Quelle: Computernetzwerke: Der Top-Down-Ansatz, 6. Ausgabe
Jim Kurose, Keith Ross; Pearson, März 2014

# Technische Grundlagen der Informatik 2
# Teil 5: Sicherheit in Layer 4/7

Philipp Rettberg / Sebastian Harnau

# Block 9/18

Sicherheit (Layer 4 / 7)

Email, Absicherung anderer Dienste

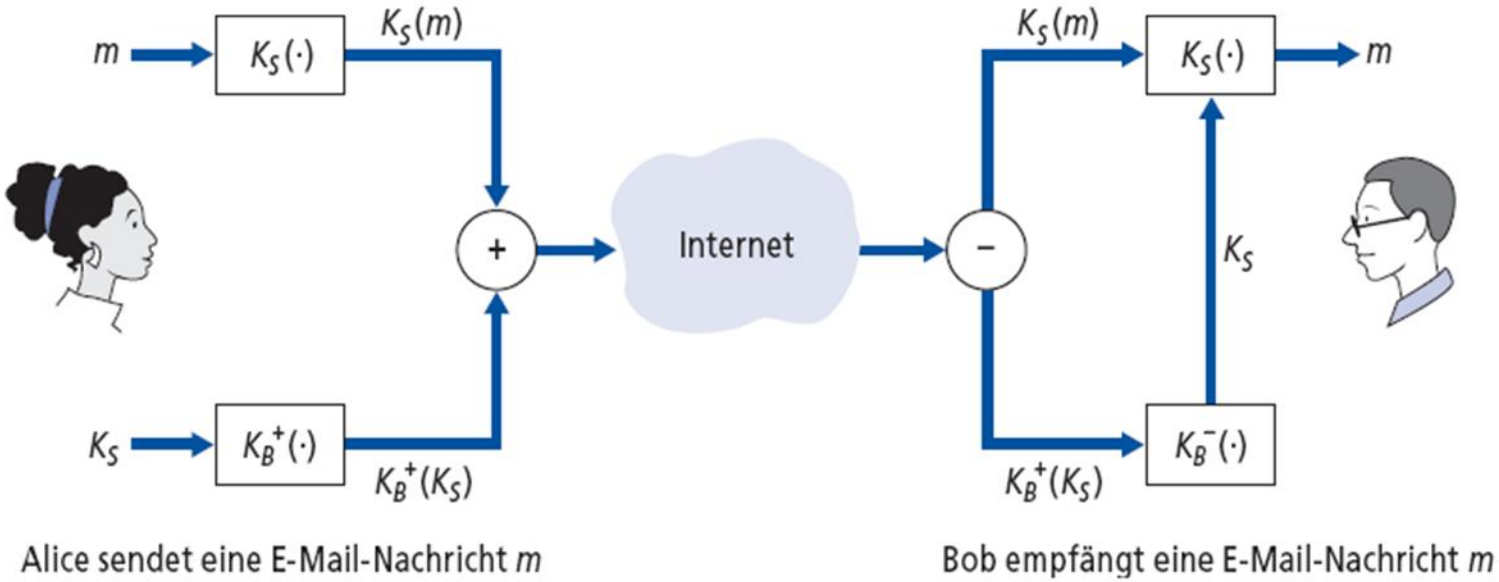
# Absichern von E-Mail

# Sichere E-Mail

Alice möchte eine vertrauliche E-Mail m an Bob schicken

**Alice:**

- erzeugt einen zufälligen symmetrischen Schlüssel $K_S$
- verschlüsselt die Nachricht mit $K_S$ (aus Effizienz-gründen)
- verschlüsselt $K_S$ mit Bobs öffentlichem Schlüssel
- sendet sowohl $K_S(m)$ als auch $K_B(K_S)$ an Bob




Alice sendet eine E-Mail-Nachricht *m*

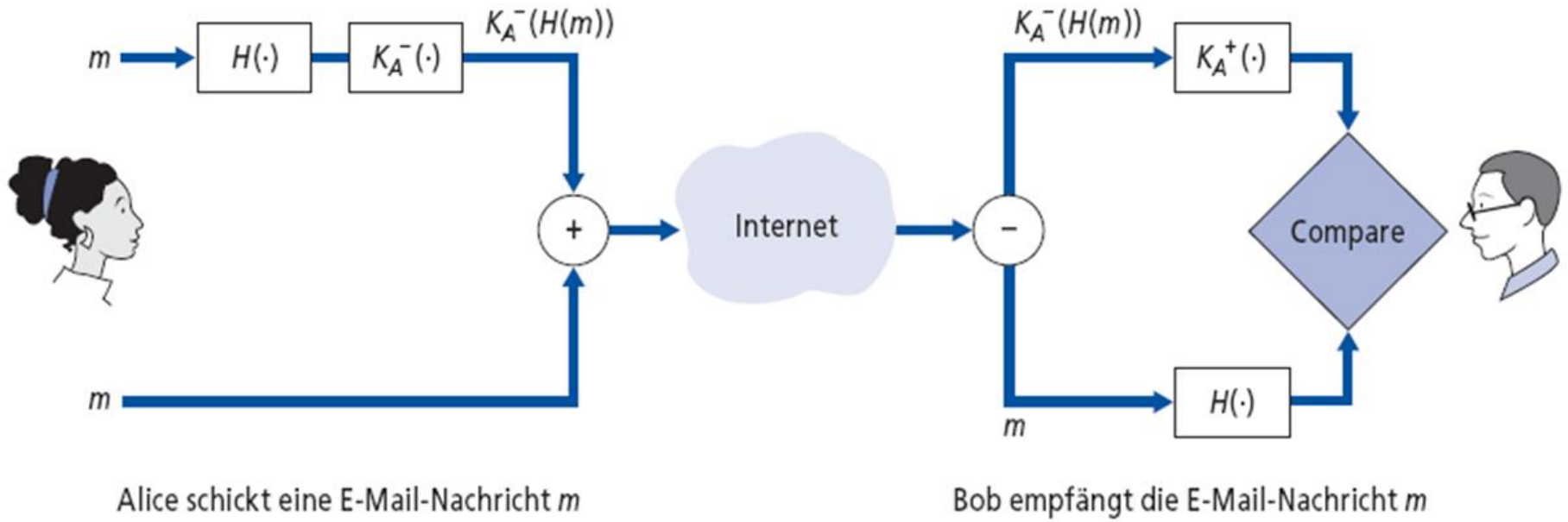
Bob empfängt eine E-Mail-Nachricht *m*

**Bob:**

- verwendet seinen privaten Schlüssel, um KS zu erhalten
- verwendet KS, um KS(m) zu entschlüsseln und m zu lesen

# Sichere E-Mail

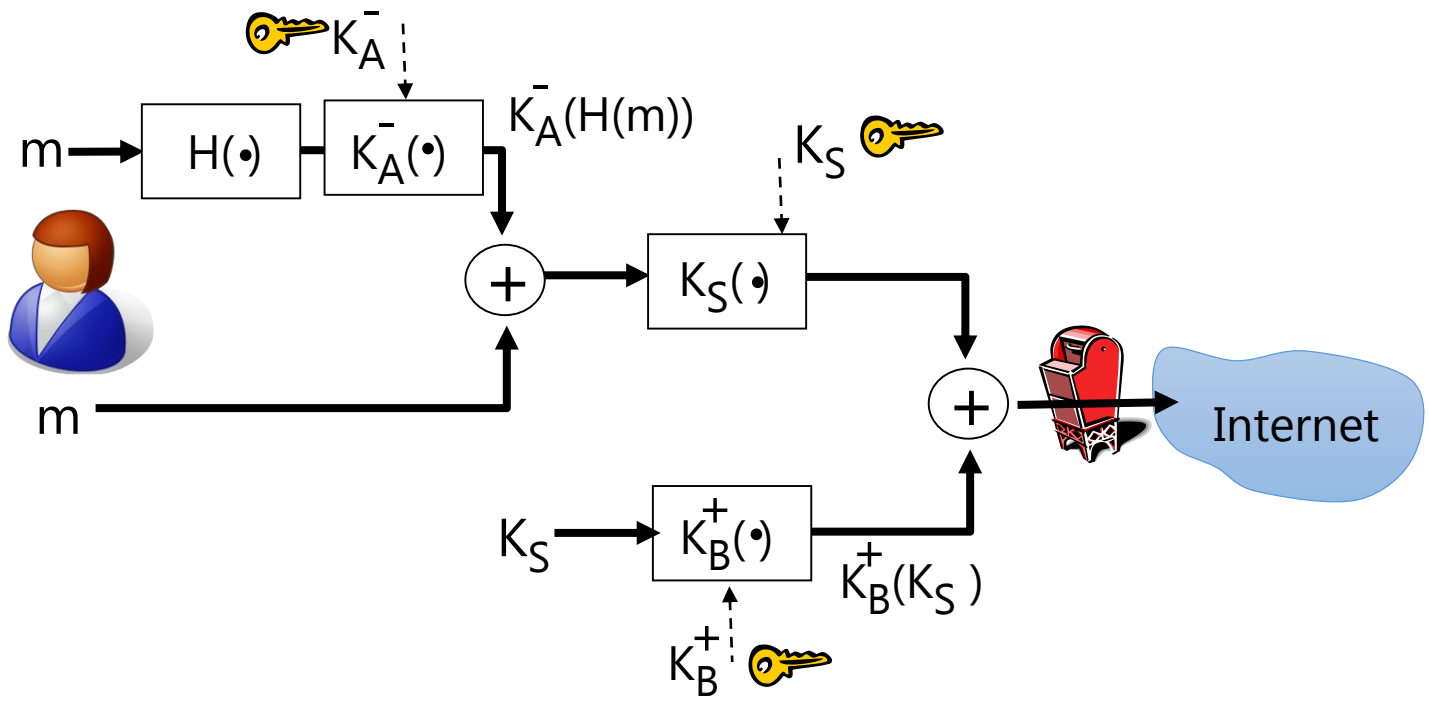
Alice möchte Absenderauthentifizierung und Nachrichtenintegrität sicherstellen




- Alice unterschreibt die Nachricht digital
- sie schickt sowohl die Nachricht als auch die Signatur

# Sichere E-Mail

Alice möchte Vertraulichkeit, Absenderauthentifizierung und Nachrichtenintegrität sicherstellen




**Alice verwendet drei Schlüssel:** ihren privaten Schlüssel, Bobs öffentlichen Schlüssel und einen neu erstellten symmetrischen Schlüssel

# Pretty Good Privacy (PGP) / GnuPG

- Verfahren für E-Mail-Verschlüsselung im Internet, De-facto-Standard
- verwendet symmetrische Kryptographie, Public-Key-Kryptographie, Hashfunktionen und digitale Unterschriften wie beschrieben
- bietet Vertraulichkeit, Absenderauthentifizierung, Integrität

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town
tonight.Passionately yours,
Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mq
JhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

# PGP/GnuPG Probleme

- PGP / GnuPG erfordert zusätzliche Software auf Sender-/Empfängerseite
- Handling der Schlüssel nicht trivial
- Keyserver für öffentliche Schlüssel sind Schwachstelle, hier kann das System durch gezielte Verbreitung von nicht funktionierenden Schlüsseln gestört werden
- es kann wg. SMTP nur der Mail-Body verschlüsselt werden, eine Mail beinhaltet jedoch viele Metadaten:
    - Sender
    - Empfänger
    - Betreff
    - Sendezeit
    - Verlauf der Übertragung über beteiligte Systeme
    - genutzte Softwarekomponenten
    - …

# Darkmail (Konzept)

- Aufteilung der Mail in verschiedene Bereiche (Envelopes), die einzeln mit Public-Key-Verfahren für den jeweiligen Informationsempfänger verschlüsselt werden. Jede beteiligte Station kann nur genau die Inhalte der Mail entschlüsseln, die für die Zustellung benötigt werden, z.B.:
    - Eigener Mailserver: Absender, Empfangsserver
    - Empfänger-Mailserver: Absendeserver, Empfänger
    - Mails liegen komplett verschlüsselt auf den Servern
    - Bereitstellung der öffentlichen Schlüssel per DNS

**Siehe auch: http://darkmail.info/**

# DomainKeys Itentified Mail (DKIM)

- Identifikationsprotokoll zur Sicherstellung der Authentizität von E-Mail-Absendern
- DomainKeys basiert auf asymmetrischer Verschlüsselung
- E-Mail wird mit einer Digitalen Signatur versehen, die der empfangende Server anhand des öffentlichen Schlüssels, der im Domain Name System (DNS) der Domäne verfügbar ist, verifizieren kann.
- Schlägt dies fehl, hat der empfangende Mail Transfer Agent (MTA) oder das empfangende Anwendungsprogramm die Möglichkeit, die E-Mail zu verweigern oder auszusortieren.

- Siehe: http://tools.ietf.org/wg/dkim/
- Vgl.: Sender-Policy-Framework SPF: http://de.wikipedia.org/wiki/Sender_Policy_Framework

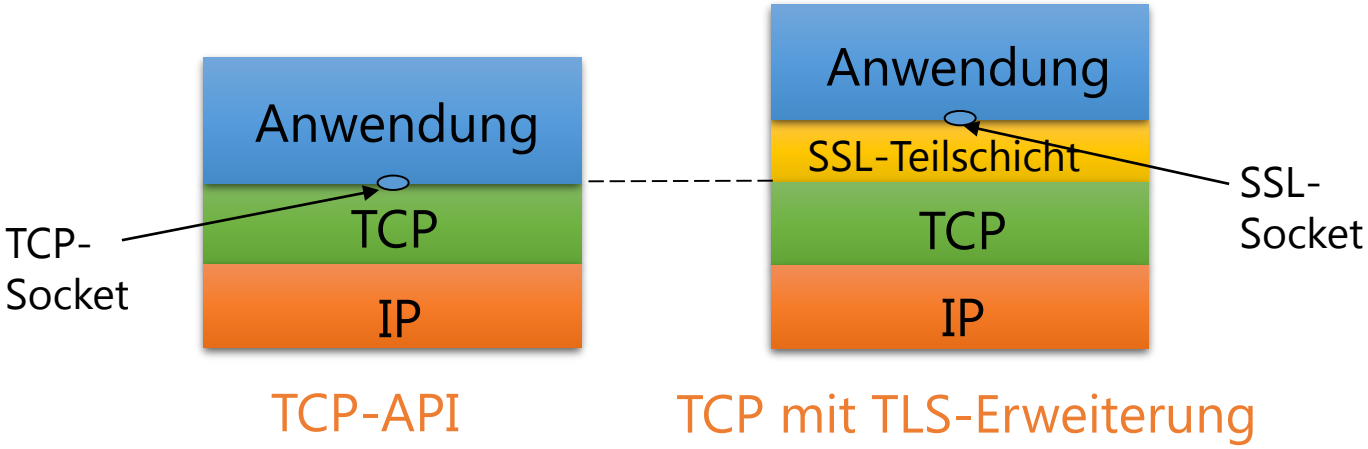
# Absichern anderer Dienste (Beispiele)

# DNS - DNSSec

- DNSSEC verwendet ein asymmetrisches Kryptosystem.
- Der „Besitzer“ einer Information – in der Regel der Master-Server, auf dem die abzusichernde Zone liegt – unterzeichnet jeden einzelnen Record mittels seines geheimen Schlüssels (engl. private key).
- DNS-Clients können diese Unterschrift mit dem öffentlichen Schlüssel (verfügbar als DNS-Eintrag) des Besitzers validieren und damit Authentizität und Integrität überprüfen.

# TLS für HTTP, FTP…

- Nutzung von **Transport-Layer-Security (Transportschicht-Sicherheit)** für beliebige TCP-basierte Anwendungen
  - z.B. zwischen Webbrowser und Webserverserver für E-Commerce (HTTPS)
  - Z.B. zwischen FTP-Client und FTP-Server (FTPS)
- Sicherheitsdienste:
  - Serverauthentifizierung
  - Datenverschlüsselung
  - Clientauthentifizierung (optional)

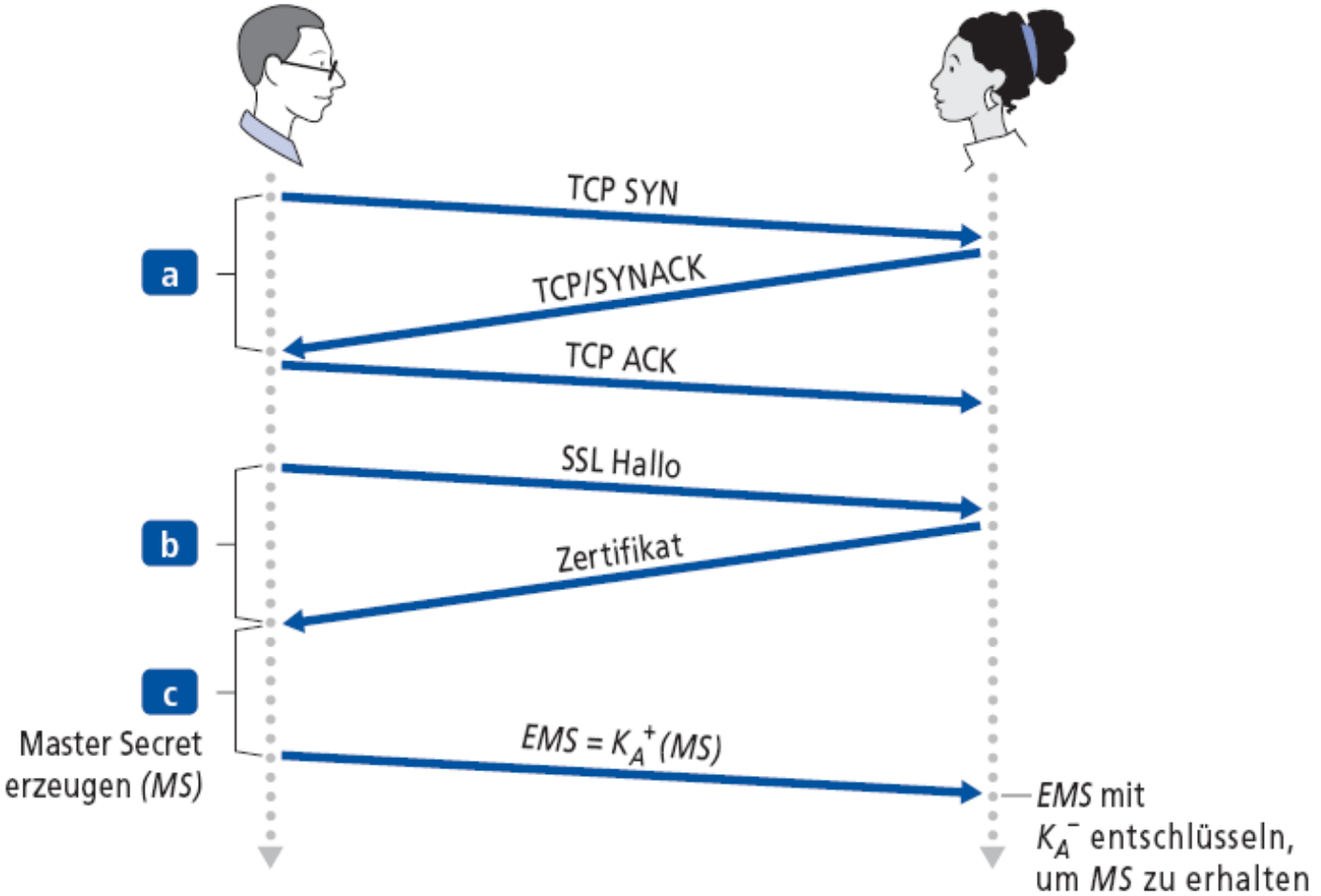


# TLS: drei Phasen

**1. Handshake:**

- Bob baut eine TCP-Verbindung zu Alice auf
- authentifiziert Alice über ein CA-signiertes Zertifikat
- erzeugt, verschlüsselt (mit Alices öffentl. Schlüssel) und verschickt ein Master Secret an Alice

Nonce-Austausch wird hier nicht gezeigt



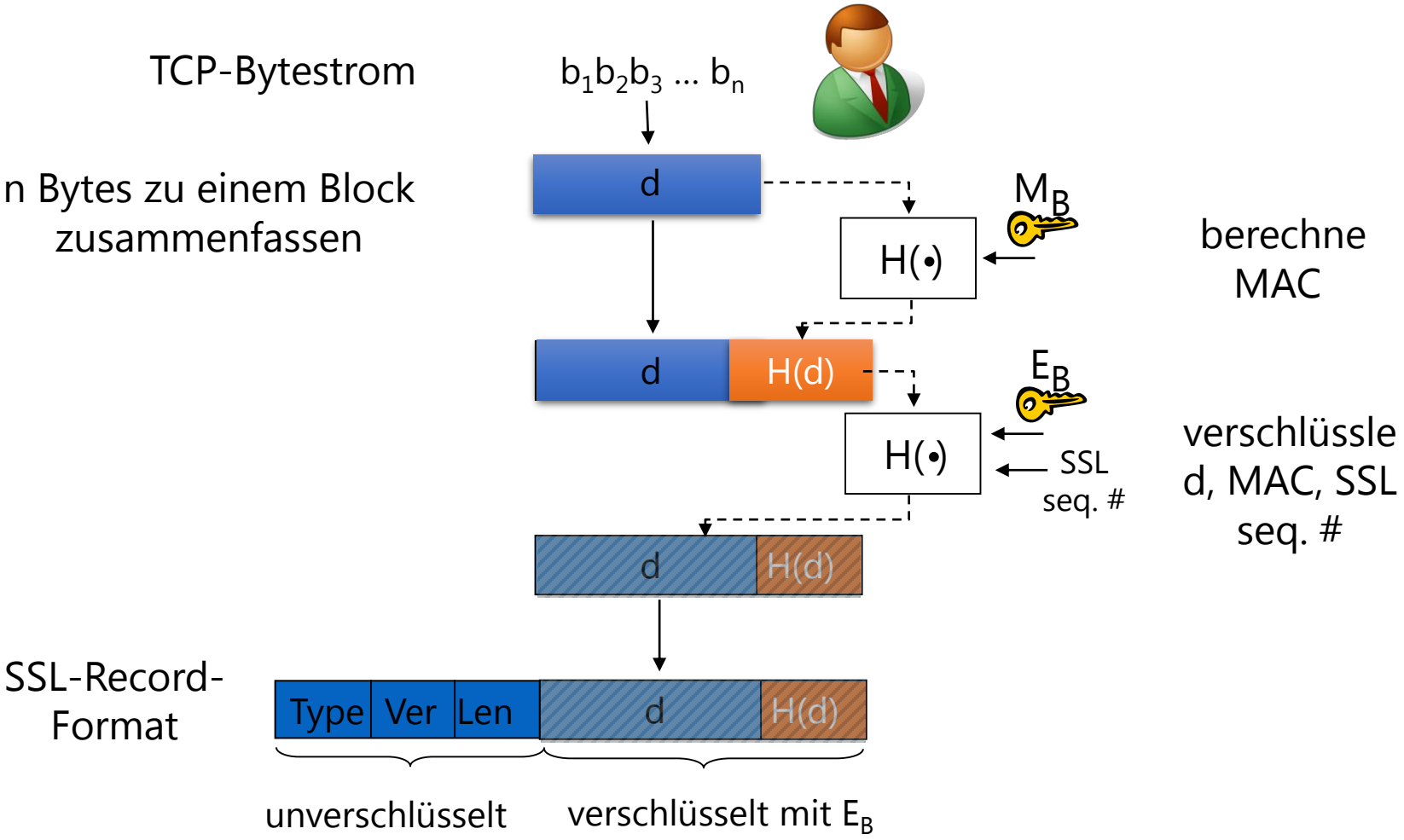
# TLS: drei Phasen

**2. Schlüsselableitung:**

- Alice und Bob verwenden das Master Secret, um vier Schlüssel zu erzeugen:
  - EB: Schlüssel für Verschlüsselung Bob->Alice
  - EA: Schlüssel für Verschlüsselung Alice->Bob
  - MB: MAC-Schlüssel Bob->Alice
  - MA: MAC-Schlüssel Alice->Bob
- Verschlüsselungs- und MAC-Algorithmen können zwischen Bob und Alice ausgehandelt werden

# TLS: drei Phasen

**3. Datentransfer**



# TLS-Beispiel: Apache Webserver

- Konfiguration vhost unverschlüsselt:

```
Listen 80

<VirtualHost *:80>
    DocumentRoot "/www/example1"
    ServerName www.example.com
</VirtualHost>
```

- Konfiguration vhost verschlüsselt:

```
Listen 443

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /root/ssl/domain-name.crt
    SSLCertificateKeyFile /root/ssl/domain-name.key
    DocumentRoot "/www/example1"
    ServerName www.example.com
</VirtualHost>
```

# Beispiel: Erstellung Zertifikate & Signieren

- Erzeugen des privaten Schlüssels:
  - **`openssl genrsa -out domain-name.key 2048`**
- Erzeugen eines sog. Certificate Signing Requests
  - **`openssl req -new -key domain-name.key -out domain-name.csr`**
- Signieren des Zertifikats
  - **`openssl x509 -req -days 365 -in domain-name.csr -signkey domain-name.key -out domain-name.crt`**