilt, also SHA-256(y) = '`0000 ... 0000`', sind alle Bitcoins erschöpft. Die 'einfache' Berechenbarkeit von SHA-256 macht Bitcoin-Mining attraktiv, speziell auf geeigneter Hardware. Allerdings mit

dem negativen Seiteneffekt, dass enorm viel elektrische Energie für sinnlos generierte Hashwerte benötigt wird.

2.2.4 Signatur-Primitiv σ

Beim Signatur-Primitiv σ drehen wir den Verwendungszweck des *public key* p und des *private key* $\bar{p}$ und gehen davon aus, dass sie dem Unterzeichner (*Originator*) zugewiesen sind – also p_o und $\bar{p}_o$ –, der ein Dokument x signieren möchte:

- *Signaturoperation* des Unterzeichners: $sig = \sigma(x, \bar{p}_o)$
- *Verifikationsoperation* beim Empfänger: $x = \bar{\sigma}(sig, p_o)$

Message Digest

Der *public key* des Unterzeichners p_o ist öffentlich bekannt bzw. wird zusammen mit dem Dokument x und der Signatur s dem Empfänger zur Verfügung gestellt. Aus praktischen Gründen ersetzen wir allerdings das Dokument x durch dessen Hashwert: $h = h(x)$, was als *Message Digest* bezeichnet wird. Die Signatur ist wiederum *zustandslos*.

Üblicherweise wird daher ein Zeitstempel zur Signatur hinzugefügt [Abb. 2.2-2]: $sig(t) = \sigma(h(x) + time, \bar{p}_o)$. Hierdurch wird die Signatur auch transaktions-sicher.

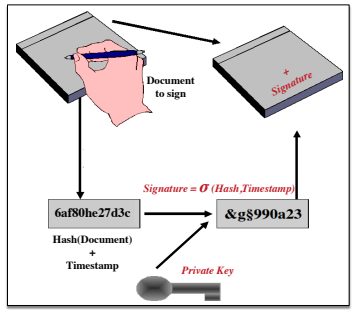


Abb. 2.2-2: Erzeugung einer digitalen Signatur

Ziele digitaler Signaturen

Laut *Bruce Schneier* [Sch96] lassen sich mit digitalen Signaturen folgende Ziele realisieren:

1. *Verifiable*: Der Empfänger kann einfach überprüfen, dass die Nachricht tatsächlich vom Unterzeichner stammt.
2. *Un-Forgeable*: Nur der Unterzeichner kann die Signatur dem Dokument beifügen; diese kann somit nicht gefälscht werden.
3. *None-Reusable*: Die digitale Signatur ist für dieses Dokument einmalig und kann nicht für andere benutzt werden.
4. *Un-Alterable*: Eine Änderung des Dokuments nach Berechnung der Signatur zerstört die Bindung zwischen beiden: Fälschungssicherheit.
5. *None-Deniable*: Der Unterzeichner kann nicht bestreiten, dass er das Dokument signiert hat.

2.2.5 Zusammenspiel der Krypto-Primitive

Die gesamtem moderne IT-Security macht sich die Krypto-Primitive

- Verschlüsselung $\mathbb{C}$,
- Schlüsseltausch κ,
- Signatur σ sowie
- Hashes h

zunutze und baut mit den unterschiedlichen Implementierungen entsprechende Krypto-Lösungen auf, wie Abb. 2.2-3 in Anlehnung an XKCD[15] zeigt.

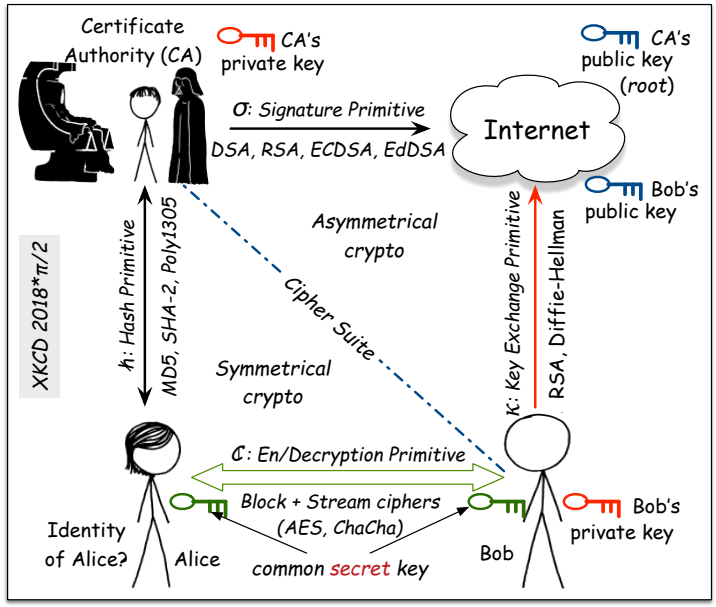


Abb. 2.2-3: Kryptographische Primitive und ihre Implementierung
AES: Advanced Encryption Standard, MD5: Message Digest 5,DSA: Digital Signature Algorithm, DSS: Digital Signature Standard, RSA: Rivest/Shamir/Adleman Algorithmus, ECD: Elliptic Curve DSA, EdDSA: Edwards Kurven DSA, SHA: Secure Hash

Verschlüsselungs-Primitiv

Die Verschlüsselungs-Primitive $\mathbb{C}$ können wir grob in Block- und Stromschlüssel einordnen, worauf wir später noch im Detail eingehen werden. Allerdings werden bis dato bei starker Verschlüsselung nahezu ausschließlich Blockchiffren – wie AES – in einem verschränkten Betriebsmodus eingesetzt. Eine Mischform von Block- und Stromchiffren gilt bei richtiger Wahl der Komponenten als besonders sicher.

Es besteht die Meinung, dass vor allem die Länge des Schlüssels die Qualität der Verschlüsselung bestimmt. Bei Blockchiffren sind die einzelnen Nachrichtenblöcke immer gleich lang wie der Schlüssel, was dazu führt, dass wir in einer Verschlüsselungsoperation immer genau n Bits der Blocklänge als auswertbare Nutzinformation

[15] *A webcomic of romance, sarcasm, math, and language* von *Randall Munroe*, siehe: https://xkcd.com/

mitgeben, d.h. die Verschlüsselung wird 'stärker', aber um den Preis von mehr Informationen, die wir automatisch liefern.

Schlüsseltausch-Primitiv

Die Schlüsseltausch-Operation σ basiert wie auch die Signatur-Operation σ auf dem bekannten RSA- bzw. Diffie-Hellman-Algorithmus. Beide Verfahren arbeiten deterministisch, d.h. mit der Vollendung des Schlüsseltauschs $\bar{\kappa}$ ist klar, dass beide Partner über die gleichen Parameter verfügen. Zukünftige, z.B. Gitter-basierte Verfahren, operieren dagegen *probabilistisch* bzw. *optimistisch*:

> In den meisten Fällen stimmen die ausgehandelten Parameter überein, der Algorithmus garantiert dies aber nicht. Sollte die anschließende Verschlüsselung nicht klappen, muss der Schlüsseltausch nochmals angestoßen werden.

Signatur-Primitiv

Im praktischen Einsatz des Signatur-Primitivs σ ist man auf eine tragende Infrastruktur angewiesen, wo von einem 'Trust-Anker' aus Vertrauen bezogen werden kann. Für die *Public Key Infrastructure* PKI und den *Certificate Authorities* ist dieses Vertrauen verloren gegangen, und so stehen Alternativen wie das dezentral organisierte *Domain Name System* DNS derzeit im Zentrum einer Neuausrichtung.

Hash-Primitiv

Das Hash-Primitiv h hat sich als 'Schweizer-Messer' der Kryptographie entwickelt, wird es doch sowohl zur Integritätsprüfung lang- und kurzfristiger gültiger Datenblöcke benutzt, wobei hier auf den Datenkontext im vorigen Abschnitt verwiesen werden kann. Der Hashwert liefert somit einen technischen Kontext zur Nutzinformation, der in manchen Fällen – wie unter Einsatz von TLS Abb. 2.3-1 – zwar 'Integrität' sichert, aber auch 'Data Leakage' mit sich bringen kann.

Hashfunktionen und Verschlüsselungsfunktionen zählen zu den 'klassischen/symmetrischen' Krypto-Verfahren. Deren Implementierung ist auch erstaunlich ähnlich und basiert auf den mathematischen Ideen der *Permutation*, *Transposition* und – wichtig – *Substitution*. Die letztere Operation macht das Ergebnis quasi 'unvorhersehbar' und abhängig von den Eingangswerten. Schlüsseltausch- und Signaturoperation fallen hingegen in den Bereich der 'asymmetrischen' Kryptographie [Abb. 2.2-3] und werden heute technisch vor allem mittels *elliptischer Kurven* unter Einsatz des *Diffie-Hellman-Protokolls* vorgenommen.

Keine Authentisierung von Alice

Bei den derzeitigen Krypto-Lösungen wird im Rahmen der Transaktion nur die Identität des Empfängers (also Bob; vgl. Abb. 2.2-3) verlangt; Alice handelt hingegen anonym, und die Berechtigungen, über die Alice verfügt, müssen in der Applikation geklärt werden.

> Will man das ändern, muss Alice ebenfalls über einen *public* und *private key* verfügen, was technisch möglich, praktisch jedoch mit einigem Aufwand verbunden ist. In den meisten Fällen begnügt man sich damit, Alice eine *lokale Identität* zuzuweisen. Globale Identitäten, die z.B. an Zertifikate im Personalausweis geknüpft sind, wurden zwar in den letzten Jahrzehnten entwickelt, haben sich aber immer noch nicht auf breiter Front durchgesetzt.

2.3 Hashfunktionen und ihr Einsatz

Die zentralen Eigenschaften von Hashfunktionen und Hashwerten wurden bereits bei der Diskussion des Hash-Primitivs h vorgestellt. Hierbei wurde eine Hashfunktion als Rechenvorschrift eingeführt, mit der eine 'Eingangs'-Zeichenfolge beliebiger Länge in eine 'Ausgangs'-Zeichenfolge konstanter (im Allgemeinen kürzerer) Länge umgewandelt wird: der *Hashwert* oder die *Hashsumme*. Diese Einweg-Hashfunktion funktioniert immer nur in eine Richtung, d.h. aus der 'Eingangs'-Zeichenfolge lässt sich einfach die 'Ausgangs'-Zeichenfolge berechnen, aber es ist aufgrund des *Lawinen-Effekts* praktisch unmöglich, zu einer 'Ausgangs'-Zeichenfolge passende 'Eingangs'-Zeichenfolgen zu berechnen: Ändert sich bei der 'Eingangs'-Zeichenfolge auch nur ein Bit, besitzt der erzeugte Hashwert einen vollkommen anderen, unvorhersehbaren Wert, was eine 'Hashsumme' fundamental von einer '*Checksumme*' unterscheidet.

Ein weiteres wichtiges Merkmal, was Hashsummen von Checksummen unterscheidet, ist ihre Nicht-Additivität: Wenn wir zu einem bekannten Wert x, für den $h(x)$ gebildet werden soll, einen weiteren Wert k hinzufügen, sagen wir in Form einer Konkatenierung $h = h(k \parallel x)$, benötigt man das Wissen sowohl von x als auch von k, um den Hashwert h zu berechnen.

Salted Hash = Keyed Hash

Das Hinzufügen eines weiteren Wertes in der Hashsumme wird als *Salt* oder auch *key*, der erzeugte Hashwert demzufolge als 'salted' oder 'keyed' hash bezeichnet. Welche Eigenschaften soll nun k als Salt besitzen? Hierfür gibt es folgende Szenarien:

Key $k = \ldots$	**Bedeutung**	**Hashfunktion** $h(\ldots)$	**Bemerkung**	**Zustands-** **los**	**behaftet**
$\varnothing$	kein *key*	$h(x)$	unverändert	$\surd$	
$[0,1]$	protokollspezifische Füllbits	$h([0,1] \parallel x)$	*Obfuskation* trivialer Werte von x	$\surd$	
$\{0,1..,i,...,n\}$	Index i	$h(i \parallel x)$	*Sequenzierung* von x		$\surd$
random	Nonce/Challenge/Salt	$h(\text{random} \parallel x)$	*random* kann öffentlich sein		$\surd$
shared secret	Geheimes Passwort	$h(\text{secret} \parallel x)$	Nachrichten-Authentisierung	$\surd$	$\surd$

Tab. 2.3-1: Verwendung von Hashfunktionen mit Salt;
$\parallel$: Konkatinierungs-Operator, MIC: Message Integrity Check, Nonce: oNly use once

Rainbow Tables und Salt

Salted Hashes schützen – unabhängig von der Hashfunktion – den Hashwert zuverlässig gegen *Rainbow Tables*. Bei einigen Verfahren wie bei *Challenge/Response* wird für die Benutzerauthentisierung sowohl von Füllbits als auch von Challenges Gebrauch gemacht, was die Gefahr des Erratens trivialer Passwörter stark senkt.

2.3.1 Hashfunktionen zur Nachrichtensicherung

An die Hashfunktionen zur Nachrichtensicherung werden zwei wesentliche Anforderungen gestellt:

1. Sie müssen schnell sein, wenig Rechnerressourcen verbrauchen und ggf. parallelisierbar sein, was sowohl einzelne große Datenmengen als auch viele kleine zu hashende Datenpakete (z.B. IP-Pakete) betrifft.

2. Sie müssen kollisionsresistent sein, d.h. für die zu verarbeitende Datenmenge sollten keine bekannten Kollisionen vorliegen.

Einige bekannte Hashfunktionen sind:

- MD5 (*Message Digest 5*) [RFC 1321]: Die erzeugte 'Ausgangs'-Zeichenfolge, der Hashwert, hat eine Länge von 128 Bit (32 hexadezimale Zeichen). Allerdings sind hier Kollisionen bekannt, sodass MD5 nicht für zustandslose und langfristig gültige Hashes benutzt werden sollte.
- RIPEMD (*RIPE Message Digest*) wurde für das RIPE-Projekt (RACE Integrity Primitives Evaluation) der EU entwickelt und 1996 erstmals veröffentlicht. RIPEMD-160 ist eine Hashfunktion, die eine Prüfsequenz mit der Länge von 160 Bit erzeugt. Es existieren auch die 128-, 256- und 320-Bit-Versionen von RIPEMD, die man entsprechend als RIPEMD-128, RIPEMD-256 bzw. RIPEMD-320 bezeichnet.
- Salsa10[16] ist eine von D. J. Bernstein entwickelte und sehr schnelle Hashfunktion, die einen 64 Byte (512 Bit) langen Hashwert liefert.
- SHA(-1) (*Secure Hash Algorithm*) [RFC 3174] und einem Hashwert von 224 Bit, entsprechend 40 hexadezimalen Zeichen in der Ausgabe.
- SHA-2 *Secure Hash Algorithm* mit Längen von 256, 384 und 512 Bit [RFC 4634]. SHA-2 ist die weit verbreitetste Hashfunktion, die speziell auch für digitale Signaturen eingesetzt wird. Dass die Länge des erzeugten Hashwertes nicht mit der Rechengeschwindigkeit zu tun hat, zeigt augenfällig SHA-512, das aufgrund des effizienteren Algorithmus schneller ist als SHA-256.
- GMAC (*Galois Message Authentication Code*) folgt dem im Abb. 2.4-6 dargestellten Verfahren, nur dass auf eine explizite Verschlüsselung verzichtet wird.

2.3.2 Message Authentication Codes

Mittels der Hashfunktion wird eine Prüfsumme `MAC` (*Message Authentication Code*) berechnet, mit der eine zu sendende Nachricht bzw. ein IP-Paket signiert werden kann.

Eine besondere Form der Keyed-Hashfunktion wird als HMAC-Funktion (*Hash based Message Authentication Code*) bezeichnet. Sie kann mit jeder beliebigen Hashfunktion (z.B. MD5 als auch SHA) benutzt werden. Die HMAC-Funktion auf der Basis der Hashfunktion h ist folgendermaßen definiert [RFC 2104]: HMAC

$$Y = \mathrm{HMAC}(k, X) = h(k \otimes opad, h(k \otimes ipad, X))$$

Erläuterung: k: gemeinsamer und geheimer Schlüssel, X: die zu übertragenden Daten, Y: Prüfsumme als Signatur der Daten; $\otimes$: Operation `Bitwise_Exclusive_OR`, *ipad*: Sequence von B Byte '`0x36`', *opad*: Sequenz von B Byte '`0x5c`'.

Zur Berechnung der Hashfunktion h werden zunächst die Daten in Blöcke mit der Länge von B Byte aufgeteilt ($B = 64$ und ggf. durch die Padding-Bytes `0x00` aufgerundet). Anschließend erfolgt die Berechnung des Hashwertes iterativ.

[16] siehe: https://cr.yp.to/salsa10.html

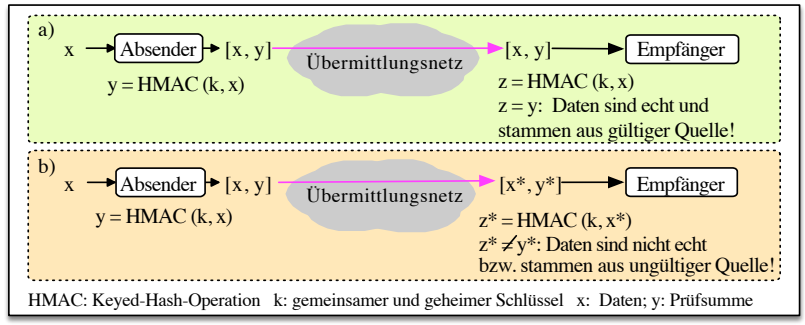


Abb. 2.3-1: Authentisierung der Datenquelle und Überprüfung der Datenintegrität
a) Ergebnis ist positiv, b) Ergebnis ist negativ

Arbeitsweise von HMAC

Abb. 2.3-1 erläutert das Prinzip, wie die HMAC-Funktion genutzt werden kann, um die Echtheit und die Herkunft von Daten zu überprüfen, bei der beide Kommunikationspartner zunächst ein *shared secret* k besitzen. Wie ersichtlich, erstellt der Absender (Quellrechner) zusätzlich zur eigentlichen Nachricht x eine Prüfsequenz y mittels der HMAC-Funktion, in die die Nachricht x und das *shared secret* k einfließt, und fügt diese der Nachricht bei. Der Empfänger entnimmt nun die Nachricht x und berechnet wiederum mittels des *shared secrets* k auf die gleiche Weise den Hashwert z. Sind y und z gleich, [Abb. 2.3-1a] ist sowohl die Herkunft der Nachricht als auch ihre Unverfälschtheit sichergestellt. Ist der ermittelte Wert $z*$ nicht identisch mit dem Wert $y*$ für die Nachricht $x*$, wurde diese entweder bei der Übertragung verändert oder aber der Absender besitzt nicht das gleiche *shared secret* [Abb. 2.3-1b].

HMAC-Funktionen

Üblicherweise stehen folgende HMAC-Funktionen zur Verfügung:

- HMAC-MD5 – 128 Bit Länge
- HMAC-SHA-1 – 160 Bit Länge sowie
- RIPEMD-160 – 160 Bit Länge.

Welche Hashfunktion h eingesetzt wurde, lässt sich aus der Länge des Hashwertes zwanglos ermitteln, sofern die bekannten Hashes eingesetzt werden. Liegt eine Hashlänge von 160 Bit vor, muss sowohl auf SHA-1 als auch auf RIPE getestet werden.

AES-CMAC

Weitere vom NIST[17] spezifizierte Hashfunktionen zur Nachrichtenauthentisierung stellen AES-CMAC-128 bzw. AES-CMAC-256 dar. Beide unterscheiden sich nur in der Länge des Hashwerts. Abb.2.3-2 illustriert die Arbeitsweise von AES-CMAC, und gemeinsam mit Abb. 2.3-3 verdeutlicht dies die Ähnlichkeit zwischen symmetrischer Verschlüsselung und Hashfunktion. Die Nachricht muss hier in n-Blöcke mit jeweils der Schlüssellänge heruntergebrochen werden. Der letzte Block ist um die fehlenden Bits zu ergänzen werden, wobei das Padding mit dem Bit '1' beginnt und dann die übrigen Bit mit '0' deklariert werden.

Poly1305

Bleibt das *shared secret* aus Tab.2.3-1 unverändert, arbeiten diese HMAC-Funktionen zustandslos. Wird aber zusätzlich ein Nonce hinzugefügt, ist der HMAC automatisch zustandsbehaftet wie beim Poly135-AES Verfahren [Ber05], das in RFC 7905 standardisiert ist und zunächst für die Blockchiffre AES entwickelt wurde. Hierbei wird

[17] sieh: https://csrc.nist.gov/publications/detail/sp/800-38b/final

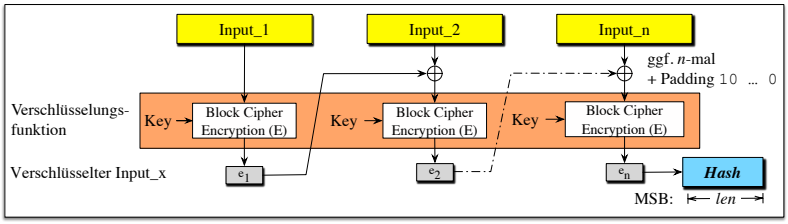


Abb. 2.3-2: Authentisierung einer Nachricht mittels der AES-CMAC-Hashfunktion

neben dem 16 Byte langen AES-Schlüssel k je ein weiterer 16-Byte-Schlüssel r sowie ein *Nonce* n genutzt, um den *Authenticator* a zu bauen:

$$a = \text{Poly1305}_r(m, \text{AES}_k(n))$$

Die Nachricht liegt als m vor, und an diese wird der Authenticator a hinzugefügt. Gegenüber den weiter oben genannten Hashfunktionen, die im wesentlichen auf Substitutionsboxen aufbauen, wird bei Poly1305 eine Polynom-Berechnung modulo (der Primzahl als Restwert) $2^{130} - 5$ durchgeführt. Dies besitzt nicht nur den Vorteil besonders schnell und parallelisierbar, sondern auch *beweisbar* sicher zu sein. Ein potenzieller Angreifer muss zur Berechnung von a neben m auch die Schlüssel k und r sowie das Nonce n kennen bzw. schätzen.

2.3.3 Hashfunktionen für Passwörter

Bei der Verwendung von Hashfunktionen im Zusammenhang mit Passwörtern ergeben sich zwei interessante Szenarien:

1. Bildung des Hashes h_p eines Passworts p zur Ablage des verschleierten Passworts: $p \rightarrow h_p = h(p)$.
2. Nutzung der Eigenschaft von Hashfunktionen, quasi zufällige Bytemuster zu erzeugen, die dann als Passwörter genutzt werden können: *Password-Based Key Derivation Functions* PBKDF.

Will man Passwörter, insbesondere von Dritten sorgfältig aufbewahren, kommt man nicht drumherum, diese verschlüsselt abzulegen. Hierbei steht man vor dem Dilemma, entweder alle oder zumindest einen Satz von Passwörtern mit dem gleichen Schlüssel zu verschlüsseln, oder aber jeden Eintrag einzeln, was einen erhöhten Aufwand bedeutet.

Unix crypt als Hashfunktion

Das Unix-Betriebssystem weist demgegenüber die *crypt*-Funktion auf, die das Passwort zustandsbehaftet gemeinsam mit dem Salt ablegt. Ursprünglich diente hierzu der Trick, die DES-Blockchiffre zu nutzen, indem das Passwort als Schlüssel, die Sequenz $8 * 0$ Byte als Nachricht herangezogen wurde [Abb. 2.3-3]. Dies hat allerdings den Nachteil, dass hierdurch die Passwortlänge auf maximal 8 Zeichen beschränkt ist, was mittels moderner Hashfunktionen natürlich nicht mehr gegeben ist. Allerdings

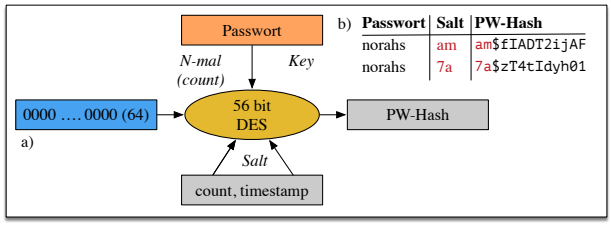


Abb. 2.3-3: Ablauf der ursprünglichen Unix-Funktion crypt
a) Das Passwort wird zur Verschlüsselung der mit 'count' und 'timestamp' als implizitem und explizitem Salt angereicherten 0-Byte-Sequenz genutzt und das Ergebnis der Berechnung mehrmals (64 mal) wiederholt. b) Salt und Passwort-Hash werden in der Daten `/etc/passwd` abgelegt, die nur vom Unix-User *root* lesbar ist. Wird das (gleiche) Passwort neu erzeugt und abgelegt, ist der Hashwert stets verschieden.

gilt der Grundsatz, dass der Passwortvergleich relativ 'teuer' sein sollte, sodass ein potenzieller Angreifer nicht zu schnell zum Ziel kommt.

Standard-Hashes unterlaufen diese Forderung, und so wurden die speziellen Hashfunktionen *bcrypt* und *scrypt* entwickelt, die bei einigen Unix-Derivaten wie OpenBSD[18] zum Einsatz kommen.

PBKDF

Neben diesem quasi passiven Einsatz von Hashfunktionen kann man deren Eigenschaften auch aktiv nutzen, um 'gute' Passwörter zu generieren, was aus der hohen Zufälligkeit und Entropie des Hashwertes folgt. Hierbei gehen wir folgt vor:

kurzes, statisches Passwort $p \rightarrow h(salt, p) \rightarrow$ Passwort zur aktuellen Verschlüsselung

PKCS#5

Wir nutzen also die Hashfunktion, um abgeleitete Passwörter zu produzieren, was in RFC 8018 als *Password-Based Key Derivation Functions* PBKDF spezifiziert ist und unter der Kennzeichnung `PKCS#5` läuft. Ein aktueller Vorschlag besteht in der Kombination der *scrypt* Hashfunktion mit einer auf acht Runden reduzierten Version von *Salsa* (Salsa20/8)[19].

WLAN-Schlüssel

> Für PBKDF gibt es bei der IP-Kommunikation viele Anwendungsfälle. So wird z.B. das WLAN-WPA2-Passwort, das für den Zutritt zum Netzwerk im Falle von *Pre-Shared Keys* PSK eingegeben werden muss, niemals als Verschlüsselungspasswort benutzt, sondern es durchläuft eine PBKDF. Diese ist für den Benutzer im WLAN nicht sichtbar, vergrößert die Passwortlänge auf das gewünschte Maß und erhöht somit die Sicherheit gegenüber möglichem *Brute Force*-Erraten des Passworts deutlich; speziell dann, wenn das Eingangspasswort eine ausreichende Zufälligkeit besitzt.

HMAC-based Key Derivation Function

Eine Weiterentwicklung der PBKDF stellt die *HMAC-based Key Derivation Function* HKDF dar [Kra10], die in RFC 5896 standardisiert ist. Ausgangsüberlegung für diese Funktion ist, die vorliegende Entropie beim Erzeugen von Passwörtern optimal zu nutzen. Tatsächlich kann der Fall eintreten, dass beim Hashen einer Zufallszahl die resultierende Entropie nicht nur zu-, sondern sondern auch abnehmen kann[20]. Entropie kann im Grunde genommen nicht erzeugt, sondern nur genutzt werden.

[18] siehe: `https://www.openbsd.org/papers/bcrypt-paper.pdf`
[19] siehe: `https://tools.ietf.org/html/draft-josefsson-scrypt-kdf-01`
[20] siehe: `https://blog.cr.yp.to/20140205-entropy.html`

Eine HKDF nutzt wie auch eine PBKDF einen (schwachen) Eingabewert und optional ein Salt zur Erzeugung einer Pseudo-Zufallszahl – dem *key* – in drei Schritten:

1. Generierung eines *Pseudo-Random Keys* PRK = HMAC-Hash$(input, salt)$.
2. Iterative Erzeugung des *Output Key Materials* in mehreren Runden: OKM = HKDF-Expand(PRK, info, len); info sind ergänzende Kontextdaten, die optional sind, len ist die Länge des Hashes. Bei der HKDF-Expand-Funktion wird von einem Wert $T(i = 0) =$ "" gestartet und dann in Folge $T(i)$ berechnet als $T(i) =$ HMAC-Hash(PRK, $T(i-1)||\text{info}||0x01$). Der Wert von $T(k)$ hängt also von den früheren Werten $T(k-i)$ ab.
3. Das finale *Output Key Material* OKM wird aus diesem konkatenierten Hash in der gewünschten Länge geschnitten.

Mit einer HKDF lassen sich zwei Anwendungsfälle realisieren:

- Die Ableitung eines kurzen, qualifizierten *Passworts* mit genügend Entropie aus größeren Eingabewerten. Extract
- Die Erzeugung von umfangreichem, qualifiziertem *Key Material* aus einer bereits vorhandenen *pseudo random*-Bytefolge durch Diffusion. Expand

Die letzte Eigenschaft wird in der neuen TLS-Version 1.3 genutzt.

2.4 Symmetrische Verschlüsselung

Verschlüsselung ist über die letzten Jahrtausende immer eine Anforderung der Kriegsführung gewesen. Bereits seit der Antike sind *Chiffren* (Verschlüsselungsverfahren) bekannt. Zu den bekanntesten zählen die Verschiebeschiffren, von denen verbrieft ist, dass diese schon von Julius Caesar (100 bis 44 v.d.Z.) eingesetzt wurden. Historisch von Bedeutung sind beispielsweise

- Verschiebechiffren,
- multiplikative Chiffren sowie
- affine bzw. Tauschchiffren

die als *text-alphabetische* Chiffren bekannt sind und allesamt den Nachteil besitzen, dass mittels Analyse der statistischen Häufigkeiten der Buchstaben leicht festgestellt werden kann, aus welcher Sprache sie entstammen.

Später wurden *CodeBooks*, also Kodierbücher entwickelt, die eine Übersetzung der Texte ermöglichen. Die *Enigma*-Maschine stellte den Versuch dar, die Verschlüsselung zu automatisieren und zugleich einer zentralen Anforderung der modernen Kryptographie zu entsprechen: Kerckhoff'sches Prinzip (1883)

> Bei kryptographisch 'sicheren' Verschlüsselungsmethoden darf die Sicherheit bzw. der Schutz der verschlüsselt übertragenen Nachricht nicht vom *Verschlüsselungsalgorithmus* abhängen, sondern ausschließlich von der Geheimhaltung des *Schlüssels* sowie seiner *Entropie* und der *Schlüssellänge*.

Abb. 2.4-1 gibt eine Überblick über die aktuellen Verfahren der symmetrischen Verschlüsselung, die wir generell einteilen in Verfahren der

- *Blockverschlüsselung* mit zustandslosem Schlüssel und
- *Datenstromverschlüsselung* mit zustandsbehafteten Schlüsseln.

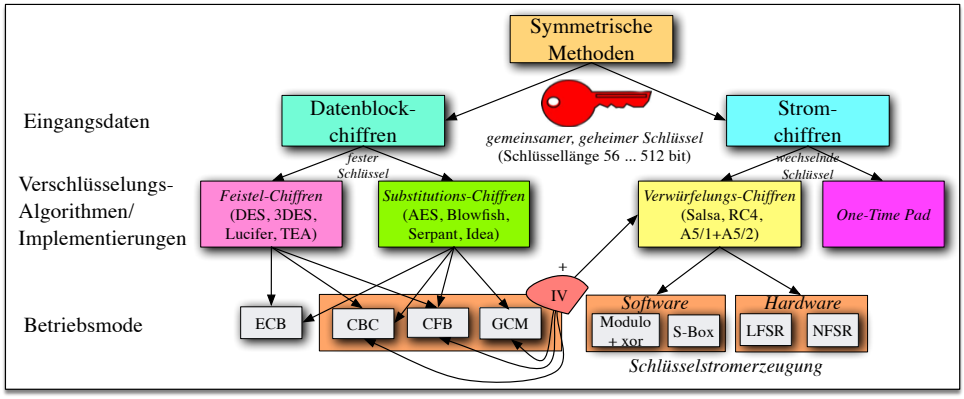


Abb. 2.4-1: Überblick über die symmetrischen Verschlüsselungsverfahren
ECB: Encoding Code Book, CBC: Cipher Block Chaining, CFB: Cipher Feedback Mode, GCM: Galois Counter Mode, LFSR: Linear Feedback Shift Register, NFSR: None-linear Feedback Shift Register, IV: Initialisierungsvektor

Im praktischen Einsatz finden zwei unterschiedliche Schlüssel Verwendung:

- $A \xrightarrow{\text{Schlüssel}\,(A \to B)} B$
- $B \xrightarrow{\text{Schlüssel}\,(B \to A)} A$

Schlüssel = zufällige Bitsequenz

Unter der Annahme, dass der Schlüssel eine weitgehend zufällige Bitsequenz[21] darstellt und daher nicht trivial geraten werden kann, ist die Schlüssellänge Garant für die Verschlüsselung selbst. Will man eine Verschlüsselung brechen, sind statistisch gesehen $2^{\text{Schlüssellänge}/2}$ Operationen notwendig, bis der 'richtige' Schlüssel durch Probieren gefunden wird. Der unter diesen Umständen aus dem Chiffretext abgeleitete Klartext sollte anschließend 'Sinn' machen. Unter den Bedingungen der aktuellen Rechnerperformance sind Schlüssellängen unter 100 Bit angreifbar.

Generell versuchen die Kryptologen, Chiffrecodes zu entwickeln, die mathematisch als 'schwer zu brechen' nachgewiesen werden können. Zentrales Merkmal hierbei ist, wie viele CPU-Instruktionen benötigt werden, den verschlüsselten Text in seine Ursprungsform zurückzuführen, wobei der Verschlüsselungsalgorithmus bekannt ist.

Üblicherweise werden zum Brechen der Verschlüsselung folgende Angriffsszenarien entwickelt:

Ciphertext-only
- Aus einer beschränkten Anzahl von abgefangenen Geheimnachrichten soll der Schlüssel und somit die Ursprungsnachricht berechnet werden.

Known-plaintext
- Der Angreifer ist im Besitz von Geheimnachrichten als auch der ursprünglichen Nachrichten und kann darüber den verwendeten Schlüssel bestimmen.

- Kennt der Angreifer Teile des ursprünglichen und verschlüsselten Textes und ihre Zuordnung und kann Teile des Schlüssels entziffern, um mit diesen Teiltexte gezielt zu verschlüsseln. Chosen-plaintext
- Hier gilt das Umgekehrte wie der *Chosen-plaintext*, da nun der Angreifer durch Kenntnis eines Teilschlüssels geheime Dokumente teilweise entschlüsseln kann. Chosen-ciphertext

Die notwendigen Hilfsmittel hierzu stellt die lineare bzw. differenzielle *Kryptanalyse* zur Verfügung [Mat93; OM93], und hierbei speziell das *Piling-Up Lemma*, das von *Mitsuru Matsui* im Rahmen seiner Analyse der DES3-Blockchiffre entwickelt wurde. Piling-Up Lemma

Alle Verschlüsselungsverfahren sind im Prinzip angreifbar, wie z.B. letztlich *Efail* [Pod+18] gegen die E-Mail-Verschlüsselung gezeigt hat. Lediglich das *One-Time Pad* ist bei richtiger Anwendung 'sicher' gegenüber solchen Angriffen.

2.4.1 Stromchiffren

Stromchiffren werden für Nachrichten/Datenblöcke mit unbekannter Länge eingesetzt, z.B. einem Telefongespräch in einem digitalen Netz. Sie finden daher u.a. bei den Mobilfunknetzen GSM und UMTS Verwendung (mit den Verfahren A5/1 und A5/2).

Im Bereich des Internet fand ursprünglich die Variante (A)RC4 Einsatz [KR97]; da sie hier ohne *Initialisierungsvektor* Abb. 2.4-2 eingesetzt wurde, galt deren Verwendung bei der *Transport Layer Security* (TLS) unsicher [AlF+13] und wurde hieraus entfernt. RC4

Hingegen gelten die (X)*Salsa*- bzw. *ChaCha*-Implementierungen von *Dan Bernstein* heute entsprechend RFC 7539 das Werkzeug der Wahl bei der Internet-Kommunikation; speziell dann, wenn Systeme eingesetzt werden, die über keine AES-Hardwarebeschleunigung verfügen; wie dies üblicherweise bei IoT-Geräten der Fall ist. Salsa und ChaCha

Bei allen Stromchiffren müssen neben dem initialen Schlüssel, der typischerweise relativ kurz ist, deterministisch neue Schlüssel generiert werden: der *Schlüsselstrom*. Dies muss unabhängig von den Eingangsdaten erfolgen und geschieht entweder per Software in *S-Boxen* (Substitutionsboxen) bzw. mittels `modulo`- und `XOR`-Operationen oder in Hardware unter Nutzung von *Linear*- bzw. *None-linear Feedback Shift Register*. In allen Fällen bedarf es für die erste Sequenz der Daten eines *Initialisierungsvektors* IV. Schlüsselstrom

Eine besondere Stromchiffre stellt das *One-Time Pad* dar. Hier ist der geheime Schlüssel gleich lang wie die Nachricht selbst und darf darüber hinaus nicht wieder verwendet werden. Unter diesen Umständen lässt sich beweisen, dass der resultierende Chiffretext nicht algorithmisch entschlüsselt werden kann: Der Code ist nicht 'knackbar'. One-Time Pad

Abb. 2.4-2 illustriert die Arbeitsweise von Stromchiffren, die folgende Kernbestandteile aufweist:

- Initial wird ein gemeinsamer, geheimer Schlüssel gewählt, der relativ 'kurz' ist (zwischen 40 und 1024 Bit).
- Der Schlüsselstrom wird hierbei entweder

[21] Eine solche Bit- bzw. Ziffernfolge wird im Folgenden auch als *Nonce* (oNly use once) bezeichnet.

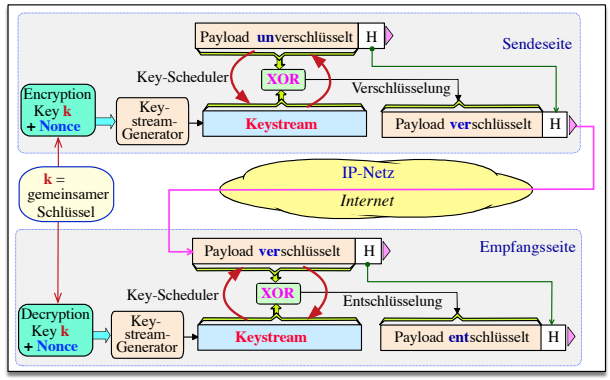


Abb. 2.4-2: Ablauf des Schlüsselstromverfahrens
Verschlüsselung (oben) und Entschlüsselung (unten) von Nachrichten; H: unverschlüsselter Header

- ▷ auf *Hardware-Basis*, z.B. mit Shift-Registern bitweise erzeugt oder
- ▷ per *Software* blockweise mittels Permutationen der S-Boxen, oder – wie bei Salsa – in fixen 64-Byte-Intervallen gebildet.

  Als Ergebnis liegt nun eine Folge von Pseudo-Zufallszahlen vor, deren Güte die eigentliche Verschlüsselung bestimmt. Diese 'Güte' wird durch die vorhandene Entropie und den Algorithmus begrenzt. Der Algorithmus sollte so beschaffen sein, dass keine Korrelation zwischen 'Güte' des Schlüssels und genutzter Rechenzeit besteht, was für *Seitenkanalangriffe* genutzt werden könnte.
- Der Klartext wird mit den ständig wechselnden Schlüsseln per `XOR`-Operation in den Chiffretext überführt.
- Damit die Verschlüsselung auch unter der Bedingung eines kurzen Initialschlüssels k (der unter Umständen konstant bleibt) einmalig ist, wird der *Nonce* bzw. *Initialisation Vector* IV zur Erzeugung des Schlüsselstroms hinzugefügt.

Ein wichtiges Designmerkmal von Stromchiffren besteht darin, den Schlüsselstrom unabhängig von den Nutzdaten zu generieren. Hierbei stehen ursprünglich nur der Schlüssel k und das Nonce n zur Verfügung, die als Eingabe für eine Hashoperation herangezogen werden können. Die Authentisierung der Nachricht erfolgt bei den Stromchiffren daher immer vor der Verschlüsselung: entweder als HMAC in Ergänzung zur Nachricht wie bei TLS 1.2 oder aber im modernen AEAD-Modus, wie für TLS 1.3 vorgesehen.

Coding

Einen anderen Ansatz als die bislang diskutierten Verfahren stellt das von Jürgen Müller entwickelte *Coding2* dar [Mül13]: Hier wird der Schlüsselstrom extrem aufwändig gebildet und Blockweise zwischen den Kommunikationsteilnehmern abgestimmt. Diese Abstimmung wird in *Perfect Forward Secrecy* Manier vorgenommen, sodass aus dem Wissen über frühere Schlüssel kein Nutzen für die folgenden verschlüsselten Sequenzen gezogen werden kann. Die initiale(!) Schlüssellänge beläuft sich auf ca. 50MByte.

Kennzeichnend bei Coding2 ist, das hier im Gegensatz zu anderen Stromschlüsselverfahren nicht von einem relativ kleinen initialen Schlüssel bzw. Initialisierungsvektor (evtl. ergänzt um ein Nonce) gesprochen werden kann, sondern die effektive Schlüssellänge sodass Verfahren der linearen bzw. differentiellen ist so groß wie die zu verschlüsselnde Datenmenge. In diesem Zusammenhang könnte man auch von einem *Pseudo-One-Time-Pad* sprechen.

2.4.2 Blockchiffren

Bei *Blockchiffren* wird die Nachricht in Blöcke aufgeteilt, deren Länge der des Schlüssels entspricht, also z.B. jeweils 128 Bit. Der letzte Block der Nachricht muss auf die Länge des Schlüssels mittels *Padding* aufgefüllt werden. Jeder Block wird mit dem gleichen, *zustandslosen* Schlüssel in den Chiffretext überführt. Als Folge hiervon ist der Chiffretext immer deterministisch mit dem Eingabetext verknüpft.

S-Boxen

Blockchiffren nutzten in der Regel *Substitutionsboxen* (S-Boxen), bei denen eine Abfolge von Permutationen, Transpositionen und Substitutionen vorgenommen wird.

Blockverschlüsselung anhand des Zauberwürfels

> Wir können uns die grundlegende Operation von S-Boxen anhand eines 'Zauberwürfels' vorstellen, indem der Eingabewert auf dem Würfel verteilt und dieser anschließend mehrfach gedreht wird. Welche Drehungen hierbei vorgenommen werden, entspricht dabei unserem Schlüssel und bleibt geheim. Führt man die Operationen in umgekehrter Reihenfolge aus, erhält man wiederum den Ursprungstext.
>
> Wenn wir unsere Eingangsnachricht enummerieren, würde sich dies mit dem Text 'geheim' wie in Abb. 2.4-3a darstellen. Bei einem 'Known-Plaintext'- oder 'Chosen-Plaintext'-Angriff ließe sich hieraus die Struktur der Eingangswerte ermitteln. Sinnvoller ist das Vorgehen, wie in Abb. 2.4-3b gezeigt: Wir verteilen den Eingangstext 'zufällig' auf dem Würfel – was als *Konfusion* bezeichnet wird – und drehen dann [Abb. 2.4-3c]. Hierdurch entfällt der 'Grundzustand' des Würfels, der Rückschlüsse auf die Eingangsdaten liefern könnte. Führen wir dieses Verfahren auch *nach* der Verschlüsselung [Abb. 2.4-3d] durch, sprechen wir von *Diffusion*: Einem Angreifer wird nun auch das Erraten des Schlüssels schwerer gemacht, da die Korrelation von Chiffretext und Schlüssel aufgehoben wird.

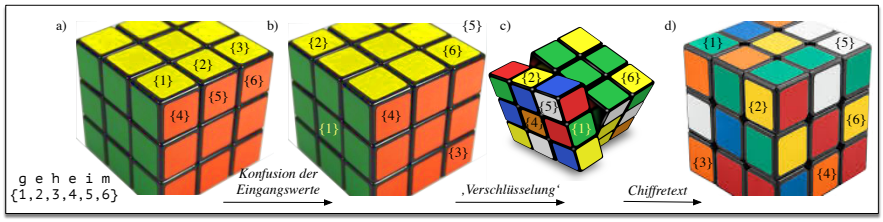


Abb. 2.4-3: Illustration des Konfusions- und Verschlüsselungsprinzips bei der Blockverschlüsselung; der Wert '5' liegt anfangs auf der abgewandten Seite des Würfels
a) Naiver Ansatz mit deterministischer Verteilung der Anfangswerte, b) Konfusion: Zufällige Verteilung der Anfangswerte, c) Verschlüsselung durch Rotation, d) Ergebnis der Verschlüsselung (ohne Diffusion)

DES, AES, IDEA, Blowfisch

Während das DES-Verfahren – auch in seiner verbesserten Form 3DES – heute kaum mehr eine Bedeutung besitzt, spielen neben AES nur noch das 1990 an der ETH Zürich von *James Massey* und *Xueija Lai* entwickelte IDEA- (International Data Encryption

Algorithm) sowie *Bruce Schneiers* Blowfisch-Verfahren eine gewisse Rolle. Da das IDEA-Verfahren ursprünglich patentgeschützt war, konnte es bis zum Auslaufen des Patents in 2011 in freier Software nicht eingesetzt werden.

Neben der eigentlichen Verschlüsselung, die in der Regel über *Substitutionsboxen* erfolgt, ist der Umgang mit dem Schlüssel ein wichtiges Implementierungsdetail. So besitzt der DES-Mechanismus zwar eine Schlüssellänge von 64 Bit, von denen aber nur 56 Bit kryptographisch wirksam und die weiteren 8 Bit als Paritätsbits zu betrachten sind. Beim AES-Verfahren können hingegen unterschiedliche Schlüssellängen genutzt werden, was dieses Verfahren einerseits sehr flexibel macht und andererseits auf einer universellen Implementierung aufbaut, die schneller ist als DES.

Konfusion, Verschlüsselung, Diffusion

Alle Verfahren sind so gewählt, dass

- die Verwürfelung (*Konfusion*) der Eingangsdaten,
- die eigentliche *Verschlüsselung* der so erzeugten Datensequenz und
- eine abschließende *Diffusion* der Ausgangsdaten

gewährleistet wird. Aus dem erzeugten Chiffretext kann also unter keinen Umständen mehr auf den den ursprünglichen Inhalt geschlossen werden. Weder *Known-* noch *Chosen-Plaintext*-Attacken sind daher von Erfolg gekrönt.

Enigma-Code

Das 'Brechen' des Enigma-Codes, der von der Deutschen Wehrmacht im Zweiten Weltkrieg benutzt wurde, durch polnische Informatiker, aber vor allen vom den Briten *Alan Turing*[22], einem der Grundväter der Informatik, war auf der Tatsache begründet, dass zwar die Konfusion bekannt und implementiert war, aber (noch) nicht die Diffusion.

Electronic Code Book

Zentrales Problem der Blockchiffren ist ihre *Zustandslosigkeit*: Ein Datenblock mit festem Inhalt wird immer als Chiffretext mit konstantem Inhalt umgewandelt. Wir bezeichnen dieses Vorgehen auch als *Electronic Code Book* ECB, da die Verschlüsselung quasi über eine *Lookup-Tabelle* erfolgt. Unabhängig vom Inhalt der Nachricht weiß ein potenzieller Angreifer immer, dass es sich um dieselbe Nachricht handelt. Inhalte und Schlüssel können zwar *obfuskiert* werden, doch oft reicht schon das Wissen um die Unveränderlichkeit einer Nachricht, um qualifizierte Rückschlüsse zu ziehen[23].

2.4.3 Klassische Betriebsarten

Der ECB-Modus verschlüsselt bei der Blockverschlüsselung und bei Plaintext-Nachrichten unabhängig voneinander. Ein fehlerhaft übertragener Block besitzt somit keinen Einfluss auf die anderen empfangenen Nachrichten. Alternativ können die Datenblöcke während ihrer Verschlüsselung auch untereinander verschränkt werden: *Blockverschlüsselung* → *Stromverschlüsselung*.

Je nachdem, wie diese Verschränkung vorgenommen wird, ergeben sich die sogenannten Betriebsarten, von denen zunächst die folgenden Klassiker vorgestellt werden sollen [Abb. 2.4-4]:

[22] vgl. Wikipedia https://de.wikipedia.org/wiki/Alan_Turing
[23] ein schönes Beispiel hierfür ist der 'Tux' bei Wikipedia https://de.wikipedia.org/wiki/Electronic_Code_Book_Mode

- *Cipher Block Chaining* (CBC),
- *Cipher Feedback Mode* (CFB), sowie
- *Output Output Feedback* (OFM).

Vergleichbar den Stromchiffren, muss auch hier ein *Initialisierungsvektor* beiden Kommunikationspartnern zu Anfang bekannt sein, der die gleiche Länge wie der Block bzw. der Schlüssel besitzen muss.

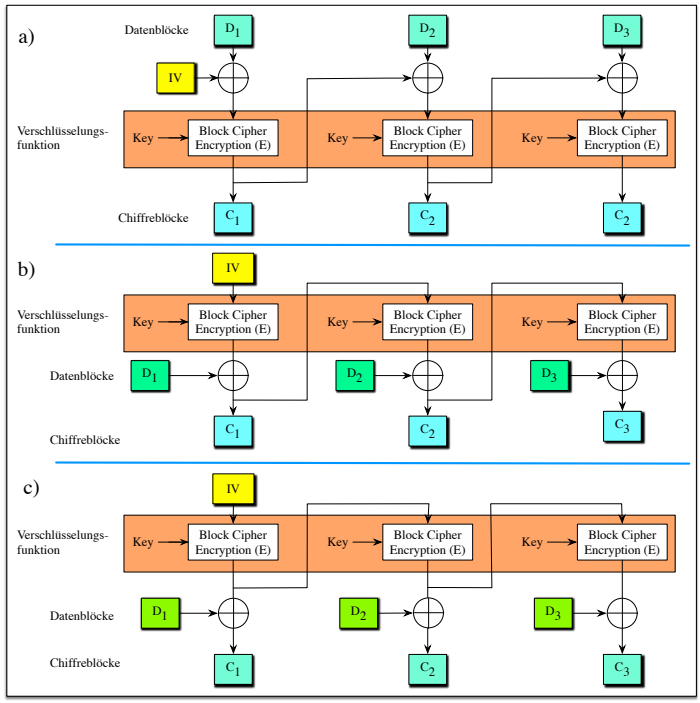


Abb. 2.4-4: Verschlüsselung bei den Betriebsmoden a) CBC, b) CFB und c) OFM für Blockchiffren; $\oplus$ ist die XOR-Operation, IV: Initialisierungsvektor

Cipher Block Chaining

Im *CBC-Mode* [Abb. 2.4-4a] wird der erste Datenblock D_1 mit dem IV per XOR-Operation zusammengeführt und dann gemeinsam mit dem Schlüssel k verschlüsselt. Der resultierende Chiffreblock C_1 liegt nur vor und wird zudem als Input für die XOR-Operation des zweiten Datenblocks D_2 genutzt. Während dies für den Sender eine einfache Rekursion darstellt, muss der Empfänger über alle Chiffreblöcke C_n in Reihe und vollständig verfügen. Fehler in einem empfangenen Chiffreblock C_i führen dazu, dass keine Entschlüsselung des Chiffreblocks C_{i+1} möglich ist; wohl aber wieder C_{i+2}, sofern C_{i+1} korrekt eingegangen ist, was durch die Unabhängigkeit der XOR-Operationen begründet ist. Durch dieses Verhalten lässt sich die Entschlüsselung im Gegensatz zur Verschlüsselung auch parallelisieren

Cipher Feedback Mode

Im *CFB-Mode* [Abb. 2.4-4b] wird hingegen zunächst der IV verschlüsselt und das Ergebnis mit dem ersten Datenblock D_1 per XOR in den Chiffretext C_1 überführt und zudem als Input für die nächste Iteration, d.h. die Verschlüsselung von D_2 benutzt.

Auch hier gilt, dass Fehler in den empfangenen Chiffreblöcken nur eine begrenzte Auswirkung besitzen, sodass von einer *Selbstsynchronisation* gesprochen wird. Wie dargestellt, ist die letzte Operation ein `XOR` und keine eigentliche Verschlüsselung, sodass im CFB-Mode auch Nachrichten übertragen werden können, die nicht ein Vielfaches der Block- bzw. Schlüssellänge entsprechen, wodurch das *Padding* entfallen kann.

Output Feedback Mode

Der *OFB-Mode* [Abb. 2.4-4c] nutzt auch den IV als Eingabe zur ersten Verschlüsselungsoperation. Im Gegensatz vom CFB wird nur der so generierte Wert sowohl als Input für die `XOR`-Operation mit dem Datenblock D_1 als auch für die folgende Blockverschlüsselung genutzt. Als Konsequenz geht die Selbstsynchronisation verloren. Auf der Habenseite des Verfahrens steht, dass die eigentliche Verschlüsselung über die `XOR`-Operation erfolgt; die Blockverschlüsselung dient so im Wesentlichen als Schlüsselgenerator. Demzufolge kann anstatt eines Blockverschlüsselungsalgorithmus auch eine Hashfunktion genutzt werden. In diesem Zusammenhang spricht man auch von einem *Pseudo One-Time Pad*.

Obwohl die hier dargestellten Betriebsarten der einfachen ECB-Verschlüsselung vorzuziehen sind, bergen sie bei der naiven Nutzung durchaus Stolperfallen. So kann z.B. das beim CBC notwendige *Padding* dazu genutzt werden, einen *Known-Plaintext-Angriff* durchzuführen. Ähnliche Angriffsszenarien ergeben sich unter Berücksichtigung bekannter Informationen wie z.B. über den HTTP-Header, was als *Poodle-Angriff* bekannt wurde [MDK14], sodass speziell CBC im Endeffekt die Sicherheit nicht erhöhen, sondern sogar verringern kann.

2.4.4 Counter Mode und AEAD

Mittels der Blockverschränkung wird aus der zustandslosen Blockchiffre eine zustandsbehaftete Stromchiffre, d.h. es liegt im Grunde ein dualer Betriebsmode vor. Mit Blick auf Tab. 2.3-1 lässt sich vermuten, dass die oben dargestellten Verfahren, die auf der Nutzung eines Initialisierungsvektors IV als *Nonce* beruhen, weitere Möglichkeiten offenlassen, bei denen

- die Nachrichtenabfolge *sequenziert* wird, was als *Counter-Mode* (Zähler-Mode) bezeichnet wird, oder aber
- statt eines zufälligen IVs die Nachrichtenblöcke durch die Angabe eines *Authenticators* ergänzend markiert werden können. Dies ist Gegenstand des *Galois Counter Mode* (GCM) und wird auch als *Authenticated Encryption with Associated Data* AEAD bezeichnet und stellt heute den Stand der Technik dar.

Counter-Mode

Der *Counter-Mode* CM geht zunächst ähnlich wie der CFB bzw. OFM vor, bei denen zunächst der Initialisierungsvektor mit der Blockchiffre verschlüsselt wird. Allerdings besitzt hier der IV eine andere Bedeutung:

- Ein Teil des IV wird als *Nonce* interpretiert, während
- der zweite Teil ein *Zähler* ist, der monoton hochgezählt wird, wobei natürlich auf einen Überlauf des Zählers als Integer zu achten ist.

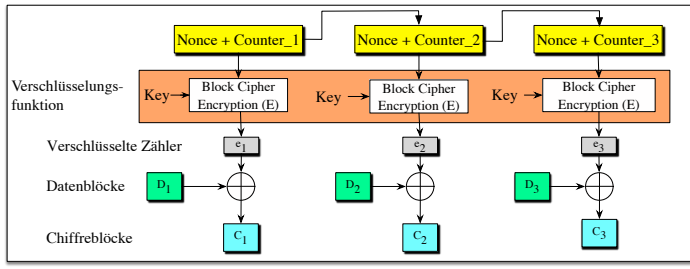


Abb. 2.4-5: Counter-Mode-Blockverschränkung; $\oplus$ ist die XOR-Operation, e_i sind die mit k verschlüsselten Zähler

Im Gegensatz zu CFB/OFM werden nun nicht die Nachrichtenblöcke untereinander verschränkt, sondern dies geschieht ausschließlich über den Zähler e_i. Damit die verschlüsselten Nachrichtenblöcke entschlüsselt werden können, muss die Reihenfolge korrekt und vollständig erhalten bleiben. Diese Entkopplung führt allerdings auch dazu, dass die Erzeugung der Einzelblöcke (und auch die Wiederherstellung der Ursprungsinformation) parallelisiert werden kann.

Galois Counter Mode

Der *Galois Counter Mode* (GCM) ist die erste nicht-naive Blockverschränkungsmethode, bei der zusätzlich in den verschränkten und verschlüsselten Chiffreblöcken eine Authentisierungsinformation untergebracht wird. GCM wurde im Jahre 2004 von *David A. McGrew* und *John Viega* vorgestellt [MV04] und findet seit etwa 2007 zunehmend Einsatz. GCM weist zwei zentrale Komponenten auf [Abb. 2.4-6]:

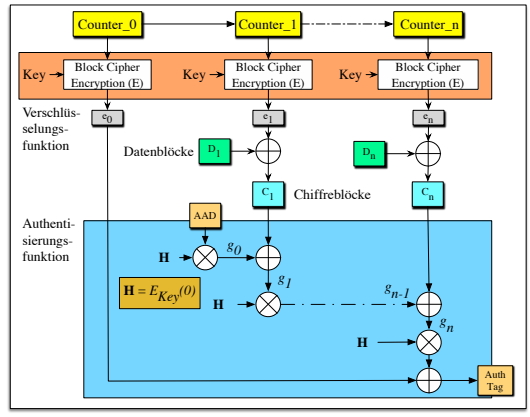


Abb. 2.4-6: Galois Counter Mode mit den Verschlüsselungs- und Authentisierungsoperationen; $\otimes$ bezeichnet Multiplikation auf dem Galois-Feld, $\oplus$ ist die XOR-Operation

1. Die Verschlüsselung findet mittels Blockchiffren statt, bei denen die Blocklänge 128 Bit beträgt und im Counter Mode abläuft, d.h. es werden die Zähler mit k verschlüsselt ($k_i \ \forall_{i=0\ldots n}$) und die Chiffreblöcke über ein XOR gebildet.
2. Die Authentisierung erfolgt auf Grundlage einer verketteten Multiplikation im *Galois-Feld*, für das das irreduzible Polynom $P(x) = x^{128} + x^7 + x^1 + x + 1$ als GF(2^{128}) genutzt wird. Diese Rechenoperation muss für jeden Nachrichtenblock

nur ein einziges Mal durchgeführt und erklärt auch die Beschränkung auf 128 Block- bzw. Schlüssellängen.

Wie in Abb. 2.4-6 dargestellt, wird für jeden zu verschlüsselnden Block ein *Authentication-Parameter* g_i iterativ gebildet:

- Zunächst wird ein sogenannter *authentication subkey* `H` berechnet, der mittels des Blockschlüssels k den verschlüsselten Wert von '0' aufweist: $H = E_k(0)$.
- Ausgehend von einem öffentlich bekannten Wert, einer Zeichenkette, die als *Additional Authenticated Data* bezeichnet wird, erfolgt eine Multiplikation im Galois-Feld, und es ergibt sich: $g_0 = AAD \otimes H$.
- Für jeden weiteren Block i wird berechnet: $g_i = (g_{i-1} \oplus C_i) \times H$
- Der letzte Datenblock erhält ein sogenanntes *Authentication-Tag* mittels: AuthTag = $(g_n \otimes H) \oplus e_0$.

Der Empfänger der verschlüsselten Blockreihe kann aus diesen Informationen unter Kenntnis des Initialisierungsvektors IV und des Blockschlüssels k sowohl die *Integrität* als auch die *Authentizität* der Nachricht verifizieren. Für nähere Details sei auf [Dwo07] verwiesen.

2.5 Schlüsseltauschverfahren

Die Entwicklung der *asymmetrischen Verschlüsselung* hat für die Kryptographie eine vergleichbare Bedeutung wie die *Kopernikanische Wende* für die Astronomie und stellt neben der *Digitalisierung* der Informationen und dem *Internet* als Kommunikationsplattform ein Fundament des dritten Jahrtausends dar.

Asymmetrische Verschlüsselung

Erst Mitte der 70er Jahre von *Diffie-Hellman*[24] (DH) und *Rivest/Shamir/Adleman* (RSA) mit unterschiedlichen Ansätzen 'erfunden', wurden die Konsequenzen der asymmetrischen Verschlüsslung und des sicheren 'Schlüsseltauschs' sehr schnell erkannt. Anfang der 90er Jahre des letzten Jahrhunderts wurden entsprechende Algorithmen in Software umgesetzt und als Internetanwendungen und -protokolle (RFC 2437/RSA und 2631/DH) allgemein verfügbar gemacht.

Wie bereits in Abschnitt 2.2 dargestellt, lassen sich mit den Mitteln der 'asymmetrischen' Kryptographie die beiden Probleme

1. Schlüsseltausch (*key exchange*) mittels des κ-Primitives und
2. Beglaubigung (*signature*) über das σ-Primitiv

lösen. Im Folgenden wollen wir schwerpunktmäßig auf den Schlüsseltausch eingehen und die Frage der digitalen Signaturen bei der Diskussion von TLS (Abschnitt 7.2) wieder aufgreifen.

[24] DH basiert auf *Merkle's puzzle* [Mer80], das als 'The Problem – Ready for Ralph' sogar als Hommage zum Gegenstand eines Popsongs wurde [Godley & Creme: *Ismism*, 1981].

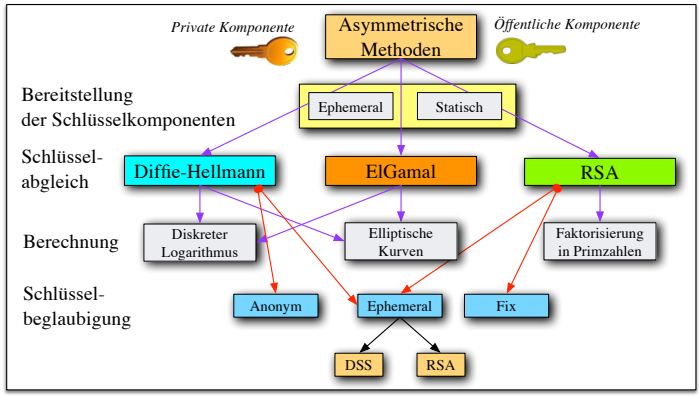


Abb. 2.5-1: Überblick über die asymmetrischen Verschlüsselungsverfahren
DSS: Digital Signature Standard, RSA: Rivest/Shamir/Adleman-Algorithmus

Einordnung der asymmetrischen Verfahren

Abb. 2.5-1 liefert einen Überblick über die gängigen asymmetrischen Verfahren und ihren Einsatz. Folgende wichtige Elemente wollen wir kurz diskutieren:

- Generell können der Schlüssel, also der *private key* sowie der *public key statisch* zur Verfügung gestellt werden, oder aber *ephemeral*, werden also *während* des Verbindungsaufbaus erzeugt. Bei RSA-Verfahren wird in der Regel von statischen Schlüsseln Gebrauch gemacht. *On the fly*, d.h. während der Verbindungsaufnahme erzeugte RSA-Schlüssel haben eine mindere Qualität, stehen aber als sogenannte Exportschlüssel zur Verfügung.

  Schlüssel-bereitstellung

  Diffie-Hellman benötigt zunächst die öffentlichen *DH-Domain-Parameter* (DH), die festlegen, in welchem Teil des Lösungsraums die Schlüssel liegen. Die DH-Parameter sind daher nicht spezifisch, sondern stellen ein Protokollartefakt dar, auf das allerdings nicht verzichtet werden kann, und teils als externe Parameter, teils als feste Werte in der Software vorliegen. Der *private key* wird – gemeinsam mit dem *public key* während des Ablaufs des DH-Verfahrens von beiden Kommunikationspartnern durch Zufallszahlen gebildet (die aus der DH-Parameter 'Lösungsmenge' abgeleitet werden), sodass man auch von *Perfect Forward Secrecy* (PFS) spricht.

  PFS

- Beim interaktiven Schlüsselabgleich findet entweder *Diffie-Hellman* oder aber *RSA* Verwendung. *ElGamal* – als Weiterführung von Diffie-Hellman – wird bei den Übertragungsprotokollen eingesetzt, wo es um einen Nachrichtenaustausch geht.

  Schlüsselab-gleichsverfahren

- Ausgehend von der vorliegenden Schlüsselkomponente, muss nun überprüft werden, ob der Kommunikationspartner das passende Gegenstück besitzt.
  RSA und DH setzen hierbei auf Funktionen, bei denen in der 'Vorwärtsrichtung' die Berechnung einfach (z.B. Multiplikation), in der 'Rückwärtsrichtung' aber schwierig ist: das *multiplikative Inverse*, falls die Ordnung der Gruppe nicht bekannt ist. Bei RSA fußt die Multiplikation auf großen Primzahlen, während bei Diffie-Hellman und ElGamal das Problemen des *diskreten Logarithmus* oder der Multiplikation auf *elliptischen Kurven* zu lösen ist.
  Alle asymmetrischen Verfahren fußen auf der Annahme, dass die Berechnung des

  Schlüssel-abgleich

Gegenstücks, also z.B. des *private key* für jemanden, der lediglich den *public key* kennt, ausgesprochen schwierig ist, d.h. nicht in sinnvoller Zeit realisiert werden kann. Bei RSA ist diese Aussage mit dem Bestimmen der Primfaktoren p, q aus dem bekannten Modulus $n = p * q$ verknüpft. Bei Diffie-Hellman besteht hingegen bei bekannten DH-Parametern die Möglichkeit der A-priori-Faktorisierung eines Teils des Schlüssels [Adr+15].

Schlüsselbeglaubigung

- In der Regel wird das ausgetauschte Schlüsselmaterial vom Server während der Verbindungsaufnahme zusätzlich *beglaubigt*. Dies erfordert ein *Vertrauenssystem*, das in Form der *Public Key Infrastructure* (PKI) existiert, aber zunehmend kompromittiert ist.

Anonymous DH

Die Beglaubigung kann auch entfallen; hierbei sprechen wir dann von einem *anonymen Schlüsseltausch*, der aber die Gefahr von *Man-in-the-Middle*-Angriffen (MitM) mit sich bringt.

2.5.1 Ablauf des RSA-Schlüsseltauschs

Beim RSA-Verfahren braucht der Server sowohl einen *public* als auch einen *private key* zu besitzen, die beide statisch sein können. In der Regel liegt der *public key* in einem *X.509-Zertifikat* vor [Abschnitt 2.7], der *private key* in einem sogenannten *key file*, der häufig durch ein Passwort geschützt ist.

Schlüsselmaterial

Will man qualifizierte RSA-Schlüssel erzeugen, sind zunächst große Primzahlen p und q bereitzustellen, die den möglichen Lösungsraum beschreiben. Teilweise werden aber kleine Primzahlen nicht zugelassen, was diesen beschränkt. Ein weiteres Problem besteht in der Qualität großer Primzahlen, die – sofern nicht bekannt bzw. tabelliert – in ihrer *Primalität* nur abgeschätzt werden können. Für den Exponenten e wird in der Regel ein pragmatisches Verfahren gewählt, indem dieser auf die *5te Fermat'sche Zahl* $2^{16} + 1 = 65537$ gesetzt wird. Bei der freien Wahl von e ist zu darauf zu achten, dass e und $\varphi(n) = (p-1) \cdot (q-1)$ teilerfremd sind und somit ein ein-eindeutiger *private key* d gebildet werden kann. Dies gilt trivial, falls e ebenfalls eine Primzahl ist.

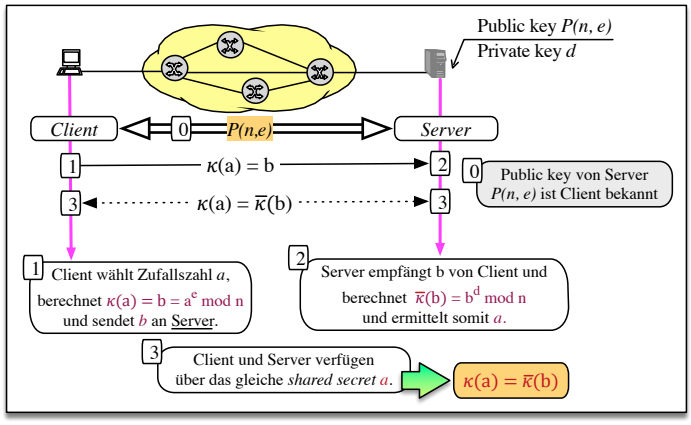


Abb. 2.5-2: Schlüsseltausch beim RSA-Verfahren

Schlüssel-bereitstellung

Zunächst der RSA *public key* als $P(n,e)$ abgeleitet, der sich aus dem Produkt der beiden Primzahlen (dem *Modulus* $n = p*q$) und dem *Exponenten* ergibt. Die Berechnung des (inversen) *private key* d erfolgt durch den Einsatz des *erweiterten Euklid'schen Algorithmus*: $d = \frac{1+\varphi(n)}{e}$ wobei dieser teilerfremd zum Exponenten sein muss.

Fermat Lösung

Da n in Klartext übermittelt wird, lässt sich zunächst eine Zahl $k = \sqrt{n}$ berechnen, von der auf die benachbarten Primzahlen p' und q' geschlossen werden kann. Ergibt das Produkt beider Zahlen den *Modulus* n ist der RSA-Algorithmus gebrochen, da sich aus dieser Kenntnis der *private key* d berechnen lässt. Deswegen verlangt das RSA-Verfahren, dass die gewählten Primzahlen einen genügend großen 'Abstand' voneinander besitzen.

Schritt	**Berechnung**	**Bemerkung**
Schlüsselerzeugung	$p, q \in \mathbb{P}$	*Primzahlen* mit deutlich verschiedenen Längen
	$n = p \cdot q$	*Modulus*
	$\varphi(n) = (p-1) \cdot (q-1)$	*Euler'sche Zahl*
	e	*Exponent*, z.B.: $e = 3; e = 2^{16} + 1 = 65537$
Public key	$P = (n, e)$	ist öffentlich oder wird mitgeteilt
Private key	$d \equiv e^{-1} \bmod \varphi(n)$	'inverse von e' über *erweiterten Euklid'schen Algorithmus*
Initiierung Schlüsseltausch	$\kappa(a) = a^e \bmod n = b$	a wird gewählt, $\kappa(a)$ berechnet, $b \rightsquigarrow$ Server
Vollendung Schlüsseltausch	$\bar{\kappa}(b) = b^d \bmod n = a$	b wird empfangen und a mittels $\bar{\kappa}(b)$ berechnet

Tab. 2.5-1: Schlüsselerzeugung und -tausch beim RSA-Verfahren

RSA-Schlüsseltausch

Damit der Schlüsseltausch [Abb. 2.5-2] vorgenommen werden kann, muss der Client über den *public key* des Servers verfügen. Vom Client wird eine Zufallszahl als *shared secret* erzeugt, das mittels des *public key* des Servers vertraulich übertragen wird und von diesem über dessen *private key* entnommen werden kann. Beiden Seiten steht nun das *shared secret* zur Verfügung, was als Ausgangspunkt für weitere kryptographische Operationen genutzt wird.

2.5.2 Ablauf des DH-Verfahrens

Diffie-Hellman (kurz DH) ist ein Verfahren zur Verständigung auf einen *gemeinsamen Schlüssel*, wobei (derzeit) zwei unterschiedliche mathematische Verfahren genutzt werden können:

1. Der *diskrete Logarithmus* (DL), wobei der Sachverhalt ausgenutzt wird, dass das Potenzieren schnell, die Umkehrfunktion – der Logarithmus – aber eine vergleichsweise viel schwierigere mathematische Operation ist.
2. *elliptische Kurven*, bei der geometrische Transformationen auf geeigneten Kurven zweiten Grades vorgenommen werden.

DH-Domain-Parameter

Ausgangspunkt des DH-Verfahrens ist zunächst die Bereitstellung der Diffie-Hellman-*Domain-Parameter* $DH(g,p)$ [Tab. 2.5-2]. Diese sind *per constructionem* öffentlich und müssen sowohl dem Server als auch dem Client bekannt sein. Während die Primzahl p primär für die Größe des Lösungsraums verantwortlich ist – und somit den Schwierigkeitsgrad für eine potentiellen Angreifer bestimmt – wird der Generator g (vergleichbar e beim RSA-Verfahren) trivial vorgegeben. $g = 2$ und $g = 3$ sind eine