Kapitel 3: Anwendungsschicht

3.1 Netzanwendungen

3.2 Web und HTTP (HyperText Transfer Protocol)

3.3 DNS (Domain Name System)

3.4 Weitere Anwendungsprotokolle: Mail und FTP

Lernziel:

- Verständnis für Anforderungen von Anwendungen
- Entwicklung und Kerneigenschaften von HTTP als wichtigstem - mit Trend zu universellem – Anwendungsprotokoll
- Funktionsweise von DNS –einer global verteilten Datenbank

Anwendungen

- Web surfen inkl. soziale Netze
- Audio- und Video Streaming (live/on-demand)
- Sprach- und Videotelefonie, Audio- und Videokonferenz
- Dateiübertragung (z.B. Softwareinstallation, Updates, BackUp, FileSharing, …)
- Email
- Gaming (runden-basiert, Echtzeit)
- Cloud Computing (z.B. Google Docs)
- Remote Access, Remote Desktop
- SAP, Banking, Datenbankanwendungen, …
- IoT-Anwendungen
- … und viele, viele mehr

Anforderungen von Applikationen/Anwendungen?

Datenintegrität (data integrity)
(Toleranz gegen Datenverlust)

- einige Anwendungen können Datenverlust tolerieren (z.B. Audioübertragungen)
- andere Anwendungen benötigen einen absolut zuverlässigen Datentransfer (z.B. Dateitransfer)

Zeitanforderungen (timing)

- einige Anwendungen wie Internettelefonie oder Netzwerkspiele tolerieren nur eine sehr geringe Verzögerung

Sicherheit (security)

- Verschlüsselung
- Authentisierung
- Datenintegrität

Durchsatz/Bandbreite (throughput)

- einige Anwendungen (z.B. Multimedia-Streaming) brauchen eine Mindestbandbreite, um zu funktionieren
- andere Anwendungen verwenden einfach die verfügbare Bandbreite (bandbreitenelastische Anwendungen)

Anmerkung:

Datenintegrität, Durchsatz und Verzögerungen beziehen sich auf die Kommunikation von Nachrichten über „Sockets“ nicht auf „Pakete“; Paketverlust bedeutet beispielsweise nicht unbedingt Datenverlust.

Anwendungen und zugehörige Protokolle

Applikation Anwendung	Protokoll der Anwendungsschicht (Application Layer Protocol)	Transport-Protokoll
E-Mail	SMTP, POP3, IMAP (RFCs 5321, 1939, 3501)	TCP
Remote-Terminal	Telnet (RFC 854)	TCP
	SSH (mehrere RFCs)	TCP
Remote-Desktop	RDP (proprietär, Microsoft)	TCP
Web	HTTP (RFC 2616, RFCs 7230-7235)	TCP
	HTTP 3.0 (RFC 9114)	UDP (QUIC)
File-Transfer	FTP (RFC 959)	TCP
Multimedia Streaming	HTTP (z.B. YouTube)	TCP
	RTP (rückläufig, Live-Streaming)	UDP und TCP
Internet-Telefonie	SIP (3261), RTP (RFC 3550)	UDP
	Skype (proprietär)	UDP oder TCP
Video-Konferenzen (Webex)	SRTP (Secure Real Time Protocol) Fallback HTTPS	UDP (Fallback TCP) TCP

Transport-Dienste im Internet

TCP-Dienste:

- Zuverlässiger Transport von Daten zwischen Sender und Empfänger
  - alle Daten ohne Verlust in gleicher Reihenfolge
- Flusskontrolle: Sender überflutet Empfänger nicht mit Daten
- Überlastkontrolle: Drosseln des Senders bei Überlast im Netz
- Keinerlei Garantien für Qualität der Übertragung (Dauer, Durchsatz, Sicherheit)
- Verbindungsorientierung: Herstellen einer Verbindung zwischen Client und Server

UDP-Dienste:

- Unzuverlässiger Transport von Daten zwischen Sender und Empfänger
  - Verlust und Änderung der Reihenfolge möglich
- Keine Verbindungsorientierung, Zuverlässigkeit, Flusskontrolle, Überlastkontrolle, Garantien für Qualität der Übertragung

Frage:
Wozu soll das gut sein?
Warum gibt es UDP?

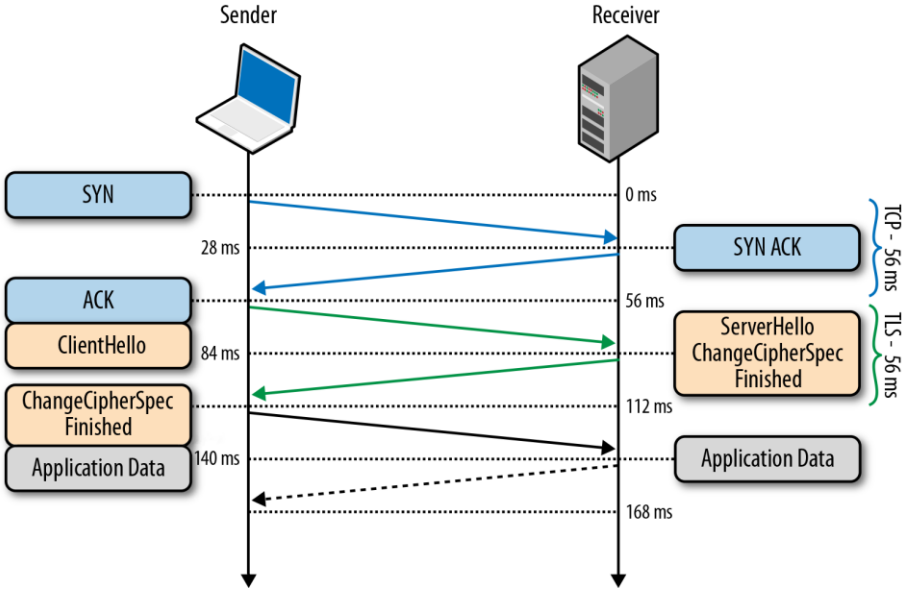
... Antwort auf “Warum UDP?”

- Vor wenigen Jahren wäre die Antwort gewesen
  - Internettelefonie
    - Sprachübertragung, vereinzelt Videokonferenz
  - einfache einmalige „ein-Paket“ Request-Response-Muster wie bei DNS
  - Multicast
  - verschwindet bei Streaming
- Heute
  - Videokonferenzen
  - zunehmender Anteil des HTTP Verkehrs
  - verschwindet bei DNS wegen Sicherheit (DNS über HTTP) und kommt wieder bei HTTP/3.0
  - Streaming nach wie vor über TCP ... oder über HTTP/3.0 und UDP

Sichere Datenübertragung mit TCP

TCP & UDP

- keine Verschlüsselung der Daten
- Passwörter werden im Internet so übertragen, wie sie in den Socket geschrieben werden:
  - wenn Klartext in den Socket, dann auch Klartext im Internet



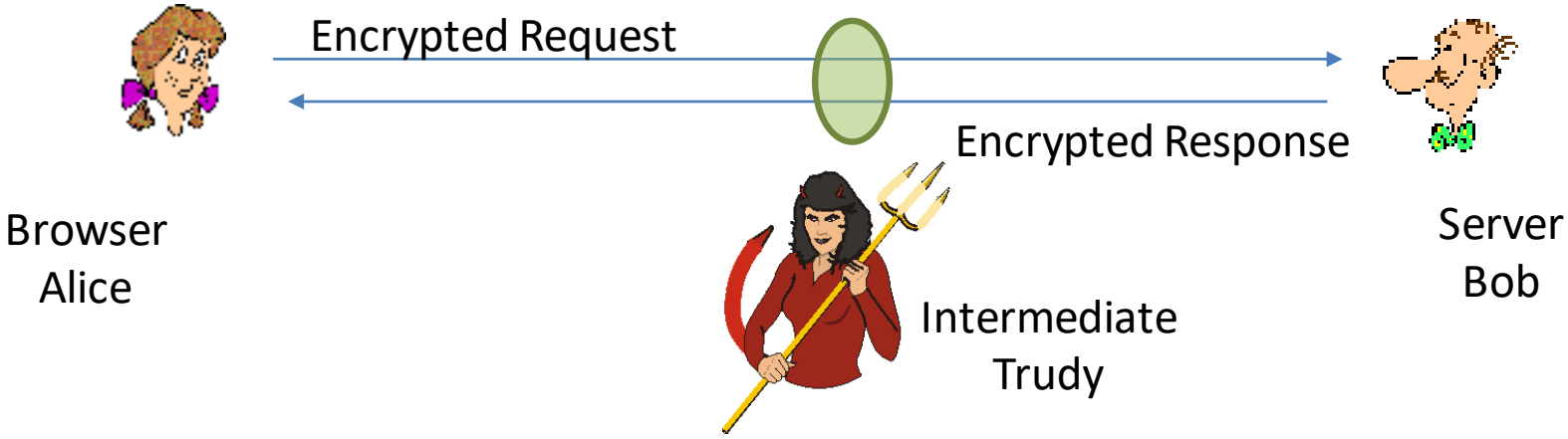
Quelle: O'Reilly: Browser Networking

SSL/TLS (Secure Sockets Layer/Transport Layer Security)

- bietet verschlüsselte TCP-Verbindungen
- Datenintegrität
- Authentisierung der End-Punkte
- Anwendungen nutzen SSL/TLS Bibliotheken zum Zugriff auf TCP
- SSL/TLS Socket API
  - Klartext im Socket, verschlüsselt im Internet
- Sichere Varianten der Protokolle beruhen auf SSL/TLS
  - https, ftps, imaps, smtps, …
  - http/2.0, http/3.0

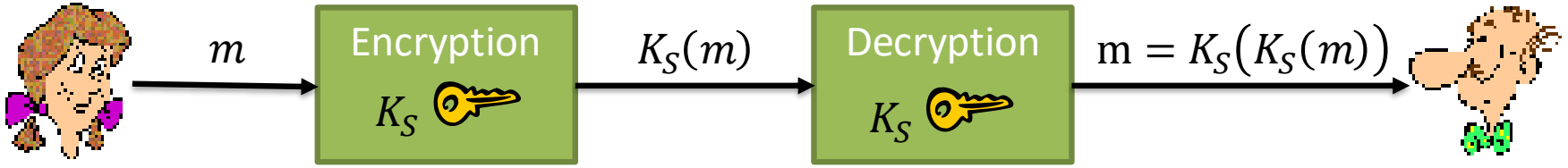
Ende-zu-Ende-Verschlüsselung mit TLS

- Problem:
  - Browser Alice möchte von Web-Server Bob eine Web-Seite laden
  - Intermediate Trudy kann den Netzwerkverkehr mitlesen und auch manipulieren
  - Alice und Bob müssen also die Nachrichten verschlüsselt austauschen
  - Problem: wie können Alice und Bob sich auf einen gemeinsamen Schlüssel einigen, ohne dass Trudy den Schlüssel kennt
  - und wie kann Alice sicherstellen, dass der Schlüssel von Bob und nicht von Alice kommt (Man-In-the-Middle-Attack)



Symmetrische und Asymmetrische Verschlüsselung

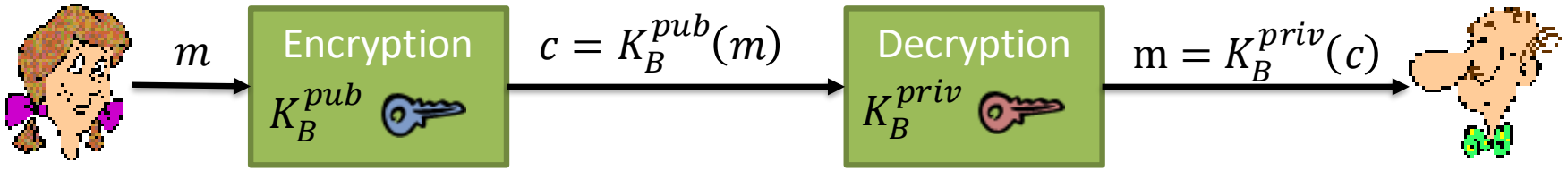
- Symmetrische Verschlüsselung:
  - gleicher Schlüssel K_S zum Ver- und Entschlüsseln einer Nachricht m



  - bekannte Verfahren: DES (Data Encryption Standard), AES (Advanced Encryption Standard), Camellia
  - längere Schlüssel bieten größere Sicherheit
    - 256 Bit-AES gilt als sicher
  - symmetrische Verschlüsselung ist weniger rechenintensiv als asymmetrische Verschlüsselung und wird in TLS zur eigentlichen Verschlüsselung der Nachrichten eingesetzt
  - Problem: beide Seiten benötigen den gleichen Schlüssel

Symmetrische und Asymmetrische Verschlüsselung

- Asymmetrische Verschlüsselung (Public Key Verfahren):
  - verwendet ein Paar von Schlüsseln, dem Public Key zum Verschlüsseln einer Nachricht und dem Private Key zum Entschlüsseln der Nachricht
  - Alice verschlüsselt die Nachricht mit Bob's öffentlich verfügbarem Public Key K_B^{pub} und Bob entschlüsselt die Nachricht mit dem geheimen nur ihm bekannten Private Key K_B^{priv}



  - bekannte Verfahren: RSA (Rivest, Shamar, Adleman), Diffie-Hellman, elliptische Kurven
  - asymmetrische Verschlüsselung ist sehr rechenintensiv und wird in TLS zum initialen Schlüsselaustausch verwendet

Digitale Signatur

- Alice signiert eine Nachricht mit ihrem privaten Schlüssel K_A^{priv}, in dem Sie einen Hashwert der Nachricht codiert
- Bob verifiziert den codierten Hashwert mit Alice öffentlichem Schlüssel K_A^{pub}

m

Hash function

$H(m)$

Encryption K_A^{priv}

$K_A^{priv}(H(m))$

Signieren

$m, K_A^{priv}(H(m))$

"Splitten"

$K_A^{priv}(H(m))$

Decryption K_A^{pub}

$K_A^{pub}\left(K_A^{priv}(H(m))\right)$

Vergleich

$H(m)$

Hash function

m

Digitale Signatur

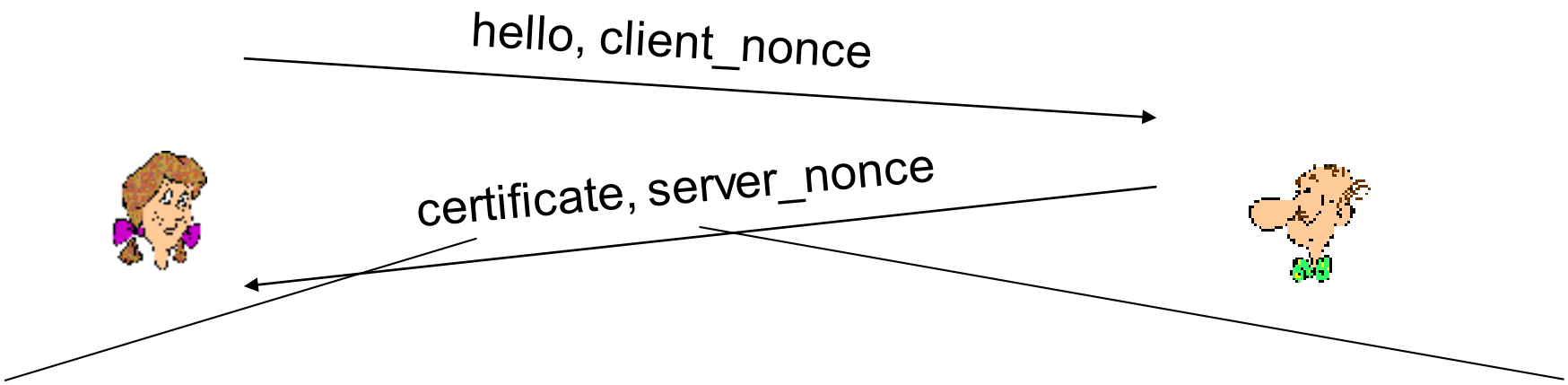
- Digitale Signaturen beruhen ebenfalls auf einem Paar aus privatem und öffentlichem Schlüssel
- Alice signiert eine Nachricht m, indem sie einen Hashwert $H(m)$ der Nachricht erzeugt, diesen mit ihrem privaten Schlüssel K_A^{priv} verschlüsselt und den verschlüsselten Hashwert $K_A^{priv}\big(H(m)\big)$ zusammen mit der Nachricht überträgt
- Bob überprüft die Signatur, indem er den empfangenen verschlüsselten Hashwert mit dem öffentlichen Schlüssel K_A^{pub} von Alice entschlüsselt und das Ergebnis mit dem Hashwert der Nachricht vergleicht. Wenn beide gleich sind, kommt die Nachricht sicher von Alice.
- Die Hash-Funktion dient dazu, den Rechenaufwand und die übertragene Datenmenge zu reduzieren
  - Hash-Funktionen: MD5, SHA-1

Mathematisches Prinzip (RSA)

- Verfahren basiert auf zwei große Primzahlen p und q mit $n = p \cdot q, z = (p-1) \cdot (q-1)$
- Private und Public Key ergeben sich aus zwei Zahlen e und d
  - von denen e teilerfremd zu z ist
  - und $e \cdot d - 1$ durch z teilbar ist, d.h. $e \cdot d \; mod \; z = 1$
- Verschlüsselung und Entschlüsselung
  - Verschlüsselung: $m^e \; mod \; n = c$
  - Entschlüsselung: $c^d \; mod \; n = (m^e \; mod \; n)^d \; mod \; n = m^{e \cdot d} \; mod \; n = m$
    - mathematischer Kern: $m^{e \cdot d} \; mod \; n = m^{\overbrace{(e \cdot d \; mod \; z)}^{=1}} mod \; n = m$
- Insbesondere gilt somit, dass auch eine Verschlüsselung mit dem Private Key mit dem Public Key entschlüsselt werden kann, was die Grundlage der Digitalen Signatur ist

$$K_B^{priv}\left(K_B^{pub}(m)\right) = m = K_B^{pub}\left(K_B^{private}(m)\right)$$

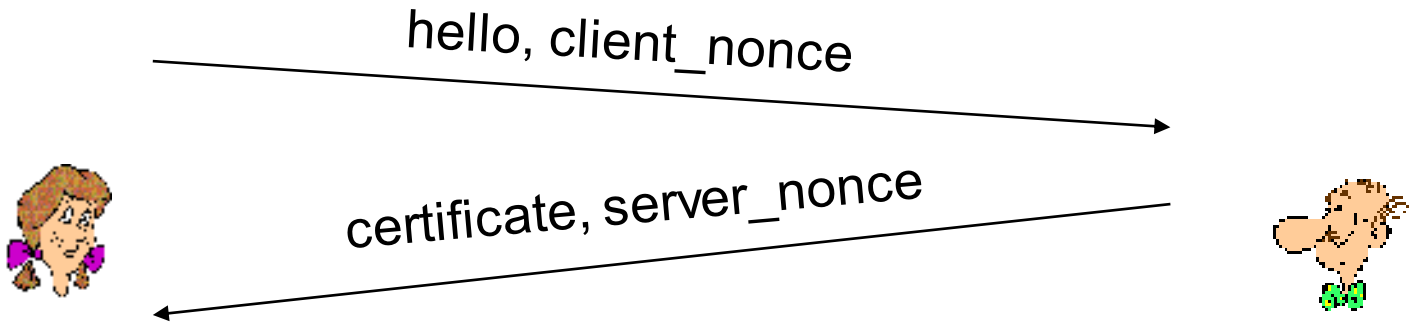
Prinzip TLS



Zertifikat:

- enthält Public Key des Kommunikationspartners (Person, Rechner, Institution) zusammen mit Informationen, die diesen eindeutig identifizieren, und Daten, die den Rahmen des Zertifikats festlegen
  - URL
  - Namen des "Subjects" (Kommunikationspartner)
  - Namen des "Issuers" (Zertifizierungsstelle)
  - Gültigkeit
- wird mit der digitalen Signatur einer Zertifizierungsstelle versehen
  - zur Erinnerung: Hash der Daten inkl. Public Key wird mit Private Key der Zertifizierungsstelle verschlüsselt
  - populäre Zertifizierungsstellen: Sectigo, DigiCert

Prinzip TLS



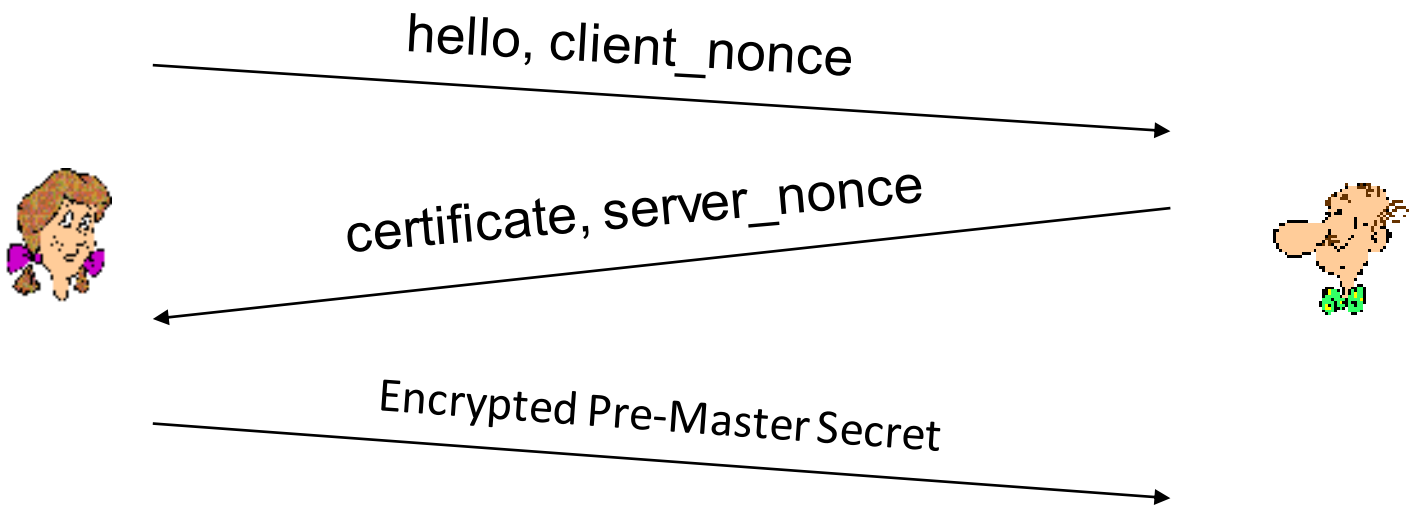
Überprüfen des Zertifikats im Browser:

- Browser enthält Public Keys der wichtigsten Zertifizierungsstellen
- Browser überprüft Signatur des Zertifikats anhand des Public Keys der Ausstellungsstelle
- Browser überprüft Gültigkeit der Daten im Zertifikat, z.B. URL und Ablaufdatum
- stimmt etwas nicht, gibt es eine Fehlermeldung
  - hier sollte vor allem auf falsche URLs geachtet werden

Sicherheit:

- Zertifikat vermeidet Man-In-The-Middle Attack
- Trudy kann Alice nicht ihren Public Key anstelle von Bob's unterjubeln
- Zertifikat bindet Bob's Public Key an seine URL
- Problem: Trudy kann gültiges Zertifikat für ähnliche URL senden

Prinzip TLS

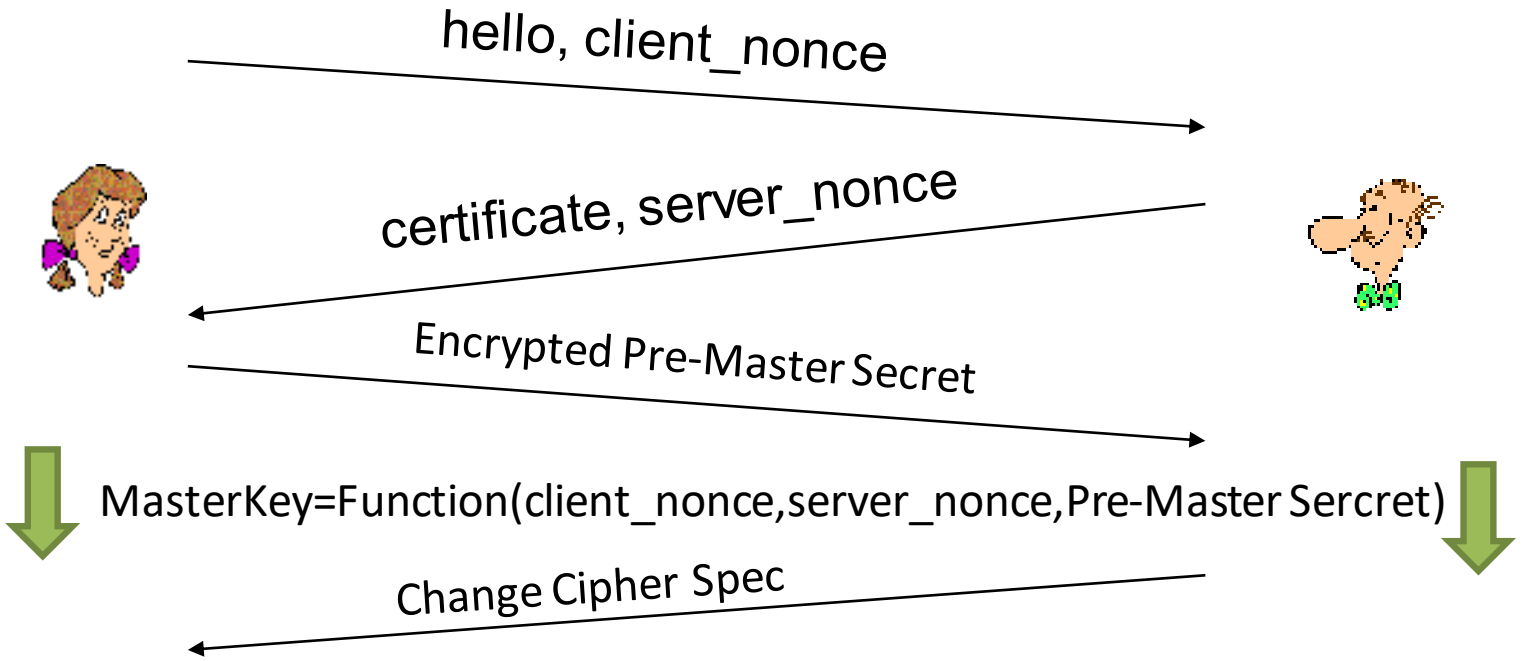


Pre-Master Secret

- Alice erstellt ein "Pre-Master Secret"
  - Diffie-Hellman-Parameter
  - RSA verschlüsseltes "Geheimnis"
- sendet es an Bob, verschlüsselt mit Bob‘s Public Key aus dem Zertifikat

Alice und Bob haben jetzt ein erstes gemeinsames Geheimnis. Trudy kennt das Geheimnis nicht, da sie Bob‘s Private Key zum Entschlüsseln nicht kennt

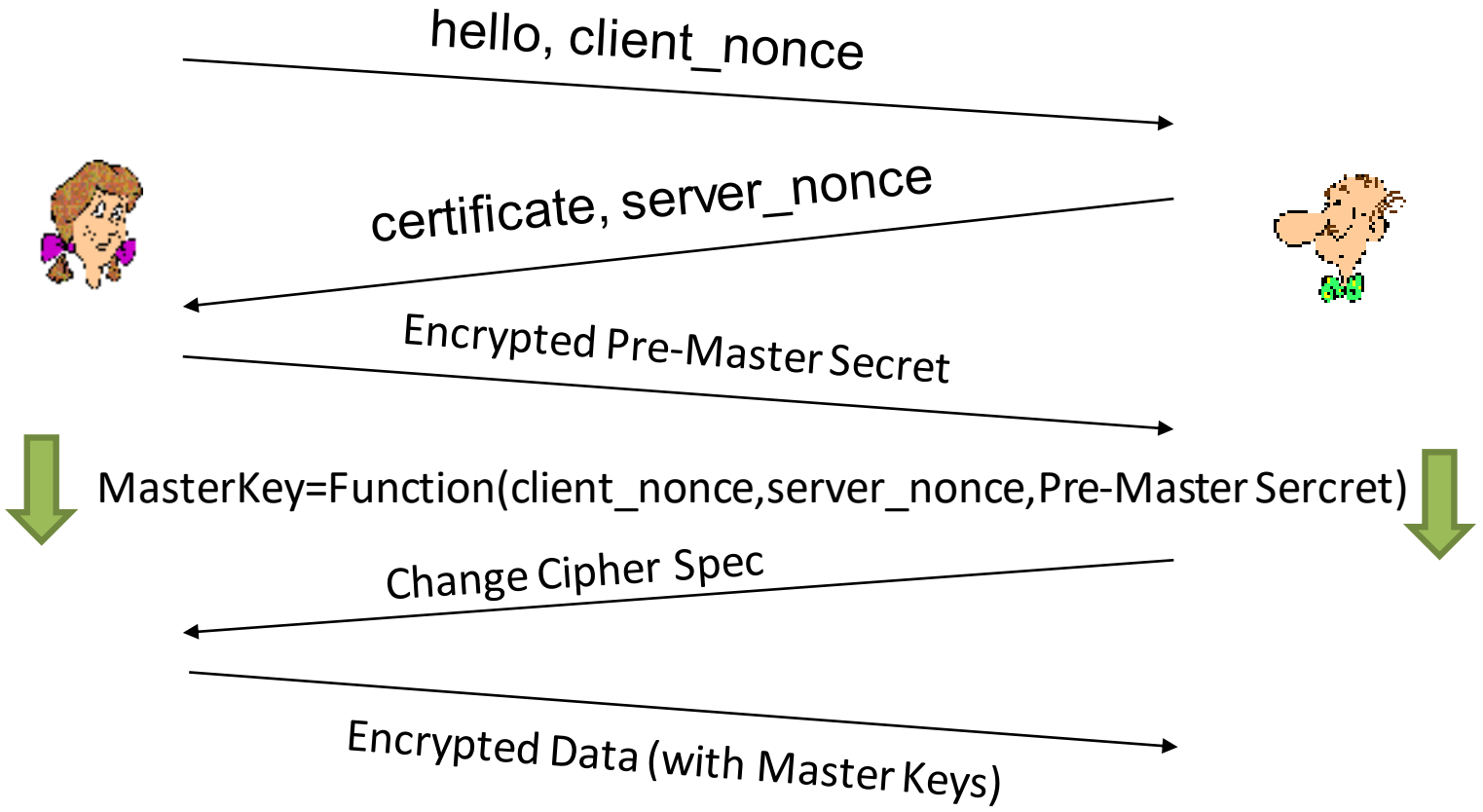
Prinzip TLS



Alice und Bob berechnen nun einen gemeinsamen Schlüssel, den MasterKey. Grundlage für die Berechnung sind das Pre-Master Secret sowie die beiden Nonces. Das sind zwei Zufallszahlen, die eine Wiederholung des gleichen Master Keys und somit eine Replay-Attacke verhindern.

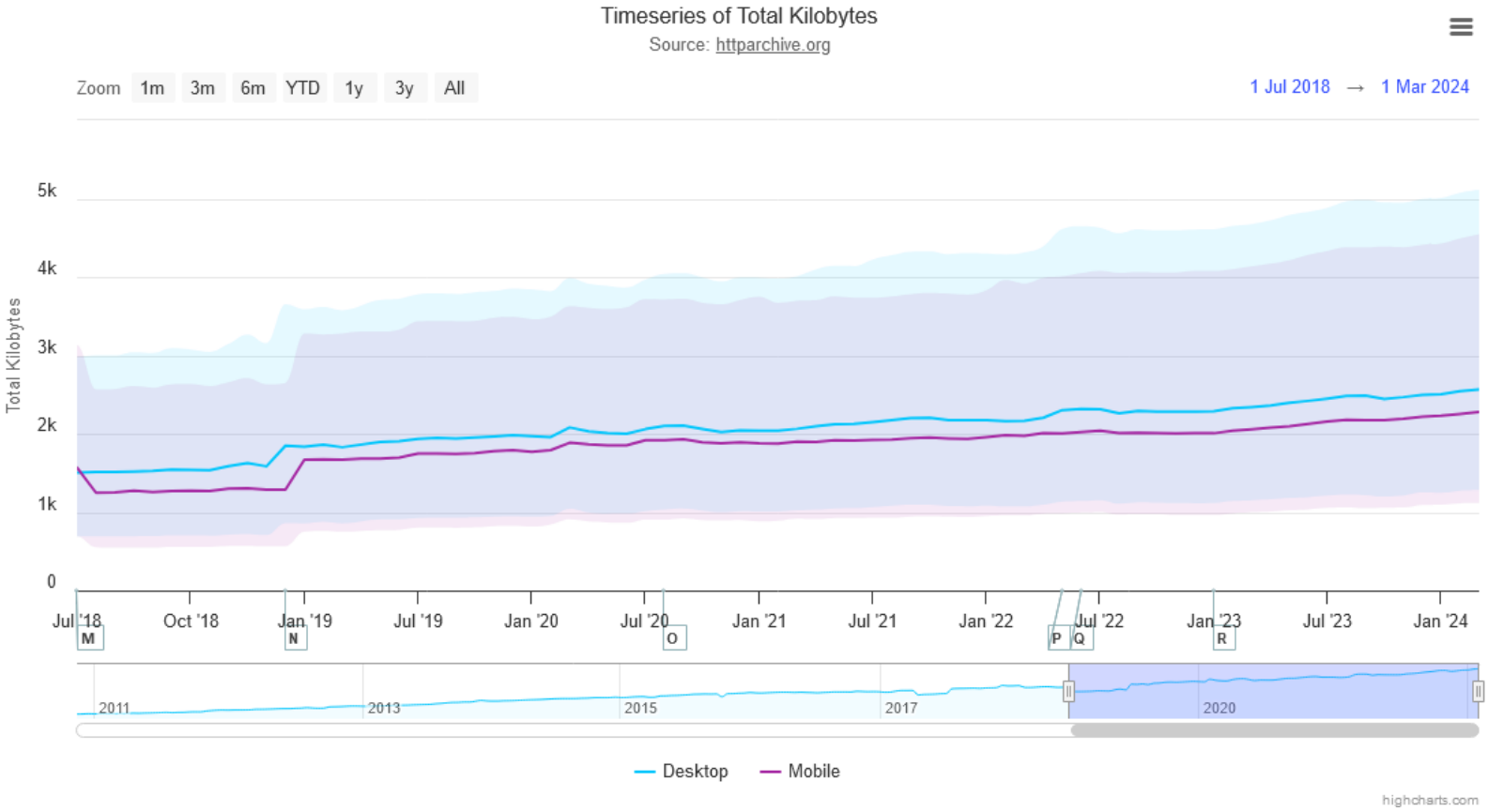
Eigentlich wird nicht nur ein Master Key berechnet sondern vier Master Keys, jeweils einen für die Integritätsprüfung und die Verschlüsselung auf Client- und Serverseite.

Prinzip TLS



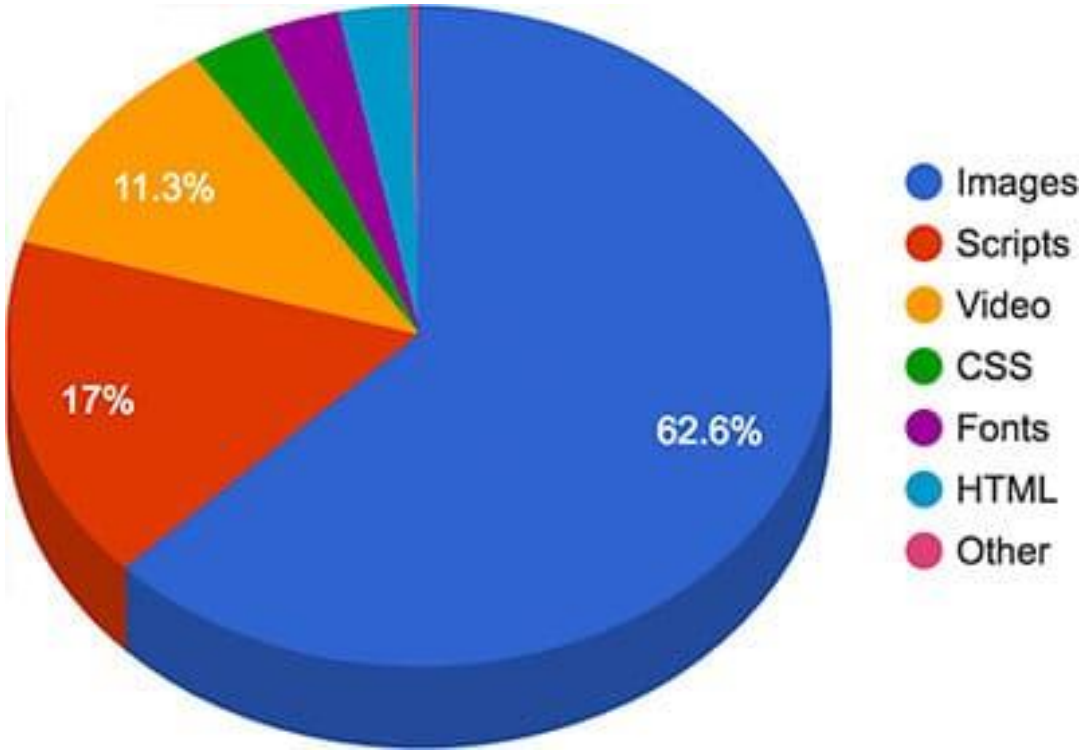
Mit der Nachricht Change Cipher Spec wird der Beginn der Übertragung von verschlüsselten Daten initialisiert.

Größe einer Web-Seite



- Median der Seitengröße:
  - 2021: Desktop: 1828 kB, Mobil: 1669 kB
  - 2024: Desktop: 2566 kB, Mobil: 2277 kB
- Median der Anzahl Requests/Objekte pro Seite:
  - 2021: Desktop: 75, Mobile: 69 (konstant)
  - 2024: Desktop: 73, Mobile: 68 (konstant/rückläufig)

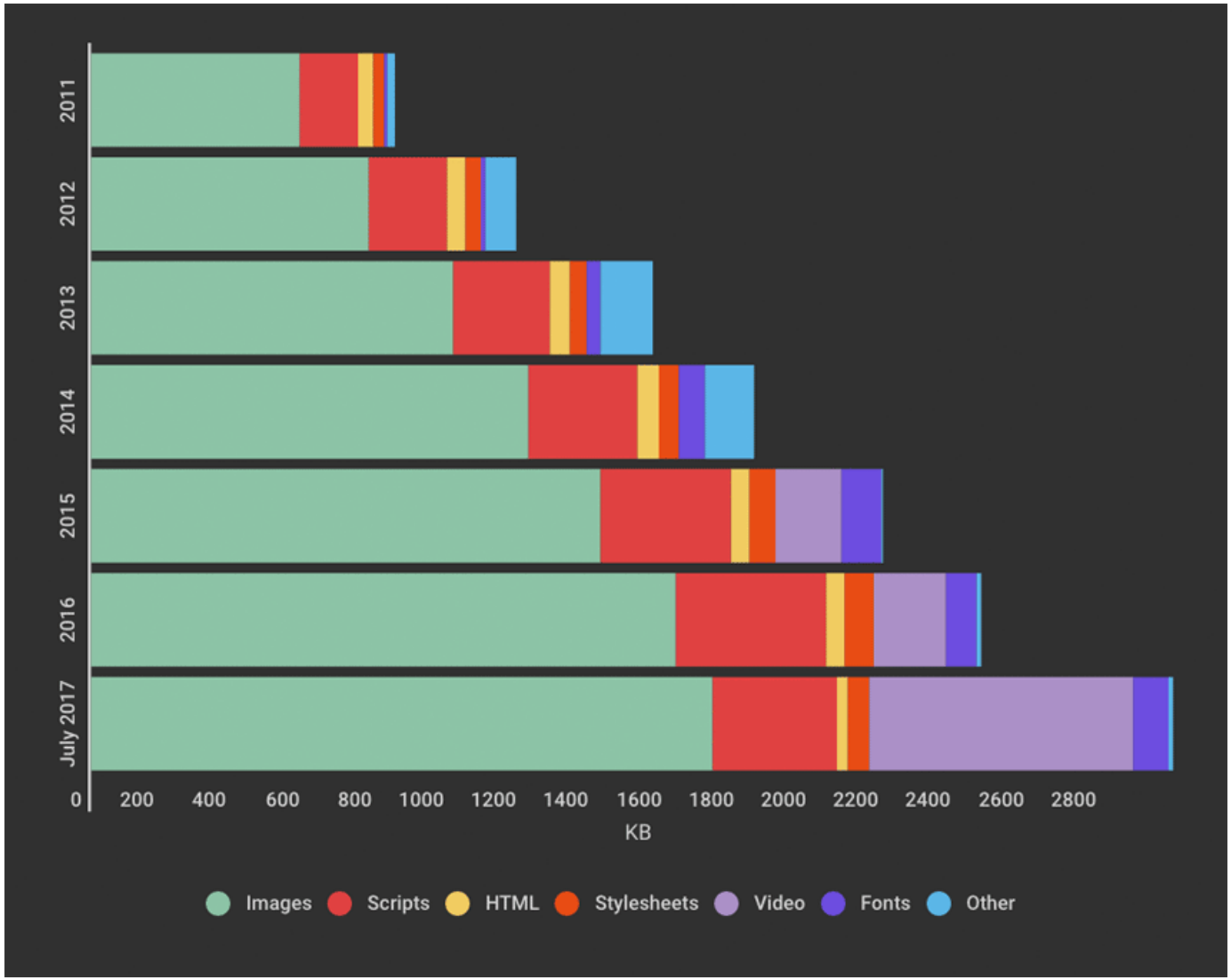
Aufbau einer Web-Seite: Die Objekte



Quelle: developers.google.com

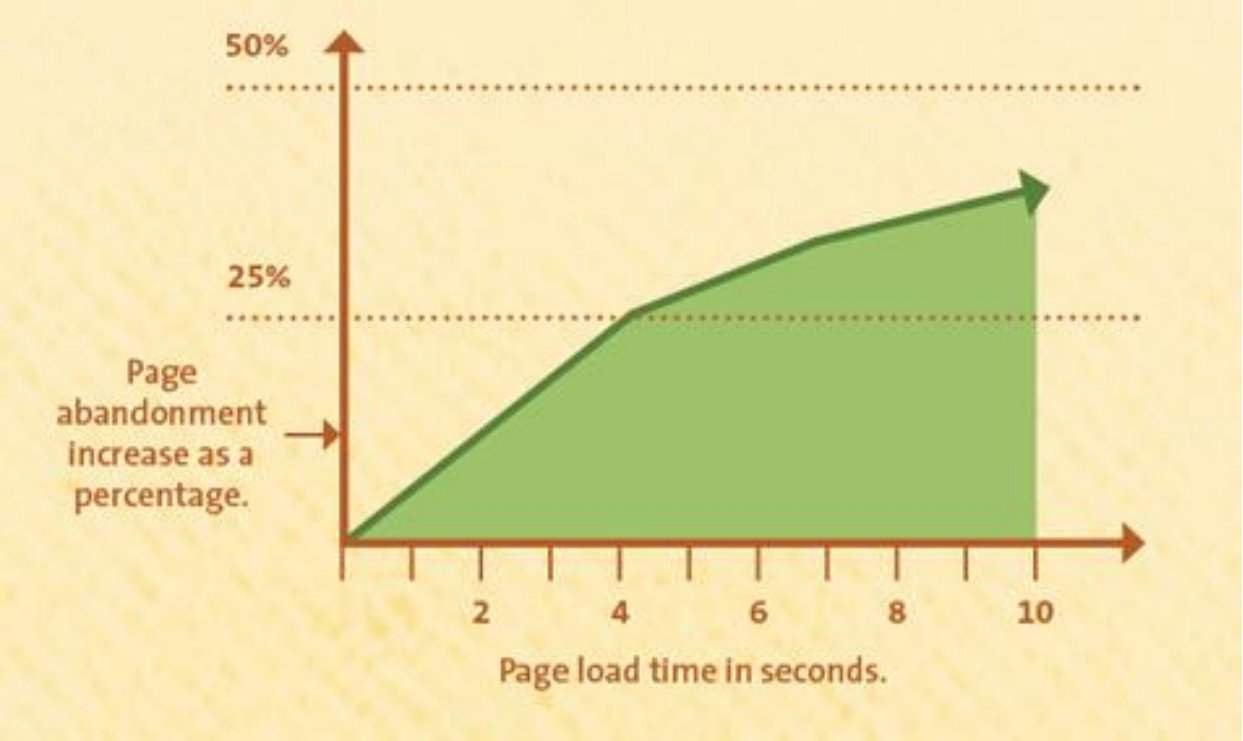
- Bilder machen ca. 60% des Volumens einer Web-Seite aus
- HTML-Code liegt bei unter 5 Prozent des Volumens

Wachstum: Images, Scripts und Videos



Quelle: assets.speedcurve.com

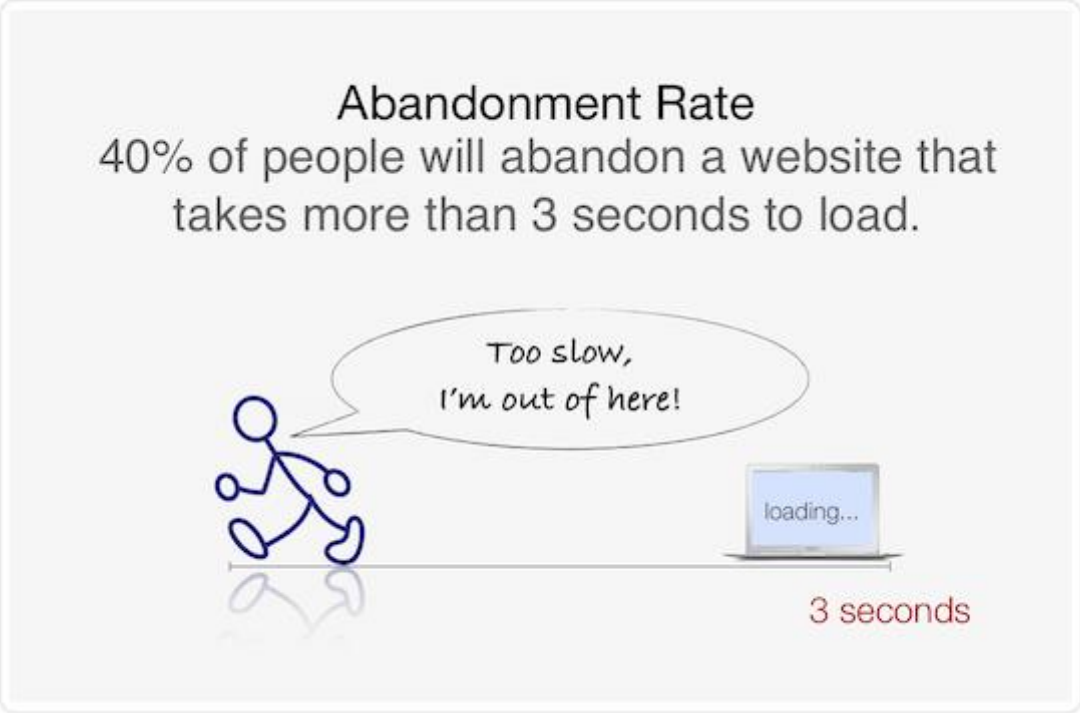
Ladezeiten



Quelle: kiss-metrics.com

- Abbruchraten für Web-Sites wachsen mit der Downloadzeit
- Einer von vier Kunden bricht nach 4s ab
- Web-Seite sollte in wenigen Sekunden (Akamai: 2s) geladen sein

Ladezeiten



Quelle: https://www.relentlesstechnology.com/insights/page-speed.html

- Abbruchraten für Web-Sites wachsen mit der Downloadzeit
- Einer von vier Kunden bricht nach 4s ab
- Web-Seite sollte in wenigen Sekunden (Akamai: 2s) geladen sein

Web-Surfen

- Welche Bandreite wird zum Web-Surfen benötigt?
  - Spitzendatenrate: 2 MBytes pro 2s ➔ 8 Mbps
  - Durchschnittliche Rate:
    - durchschnittliche Verweildauer auf einer Seite:
      - 30s ➔ 2 MBytes pro 32s ➔ 500 kbps
      - 62s ➔ 2 MBytes pro 64s ➔ 250 kbps
- Welche Bandbreite wird für 1000 aktive Web-Surfer benötigt bei 30s Verweildauer pro Seite?
  - Minimal: 1000 x 500 kbps -> 500 Mbps
    - Auslastung: 100%
  - Optimal: 1000 x 8 Mbps -> 8 Gbps
    - Auslastung: 500Mbps/8 Gbps➔1/16=6,25%
  - Realistisch (25-50% Auslastung): 1-2Gbps

Bitraten-Raten und Anforderungen

- Audio-Streaming?
  - oft Datenraten von 64-160 kbps, auch 32kbps und 320kbps
  - relativ konstante Bandbreite, meist geringerer Spielzeitpuffer als bei Video-Streaming
- Video-Streaming?
  - Datenraten von 500 kbps – 5 Mbps, 25 Mbps für Ultra HD
  - gute mittlere Bandbreite, Schwankungen können toleriert werden, hohe Bandbreite am Anfang
- Sprach-Telefonie?
  - Datenraten von 30kbps-100kbps (Skype empfiehlt 100kbps)
  - konstante Delays von unter 180 ms
- Video-Konferenz?
  - ab 300 kbps, 500 kbps bis 1,5 Mbps für gute Qualität
  - Webex: Standard (0.5 Mbps), High Quality (1,0 Mbps Receive, 1,5 Mbps Send), High Definition Video (2,5 Mbps Receive, 3,0 Mbps Send)

Webex Meetings - Bandbreitenverbrauch

Table 4. Webex Meetings Bandwidth per Resolution Table

Layer	Bandwidth Range
90p active thumbnail (each)	~60-100 kb/s
180p main video	125-200 kb/s
360p main video	470-640 kb/s
720p main video	900k-1.5 mb/s
Content sharing (sharpness, 1080p/5)	120k – 1.3 mb/s
Content sharing (motion, 720p/30)	900k – 2.5 mb/s

Cisco (2021): Bandwidth Planning in your Cisco Webex Meetings Environment White Paper
Quelle: https://www.cisco.com/c/en/us/products/collateral/conferencing/webex-meetings/white_paper_c11-691351.html

Übersicht zu Anwendungen und Anforderungen

Delay-Anforderungen	Bandbreiten-Anforderung: keine/gering	mittel	hoch: durchschnittliche Bandbreite	hoch: konstante Bandbreite
keine/gering	Datentransfer im Hintergrund (Software Upgrades, Email, Backup)	App-Installation, Email	Video-Streaming (on-demand)	
niedrig	Chat	Web-Surfen, runden-basierte Spiele, Audio-Streaming		Live-Streaming (kleiner Puffer)
hoch	Online-Trading, Alarme von Sensoren	Cloud-Computing, Echtzeit-Spiele, Sprache		Video-Telefonie/-konferenz