
lrz
Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

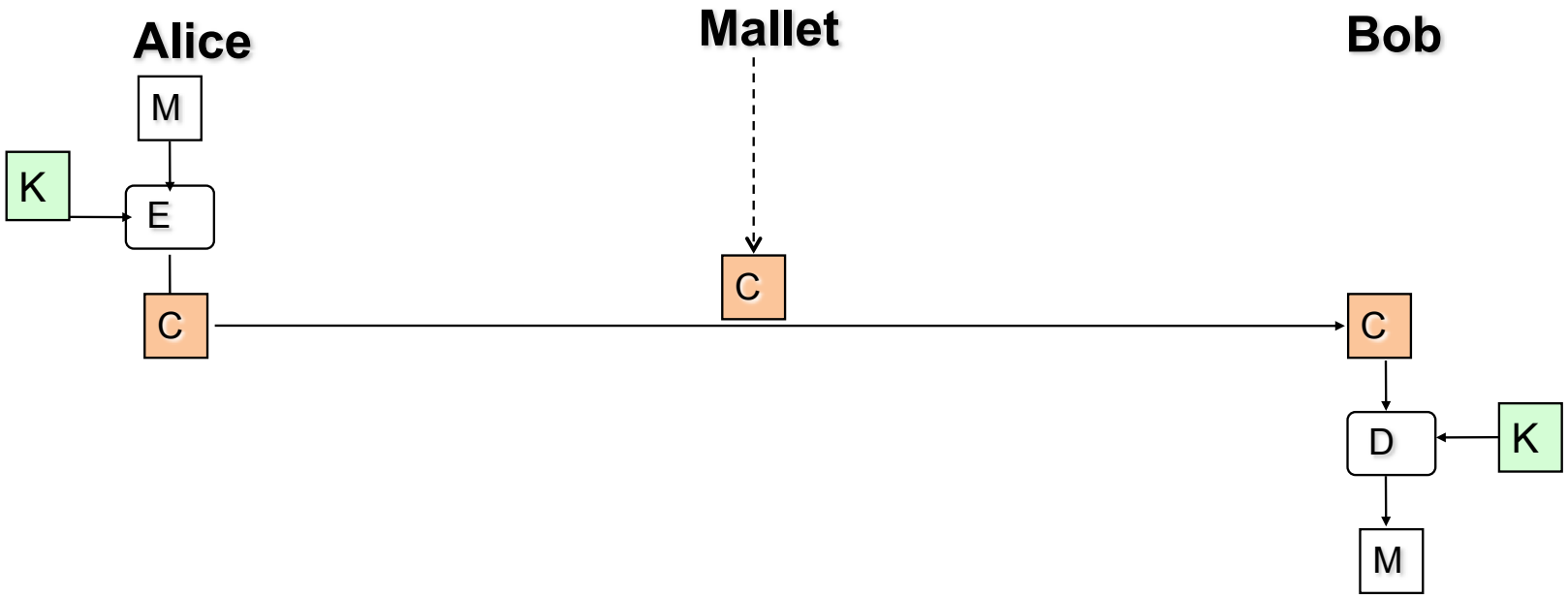
Kapitel 10: Sicherheitsmechanismen

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - ❑ Mandatory Access Control (MAC)
    - ❑ DAC
5. Identifizierung

Vertraulichkeit (Confidentiality)

- Schutz der Daten vor unberechtigter Offenlegung
- Wie kann Vertraulichkeit realisiert werden?
  - Durch Verschlüsselung (Encryption)
  - Mallet kann Chiffrentext mangels Kenntnis des Schlüssels nicht nutzen



Integrität

- Erkennung von Modifikationen, Einfügungen, Löschungen, Umordnung, Duplikaten oder Wiedereinspielung von Daten
- Wie kann Integrität gewährleistet werden?
  - Modifikation, Einfügung, Löschung, Umordnung?
    - Kryptographischer Hash-Wert über die Daten
  - Duplikate, Wiedereinspielung von Daten?
    - Kryptographischer Hash-Wert + „gesicherte“ Sequenznummern und/oder Zeitstempel

Integrität durch Verschlüsselung?

- Ist Verschlüsselung ein Mechanismus zur Integritätssicherung?
  - In Allgemeinheit NEIN: „Blinde“ Modifikation des Chiffrentextes möglich
  - Abhängig vom Verschlüsselungsverfahren und den Daten kann es passieren, dass die Veränderung nicht automatisch erkannt wird
  - Auch mit semantischem Wissen kann Veränderung unbemerkt bleiben
  - Unwahrscheinliches aber mögliches Bsp.: Angreifer kippt Bit in verschlüsselter Überweisung; Entschlüsselung liefert 1000 statt 10 €

Angriff auf Mechanismen zur Integritätssicherung

- Angreifer verändert unbemerkt Daten und Hash-Wert
- Deshalb: Hash-Wert und ggf. Sequenznummern müssen vor Veränderungen geschützt werden
  - Sequenznummern oder Timestamp als Teil der geschützten Daten werden (automatisch) durch Hash geschützt
  - Sequenznummern im Protokoll-Header sind gesondert (durch Hash) zu schützen
  - Hash selbst wird z.B. durch Verschlüsselung geschützt
    - In diesem (Spezial-)Fall ist Verschlüsselung ein wichtiger Beitrag zur Integritätssicherung
    - Bei verschlüsselten Hashes lassen sich „blinde“ Veränderungen am Chiffrentext automatisch erkennen
    - Übertragen wird < m, E(H(m)) >
    - Test beim Empfänger: Ist D(E(H(m))) gleich dem selbst berechneten Wert von H(m)?

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - Mandatory Access Control (MAC)
    - DAC
5. Identifizierung

Arten

- Bei Authentisierung wird unterschieden zwischen:
  1. Authentisierung des Datenursprungs
  2. Benutzerauthentisierung
  3. Peer Entity Authentisierung
     - Einseitig (z.B. Client prüft Server, aber nicht umgekehrt), oder
     - Zwei- bzw. mehrseitige Authentisierung

- Grundsätzliche Möglichkeiten zur Authentisierung:
  1. Wissen (Something you know)
  2. Besitz (Something you have)
  3. Persönliche Eigenschaft (Something you are)
  4. Kombinationen aus 1. – 3.
  5. (Delegation - Someone who knows you)

Benutzerauthentisierung

- Wissen
  - Passwort, Passphrase (Unix Passwort Verfahren, vgl. Kap. 3)
  - Einmal-Passwort
  - PIN
  - .....
- Besitz
  - Smartcard, Token, („physischer“) Schlüssel, Token-App auf Smartphone
  - Kryptographischer Schlüssel als Datei
- Eigenschaft
  - Biometrie:
    - Fingerabdruck
    - Stimmerkennung
    - Gesichtserkennung
    - Iris-Scan
    - Hand-Geometrie; Venenbild der Hand
    - Behavioral Biometrics, z.B.
      - Anschlags- oder Andruck-Charakteristik beim Schreiben
      - Lippenbewegungen

Einmalpasswörter

- Motivation
  - Nutzung nicht vertrauenswürdiger Geräte
  - Erwartetes „Shoulder-Surfing“, z.B. bei Messen / Präsentationen

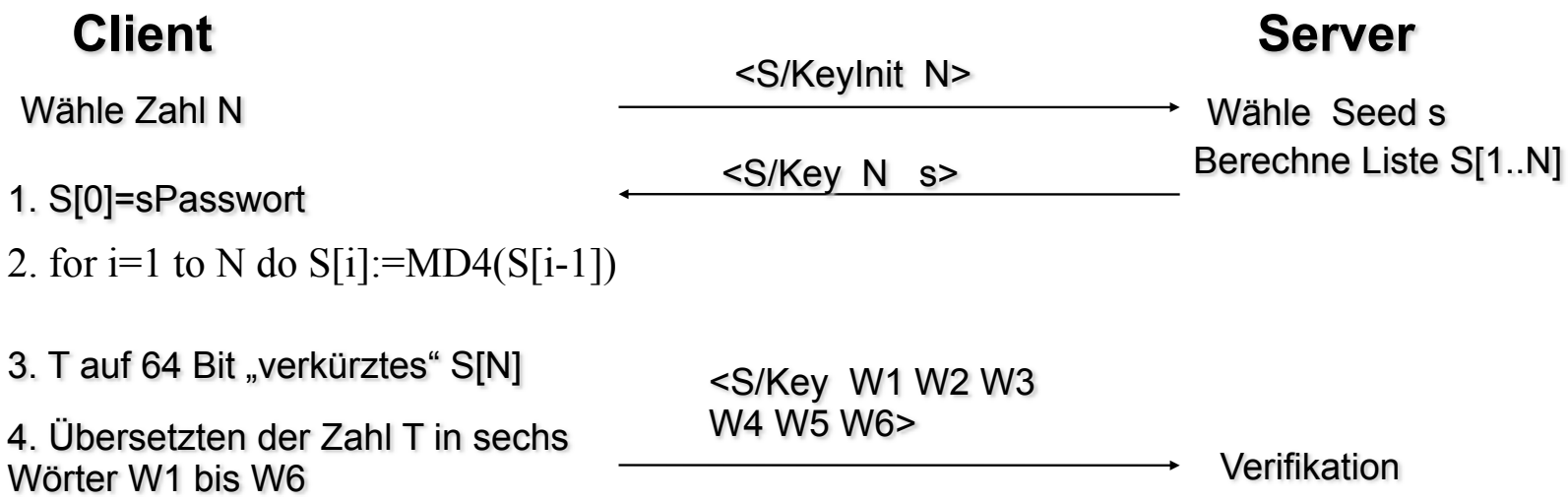
- Abgehörtes Passwort soll für den Angreifer möglichst nutzlos sein:
  - Passwort kann nicht mehrfach verwendet werden
  - Begrenzte Gültigkeitsdauer nach Beginn der Nutzung
  - Aus dem (n-1)ten Passwort lässt sich das n. Passwort nicht ableiten

- Design-Kriterien aus den 1990ern:
  - Benutzer gibt Anzahl der Einmalpasswörter vor
  - Keine Verschwendung von kostbarem Speicherplatz durch Passwort-Listen
  - Keine Out-of-Band-Kommunikation (z.B. Nutzung eines Mobiltelefons)

- Bekannte Verfahren: S/Key und OTP

Einmal-Passwort Verfahren: S/Key (1995)

- Authentisierungsserver kennt Passwort des Benutzers



- Bei nächster Authentisierung wird S[N-1] verwendet, dann S[N-2], usw.
- Entwickelt von Bellcore [RFC 1760]

Details

- Verkürzungsfunktion
  - T := S[N] (128 Bit lang)
    T[0-31] := T[0-31] XOR T[64-95]
    T[32-63] := T[32-63] XOR T[96-127]
  - Weiter verwendet wird T[0-63]

- Eingabe einer 64 Bit Zahl ist fehleranfällig, daher

- Übersetzungsfunktion für T
  - Ergebnis 6 kurze (1 bis 4 Zeichen lange) englische Wörter
  - Wörterbuch mit 2048 Wörtern (in RFC 1760 enthalten)
  - Je 11 Bit von T liefern - als Zahl interpretiert - die Nummer des Wortes

  - Bsp. für einen solchen „Satz“: HIT HARD LIKE A DOOM GOAT

Bewertung

- Gute Hashfunktionen bieten ausreichend Schutz vor dem Ableiten des n. Passworts aus den vorherigen n-1 Passwörtern
- Ohne weitere Schutzmaßnahmen anfällig für Man-in-the-Middle Angriffe
- Benutzer muss Reihenfolge der Passwörter genau einhalten

OTP (One Time Password System)

- Entwickelt von Bellcore [RFC 2289] als Nachfolger für S/Key
- Schutz vor Race Angriff:
  - S/Key Implementierungen erlauben i.d.R. mehrere gleichzeitige Sessions mit einem Passwort
  - Angreifer kann abgehörtes Passwort für kurzen Zeitraum nutzen (Replay Angriff)
- Jede Anmeldung mit OTP braucht eigenes One-Time Passwort
- Sonst nur marginale Änderungen

- Unterstützt verschiedene Hash-Funktionen (MD4, MD5, SHA,..)
- Akzeptiert Passwort auch in Hexadezimal-Notation
- Passwort muss mind. 10 und kann bis 64 Zeichen lang sein
- Verwendung von IPSec wird „empfohlen“

Angriffe

- Dictionary Attack:
  - Alle Nachrichten werden im Klartext übertragen, z.B.
  - Angreifer kann mit diesen Informationen versuchen, das Passwort des Benutzers zu brechen, z.B.:
    Wort 1: Automobile: BAD LOST CRUMB HIDE KNOT SIN
    Wort k: wireless-lan: A GUY SWING GONE SO SIP
  - Daher empfiehlt OTP die Verschlüsselung über IPSec
- Sicherheit hängt essentiell von der Sicherheit des gewählten Passwortes ab
- Spoofing-Angriff:
  - Angreifer gibt sich als Authentisierungs-Server aus
  - Damit Man-in-the-Middle Angriff möglich
  - Auch hier: OTP empfiehlt die Verwendung von IPSec zur Authentisierung des Servers

Smartcards

- Klassifikation und Abgrenzung:
  1. Embossing Karten (Prägung auf der Karte, z.B. Kreditkarte)
  2. Magnetstreifen-Karten; nur Speicherfunktion (alte EC-Karte)
  3. Smartcard (eingebettete Schaltung):
     - Speicherkarten
     - Prozessor-Karten
     - Kontaktlose Karten

       - Bsp.: Prozessor-Karte mit Fingerabdruck-Sensor

- Zugangsdaten werden auf Karte gespeichert oder erzeugt
  - Schutz der Daten ggf. durch PIN/Passwort und/oder Verschlüsselung
  - PIN-/Passworteingabe setzt vertrauenswürdiges Eingabegerät

RSA SecurID

- SecurID Token
  - generiert jede Minute eine neue Zahl, die nur durch den zentralen Authentifizierungsserver vorhersagbar ist
  - Diese 6- bis 8-stellige Zahl muss zusammen mit dem Benutzerpasswort eingegeben werden (= 2-Faktor-Authentisierung)
- Unterstützung in kommerziellen VPN-Gateways und OpenSSH
- Zahl wird per AES „berechnet“; Eingabe ist eine „echte“ Zufallszahl (Seed) bei der Fertigung des Tokens.
- Aktuelle Produktversion hat USB-Schnittstelle, die als Smartcard / Zertifikatsspeicher dient. Auch als App verfügbar.

Details

- Die angezeigte Zahl ist eine AES-Verschlüsselung
  - der Anzahl der seit 01.01.1986 00:00 Uhr vergangenen Sekunden (Klartext)
  - mit der bei der Fertigung gewählten Zufallszahl als Schlüssel

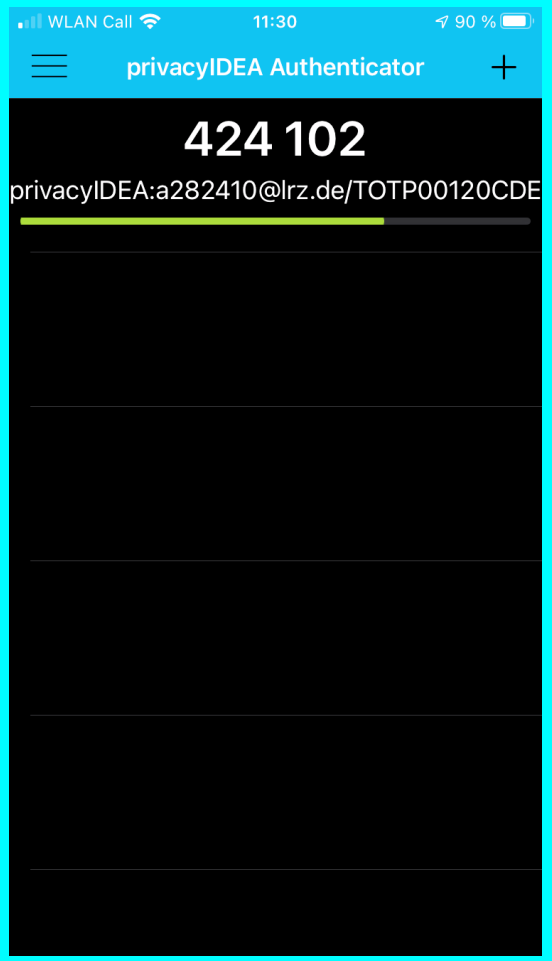
- Damit auch Zeitabweichungen der Quartzuhren in den Token berücksichtigbar

- „Lebensdauer“ je nach Modell 1-5 Jahre; das Gerät schaltet sich zu einem vorgegebenen Zeitpunkt ab.

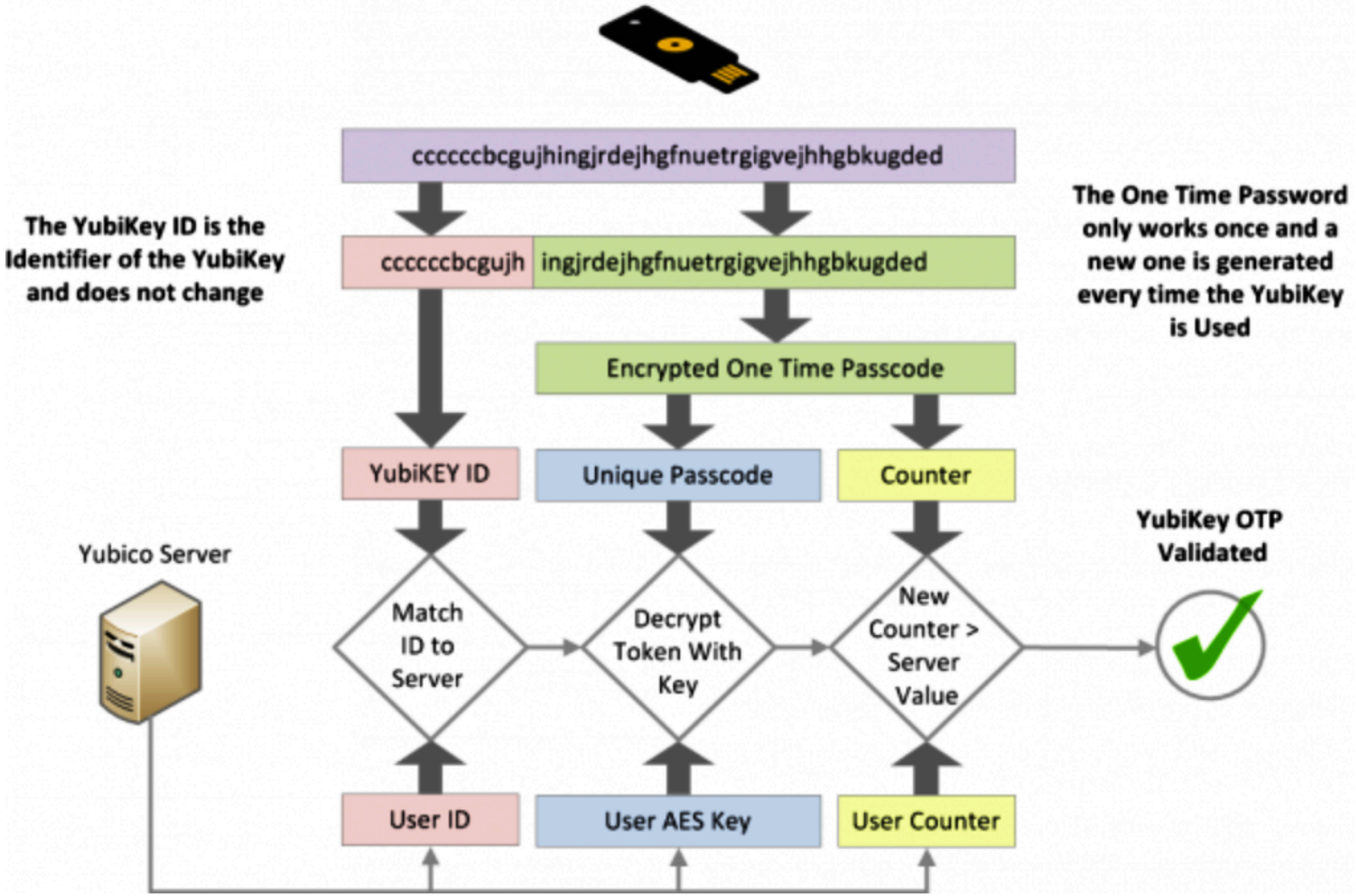
- Kein „Batteriewechsel“: Hardwaremanipulation führt immer zu Hardwarebeschädigung / -zerstörung
- Kosten ca. 25 Euro pro Token (je nach Mengenrabatt)

2FA im LRZ

- Serverseitig PrivacyIDEA - ermöglicht Vielzahl von Faktoren
- Client-seitig
  - PrivacyIDEA App mit TOTP
  - YubiKey mit OTP im AES Mode
- TOTP (RFC 6238)
  - TOTP = HMAC(Secret Key, Current Time)
  - TOTP wird zusätzlich zum Passwort eingegeben



2FA im LRZ: Yubikey

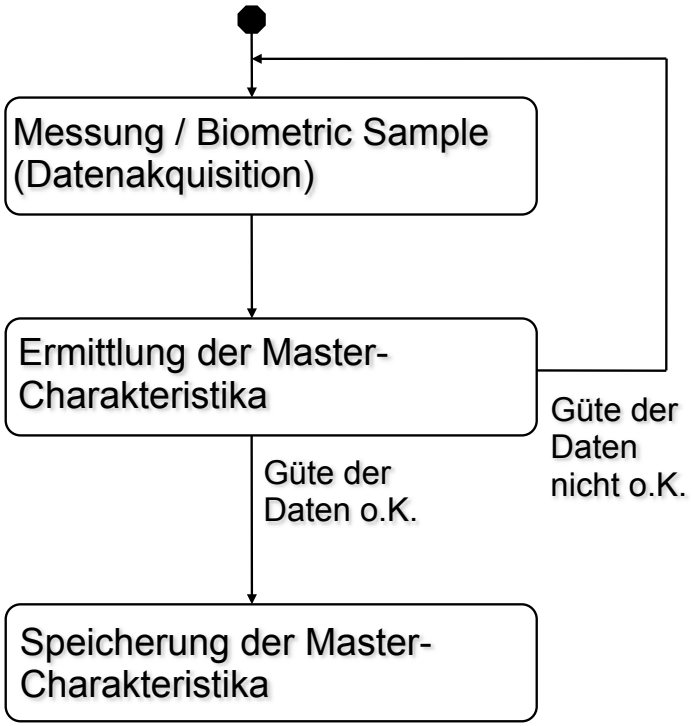


Inhalt

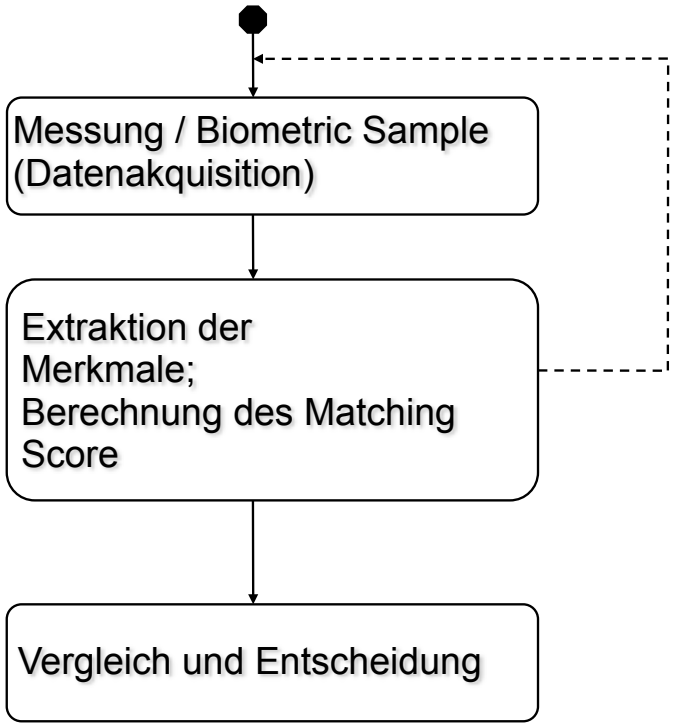
1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - Mandatory Access Control (MAC)
    - DAC
5. Identifizierung

Biometrie: allgemeines Vorgehen

- Initialisierung des Systems pro Nutzer
  - Viele Messungen möglich



- Authentisierung
  - I.d.R. nur eine oder sehr wenige Messungen möglich



Anwendungen

- Anmeldung an PCs / Notebooks
- Zutrittskontrolle
  - zu Räumen in Bürogebäuden, Rechenzentren, ...
  - Zoo Hannover hat Gesichtserkennungssystem
  - Fingerabdruckleser in Fitness-Studios etc.
- Biometrischer Reisepass
- Kriminalistik, z.B.
  - Fingerabdruck
  - Gebissabdruck
- Bezahlen im Supermarkt (Datenschutz?)

- Warum ist ein Geldautomat mit Fingerabdruckleser keine gute Idee?

Beispiel Fingerabdruck

- Identifikation anhand des Fingerabdrucks hat lange Geschichte
- Merkmale von Fingerabdrücken sind gut klassifiziert

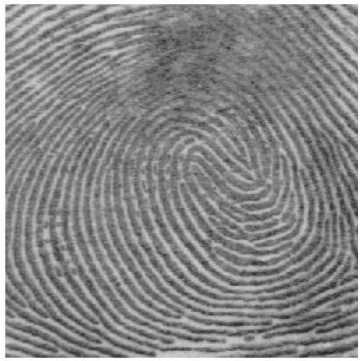
Bogen

gespannter Bogen

linke Schleife

rechte Schleife

Knäuel

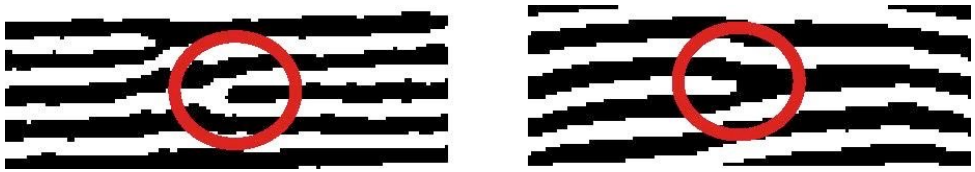


Doppelschleife

Karu, K. und A. Jain: Fingerprint Classification. Pattern Recognition, 29(3):389–404, 1996.

Fingerabdruck: Merkmalsextraktion

- Die vorgestellten Klassen lassen sich leicht unterscheiden
- Extraktion sogenannter Minuzien (Minutiae):
  - Repräsentation basierend auf charakteristischen Rillenstrukturen
  - Problem der Invarianz bei unterschiedlicher Belichtung oder unterschiedlichem Druck
    Folgende Beispiele sind äquivalent (entstanden durch untersch. Druck)

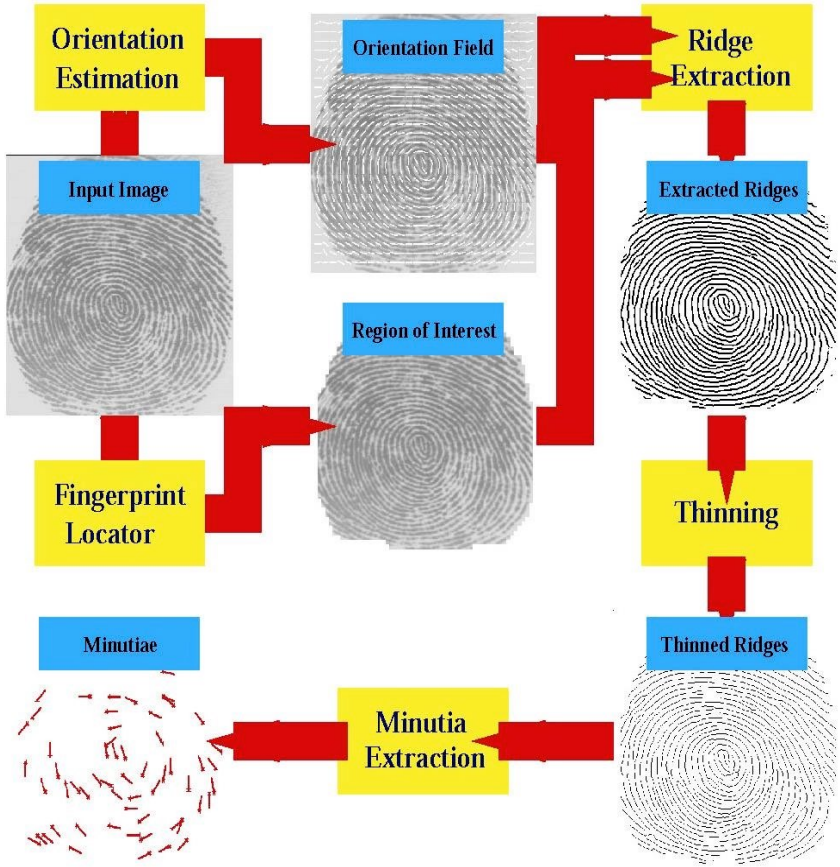


Rillen-Ende Rillen-Verzweigung

  - Solche äquivalente Rillenstrukturen werden zu einer Minuzie zusammen-gefasst
- Merkmale: Lage der Minuzien
  - Absolut bezüglich des Abdrucks und relativ zueinander
  - Orientierung bzw. Richtung

Fingerabdruck: Minutiae Extraktion

- Algorithmus: Beispiel aus [JHPB 97]

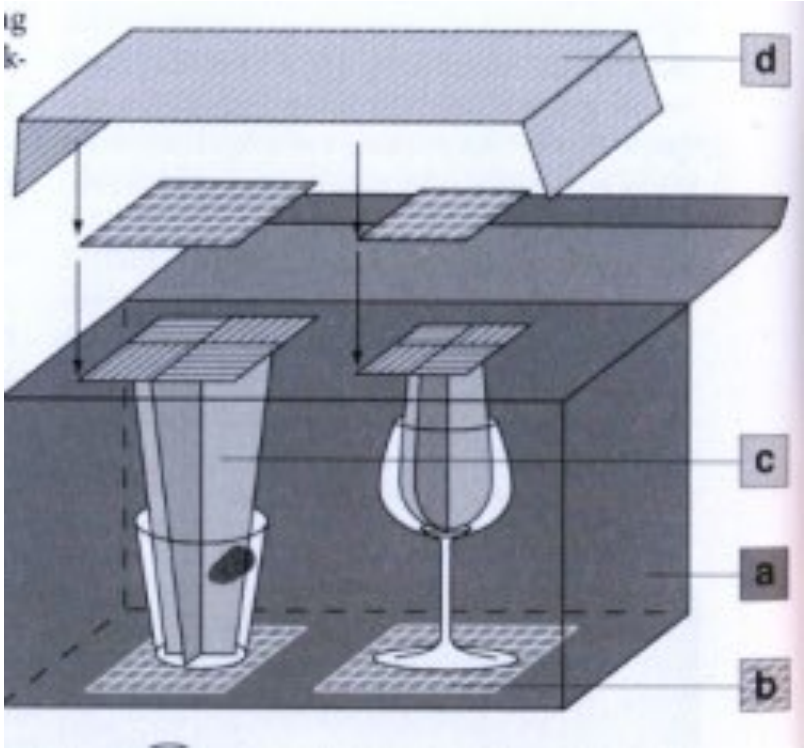


Fingerabdruck: Angriffe

- Sicherheit hängt auch von der Art des Sensors ab
  - Optische Sensoren (Lichtreflexion)
  - Kapazitive Sensoren (elektrische Leitfähigkeit, Kapazität)
  - Temperatur, Ultraschall,.....
- Optische Sensoren können einfach „betrogen“ werden [MaMa 02, Mats 02]
  - Finger-Form mit Hilfe von warmem Plastik abnehmen
  - Form mit Silikon oder Gummi ausgießen
  - Gummi-Finger verwenden
  - Akzeptanzrate bei vielen optischen Sensoren über 80 %
  - Finger-Form kann auch mit einem Fingerabdruck auf Glas erzeugt werden, d.h. der „Original-Finger“ ist nicht erforderlich
- Kapazitive Sensoren weisen Gummi-Finger i.d.R. zurück
- Verbesserung durch kombinierte Sensoren
- iPhone-Sensor: http://www.heise.de/ct/artikel/Der-iPhone-Fingerabdruck-Hack-1965783.html

2008: CCC veröffentlicht Schäuble-Fingerabdruck

- Protest gegen zunehmende Erfassung biometrischer Daten, z.B. für Reisepässe
- Von einem Wasserglas während einer politischen Veranstaltung genommen
- Fingerabdruck-Attrappe über Mitgliederzeitschrift verteilt
- Bundesinnenministerium sah E-Pass dadurch nicht in Frage gestellt



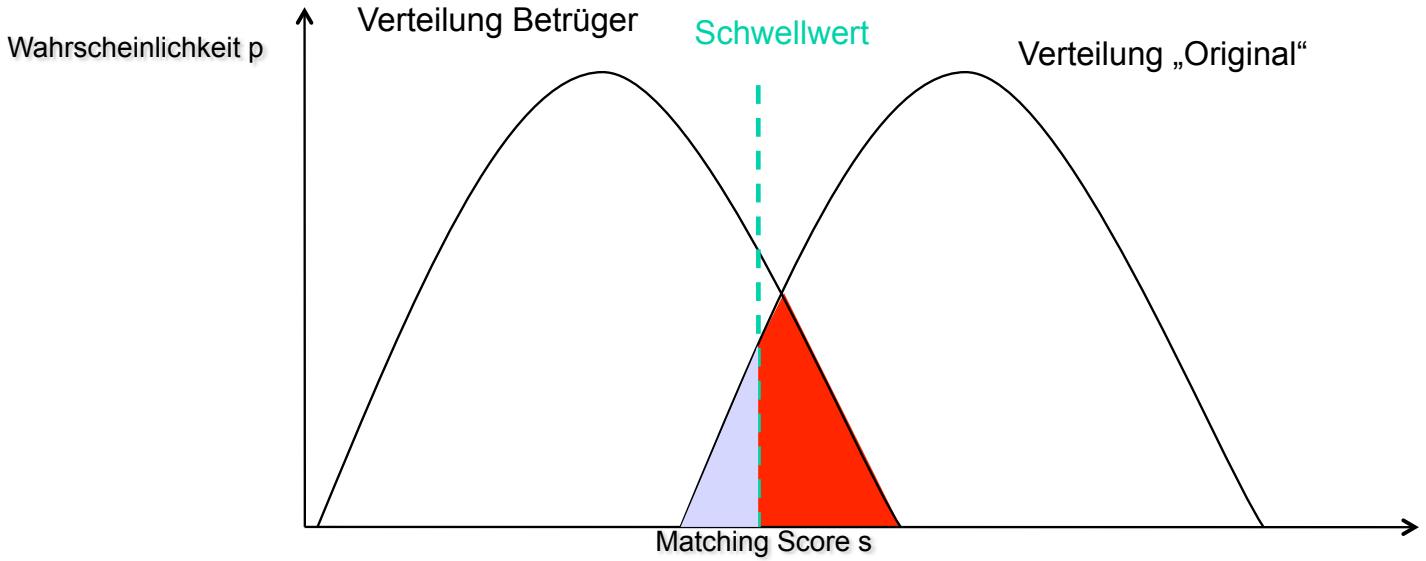
- Im Rückblick: Aktion hatte nur kurze Medien-Wirksamkeit

Fingerabdruckscanner: Lebenderkennung

- Puls
- Tiefenmuster
- Wärmebild
  - totes Gewebe absorbiert Infrarotlicht
- Blutzirkulation
- Messen der Sauerstoff-Sättigung
- Messen des elektrischen Widerstands
- Feuchtigkeit

Fehlerarten

- Biometrische Systeme sind fehlerbehaftet
- Fehlerarten:
  - Falsch Positiv / Falschakzeptanzrate (Mallet wird als Alice authentisiert)
  - Falsch Negativ / Falschrückweisungsrate (Alice wird nicht als Alice identifiziert)
- Fehler sind abhängig von Schwellwerteinstellungen



Fehlerraten

- Abschätzung der Fehlerraten:
  N: Anzahl der Identitäten
  FP: Falsch Positiv (Falschakzept.)
  FN: Falsch Negativ (Falschrückw.)
- Es gilt [PPK03]:

$$FN(N) \cong FN$$

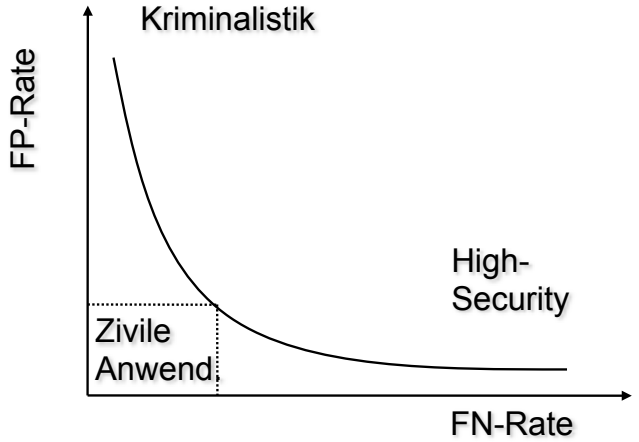
$$FP(N) \cong 1 - (1 - FP)^N \cong N \times FP$$

  falls

$$N \times FP < 0,1$$

- Anwendungsbeispiel:
  - N = 10.000
  - FP = 0,00001 (0,001 %)
  - Damit FP(N) = 0,1
  - D.h. Fehlerrate von 10 %; Angreifer probiert seine 10 Finger und hat nennenswerte Chance
  - Praxisforderung: FP(N) < 1/100.000

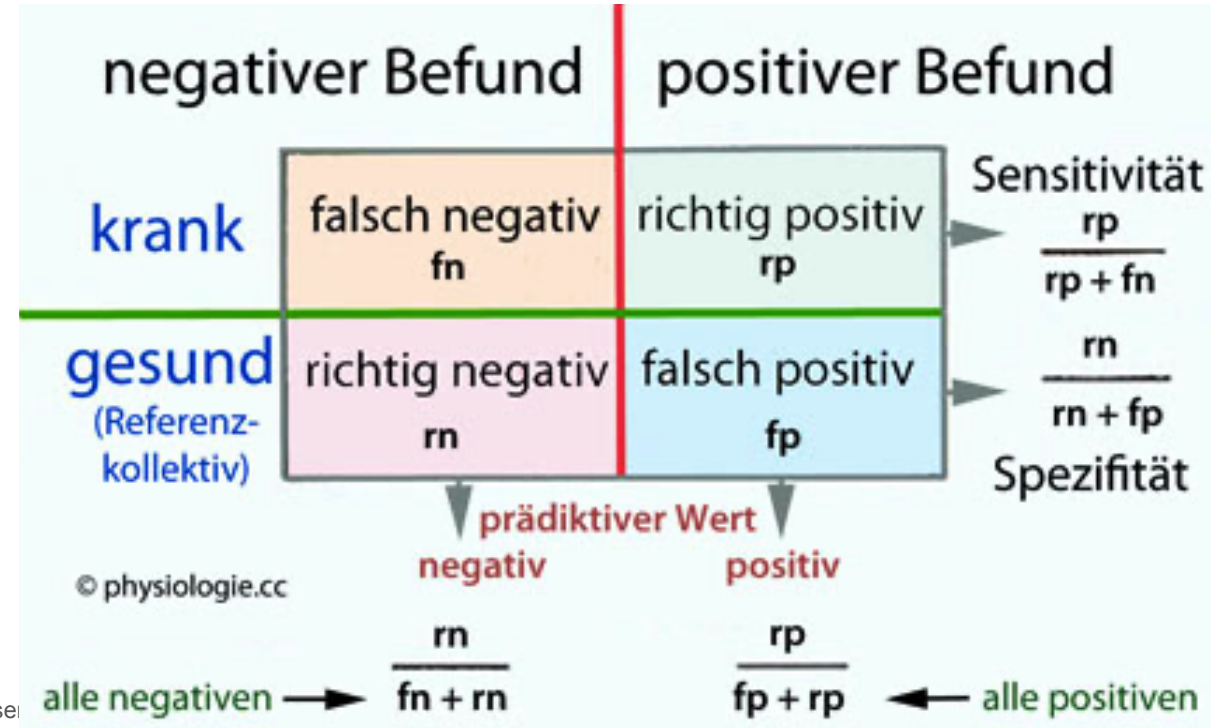
- Fehlerraten, bzw. Einstellung der Schwellwerte abhängig vom Anwendungsszenario



- Platzierung von Anwendungen?
  - Hohe Sicherheitsanforderungen
  - Kriminalistische Anwendungen
  - “Zivile” Anwendungen

Fehlerraten in der Medizin

- Sensitivität und Spezifität medizinischer Tests
- Am Bsp. von Covid-19 Tests
  - Sensitivität - Erfasst die Sicherheit der Erkrankung
  - Spezifität - Wahrscheinlichkeit, dass gesunde als gesund erkannt werden



[http://physiologie.cc/I.10.htm]

Multimodale Systeme

- Sicherheit lässt sich durch multimodale Systeme deutlich erhöhen
- Multimodale Systeme kombinieren verschiedene Verfahren

	Wissen	Besitz	Biometrie
Wissen			
Besitz			
Biometrie			

- Auch verschiedene biometrische Verfahren lassen sich kombinieren:
  - Erhöhung der Sicherheit
  - Verringerung der Fehlerraten
  - Z.B. Iris-Scan mit Spracherkennung kombiniert

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - ❑ Mandatory Access Control (MAC)
    - ❑ DAC
5. Identifizierung

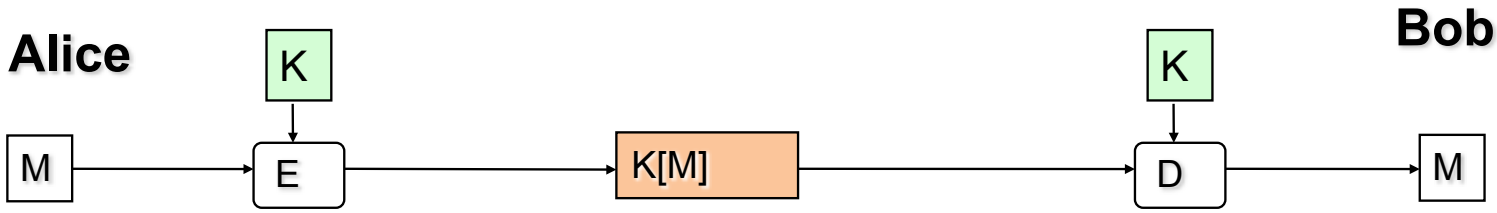
Authentisierung des Datenursprungs

- Möglichkeiten zur Authentisierung des Datenursprungs bzw. zur Peer-Entity-Authentication:

1. Verschlüsselung der Nachricht (Authentisierung erfolgt mittelbar durch Wissen, d.h. Kenntnis des Schlüssels)
2. Digitale Signatur
3. Message Authentication Code (MAC)
   MAC = Hashverfahren + gemeinsamer Schlüssel
4. Hashed Message Authentication Code (HMAC)

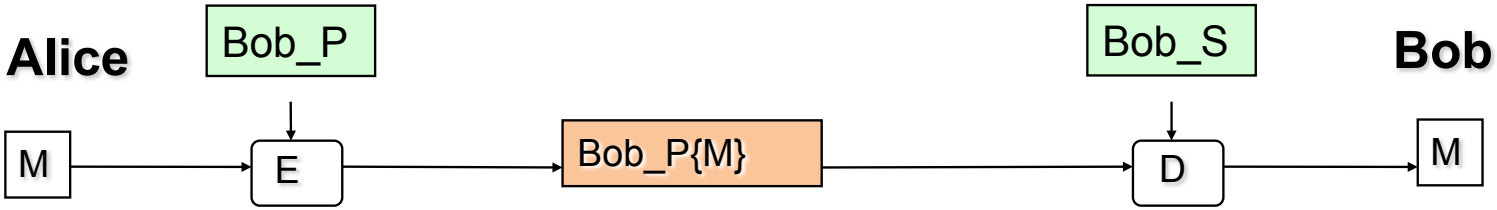
   - Kombinationen der angegebenen Verfahren

Authentisierung durch symm. Verschlüsselung



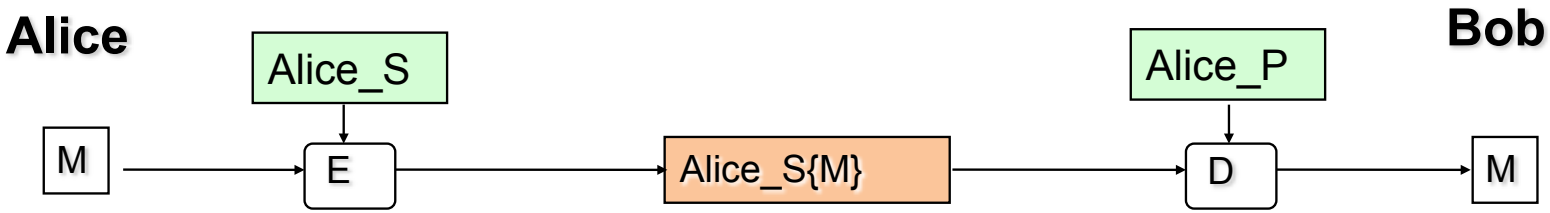
- Merkmale:
  - Authentisierung des Datenursprungs (Nachricht kann nur von Alice stammen, wenn der Schlüssel nur Alice und Bob bekannt ist)
  - Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten (nur Alice und Bob kennen K)
- „Nachteile“:
  - Sender kann die Sendung leugnen (Bob könnte sich die Nachricht auch selbst geschickt haben)
  - Alice / Bob können Zugang / Empfang nicht beweisen

Authentisierung durch asym. Verschlüsselung



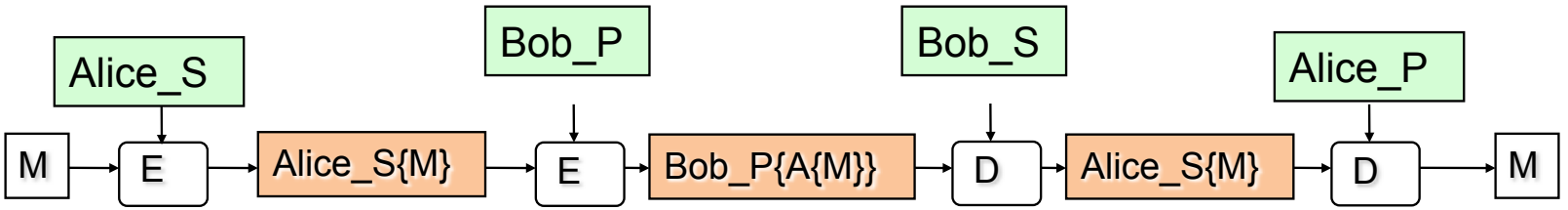
- Merkmale:
  - Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten (nur Bob kennt seinen privaten Schlüssel)
  - KEINE Authentisierung des Datenursprungs
    (Jeder kann senden, weil jeder Bobs Public Key haben kann)
  - Sender kann die Sendung leugnen
    (könnte irgendjemand anderes gewesen sein)
  - Alice / Bob können Zugang / Empfang nicht beweisen

Digitale Signatur



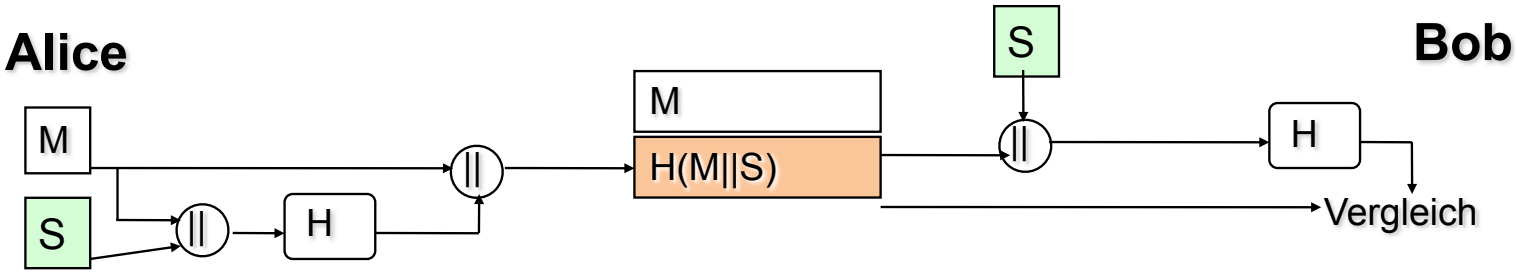
- Merkmale:
  - ❑ Authentisierung des Datenursprungs (Nachricht kann nur von Alice stammen; nur Alice kennt ihren geheimen Schlüssel)
  - ❑ Jeder kann die Signatur verifizieren (auch ohne Mithilfe von Alice)
  - ❑ Alice kann die Sendung nicht leugnen
  - ✶ Bob wird nicht authentisiert
  - ✶ Keine Vertraulichkeit (Jeder kann Nachricht lesen, jeder „kennt“ öffentlichen Schlüssel von Alice)
  - ✶ Alice kann Zugang nicht beweisen

Asym. Verschlüsselung + Signatur



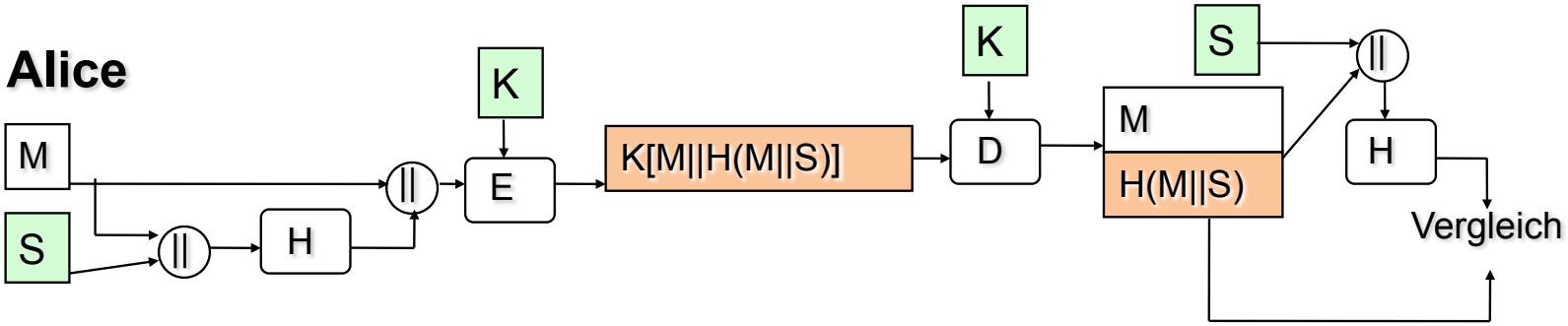
- Merkmale:
  - Authentisierung des Datenursprungs
  - Nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten
  - Vertraulichkeit der Signatur
  - Alice kann Sendung nicht leugnen
  - Operationen für Signatur und asymmetrische Verschlüsselung sind „teuer“
  - Alice kann Zugang nicht beweisen
  - Bei allen Verfahren bisher keine Integritätssicherung („blinde“ Modifikation des Chiffretextes wird nicht erkannt)

Verwendung von Hash-Fkt. zur Authentisierung



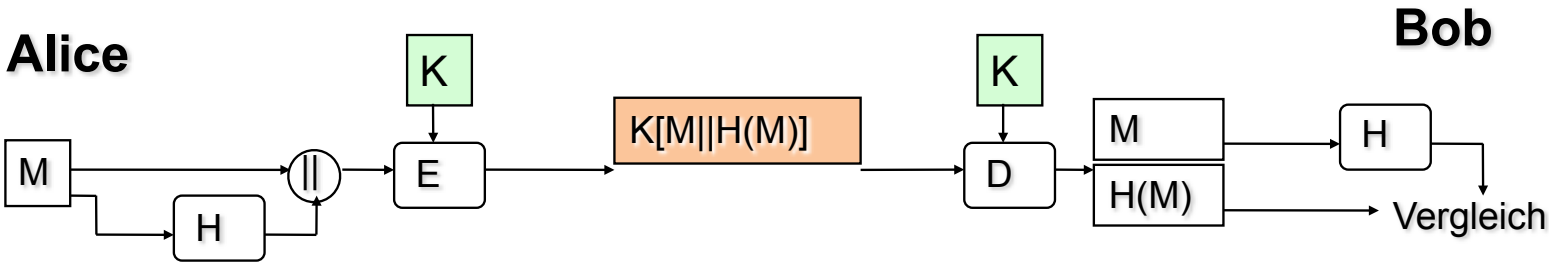
- Authentisierung des Datenursprungs (durch „Geheimnis“ S)
  - Nachricht wird mit S konkateniert und dann der Hash berechnet
- (Daten-) Integrität (durch Hash)
- Keine Vertraulichkeit, jeder kann M lesen
- Alice kann Sendung leugnen
- Alice/Bob können Zugang / Empfang nicht beweisen

Verwendung von Hash-Fkt. zur Authentisierung

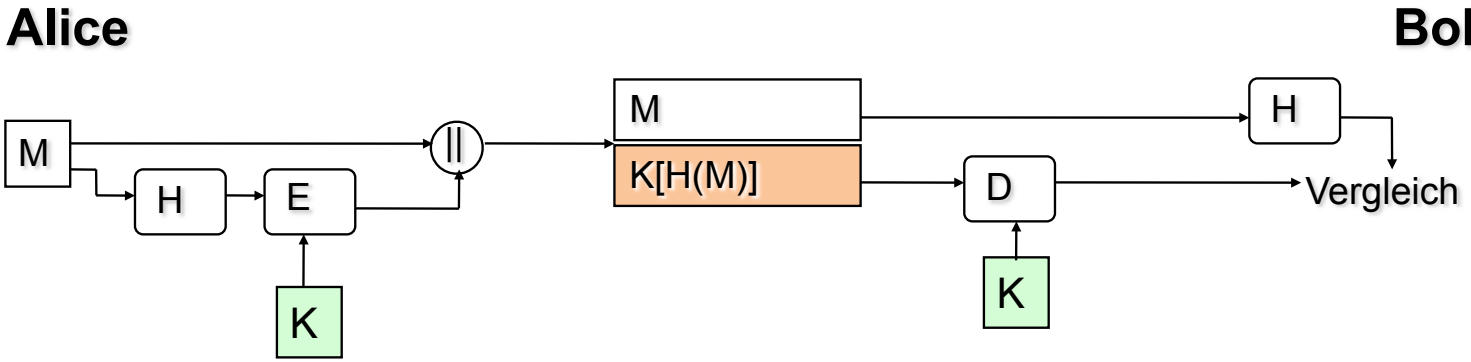


- Zusätzlich Vertraulichkeit durch Verschlüsselung
- Alice kann Sendung leugnen
- Alice/Bob können Zugang / Empfang nicht beweisen

Verwendung von Hash-Fkt. zur Authentisierung

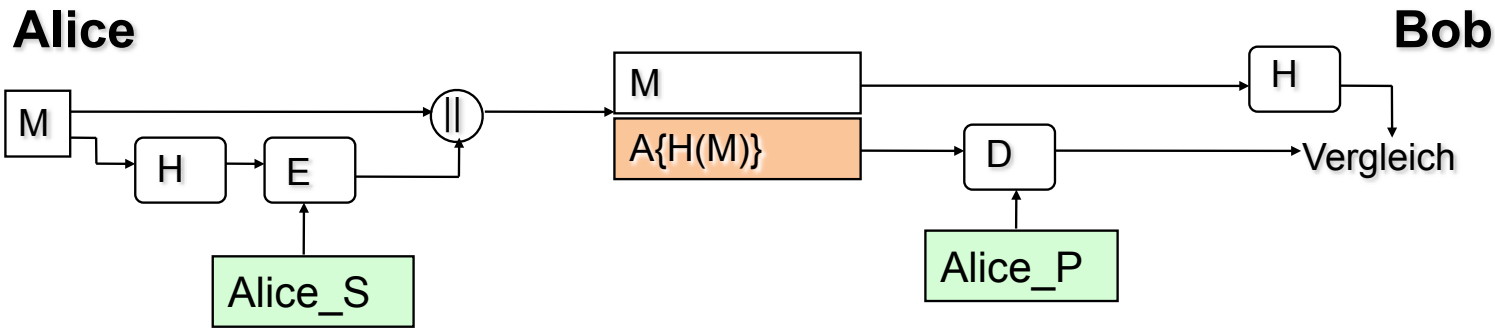


- Authentisierung des Datenursprungs (durch Schlüssel K)
- Vertraulichkeit
- Integrität



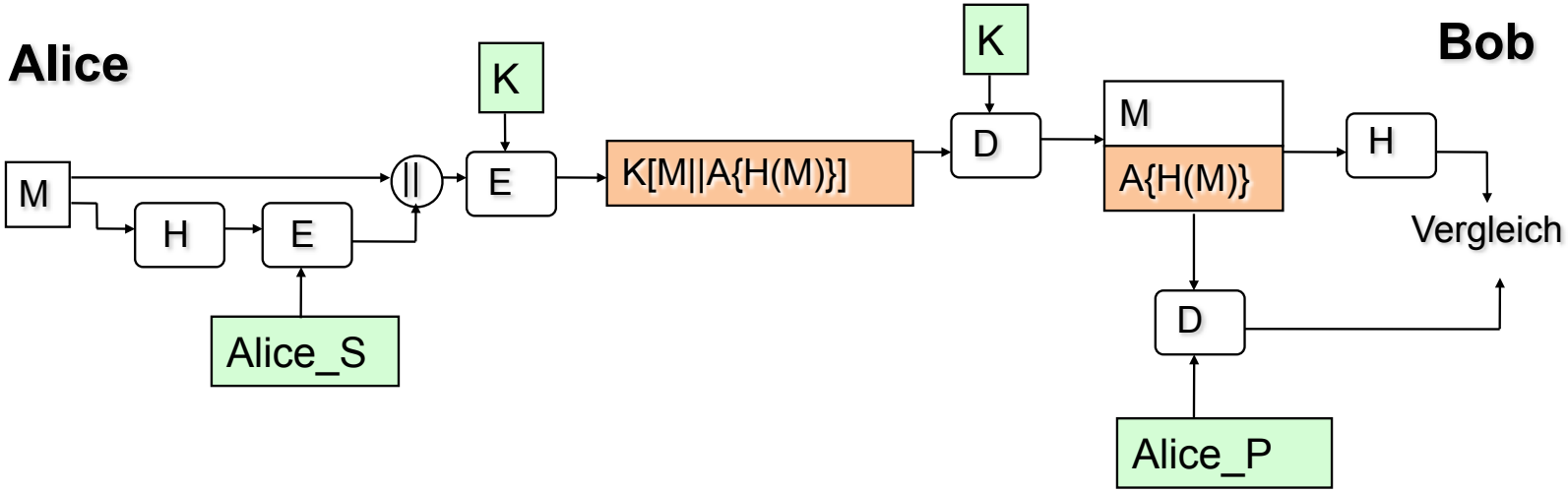
- Authentisierung und Integrität, keine Vertraulichkeit

Verwendung von Hash-Fkt. zur Authentisierung



- Authentisierung des Datenursprungs durch digitale Signatur
  - Alice signiert Hash
- (Daten-) Integrität (durch Hash)
- Keine Vertraulichkeit, jeder kann M lesen
- Alice kann Zugang nicht beweisen

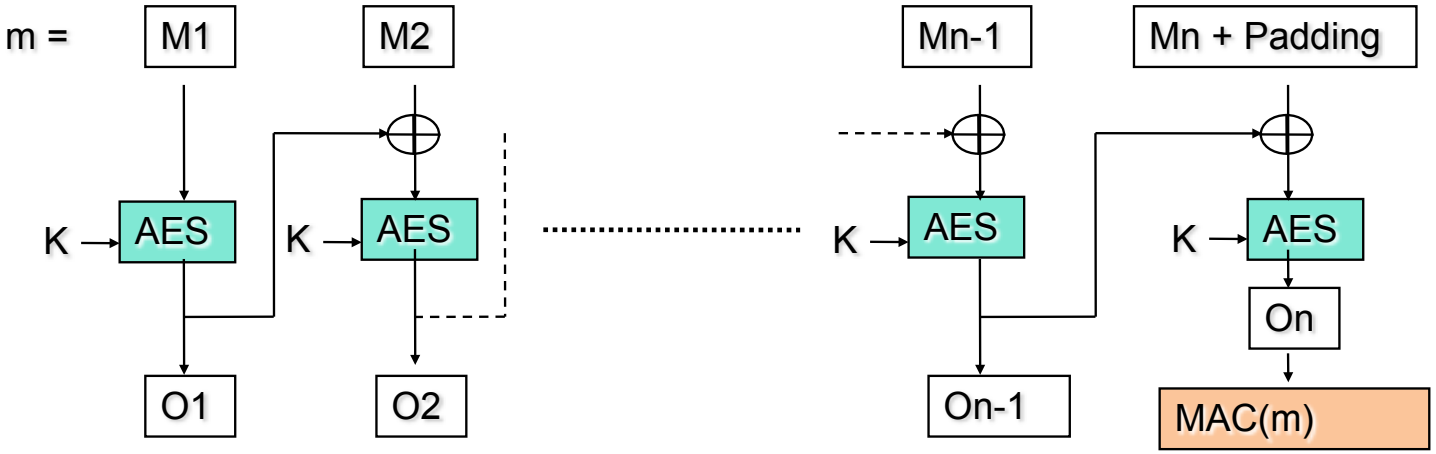
Verwendung von Hash-Fkt. zur Authentisierung



- Zusätzlich Vertraulichkeit durch (symmetrische) Verschlüsselung
- Am häufigsten verwendetes Verfahren
- Alice kann Zugang nicht beweisen

Authentisierung: MAC

- Message Authentication Code (MAC) für Nachricht M
- Idee: Kryptographische Checksumme wird mit Algorithmus A berechnet, A benötigt einen Schlüssel K
- MAC = A(K,M)
- Authentisierung über Schlüssel K (kennen nur Alice und Bob)
- Beispiel?



- AES im CBC Mode

Sicherheit von MACs

- Wie kann der MAC angegriffen werden?
- Brute Force:
  - MAC ist n Bits lang, Schlüssel K ist k Bits lang mit k > n
  - Angreifer kennt Klartext m und MAC(m,K)
  - Für alle Ki berechnet der Angreifer: MAC(m,Ki) == MAC(m,K)?
  - D.h. der Angreifer muss 2^k MACs erzeugen
  - Es existieren aber nur 2^n verschiedene MACs ($2^n < 2^k$)
  - D.h. mehrere Ki generieren den passenden MAC ($2^{(k-n)}$ Schlüssel)
  - Angreifer muss den Angriff iterieren:
    1. Runde liefert für 2^k Schlüssel ca. $2^{(k-n)}$ Treffer
    2. Runde liefert für $2^{(k-n)}$ Schlüssel $2^{(k-2n)}$ Treffer
    3. Runde liefert .... $2^{(k-3n)}$ Treffer
  - Falls k < n, liefert die erste Runde bereits den korrekten Schlüssel

length extension attack

- Möglich wenn Hash-Funktion mit Merkle-Damgard-Konstruktion verwendet wird (z.B. MD5, SHA, SHA-1)
- MAC(k,m), z.B. SHA-1(k||m)
- Dienst liefert für m MAC als Ausweis für Dienstnutzung

- Angreifer kennt Blocklänge und Länge der Nachricht

- Angreifer kann Nachricht verlängern ohne k zu kennen
- SHA-1(k||mm') liefert „gültigen" Hash auch ohne Kenntnis von k
- Beispiel m=Überweise-100-€
- Beispiel m'=\x00\x00\x00&Überweise-10000000000000-$

Hashed MAC (HMAC)

- Gesucht: MAC, der nicht symm. Verschlüsselung, sondern kryptographische Hash-Funktion zur Kompression verwendet
  - Hashes wie SHA-3 sind deutlich schneller als z.B. DES
- Problem: Hash-Funktionen verwenden keinen Schlüssel
- Lösung HMAC
  - Beliebige Hash-Funktion H verwendbar, die auf (Input) Blöcken arbeitet
  - Sei b die Blocklänge (meist 512 Bits)
  - Beliebige Schlüssel K mit Länge |K| = b verwendbar
  - Falls |K| < b:
    - Auffüllen mit Null-Bytes bis |K+| = b; d.h. K+ = K||0....0
  - Falls |K| > b:
    - K = H(K)
  - Schlüssel wird mit konstanten Input- (ipad) bzw. Output-Pattern (opad) XOR verknüpft:
    - ipad = 0x36 (b mal wiederholt), opad = 0x5c (b mal wiederholt)

HMAC Algorithmus

$$HMAC(m) = H\Big[(K^+ \oplus opad)||H[(K^+ \oplus ipad)||m]\Big]$$

1. $K^+ :=$ Schlüssel K auf Länge von b Bits gebracht
2. b Bits langer Block $S_i := K^+ \text{ XOR } ipad$
3. Nachricht m mit dem Block konkatenieren
4. Hash-Wert von $S_i||m$ berechnen
5. b-Bit-Block $S_o := K^+ \text{ XOR } opad$
6. mit dem Ergebnis von 4. konkatenieren
7. Hash-Wert über das Ergebnis von 6. berechnen

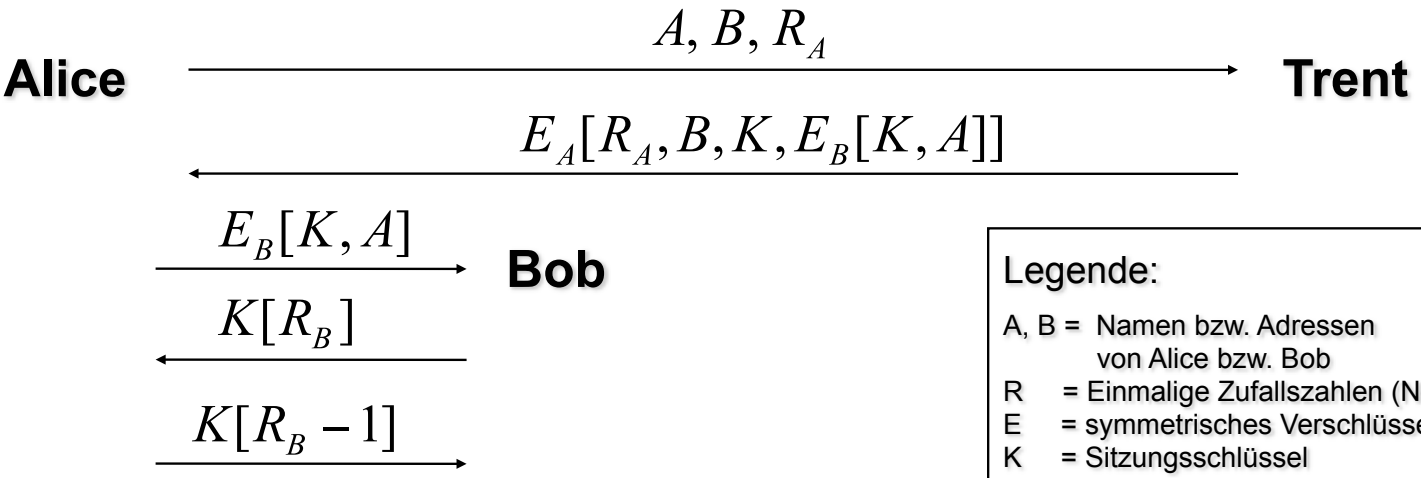
- Es muss verhindert werden, dass ein Angreifer eigenen Text an die Nachricht m anhängt und einfach den (zweiten, inneren) Hashwert weiterrechnet.
- Die äußere Hashfunktion sichert also nicht den ursprünglichen Nachrichteninhalt, sondern „das Ende“ der Nachricht.

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - ❑ Mandatory Access Control (MAC)
    - ❑ DAC
5. Identifizierung

Needham-Schröder

- Entwickelt von Roger Needham u. Michael Schroeder (1979)
- Verwendet vertrauenswürdigen Dritten Trent neben Alice und Bob (Trusted Third Party, TTP)
- Optimiert zur Verhinderung von Replay-Angriffen
- Verwendet symmetrische Verschlüsselung
- Trent teilt mit jedem Kommunikationspartner eigenen Schlüssel

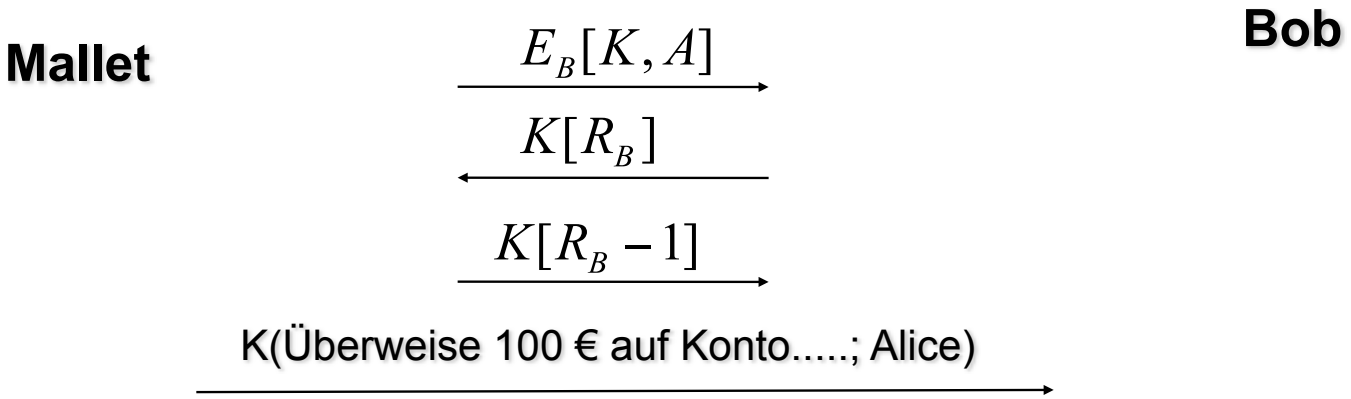


Legende:

A, B = Namen bzw. Adressen von Alice bzw. Bob
R = Einmalige Zufallszahlen (Nonces)
E = symmetrisches Verschlüsselungsverf.
K = Sitzungsschlüssel

Needham-Schröder-Protokollschwäche

- Problem: Alte Sitzungsschlüssel K bleiben gültig
- Falls Mallet an alten Schlüssel gelangen und die 1. Nachricht von Alice an Bob wiedereinspielen konnte, wird Maskerade möglich
- Mallet braucht keine geheimen Schlüssel von Trent (K_A,T, K_B,T)



- Lösungsidee:
  - Sequenznummer oder Timestamps einführen
  - Gültigkeitsdauer von Sitzungsschlüsseln festlegen

Kerberos

- Trusted Third Party Authentisierungsprotokoll
- Entwickelt für TCP/IP Netze
  - Im Rahmen des MIT Athena Projektes (X-Windows)
  - 1988 Version 4; 1993 Version 5
- Client (Person oder Software) kann sich über ein Netz bei Server(n) authentisieren
- Kerberos-Server kennt Schlüssel aller Clients
- Basiert auf symmetrischer Verschlüsselung
- Abgeleitet vom Needham-Schröder-Protokoll
- Hierarchie von Authentisierungsservern möglich; jeder Server verwaltet einen bestimmten Bereich (sog. Realm)
- Über Kooperationsmechanismen der Kerberos-Server kann Single-Sign-On realisiert werden

Authentisierungsdaten

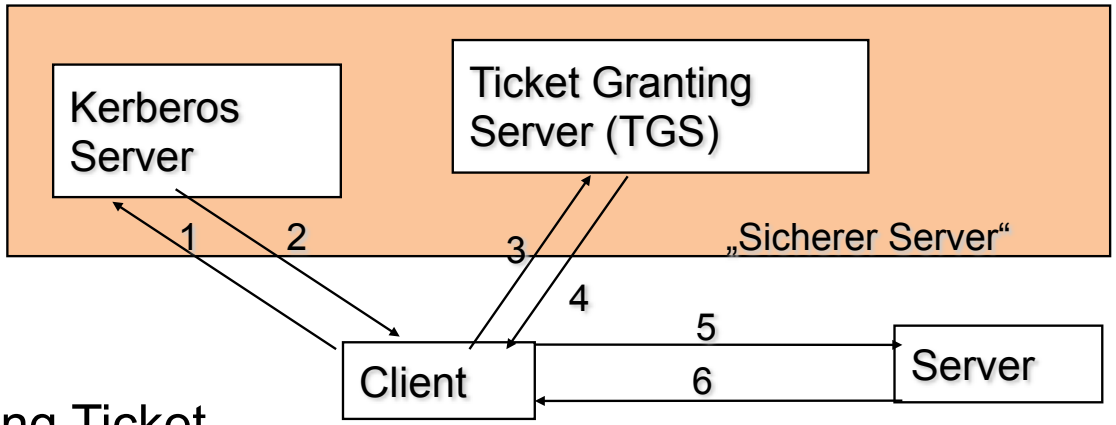
- Authentisierung basiert auf gemeinsamem (Sitzungs-)Schlüssel
- Kerberos arbeitet mit Credentials; unterschieden werden
  1. Ticket
  2. Authenticator
- Ticket
  - als „Ausweis“ für die Dienstnutzung; nur für einen Server gültig
  - wird vom Ticket Granting Server erstellt
  - keine Zugriffskontrolle über Ticket (nicht mit Capability verwechseln!)

$$T_{c,s} = s, c, addr, timestamp, lifetime, K_{c,s}$$

- **Authenticator**
  - „Ausweis“ zur Authentisierung; damit Server ein Ticket verifizieren kann
  - vom Client selbst erzeugt
  - Wird zusammen mit dem Ticket verschickt

$$A_{c,s} = c, addr, timestamp$$

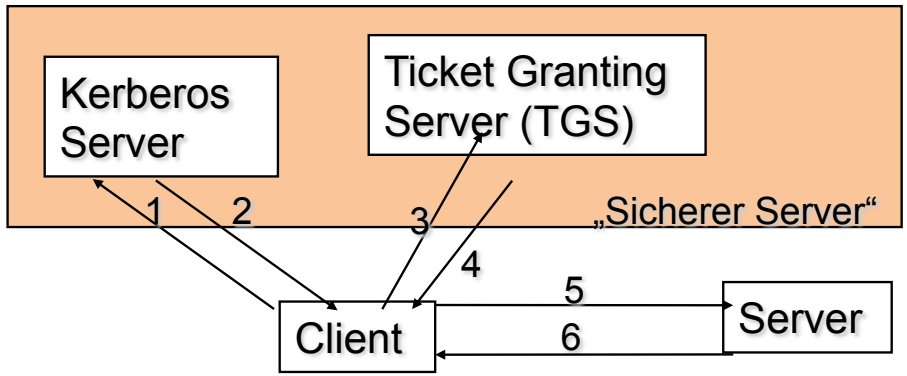
Kerberos Modell



1. Request für Ticket Granting Ticket
2. Ticket Granting Ticket
3. Request für Server Ticket
4. Server Ticket
5. Request für Service
6. Authentisierung des Servers (Optional)

- Im folgenden Kerberos V5 vereinfacht, d.h. ohne Realms und Optionenlisten; exaktes Protokoll [RFC 1510, Stal98, RFC 4120]

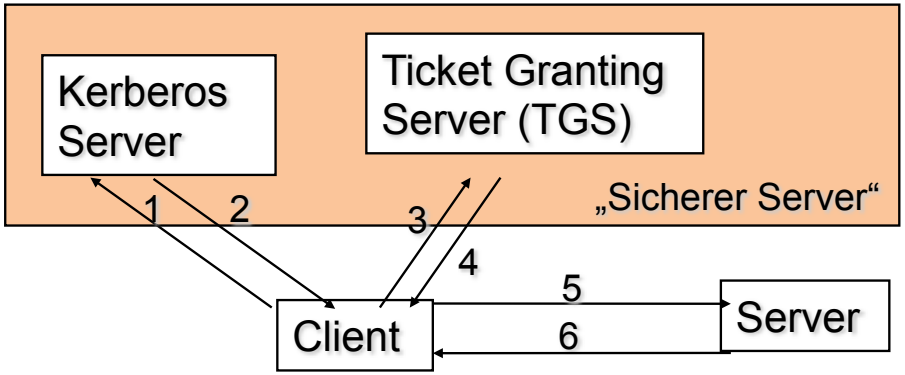
Kerberos: Initiales Ticket (ein Mal pro Sitzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

1. Request für Ticket Granting Ticket:
   c, tgs (Kerberos überprüft, ob Client in Datenbank)
2. Ticket Granting Ticket:
   $K_c[K_{c,tgs}], K_{tgs}[T_{c,tgs}]$ mit $T_{c,tgs} = tgs, c, a, t, v, K_{c,tgs}$

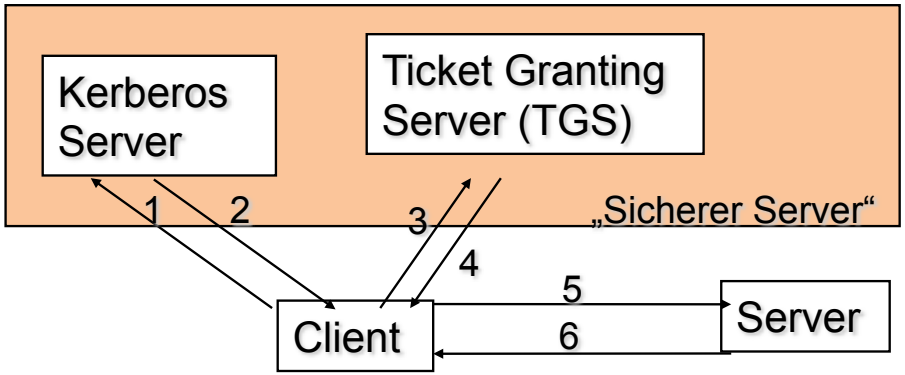
Kerberos: Request für Server Ticket



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

3. Request für Server Ticket:
   $s, K_{c,tgs}[A_{c,tgs}], K_{tgs}[T_{c,tgs}]$ mit $A_{c,tgs} = c,a,t$ $T_{c,tgs} = tgs,c,a,t,v,K_{c,tgs}$
4. Server Ticket:
   $K_{c,tgs}[K_{c,s}], K_s[T_{c,s}]$ mit $T_{c,s} = s,c,a,t,v,K_{c,s}$

Kerberos: Request für Service (pro Service-Nutzung)



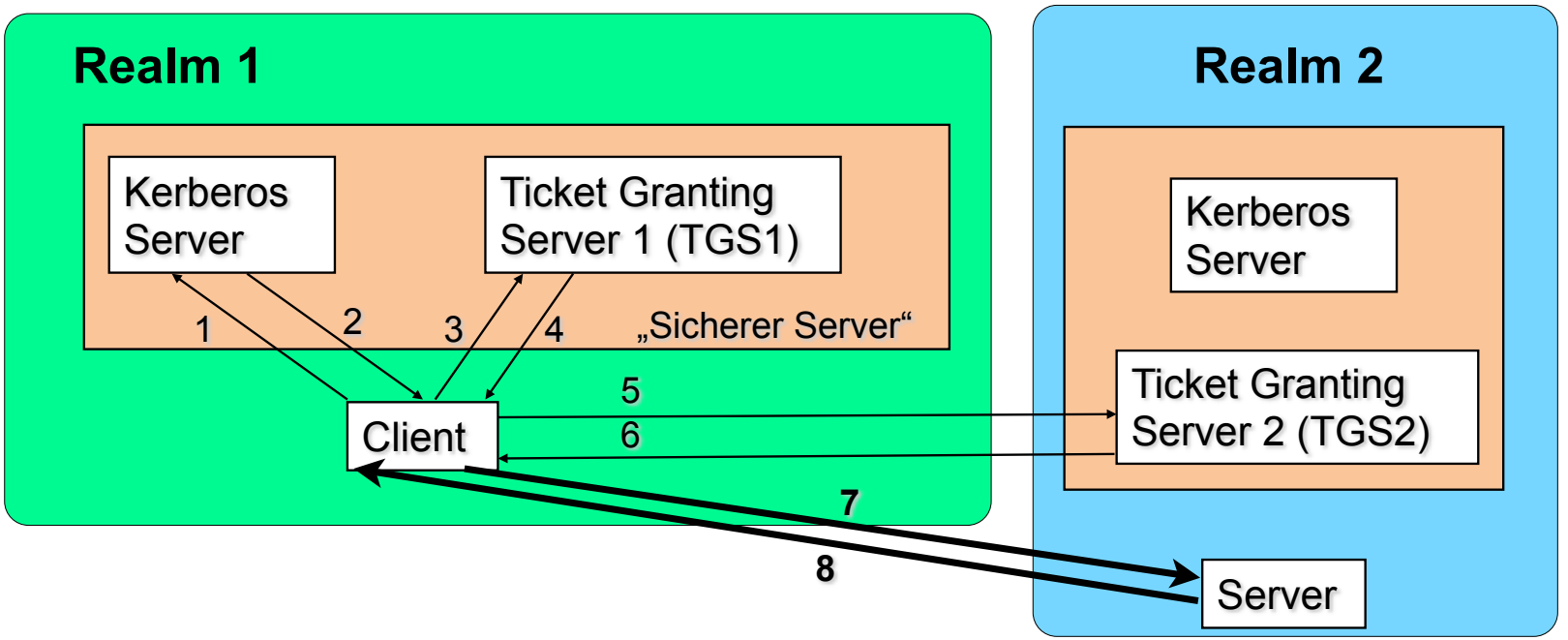
c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

5. Request für Service:
   $K_{c,s}[A_{c,s}], K_s[T_{c,s}]$ mit $A_{c,s} = c, a, t, key, seqNo$ $T_{c,s} = s, c, a, t, v, K_{c,s}$
6. Server Authentication:
   $K_{c,s}[t, key, seqNo]$

Multi-Domain-Kerberos

- Kerberos-Server immer für eine Domäne (Realm) zuständig
- Domänenübergreifendes Kerberos wird benötigt
  (z.B. Kooperation von zwei unabhängigen Unternehmen)
- Idee:

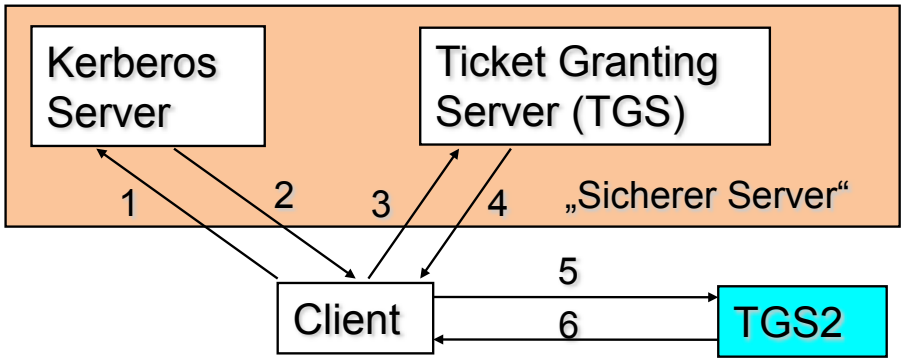
TGS der fremden Realm wird „normaler“ Server



Multi-Domain Kerberos

- Domänenübergreifende Authentisierung
- Erfordert Schlüsselaustausch zwischen TGS1 und TGS2:
  KTGS1,TGS2
- Vertrauen (Trust) erforderlich:
  - Besuchende Domäne muss Authenticator und TGS der Heimat-Domäne vertrauen
  - Beide Domänen müssen sich auf „sichere“ Implementierung verlassen
- Skalierungsproblem:
  n Realms erfordern n * (n-1) / 2 Schlüssel, d.h. $O(n^2)$

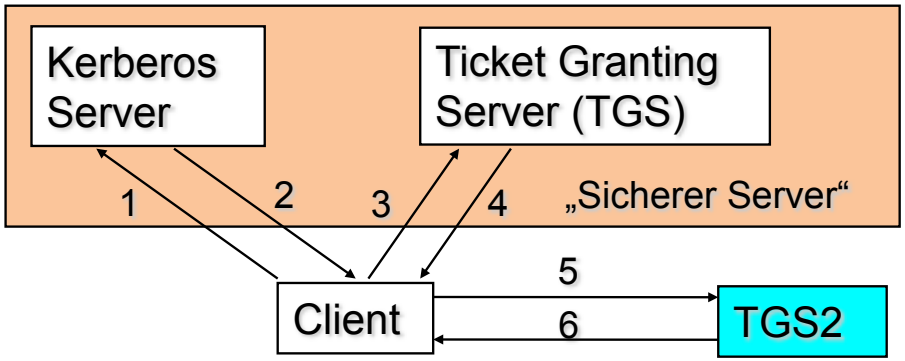
Multi-Domain Kerberos: Erweiterungen



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

3. Request für Server Ticket für fremden TGS (TGS2):
   tgs2, $K_{c,tgs1}[A_{c,tgs1}]$, $K_{tgs1}[T_{c,tgs1}]$
   mit $A_{c,tgs1}$=c,a,t ; $T_{c,tgs1}$=tgs1,c,a,t,v,$K_{c,tgs1}$
4. Server Ticket:
   $K_{c,tgs1}[\mathbf{K_{c,tgs2}}]$, $K_{tgs2}[T_{c,tgs2}]$ mit $T_{c,tgs2}$ = tgs2,c,a,t,v,$K_{c,tgs2}$

Kerberos: Request for Service (pro Service-Nutzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Gültigkeitsdauer
t	=	Zeitstempel
K_x	=	Schlüssel von x
$K_{x,y}$	=	Sitzungsschlüssel von x u. y
$T_{x,y}$	=	Ticket für x um y zu nutzen
$A_{x,y}$	=	Authenticator von x für y

5. Request for Server Ticket beim TG2:
   s, $K_{c,tgs2}[A_{c,tgs2}]$, $K_{tgs2}[T_{c,tgs2}]$
   mit $A_{c,tgs2}$ = c,a,t $T_{c,tgs2}$ = tgs2,c,a,t,v,$K_{c,tgs2}$
6. Server Ticket:
   $K_{c,tgs2}[K_{c,s}]$, $K_s[T_{c,s}]$
7. Weiterer Ablauf wie bei single Domain Kerberos

Bewertung

- Sichere netzweite Authentisierung auf Ebene der Dienste

- Authentisierung basiert auf IP-Adresse
  - IP-Spoofing u.U. möglich
  - Challenge Response Protokoll zur Verhinderung nur optional
- Sicherheit hängt von der Stärke der Passworte ab (aus dem Passwort wird der Kerberos-Schlüssel abgeleitet)
- Lose gekoppelte globale Zeit erforderlich (Synchronisation)
- Kerberos-Server und TGS müssen (auch physisch) besonders gut gesichert werden und sind potenziell „Single Point of Failure“
- Verlässt sich auf „vertrauenswürdige“ Software (Problem der Trojanisierung, vgl. CA-2002-29)
- Administrationsschnittstelle und API nicht standardisiert

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
    1. Peer Entity / Benutzer
        - Passwort, Einmalpasswort, Biometrie
    2. Datenursprung
        - Verschlüsselung
        - Message Authentication Code (MAC) und Hashed MAC (HMAC)
    3. Authentisierungsprotokolle
        - Needham-Schröder
        - Kerberos
4. Autorisierung und Zugriffskontrolle
    - ❑ Mandatory Access Control (MAC)
    - ❑ DAC
5. Identifizierung

Autorisierung und Zugriffskontrolle

- Autorisierung: Vergabe / Spezifikation von Berechtigungen
- Zugriffskontrolle: Durchsetzung dieser Berechtigungen
- Häufig werden Autorisierung und Zugriffskontrolle zusammengefasst

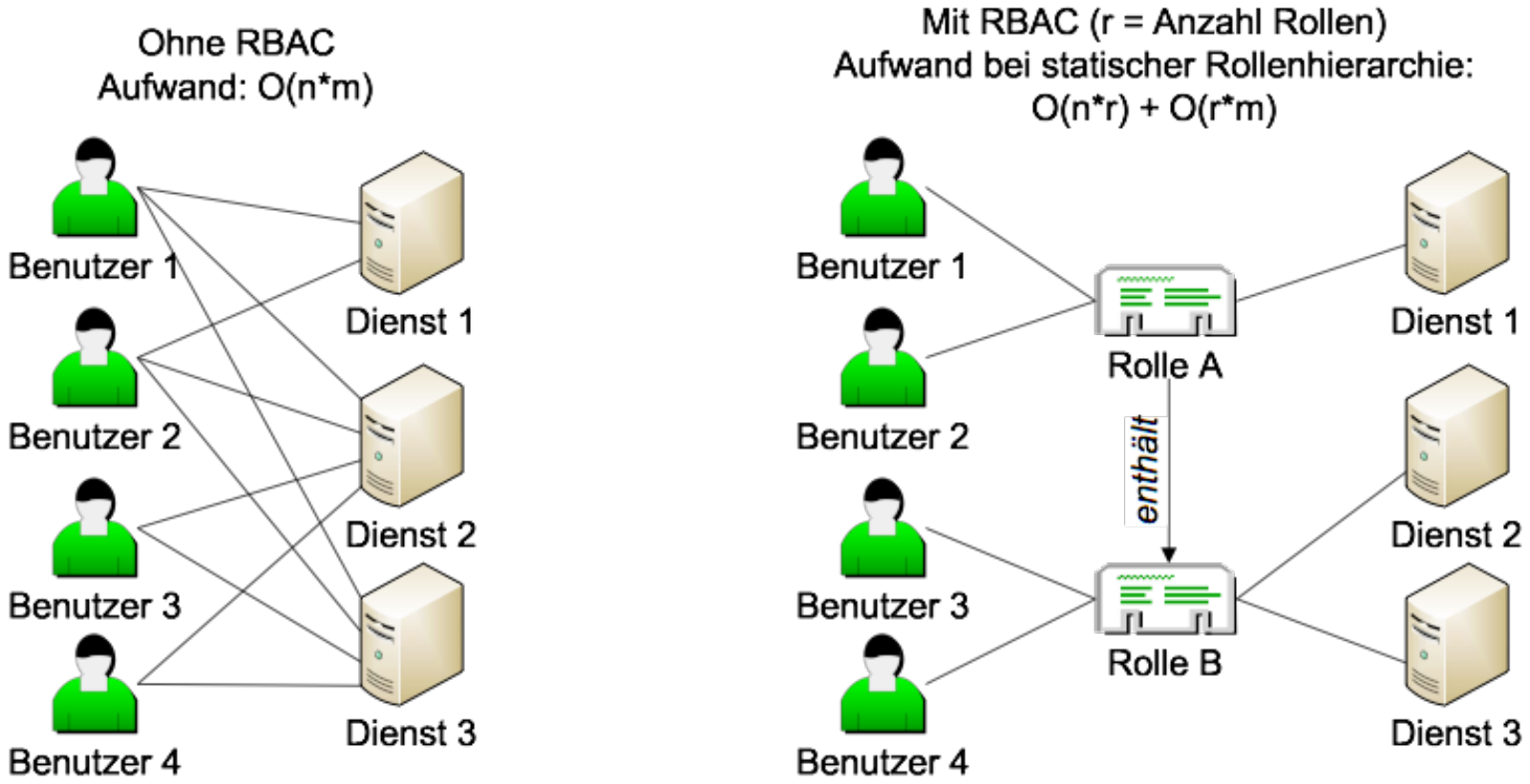
- Handelnde werden als Subjekt bezeichnet
- Berechtigungen werden an Subjekte erteilt
- Berechtigungen gelten für Objekte
- Objekte sind die schützenswerten Einheiten im System

Klassifikation

- DAC (Discretionary Access Control)
  - Basieren auf dem Eigentümerprinzip
  - Eigentümer spezifiziert Berechtigungen an seinen Objekten
  - Zugriffsrechte auf Basis der Objekte vergeben
- MAC (Mandatory Access Control)
  - Regelbasierte Festlegung der Rechte
  - Systemglobal
  - Z.B. Bell-LaPadula; Regeln werden über Sicherheitsklassen (unklassifiziert, vertraulich, geheim, streng geheim) spezifiziert
- RBAC (Role-based Access Control)
  - Trennung von Subjekt und Aufgabe
  - Berechtigungen werden nicht mehr an Subjekt, sondern an bestimmte Aufgabe geknüpft
  - Subjekte erhalten Berechtigung über Rollenmitgliedschaft(en)

RBAC: Rollenhierarchie und Aufwand

Kontinuierlich zu pflegende Berechtigungszuordnungen bei n Benutzern und m Diensten:



Zugriffsmatrix

- Schutzzustand eines Systems zum Zeitpunkt t wird durch Matrix M(t) modelliert:
  - M(t) = S(t) x O(t); es gilt M(t): S(t) x O(t) $\longrightarrow$ 2^R
  - R ist die Menge der Zugriffsrechte
  - Subjekte S bilden die Zeilen der Matrix
  - Objekte O bilden die Spalten
  - Ein Eintrag M(t,s,o) = $\{r_1,r_2,...,r_n\}$ beschreibt die Menge der Rechte des Subjekts s zum Zeitpunkt t am Objekt o

	Datei1	Datei2	Prozess 1
Prozess 1	*read*	*read*	
Prozess 2		*read, write*	*signal*
Prozess 3	*read, write, owner*		*kill*

  - Implementierung „spaltenweise“: Zugriffskontrolllisten (z.B. UNIX)
  - Implementierung „zeilenweise“: Capabilities

Referenzmonitor

- Zur Realisierung der Zugriffskontrolle ist eine sichere, „vertrauenswürdige“ Systemkomponente erforderlich

- Häufig als Referenzmonitor oder Access Control Monitor bezeichnet

- Erfüllt folgende Anforderungen:
  - Zugriff auf Objekte nur über den Monitor möglich
  - Monitor kann Aufrufenden (Subjekt) zweifelsfrei identifizieren (Authentisierung)
  - Monitor kann Objektzugriff unterbrechen bzw. verhindern

Inhalt

1. Vertraulichkeit
2. Integritätssicherung
3. Authentisierung
   1. Peer Entity / Benutzer
      - Passwort, Einmalpasswort, Biometrie
   2. Datenursprung
      - Verschlüsselung
      - Message Authentication Code (MAC) und Hashed MAC (HMAC)
   3. Authentisierungsprotokolle
      - Needham-Schröder
      - Kerberos
4. Autorisierung und Zugriffskontrolle
   - ❑ Mandatory Access Control (MAC)
   - ❑ DAC
5. Identifizierung

Identifikation (Identification)

- Zweifelsfreie Verbindung (Verknüpfung) von digitaler ID und Real-World Entity (Person, System, Prozess,....)
- Ohne sichere Identifikation kann es keine zuverlässige Authentisierung geben
- Mindestens zweistufiger Prozess:
  - Personalisierung:
    Zweifelsfreie Ermittlung der Real-World Identität (bei Personen z.B. durch Personalausweis) und Vergabe einer digitalen ID (z.B. Benutzername)
  - Identifikation:
    Verbindung von digitaler ID mit Informationen, die nur die Entität nutzen / kennen kann (z.B. Passwort, Schlüsselpaar, bzw. öffentlicher Schlüssel)

- Problem: Falls der Angreifer in der Lage ist, seine Informationen mit fremder ID zu verbinden, kann er Maskerade-Angriffe durchführen

Identifikation durch digitale Signatur / Zertifikat

- Grundidee: Trusted Third Party (TTP) bürgt durch Unterschrift (digitale Signatur) für die Identität einer Entität (vergleichbar mit einem Notar)

- Begriffe:
  - Zertifikat: Datenstruktur zur Verbindung von Identitätsinformation und öffentlichem Schlüssel der Entität; digital signiert von einer
  - Certification Authority (CA) / Trust Center: Trusted Third Party
  - Realm: Benutzerkreis der CA
    - Alle Benutzer in einer Realm „vertrauen“ der CA, d.h.
    - „Aussagen“ der CA werden von allen Benutzern als gültig, richtig und wahr angenommen
  - (Local) Registration Authority (LRA): Nimmt Anträge auf ein Zertifikat (Certification Request) entgegen; führt Personalisierung durch

Identifikation: Aufgabenspektrum einer CA

- **Generierung von Zertifikaten (Certificate Issuance):**
  Erzeugung der Datenstrukturen und Signatur
- **Speicherung (Certification Repository):**
  Allgemein zugängliches Repository für Zertifikate
- **Widerruf und Sperrung (Certificate Revocation):**
  Z.B. falls geheimer Schlüssel des Zertifizierten kompromittiert wurde
- **Aktualisierung (Certifcation Update):**
  Erneuerung des Zertifikates nach Ablauf der Gültigkeit
- **Schlüsselerzeugung (Key Generation)**
- **Historienverwaltung (Certification History):**
  Speicherung nicht mehr gültiger Zertifikate (zur Beweissicherung)
- **Beglaubigung (Notarization):**
  CA signiert Vorgänge zwischen Benutzern (z.B. Verträge)
- **Zeitstempeldienst (Time Stamping):** CA bindet Info an Zeit
- **Realm-übergreifende Zertifi-zierung (Cross-Certification):**
  Eigene CA zertifiziert fremde CAs
- **Attribut-Zertifikate (Attribute Certificate):**
  Binden von Attributen an eine Identität (z.B. Berechtigungen, Vollmachten, ...)

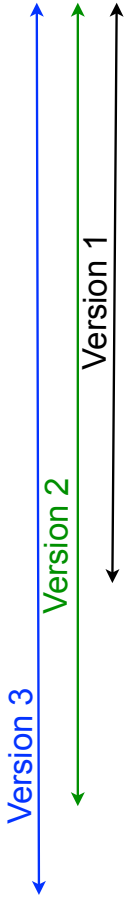
Ablauf der Benutzerzertifizierung

1. Schlüsselgenerierung:
   - Zentral durch CA oder dezentral durch Benutzer
   - „Ausreichend sichere“ Schlüssel müssen erzeugt werden
   - Nur der Zertifizierte darf geheimen Schlüssel kennen
2. Personalisierung, Certification Request:
   - Benutzer beantragt ein Zertifikat (Certification Request)
   - Feststellung der Identität des Benutzers (z.B. durch pers. Erscheinen)
   - Benutzer muss belegen, dass er im Besitz des passenden privaten Schlüssels ist (z.B. durch Challenge-Response-Protokoll)
3. Generierung der Datenstruktur für das Zertifikat:
   - Entsprechende Attribute werden aus dem Certification Request des Benutzers entnommen
   - Im Folgenden X.509v3-Zertifikate als Beispiel
4. Digitale Signatur durch die CA

X.509v3 Zertifikat: Attribute

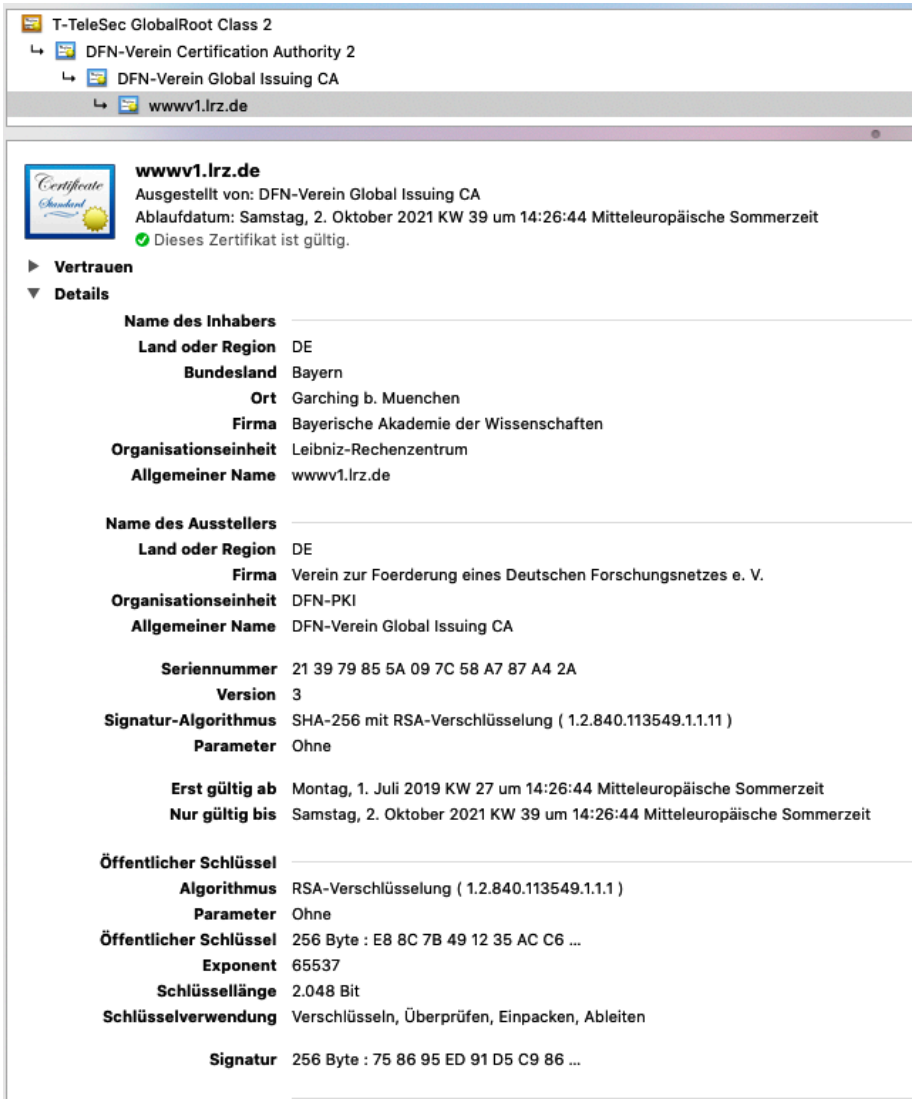
- X.509 internationaler ITU-T Standard als Teil der X.500 Serie:
  - Verzeichnisdienst
  - X.500 - X.530 wurde nie vollständig implementiert
- X.509 hat sich auf breiter Basis durchgesetzt
- Drei Versionen:
  - V1: 1988
  - V2: 1993
  - V3: 1995
- Definiert:
  - Datenformat für Zertifikat
  - Zertifikatshierarchie
  - Widerrufslisten (Certificate Revocation Lists, CRL)

X.509v3 Zertifikat: Attribute



Version	Versionsnummer (1,2,3); Default 1
SerialNumber	Pro CA eindeutige Nummer des Zertifikates
SignatureAlgorithm	Verw. Algorithmus für die digitale Signatur
Issuer	Distinguished Name (DN, vgl. X.500) der CA
Validity	Gültigkeitsdauer; Angegeben in notBefore und notAfter
Subject	„Gegenstand“ des Zert.; z.B. DN des Zertifizierten
SubjectPublicKey-Info	Öffentlicher Schlüssel, des Zertifizierten; Algorithmus für den Schlüssel; ggf. weitere Parameter
IssuerUnique-Identifier	Eindeutiger Bezeichner der CA (ab Version 2 optional); vgl. auch Issuer Feld
SubjectUnique-Identifier	Zusätzliche Info über Subject des Zertifikates (ab Version 2 optional)
Extensions	Ab v3: Einschränkungen, Bedingungen, Erweiterungen
Signature	digitale Signatur der gesamten Datenstruktur

DFN-PKI Zertifikat: Beispiel Web-Server



T-TeleSec GlobalRoot Class 2
DFN-Verein Certification Authority 2
DFN-Verein Global Issuing CA
wwwv1.lrz.de

wwwv1.lrz.de
Ausgestellt von: DFN-Verein Global Issuing CA
Ablaufdatum: Samstag, 2. Oktober 2021 KW 39 um 14:26:44 Mitteleuropäische Sommerzeit
Dieses Zertifikat ist gültig.

Vertrauen

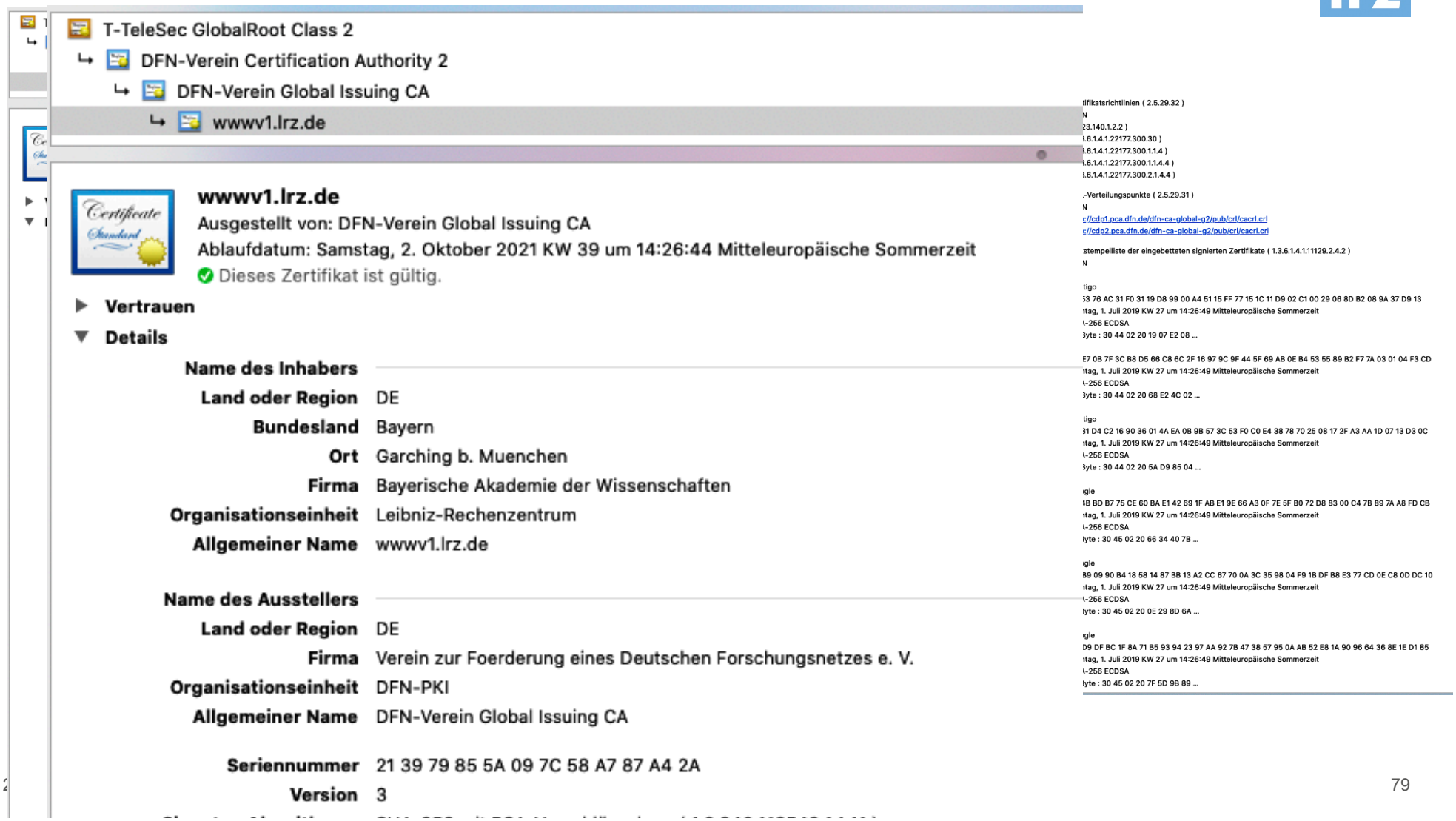
Details

Name des Inhabers	
Land oder Region	DE
Bundesland	Bayern
Ort	Garching b. Muenchen
Firma	Bayerische Akademie der Wissenschaften
Organisationseinheit	Leibniz-Rechenzentrum
Allgemeiner Name	wwwv1.lrz.de
Name des Ausstellers	
Land oder Region	DE
Firma	Verein zur Foerderung eines Deutschen Forschungsnetzes e. V.
Organisationseinheit	DFN-PKI
Allgemeiner Name	DFN-Verein Global Issuing CA
Seriennummer	21 39 79 85 5A 09 7C 58 A7 87 A4 2A
Version	3
Signatur-Algorithmus	SHA-256 mit RSA-Verschlüsselung (1.2.840.113549.1.1.11)
Parameter	Ohne
Erst gültig ab	Montag, 1. Juli 2019 KW 27 um 14:26:44 Mitteleuropäische Sommerzeit
Nur gültig bis	Samstag, 2. Oktober 2021 KW 39 um 14:26:44 Mitteleuropäische Sommerzeit
Öffentlicher Schlüssel	
Algorithmus	RSA-Verschlüsselung (1.2.840.113549.1.1.1)
Parameter	Ohne
Öffentlicher Schlüssel	256 Byte : E8 8C 7B 49 12 35 AC C6 …
Exponent	65537
Schlüssellänge	2.048 Bit
Schlüsselverwendung	Verschlüsseln, Überprüfen, Einpacken, Ableiten
Signatur	256 Byte : 75 86 95 ED 91 D5 C9 86 …

Schlüsselverwendung (2.5.29.15)	
JA	
Digitale Signatur, Verschlüsseln von Schlüsseln	
Basiseinschränkungen (2.5.29.19)	
NEIN	
NEIN	
Erweiterte Schlüsselverwendung (2.5.29.37)	
NEIN	
Serverauthentifizierung (1.3.6.1.5.5.7.3.1)	
Schlüsselkennung des Antragstellers (2.5.29.14)	
NEIN	
EE 7E 6D ED 2B 6C E5 0C DE AA D4 86 D0 B5 E8 9B 7	
Schlüsselkennung (2.5.29.35)	
NEIN	
6B 3A 98 8B F9 F2 53 89 DA E0 AD B2 32 1E 09 1F E8	
Alternativer Name des Inhabers (2.5.29.17)	
NEIN	
girlsday.lrz.de	
intern.lrz-muenchen.de	
intern.lrz.de	
intern.xn--lrz-mnchen-eeb.de	
lrz.de	
phpmyadmin.intern.lrz.de	
phpmyadmin.lrz-muenchen.de	
phpmyadmin.lrz.de	
phpmyadmin.xn--lrz-mnchen-eeb.de	
secincmgmt.sec.lrz.de	
serverbetrieb.lrz.de	
test.wwwv1.lrz.de.devweb.mwn.de	
v2c.lrz.de	
www.girlsday.lrz.de	
www.grid.lrz.de	
www.l-rz.de	
www.leibniz-supercomputing-centre.de	
www.leibniz-supercomputing-centre.eu	
www.lrz-muenchen.de	
www.lrz-munich.eu	
www.lrz.de	
www.lrz.eu	
www.v2c.lrz.de	
www.xn--lrz-mnchen-eeb.de	
wwwv1.lrz.de	

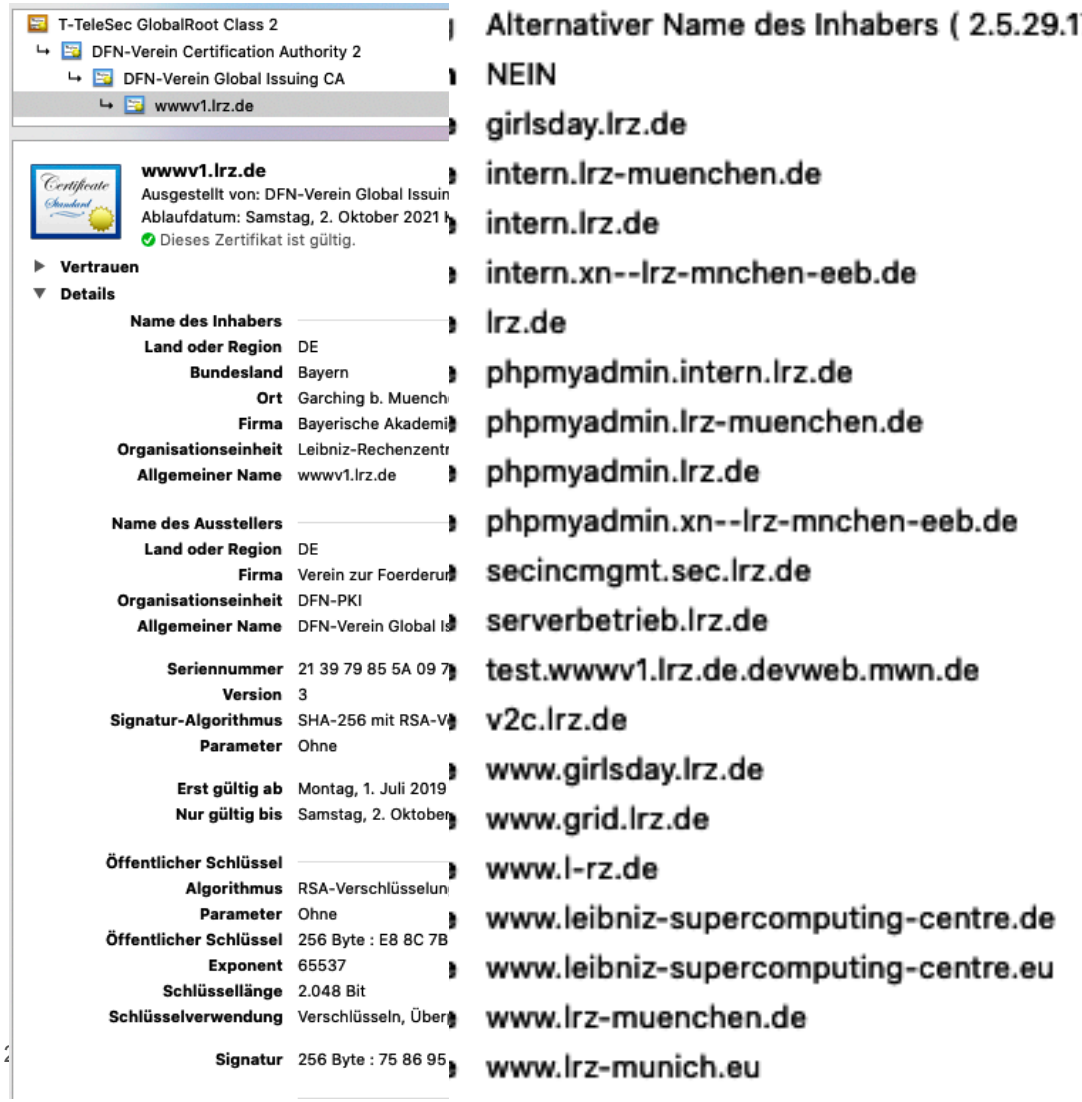
Erweiterung	Zertifikatsrichtlinien (2.5.29.32)
Kritisch	NEIN
Policy-ID #1	(2.23.140.1.2.2)
Policy-ID #2	(1.3.6.1.4.1.22177.300.30)
Policy-ID #3	(1.3.6.1.4.1.22177.300.1.1.4)
Policy-ID #4	(1.3.6.1.4.1.22177.300.1.1.4.4)
Policy-ID #5	(1.3.6.1.4.1.22177.300.2.1.4.4)
Erweiterung	CRL-Verteilungspunkte (2.5.29.31)
Kritisch	NEIN
URI	http://cdp1.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
URI	http://cdp2.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
Erweiterung	Zeitstempelliste der eingebetteten signierten Zertifikate (1.3.6.1.4.1.11129.2.4.2)
Kritisch	NEIN
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	6F 53 76 AC 31 F0 31 19 D8 99 00 A4 51 15 FF 77 15 1C 11 D9 02 C1 00 29 06 8D B2 08 9A 37 D9 13
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 19 07 E2 08 …
SCT-Version	1
Schlüssel-ID protokollieren	AA E7 0B 7F 3C B8 D5 66 C8 6C 2F 16 97 9C 9F 44 5F 69 AB 0E B4 53 55 89 B2 F7 7A 03 01 04 F3 CD
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 68 E2 4C 02 …
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	55 81 D4 C2 16 90 36 01 4A EA 0B 9B 57 3C 53 F0 C0 E4 38 78 70 25 08 17 2F A3 AA 1D 07 13 D3 0C
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 5A D9 85 04 …
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	EE 4B BD B7 75 CE 60 BA E1 42 69 1F AB E1 9E 66 A3 0F 7E 5F B0 72 D8 83 00 C4 7B 89 7A A8 FD CB
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 66 34 40 7B …
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	A4 B9 09 90 B4 18 58 14 87 BB 13 A2 CC 67 70 0A 3C 35 98 04 F9 1B DF B8 E3 77 CD 0E C8 0D DC 10
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 0E 29 8D 6A …
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	BB D9 DF BC 1F 8A 71 B5 93 94 23 97 AA 92 7B 47 38 57 95 0A AB 52 E8 1A 90 96 64 36 8E 1E D1 85
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 7F 5D 9B 89 …

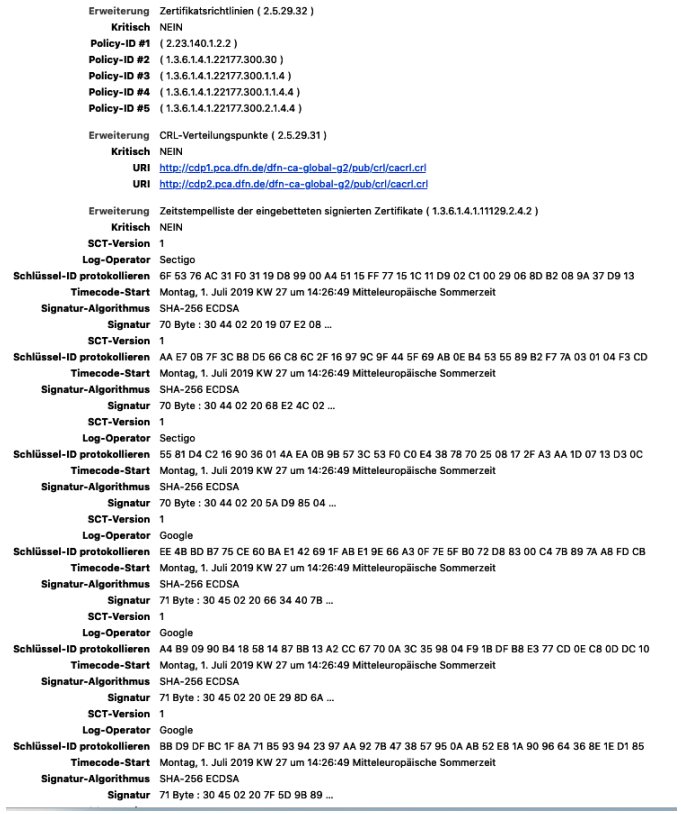
DFN-PKI Zertifikat: Beispiel Web-Server



DFN-PKI Zertifikat: Beispiel Web-Server

DFN-PKI Zertifikat: Beispiel Web-Server





Erweiterung	Zertifikatsrichtlinien (2.5.29.32)
Kritisch	NEIN
Policy-ID #1	(2.23.140.1.2.2)
Policy-ID #2	(1.3.6.1.4.1.22177.300.30)
Policy-ID #3	(1.3.6.1.4.1.22177.300.1.1.4)
Policy-ID #4	(1.3.6.1.4.1.22177.300.1.1.4.4)
Policy-ID #5	(1.3.6.1.4.1.22177.300.2.1.4.4)
Erweiterung	CRL-Verteilungspunkte (2.5.29.31)
Kritisch	NEIN
URI	http://cdp1.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
URI	http://cdp2.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
Erweiterung	Zeitstempelliste der eingebetteten signierten Zertifikate (1.3.6.1.4.1.11129.2.4.2)
Kritisch	NEIN
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	6F 53 76 AC 31 F0 31 19 D8 99 00 A4 51 15 FF 77 15 1C 11 D9 02 C1 00 29 06 8D B2 08 9A 37 D9 13
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 19 07 E2 08 ...
SCT-Version	1
Schlüssel-ID protokollieren	AA E7 08 7F 3C B8 D5 66 C8 6C 2F 16 97 9C 9F 44 5F 69 AB 0E B4 53 55 89 B2 F7 7A 03 01 04 F3 CD
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 68 E2 4C 02 ...
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	55 81 D4 C2 16 90 36 01 4A EA 0B 9B 57 3C 53 F0 C0 E4 38 78 70 25 08 17 2F A3 AA 1D 07 13 D3 0C
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 5A D9 85 04 ...
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	EE 4B BD B7 75 CE 60 BA E1 42 69 1F AB E1 9E 66 A3 0F 7E 5F B0 72 D8 83 00 C4 7B 89 7A A8 FD CB
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 66 34 40 7B ...
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	A4 B9 09 90 B4 18 58 14 87 BB 13 A2 CC 67 70 0A 3C 35 98 04 F9 1B DF B8 E3 77 CD 0E C8 0D DC 10
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 0E 29 8D 6A ...
SCT-Version	1
Log-Operator	Google
Schlüssel-ID protokollieren	BB D9 DF BC 1F 8A 71 B5 93 94 23 97 AA 92 7B 47 38 57 95 0A AB 52 E8 1A 90 96 64 36 8E 1E D1 85
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	71 Byte : 30 45 02 20 7F 5D 98 89 ...

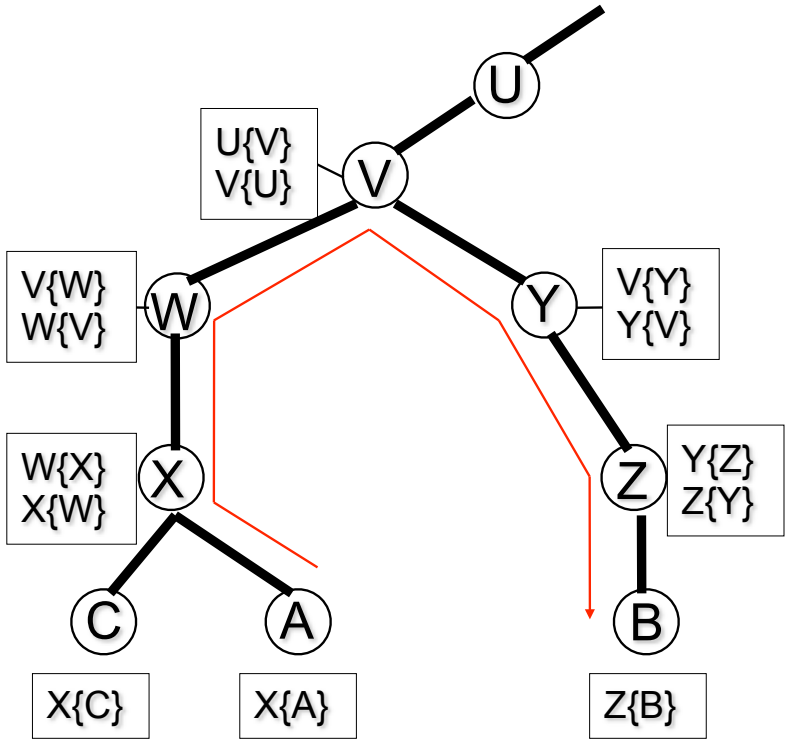
Alternativer Name des Inhabers (2.5.29.17)
NEIN
girlsday.lrz.de
intern.lrz-muenchen.de
intern.lrz.de
intern.xn--lrz-mnchen-eeb.de
lrz.de
phpmyadmin.intern.lrz.de
phpmyadmin.lrz-muenchen.de
phpmyadmin.lrz.de
phpmyadmin.xn--lrz-mnchen-eeb.de
secincmgmt.sec.lrz.de
serverbetrieb.lrz.de
test.wwwv1.lrz.de.devweb.mwn.de
v2c.lrz.de
www.girlsday.lrz.de
www.grid.lrz.de
www.l-rz.de
www.leibniz-supercomputing-centre.de
www.leibniz-supercomputing-centre.eu
www.lrz-muenchen.de
www.lrz-munich.eu

DFN-PKI Zertifikat: Beispiel Web-Server

Erweiterung	Zertifikatsrichtlinien (2.5.29.32)
Kritisch	NEIN
Policy-ID #1	(2.23.140.1.2.2)
Policy-ID #2	(1.3.6.1.4.1.22177.300.30)
Policy-ID #3	(1.3.6.1.4.1.22177.300.1.1.4)
Policy-ID #4	(1.3.6.1.4.1.22177.300.1.1.4.4)
Policy-ID #5	(1.3.6.1.4.1.22177.300.2.1.4.4)
Erweiterung	CRL-Verteilungspunkte (2.5.29.31)
Kritisch	NEIN
URI	http://cdp1.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
URI	http://cdp2.pca.dfn.de/dfn-ca-global-g2/pub/crl/cacrl.crl
Erweiterung	Zeitstempelliste der eingebetteten signierten Zertifikate (1.3.6.1.4.1.11129.2.4.2)
Kritisch	NEIN
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	6F 53 76 AC 31 F0 31 19 D8 99 00 A4 51 15 FF 77 15 1C 11 D9 02 C1 00 29 06 8D B2 08 9A 37 D9 13
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 19 07 E2 08 ...
SCT-Version	1
Schlüssel-ID protokollieren	AA E7 0B 7F 3C B8 D5 66 C8 6C 2F 16 97 9C 9F 44 5F 69 AB 0E B4 53 55 89 B2 F7 7A 03 01 04 F3 CD
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 68 E2 4C 02 ...
SCT-Version	1
Log-Operator	Sectigo
Schlüssel-ID protokollieren	55 81 D4 C2 16 90 36 01 4A EA 0B 9B 57 3C 53 F0 C0 E4 38 78 70 25 08 17 2F A3 AA 1D 07 13 D3 0C
Timecode-Start	Montag, 1. Juli 2019 KW 27 um 14:26:49 Mitteleuropäische Sommerzeit
Signatur-Algorithmus	SHA-256 ECDSA
Signatur	70 Byte : 30 44 02 20 5A D9 85 04 ...
SCT-Version	1

Kopplung von Realms; Zertifizierungspfade

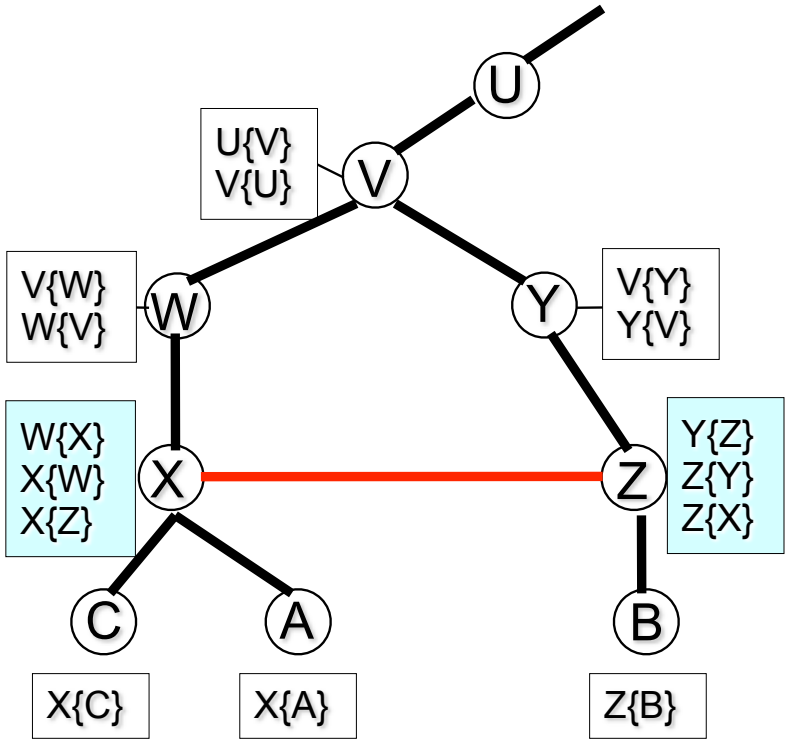
- Bisher wurde nur eine CA betrachtet, nun
- CA Hierarchie:



- Legende: X{A} = Zertifikat ausgestellt von X für A
  (X zertifiziert A)
- A kommuniziert mit B und möchte dessen Zertifikat verifizieren
- Dazu Aufbau eines Zertifizierungs-pfades erforderlich:
  - A braucht folgende Zertifikate X{W},W{V},V{Y},Y{Z},Z{B}
  - Alle Zertifikate längs dieses Pfades müssen verifiziert werden
  - D.h. A braucht öffentliche Schlüssel von: X, W, V, Y und Z
- Im Bsp. eine streng hierarchische CA Infrastruktur
- Optimierung des Pfades?

Kopplung von Realms; Zertifizierungspfade

- Bisher wurde nur eine CA betrachtet, nun
- CA Hierarchie:



- Cross-Zertifizierung nicht entlang der Hierarchieebenen
- Damit Aufgeben des hierarchischen Ansatzes
- Vermaschte bzw. vernetzte CA-Infrastruktur
- Es entsteht ein „Web of Trust“ (vergleichbar mit PGP)
- Pfade deutlich kürzer
- Pfadermittlung und Pfadverwaltung damit aber u.U. deutlich aufwendiger

Widerruf von Zertifikaten

- Falls Schlüssel kompromittiert wurde, muss Zertifikat widerrufen werden
- Dazu Certificate Revocation Lists (CRLs):
  Liste jeder Zertifikats-ID mit Datum der Ungültigkeit; digital signiert von CA
- Problem der Informationsverteilung:
  - Zeitnah, d.h. möglichst aktuell
  - Vollständig
  - Effiziente Verteilung
- Grundsätzliche Ansätze:
  - Push-Modell (regelmäßige Übersendung der CRL)
  - Pull Modell (Verifikator fragt bei Überprüfung aktuell nach, ob Zertifikat noch gültig, oder lädt sich CRL)
  - Vollständige CRL oder Delta-Listen

Online Certificate Status Protocol (OCSP)

- Ermöglicht Clients die Abfrage des Zertifikatszustandes (zeitnah) bei einem Server (OCSP-Responder)
- OCSP-Responder i.d.R. betrieben von ausstellender CA
- Ablauf:
  - Client schickt Hash des zu verifizierenden Zertifikats
  - Responder prüft und antwortet mit einer der folgenden signierten Nachrichten:
    - „Good“ (Zertfikat ist gültig)
    - „Revoked“ (Zertifikat ist widerrufen, mit entsprechender Zeitangabe)
    - „Unknown“ (Responder kennt das Zertifikat nicht)
  - Replay Protection über optionale Zufallszahl (in Client-Nachricht)
  - Client kann Positiv-Antwort fordern; Responder antwortet dann mit Hash des gültigen Zertifikates
- Kein eigenes Transportprotokoll; verwendet HTTP oder HTTPS

OSCP Diskussion

- Vorteile:
  - Geschwindigkeitsvorteil gegenüber CRL
  - Möglichkeit, gesperrte von gefälschten Zertifikaten zu unterscheiden:
    - Responder darf „Good“ nur liefern, wenn Zertifikat gültig
      (Standard erlaubt Good auch wenn Zertifikat nicht in Sperrliste)
  - Individuelle Abfrage für aktuell verwendetes Zertifikat

- Nachteile:
  - Aktualität hängt von Implementierung ab; es gibt Responder, die CRL nutzen
  - Zertifikatskette muss vom Client geprüft werden
    (lässt sich ggf. über Server-based Certification Validation Protocol (SCVP) an den Server auslagern)

Literatur

[MaMa 02] Matsumoto, T. und H. Matsumoto: Impact of artificial ”gummyfingers on finger-print systems. In: Renesse, R. L. van (Herausgeber): Optical Security and Counterfeit Deterrence Techniques IV, Nummer 4677 in Proceedings of SPIE, Januar 2002.

[Mats 02] Matsumotu, T.: Importance of Open Discussion on Adversarial Analyses for Mobile Security Technologies — A Case Study for User Identification —. Presentation, ITU- T Workshop on Security, Seoul, 2002, http://www.itu.int/itudoc/itu-t/workshop/security/present/.

[PPK 03] Prabhakar, S., S. Pankanti und A. K. Jain: Biometric Recognition: Security and Privacy Concerns. IEEE Security and Privacy, 1(2):33–42, March 2003.

[Stal 98] Stallings, W.: Cryptography and Network Security — Principles and Practice. Prentice Hall, 1998.