3 Implementierung

- Globale Tabelle/Matrix
  - System hält eine Datenstruktur und prüft im betreffenden Eintrag die Berechtigungen
  - Tabelle üblicherweise recht groß: passt evtl. nicht in den Speicher
- Zugriffslisten an den Objekten
  - jedes Objekt hält eine Liste der Berechtigungen (z.B. Unix Datei: Inode)
  - verringert üblicherweise den Platzbedarf für die Einträge (unnötige Felder der Matrix werden nicht repräsentiert)
- Zugriffslisten an den Subjekte
  - jedes Subjekt hält eine Liste von Objekten und den Berechtigungen, die das Subjekt für das Objekt hat
  - ähnlich Capabilities

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.3 Schutzmodell nach Bell-La Padula

- Sicherheitsgrad
  - Tupel: (Geheimhaltungsstufe, Schutzkategorie)
  - Geheimhaltungsstufe (g): ein Element aus einer vollständig geordneten Menge (z.B. vertraulich, geheim, streng geheim)
  - Schutzkategorie (s): Teilmenge von systemspezifischen Sachgebieten (z.B. Arbeiter, Angestellte, leit. Angestellte, Post)
  - Jedem Objekt und jedem Subjekt ist ein Sicherheitsgrad zugeordnet
- Sicherheitseigenschaft
  - Ein Subjekt kann nur Objekte mit gleichem oder niedrigerem Sicherheitsgrad lesend oder schreibend zugreifen.
  - Dabei gilt: $(g,s) \leq (g',s') \Rightarrow g \leq g' \wedge s \subseteq s'$

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.3 Schutzmodell nach Bell-La Padula

- *-Eigenschaft
  - Ein Subjekt kann nur dann gleichzeitig zu einem Objekt A lesenden und zu einem Objekt B schreibenden Zugriff haben, wenn B den gleichen oder einen höheren Sicherheitsgrad besitzt als A
- ★ Es ist nicht möglich Informationen eines hohen Sicherheitsgrads zu einem Objekt niedrigeren Sicherheitsgrads zu transportieren

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel

- Subjekte und Objekte mit Sicherheitsgraden
  - D_{LA} = Personaldaten der leitenden Angestellten:
    (streng geheim, {})
  - D_{AN} = Personaldaten der sonstigen Angestellten:
    (geheim, {})
  - D_{AR} = Personaldaten der Arbeiter:
    (geheim, {})
  - S_{pers} = Leiter des Personalbüros:
    (streng geheim, {Post, leit. Angestellte, Arbeiter, Angestellte})
  - S_{stellv} = Sachbearb. leitende Angestellte, stellvertr. Leiter Personalbüro:
    (streng geheim, {leit. Angestellte})
  - S_{sach} = Sachbearbeiter Angestellte u. Arbeiter:
    (geheim, {Arbeiter, Angestellte})
  - S_{post}: Poststelle:
    (streng geheim, {Post})

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

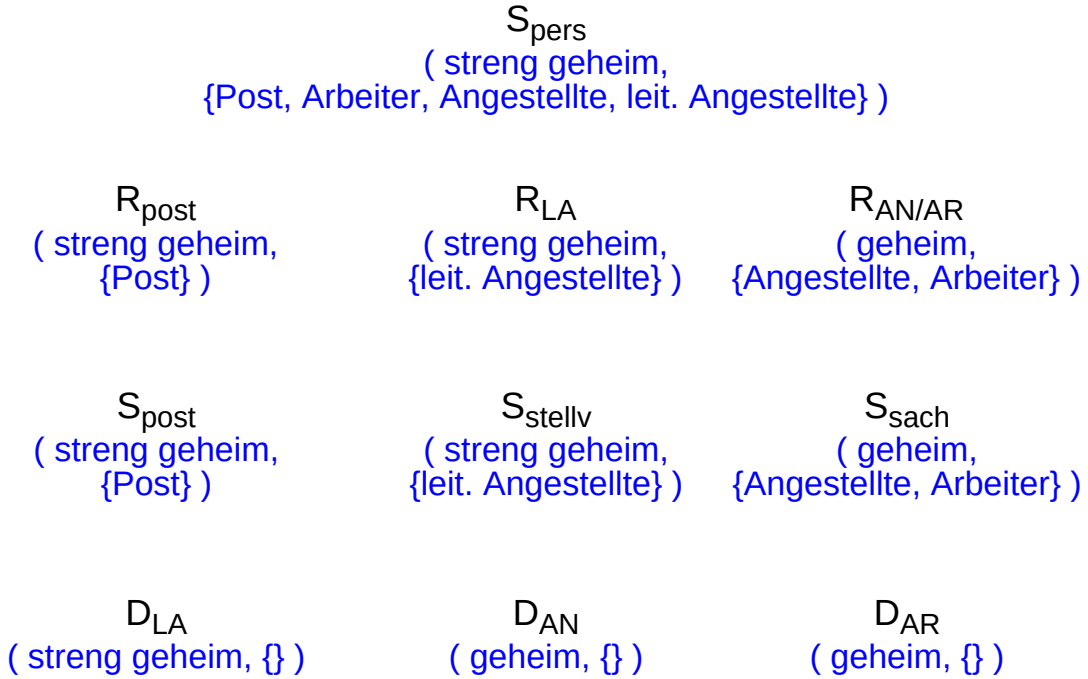
1 Beispiel (2)

- Prozeduren mit Sicherheitsgraden
  - R_{LA} = Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{LA}:
    (streng geheim, {leit. Angestellte})
  - $R_{AN/AR}$ = Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{AN} oder D_{AR}:
    (geheim, {Angestellte, Arbeiter})
  - R_{post} = Lesen von Name, Abteilung und Pers.-Nr.:
    (streng geheim, {Post})

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel (3)

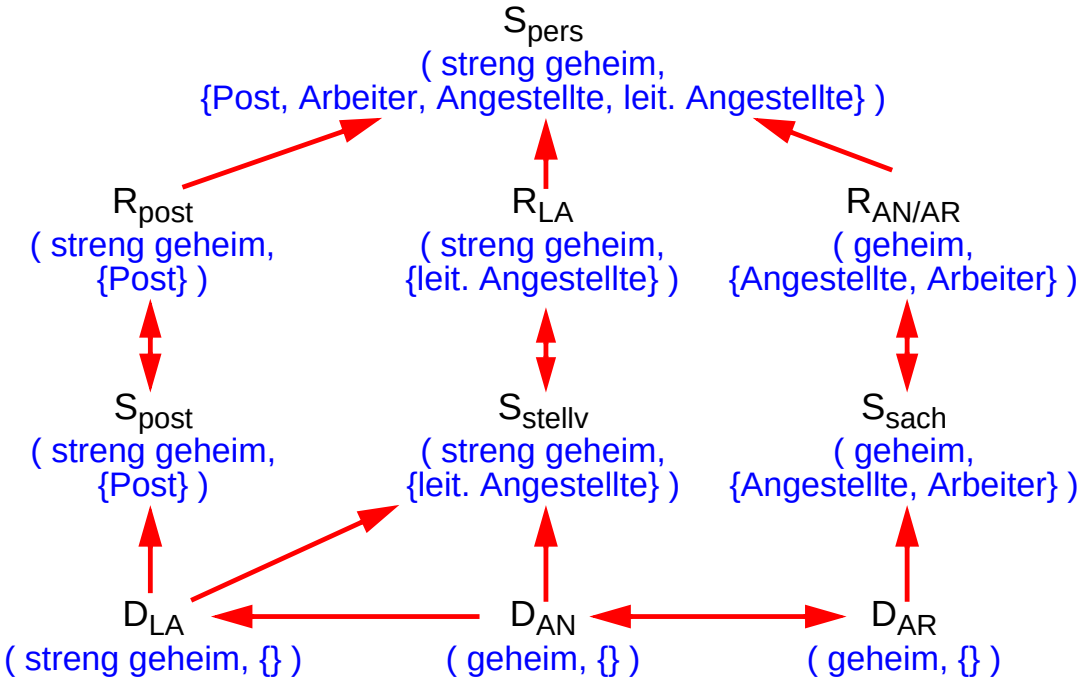
- Informationsflusskontrollgraph:



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel (4)

- Informationsflusskontrollgraph:



© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997–2000
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel (4)

- Sicherheitsgrade verhindern bestimmte Informationsflüsse unabhängig von der Schutzmatrix
  - z.B. kann Information aus R_{post} nach S_{post} gelangen, von dort aber nicht mehr an S_{sach} weitergegeben werden
- Umgekehrt kann die Schutzmatrix Einschränkungen treffen, die nicht durch die Sicherheitsgrade allein verhindert werden
  - z.B. kann S_{stellv} nicht den kompletten Inhalt von D_{LA} lesen, obwohl der Informationsflusskontrollgraph dies erlauben würde

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997–2000
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Bewertung

- Probleme
  - Information erlangt immer höhere Sicherheitsgrade und kann dann nicht mehr weitergegeben werden
  - Beispiel: Programm zur Steuererklärung greift auf streng geheime Buchhaltungsdaten zu ➔ Steuererklärung ist streng geheim
- Einführung von vertrauenswürdigen Prozeduren, die die *-Eigenschaft umgehen können
  - Informationen können im Sicherheitsgrad wieder heruntergestuft werden
- vertrauenswürdige Prozeduren stellen wiederum ein Sicherheitsrisiko dar
  - Verifikation nötig, aber schwierig

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.4 Schutz durch Speicherverwaltung

- Schutz vor gegenseitigem Speicherzugriff
  - Segmentierung und Seitenadressierung erlauben es, jedem Prozess nur den benötigten Speicher einzublenden
  - Segmentverletzung löst Unterbrechung aus
- Systemaufrufe
  - definierter Weg von einer Schutzumgebung (der des Prozesses) in eine andere (der des Betriebssystems)
- Erweiterung dieses Konzepts:
  - allgemeine Prozeduraufrufe zwischen verschiedenen Schutzumgebungen, realisiert mit der Speicherverwaltung und deren Hardware (MMU)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

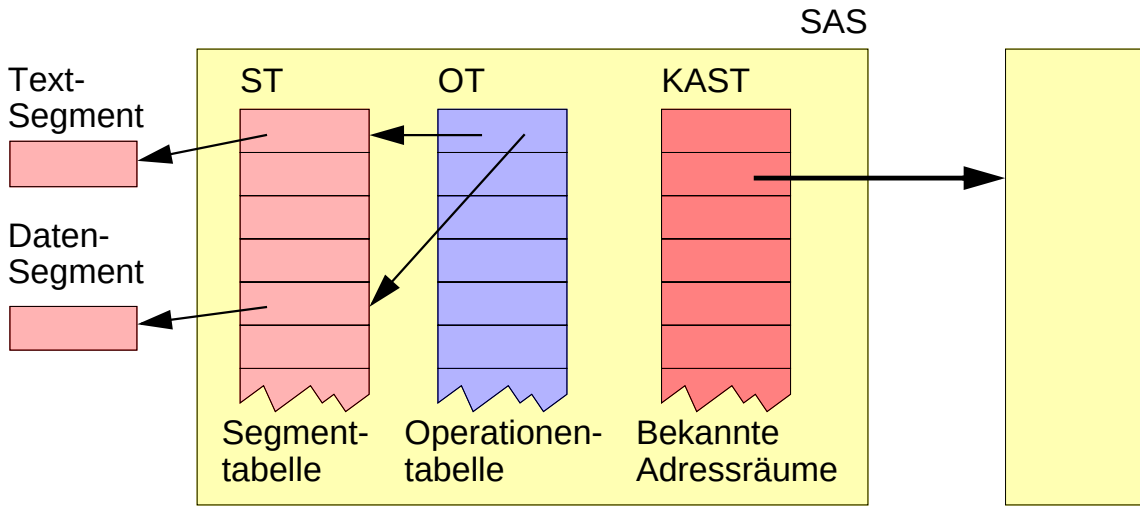
1 Modulkonzept von Habermann

- Idee (von 1976)
  - ◆ Adressräume (Module) bilden Schutzumgebungen
  - ◆ Adressräume bieten definierte Operationen an (ähnlich wie das Betriebssystem Systemaufrufe anbietet)
  - ◆ Parameter werden in speziellen Segmenten übergeben
- ★ Bietet allgemeinen Schutz der Module und erlaubt kontrollierte Interaktionen
- Module besitzen einen statischen Adressraum *(SAS, Static address space)*
  - ◆ enthält Liste von Segmenten, die zu dem Modul gehören bzw. von dem Modul zugegriffen werden dürfen
  - ◆ enthält Liste von angebotenen Operationen mit den Angaben, welche Segmente jede Operation benötigt (u.a. Segment für die auszuführenden Instruktionen)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Modulkonzept von Habermann (2)

- ◆ enthält Liste von bekannten Adressräumen anderer Module (dort können dann Operationen aufgerufen werden)

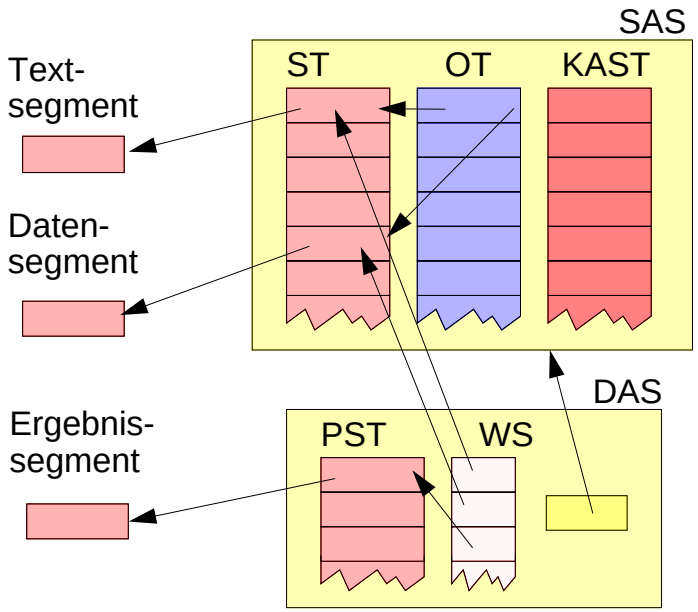


KAST = *Known address space table*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Modulkonzept nach Habermann (3)

- Aktivitätsträger sind einen dynamische Adressraum zugeordnet *(DAS, Dynamic address space)*

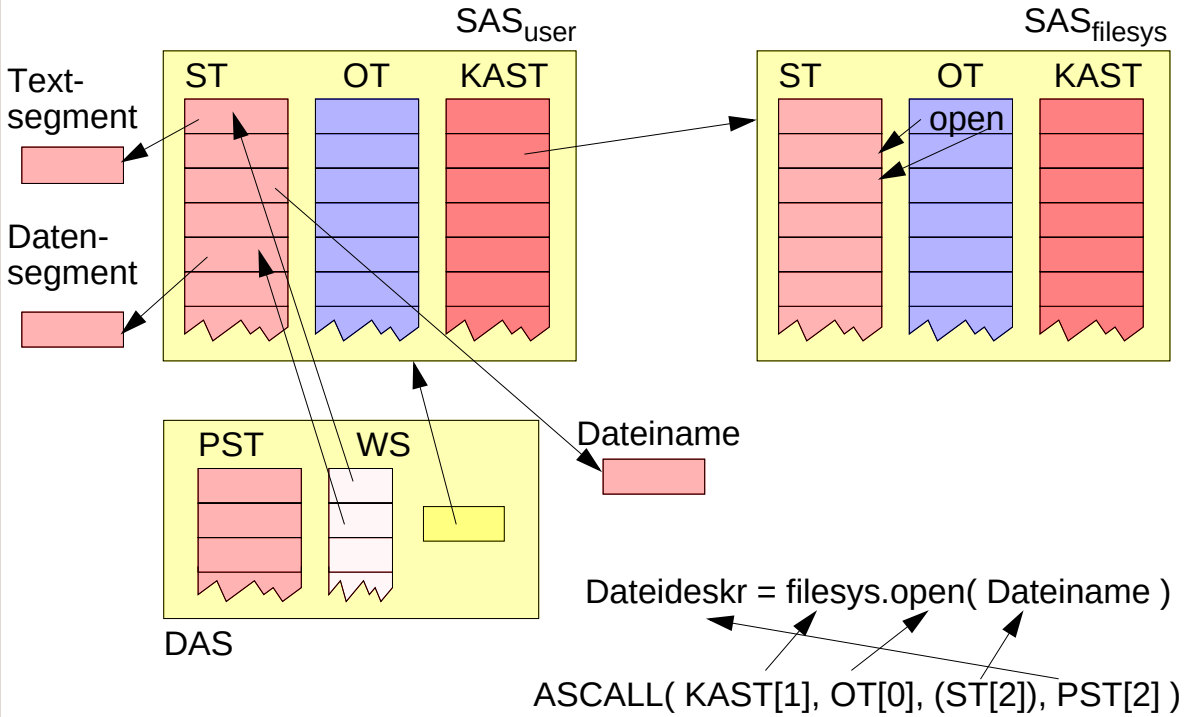


- enthält Segmenttabelle für Parameter und Ergebnisse *(PST)*
- enthält Verweise auf Segmenttabellen, die den Adressraum zusammenstellen *(WS= Working space)*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Beispielaufruf

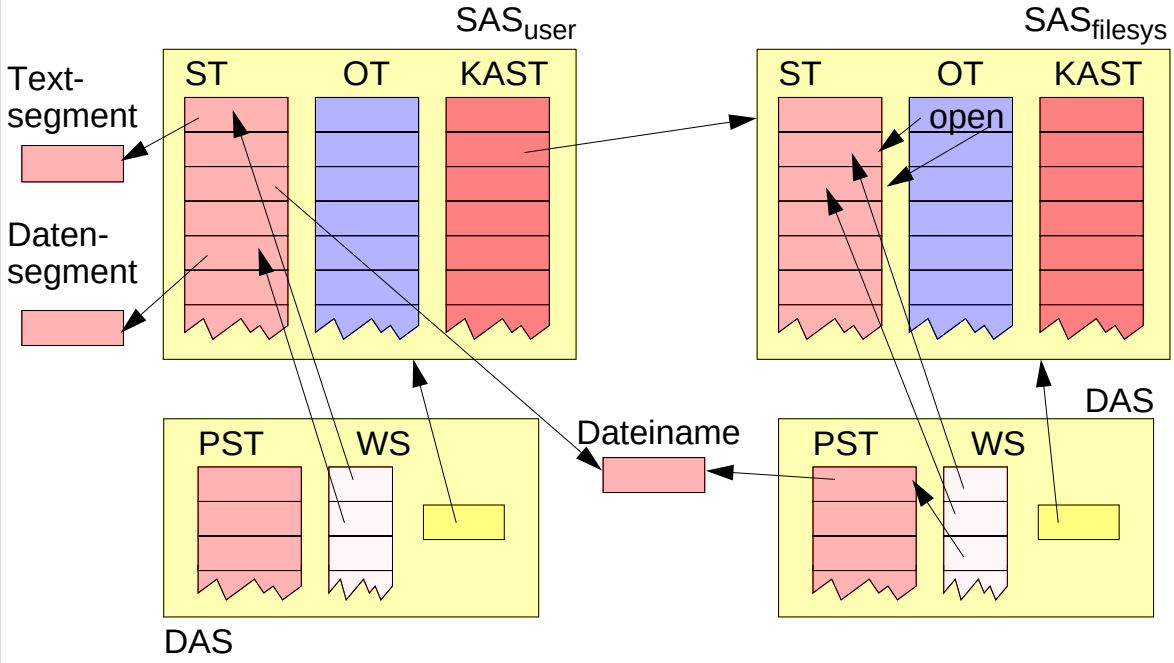
- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Beispielaufruf (2)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

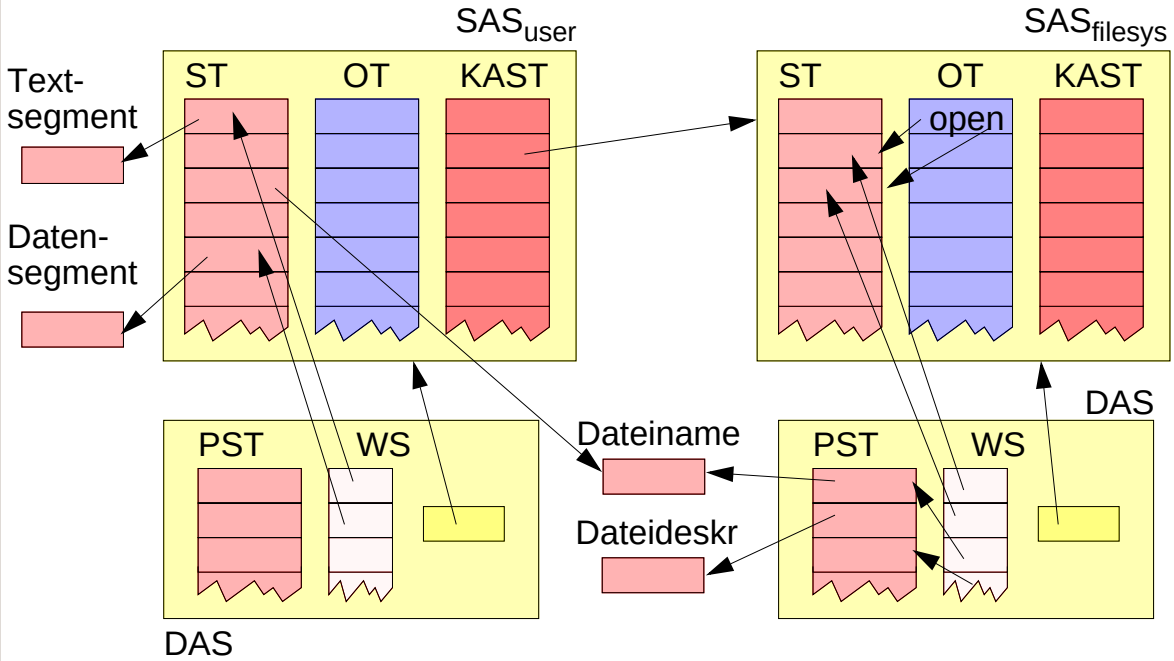


  - für den Aufruf von „open“ wird ein neuer DAS erzeugt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Beispielaufruf (3)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

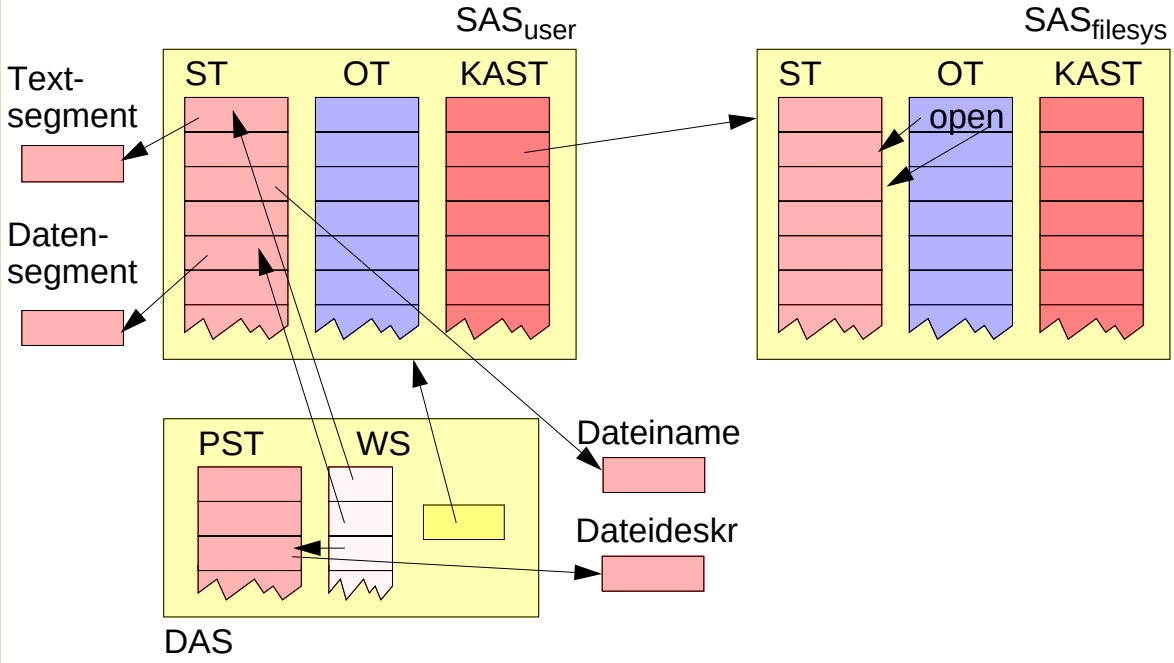


  - „open“ erzeugt neues Segment für den Dateideskriptor

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Beispielaufruf (4)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf



  - Ergebnissegment wird an den Aufrufer zurückgegeben

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

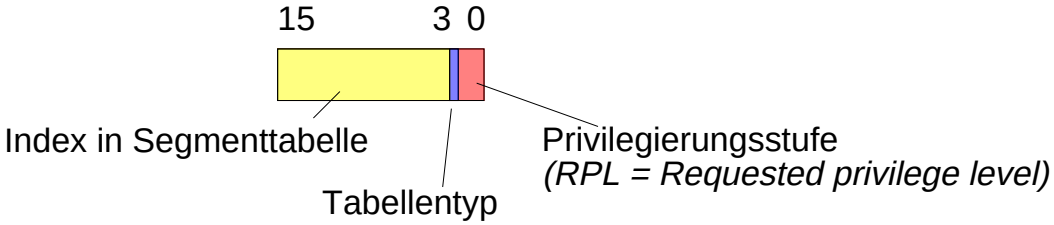
3 Beispiel: Pentium

- Privilegierungsstufen
  - Stufe 0: höchste Privilegien (privilegierte Befehle, etc.): BS Kern
  - Stufe 1: BS Treiber
  - Stufe 2: BS Erweiterungen
  - Stufe 3: Benutzerprogramme
- Merke:
  - kleine Stufe: hohe Privilegien
  - große Stufe: kleine Privilegien

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (2)

- Segmentselektoren enthalten Privilegierungsstufe

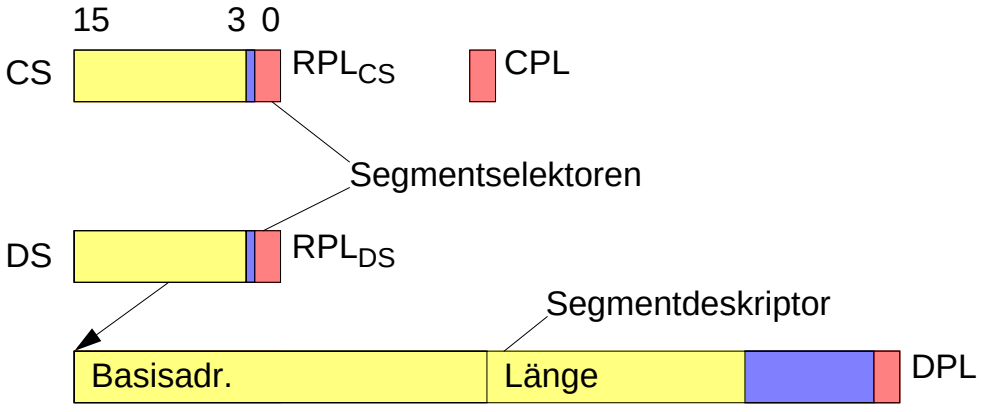


- Codesegmentselektor (CS) bestimmt aktuelle Privilegierungsstufe
  - CPL = *Current privilege level*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (3)

- Datenzugriff (z.B. auf Datensegment DS)



  - DPL = *Descriptor privilege level*
  - CPL ist normalerweise gleich RPL_{CS}
  - Zugriff wird erlaubt, wenn: DPL ≥ max(CPL, RPL_{DS})
  - ansonsten wird Unterbrechung ausgelöst (Schutzverletzung)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

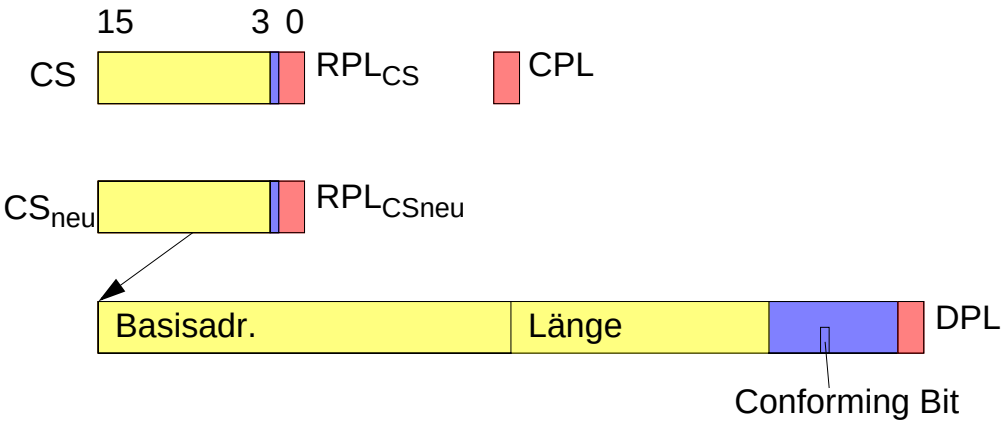
3 Beispiel: Pentium (4)

- Erläuterung:
  - ◆ DPL < CPL: *Schutzverletzung*
    augenblickliche Privilegierungsstufe hat weniger Privilegien als im Deskriptor verlangt
    (CPL hat höhere Stufe als der Deskriptor)
  - ◆ DPL ≥ CPL: augenblickliche Privilegierungsstufe ist mindestens so hoch wie die im Deskriptor
  - ◆ Segment kann nur angesprochen werden, wenn augenblickliche Stufe die gleichen oder mehr Privilegien benötigt.
  - ◆ DPL < RPL(ds): *Schutzverletzung*
    Selektor hat weniger Privilegien als der Deskriptor
    (Selektor hat höhere Stufe als der Deskriptor)
  - ◆ DPL ≥ RPL(ds): Selektor hat mindestens die gleiche Privilegierungsstufe wieder Deskriptor
  - ◆ Selektor darf keine niedrigeren Privilegien versprechen als wirklich verlangt.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (5)

- Sprünge in andere Codesegmente *(Far Call)*

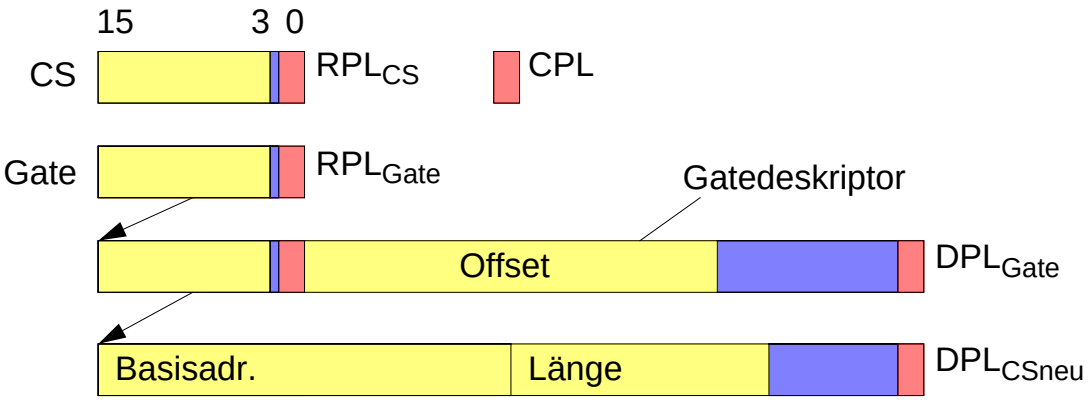


- ◆ Sprung wird erlaubt, falls: DPL = CPL oder Conforming Bit gesetzt und DPL ≤ CPL
- ◆ Im Falle von DPL ≤ CPL wird jedoch CPL nicht geändert (Codesegment hat höheres Privileg, CPL bleibt aber unverändert)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (6)

- Kontrolltransfer mit einem *Gate*

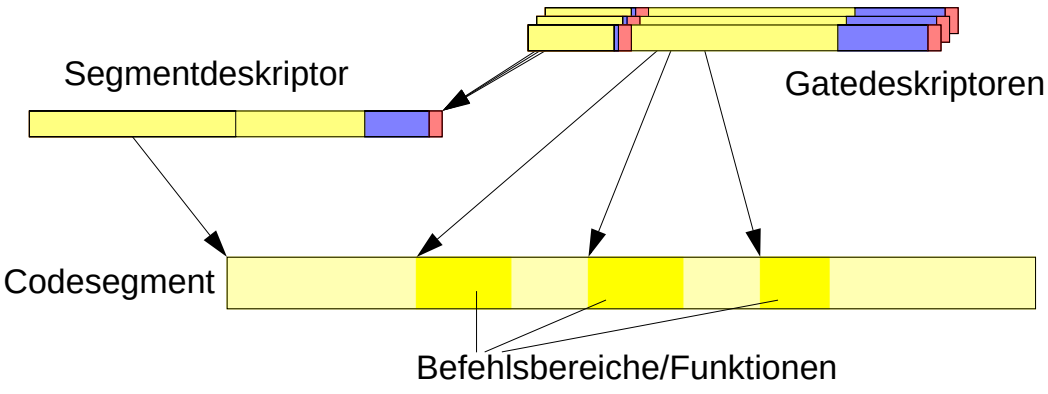


- Gatedeskriptoren stehen wie Segmentdeskriptoren in der Segmenttabelle
- Gatedeskriptor enthält Segmentselektor für das Codesegment und einen Offset zu diesem Segment, an dem der Einsprungpunkt liegt
- Kontrolltransfer (*CALL* Aufruf) wird erlaubt, falls:
  $DPL_{Gate} \geq \max(CPL, RPL_{Gate})$ und $DPL_{CSneu} \leq CPL$

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (7)

- Gates erlauben den kontrollierten Sprung in ein privilegierten Befehlsbereich

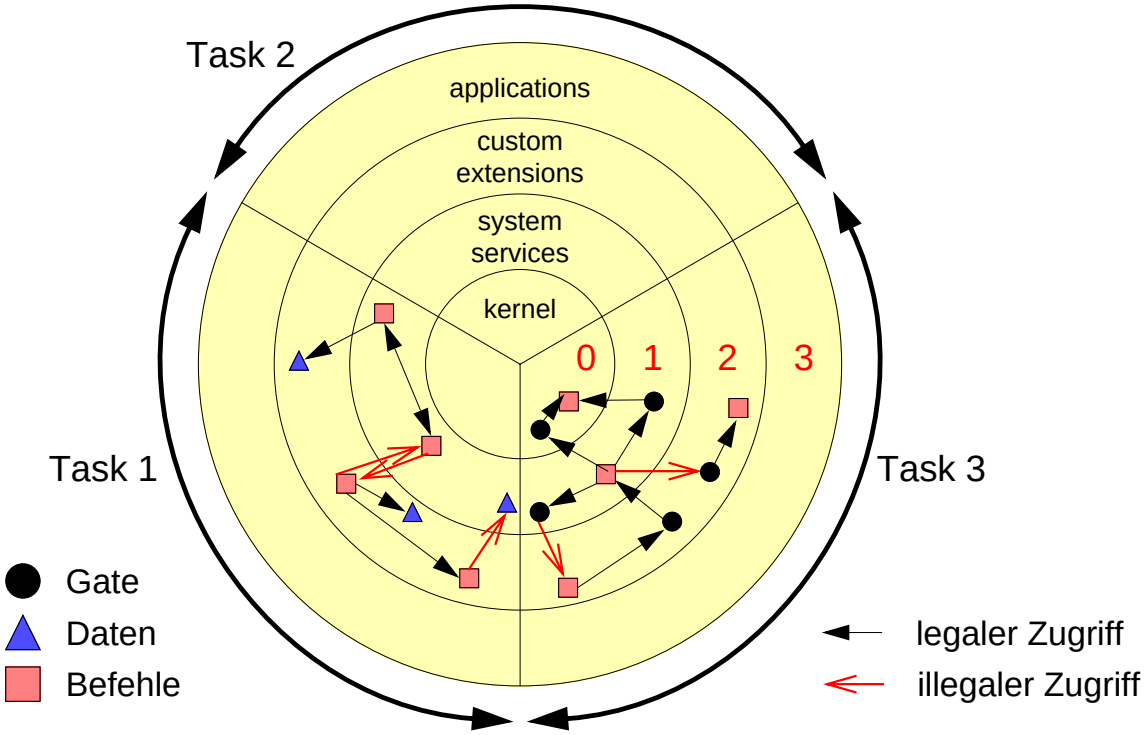


- es existiert auch der entsprechende Rücksprung
- für jede Privilegierungsstufe gibt es einen eigenen Stack; dieser wird mit umgeschaltet
- Parameter werden automatisch auf den neuen Stack kopiert (Anzahl wird im Gatedeskriptor vermerkt)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Beispiel: Pentium (8)

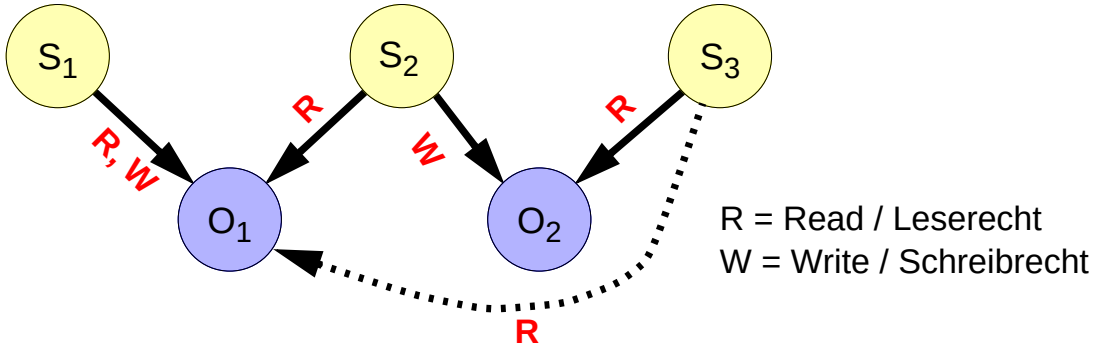
- Beispiel eines realisierten Schutzsystems



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.5 Capability-basierte Systeme

- Ein Benutzer (Subjekt) erhält eine Referenz auf ein Objekt
  - die Referenz enthält alle Rechte, die das Subjekt an dem Objekt besitzt
  - bei der Nutzung der Capability (Zugriff auf das Objekt) werden die Rechte überprüft



Subjekte und Objekte; Weitergabe einer Capability (O_1 von S_2 nach S_3)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.5 Capability-basierte Systeme (2)

★ Vorteile

- keine Speicherung von Rechten beim Objekt oder Subjekt nötig; Capability enthält Zugriffsrechte
- leichte Vergabe von individuellen Rechten
- einfache Weitergabe von Zugriffsrechten möglich

▲ Nachteile

- Weitergabe nicht kontrollierbar
- Rückruf von Zugriffsrechten nicht möglich
- Capability muss vor Fälschung und Verfälschung geschützt werden (z.B. durch kryptographische Mittel oder durch Speicherverwaltung)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel: Hydra

■ Hydra ist ein Capability-basiertes Betriebssystem

- entwickelt Mitte der Siebziger Jahre an der Carnegie-Mellon University
- lief auf einem speziellen Multiprozessor namens **C.mmp**
- Capability-Mechanismen sind integraler Bestandteil des Betriebssystems

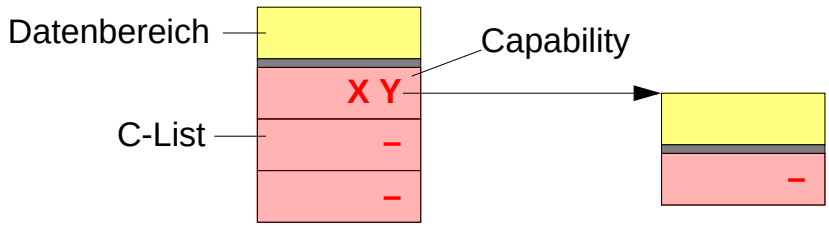
■ Objekte in Hydra werden durch Capabilities angesprochen und geschützt

- Objekte haben einen Typ (z.B. Prozeduren, Prozesse/LNS, Semaphoren, Datei etc.)
- Capabilities haben entsprechenden Typ
- benutzerdefinierte Typen sind möglich

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel: Hydra (2)

- generische Operationen für alle Typen implementiert durch das Betriebssystem
- Objekte besitzen eine Liste von Capabilities auf andere Objekte (genannt *C-List*)
- Capabilities enthalten Rechte
- Objekte besitzen einen Datenbereich (implementiert durch geschütztes Segment)



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel: Hydra (3)

- Prozesse (Subjekte)
  - Prozesse besitzen einen aktuellen Kontext, den LNS *(Local name space)*
  - LNS ist ein Objekt
  - zum LNS gehört einen Aktivitätsträger (Thread)
  - LNS kann nur auf Objekte zugreifen, die in seiner C-List stehen (mehrstufige Zugriffe, z.B. auf die C-List eines Objekts, dessen Capabilities in der C-List des LNS steht, sind möglich; Pfad zur eigentlichen Capability)
- Capabilities
  - Prozesse können nur über Systemaufrufe ihre Capabilities bzw. ihre C-List bearbeiten
  - Capabilities können nicht gefälscht oder verfälscht werden
  - Betriebssystem kann sicheres Schutzkonzept basierend auf Capabilities implementieren

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

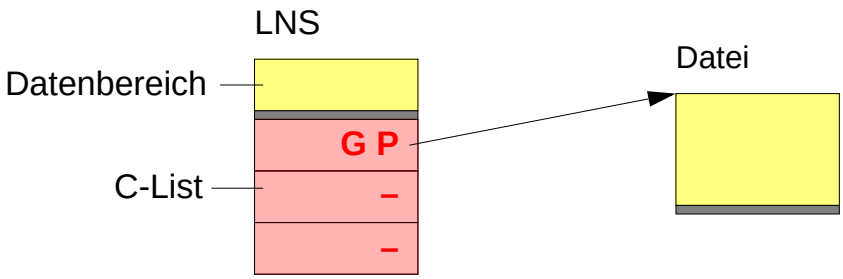
2 Datenzugriff in Hydra

- Operationen auf dem Datenbereich
  - *Getdata:* kopiere Abschnitt aus dem Datenbereich eines Objekts in den Datenbereich des LNS
  - *Putdata:* kopiere Abschnitt aus dem Datenbereich des LNS in den Datenbereich eines Objekts
  - *Adddata:* füge Daten zu dem Datenbereich eines Objekts hinzu
- Dazugehörige Rechte:
  - ***G****ETRTS:* erlaubt den Aufruf von *Getdata*
  - ***P****UTRTS:* erlaubt den Aufruf von *Putdata*
  - ***AD****DRTS:* erlaubt den Aufruf von *Adddata*
- Rechte müssen in der Capability zum Objekt gesetzt sein

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Datenzugriff in Hydra (2)

- Beispiel: Implementierung von Dateien
  - *Getdata* erlaubt das Lesen von Daten
  - *Putdata* erlaubt das Schreiben von Daten
  - *Adddata* erlaubt das Anhängen von Daten
- Entsprechende Rechte können pro Capability gesetzt werden



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

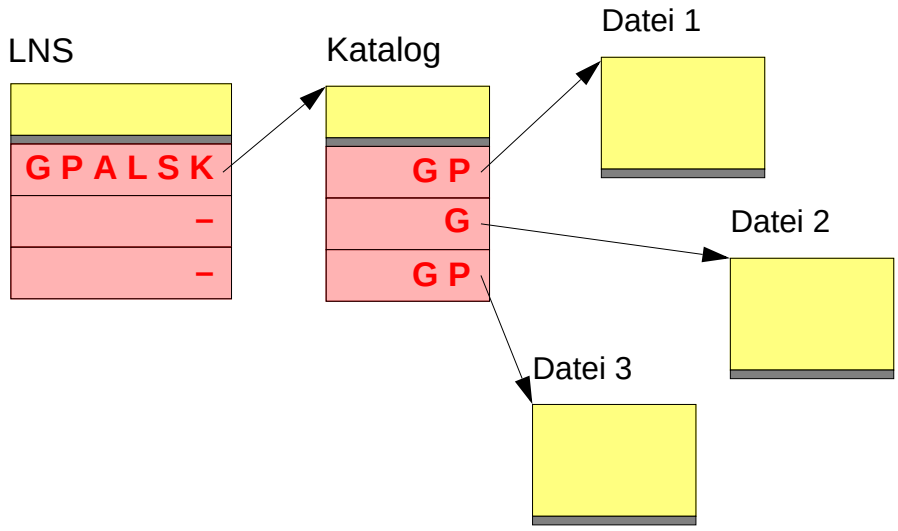
3 Zugriff auf Capabilities in Hydra

- Operationen auf der C-List:
  - *Load:* kopieren einer Capability aus der C-List eines Objekts in die C-List des LNS
  - *Store:* kopieren einer Capability aus der C-List des LNS in die C-List eines Objekts (dabei können Rechte maskiert werden)
  - *Append:* anfügen einer Capability in die C-List eines Objekts
  - *Delete:* löschen einer Capability aus der C-List eines Objekts
- Rechte:
  - ***L**OADRTS:* erlaubt Aufruf von *Load*
  - ***S**TORTS:* erlaubt Aufruf von *Store*
  - ***A**PPRTS:* erlaubt Aufruf von *Append*
  - ***K**ILLRTS:* erlaubt Aufruf von *Delete*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Zugriff auf Capabilities in Hydra (2)

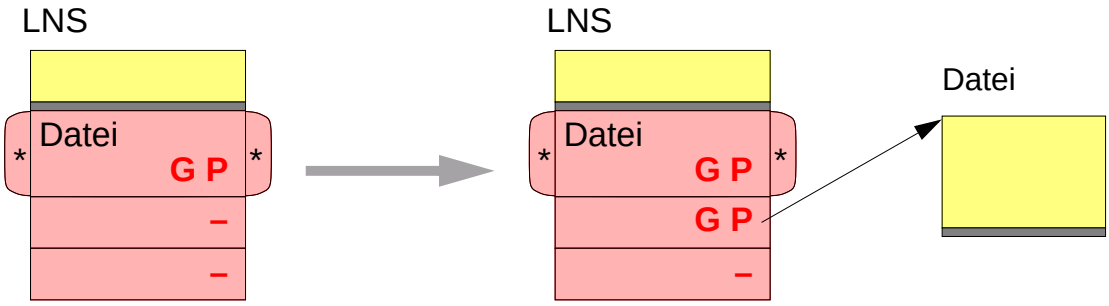
- Beispiel: Implementierung von Katalogen
  - *Load* erlaubt das Auflösen von Namen (Aufrufer bekommt die Capability)
  - *Store* und *Append* erlauben das Hinzufügen von Dateien zum Katalog
  - *Delete* erlaubt das Austragen von Dateien aus dem Katalog



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Objekterzeugung in Hydra

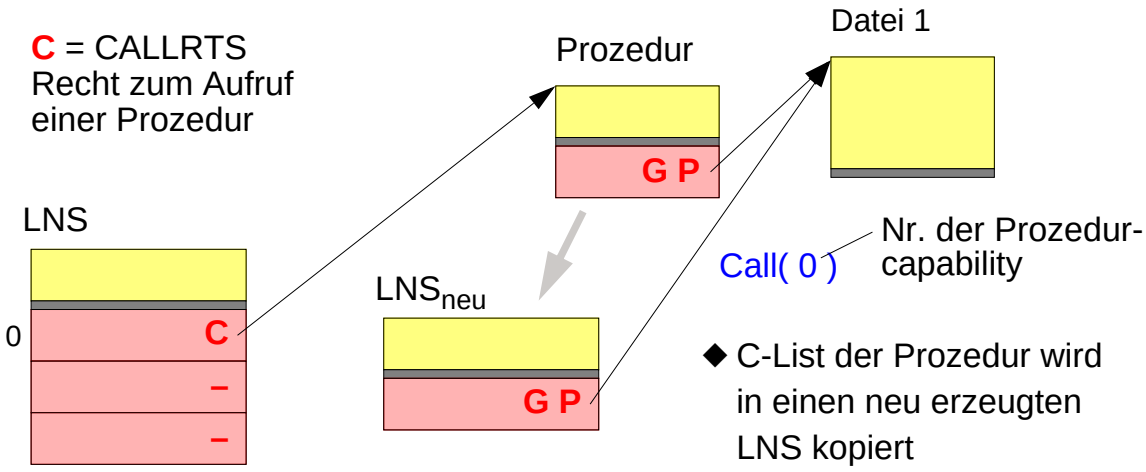
- Objekterzeugung über Erzeugungsschablonen *(Creation Templates)*
  - Erzeugungsschablone enthält den Typ des neu zu erzeugenden Objektes und eine Rechtemaske
  - nur die in der Maske angeschalteten Rechte werden dem Aufrufer in einer neuen Capability gegeben



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra

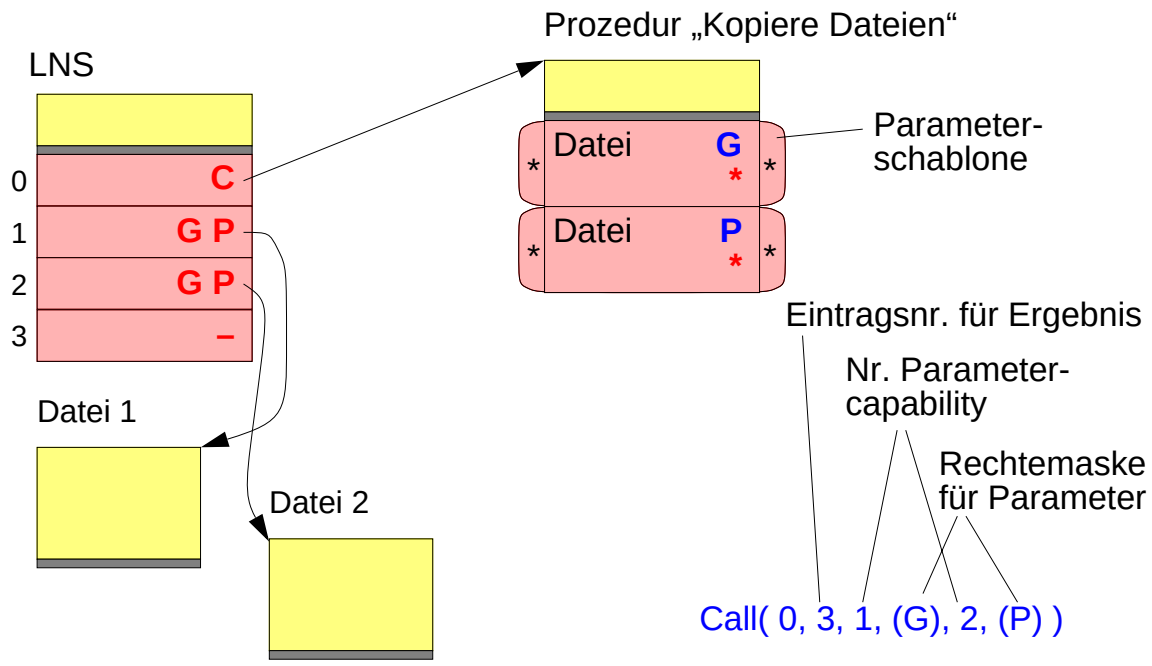
- Prozedur ist ein Objekt, aus dem beim Aufruf ein LNS des laufenden Prozesses erzeugt wird
  - neuer LNS wird aktueller Kontext (alte LNS stehen auf einem Stack; sie werden wieder aktiviert, wenn Prozedur zu Ende)
- Aufruf einer Prozedur



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (2)

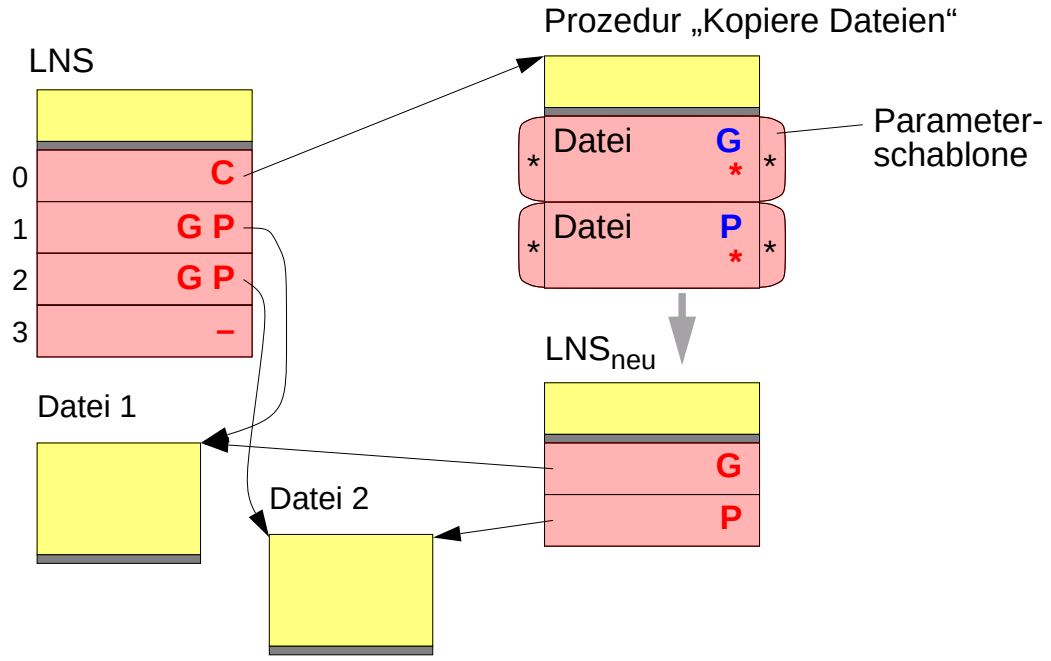
- Übergabe von Parametern
  - Beispiel: Prozedur zum Kopieren von Dateiinhalten



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (3)

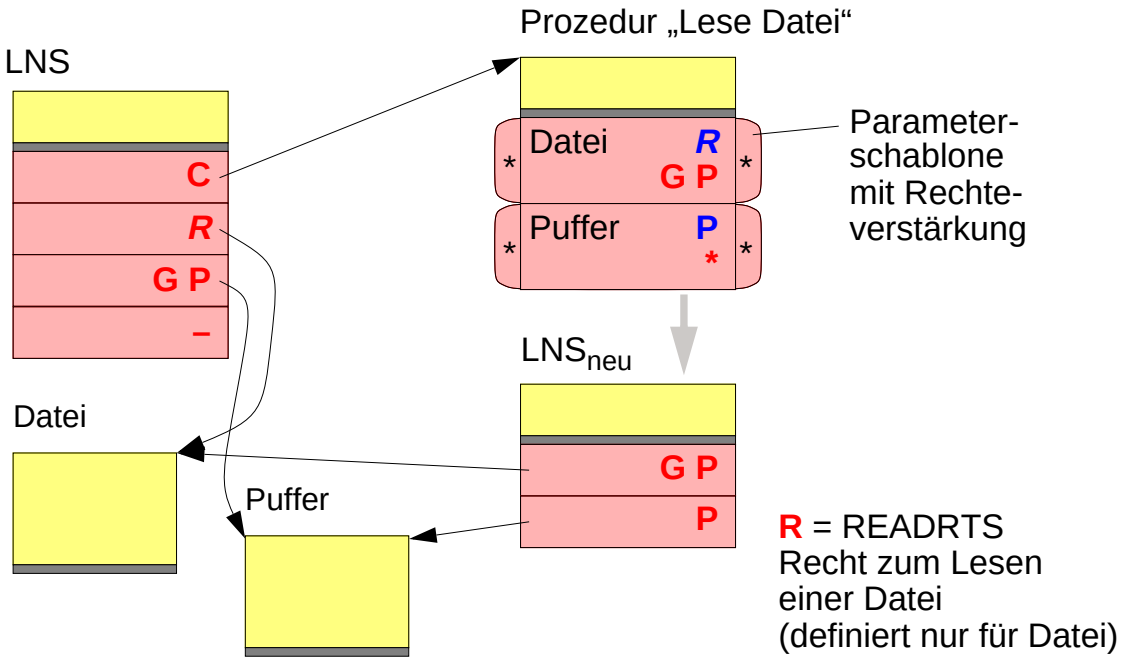
- Übergabe von Parametern
  - Beispiel: Prozedur zum Kopieren von Dateiinhalten



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (3)

- Verstärken von Rechten
  - Beispiel: Prozedur zum Lesen von Dateiinhalten in einen Puffer



© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997–2000
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Problem: Gegenseitiges Misstrauen

- Aufrufer misstraut einer Prozedur
  - Aufrufer möchte der Prozedur nur soviel Rechte einräumen wie nötig
- Aufgerufene Prozedur misstraut dem Aufrufer
  - Aufrufer soll nur soviel Rechte und Zugang bekommen wie erforderlich
- ★ Hydra Prozeduraufruf unterstützt diese Forderungen direkt
  - Aufrufer übergibt Capabilities, die nötig sind
  - Aufrufer kann Rechte bei der Übergabe maskieren und damit ausschalten
  - Aufrufer erhält nur Zugang zu einem definierten Ergebnis
  - Prozedur kann eigene Capabilities besitzen, die einem LNS zur Verfügung stehen und die dem Aufrufer verborgen bleiben können

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997–2000
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Problem: Gegenseitiges Misstrauen (2)

▲ Rechteverstärkung als Sicherheitslücke?

- ◆ Verstärkungsschablone wird nur an vertrauenswürdige Prozeduren ausgegeben und kann nicht einfach erzeugt werden

7 Problem: Modifikationen

■ Aufrufer möchte Modifikationen an und über Parameter ausschließen

- ◆ eine Prozedur soll nichts verändern können

■ Wegnehmen der entsprechenden Rechte langt nicht

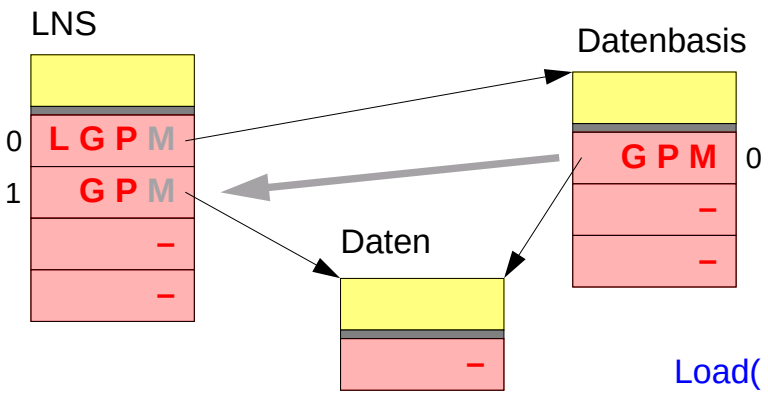
- ◆ Prozedur kann lesend zu neuen Capabilities gelangen und über diese Änderungen vornehmen (Transitivität)
- ◆ Rechteverstärkung könnte angewandt werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Problem: Modifikation (2)

★ Einführung des Modifikationsrechts ***M***DFYRTS

- ◆ für alle modifizierenden Operationen an Datenbereichen und C-Lists muss zusätzlich das Modifikationsrecht für das Objekt vorhanden sein
- ◆ Modifikationsrecht wird automatisch gelöscht, wenn eine Capability über einen Pfad geladen wird, auf dem eine der Capabilities kein Modifikationsrecht besitzt
- ◆ Modifikationsrecht kann nicht über Rechteverstärkung erlangt werden



es ist nicht möglich für den LNS, eine Capability auf die Daten zu erlangen, die Modifikationen zulässt

Load((0, 0), 1)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Problem: Modifikation (3)

- Parameterübergabe
  - Wegnahme des Modifikationsrecht bei Parametern stellt sicher, dass die aufgerufene Prozedur keinerlei Veränderungen beim Aufrufer durchführen kann

8 Problem: Ausbreitung von Capabilities

- Aufrufer will verhindern, dass eine übergebene Capability vom Aufgerufenen an einen Dritten weitergegeben wird *(Propagation Problem)*
  - Beispiel: Prozedur „Drucken“ soll niemandem eine Referenz auf die zu druckenden Daten weitergeben können

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

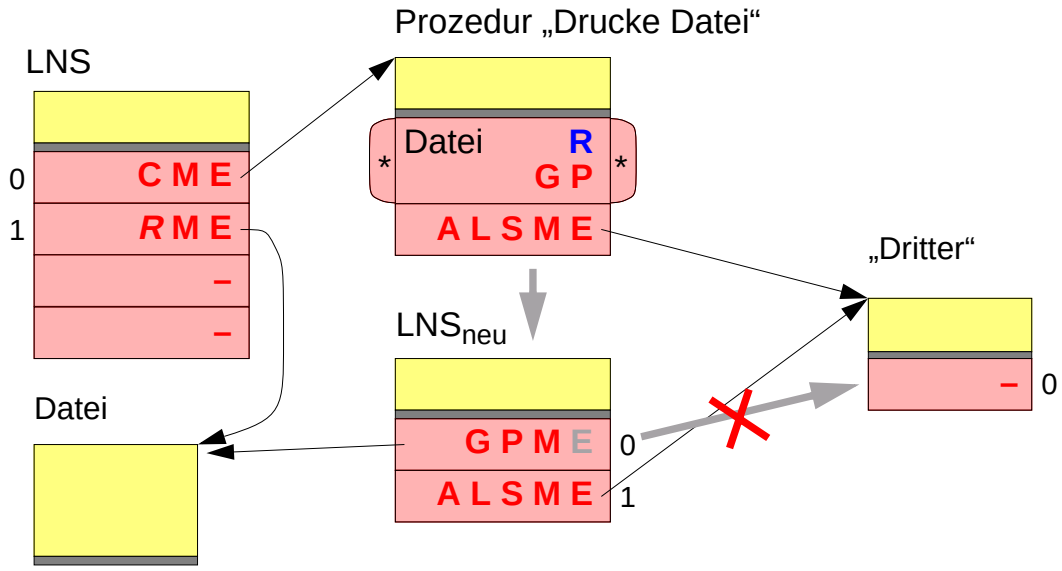
8 Problem: Ausbreitung von Capabilities (2)

- ★ Einführung des Environment-Rechts ***E****NVRTS*
  - für das Speichern oder Anfügen einer Capability an eine C-List muss die zu speichernde Capability selbst das Environment-Recht besitzen
  - Environment-Recht wird automatisch gelöscht, wenn eine Capability über einen Pfad geladen wird, auf dem eine der Capabilities kein Environment-Recht besitzt
  - Environment-Recht kann nicht über Rechteverstärkung erlangt werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Problem: Ausbreitung von Capabilities (3)

- Versuchte Weitergabe einer Capability an einen Dritten



Call(0, 2, 1, (–E)) Store(0, (1, 0))

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Problem: Aufbewahrung von Capabilities

- Aufrufer möchte sicher sein, dass Aufgerufener keine Capabilities nach der Bearbeitung des Aufrufs zurückbehalten kann *(Conservation Problem)*

★ Environment-Recht zusammen mit dem Aufrufmechanismus genügt

- ◆ Aufgerufener kann Capability ohne ENVRTS nicht weitergeben und folglich nicht abspeichern
- ◆ der LNS des Aufrufs wird mit Beendigung des Aufrufs vernichtet, so dass die übergebenen Capabilities nicht zurückbehalten werden können
- ◆ ENVRTS wirkt transitiv, so dass auch die über eine Parameter-Capability gewonnenen Capabilities nicht weitergegeben werden können

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

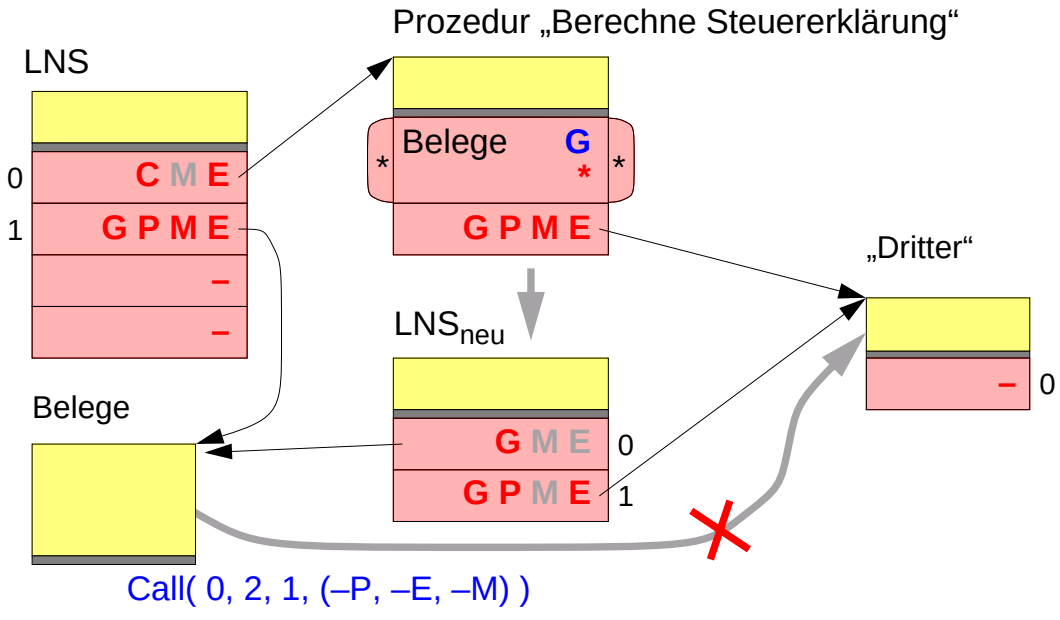
10 Problem: Informationsflussbegrenzung

- Aufrufer möchte die Verbreitung von Informationen aus übergebenen Parametern einschränken *(Confinement Problem)*
  - selektiv: bestimmte Informationen sollen nicht nach außen gelangen
  - global: gar keine Informationen sollen nach außen gelangen
  - ENVRTS ist nicht ausreichend, da Prozedur den Dateninhalt von Parameterobjekten kopieren könnte
    (ENVRTS wirkt nur auf die Weitergabe von Capabilities)
- Hydra realisiert nur globale Informationsflussbegrenzung
- ★ Modifikationsrecht auf der Prozedur-Capability
  - wenn kein Modifikationsrecht vorhanden ist, werden bei allen in den LNS übernommenen Capabilities die Modifikationsrechte ausgeschaltet
    (gilt jedoch nicht für Parameter)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

10 Problem: Informationsflussbegrenzung (2)

- Beispiel: Prozedur zur Steuerberechnung
  - die übergebenen Beleg- und Buchhaltungsdaten sollen nicht weitergegeben werden können



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

11 Problem: Initialisierung

- Initialisierung von Objekten durch Prozeduren
  - Übergabe eines Objekts und verschiedener Capabilities, mit denen das Objekt initialisiert werden soll
  - Problem: Parameter-Capabilities müssen Environment-Recht besitzen (sonst ist das zu initialisierende Objekt nicht arbeitsfähig), gleichzeitig soll aber die Ausbreitung solcher Capabilities eingeschränkt werden
    - Lösung: Wegnahme des Modifikationsrechts auf der Prozedurcapability
  - Problem: Es muss verhindert werden, dass die Prozedur in das zu initialisierende Objekt eigene oder fremde Capabilities einsetzt, so dass es später Einfluss auf das zu initialisierende Objekt nehmen kann
- Beispiel: Prozedur zur Initialisierung eines Katalogs bekommt Capabilities auf die entsprechenden Dateien
  - es soll sichergestellt werden, dass Prozedur keine eigenen Dateicapabilities in den Katalog einfügt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

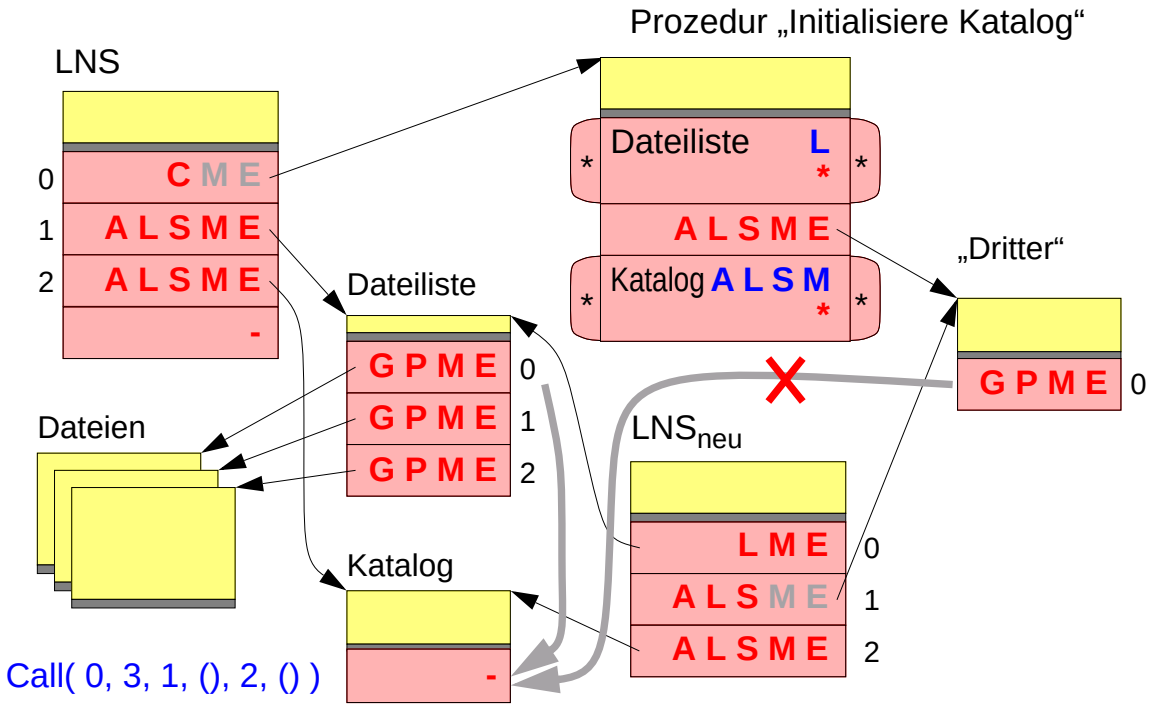
11 Problem Initialisierung (2)

- ★ Environment-Recht auf der Prozedur-Capability
  - wenn kein Environment-Recht vorhanden ist, werden bei allen in den LNS übernommenen Capabilities die Environment-Rechte ausgeschaltet (gilt jedoch nicht für Parameter)
  - durch das fehlende Environment-Recht können alle bereits vorhandenen Capabilities nicht in das zu initialisierende Objekt gespeichert werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

11 Problem: Initialisierung (3)

- Beispiel: Initialisierung eines Katalogs



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

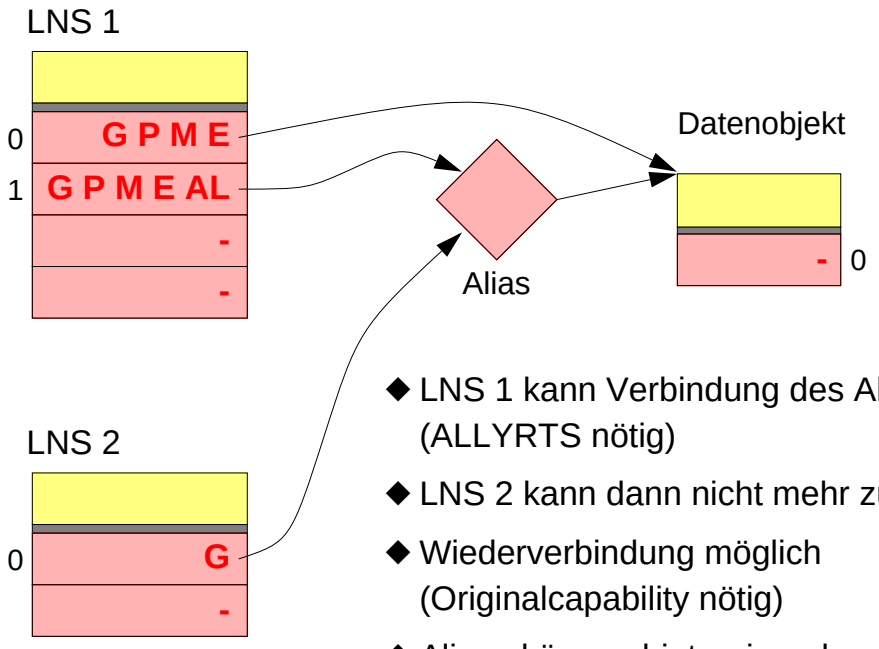
12 Rückruf von Capabilities

- Anwender möchte ausgegebene Capabilities für ungültig erklären
  - sofortiger Rückruf — Rückruf nach einiger Zeit erst wirksam
  - dauerhafter Rückruf — Rückruf nur zeitlich begrenzt wirksam
  - selektiver Rückruf — Rückruf für alle Benutzer eines Objekts
  - partieller Rückruf — Rückruf aller Rechte an einem Objekt
  - Recht zum Rückruf; Rückruf des Rückrufrechts
- ★ Hydra setzt sogenannte Aliase ein
  - Alias ist eine Indirektionsstufe zu Capabilities
  - Statt auf ein Objekt können Capabilities auf Aliase verweisen und diese wiederum auf andere Aliase oder schließlich auf das eigentliche Objekt
  - Verbindung vom Alias zum Objekt kann gelöst werden: Fehler beim Zugriff
  - Recht zum Lösen der Verbindung ***AL***LYRTS
    (engl. *ally* = verbünden)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

12 Rückruf von Capabilities (2)

- Beispiel: Weitergabe einer rückrufbaren Capability

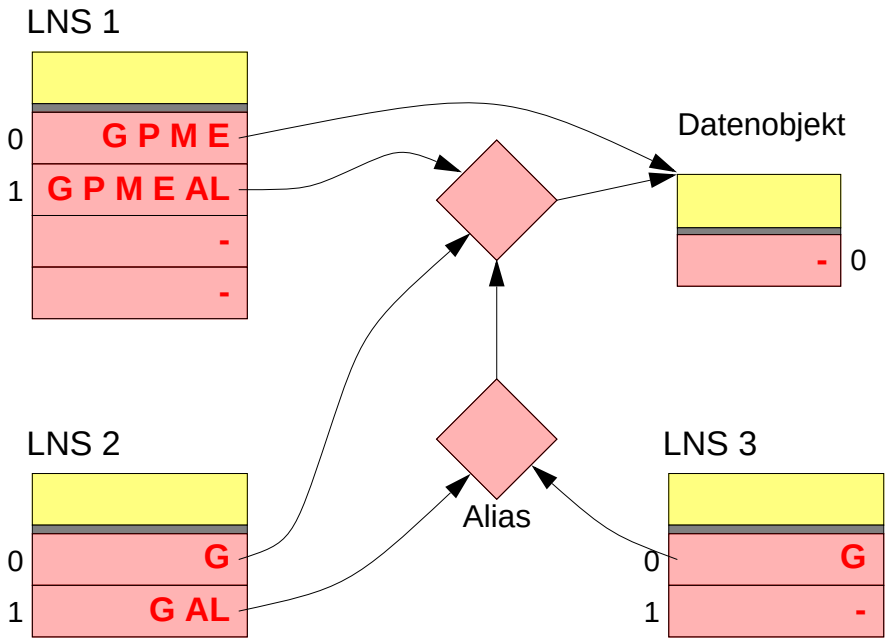


- LNS 1 kann Verbindung des Alias lösen (ALLYRTS nötig)
- LNS 2 kann dann nicht mehr zugreifen
- Wiederverbindung möglich (Originalcapability nötig)
- Aliase können hintereinander auftreten

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

12 Rückruf von Capabilities (3)

- Beispiel: Aliasketten



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

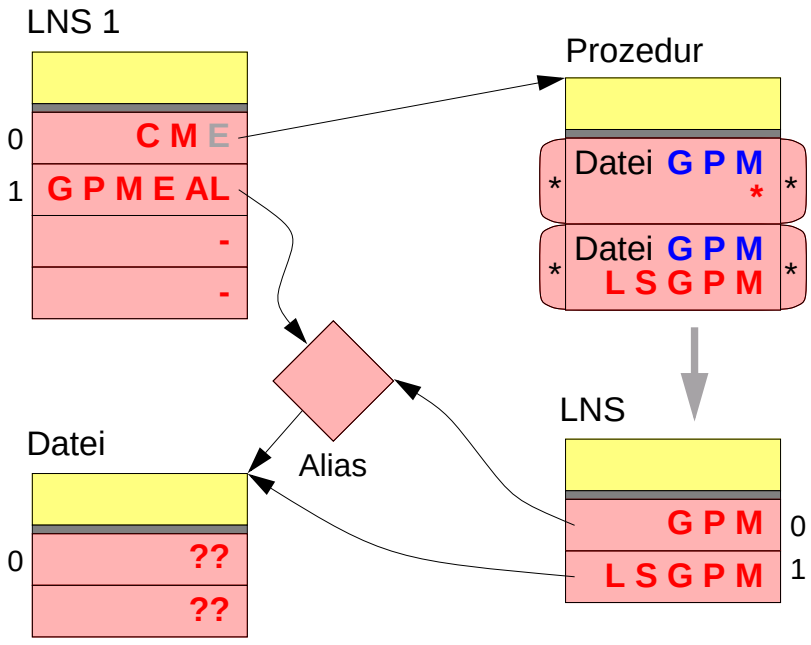
12 Rückruf von Capabilities (4)

- ▲ Problem: Rückruf während der Bearbeitung eines Objekts
  - ◆ inkonsistente Zustände möglich
- ★ Lösung in Hydra
  - ◆ Parameter-Capabilities, die durch eine rechteverstärkende Parameterschablone angenommen werden, zeigen auf das Originalobjekt
- ▲ Nachteil
  - ◆ nicht vertrauenswürdige Prozeduren können rückruffreie Capability erlangen
  - ◆ Problem fällt in die selbe Kategorie wie rechteverstärkende Parameterschablonen an sich

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

12 Rückruf von Capabilities (5)

- ■ Beispiel:



Call(0, 2, 1, (–E), 1, (–E))

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

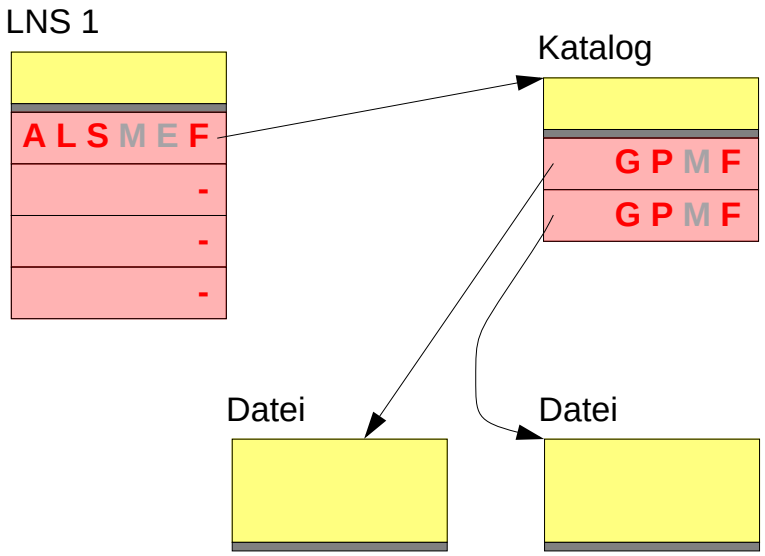
13 Garantierter Zugriff

- Schutz vor Rückruf
  - ◆ Modifizierende Benutzer eines Objekts
    - kooperierende Benutzer: Rückruf keine Gefahr
    - nicht kooperierende Benutzer: Rückruf nötig
  - ◆ Benutzer eines Objekts ohne modifizierende Zugriffe
    - Aufruf von Prozeduren ist unkritisch, da Aufrufe nicht rückrufbar sind
    - lesende Zugriffe auf Objekte sind kritisch:
      Verhindern des Rückrufs ist jedoch nicht ausreichend
      – Daten im Objekt könnten gelöscht oder verfälscht werden
      – Capabilities im Objekt oder deren Rechte könnten entfernt werden
- ★ Hydra führt das Einfrierrecht *(Freeze right **F**RZRTS)* ein
  - ◆ Einfrieren nimmt Modifikationsrecht weg
  - ◆ ein Objekt kann nur gefroren werden, wenn alle Capabilities in der C-List bereits das Einfrierrecht haben

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

13 Garantierter Zugriff (2)

- Beispiel:



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

14 Bewertung von Hydra

- Hydra demonstrierte die Beherrschbarkeit einer ganzen Reihen von Sicherheitsproblemen
  - ◆ Ergebnisse flossen in eine ganze Reihe von Systemen
  - ◆ reine Capability-basierte Systeme haben sich jedoch nie durchgesetzt
- Hydras Probleme
  - ◆ lagen im wesentlichen nicht am Capability-Mechanismus
  - ◆ es gabe keine vernünftigen Editoren und Compiler
  - ◆ Hardware besaß keine Spezialhardware zur Unterstützung von Paging

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.6 Kryptographische Maßnahmen

- Verschlüsseln und Entschlüsseln vertraulicher Daten
  - ◆ aus den verschlüsselten Daten soll die Originalinformation nur mit Hilfe eines Schlüssels restauriert werden können
  - ◆ Schlüssel bleibt geheim
- Authentisierung
  - ◆ Empfänger kann verifizieren, wer der Absender ist
- Sicherer Kanal
  - ◆ gesendete Informationen können nicht gefälscht und verfälscht werden
  - ◆ nur der adressierte Empfänger kann die Informationen lesen
  - ◆ Empfänger kann den Absender authentisieren

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.6 Kryptographische Maßnahmen (2)

- Funktionen
  - Verschlüsselungsfunktion *E (encrypt): E(K_1, T)* → *C*
  - Entschlüsselungsfunktion *D (decrypt): D(K_2, C)* → *T*
  - *K* = Schlüssel, *T*= zu verschlüsselnder Text/Daten
- Verwandte Schlüssel
  - es gilt: K_1 und K_2 sind verwandt, wenn gilt: $\forall T: D(K_2, E(K_1, T)) = T$
- Symmetrisches Verschlüsselungsverfahren
  - es gilt: $K_1 = K_2$

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.6 Kryptographische Maßnahmen (3)

- Forderungen an ein Verschlüsselungsverfahren
  - Wenn K_2 unbekannt ist, soll es sehr aufwendig sein aus $E(K_1, T)$ das *T* zu ermitteln (Entschlüsselungsangriff)
  - Es soll sehr aufwendig sein aus *T* und $E(K_1, T)$ den Schlüssel K_1 zu ermitteln (Klartextangriff)
  - Bei asymmetrischen Verfahren soll es sehr aufwendig sein, aus K_1 den Schlüssel K_2 zu ermitteln und umgekehrt.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Monoalphabetische Verfahren

- Verfahren nach Caesar

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E

  - Verschlüsselungsfunktion: $E: M \rightarrow (M + k) \bmod 26$
  - k ist variierbar (26 Möglichkeiten)

- Zufällige Substitution

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	H	A	J	U	L	G	N	S	P	W	R	O	T	C	V	Y	X	M	Z	K	B	I	D	E

  - 26! Möglichkeiten

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Monoalphabetische Verfahren (2)

- ★ Nachteil
  - vollständiges Ausprobieren möglich bei Caesar
  - Häufigkeitsanalyse der Buchstaben
    - für eine Sprache gibt es häufigere Buchstaben, z.B. e im Deutschen
    - durch die Häufigkeitsanalyse können die Möglichkeiten stark eingeschränkt werden; vollständiges Probieren wird ermöglicht

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Polyalphabetische Verschlüsselung

- Einsatz von vielen Abbildungen, die durch einen Schlüssel ausgewählt werden
  - Beispiel: Vigenére (Caesar-Verschlüsselung mit zyklisch wiederholten Folgen von Verschiebungswerten)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
													...													
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Polyalphabetische Verschlüsselung (2)

- Auswahl der Zeile durch den entsprechenden Buchstaben des Schlüsselwortes

	W	I	C	H	T	I	G	E	N	A	C	H	R	I	C	H	T
	G	E	H	E	I	M	G	E	H	E	I	M	G	E	H	E	I
	C	M	J	L	B	U	M	I	U	E	K	T	X	M	J	L	B

Originaltext

Schlüsselwort (wiederholt)

verschlüsselter Text

- Gilt als nicht sicher
  - Koinzidenzanalyse
  - Häufigkeitsanalysen und Brute force Attacke

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Polyalphabetische Verfahren (3)

- Koinzidenz
  - Wahrscheinlichkeit für zwei gleiche Buchstaben untereinander bei umbrechendem Text
    - zufällige Buchstabenwahl: 3,8%
    - englischer Text: 6,6%
  - Brechen polyalphabetischer Verfahren
    - Bestimmen der Koinzidenz für verschiedene Textlängen
    - Textlänge mit höchster Koinzidenz ist wahrscheinlich Schlüsseltextlänge
    - danach Häufigkeitsanalyse pro Buchstabe des Schlüsseltexts

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

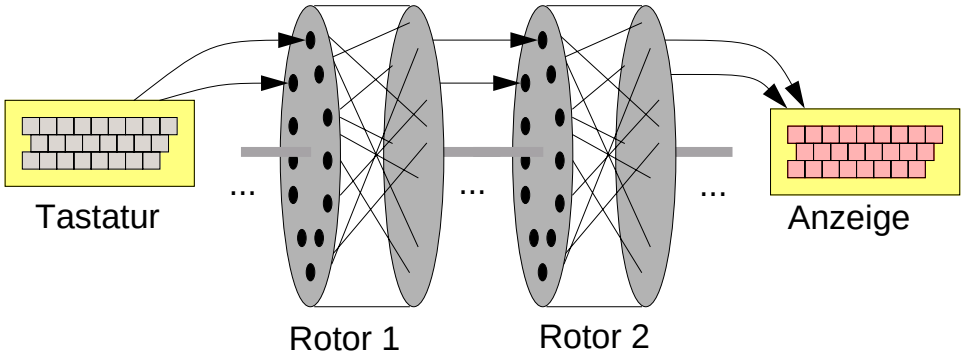
3 One-Time Pad Verfahren

- Theoretisch sicheres Verfahren
  - Liste von Zufallszahlen (soviele wie Zeichen in der Nachricht): *r[i]*
  - Zeichen *z[i]* der Nachricht wird verschlüsselt mit $c[i] = (z[i] + r[i]) \bmod 26$
  - Empfänger braucht die gleiche Liste
  - theoretisch sicher, da aus dem *c[]* nicht auf *z[]* geschlossen werden kann
- Praktisch unbrauchbar
  - echte Zufallszahlen nötig
  - lange Liste nötig
    - jede Liste kann nur einmal verwendet werden
    - Liste muss so lang wie die Nachricht sein

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Rotormaschinen

- Drehende Scheiben verändern ständig die Permutation



  - Einstellen einer Anfangsposition für die Rotoren
  - bei jedem Zeichen wird erster Rotor um eine Position weitergedreht
  - zweiter Rotor rotiert mit niedrigerer Geschwindigkeit
  - zum Entschlüsseln sind entsprechende Gegenstücke nötig

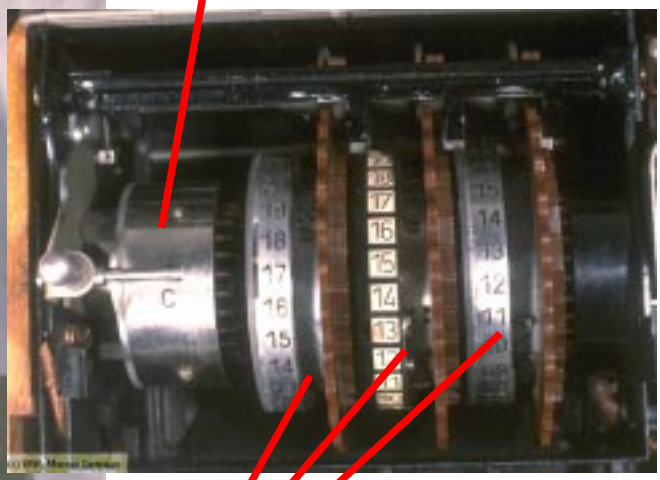
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Rotormaschinen (2)

- Enigma
  - deutsche Chiffriermaschine aus dem zweiten Weltkrieg
  - drei Rotore und Reflektor
    - Reflektor leitet Strom wieder bei einer anderen Position durch die Rotoren zurück: Verfahren wird symmetrisch
    - Entschlüsseln mit den gleichen Rotoren möglich
- Verfahren gilt als nicht sicher
  - Brute force Attacke: *Collosus* Computer
- Schlüsseldemo
  - http://www.ugrad.cs.jhu.edu/~russell/classes/enigma/

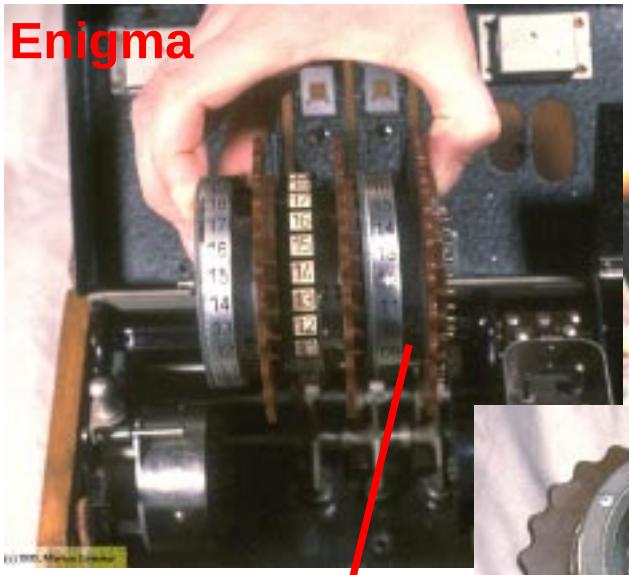
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Rotormaschinen



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Rotormaschinen (2)



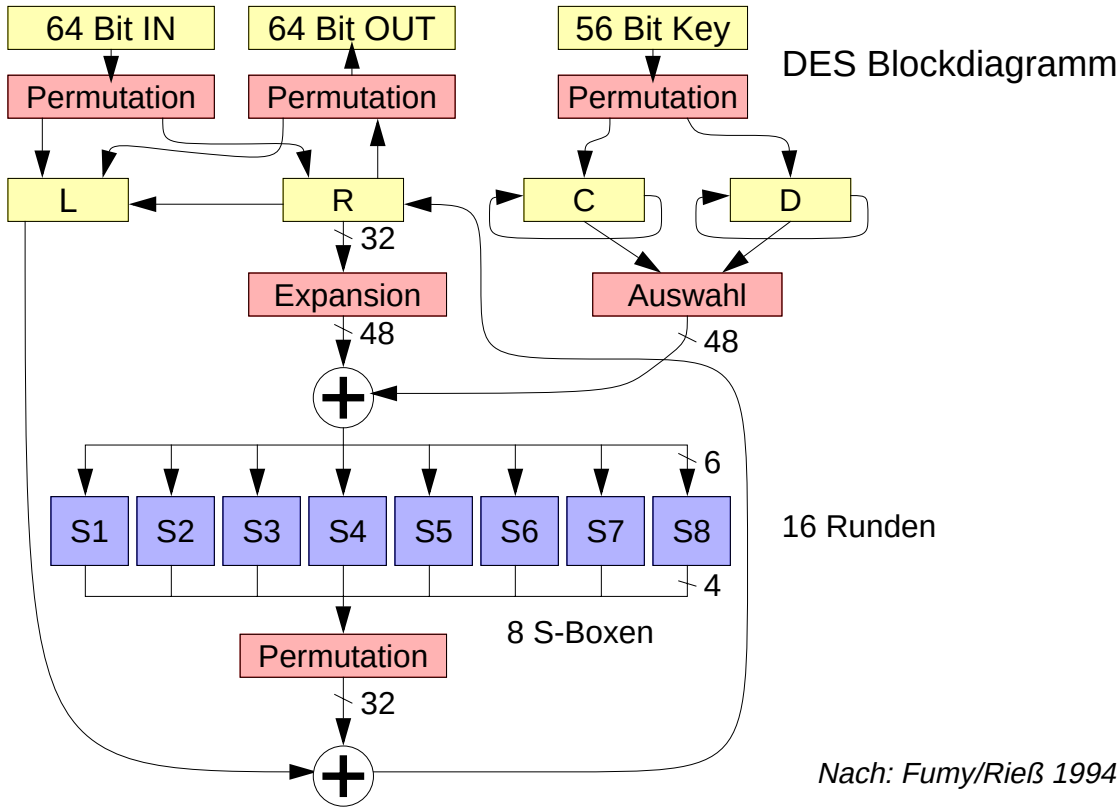
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Heutige symmetrische Verfahren

- DES *(Data Encryption Standard, 1977)*
  - entwickelt von IBM
  - amerikanischer Standard (Kriegswaffe)
  - blockorientiertes Verfahren (64 Bit Block, 56 Bit Schlüssel)
  - 16 Runden
  - gilt heute als nicht mehr ganz sicher, da Rechenleistung von Großrechnern oder Rechenverbünden zum Brechen manchmal ausreicht

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Heutige symmetrische Verfahren (2)



DES Blockdiagramm

Nach: Fumy/Rieß 1994

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Heutige symmetrische Verfahren (3)

- Tripple DES
  - ◆ dreifache Verschlüsselung mit DES
  - ◆ Nutzung von drei oder mindestens zwei verschiedenen Schlüsseln
- IDEA *(International Data Encryption Algorithm)*
  - ◆ Alternative zu DES
  - ◆ 64 Bit Blockgröße
  - ◆ 128 Bit Schlüssel
  - ◆ keine Permutationen und S-Boxen
  - ◆ stattdessen: Addition, Multiplikation und XOR
  - ◆ 8 Runden und Output-Transformation
  - ◆ Einsatz: z.B. PGP *(Pretty Good Privacy)*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Beispiel: UNIX Passwörter

- Passwörter wurden zunächst im Klartext gespeichert
  - ◆ Passwortdatei muss streng geschützt werden
  - ◆ strenger Schutz oft nicht möglich (z.B. Backup der Platte)
  - ◆ Superuser kann die Passwörter von Benutzern einsehen
- Verschlüsseln der Passwörter
  - ◆ nur die verschlüsselte Version wird gespeichert
  - ◆ verschlüsselte Passwörter dürfen nicht leicht entschlüsselt werden können
- ▲ Ausprobieren von Passwörtern
  - ◆ Benutzer wählen Namen und Gegenstände als Passwort
  - ◆ Verschlüsseln von gängigen Begriffen und Vergleich mit verschlüsselt gespeicherten Passwörtern
  - ◆ Verschlüsselungszeit fließt mit ein in die Sicherheitsbetrachtung

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Beispiel: UNIX Passwörter (2)

- Heutiges Verfahren
  - zufällige Auswahl eines von 4096 Werten *(Salt)*
  - der Salt fließt mit in die Verschlüsselung ein, so dass ein und dasselbe Passwort in 4096 Varianten vorkommen kann
  - Verschlüsselung mit DES
  - Zugriff auf verschlüsselte Passwörter wird weitestgehend verhindert (Shadow-Passwortdatei)
- ★ Vorteil
  - Ausprobieren von Passwörtern benötigt mehr Zeit
  - Vergleich zweier Passwörter weitgehend unmöglich

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Beispiel: UNIX Passwörter (3)

- Politik am IMMD
  - Mindestlänge 8 Zeichen
  - mindestens 5 verschiedene Zeichen
  - mindestens 3 Zeichenklassen (Groß-, Kleinbuchstaben, Ziffern, Sonderzeichen)
  - keine Wiederholungen von Zeichenfolgen erlaubt
  - keine aufeinanderfolgenden Zeichen erlaubt, z.B. "123"
  - ...
  - Begriffe, Namen, etc. werden ausgeschlossen und müssen hinreichend verfremdet sein
- ★ Angriff durch Ausprobieren wird weitestmöglich erschwert

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Heutige asymmetrische Verfahren

- RSA (Rivest, Shamir und Adleman)
  - Öffentlicher Schlüssel (zum Verschlüsseln) besteht aus *(e, N)*
  - Ein Block *M* wird verschlüsselt durch: $C = E((e, N), M) = M^e \bmod N$
  - *C* wird entschlüsselt durch: $M = D((d, N), C) = C^d \bmod N$
  - Wahl der Schlüssel:
    - Es muss gelten $\forall M : (M^e)^d = M \bmod N$
    - Aus Kenntnis von *e* und *N* darf *d* nur mit hohem Aufwand ermittelbar sein
  - Lösung:
    - $N = pq$ mit *p* und *q* zwei hinreichend große Primzahlen
    - zufällige Wahl von *d*, teilerfremd zu $(p-1)(q-1)$
    - Berechnung von *e* aus der Bedingung: $ed = 1 \bmod ((p-1)(q-1))$
  - Es ist aufwendig, die Primfaktoren von *N* zu berechnen (mit diesen wäre es möglich *d* zu ermitteln)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

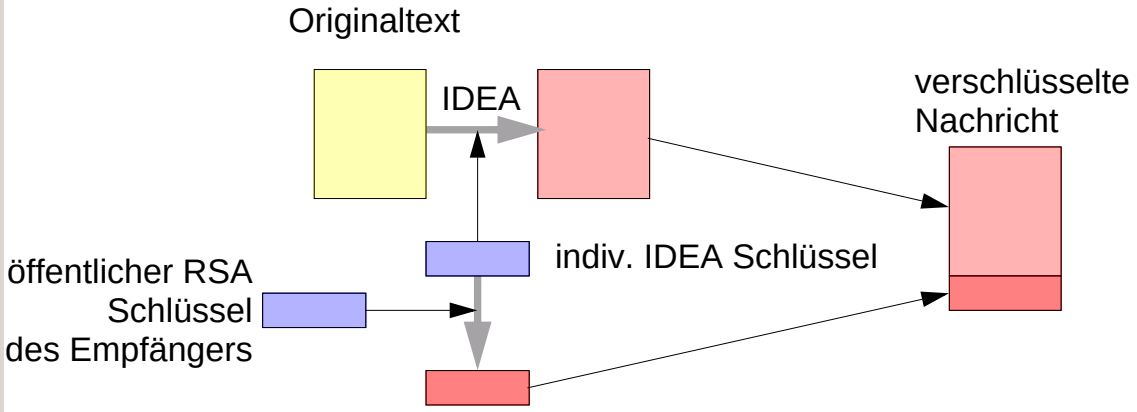
7 Heutige asymmetrische Verfahren (2)

- Vorteil asymmetrischer Verfahren (*Public key*-Verfahren)
  - nur ein Schlüsselpaar pro Teilnehmer nötig
    (sonst ein Schlüsselpaar pro Kommunikationskanal!)
  - Schlüsselverwaltung erheblich vereinfacht
    - jeder Teilnehmer erzeugt sein Schlüsselpaar und
    - veröffentlicht seinen öffentlichen Schlüssel
  - Authentisierung durch digitale Unterschriften möglich
  - gilt als sicher bei hinreichend großer Schlüssellänge (1024 Bit)
- Nachteil
  - relativ langsam berechenbar
  - gemischter Betrieb von asymmetrischen und symmetrischen Verfahren zur Geschwindigkeitssteigerung

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Heutige asymmetrische Verfahren (3)

- Beispiel: PGP Verschlüsselung



  - Daten werden mit einem individuellen Schlüssel IDEA-verschlüsselt
  - IDEA-Schlüssel wird RSA-verschlüsselt der Nachricht angehängt

- ★ Nachricht an mehrere Adressaten verschickbar
  - lediglich der IDEA-Schlüssel muss in mehreren Varianten verschickt werden (je eine Version verschlüsselt mit dem öffentl. Schlüssel des Empfängers)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften

- Authentisierung des Absenders
  - Bilden eines Hash-Wertes über die zu übermittelnde Nachricht
    - Hash-Wert ist ein Codewort fester Länge
    - es ist unmöglich oder nur mit hohem Aufwand möglich, für einen gegebenen Hash-Wert eine zugehörige Nachricht zu finden
  - Verschlüsseln des Hash-Wertes mit dem geheimen Schlüssel des Absenders (digitale Unterschrift, digitale Signatur)
  - Anhängen des verschlüsselten Hash-Wertes an die Nachricht
  - Empfänger kann den Hash-Wert mit dem öffentlichen Schlüssel des Absenders dechiffrieren und mit einem selbst berechneten Hash-Wert der Nachricht vergleichen
  - stimmen beide Werte überein muss die Nachricht vom Absender stammen, denn nur der besitzt den geheimen Schlüssel

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften (2)

- Kombination mit Verschlüsselung
  - erst signieren
  - dann mit dem öffentlichen Schlüssel des Adressaten verschlüsseln
- Reihenfolge wichtig
  - Man signiere nichts, was man nicht entschlüsseln kann
- Heute gängiges Hash-Verfahren
  - MD5
  - 128 Bit langer Hash-Wert

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften (3)

- Woher weiß ich, dass ein öffentlicher Schlüssel authentisch ist?
  - Ich bekomme den Schlüssel vom Eigentümer (persönlich, telefonisch).
    - Hash-Wert auf öffentlichen Schlüsseln, die leichter zu überprüfen sind (Finger-Prints)
  - Ich vertraue jemandem (Bürge), der zusichert, dass der Schlüssel authentisch ist.
    - Schlüssel werden von dem Bürgen signiert.
    - Bürge kann auch eine ausgezeichnete Zertifizierungsstelle sein.
    - Netzwerk von Zusicherungen auf öffentliche Schlüssel *(Web of Trust)*
  - Möglichst weite Verbreitung von öffentlichen Schlüsseln ereichen (z.B. PGP: Webserver als Schlüsselserver)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften (4)

- Mögliche Probleme von Public key-Verfahren
  - Geheimhaltung des geheimen Schlüssels
    (Time sharing-System, Backup; Schlüsselpasswort / Pass phrase)
  - Vertrauen in die Programme (z.B. PGP)
  - Ausspähung während des Ver- und Entschlüsselungsvorgangs

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.7 Authentisierung im Netzwerk

- Viele Klienten, die viele Dienste in Anspruch nehmen wollen
  - Dienste *(Server)* wollen wissen welcher Benutzer *(Principal)*, den Dienst in Anspruch nehmen will (z.B. zum Accounting, Zugriffsschutz, etc.)
  - Im lokalen System reicht die (durch das Betriebssystem) geschützte Benutzerkennung (z.B. UNIX UID) als Ausweis
  - Im Netzwerk können Pakete abgefangen, verfälscht und gefälscht werden (einfache Übertragung einer Benutzerkennung nicht ausreichend sicher)
- Public key-Verfahren
  - Authentisierung durch digitale Unterschrift (mit geheimen Schlüssel des Senders) und Verschlüsseln (mit öffentlichem Schlüssel des Empfängers)
  - Nachteile
    - jeder Dienst benötigt sicheren Zugang zu allen öffentlichen Schlüsseln
    - Verschlüsseln und Signieren mit RSA ist sehr teuer

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.7 Authentisierung im Netzwerk (2)

★ Einsatz von Authentisierungsdiensten

- ◆ zentraler Server, der alle Benutzer kennt
- ◆ Authentisierungsdienst garantiert einem Netzwerkdienst, dass ein Benutzer auch der ist, der er vorgibt zu sein

■ Benutzerausweis

- ◆ der Authentisierungsdienst erkennt den Benutzer anhand eines geheimen Schlüssels oder Passworts
- ◆ der Schlüssel ist nur dem Authentisierungsdienst und dem Benutzer bekannt

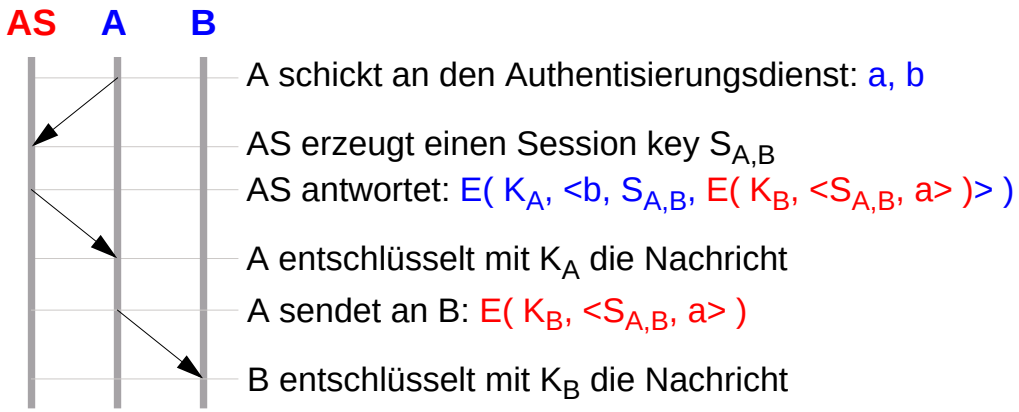
■ Vorgang

- ◆ Benutzer *(Principal)* will mit einem Programm *(Client)* einen Dienst *(Server)* in Anspruch nehmen
- ◆ durch geeignetes Protokoll erhalten Client und Server jeweils einen nur ihnen bekannten Schlüssel, mit dem sie ihre Kommunikation verschlüsseln können *(Session key)*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Authentisierungsdienst

■ A will den Dienst B in Anspruch nehmen (Nach Needham-Schröder):



- ◆ K_X ist der geheime Schlüssel, den nur Authentisierungsdienst und X kennen
- ◆ nach dem Protokollablauf kennen sowohl A und B den Session key
- ◆ A weiß, dass nur B den Session key kennt
- ◆ B weiß, dass nur A den Session key kennt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Authentisierungsdienst (2)

▲ Problem

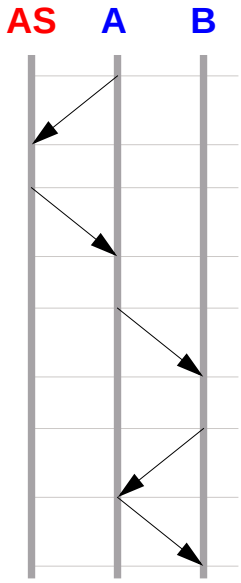
- ◆ letzte Nachricht von A an B könnte aufgefangen und später erneut ins Netz gegeben werden *(Replay attack)*
- ◆ Folge: Kommunikation zwischen A und B kann gestört werden

★ Korrektur durch zusätzliches Versenden einer Verbindungsbestätigung durch B und A

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Authentisierungsdienst mit Bestätigung

■ Bestätigung enthält Einmalinformation *(Nonce)*



A schickt an den Authentisierungsdienst: a, b, i_a

AS erzeugt einen Session key $S_{A,B}$

AS antwortet: $E(K_A, <i_a, b, S_{A,B}, E(K_B, <S_{A,B}, a>)>)$

A entschlüsselt mit K_A die Nachricht

A sendet an B: $E(K_B, <S_{A,B}, a>)$

B entschlüsselt mit K_B die Nachricht

B schickt an A: $E(S_{A,B}, i_b)$

A antwortet an B: $E(S_{A,B}, i_b+1)$

- ◆ ein Wiedereinspielen der Nachricht $E(K_B, <S_{A,B}, a>)$ oder $E(S_{A,B}, i_b+1)$ wird erkannt und kann ignoriert werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Authentisierungsdienst mit Bestätigung (2)

▲ Problem

- ◆ Aufzeichnen von $E(K_B, <S_{A,B}, a>)$ und
- ◆ Brechen von $S_{A,B}$ erlaubt das Aufbauen einer Verbindung.
- ◆ ein Dritter kann dann die erste Bestätigung abfangen und die zweite Bestätigung verschicken

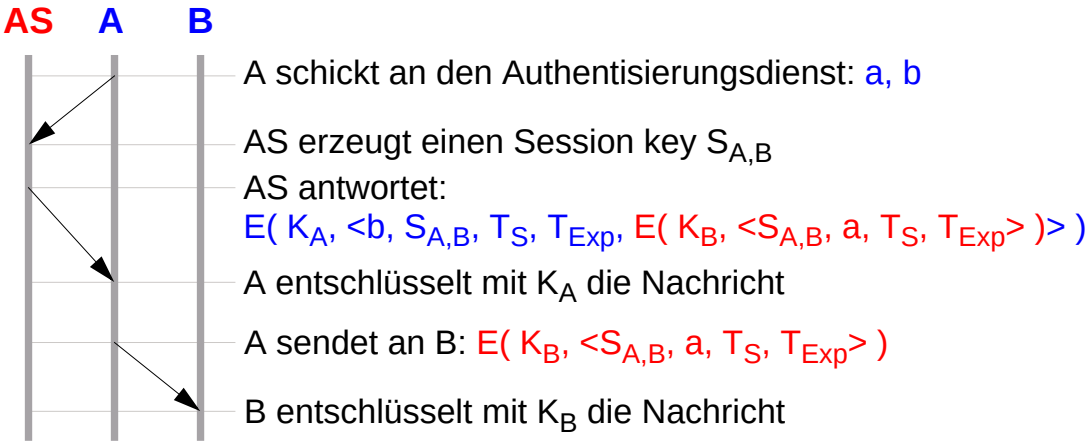
★ Lösung

- ◆ Einführung von Zeitstempeln *(Time stamp)* und Angaben zur Lebensdauer *(Expiration time)*

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Authentisierungsdienst mit Zeitstempeln

■ Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



- ◆ T_S = Zeitstempel der Nachrichtenerzeugung
- ◆ T_{Exp} = maximale Lebensdauer der Nachricht
- ◆ aufgezeichnete Nachricht kann nach kurzer Zeit (z.B. 5min) nicht nocheinmal zum Aufbau einer Verbindung verwendet werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

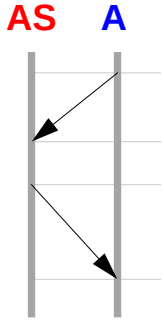
4 Beispiel: Kerberos

- Kerberos V5
  - Softwaresystem implementiert Weiterentwicklung des Needham-Schröder-Protokolls
  - entwickelt am MIT seit 1986
- Ziel
  - Authentisierung und Erzeugung eines gemeinsamen Schlüssels durch den vertrauenswürdigen Kerberos-Server
- Idee
  - Trennung von Authentisierungsdienst und Schlüsselerzeugung
  - reduziert die nötige Übertragung einer Identifikation oder eines Passworts zum Kerberos-Server

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos (2)

- Benutzer holt sich zunächst ein Ticket vom Authentisierungsdienst



A schickt an den Authentisierungsdienst: a, tgs, T_{Exp}, i_a

AS erzeugt ein Ticket für den Ticket Server tgs
AS antwortet:
E(K_A, <$S_{A,TGS}$, tgs, T_{Exp}, i_a, E(K_{TGS}, <$S_{A,TGS}$, a, T_{Exp}>)>)

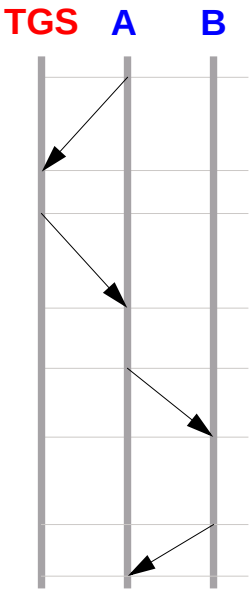
A entschlüsselt mit K_A die Nachricht

- das Ticket besteht aus <$S_{A,TGS}$, a, T_{Exp}>
- es enthält einen Session key für die Kommunikation mit einem Ticket granting server, der dann die Verbindung zu einem Netzwerkdienst bereitstellen kann

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos (3)

- Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



A schickt an den Ticket granting server:
$E(S_{A,TGS}, T_s), E(K_{TGS}, <S_{A,TGS}, a, T_{Exp}>), b, T_{Exp}, i_a'$

TGS erzeugt einen Session key $S_{A,B}$

TGS antwortet:
$E(S_{A,TGS}, <S_{A,B}, b, T_{Exp}, i_a'>), E(K_B, <S_{A,B}, a, T_{Exp}>)$

A entschlüsselt mit $S_{A,TGS}$ die Nachricht

A sendet an B: $E(S_{A,B}, <T_s, C, S_{sub}>), E(K_B, <S_{A,B}, a, T_{Exp}>)$

B entschlüsselt mit K_B den zweiten Teil der Nachricht (Ticket) und mit $S_{A,B}$ den ersten Teil

B antwortet (optional): $E(S_{A,B}, T_s)$

- C = Checksumme zur Überprüfung der richtigen Entschlüsselung
- A kann mehrere Verbindungen mit seinem Ticket öffnen

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

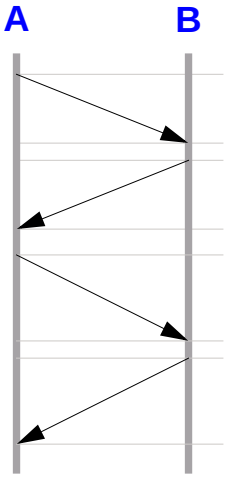
4 Beispiel: Kerberos (4)

- Unterscheidung zwischen Benutzer *(User)* und Benutzerprogramm *(Client)*
  - Wie kann ein Benutzerprogramm seinen Benutzer identifizieren?
  - Geheimer Schlüssel vom Benutzer (K_X) hängt von einem Passwort ab
  - mittels einer Einwegfunktion wird aus dem Passwort der Schlüssel K_X erzeugt
  - Benutzerprogramm braucht also das Passwort zur Verbindungsaufnahme
- Beispiel: *kinit, klogin*
  - Anmeldung beim Authentisierungsdienst mit *kinit* und Passworteingabe
  - Ticket wird im Benutzerkatalog gespeichert
  - *klogin* erlaubt das Einloggen auf einem entfernten Rechner mit Datenverschlüsselung und ohne Passwort

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel

- A und B tauschen ihre öffentlichen Schlüssel aus



A schickt an B seinen öffentlichen Schlüssel: K_A

B schickt an A seinen öffentlichen Schlüssel: K_B

A verschickt signierte und verschlüsselte Nachricht:
$E(K_B, E(K_A^{-1}, T_1))$

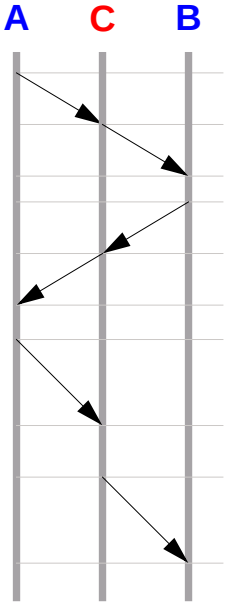
B verschickt signierte und verschlüsselte Nachricht:
$E(K_A, E(K_B^{-1}, T_2))$

- Problem
  - A und B können nicht sicher sein, dass der öffentliche Schlüssel wirklich vom jeweils anderen stammt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel (2)

- Aktiver Mithörer C fängt Datenverbindungen ab *(Man in the middle attack)*



A versucht an B seinen öffentlichen Schlüssel zu schicken: K_A

C fängt Nachricht ab und schickt an B seinen eigenen öffentlichen Schlüssel: K_C

B versucht an A seinen öffentlichen Schlüssel zu schicken: K_B

C fängt Nachricht ab und schickt an A seinen eigenen öffentlichen Schlüssel: K_C

A versucht an A eine Nachricht zu schicken:
$C_1 = E(K_C, E(K_A^{-1}, T_1))$

C dekodiert die Nachricht: $T_1 = D(K_C^{-1}, D(K_A, C_1))$

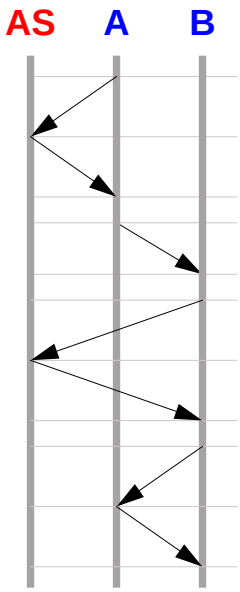
C schickt die Nachricht weiter: $E(K_B, E(K_C^{-1}, T_1))$

B glaubt eine Nachricht von B zu empfangen

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel (3)

- Einsatz eines Authentisierungsdienstes



A kontaktet AS und möchte Verbindung mit B: a, b

AS verschickt den Schlüssel von B: $E(K_{AS}^{-1}, <K_B, b>)$

A schickt initiale Nachricht an B: $E(K_B, <a, i_a>)$

B kontaktet AS und möchte den Schlüssel von A: b, a

AS verschickt den Schlüssel von A: $E(K_{AS}^{-1}, <K_A, a>)$

B antwortet an A: $E(K_A, <i_a, i_b>)$

A bestätigt: $E(K_B, <i_b>)$

- Replay-Probleme
  - Hinzunahme von Zeitstempel und Lebendauer

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

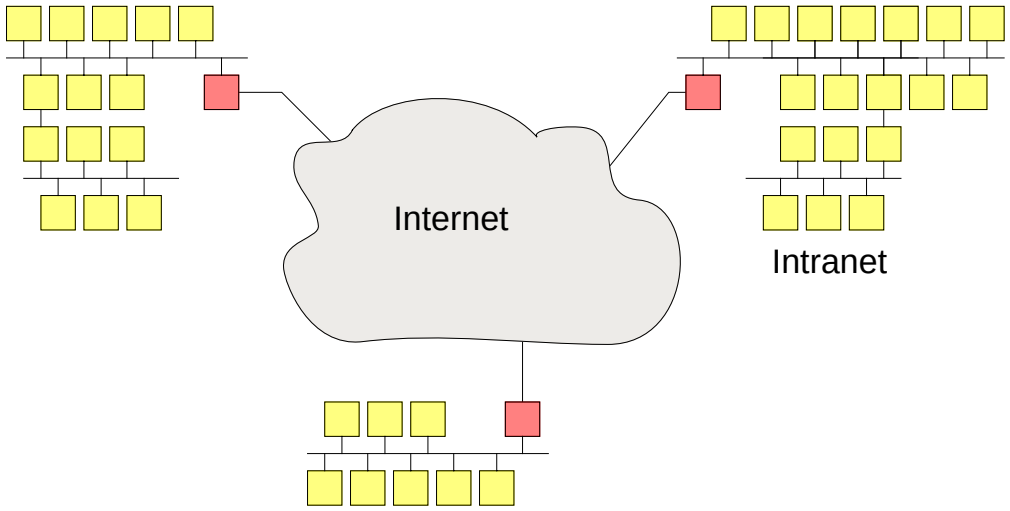
I.8 Firewall

- Trennung von vertrauenswürdigen und nicht vertrauenswürdigen Netzwerksegmenten durch spezielle Hardware *(Firewall)*
  - Beispiel: Trennen des firmeninternen Netzwerks (Intranet) vom allgemeinen Internet
- Funktionalität
  - Einschränkung von Diensten
    - von innen nach außen, z.B. nur Standarddienste
    - von außen nach innen, z.B. kein Telnet, nur WWW
  - Paketfilter
    - Filtern „defekter“ Pakete, z.B. SYN-Pakete
  - Inhaltsfilter
    - Filtern von Pornomaterial aus dem WWW oder News
  - Authentisieren von Benutzern vor der Nutzung von Diensten

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.8 Firewall (2)

- Virtual private network
  - ◆ Verbinden von Intranet-Inseln durch spezielle Tunnels zwischen Firewalls
  - ◆ getunnelter Datenverkehr wird verschlüsselt
  - ◆ Benutzer sieht ein „großes Intranet“ (nur virtuell vorhanden)



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.9 Richtlinien für den Benutzer

1 Passwörter

- Wahl eines Passworts
  - ◆ hinreichend komplexe Passwörter wählen
  - ◆ Schutz vor Wörterbuchangriffen
  - ◆ verschiedene Passwörter für verschiedene Aufgaben (z.B. PPP Passwort ungleich Benutzerpasswort)
- Aufbewahrung
  - ◆ möglichst nirgends aufschreiben
  - ◆ nicht weitergeben
  - ◆ kein Abspeichern auf einem Windows-Rechner (Option immer wegklicken)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Passwörter

- Eingabe
  - ◆ niemals über eine unsichere Rechnerverbindung eingeben
    - **`ftp, telnet, rlogin`** Dienste vermeiden
    - nur sichere Dienste verwenden: **`ssh, slogin`**
    - Datenweg beachten, über den das Passwort läuft: ein unsicheres Netzwerk ist bereits genug
- Änderung
  - ◆ Passwörter regelmäßig wechseln
  - ◆ alte Passwörter nicht wiederverwenden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Schlüsselhandhabung

- Einsatz von PGP oder S/MIME
  - ◆ Zugang zu den privaten Schlüsseln für andere verhindern
  - ◆ Dateirechte auf der Schlüsseldatei prüfen
  - ◆ privater Schlüssel nur auf Diskette
  - ◆ Passphrase wie ein Passwort behandeln
  - ◆ privaten Schlüssel nie über unsichere Netze transportieren

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 E-Mail

- Authentisierung
  - Bei elektronischer Post ist der Absender nicht authentisierbar
  - Digitale Unterschriften einsetzen (z.B. mit PGP oder S/MIME)
- Abhören
  - Elektronische Post durchläuft viele Zwischenstationen und kann dort jeweils gelesen und verfälscht werden
  - Verschlüsselung einsetzen (z.B. mit PGP oder S/MIME)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Programmierung

- S-Bit Programme vermeiden
  - Oft kann das S-Bit durch geschickte Vergabe von Benutzergruppen an Dateien vermieden werden
- Verwendung zusätzlicher Rechte (z.B. durch S-Bit) nur in Abschnitten
  - Trusted Computing Base (TCB)

```
seteuid(getuid());/* am Programmanfang Rechte wegnehmen */
...
seteuid(0);       /* setzt root Rechte */
fd = open("/etc/passwd", O_RDWR);
seteuid(getuid());/* nimmt root Rechte wieder weg */
...
```

- Sorgfältige Programmierung
  - Funktionen wie **`strcpy, strcat, gets, sprintf, scanf, sscanf, system, popen`** vermeiden oder durch **`strncpy, fgets, snprintf`** ersetzen

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 World Wide Web

- Cookies
  - Akzeptieren von Cookies erlaubt einer Website die angesprochenen Seiten genau einem Benutzer zuzuordnen
  - funktioniert über Sessions hinweg
- JavaScript
  - schwere Sicherheitslücken erlauben es, alle für den Benutzer lesbare Dateien an einen Dritten weiterzugeben
  - Ausschalten!
- Abhören
  - WWW-Verbindungen können abgehört werden
  - keine privaten Daten, wie z.B. Kreditkartennummern übertragen
  - Secure-HTTP mit SSL-Verschlüsselung benutzen; https-URLs

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.