
Computersystemsicherheit

Niclas Kusenbach

LaTeX version:  SCHOUTER

Table of Contents

Contents

1 Kryptografie	3	2.4.2 Challenge-Response-Verfahren (CHAP)	16
1.1 Symmetrische Kryptografie	3	2.5 Single Sign On (SSO)	17
1.1.1 Blockchiffren	3	2.5.1 Kerberos (Ein SSO-Protokoll) . . .	17
1.2 Kryptografische Hashfunktionen	7	2.6 Autorisierung (Zugriffskontrollmodelle) . .	18
1.3 Asymmetrische Kryptografie	7	2.6.1 Discretionary Access Control (DAC) .	18
1.3.1 Grundlagen	7	2.6.2 Role-based Access Control (RBAC) .	19
1.4 RSA-Kryptosystem	8	2.6.3 Mandatory Access Control (MAC) .	19
1.4.1 Schlüsselerzeugung	8	3 Netzwerkgrundlagen & Sicherheit	20
1.4.2 Verschlüsselung und Entschlüsselung .	9	3.1 Einführung und Modelle	20
1.5 ElGamal-Kryptosystem	9	3.1.1 OSI-Modell (Open System Inter-connection)	20
1.5.1 RSA-Signaturen	10	3.1.2 TCP/IP Modell vs. OSI	20
1.5.2 Sicherheitsbegriffe für Signaturen .	11	3.2 Die Schichten im Detail	20
1.6 Schlüsselverteilung und Schlüsselaustausch	11	3.2.1 Layer 1: Physical Layer	21
1.6.1 Schlüsselarten	11	3.2.2 Layer 2: Data Link Layer	21
1.6.2 Schlüsselaustauschprotokolle . . .	11	3.2.3 Layer 3: Network Layer	21
1.6.3 Diffie-Hellman-Schlüsselaustausch .	12	3.2.4 Layer 4: Transport Layer	21
2 Authentifizierung	13	3.2.5 Layer 5-7: Höhere Schichten	22
2.1 Grundlagen	13	3.3 Angriffsmodelle im Netzwerk	22
2.1.1 Begriffsdefinitionen	13	3.4 Netzwerkprotokolle und spezifische Angriffe	23
2.1.2 Drei Ansätze der Authentisierung .	13	3.4.1 ARP (Address Resolution Protocol) .	23
2.2 Authentisierung durch Besitz	13	3.4.2 DHCP (Dynamic Host Configuration Protocol)	23
2.2.1 Beispiel: ATHENE KARTE (SmartCard)	13	3.4.3 ICMP und (D)DoS Angriffe	24
2.2.2 Challenge-Response-Verfahren (mit Kryptoprozessor)	14	3.5 Netzwerkschutzmechanismen	25
2.2.3 Probleme	14	3.5.1 Firewall	25
2.3 Authentisierung durch Merkmale (Biometrie)	14	3.5.2 IDS vs. IPS	25
2.3.1 Anforderungen an biometrische Merkmale	14	4 Border Gateway Protocol (BGP)	26
2.3.2 Prozesse und Fehlerraten	15	4.1 Grundlagen des Routings: LAN vs. Internet	26
2.3.3 Beispiel: Fingerabdruck	15	4.2 Lokale Netzwerke (LAN)	26
2.3.4 Probleme der Biometrie	15	4.3 Internet Routing	26
2.4 Authentisierung durch Wissen	16	4.4 Border Gateway Protocol (BGP)	26
2.4.1 Passwörter	16	4.5 Funktionsweise von BGP	26
		4.6 BGP Varianten	26
		4.7 Routing-Entscheidungen	27

4.8	Angriffe gegen BGP	27
4.9	BGP Hijacking	27
4.10	Weitere Angriffsvektoren	27
4.11	Ziele der Angriffe	27
4.12	Reale Angriffsbeispiele	28
4.13	Gegenmaßnahmen	28
4.14	Organisatorische Basis	28
4.15	Internet Routing Registry (IRR)	28
4.16	Resource Public Key Infrastructure (RPKI)	28
4.17	BGPsec	29
5	Domain Name System	30
5.1	Hierarchie und Namensraum	30
5.2	Domain vs. Zone	30
5.3	DNS Resource Records (RR)	30
5.3.1	Wichtige Record-Typen	31
5.4	Nachrichtenübermittlung und Auflösung	31
5.4.1	Kommunikation	31
5.4.2	Server-Typen und Rollen	31
5.4.3	Auflösungsmethoden	31
5.5	DNS Cache Poisoning	32
5.5.1	Angriffsmethoden	32
5.5.2	Bailiwick-Regel (Gegenmaßnahme)	32
5.6	Gegenmaßnahme: DNSSEC	33
5.6.1	Funktionsweise	33
5.6.2	Neue Record-Typen	33
5.6.3	Problem: Zone Enumeration	33
5.7	Transport-Sicherheit (Privacy)	33
5.8	Weitere Angriffe auf DNS-Infrastruktur	33
5.8.1	DNS Amplification DDoS	33
5.8.2	DNS Tunneling	34
5.9	DNS-basierte Sicherheitsmechanismen für E-Mail	34
5.9.1	SPF (Sender Policy Framework)	34
5.9.2	Weitere Mechanismen	34

1 Kryptografie

Die Kryptografie wird in drei Hauptkategorien unterteilt:

1. Symmetrische Kryptografie
2. Hashfunktionen
3. Asymmetrische Kryptografie

1.1 Symmetrische Kryptografie

Symmetrische Kryptografie ist eine Menge von kryptografischen Protokollen, bei der derselbe geheime Schlüssel für die Ver- und Entschlüsselung von Daten verwendet wird.

Symmetrische Kryptosysteme

Ein symmetrisches Kryptosystem ist ein 5-Tupel (M, K, C, e, d) bestehend aus:

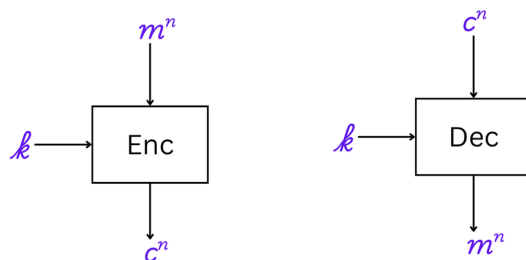
- einer Menge M von Klartexten,
- einer Menge K von Schlüsseln,
- einer Menge C von Chiffretexten,
- einer Verschlüsselungsfunktion $e : M \times K \rightarrow C$,
- einer Entschlüsselungsfunktion $d : C \times K \rightarrow M$,

so dass für alle Klartexte $m \in M$ und alle Schlüssel $k \in K$ gilt, dass $d(e(m, k), k) = m$.

1.1.1 Blockchiffren

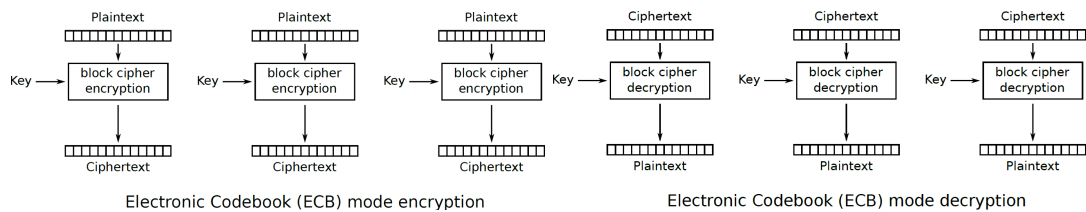
Definition

Blockchiffren sind Kryptosysteme, die nur Blöcke fester Länge verschlüsseln können.



- Ein Blockchiffre arbeitet auf einem Klartextblock der Länge b , um einen Chiffretextblock der Länge b zu erzeugen.
- Der gleiche Schlüssel kann mehrmals auf unterschiedliche Blöcke verwendet werden.
- Beispiele von Blockchiffren: AES, DES, 3DES, Serpent, Twofish, Blowfish, etc.

Electronic Code Book (ECB) Modus



Wenn die Blöcke nicht die Länge n haben, können trotzdem beliebige Nachrichten verschlüsselt werden, da eine **Auffüllfunktion** (Padding function) benutzt wird.

Bei vielen Padding-Verfahren wird *immer* ein Padding hinzugefügt, auch wenn die Nachricht bereits ein Vielfaches der Blocklänge n hat. Dies ist notwendig, damit die *unpad()*-Funktion eindeutig feststellen kann, wie viele Bytes entfernt werden müssen. Eine gute Auffüllfunktion sollte umkehrbar sein, d.h. es muss eine *unpad()*-Funktion geben mit $unpad(pad(x)) = x \quad \forall x \in M^*$.

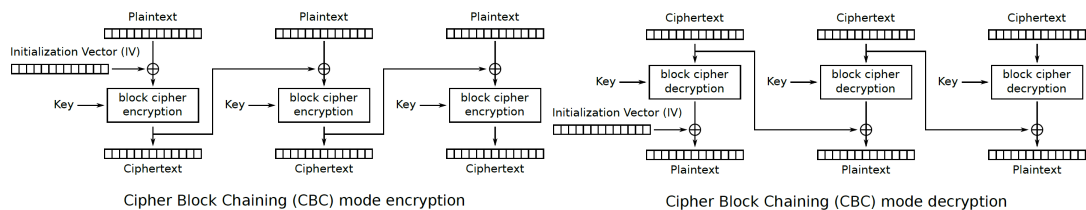
Vorteile:

- Unkomplizierte Bedienung. Jeder Block wird unabhängig bearbeitet.
- Parallelisierbarkeit von Ver- und Entschlüsselungsverfahren.
- Beschädigte Datenblöcke beeinflussen keine anderen Blöcke (Fehlertoleranz).

Nachteile:

- *Deterministisch*: Muster im Klartext sind sichtbar. Identische Klartextblöcke ergeben immer identische Chiffretextblöcke.
- *Keine Diffusion*: Kleine Änderungen im Klartext führen zu lokalisierten Änderungen im Geheimtext.

Cipher Block Chaining (CBC) Modus



Zur Formalisierung von CBC benötigen wir randomisierte Kryptosysteme. Der Zufallswert r wird hier als Initialisierungsvektor (IV) bezeichnet.

Randomisierte symmetrische Kryptosysteme

Ein randomisiertes symmetrisches Kryptosystem ist ein 5-Tupel (M, K, C, e, d) bestehend aus:

- einer Menge M von Klartexten,
- einer Menge K von Schlüsseln,
- einer Menge C von Chiffretexten,
- einer Menge R von Zufallswerten (z.B. IVs),
- einer Verschlüsselungsfunktion $e : M \times K \times R \rightarrow C$,
- einer Entschlüsselungsfunktion $d : C \times K \rightarrow M$,

(Anmerkung: Die Entschlüsselung d benötigt den IV r , dieser wird aber typischerweise als Teil des Chiffretextes C übermittelt und nicht als separater Zufallseingang für d selbst.)

Sei $r \in R$ der Initialisierungsvektor (IV). **Verschlüsselung**

$$e^*(x_0x_1 \dots x_n, k, r) = y_0y_1 \dots y_n \text{ mit } y_0 = e(x_0 \oplus r, k) \quad \text{und} \quad y_i = e(x_i \oplus y_{i-1}, k) \quad \text{für } i \geq 1$$

Entschlüsselung

$$d^*(y_0y_1 \dots y_n, k, r) = x_0x_1 \dots x_n \text{ mit } x_0 = d(y_0, k) \oplus r \quad \text{und} \quad x_i = d(y_i, k) \oplus y_{i-1} \quad \text{für } i \geq 1$$

- Zur Verschlüsselung muss ein Wert $r \in R$ (der IV) gewählt werden.
- Zufallswerte aus R (IVs) sind nicht geheim, sie können unverschlüsselt mit dem Chiffre gespeichert und verschickt werden (meist als erster Block).
- Wir wollen $e(x, k, r^1) \neq e(x, k, r^2)$ für $r^1 \neq r^2$.
- Wichtig für die Sicherheit ist, dass der IV (Zufallswert r) **einzigartig** (nie doppelt für denselben Schlüssel) und **unvorhersagbar** ist.
- Muster im Klartext sind im Chiffre nicht mehr erkennbar.
- Gleiche Klartextblöcke werden unterschiedlich verschlüsselt.
- Ein fehlerhafter Chiffreblock y_i führt nur zur fehlerhaften Entschlüsselung des aktuellen Blocks x_i und des unmittelbar nachfolgenden Blocks x_{i+1} .
- Verschlüsselung ist **nicht** parallelisierbar (sequenziell), Entschlüsselung ist parallelisierbar.

CBC Padding Oracle Attack **CBC ist anfällig für Padding-Oracle-Angriffe**

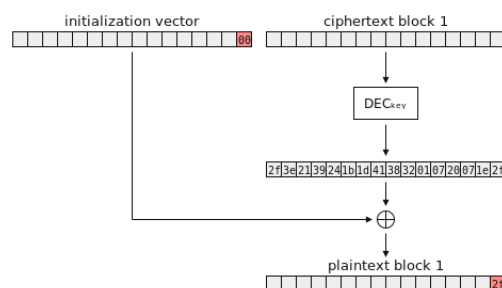
Ein solcher Angriff ermöglicht es, einen Geheimtext Schritt für Schritt zu entschlüsseln, ohne den Verschlüsselungsschlüssel zu kennen. Der Angreifer nutzt aus, wie ein Server auf fehlerhaftes Padding reagiert.

Der Angreifer:

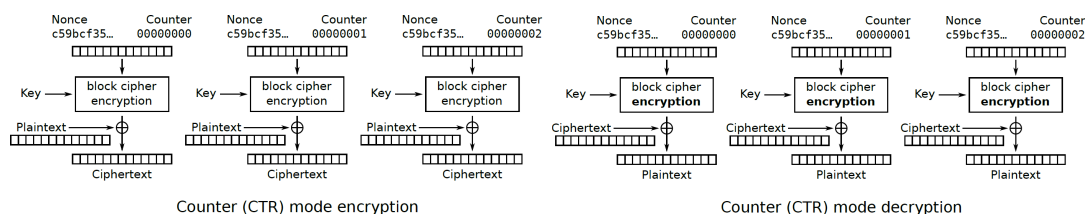
- hat keinen Zugriff auf den geheimen Schlüssel.
- ist in der Lage, gültige Chiffretexte abzufangen.
- ist in der Lage, modifizierte Versionen des Chiffretextes an das Orakel/Server zu senden und dessen Antworten zu beobachten.

Das Orakel (Server):

- muss ein überprüfbares Padding-Schema verwenden.
- muss dem Angreifer verraten, ob ein entschlüsselter Text ein gültiges (oder nicht) Padding hat. Dies kann geschehen durch:
 1. direkte Fehlermeldungen (z.B. HTTP 500) oder
 2. Side-Channel-Messungen (z.B. Unterschiede im Antwortverhalten).



Counter Mode (CTR) Modus



Um diesen Modus zu formalisieren, benötigen wir eine randomisierte Zählfunktion, die einen "Nonce" (Number used once) r verwendet.

- Ein randomisierter Zähler (Funktion $ctr(r, i)$) bildet einen Zufallswert (Nonce r) und eine natürliche Zahl i (Zähler) auf eine Bitkette fester Länge ab.
- Eine einfache Implementierung benutzt die Binärdarstellung der natürlichen Zahl (LSB- oder MSB-Kodierung) mit 0-Padding, konkateniert mit der Nonce.
- Ein randomisierter Zähler $ctr(r, \cdot)$ sollte injektiv sein (für ein festes r). Man sollte die Periode (Wiederholung) so lang wie möglich wählen.
- Die Nonce r muss für denselben Schlüssel **nie** wiederverwendet werden.

Verschlüsselung:

$$e^*(x_0x_1 \dots x_n, k, r) = y_0y_1 \dots y_n \text{ mit } y_i = e(ctr(r, i), k) \oplus x_i$$

Entschlüsselung:

$$d^*(y_0y_1 \dots y_n, k, r) = x_0x_1 \dots x_n \text{ mit } x_i = e(ctr(r, i), k) \oplus y_i$$

Der CTR Modus unterscheidet sich stark von den vorher betrachteten Betriebsmodi:

- Ver- und Entschlüsselung nutzen beide die Verschlüsselungsfunktion e der Blockchiffre; die Entschlüsselungsfunktion d selbst wird nicht benötigt.
- Ver- und Entschlüsselung sind identisch (XOR mit dem Keystream).
- Die Berechnung des Keystreams $e(ctr(r, i), k)$ ist unabhängig vom zu verschlüsselnden Text.
- Ver- und Entschlüsselung können vollständig parallelisiert werden.
- CTR ist eine One-Time-Pad-Konstruktion, bei der die Blockchiffre als Pseudozufallsgenerator (Keystream-Generator) dient.

Advanced Encryption Standard (AES)

- Blocklänge ist 128 bereits
- AES-Schlüssel können 128, 192, oder 256 bits lang sein

Sicherheit

- AES ist sicher solange die Implementierung und dazugehörige Systeme richtig konfiguriert sind (s. CBC Padding Attack)
- Schwache Schlüssel und IV-Generierung kann die Sicherheit von AES gefährden
- Side-channel Angriffe wie cache-timing und power analysis können verwendet werden, um den Schlüssel abzuleiten

Gegenmaßnahmen

- Konstantzeit-Implementierung (gegen Timing Angriffe): Ausführungszeit von Code soll unabhängig von den verarbeiteten geheimen Daten sein
-

Stromchiffren Stromchiffren können beliebig lange Bitketten verschlüsseln. Dabei sind Klar- und Chiffretexte beliebiger Länge, nur der Schlüssel hat eine feste Länge. Aus dem Schlüssel wird ein pseudozufälliger Schlüsselstrom erzeugt. Pseudozufallszahlen hängen von ihren Startparametern ab (seed) ab - gleiche Parameter liefern gleiche Zufallszahlen. Ver- und Entschlüsselung ist ein bitweise exklusives oder (XOR) mit dem Schlüsselstrom. Ein Kryptosystem heißt Stromchiffre, wenn es eine Funktion $keystream(x, z) = |x|$ gibt, so dass $e(x, y) = d(x, z) = x \oplus keystream(x, y)$. Die Funktion $keystream$ nennen wir Schlüsselstromgenerator und ihren Funktionswert Schlüsselstrom.

- Keystream sollte ein Pseudozufallszahlengenerator sein
- Keystream kann unabhängig vom Inhalt der ersten Variable sein, also $keystream(x_1, k) = keystream(x_2, k)$ für beliebige x_1 und x_2 mit $|x_1| = |x_2|$
- Falls der Schlüsselstrom sich wiederholt, ist die Stromchiffre nicht mehr sicher

ChaCha20 ist eine moderne Stromchiffre, die als Alternative zu AES entwickelt wurde.

1.2 Kryptografische Hashfunktionen

Eine Hashfunktion ist ein Algorithmus, der eine Eingabe beliebiger Größe in einen Hashwert mit einer festen Länge umwandelt. Hashfunktionen sind deterministisch, erlauben eine schnelle Berechnung und bieten Integritätsschutz (Änderung der Eingabe ändert den Hash) Eigenschaften einer Hashfunktion:

1. Pre-Image Resistance

Bei gegebenem Hashwert h ist es rechnerisch unmöglich, die ursprüngliche Nachricht m zu finden, so dass $H(m) = h$.

2. Second-Image Resistance

Bei gegebener Nachricht m_1 ist es rechnerisch unmöglich, eine andere Nachricht m_2 zu finden, die denselben Hashwert erzeugt, so dass $H(m_1) = H(m_2)$

3. Collision Resistance

Es ist rechnerisch unmöglich irgendwelche zwei unterschiedlichen Nachrichten m_1 und m_2 zu finden, die denselben Hashwert erzeugen, so dass $H(m_1) = H(m_2)$

Hashfunktion	Output	Sicherheit	Anwendung
MD5	128 Bits	Unsicher	X
SHA-1	160 Bits	Unsicher seit 2017	X
SHA-256	256 Bits	Sicher	TLS/SSL, hashing, Blockchain
SHA-3/Keccak	224/256/384/512 Bits	Sicher	Ähnlich wie SHA-2 (aber langsamer ohne Hardware Unterstützung)

Table 1: Vergleich von Hashfunktionen

1.3 Asymmetrische Kryptografie

1.3.1 Grundlagen

Bei einem asymmetrischem Kryptosystem gibt es verschiedene Schlüssel.

1. Öffentliche Schlüssel werden frei für alle interessierten Mitredner veröffentlicht.
2. Eine geheime Nachricht muss erst mit dem öffentlichen Schlüssel verschlüsselt an den Empfänger zugeschickt werden.

Asymmetrische Kryptografie

Ein asymmetrisches Kryptosystem ist ein 7-Tupel $(M, K_s, K_p, K, C, e, d)$ bestehend aus

- einer Menge M von Klartexten,
- einer Menge K_s von geheimen/privaten Schlüsseln,
- einer Menge K_p von öffentlichen Schlüsseln
- einer Menge $K \subset K_s \times K_p$ von Schlüsselpaaren,
- einer Menge C von Chiffretexten,
- einer Verschlüsselungsfunktion $e : M \times K_p \rightarrow C$,
- einer Entschlüsselungsfunktion $d : C \times K_s \rightarrow M$,

so dass für alle Klartexte $m \in M$ und alle Schlüsselpaare $(s, p) \in K$ gilt, dass $d(e(m, p), s) = m$.

Prinzip:

- Verschlüsselung mit **öffentlichem Schlüssel** des Empfängers
- Entschlüsselung mit **privatem Schlüssel** des Empfängers
- Kein vorheriger Schlüsselaustausch nötig (im Gegensatz zu symmetrischer Kryptographie)

1.4 RSA-Kryptosystem

Idee

Das RSA-Kryptosystem (nach **Rivest, Shamir, Adleman**, 1977) ist das bekannteste Verfahren der asymmetrischen Kryptographie. Es basiert auf der Schwierigkeit, eine große Zahl $n = p \cdot q$ in ihre Primfaktoren zu zerlegen.

1.4.1 Schlüsselerzeugung

1. Wähle zwei große Primzahlen p, q mit $p \neq q$.
2. Berechne das **RSA-Modul**:

$$n = p \cdot q$$

3. Berechne die **Eulersche Totientfunktion**:

$$\varphi(n) = (p-1)(q-1)$$

Diese gibt die Anzahl der zu n teilerfremden Zahlen an.

4. Wähle den **Verschlüsselungsexponenten** e mit

$$1 < e < \varphi(n), \quad \gcd(e, \varphi(n)) = 1$$

(d. h. e und $\varphi(n)$ sind teilerfremd).

5. Berechne den **Entschlüsselungsexponenten** d als **modulares Inverses** von e :

$$d \cdot e \equiv 1 \pmod{\varphi(n)}$$

Dies geschieht mit dem **erweiterten Euklidischen Algorithmus**.

Beispiel: Erweiterter Euklidischer Algorithmus

Gegeben $e = 17$ und $\varphi(n) = 3120$:
Wir suchen d mit $17d \equiv 1 \pmod{3120}$.

$$3120 = 17 \cdot 183 + 9$$

$$17 = 9 \cdot 1 + 8$$

$$9 = 8 \cdot 1 + 1$$

$$8 = 1 \cdot 8 + 0$$

Rückwärtseinsetzen:

$$1 = 9 - 8 = 9 - (17 - 9) = 2 \cdot 9 - 17 = 2(3120 - 17 \cdot 183) - 17 = 2 \cdot 3120 - 367 \cdot 17$$

Daraus folgt:

$$d \equiv -367 \equiv 2753 \pmod{3120}$$

Ergebnis: $d = 2753$.

Damit gilt:

Öffentlicher Schlüssel: (e, n) , Privater Schlüssel: (d, n)

1.4.2 Verschlüsselung und Entschlüsselung

- **Verschlüsselung:**

$$c = m^e \bmod n$$

- **Entschlüsselung:**

$$m = c^d \bmod n$$

Sicherheit von RSA:

- Basierend auf **Faktorisierungsproblem**: Schwierigkeit, n in p und q zu zerlegen
- Kenntnis von p , q oder $\varphi(n)$ ermöglicht Berechnung von d
- **Multiplikative Eigenschaft**: $(m_1^e \bmod n) \cdot (m_2^e \bmod n) \bmod n = (m_1 m_2)^e \bmod n \rightarrow$ problematisch für Sicherheit
- Durch **Quantencomputer** (Shor-Algorithmus) brüchbar
- p und q sollten groß und ähnlich groß sein (gleiche Bitlänge)

1.5 ElGamal-Kryptosystem

ElGamal Schlüsselerzeugung (Alice)

- Wähle eine **zyklische Gruppe** $G = (\mathbb{G}, \circ, e)$ mit großem Primzahlmodulus (z. B. \mathbb{Z}_p^\times) und einem **Erzeuger** g
- Wähle einen privaten Exponenten $a \in \{1, \dots, \text{ord}(g) - 1\}$ und berechne $A = g^a \bmod p$
- **Privater Schlüssel**: a
- **Öffentlicher Schlüssel**: (G, g, A)

Verschlüsselung (an Alice):

- Wähle zufällig $r \in \{1, \dots, \text{ord}(g) - 1\}$
- Berechne $R = g^r \bmod p$
- Berechne gemeinsamen Schlüssel $K = A^r = (g^a)^r = g^{ar} \bmod p$
- Berechne $C = (R, m \cdot K \bmod p)$ und sende C

Entschlüsselung (Alice):

- Berechne $K = R^a = (g^r)^a = g^{ra} \bmod p$
- Berechne das Inverse $K^{-1} \bmod p$
- Entschlüssele $m = C_2 \cdot K^{-1} \bmod p$

Sicherheit von ElGamal:

- Sicherheit basiert auf dem **Diskreten Logarithmusproblem (DLP)**: gegeben (g, g^a) ist a schwer zu bestimmen
- Angreifbar durch Quantencomputer (Shor-Algorithmus)
- **Probabilistisches Verfahren** durch Zufallswert $r \rightarrow$ semantisch sicher, wenn das **Decisional Diffie-Hellman-Problem (DDH)** schwer ist
- Aus einem gültigen Chiffre (c_1, c_2) lässt sich leicht $(c_1, g \cdot c_2)$ für beliebiges $g \in G$ konstruieren — **nicht deterministisch**, daher keine Wiederverwendung von r

Hinweis: Bei allen Potenzoperationen und Multiplikationen muss stets das **Modulus** p angewendet werden. Der in der Vorlesung gezeigte Fehler (fehlendes $\bmod p$ bei K) wurde hier korrigiert.

- Kombination von asymmetrischer und symmetrischer Kryptographie

- Nachricht wird mit **symmetrischem Verfahren** verschlüsselt (schnell, effizient)
- Symmetrischer Schlüssel wird mit **asymmetrischem Verfahren** verschlüsselt (sicherer Schlüsselaustausch)
- Vorteile: Effizienz + Sicherheit, einfaches Teilen mit mehreren Empfängern
- Nachteil: Sicherheit von beiden Systemen abhängig

5. Digitale Signaturen

Zweck digitaler Signaturen

- **Authentizität**: Nachweis des Urhebers
- **Integrität**: Nachweis der Unverändertheit
- **Nicht-Abstreitbarkeit (Non-Repudiation)**: Unterzeichner kann Unterschrift nicht abstreiten

Rechtlicher Rahmen (eIDAS/VDG):

- **Einfache elektronische Signatur**: Keine besondere rechtliche Vermutung
- **Fortgeschrittene elektronische Signatur**: Eindeutige Zuordnung, hohes Vertrauen
- **Qualifizierte elektronische Signatur**: Rechtliche Gleichstellung mit handschriftlicher Unterschrift

1.5.1 RSA-Signaturen

RSA-Signaturverfahren

- Schlüsselgenerierung wie bei RSA
- Signieren: $s = (h(m))^d \mod n$
- Verifizieren: Teste ob $h(m) = s^e \mod n$
- **Hashfunktion h notwendig** zur Vermeidung von Angriffen

Digital Signature Algorithm (DSA)

DSA Parametergenerierung

1. Wähle Primzahl q (160/224/256 Bit)
2. Wähle Primzahl p (1024/2048/3072 Bit) mit $q \mid (p - 1)$
3. Finde $g \in \mathbb{Z}_p^\times$ mit $\text{ord}(g) = q$
4. Parameter (p, q, g) sind öffentlich

DSA Schlüsselgenerierung und Signatur

- Privater Schlüssel: x mit $1 < x < q$
- Öffentlicher Schlüssel: $y = g^x \mod p$
- Signieren: Wähle k , berechne $r = (g^k \mod p) \mod q$, $s = k^{-1} \cdot (H(m) + r \cdot x) \mod q$
- Verifizieren: Berechne $w = s^{-1} \mod q$, $u_1 = H(m) \cdot w \mod q$, $u_2 = r \cdot w \mod q$, $v = (g^{u_1} \cdot y^{u_2} \mod p) \mod q$, akzeptiere wenn $v = r$

1.5.2 Sicherheitsbegriffe für Signaturen

Angriferwissen

- **Key-Only Attack:** Nur öffentlicher Schlüssel bekannt
- **Known Signature Attack:** Nachricht-Signatur-Paare bekannt
- **Chosen Message Attack:** Signaturen für selbstgewählte Nachrichten erhältlich
- **Adaptive Chosen Message Attack:** Signaturen auch während Angriff erhältlich

Angriferziele

- **Existential Forgery:** Neues gültiges Nachricht-Signatur-Paar
- **Selective Forgery:** Signatur für bestimmte neue Nachricht
- **Universal Forgery:** Signatur für beliebige Nachricht
- **Total Break:** Bestimmung des privaten Schlüssels

1.6 Schlüsselverteilung und Schlüsselaustausch

1.6.1 Schlüsselarten

- **Langzeitschlüssel:** Lange Gültigkeit (Monate/Jahre), häufig für Authentifizierung
- **Sitzungsschlüssel (Session Keys):** Kurze Gültigkeit (eine Sitzung), reduziert Risiko bei Kompromittierung

Public Key Infrastructure (PKI)

Zertifikate

- Bestätigung der Zuordnung von öffentlichen Schlüsseln zu Identitäten durch vertrauenswürdige dritte Partei
- Enthalten: Öffentlicher Schlüssel, Name, Gültigkeitszeitraum, Aussteller, Signatur
- **X.509:** Hierarchisches, zentralisiertes System mit Root-Zertifikaten
- **Web of Trust:** Dezentrales System (PGP), gegenseitige Zertifizierung

1.6.2 Schlüsselaustauschprotokolle

Dolev-Yao-Angreifermodell

- Angreifer hat volle Kontrolle über Kommunikationskanal
- Kann: Abfangen, Verzögern, Unterdrücken, Ersetzen, Unter falscher Identität senden
- Kann **nicht**: Kryptographische Primitive brechen (perfekte Kryptographie angenommen)

Needham-Schroeder

- Symmetrische Version anfällig für Replay-Angriffe (veraltete Schlüssel)
- Asymmetrische Version sicherer, aber anfällig gegen aktive Angreifer ohne Authentifizierung

1.6.3 Diffie-Hellman-Schlüsselaustausch

Diffie-Hellman Protokoll

Dabei geht es um den Schlüsselaustausch um danach ein symmetrisches Kommunikationsverfahren anzuwenden. g^{ab} ist der gemeinsame symmetrische Schlüssel am Ende. Die öffentlichen Elemente sind: $g, p, g^a \bmod p, g^b \bmod p$

1. Einigung auf Primzahl p und Generator g von \mathbb{Z}_p^\times
 2. A wählt a , sendet $g^a \bmod p$ an B
 3. B wählt b , sendet $g^b \bmod p$ an A
 4. A berechnet $(g^b)^a = g^{ab} \bmod p$
 5. B berechnet $(g^a)^b = g^{ab} \bmod p$
- Gemeinsamer Schlüssel: $K = g^{ab} \bmod p$

Sicherheit:

- Basierend auf **Computational Diffie-Hellman Problem (CDH)**: Berechnung von g^{ab} aus g, g^a, g^b
- **Man-in-the-Middle-Angriff** möglich: Angreifer führt zwei separate DH-Protokolle
- Lösung: **Authenticated Diffie-Hellman** oder **Station-to-Station (STS)** Protokoll mit Signaturen

Station-to-Station (STS) Protokoll

- $A \rightarrow B: g^a$
 - $B \rightarrow A: g^b, \{\text{sig}(sk_B, (g^a, g^b))\}_K$ mit $K = g^{ab}$
 - $A \rightarrow B: \{\text{sig}(sk_A, (g^a, g^b))\}_K$
- Signatur gewährleistet Authentizität und Integrität.

Logjam-Angriff (2015):

- Vorberechnung des diskreten Logarithmus für häufig verwendete Primzahlen
- Betraf 512/768 Bit, Abschätzung für 1024 Bit möglich
- Lösung: Verwendung größerer, individueller Primzahlen

2 Authentifizierung

2.1 Grundlagen

2.1.1 Begriffsdefinitionen

Identität

Eine Menge von Attributen, die eine Entität beschreiben (z.B. Name, Geburtsdatum, Wohnort).

Authentisierung

Die **Bereitstellung** von Unterlagen oder Nachweisen, die es ermöglichen, die Identität zu prüfen (z.B. das Vorzeigen eines Personalausweises).

Authentifikation / Authentifizierung

Die **Prüfung** und Echtheitsbezeugung der vorgelegten Unterlagen zur Identitätsfeststellung (z.B. der Vergleich des Fotos auf dem Ausweis mit der Person).

Autorisierung

Die Gewährung oder Verwehrung von Rechten an eine (authentifizierte) Entität.

2.1.2 Drei Ansätze der Authentisierung

Ziel ist die Identifikation von Subjekten (Menschen, Systeme, Dienste) und der Nachweis ihrer Identität.

- **Durch Wissen:** z.B. PIN, Passwort, kryptographischer Schlüssel.
- **Durch Besitz:** z.B. Smartcard, Token, SIM-Karte.
- **Durch Merkmale:** z.B. Biometrie (physiologische Eigenschaften).

2.2 Authentisierung durch Besitz

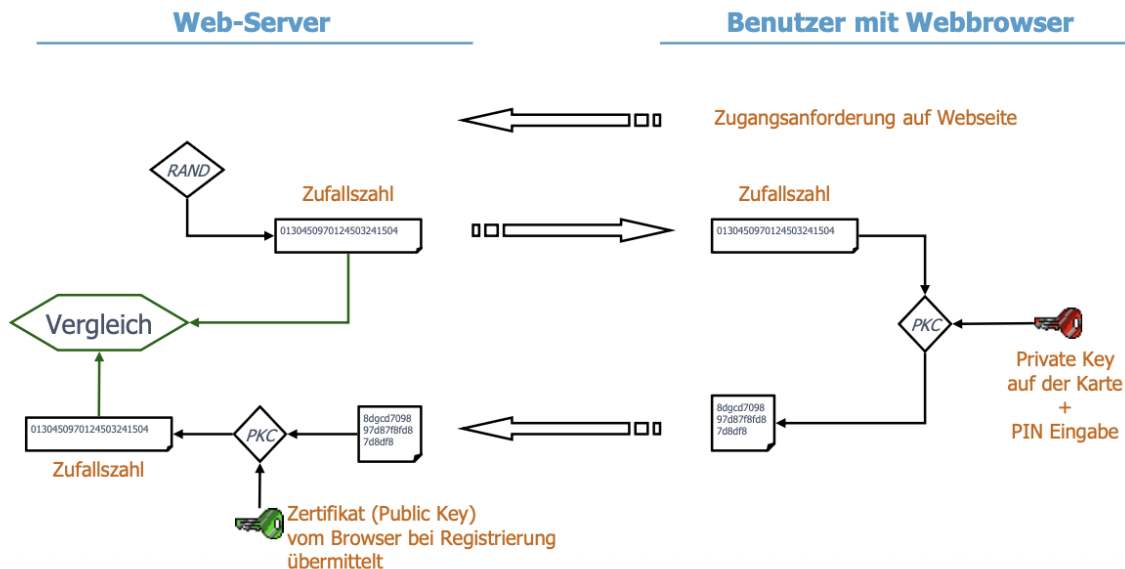
- **Statische Tokens:** Ein gespeichertes Geheimnis (z.B. privater Schlüssel) wird direkt genutzt.
- **Dynamische Tokens:** Das Geheimnis wird zur Berechnung von Authentifizierungs-Informationen genutzt (z.B. Challenge-Response).
- **Hardware-Tokens:** Schlüssel, SmartCard, Transponder.
- **Software-Tokens:** Cookie, Client-Zertifikat.

2.2.1 Beispiel: ATHENE KARTE (SmartCard)

- Besitzt einen **Kryptoprozessor** (z.B. CardOS 4.3b).
- Träger eines privaten Schlüssels (z.B. 2048 Bit RSA) und eines öffentlichen Zertifikats (digitale ID).
- Der **private Schlüssel ist nicht auslesbar** und zusätzlich durch eine PIN geschützt.

2.2.2 Challenge-Response-Verfahren (mit Kryptoprozessor)

Dies ist ein dynamisches Verfahren, das den privaten Schlüssel nutzt, ohne ihn preiszugeben.



1. **Benutzer (mit Webbrowser)** initiiert eine Zugangsanforderung auf einem Web-Server.
2. **Web-Server** generiert eine **Zufallszahl** (die "Challenge") und sendet sie an den Benutzer.
3. **Benutzer** gibt seine PIN ein, um den Kryptoprozessor der Karte freizuschalten. Die Karte "signiert" die Zufallszahl mit dem **privaten Schlüssel**.
4. **Benutzer** sendet die signierte Zufallszahl zurück an den Server.
5. **Web-Server** nutzt das **öffentliche Zertifikat** (Public Key) des Benutzers (das er z.B. bei einer Registrierung erhalten hat), um die Signatur zu prüfen.
6. Stimmt die verifizierte Zufallszahl mit der ursprünglich gesendeten überein, ist der Benutzer authentifiziert.

2.2.3 Probleme

- **Diebstahl:** Offensichtliches Problem.
- **Gegenmaßnahme:** Sicherung des Tokens durch ein zusätzliches Merkmal, z.B. Wissen (PIN für Hardware-Crypto, Passwort für Software-Crypto) oder 2. Faktor.
- **Extraktion der Schlüssel:** Angriffe auf die Token-Hardware.
- **Methoden:** Schwachstellen in der Firmware oder **Side-Channel-Angriffe** (z.B. Monitoring des Stromverbrauchs), um den privaten Schlüssel auszulesen.

2.3 Authentisierung durch Merkmale (Biometrie)

2.3.1 Anforderungen an biometrische Merkmale

- **Universalität:** Jede Person besitzt das Merkmal.
- **Eindeutigkeit:** Merkmal ist für jede Person verschieden.
- **Beständigkeit:** Merkmal ist (weitgehend) unveränderlich.
- **Quantitative Erfassbarkeit:** Messbar mittels Sensoren.
- **Performanz:** Genauigkeit und Geschwindigkeit der Erfassung/Prüfung.
- **Akzeptanz:** Benutzer müssen bereit sein, das Merkmal zu nutzen.

- **Fälschungssicherheit:** Schutz gegen Angriffe.

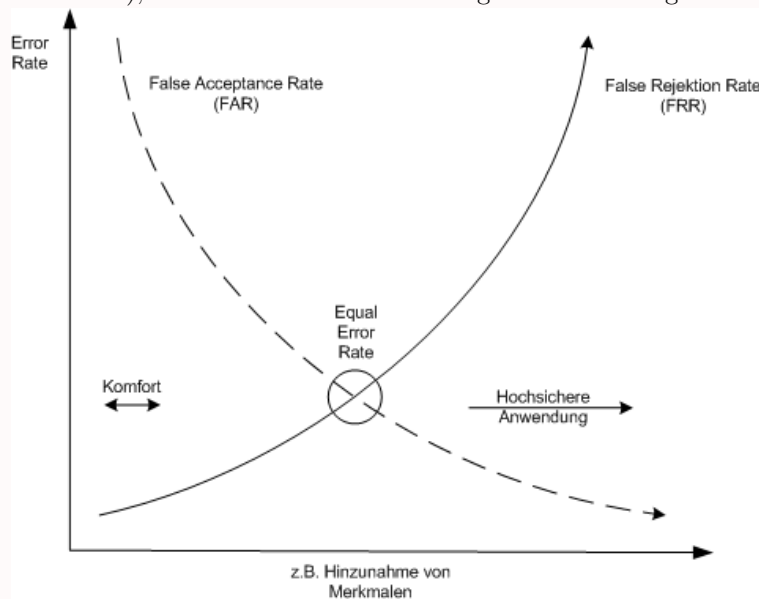
2.3.2 Prozesse und Fehlerraten

- **Enrollment:** Erstmalige Registrierung eines Benutzers und Erfassung seines Merkmals (Template).
- **Verifikation:** Erneute Erfassung und Vergleich mit dem gespeicherten Template.
- **Biometrie ist immer fehlerbehaftet** (ein statistischer Test).

Fehlerraten

- **False Acceptance Rate (FAR):** Ein Unberechtigter wird fälschlicherweise authentifiziert. (**Sicherheitsproblem!**)
- **False Rejection Rate (FRR):** Ein Berechtigter wird fälschlicherweise abgewiesen. (**Benutzbarkeits-/Akzeptanzproblem!**)
- **Equal Error Rate (EER):** Der Punkt, an dem $FAR = FRR$.

Man muss für den Anwendungsfall abwägen: Eine hochsichere Anwendung optimiert auf eine niedrige FAR (auf Kosten der Bequemlichkeit), eine komfortable Anwendung auf eine niedrige FRR.



2.3.3 Beispiel: Fingerabdruck

- Es wird nicht das Bild gespeichert, sondern ein Template aus **Minutien** (Verzweigungen, Endpunkte, etc.) mit relativen Koordinaten und Winkeln.
- Problem: Schlechte Abdrücke können zu fehlenden Minutien führen (-> höhere FRR).

2.3.4 Probleme der Biometrie

- **Datenschutz:** Biometrische Merkmale können "intrusiv" sein und sensible Daten enthüllen (z.B. Venenmuster, DNA -> Gesundheitsdaten).
- **Speicherung:** Es sollten Referenzdaten (Templates) gespeichert werden, aus denen das Merkmal nicht rekonstruiert werden kann.
- **Keine oder begrenzte Widerrufbarkeit:** Ein kompromittierter Fingerabdruck (oder Iris, DNA) kann nicht einfach "gesperrt" und ersetzt werden wie ein Passwort.
- **Kompromittierung:** Wenn eine Kopie erstellt werden kann (z.B. von einem Lesegerät gestohlen), wird die Authentifizierung durch *Merkmal* ("Wer bin ich?") zu einer Authentifizierung durch *Wissen* ("Wie sieht Merkmal X aus?").

- **Gegenmaßnahme: Lebendverifikation** (Liveness Detection) prüft, ob ein echter Finger (keine Plastik-Kopie) aufliegt.

2.4 Authentisierung durch Wissen

2.4.1 Passwörter

Gängigste Methode. Werden (idealerweise) nicht im Klartext, sondern als **Hash-Wert** gespeichert (z.B. in `/etc/shadow` (Linux) oder via LSASS (Windows)).

Evolution der Passwort-Authentifizierung

1. **Plaintext-Übertragung:** Passwort wird im Klartext gesendet (z.B. `telnet`, `ftp`).
2. **Problem:** Passiver Angreifer (Sniffer) im Netz sieht alle Passwörter.
3. **Übertragung mit TLS:** Der Kanal ist geschützt (z.B. HTTPS).
4. **Problem:** Wenn der Server das Passwort im Klartext in seiner Datenbank speichert, erlangt ein Angreifer bei einem Datenbank-Leak alle Passwörter.
5. **Server speichert Hash:** Server speichert $H_{ID} := h(P_{ID})$. Beim Login sendet der Nutzer P_{ID} , der Server berechnet $h(P_{ID})$ und vergleicht es mit dem gespeicherten H_{ID} .
6. **Problem: Rainbow-Tables.** Da $h(\text{"Passwort123"})$ für alle Nutzer gleich ist, kann ein Angreifer eine Tabelle mit Hashes für Millionen gängiger Passwörter vorab berechnen und die gehashte Datenbank sehr schnell knacken.
7. **Server speichert Hash mit Salt:** Server speichert $H_{ID} := h(P_{ID}, s_{ID})$, wobei s_{ID} ein einzigartiger, zufälliger **Salt** pro Nutzer ist (wird mit H_{ID} gespeichert).
8. **Vorteil:** $h(\text{"Passwort123"}, s_1) \neq h(\text{"Passwort123"}, s_2)$.
9. **Eine Rainbow-Table muss nun für jeden Nutzer (jeden Salt) separat erstellt werden**, was den Geschwindigkeitsvorteil zunichte macht und den Angriff stark verlangsamt.

Passwort-Manager Da man für unterschiedliche Dienste unterschiedliche, starke Passwörter nutzen soll, ist das Auswendiglernen unmöglich.

- **Lösung:** (Lokale) Passwort-Manager, die mit einem starken Masterpasswort geschützt sind.

2.4.2 Challenge-Response-Verfahren (CHAP)

Authentisierung durch Wissen, ohne das "Wissen" (Passwort) zu übertragen. Nutzt symmetrische Kryptographie (HMAC).

- **Voraussetzung:** Alice (Nutzer) und Bob (Server) teilen ein Geheimnis (P_{ID} , z.B. das Passwort).
- **Ablauf (CHAP):**
 1. Alice \rightarrow Bob: ID
 2. Bob \rightarrow Alice: $RAND$ (Zufallszahl, "Challenge")
 3. Alice \rightarrow Bob: $c = \text{HMAC}(P_{ID}, RAND)$
 4. Bob prüft: Berechnet $c' = \text{HMAC}(P_{ID}, RAND)$ und testet, ob $c' == c$.
- **Probleme:**
 - Der Klartextraum für $RAND$ muss groß sein, sonst **Replay-Attacke** (Angreifer kann mehrfach genutzte Challenges korrekt beantworten).
 - Server muss P_{ID} im Klartext kennen. Speichert er stattdessen $h(P_{ID})$, braucht der Angreifer bei einem Leak auch nur noch den Hash (und nicht das Passwort), um sich zu authentifizieren.
 - Schützt nur die Authentisierung, nicht den restlichen Kommunikationskanal (Integrität).

2.5 Single Sign On (SSO)

- **Problem:** "Passwort-Müdigkeit" – zu viele Dienste erfordern eigene Passwörter.
- **Definition:** Eine Authentisierungsmethode, die es einem Benutzer ermöglicht, sich mit **einem einzigen Satz** von Anmeldeinformationen bei **mehreren unabhängigen** Softwaresystemen anzumelden.
- **Idee:** Ein zentraler, vertrauenswürdiger **Provider** bestätigt die Identität des Nutzers gegenüber allen anderen **Services**.
- **Vorteile:** Weniger Passwörter (nur ein starkes nötig), erhöhte Sicherheit (wenn gut implementiert), Komfort, bessere Kontrolle.
- **Nachteil: Single Point of Failure.** Wird das SSO-Login kompromittiert, hat ein Angreifer Zugriff auf *alle* verbundenen Dienste.

2.5.1 Kerberos (Ein SSO-Protokoll)

- **Ziele:** Authentifizierung von *Principals* (Benutzer, Server), Austausch von Sitzungs-Schlüsseln, SSO innerhalb einer administrativen Domäne (*Realm*).
- **Design:**
 - Pro *Realm* ein **Key Distribution Center (KDC)**.
 - **KDC = Authentication Server (AS) + Ticket Granting Server (TGS)**.
 - Basiert auf **Pre-Shared Secrets**: Das KDC kennt einen geheimen Schlüssel (K) für jeden Principal in seinem Realm (z.B. K_{Bob} , K_{TGS} , K_{SMB}). Für Benutzer wird K_{Bob} aus deren Passwort-Hash generiert.

Kerberos-Ablauf (vereinfacht) Ziel: Benutzer Bob (C) möchte auf den SMB-Server (S) zugreifen.

1. Login + TGT-Anfrage:

- Bob gibt sein Passwort ein. Client C generiert $K_{Bob} = \text{Hash}(\text{Passwort})$.
- $C \rightarrow AS: (K_{Bob}(\text{timestamp}), \text{Bob}, \text{TGS})$
(Bob bittet den AS um ein Ticket für den TGS, authentifiziert sich mit einem verschlüsselten Timestamp).

2. AS-Antwort (TGT):

- $AS \rightarrow C: \{K_{Bob, TGS}\}_{K_{Bob}} + \{TGT\}_{K_{TGS}}$
- Der AS prüft den Timestamp. Wenn gültig:
- Er sendet den **Sitzungsschlüssel** für C und TGS ($K_{Bob, TGS}$), verschlüsselt mit Bobs Schlüssel (K_{Bob}).
- Er sendet das **Ticket Granting Ticket (TGT)**, welches ($K_{Bob, TGS}, \text{Bob}, \dots$) enthält, alles verschlüsselt mit dem geheimen Schlüssel des TGS (K_{TGS}). **Der Client kann das TGT nicht lesen.**

3. Service-Ticket-Anfrage:

- $C \rightarrow TGS: \{A_{Bob}\}_{K_{Bob, TGS}} + \{TGT\}_{K_{TGS}} + \text{"SMB"}$
- C entschlüsselt $\{K_{Bob, TGS}\}_{K_{Bob}}$, um den Sitzungsschlüssel $K_{Bob, TGS}$ zu erhalten.
- C erstellt einen *Authenticator* $A_{Bob} = (\text{Bob}, \text{IP}, \text{timestamp})$ und verschlüsselt ihn mit $K_{Bob, TGS}$.
- C sendet den Authenticator, das (unlesbare) TGT und den Namen des Zieldienstes ("SMB") an den TGS.

4. TGS-Antwort (Service Ticket):

- $TGS \rightarrow C: \{K_{Bob, SMB}\}_{K_{Bob, TGS}} + \{T_{Bob, SMB}\}_{K_{SMB}}$
- TGS entschlüsselt das TGT (mit K_{TGS}) und den Authenticator (mit $K_{Bob, TGS}$) und prüft sie.
- Er generiert einen neuen Sitzungsschlüssel für Bob und den SMB-Server ($K_{Bob, SMB}$).
- Er sendet $K_{Bob, SMB}$, verschlüsselt mit $K_{Bob, TGS}$.

- Er sendet das **Service Ticket** ($T_{Bob,SMB}$), welches $(K_{Bob,SMB}, Bob, \dots)$ enthält, alles verschlüsselt mit dem geheimen Schlüssel des SMB-Servers (K_{SMB}).

5. Zugriff auf Dienst:

- $C \rightarrow SMB: \{A'_{Bob}\}_{K_{Bob,SMB}} + \{T_{Bob,SMB}\}_{K_{SMB}}$
- C entschlüsselt $K_{Bob,SMB}$.
- C erstellt einen *neuen* Authenticator A'_{Bob} und verschlüsselt ihn mit $K_{Bob,SMB}$.
- C sendet den neuen Authenticator und das (unlesbare) Service Ticket an den SMB-Server.

6. Verifikation:

- Der SMB-Server entschlüsselt das Service Ticket (mit K_{SMB}) und den Authenticator (mit $K_{Bob,SMB}$) und prüft sie.
- Wenn alles gültig ist, ist Bob authentifiziert und Bob und SMB teilen sich den Sitzungsschlüssel $K_{Bob,SMB}$ für die weitere Kommunikation.

Kerberos-Angriffe

- **Pass the Hash:** Der Angreifer stiehlt den Hash K_{Bob} (z.B. aus dem LSASS-Prozessspeicher). Da K_{Bob} das "Geheimnis" ist, das Kerberos verwendet, kann der Angreifer **Schritt 1** des Protokolls direkt ausführen und ein TGT erhalten, **ohne das Klartextpasswort zu kennen**.
- **Golden Ticket:** Der Angreifer kompromittiert das KDC und stiehlt den geheimen Schlüssel des TGS selbst (den Hash des KRBTGT-Kontos). Mit diesem Schlüssel kann der Angreifer **offline** ein TGT für **jeden beliebigen Benutzer** (z.B. Administrator) mit **beliebiger Gültigkeitsdauer** fälschen. Dies gewährt dem Angreifer uneingeschränkten, persistenten Zugriff auf die gesamte Domäne.

2.6 Autorisierung (Zugriffskontrollmodelle)

Nach der Authentifizierung (Wer bist du?) folgt die Autorisierung (Was darfst du?).

- **Referenzmonitor:** Ein abstraktes Konzept, das jede Anfrage eines **Subjekts** (Prozess, Benutzer) auf ein **Objekt** (Datei, Speicher) prüft und anhand einer Rechedatenbank entscheidet (gewährt / abgelehnt).
- **Schutzziele:** Integrität und Vertraulichkeit.

2.6.1 Discretionary Access Control (DAC)

	Datei1	Datei2	Datei3	Prozess1	Prozess2
Prozess1	{ read, write }		{ read, write }		{ send, receive }
Prozess2				{ send, receive }	
Prozess3		{ owner, execute }		{ signal }	

- **Definition:** Der **Eigentümer** eines Objekts ist für die Vergabe von Zugriffsrechten verantwortlich ("at his discretion").
- **Modell: Zugriffsmatrix** ($M : S \times O \rightarrow \mathcal{P}(R)$), die Rechte von Subjekten S auf Objekte O abbildet.

- **Implementierung (Speicherung der Matrix):**
 - **Spaltenweise (Access Control Lists, ACLs):** Jedes *Objekt* hat eine Liste, die alle Subjekte und deren Rechte aufführt. (z.B. Dateiberechtigungen in Windows/Linux).
 - *Vorteil:* Effizient zu bestimmen: "Wer darf auf diese Datei zugreifen?"
 - **Zeilenweise (Capability Lists, CLs):** Jedes *Subjekt* hat eine Liste (Capability) mit allen Objekten, auf die es zugreifen darf, und den jeweiligen Rechten.
 - *Vorteil:* Effizient zu bestimmen: "Worauf darf dieser Benutzer zugreifen?"
- **Nachteile:** Keine formalen Garantien für Informationsfluss (Problem "Trojanisches Pferd": Ein Programm, das im Kontext des Nutzers läuft, kann dessen Rechte missbrauchen, z.B. eine Datei kopieren und unerlaubt weitergeben).

2.6.2 Role-based Access Control (RBAC)

- **Definition:** Berechtigungen werden nicht direkt an Benutzer, sondern an **Rollen** (z.B. "Arzt", "Buchhalter", "Admin") vergeben. Benutzer werden dann diesen Rollen zugewiesen.
- **Vorteile:** Bildet Organisationsstrukturen gut ab; erleichtert Prinzipien wie "Need-to-Know" und "Separation-of-Duty".

2.6.3 Mandatory Access Control (MAC)

- **Definition:** Systembestimmte (regelbasierte) Festlegung von Sicherheitseigenschaften. **Systemregeln dominieren (überschreiben) Benutzerwünsche** (DAC-Einstellungen).
- **Ziel:** Kontrolle des Informationsflusses.

Beispiel: Bell-La Padula (BLP) Modell Ein MAC-Modell, das sich auf **Vertraulichkeit** (Confidentiality) konzentriert.

- **Konzept:** Subjekte und Objekte erhalten **Sicherheitsklassen (Labels)**, z.B. (Level, {Kategorien}).
- **Level:** Haben eine totale Ordnung (z.B. unklassifiziert < vertraulich < geheim < streng geheim).
- **Kategorien:** Eine Menge von Zuständigkeiten (z.B. {Buchhaltung}, {Forschung}).
- **Dominanz (\geq):** Ein Subjekt S dominiert ein Objekt O ($SC(S) \geq SC(O)$), wenn S ein höheres oder gleiches Level hat **und** die Kategoriemenge von O eine Teilmenge der Kategoriemenge von S ist ($L_S \geq L_O \wedge C_O \subseteq C_S$).

Bell-La Padula Regeln

- **1. Simple-Security-Property (No-Read-Up):**
 - Ein Subjekt S darf ein Objekt O nur **lesen**, wenn $SC(S) \geq SC(O)$ (Subjekt dominiert Objekt).
 - (Ein "geheimer" Sekretär darf keine "streng geheimen" Bilanzdaten lesen).
- **2. *-Property (No-Write-Down):**
 - Ein Subjekt S darf ein Objekt O nur **schreiben**, wenn $SC(S) \leq SC(O)$ (Objekt dominiert Subjekt).
 - (Ein "strenger geheimer" CEO darf Bilanzdaten nicht in eine "unklassifizierte" Website schreiben und so leaken).

- **Nachteile BLP:** Informationen fließen sukzessive nur "nach oben". Erlaubt "blindes Schreiben" (Schreiben in ein Objekt, das man nicht mehr lesen darf → Integritätsproblem). Modelliert keine "Covert Channels".

3 Netzwerkgrundlagen & Sicherheit

3.1 Einführung und Modelle

Dieser Abschnitt behandelt die Grundlagen von Netzwerken, Kommunikationsmodellen und spezifischen Angriffen sowie deren Abwehr auf verschiedenen Schichten.

3.1.1 OSI-Modell (Open System Interconnection)

Das OSI-Modell ist ein Referenzmodell für Netzwerkprotokolle, unterteilt in 7 Schichten. Jede Schicht bietet Dienste für die darüberliegende Schicht an.

- **Layer 7: Application Layer** (Anwendungsschicht)
Stellt Funktionen für Anwendungen bereit (nicht die Anwendung selbst).
HTTPS/S, FTP, SMTP, DHCP, DNS
- **Layer 6: Presentation Layer** (Darstellungsschicht)
Datenformatierung, Kompression, Verschlüsselung.
SSL/TLS
- **Layer 5: Session Layer** (Sitzungsschicht)
Sitzungsmanagement (Aufbau, Abbau), Authentifizierung.
RPC, SMPP
- **Layer 4: Transport Layer** (Transportschicht)
Ende-zu-Ende Kommunikation, TCP/UDP.
TCP, UDP
- **Layer 3: Network Layer** (Vermittlungsschicht)
Logische Adressierung (IP), Routing.
IPv4, IPv6, ARP, ICMP
- **Layer 2: Data Link Layer** (Sicherungsschicht)
Physische Adressierung (MAC), Zugriff auf das Medium.
SDLC, SLIP, NCP
- **Layer 1: Physical Layer** (Bitübertragungsschicht)
Übertragung von Bits über ein Medium (Kabel, Funk).
Ethernet, Wi-Fi

3.1.2 TCP/IP Modell vs. OSI

Das TCP/IP-Modell ist eine vereinfachte, praxisorientierte Version des OSI-Modells (oft 4 Schichten).

Vergleich der Dateneinheiten (Encapsulation)

Beim Durchlaufen der Schichten von oben nach unten werden Daten **gekapselt** (Encapsulation). Jede Schicht fügt ihren Header (und teilweise Trailer) hinzu.

- **Application Layer:** Daten / Message (M)
- **Transport Layer:** **Segments** (Header $H_t + M$)
- **Internet Layer:** **Packets** (Header $H_i + H_t + M$)
- **Link Layer:** **Frames** (Header $H_l + \dots + TrailerT_l$)

3.2 Die Schichten im Detail

3.2.1 Layer 1: Physical Layer

-
- **Funktion:** Konvertierung von Daten in physikalische Signale zur Übertragung zwischen Geräten.
 - **Medien:**
 - Elektrische Impulse (Kupferkabel)
 - Lichtimpulse (Glasfaser)
 - Funksignale (Wi-Fi)

3.2.2 Layer 2: Data Link Layer

- **Funktion:** Verbindung zwischen zwei Geräten im *selben* Netzwerk (Hop-to-Hop).
- **Hardware:** Switches.
- **Adressierung:** **MAC-Adresse** (Media Access Control).
 - Weltweit eindeutig (theoretisch).
 - 48 Bit lang (6 Bytes).
 - **Aufbau:** Erste 3 Bytes = Hersteller-Kennung (OUI), Letzte 3 Bytes = Seriennummer.

3.2.3 Layer 3: Network Layer

- **Funktion:** Logische Adressierung und Weiterleitung (Routing) über Netzwerkgrenzen hinweg.
- **Protokolle:** IPv4, IPv6, ICMP.
- **Hardware:** Router.
- **Wichtig:** IP ist ein **unzuverlässiges** Protokoll (Best Effort). Es gibt keine Garantie für die Ankunft der Pakete.

3.2.4 Layer 4: Transport Layer

Stellt die Ende-zu-Ende-Kommunikation sicher.

- **Multiplexing:** Nutzung von **Ports**, um verschiedene Dienste (z.B. Web, Mail) gleichzeitig auf einem Host zu betreiben.
- **Segmentierung:** Aufteilen großer Datenmengen.
- **Fehlererkennung:** Checksummen.
- **Flusskontrolle:** Vermeidung von Überlastung des Empfängers.

TCP vs. UDP

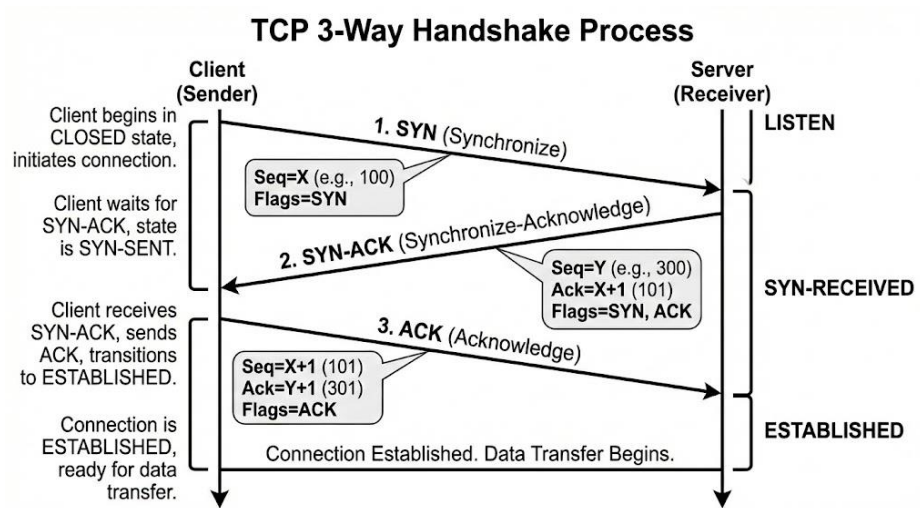
TCP (Transmission Control Protocol):

- **Verbindungsorientiert** (Handshake notwendig).
- **Zuverlässig** (ACKs für Pakete, Neuversand bei Verlust).
- **Reihenfolge:** Garantiert (Sequenznummern).
- **Einsatz:** Web (HTTP), Email (SMTP), Dateitransfer (FTP).

UDP (User Datagram Protocol):

- **Verbindungslos** (Fire-and-Forget).
- **Unzuverlässig** (Keine ACKs, kein Neuversand).
- **Schnell:** Geringer Overhead (nur 8 Byte Header).
- **Einsatz:** Streaming, Gaming, DNS, DHCP.

TCP 3-Way Handshake (Verbindungsaufbau) Um eine Verbindung aufzubauen, nutzen Client und Server folgenden Ablauf:



1. **SYN:** Client sendet Seq=X, Flags=SYN. (Status: SYN-SENT)
2. **SYN-ACK:** Server antwortet Seq=Y, Ack=X+1, Flags=SYN,ACK. (Status: SYN-RECEIVED)
3. **ACK:** Client bestätigt Seq=X+1, Ack=Y+1, Flags=ACK. (Status: ESTABLISHED)

TCP Connection Termination (Verbindungsabbau) Der Abbau erfolgt in der Regel über einen 4-Schritte-Prozess unter Nutzung des **FIN**-Flags:

1. **FIN:** Client möchte schließen, sendet Flags=FIN. (Status: FIN-WAIT-1)
2. **ACK:** Server bestätigt den Erhalt mit Flags=ACK. (Status: CLOSE-WAIT beim Server, FIN-WAIT-2 beim Client)
3. **FIN:** Server ist bereit zum Schließen, sendet ebenfalls Flags=FIN. (Status: LAST-ACK)
4. **ACK:** Client bestätigt den Erhalt mit Flags=ACK. (Status: TIME-WAIT, danach CLOSED)

3.2.5 Layer 5-7: Höhere Schichten

- **Session Layer:** Authentifizierung, Verwaltung von Sitzungen (z.B. RPC).
- **Presentation Layer:** Datenkonvertierung (z.B. ASCII → ASN.1), Verschlüsselung (SSL/TLS wird oft hier eingeordnet), Kompression.
- **Application Layer:** Protokolle für Anwendungen. Ports definieren den Service:
 - HTTP/S: Port 80/443
 - FTP: Port 20/21
 - SMTP: Port 25

3.3 Angriffsmodelle im Netzwerk

- **Eavesdropping (Abhören):** Passiver Angreifer. Liest Daten mit, verändert sie aber nicht. Abwehr: Verschlüsselung.
- **On-Path / Man-in-the-Middle (MitM):** Angreifer sitzt *auf* dem Kommunikationsweg (z.B. kontrolliert Router). Kann Daten lesen, **verändern**, **blockieren** oder einschleusen.

- **Off-Path:** Angreifer sitzt *nicht* auf dem direkten Weg. Kann Daten nicht mitlesen oder blockieren, aber Daten einschleusen (z.B. Spoofing mit gefälschter Absenderadresse).

3.4 Netzwerkprotokolle und spezifische Angriffe

3.4.1 ARP (Address Resolution Protocol)

Funktion: Auflösung einer bekannten IP-Adresse zu einer unbekannten MAC-Adresse im lokalen Netzwerk (Layer 2).

Ablauf ARP

1. **Request:** "Wer hat IP 10.23.4.38?" → Gesendet als **Broadcast** (FF:FF:FF:FF:FF:FF). Alle Geräte empfangen es.
2. **Reply:** "Ich (10.23.4.38) habe MAC 11:AB:..." → Gesendet als **Unicast** an den Anfragenden.

ARP Spoofing / Cache Poisoning Da ARP **zustandslos** ist (Clients akzeptieren Antworten auch ohne vorherige Anfrage), kann ein Angreifer gefälschte ARP-Replies senden.

- **Angriff:** Angreifer sendet: "Ich bin IP des Routers" an das Opfer und "Ich bin IP des Opfers" an den Router.
- **Folge:** Der ARP-Cache der Opfer wird "vergiftet". Der Angreifer wird zum *Man-in-the-Middle*.
- **Gegenmaßnahmen:**
 - Statische ARP-Einträge (aufwendig).
 - ARP-Monitoring Tools (z.B. Arpwatch, Snort).
 - Nutzung von IPv6 (nutzt NDP + SEND, sicherer).
 - Netzwerksegmentierung.

MAC Spoofing: MAC-Adressen sind in Software leicht änderbar. MAC-Filter sind daher kein verlässlicher Schutz.

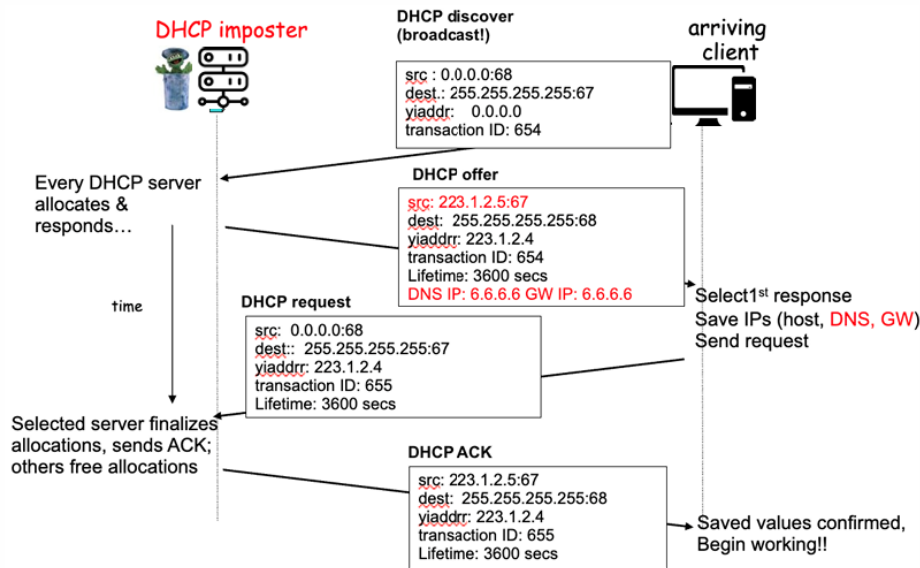
3.4.2 DHCP (Dynamic Host Configuration Protocol)

Funktion: Automatische Zuweisung von IP-Adressen, Subnetzmasken, Gateway und DNS an Clients. Nutzt UDP (Ports 67/68).

Ablauf (DORA-Prinzip)

1. **Discover** (Broadcast): Client sucht DHCP-Server.
2. **Offer** (Unicast/Broadcast): Server bietet IP an.
3. **Request** (Broadcast): Client fordert die angebotene IP an.
4. **Ack** (Unicast/Broadcast): Server bestätigt und verleast IP.

DHCP Spoofing (Rogue DHCP) Ein Angreifer stellt einen falschen DHCP-Server im Netz auf. Wenn er schneller antwortet als der echte Server (Race Condition), übernimmt er die Konfiguration des Clients.



- **Gefahr:** Angreifer setzt sich selbst als Gateway oder DNS-Server (MitM).
- **Gegenmaßnahme: DHCP Snooping** auf Switches.
 - Ports werden in **Trusted** (nur hier darf ein DHCP-Server hängen) und **Untrusted** unterteilt.
 - DHCP-Offers von Untrusted Ports werden blockiert.

3.4.3 ICMP und (D)DoS Angriffe

ICMP (Internet Control Message Protocol): Dient dem Austausch von Informations- und Fehlermeldungen (z.B. ping zur Latenzmessung).

(D)DoS - (Distributed) Denial of Service Ziel ist es, die Verfügbarkeit eines Dienstes zu stören.

- **DoS:** Ein Angreifer.
- **DDoS:** Viele Angreifer (Botnet).

Spezifische Angriffe

- **Ping of Death:** Senden von malformierten (z.B. zu großen) ICMP-Paketen, die beim Zusammensetzen den Server zum Absturz bringen. (Heute meist gepatcht).
- **Smurf Attack (Amplification):**
 - Angreifer sendet Ping an die **Broadcast-Adresse** eines Netzwerks.
 - Absender-Adresse ist gefälscht auf die **Opfer-IP**.
 - Alle Hosts im Netz antworten dem Opfer → Überlastung.
 - **Schutz:** Broadcast-Pings im Router deaktivieren.
- **SYN Flood:**
 - Angreifer sendet viele TCP-SYN-Pakete, antwortet aber nie auf das SYN-ACK.
 - Server hält Ressourcen für "halboffene Verbindungen" reserviert, bis er überlastet ist.
 - **Schutz: SYN Cookies** (Zustand wird nicht gespeichert, sondern kryptographisch in der Sequenznummer der Antwort kodiert).

3.5 Netzwerkschutzmechanismen

3.5.1 Firewall

Ein System, das den Netzwerkverkehr zwischen Zonen (z.B. LAN und Internet) überwacht und filtert.

- Filtert basierend auf Regeln (IPs, Ports, Protokolle).
- Ermöglicht Netzwerksegmentierung.

3.5.2 IDS vs. IPS

IDS und IPS Vergleich

IDS (Intrusion Detection System):

- **Passiver** Beobachter (nicht im Datenpfad/Inline).
- Analysiert Kopien des Verkehrs ("Mirror Port").
- Meldet Alarme, blockiert aber nicht selbstständig.

IPS (Intrusion Prevention System):

- **Aktiver** Schutz (Inline im Datenpfad).
- Kann bösartige Pakete in Echtzeit verwerfen/blockieren.

4 Border Gateway Protocol (BGP)

4.1 Grundlagen des Routings: LAN vs. Internet

Bevor BGP behandelt wird, ist es wichtig, den Unterschied zwischen lokalem Switching und globalem Routing zu verstehen.

4.2 Lokale Netzwerke (LAN)

In lokalen Netzen (Layer 2) erfolgt die Weiterleitung durch **Switches** basierend auf MAC-Adressen.

- **Funktionsweise:** Switches führen eine *Address Table* (MAC-Adresstabelle), die MAC-Adressen physischen Ports zuordnet.
- **Lernphase:** Ist die Ziel-MAC unbekannt, wird das Paket an *alle* Ports gesendet (*Flooding*). Antwortet der Empfänger, speichert der Switch die Port-Zuordnung.

4.3 Internet Routing

Das Internet basiert auf IP-Adressen (Layer 3). Hier übernehmen **Router** die Weiterleitung.

- **Routingtafel:** Ordnet IP-Adressbereiche (**Prefixes**) bestimmten Ausgängen (Interfaces/Ports) zu.
- **Problemstellung:** Während Switches lokal lernen können, benötigt das Internet ein Protokoll, um Routingtabellen global auszutauschen. Hier kommt BGP ins Spiel.

4.4 Border Gateway Protocol (BGP)

Autonomes System (AS)

Ein **Autonomes System (AS)** ist ein Verbund von IP-Netzwerken, der unter der Kontrolle einer einzigen administrativen Instanz steht und eine einheitliche Routing-Policy verfolgt. Jedes AS wird durch eine eindeutige Nummer identifiziert, die **ASN** (Autonomous System Number).

Beispiele für AS sind ISPs (Deutsche Telekom), Content Provider (Google) oder große Institutionen (TU Darmstadt). Das Internet ist ein Verbund zehntausender solcher AS.

4.5 Funktionsweise von BGP

BGP organisiert die Kommunikation *zwischen* diesen Autonomen Systemen.

- **Protokoll-Typ:** BGP ist ein **Path-Vector-Protokoll**. Es speichert nicht nur die Kosten, sondern den gesamten Pfad (Liste der ASNs) zum Ziel, um Schleifen zu vermeiden.
- **Transport:** BGP nutzt **TCP Port 179** für eine zuverlässige Übertragung.
- **Peering:** Zwei Router bauen eine direkte Nachbarschaft auf („Peers“), um Routeninformationen auszutauschen.
- **NLRI:** Ausgetauscht werden *Network Layer Reachability Information* (Erreichbarkeitsinformationen für IP-Präfixe).

4.6 BGP Varianten

1. **External BGP (EBGP):** Verbindet Router in *unterschiedlichen* AS.

- **Sicherheitsregel:** Die TTL (Time to Live) ist standardmäßig auf 1 gesetzt. Das erzwingt eine physische Direktverbindung.

2. **Internal BGP (IBGP):** Verbindet Router *innerhalb desselben* AS.

- Dient dazu, extern gelernte Routen im eigenen Netz zu verteilen.
- Erfordert oft ein Full-Mesh (jeder mit jedem) oder Route Reflectors.

4.7 Routing-Entscheidungen

BGP-Router müssen entscheiden, welchen Weg sie für ein Paket wählen, wenn mehrere Routen zum gleichen Ziel existieren. Die Hierarchie der Entscheidungskriterien ist für das Verständnis von Angriffen essenziell (Reihenfolge ist wichtig):

1. **Longest Prefix Match (Spezifität):** Das spezifischere Präfix gewinnt *immer*.
 - *Beispiel:* AS2 kennt Route A zu 1.1.0.0/16 und Route B zu 1.1.1.0/24.
 - Obwohl 1.1.1.0 Teil von 1.1.0.0 ist, wird für eine IP wie 1.1.1.5 die Route B gewählt, da /24 spezifischer (länger) ist als /16.
 - **Wichtig:** Diese Regel schlägt alle anderen Metriken, sogar die Pfadlänge! Dies ist die Grundlage für *Sub-Prefix Hijacking*.
2. **Shortest AS Path:** Bei gleicher Präfix-Länge gewinnt die Route, die über weniger Autonome Systeme führt.

4.8 Angriffe gegen BGP

BGP wurde ursprünglich ohne Sicherheitsmechanismen entwickelt („Vertrauensbasis“). Dies ermöglicht verschiedene Angriffe.

4.9 BGP Hijacking

Ein Angreifer (ein feindliches AS) kündigt IP-Präfixe an, die ihm nicht gehören.

Hijacking Varianten

- **Same-prefix Hijack:** Der Angreifer kündigt exakt das gleiche Präfix an wie das Opfer (z.B. Opfer: 10.10.0.0/24, Angreifer: 10.10.0.0/24).
 - *Effekt:* Das Internet teilt sich auf. Nur Router, die "näher" (kürzerer AS-Pfad) am Angreifer sind, leiten den Verkehr falsch um.
- **Sub-prefix Hijack:** Der Angreifer kündigt ein *spezifischeres* Teilnetz an (z.B. Opfer: 10.10.0.0/24, Angreifer: 10.10.0.0/25).
 - *Effekt:* Aufgrund der *Longest Prefix Match*-Regel gewinnt der Angreifer global den gesamten Verkehr für dieses Subnetz, unabhängig von der Pfadlänge. Dies ist der mächtigere Angriff.

4.10 Weitere Angriffsvektoren

- **AS_PATH Fälschung:** Der Angreifer manipuliert den AS-Pfad in seinem Announcement, um legitim zu erscheinen (fügt z.B. das Opfer-AS in den Pfad ein), oder um Pfade künstlich attraktiv zu machen.
- **Route Leaks:** Ein AS verbreitet Routen, die es gelernt hat, versehentlich weiter (oft Konfigurationsfehler). Dies kann dazu führen, dass globaler Verkehr durch ein kleines, überlastetes Netz geleitet wird.

4.11 Ziele der Angriffe

- **Blackholing (DoS):** Verkehr wird angezogen und verworfen.
- **Redirection / Man-in-the-Middle:** Verkehr wird durch den Angreifer geleitet, analysiert/manipuliert und dann zum Ziel weitergeleitet (schwer zu entdecken).
- **Subversion:** Umgehen von Zensur oder Geolokalisierung.

4.12 Reale Angriffsbeispiele

Die Vorlesung nennt drei prominente Beispiele, die die theoretischen Konzepte verdeutlichen:

1. KLAYswap (2022) - Redirection & Diebstahl:

- *Ziel:* Krypto-Dienst KLAYswap.
- *Methode:* Angreifer hijackten den IP-Bereich einer Drittanbieter-Bibliothek (KakaoTalk Messenger), die von KLAYswap geladen wurde.
- *Folge:* Der Angreifer lieferte schadhafte Code aus, da er durch den Hijack gültige SSL-Zertifikate ausstellen konnte. Nutzer überwiesen Krypto-Währung an den Angreifer.

2. China Telecom (2015-2017) - Route Leak / Subversion:

- *Vorfall:* China Telecom kündigte sich fälschlicherweise als Transit-Provider für US-Netze (Verizon) an.
- *Folge:* Inneramerikanischer Verkehr (USA → USA) wurde über China umgeleitet. Ermöglichte Spionage/-Analyse. Dauerte ca. 2,5 Jahre.

3. Cloudflare / Eletronet (2024) - Blackholing:

- *Vorfall:* Ein kleiner brasilianischer ISP kündigte versehentlich (Route Leak) das spezifische Präfix 1.1.1.1/32 an.
- *Mechanismus:* Cloudflare kündigt normalerweise 1.1.1.0/24 an. Da /32 spezifischer ist als /24, zog der brasilianische ISP den globalen DNS-Verkehr an.
- *Ergebnis:* Globaler Ausfall des DNS-Dienstes 1.1.1.1.

4.13 Gegenmaßnahmen

Es gibt keinen eingebauten Schutz in BGP. Sicherheit muss "aufgesetzt" werden.

4.14 Organisatorische Basis

- **RIR (Regional Internet Registries):** Organisationen wie RIPE (Europa) verwalten IP-Adressen und ASNs.
- IP-Adressen sind Eigentum. Wer sie "besitzt", darf sie announce.

4.15 Internet Routing Registry (IRR)

Ein Netzwerk verteilter Datenbanken, in denen Betreiber dokumentieren, welche Routen ihnen gehören.

- **Problem:** Rein manuell gepflegt, oft veraltet, ungenau. Dient nur als sekundäre Informationsquelle.

4.16 Resource Public Key Infrastructure (RPKI)

Der aktuelle Standard zur Absicherung des *Ursprungs* (Origin) einer Route. Nutzt Kryptographie.

1. **ROA (Route Origin Authorization):** Ein kryptografisch signiertes Objekt in der RPKI-Datenbank. Es legt fest:
 - Welches **AS** darf das Präfix announce? (Origin ASN)
 - Welches **Präfix** (z.B. 10.20.0.0/16)?
 - **Max Length:** Die maximal erlaubte Präfixlänge (z.B. /24). Verhindert Sub-Prefix Hijacking.
2. **ROV (Route Origin Validation):** Der Router lädt ROAs herunter und prüft eingehende BGP-Announcements.
 - **Valid:** Announcement stimmt mit ROA überein. → Route wird akzeptiert.
 - **Invalid:** AS stimmt nicht oder Präfix ist spezifischer als *Max Length* erlaubt. → Route wird verworfen ("Do Not Route").

- **Not Found:** Keine ROA vorhanden. → Route wird meist akzeptiert (da RPKI noch nicht flächendeckend ist).

Grenzen von RPKI

RPKI schützt nur den **Origin** (Wer darf announce?). Es schützt **nicht** den Pfad (**AS_PATH**).

- Ein Angreifer kann immer noch den Pfad manipulieren, solange er den korrekten Ursprung im Announcement lässt (Path-Manipulation ist weiterhin möglich).
- Implementierungsfehler in Routern können RPKI wirkungslos machen.

4.17 BGPSec

Eine Erweiterung, um auch den Pfad zu schützen.

- **Konzept:** Jedes AS signiert kryptografisch das Announcement an das nächste AS. Es entsteht eine lückenlose Signaturkette.
- **Vorteil:** Schützt vor Pfad-Manipulationen und AS-Spoofing.
- **Nachteile (Warum es kaum genutzt wird):**
 - Sehr hoher Rechenaufwand auf den Routern.
 - Erfordert lückenlose Unterstützung: Wenn ein Router im Pfad kein BGPSec spricht, bricht die Kette ("Chain of Trust").

5 Domain Name System

Das **Domain Name System (DNS)** ist ein fundamentaler Dienst des Internets, oft bezeichnet als das „Telefonbuch des Internets“. Es wurde ca. 1985 entworfen (Ursprung im ARPANET) und ursprünglich **ohne** Sicherheitsfeatures konzipiert.

Kernfunktion

Das DNS ist eine **globale, verteilte Datenbank**, die hierarchisch verwaltet wird. Die Hauptaufgabe ist die Übersetzung (Auflösung) von menschenlesbaren Hostnamen (z. B. **www.example.org**) in maschinenlesbare IP-Adressen (z. B. **93.184.216.34**).

5.1 Hierarchie und Namensraum

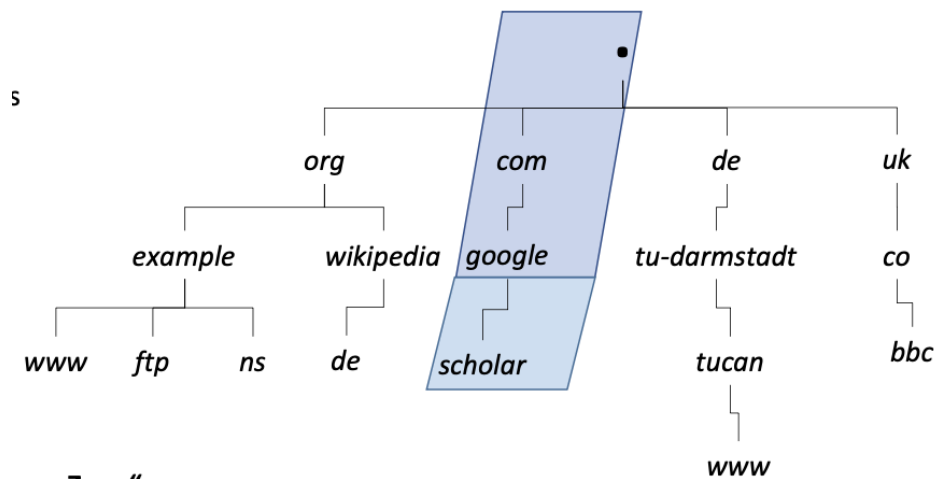
Das DNS ist als Baumstruktur organisiert:

- **Root (.)**: Die Wurzel des Baums (oft als Punkt am Ende dargestellt).
- **Top-Level-Domains (TLDs)**: Unterhalb der Root (z. B. **org**, **com**, **de**).
- **Second-Level-Domains**: Z. B. **example** in **example.org**.
- **Subdomains**: Weitere Unterteilungen, z. B. **www** oder **ftp**.

5.2 Domain vs. Zone

Es ist wichtig, zwischen einer logischen Domain und einer administrativen Zone zu unterscheiden.

- **Domain**: Ein logisch abgegrenzter Teilbereich des Internets mit eindeutigem Namen.
- **Zone**: Ein von einer **einzigen Autorität** verwalteter Bereich. Eine Zone kann eine Domain umfassen, schließt aber Subdomains aus, die an andere Autoritäten delegiert wurden (z. B. wird eine Subdomain administrativ ausgegliedert, bildet sie eine eigene Zone).



5.3 DNS Resource Records (RR)

Informationen im DNS werden in sogenannten **Resource Records** gespeichert. Ein Record besteht aus folgenden Feldern:

1. **Name (Owner)**: Identifikator (FQDN - Fully Qualified Domain Name).

2. **Type:** Art des Datensatzes (siehe Tabelle).
3. **Class:** Meist IN (Internet).
4. **TTL (Time to Live):** Gültigkeitsdauer in Sekunden (für Caching).
5. **RDLenght:** Länge der Daten in Bytes.
6. **RData:** Der eigentliche Wert (z. B. die IP-Adresse).

5.3.1 Wichtige Record-Typen

Typ	Beispiel-Daten	Zweck
A	93.184.216.34	IPv4-Adresse zum Hostnamen.
AAAA	2606:2808::1	IPv6-Adresse zum Hostnamen.
MX	mail.example.org	Mail Exchange: Mailserver für die Domain.
NS	ns.example.org	Name Server: Autoritativer Server für eine Zone.
CNAME	server1.blau.de	Canonical Name: Alias auf einen anderen Namen.
TXT	v=spf1 -all	Beliebiger Text (oft für Sicherheitsmechanismen wie SPF).

RRset

Ein **Resource Record Set (RRset)** ist die Menge aller Records mit **gleichem Namen, Typ und Klasse**. DNS überträgt immer ganze RRsets, nie einzelne Records aus einem Set (z. B. beim Load-Balancing mit mehreren A-Records für eine Domain).

5.4 Nachrichtenübermittlung und Auflösung

5.4.1 Kommunikation

DNS verwendet ein Client/Server-Modell.

- **Transport:** Standardmäßig **UDP Port 53**. TCP Port 53 wird bei großen Antworten (Zone Transfers, große DNSSEC-Pakete) verwendet.
- **Format:** Anfragen und Antworten haben dasselbe Format (Header, Question, Answer, Authority, Additional).

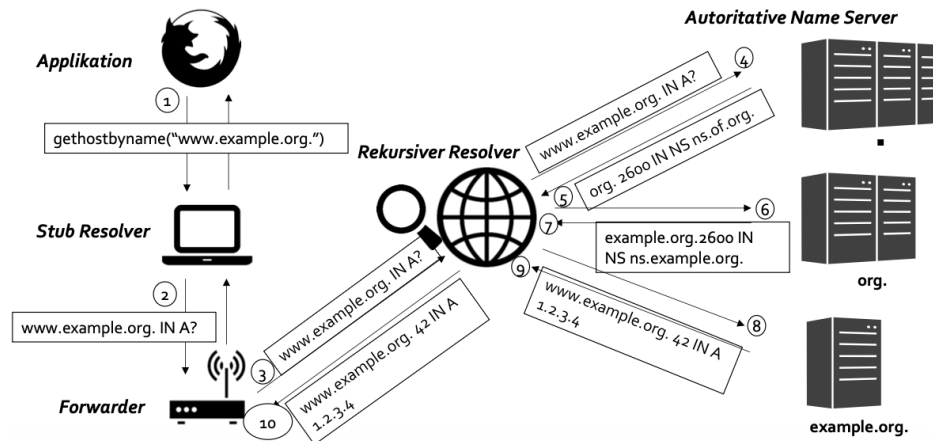
5.4.2 Server-Typen und Rollen

- **Stub Resolver:** Einfacher Client auf dem Endgerät (PC/Laptop), stellt nur Anfragen.
- **Forwarder:** Leitet Anfragen weiter (z. B. Router im Heimnetz).
- **Rekursiver Resolver:** "Der Suchende". Übernimmt die komplette Auflösung für den Client, fragt verschiedene Server ab und cacht Ergebnisse (z. B. Server beim ISP oder Google 8.8.8.8).
- **Autoritativer Name Server:** "Der Wissende". Hat die Hoheit über eine Zone und liefert die endgültigen Antworten.

5.4.3 Auflösungsverfahren

Rekursive vs. Iterative Anfragen

- **Rekursiv (RD-Flag=1):** „Besorge mir die Antwort.“ Der angefragte Server übernimmt die Arbeit und liefert das Endergebnis. Typisch zwischen *Stub Resolver* und *Rekursivem Resolver*.
- **Iterativ (RD-Flag=0):** „Gib mir die Antwort oder sag mir, wen ich fragen soll.“ Der Server liefert entweder die Daten oder einen Verweis (Referral) auf den nächsten zuständigen Server. Typisch zwischen *Rekursivem Resolver* und *Autoritativen Servern*.



5.5 DNS Cache Poisoning

Da DNS ursprünglich keine Authentifizierung besaß, vertrauen Resolver den Antworten, die sie erhalten (UDP ist verbindungslos und leicht zu fälschen).

Cache Poisoning

Einspeisung gefälschter DNS-Einträge in den Cache eines Resolvers. Ziel ist meist die **Impersonation** (Umlenkung von Nutzern auf Angreifer-Server).

5.5.1 Angriffsmethoden

1. **Klassisches Poisoning (Pre-Bailiwick):** Angreifer sendet Antwort mit zusätzlichen, gefälschten Records für fremde Domains (z. B. „Hier ist die IP für **example.org**, und übrigens ist die IP für **google.com** 6.6.6.6“).
2. **Off-Path Angriff:** Der Angreifer kann den Verkehr nicht mitlesen (Blind spoofing). Er muss die Anfrage des Resolvers an den autoritativen Server erraten und schneller antworten als der echte Server.
 - *Herausforderung:* Erraten der 16-Bit **Transaction-ID** und des **UDP-Quellports**.
3. **Kaminsky Angriff (2008):** Ein ausgeklügelter Off-Path Angriff. Um das Problem zu umgehen, dass ein Cache-Eintrag (selbst ein fehlgeschlagener) eine TTL hat und weitere Angriffsversuche blockiert:
 - Angreifer fragt nicht-existente Subdomains an (z. B. **1.bank.com**, **2.bank.com**).
 - Resolver muss jedes Mal neu beim autoritativen Server fragen.
 - Angreifer flutet gefälschte Antworten, die einen neuen, gefälschten Nameserver für die Ziel-Zone (**bank.com**) einschmuggeln.
 - *Gegenmaßnahme:* Randomisierung des UDP-Quellports (zusätzlich zur Transaction-ID), was den Suchraum auf ca. 32-Bit erhöht.
4. **Man-in-the-Middle (MITM):** Angreifer kann Verkehr mitlesen (z. B. im WLAN oder via BGP-Hijacking). Transaction-ID und Ports sind sichtbar → Triviales Poisoning möglich.

5.5.2 Bailiwick-Regel (Gegenmaßnahme)

Ein Resolver akzeptiert nur Informationen, die in den Zuständigkeitsbereich (Zone) des antwortenden Servers fallen.

- Ein Server für **example.org** darf keine Records für **google.com** liefern.
- Er darf aber Records für **www.example.org** liefern.

5.6 Gegenmaßnahme: DNSSEC

Um MITM und fortgeschrittenes Cache Poisoning zu verhindern, muss die Authentizität der Daten sichergestellt werden. **DNSSEC** (DNS Security Extensions) bietet Integrität und Authentizität durch kryptographische Signaturen, aber **keine** Vertraulichkeit (Daten sind lesbar).

5.6.1 Funktionsweise

DNSSEC bildet eine **Chain of Trust** von der Root-Zone bis zur Ziel-Domain.

- Records werden nicht verschlüsselt, sondern **signiert**.
- Eltern-Zonen signieren den Hash der Schlüssel ihrer Kinder (Delegation).

5.6.2 Neue Record-Typen

- **RRSIG**: Enthält die digitale Signatur eines RRsets.
- **DNSKEY**: Enthält den öffentlichen Schlüssel (Public Key) zum Überprüfen der Signatur.
- **DS** (Delegation Signer): Fingerprint (Hash) des Schlüssels der Unterzone (liegt in der Elternzone, stellt die Vertrauenskette her).
- **NSEC / NSEC3**: Dient dem *Authenticated Denial of Existence* (Beweis, dass ein Name **nicht** existiert).

5.6.3 Problem: Zone Enumeration

Da DNSSEC beweisen muss, dass ein Name *nicht* existiert, geben NSEC-Records Informationen über den „nächsten“ existierenden Namen preis.

- **NSEC Walking**: Angreifer fragt nacheinander Namen ab und erhält durch die NSEC-Antworten („Zwischen A und F gibt es nichts“) die Liste aller existierenden Domains.
- **NSEC3**: Hasht die Namen. Angreifer können die Hashes jedoch offline via Brute-Force (GPU) knacken, da der Namensraum (z. B. www, mail) klein ist.
- **Lösung (Live Signing / White Lies)**: Server berechnet Signaturen on-the-fly. Bei Anfrage nach `ghost.example.com` behauptet der Server: „Der Vorgänger ist **ghost** und der Nachfolger ist **ghost\000**“. Der Bereich ist so klein, dass er nur den angefragten Namen abdeckt. Verhindert Enumeration effektiv.

5.7 Transport-Sicherheit (Privacy)

DNSSEC schützt die Daten, verschlüsselt aber nicht den Transport. Wer wissen will, welche Webseiten ein Nutzer besucht, kann den DNS-Verkehr mitlesen.

DoT vs. DoH

Beide Protokolle verschlüsseln die Kommunikation zwischen Stub-Resolver und Rekursivem Resolver (Last-Mile-Security).

- **DoT (DNS over TLS)**: Dedizierter Port (TCP 853).
- **DoH (DNS over HTTPS)**: Versteckt DNS im HTTPS-Traffic (TCP 443). Schwerer zu blockieren/-filtern.

Wichtig: Sie schützen vor Lauschern auf der Leitung, garantieren aber **nicht** die Echtheit der Daten vom autoritativen Server (dafür wird DNSSEC benötigt).

5.8 Weitere Angriffe auf DNS-Infrastruktur

5.8.1 DNS Amplification DDoS

Ein **Reflection**-Angriff unter Ausnutzung des UDP-Protokolls.

1. Angreifer sendet Anfrage an offene DNS-Server.

2. **IP-Spoofing:** Absender-Adresse ist die des Opfers.
3. **Amplification:** Die Anfrage ist klein (z. B. 60 Byte), die Antwort ist riesig (z. B. 3000 Byte, Faktor 50x).
4. Der DNS-Server flutet das Opfer mit den großen Antworten.

Gegenmaßnahmen: Response Rate Limiting (RRL) auf Servern, Verhinderung von IP-Spoofing im Netzwerk (BCP 38).

5.8.2 DNS Tunneling

Umgehung von Firewalls oder Exfiltration von Daten.

- Daten werden in Subdomains kodiert (z. B. `geheimespasswort.angreifer.com`).
- Der autoritative Server des Angreifers empfängt die Anfrage und dekodiert die Daten.
- Antworten können Steuerbefehle (C2) enthalten (via TXT oder CNAME Records).

5.9 DNS-basierte Sicherheitsmechanismen für E-Mail

Das DNS wird genutzt, um die Sicherheit anderer Dienste (v.a. E-Mail) zu erhöhen.

5.9.1 SPF (Sender Policy Framework)

Schutz gegen E-Mail-Spoofing (Versand unter falschem Namen).

- Ein **TXT-Record** in der Domain definiert, welche IP-Adressen Mails für diese Domain versenden dürfen.
- **Syntax:** `v=spf1 [Mechanismen] [Qualifier]all`

Qualifier	Bedeutung
+	Pass (Standard, wenn weggelassen).
-	Fail (Mail ablehnen).
~	Soft-Fail (Mail annehmen, aber markieren/Spam-Ordner).
?	Neutral.

Beispiel: `v=spf1 mx ip4:1.2.3.0/24 -all`

Bedeutet: Die MX-Server und das Subnetz 1.2.3.0/24 dürfen senden. Alles andere (`-all`) wird abgelehnt.

5.9.2 Weitere Mechanismen

- **DKIM:** Signieren von E-Mails; Public Key liegt im DNS.
- **DANE:** Bindung von TLS-Zertifikaten an DNS-Namen (via TLSA-Records), benötigt zwingend DNSSEC.