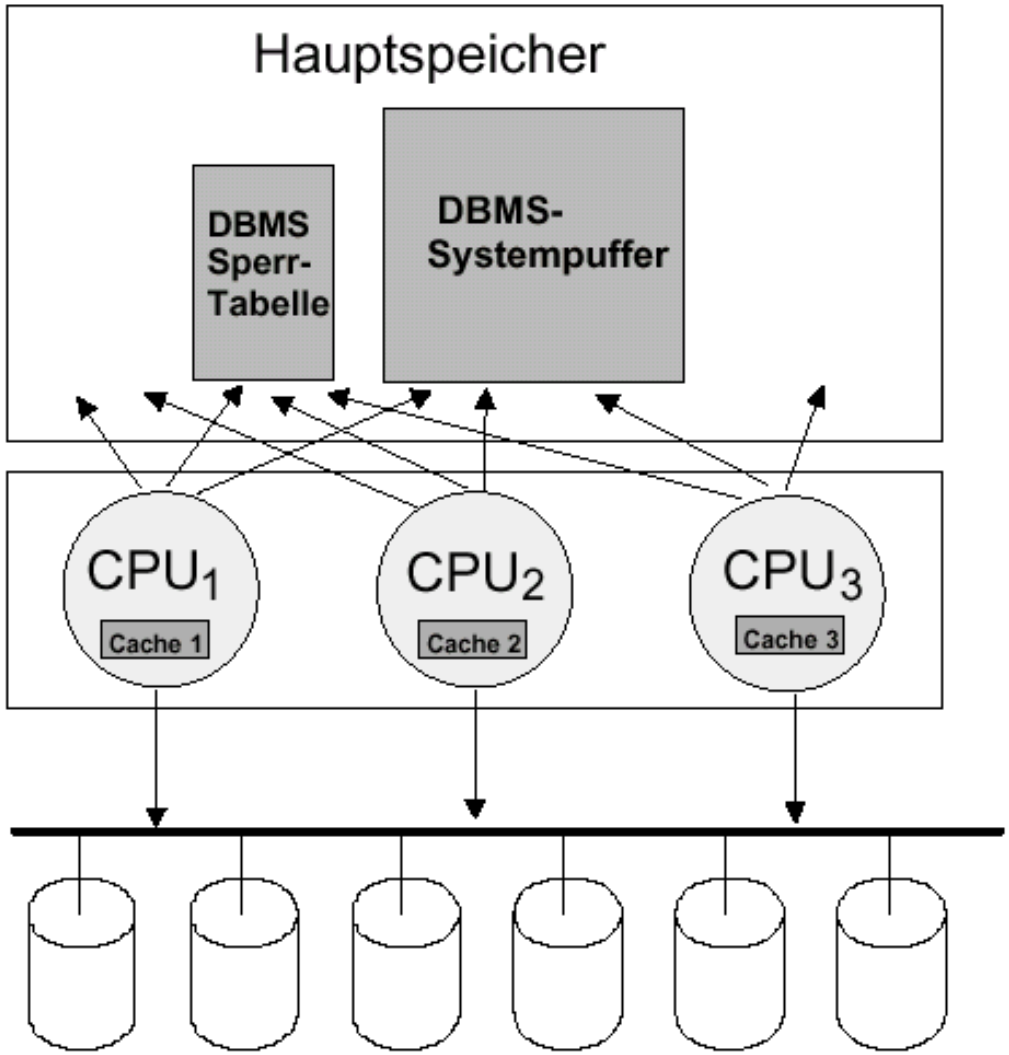
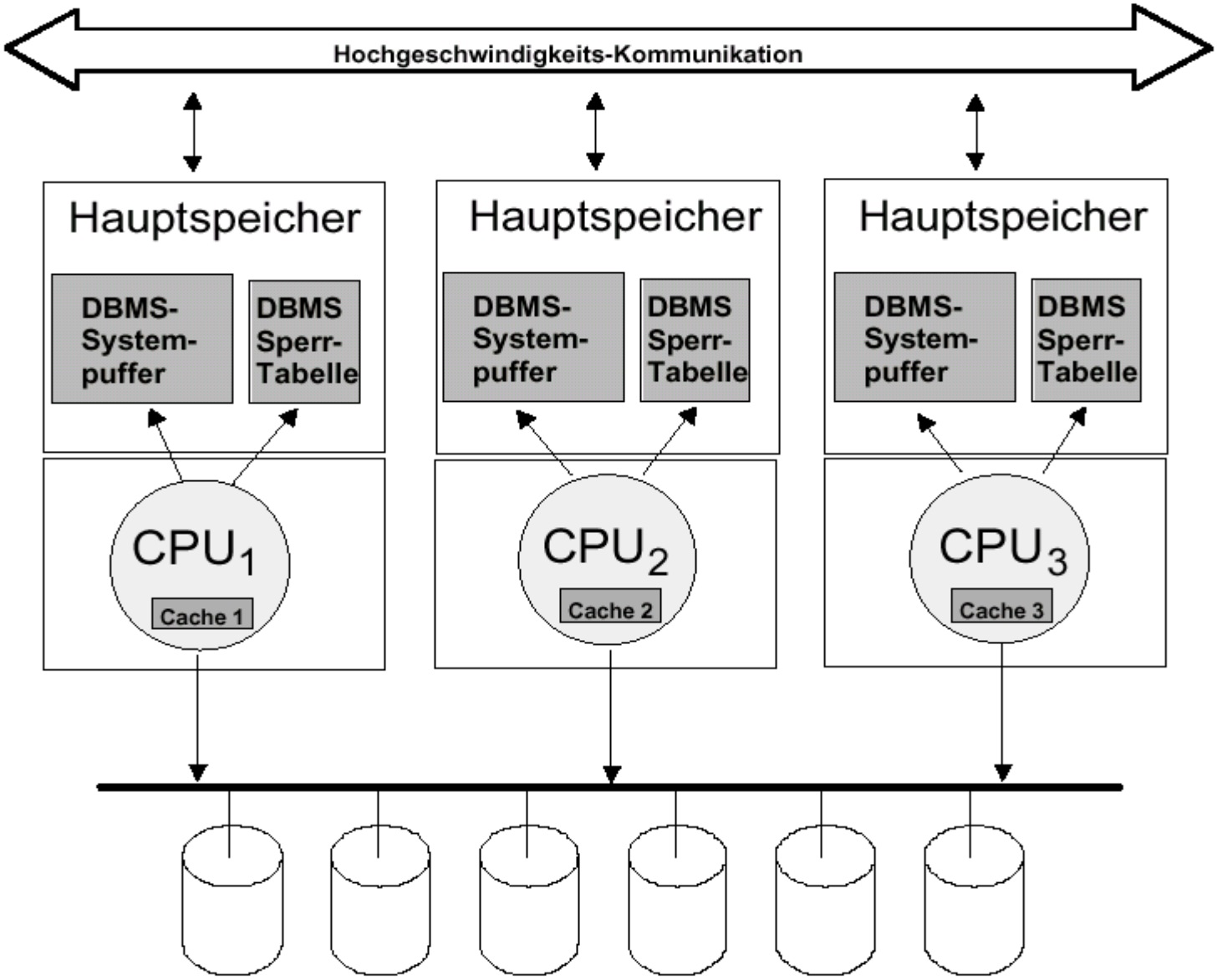
Mehrrechner-Datenbanksysteme

Grundlegende Architekturen
Anfragebearbeitungstechniken

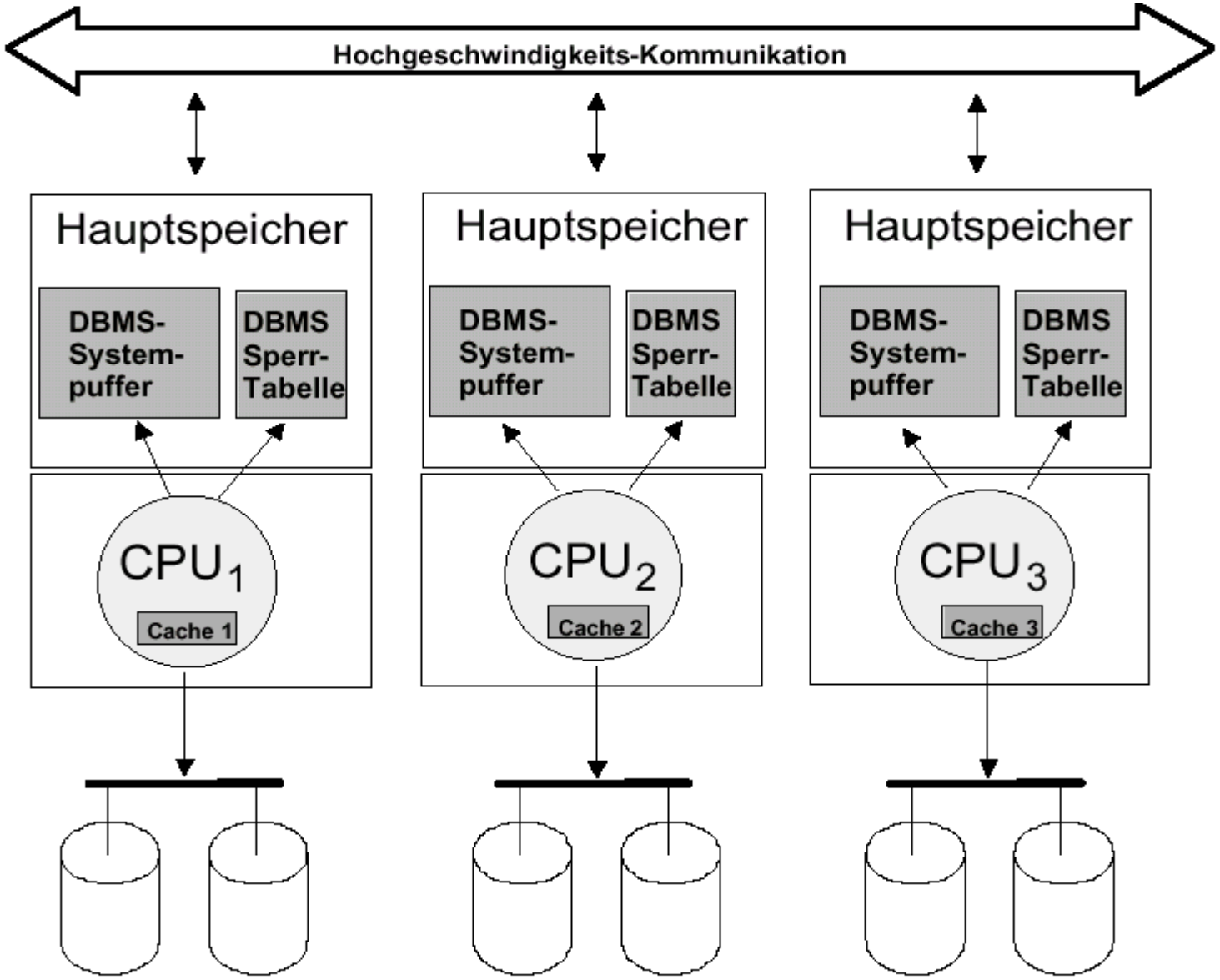
Shared Everything Architektur:
Shared Disk & Shared Memory



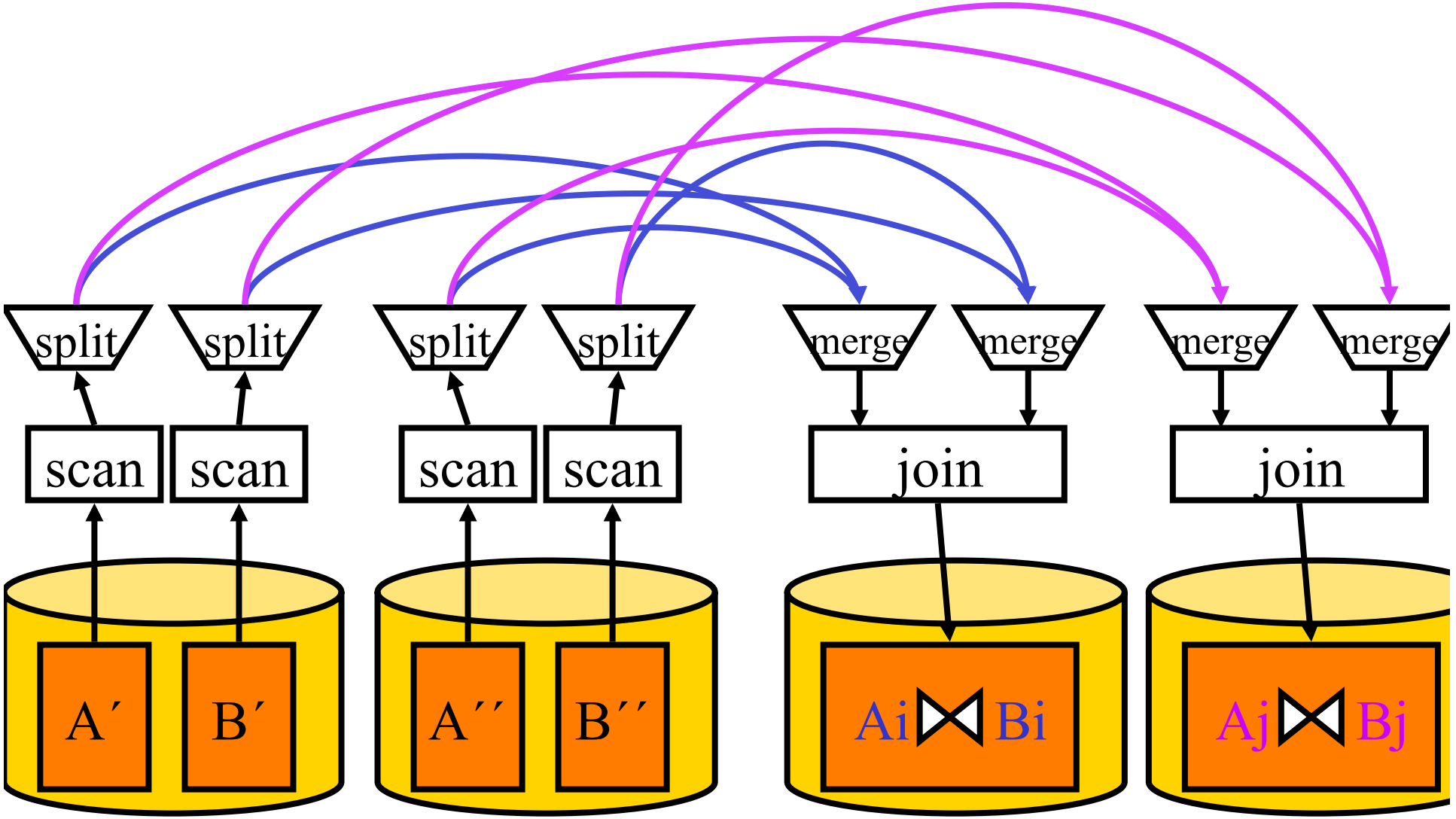
Shared Disk-Architektur



Shared Nothing-Architektur



Parallele Anfragebearbeitung: Hash Join



Paralleler Hash Join – im Detail

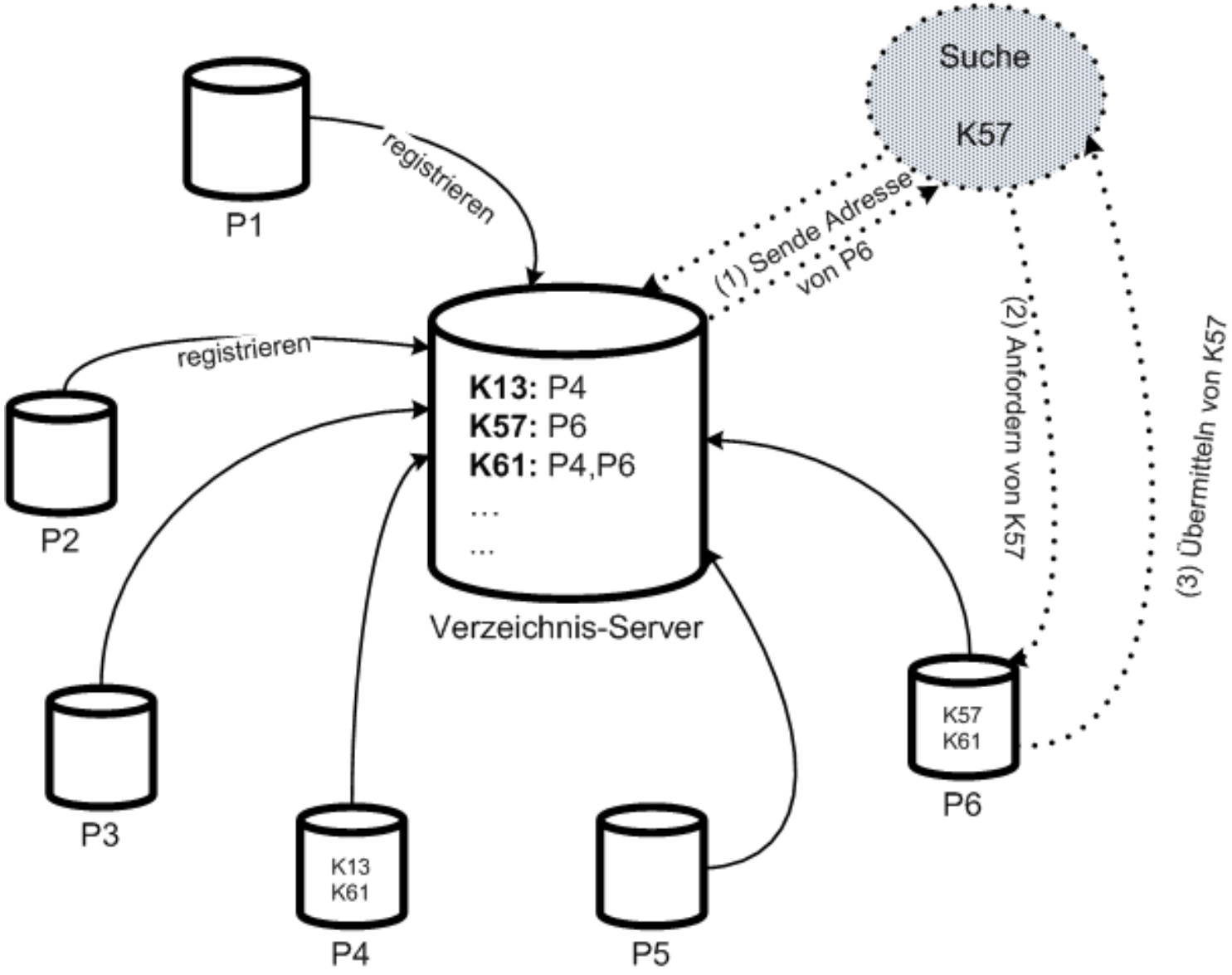
1. An jeder Station werden mittels Hash-Funktion h1 die jeweiligen Partitionen von A und B in A1,...,Ak und B1,...,Bk zerlegt
   - h1 muss so gewählt werden, dass alle Ai´s aller Stationen in den Hauptspeicher passen
2. Für alle 1 <= i <= n: Berechne jetzt den Join von Ai mit Bi wie folgt
   a. Wende eine weitere Hash-Funktion h2 an, um Ai auf die l Stationen zu verteilen
      - Sende Tupel t an Station h2(t)
   b. Eintreffende Ai-Tupel werden in die Hash-Tabelle an der jeweiligen Station eingefügt
   c. Sobald alle Tupel aus Ai „verschickt" sind, wird h2 auf Bi angewendet und Tupel t an Station h2(t) geschickt
   d. Sobald ein Bi-Tupel eintrifft, werden in der Ai-Hashtabelle seine Joinpartner ermittelt.

Peer to Peer-Informationssysteme

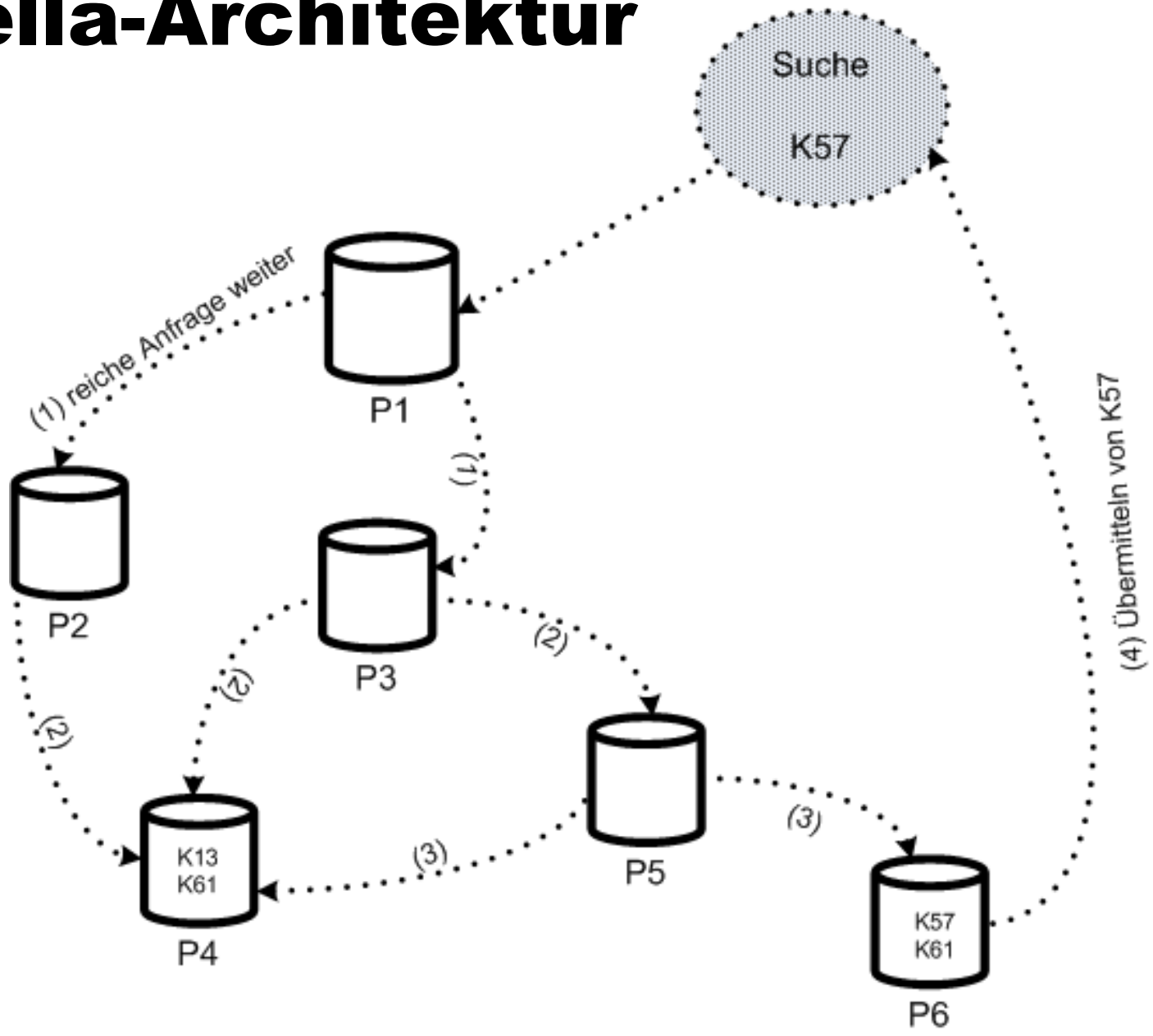
- Seti@Home
  - P2P number crunching

- Napster
  - P2P file sharing / Informationsmanagement

Napster-Architektur



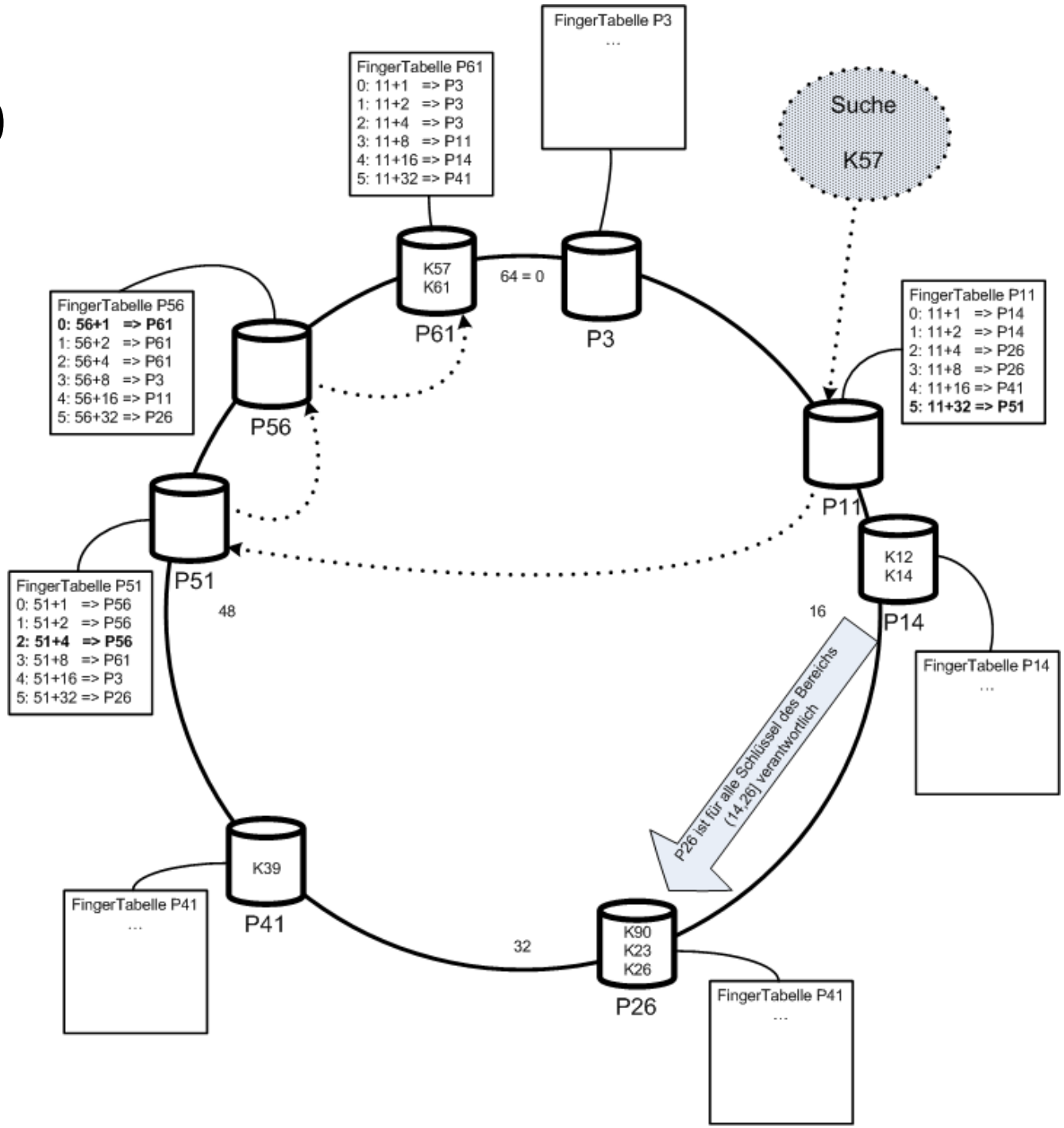
Gnutella-Architektur



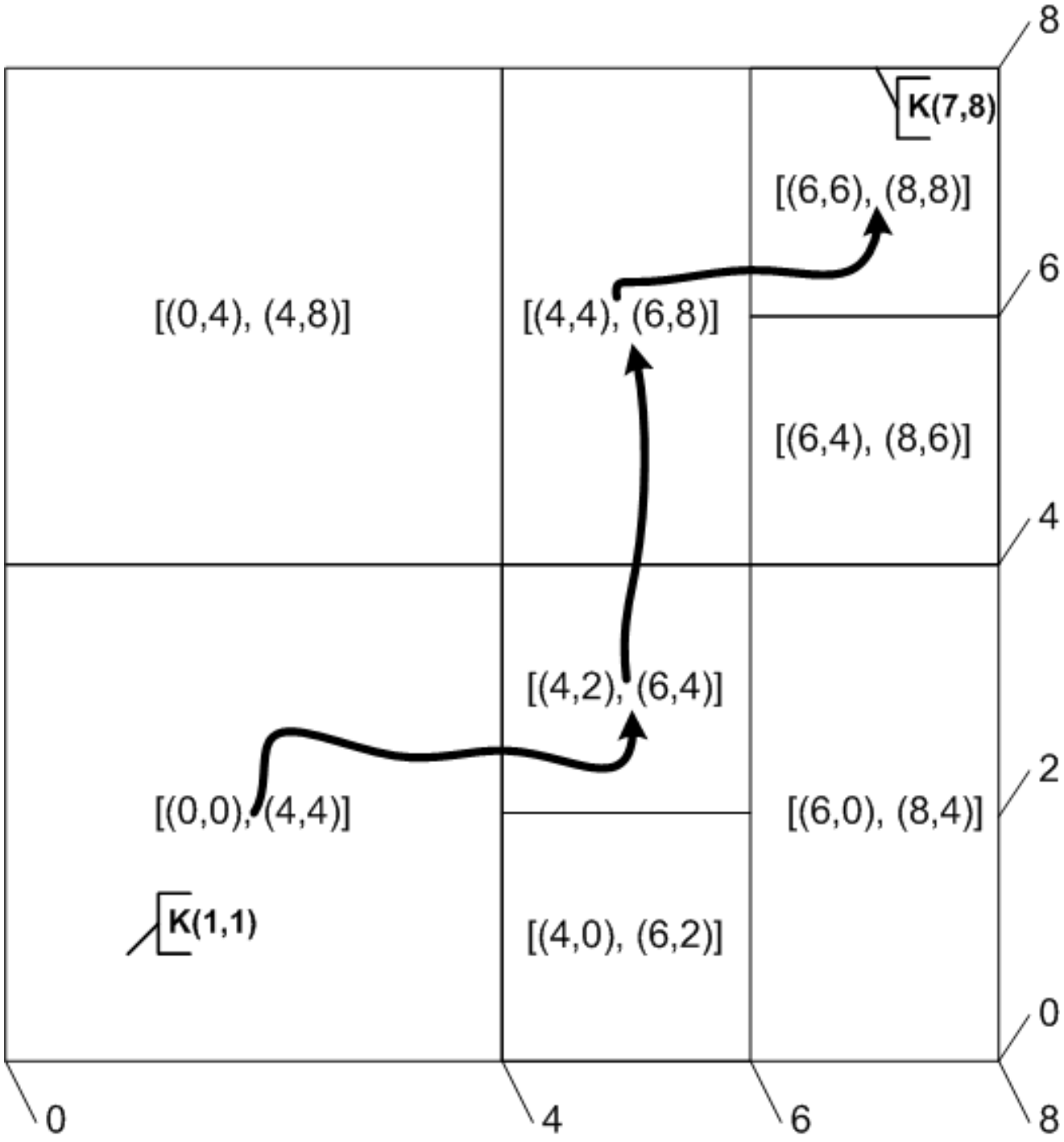
DHT: Distributed Hash Table

- Basieren auf „consistent hashing“
- Vollständige Dezentralisierung der Kontrolle
- Dennoch zielgerichtete Suche

CHORD


FingerTabelle P3
...
FingerTabelle P61
0: 11+1 => P3
1: 11+2 => P3
2: 11+4 => P3
3: 11+8 => P11
4: 11+16 => P14
5: 11+32 => P41
Suche
K57
K57
K61
P61
64 = 0
P3
FingerTabelle P56
0: 56+1 => P61
1: 56+2 => P61
2: 56+4 => P61
3: 56+8 => P3
4: 56+16 => P11
5: 56+32 => P26
P56
FingerTabelle P11
0: 11+1 => P14
1: 11+2 => P14
2: 11+4 => P26
3: 11+8 => P26
4: 11+16 => P41
5: 11+32 => P51
P11
P51
FingerTabelle P51
0: 51+1 => P56
1: 51+2 => P56
2: 51+4 => P56
3: 51+8 => P61
4: 51+16 => P3
5: 51+32 => P26
48
16
K12
K14
P14
FingerTabelle P14
...
P26 ist für alle Schlüssel des Bereichs (14,26] verantwortlich
K39
P41
FingerTabelle P41
...
32
K90
K23
K26
P26
FingerTabelle P41
...

CAN



No-SQL Datenbanken

- Internet-scale Skalierbarkeit
- CAP-Theorem: nur 2 von 3 Wünschen erfüllbar
  - Konsistenz (Consistency)
  - Zuverläassigkeit/Verfügbarkeit (Availability)
  - Partitionierungs-Toleranz
- No-SQL Datenbanksysteme verteilen die Last innerhalb eines Clusters/Netzwerks
  - Dabei kommen oft DHT-Techniken zum Einsatz

Schnittstelle der No-SQL Datenbanken

- Insert(k,v)
- Lookup(k)
- Delete(k)

- Extrem einfach ➔ effizient
- Aber: wer macht denn die Joins/Selektionen/…
  - ➔ das Anwendungsprogramm

Konsistenzmodell: ~~C~~AP

- Relaxiertes Konsistenzmodell
  - Replizierte Daten haben nicht alle den neuesten Zustand
    - Vermeidung des (teuren) Zwei-Phasen-Commit-Protokolls
  - Transaktionen könnten veraltete Daten zu lesen bekommen
  - Eventual Consistency
    - Würde man das System anhalten, würden alle Kopien irgendwann (also eventually) in denselben Zustand übergehen
  - Read your Writes-Garantie
    - Tx leist auf jeden Fall ihre eigenen Änderungen
  - Monotonic Read-Garantie
    - Tx würde beim wiederholten Lesen keinen älteren Zustand als den vorher mal sichtbaren lesen

BASE - Modell

- Basically Available
- Soft State
- Eventually Consistent

JSON Objekt/Dokument Modell

- Schema-los: frei kombinierbare Konstruktoren
- Atomare Typen
  - Unicode Strings, wrapped in quote characters.
    - “Gregory House”
    - “Name”
  - Numbers, which are double-precision IEEE floats.
    - 255.76
  - Boolean values, which are true or false.
    - true
  - Null, to denote a null value.
- Objekte { ... }
  - {“Name“ : “Gregory House“,
    “Titel“ : “Dr“}
- Arrays [...]

Zwei Beispiel-Objekte einer Kollektion People

```
{   "name":      "George Bluth",
    "age":       58,
    "indicted": true,
    "kids": ["Gob", "Lindsay", "Buster",
        {
            "name": "Michael",
            "age":  38,
            "kids": ["George-Michael"]
        }],
    "rival":     "Stan Sitwell" }

{   "name":             "Stan Sitwell",
    "age":              "middle-aged",
    "charity_giving": 250120.5,
    "kids":             ["Sally"] }
```

Figure 1: A pair of valid JSON objects.

Systeme

- MongoDB
- Cassandra
- Dynamo
- BigTable
- Hstore
- SimpleDB
- S3

Relationale Modellierung der JSON Objekte

argo_people_str

objid	keystr	valstr
1	name	George Bluth
1	kids[0]	Gob
1	kids[1]	Lindsay
1	kids[2]	Buster
1	kids[3].name	Michael
1	kids[3].kids[0]	George-Michael
1	rival	Stan Sitwell
2	name	Stan Sitwell
2	age	middle-aged
2	kids[0]	Sally

argo_people_num

objid	keystr	valnum
1	age	58
1	kids[3].age	38
2	charity_giving	250120.5

argo_people_bool

objid	keystr	valbool
1	indicted	true

Client/Server-Datenbanksysteme und Datenbankanwendungen

Architekturen
Basistechnologien
Sicherheitsaspekte

CS-Architekturen

- Client/Server war das Modewort der 90-er Jahre
  - Downsizing/Rightsizing
  - CS hat uns das WWW gebracht (Webserver/Browser)
- Anforderung/Initiative geht vom Client aus
- Server bedient den Client
  - Server bedienen viele Klienten gleichzeitig
    - Deshalb sollte der Server unbedingt immer „multi-threaded“ realisiert werden
    - Dadurch werden die Rechnerressourcen besser ausgenutzt
  - Server kann im Rahmen der Bearbeitung selbst zum Client werden

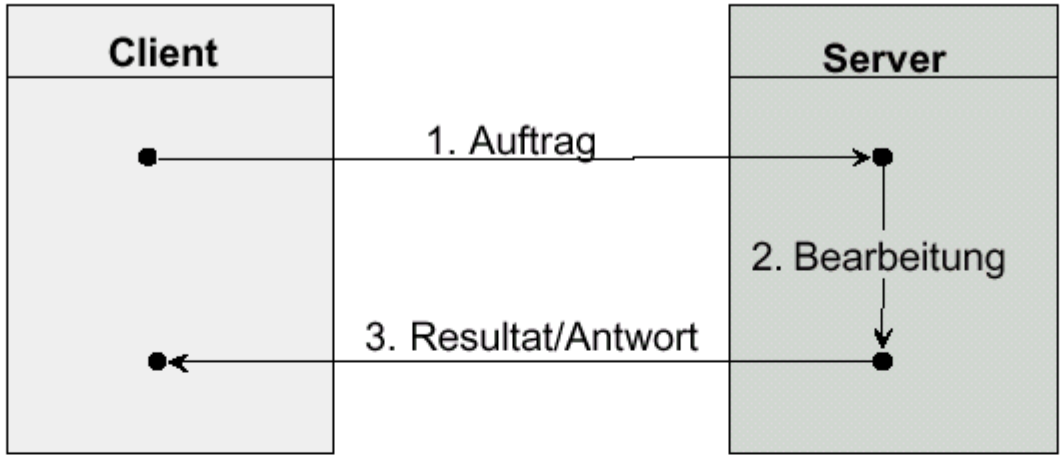


Abb. 12-1: Interaktionen im Client/Server-Modell

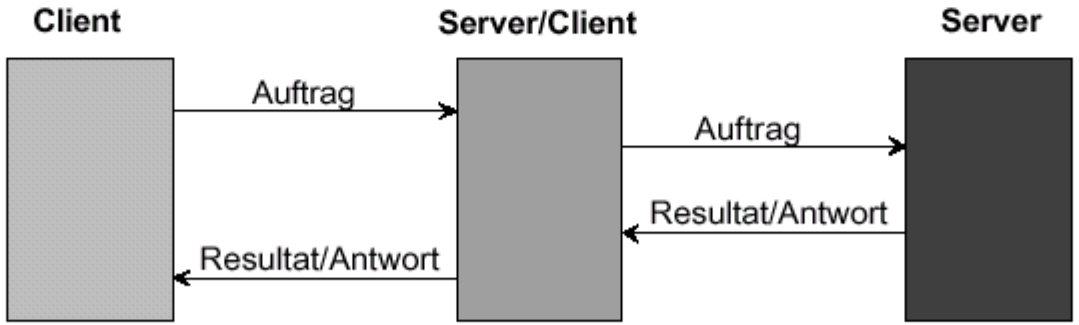
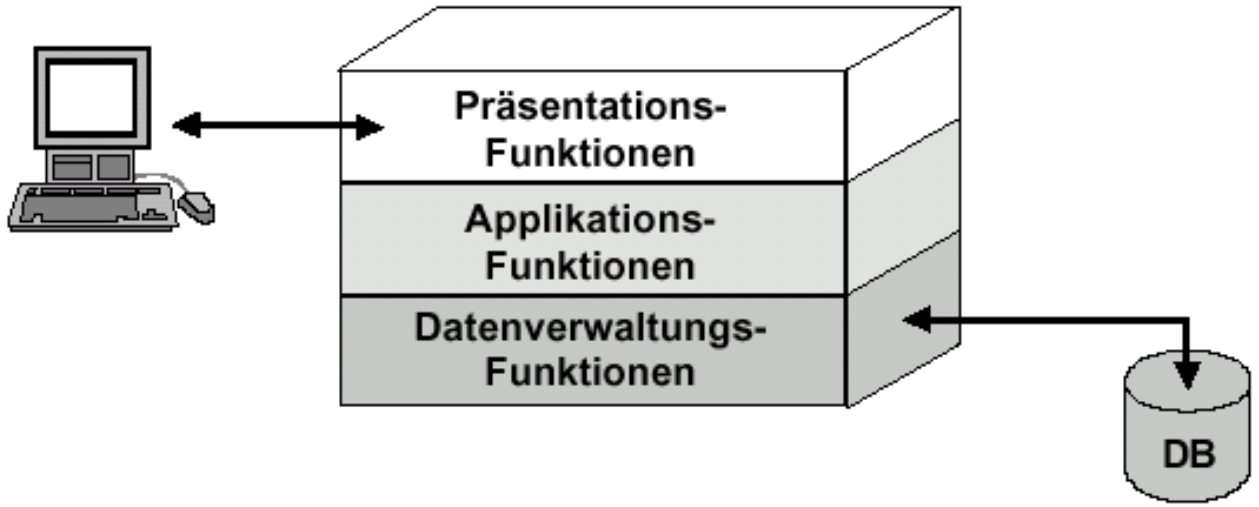


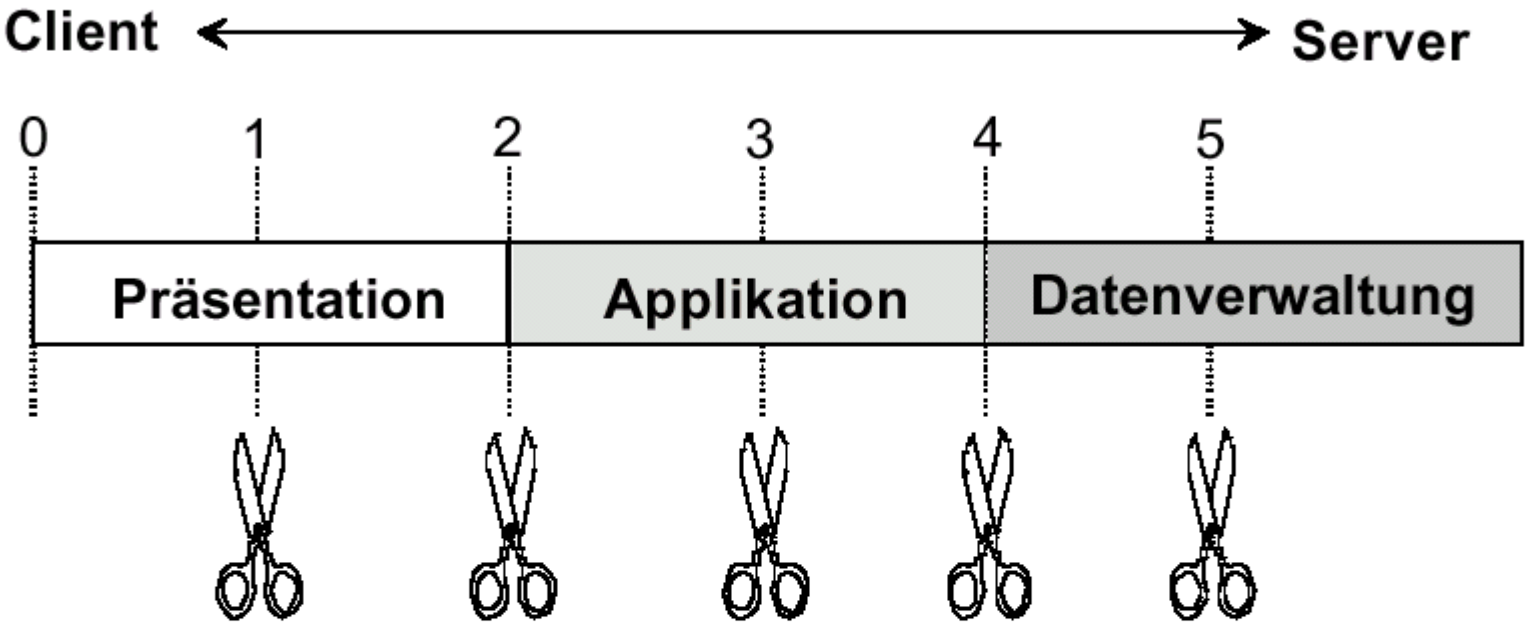
Abb. 12-2: Verschiedene Rollen bei Auftragsausführung

Komponenten einer Anwendung

1. Präsentationsfunktion
   - GUI
2. Applikationsfunktion
   - Programm- und Ablauflogik
   - Aufbereitung der SQL Statements zur Interaktion mit der DB
3. Datenverwaltungsfunktion



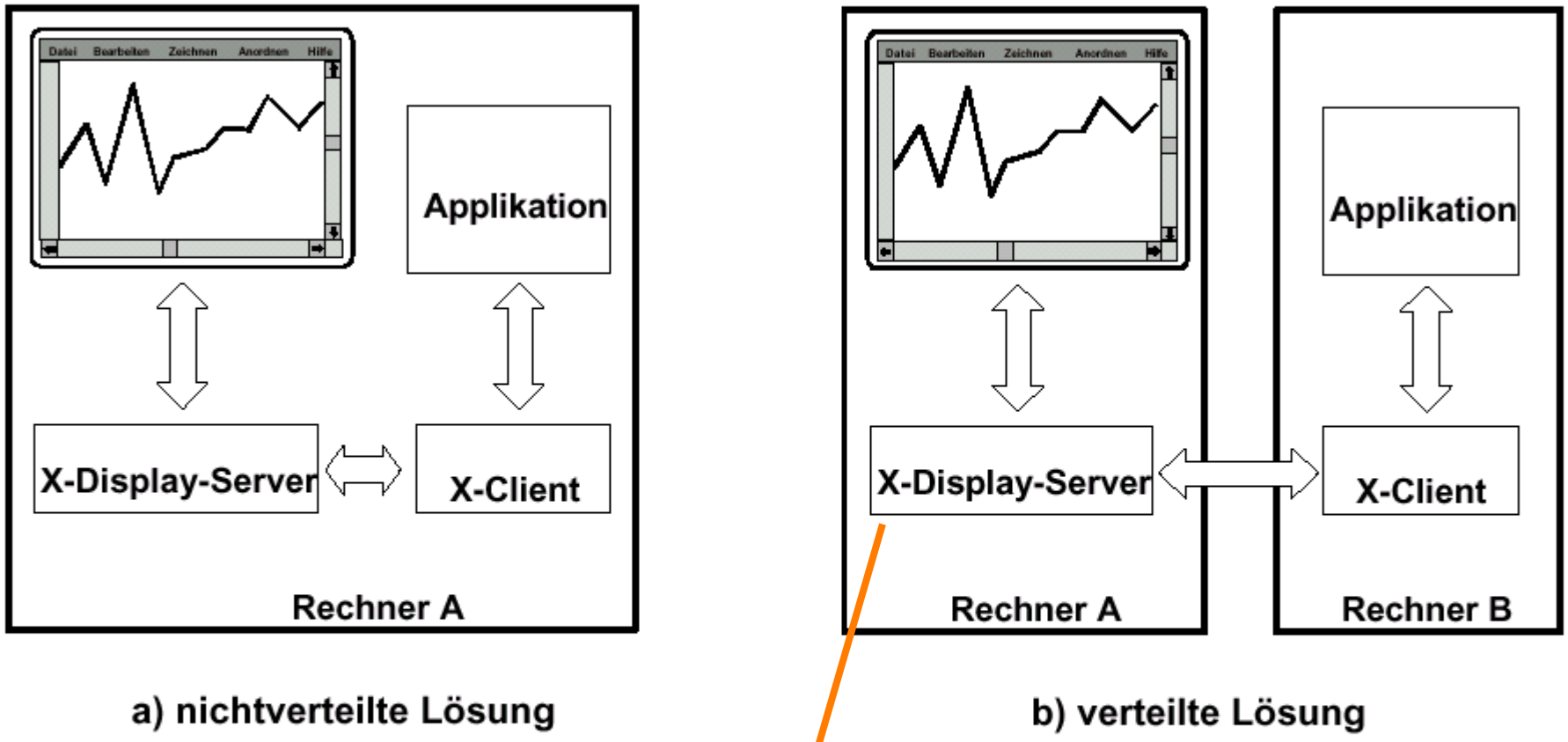
Architektur-Entscheidungen: Funktionsverteilung



1. Verteilte Präsentation
2. Entfernte Präsentation
3. Verteilte Applikationsfunktion
4. CS mit entferntem Datenbankzugriff
5. Datenbankserver mit entferntem Dateiserver (file server, NFS)

Verteilte Präsentation: X-Windows

(Server läuft auf Displaystation, Client in der Anwendung)



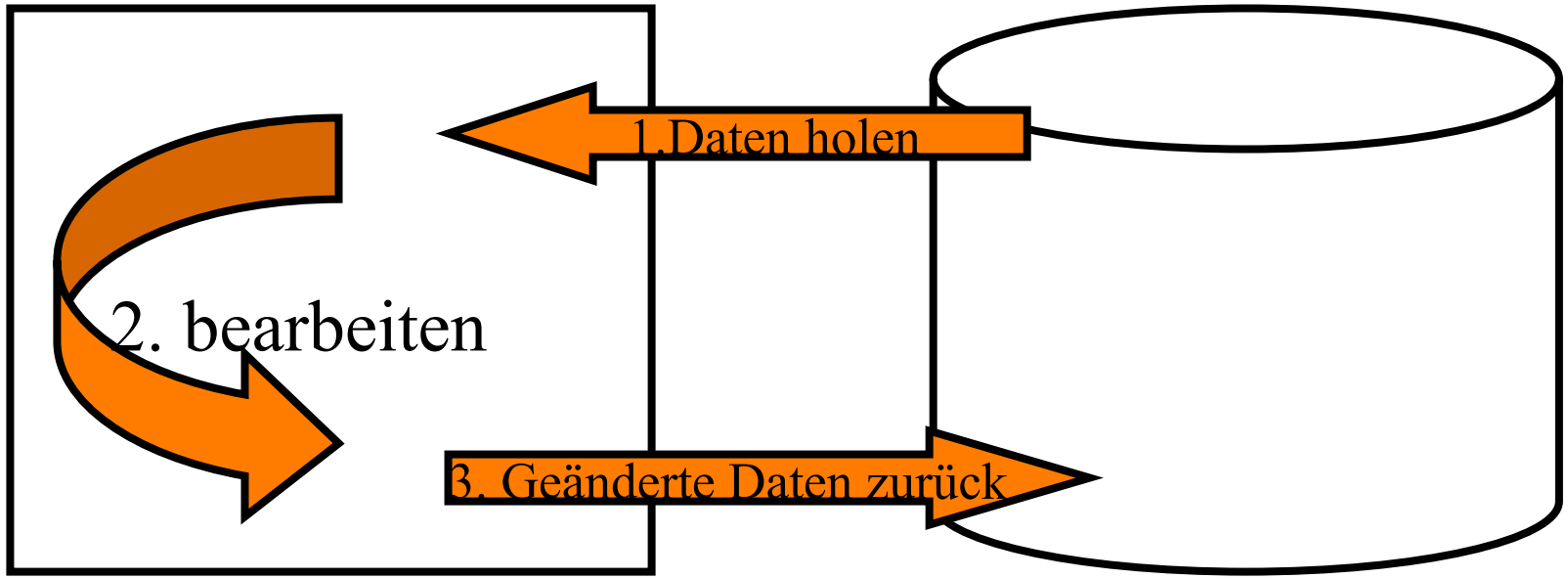
a) nichtverteilte Lösung

b) verteilte Lösung

Etwas gewöhnungsbedürftig, da der X-Server auf dem Client-Rechner läuft

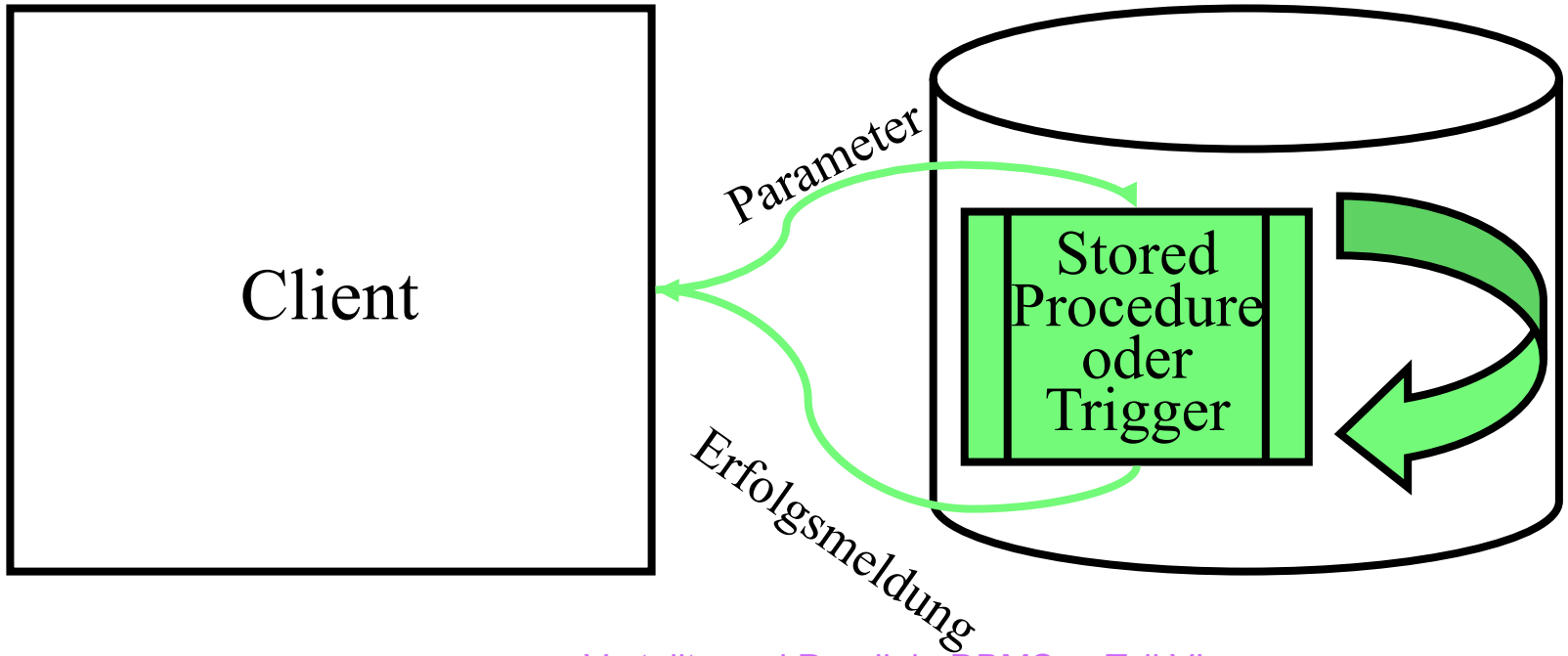
Remote Database Access

- Zugriff auf entfernte Datenbanken
- Geht heute meist über die standardisierten Schnittstellen
  - ODBC
  - JDBC
- Design-Entscheidung wo Arbeit ausgeführt wird
  - Client oder Server



Remote Database Access

- Design-Entscheidung wo Arbeit ausgeführt wird
  - Client oder Server



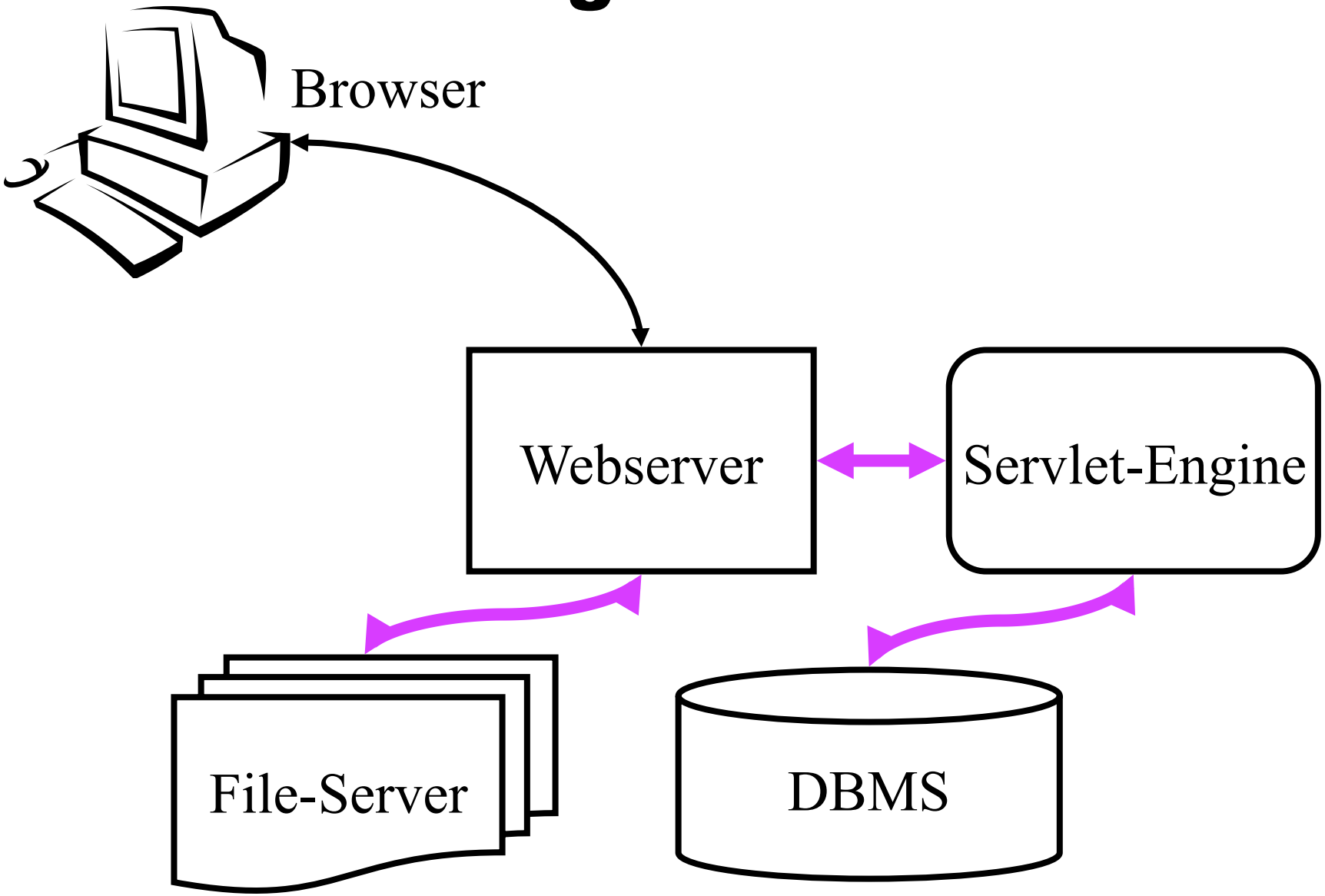
Einordnung einiger Systeme

- SAP R/3
  - Dreistufige Client/Server-Architektur
    1. Präsentations-Rechner
    2. Anwendungsserver
       - Versucht, den DB-Server so wenig wie möglich zu belasten
       - Macht die gesamte Bearbeitung
       - Sogar die Synchronisation (Sperren von Anwendungsobjekten)
    3. Datenbankserver
       - Wird im Wesentlichen als Storage Server verwendet
       - Anwendungsserver haben sehr viel Datenbankfunktionalität nachgebaut
         - Anfragebearbeitung
         - Synchronisation
         - Pufferung

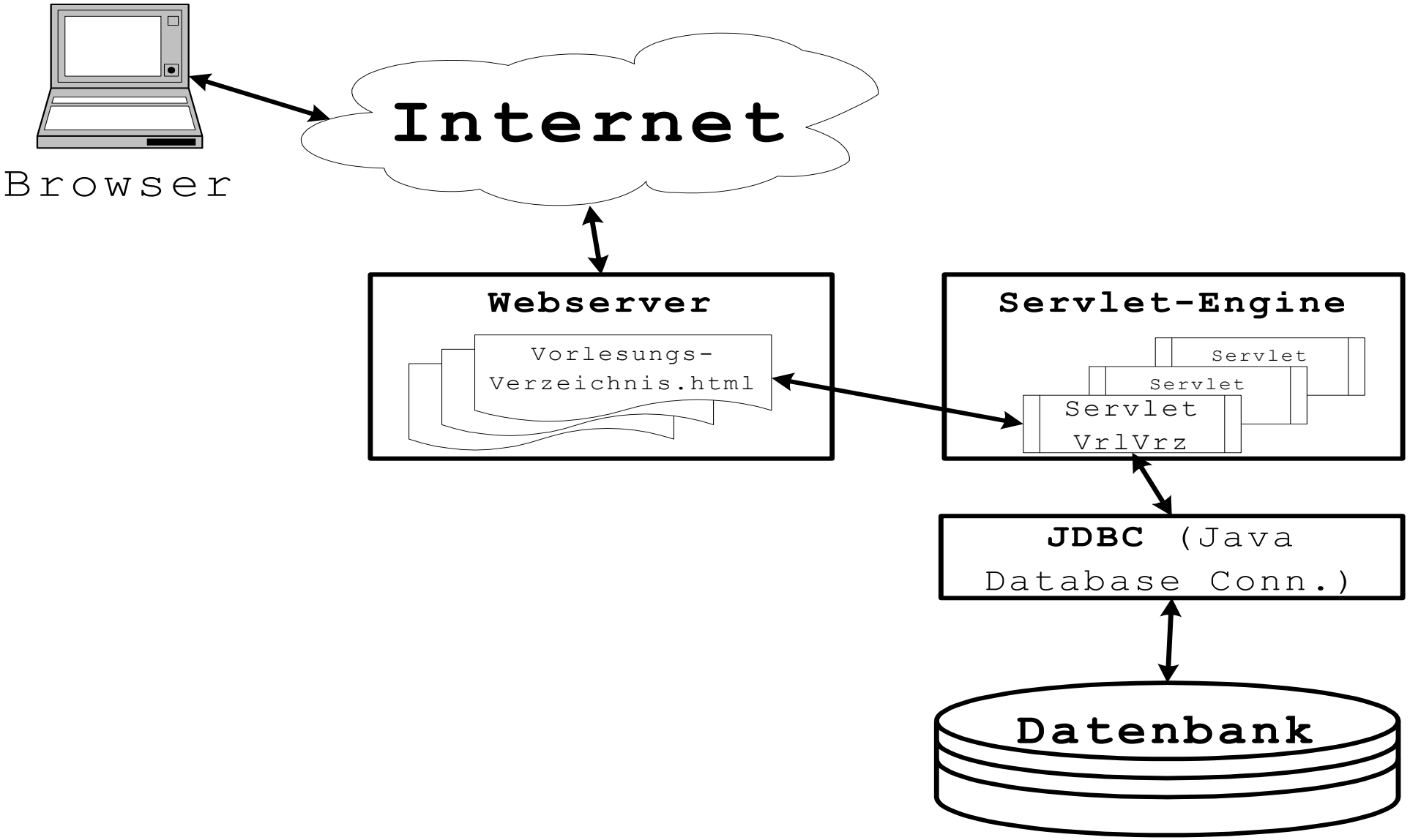
Einordnung einiger Systeme (cont´d)

- Web-Anwendungen mit Datenbankanbindung, wie Mitfahrbörse
  - Heute meistens mit Java-Servlets realisiert
  - Oder mit CGI-Skripten (veraltete Architektur, Amazon)
  - Vielstufige CS-Architektur
    - Webbrowser ~ Client
    - Webserver
      - Server für Browser
      - Client der Servlet-Engine
    - Servlet-Engine
      - Server für Webbrowser
      - Client des/der DBMSs(per JDBC angesteuert)
    - DBMS ist Server der Servlet-Engine

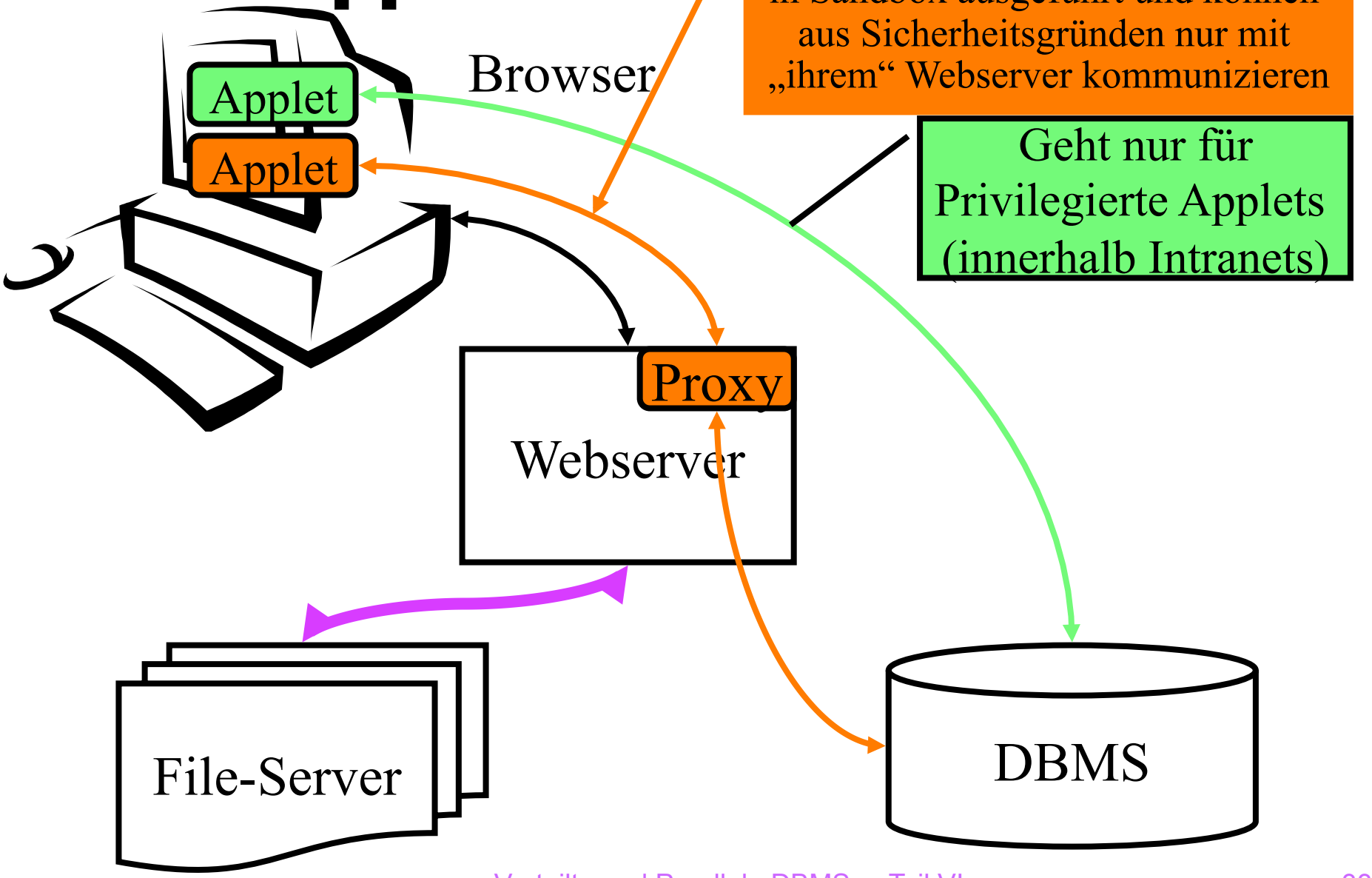
Architektur Datenbank-basierter Webanwendungen



Web-Anbindung von Datenbanken via Servlets



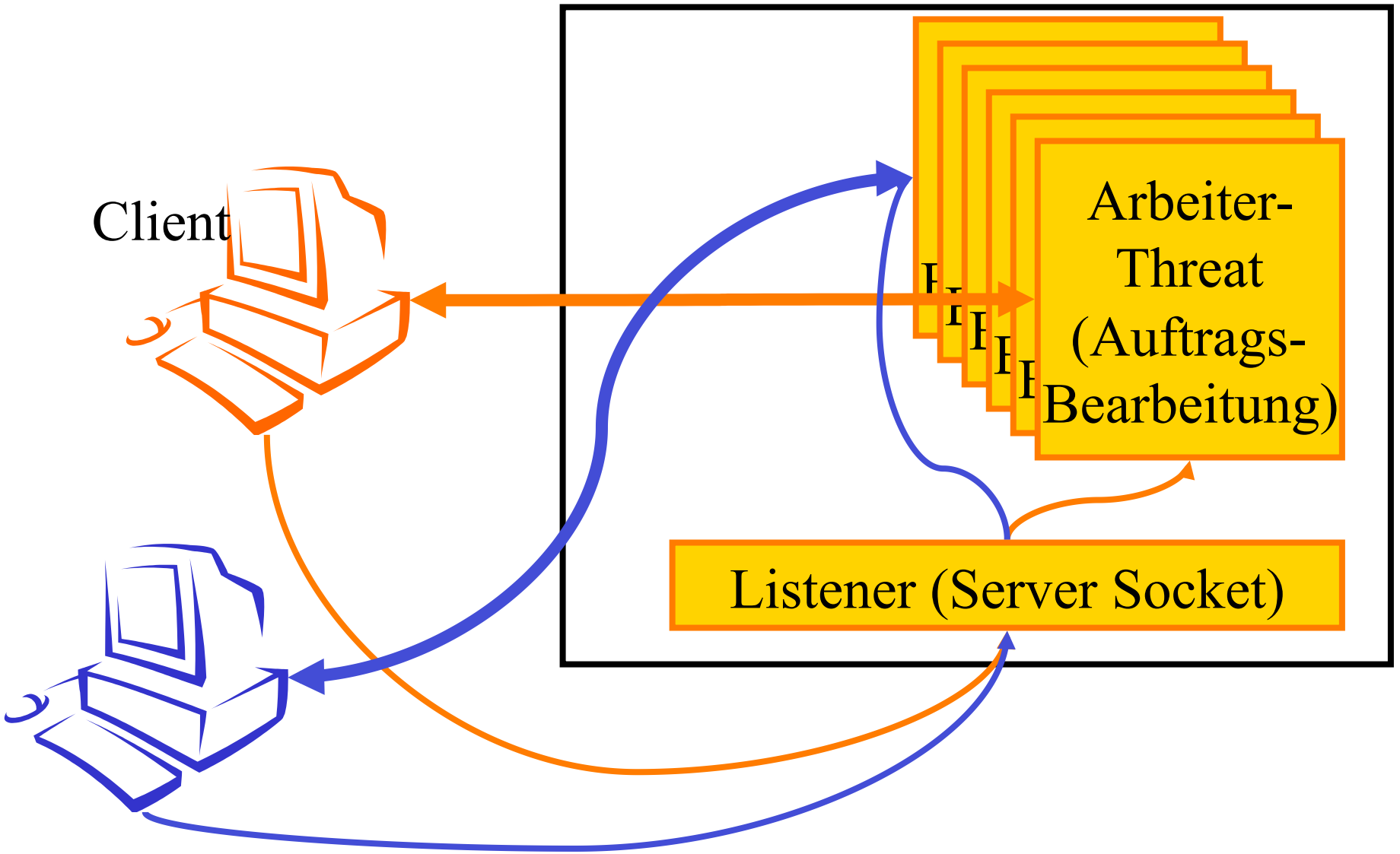
Architektur der DB-Anbindung mittels Applet



Basistechnologie für Client/Server-Datenbankanwendungen

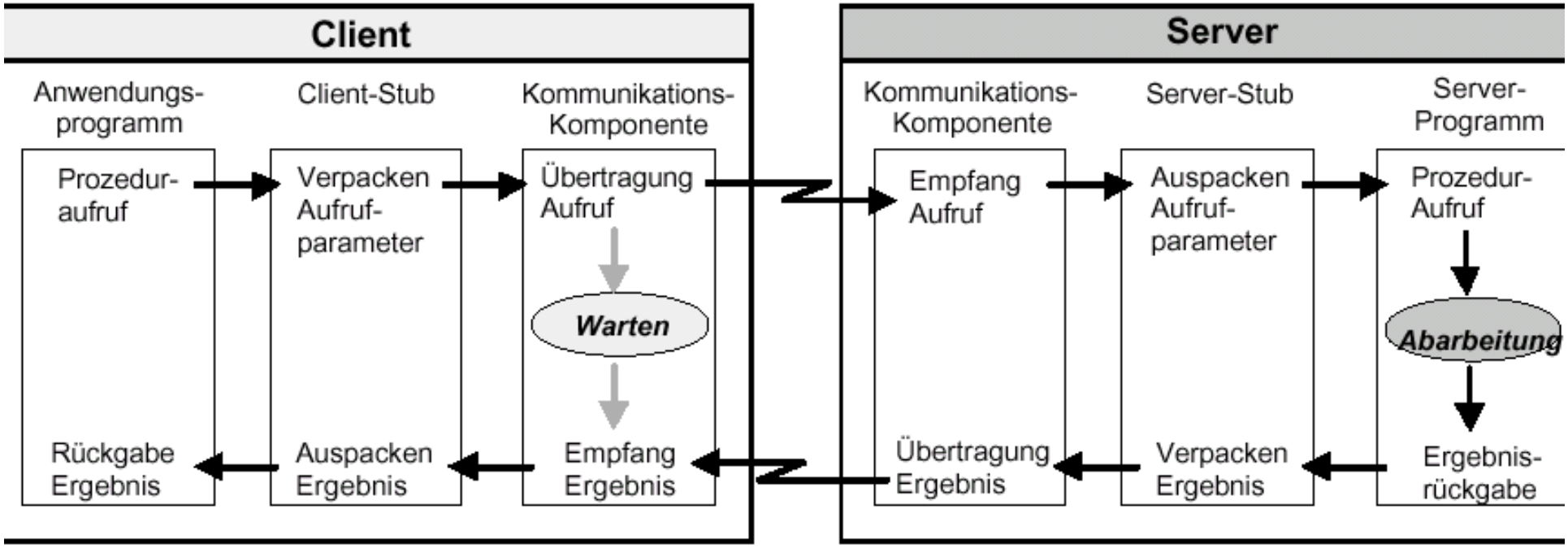
- Socket-Schnittstelle für die Kommunikation
  - TCP (stream-orientiert)
  - UDP (Datagramm-orientiert)
- Threads für die Realisierung „multi-threaded" Server
  - Ein „Listener" (Server-Socket-Prozess) lauscht auf Anforderungen
  - Aufträge werden dann an die Arbeiter-Threats zur Bearbeitung weitergeleitet
  - Optimierung: Arbeiter-Threat sterben nicht nach getaner Arbeit sondern warten auf den nächsten Auftrag
  - Siehe Prakt. Informatik II
- Abhörsichere Socketverbindungen mittels SSL
  - Siehe später

Multi-Threaded Server



Basistechnologie für Client/Server-Datenbankanwendungen (cont´d)

- Remote Procedure Call
  - SUN-RPC
  - DCE-RPC bzw. OSF (Open Software Foundation
  - RMI (Remote Method Invocation, Sun, Java)
- Ablauf des RPC wie folgt:



RPC-Ausführungssemantiken

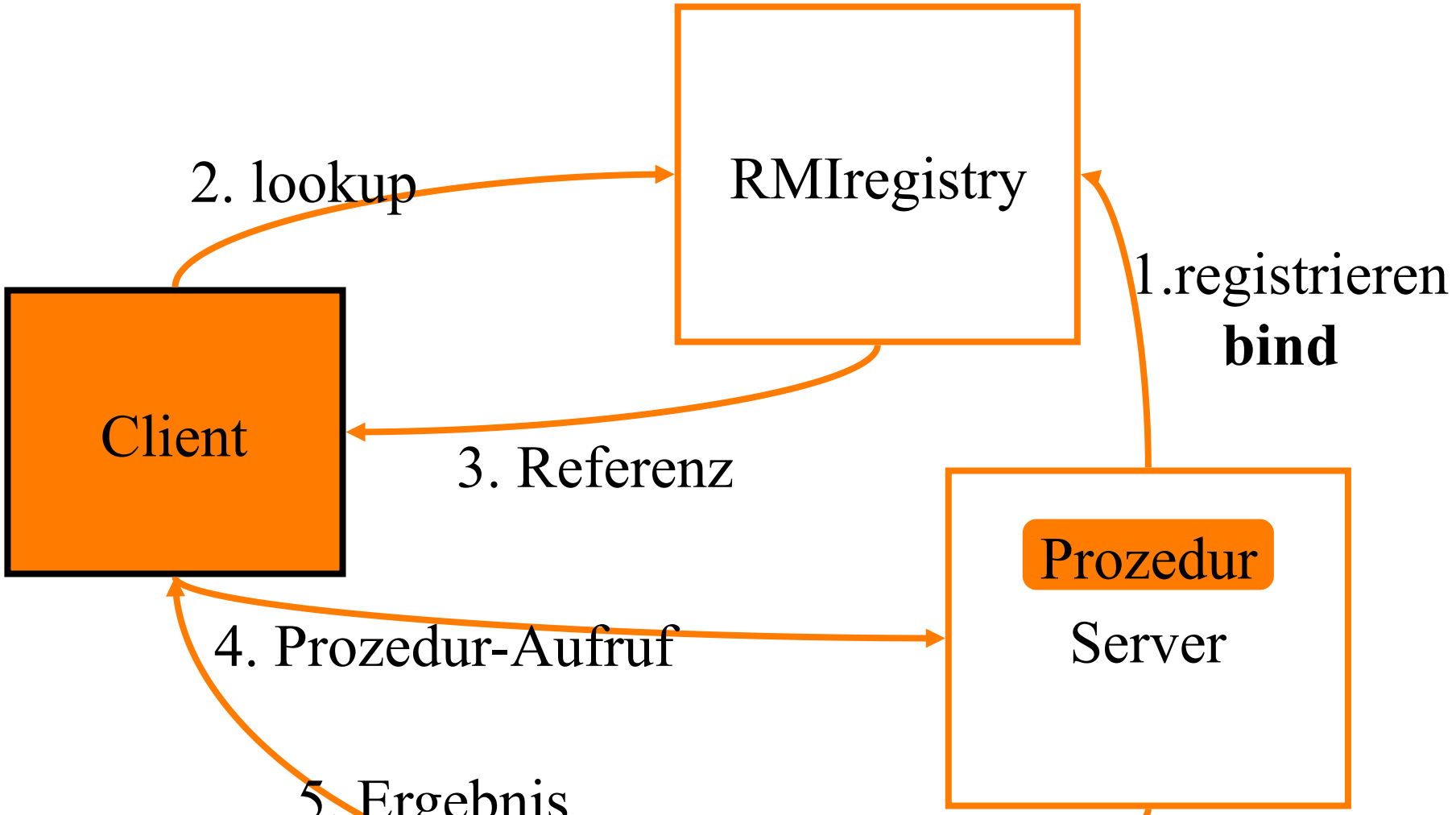
- "***kann sein***" (***may-be***)-*Semantik*: Ausführung ohne Garantie über die Häufigkeit der Prozedurausführungen in Fehlerfällen

- "***höchstens einmal***" (***at most once***)-*Semantik*: maximal einmalige Ausführung der Prozedur auf Serverseite, ggf. mehrfach auf Serverseite eintreffende Aufruf-aufforderungen werden ignoriert,

- "***mindestens einmal***" (***at least once***)-*Semantik*: die Prozedur wird mindestens einmal auf dem Server ausgeführt (evtl. auch mehrfach),

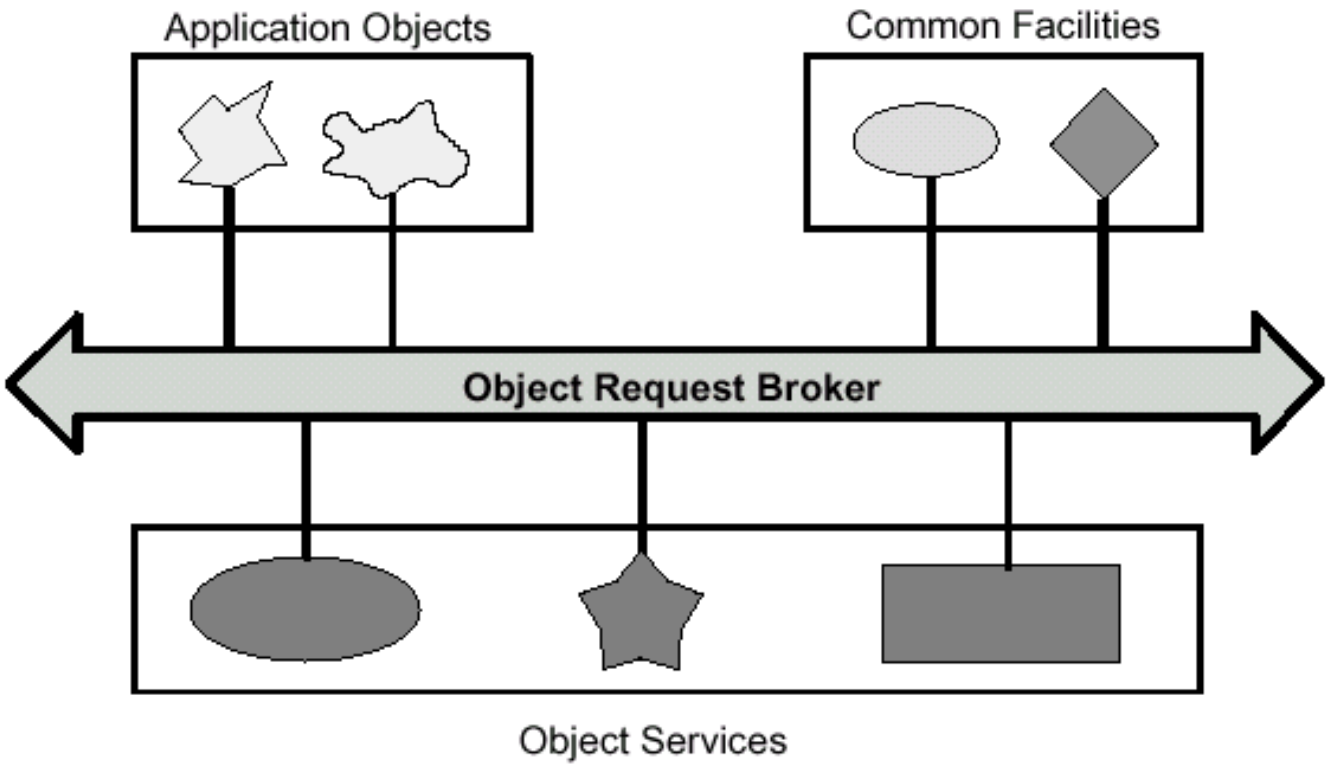
- "***genau einmal***" (***exactly once***)-*Semantik*: die Prozedur wird genau einmal ausgeführt.

RMI: Client, RMIregistry und Server

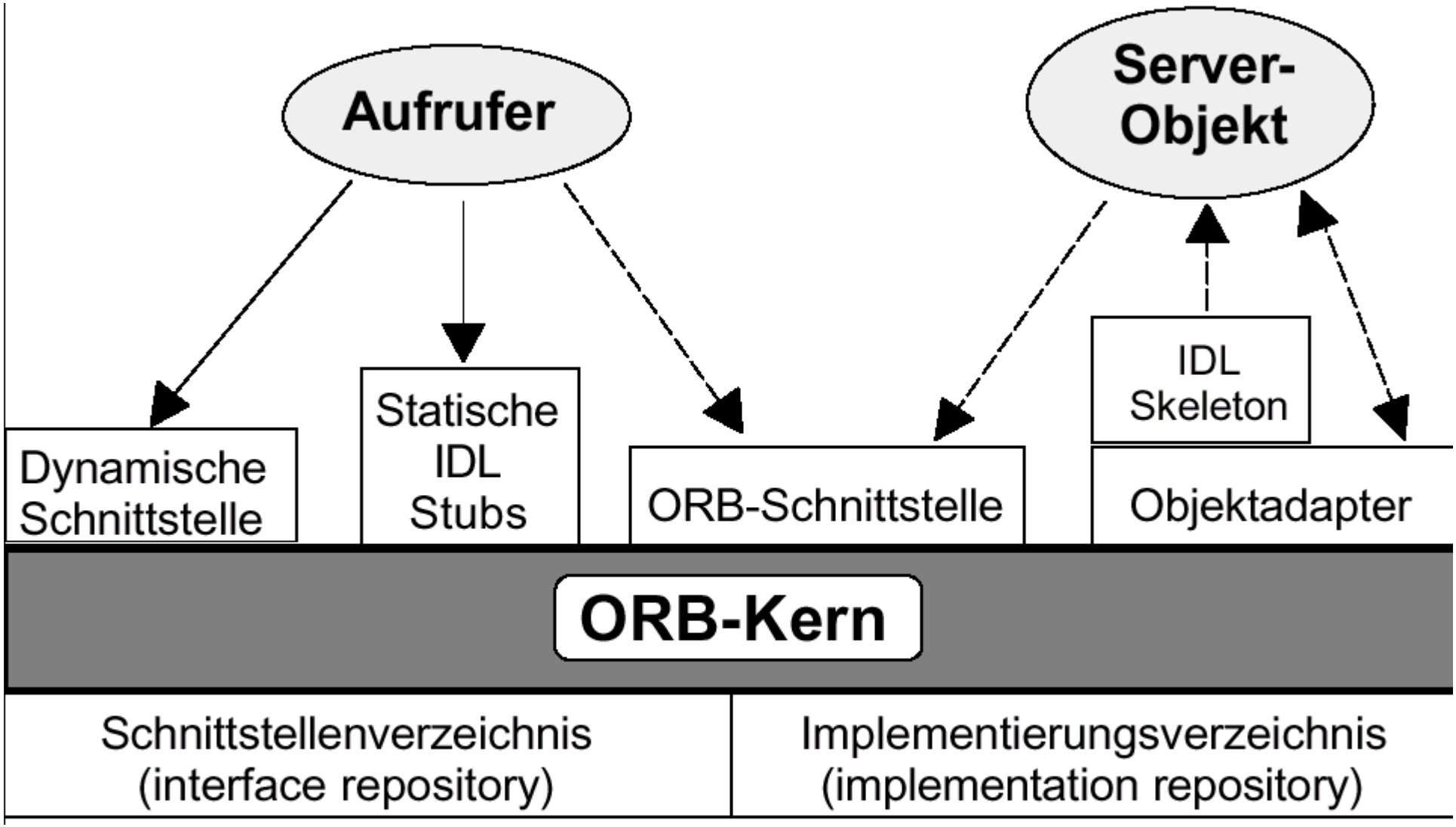


CORBA: Common Object Request Broker

- Middleware für verteilte objekt-orientierte Systeme
  - RMI ist Java-spezifisch
  - CORBA ist sprach-neutral



Nutzung entfernter Objekte via Object Request Broker



Sicherheitsprobleme: SQL-Injection Attacken

- Hinter den meisten Web-Applikationen verbergen sich Datenbanksysteme
- Aus den Eingabe-Parametern werden SQL-Anfragen generiert
- Man darf diesen Eingabe-Parametern NIEMALS trauen, da sie „ausführbaren“ SQL-Code enthalten könnten

Naive Authentifizierung

Studenten			
MatrNr	Name	Semester	Passwort
24002	Xenokrates	18	AlterGrieche
25403	Jonas	12	Bruno
26120	Fichte	10	Idealismus
26830	Aristoxenos	8	Halbton
27550	Schopenhauer	6	WilleUndVorstellung
28106	Carnap	3	logischeAnalyse
29120	Theophrastos	2	Peripatos
29555	Feuerbach	2	Naturalismus

prüfen			
MatrNr	PersNr	VorlNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Mit jeder Anfrage wird das Passwort übergeben

```
Select *
From Studenten s join prüfen p on s.MatrNr = p.MatrNr
Where s.Name = … and s.Passwort = …
```

```
Select *
From Studenten s join prüfen p on s.MatrNr = p.MatrNr
Where s.Name = 'Schopenhauer' and
                    s.Passwort = 'WilleUndVorstellung'
```

prüfen			
MatrNr	PersNr	VorlNr	Note
27550	4630	2137	2

Attacke ...

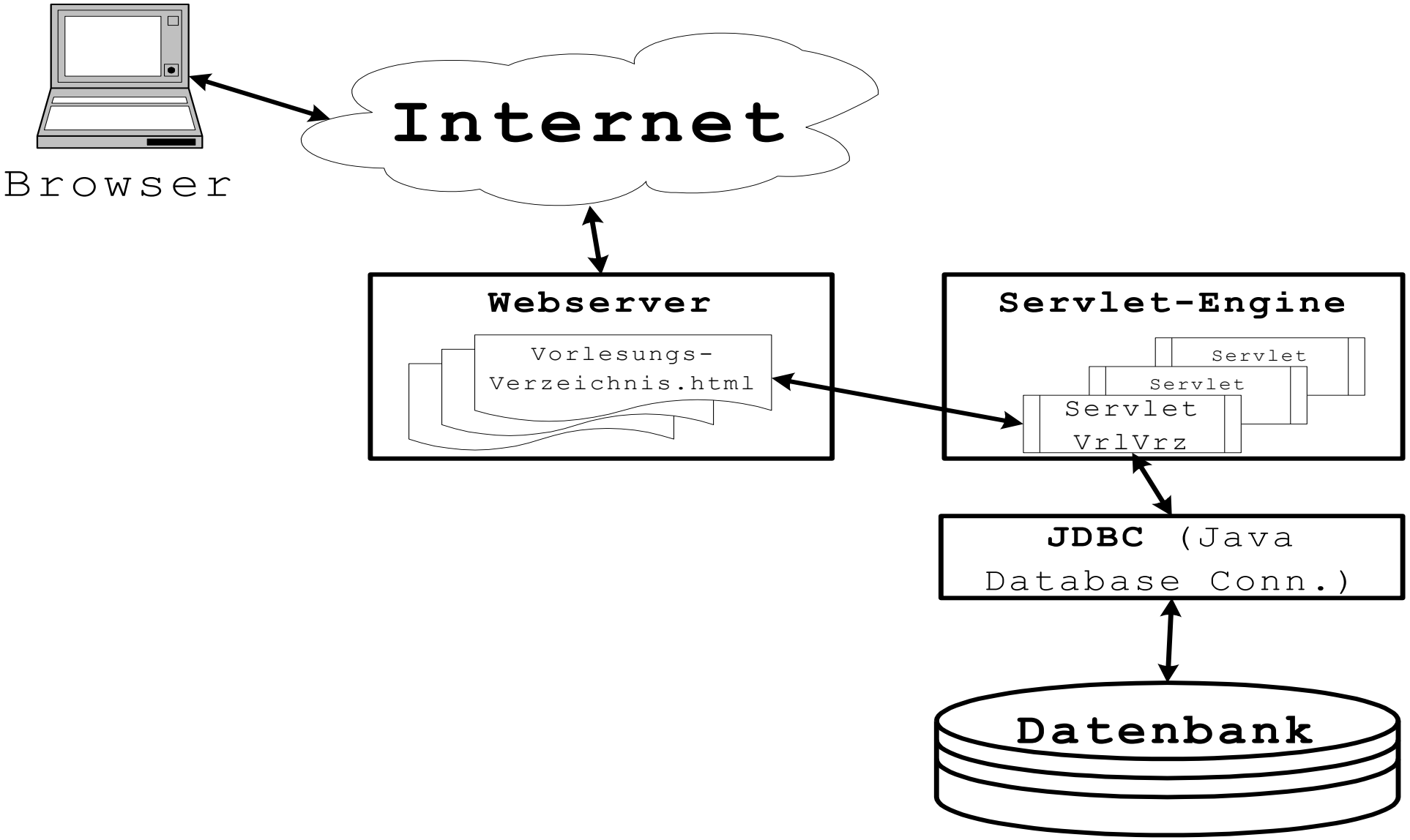
Name: Schopenhauer

Passwort: WilleUndVorstellung‘ **or** ‘x‘ = ‘x‘

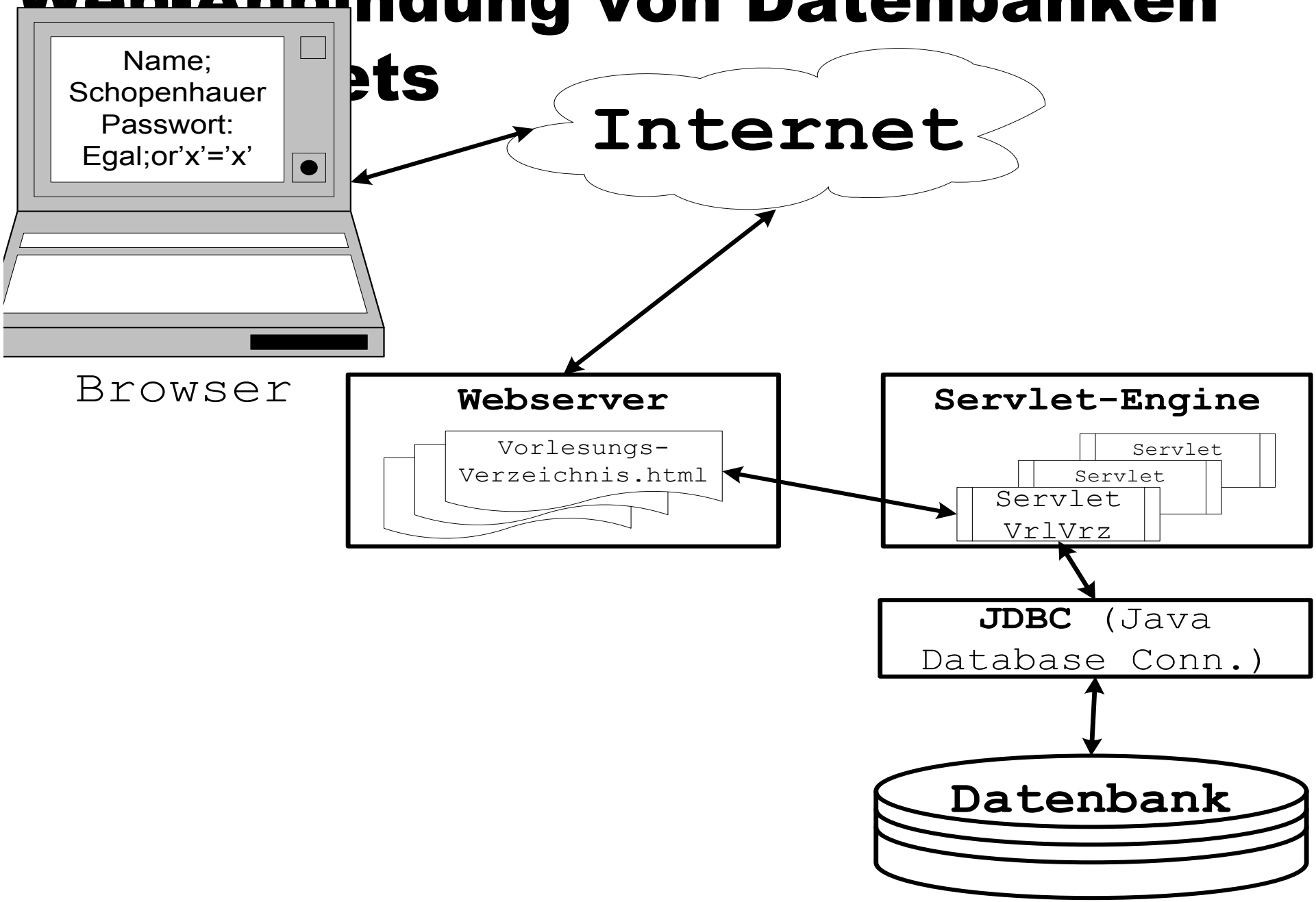
Select *
From Studenten s **join** prüfen p **on** s.MatrNr = p.MatrNr
Where s.Name = ‘Schopenhauer‘ **and**
s.Passwort = ‘WilleUndVorstellung‘ **or** ‘x‘ = ‘x‘

prüfen			
MatrNr	PersNr	VorlNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Web-Anbindung von Datenbanken via Servlets



Web-Anbindung von Datenbanken
ets



SQL-Injektion via Web-Schnittstelle

```
String _name = ... //Auslesen aus der Session etc = Benutzereingabe
String _pwd = ... // analog

String _query =
    "select * " +
    "from Studenten s join prüfen p on s.MatrNr = p.MatrNr" +
    "where s.Name = '" + _name +
                              "' and s.Passwort = '" + _pwd +"';";

// initialisiere Connection c;
Statement stmt = c.createStatement;
ResultSet rs = stmt.execute(_query); // oder ähnlich;
```

Attacke ...

Name: Schopenhauer

Passwort: weissIchNichtAberEgal‘ **or** ‘x‘ = ‘x‘

Select *
From Studenten s **join** prüfen p **on** s.MatrNr = p.MatrNr
Where s.Name = ‘Schopenhauer‘ **and**
s.Passwort = ‘weissIchNichtAberEgal‘ **or** ‘x‘ = ‘x‘

prüfen			
MatrNr	PersNr	VorlNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Attacke ...

Name: Schopenhauer

Passwort: Egal‘; **delete from** prüfen **where** ‘x‘ = ‘x‘

Select *
From Studenten s **join** prüfen p **on** s.MatrNr = p.MatrNr
Where s.Name = ‘Schopenhauer‘ **and**
s.Passwort = ‘Egal‘; **delete from** prüfen **where** ‘x‘ = ‘x‘

prüfen			
MatrNr	PersNr	VorlNr	Note
~~28106~~	~~5001~~	~~2126~~	~~1~~
~~25403~~	~~5041~~	~~2125~~	~~2~~
~~27550~~	~~4630~~	~~2137~~	~~2~~

Attacke ...

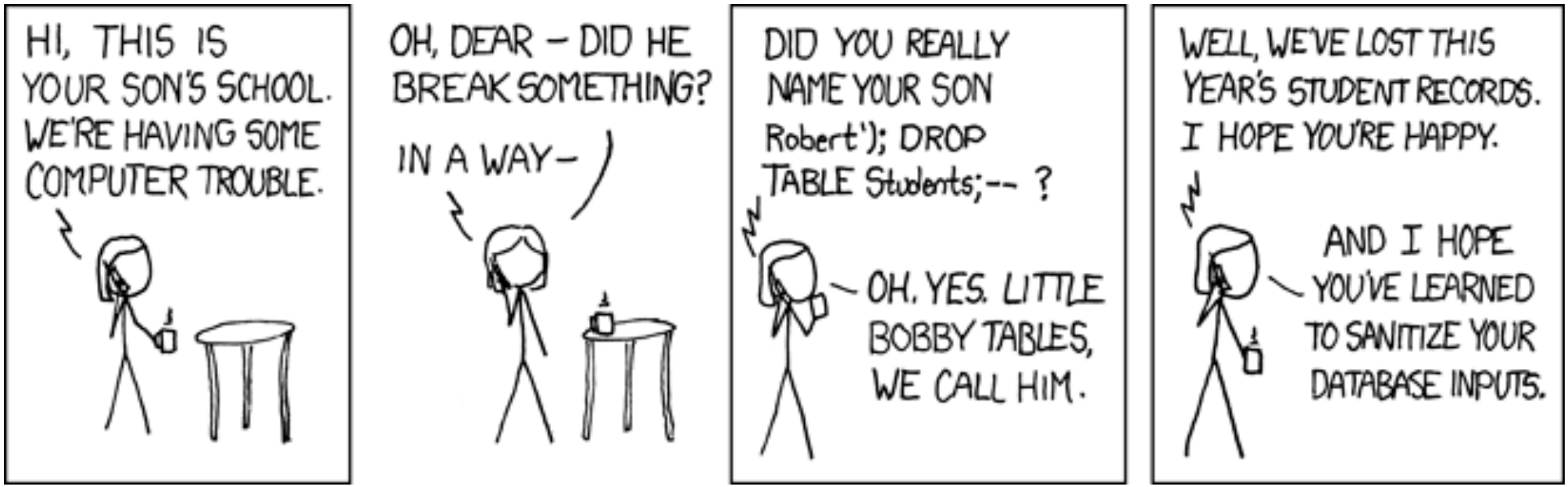
Name: Schopenhauer

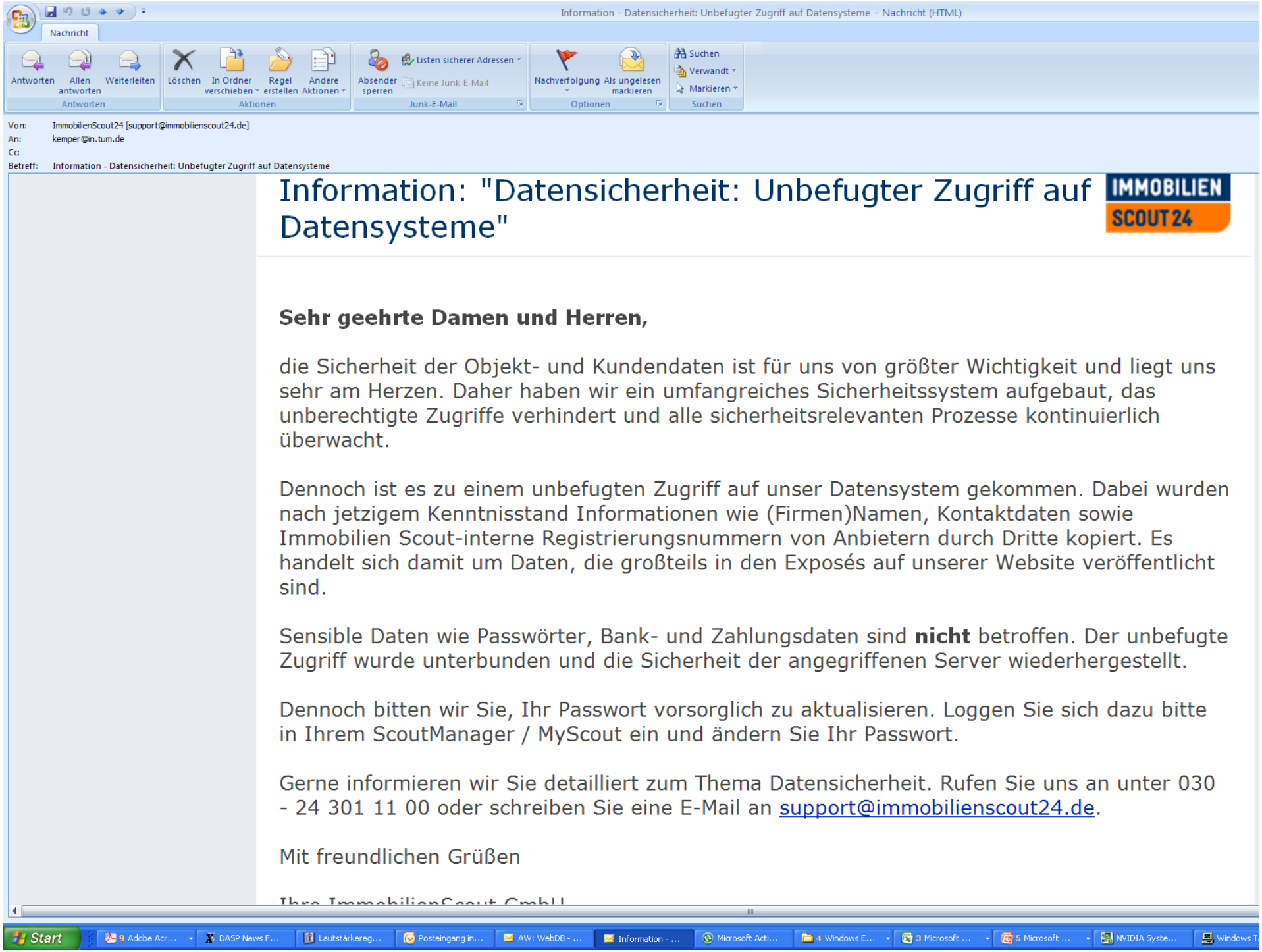
Passwort: Egal‘; **update** prüfen **set** Note = 1**where** MatrNr = 25403;

Select *
From Studenten s **join** prüfen p **on** s.MatrNr = p.MatrNr
Where s.Name = ‘Schopenhauer‘ **and**
s.Passwort = ‘Egal‘; **update** prüfen **set** Note = 1
where MatrNr = 25403**;**

prüfen			
MatrNr	PersNr	VorlNr	Note
28106	5001	2126	1
25403	5041	2125	~~2~~ 1
27550	4630	2137	2

Karikatur
Quelle: xkcd




Information - Datensicherheit: Unbefugter Zugriff auf Datensysteme - Nachricht (HTML)
Nachricht
Antworten
Allen antworten
Weiterleiten
Löschen
In Ordner verschieben
Regel erstellen
Andere Aktionen
Absender sperren
Listen sicherer Adressen
Keine Junk-E-Mail
Junk-E-Mail
Nachverfolgung
Als ungelesen markieren
Optionen
Suchen
Verwandt
Markieren
Suchen
Von: ImmobilienScout24 [support@immobilienscout24.de]
An: kemper@in.tum.de
Cc:
Betreff: Information - Datensicherheit: Unbefugter Zugriff auf Datensysteme
Information: "Datensicherheit: Unbefugter Zugriff auf Datensysteme"
IMMOBILIEN SCOUT24
Sehr geehrte Damen und Herren,
die Sicherheit der Objekt- und Kundendaten ist für uns von größter Wichtigkeit und liegt uns sehr am Herzen. Daher haben wir ein umfangreiches Sicherheitssystem aufgebaut, das unberechtigte Zugriffe verhindert und alle sicherheitsrelevanten Prozesse kontinuierlich überwacht.
Dennoch ist es zu einem unbefugten Zugriff auf unser Datensystem gekommen. Dabei wurden nach jetzigem Kenntnisstand Informationen wie (Firmen)Namen, Kontaktdaten sowie Immobilien Scout-interne Registrierungsnummern von Anbietern durch Dritte kopiert. Es handelt sich damit um Daten, die großteils in den Exposés auf unserer Website veröffentlicht sind.
Sensible Daten wie Passwörter, Bank- und Zahlungsdaten sind nicht betroffen. Der unbefugte Zugriff wurde unterbunden und die Sicherheit der angegriffenen Server wiederhergestellt.
Dennoch bitten wir Sie, Ihr Passwort vorsorglich zu aktualisieren. Loggen Sie sich dazu bitte in Ihrem ScoutManager / MyScout ein und ändern Sie Ihr Passwort.
Gerne informieren wir Sie detailliert zum Thema Datensicherheit. Rufen Sie uns an unter 030 - 24 301 11 00 oder schreiben Sie eine E-Mail an support@immobilienscout24.de.
Mit freundlichen Grüßen
Start
9 Adobe Acr...
DASP News F...
Lautstärkereg...
Posteingang in...
AW: WebDB - ...
Information - ...
Microsoft Acti...
4 Windows E...
3 Microsoft ...
5 Microsoft ...
NVIDIA Syste...
Windows T

Schutz vor SQL-Injection-Attacken

- **Prepared Statements**

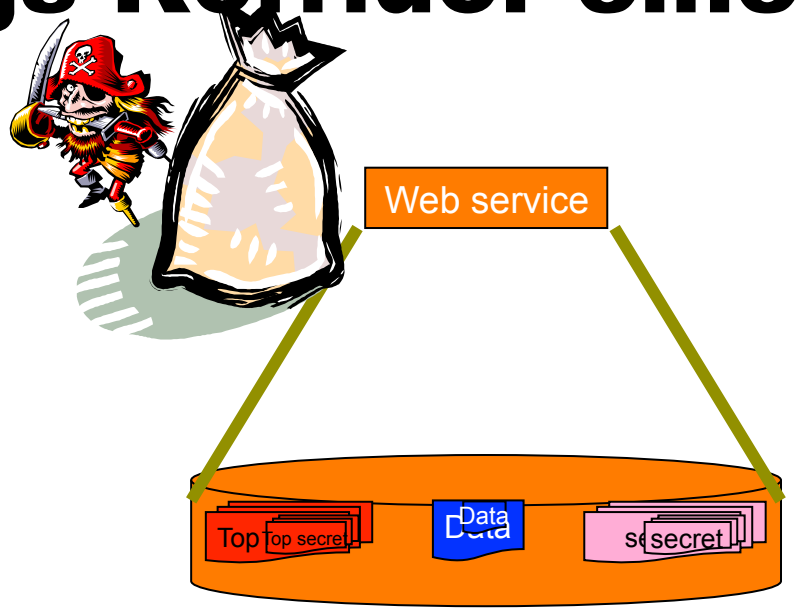
```
PreparedStatement stmt = conn.prepareStatement(
                    "select * from Vorlesungen v join Professoren p
                                        on v.gelesenVon = p.PersNr
                     where v.Titel = ? and p.Name = ? ");

String einzulesenderTitel = "Logik";
String einzulesenderName = "Sokrates";

stmt.setString(1, einzulesenderTitel);
stmt.setString(2, einzulesenderName);

ResultSet rs = stmt.executeQuery();
```

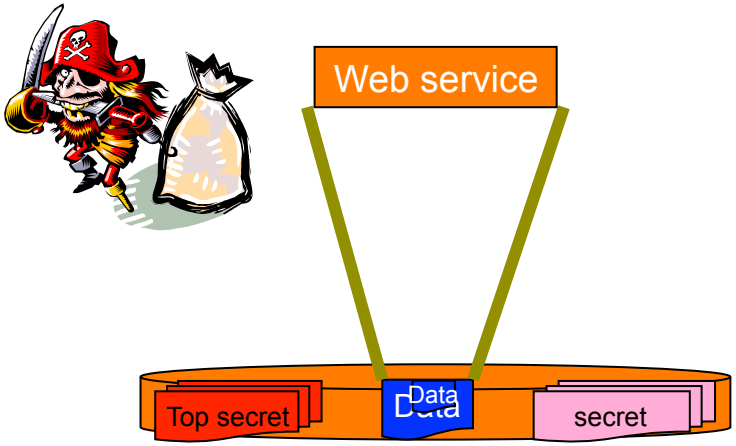
Schutz vor SQL-Injection-Attacken

- Filterung der Eingabe-Parameter

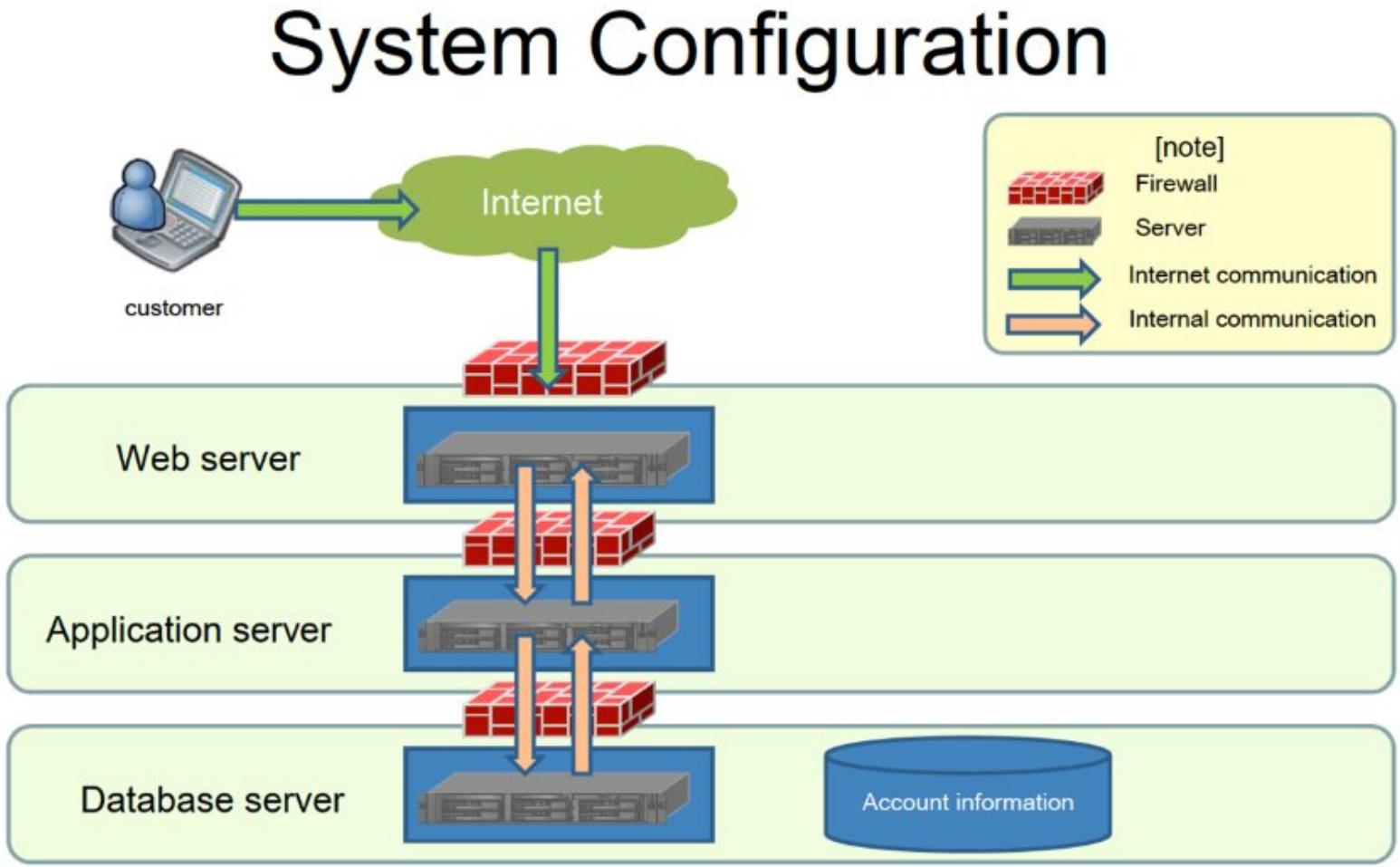
- Restriktive Autorisierungskorridore für die Anwendungen

Autorisierungs-Korridor einer Web-Anwendung

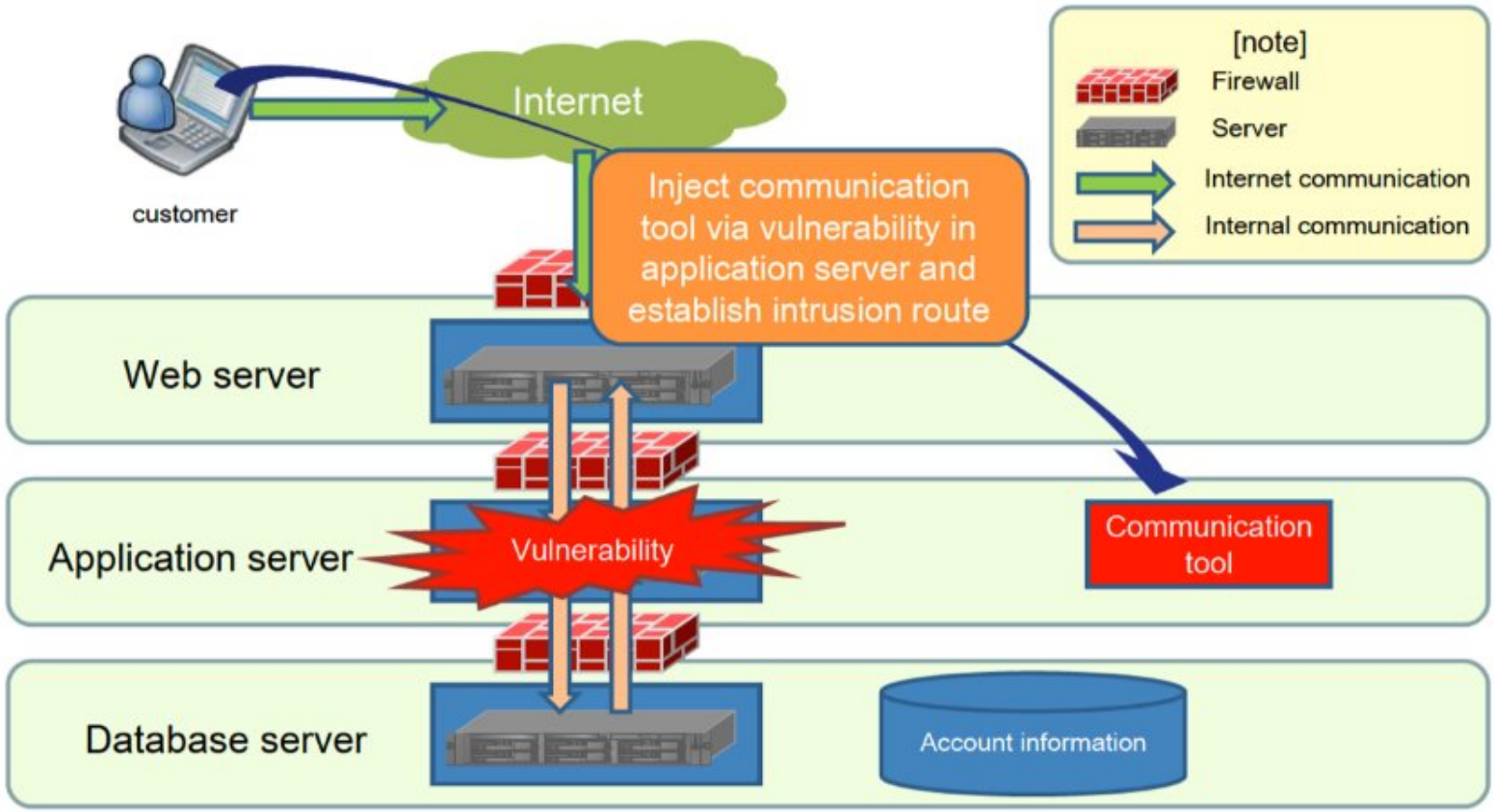




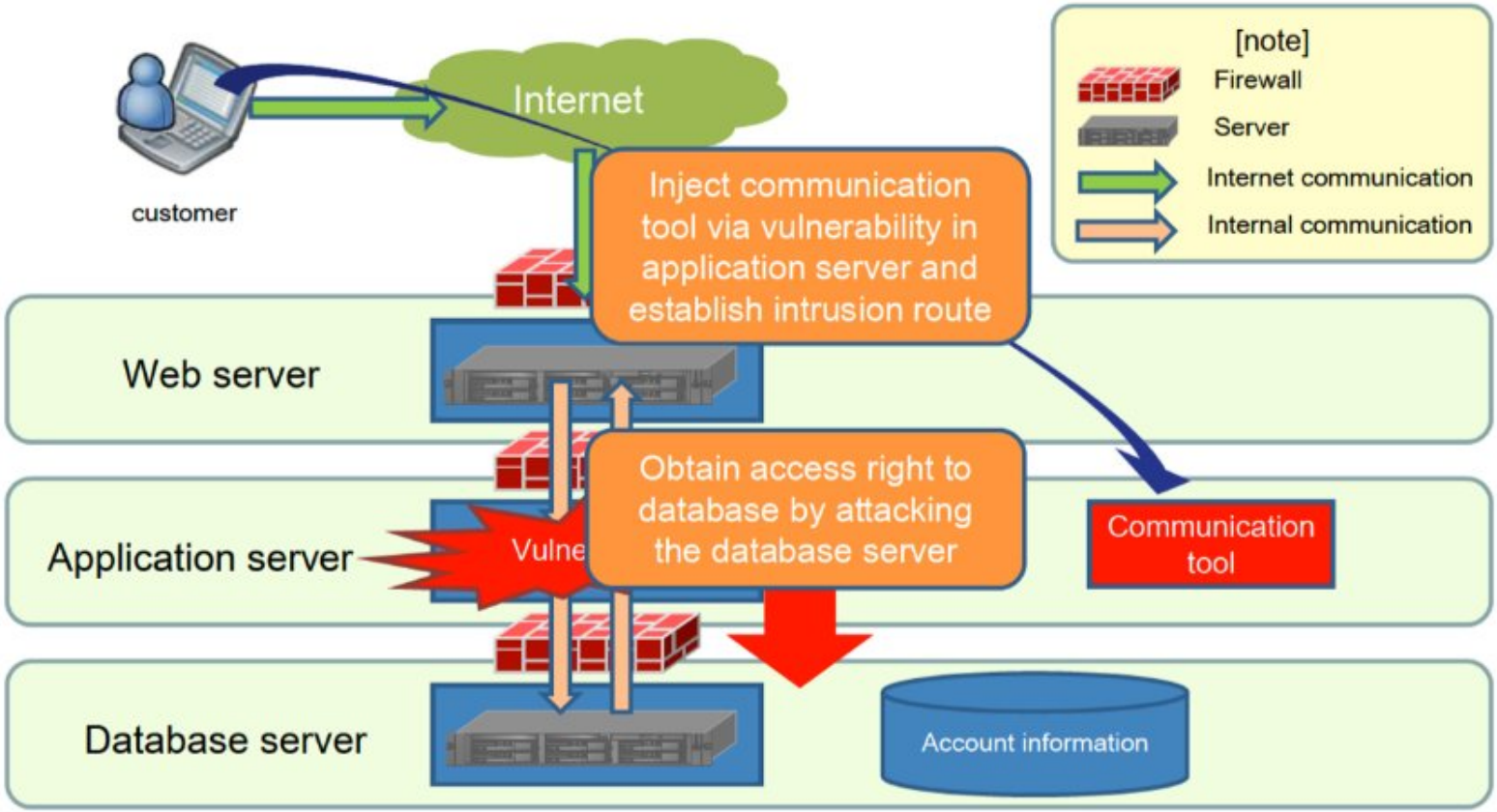
Sony Datendiebstahl
Quelle: Spiegel online



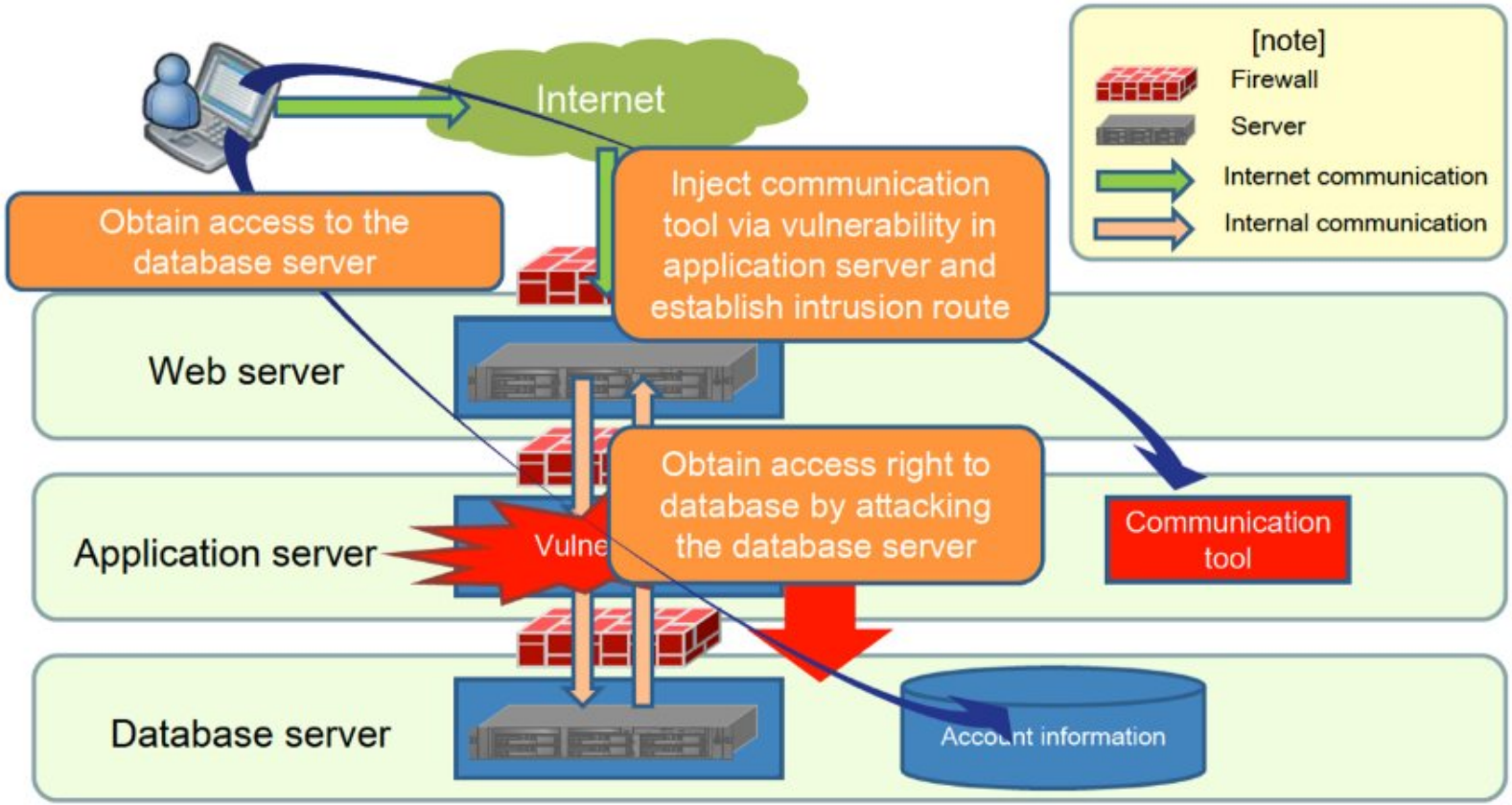
Intrusion route to the system



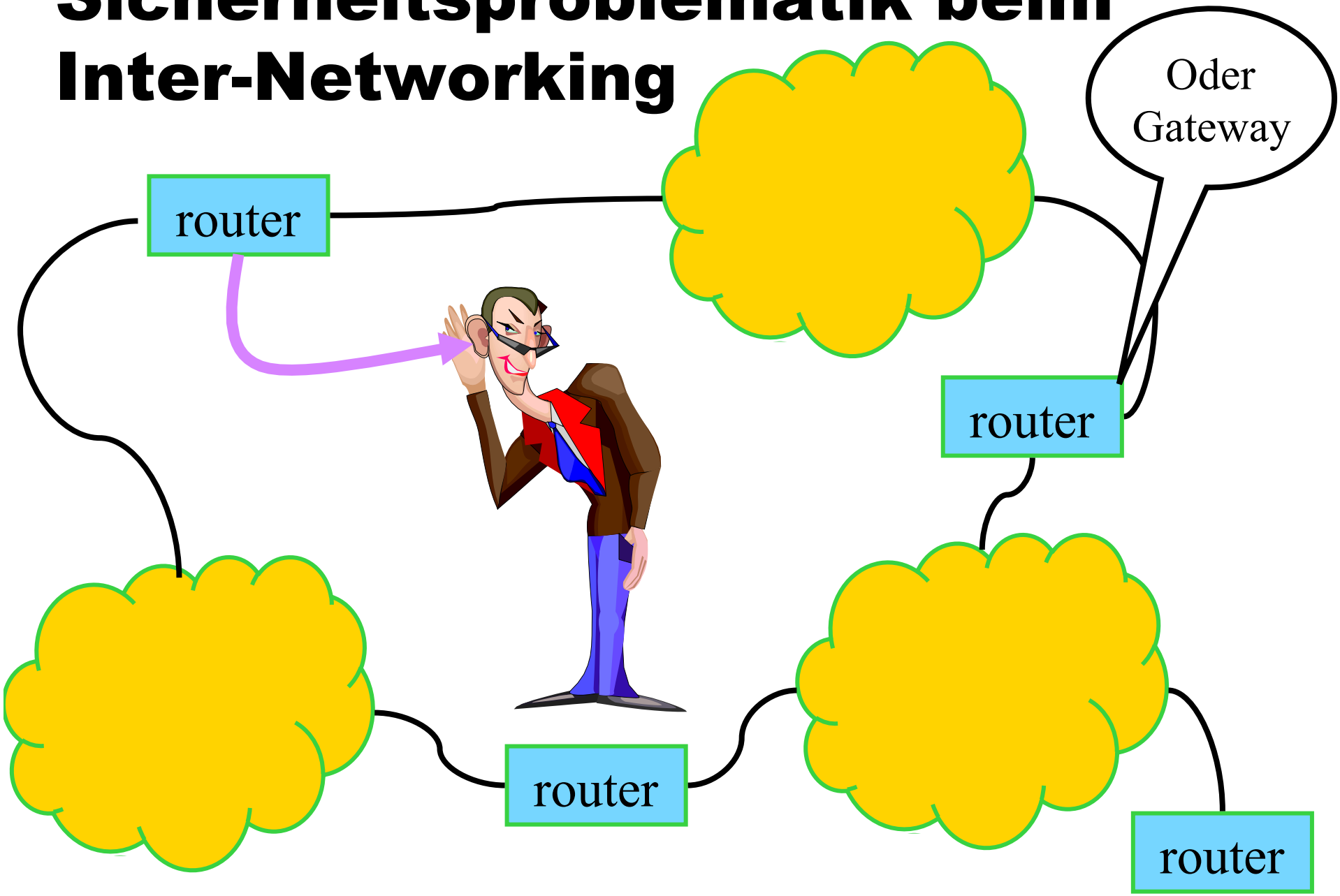
Intrusion route to the system



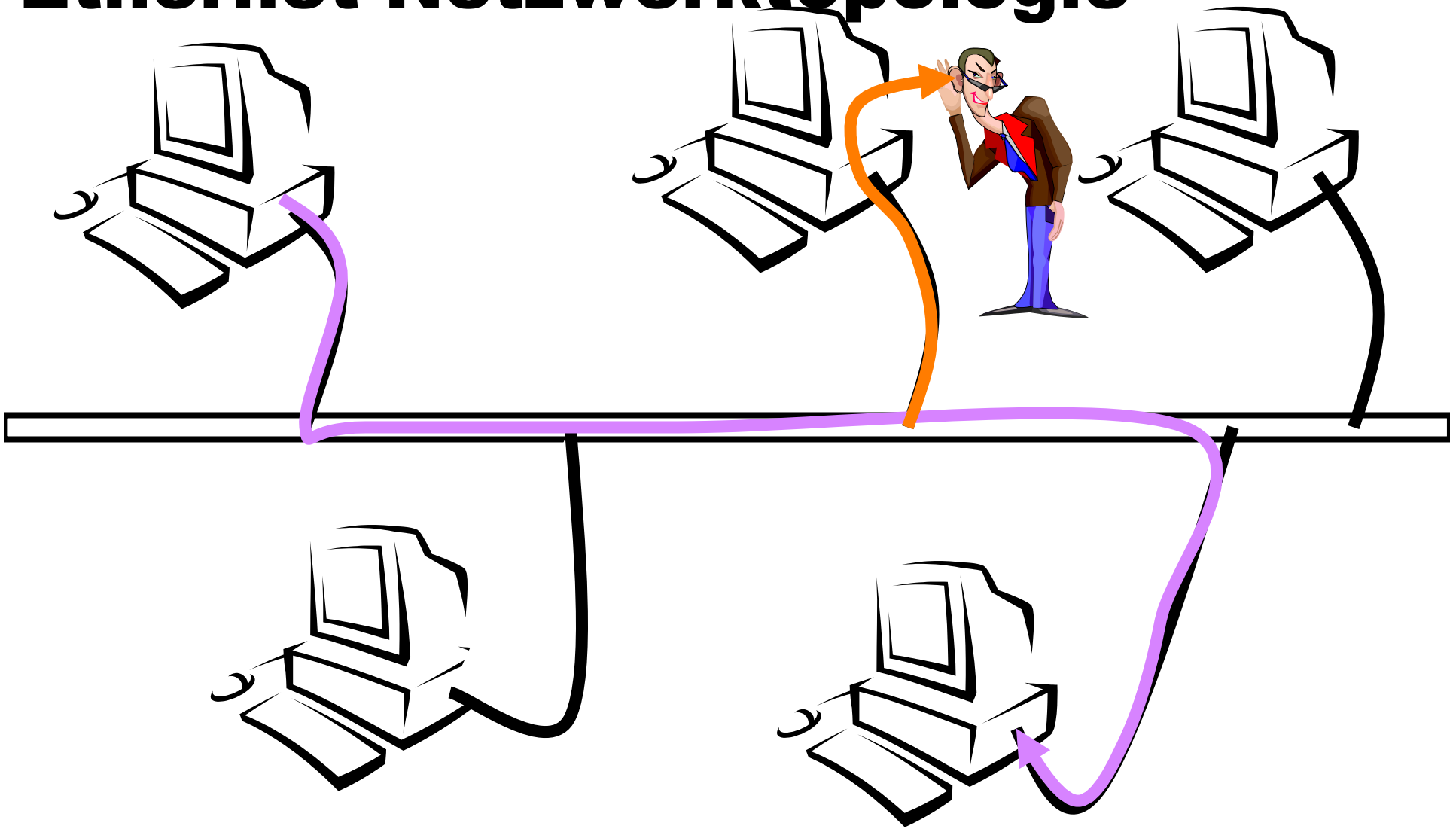
Intrusion route to the system



Sicherheitsproblematik beim Inter-Networking



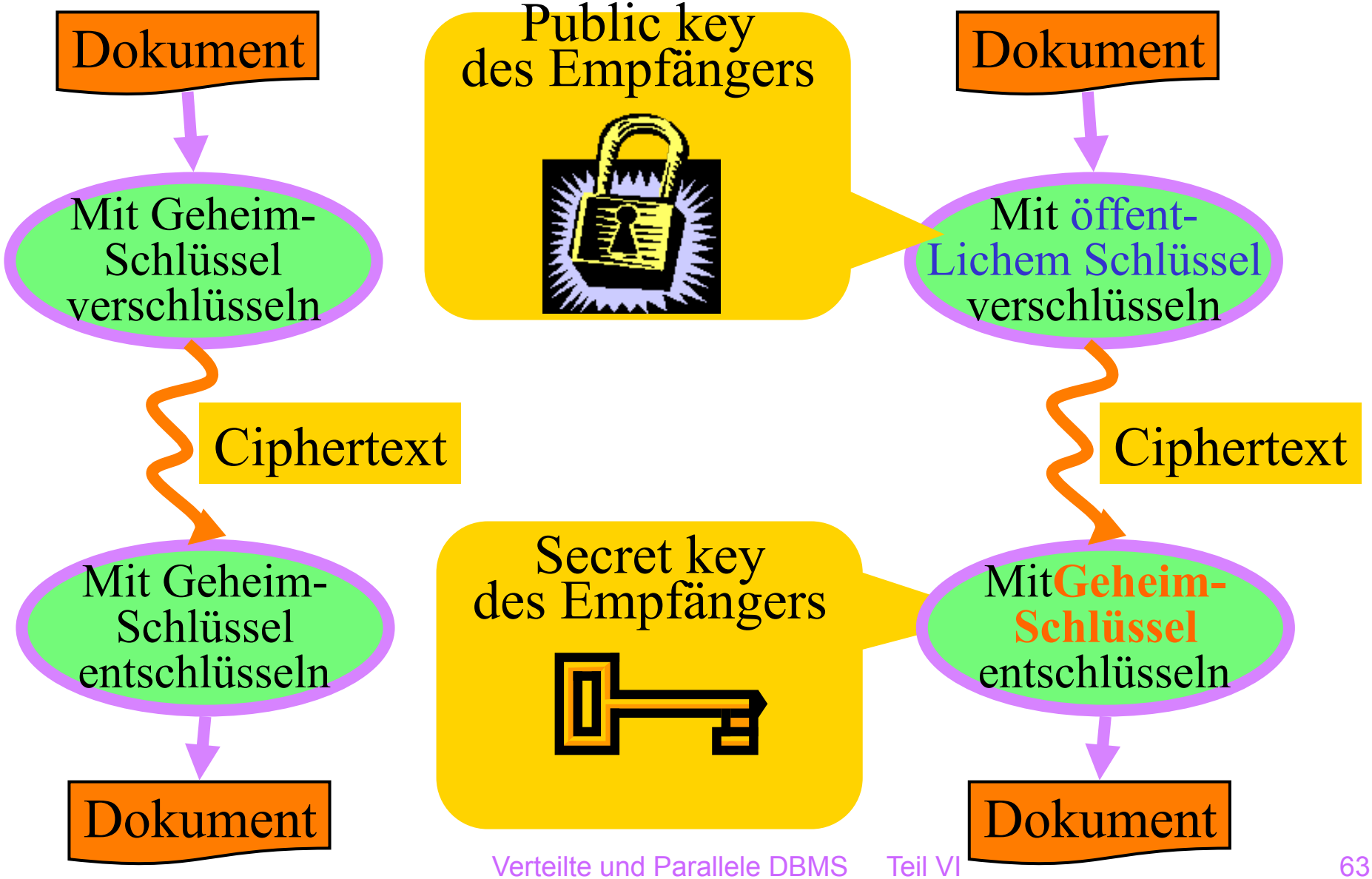
Ethernet-Netzwerktopologie



Sicherheitsaspekte

- Kryptographische Algorithmen
  - Symmetrische Verschlüsselungsverfahren
    - Geheime Schlüssel
    - zB DES
  - Asymmetrische Verschlüsselung (Public Key Kryptographie)
    - Geheimer Schlüssel
    - Öffentlicher Schlüssel
    - zB RSA
  - Message Digest
    - Digitaler Fingerabdruck
    - Wird in Verbindung mit dem Public Key-Verfahren zur digitalen Signatur verwendet
    - zB MD5
- Sicherheitsdienste
  - Privacy (Geheimhaltung, Diskretion)
  - Authentifizierung
  - Nachrichtenintegrität

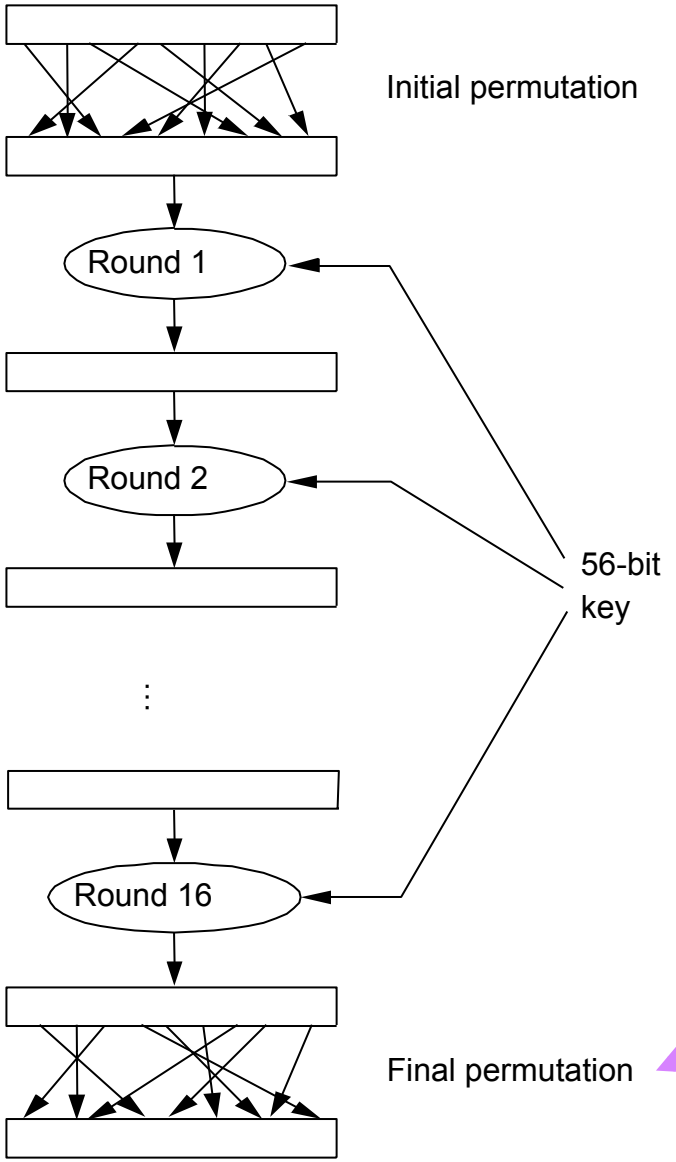
Geheimschlüssel vs Public Key



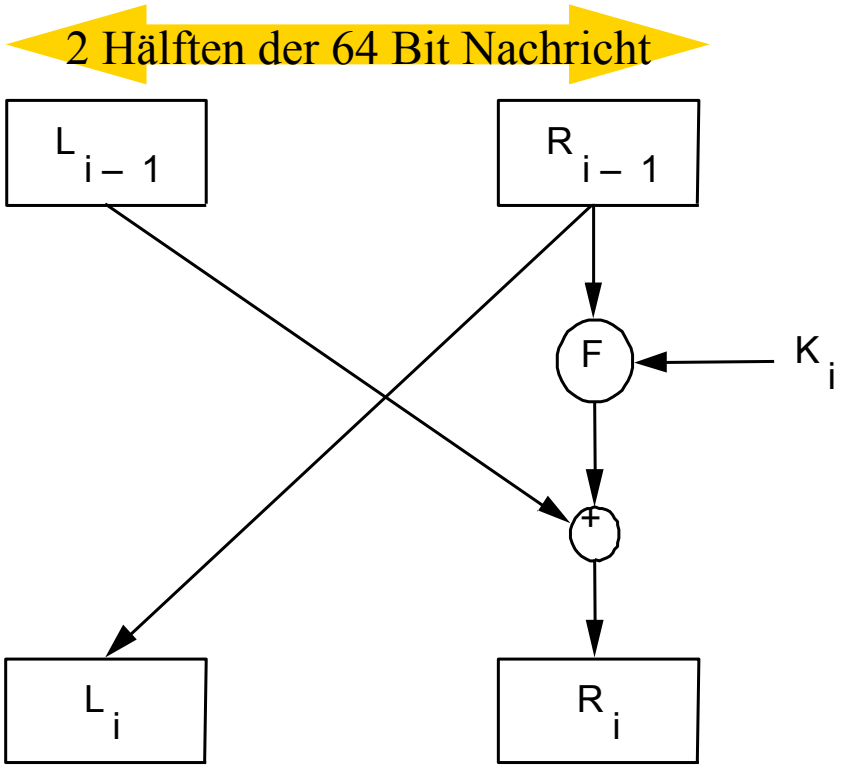
DES: Data Encryption Standard

- Geheimschlüssel/Symmetrisches Verfahren
- 64 Bit Geheimschlüssel (nur 56 werden benutzt; jedes achte Bit speichert die Parität der vorhergehenden 7 Bits)
- Es werden jeweils 64 Bit-Blöcke codiert (lange Nachrichten werden entsprechend aufgespalten)
- DES durchläuft drei Phasen bei der Verschlüsselung bzw. Entschlüsselung
  1. Die 64 Bit werden permutiert (Mischen)
  2. Sechzehn mal wird dieselbe Operation auf den Daten und dem Schlüssel angewendet
  3. Das Inverse der in 1. Durchgeführten Permutation wird angewendet

- 64-bit key (56-bits + 8-bit parity)
- 16 Runden

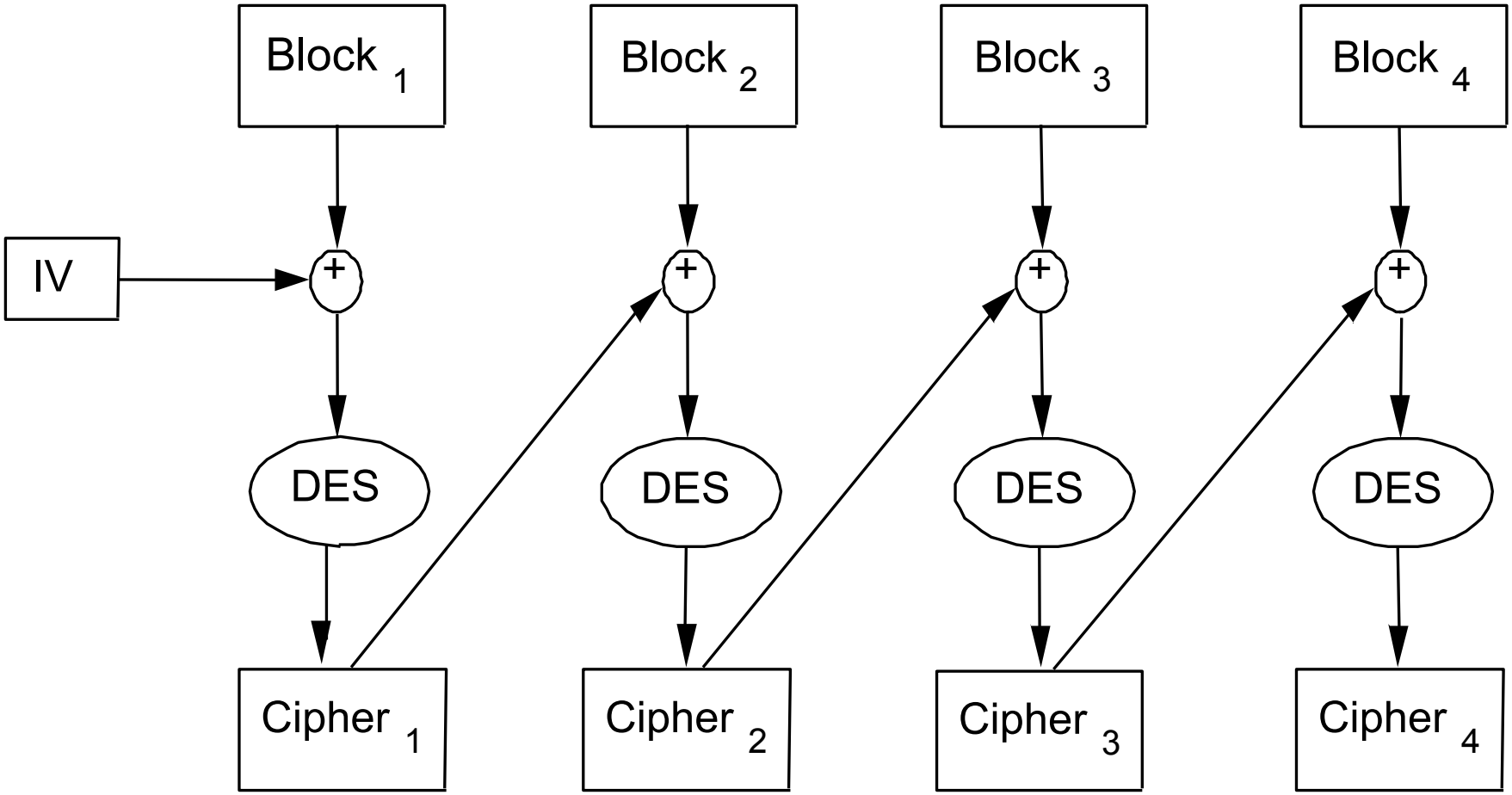


- jede Runde i



Permutationen sind
kein Sicherheitsgewinn
„Augenwischerei“

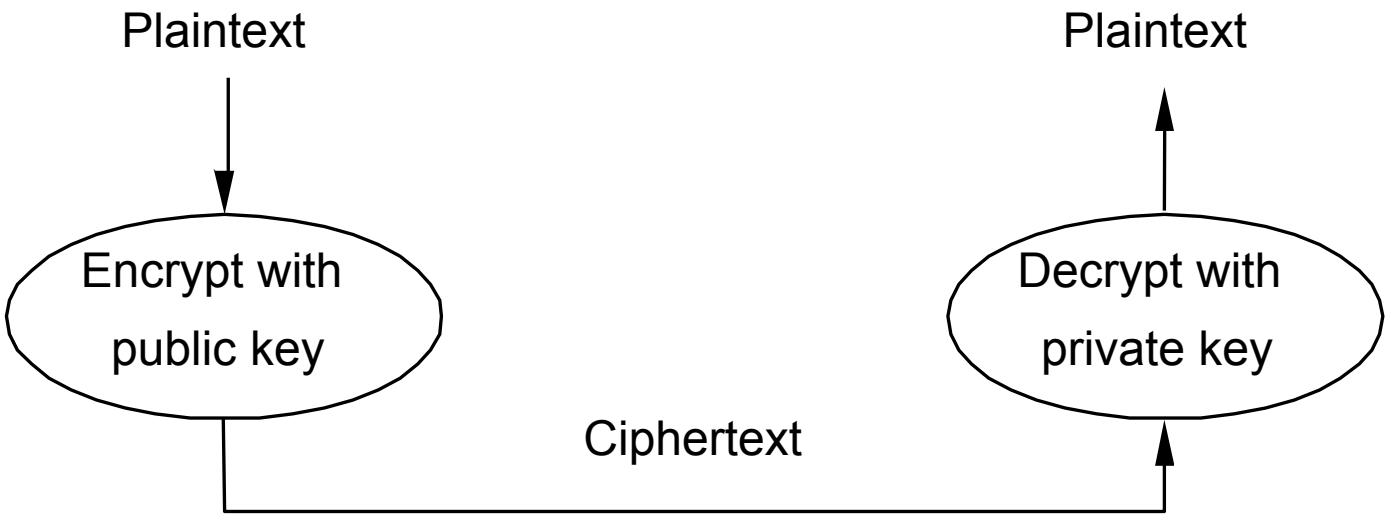
Verschlüsselung längerer Nachrichten



Sicherheit von DES

- Es gibt keinen mathematischen Beweis der Sicherheit von DES
- DES basiert auf „Konfusion und Diffusion“
- Offensichtlich Attacke
  - Versuche alle 2^{56} möglichen Schlüssel
  - Eine Verschlüsselung auf moderner Workstation dauert ca 4µs
  - Für die Hälfte aller 2^{56} möglichen Schlüssel benötigt man dann also ca 4500 Jahre
    - Mit 900 Computern aber nur 6 Monate
- Erhöhung der Sicherheit durch Triple-DES
  - 2 Schlüssel: erster und dritter DES-Durchgang mit demsleben Schlüssel
  - 3 unteschiedliche Schlüssel
  - Wird heute schon eingesetzt

Public Key (RSA)

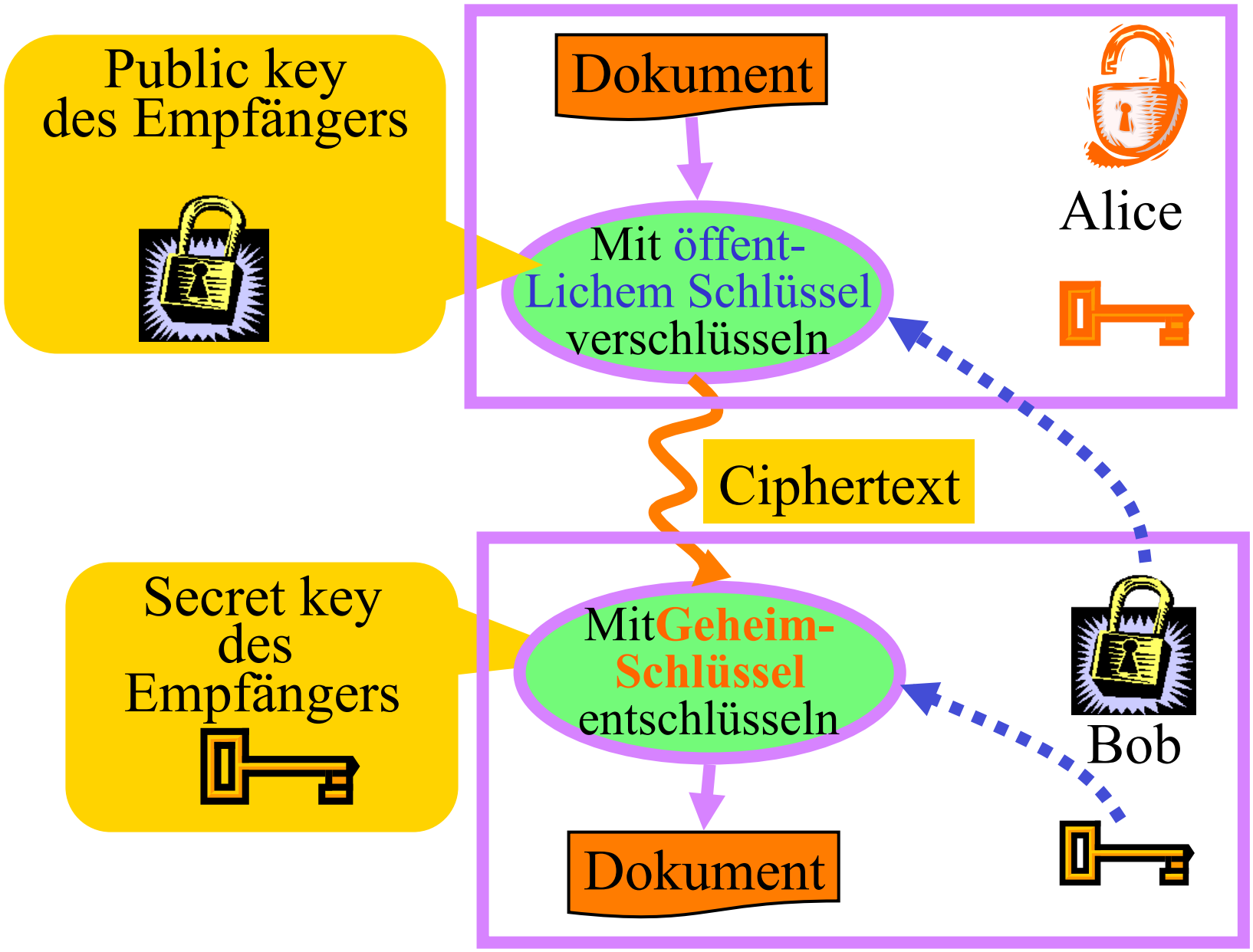


- Encryption & Decryption

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

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

Public Key Kryptographie: RSA



Falltür- ("*trap-door*")-Funktionen[1]

$$E, D : \{0, 1, \ldots, n-1\} \rightarrow \{0, 1, \ldots, n-1\}$$

(a) $D(E(M)) = M$

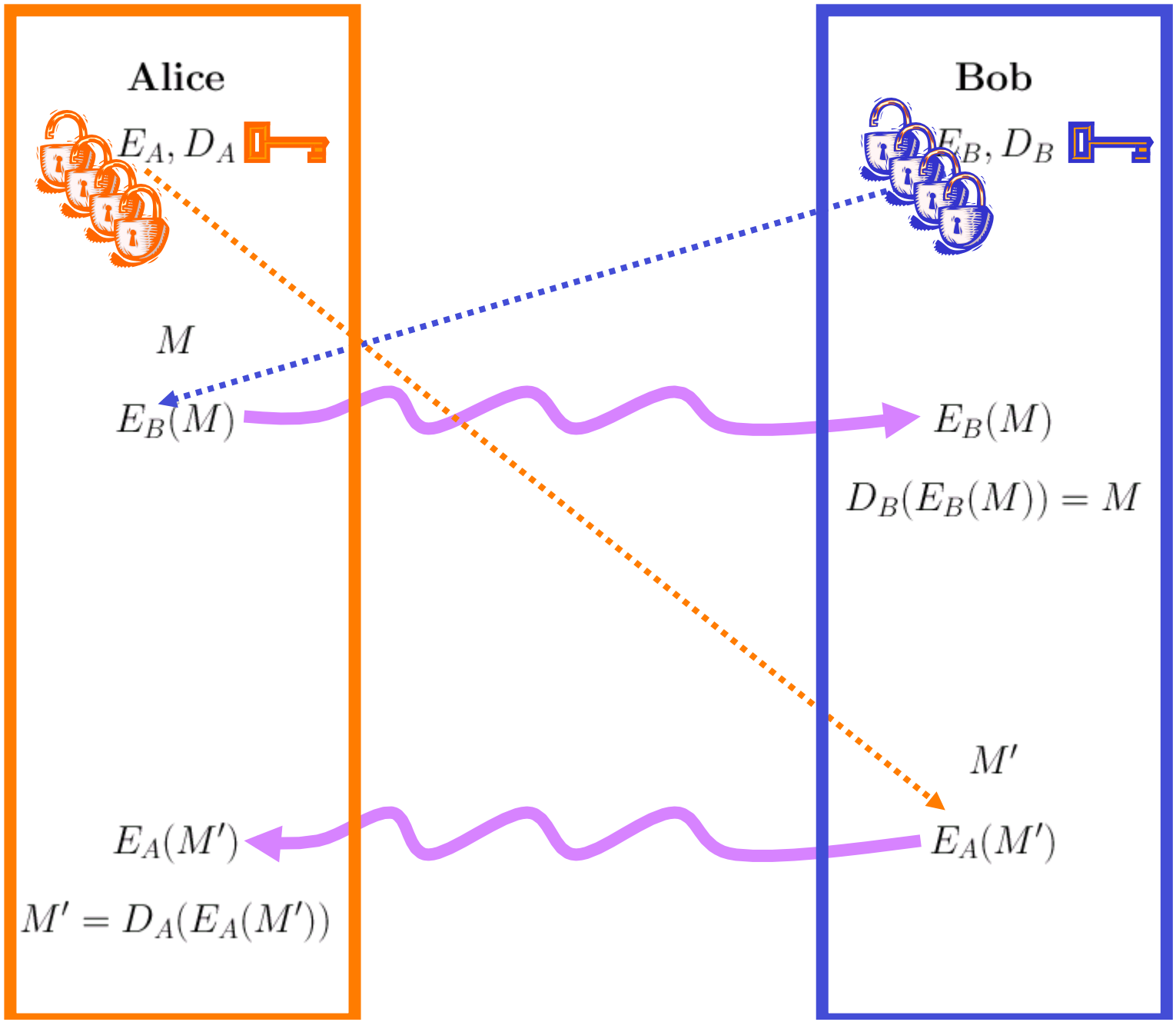
(b) D und E sind "einfach" zu berechnen

(c) es ist praktisch unmöglich (*infeasible*), D aus E herzuleiten

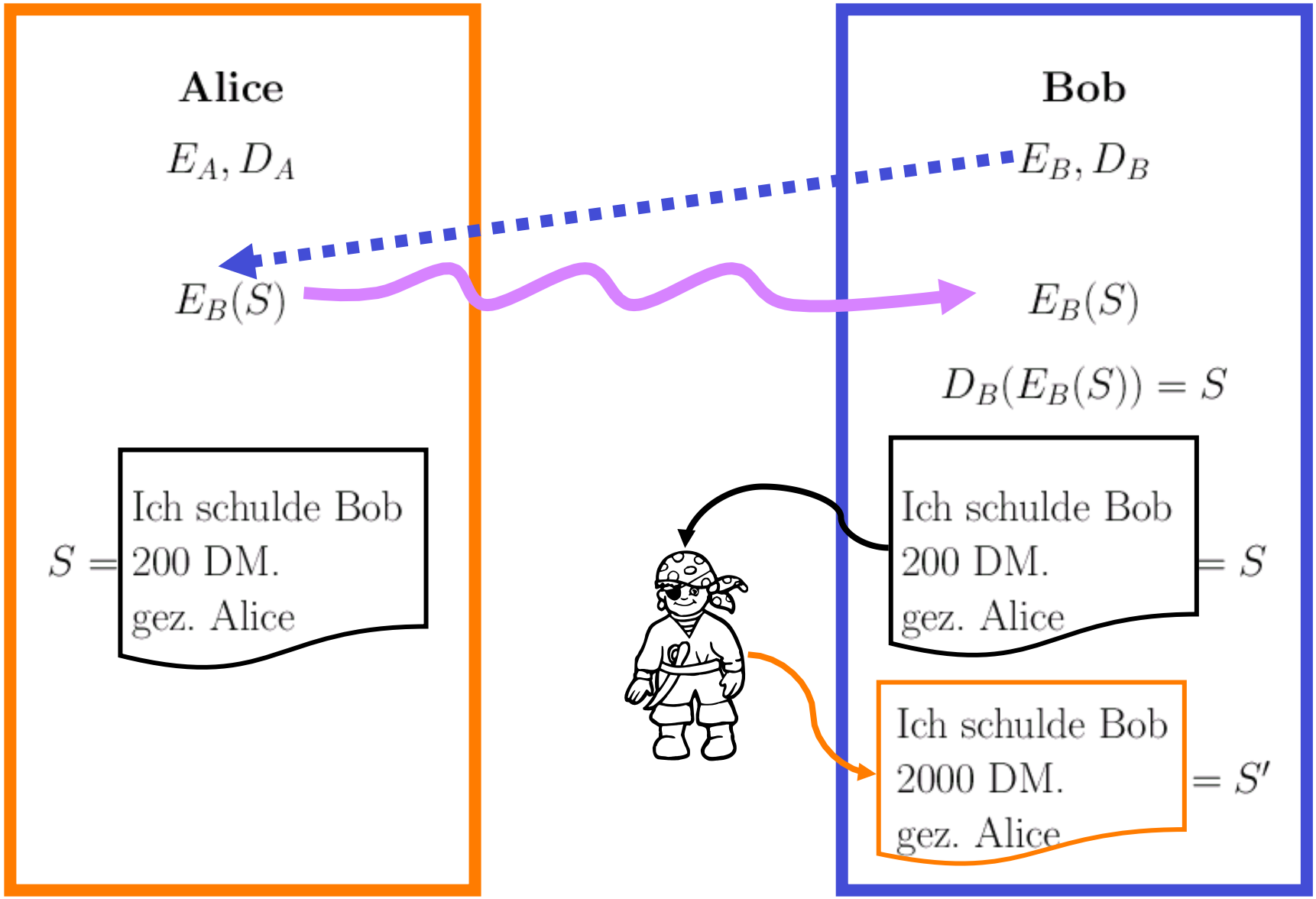
(d) $E(D(M)) = M$

- (a)–(c): "*trap-door one-way*"-Funktion
  - ausreichend für Verschlüsselung und anschließende Entschlüsselung
- (a)–(d): "*trap-door one-way*"-Permutation
  - notwendig für digitale Signaturen (siehe unten)

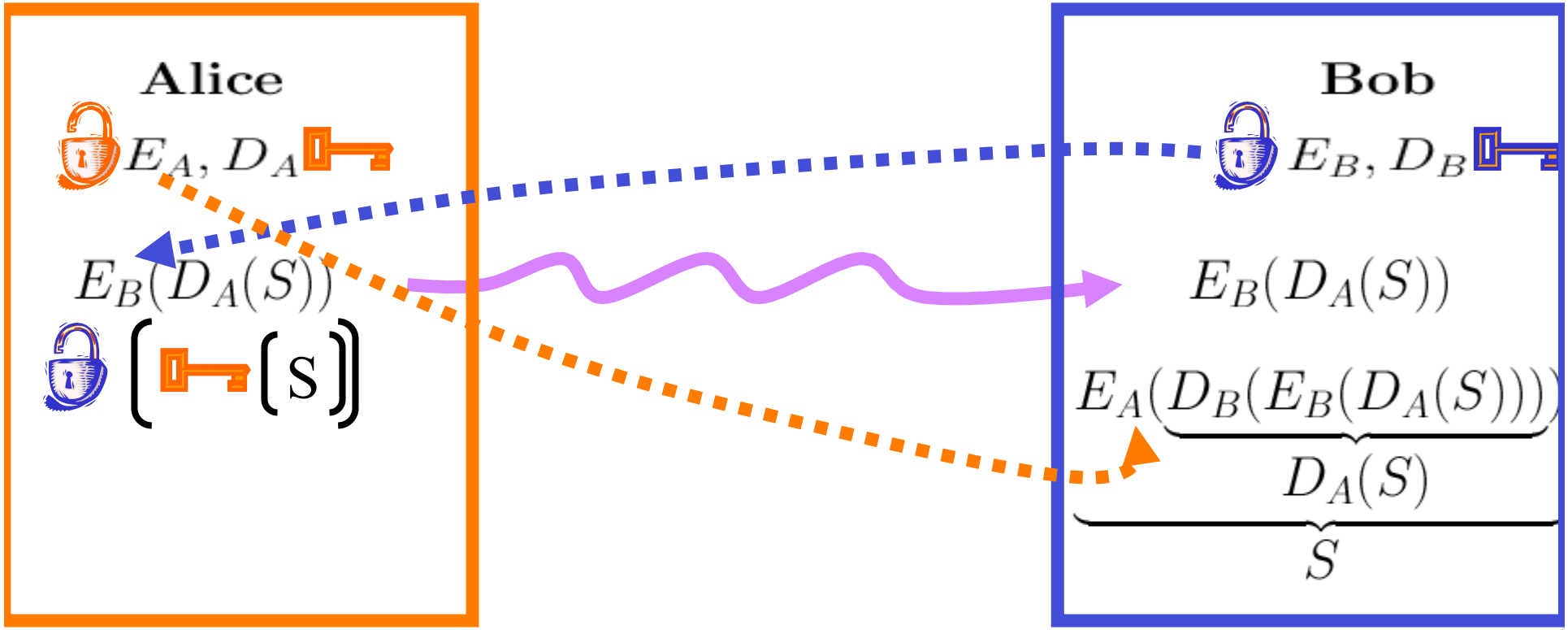
Austausch von Nachrichten



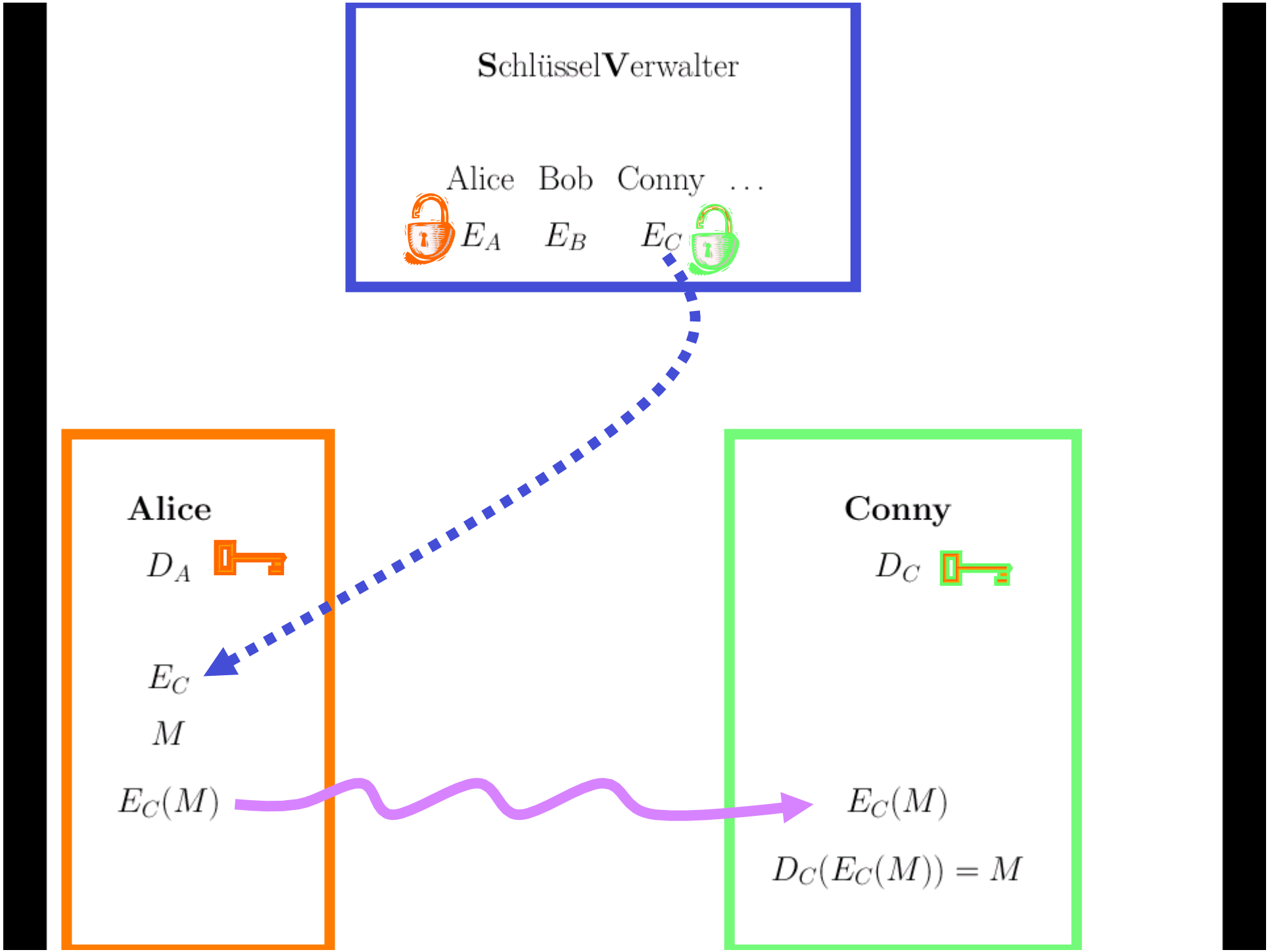
Motivation für Digitale Signaturen

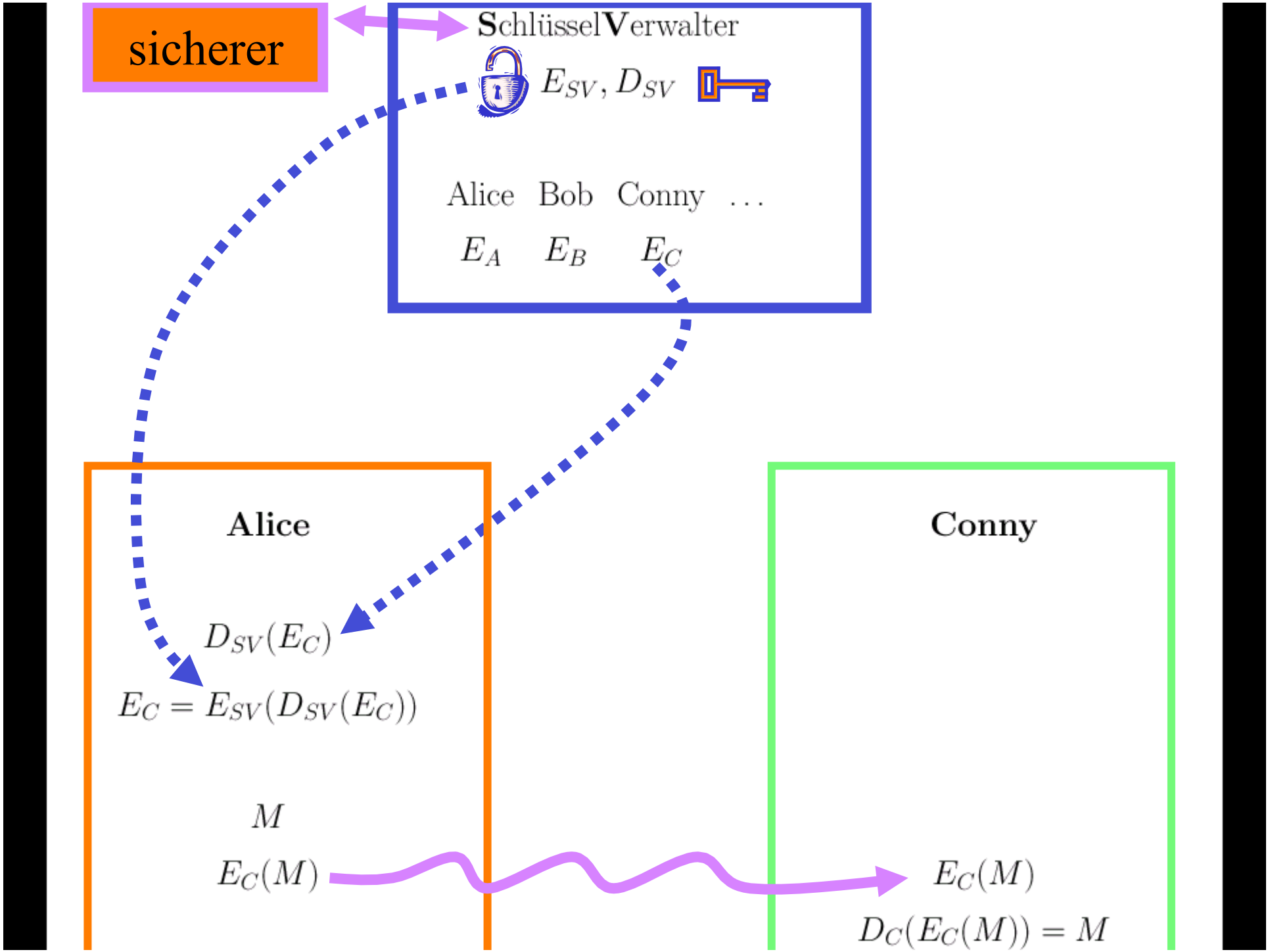


Digitale Signatur: Dokumenten-Integrität



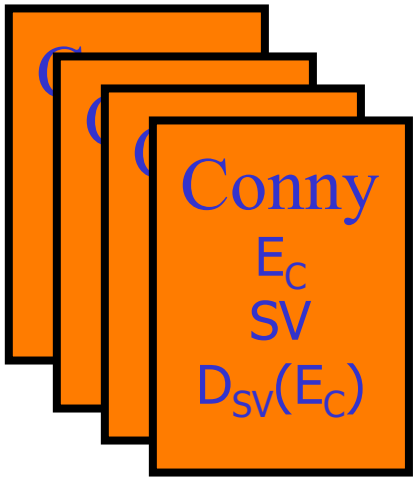
- Bob kann nicht zu S' das korrespondierende $D_A(S')$ generieren.
- Alice kann nicht bestreiten, S geschrieben zu haben
  - weil nur sie $D_A(S)$ erzeugt haben kann


SchlüsselVerwalter
Alice Bob Conny ...
E_A E_B E_C
Alice
D_A
E_C
M
$E_C(M)$
Conny
D_C
$E_C(M)$
$D_C(E_C(M)) = M$


sicherer
SchlüsselVerwalter
E_{SV}, D_{SV}
Alice Bob Conny ...
E_A E_B E_C
Alice
$D_{SV}(E_C)$
$E_C = E_{SV}(D_{SV}(E_C))$
M
$E_C(M)$
Conny
$E_C(M)$
$D_C(E_C(M)) = M$

Verwaltung und Verteilung der öffentlichen Schlüssel

- X.509 – Standard
- Digitale Zertifikate
- Certification Authorities (CA)
  - Banken,Telekom,Firmen (Verisign, ...)
  - Ein Zertifikat von CA X ist nur für den sinnvoll, der den öffentlichen Schlüsssel von X kennt
- Ein X.509 – Zertifikat enthält
  - Name der Organisation/Person: Conny
  - Öffentlichen Schlüssel: E_C
  - Name der Zertifizierungsautorität: SV
  - Digitale Signatur der CA: $D_{SV}(E_C)$
- Besitz eines Zertifikats sagt gar nichts aus
  - Zertifikate werden kopiert, gepuffert, etc.
  - Nur der Besitz des zugehörigen geheimen Schlüssels authentifiziert den rechtmäßigen Besitzer
- Hierarchie von CAs: X zertifiziert Y zertifiziert Z
  - Wenn ich X kenne kann ich dem Zertifikat für Z trauen, ohne Y zu kennen



Das RSA-Verfahren

- Rivest, Shamir und Adleman (1978)
- das älteste "*public key*"-Kryptographieverfahren
- beruht auf der "Erfahrung", daß Faktorisierung ein "hartes" Problem ist

- **Verschlüsselung einer Nachricht** M

öffentlicher Schlüssel: $(e, n) \approx E$

1. repräsentiere die Nachricht M als natürliche Zahl, so daß gilt: $0 \leq M \leq n-1$
   - längere Nachrichten sind entsprechend aufzuspalten
2. berechne $C = E(M) := M^e \bmod n$

- **Verschlüsselung einer Nachricht** M

$$\text{öffentlicher Schlüssel: } (e, n) \approx E$$

1. repräsentiere die Nachricht M als natürliche Zahl, so daß gilt: $0 \leq M \leq n - 1$
   - längere Nachrichten sind entsprechend aufzuspalten
2. berechne $C = E(M) := M^e \bmod n$

- **Entschlüsselung von** $C = E(M)$

$$\text{geheimer Schlüssel: } (d, n) \approx D$$

$$D(C) := C^d \bmod n$$

$$M^{14}$$

$$14 = \overbrace{(1110)}^{E}$$

$$(M^2 * M) \bmod n$$

$$\left(M^3\right)^2 * M \bmod n$$

$$\left(M^7\right)^2 = M^{14} \bmod n$$

- öffentlicher Schlüssel: (e, n)
- geheimer Schlüssel: (d, n)

Auswahl der Schlüssel

1. Wähle zwei (sehr große) Primzahlen p und q
   - mindestens 100-stellig
   - zufällig ausgewählt
     (etwa jede 115-te ungerade 100-stellige Zahl ist prim)
2. Berechne $n := p * q$
3. Wähle eine "große" Zahl d, für die gilt:

$$\mathrm{ggT}(d, \underbrace{(p-1)*(q-1)}_{\phi(n)}) = 1$$

   - man wähle z.B. eine Primzahl $d > \max(p, q)$
4. Berechne e, so daß gilt:

$$e * d \equiv 1 \pmod{(p-1)*(q-1)}$$

   - e ist das "multiplikative Inverse" von d
     (im *Ring* $\mathbf{Z}_{\phi(n)}, +_{\phi(n)}, *_{\phi(n)}$)
   - e existiert und ist eindeutig
     (weil $\mathrm{ggT}(d, \phi(n)) = 1$)

Mathematische Grundlagen

Definition 1 (Eulers ϕ-Funktion)

$$\phi(n) = \#\{i | i < n \text{ und } ggT(i,n) = 1\}$$

- Für eine Primzahl p gilt (natürlich):

$$\phi(p) = p - 1$$

- Für $n = p * q$ gilt:

$$\begin{aligned} \phi(n) &= \phi(p * q) \\ &= \phi(p) * \phi(q) \\ &= (p-1) * (q-1) \\ &= n - (p+q) + 1 \end{aligned}$$

Theorem 2 (Eulers Theorem)
Für alle $n > 1$ und alle $0 < M < n$ gilt:

$$ggT(M,n) = 1 \quad \Rightarrow \quad M^{\phi(n)} \equiv 1 \pmod{n}$$

Theorem 3 (Fermats "kleines" Theorem)
Für alle Primzahlen p und alle $0 < M < p$ gilt:

$$M^{(p-1)} \equiv 1 \pmod{p}$$

Existenz von e

Theorem 4

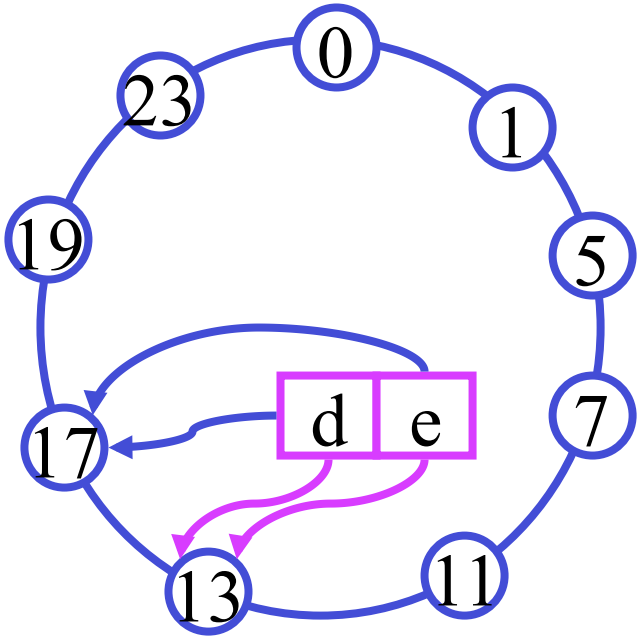
$$ggT(d, \phi(n)) = 1 \quad \Rightarrow \quad \exists e : e * d \equiv 1 \pmod{\phi(n)}$$

Beweis(idee)

- $(\mathbf{Z}^*_{\phi(n)}, *_{\phi(n)})$ ist eine (kommutative) Gruppe
  - $\mathbf{Z}^*_{\phi(n)} = \{a | 0 < a < \phi(n) \text{ und } \mathrm{ggT}(a, \phi(n)) = 1\}$
  - $a *_{\phi(n)} b = a * b \bmod \phi(n)$ für $a, b \in \mathbf{Z}^*_{\phi(n)}$
- Zu jedem Element einer Gruppe gibt es ein eindeutiges Inverse

Beispiel

- Seien folgende Primzahlen gewählt
  - p=5
  - q=7
- $\Phi(p*q) = \Phi(n) = 4*6 = 24$
- $Z^*_{\Phi(n)}$ ist dann wie folgt:



$13*13 = 1 \bmod 24$

$17*17 = 1 \bmod 24$

Berechnung von e

- $\mathrm{ggT}(a,b) = \min'(\{x*a+y*b | x,y \in \mathbf{Z}\})$
  wobei $\min'$ das kleinste *positive* Element auswählt
- $\mathrm{ggT}(\phi(n),d) = 1 \in \{x*\phi(n)+y*d | x,y \in \mathbf{Z}\}$
- Also existieren $x', y' \in \mathbf{Z}$:

$$\begin{aligned} x'*\phi(n)+y'*d &= 1 \\ y'*d &= 1 - x'*\phi(n) \\ y'*d &\equiv 1 \pmod{\phi(n)} \end{aligned}$$

Euklid(a,b) **is**
  if $b = 0$
    then return a
    else return *Euklid*$(b, a \bmod b)$;
end *Euklid*.

ErweiterterEuklid(a,b) **is**
  if $b = 0$
    then return $(a,1,0)$;
  $(r',x',y') :=$ *ErweiterterEuklid*$(b, a \bmod b)$;
  $(r,x,y) := (r', y', x' - \lfloor a/b \rfloor * y')$;
  return (r,x,y);
end *ErweiterterEuklid*.

- $(r,x,y) :=$ *ErweiterterEuklid*(a,b);
  - $r = x*a+y*b = \mathrm{ggT}(a,b)$
- $(\_,\_,e) :=$ *ErweiterterEuklid*$(\phi(n),d)$;

Beispielberechnung

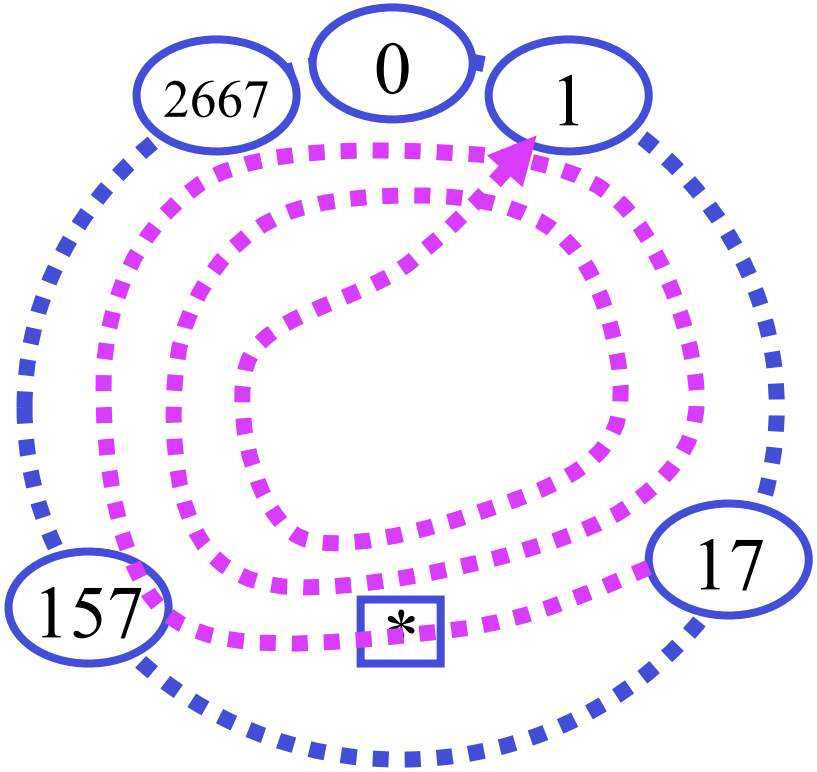
- Folgende Werte seien gewählt
  - p=47
  - q=59
  - n=p*q=2773
  - Φ(n)=46*58=2668
  - d=157

Euklid „in action“

a	b	r´	x´	y´	r	x	y
2668	157	1	1	-1	1	-1	17
157	156	1	0	1	1	1	-1
156	1	1	1	0	1	0	1
1	0				1	1	0

  - -1 * 2668 + 17*157 = 1
    - 17*157 = 1 + 2668
    - 17*157 = 1 mod 2668
    - Also ist 17 das multiplikative Inverse von 157 im Zahlenring

Illustration von e=157 und d=17 im Zahlenring Z^*_{2668}



Beispiel-Verschlüsselung

- Simple Codierung
  - 00 : blank
  - 01 : A
  - 02 : B
  - ...
  - 26 : Z
- p=47
- q=59
- n=p*q=2773
- e=17
- d=157

- ITS ALL GREEK TO ME
- 0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
- E(M) = M**17 mod n
- E(920) = 920^{17} mod 2773 = 948
- E(1900) = 2342
- 0948 2342 1084 1444 2663 2390 0778 0774 0219 1655
- D(E(M) = E(M)^{157} mod n
- D(948) = 948**157 mod 2773 = 0920
- D(2342) = 1900
- 0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
- ITS ALL GREEK TO ME

Korrektheit von RSA

(a) $D(E(M)) \equiv (E(D(M)) \equiv (M^e)^d \equiv M^{e*d} \pmod{n}$

(d) $E(D(M)) \equiv (D(M))^e \equiv (M^d)^e \equiv M^{e*d} \pmod{n}$

$n = p * q$

$\phi(n) = (p-1) * (q-1)$

$e * d \equiv 1 \pmod{\phi(n)}$

$e * d = 1 + k * (p-1) * (q-1)$

Falls $M \not\equiv 0 \pmod{p}$ gilt:

$$\begin{aligned} M^{e*d} &\equiv M * (M^{p-1})^{k*(q-1)} && (\pmod{p}) \\ &\equiv M * (1)^{k*(q-1)} && (\pmod{p}) \\ &\equiv M && (\pmod{p}) \end{aligned}$$

Fermats Theorem
$M^{(p-1)} \equiv 1 \pmod{p}$

Falls $M \equiv 0 \pmod{p}$ gilt (trivialerweise) auch:

$M^{e*d} \equiv M \pmod{p}$

Also: $M^{e*d} \equiv M \pmod{p}$ für alle M

Analog: $M^{e*d} \equiv M \pmod{q}$ für alle M

$M^{e*d} \equiv M \pmod{n}$ für alle M. [chinesischer Restsatz]

- Also sind E und D inverse Permutationen!

- Berechnung von $M^e \bmod n$ bzw. $C^d \bmod n$
- modulare Exponentiation durch wiederholte Quadrierung

ModulareExponentiation(M, e, n) **is**
  $c := 0;$
  $r := 1;$
  $E := [e_k, e_{k-1}, \ldots, e_1, e_0];$!! binäre Repräsentation von e
  for $i := k$ **downto** 0 **do**
    begin
      $c := 2 * c;$
      $r := (r * r) \bmod n;$
      if $E[i] = 1$ **then**
        begin
          $c := c + 1;$
          $r := (r * M) \bmod n;$
        end
    end
  return $r;$
end *ModulareExponentiation*.

$$M^{14}$$

$$14 = \overbrace{(1110)}^{E}$$

$$(M^2 * M) \quad \bmod n$$

$$\left(M^3\right)^2 * M \quad \bmod n$$

$$\left(M^7\right)^2 = M^{14} \quad \bmod n$$

- Korrektheit

$$M^{2c} \bmod n = (M^c)^2 \bmod n$$
$$M^{2c+1} \bmod n = M * (M^c)^2 \bmod n$$

- Komplexität
  - $O(\beta^3)$

Probabilistische Primzahl-Bestimmung

1. Rate eine willkürliche ungerade Zahl p (entsprechend groß, z.B. 100 Stellen)
2. Wähle willkürlich $m = 50$ Zahlen (sogenannte Zeugen) $\{x_1, x_2, \ldots, x_m\}$ zwischen 1 und p
3. für jede dieser Zahlen $x_i (1 \leq i \leq m)$ führe die beiden folgenden Tests durch:
   (a) Gilt $\mathrm{ggT}(x_i, p) = 1$? Wenn nein, dann ist p keine Primzahl und fange bei Schritt 1. neu an.
   (b) Gilt $J(x_i, p) \equiv x_i^{(b-1)/2}(\quad \pmod p)$? Wenn nein, dann ist p keine Primzahl und fange bei Schritt 1. neu an.

$J(x_i, p)$ ist das Jacobi-Symbol.

- Für eine nicht-prime Zahl p sind mehr als die Hälfte aller Zahlen zwischen 1 und p Zeugen dafür, daß p nicht-prim ist.
- Deshalb ist für jedes x_i die Wahrscheinlichkeit, daß x_i eine nicht-prime Zahl p als zusammengesetzt "entlarvt", gleich 1/2.
- Daraus folgt daß der obige Algorithmus mit einer Wahrscheinlichkeit von $1 - (1/2)^m$ korrekte Primzahlen ausgibt.

Jakobi-Symbol („wen´s interessiert“)

$$J(a,b) = \begin{cases} 1 \text{ falls } a = 1 \\ J(a/2,b) * -1^{(b^2-1)/8} & \text{falls a gerade} \\ J(b \bmod a, a) * -1^{(a-1)(b-1)/4} & \text{sonst} \end{cases}$$

Sicherheit von RSA

- es gibt prinzipiell keine Möglichkeit, die Sicherheit von Kryptographiesystemen zu beweisen
- RSA basiert auf der Komplexität, große Zahlen zu faktorisieren
  - alle offensichtlichen Methoden, RSA zu "brechen", sind äquivalent zur Faktorisierung von n
- Aber: es konnte (noch) nicht gezeigt werden, daß Faktorisierung ein NP-vollständiges Problem ist
- es gibt auch keinen Beweis dafür, daß man RSA nur "brechen" kann, wenn man die Faktoren von n (also p und q) kennt
- andererseits, nach Faktorisierung von $n = p * q$ kann man $\phi(n) = (p-1)(q-1)$ sehr leicht berechnen
- Faktorisierung ist aber z.Zt. nicht praktikabel für 100 – 200-stellige Zahlen
- Faktorisierung ist ein intensiv erforschtes Problem
  - Fermat (1601 – 1665)
  - Legendre (1752 – 1833)

Sicherheit von RSA – Herausforderung

- 1977 haben Rivest, Shamir und Adleman die Herausforderung gestellt, eine Nachricht zu entschlüsseln, die mit einem 129-stelligen Schlüssel (also 428 Bits) verschlüsselt wurde
- Damals glaubte man, dass dieser Code nicht zu brechen sei
  - 40 Quadrillion Jahre hätte es mit damaligen Faktorisierungsalgorithmen auf damaliger Hardware gedauert
- 1994 wurde diese Nachricht tatsächlich entschlüsselt
  - Dazu wurde die Faktorisierung auf Tausende von Rechnern im Internet parallelisiert
  - Der Rechenaufwand betrug ca 5000 MIP-Jahre
  - Es kann natürlich sein, dass der Algorithmus Glück hatte (zufällig günstige Primzahl-Konfiguration)
- Wortlaut der Nachricht: THE MAGIC WORDS ARE SQUEAMISH AND OSSIFRAGE
- Heute werden 512 Bit RSA-Schlüssel verwendet (jetzt, nach Lockerung der US-Ausfuhrbeschränkungen auch schon 768 oder 1024 Bit)

Sicherheit von RSA – Herausforderung (cont´d)

- Die zwei 64 bzw 65-stelligen Primzahlen
  - P=3490529510847650949147849619903898133417764638493387843990820577
  - Q=32769132993266709549961988190834461413177642967952942539798288533
- N = P * Q
  - 129-stellig (also 428 Bits)
  - Wurde 1993 von ca 600 Stationen im Internet innerhalb von einem Jahr faktorisiert
- Heute werden standardmäßig 512 Bit RSA-Schlüssel verwendet (jetzt, nach Lockerung der US-Ausfuhrbeschränkungen auch schon 768 oder 1024 Bit)

Andere Ansätze, RSA zu "brechen"

- **Berechnung von $\phi(n)$ ohne Faktorisierung**
  - $e * d \equiv 1 \pmod{\phi(n)}$
    - Wenn e und $\phi(n)$ bekannt sind, kann d berechnet werden (*ErweiterterEuklid*).
  - Berechnung von $\phi(n)$ so schwer wie Faktorisierung:

$$\begin{aligned} \phi(n) &= n - (p+q) + 1 \\ p - q &= \sqrt{(p+q)^2 - 4n} \\ \text{Also:} \quad q &= 1/2((p+q) + (p-q)) \end{aligned}$$

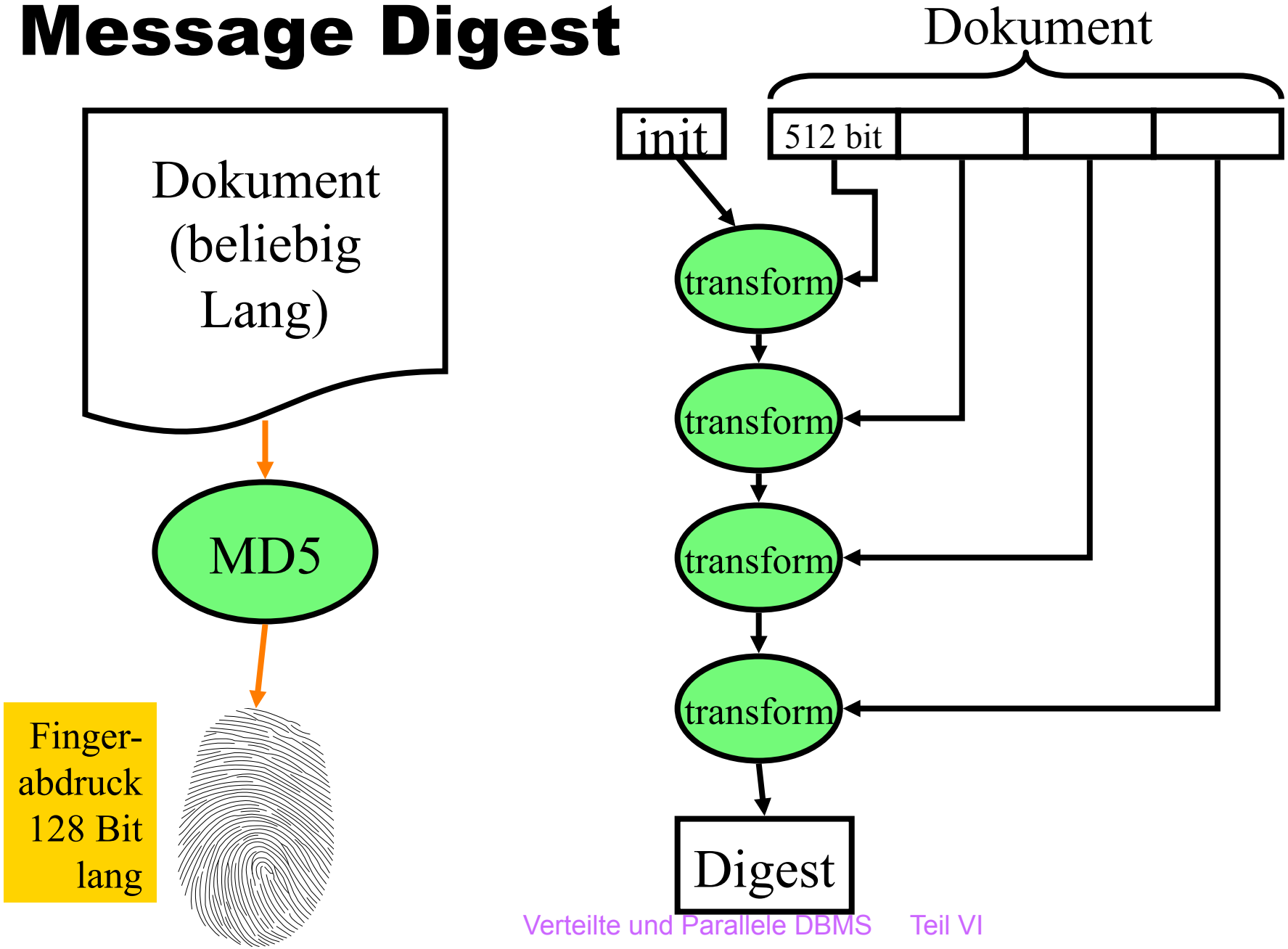
  - deshalb darf n keine Primzahl sein, da sonst die Berechnung von $\phi(n)$ trivial wäre.

- **Berechnung von d**
  - ohne Faktorisierung von n oder Bestimmung von $\phi(n)$
  - $e * d - 1$ ist ein Vielfaches von $\phi(n)$
  - es gibt effiziente Faktorisierungsalgorithmen für n, wenn ein Vielfaches von $\phi(n)$ bekannt ist
- **Berechnung eines d', das äquivalent zu d ist**
  - sehr selten
  - liegen im Abstand von $\mathrm{lcm}(p-1, q-1)$ (least common multiplier)
  - erschöpfende Suche also nicht möglich

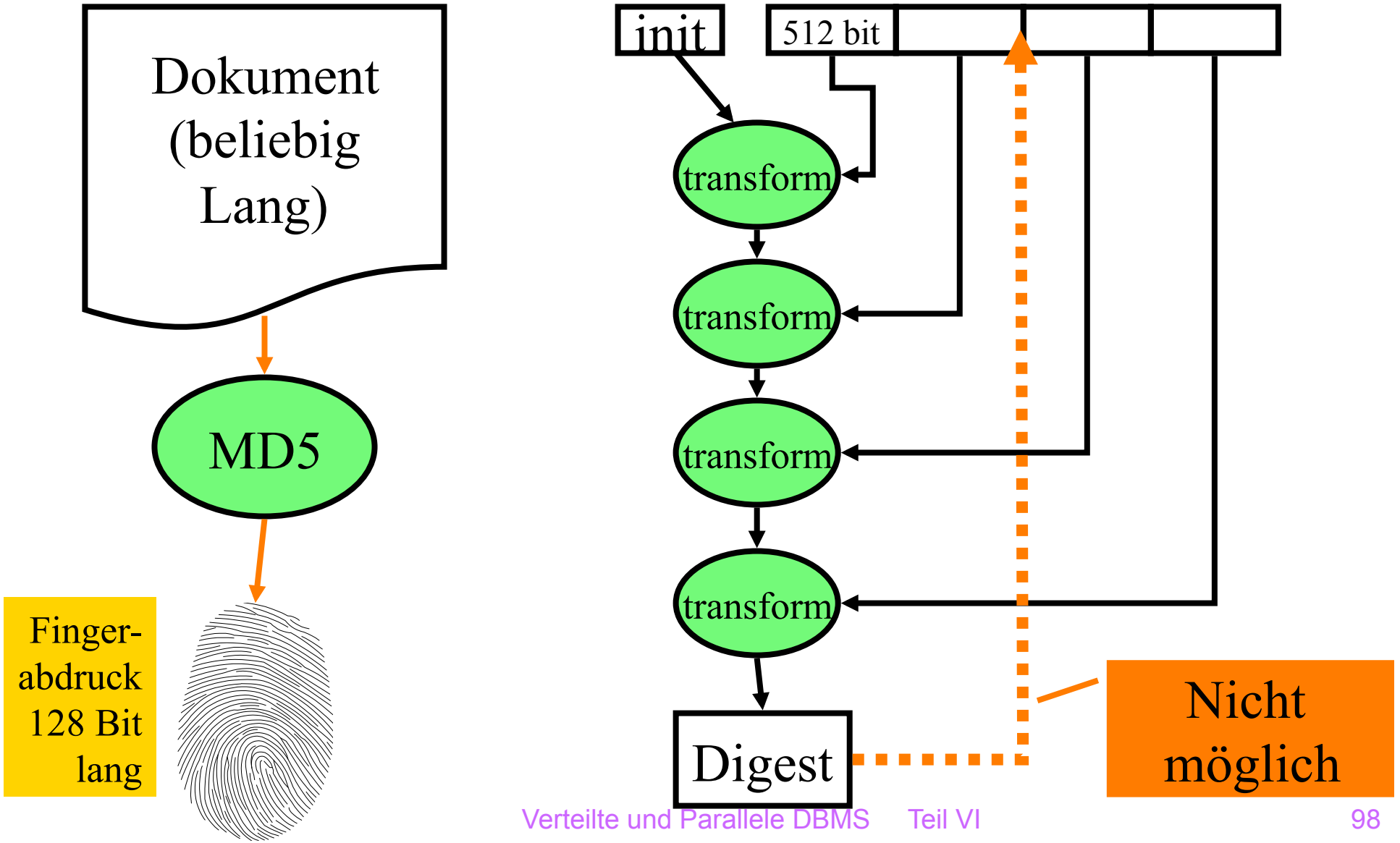
Literatur

- `http://www.rsa.com/`
- R.L. Rivest, A. Shamir und L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM (Association for Computing Machinery), Volume 21, Number 2, February 1978, S. 120–126.
- zu Enigma: `http://www.cs.uoregon.edu/~inoble/enigma/text/`
- F.L. Bauer: Kryptologie. Methoden und Maximen. 2. Auflage, Springer Verlag, Berlin, 1994.
- L. Peterson und B. Davie: Computer Networks. Morgan Kaufmann, 2000. Kapitel 8, S 568 – 615.

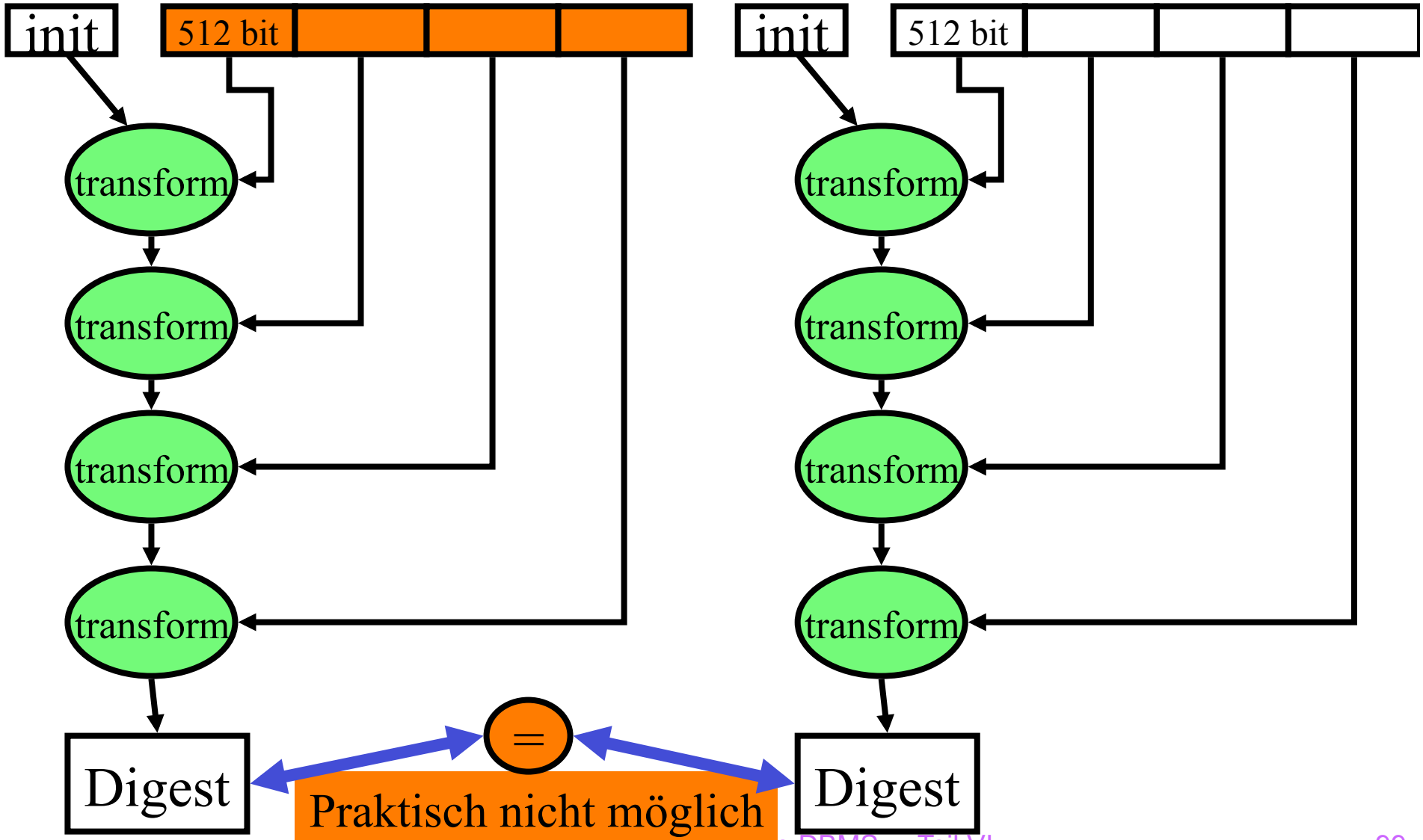
Message Digest



Message Digest



Message Digest: man kann sich kein Dokument zu einem Digest „konstruieren“



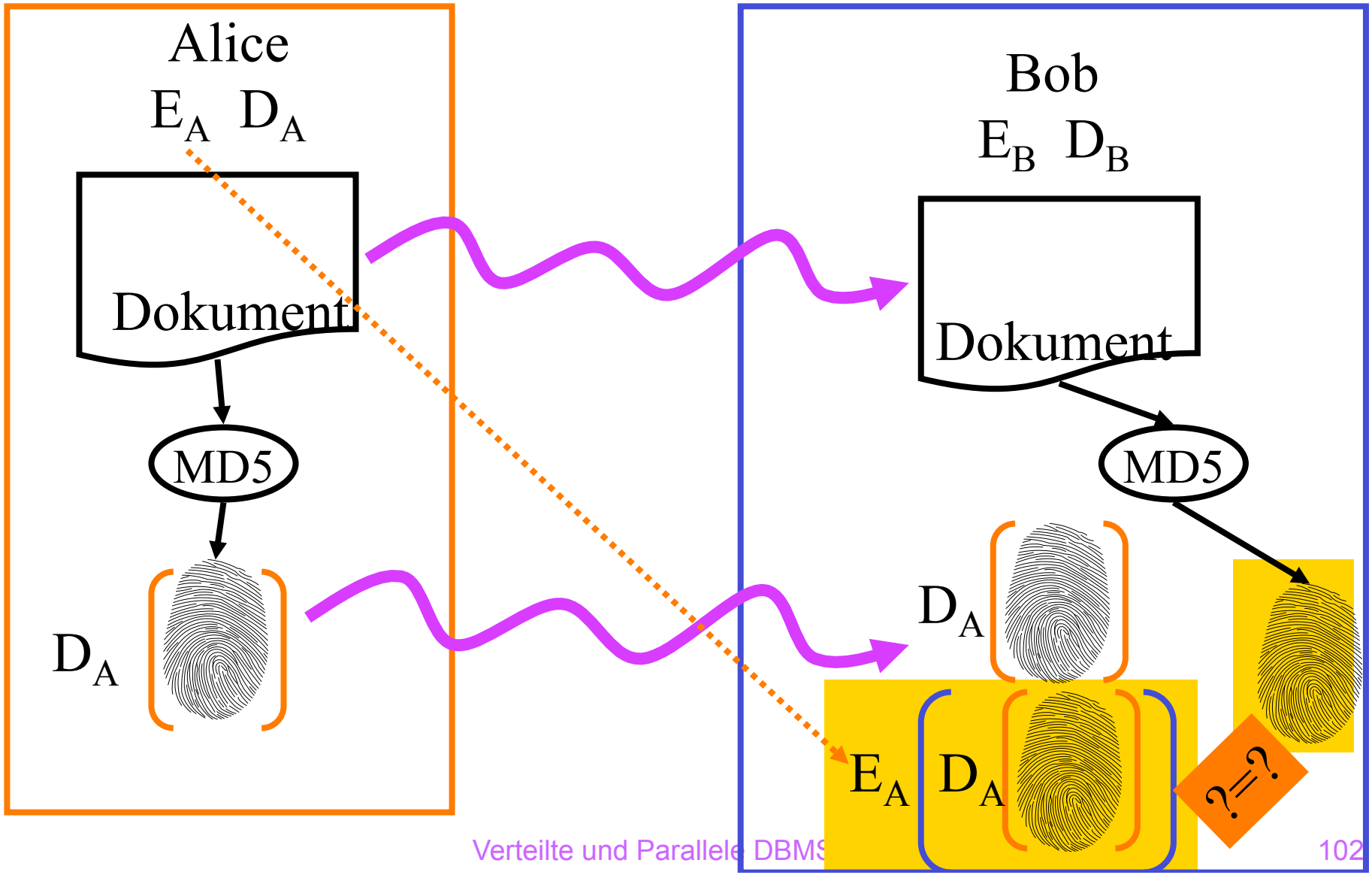
Message Digest

- Cryptographic checksum
  - just as a regular checksum protects the receiver from accidental changes to the message, a cryptographic checksum protects the receiver from malicious changes to the message.
- One-way function
  - given a cryptographic checksum for a message, it is virtually impossible to figure out what message produced that checksum; it is not computationally feasible to find two messages that hash to the same cryptographic checksum.
- Relevance
  - if you are given a checksum for a message and you are able to compute exactly the same checksum for that message, then it is highly likely this message produced the checksum you were given.

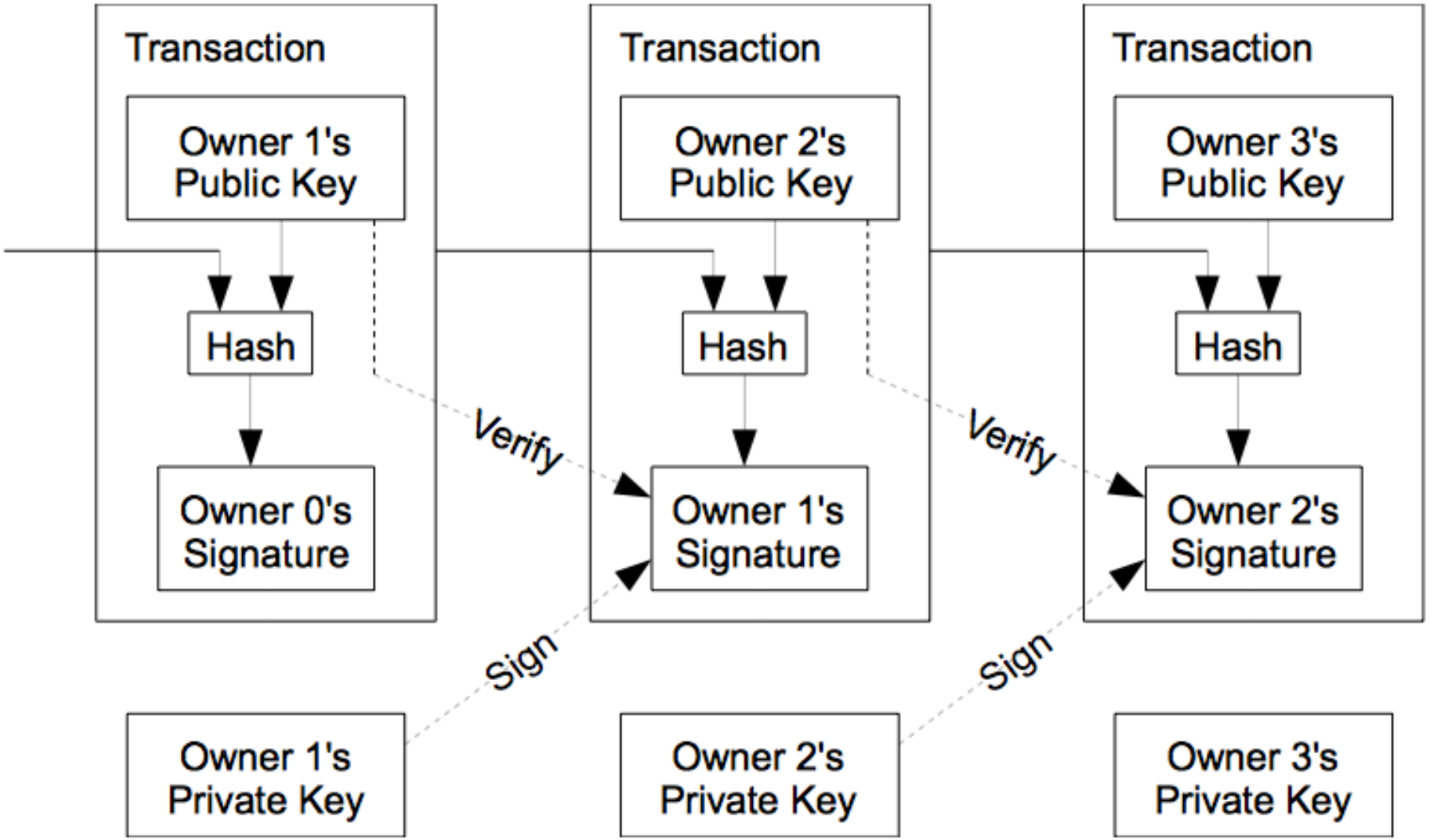
Message Integrity Protocols

- Digital signature using RSA
  - special case of a message integrity where the code can only have been generated by one participant
  - compute signature with private key and verify with public key
- Keyed MD5
  - sender: m + MD5(m + k) + E(k, *private*)
  - receiver
    - recovers random key using the sender's public key
    - applies MD5 to the concatenation of this random key message
- MD5 with RSA signature
  - sender: m + E(MD5(m), *private*)
  - receiver
    - decrypts signature with sender's public key
    - compares result with MD5 checksum sent with message

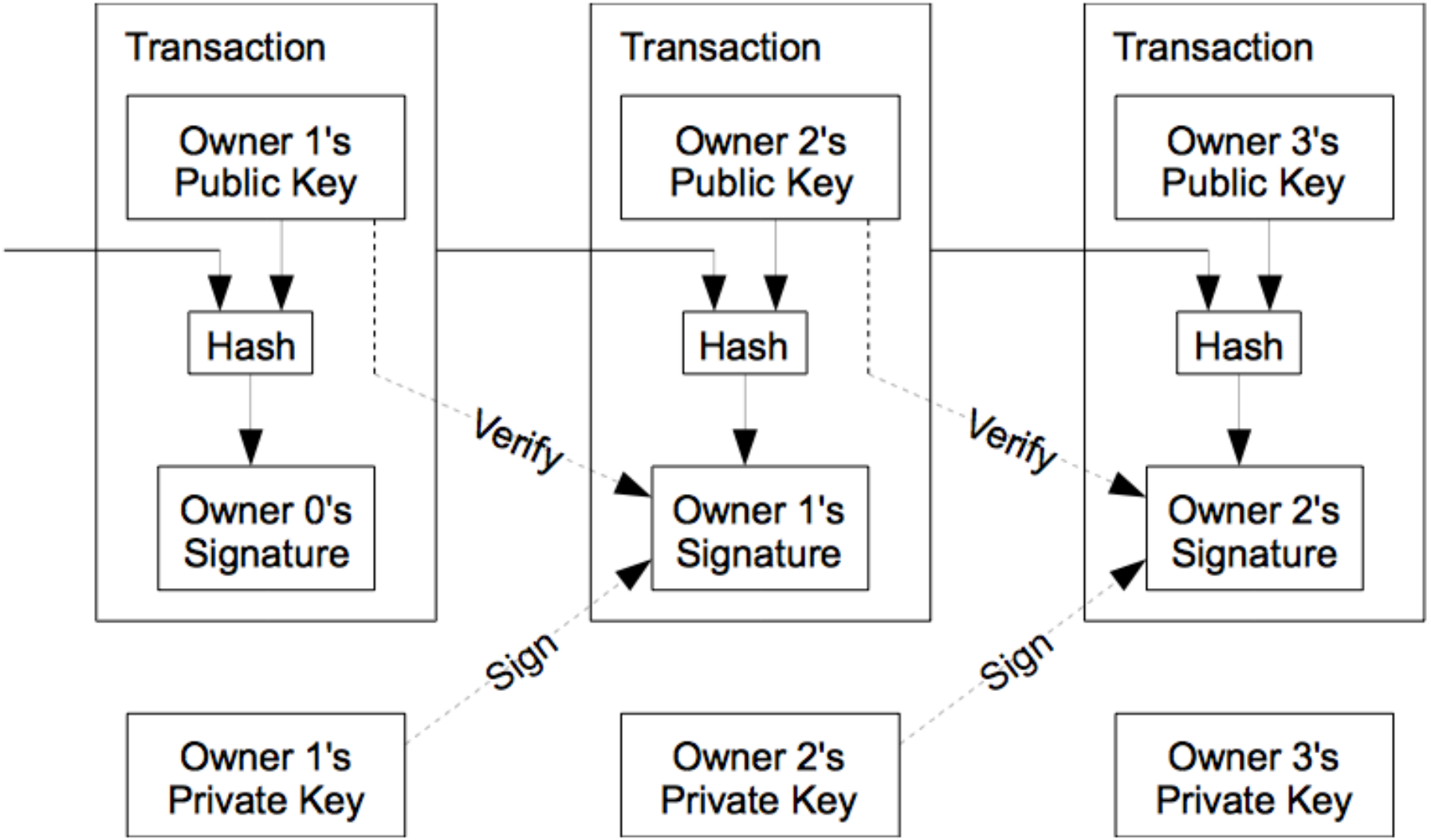
MD5 mit RSA-Signatur



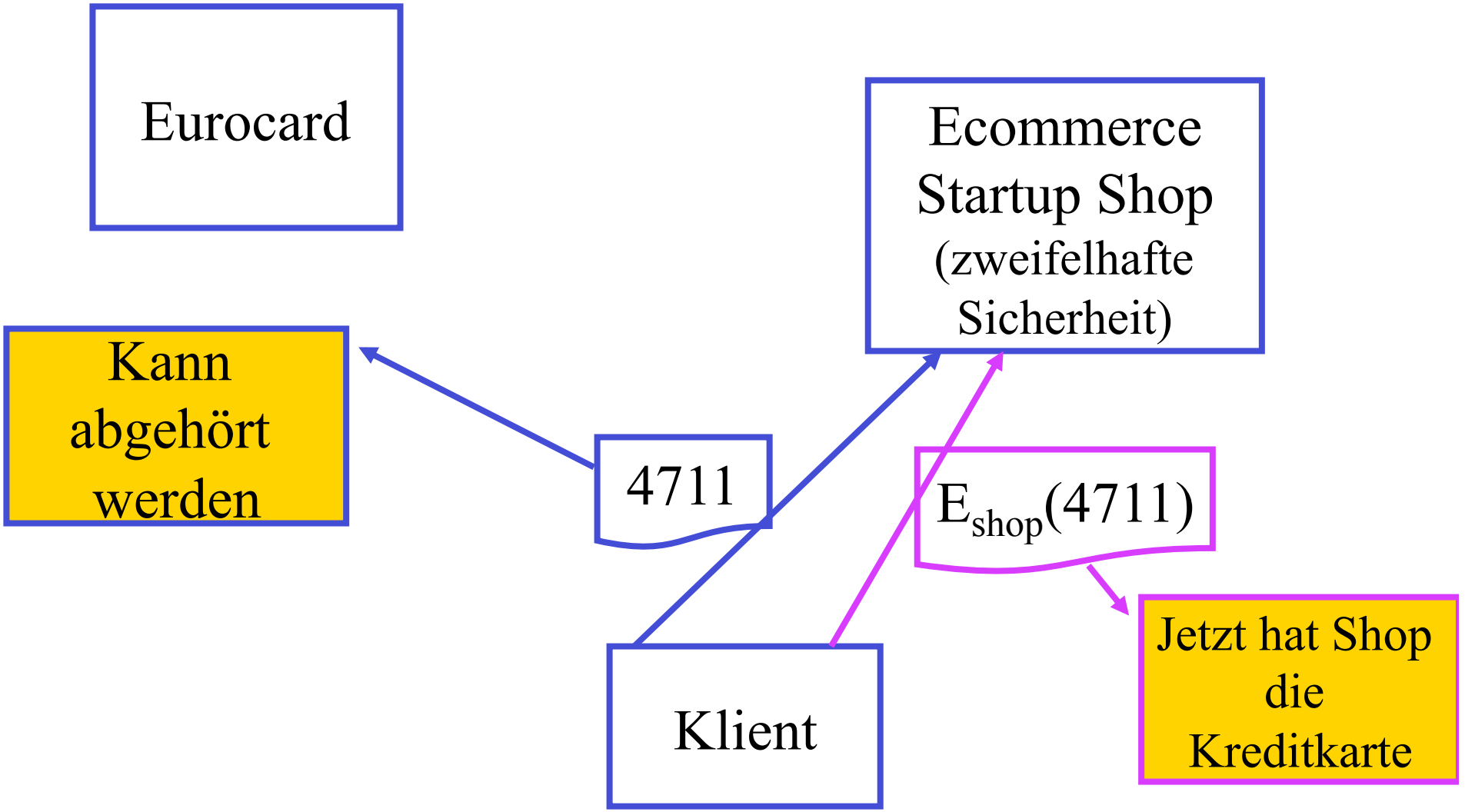
Was ist das?


Transaction
Owner 1's Public Key
Hash
Owner 0's Signature
Transaction
Owner 2's Public Key
Hash
Owner 1's Signature
Transaction
Owner 3's Public Key
Hash
Owner 2's Signature
Verify
Verify
Sign
Sign
Owner 1's Private Key
Owner 2's Private Key
Owner 3's Private Key

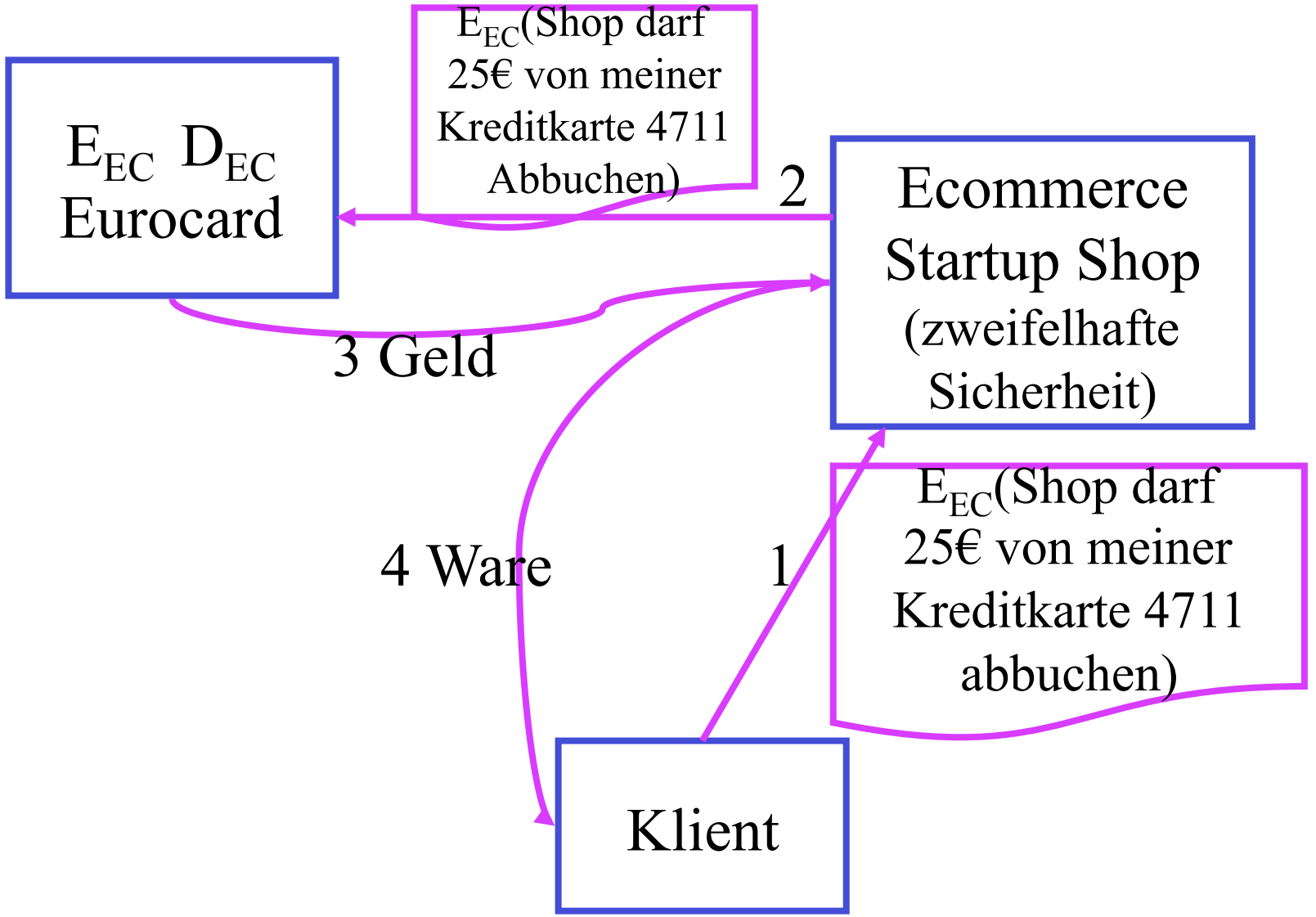
Transaktionskette eines Bit-Coins von Owner1 über Owner2 zu Owner3



Sicheres Zahlen per Kreditkarte



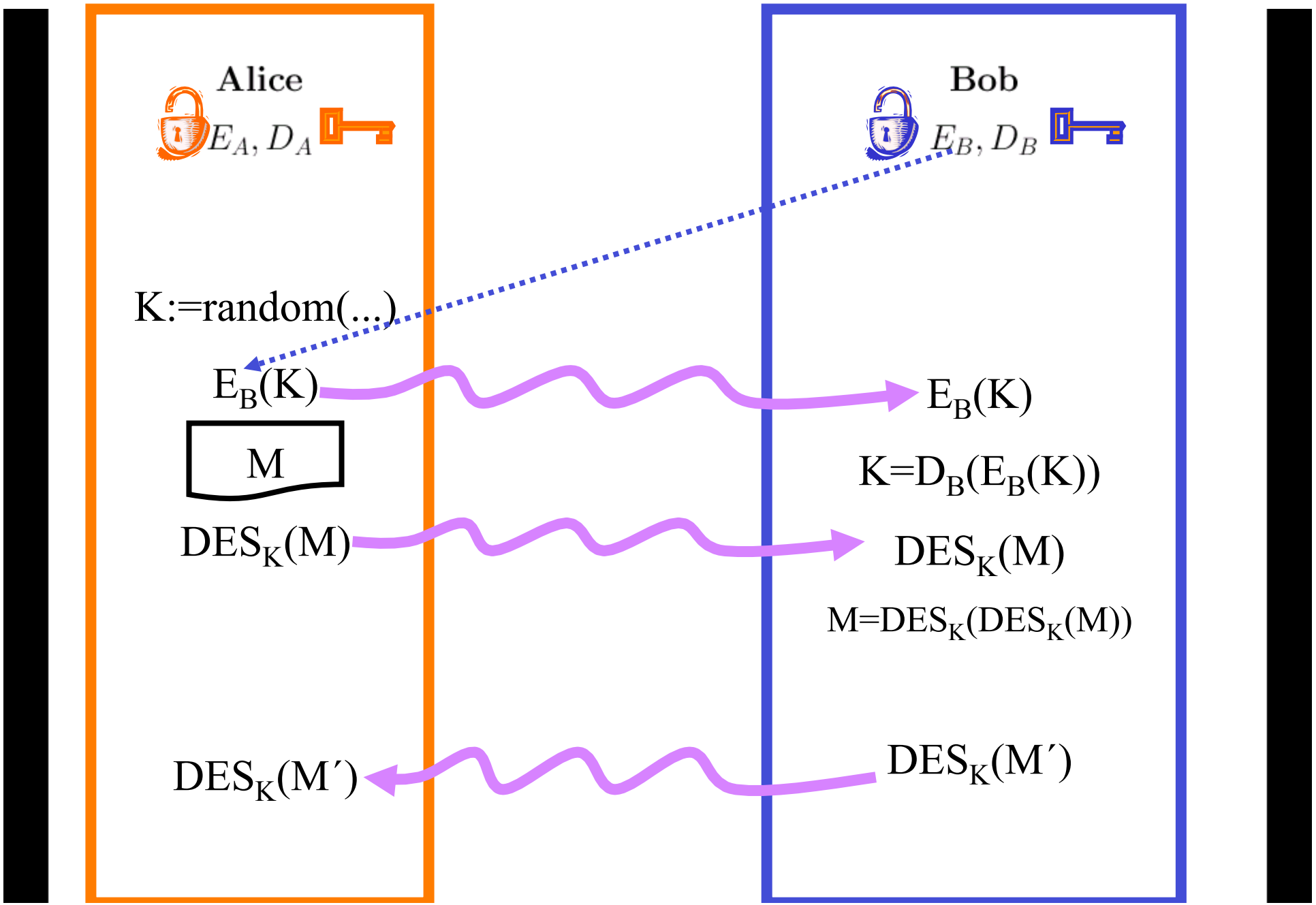
Sicheres Zahlen per Kreditkarte: SET-Standard



Leistungsfähigkeit der Verschlüsselungsverfahren

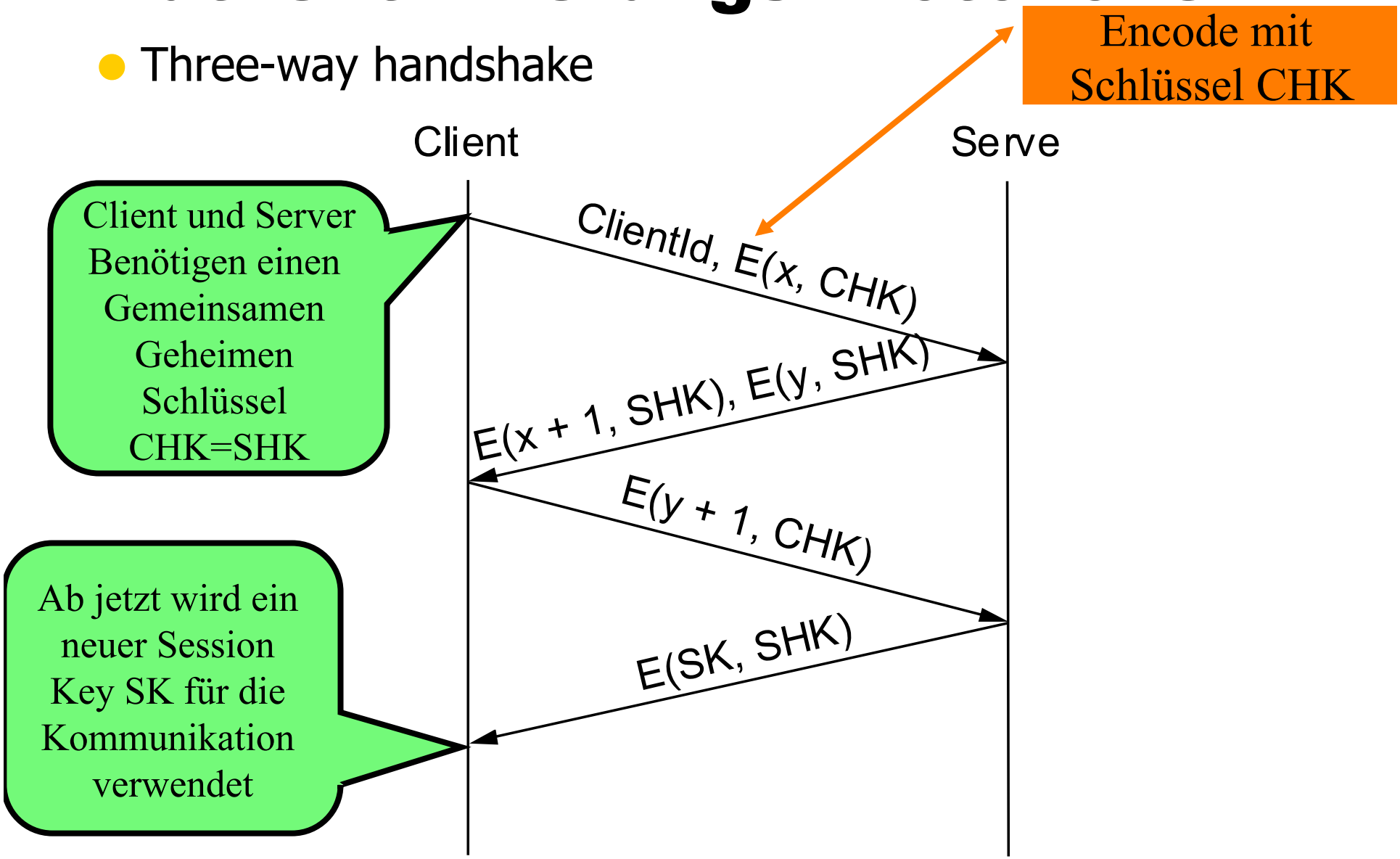
- DES und MD5 sind mehrere Größenordnungen schneller als RSA
- Softwareimplemtierung auf heutiger Hardware
  - DES: 36 Mbps
  - MD5: 85 Mbps
  - RSA: 1 Kbps
- Hardware-Implementierung
  - DES und MD5: x00 Mbps (x>2)
  - RSA: 64 Kbps
- Also ist RSA nicht für die Codierung von Nachrichten geeignet
  - Es wird für den DES-Schlüsselaustausch verwendet
  - Und für die Authentifizierung
  - Und für die digitale Signatur von MD5-Digests

Austausch von Schlüsseln (Session Keys)

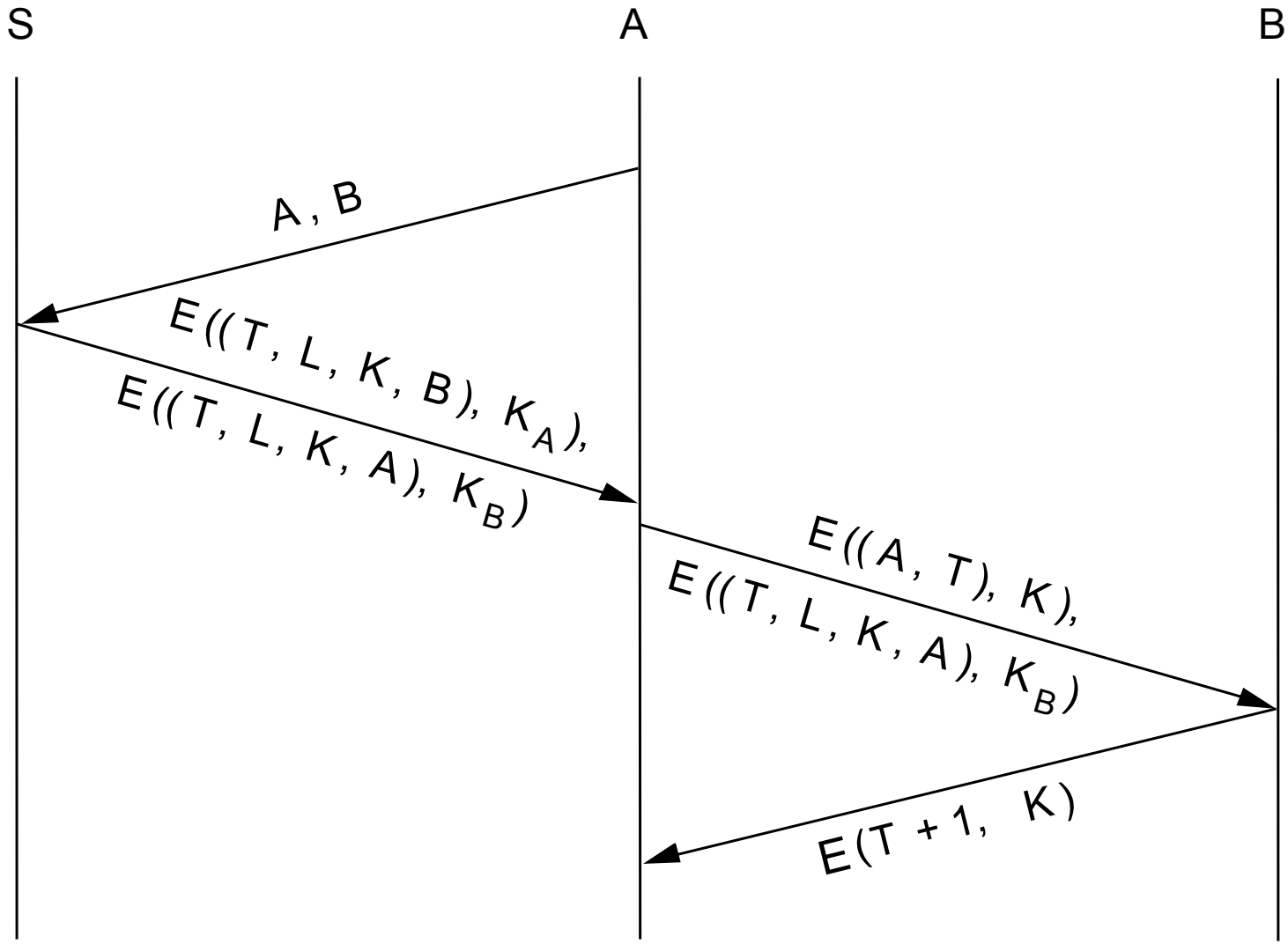


Authentifizierungs-Protokolle

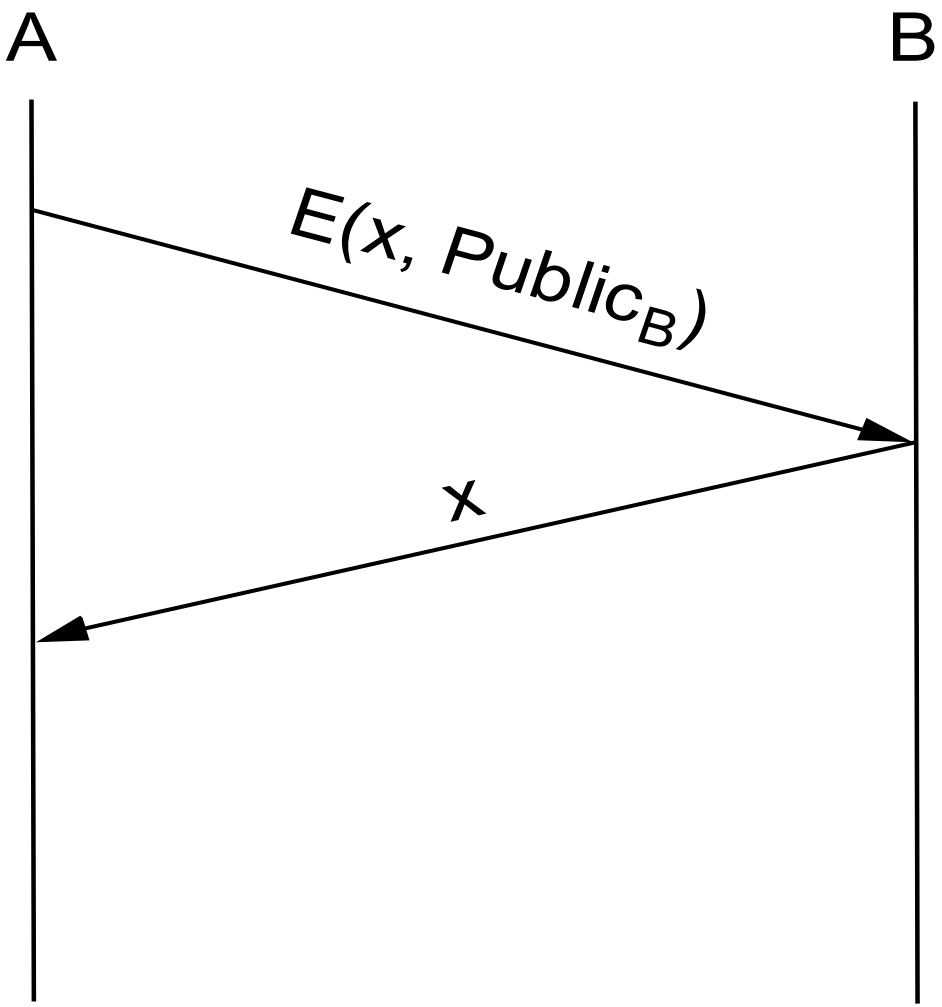
- Three-way handshake



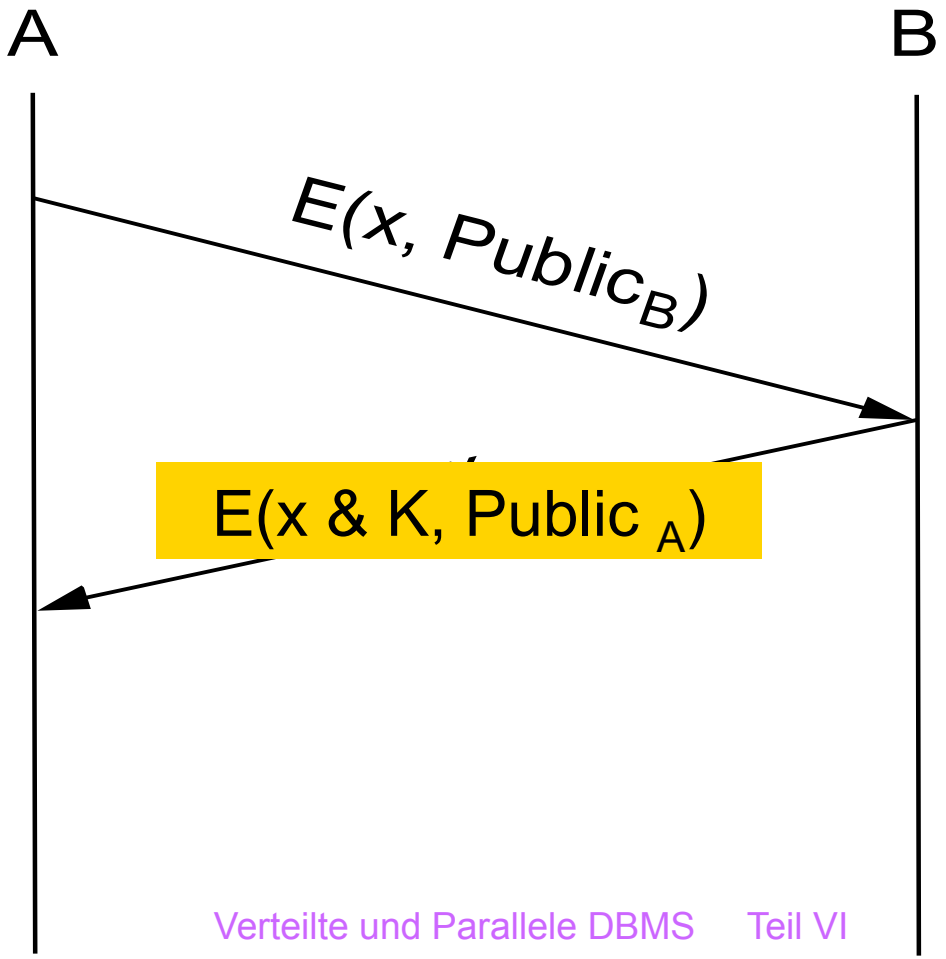
Trusted third Party (Kerberos)



- Public Key Authentifizierung



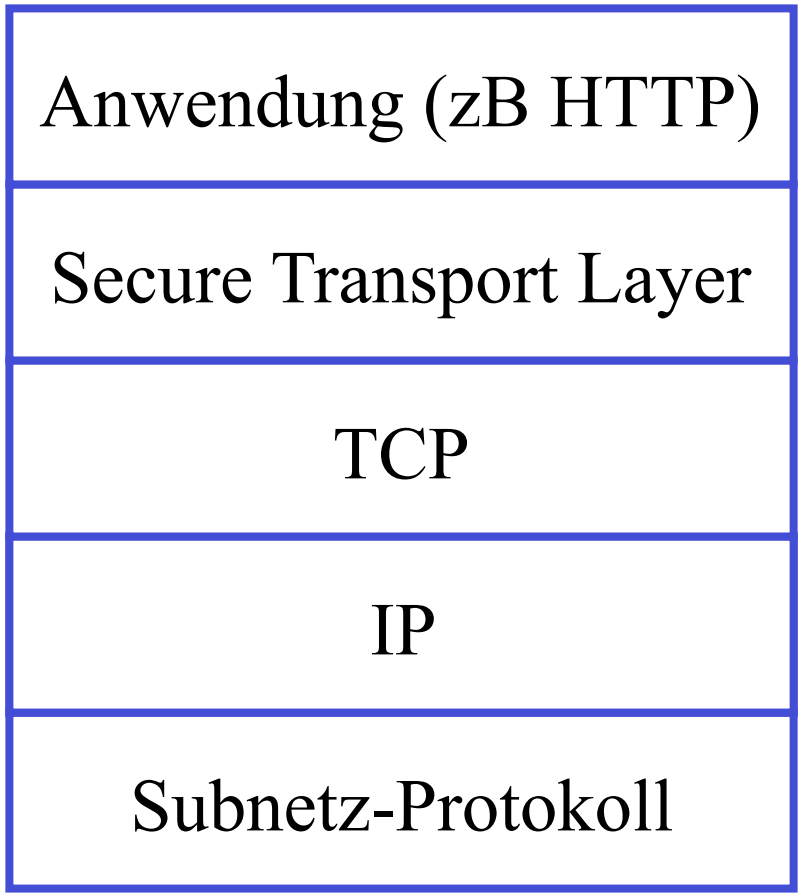
- Public Key Authentifizierung
- Abwehr des “man in the middle”
- Ab jetzt verwenden A und B den Schlüssel K für die (symmetrische) Verschlüsselung Ihrer Kommunikation
- Es ist sichergestellt, dass nur B diesen Schlüssel übermitteln konnte – und kein man in the middle. Warum?

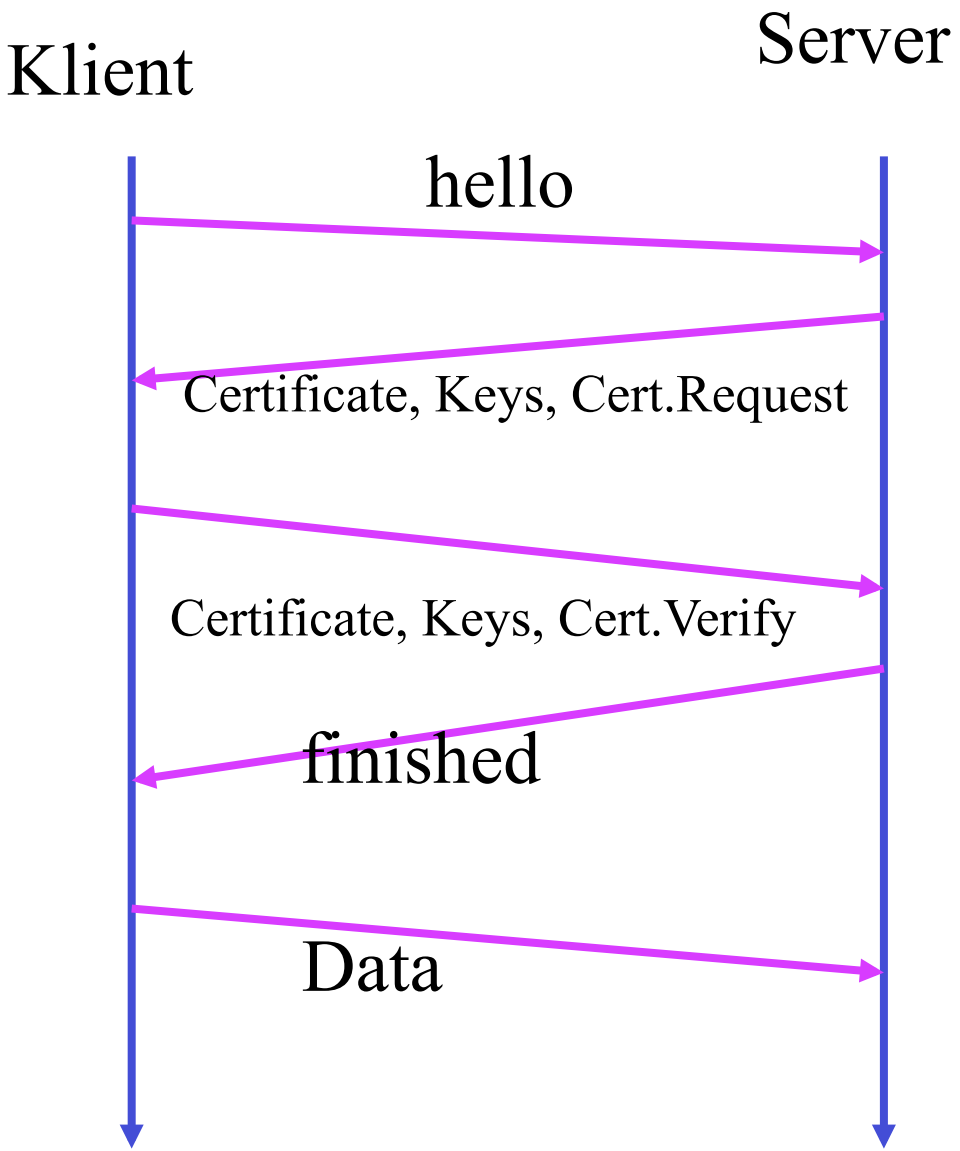


Beispielsysteme

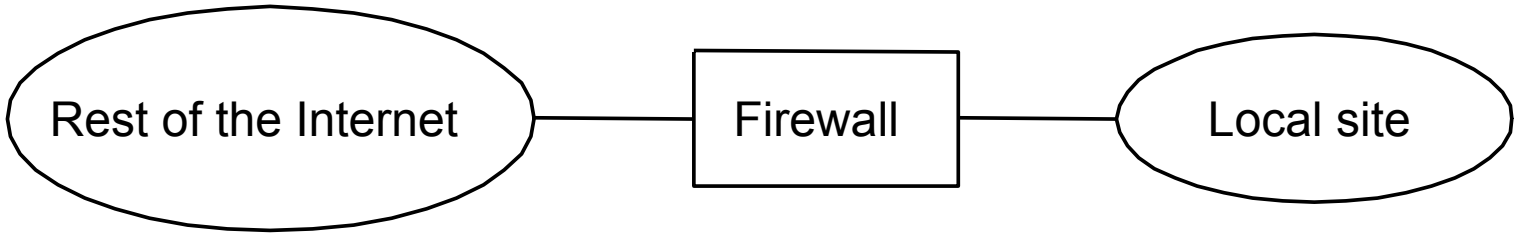
- PEM
  - Privacy Enhanced Email
  - A schickt an B eine Email M mit Signatur:
    - M + E(MD5(M),$private_A$)
    - Rechtlich bindend
- PGP
  - Pretty Good Privacy
  - Umgehung des US-Ausfuhrverbots für sichere Kryptographie
  - Wurde mittlerweile gelockert
- SSL bzw TLS
  - Secure Socket Layer
  - Transport Layer Security

TLS bzw SSI


Anwendung (zB HTTP)
Secure Transport Layer
TCP
IP
Subnetz-Protokoll


Klient
Server
hello
Certificate, Keys, Cert.Request
Certificate, Keys, Cert.Verify
finished
Data

Firewalls



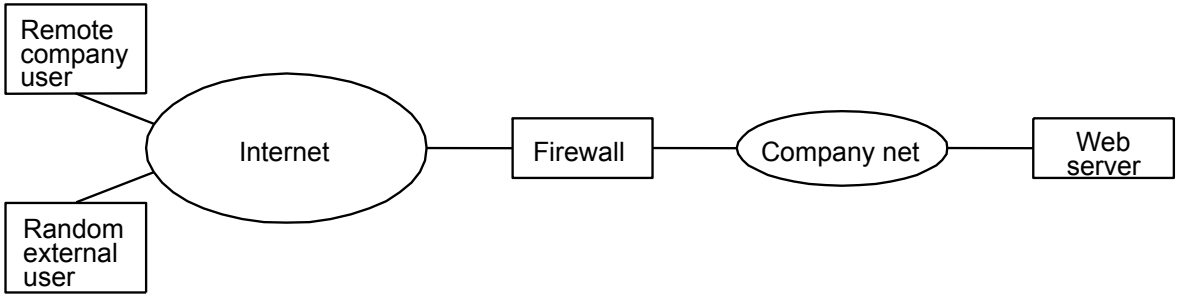
- Filter-Basierte Lösung
- Beispiel

  (192.12.13.14, 1234, 128.7.6.5, 80)

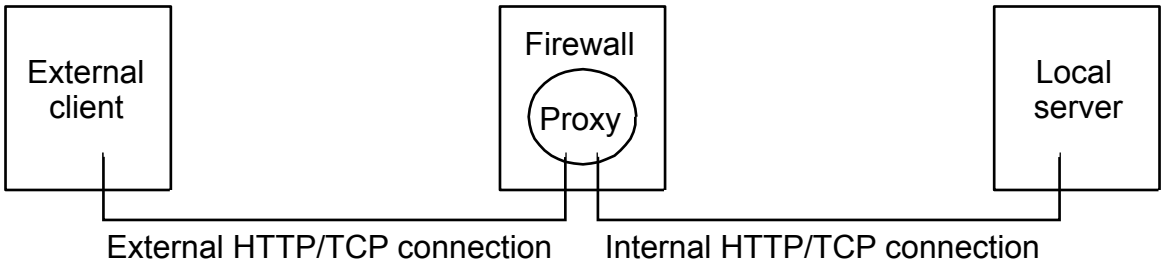
  (*,*, 128.7.6.5, 80)
  - Default: forward or not forward?
    - D.h. negative oder positive Autorisierung
  - Wie dynamisch?

Proxy-Basierte Firewalls

- Problem: komplexe Policy
- Beispiel: web server



- Lösung: proxy



- Design: transparent vs. classical
- Limitation: Attacke "von innen"