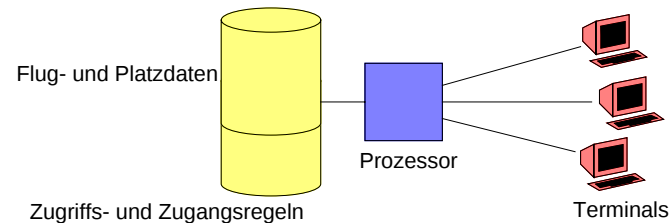


I Datensicherheit und Zugriffsschutz

I.1 Problemstellung

- Beispiel: Zugang zu einer Datenbank zur Flugreservierung und -buchung



- ◆ Was sind mögliche Beeinträchtigungen der Datensicherheit?

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.1

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

I.1 Problemstellung (3)

- Illegaler Datenzugriff
 - ◆ Daten sind zugreifbar, die vertraulich behandelt werden sollen
- Illegales Löschen von Daten
 - ◆ Kein Zugriff, aber Daten werden gelöscht
- Illegales Manipulieren von Daten
 - ◆ Daten werden in böswilliger Absicht verändert
- Zerstörung von Rechensystemen
 - ◆ physisches Zerstören von Teilen der Rechananlage

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

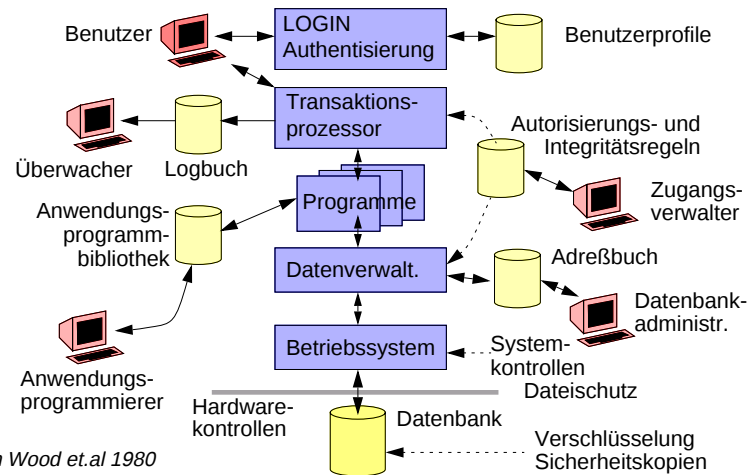
I-Security.fm 1999-02-02 09.53

I.3

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

I.1 Problemstellung (2)

- Überprüfungen beim Transaktionsbetrieb (Datenbankanwendung)



Nach Wood et.al 1980

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.2

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

1 Umgebung der Rechananlage

- ▲ Naturkatastrophen
 - ◆ Erdbeben, Vulkanausbrüche etc. können Rechananlage und Datenbestand zerstören
- ▲ Unfälle
 - ◆ Gasexplosion, Kühlwasserlecks in der Klimaanlage oder Ähnliches zerstören Rechner und Daten
- ▲ Böswillige Angriffe
 - ◆ Zerstörung der Rechananlage und des Datenbestands durch Sabotage (Bombenanschlag, Brandanschlag etc.)
- ▲ Unbefugter Zutritt zu den Räumen des Rechenzentrums
 - ◆ Diebstahl von Datenträgern
 - ◆ Zerstörung von Daten
 - ◆ Zugang zu vertraulichen Daten

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.4

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

2 Systemsoftware

- ▲ Versagen der Schutzmechanismen
 - ◆ System läßt Unbefugte auf Daten zugreifen oder Operationen ausführen
- ▲ Durchsickern von Informationen
 - ◆ Anwender können anhand scheinbar unauffälligen Systemverhaltens Rückschlüsse auf vertrauliche Daten ziehen (*Covert channels*)
- Beispiel: verschlüsselt abgespeicherte Paßwörter sind zugänglich
 - ◆ Entschlüsselungsversuch der Paßwörter außerhalb der Rechenanlage
 - ◆ "Wörterbuchattacke": Raten von Paßwörtern möglich

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.5

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

3 Systemprogrammierer (2)

- ◆ Buffer overflow Fehler bei Funktionen **gets()**, **strcpy()**, **strcat()**, **sprintf("%s",...)**
 - Zu lange Eingaben überschreiben den Stackspeicher des Prozessors
 - Mit genauer Kenntnis ist das Ausführen beliebigen Codes erzwingbar
- ◆ Fehlerhafte Parameterprüfung beim Aufruf von Funktionen **system()**
- ◆ Beispiel: Löschen einer Datei
 - Name der Datei wurde in Variable **file** eingelesen
 - Aufruf von **system** mit Parameter **strcat("rm ", file)**
 - Gibt man für den Dateinamen den String **"fn ; xterm -display myhost:0 &"** ein, bekommt man ein Fenster auf der aufgebrochenen Maschine

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.7

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

3 Systemprogrammierer

- ▲ Umgehen oder Abschalten der Schutzmechanismen
- ▲ Installation eines unsicheren Systems
 - ◆ erlaubt dem Systemprogrammierer die Schutzmechanismen von außen zu umgehen
- ▲ Fehler beim Nutzen von Bibliotheksfunktionen innerhalb sicherheitskritischer Programme
 - ◆ S-Bit Programme unter UNIX laufen mit der Benutzerkennung des Dateibesitzers, nicht unter der des Aufrufers
 - Fehlerhafte S-Bit Programme können zur Ausführung von Code unter einer fremden Benutzerkennung gebracht werden
 - S-Bit Programme mit "root-Rechten" besonders gefährlich

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.6

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

4 Rechnerhardware

- ▲ Versagen der Schutzmechanismen
 - ◆ erlauben nicht-autorisierten Zugriff
- ▲ Fehlerhafte Befehlsausführung
 - ◆ Zerstörung von wichtigen Daten
- ▲ Abstrahlungen
 - ◆ erlaubt Ausspähen von Daten

5 Datenbasis

- ▲ Falsche Zugriffsregeln
 - ◆ erlauben nicht-autorisierten Zugriff

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.8

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

6 Operateur

- ▲ Kopieren vertraulicher Datenträger
- ▲ Diebstahl von Datenträgern
- ▲ Initialisierung mit unsicherem Zustand
 - ◆ Operateur schaltet beispielsweise Zugriffskontrolle ab
- ▲ Nachlässige Rechnerwartung
 - ◆ Nachbesserungen der Systemsoftware (*Patches*) werden nicht eingespielt
 - Sicherheitslücken werden nicht gestopft

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.9

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

7 Sicherheitsbeauftragter

- ▲ Fehlerhafte Spezifikation der Sicherheitspolitik
 - ◆ dadurch Zugang für Unbefugte zu vertraulichen Daten oder
 - ◆ Änderungen von Daten durch Unbefugte möglich
- ▲ Unterlassene Auswertung von Protokolldateien
 - ◆ Einbrüche und mögliche Sicherheitslücken werden nicht rechtzeitig entdeckt

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.10

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

8 Kommunikationssystem

- ▲ Abhören der Kommunikationsleitungen (*Sniffing*)
 - ◆ z.B. Telefonverbindung bei Modemnutzung oder serielle Schnittstellen
 - ◆ z.B. Netzwerkverkehr auf einem Netzwerkstrang
- ◆ Ermitteln von Paßwörtern und Benutzerkennungen
 - manche Dienste übertragen Paßwörter im Klartext (z.B. ftp, telnet, rlogin)
- ◆ Zugriff auf vertrauliche Daten
- ◆ unbefugte Datenveränderungen
 - Verfälschen von Daten
 - Übernehmen von bestehenden Verbindungen (*Hijacking*)
 - Vorspiegeln falscher Netzwerkadressen (*Spoofing*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.11

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

8 Kommunikationssystem (2)

- ▲ Illegale Nutzung von Diensten über das Netzwerk
 - ◆ Standardsysteme bieten eine Menge von Diensten an (z.B. ftp, telnet, rwho u.a.)
- ◆ Sicherheitslücken von Diensten werden publik gemacht und sind auch von "dummen" Hackern nutzbar (*Exploit scripts*)
<http://www.rootshell.org>
- ◆ Auch bei temporär am Netzwerk angeschlossenen Computern eine Gefahr
 - z.B. Linux-Maschine mit PPP-Verbindung an das Uni-Netz
 - Voreinstellungen der Standardinstallation meist unsicher

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.12

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

9 Terminal

- ▲ Ungeschützter Zugang zum Terminal
 - ◆ Nutzen einer fremden Benutzerkennung
 - ◆ Zugriff auf vertrauliche Daten
 - ◆ unbefugte Datenveränderungen

10 Benutzer

- ▲ Nutzen anderer Kennungen
 - ◆ erlauben nicht-autorisierten Zugriff
 - ◆ unbefugte Datenveränderungen
 - ◆ unbefugte Weitergabe von Informationen
- ▲ Einbruch von Innen
 - ◆ Benutzer hat leichter Zugang zu möglichen Sicherheitslöchern (z.B. bei Diensten)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.13

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

11 Anwendungsprogrammierung

- ▲ Nichteinhalten der Spezifikation
 - ◆ Umgehen der Zugriffskontrollen
- ▲ Einfügen von „böartigen“ Befehlsfolgen
 - ◆ *Back door*: Hintertür gibt dem Programmierer im Betrieb Zugang zu vertraulichen Daten oder illegalen Operationen
 - ◆ *Trojan horse*: Unter bestimmten Bedingungen werden illegale Operationen ohne Trigger von außen angestoßen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.14

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

12 "Tracker Queries"

- Beispiel: Datenbanksysteme
 - ◆ Zugriff auf Einzelinformationen ist verboten (Vertraulichkeit)
 - ◆ statistische Informationen sind erlaubt
- ▲ Grenzen möglicher Sicherheitsmaßnahmen:
Zugriff auf Einzelinformationen dennoch möglich
 - ◆ geeignete Anfragen kombinieren (*Tracker queries*)

- Beispiel: Gehaltsdatenbank

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.15

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

12 "Tracker Queries" (2)

- Tabelle der Datenbankeinträge:

Nr.	Name	Geschl.	Fach	Stellung	Gehalt	Spenden
1	Albrecht	m	Inf.	Prof.	60.000	150
2	Bergner	m	Math.	Prof.	45.000	300
3	Cäsar	w	Math.	Prof.	75.000	600
4	David	w	Inf.	Prof.	45.000	150
4	Engel	m	Stat.	Prof.	54.000	0
5	Frech	w	Stat.	Prof.	66.000	450
6	Groß	m	Inf.	Angest.	30.000	60
8	Hausner	m	Math.	Prof.	54.000	1500
9	Ibel	w	Inf.	Stud.	9.000	30
10	Jost	m	Stat.	Angest.	60.000	45
11	Knapp	w	Math.	Prof.	75.000	300
12	Ludwig	m	Inf.	Stud.	9.000	0

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.16

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

12 "Tracker Queries" (3)

■ Anfragen und Antworten:

- ◆ Anzahl('w'): 5
- ◆ Anzahl('w' und (nicht 'Inf' oder nicht 'Prof.')): 4
- ◆ mittlere Spende('w'): 306
- ◆ mittlere Spende('w' und (nicht 'Inf.' oder nicht 'Prof.')): 345

■ Berechnung:

- ◆ Spende('David'): $306 * 5 - 345 * 4 =$
 $1530 - 1380 =$
150

1 Beispiel für Zugriffslisten

■ Personaldatensatz

- ◆ besteht aus: Name, Abteilung, Personalnummer, Lohn- oder Gehaltsgruppe

■ Personaldateien (Objekte)

- ◆ D_{LA} : Personaldaten der leitenden Angestellten
- ◆ D_{AN} : Personaldaten der sonstigen Angestellten
- ◆ D_{AR} : Personaldaten der Arbeiter

■ Prozeduren (gehören zu den Aktionen)

- ◆ R_{LA} : Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{LA}
- ◆ $R_{AN/AR}$: Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{AN} oder D_{AR}
- ◆ R_{post} : Lesen von Name, Abteilung und Pers.-Nr.

I.2 Zugriffslisten

■ Identifikation von Subjekten, Objekten und Berechtigungen

- ◆ Subjekt: Person oder Benutzerkennung im System
(repräsentiert jemanden, der Aktionen ausführen kann)
- ◆ Objekt: Komponente des Systems
(repräsentiert Ziel einer Aktion)
- ◆ Berechtigung: z.B. Leseberechtigung auf einer Datei
(repräsentiert die Erlaubnis für die Ausführung einer Aktion)

■ Erfassung der Berechtigungen in einer Subjekt-Objekt-Matrix: Zugriffsliste (*Access control list, ACL*)

1 Beispiel für Zugriffslisten (2)

■ Benutzer (Subjekte)

- ◆ S_{pers} : Leiter des Personalbüros
 - Besitzer aller Dateien und Prozeduren
 - Lese- und Schreibrecht für alle Dateien
 - Aufrufrecht für alle Prozeduren
- ◆ S_{stellv} : Sachbearb. leitende Angestellte, stellvertr. Leiter Personalbüro
 - Lese- und Schreibrecht für D_{AN} und D_{AR}
 - Aufrufrecht für R_{LA}
- ◆ S_{sach} : Sachbearbeiter Angestellte u. Arbeiter
 - Aufrufrecht für $R_{AN/AR}$
- ◆ S_{post} : Poststelle
 - Aufrufrecht für R_{post} auf alle Dateien

1 Beispiel für Zugriffslisten (3)

- Berechtigungen werden in Matrix ausgedrückt:

	D _{LA}	D _{AN}	D _{AR}	R _{LA}	R _{AN/AR}	R _{post}
S _{pers}	O, R, W	O, R, W	O, R, W	O, I	O, I	O, I
S _{stellv}		R, W	R, W	I		
S _{sach}					I	
S _{post}						I
R _{LA}	R					
R _{AN/AR}		R	R			
R _{post}	R	R	R			

- O = *Owner*; Besitzer der Datei oder Prozedur
- R = *Read*; volle Leseberechtigung
- W = *Write*; volle Schreibberechtigung
- I = *Invoke*; Aufrufberechtigung

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

I.21

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

2 Beispiel: UNIX

- *Superuser*

- ◆ Benutzer *root* hat automatisch alle Zugriffsrechte

- S-Bit-Programme

- ◆ S-Bit ist ein besonderes Recht auf der Binärdatei des Programms
- ◆ Besitzer der Datei wird bei der Ausführung auch Besitzer des Prozeß (sonst wird Aufrufer Besitzer des Prozeß)

- ★ Vorteil

- ◆ Bereitstellen von Prozessen, die kontrolliert Aufrufern höhere Zugriffsberechtigungen erlauben

- ▲ Nachteil

- ◆ Fehler im Prozeß gibt Aufrufer volle Rechte des Programmbesitzers
- ◆ fatal, falls das Programm *root* gehört

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

I.23

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

2 Beispiel: UNIX

- Zugriffslisten für

- ◆ Dateien und Geräte
- ◆ Shared memory-Segmente
- ◆ Message queues
- ◆ Semaphore
- ◆ etc.

- Berechtigungen:

- ◆ Lesen (*read*), Schreiben (*write*), Ausführen (*execute*)
- ◆ für Besitzer, Gruppe und alle anderen unterscheidbar

- Subjekte:

- ◆ Prozesse
- ◆ Besitzer (Benutzer) und Zugehörigkeit zu einer oder mehreren Gruppen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

I.22

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

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

I.24

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.25

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.26

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.27

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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})

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

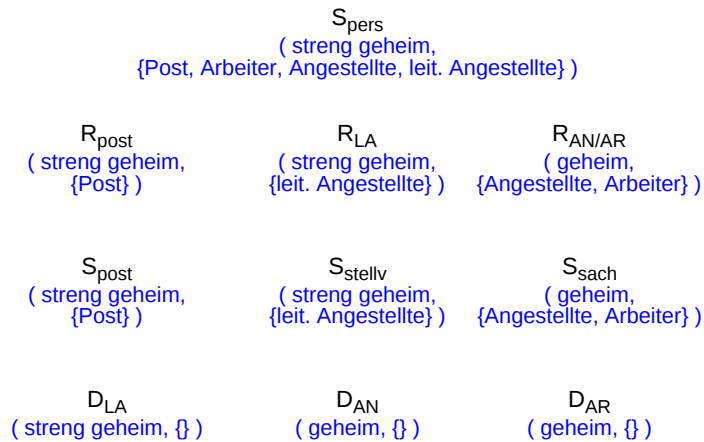
I-Security.fm 1999-02-02 09.53

I.28

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

1 Beispiel (3)

Informationsflußkontrollgraph:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.29

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

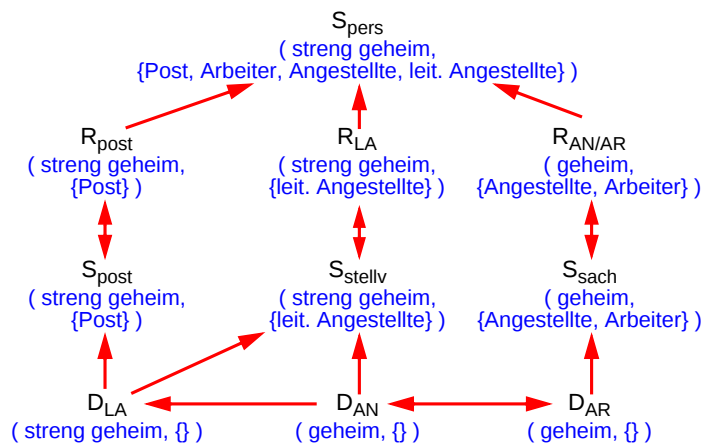
I-Security.fm 1999-02-02 09.53

I.31

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

1 Beispiel (4)

Informationsflußkontrollgraph:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.30

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.32

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

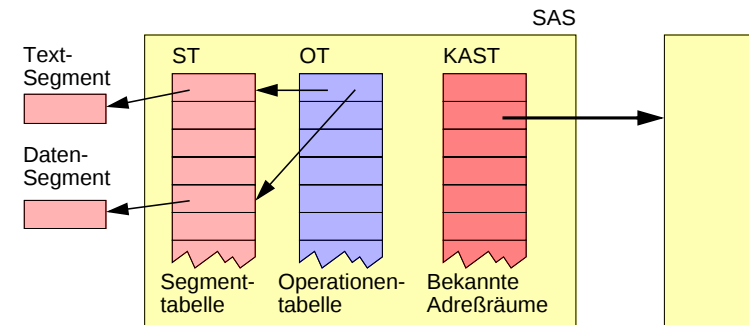
I-Security.fm 1999-02-02 09.53

I.33

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.35

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

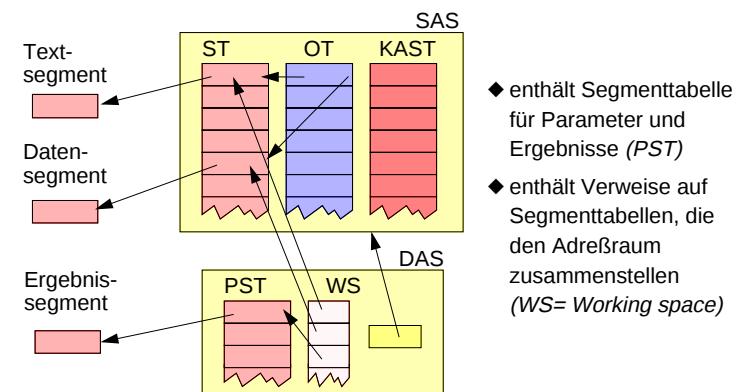
I-Security.fm 1999-02-02 09.53

I.34

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

1 Modulkonzept nach Habermann (3)

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



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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

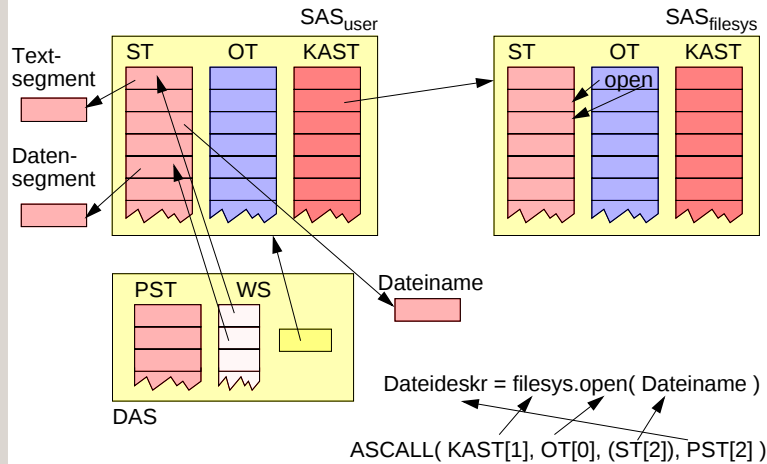
I-Security.fm 1999-02-02 09.53

I.36

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

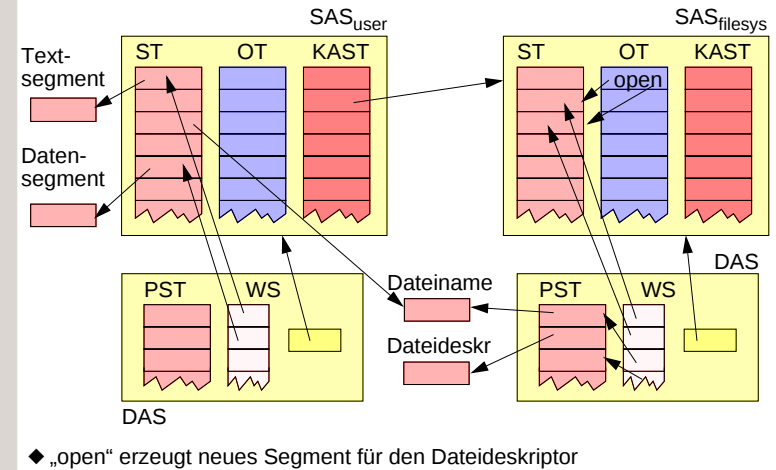
I-Security.fm 1999-02-02 09.53

I.37

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

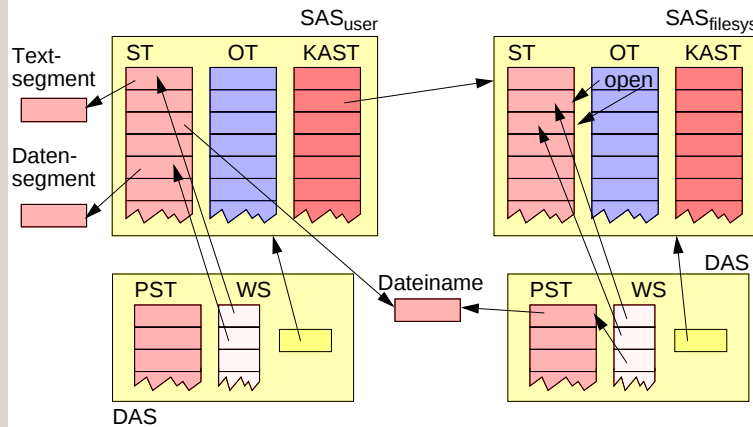
I-Security.fm 1999-02-02 09.53

I.39

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

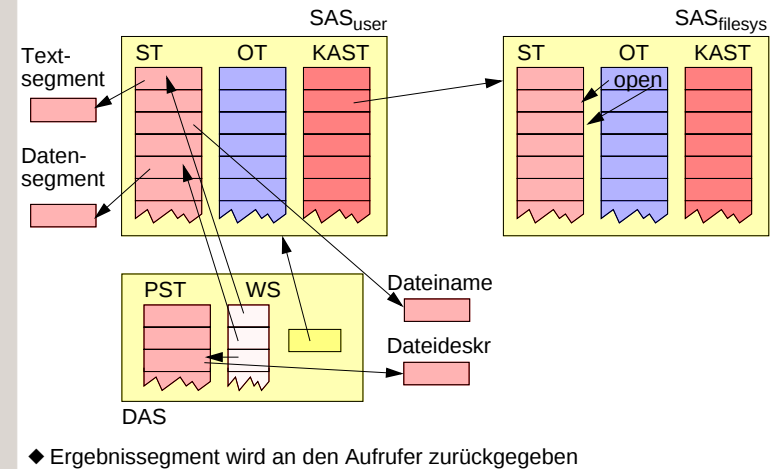
I-Security.fm 1999-02-02 09.53

I.38

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

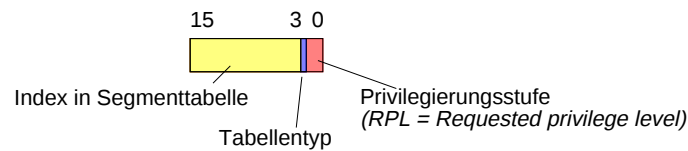
I.40

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

- Segmentselektoren enthalten Privilegierungsstufe



- Codesegmentselektor (CS) bestimmt aktuelle Privilegierungsstufe

- ◆ $CPL = \text{Current privilege level}$

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.41

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

3 Beispiel: Pentium (3)

- Erläuterung:

- ◆ $DPL < RPL(ds)$: Schutzverletzung
Selektor hat geringere Privilegierungsstufe als der Deskriptor
- ◆ $DPL \geq RPL(ds)$: Selektor hat mindestens die gleiche Privilegierungsstufe wieder Deskriptor
- ◆ $DPL < CPL$: Schutzverletzung
augenblickliche Privilegierungsstufe ist niedriger als im Deskriptor verlangt
- ◆ $DPL \geq CPL$: augenblickliche Privilegierungsstufe ist mindestens so hoch wie die im Deskriptor

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

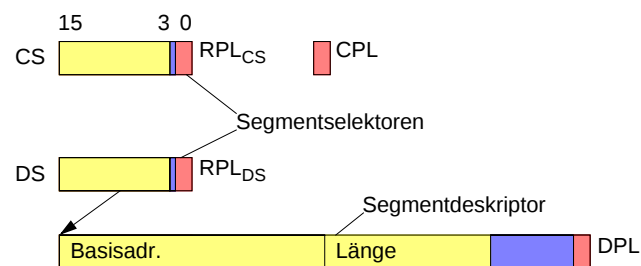
I-Security.fm 1999-02-02 09.53

I.43

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

3 Beispiel: Pentium (2)

- Datenzugriff (z.B. auf Datensegment DS)



- ◆ $DPL = \text{Descriptor privilege level}$
- ◆ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

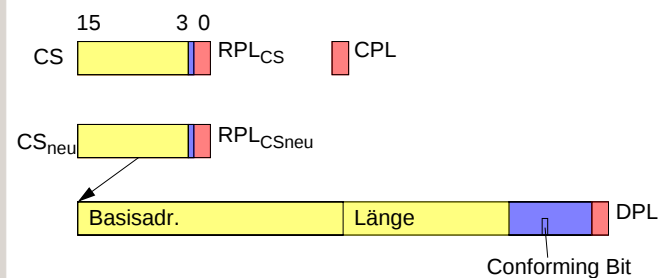
I-Security.fm 1999-02-02 09.53

I.42

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

3 Beispiel: Pentium (4)

- Sprünge in andere Codesegmente (*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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

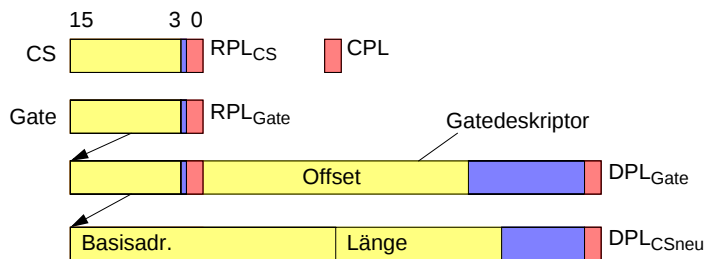
I-Security.fm 1999-02-02 09.53

I.44

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

3 Beispiel: Pentium (5)

■ 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 $DPL_{CSneu} \leq CPL$

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

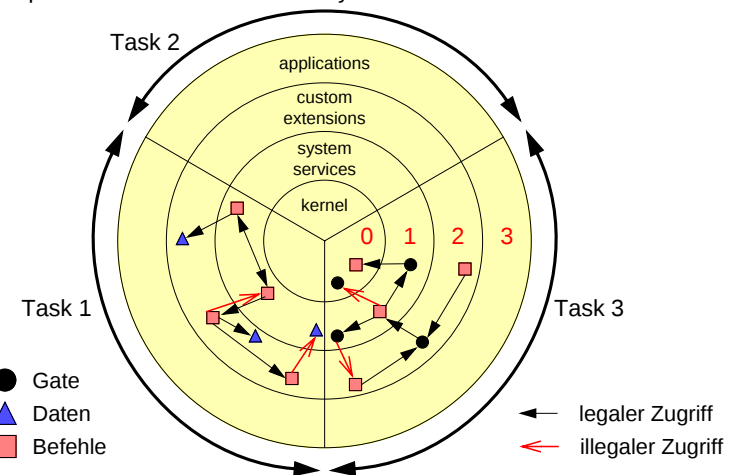
I-Security.fm 1999-02-02 09:53

I.45

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

3 Beispiel: Pentium (7)

■ Beispiel eines realisierten Schutzsystems



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

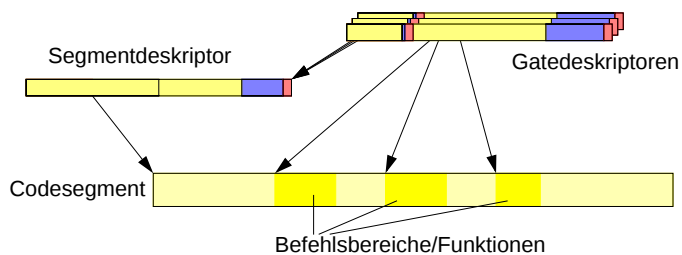
I-Security.fm 1999-02-02 09:53

I.47

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

3 Beispiel: Pentium (6)

■ 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)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

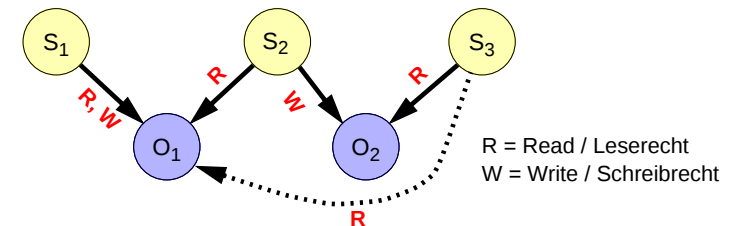
I.46

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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 (O₁ von S₂ nach S₃)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09:53

I.48

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.49

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

1 Beispiel: Hydra (2)

- Prozesse (Subjekte)
 - ◆ Prozesse besitzen einen aktuellen Kontext, den LNS (*Local name space*)
 - ◆ LNS ist ein Objekt
 - ◆ LNS besitzt 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
 - ◆ Capabilities haben einen Typ (z.B. Prozedur, LNS etc.)
 - ◆ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.51

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Semaphoren, Datei etc.)
 - ◆ benutzerdefinierte Typen sind möglich
 - ◆ generische Operationen für alle Typen implementiert durch das Betriebssystem
 - ◆ Objekte besitzen eine Liste von Capabilities auf andere Objekte (genannt *C-List*)
 - ◆ Objekte besitzen einen Datenbereich (implementiert durch geschütztes Segment)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.50

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

1 Beispiel: Hydra (3)

- Datenbereich
 - ◆ Operationen:
 - *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
 - ◆ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.52

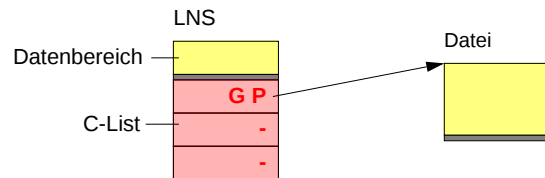
Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiel: Hydra (4)

■ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

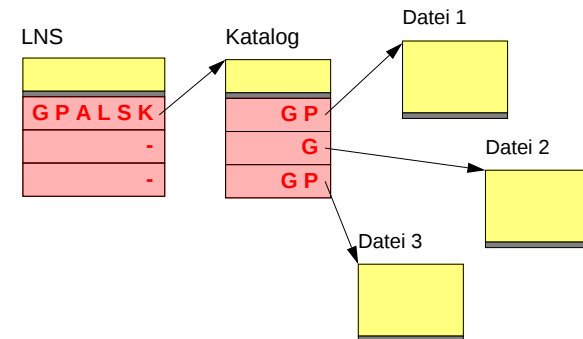
I.53

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

1 Beispiel: Hydra (6)

■ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.55

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

1 Beispiel: Hydra (5)

■ C-List

◆ Operationen:

- *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*

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

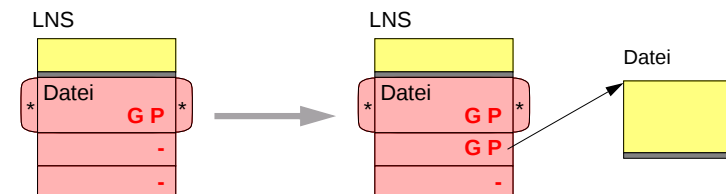
I.54

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

1 Beispiel: Hydra (7)

■ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.56

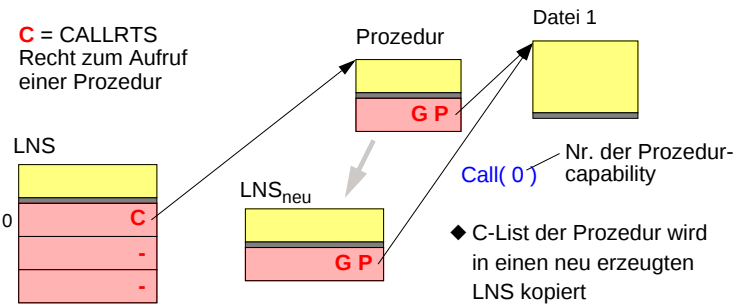
Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Hydra Prozeduraufruf

- 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

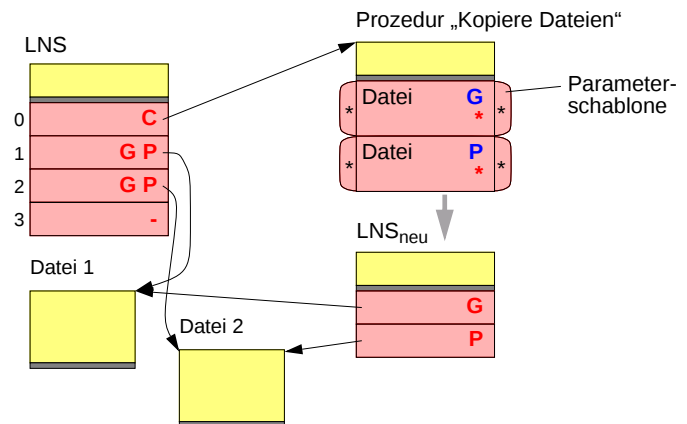
I.57

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

2 Hydra Prozeduraufruf (3)

- Übergabe von Parametern

◆ Beispiel: Prozedur zum Kopieren von Dateiinhalten



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

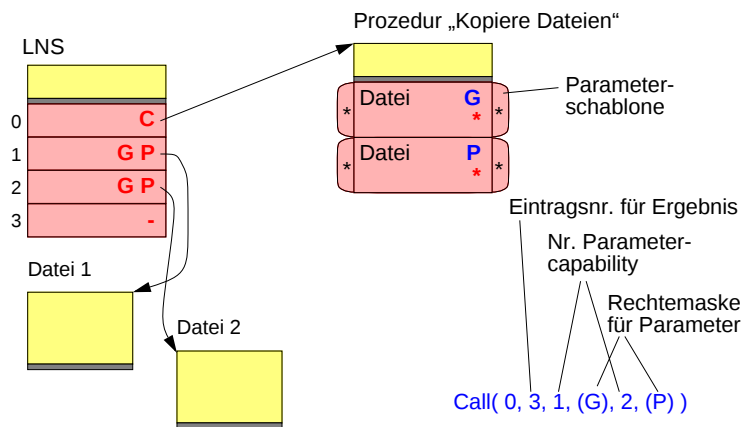
I.59

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

2 Hydra Prozeduraufruf (2)

- Übergabe von Parametern

◆ Beispiel: Prozedur zum Kopieren von Dateiinhalten



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

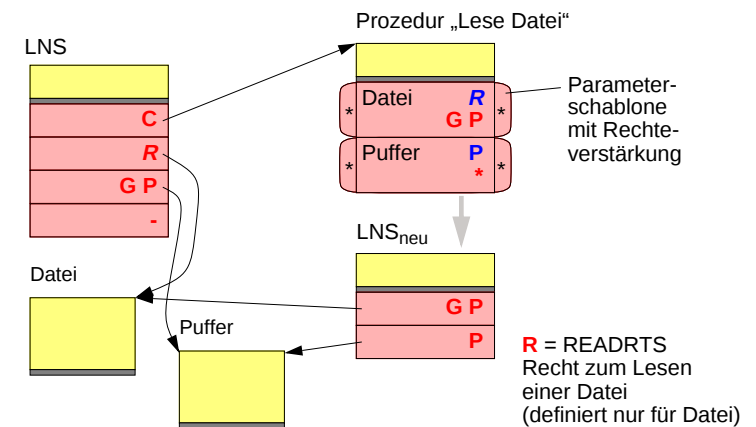
I.58

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

2 Hydra Prozeduraufruf (3)

- Verstärken von Rechten

◆ Beispiel: Prozedur zum Lesen von Dateiinhalten



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.60

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

3 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

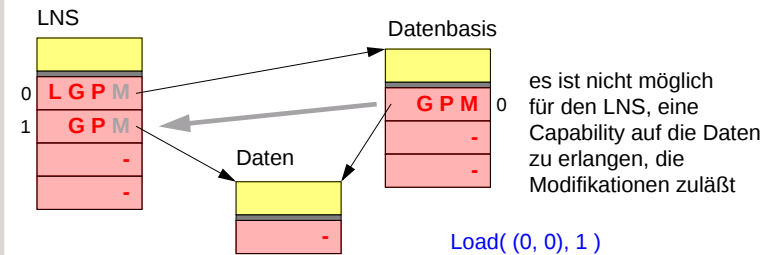
I-Security.fm 1999-02-02 09.53

I.61

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

4 Problem: Modifikation (2)

- ★ Einführung des Modifikationsrechts **MDFYRTS**
 - ◆ für alle modifizierenden Operationen an Datenbereichen und C-Lists muß zusätzlich das Modifikationsrecht 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.63

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

3 Problem: Gegenseitiges Mißtrauen (2)

- ▲ Rechteverstärkung als Sicherheitslücke?
 - ◆ Verstärkungsschablone wird nur an vertrauenswürdige Prozeduren ausgegeben und kann nicht einfach erzeugt werden

4 Problem: Modifikationen

- Aufrufer möchte Modifikationen an und über Parametern 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.62

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

4 Problem: Modifikation (3)

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

5 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.64

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

5 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.65

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

6 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

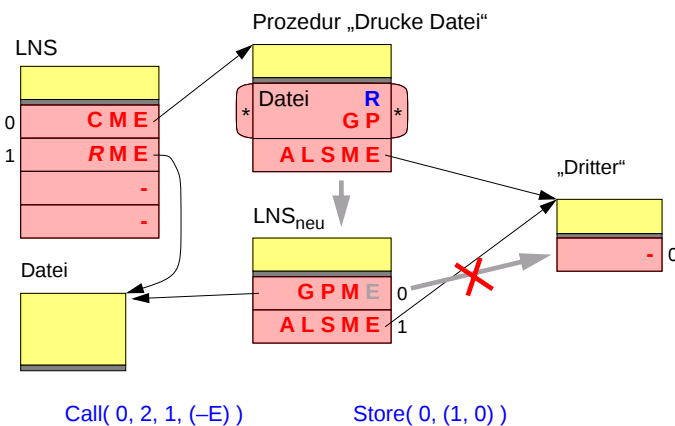
I-Security.fm 1999-02-02 09.53

I.67

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

5 Problem: Ausbreitung von Capabilities (3)

- Versuchte Weitergabe einer Capability an einen Dritten



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.66

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

7 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

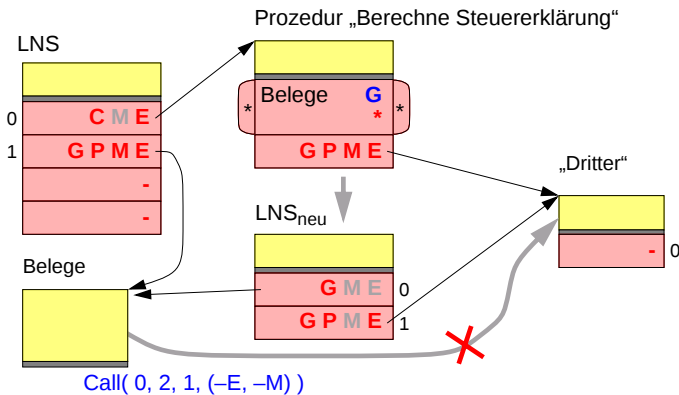
I.68

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

7 Problem: Informationsflußbegrenzung (2)

■ Beispiel: Prozedur zur Steuerberechnung

- ◆ die übergebenen Beleg- und Buchhaltungsdaten sollen nicht weitergegeben werden können



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.69

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

8 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.71

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

8 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

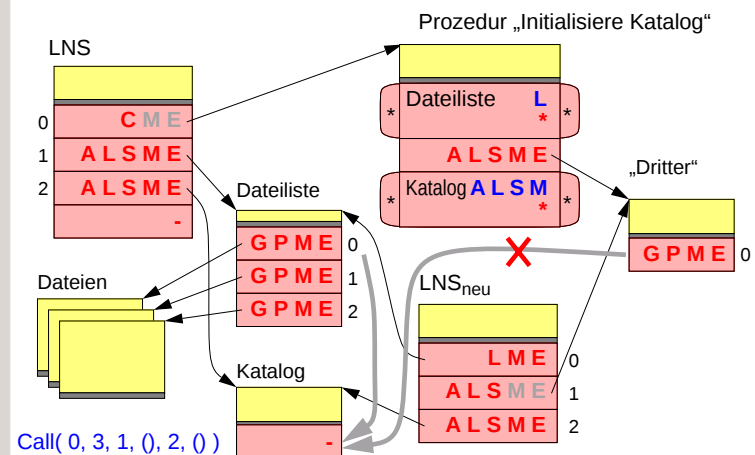
I-Security.fm 1999-02-02 09.53

I.70

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

8 Problem: Initialisierung (3)

■ Beispiel: Initialisierung eines Katalogs



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.72

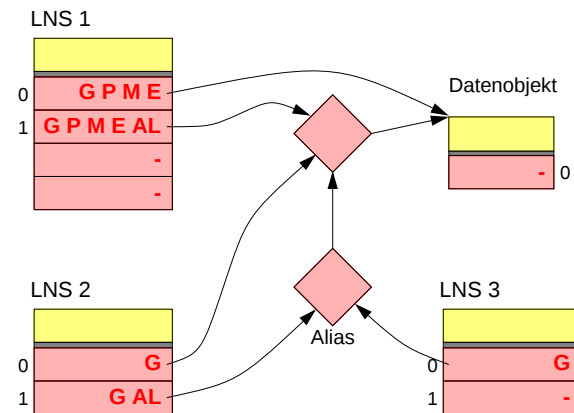
Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 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* = verbunden)

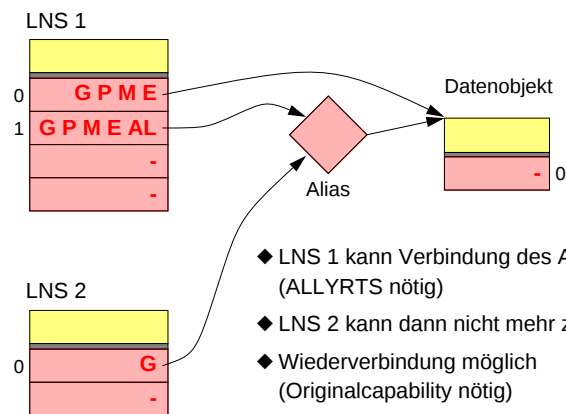
9 Rückruf von Capabilities (3)

- Beispiel: Aliasketten



9 Rückruf von Capabilities (2)

- Beispiel: Weitergabe einer rückrufbaren Capability



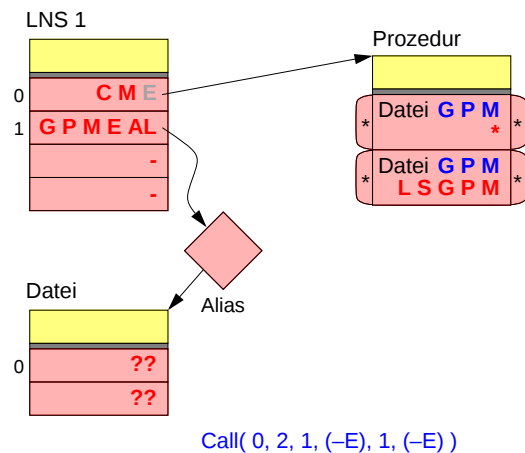
- ◆ LNS 1 kann Verbindung des Alias lösen (ALLYRTS nötig)
- ◆ LNS 2 kann dann nicht mehr zugreifen
- ◆ Wiederverbindung möglich (Originalcapability nötig)
- ◆ Aliase können hintereinander auftreten

9 Rückruf von Capabilities (4)

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

9 Rückruf von Capabilities (5)

■ Beispiel:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.77

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

10 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

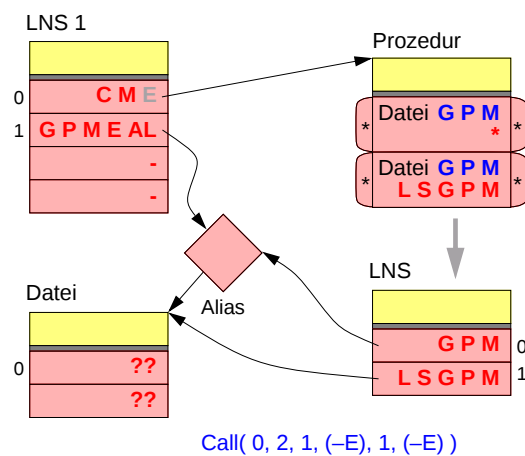
I-Security.fm 1999-02-02 09.53

I.79

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

9 Rückruf von Capabilities (6)

■ Beispiel:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

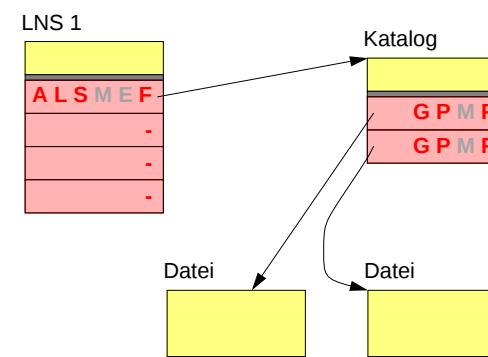
I-Security.fm 1999-02-02 09.53

I.78

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

10 Garantierter Zugriff (2)

■ Beispiel:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.80

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

11 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 Hardware zur Unterstützung von Paging

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.81

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*): $E(K, T) = C$
 - ◆ Entschlüsselungsfunktion D (*decrypt*): $D(K, C) = T$
 - ◆ 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: $K_1 = K_2$

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.83

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.82

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.

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.84

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.85

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.86

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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: Vigenère (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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.87

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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		Originaltext
G	E	H	E	I	M	G	E	H	E	I	M	G	E	H	E	I		Schlüsselwort (wiederholt)

C	M	J	L	B	U	M	I	U	E	K	T	X	M	J	L	B		verschlüsselter Text
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	----------------------

▲ Gilt als nicht sicher

◆ Koinzidenzanalyse

◆ Häufigkeitsanalysen und Brute force Attacke

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.88

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.89

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

3 One-Time Pad Verfahren

■ Theoretisch sicheres Verfahren

- ◆ Liste von Zufallszahlen (so viele 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 $c[i]$ nicht auf $z[i]$ 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

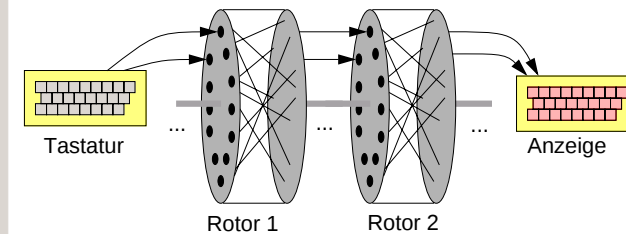
I-Security.fm 1999-02-02 09.53

I.90

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.91

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

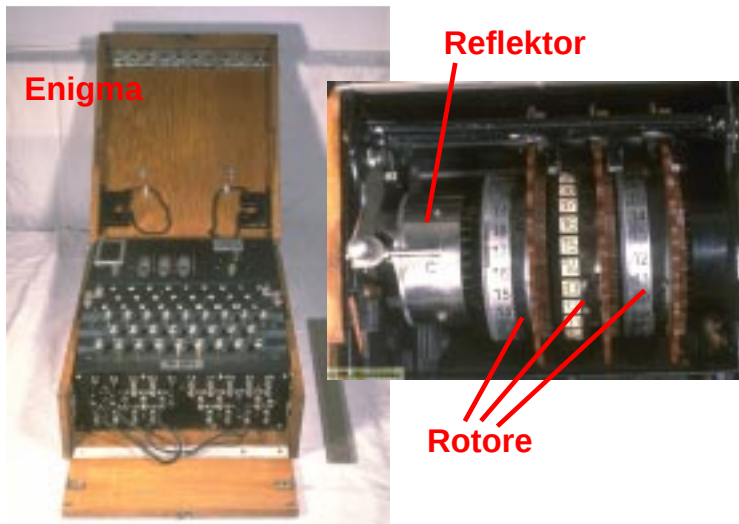
© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.92

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

4 Rotormaschinen



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.93

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

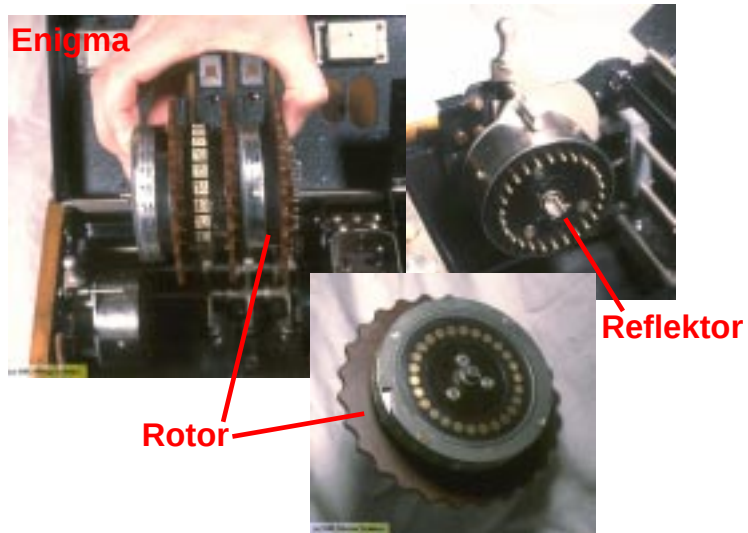
© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.95

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

4 Rotormaschinen



SP I

Systemprogrammierung I

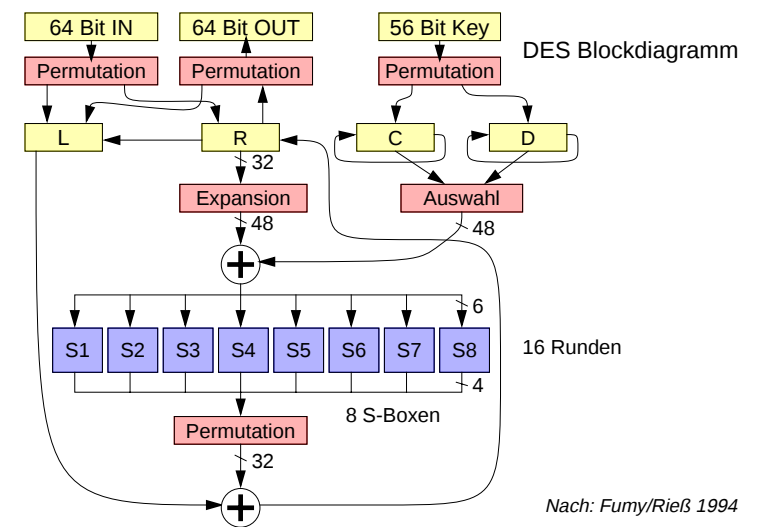
© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.94

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

5 Heutige symmetrische Verfahren (2)



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.96

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.97

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.99

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.98

Reproduktion jeder Art oder Verwendung dieser Unterlagen, 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, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-02 09.53

I.100

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