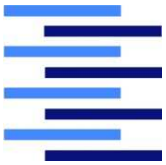


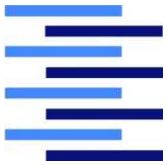
Kapitel 5
Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
3. Modelle für die Autorisation (Zugriffskontrolle)
4. Autorisation in UNIX / Windows

Benutzerauthentifikation

- Identifikation:
  - "Wer ist diese Person?" (→ Zuordnung von gespeicherten Attributen)
- Authentifikation:
  - "Ist die Person wirklich diejenige, für die sie sich ausgibt?"
- Basisprinzipien der Authentifikation:
  - Authentifikation durch (geheimes) Wissen
    - z.B. Passworte, PINs
  - Authentifikation durch (persönlichen) Besitz
    - z.B. Smart-Card (mit privatem Schlüssel), SIM-Karte (Handy), USB-Token
  - Authentifikation durch (biometrische) Merkmale
    - z.B. Fingerabdruck
- Die Authentifikation muss vor Zugang zum System geschehen!

Passwort-Richtlinien



- Ein Passwort sollte
  - ... mindestens 8 Zeichen lang sein
  - ... kein Name oder Begriff aus dem Lexikon sein
  - ... Buchstaben, Zahlen und evtl. Sonderzeichen in nicht-sinnvollen Kombinationen enthalten *(→ Hashverfahren über Merksatz!)*
  - *... in regelmäßigen Abständen geändert werden (???)*
- Das Betriebssystem sollte
  - ... die Regeln zur Passwortbildung überwachen
  - ... nur eine geringe Zahl von Fehlversuchen bei der Anmeldung gestatten und danach die Kennung sperren

Authentifikation durch Besitz

- **Magnetstreifenkarte**
  - z.B. Parkhaus-Karte (Einweg-Karte)
- **Chipkarte**
  - **Speicherkarte**: Nicht-flüchtiger Speicher (EEPROM)
    - keine CPU, nur genau eine Funktion, sehr billig
    - Evtl. zusätzliche Sicherheitslogik zur PIN-Speicherung und Überprüfung, Zähler für Fehlversuche
  - **Smartcard**: Chip mit Mikroprozessor, programmierbarem Speicher und ggf. RFID-Schnittstelle oder zusätzlichem Magnetstreifen
    - z.B. HAW-Chipkarte, ePass, ePersonalausweis, …
    - Speicherung privater Schlüssel möglich
- **USB-Token**
  - Funktionale Kombination von Smartcard und Chipkartenleser
  - Erzeugung von Zufallszahlen und privaten RSA-Schlüsseln möglich

Authentifikation durch biometrische Merkmale

- Allgemeine Vorgehensweise:
  Messen von biometrischen Merkmalen einer Person und Abgleich mit gespeicherten Werten

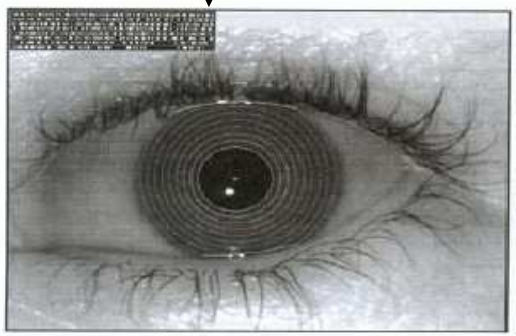
1. Schritt: Messdatenerfassung durch biometrischen Sensor und Vorverarbeitung („Feature Extraction“)
2. Registrierung eines Benutzers:
   - ➢ Speicherung der Daten als Referenzwerte in einer Datenbank
3. Authentifikation eines Benutzers:
   - ➢ Suchen des Benutzers in der Datenbank
   - ➢ Vergleichen der Daten mit gespeicherten Referenzwerten (Toleranz!?)

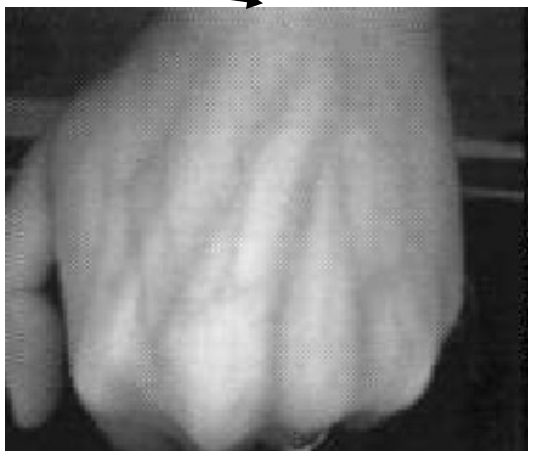
Anforderungen an ein biometrisches Merkmal

- Eindeutigkeit: Merkmal ist für jede Person verschieden
- Beständigkeit: Merkmal ist unveränderlich
- Fälschungssicherheit
- quantitative Erfassbarkeit
- Performance: hohe Prüfungsgenauigkeit und -geschwindigkeit
- Akzeptanz des Merkmals beim Benutzer

Klassen biometrischer Merkmale

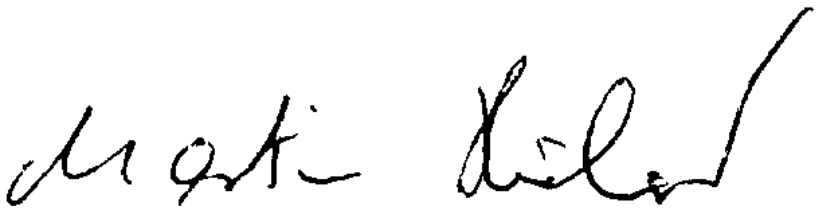
- Physiologische Merkmale (statisch):
  - Feststehende physikalische Merkmale
  - keine oder nur sehr begrenzte Möglichkeiten zur Auswahl verschiedener Referenzdaten oder Änderung von Referenzdaten
  - Beispiele: Fingerabdruck, Gesichtsbild, Retina, Handgeometrie, Iris, Venenmuster





Klassen biometrischer Merkmale

- Verhaltensmerkmale (dynamisch):
  - Biometrisches Merkmal ist nur bei bestimmter Aktion des Benutzers vorhanden
  - Zeitlicher Verlauf liefert ebenfalls Kennwerte (Druckänderungen, Frequenzänderungen, …)
  - Beispiele: Unterschriften-Dynamik (Form, Druckverlauf, ..), Sprache, Tippverhalten (Keystroke)



- **Bewertung**:
  - Nicht so beständig wie physiologische Merkmale
  - Oft Kombination mit Passwortabfrage
    → Verifikation (Identität wird vorgegeben) statt Identifikation eines Benutzers

Beispiel: Fingerabdruck

Sensor

Graustufen-Bild eines Finger-abdrucks

- Muster von Rillen
- Merkmal: **Minutien** (lat. für Kleinigkeit, Detail):
  - End- und Verzweigungspunkte von Rillen
  - Lage und Anzahl ist charakteristisch für jeden Menschen

Verarbeitung von Minutien

Rillenmuster nach dem Herausfiltern des Hintergrunds, z.B. Filtern von Veränderungen durch Schmutz, Verletzungen etc.

Feature-Extraktion: Bestimmen der Minutien (10 –100) als Endpunkte und Verzweigungswinkel
Danach: Abgleich des Minutien-Musters mit Referenzwerten

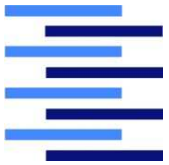
Biometrie: Mögliche Probleme

- Sicherheit der Erkennung (Abweichungen durch Umweltbedingungen: Lichtverhältnisse, Körperzustand, ...)
  - Fehler: Akzeptieren eines Unberechtigten
  - Fehler: Abweisen eines Berechtigten
- Fälschung von Körpermerkmalen (Handabdruck aus Gips, Verwenden von Fotos, ...)
  - Gewaltkriminalität
- „Privater Schlüssel“ ist unveränderlich
  - Kompromittierung der Referenzwerte: Schlüssel kann nicht gewechselt werden
- Unbemerkte Registrierung (Videoüberwachung, …)
  - Informationelle Selbstbestimmung
- Benutzerakzeptanz (Blut-, Urinproben???)

Kapitel 5
Authentifikation und Autorisation

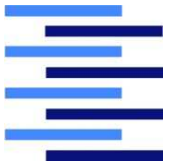
1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
    1. Einführung
    2. Kerberos
    3. Web-Authentifikationsdienste
3. Modelle für die Zugriffskontrolle (Autorisation)
4. Autorisation in UNIX / Windows

Passwort-Verfahren für verteilte Systeme



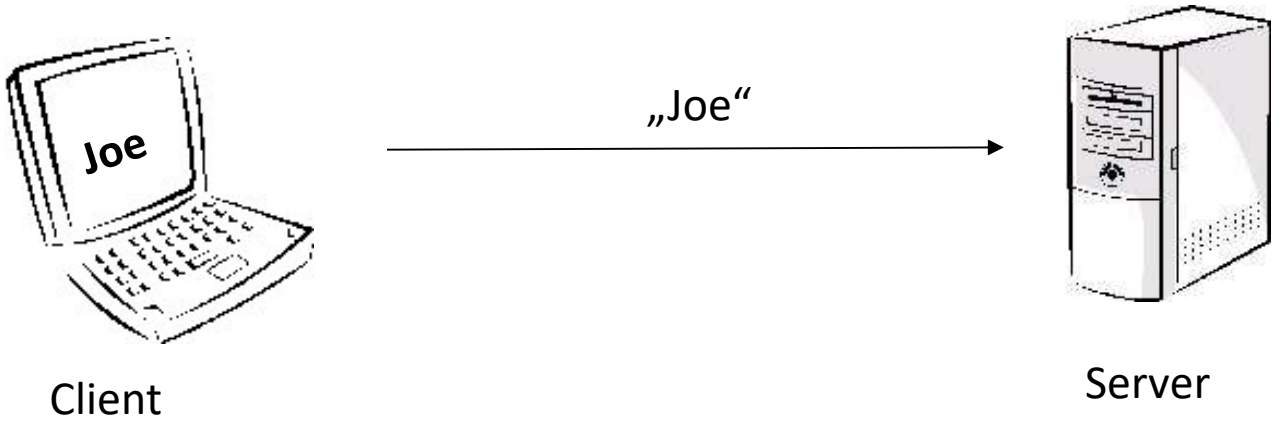
- Einmal-Passwörter
  - Der Benutzer erhält eine Liste mit systemgenerierten Passwörtern *(z.B. mehrfache Anwendung einer kryptographischen Hashfunktion)*
  - Nach Verwendung eines Passworts wird es verworfen und bei der nächsten Anmeldung
    - das nächste Passwort aus einer Liste genommen (z.B. TAN) oder
    - übermittelt (z.B. mTAN) oder
    - generiert (z.B. ChipTAN).
  - Beispiel: Online-Banking
- Challenge-Response-Verfahren
  - Der Server stellt zur Authentifikation eine Frage *(Challenge)* und prüft, ob die Antwort *(Response)* des Clients korrekt ist

Authentifizierung – Protokollversuch 0.1



Ziel: Der Server möchte, dass Joe ihm seine Identität “beweist”

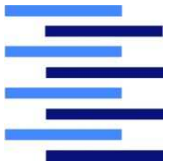
Protokollversuch 0.1: Client sendet “ich bin Joe” (als Benutzerkennung)



Angriffsszenario?

Jeder Angreifer kann einfach behaupten, Joe zu sein!

Authentifizierung – Protokollversuch 0.2



Protokollversuch 0.2:

Client sendet “ich bin Joe” in einem IP-Paket,
das die Quell-IP-Adresse von Joes Client enthält (die der Server kennt)

Client

„Joe“, IP-Adresse (Client)

Server

Angriffsszenario?

Ein Angreifer kann ein Paket mit gefälschter Absenderadresse erzeugen (“IP-Spoofing”)

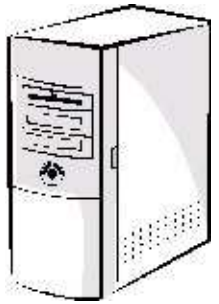
Authentifizierung – Protokollversuch 0.3

Protokollversuch 0.3:

Client sendet “Ich bin Joe” und schickt Joes geheimes Passwort als “Beweis” mit (der Server kennt das Passwort).

Client

„Joe“, „geheimesPasswort“

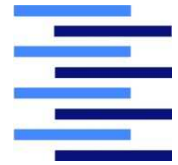


Server

Angriffsszenario?

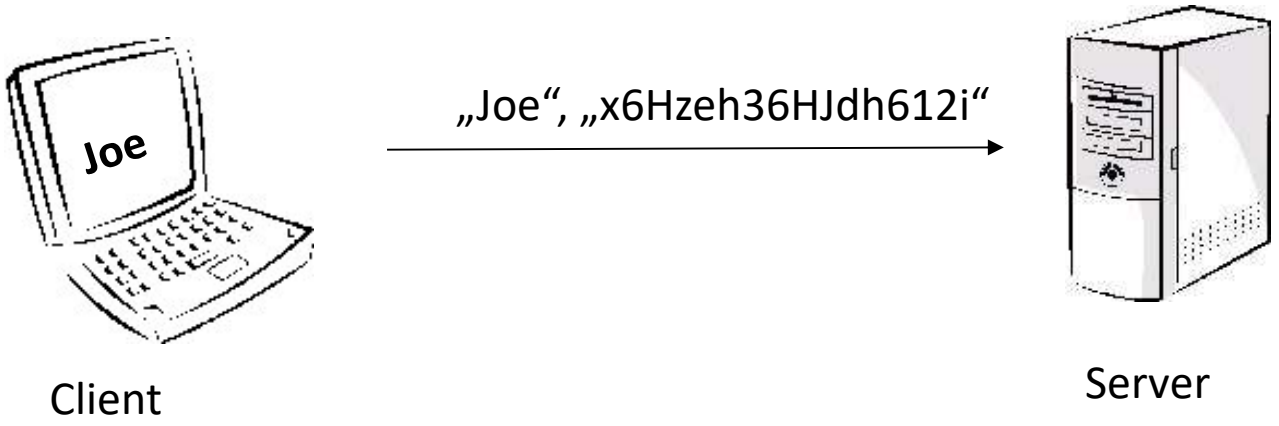
Sniffing-Angriff: Ein Angreifer liest das unverschlüsselte Passwort mit und wiederholt es später in seiner eigenen Anfrage an den Server

Authentifizierung – Protokollversuch 0.4



Protokollversuch 0.4:

Client sendet “Ich bin Joe” und schickt sein *verschlüsseltes* geheimes Passwort als “Beweis” mit (der Server kennt den Schlüssel und das Passwort).



Angriffsszenario?

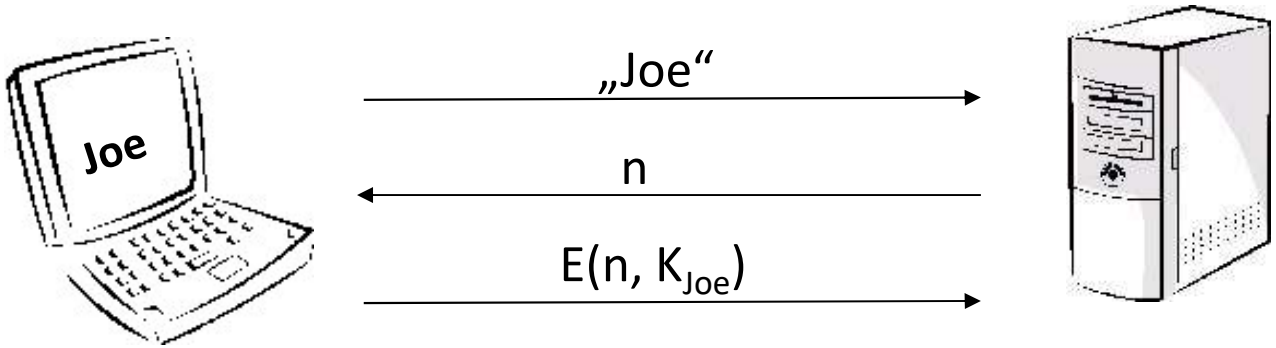
Replay-Angriff: Ein Angreifer zeichnet das vom Client gesendete Datenpaket auf *(ohne das Passwort zu entschlüsseln)* und benutzt es später für seine eigene Anfrage an den Server

Challenge-Response-Protokoll 1.0

Ziel: Replay-Angriff verhindern

Nonce: Zufallszahl n, die genau einmal als “Challenge” verwendet wird (“Number used once”)

Challenge-Response-Protokoll 1.0: Um zu beweisen, dass Joe “live” an der Kommunikation teilnimmt, schickt der Server eine Nonce n, die Joe symmetrisch verschlüsseln (Schlüssel = Joes Passwort) und als “Response” zurückschicken muss (der Server kennt Joes Passwort).



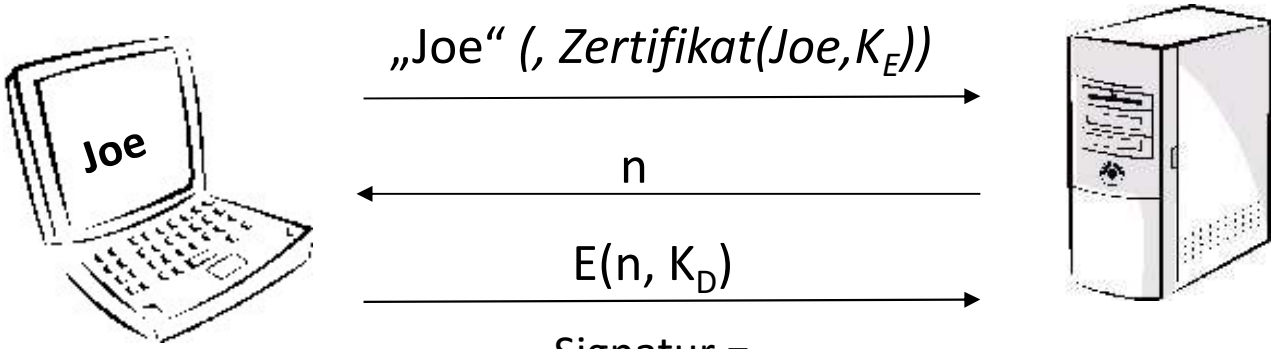
Der Server weiß: Wenn $D(E(n, K_{Joe}), K_{Joe}) = n$, dann muss gelten:
a) Die Response $E(n, K_{Joe})$ kommt von Joe, denn nur Joe kennt außer dem Server den richtigen Schlüssel (Passwort)
b) Die Antwort kann nicht älter als der Nonce sein → kein Replay

Challenge-Response-Protokoll 2.0

Wie Challenge-Response-Protokoll 1.0, aber mit Public Key-Verfahren:

Der Server kennt den öffentlichen Schlüssel K_E von Joe (ggf. über Zertifikat)

→ Joe kann den Nonce signieren!



Der Server weiß: Wenn D(E(n, K_D), K_E) = n, dann muss gelten:
a) Die Response E(n, K_D) kommt von Joe, denn nur Joe kennt den privaten Schlüssel K_D zum öffentlichen Schlüssel K_E
b) Die Antwort kann nicht älter als der Nonce sein → kein Replay

Kapitel 5
Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
   1. Einführung
   2. Kerberos
   3. Web-Authentifikationsdienste
3. Modelle für die Zugriffskontrolle (Autorisation)
4. Autorisation in UNIX / Windows

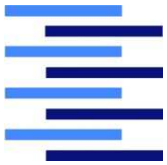
Fallbeispiel: Kerberos (RFC 4120)

- **Aufgaben**
  - Sichere Authentifikation von Benutzern und Computern in einem (lokalen) Netz
  - Realisierung eines **Single-Sign-On**-Service für Benutzer:
    - Einmal anmelden (Kennung und Passwort)
    - Auf alle durch Kerberos verwalteten Server authentifiziert zugreifen
- Am MIT Mitte der 80er Jahre entwickelt (Projekt Athena)
- Sowohl als Open Source als auch in kommerzieller Software verfügbar *(z.B. Bestandteil von Windows - ActiveDirectory)*

Kerberos - Design

- Jeder Benutzer und jeder Server(dienst) hat einen eigenen geheimen Schlüssel (bei Benutzern aus dem Passwort abgeleitet)
- Die einzige Instanz, die alle Schlüssel (Passwörter) kennt, ist der Kerberos Server, auch **Key Distribution Center (KDC)** genannt
- Der Benutzer muss
  - beim KDC registriert sein
  - beim Einloggen einmalig seine Authentizität durch Angabe von Kennung und Passwort beweisen
- Passwörter/Schlüssel werden nie als Klartext, sondern immer verschlüsselt über das Netzwerk versendet
- Verschlüsselt wird symmetrisch mit DES, 3DES oder AES (in Kerberos V5)
- Datenrepräsentation (Protokollspezifikation) mittels ASN.1

Kerberos – Begriffe (1)

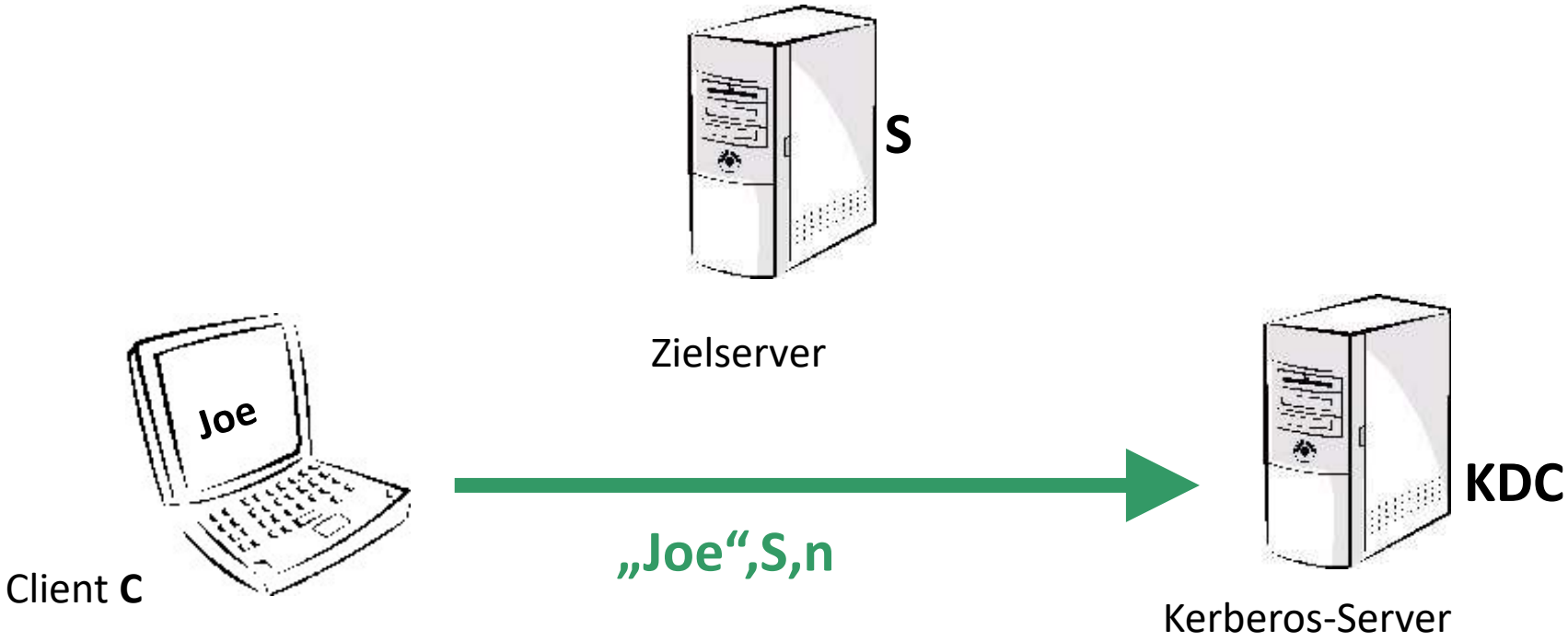


- **Principal**
  - Eindeutig benannter Benutzer (Client) oder Server(dienst), der an der Netzwerkkommunikation teilnimmt
- **Session key („Sitzungsschlüssel“)**
  - Eine Zufallszahl, die zeitlich befristet als geheimer Schlüssel genutzt wird
- **Ticket**
  - Eine Nachricht, die beweist, dass sich ein Principal (Client) vor kurzem gegenüber dem KDC authentifiziert hat

Kerberos – Begriffe (2)

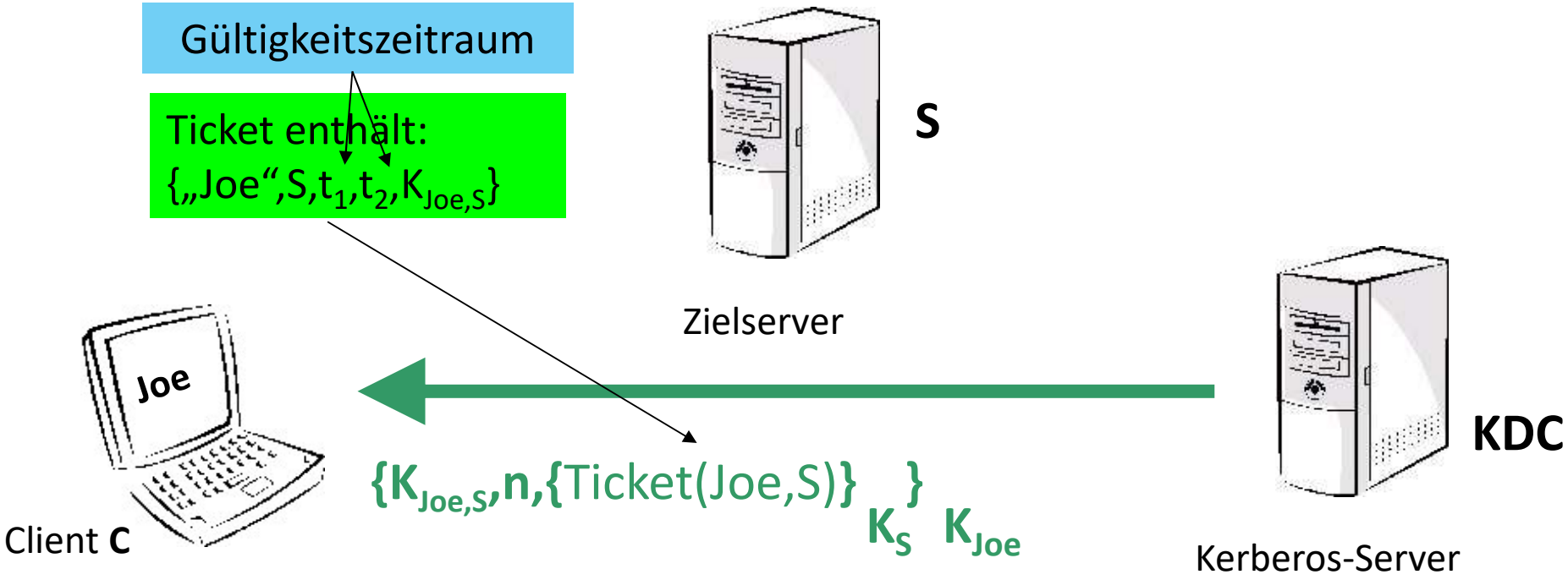
- **Nonce („Einmalstempel“)**
  - Neu generierte Zufallszahl, die einer Nachricht hinzugefügt wird, um ihre Aktualität zu beweisen
    Notation: n
- **Time stamp („Zeitstempel“)**
  - Eine Zahl, die das aktuelle Datum und die genaue Zeit darstellt
  - Notation: t

Kerberos – Grundprinzip (1) [vereinfacht]



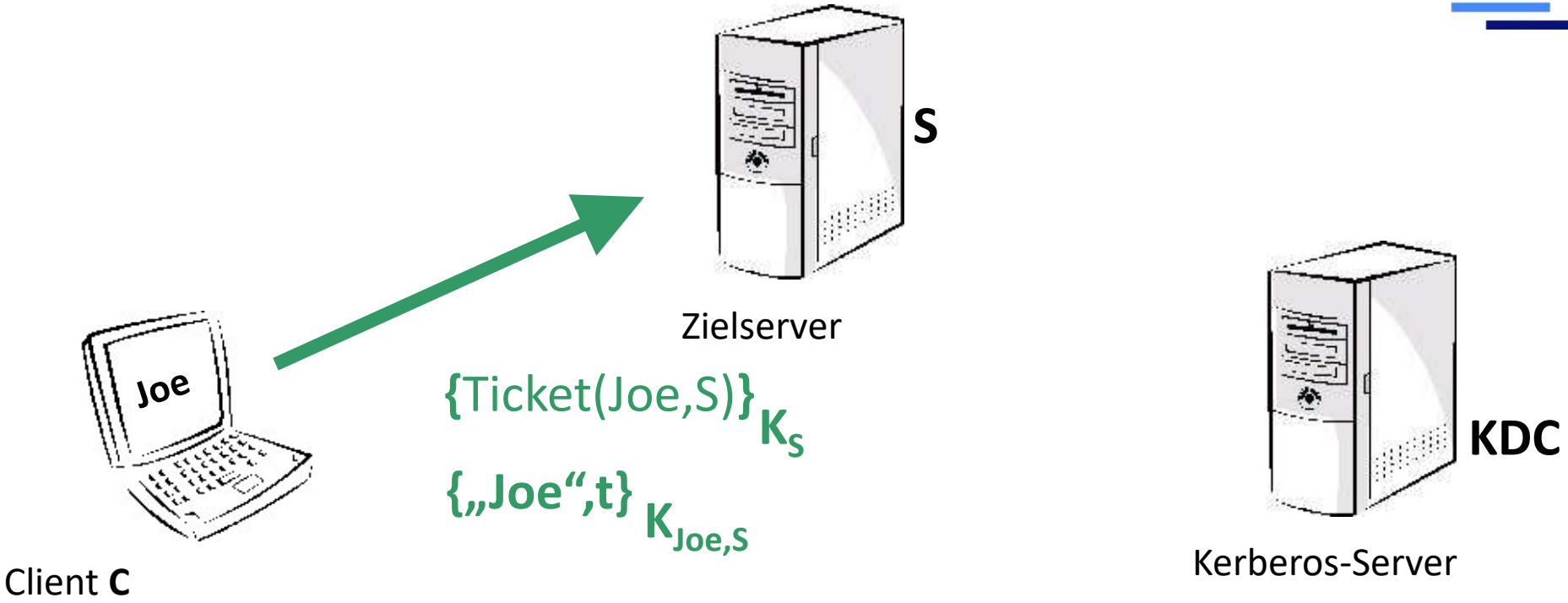
1. Der Client sendet eine Anforderung für die Kommunikation mit dem Zielserver S an den KDC (inkl. Nonce-Wert):
   Benutzerkennung, Zielservername, Nonce

Kerberos – Grundprinzip (2) [vereinfacht]



2. Der KDC gibt eine mit dem geheimen Schlüssel von Joe verschlüsselte Nachricht zurück, die einen neu erzeugten Sitzungsschlüssel $K_{Joe,S}$ für Joe und den Zielserver enthält, ebenso wie ein Ticket, das mit dem geheimen Schlüssel K_S von S verschlüsselt ist.

Kerberos – Grundprinzip (3) [vereinfacht]

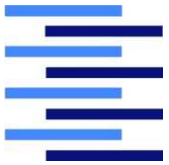


3. Der Client sendet das mit K_S verschlüsselte Ticket mit einer neu erzeugten Authentifizierungsnachricht (Name und Zeitstempel, verschlüsselt mit dem gemeinsamen Sitzungsschlüssel $K_{Joe,S}$) an den Zielserver S

→ Was kann der Server prüfen?

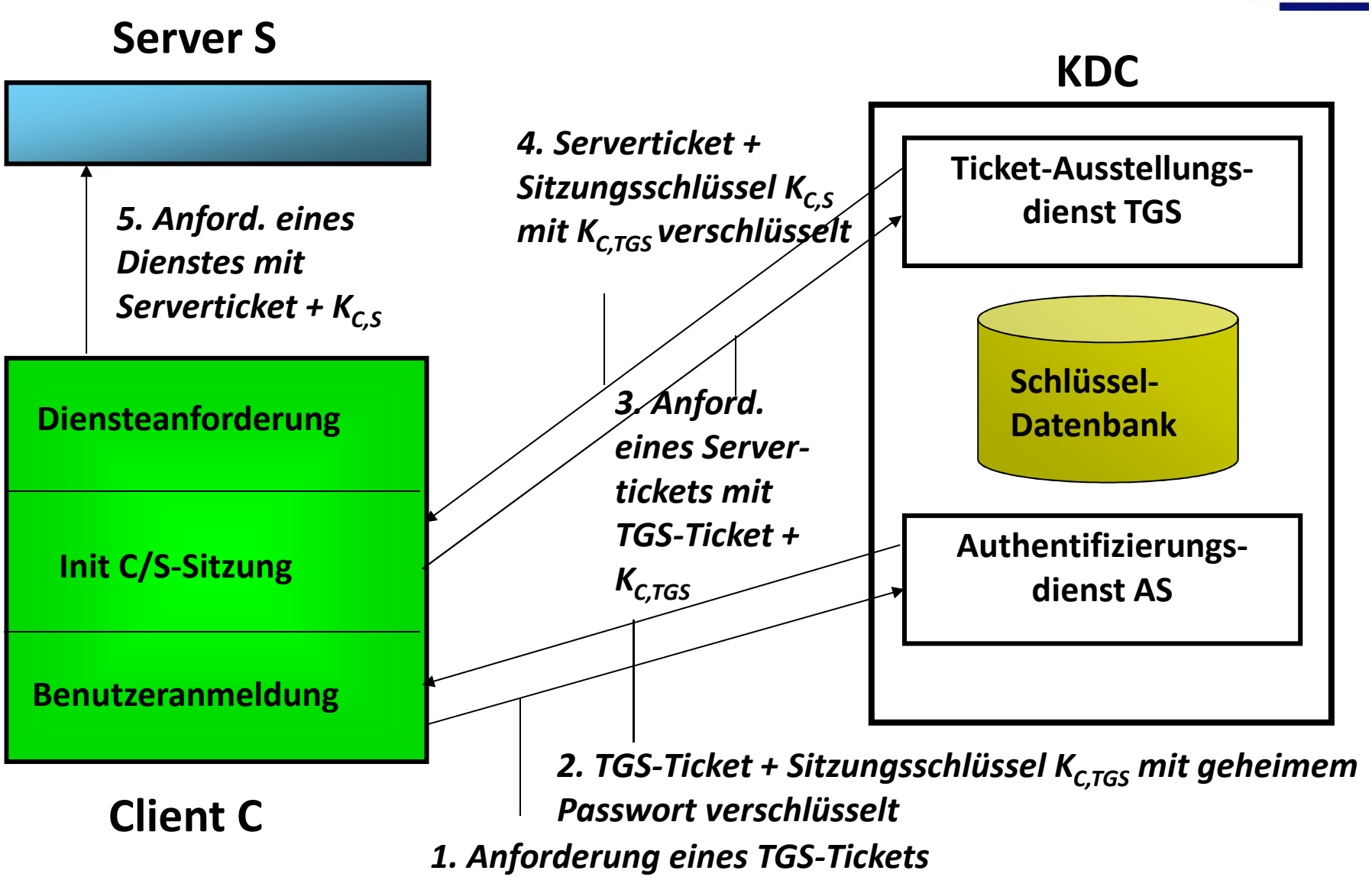
→ Wie kann sich der Server gegenüber dem Client authentifizieren?

Problem der vereinfachten Lösung



- Da für jeden Serverdienst ein eigenes Ticket nötig ist, muss das Benutzer-Passwort (zur Ableitung des Benutzerschlüssels)
  - **vom Benutzer mehrfach eingegeben werden**
    - lästig, nicht zumutbar oder
  - **das Passwort im Speicher des Client gehalten werden**
    - zu gefährlich!!
- **Lösung**: Erweiterung des KDC um einen zusätzlichen **„Ticket Granting Service“ (TGS)** zur Ausstellung von Tickets unabhängig von der Passwort-Authentifizierung
- Statt des Passworts muss nun nur noch das TGS-Ticket und der Sitzungsschlüssel zur Kommunikation mit dem TGS im Client-Speicher gehalten werden → einmaliges Eingeben des Passwortes reicht aus!!
- Außerdem: Durch den TGS wird eine bereichsübergreifende Authentifizierung möglich (kommt später)

Kerberos – Prinzip [komplett]



Kerberos – Protokollbeschreibung

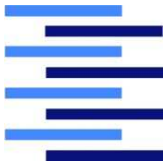
(Version 4)

	Von	An	Nachricht
1	Client C	KDC (AS)	C, TGS, n_1
2	KDC (AS)	Client C	$\{K_{C,TGS}, n_1, \underbrace{\{C,TGS,t_1,t_2,K_{C,TGS}\}}_{\textit{TGS-Ticket}}\}_{K_{TGS}}\}_{K_C}$
3	Client C	KDC (TGS)	$\underbrace{\{C,TGS,t_1,t_2,K_{C,TGS}\}}_{\textit{TGS-Ticket}}\}_{K_{TGS}}, \{C,t\}_{K_{C,TGS}}, S, n_2$
4	KDC (TGS)	Client C	$\{K_{C,S}, n_2, \underbrace{\{C,S,t_1,t_2,K_{C,S}\}}_{\textit{Serverticket}}\}_{K_S}\}_{K_{C,TGS}}$
5	Client C	Applik.-Server S	$\underbrace{\{C,S,t_1,t_2,K_{C,S}\}}_{\textit{Serverticket}}\}_{K_S}, \{C,t\}_{K_{C,S}}$

Bereichsübergreifende Authentifizierung

- Ab gewisser Größe oder Topologie des Netzwerkes: Aufteilung in mehrere Bereiche („Realms“) mit separaten KDC (AS + TGS)
- Ein Client kann von seinem lokalen TGS ein Ticket für einen TGS in einem anderen Realm erhalten
- Jeweils zwei TGS in unterschiedlichen Realms teilen sich einen gemeinsamen Schlüssel (V4)
- Ab V5:
  - Realms sind hierarchisch angeordnet (Baum)
  - Schlüsselaustausch nur zwischen Eltern und Kindern
  - Zugriff auf TGS in einem entfernten Realm erfordert ggf. Durchlauf des Baums (jeweils Ticketanforderung)

Grenzen und Einsatzgebiet von Kerberos



- Alle TGS-Tickets sind mit dem gleichen Schlüssel chiffriert, dem Kerberos Master Key
- Kein Schutz vor Systemsoftwaremodifikationen
- Alles muss „kerberorisiert“ werden
- Kerberos Server muss funktionieren („single point of failure“)

➔ **Einsatz in homogenen Umgebungen**

- **Firmennetz / Campusnetz**
- **im Rahmen eines Verzeichnisdienstes**

Kerberos-Implementierungsbeispiel: ActiveDirectory

- Verzeichnisdienst nach X.500 – Standard
- Bestandteil von Windows 2000 / 2003 / 2008 / 2012 / 2016 Server
- **Verwendet ausschließlich Kerberos V5 zur Authentifikation**
- *Kerberos-Erweiterung u.a.: Nutzung einer PKI-Infrastruktur*

ActiveDirectory – Eigenschaften (1)

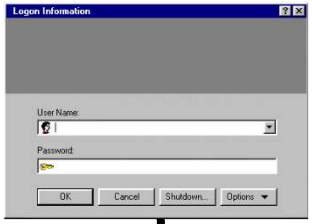
- Hierarchische Struktur von Objekten:
  - Forest
    - Domain (ggf. verbunden durch Vertrauensbeziehungen)
      ~ Realms mit gemeinsamem TGS-Schlüssel
      - Organizational Unit (OU)
        - Benutzer, Computer, Drucker, …
- Domains und OUs können eigene Teilbäume bilden (mehrere Ebenen)
- Innerhalb eines „Forest“ ..
  - Einheitliches Datenbankschema
  - Austausch von bestimmten Informationen zwischen sich vertrauenden Domains („global catalog“)
- Verwendet DNS auch zur Namensbildung
  - Bsp: „Huebner.Informatik.HAW-Hamburg.de“

Active Directory - Eigenschaften (2)

- Zugriff / Abfragen über **LDAP** oder ADSI möglich
- Implementierung als verteilte, objekt-orientierte Datenbank
  - Physikalisch verteilte ActiveDirectory - Server („Domain Controller“)
    - Domain Controller enthält Kerberos - KDC
    - mehrere Domain Controller innerhalb einer Domain möglich
      → Replikation derselben Datenbank

Smartcard-Anmeldung am ActiveDirectory

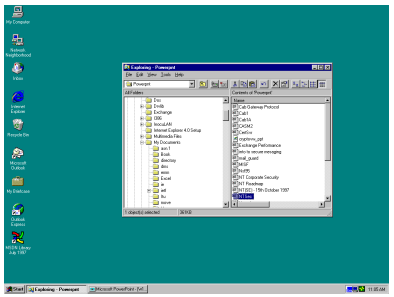
1 Beim Einführen der Karte zeigt Winlogon GINA an



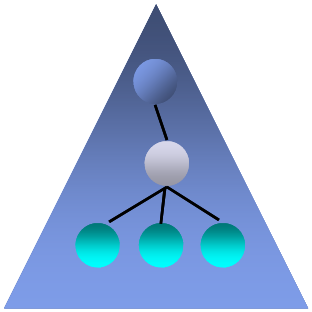
2 Der Benutzer gibt die PIN ein

3 GINA gibt die PIN an LSA weiter

4 LSA greift auf die Smartcard zu und ruft das Zertifikat ab



8 Die Smartcart entschlüsselt das TGT mit seinem privaten Schlüssel, damit LSA eine Benutzer-anmeldung zulässt



6 Der KDC überprüft das Zertifikat und sucht dann den Benutzer im Verzeichnis-dienst

5 Kerberos sendet das Zertifikat in einer PKINIT-Anmelde-anforderung an den KDC

7 Der KDC gibt das TGT mit einem Sitzungsschlüssel, der wiederum mit dem öffentlichen Schlüssel des Benutzers verschlüsselt wurde, verschlüsselt zurück

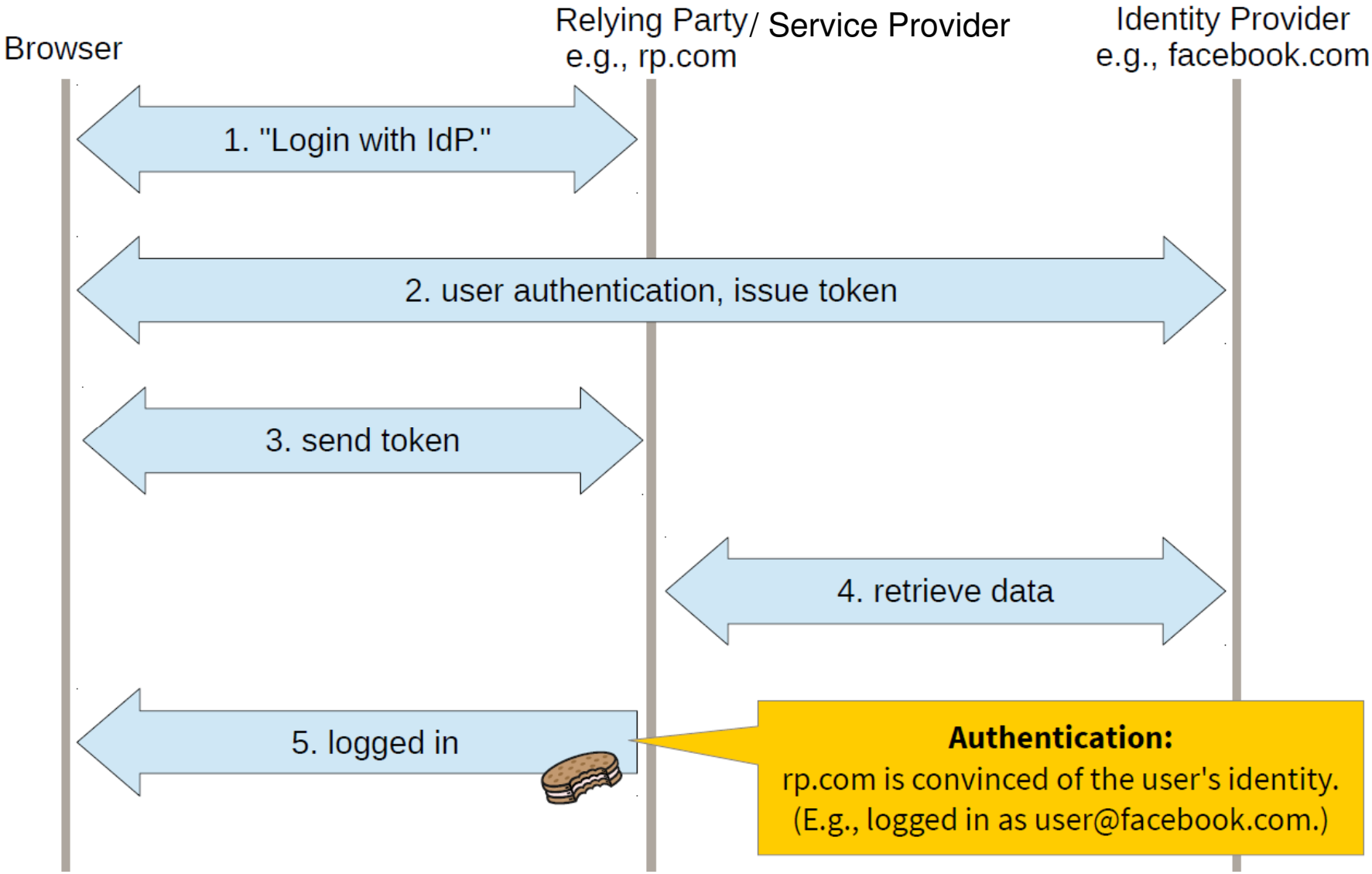
LSA Kerberos

Kerberos KDC

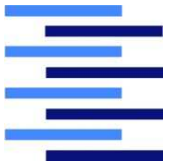
Kapitel 5
Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
   1. Einführung
   2. Kerberos
   3. Web-Authentifikationsdienste
3. Modelle für die Zugriffskontrolle (Autorisation)
4. Autorisation in UNIX / Windows

Web-Authentifikation: Prinzip



Übersicht Web-Authentifikationsdienste



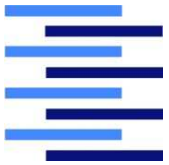
- Standard-Spezifikationen (https://www.oasis-open.org/standards)
  - Web-Service Sicherheitsstandards
  - Security Assertion Markup Language (SAML) 2.0
- Protokolle / Implementierungen *(Auswahl)*
  - Shibboleth: Basis SAML
  - Central Authentication Service (CAS): Basis SAML
  - OpenID Connect: Sichere Authentifikation für OAUTH 2.0 (Basis: JSON Web Token *JWT*)
  - ...

SAML (Security Assertion Markup Language)

- XML-basierter Standard zum Austausch von Authentifikations- und Autorisierungsinformationen
- Format: **Assertion** *(„Ticket“), Beispiel:*

```
<saml:Assertion
        MajorVersion= 1 MinorVersion= 0
        AssertionID=128.9.167.32.12345678
        Issuer=Smith Corporation
        IssueInstant=2016-12-03T10:02:00Z />
        <saml:Conditions
                NotBefore=2016-12-03T10:00:00Z
                NotAfter=2016-12-03T10:05:00Z />
        <saml:AuthenticationStatement
                AuthenticationMethod=password
                Authenticationinstant=2016-12-03T10:02:00Z>
                <saml:Subject>
                        <saml:NameIdentifier
                                SecurityDomain=smithco.com
                                Name=joeuser />
                </saml:Subject>
        </saml:AuthenticationStatement>
</saml:Assertion>
```

SAML (Security Assertion Markup Language)

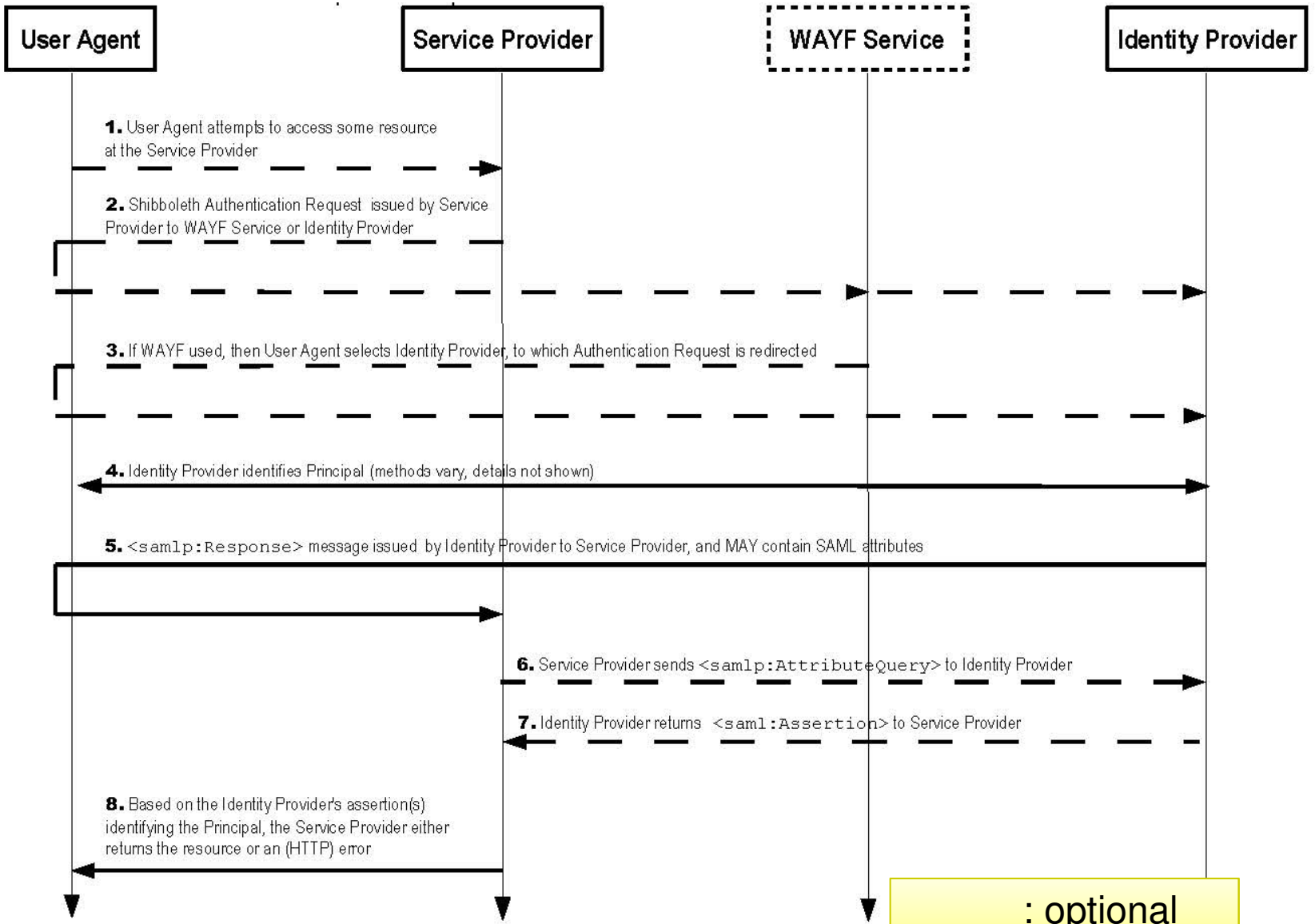


- Generische Protokollspezifikationen:
  - SAML Authority: Stellt Assertions aus
    (analog zu einer „CA“, „Identity Provider“)
  - Relying Party: vertraut den Assertions
- SAML-Sicherheit:
  - Keine Sicherheitskonzepte in der SAML-Spezifikation enthalten *(Schutz vor Sniffing, Spoofing, ...)!*
  - Empfohlen:
    - Verschlüsselung / Signierung der Assertions über *XML-Encryption / XML-Signature* – Standard
    - Transportsicherung über *SSL/TLS*

Shibboleth - Anwendung

- ***Authentifizierung:*** Ein Benutzer (UserAgent) will über das Web auf eine geschützte Ressource zugreifen. Der ServiceProvider nimmt die Anfrage entgegen und prüft, ob der Benutzer bereits authentifiziert ist. Wenn nicht, wird er evtl. zu einem Lokalisierungsdienst *(„Where Are You From?“ WAYF)* weitergeleitet. Der Lokalisierungsdienst bietet eine Auswahl von IdentityProvidern an. Der Benutzer wählt seinen IdentityProvider aus und wird zu diesem weitergeleitet. Der IdentityProvider prüft, ob der Benutzer bereits authentifiziert ist. Ist dies nicht der Fall, wird der Benutzer aufgefordert, dies zu tun (zum Beispiel mit Benutzerkennung und Passwort oder Chipkarte). Der IdentityProvider stellt einen "digitalen Ausweis" ***(signierte SAML-Assertion)*** aus und leitet den Benutzer zum ServiceProvider zurück. Der ServiceProvider prüft den Inhalt des digitalen Ausweises.
- ***Autorisierung:*** Benötigt der ServiceProvider weitere Informationen über den Benutzer, so fragt er direkt beim IdentityProvider des Benutzers nach. Der ServiceProvider prüft über das eigene System, ob der Benutzer auf die Ressource zugreifen darf, und gestattet den Zugriff oder lehnt ihn ab.

Shibboleth - Anwendung



Shibboleth - Technik

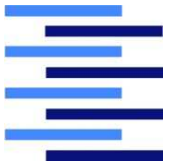
- SAML
- XML-Encryption / XML-Signature
- http / SOAP (jeweils über TLS)

OpenSource-Implementierung verfügbar mit folgenden Beschränkungen:

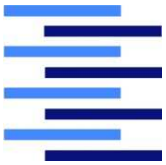
- The Shibboleth Service Provider is a web server module. The supported web servers are Apache's HTTPD, Microsoft's IIS, and Sun's Java System Web Server
- The Shibboleth Identity Provider is a Java web application and must be deployed in a standard web application container like Apache's Tomcat, Mortbay's Jetty, or JBoss's JBoss Application Server.

siehe https://shibboleth.net und https://wiki.shibboleth.net

Shibboleth-Föderation von Identity Providern



- Voraussetzungen:
  - Hierarchie von Identity Providern → Assertions wird gegenseitig vertraut *(Zertifikate)*
  - Einheitliches Attributschema
  - Einheitliche Policies *(u.a. zur lokalen Authentifikation)*
- Vorteile:
  - Nur *ein* zentraler Vertragspartner für Service Provider nötig („Root“-Identity Provider)
  - Betrieb eines zentralen Lokalisierungsdienstes möglich
- Im Hochschulbereich verbreitet (siehe https://www.aai.dfn.de)



Kapitel 5
Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
3. Modelle für die Zugriffskontrolle (Autorisation)
4. Autorisation in UNIX / Windows

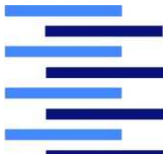
Zugriffskontrolle: Sicherheitsmodelle

Ziel: Nur autorisierte Zugriffe auf ein Objekt zulassen!

- Discretionary Access Control [DAC] (Zugriffsmatrix)
  - Access Control Lists
  - Capabilities
- Mandatory Access Control [MAC] (Multilevel Security)
  - Bell-La Padula
  - Biba
- Role Based Access Control [RBAC]

In der Praxis: Kombination der Modelle ist möglich!

Discretionary Access Control (DAC)



- Idee:
  - Zugriffskontrolle liegt in der Verantwortung der Subjekte
  - Ein **Subjekt** (Benutzer) kann „Eigentümer“ (owner) eines **Objekts** (z.B. einer Datei) sein
  - Der Eigentümer kann anderen Subjekten **Zugriffsrechte** (z.B. read, write, append, execute, search, ..) auf sein Objekt geben
- Modellierung der Zugriffsrechte:
  - Explizite Speicherung der Zugriffsrechte für alle Subjekte und Objekte in einer **Zugriffsmatrix**
  - Zugriff wird (vom „Sicherheitsmonitor“) nur gewährt, wenn in der Zugriffsmatrix die entsprechenden Rechte eingetragen sind

Beispiel: Zugriffsmatrix

Subjekte S	Objekte O				
	Datei 1	**Datei 2**	**…**	**…**	
Fritz	owner,r,w,x	-			
Karl	r,x	w			
Anne	-	owner,r,w			

Subjekt-Sicht

Objekt-Sicht

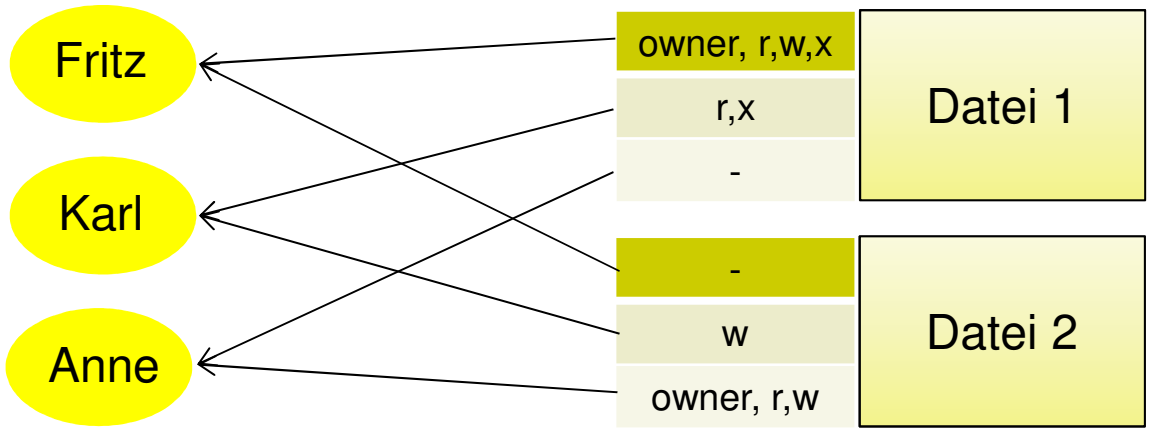
r: Leseberechtigung („**r**ead“)
w: Schreibberechtigung („**w**rite“)
x: Berechtigung, Programme auszuführen („e**x**ecute“)

nach [Eckert]

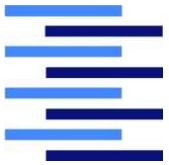
Implementierung einer Zugriffsmatrix durch Zugriffskontrolllisten

- Gebräuchliche Bezeichnung: **ACL** („Access Control List“)
- Objekt-Sicht: Spaltenweise Speicherung der Zugriffsmatrix
- Die ACL eines Objekts (z.B. einer Datei) speichert für jedes Subjekt, welche Zugriffsrechte existieren
- ACL = Liste von Paaren (Subjekt-Id, Rechte)

ACE (Access Control Element)

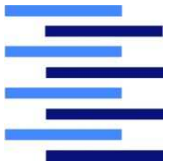


Vor- und Nachteile von Zugriffskontrolllisten (ACLs)

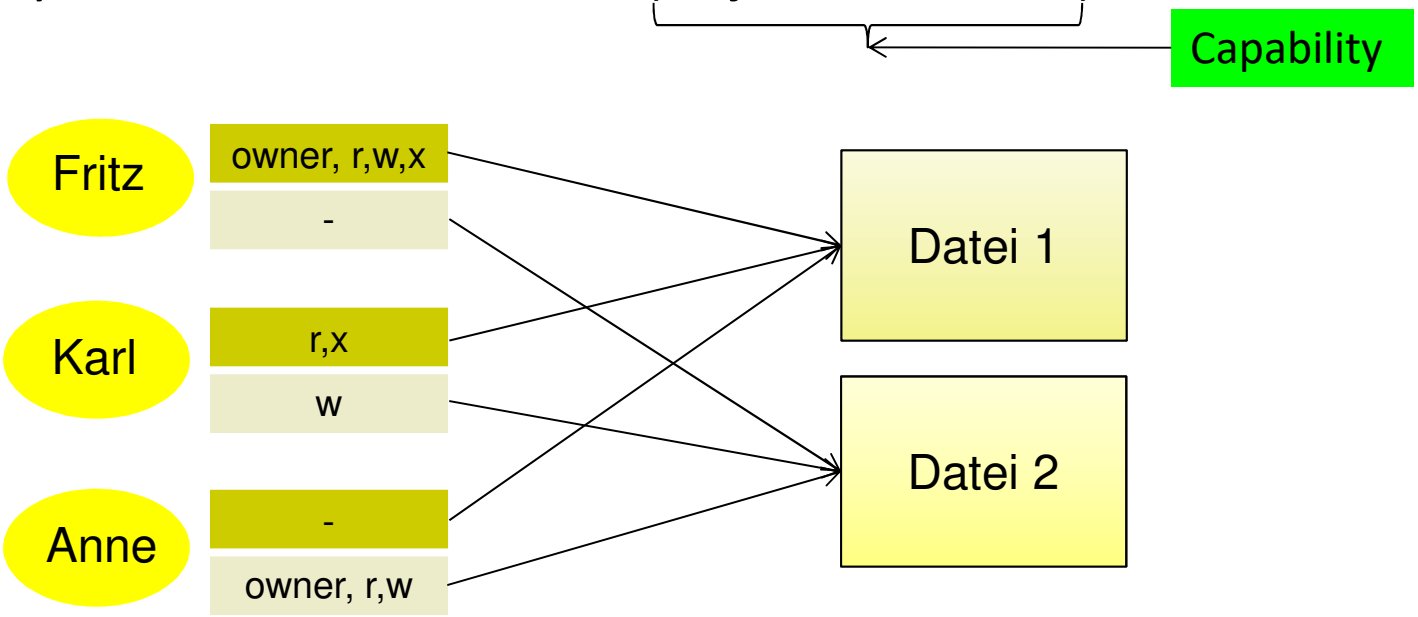


- Vorteile:
  - Einfache Verwaltung
  - Inkonsistenzen durch das Löschen von Objekten sind nicht möglich
  - Bei Verwendung von Rollen (Gruppen) statt Subjekten (Benutzern) effiziente Implementierung eines einfachen **RBAC-Modells** möglich
- Nachteile:
  - Aufwändige Rechteüberprüfung bei langen ACLs
  - Keine dynamische Weitergabe von Rechten möglich

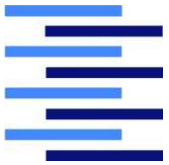
Implementierung einer Zugriffsmatrix durch Zugriffsausweise



- Gebräuchliche Bezeichnung: **Capability *(Token)***
- Subjekt-Sicht: Zeilenweise Speicherung der Zugriffsmatrix
- Die Capability-Liste eines Subjekts (z.B. eines Prozesses) speichert für jedes Objekt, welche Zugriffsrechte existieren
- Capability-Liste = Liste von Paaren (Objekt-Id, Rechte)

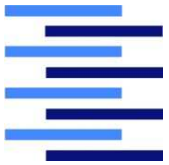


Vor- und Nachteile von Zugriffsausweisen (Capabilities)



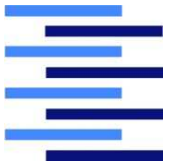
- Vorteile:
  - Effiziente Rechteüberprüfung zur Laufzeit
  - Dynamische Weitergabe von Rechten möglich
    ➔ effiziente Implementierung von **Schutzdomänen**
- Nachteile:
  - Komplexe Verwaltung
  - Inkonsistenzen durch das Löschen von Objekten müssen explizit verhindert werden

Kombination von Zugriffskontrolllisten und Zugriffsausweisen



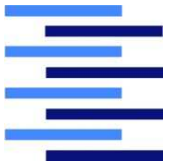
- Erstmaliger Zugriff auf ein Objekt (z.B. Öffnen einer Datei):
  - Überprüfung der Berechtigung anhand der Zugriffskontrollliste (ACL) des Objekts durch vertrauenswürdigen Berechtigungskontrolleur
  - Ausstellung eines Zugriffsausweises (Token) für das Subjekt bzgl. des gewünschten Objekts
- Weitere Zugriffe auf das Objekt
  - Überprüfung des Zugriffsausweises (ggf. durch dezentrale Kontrolleure) genügt

Mandatory Access Control (Multilevel-Security)



- Idee:
  - Zugriffskontrolle wird durch systembestimmte Regeln festgelegt
  - Definition von **Sicherheitsstufen** („Security-Level“) *(z.B. Top Secret > Secret > Confidential > Unclassified)*
  - Sicherheitsstufe eines Objekts („Classification“): L(O)
  - Sicherheitsstufe eines Subjekts („Clearance“): L(S)
- Modellierung der Zugriffsrechte:
  - Festlegung von Regeln, welche Zugriffsrechte Subjekte einer bestimmten Sicherheitsstufe auf Objekte haben

Bell-La Padula - Modell



- Anwendungsziel: Geheimhaltung! *(z.B. beim Militär)*
- *Grundhaltung: Hoher Level = hohe Vertraulichkeit*
- Zugriffsregeln:

„No-read-up“-Regel:

Ein Subjekt S darf ein Objekt O nur **lesen**, wenn gilt:

$L(O) \leq L(S)$

„No-write-down“-Regel:

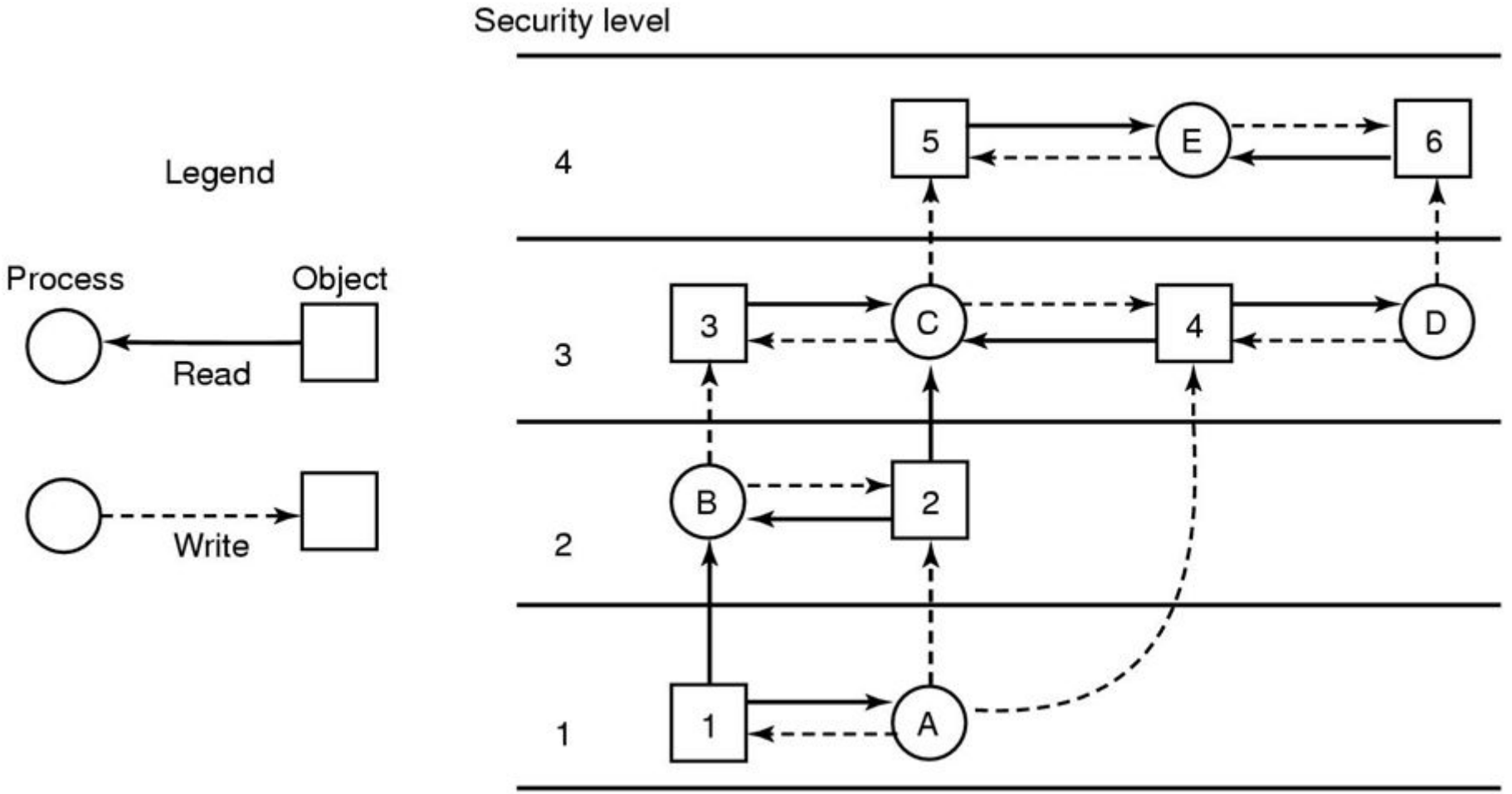
Ein Subjekt S darf ein Objekt O nur **schreiben**, wenn gilt:

$L(S) \leq L(O)$

„Von unten lesen, nach oben schreiben (berichten)!“

*→ Informationen fließen nach **oben**!*

Bell-La Padula – Modell: Informationsflüsse



Informationsflüsse können nur nach oben gehen!

[Tanenbaum]

Biba- Modell

- Anwendungsziel: Datenintegrität! *(z.B. in Unternehmen)*
- *Grundhaltung:*
  Hoher Level = hohe Integrität (Verlässlichkeit, Bedeutung)
- Zugriffsregeln:

„No-read-down“-Regel:

Ein Subjekt S darf ein Objekt O nur **lesen**, wenn gilt:

$L(S) \leq L(O)$

„No-write-up“-Regel:

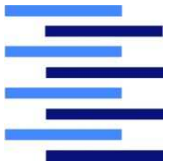
Ein Subjekt S darf ein Objekt O nur **schreiben**, wenn gilt:

$L(O) \leq L(S)$

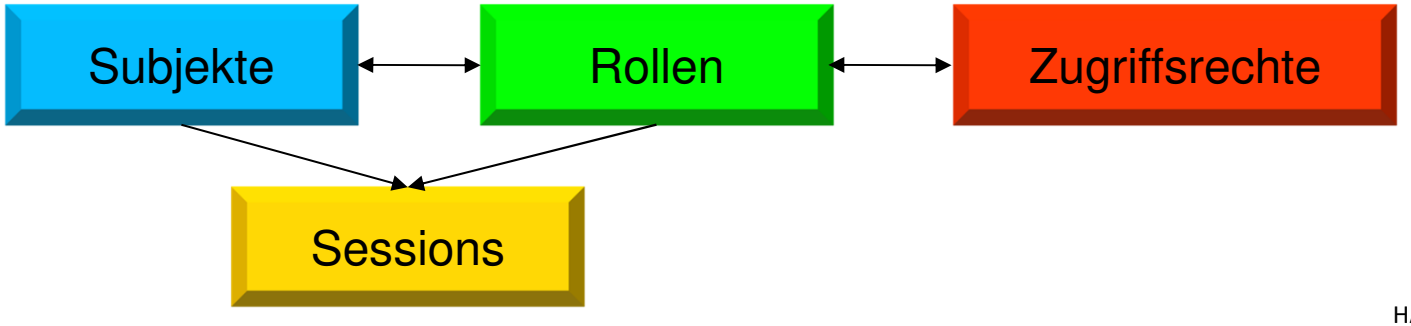
„Von oben lesen, nach unten schreiben!“

*→ Informationen fließen nach **unten**!*

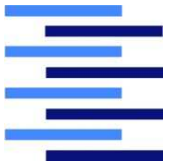
Role Based Access Control (RBAC)



- Idee:
  - Systembestimmte Zugriffskontrolle durch Aufgaben-orientierte Rechtevergabe
  - Eine **Rolle** beschreibt eine bestimmte Aufgabe mit damit verbundenen Verantwortlichkeiten und Berechtigungen
- Modellierung der Zugriffsrechte:
  - Zuordnung von Subjekten (Benutzern) zu Rollen (n:m)
  - Zuordnung der Rollen zu Zugriffsrechten
  - Ein Subjekt wählt für eine **Session** die angemessene Rolle aus



RBAC-Beispiel: Bank



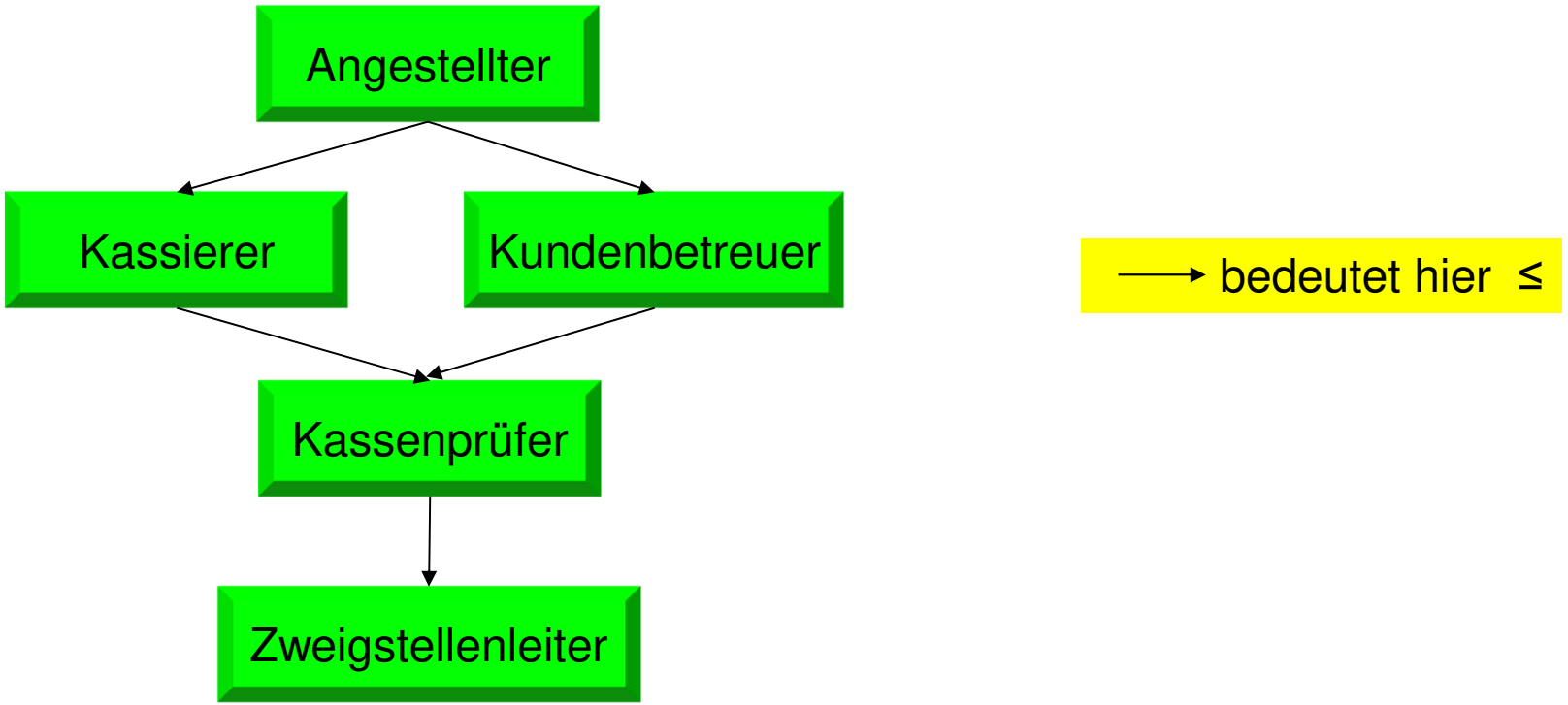
- Subjekte:
  - Herr Müller, Frau Schmidt, …
- Rollen:
  - {Angestellter, Kassierer, Kundenbetreuer, Kassenprüfer, Zweigstellenleiter}
- Zugriffsrechte:
  - {Konto_sperren, Kreditrahmen_erhöhen, Einzahlung_auf_Kundenkonto, Abheben_von_Kundenkonto, …}
- Rollenzuordnung:
  - Müller → {Zweigstellenleiter, Kassierer}
  - Schmidt → {Kassierer}
- Zugriffsrechtezuordnung:
  - Zweigstellenleiter → {Konto_sperren, Kreditrahmen_erhöhen}
  - Kassierer → {Einzahlung_auf_Kundenkonto, Abheben_von_Kundenkonto}
- Welche Sessions sind möglich?

RBAC-Erweiterung: Rollenhierachien

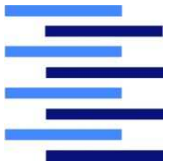
Hierarchische Anordnung der Rollen (partielle Ordnung):

$R_1 \leq R_2 \Rightarrow R_2$ besitzt mindestens alle Zugriffsrechte von R_1

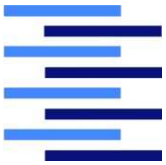
→ *Rechtevererbung wird möglich!*



RBAC-Erweiterung: Aufgabentrennung



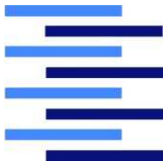
- Statische Aufgabentrennung
  (Regelung der Zuordnung Subjekt ↔ Rolle)
  - Ein Subjekt darf nicht Mitglied von R_1 und R_2 sein!
  - Beispiel:
    - R_1 = Kassierer
    - R_2 = Kassenprüfer
- Dynamische Aufgabentrennung
  (Regelung der Sessionanmeldung)
  - Ein Subjekt S darf nicht gleichzeitig in zwei Sessions (S, R_1) und (S, R_2) angemeldet sein
  - Beispiel:
    - S = Herr Meier → {Kundenberater, Kunde}



Kapitel 5
Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
3. Modelle für die Zugriffskontrolle (Autorisation)
4. Autorisation in UNIX / Windows

Zugriffskontrolle in UNIX



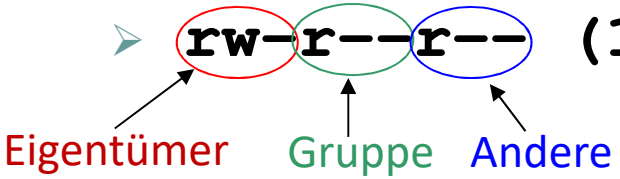
Basis: Discretionary Access Control

- Subjekte
  - Benutzeridentifikation: UID (16 Bit-Integer)
  - Gruppe(n) des Benutzers: GID (16 Bit-Integer)
- Objekte
  - Dateien
  - Verzeichnisse
- Zugriffsrechte
  - r: Lesen
  - w: Schreiben
  - x: Ausführen (Datei) / Durchsuchen (Verzeichnis)

Implementierung eines einfachen RBAC-Modells möglich!

UNIX: Zugriffsrechte auf Objekte

- Unterscheidung nach Subjekten
  - Rechte des Eigentümers („owner“)
  - Rechte aller Mitglieder der Gruppe(n) des Eigentümers
  - Rechte aller anderen Subjekte (Benutzer)
- Darstellung durch 3 * 3 = 9 Angaben (Flags)
- Flag gesetzt = Zugriff erlaubt
- Beispiele:
  - **rwxrwxrwx (111111111)** Alle dürfen alles
  - **rw-------- (110000000)** Nur der Eigentümer darf lesen und schreiben
  - **rw-r--r--** **(110100100)** **?**

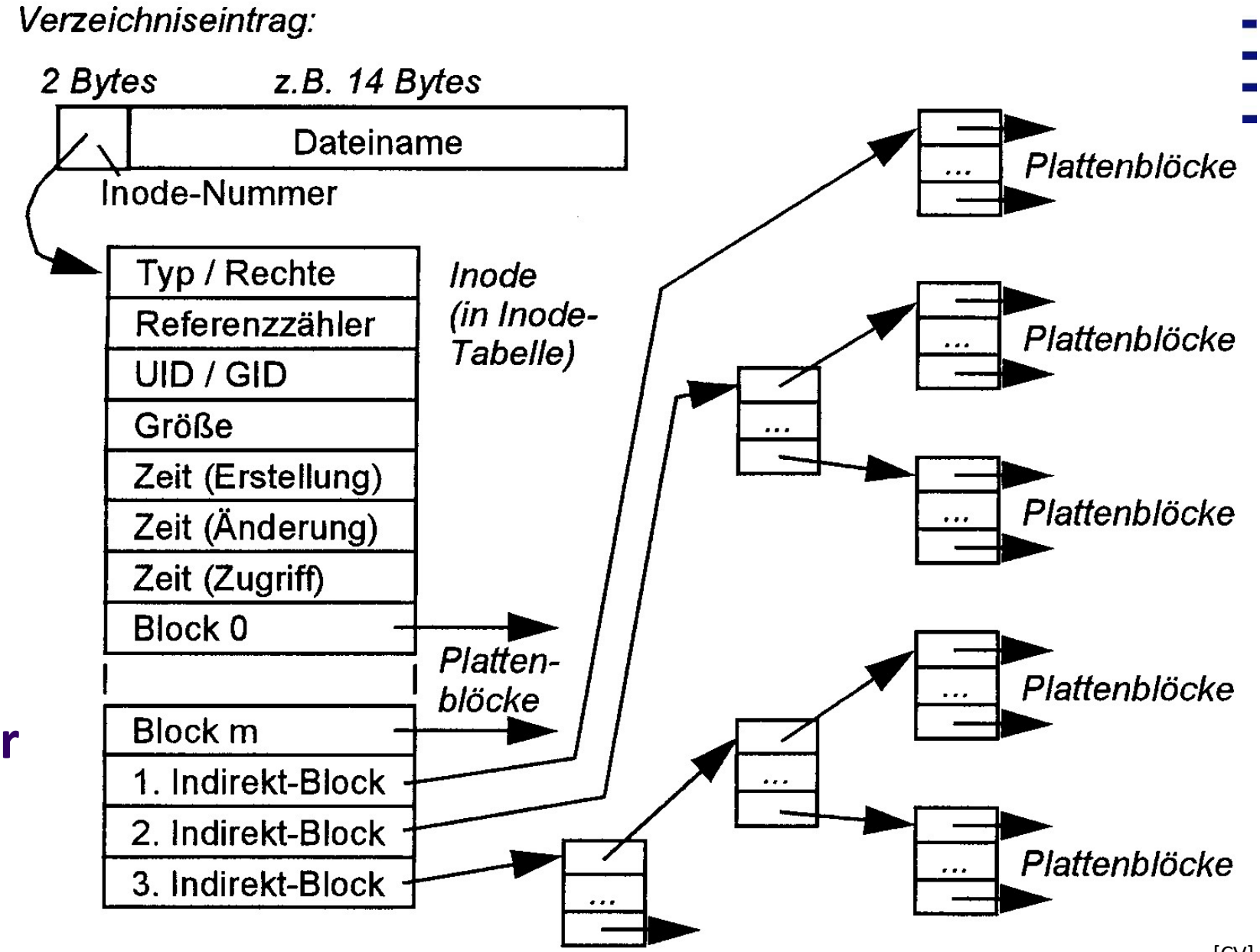


Eine einfache ACL!

UNIX: Zuweisung von Zugriffsrechten

- Jeder Prozess hat als Default die UID und GID seines Eigentümers (derjenige, der ihn gestartet hat)
- Wenn das SETUID-Bit in der Programmdatei des Prozesses gesetzt ist, so wird als Prozess-UID diejenige des Eigentümers der Programmdatei eingetragen *(Bsp.: Druck-Spooler-Aufruf!)*
- Jeder neuen Datei wird die UID und GID des erstellenden Prozesses als Eigentümer zugewiesen
- Der Superuser („root“ - UID 0) darf auf alle Dateien zugreifen und z.T. geschützte Systemaufrufe ausführen!

UNIX: I-Node-Struktur

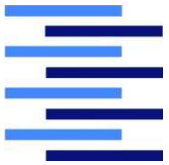


[CV]

UNIX: Zugriff auf Dateien

- Datei öffnen (**`open`**):
  - Das System vergleicht die im I-Node gespeicherten Schutzrechte mit der UID/GID des Prozesses (im PCB)
  - Falls Zugriff nicht erlaubt → Fehler!
  - Zugriff erlaubt: Rückgabe eines File-Descriptors *(Capability!)*
- Zugriff auf eine geöffnete Datei (**`read, write`**):
  - Anhand des File-Descriptors ohne weitere Überprüfung
  - Zwischenzeitlich vorgenommene Änderungen der Zugriffsrechte sind folglich bis zum Schließen der Datei (**`close`**) unwirksam!

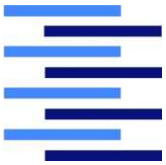
Zugriffskontrolle in Windows



Basis: Discretionary Access Control

- Subjekte
  - Benutzer und Gruppen
  - n:m – Beziehungen sind möglich
  - Identifikation über „Security Identifier“ (SID)
- Objekte
  - Dateien, Geräte, Jobs, Prozesse, Threads, Events, Mutexe, Semaphore, Shared Memory, Netzwerkfreigaben, Registrierungsschlüssel …
- Zugriffsrechte (zugelassen / verweigert)
  - Vollzugriff
  - Lesen
  - Schreiben
  - Ausführen
  - 10 weitere Kombinationen
  - spezielle Rechte (ca. 700 Richtlinienobjekte)

Security Identifier (SID)



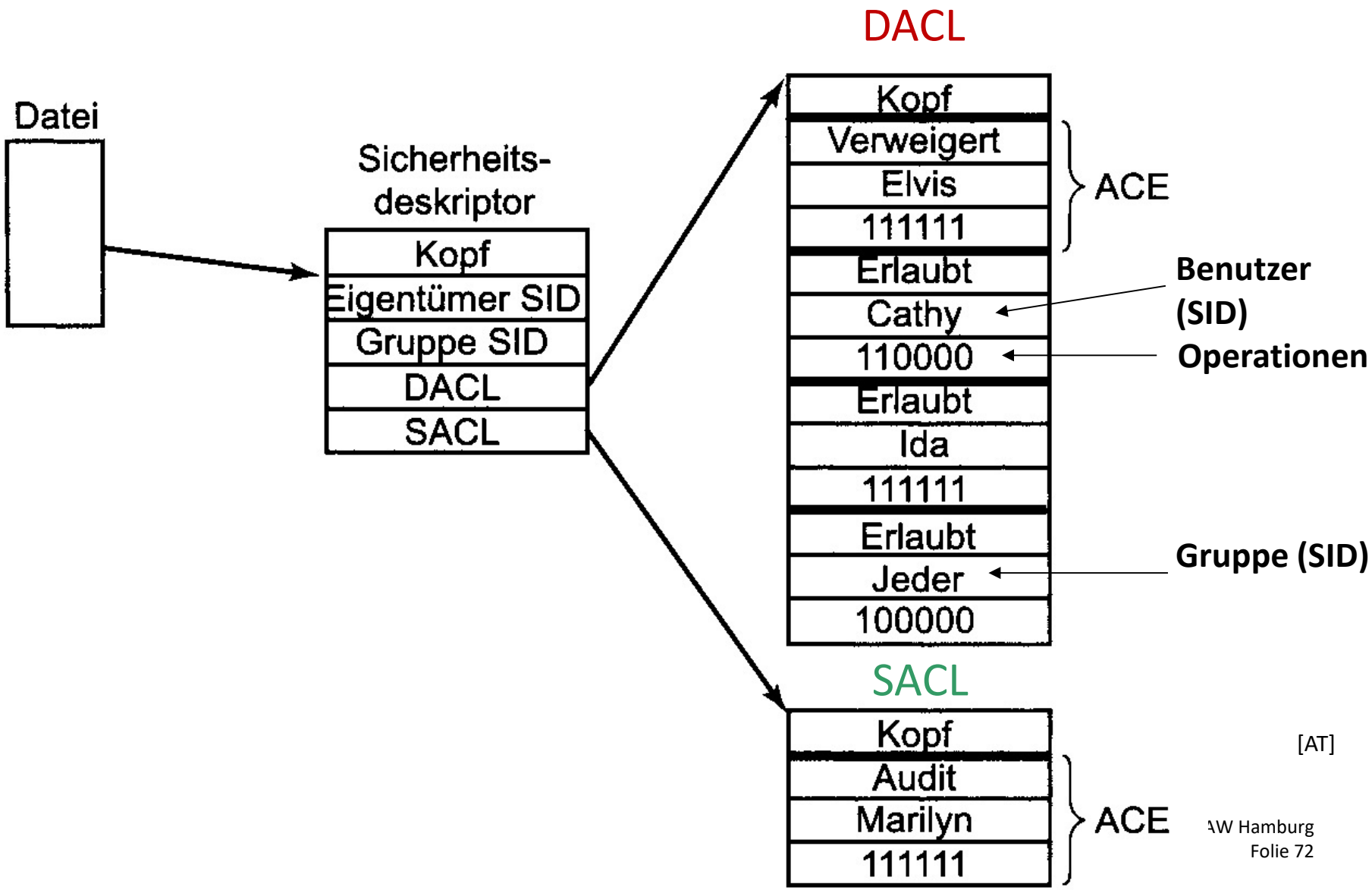
- SID-Struktur:
  - S-<Versionsnr.>-<ausstellende Autorität>-<untergeordnete Autoritäten>* -RID
- Beispiel:
  - S-1-5-21-2232374393-3596432456-3763152189-1005
    - Ausstellende Autorität: Zuständiges Sicherheitssystem (lokaler Rechner oder Domain Controller)
    - Untergeordnete Autoritäten: Dezentrale Einheiten
    - RID: Relative ID = Zufallszahl

➔ Weltweit eindeutig *(wahrscheinlich)*!

Sicherheitsdeskriptor

- Speichert die Zugriffsrechte auf ein (beliebiges) Objekt
- Struktur:
  - Versionsnummer
  - verschiedene Flags
  - Besitzer-SID
  - SID der Primärgruppe (nur von POSIX verwendet)
  - Discretionary ACL (**DACL**): Liste von ACEs (Access Control Elements) zur Zugriffskontrolle
  - System ACL (**SACL**): Liste von ACEs zur Steuerung der Ereignisprotokollierung im Systemprotokoll

Beispiel eines Sicherheitsdeskriptors



[AT]

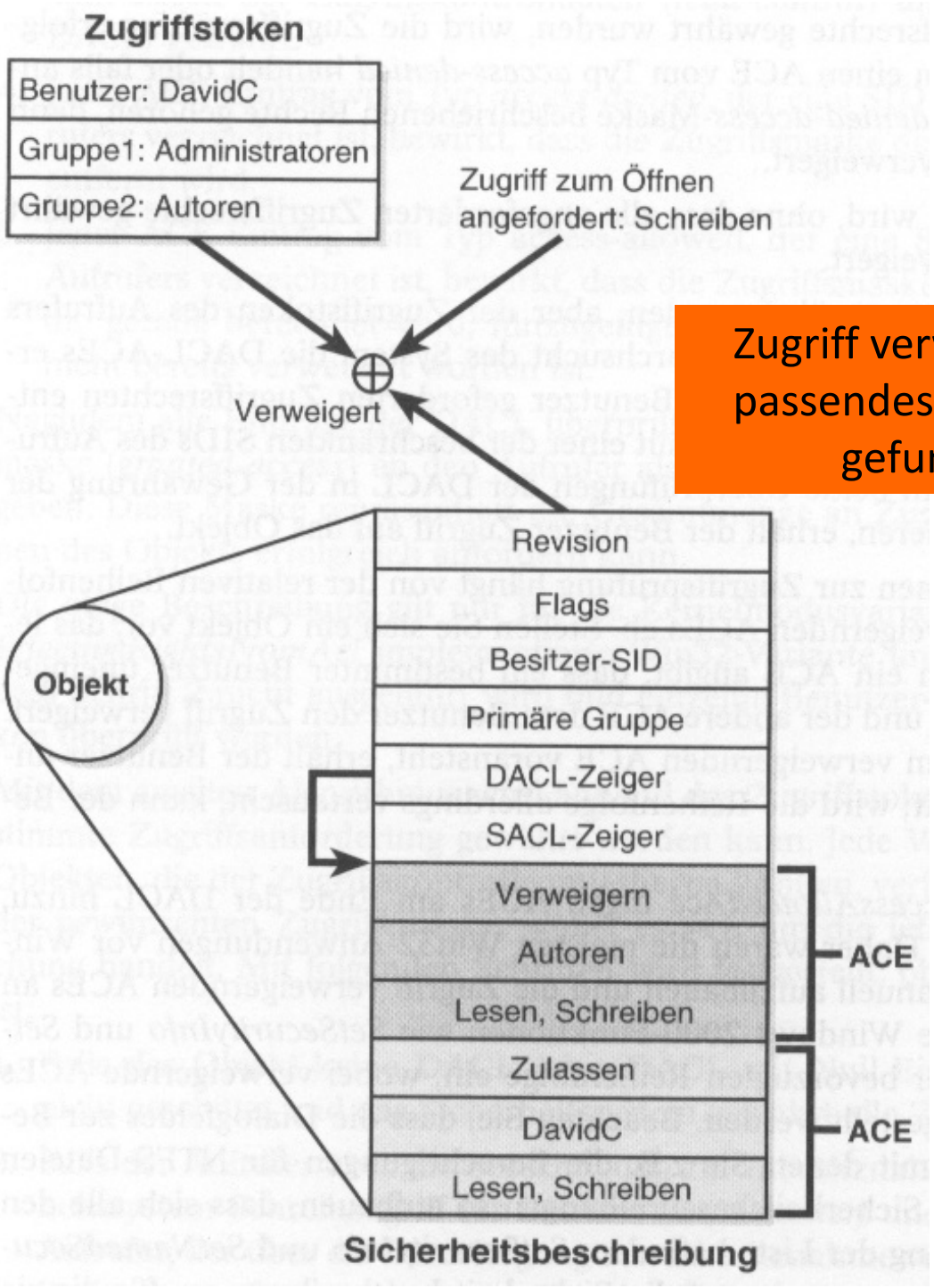
Access Token

- Speichert die SIDs und die Rechte eines Prozesses (→ Capability)
- Kann weitergegeben / vererbt werden (z.B. an Threads)
- Standard-DACL: Default- ACL für erzeugte Objekte
- (Sonder-)Rechte: Aufteilung der Superuser-Rechte auf verschiedene Benutzer („Rollen“) ist möglich!
  Beispiele:
  Herunterfahren des Rechners, Zugriff auf bestimmte Systemdateien

Struktur eines Access Token:

Kopf (IDs, ...)
Gültigkeitsdauer
Standardprimärgruppe
Standard-DACL
Benutzer-SID
SID von Gruppe 1
.... SID von Gruppe n
....
Recht 1
... Recht m

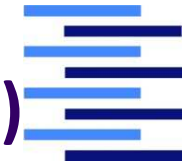
Beispiel: Zugriff eines Prozesses auf ein Datei-Objekt unter Windows



Zugriff verweigert, da passendes ACE zuerst gefunden!

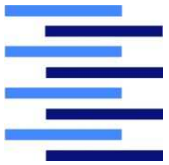
[DS/MR]

Encrypting File System (EFS): Dateiverschlüsselung (1)



- Vorbereitung:
  - Erzeugung eines RSA-Schlüsselpaares für einen Benutzer bei erstmaliger EFS-Nutzung
  - Privater RSA-Schlüssel wird (mit einem aus dem Benutzer-Passwort abgeleiteten Schlüssel verschlüsselt) auf der Festplatte gespeichert
- Verschlüsselung einer Datei:
  - Erzeugung eines neuen symmetrischen Schlüssels (3DES / AES), Verschlüsselung und Speicherung der verschlüsselten Dateiinhalte
  - Verschlüsselung des symmetrischen Schlüssels mit den öffentlichen RSA-Schlüsseln des Datei-Owners sowie sämtlicher berechtigter Benutzer
  - Speicherung der öffentlichen RSA-Schlüssel und der verschlüsselten Dateischlüssel über zusätzliche NTFS-Attribute im Dateieintrag der MFT

Encrypting File System (EFS): Dateiverschlüsselung (2)



- Login:
  - Entschlüsselung des privaten RSA-Schlüssels und Speicherung im virtuellen Adressraum des Benutzerprozesses (Löschen beim Logout)
- Entschlüsselung einer Datei:
  - Entschlüsseln des symmetrischen Schlüssels durch den privaten RSA-Schlüssel und Entschlüsselung des Dateiinhalts
- Recovery-Mechanismen:
  - Export des privaten RSA-Schlüssels ist möglich (PKCS#12-Format)
  - Zusätzliche Verwendung des öffentlichen RSA-Schlüssels eines „Recovery-Agenten“

Ende des 5. Kapitels:
Was haben wir geschafft?

Authentifikation und Autorisation

1. Benutzerauthentifikation (Zugangskontrolle)
2. Authentifikation in verteilten Systemen
3. Zugriffskontrolle (Autorisation)
4. Zugriffskontrolle in UNIX / Windows