

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 Subjekten
 - ◆ jedes Subjekt hält eine Liste von Objekten und den Berechtigungen, die das Subjekt für das Objekt hat
 - ◆ ähnlich Capabilities

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.24

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.26

Reproduktion jeder Art oder Vervielfältigung 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: $(g,s) \leq (g',s') \Rightarrow g \leq g' \wedge s \subseteq s'$

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.25

Reproduktion jeder Art oder Vervielfältigung 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})

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.27

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

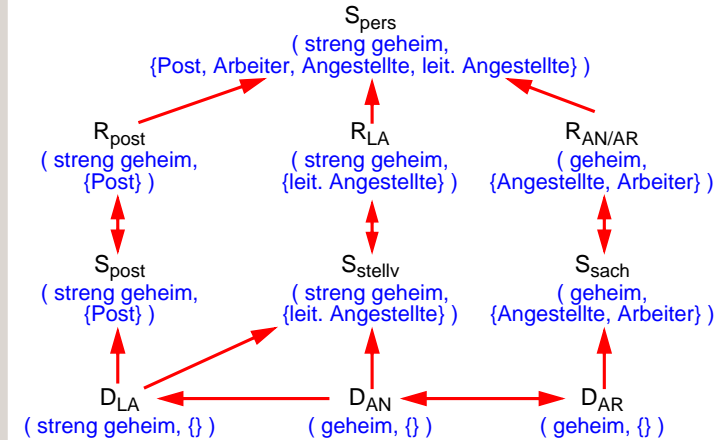
1 Beispiel (2)

■ 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})

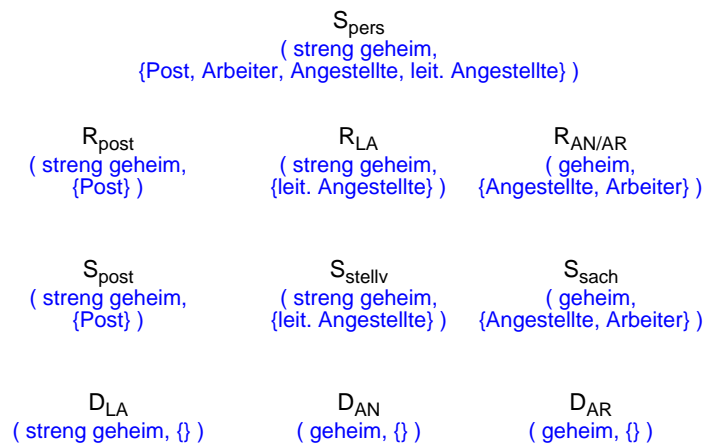
1 Beispiel (4)

■ Informationsflusskontrollgraph:



1 Beispiel (3)

■ Informationsflusskontrollgraph:



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

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

I.4 Schutz durch Speicherverwaltung

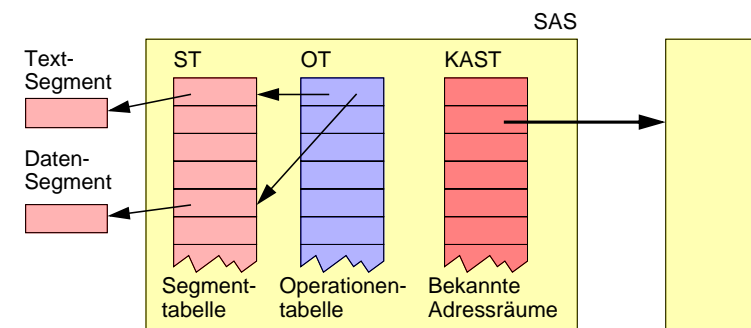
- Schutz vor gegenseitigem Speicherzugriff
 - ◆ Segmentierung und Seitenadressierung erlauben es, jedem Prozess nur den benötigten Speicher einzublenden
 - ◆ Segmentverletzung löst Unterbrechung aus
- Systemaufrufe
 - ◆ definierter Weg von einer Schutzumgebung (der des Prozesses) in eine andere (der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

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 zugriffen werden dürfen
 - ◆ enthält Liste von angebotenen Operationen mit den Angaben, welche Segmente jede Operation benötigt (u.a. Segment für die auszuführenden Instruktionen)

1 Modulkonzept von Habermann (2)

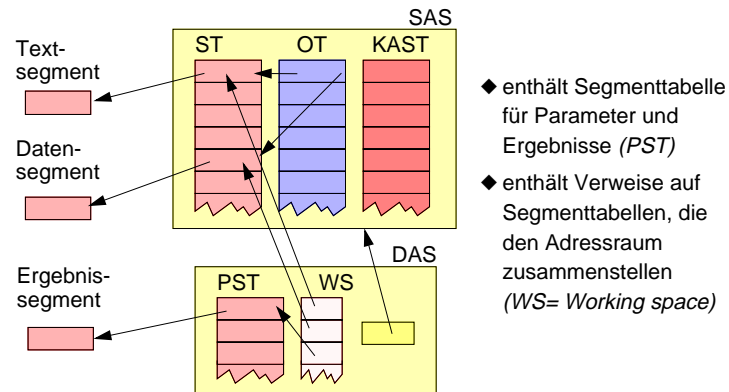
- ◆ enthält Liste von bekannten Adressräumen anderer Module (dort können dann Operationen aufgerufen werden)



KAST = Known address space table

1 Modulkonzept nach Habermann (3)

- Aktivitätsträger sind einem dynamischen Adressraum zugeordnet (DAS, Dynamic address space)



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

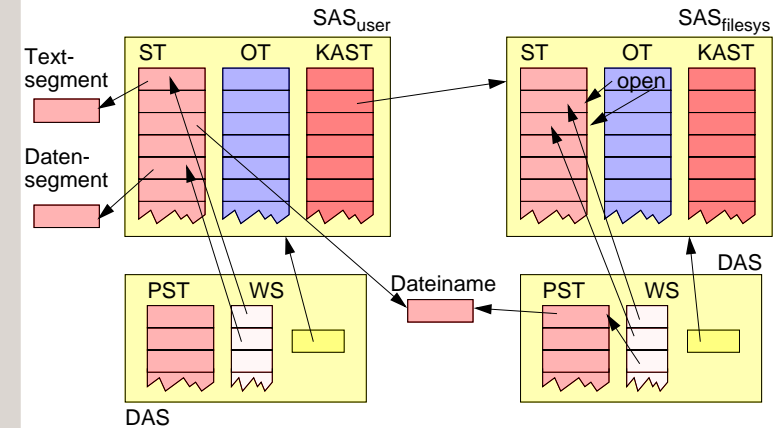
I-Security.fm 2000-02-09 09.49

I.36

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

2 Beispielaufwurf (2)

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



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

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

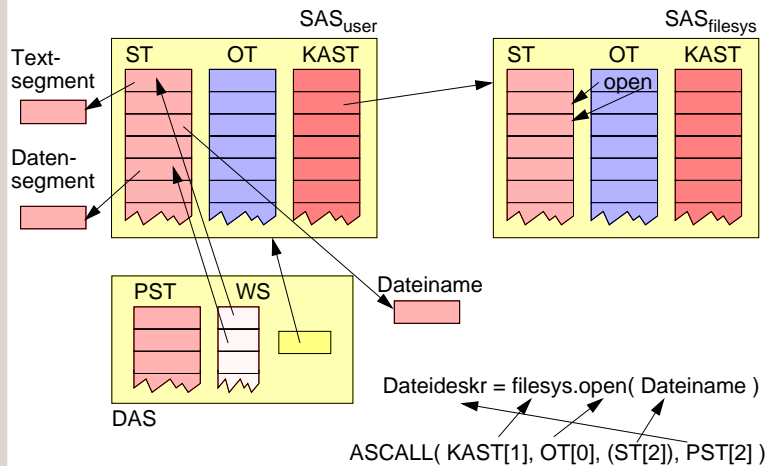
I-Security.fm 2000-02-09 09.49

I.38

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

2 Beispielaufwurf

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



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

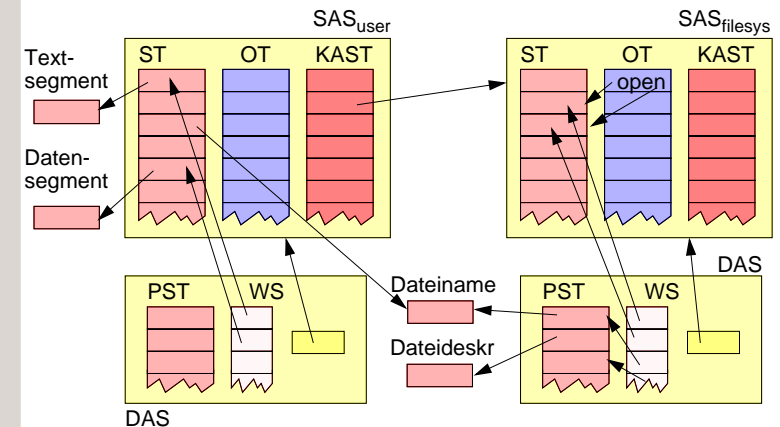
I-Security.fm 2000-02-09 09.49

I.37

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

2 Beispielaufwurf (3)

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



- ◆ „open“ erzeugt neues Segment für den Dateideskriptor

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

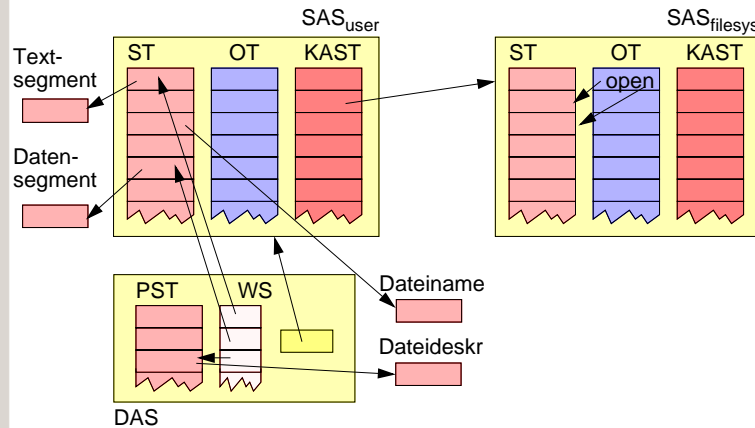
I-Security.fm 2000-02-09 09.49

I.39

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

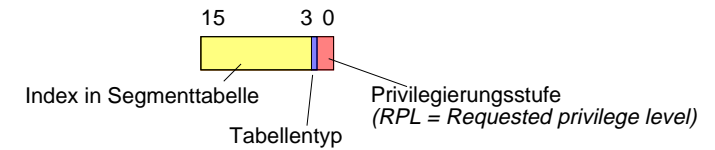
I-Security.fm 2000-02-09 09.49

I.40

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

3 Beispiel: Pentium (2)

- Segmentselektoren enthalten Privilegierungsstufe



- Codesegmentsелектор (CS) bestimmt aktuelle Privilegierungsstufe

- ◆ CPL = Current privilege level

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.42

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

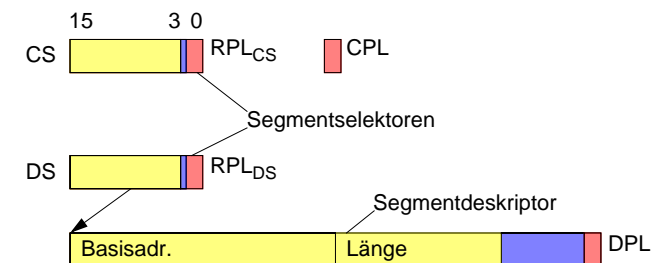
I-Security.fm 2000-02-09 09.49

I.41

Reproduktion jeder Art oder Vervielfältigung 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: $DPL \geq \max(CPL, RPL_{DS})$
- ◆ ansonsten wird Unterbrechung ausgelöst (Schutzverletzung)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.43

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

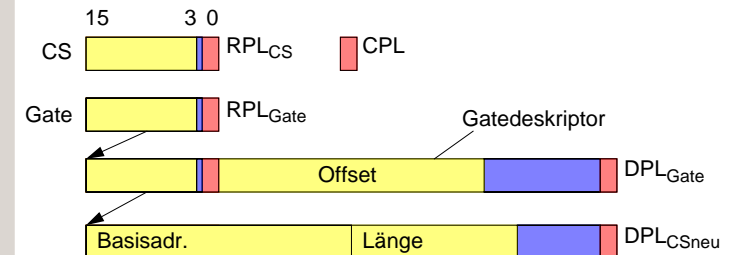
3 Beispiel: Pentium (4)

■ Erläuterung:

- ◆ **DPL < CPL:** *Schutzverletzung*
 augenblickliche Privilegiierungsstufe hat weniger Privilegien als im Deskriptor verlangt
 (CPL hat höhere Stufe als der Deskriptor)
- ◆ **DPL ≥ CPL:** augenblickliche Privilegiierungsstufe 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 Privilegiierungsstufe wie der Deskriptor
- ◆ Selektor darf keine niedrigeren Privilegien versprechen als wirklich verlangt.

3 Beispiel: Pentium (6)

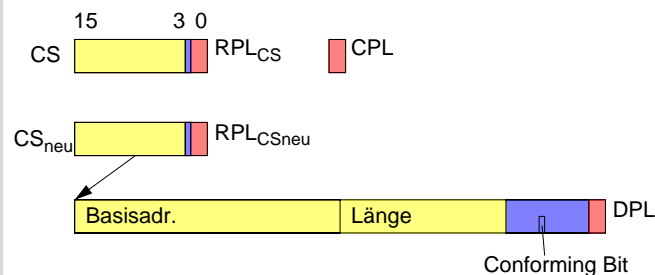
■ Kontrolltransfer mit einem Gate



- ◆ Gatedeskriptoren stehen wie Segmentdeskriptoren in der Segmenttabelle
- ◆ Gatedeskriptor enthält Segmentsелеktor 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 $DPL_{CS_{neu}} \leq CPL$

3 Beispiel: Pentium (5)

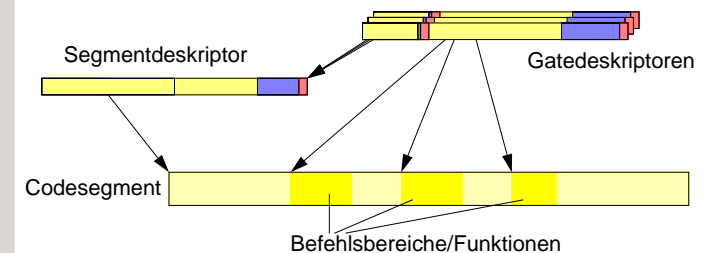
■ Sprünge in andere Code Segmente (Far Call)



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

3 Beispiel: Pentium (7)

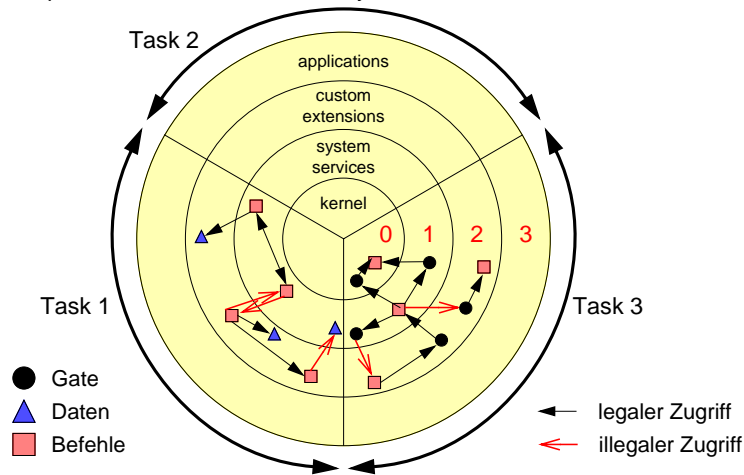
■ Gates erlauben den kontrollierten Sprung in ein privilegierten Befehlsbereich



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

3 Beispiel: Pentium (8)

■ Beispiel eines realisierten Schutzsystems



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.48

Reproduktion jeder Art oder Vervielfältigung 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

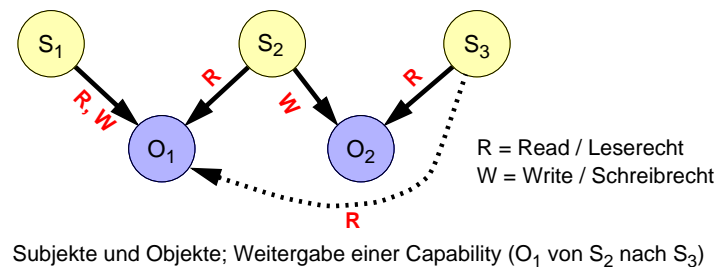
I.50

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.49

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

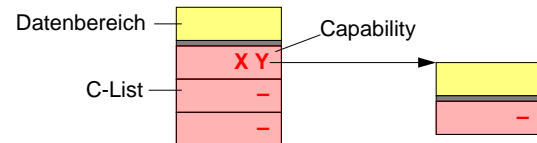
I-Security.fm 2000-02-09 09.49

I.51

Reproduktion jeder Art oder Vervielfältigung 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)



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.52

Reproduktion jeder Art oder Vervielfältigung 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:
 - ◆ **GETRTS**: erlaubt den Aufruf von *Getdata*
 - ◆ **PUTRTS**: erlaubt den Aufruf von *Putdata*
 - ◆ **ADDRTS**: erlaubt den Aufruf von *Adddata*
- Rechte müssen in der Capability zum Objekt gesetzt sein

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.54

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

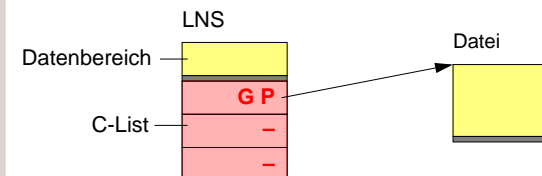
I-Security.fm 2000-02-09 09.49

I.53

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.55

Reproduktion jeder Art oder Vervielfältigung 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:
 - ◆ **LOADRTS**: erlaubt Aufruf von *Load*
 - ◆ **STORTS**: erlaubt Aufruf von *Store*
 - ◆ **APPRTS**: erlaubt Aufruf von *Append*
 - ◆ **KILLRTS**: erlaubt Aufruf von *Delete*

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

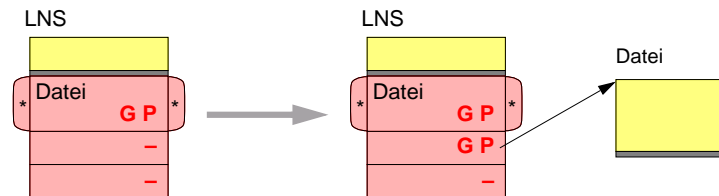
I-Security.fm 2000-02-09 09.49

I.56

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

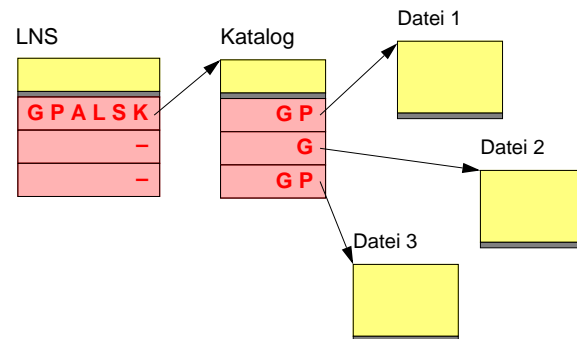
I-Security.fm 2000-02-09 09.49

I.58

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

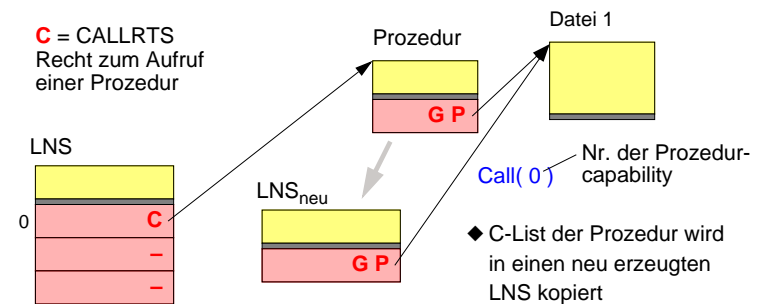
I.57

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

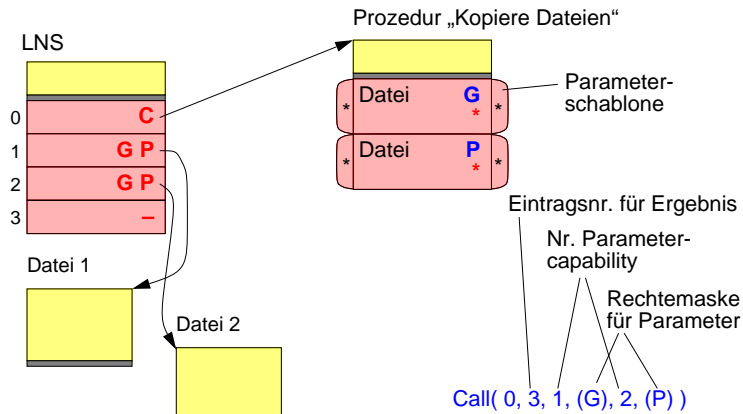
I.59

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

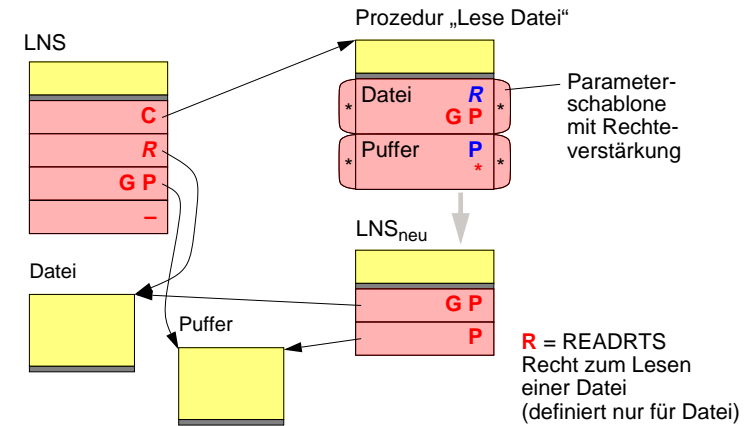
I.60

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

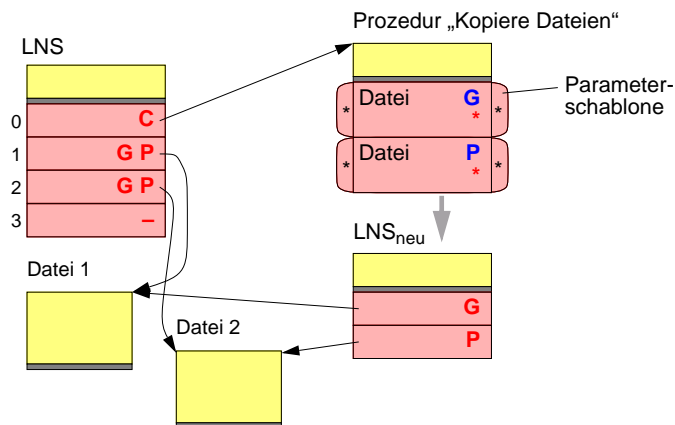
I.62

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.61

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.63

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.64

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

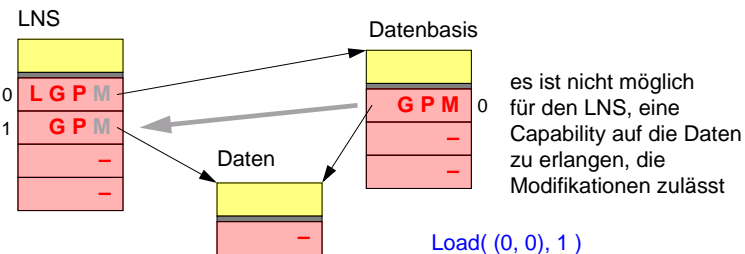
I.66

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

7 Problem: Modifikation (2)

★ Einführung des Modifikationsrechts **MDFYRTS**

- ◆ 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.65

Reproduktion jeder Art oder Vervielfältigung 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 **ENVRTS**

- ◆ 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

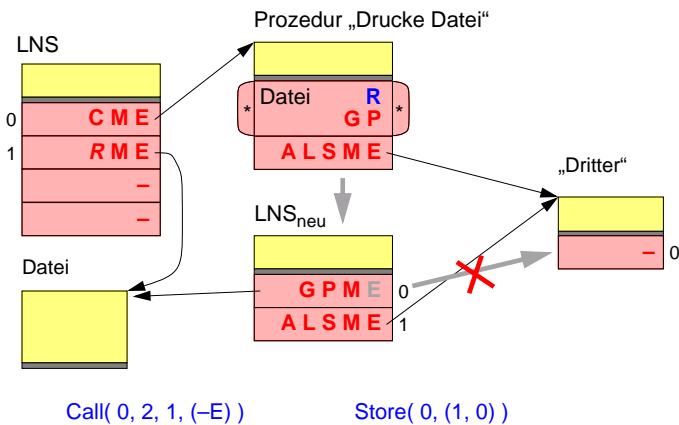
I-Security.fm 2000-02-09 09.49

I.67

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.68

Reproduktion jeder Art oder Vervielfältigung 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.70

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

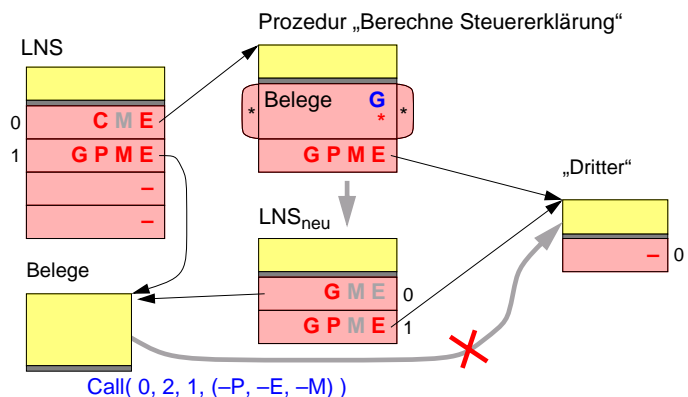
I-Security.fm 2000-02-09 09.49

I.69

Reproduktion jeder Art oder Vervielfältigung 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.71

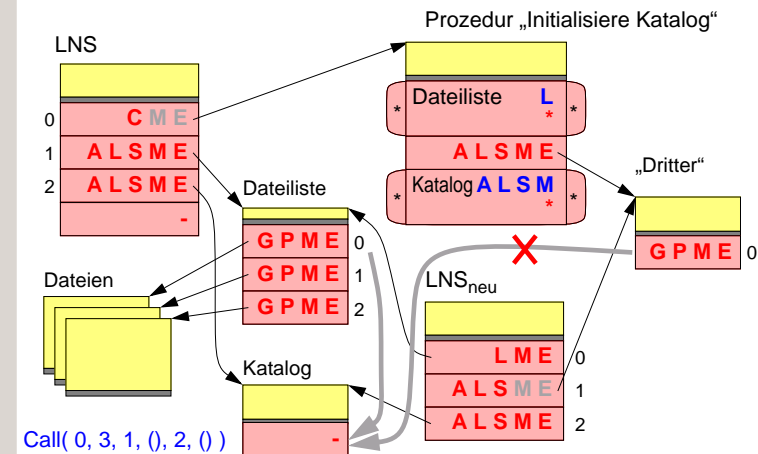
Reproduktion jeder Art oder Vervielfältigung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

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

11 Problem: Initialisierung (3)

- Beispiel: Initialisierung eines Katalogs



11 Problem Initialisierung (2)

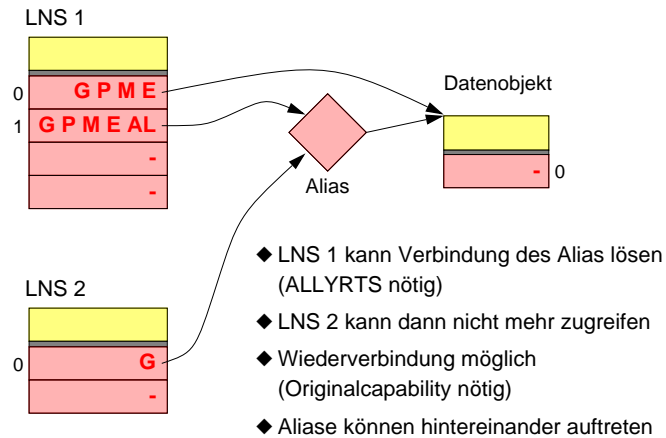
- ★ Environment-Recht auf der Prozedur-Capability
 - ◆ wenn kein Environment-Recht vorhanden ist, werden bei allen in den LNS übernommenen Capabilities die Environment-Rechte ausgeschaltet (gilt jedoch nicht für Parameter)
 - ◆ durch das fehlende Environment-Recht können alle bereits vorhandenen Capabilities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

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 **ALLYRTS** (engl. *ally* = verbünden)

12 Rückruf von Capabilities (2)

■ Beispiel: Weitergabe einer rückrufbaren Capability



12 Rückruf von Capabilities (4)

▲ Problem: Rückruf während der Bearbeitung eines Objekts

- ◆ inkonsistente Zustände möglich

★ Lösung in Hydra

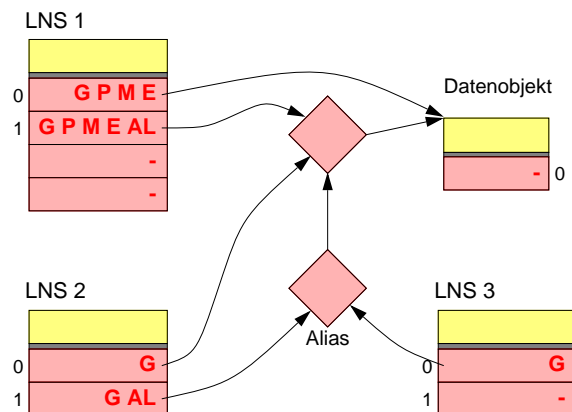
- ◆ Parameter-Capabilities, die durch eine rechteverstärkende Parameterschablone angenommen werden, zeigen auf das Originalobjekt

▲ Nachteil

- ◆ nicht vertrauenswürdige Prozeduren können rückruffreie Capability erlangen
- ◆ Problem fällt in die selbe Kategorie wie rechteverstärkende Parameterschablonen an sich

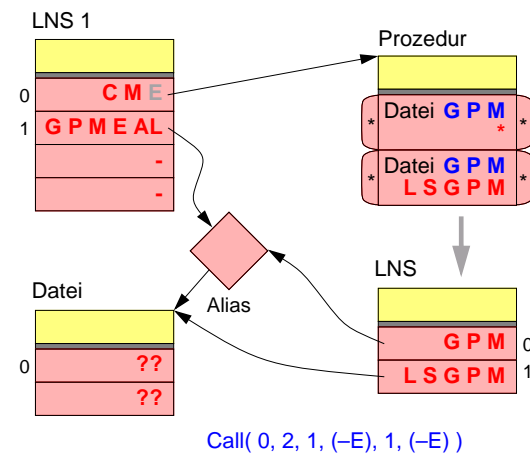
12 Rückruf von Capabilities (3)

■ Beispiel: Aliasketten



12 Rückruf von Capabilities (5)

■ Beispiel:



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 FRZRTS*) ein
 - ◆ Einfrieren nimmt Modifikationsrecht weg
 - ◆ ein Objekt kann nur gefroren werden, wenn alle Capabilities in der C-List bereits das Einfrierrecht haben

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.80

Reproduktion jeder Art oder Vervielfältigung 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 Reihe 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

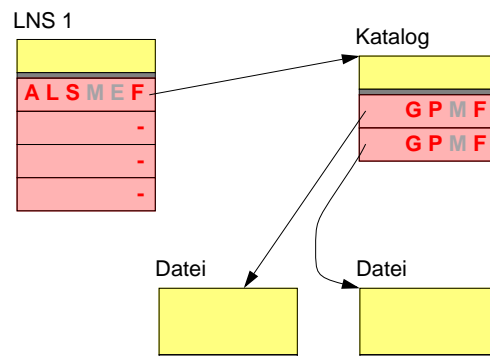
I-Security.fm 2000-02-09 09.49

I.82

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

13 Garantierter Zugriff (2)

- Beispiel:



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.81

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.83

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

I.6 Kryptographische Maßnahmen (2)

- Funktionen
 - ◆ Verschlüsselungsfunktion E (encrypt): $E(K_1, T) \rightarrow C$
 - ◆ Entschlüsselungsfunktion D (decrypt): $D(K_2, C) \rightarrow T$
 - ◆ 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: $K_1 = K_2$

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.84

Reproduktion jeder Art oder Vervielfältigung 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.

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.85

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.86

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

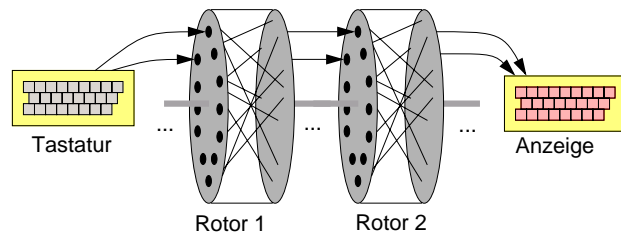
I-Security.fm 2000-02-09 09.49

I.87

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

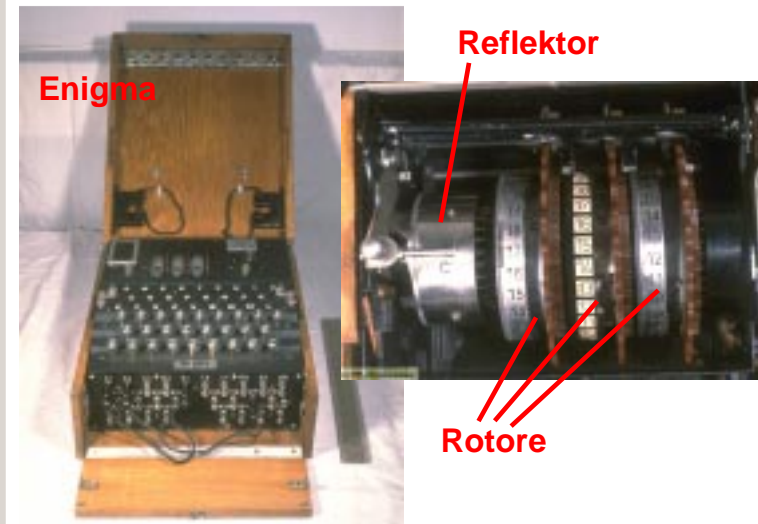
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.92

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

4 Rotormaschinen



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.94

Reproduktion jeder Art oder Vervielfältigung 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/>

SP I

Systemprogrammierung I

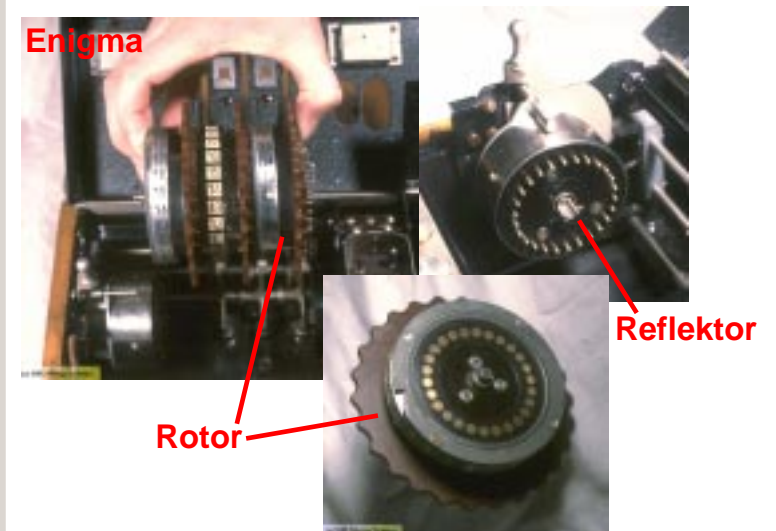
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.93

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

4 Rotormaschinen (2)



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.95

Reproduktion jeder Art oder Vervielfältigung 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

5 Heutige symmetrische Verfahren (3)

■ Triple DES

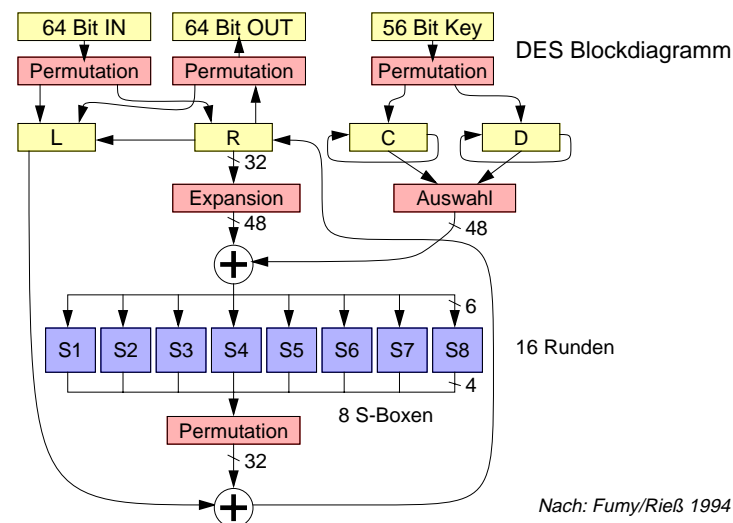
- ◆ dreifache Verschlüsselung mit DES
- ◆ Nutzung von drei oder mindestens zwei verschiedenen Schlüsseln

■ 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*)

5 Heutige symmetrische Verfahren (2)



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

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.100

Reproduktion jeder Art oder Vervielfältigung 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.102

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.101

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

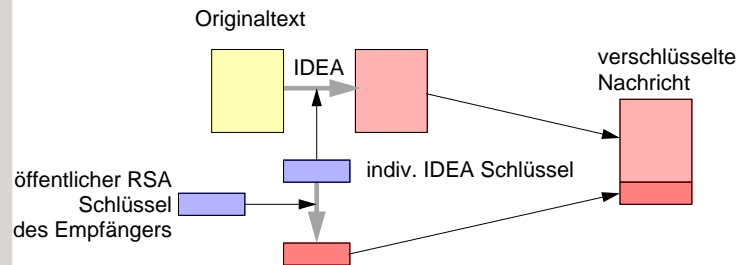
I-Security.fm 2000-02-09 09.49

I.103

Reproduktion jeder Art oder Vervielfältigung 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.104

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.106

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.105

Reproduktion jeder Art oder Vervielfältigung 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 erreichen (z.B. PGP: Webserver als Schlüsselservers)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.107

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.108

Reproduktion jeder Art oder Vervielfältigung 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*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.110

Reproduktion jeder Art oder Vervielfältigung 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 Benutzererkennung (z.B. UNIX UID) als Ausweis
 - ◆ Im Netzwerk können Pakete abgefangen, verfälscht und gefälscht werden (einfache Übertragung einer Benutzererkennung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

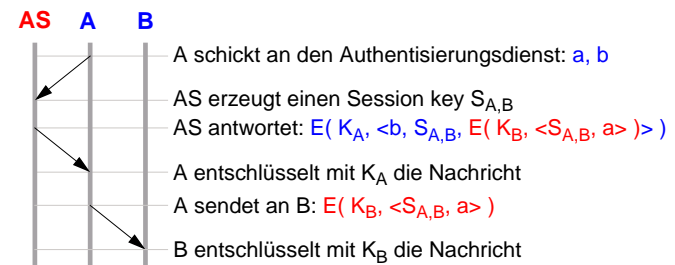
I-Security.fm 2000-02-09 09.49

I.109

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.111

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.112

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

2 Authentisierungsdienst mit Bestätigung (2)

▲ Problem

- ◆ Aufzeichnen von $E(K_B, \langle S_{A,B}, a \rangle)$ 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*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

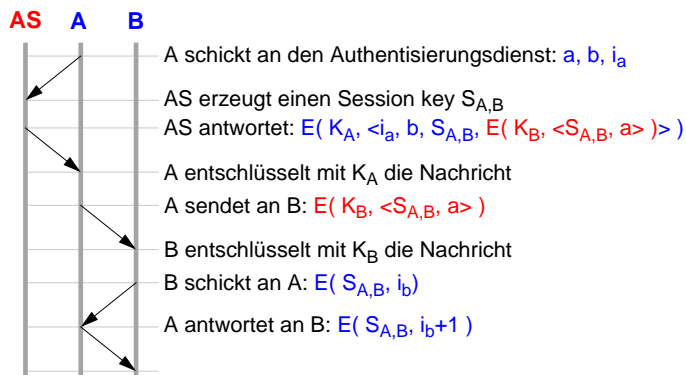
I-Security.fm 2000-02-09 09.49

I.114

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

2 Authentisierungsdienst mit Bestätigung

- Bestätigung enthält Einmalinformation (*Nonce*)



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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

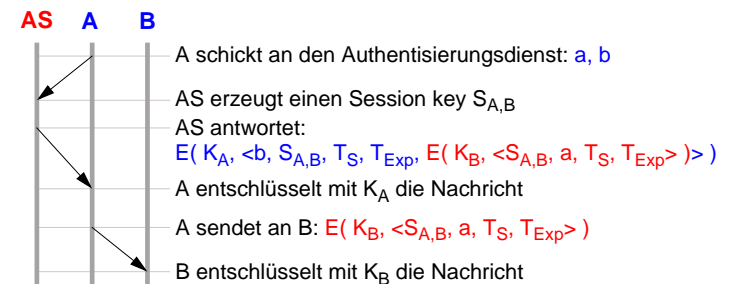
I-Security.fm 2000-02-09 09.49

I.113

Reproduktion jeder Art oder Vervielfältigung 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 noch einmal zum Aufbau einer Verbindung verwendet werden

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.115

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

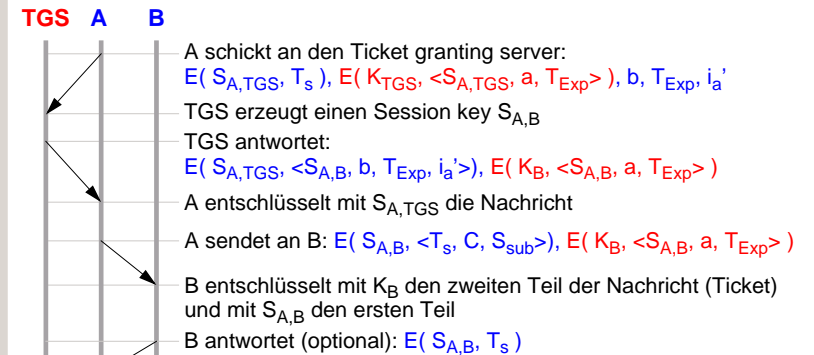
I-Security.fm 2000-02-09 09.49

I.116

Reproduktion jeder Art oder Vervielfältigung 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



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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

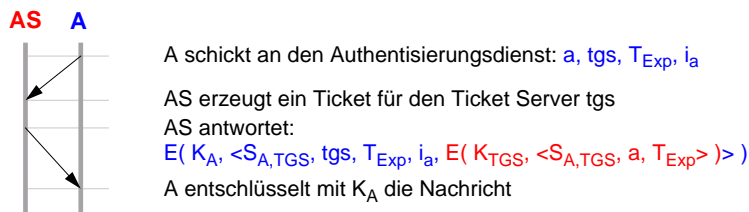
I-Security.fm 2000-02-09 09.49

I.118

Reproduktion jeder Art oder Vervielfältigung 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



- ◆ 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.117

Reproduktion jeder Art oder Vervielfältigung 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 Passwordeingabe
 - ◆ Ticket wird im Benutzerkatalog gespeichert
 - ◆ *klogin* erlaubt das Einloggen auf einem entfernten Rechner mit Datenverschlüsselung und ohne Passwort

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

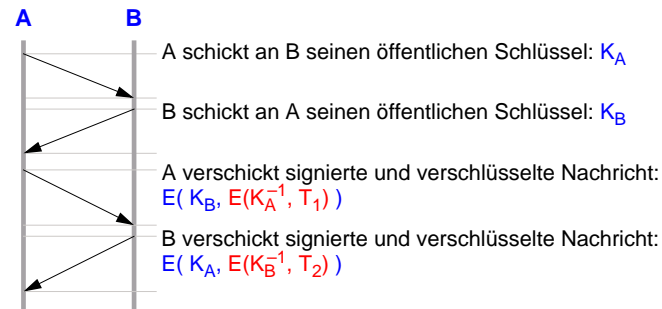
I-Security.fm 2000-02-09 09.49

I.119

Reproduktion jeder Art oder Vervielfältigung 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



▲ Problem

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

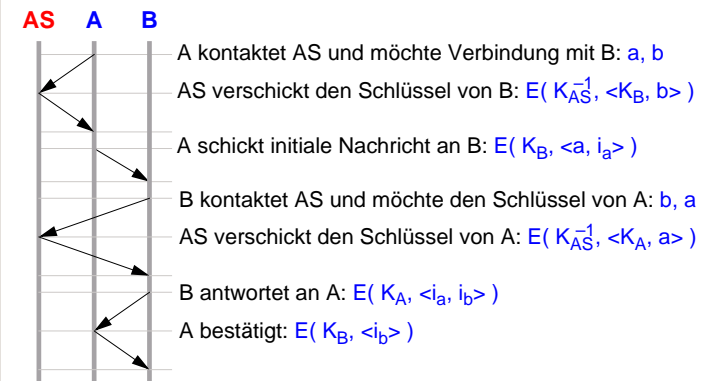
I-Security.fm 2000-02-09 09.49

I.120

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

5 Austausch öffentlicher Schlüssel (3)

- Einsatz eines Authentisierungsdienstes



▲ Replay-Probleme

- ◆ Hinzunahme von Zeitstempel und Lebendauer

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

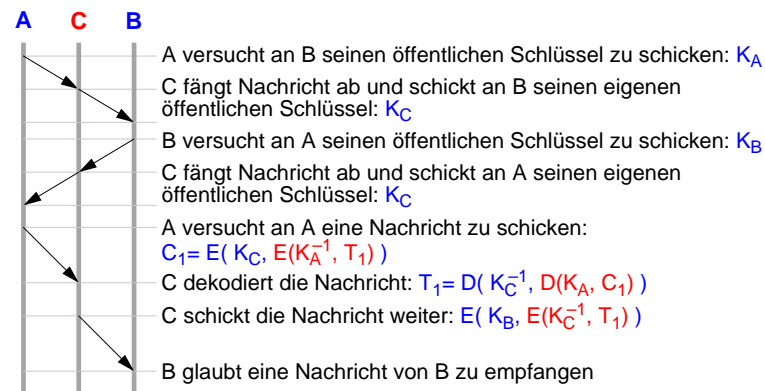
I-Security.fm 2000-02-09 09.49

I.122

Reproduktion jeder Art oder Vervielfältigung 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*)



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.121

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

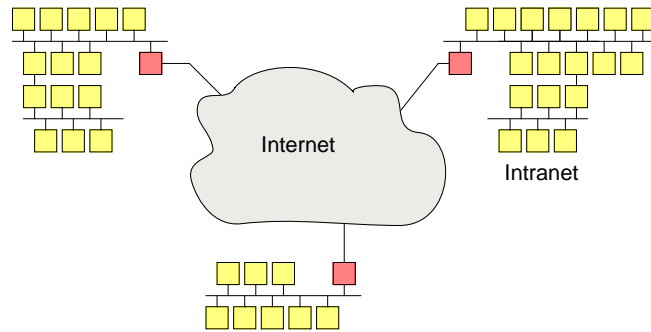
I-Security.fm 2000-02-09 09.49

I.123

Reproduktion jeder Art oder Vervielfältigung 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)



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.124

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.126

Reproduktion jeder Art oder Vervielfältigung 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.125

Reproduktion jeder Art oder Vervielfältigung 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

I-Security.fm 2000-02-09 09.49

I.127

Reproduktion jeder Art oder Vervielfältigung 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)

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

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