Sicherheit

Sicherheit

- Wahrnehmung des Themas in der Öffentlichkeit
  - Viren
    - wahrgenommen: großer eigener Schaden
    - Initiative zum Kauf von Schutzprogrammen
    - Einfach Handhabung von Virentestprogrammen: installieren, fertig
  - Würmer
    - wahrgenommen: mittlerer eigener Schaden; Schaden der anderen
    - Rechner und Netz langsam; geringere Verfügbarkeit als „schlechten Tag“ abtun

Wahrnehmung der Sicherheit

  - Würmer verbreiten sich meist per Mail
  - Keine Bereitschaft zu eingeschränktem Verhalten
- Hackerangriffe
  - wahrgenommen: geringer eigener Schaden („was wollen die schon bei mir?“)
  - Bedrohungsszenarien für viele zu abstrakt (verteilter „Denial-of-Service-Angriff“)
  - Aufwand zur Abwehr hoch (Einrichtung von Firewalls)
- Angriffe auf die Vertraulichkeit
  - selten als eigener Schaden wahrgenommen
  - „ich habe nichts zu verbergen“

Wahrnehmung der Sicherheit

- Abwehr der Gefahren mit (objektiv) großem Aufwand verbunden
- Schutzmaßnahmen unterbleiben daher meist

- Fazit
  - Schutzmaßnahmen werden gewöhnlich nur getroffen,
    - wenn der **eigene** Schaden als beträchtlich eingestuft wird
    - oder wenn der damit verbundene Aufwand als minimal empfunden wird
  - Problem
    - Viele Sicherheitstechniken wurden erst nachträglich den existierenden Anwendungen hinzugefügt
    - Verlust an Bequemlichkeit verringert Akzeptanz

Einleitung

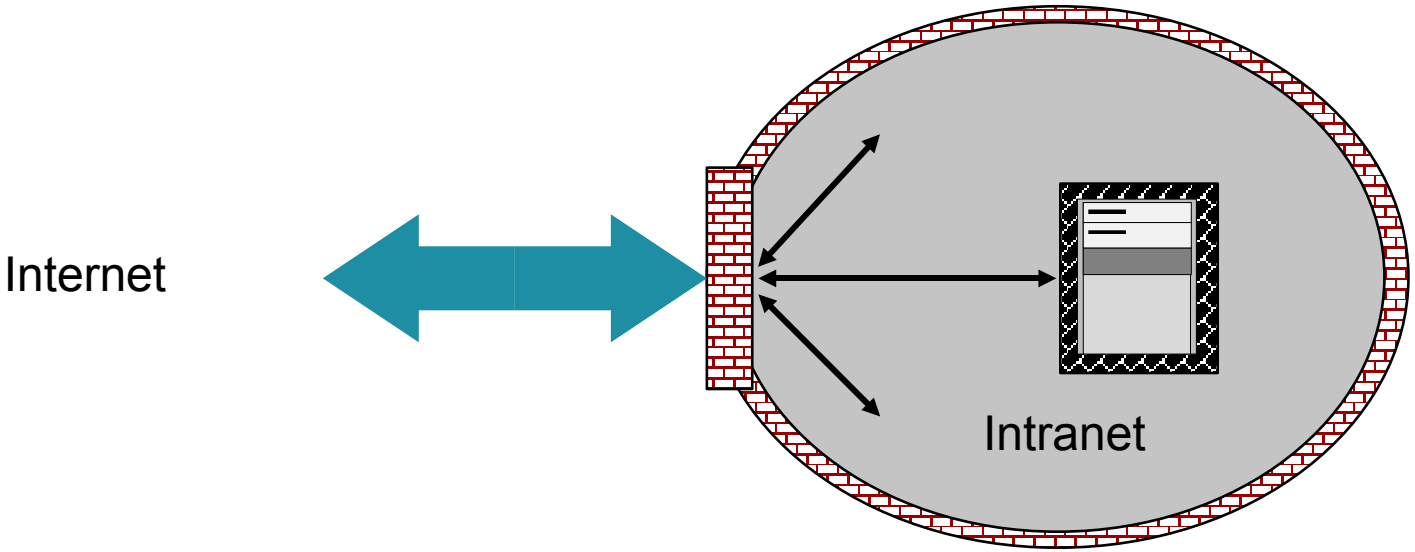
- Fragen zum Thema Sicherheit
  - Was muss geschützt werden?
  - Vor wem muss geschützt werden?
  - Gegen welche Bedrohungen?
  - Gegen welche Angriffsformen?
  - Mit welchen Hilfsmitteln ?

Aufgabenbereiche der Sicherheit

- Authentifizierung
  - Kennt mich das System?
- Autorisierung
  - Was darf ich im System?
- Vertraulichkeit
  - Wer sieht meine Daten oder Aktionen sonst noch?
- Integrität
  - Sind meine Daten unverändert?
- Nachweisbarkeit
  - Kann man mir zweifelsfrei Aktionen zurechnen?

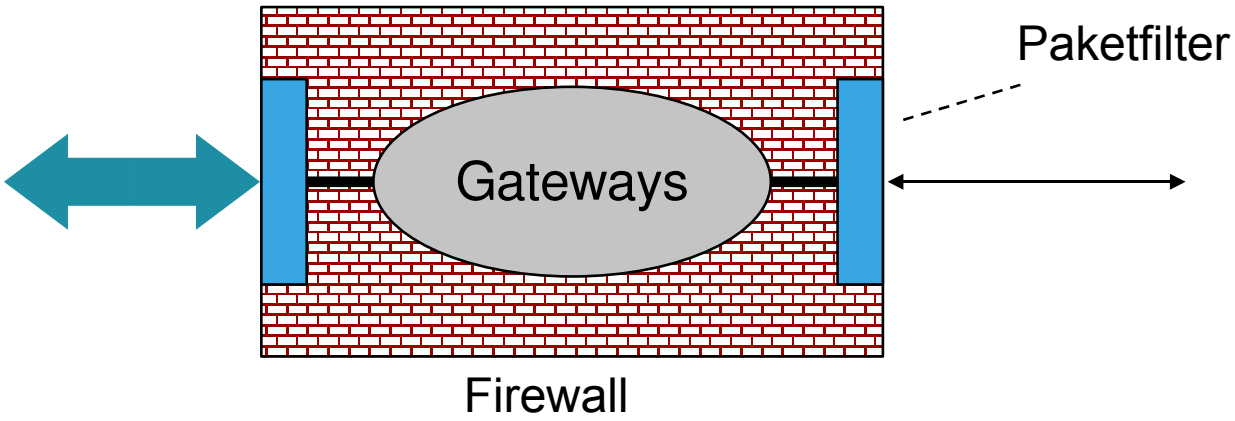
Prinzipien

- Rechner-Sicherheit
  - durch Verschlüsselung (Kryptographie) etc.
- Netz-Sicherheit
  - Schutz von Netzbereichen durch eine Barriere (Firewall)



Firewall

- Konzept
  - jeglicher Verkehr zwischen innen und außen muss die Firewall passieren
  - nur der vorgesehene Verkehr kann passieren
  - die Firewall ist geschützt gegen Angriffe



Paketfilter

- Prinzip
  - Ausfiltern von Paketen mit bestimmten Adressen oder Ports (Quelle oder Ziel)
  - Spezifizierung durch Positivliste akzeptabler bzw. Negativliste zu verwerfender Pakete

Zugang	**Empfänger**	**Port**	**Sender**	**Port**	**Kommentar**
blockiert	*	*	xyz	*	kein Vertrauen zu xyz
erlaubt	mailex	25	*	*	Post-Eingang
blockiert	*	*	*	*	Default

DNS-Name
oder IP-Adresse

Portnummer

* = alle

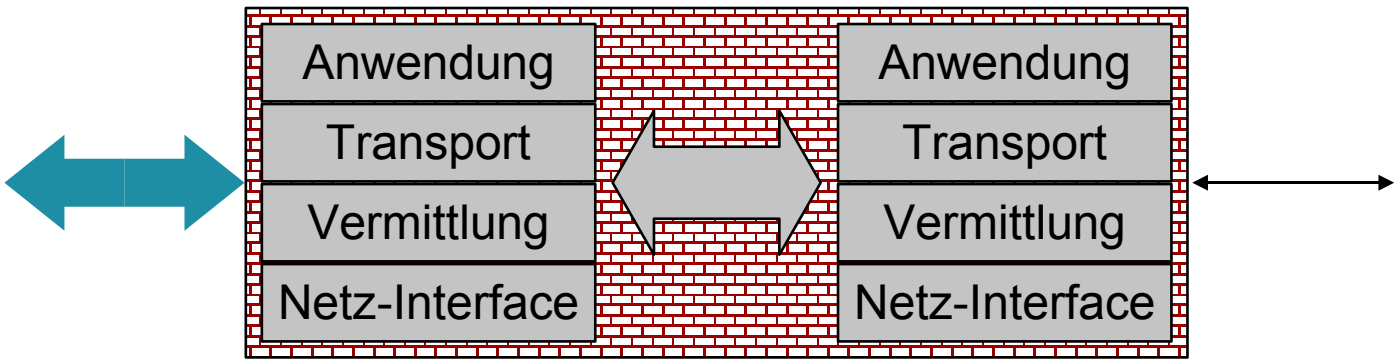
Probleme bei Paketfiltern

- Protokolle mit zufällig gewählten Port-Nummern
- Protokolle mit mehreren Verbindungen, z.B. FTP
- Rückantworten auf ausgehende Verbindungen sollten möglich sein
  - Erkennung durch gesetztes ACK-Bit in TCP
  - nicht möglich bei UDP
- bekannte Port-Nummern könnten als Ausgangspunkt eines Angriffs dienen

Paketfilter sind i.A. für ein Sicherheitskonzept nicht ausreichend.

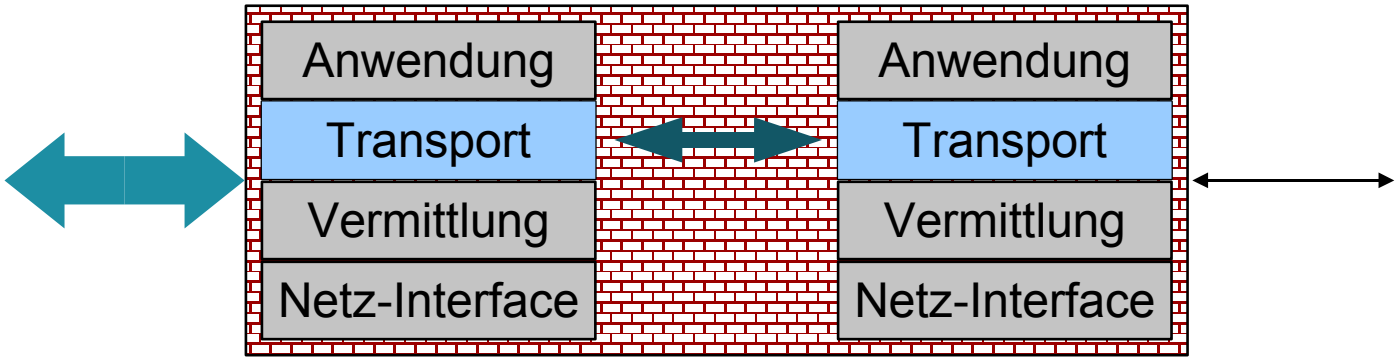
Gateways

- Prinzip
  - angesiedelt auf den höheren Protokollebenen
  - fungiert als beiderseitiger Protokoll-Endpunkt (Proxy)
  - Umsetzung der Pakete zwischen den beiden Verbindungsteilen
- Einsatz in verschiedenen Ebenen
  - Transport
  - Anwendung



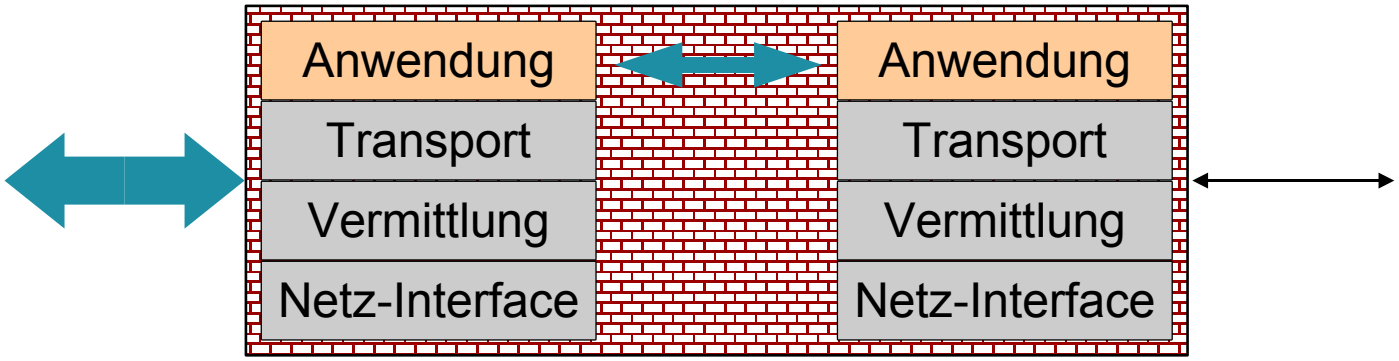
Transportschicht-Gateway

- Anwendungen kontaktieren das TS-Gateway, um eine TCP-Verbindung zu einer bestimmten Maschine aufzubauen
- Nach erfolgreichem Verbindungsaufbau arbeitet das TS-Gateway als „Draht“, d.h. schleust die Pakete durch
- Erfordert Modifikation der Anwendung, um zweistufigen Verbindungsaufbau abwickeln zu können



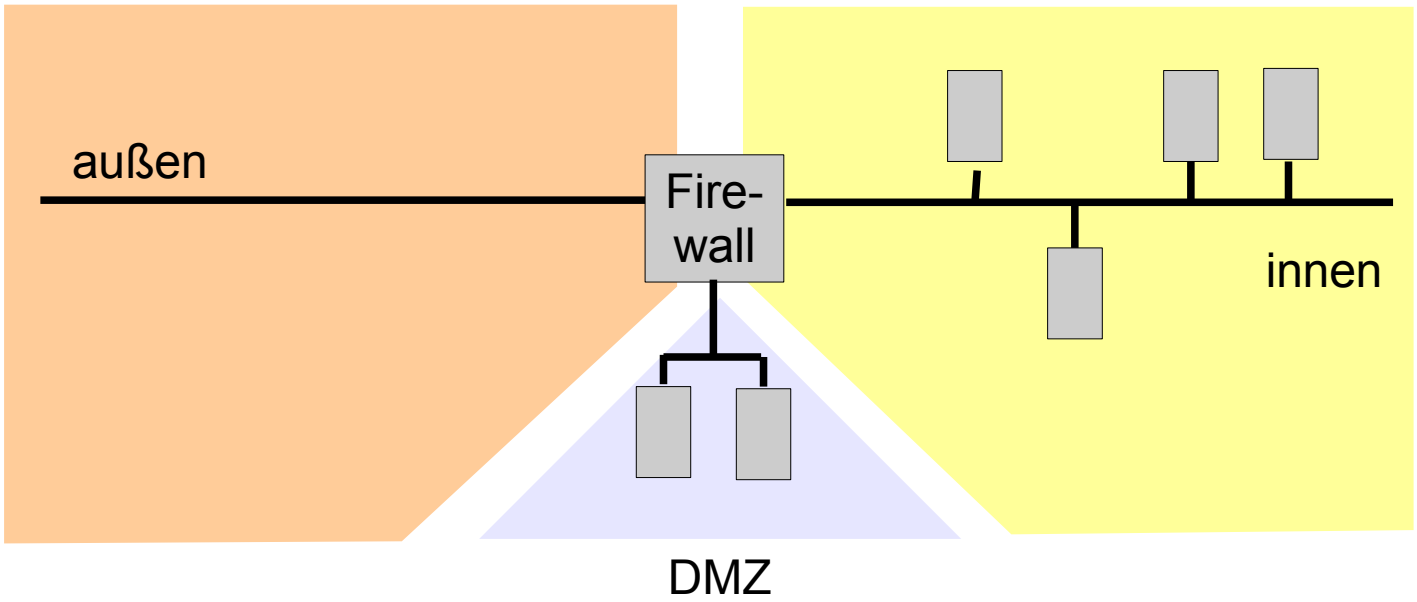
Anwendungsschicht-Gateway

- Anwendungsspezifische Kontrolle des Zugriffs
- verhält sich aus Client-Sicht wie der Anwendungsserver, z.B. Mail-Gateways
- ermöglicht weit gehende Analyse des Inhalts
  - Test auf Schadcode auf Webseiten oder in Mailnachrichten
- keine Modifikation des Klientenprogramms



Demilitarisierte Zone

- Dritter Sicherheitsbereich
  - erreichbar von außen (Webserver, Mailserver)
  - erreichbar von innen
  - Filter verhindern Zugriff von DMZ auf inneren Bereich
- Meist durch separates Netz realisiert (eigene Netzschnittstelle)



Sicherheitskonzept

- Firewalls können Angriffe von außen abwehren
  - sofern diese Angriffe durch die Verbindungsaufnahme zu eigenen Diensten bestehen
  - Blockieren unerwünschter Verbindungen
- Viren/Würmer und anderer Schadcode können nur durch Analyse entdeckt werden (Scanner)
- Sicherheit der Daten ist **nicht** gewährleistet
- Daten können
  - ausgespäht werden (Authentifikationsdaten!)
  - verändert werden
  - kopiert werden

Schutz der Daten erforderlich → Verschlüsselungstechnik (Kryptografie)

Kryptografie

- Zwei grundsätzliche Verfahren
  - geheime Schlüssel (Secret Key) = symmetrische Systeme
    - Sender und Empfänger haben ein Paar geheimer Schlüssel (K, K')
    - Sender benutzt K zum Chiffrieren
    - Empfänger dechiffriert mit inversem Schlüssel K'
  - öffentliche Schlüssel (Public Key) = asymmetrische Systeme
    - Empfänger gibt öffentlichen Schlüssel K bekannt, hält aber den privaten Schlüssel K' geheim
    - Sender chiffriert mit öffentlichem Schlüssel K
    - Empfänger dechiffriert mit privatem Schlüssel

Symmetrische Verfahren (Beispiele)

- Data Encryption Standard (DES)
  - Blockchiffre* von 64-Bit-Datenblöcken mit (effektiv) 56-Bit-Schlüssel in insgesamt 19 Schritten
  - DES wird heute als nicht mehr sicher angesehen
  - TripleDES: 3-mal DES mit verschiedenen Schlüsseln
- Advanced Encryption Standard (AES)
  - Nachfolger von DES/3DES
  - auch als Rijndael** bekannt
  - Blockchiffre mit Blockgröße 128 Bit, Schlüssellänge 128, 192, 256 Bit
  - kann sehr effizient in Software und Hardware implementiert werden
  - wird zurzeit als sehr sicher eingestuft

* Die Datenmenge wird in Blöcke aufgeteilt, die dann der Chriffrierung zugeführt werden
**als Standard jedoch auf bestimmte Blockgrößen und Schlüssellängen gegenüber dem eigentlichen Rijndael-Algorithmus eingeschränkt

Asymmetrische Verfahren

- RSA-Algorithmus (Rivest, Shamir, Adleman 1978)
  - Aufbauend auf der Grundidee des asymmetrischen Verfahrens von Diffie und Hellman (1976)
- Verwendet so genannte „One-Way Functions“
  - Die Berechnung von x bei gegebenem y = f(x) ist praktisch nicht durchführbar, d.h von hoher Komplexität
- RSA-Algorithmus beruht auf der Schwierigkeit, große Zahlen in ihre Primfaktoren zu zerlegen
- Länge der Schlüssel, z.B. 512 Bits (relativ unsicher), 1024 Bits (zurzeit gute Sicherheit)
- Wesentlich langsamer als symmetrische Verfahren!

Asymmetrische Kryptoalgorithmen

- Basierend auf
  - Primzerlegung
    - r bekannt; bestimme p und q (prim) mit p*q=r
    - Einsatz in RSA
  - Problem des diskreten Logarithmus (DLP)
    - $a^k = b \bmod q$. Bestimme k (aus q, a, b; ganzzahlige Werte).
    - Einsatz in El-Gamal-Verfahren
  - Elliptische Kurven
    - Basiert auch auf DLP, aber auf spezieller Gruppe
    - Wird als wesentlich schwieriger als das DLP vermutet
    - Sehr kurze Schlüssellänge

RSA-Prinzip

- Hier nur das Prinzip (ohne Beweis)
  - Schlüsselgenerierung
    - Wähle zwei große Primzahlen p und q (typischerweise 1024 bit)
    - Berechne n = p*q und z = (p-1)*(q-1)
    - Wähle eine zu z teilerfremde Zahl d
    - Finde e mit e*d = 1 mod z
    - Der private Schlüssel ist das Paar (d,n), der öffentliche Schlüssel ist (e,n).
  - Vorbereitung
    - Zerlege die Nachricht in gleich große Blöcke P_i von k Bits (mit k als größtem Wert mit $2^k < n$)
  - Verschlüsselung und Entschlüsselung
    - Berechne $C_i = P_i^e \bmod n$. Die Folge C_i ist dann der verschlüsselte Text.
    - Berechne $P_i = C_i^d \bmod n$. Die Folge P_i ist dann der Klartext.
  - Man kann zeigen: $T^{ed} = T \bmod n$.

Kryptooperationen

- Signatur
  - Empfänger kann vorgegebene Identität des Senders verifizieren
  - Sender kann den Inhalt und die Unterschrift nicht abstreiten
  - Empfänger kann das Ganze nicht selbst erstellen
- Verschlüsselung
  - Nur der Empfänger kann die Daten mit seinem Schlüssel wieder lesbar machen
  - ersetzt nicht die Signatur!

Abdruck (Hash/Digest)

- Bildet eine Datenmenge (insb. Text) auf eine Zahl mit vorgegebener Länge ab: n = h(Text)
  - Wird der Text geändert, dann ändert sich auch der Abdruck
- Kollision
  - Liegt vor bei h(Text1) = h(Text2) mit verschiedenen Text1 und Text2
  - Dann kann man Text1 mit Text2 ersetzen, ohne dass sich der Abdruck ändert
- Optimal, wenn es nicht möglich ist, zu einem bestimmten Wert n einen Text T zu finden, sodass h(T) = n
  - Damit könnte man sonst eine Kollision erzwingen

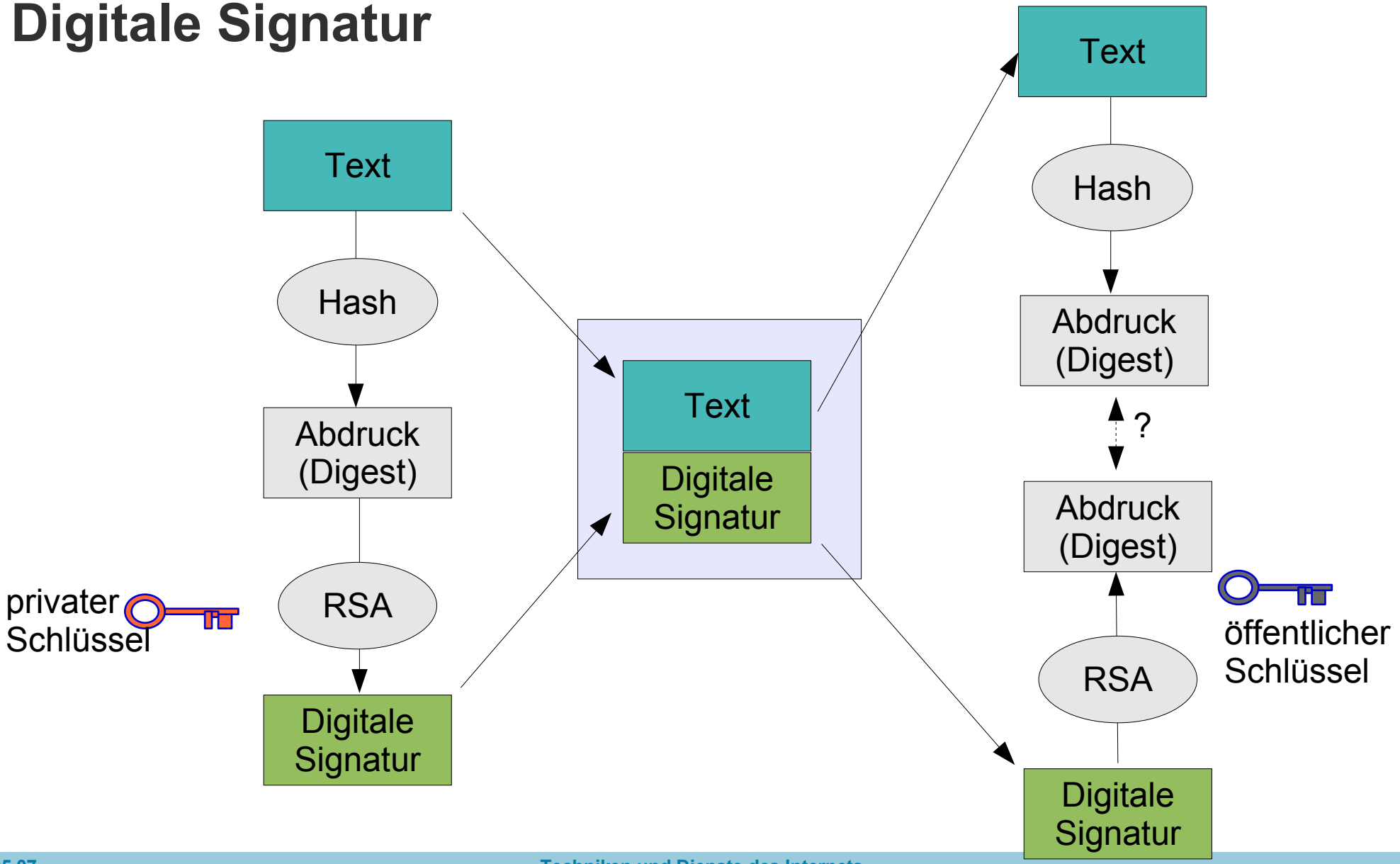
Hash

- Abdruck muss kryptografisch gesichert werden
  - Verschlüsselung mit privatem Schlüssel
  - Damit wird Nachweis erbracht, wer den Abdruck generiert hat
- Beispiele:
  - MD5 (Kollisionen mittlerweile gefunden)
  - SHA-1

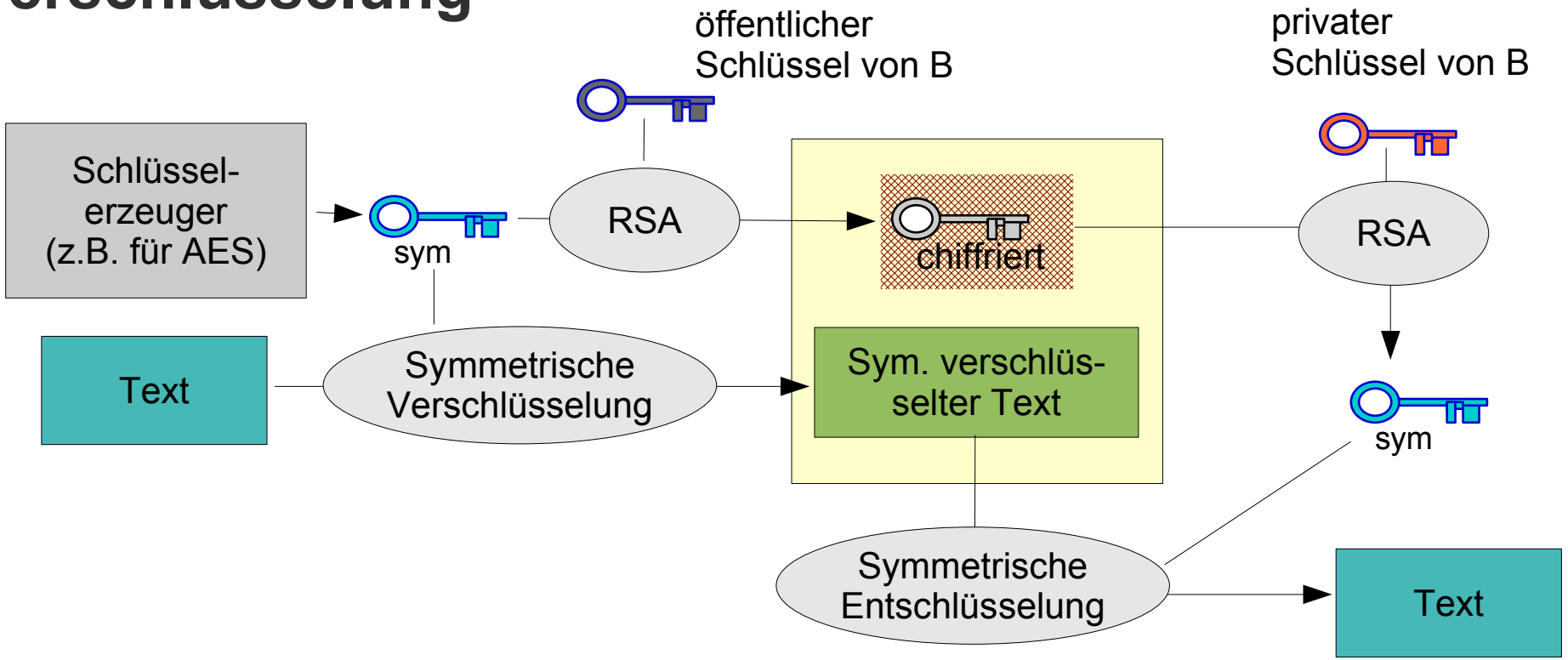
md5("Franz jagt im komplett verwahrlosten Taxi quer durch Bayern") =
a3cca2b2aa1e3b5b3b5aad99a8529074

md5("Frank jagt im komplett verwahrlosten Taxi quer durch Bayern") =
7e716d0e702df0505fc72e2b89467910

Digitale Signatur



Verschlüsselung



- Asymmetrische und symmetrische Verschlüsselung
  - nur der (relativ) kurze symmetrische Schlüssel wird asymmetrisch chiffriert
  - Nachricht symmetrisch verschlüsselt (Performanz!)

Sicherheit bei E-Mail

- Pretty Good Privacy (PGP)
- GPG (GNU Privacy Guard)
- S/MIME

PGP

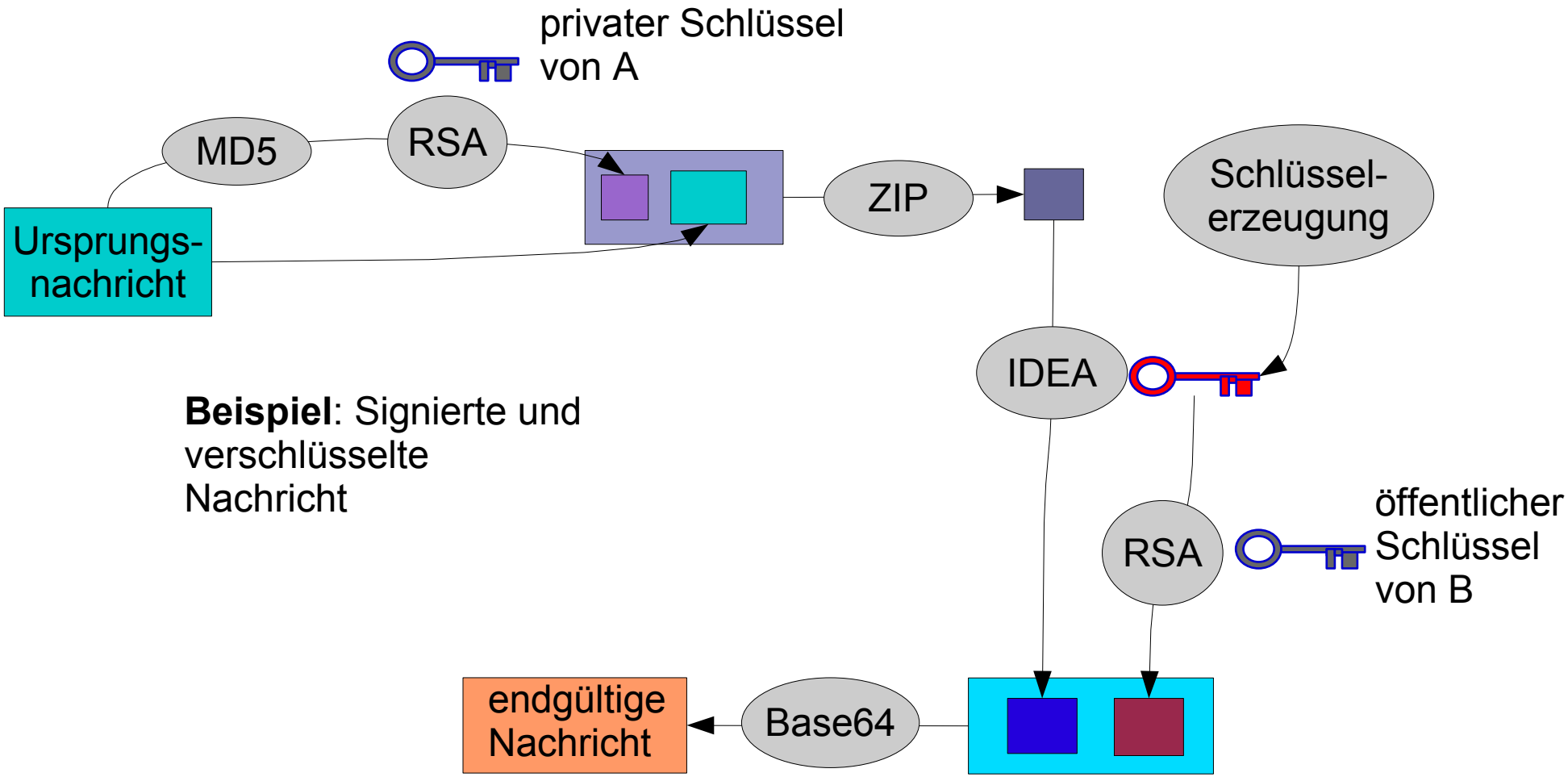
- Pretty Good Privacy (PGP) [Zimmermann, 1995]
- komplettes E-mail-Sicherheitspaket mit
  - Vertraulichkeit, Integrität, Authentifizierung
  - digitale Unterschrift
  - Kompression
- Teile des Quellcodes frei verfügbar (nicht-kommerziell)
- baut auf RSA, IDEA* und MD5 (message digest)
- Ab 1998 „OpenPGP“ auf dem Weg zum „Internet Standard“ (RFC 2440)

*symmetrisches Verschlüsselungsverfahren

GPG

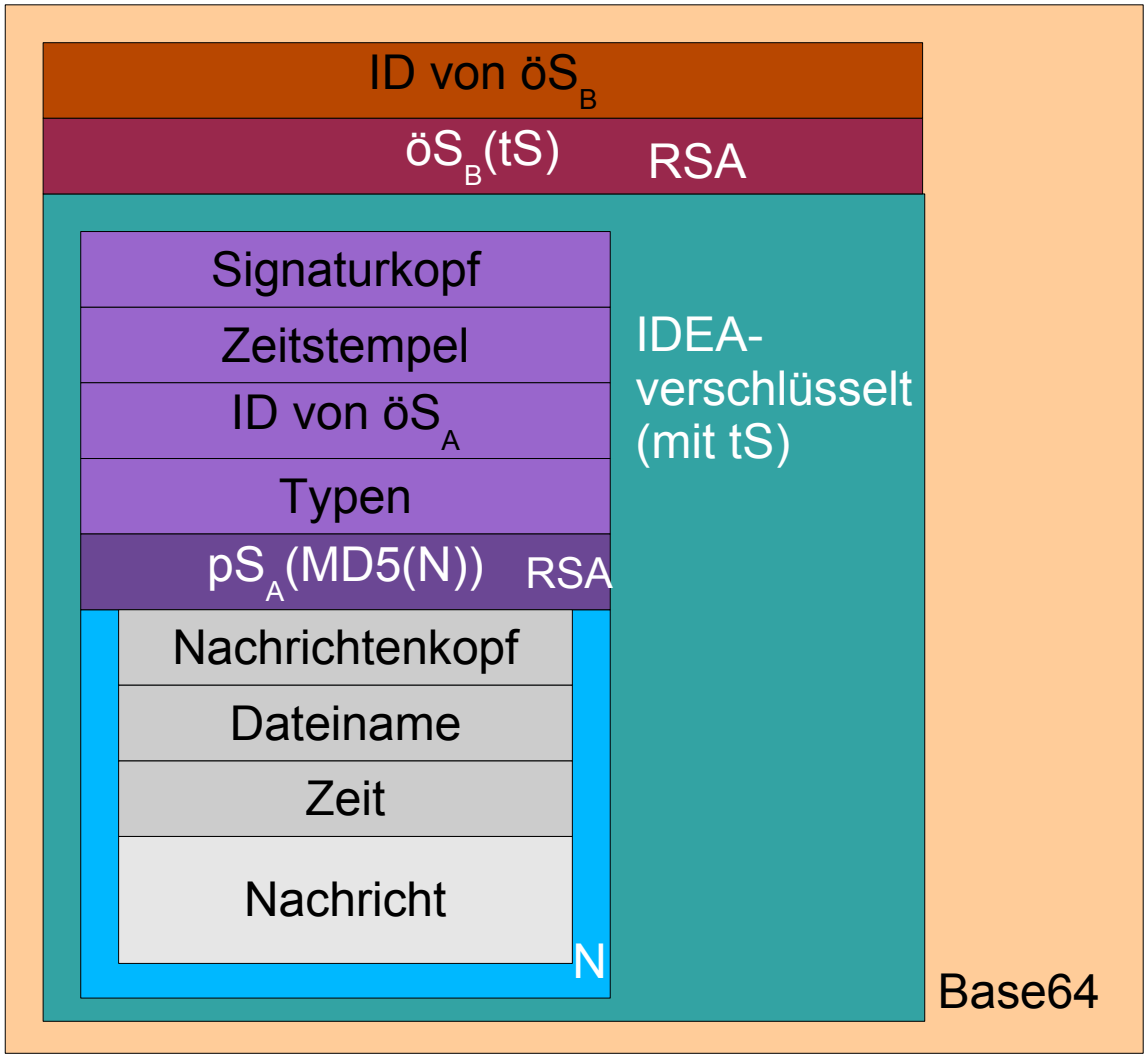
- GNU Privacy Guard
  - Implementiert OpenPGP gemäß RFC 2440
  - Open-Source-Software
- Entwickelt als Reaktion darauf, dass
  - Teile von PGP zeitweise nicht offengelegt wurden
  - Eigenschaften implementiert wurden, die als bedenklich galten (automatisches Versenden an einen Dritten)
  - PGP zeitweise nur kommerziell vertrieben wurde
  - PGP patentierte Algorithmen verwendete (z.B. IDEA)

Der Weg zur PGP-Nachricht



Beispiel: Signierte und verschlüsselte Nachricht

Aufbau einer PGP-Nachricht



tS=temporärer Schlüssel

A(B): wende A auf B an
öS$_X$=öff. Schlüssel von X
pS$_X$=priv. Schlüssel von X

Schlüsselmanagement

- Schlüsselverwaltung und Austausch oft schwächstes Glied
- PGP kennt zwei Schlüsselbunde für RSA-Schlüssel
  - Privater Schlüsselbund
    - einige persönliche private/öffentliche RSA-Schlüsselpaare
    - erlaubt das Wechseln der Schlüssel
    - Abspeicherung besonders geschützt durch Passphrase
  - öffentlicher Schlüsselbund
    - öffentliche Schlüssel möglicher Korrespondenten mit 64-Bit-ID des Schlüssels und einem Hinweis zur Vertrauensstufe

PGP-Schlüsselmanagement

- IDEA-Schlüssel für Verschlüsselung der Nachricht
  - IDEA wird wegen Performance und Sicherheit benutzt
  - jeder IDEA-Schlüssel wird nur einmal benutzt ("session key")
  - werden zufällig erzeugt unter Verwendung von Zufallseingabe, Tippgeschwindigkeit, Mausbewegung, ...
- Schlüsselverwaltung bei Mailprogrammen
  - Erweiterung für Thunderbird: Enigmail
    - ermöglicht einfachen Einsatz von GPG

Schlüsselverteilung

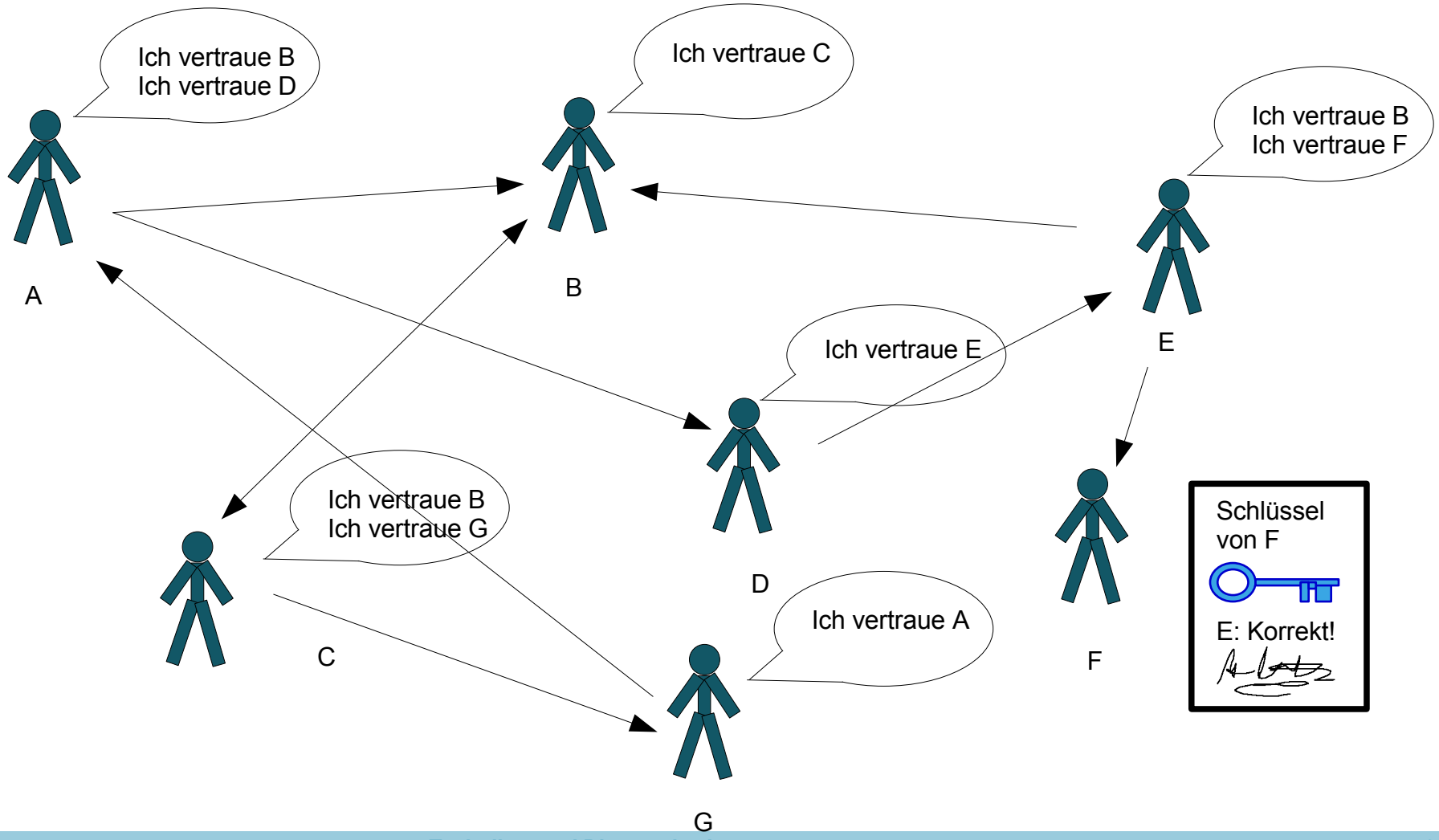
- Fragen
  - Ich möchte B eine verschlüsselte Nachricht schicken. Wie bekomme ich den öffentlichen Schlüssel von B?
  - Ich möchte B eine signierte Nachricht schicken. Wie bekommt B meinen öffentlichen Schlüssel?
  - Ich habe einen Schlüssel von B gefunden. Zumindest wird das so behauptet. Ist das dieser B?
  - Wie kann ich B von der Echtheit meines öffentlichen Schlüssels überzeugen?

Schlüsselverteilung

- RSA-Schlüssel werden verteilt z.B.
  - in E-Mails (unsicher wegen Fälschungsmöglichkeit!)
  - durch Anfrage an öffentlichen Schlüssel-Server (sicher durch Verschlüsselung der Verteilung)
  - durch Weitergabe auf Diskette, Papier usw. (sehr sicher (?))
  - durch Anfrage beim Kommunikationspartner oder einem Bekannten des Kommunikationspartners

    → Aufbau von „Vertrauensketten“ (Web of trust)

Web Of Trust


Ich vertraue B
Ich vertraue D
A
Ich vertraue C
B
Ich vertraue B
Ich vertraue F
E
Ich vertraue E
D
Ich vertraue B
Ich vertraue G
C
Ich vertraue A
G
F
Schlüssel
von F
E: Korrekt!

Privacy-Enhanced Mail

- Offizieller Internet-Standard [RFC 1421 – 1424]
- Ähnliche Zielsetzung wie PGP
- Aufwändiges, aber klar strukturiertes Schlüsselmanagement
  - Bescheinigung von Schlüsseln durch Zertifizierungsstellen
  - einfache Invalidierung von öffentlichen Schlüsseln
- Einige technische Unterschiede zu PGP
  - DES statt IDEA
  - Message Digest ist verpflichtend
  - keine Kompression
  - Mailinglisten werden unterstützt

Zertifikatsbasierte Systeme

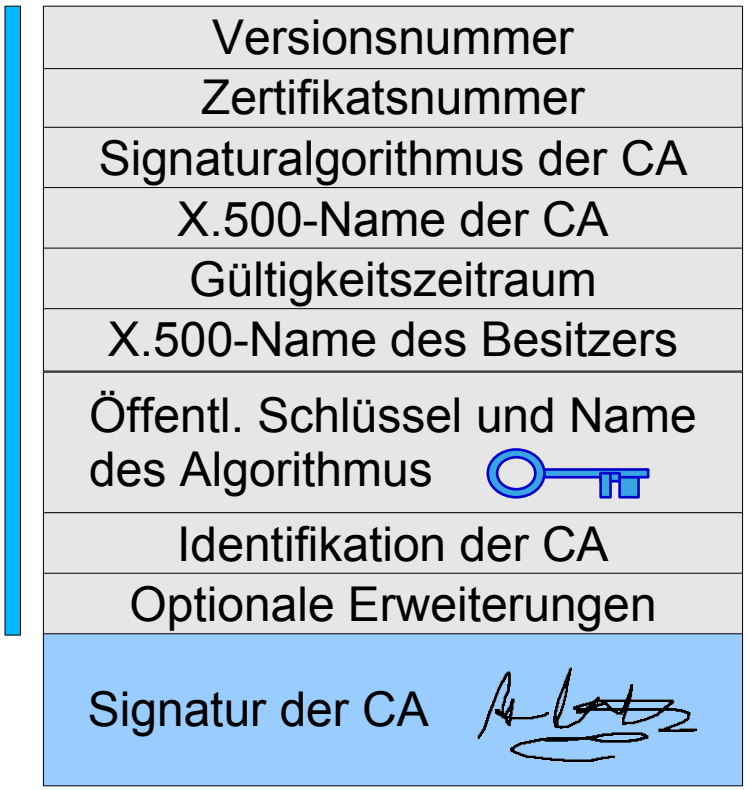
- Spezifiziert durch Public Key Cryptography Standards (PKCS)
  - von RSA Labs herausgegeben

PKCS#1	RSA Cryptography Standard	
PKCS#2		mit PKCS#1 vereinigt
PKCS#3	Diffie-Hellman-Schlüsselaustausch	
PKCS#4		mit PKCS#1 vereinigt
PKCS#5	Password-based Encryption	
PKCS#6		veraltet durch X.509v3
PKCS#7	Cryptographic Message Syntax	
PKCS#8	Private Key Information Syntax	
PKCS#9	Selected Attribute Types	
PKCS#10	Certification Request	
PKCS#11	Cryptographic Token Interface	
PKCS#12	Personal Information Exchange Syntax	
PKCS#13	Elliptic Curve Cryptography	noch in Entwicklung
PKCS#14	Pseudo-random Number Generation	noch in Entwicklung
PKCS#15	Cryptographic Token Information Format	veraltet

Zertifikate

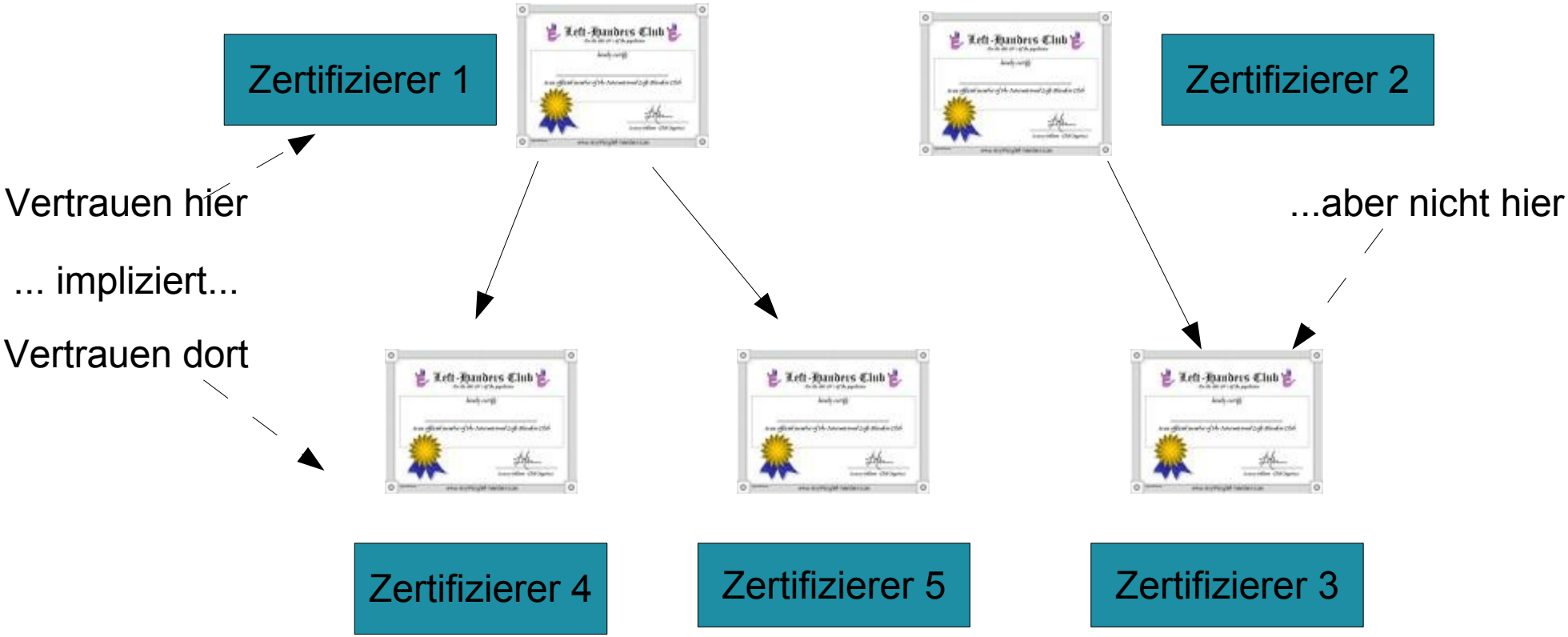
- Anforderungen
  - Zertifikat beglaubigt den
    - öffentlichen Schlüssel eines Akteurs
    - Beglaubigung durch Certification Authority (CA) (= „Internet-Notar“, Zertifizierungsinstanz, Trust Center)
  - Beteiligte vertrauen der Integrität der CA
  - Hierarchie von CAs

Aufbau eines Zertifikats nach X.509 v3



Vertrauenshierarchie

- „Infrastruktur für öffentliche Schlüssel“
  - Public Key Infrastructure (PKI)



Beispiel eines Zertifikats

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 04:26:fb:f3:00:00:00:00:00:10
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=DE, OU=Andreas Aurand, CN=CA FRS-LAB
        Validity
            Not Before: Feb 13 10:10:50 2001 GMT
            Not After : Feb 13 10:20:50 2002 GMT
        Subject: SN=7681045/unstructuredName=c2504.frs-lab.de
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:cc:bc:f5:0d:bb:3b:3a:d5:2f:9d:3b:ce:09:82:
                    15:7e:ae:5a:d2:19:d0:12:4c:6a:cf:c7:c3:e2:65:
                    cc:12:d2:ee:f2:5d:df:31:fc:4f:70:38:77:96:54:
                    37:a2:3b:dc:4e:df:7c:ea:ff:75:0c:93:43:7c:29:
                    24:8d:e7:9a:4b
                Exponent: 65537 (0x10001)
        Signature Algorithm: sha1WithRSAEncryption
            39:23:9d:21:5f:5e:90:23:e4:72:6f:a8:0c:af:b2:56:1c:bc:
            b6:28:aa:d4:a9:57:3a:52:86:3b:9f:7a:3f:c1:de:da:c0:24:
            45:3b:73:69:88:38:3f:9d:05:8e:37:67:a0:50:a7:33:44:a9:
            a9:3b:81:7b:4f:49:0e:7b:3d:49
```

Öffentlicher Schlüssel

Digitale Signatur des Zertifikats (mit dem privaten Schlüssel der CA erzeugt)

S/MIME

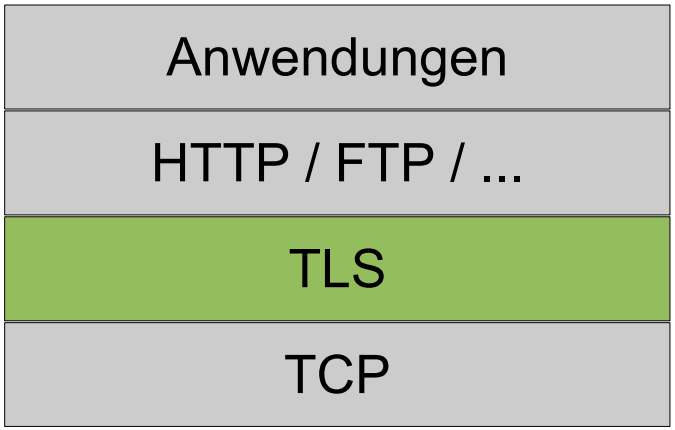
- Einsatz von PKCS für E-Mail
  - Kompatibel zu PEM
  - Anwendung von PKCS#7 (RFC 2315)
    - Übertragung von Daten, die einer kryptografischen Operation unterzogen wurden
    - Übertragung von Zertifikaten
  - PKCS#12: Speicherung des privaten Schlüssels
    - Dateiendung häufig „.p12“
  - S/MIME in aktuellen Mailprogrammen bereits integriert
- Zertifikate werden auch bei der Rechnerkommunikation eingesetzt
  - Transport Layer Security (TLS)

Transport Layer Security (TLS)

- Als SSL (Secure Socket Layer) von Netscape eingeführt
  - benutzt symmetrische (RC2/RC4) und asymmetrische Verschlüsselung (RSA)
  - Signatur mit MD5-Hash-Algorithmus
- standardisiert von IETF in 1996: TLS (Transport Layer Security)
- bietet
  - Authentifizierung für Klient und Server
  - Vertraulichkeit der Daten
  - Integrität der Daten

TLS

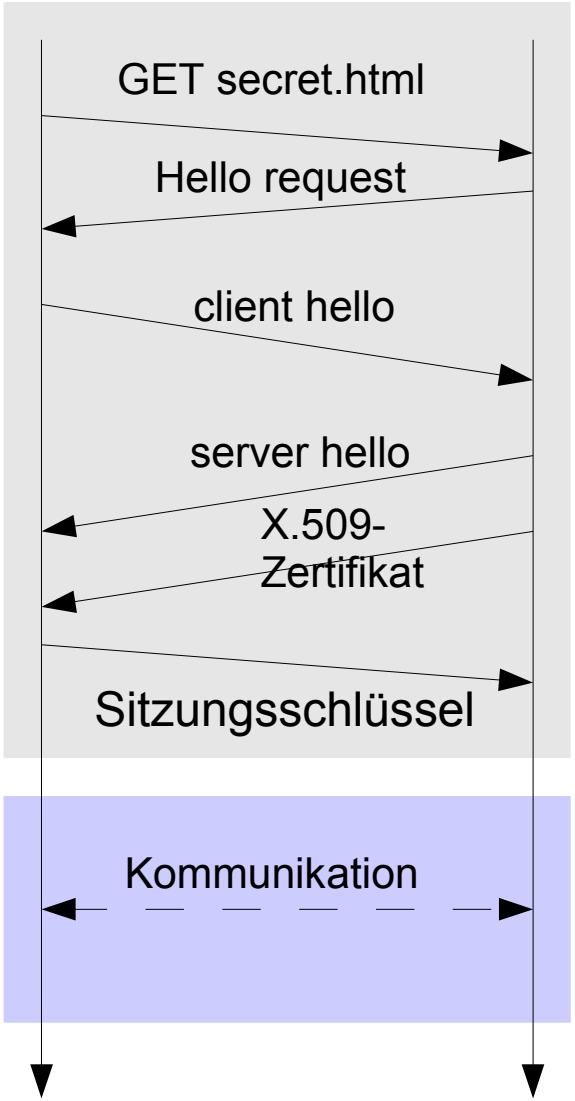
- Ziele
  - für viele Anwendungen geeignet
  - Anwendungsprogrammierer und Anwender von der Schlüsselverwaltung entlasten
  - für verbindungsorientiertes Transportprotokoll (z.B. TCP)
- Integration
  - TLS ergänzt die Transportschicht
  - erkennbar an „**https**://…“



TLS

- Asymmetrische Verschlüsselung, um Schlüssel für symmetrische Datenverschlüsselung auszuhandeln
- Algorithmen können bei der Verhandlung ausgewählt werden, auch Kompression
- Zertifikate nach X.509
- Phasen
  - Phase 1: gegenseitige Authentifizierung und Schlüsselaustausch
  - Phase 2: gesicherte Datenkommunikation
- Schlüssel werden im Cache gehalten

Ablauf von HTTPS (HTTP über TLS)



Dokument anfordern

Server antwortet; will gesicherte Verbindung aufbauen

Kryptoalgorithmen vorschlagen und Zufallswert (zc) senden

Algorithmus auswählen und Zufallswert (zs) senden

Zertifikat schicken

Sitzungsschlüssel aus zs, zc und einem „Pre-Master Key“ berechnen und senden

Kommunikation mit Sitzungsschlüssel gesichert

Protokollablauf

- Klient sendet dem Server eine „Hello“-Nachricht
  - eigene TLS-Version
  - eigene Verschlüsselungsarten
  - zufällige Bytes: zc
- Server antworter dem Klienten mit einer „Hello“-Nachricht:
  - eigene TLS-Version
  - eigene Verschlüsselungsarten
  - zufällige Bytes: zs

Protokollablauf

- Der Server sendet dem Klienten eine Zertifikatsnachricht:
  - Server-X.509-Zertifikat
    - inklusive dem passenden Schlüssel (gemäß Verschlüsselungsart)
- Der Server kann ein Klientenzertifikat anfordern (falls Klientenauthentifikation erwünscht ist)
- Der Server schickt ein „HelloDone“ an den Klienten und wartet
- Der Klient kann nun den Server authentifizieren.

Protokollablauf

- Der Klient erzeugt einen vorläufigen Schlüssel vs (46 Byte)
- Der Klient schickt dem Server
  - $öS_{Server}(vs)$
  - Falls sich der Klient authentifizieren muss:
    - Signatur(Kontrolldaten)
    - sein eigenes X.509-Zertifikat

A(x) bedeutet:
Wende A auf x an
öS=öff. Schlüssel
pS=priv. Schlüssel

- Der Klient erzeugt nun den geheimen Schlüssel aus dem Vorschlüssel, den er gesendet hat
  - S = PRF(vs, „master key“, zc + zs)

    (Pseudo Random Function, erzeugt einen Hashwert aus den Parametern)

Protokollablauf

- und der Server erzeugt denselben geheimen Schlüssel aus dem Vorschlüssel, den er erhalten hat.
  - kann er entschlüsseln durch $pS_{Server}(öS_{Server}(vs))=vs$
- Der Klient sendet eine „**ChangeCipherSpec**“-Nachricht:
  - ab sofort Verschlüsselung mit Schlüssel S
  - und schickt PRF(S, „client finished“, Verhandlungsdaten)
- Der Server schickt eine „**ChangeCipherSpec**“-Nachricht:
  - ab sofort Verschlüsselung mit Schlüssel S
  - und schickt PRF(S, „server finished“, Verhandlungsdaten)

nochmal zur Kontrolle: alle ausgetauschten Bytes (bieten sich an, denn sie sind beiden bekannt und spezifisch für diese Sitzung)

Secure HTTP (S-HTTP)

- Offizieller Standard der IETF (RFC 2660)
  - Vorgeschlagen von Enterprise Integration Technologies (EIT) 1994
- Protokoll der Anwendungsschicht
  - Kompatible Erweiterung von HTTP um Sicherheit
  - URL einer geschützten HTML-Seite beginnt mit shttp://
- Wesentliche Erweiterungen
  - Aushandlung der Mechanismen zwischen Klient und Server
    - Digitale Unterschrift, Authentifikation, Verschlüsselung
  - Bestimmung der Algorithmen, Schlüsselaustausch

S-HTTP ist praktisch nie zu bedeutsamer Verbreitung gelangt und sollte deshalb **nicht** verwendet werden *(Gefahr unerkannter Lücken)*.

Zusammenfassung

- Sicherheit in Datennetzen
  - Konzept gemäß der Bedrohungslage erarbeiten
  - Firewalls einsetzen, um unerwünschte Zugriffe von außen zu verhindern
  - Kryptografie einsetzen, um Daten zu schützen
- Kryptoverfahren
  - symmetrisch: schnell, aber Schlüsselaustausch problematisch
  - asymmetrisch: sicher, aber langsam; kann zum Schlüsselaustausch verwendet werden
  - Signaturen belegen Urheber und Ursprünglichkeit, Verschlüsselung verhindert Einsichtnahme Dritter
- Wichtigste Regel
  - Keine Kryptoverfahren selbst erfinden!