TLS-Verschlüsselung bei Qmail/Spamcontrol

Mailserver Konferenz, Berlin 2014

Erwin Hoffmann - FEHCom

FEH@FEHCOM.DE

Zu meiner Person

- Qmail Anwender und Supporter seit 1998
  - Support und Konfiguration umfangreicher Qmail-Sites
  - FreeBSD Junki mit Qmail-Ports
- Professor für IT Security und IT Governance an der
  - *Provadis* Hochschule in Frankfurt/Main und vorher an der
  - Fachhochschule Frankfurt/Main
- Autor einiger Bücher zu Rechnernetzen
  - *Technik der IP-Netze*

Überblick

- Qmail, seine Architektur und Weiterentwicklung
- Überlegungen zu Transport Layer Security bei Qmail
- Server-Daemons (**qmail-smtpd/qmail-pop3d**) mit TLS-Verschlüsselung
- **qmail-remote** mit TLS-Verschlüsselung
- Fazit und Ausblick für Qmail

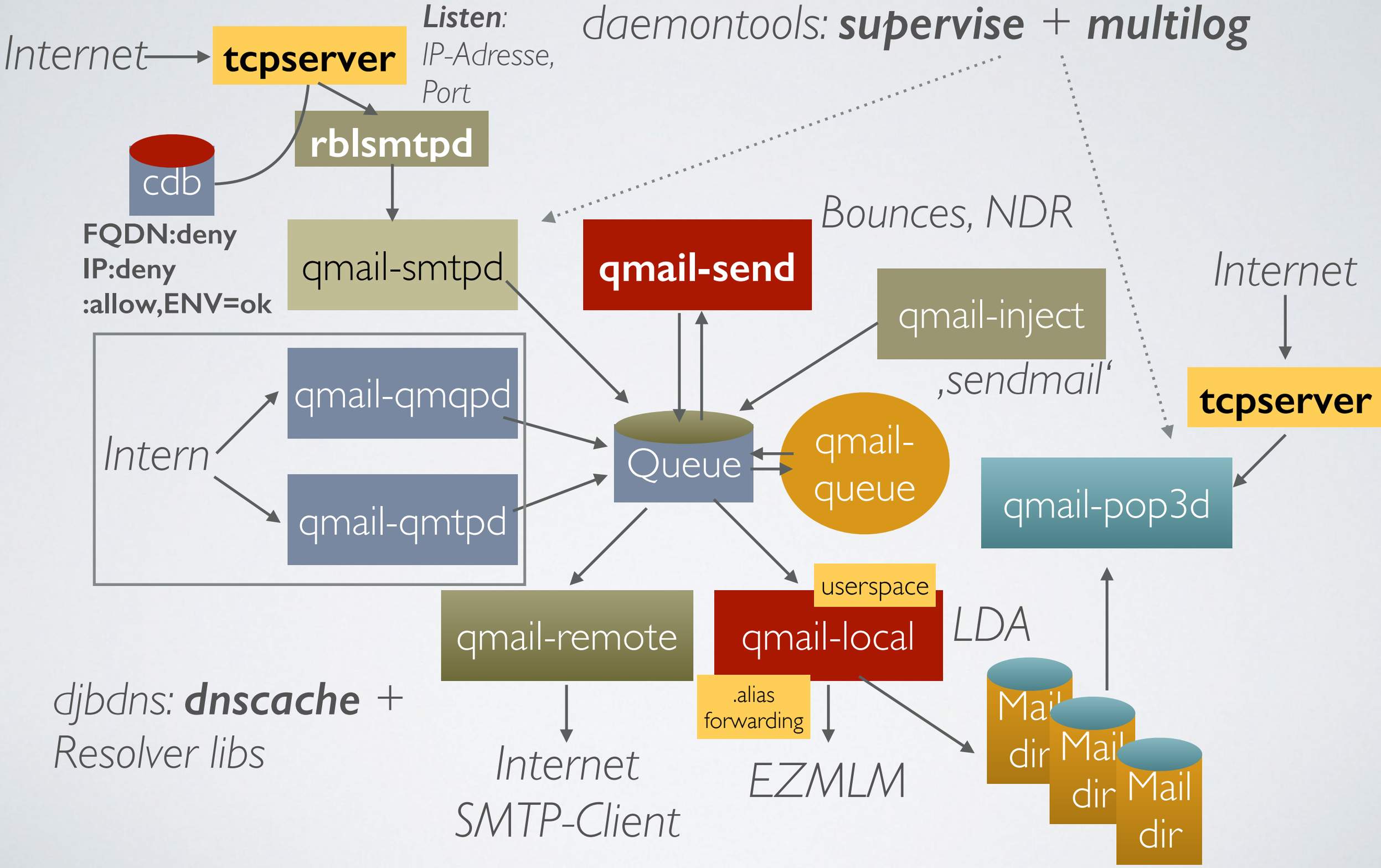
Entwicklung

- Daniel Bernstein entwickelte Qmail zwischen 1995 und 1998 (Version 1.03) als reine (E)STMP Implementierung (also ungefähr zeitgleich mit Wietse Venema's Postfix)

- Letzter ESMTP RFC, mit dem djb sich herumschlug, war RFC 2821 (‚Klensin-Draft')

- Etwa im Jahr 2004 waren die Bestandteile *Daemontools*, *djbdns* und *ucspi-tcp* vollständig entwickelt

Architektur



Vorteile

- Vorteile von Qmail:
  - Strenge Trennung der Aufgaben (einzelne Daemons); minimale Rechte
  - Modulares Konzept, kompakter Source-Code für einzelne Routinen (wenige hunderte Zeilen)
  - QMTP und Maildir-Unterstützung (von Postfix adaptiert)
  - String-Library *stralloc* von Bernstein

Nachteile

- Nachteile von Qmail (in Bezug auf SMTP):
  - Keine SMTP-Authentication
  - Keine TLS-Verschlüsslung
  - Eine Mail ist eine Transaktion (Pipelining nur bei **qmail-smtpd**)
  - Keine Plugin zum Scannen der Mail während der SMTP-Transaktion
  - Keine Recipient-Überprüfung beim SMTP-Daemon

Aktueller Stand

- Letzte ‚offizielle' Version ist qmail-1.03

  \+ einige (4 Bugfixes)

  \+ Qmail-Queue API

  ☞ **netqmail 1.06** (tarball + wenige Patches)

- AMD64 Support problematisch; **clang** ??
- Kein *utmpx.h* Support
- Kein IPv6 ;-)

Patches

- Weiterentwicklung durch Patches:
  - Qmail ist seit 2008 in der *Public Domain*; vorher waren Änderungen nur über Patches machbar

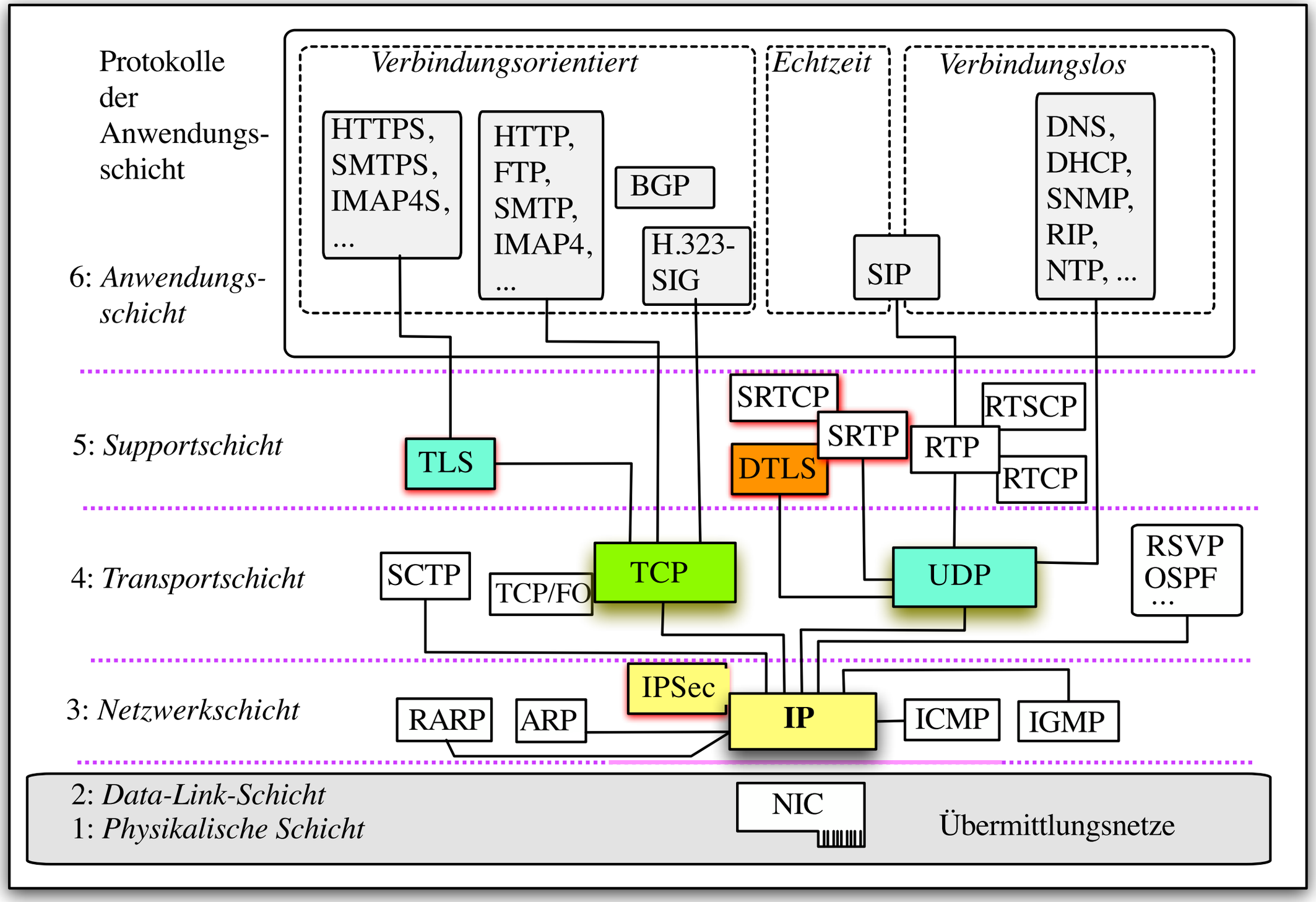
Eric Vermeulen's TLS Patch	Felix von Leitner's IPv6 Patch	Andre Oppermann's LDAP Patch	Spamcontrol Patch	Bill Shupp's „Qmail Toaster" Patchsammlung
netqmail *1.06 + 20110119*	***netqmail****-1.05-fefe3.diff.bz2*	*+ TLS* ***Status ?***		

1.0 RECIPIENTS Extension
2.0 AUTH ***qmail-smtpd***
2.3 QHPSI + WARLORD
*2.4 TLS + STARTTLS (****qmail-smtpd****)*
2.5 RECIPIENTS PAM
qmail-remote *AUTH*
2.6 TLS ***qmail-remote***
2.7 AMD64 compatibility

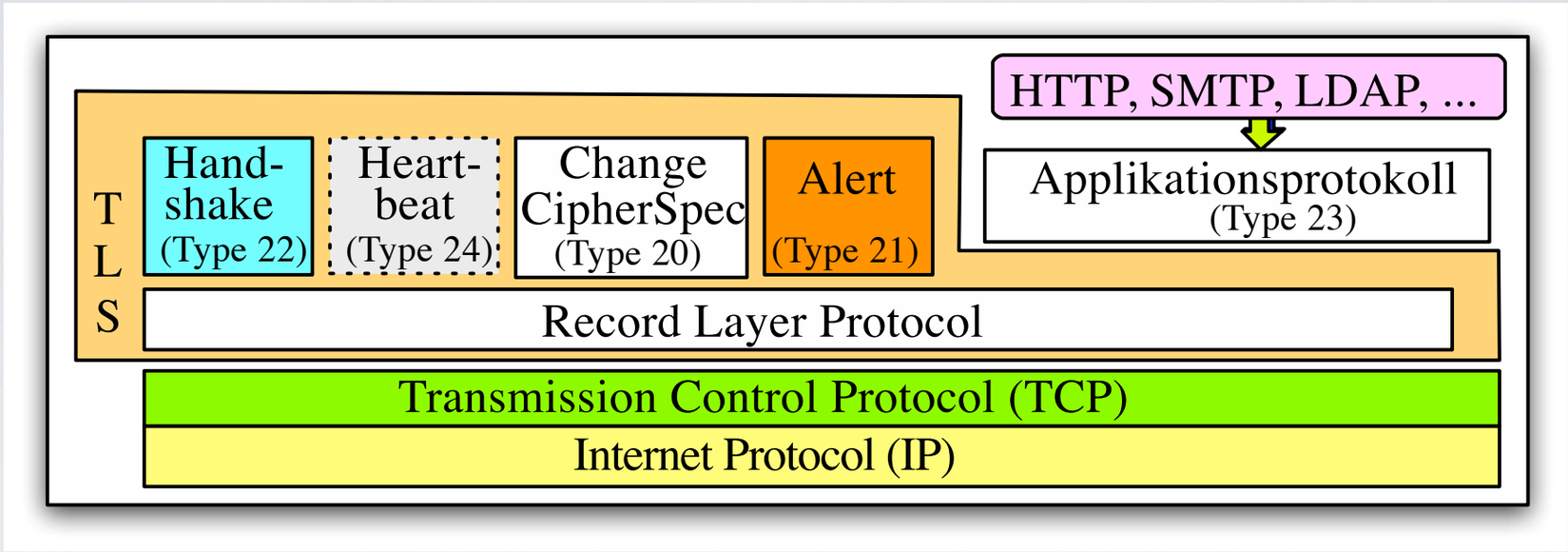
Status ?

Indimail ist ein Fork von Qmail!

TLS im Schichtenmodell

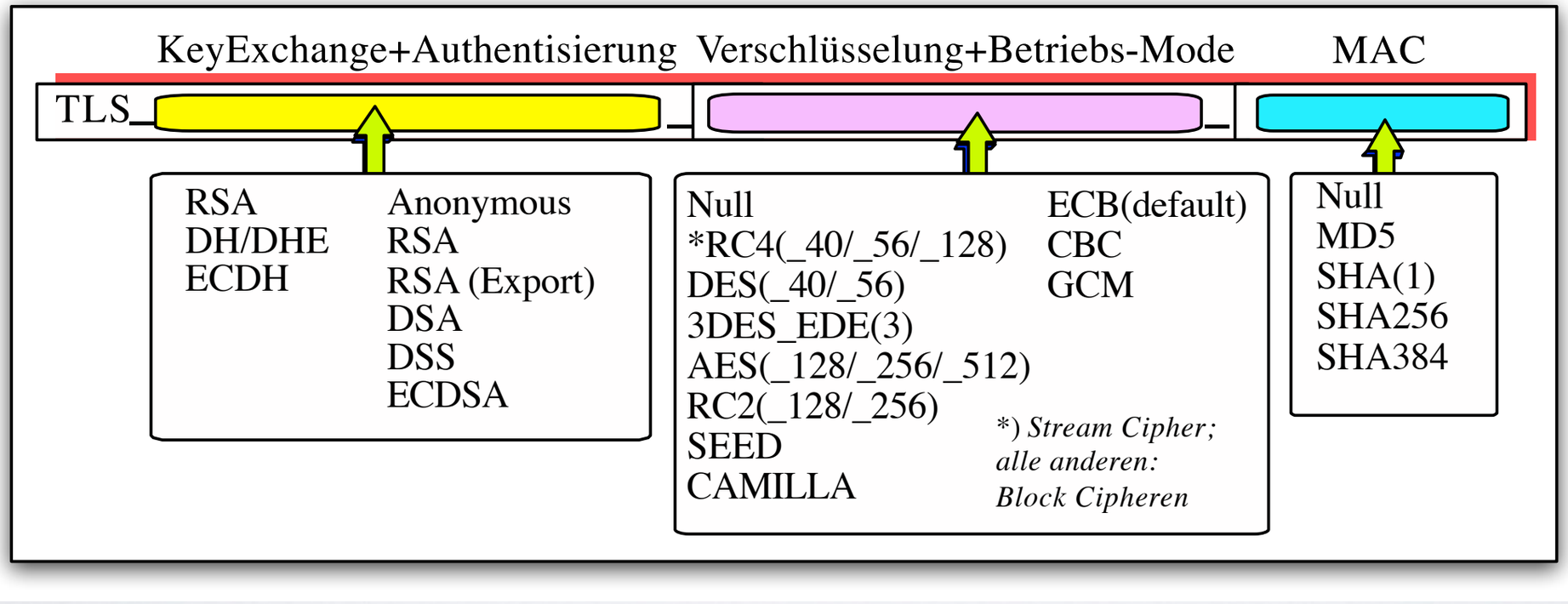


TLS hat selbst mehrere Schichten …



- Die **Record Layer** dient als Transportschicht
- **Handshake** für den Sitzungsaufbau
- *Out-of-Band Nachrichten* mittels **Change Cipher Spec**, **Alerts** und - neu - per **Heartbeat** (RFC 6520)
- ***Verschlüsselte Anwendungsdaten*** werden als Nachrichtentyp ‚23' übertragen

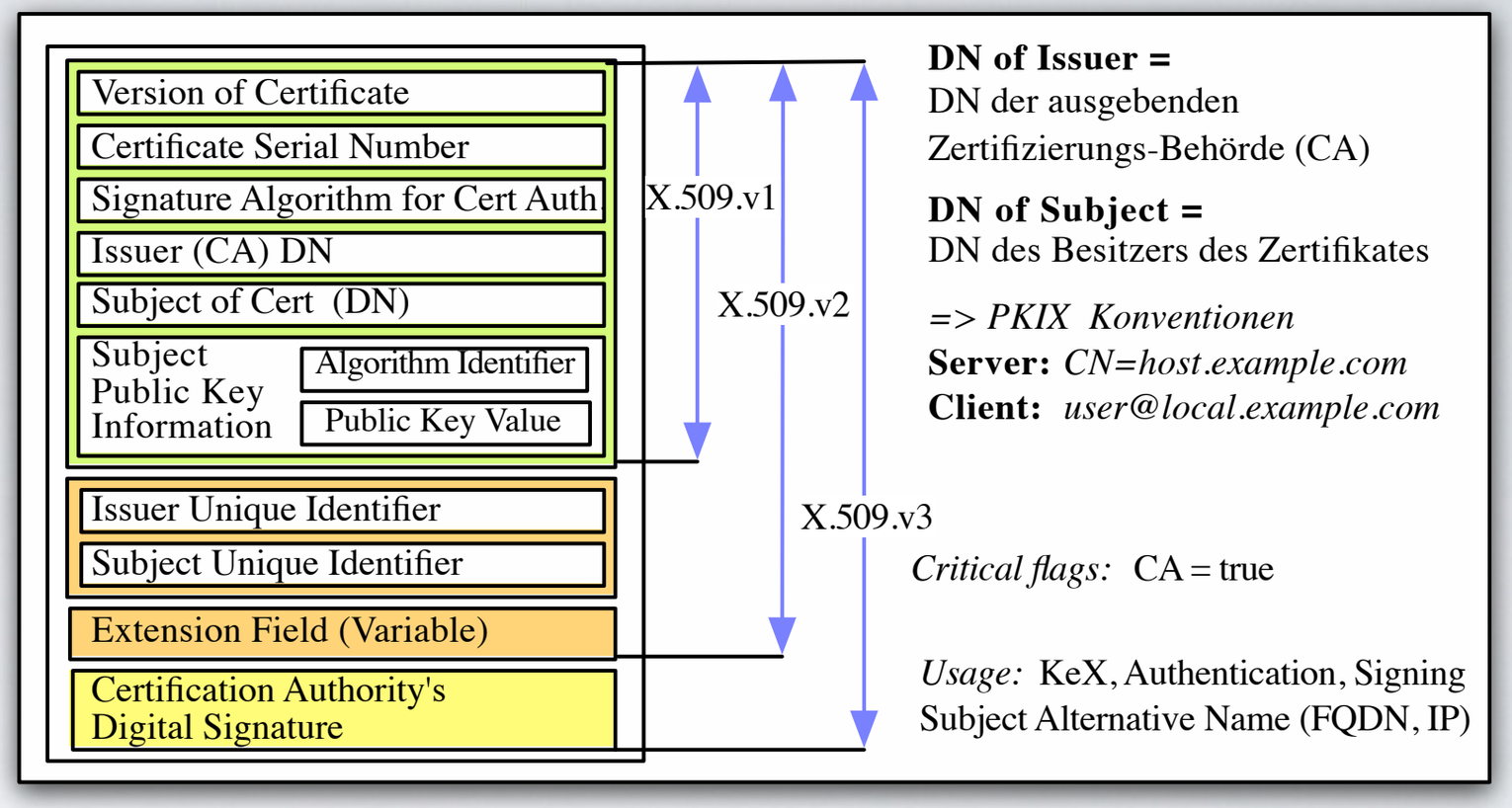
Cipher Suite



- Die ***Cipher Suite*** beschreibt den *Security-Kontext* für TLS

- Die ***Cipher Suite*** wird beim **Handshake** ausgehandelt; kann aber per ***Change Cipher Spec*** nachträglich geändert werden

X.509 Zertifikate



- **X.509 Zertifikate** werden beim *ephemeralen Diffie-Hellman* zur ***Signierung*** der DH-Parameter benötigt
- Sie dienen daher nur zur ***Authentisierung*** des Servers
- Beim ***Anonymous DH*** verzichtet der Client überhaupt auf die ***Validierung*** des Server-Certs

Transparenz von TLS + OpenSSL

- TLS war als ‚transparente' Sicherheitsschicht für die Applikation geplant; d.h. die Applikation merkt weder etwas von der stattfindenden Verschlüsselung; noch kann sie darauf Einfluss nehmen
  - Spätestens mit STARTTLS/STLS ist dieses (falsche) Paradigma nicht mehr gültig
  - Was ist mit TLS ALARM-Nachrichten ?

OpenSSL Schnittstellen

- **Input**:
  - Die *Cipher Suite*, die vorgegeben werden kann (*C+S*)
  - Der *Trust-* und *Key-Store* (bzw. dessen Inhalt)
  - Die *Diffie-Hellman Parameter* (DHPARAM)
  - *Validierung* des Zertifikates (ansonsten *Anonym*)
  - *Verifikation* des Hostname gegenüber Zertifikat
- **Output**:
  - ***mod_ssl*** (genutzte *Cipher Suite* und Partner-*DN; aber keine EKUs*)

ucspi-ssl

- Von ‚Superscript' gibt es als Ersatz von **ucspi-tcp** **ucspi-ssl**:
  - **ucspi-tcp** + OpenSSL Libraries
    - Keine Unterstützung für STARTTLS/STLS
    - Keine CRL-Unterstützung
    - Kein OCSP
- STARTTLS-Erweiterung von *Scott Gifford* !

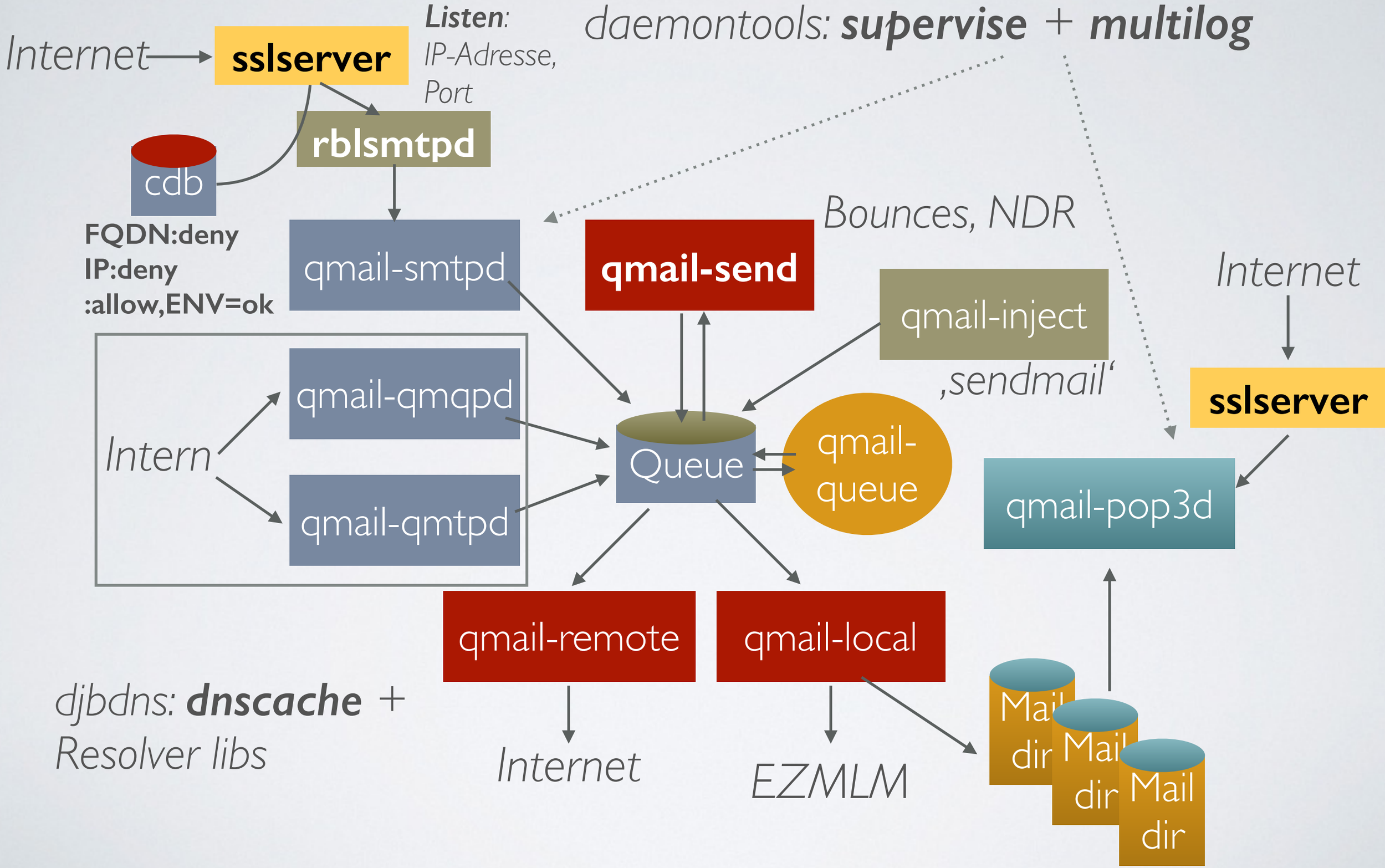
sslserver als Ersatz für tcpserver

- Qmail Empfangsprozesse:
  - **qmail-smtpd**
  - **qmail-pop3d/qmail-popup**
  - **qmail-qmtpd**
  - **qmail-qmqpd**

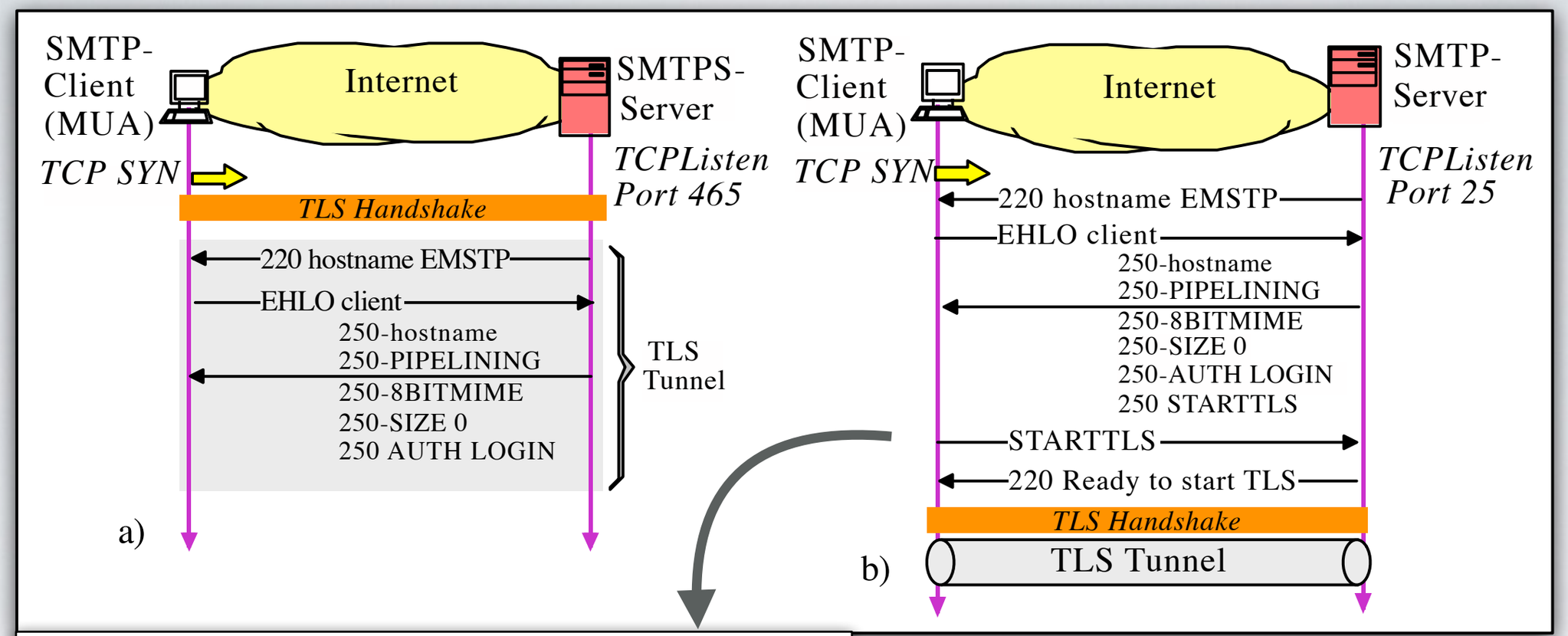
Alle diese Daemons nutzten **tcpserver** (ucspi-tcp)

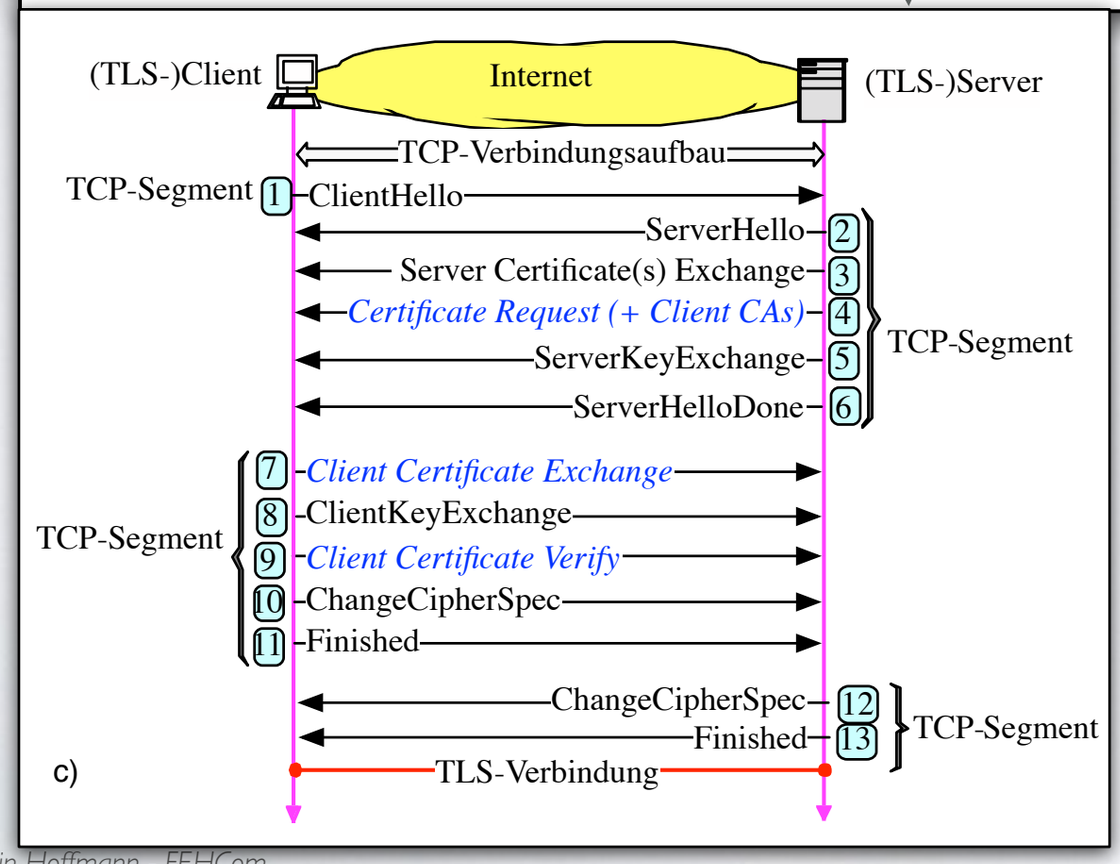
Und für den gibt es Ersatz: **sslserver** (ucspi-ssl)

sslserver im Einsatz



sslserver Anwendungsfälle

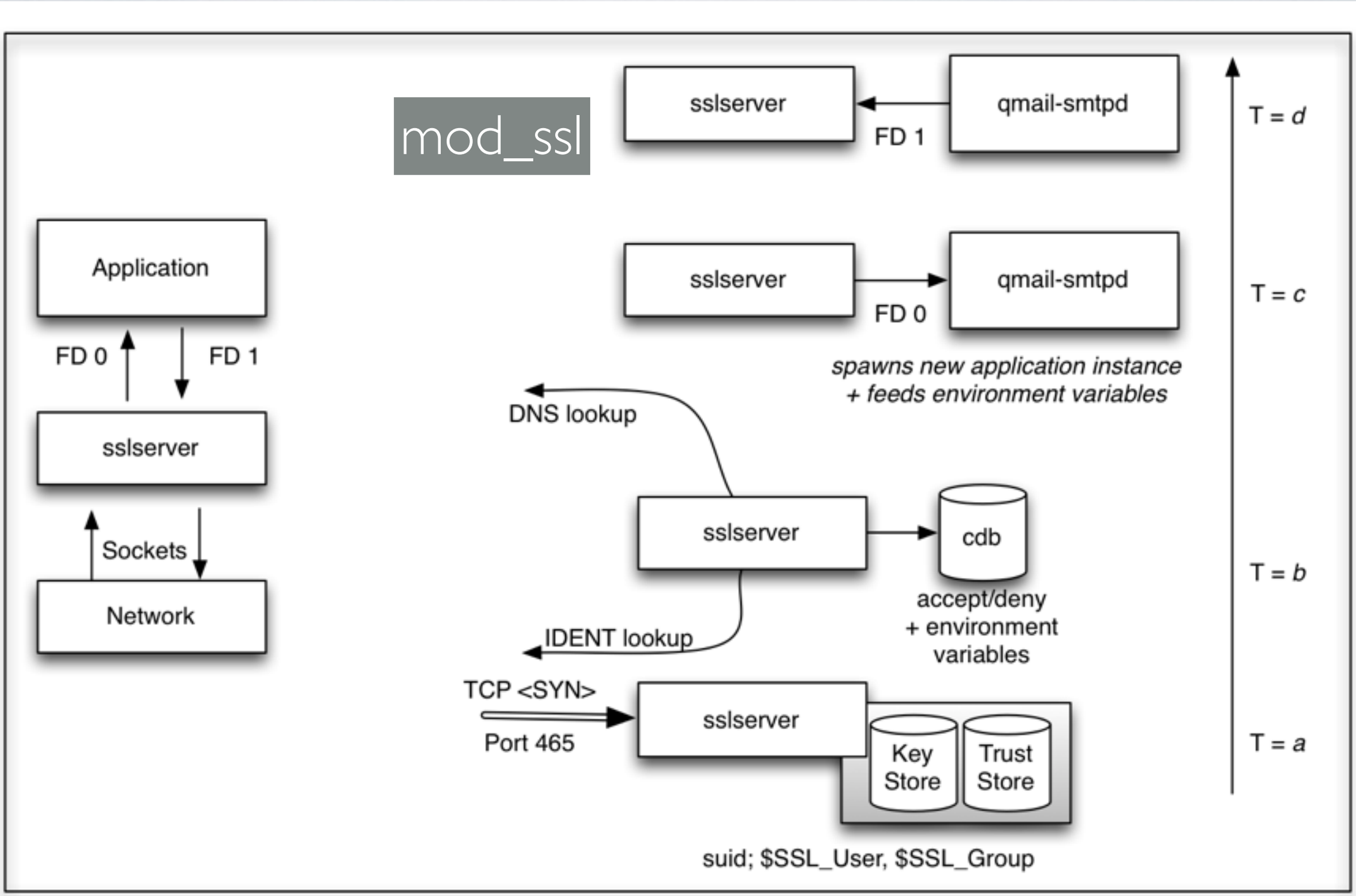




Drei Anwendungsfälle:

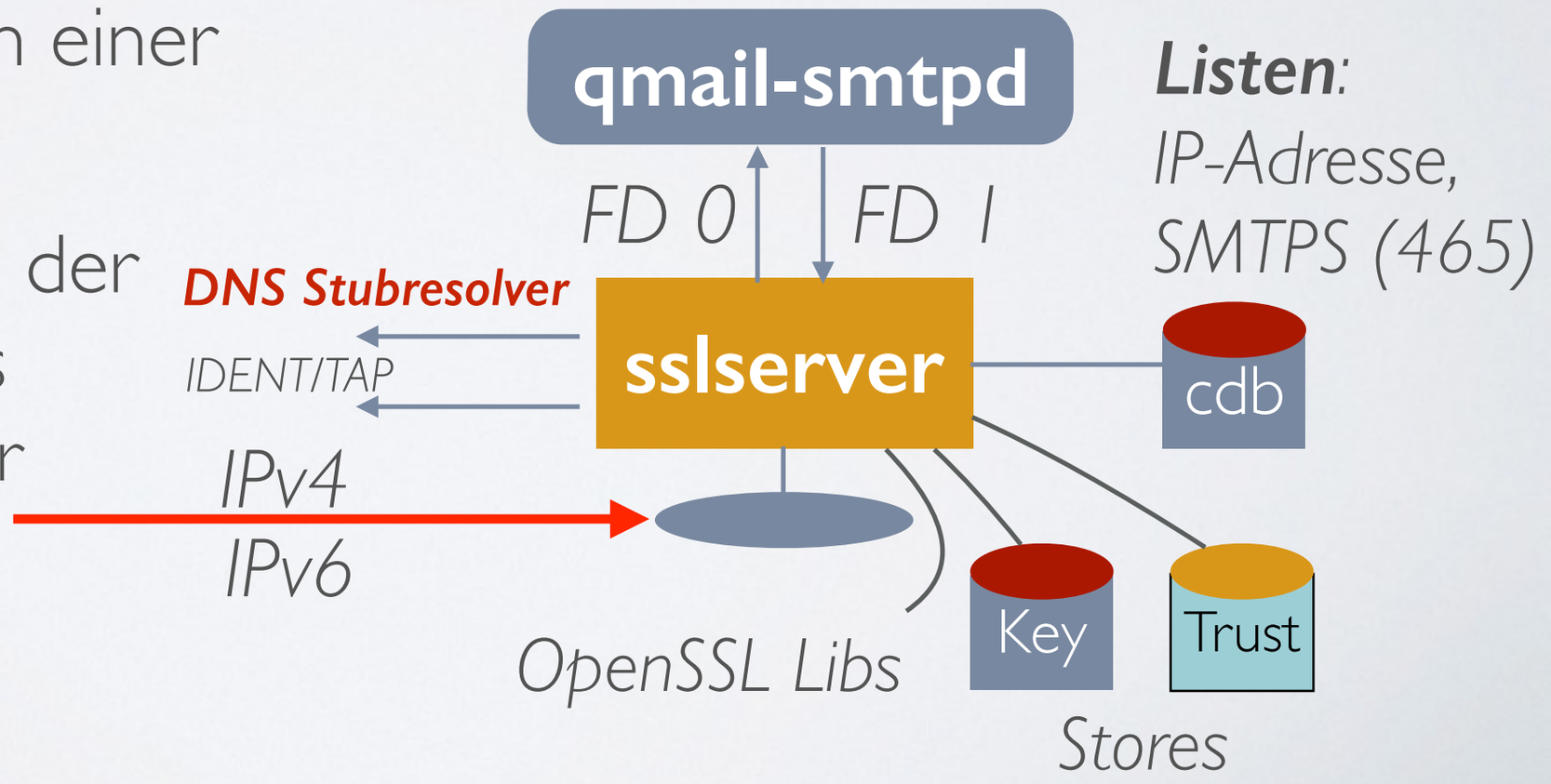
a) SMTPS
b) (E)STMP + STARTTLS
c) Client-Zertifikate

Fall a) sslserver für SMTPS



sslserver mit Stores

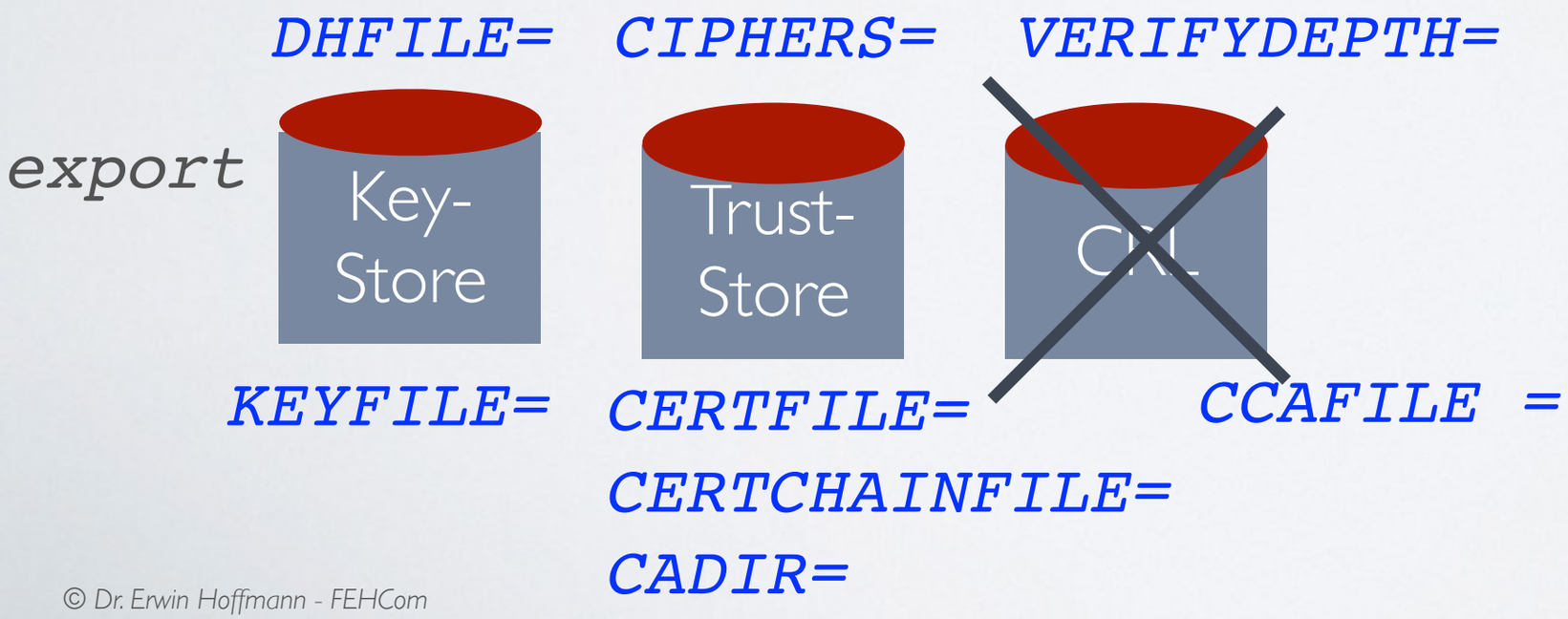
- Mittels **sslserver/sslclient** lassen sich verschlüsselte Client/Server-Anwendungen realisieren, ohne dass die Anwendung ein Socket-IF besitzen muss
  - TCP-Verbindungen können bei **sslserver** per IP-Adresse oder per FQDN kontrolliert werden
- Applikation läuft in einer *chroot*-Umgebung
- Auch das Einlesen der Zertifikate + Keys geschieht in eigener Umgebung



sslserver Syntax

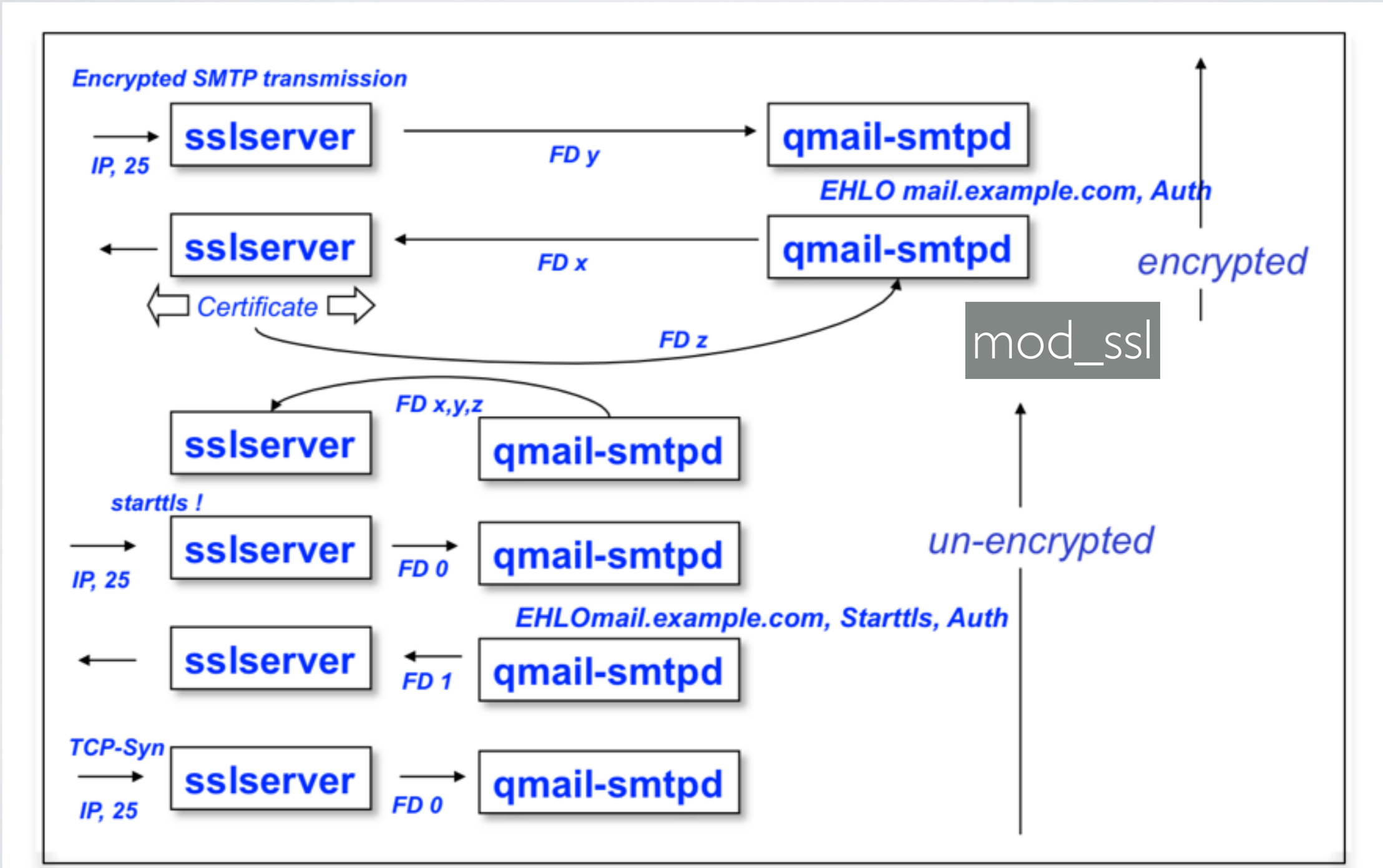
- **sslserver** *-4 | -6 -I ifname* -Rhp -x cdb \
  -sevn -m \
  -c connections \
  -u *user* -g *group* \
  -l *localhost* \
  IP-Adresse Port \
  progX *args* **progX server** *args*

Default ist immer IPv6 !



Keystore und Truststore können pro IP-Verbindung gewählt werden!

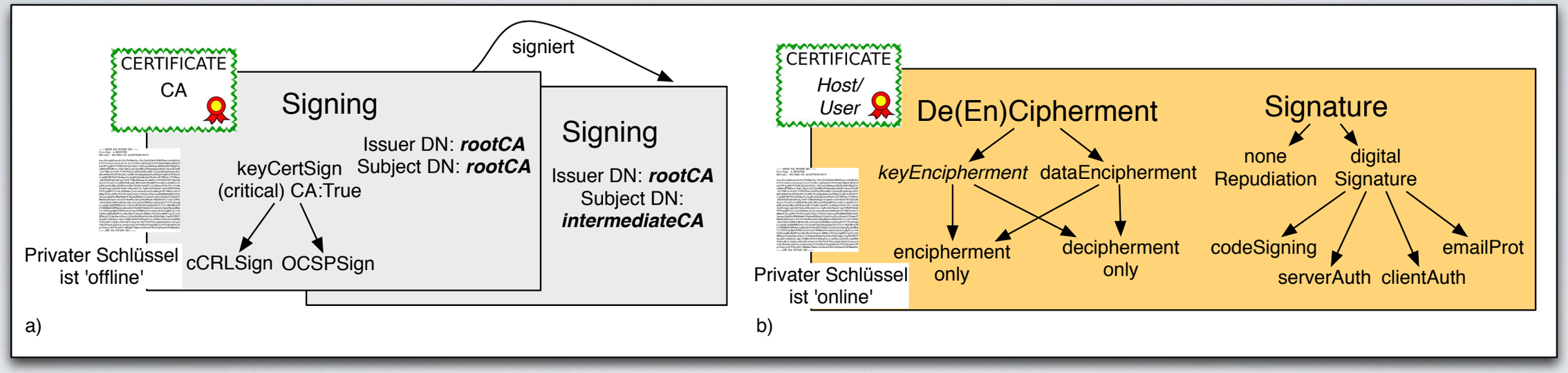
Fall b) sslserver mit STARTTLS Unterstützung



ucspi-ssl 0.94

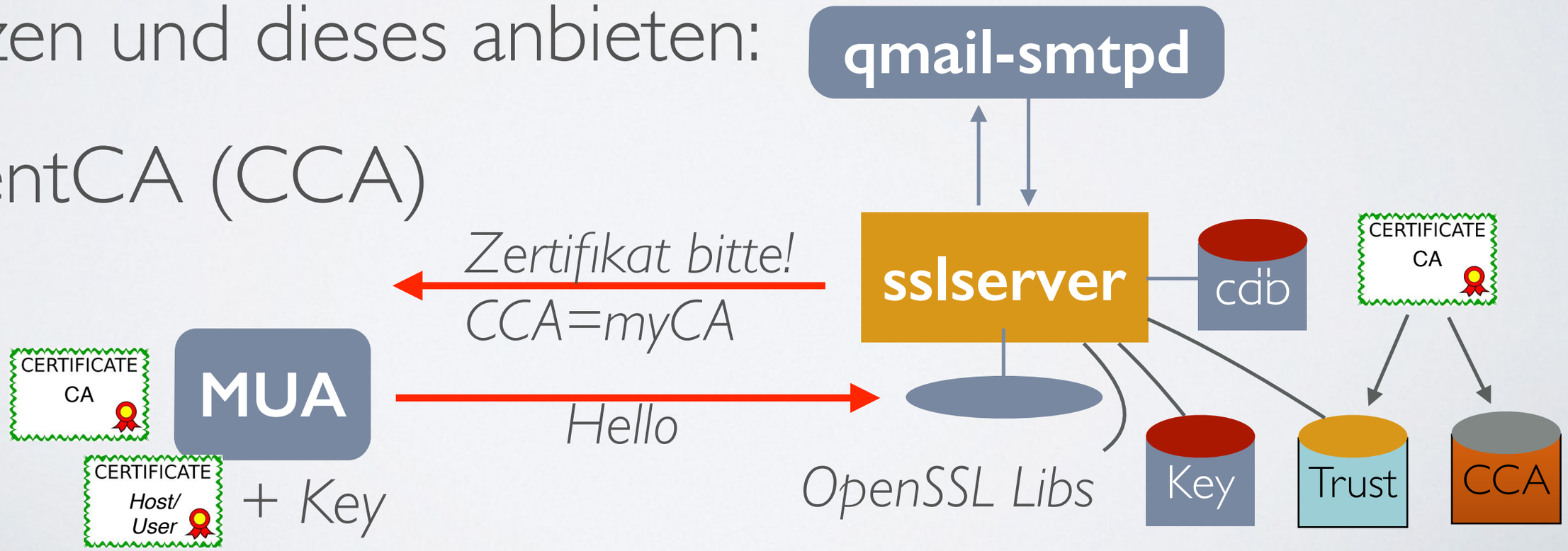
- **ucspi-ssl 0.94** ist Weiterentwicklung von Superscript's **ucspi-ssl 0.72**
- IPv6 Unterstützung + CIDR Filterung von IP-Adressen
- STARTTLS
- Zulässig nun auch ‚User' Client-Zertifikate für SMTP

Fall c) X.509 Client Certificate based Authentication



- X509 Certs (+Keys) werden an Clients ausgeliefert

- Server muss Stammzertifikat besitzen und dieses anbieten:
  - ClientCA (CCA)



Fall c) X.509 Client Certificate based Authentication - Voraussetzungen

- Eigene CA (self-signed Zertifikate sind ok)
- Personengebundene X.509 Zertifikate werden erzeugt
- Im DN des Client-Certs ist die Email-Adresse (des Benutzers) anzugeben
- Das Client-Cert wird mit der (eigenen) CA signiert
- Client-Cert + Keyfile (mit Passphrase geschützt) werden an die Benutzer/MUAs ausgerollt

Fall c) X.509 Client Certificate based Authentication - Server

- Eigenes CA Cert + Cert wird eingebunden.
- **sslserver** wird angewiesen, Client-Cert anzufordern
- Eigenes CA Cert (es können auch mehrere sein) werden an Client im erweiterten **Handshake** mitgegeben
- Client wählt passendes CA Cert aus und teilt sein Cert mit
- Server überprüft Client Cert gegen CA Cert und ob Client Key File vorliegt
  - Optionale Überprüfung der Email-Adresse
- Keine anschliessende SMTP Authentication !

qmail-remote als TLS-Client

- Anforderungen für **qmail-remote**:
  - ‚Passiver' TLS-Client: SMTPS Verbindungen auf Port 465 werden unterstützt (z.B. via **sslclient, ssltunnel**)
  - STARTTLS-fähiger Client: STARTTLS vom Server wird erkannt und in den TLS-Mode gewechselt
  - Überprüfender Client: X.509 Server-Zertifikate werden
    - *validiert* (Syntax, Signierung der DH-Parameter)
    - *verifiziert* (gegenüber einem bekannten CA Cert)
    - *Common Name* bzw. *SAN* gegenüber *Hostname*

qmail-remote Architektur

- Die TLS-Implementierung für **qmail-remote** wurde von mir von Grunde auf neu entwickelt und für Spamcontrol 2.6 implementiert

- Voraussetzung ist allerdings **ucspi-ssl**, da dessen high-level APIs auch hier genutzt werden

- Beim Kompilieren werden die **ucspi-ssl** Libraries eingebunden (und die von OpenSSL als Shared Libs)

TLS-Konfiguration qmail-remote

- Konfigurationsdatei:
  control/tlsdestinations

UCSPITLS=„“
UCSPITLS=„!“
UCSPITLS=„-„

STARTTLS (?) für alle:
TLS mit CA Cert + Cipher:
TLS mit CA Dir:
TLS für Absender:
TLS mit Host-Überprüfung:
Anonymous DH:
TLS auf Port 35:
TLS nicht für diesen:

```
*:
.example.com:
securityfirst.com:/etc/ssl/cafile||!SSLv2:HIGH
.remote.com:/etc/ssl/certdir/|3:465
mx.partner.com:/etc/ssl/partnerca||:26|mydomain.net
=mx.myfriend.com:/etc/ssl/cacert|4
-.adhonlydomain.com:|aNULL:!kRSA
=*:
hiddenpartner.org:||:35
!nosslhost.example.com:
```

qmail-remote erlaubt die Verwendung dedizierter Cipher pro Kommunikationspartner.
Bedingt durch die *Canonicalization* ist der Trigger die ‚Mail From:‘ Adresse.

TLS-Authentication mit qmail-remote

- Bei geschäftskritischer Kommunikation sollte dem Partner ein X.509 Zertifikat angeboten werden

- Konfigurationsdatei:

  control/domaincerts

Server-Zertifikat: `*:mycert|mykey|mypassword`
Domain-Zertifikat: `example.com:thiscert|thiskey|thispassword`

qmail-remote erlaubt pro sendender Domain in ‚Mail From:' die Angabe eines dedizierten X.509 Zertifikates.

Für Mailserver, die viele Domains hosten, kann statt eines *Server-Zertifikates* ein *Domain-Zertifikat* genutzt werden.

Zusammenfassung

- qmail/Spamcontrol bietet mit zusammen **ucspi-ssl** die weitgehende Nutzung von TLS für die SMTP-Kommunikation:
  - **qmail-smtpd** unterstützt unterschiedliche Key/Trust-Stores pro Verbindung
  - Dies gilt auch für **qmail-pop3d**
  - **qmail-remote** bietet umfangreiches TLS tweaking
    - Besonderheit ist die Domain-Konfiguration

Ausblick

- qmail/Spamcontrol realisiert alle meine Anforderungen an die TLS-Kommunikation
  - Es fehlt noch die Einbeziehung der OpenSSL ALARM Mitteilung (bzw. dessen Ausgabe im Log)
- Die Komplexität von qmail/Spamcontrol erlaubt keine Weiterentwicklung als Patch (➪ **s/qmail**)
  - IPv6 Integration ist notwendig
  - **djbdnscurve6** ist auf der Warteliste

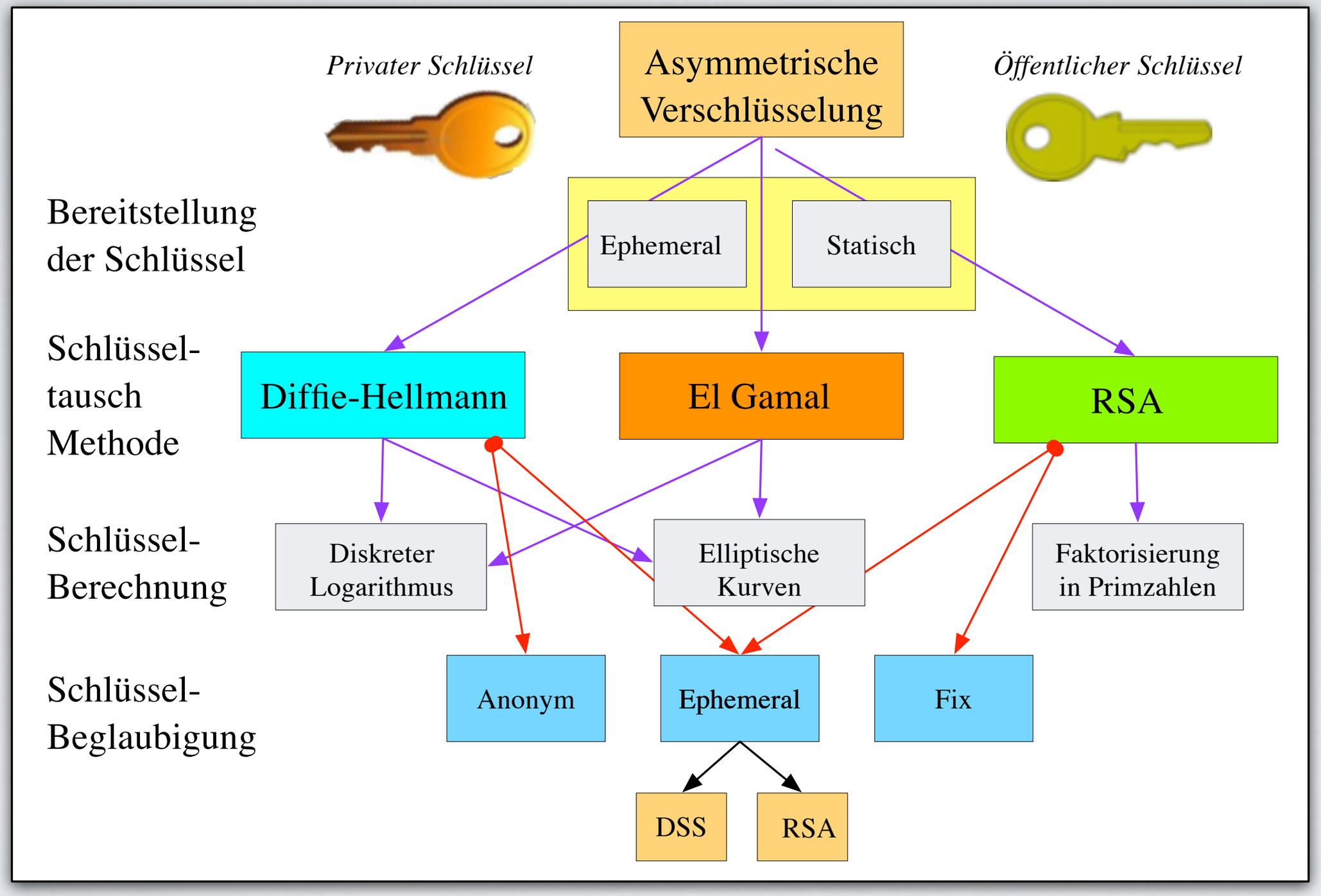
Quellen

- http://cr.yp.to/ucspi-tcp.html
- http://cr.yp.to/qmail.html
- http://www.qmail.org
- http://www.fehcom.de/qmail/qmailbook.html
- http://www.fehcom.de/qmail/smtptls.html
- http://www.fehcom.de/ipnet/ucspi-ssl.html
- http://www.fehcom.de/ipnet/ucspi-tcp6.html
- http://www.superscript.com/ucspi-ssl/index.html
- http://www.suspectclass.com/sgifford/ucspi-tls
- http://inoa.net/qmail-tls/
- http://www.qmail-ldap.org/
- http://www.fefe.de/qmail/
- http://www.qmailtoaster.com/

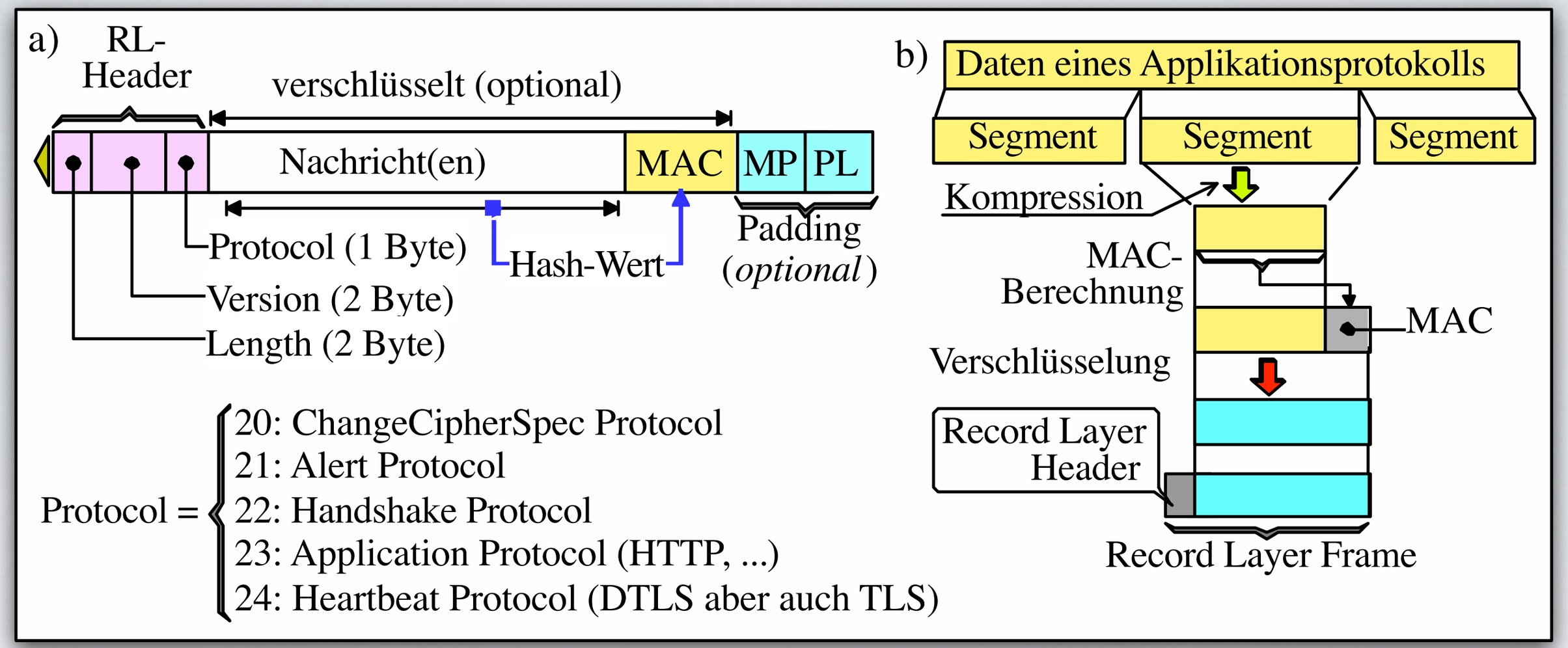
Fragen ?

Antworten !

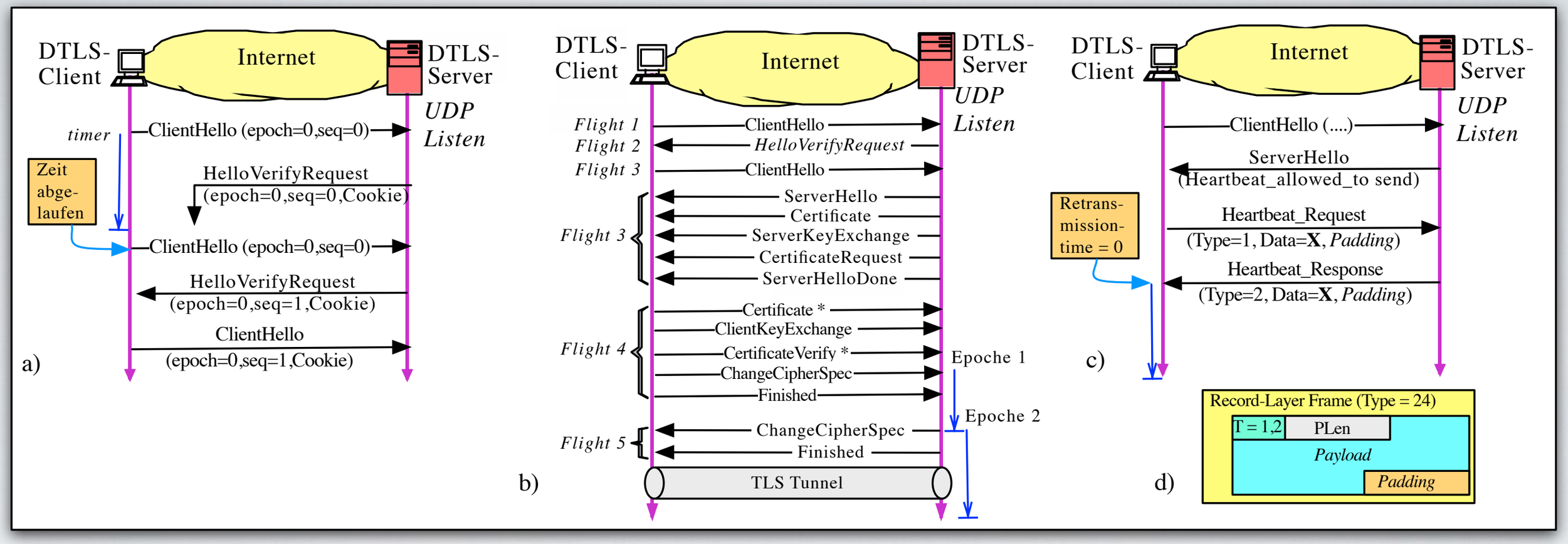
Asymmetrische Verschlüsselung



TLS Nachrichten



Heartbeart Funktion



Heartbeart Funktion

- **sslserver** hängt von OpenSSL ab.

- Daher ist **sslserver** auch vom Heartbleed Bug in OpenSSL 1.0.1 betroffen … aber

- er kann nicht ausgenutzt werden, da pro Client-IP eine **sslserver**-Instanz geöffnet wird (neuer Speicherbereich) und dieser auch nur für diesem Client.

- Nach dem Lesen der CA Certs und des Keyfiles erfolgt die eigentliche Verbindungs-ver/entschlüsselung in einer *chroot*-Umgebung statt.

- Diese werden für die aktuelle Verschlüsselung auch gar nicht benötigt; ausser zum Schlüsseltausch bei RSA (keine *Perfect Forward Secrecy* **PFS**).

- Allerdings stehen die Certs im ***Kontext*** der SSL-Verbindung.