IT-Sicherheit

- Sicherheit vernetzter Systeme -

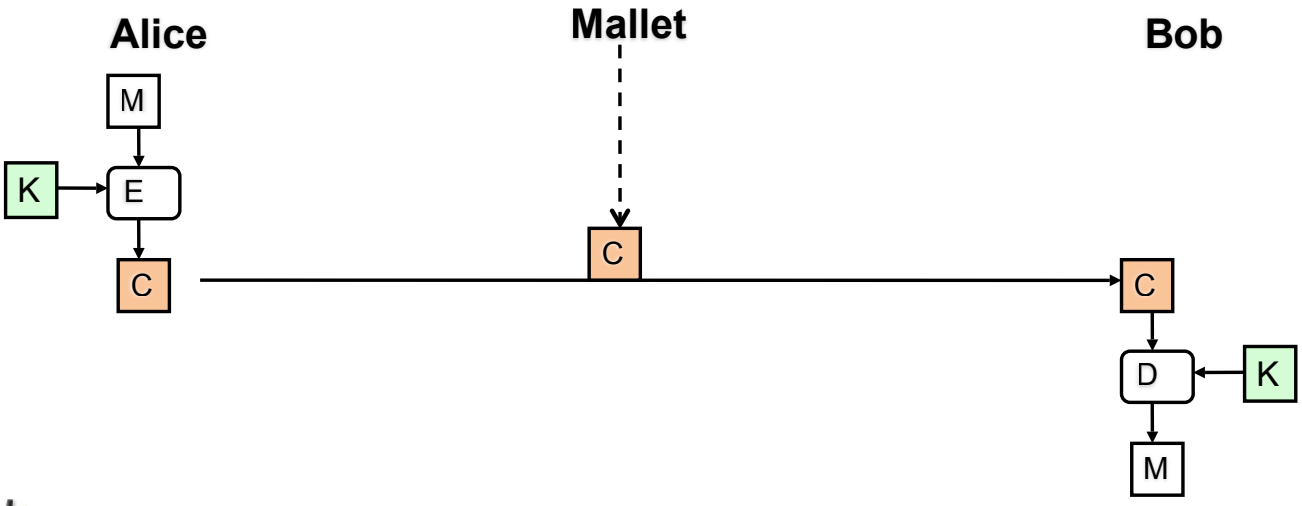
Kapitel 5: Sicherheitsmechanismen

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 **nicht** nutzen



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

Integrität

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

Einschub: US-CERT Alert TA07-352A

- Apple Updates for Multiple Vulnerabilities
- Systems affected:
  - Apple Mac OS X and Mac OS X Server Version <= 10.5.1
- Description: Vulnerabilities in
  - Address Book; Color Sync; CUPS; Desktop Services;
  - Adobe Flash Player; GNU Tar;
  - IO Storage Family; Launch Services; Mail; perl; Python; ruby
  - Quick Look; Safari; Shockwave Plugin; SMB; Spotlight; tcpdump; Xquery
- Impact:
  - Remote Code Execution; DoS;
  - Information disclosure
  - Surreptitious video conference initiation
- Solution:
  - Apply update

Einschub: US-CERT Alert TA07-355A

- Adobe Updates for Multiple Vulnerabilities
- Systems affected:
  - Adobe Flash Player; Versions:
    - 9.0.48.0 and earlier
    - 8.0.35.0 and earlier
    - 7.0.70.0 and earlier
- Description: Vulnerabilities in
  - Multiple input validation errors
  - Update to prevent Cross Site Scripting attacks
- Impact:
  - Remote Code Execution; DoS;
  - Cross Site Scripting attacks
  - DNS rebinding attacks; conduct port scans
- Solution:
  - Apply update

Authentisierung: Arten

- Authentisierung wird unterschieden in:
  1. Authentisierung des Datenursprungs
  2. Benutzerauthentisierung
  3. Peer Entity Authentisierung
  - Weitere Unterteilung von 2. und 3.
    - Einseitig oder
    - Zwei- bzw. mehrseitige Authentisierung

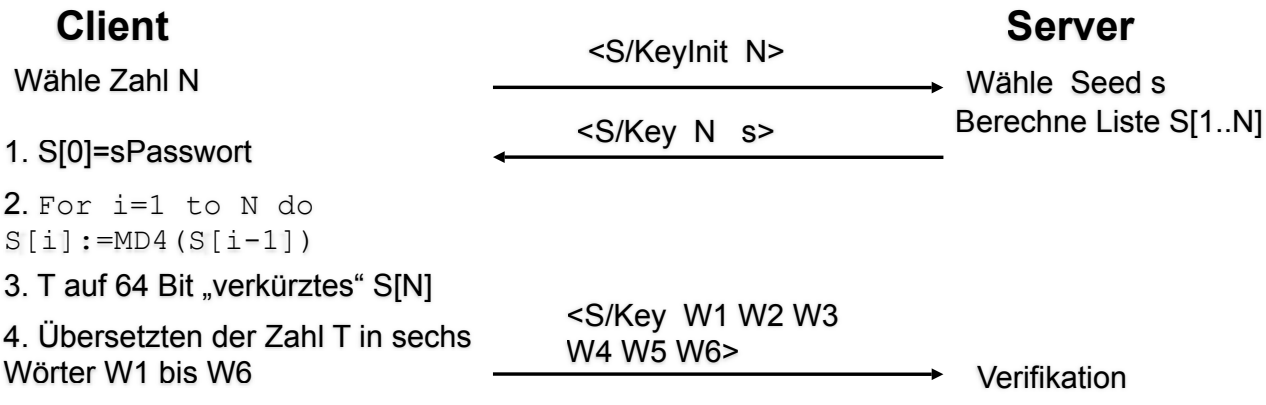
- Grundsätzliche Möglichkeiten zur Authentisierung:
  1. Wissen (Something you know)
  2. Besitz (Something you have)
  3. Persönliche Eigenschaft (Something you are)
  4. Kombinationen aus 1. – 3.

Benutzerauthentisierung

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

Einmal-Passwort Verfahren: S/Key

- Authentisierungsserver kennt Passwort des Benutzers



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

S/Key

- 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
  - Je 11 Bit von T liefern - als Zahl interpretiert - die Adresse des Wortes
  - Bsp. für einen solchen „Satz“: FORT HARD BIKE HIT SWING

OTP (One Time Password System)

- Entwickelt von Bellcore [RFC 2289] als Nachfolger für S/Key
- Schutz vor Race Angriff:
  - S/Key erlaubt 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 Hex Notation
- Passwort muss mind. 10 und kann bis 64 Zeichen lang sein
- Auf Passwort S[N] folgt S[N+1] und nicht S[N-1]
  - Damit bei Client u. Server keine Liste mehr notwendig
  - S[i+1] = Hash(S[i])
- Verwendung von IPSec wird „empfohlen“

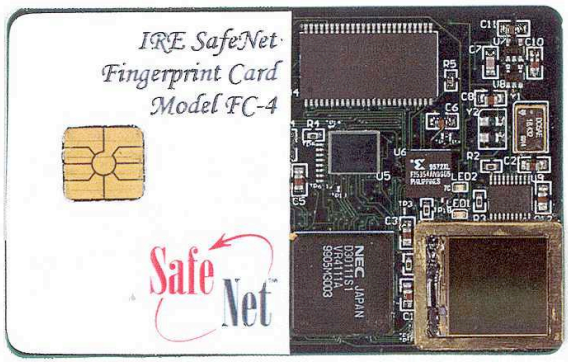
Angriffe aus S/Key und OTP

- Dictonary Attack:
  - Alle Nachrichten werden im Klartext übertragen, z.B.
    <S/Key 99 12745> <S/Key A GUY SWING GONE SO SIP>
  - Angreifer kann mit diesen Informationen versuchen 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

Authentisierung: Smart Cards

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

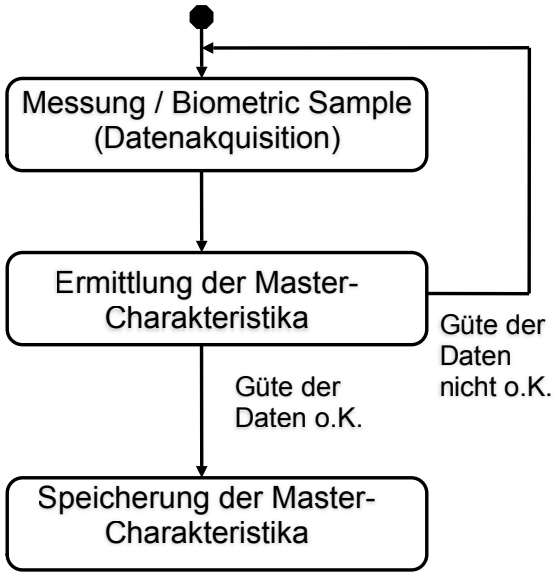
       - Bsp.: Prozessor Karte mit Fingerabdruck-Sensor



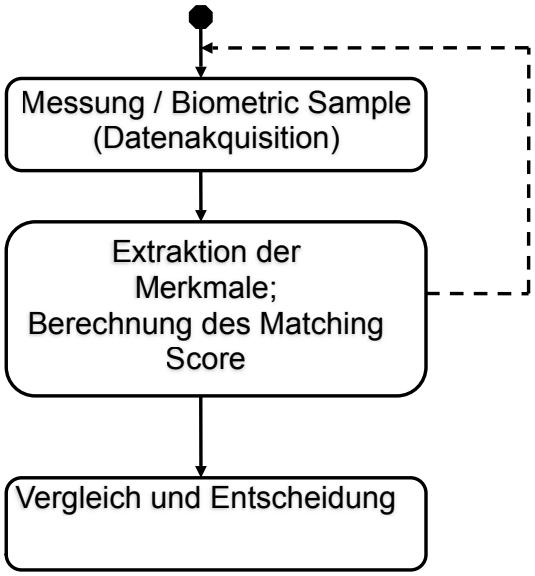
- Zugangsdaten werden auf Karte gespeichert oder erzeugt
  - Schutz der Daten ggf. durch Paßwort und/oder Verschlüsselung

Biometrie: allgemeines Vorgehen

- Initialisierung des Systems pro Nutzer
  - Viele Messungen möglich



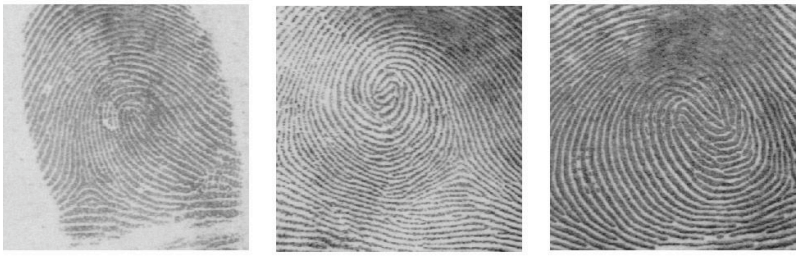
- Authentisierung
  - I.d.R. nur eine Messung möglich



Biometrie am Bsp. Fingerabdruck

- Identifikation anhand des Fingerabdrucks hat lange Geschichte
- Merkmale von Fingerabdrücken sind gut klassifiziert
  Bsp. aus [KaJa96]

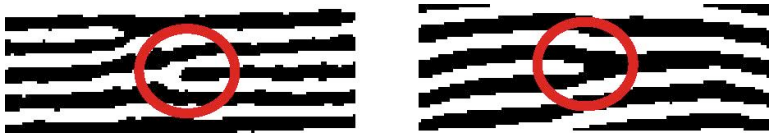
Bogen | gespannter Bogen | linke Schleife



rechte Schleife | Knäuel | Doppelschleife

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)

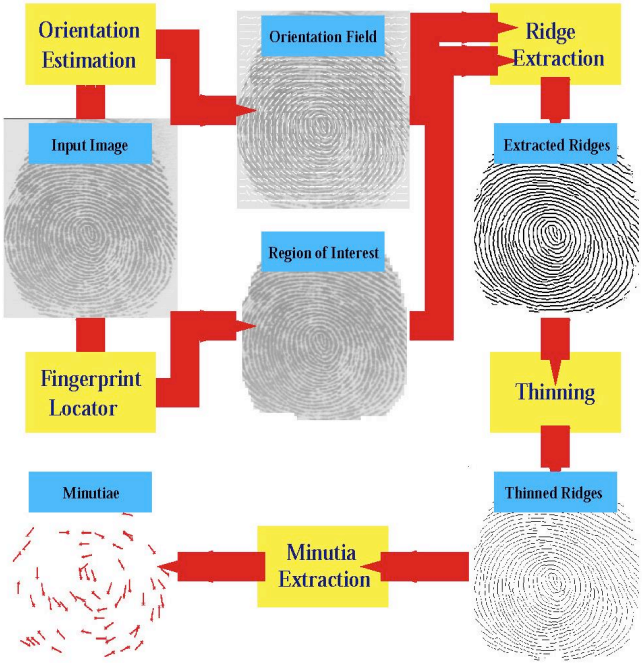


Rillen-Ende Rillen-Verzweigung

  - Solche äquivalente Rillenstrukturen werden zu einer Minuzie zusammengefasst
- Merkmale: Lage der Minuzien
  - Absolut bezüglich des Abdrucks, Relativ zueinander
  - Orientierung bzw. Richtung

Fingerabdruck: Minutiae Extraktion

- Algorithmus: Beispiel aus [JHPB 97]

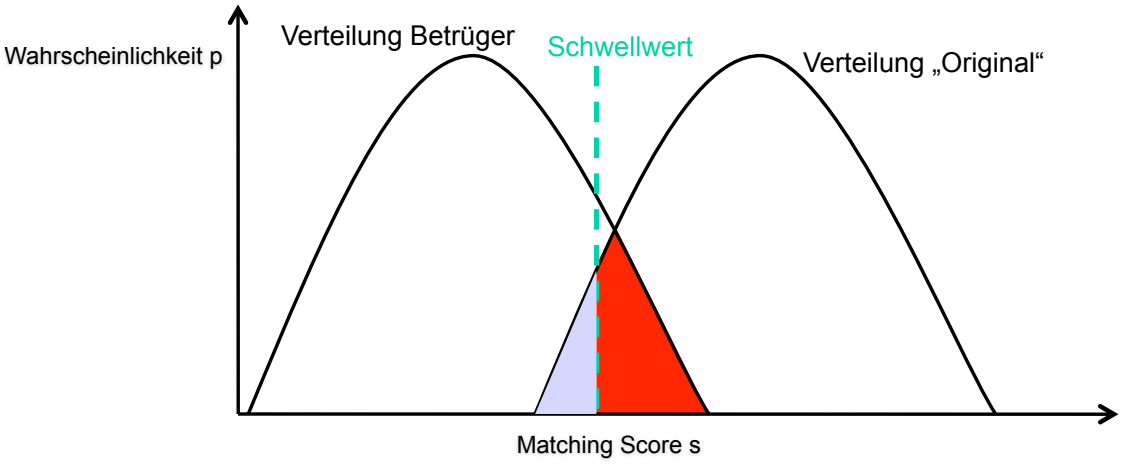


Fingerabdruck: Angriffe

- 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

Biometrischen Authentisierung: Fehlerarten

- Biometrische Systeme sind fehlerbehaftet
- Fehlerarten:
  1. Falsch Positiv (Mallet wird als Alice authentisiert)
  2. Falsch Negativ (Alice wird nicht als Alice identifiziert)
- Fehler sind abhängig von Schwellwerteinstellungen



Biometrische Authentisierung: Fehlerraten

- Abschätzung der Fehlerraten:
  N: Anzahl der Identitäten
  FP: Falsch Positiv
  FN: Falsch Negativ
- Es gilt [PPK03]:

$$FN(N) \cong FN$$

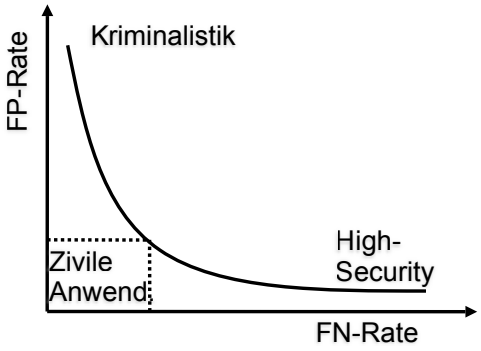
$$FP(N) \cong 1 - (1 - FP)^N \cong N \times FP$$

  falls

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

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

- Fehlerraten, bzw. Einstellung der Schwellwerte abhängig vom Anwendungsszenario



- Platzierung von Anwendungen?
  - Hohe Sicherheitsanforderungen
  - Kriminalistische Anwendungen
  - "Zivile" Anwendungen

Benutzerauthentisierung: multimodale Systeme

- Sicherheit läßt sich durch multimodale Systeme deutlich erhöhen
- Multimodale Systeme kombinieren versch. Verfahren

	Wissen	Besitz	Biometrie
Wissen			
Besitz			
Biometrie			

- Auch verschiedene biometrische Verfahren lassen sich kombinieren:
  - Erhöhung der Sicherheit
  - Verringerung der Fehlerraten
  - Z.B. Verwendung von mehr als einem Finger

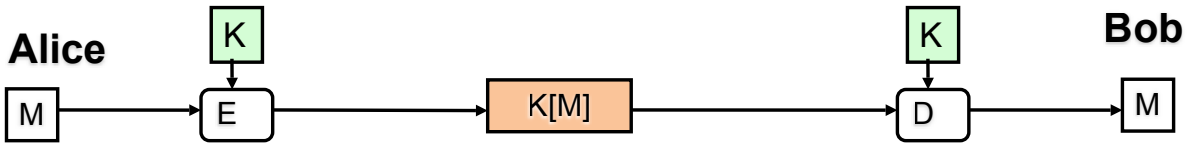
Authentisierung des Datenursprungs

- 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

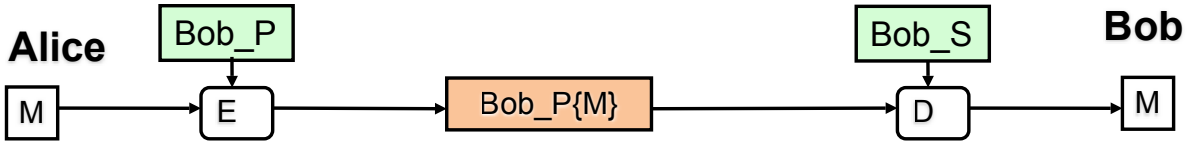
Authentisierung durch symm. Verschlüsselung



- Merkmale:
  - ❑ Authentisierung des Datenursprungs (Nachricht kann nur von Alice stammen)
  - ❑ Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - ❑ Vertraulichkeit der Daten (nur Alice und Bob kennen K)

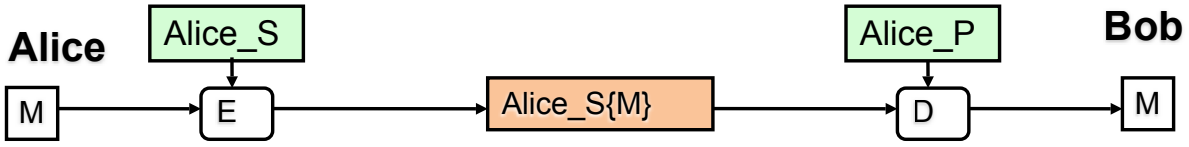
- „Nachteile“:
  - ✶ Sender kann die Sendung leugnen
  - ✶ Alice / Bob können Zugang / Empfang nicht beweisen

Authentisierung durch asym. Verschlüsselung



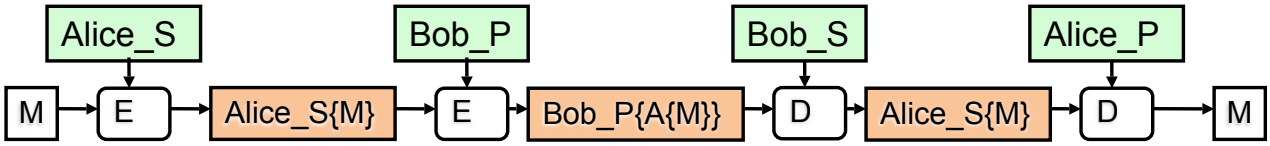
- Merkmale:
  - Bob wird nicht explizit authentisiert, aber nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten (nur Bob kennt seinen privaten Schlüssel)
  - KEINE Authentisierung des Datenursprungs (Jeder kann senden)
  - Sender kann die Sendung leugnen
  - Alice / Bob können Zugang / Empfang nicht beweisen

Authentisierung: digitale Signatur



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

Authentisierung: asym. Verschlüsselung + Signatur

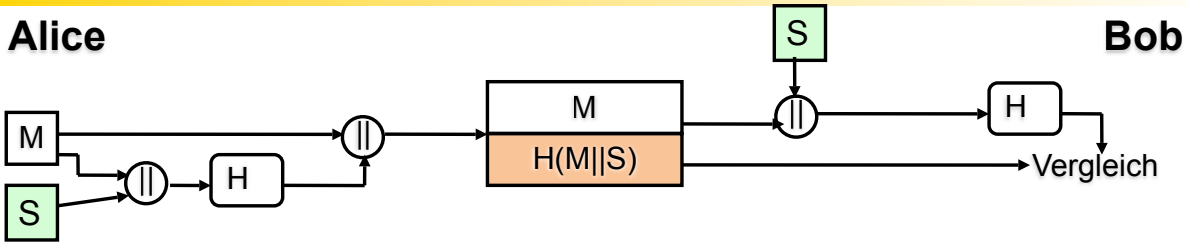


- Merkmale:
  - Authentisierung des Datenursprungs
  - Nur Bob kann Nachricht nutzen
  - Vertraulichkeit der Daten
  - Vertraulichkeit der Signatur
  - Alice kann Sendung nicht leugnen
- Operationen für Signatur und asymmetrische Verschlüsselung sind „teuer“
- Alice kann Zugang nicht beweisen
- Bei allen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

Einschub: US-CERT Alert TA08-008A

- Microsoft Updates for Multiple Vulnerabilities
- Systems affected:
  - Microsoft Windows
- Description:
  - TCP/IP kernel processes; improper handling of multicast and ICMP
  - LSASS (Local Security Authority Subsystem Service) elevation of privileges due to improper local procedure call (LPC) (only local elevation of privileges)
- Impact:
  - Remote Code Execution; DoS;
  - Gain elevated privileges
  - Crash an affected system
- Solution:
  - Apply update

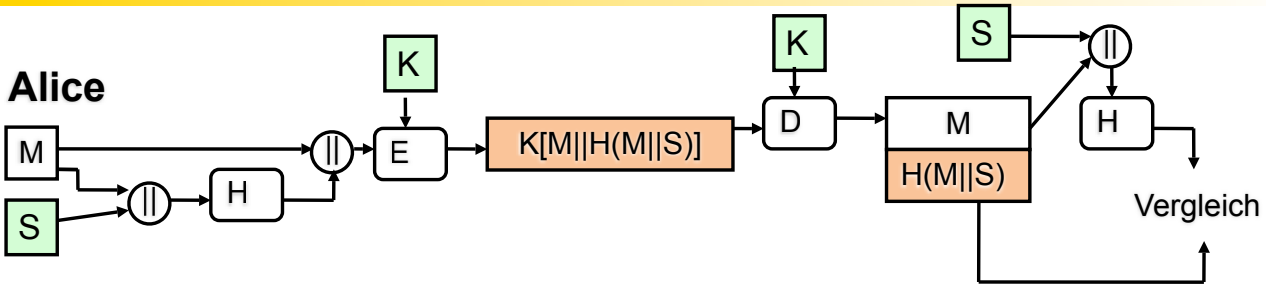
Verwendung von Hash-Fkt. zur Authentisierung



- 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

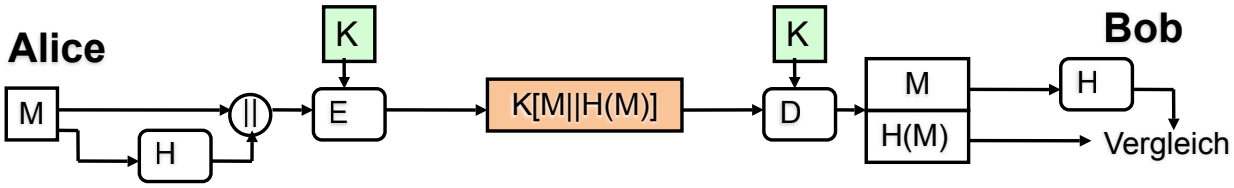
Verwendung von Hash-Fkt. zur Authentisierung



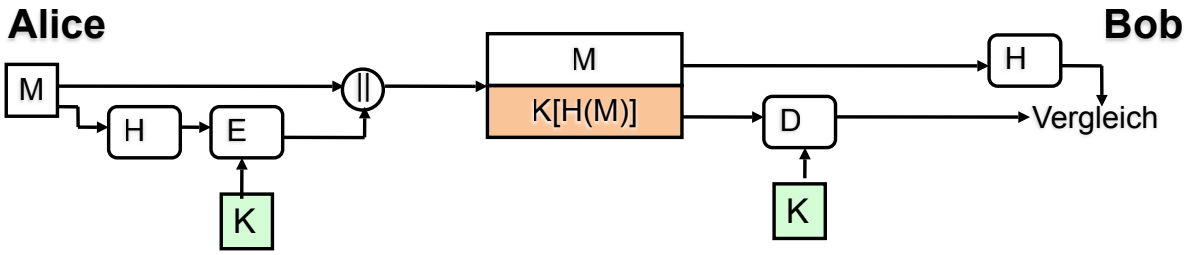
- Zusätzlich Vertraulichkeit durch Verschlüsselung

- Alice kann Sendung leugnen
- Alice/Bob können Zugang / Empfang nicht beweisen

Verwendung von Hash-Fkt. zur Authentisierung

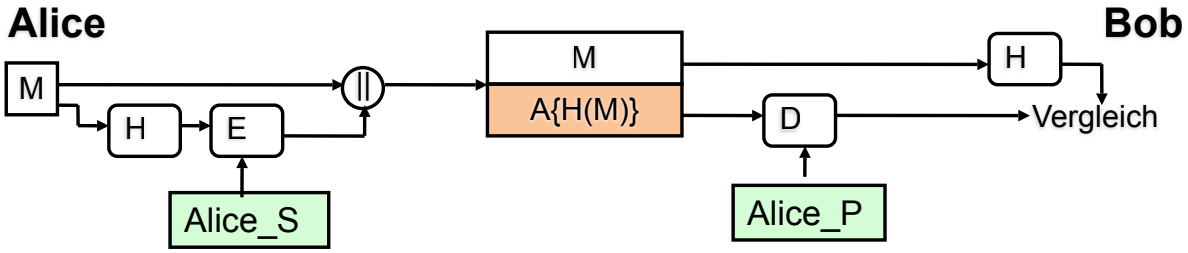


- Authentisierung des Datenursprungs (durch Schlüssel K)
- Vertraulichkeit
- Integrität



- Authentisierung und Integrität, keine Vertraulichkeit

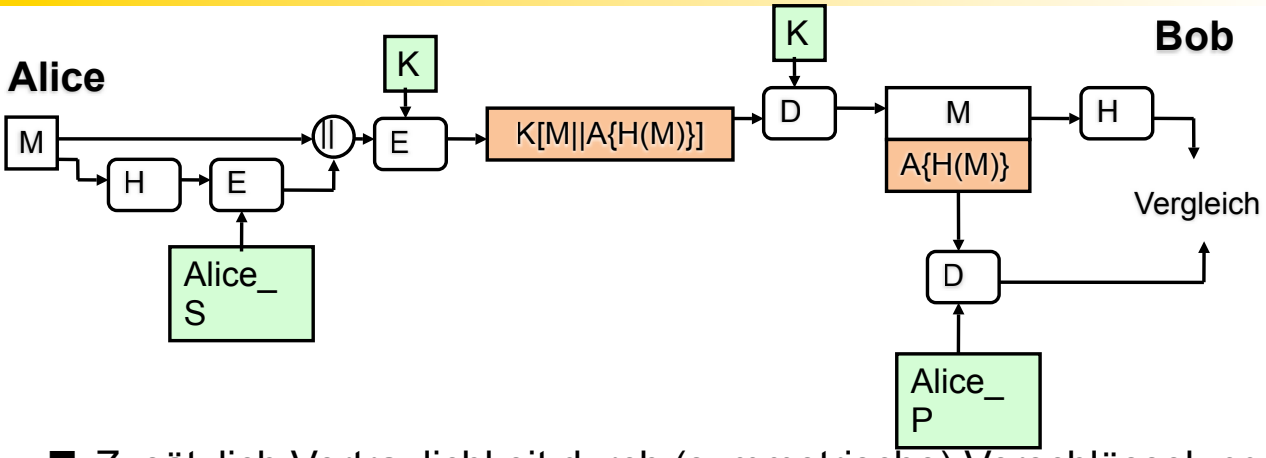
Verwendung von Hash-Fkt. zur Authentisierung



- 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

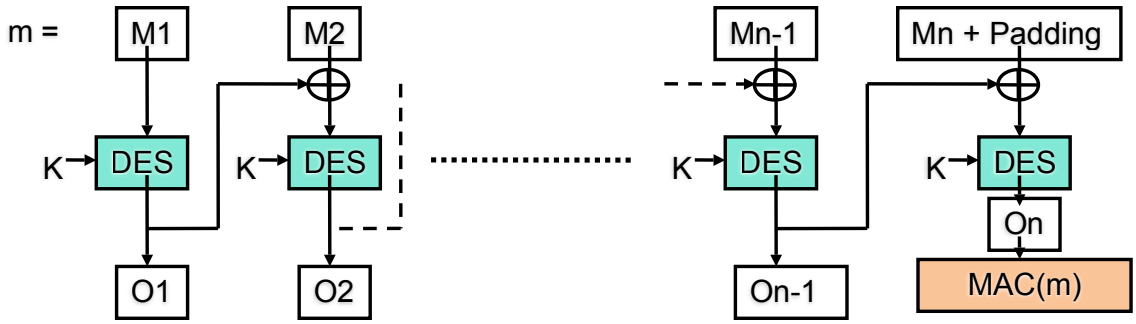


- Zusätzlich Vertraulichkeit durch (symmetrische) Verschlüsselung
- Am häufigsten verwendetes Verfahren

- Alice kann Zugang nicht beweisen

Authentisierung: MAC

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



- DES im CBC Mode

Sicherheit von MACs

- Wie kann der MAC angegriffen werden?
- Brute force:
  - MAC ist n-Bit lang, Schlüssel K ist k Bit 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

Hashed MAC (HMAC)

- Gesucht: MAC der **nicht** symm. Verschlüsselung sondern kryptographische Hash-Funktion zur Kompression verwendet
  - Hashes wie MD5 sind deutlich schneller wie bspw. DES
- Problem: Hash-Funktionen verwenden keinen Schlüssel
- Lösung HMAC
  - Beliebige Hash-Funktion H verwendbar, die auf (Input) Blöcken arbeitet
  - Sei b die Blocklänge
  - Beliebige Schlüssellänge K mit |K| <= b verwendbar
  - Falls |K| < b:
    - Auffüllen mit 0 Bytes bis |K+| = b; d.h. K+ = K||0....0
  - Schlüssel wird mit Input- (`ipad`) bzw. Output-Pattern (`opad`) XOR verknüpft
    - `ipad` = 0x36 b mal wiederholt
    - `opad` = 0x5c b mal wiederholt

HMAC Algorithmus

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

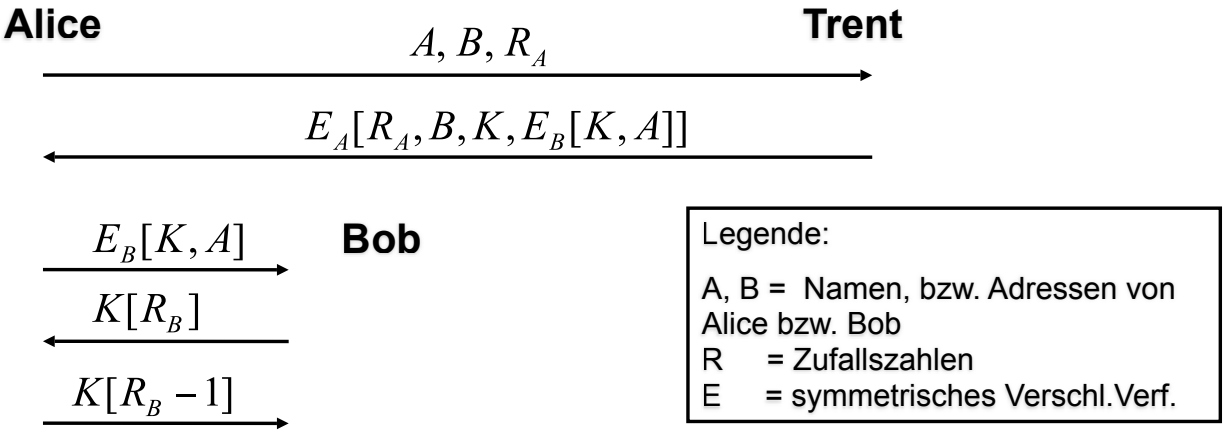
1. K^+ := Schlüssel K mit Nullen auffüllen bis dieser b Bits lang ist
2. b Bit Block $S_i := K^+ \text{ XOR } ipad$
3. Nachricht m mit dem Block S_i konkatenieren
4. Hash-Wert von $S_i||m$ berechnen
5. b Bit Block $S_o := K^+ \text{ XOR } opad$
6. S_o mit dem Ergebnis von 4. Konkatenieren
7. Hash-Wert über das Ergebnis von 6 berechnen

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

Authentisierungsprotokolle: Needham Schröder

- Verwendet vertrauenswürdigen Dritten Trent (Trusted Third Party)
- Optimiert zur Verhinderung von Replay-Angriffen
- Verwendet symmetrische Verschlüsselung
- Trent teilt mit jedem Kommunikationspartner eigenen Schlüssel

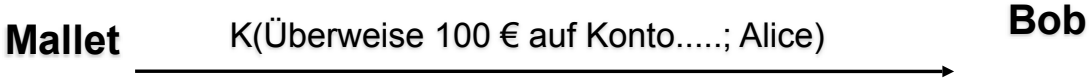


Legende:

A, B = Namen, bzw. Adressen von Alice bzw. Bob
R = Zufallszahlen
E = symmetrisches Verschl.Verf.

Needham Schröder Protokollschwäche

- Problem: Alte Sitzungsschlüssel K bleiben gültig
- Falls Mallet an alten Schlüssel gelangen kann, wird Maskerade-Angriff möglich



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

Authentisierungsprotokolle: 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 beim Server 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

Kerberos Authentisierungsdaten

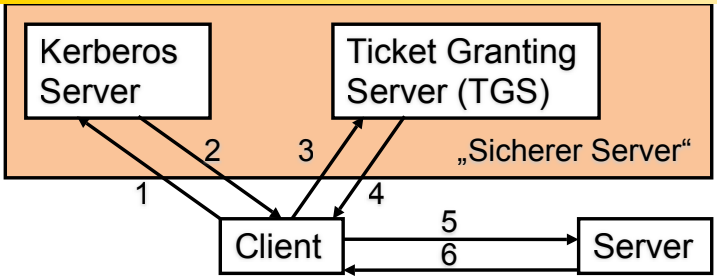
- Authentisierung basiert auf gemeinsamen Schlüssel
- Kerberos arbeitet mit Credentials, unterschieden werden
  1. Ticket
  2. Authenticator
- **Ticket**
  - als „Ausweis“ für die Dienstnutzung; nur für einen Server gültig
  - wird vom Ticket Granting Server erstellt
  - **keine** Zugriffskontrolle über 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

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

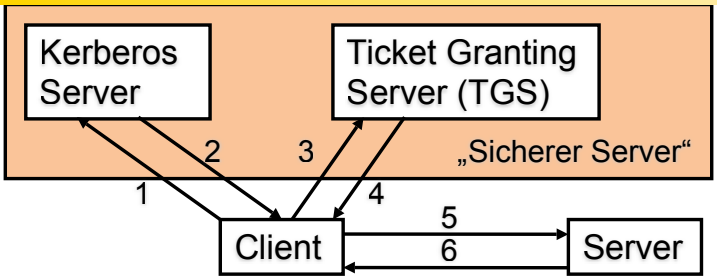
Kerberos Modell



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)

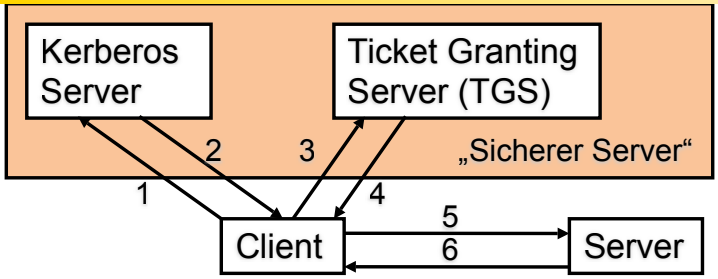
- Im folgenden Kerberos V.5 vereinfacht, d.h. ohne Realms und Optionenlisten; exaktes Protokoll [RFC 1510, Stal 98]

Kerberos: Initiales Ticket (ein Mal pro Sitzung)



c	=	Client
s	=	Server
a	=	Adresse
v	=	Lebensdauer
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 $T_{c,tgs} = tgs, c, a, t, v, K_{c,tgs}$



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

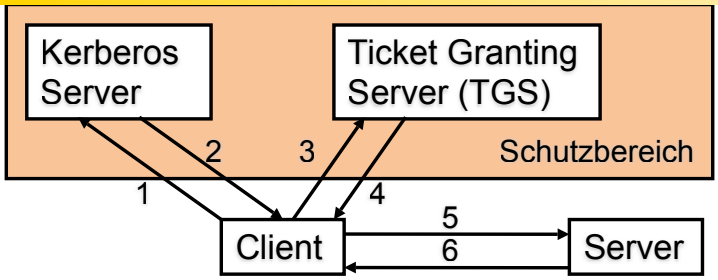
3. Request für Server Ticket:

   $s, K_{c,tgs}[A_{c,tgs}], K_{tgs}[T_{c,tgs}]$ 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 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 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:

   $K_{c,s}[t,key,seqNo]$

Kerberos Bewertung

- Sichere netzwerkweite Authentisierung auf Ebene der Dienste
- Authentisierung basiert auf IP-Adresse
  - IP Spoofing u.U. möglich
  - Challange 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) gesichert werden
- Verlässt sich auf „vertrauenswürdige“ Software (Problem der Tojanisierung, vgl. CA-2002-29)

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