zurück

Verschlüsselung – Grundlagen (TLS, HTTPS, Zertifikate, PKI, Hashing)

Verschlüsselung schützt Daten vor unerlaubtem Mitlesen und Manipulation.
Sie ist überall in der IT unverzichtbar – bei Webservern, E-Mails, VPN, WLAN, Backups, Passwortspeicherung und Identitätsverwaltung.

Diese Seite behandelt:

- symmetrische & asymmetrische Verschlüsselung
- Hashing
- TLS & HTTPS
- Zertifikate
- Public-Key-Infrastruktur (PKI)
- CA, Intermediate, Chain of Trust

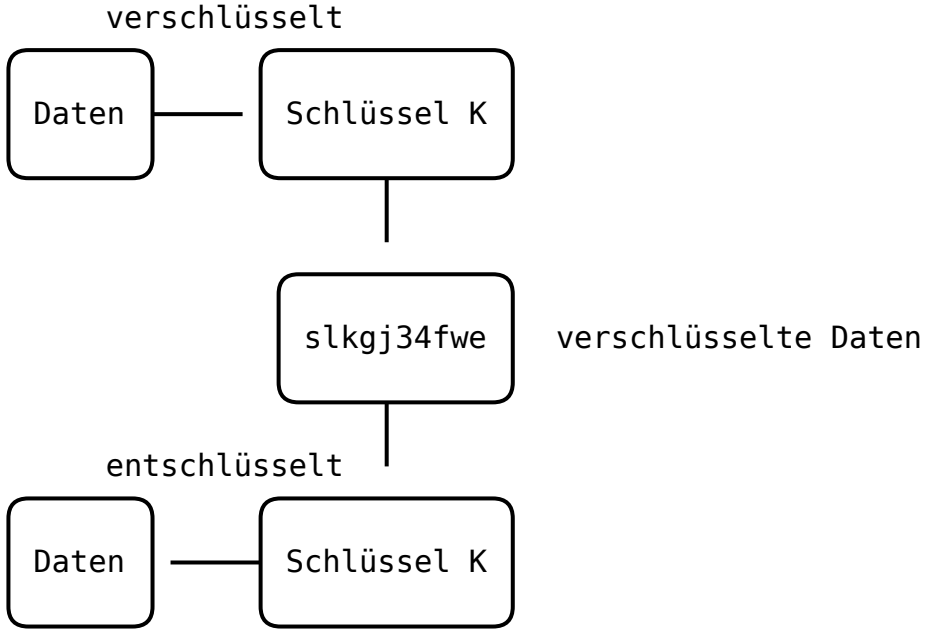
---

1. Symmetrische Verschlüsselung

Ein Schlüssel → für **Ver- und Entschlüsselung derselbe Schlüssel**.

Beispiele:

- AES (Standard heute)
- ChaCha20
- DES (veraltet)
- 3DES (veraltet)



Vorteile:

- sehr schnell
- ideal für große Datenmengen (Backups, VPN)

Nachteile:

- sicherer Schlüsselaustausch schwierig
- Schlüsselverlust = Datenverlust

---

2. Asymmetrische Verschlüsselung

Zwei Schlüssel:

- **Public Key** (öffentlicher Schlüssel)
- **Private Key** (geheimer Schlüssel)

Was der eine verschlüsselt, kann nur der andere entschlüsseln.

Beispiele:

- RSA
- ECC (moderne Alternative, schneller)
- Ed25519 (Shadowsocks, SSH etc.)

```
Public Key  → verschlüsselt
Private Key → entschlüsselt
```

Vorteile:

- einfacher Schlüsseltausch
- Grundlage für TLS, Zertifikate, digitale Signaturen

Nachteile:

- langsamer als symmetrische Verfahren

---

3. Digitale Signaturen & Integrität

Mit Private Key signieren → mit Public Key prüfen.

Schützt vor:

- Manipulation
- Identitätsdiebstahl
- gefälschten Updates

Beispiel:

- apt-get prüft digitale Signaturen

```
Private Key → Signatur
Public Key → Prüfung: gültig?
```

---

4. Hashing (wichtig für Passwörter)

Hash = Einwegfunktion → nicht rückrechenbar.

Beispiele:

- SHA-256 (stark)
- SHA-1 (veraltet)
- MD5 (gebrochen)

Moderne Passwort-Hashverfahren:

- Argon2id (Standard!)
- bcrypt
- scrypt

Eigenschaften:

- kein Entschlüsseln möglich
- kleine Änderungen → komplett anderer Hash

---

5. TLS & HTTPS

TLS (Transport Layer Security) verschlüsselt Verbindungen im Internet.
HTTPS = HTTP + TLS.

Schützt:

- Vertraulichkeit
- Integrität
- Authentizität des Servers

Ablauf in Kurzform:

1) Client verbindet sich
2) Server sendet Zertifikat
3) Client prüft Signatur & Gültigkeit
4) Es wird ein symmetrischer Sitzungsschlüssel ausgehandelt
5) Daten fließen verschlüsselt

```
Browser → TLS Handshake → Server
        ← Zertifikat
```

---

6. Zertifikate

Ein Zertifikat enthält:

- Public Key des Servers
- Name / Domain
- Gültigkeitszeitraum
- Signatur der CA
- ggf. zusätzliche Informationen (SANs)

Wichtig:

- Private Key bleibt IMMER geheim
- Zertifikate laufen ab (LE: 90 Tage)
- Chain of Trust muss vollständig sein

Arten von Zertifikaten:

DV – Domain Validated (Standard)

- bestätigt Domain-Besitz
- einfache Validierung (z. B. DNS-01 von LE)

OV – Organization Validated

- bestätigt zusätzlich Unternehmen

EV – Extended Validation

- höchste Validierungsstufe
- wird heute kaum noch genutzt

---

7. Public-Key-Infrastruktur (PKI)

Eine PKI ist ein Vertrauensmodell basierend auf Zertifikaten.

Bestandteile:

- **CA (Certificate Authority)** – stellt Zertifikate aus
- **Intermediate CA** – bildet Kette
- **Root CA** – höchstes Vertrauen
- **CRL** – Sperrliste
- **OCSP** – Online-Statusprüfung

ASCII: Chain of Trust

```
Root CA
  ↓ signiert
Intermediate CA
  ↓ signiert
Server Zertifikat
```

Der Client vertraut → Root CA → Intermediate → Server
→ wenn eine Stufe fehlt: „Zertifikat nicht vertrauenswürdig“.

---

8. Zertifikatsarten nach Einsatz

Server-Zertifikate

- TLS für Webserver (HTTPS)
- Mailserver (IMAP/SMTP/TLS)
- LDAP over TLS (StartTLS)

Client-Zertifikate

- Zugriff auf VPN
- Maschinenauthentifizierung (z. B. RADIUS)

Code-Signing Zertifikate

- Signieren von Software / Treibern

Dokumentensignatur

- PDF-Signaturen
- elektronische Unterschriften (LibreSign, eIDAS)

---

9. Zertifikatsvalidierung

Ein Client prüft:

- Ist die CA vertrauenswürdig?
- Passt die Domain zum Zertifikat?
- Ist das Zertifikat abgelaufen?
- Wurde es widerrufen?
- Ist die Chain vollständig?

---

10. Moderne Verschlüsselung in der Praxis

Webserver

- HTTPS per TLS 1.2+
- Let’s Encrypt via ACME (Traefik)

Mailserver

- STARTTLS
  * SMTPS (465)
  * DANE optional

VPN

- WireGuard (modern) → Curve25519
  * OpenVPN → TLS

Container & Microservices

- interne TLS-Kommunikation
  * mTLS (Mutual TLS) in Service Meshes (z. B. Istio)

Backups

- AES-Verschlüsselung
  * GPG für signierte Archive

—

11. Best Practices für Verschlüsselung

1) Nur moderne Protokolle verwenden

- TLS 1.2 / 1.3
  * keine alten Ciphers (RC4, 3DES, MD5, SHA1)

2) Private Keys gut schützen

- 600-Rechte
  * niemals teilen
  * nie per E-Mail versenden

3) Zertifikate automatisch erneuern

- Let’s Encrypt (ACME)

* Cronjobs oder Traefik-Auto-Renewal

4) HSTS aktivieren

- Browser erzwingt HTTPS

5) Passwort-Hashverfahren modern halten

- Argon2id
  * bcrypt

6) Nutzung von MFA erzwingen

7) Backups verschlüsselt speichern

8) interne TLS-Verbindungen prüfen

- LDAP StartTLS
  * Datenbankverbindungen (MariaDB SSL)

—

Zusammenfassung

- Symmetrisch = schnell, gleicher Schlüssel
  - Asymmetrisch = Public/Private Key, Grundlage für TLS
    * Hashes dienen der Integrität & Passwortspeicherung
    * TLS verschlüsselt Web-, Mail- und API-Verkehr
    * Zertifikate basieren auf PKI & Chain of Trust
    * Let’s Encrypt ermöglicht automatische Zertifikate
    * Moderne Sicherheit = starke Verschlüsselung + sichere Schlüsselverwaltung