1 Public Key Infrastrukturen (PKI)

Verschlüsselung allein reicht für sichere Kommunikation nicht aus. Ein Angreifer könnte sich in den Kanal einklinken (Man-in-the-Middle). Daher ist die **Authentifikation** des Kommunikationspartners essenziell.

- **Das Problem:** Woher weiß ich, dass der öffentliche Schlüssel (Public Key), den ich erhalte, wirklich zu der Person/Webseite gehört, mit der ich kommunizieren will?
- **Die Lösung:** Eine Infrastruktur, die Schlüssel an Identitäten bindet.

1.1 Vertrauensmodelle (Trust Models)

Es gibt drei fundamentale Ansätze, um Vertrauen in öffentliche Schlüssel zu etablieren.

1.1.1 1. Direct Trust

Dies ist das einfachste Modell, bei dem Schlüssel direkt zwischen den Parteien ausgetauscht werden.

Direct Trust

Direkter, manueller Austausch von öffentlichen Schlüsseln oder Fingerprints. Vertrauen entsteht durch persönliche Überprüfung.

- **Beispiel SSH:** Beim ersten Verbinden ("Trust on First Use" - TOFU) zeigt der Client den Fingerprint des Servers.
- *Warnmeldung:* "The authenticity of host... can't be established." Der Nutzer muss den Fingerprint manuell verifizieren (z.B. über einen sicheren zweiten Kanal).
- **Nachteil:** Skaliert nicht. Man kann nicht mit jedem Webseiten-Betreiber der Welt persönlich Schlüssel tauschen.

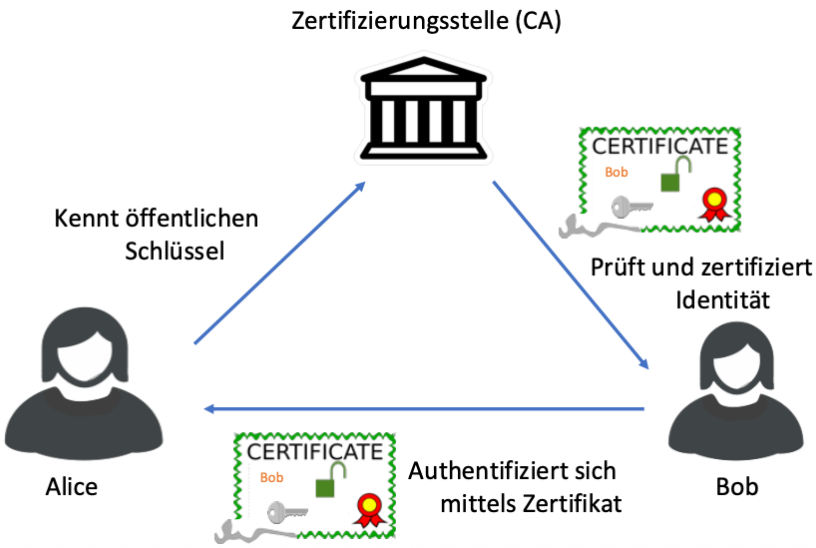
1.1.2 2. Hierarchical Trust (WebPKI)

Dieses Modell nutzt eine vertrauenswürdige dritte Partei, die als Mittelsmann fungiert. Dies ist der Standard im Web (HTTPS).

Certificate Authority (CA)

Eine **Zertifizierungsstelle** (CA) prüft die Identität eines Antragstellers und signiert dessen öffentlichen Schlüssel kryptographisch. Diese Signatur bildet ein Zertifikat.

- **Funktionsweise:** Alice vertraut der CA. Die CA bürgt für Bob. Folglich vertraut Alice Bob.
- **Verteilung:** Die Root-Zertifikate der CAs sind im Betriebssystem oder Browser vorinstalliert (Trust Store).



1.1.3 3. Web of Trust

Ein dezentraler Ansatz, der oft bei PGP (E-Mail-Verschlüsselung) genutzt wird.

- Es gibt keine zentralen CAs.
- Teilnehmer signieren gegenseitig ihre Schlüssel (**Keysigning**).
- **Transitives Vertrauen:** Alice vertraut Bob. Bob hat Carols Schlüssel signiert. Wenn Alice Bobs Urteilsvermögen vertraut, vertraut sie auch Carol.
- **Keyserver:** Dienen als Telefonbuch zum Hochladen von Schlüsseln und Signaturen.

1.2 WebPKI im Detail

Das WebPKI-System bildet das Rückgrat des sicheren Browsings.

1.2.1 Chain of Trust (Zertifikatskette)

Browser vertrauen einer Menge an **Root CAs** (Wurzelzertifikate). Eine Webseite sendet jedoch meist nicht das Root-Zertifikat, sondern eine Kette:

1. **Root Certificate:** Selbstsigniert, im Browser hinterlegt (Trust Anchor).
2. **Intermediate Certificate:** Von der Root CA (oder einer anderen Intermediate) signiert.
3. **Leaf Certificate (End-Entity):** Das eigentliche Zertifikat der Webseite, signiert von der Intermediate CA.

Der Browser validiert die Signaturen vom Leaf bis hoch zur Root.

1.2.2 Validierungsmethoden für Zertifikate

Bevor eine CA ein Zertifikat ausstellt, muss sie prüfen, ob der Antragsteller berechtigt ist.

1. **Domain Validation (DV):**
   - Prüft nur die technische Kontrolle über die Domain.
   - *Methoden:*
     - **HTTP Challenge:** CA gibt einen Token, Server muss ihn unter `http://domain/.well-known/acme-challenge/` bereitstellen.
     - **DNS Challenge:** Token muss als TXT-Record im DNS hinterlegt werden.

  - **Email Challenge:** Bestätigungslink an `admin@domain.com`.
- *Vorteil:* Schnell, automatisierbar (z.B. Let's Encrypt), kostenlos.
- *Nachteil:* Keine Prüfung der Identität der Firma dahinter.

2. **Organization Validation (OV):** Prüft zusätzlich, ob die Organisation existiert und rechtmäßiger Besitzer der Domain ist.
3. **Extended Validation (EV):** Sehr strenge Prüfung offizieller Dokumente/Register. Früher durch "grüne Adressleiste" im Browser angezeigt (heute meist entfernt, da Nutzer den Unterschied nicht verstehen).

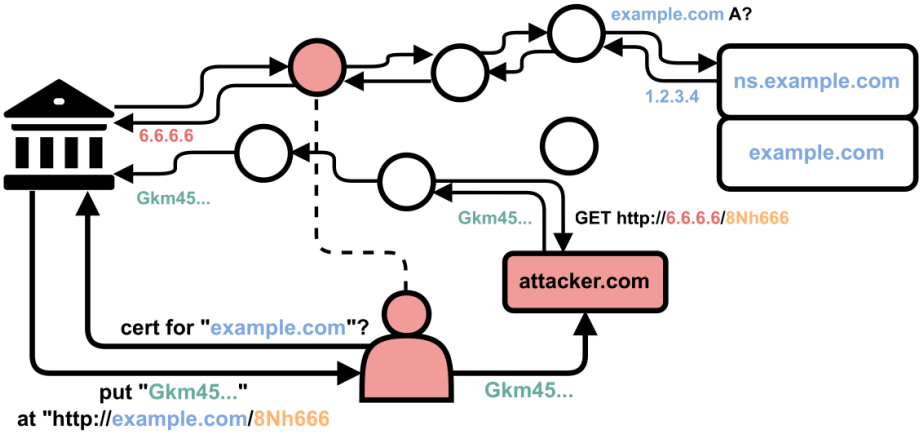
1.3 Angriffe auf die WebPKI

Das System ist nur so sicher wie das schwächste Glied (die CA) und der Validierungsprozess.

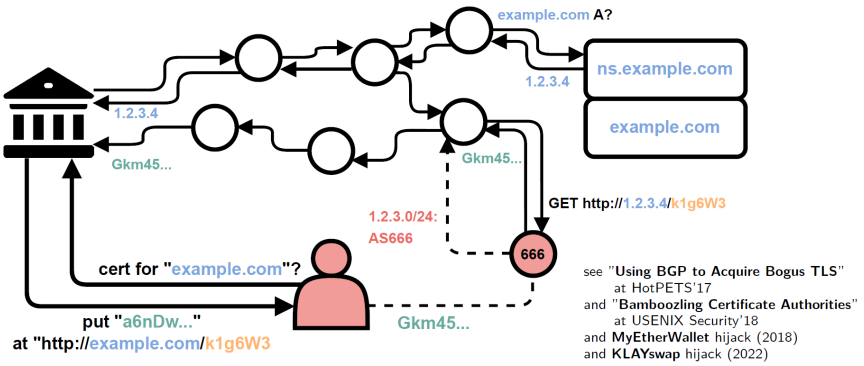
1.3.1 Angriffe auf Domain Validation

Da DV oft automatisiert abläuft, versuchen Angreifer, die Validierung (Challenge) zu manipulieren, um Zertifikate für fremde Domains zu erhalten.

- **Netzwerk-Angriffe:** Wenn der Angreifer den Traffic zwischen der CA und dem Server des Opfers abfangen kann (Man-in-the-Middle).
- **BGP Hijacking:** Der Angreifer lenkt den Internetverkehr für die IP-Adresse des Opfers auf seinen eigenen Server um. Die CA verbindet sich zur Überprüfung (HTTP Challenge) mit dem Angreifer statt dem Opfer.
- **DNS Cache Poisoning:** Der Angreifer manipuliert die DNS-Antworten, die die CA erhält, sodass die Domain auf die IP des Angreifers zeigt.



On-path position can be achieved via **BGP hijack**



Gegenmaßnahme: Verteilte Validierung Die CA sollte die Validierung (z.B. den HTTP-Request) nicht nur von einem Standort ausführen, sondern von **verteilten Validatoren** (mehrere Perspektiven weltweit). Ein lokaler BGP-Hijack oder DNS-Poisoning würde so auffallen, da nicht alle Validatoren auf den Angreifer umgeleitet werden.

1.4 Certificate Transparency (CT)

Ein großes Problem der klassischen PKI war, dass eine korrumpierte CA unbemerkt falsche Zertifikate (z.B. für google.com) ausstellen konnte.

Certificate Transparency

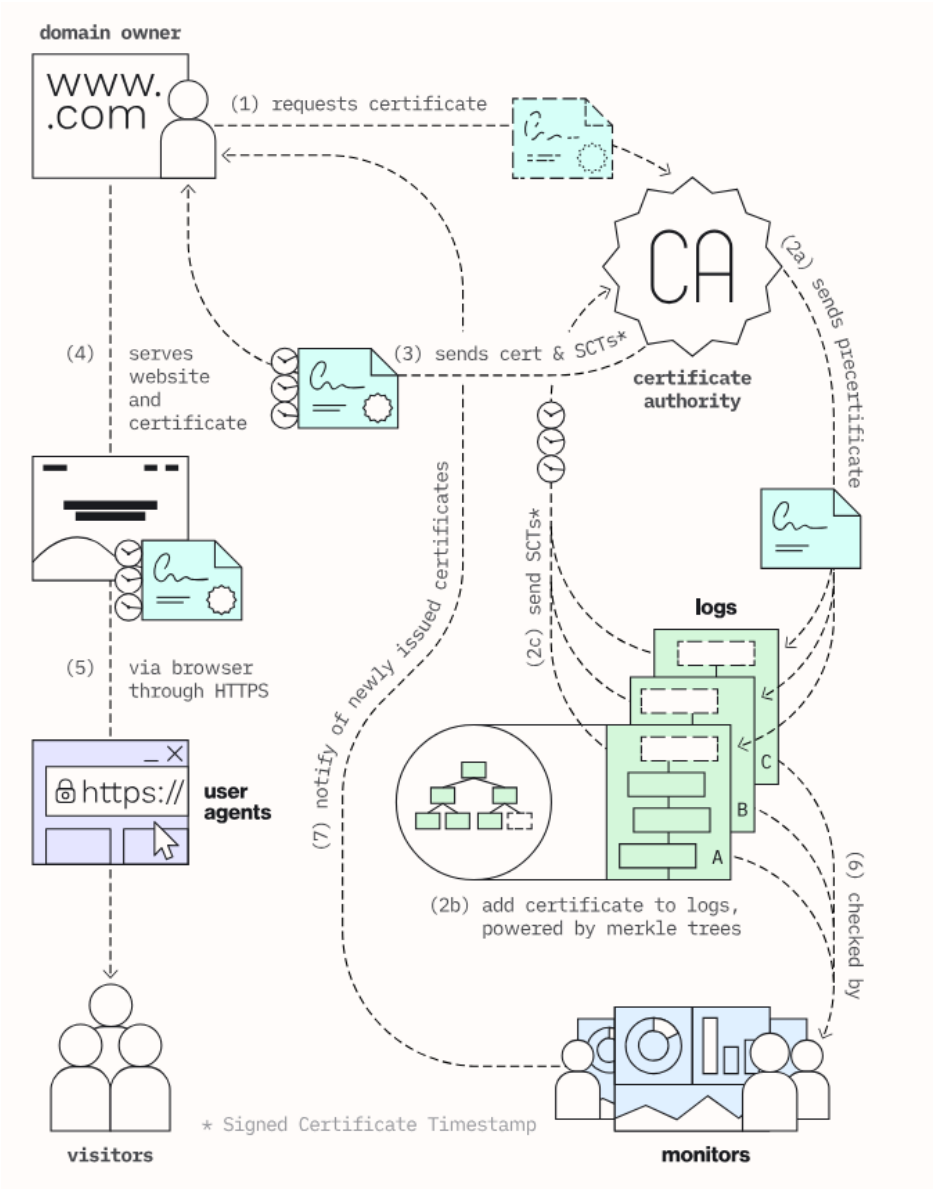
Ein System öffentlich einsehbarer, manipulationssicherer Logs, in denen alle ausgestellten Zertifikate verzeichnet werden müssen.

1.4.1 Funktionsweise

1. CA reicht ein "Pre-Certificate" bei einem CT-Log ein.
2. Das Log antwortet mit einem **SCT** (Signed Certificate Timestamp). Das ist eine kryptographische Quittung: "Ich habe dieses Zertifikat gesehen und werde es loggen."
3. Das finale Zertifikat enthält diesen SCT.
4. Browser (wie Chrome) lehnen Zertifikate ohne SCT oft ab.

1.4.2 Komponenten

- **Logs:** Nutzen **Merkle Trees**, um Append-Only-Eigenschaften zu garantieren (Inhalte können nicht nachträglich gelöscht/geändert werden).
- **Monitors:** Dienste, die die Logs überwachen und Domain-Inhaber warnen, wenn unerwartet ein Zertifikat für ihre Domain auftaucht.



1.5 Zertifikatsstruktur (X.509)

Zertifikate folgen dem X.509 Standard und werden mittels **ASN.1** (Abstract Syntax Notation One) beschrieben.

1.5.1 Wichtige Felder

- **Subject:** Für wen ist das Zertifikat? (Common Name / Domain).
- **Issuer:** Wer hat es ausgestellt? (Die CA).
- **Validity:** `Not Before` und `Not After` (Gültigkeitsdauer).
- **Subject Public Key Info:** Der eigentliche Schlüssel und der Algorithmus (z.B. RSA, ECC).
- **Signature Algorithm:** Algorithmus, mit dem die CA unterschrieben hat (z.B. SHA256withRSA).

1.5.2 Extensions

- **Key Usage / Basic Constraints:** Legt fest, was mit dem Zertifikat getan werden darf.

  - Wichtig: `CA: FALSE` verhindert, dass ein normales Webseiten-Zertifikat genutzt wird, um weitere Zertifikate zu signieren (verhindert, dass jeder User zur CA wird).
- **Subject Alternative Name (SAN):** Hier werden alle gültigen Domains (auch Subdomains) aufgelistet.

1.5.3 Widerruf (Revocation) und Gültigkeit

Wenn ein Private Key gestohlen wird, muss das Zertifikat ungültig gemacht werden.

- **CRL (Certificate Revocation List):** Liste gesperrter Zertifikate. Wird oft nicht frisch heruntergeladen (zu groß).
- **OCSP (Online Certificate Status Protocol):** Live-Abfrage bei der CA. Problem: "Fail Open" (wenn CA nicht erreichbar, akzeptiert der Browser oft trotzdem).
- **Trend:** Verkürzung der Laufzeiten (Vorschlag von Google: 90 Tage). Ersetzt komplexes Revocation-Management durch häufige Erneuerung (Automation nötig).

1.6 Alternative: DNSSEC / DANE

Statt einer WebPKI mit hunderten CAs könnte man die Hierarchie des DNS nutzen.

- **Idee:** Die "Chain of Trust" folgt der DNS-Delegation (. → .de → tu-darmstadt.de).
- **TLSA Records:** Der Fingerprint des Zertifikats wird direkt im DNS hinterlegt und per DNSSEC signiert.
- **Vorteil:** Kein "CA-Markt", logische Struktur.
- **Nachteil:** Ein einziger **Root Key** (verwaltet von der ICANN) ist der "Single Point of Failure".
- **Root Signing Ceremony:** Hochsichere Zeremonie zur Verwaltung dieses Schlüssels (Safe deposit boxes, Zeugen, physische Sicherheit).

1.7 Angriffe auf Nutzer (Typosquatting)

Selbst bei perfekter Technik (valides Zertifikat) kann der Nutzer getäuscht werden, wenn er auf der falschen Seite landet.

- **Homograph Attack:** Nutzung ähnlich aussehender Zeichen (z.B. kyrillisches 'a' statt lateinisches 'a').
- **Combo-Squatting:** Zusätze im Namen (`paypal-support.com`).
- **Homophon:** Ähnlich klingende Namen.

Hinweis: Da WebPKI (DV) nur den Besitz der Domain prüft, erhält auch der Angreifer ein völlig valides, "grünes" Schloss-Symbol für seine Phishing-Seite `goggle.com`.