Computer Security

The Protection of Information in Computer Systems

Jerome H. Saltzer and Michael D. Schroeder, 1975

Definitions

Privacy: the ability to control whether, when, and to whom information is released.

Security: techniques to control who uses or modifies a computer or the data on it.

Types of security violations:

1. Unauthorized information release
2. Unauthorized information modification
3. Unauthorized denial of use

Levels of Information Protection

- **Unprotected systems**: no security
- **All-or-nothing systems**: isolation of users, optionally with globally shared data or libraries; no nuanced levels of permissions
- **Controlled sharing**: explicit control over who can access and modify each file in a system
  - Similar to basic file permissions systems used in modern operating systems
- **User-programmed sharing controls**: user-specified *protected objects and subsystems*; user controls access to a programmable level of detail
- **Putting strings on information**: maintaining control over use and distribution of information after its release

Design Principles

1) **Economy of mechanism**: keep designs small and simple
2) **Fail-safe defaults**: default to exclusion rather than permission
3) **Complete mediation**: every access to everything must be checked
4) **Open design**: security should not depend on secrecy of the design
5) **Separation of privilege**: mechanisms using multiple separate keys are ideal
6) **Least privilege**: give the least permissive privileges required for any task
7) **Least common mechanism**: minimize the mechanisms shared across users
8) **Psychological acceptability**: mechanisms should be user-friendly

Protection Mechanisms

- Isolated virtual machines
- Authentication
  - Passwords
  - Unforgeable keys
  - Cryptographic authentication
- List-oriented protection
  - All users on a list receive access
- Ticket-oriented protection
  - Each user has a list of objects they can access

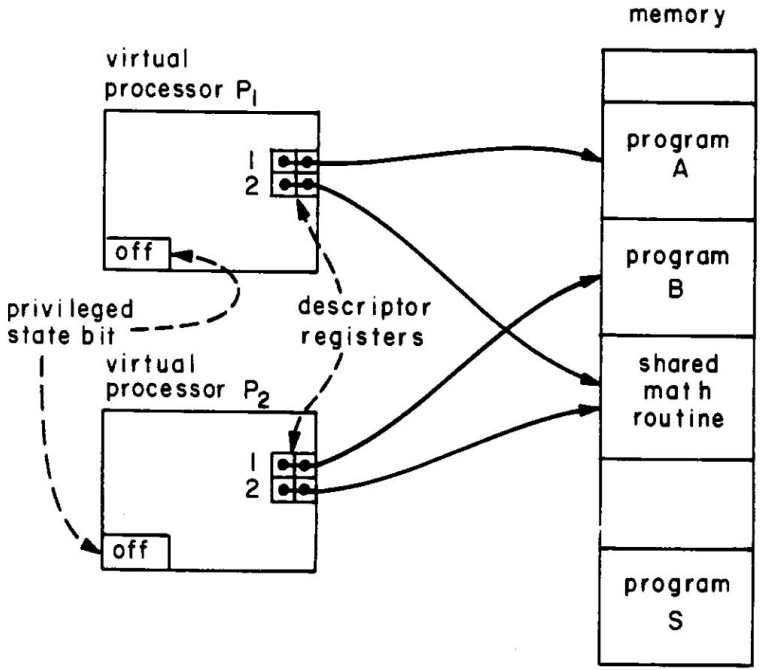


Fig. 3. Fig. 2 redrawn to show sharing of a math routine by two virtual processors simultaneously.

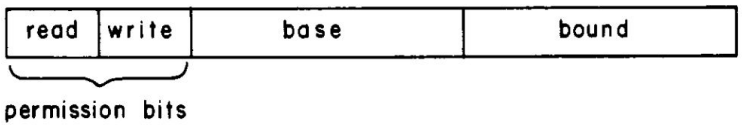


Fig. 4. A descriptor containing READ and WRITE permission bits.

Example: Capability System

- Ticket-based
- Users store protection descriptor registers in arbitrary memory locations (*capabilities*)

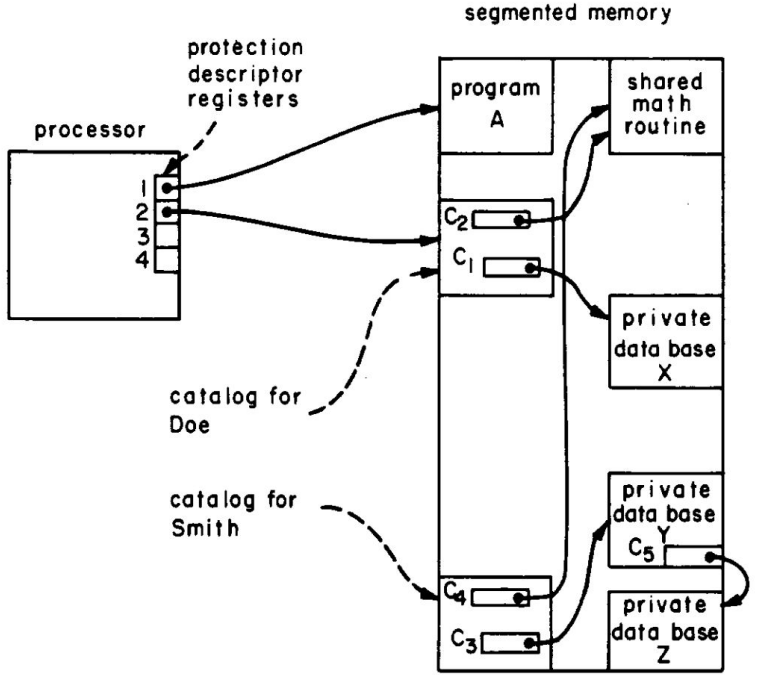


Fig. 6. A simple capability system. Program *A* is in control of the processor. Note that there is no way for the processor to address Smith's catalog or data base *Y*. On the other hand, data base *X* could be accessed by loading capability C_1 into a protection descriptor register. Capability C_1 is loadable because it is stored in a segment that can be reached from a capability already loaded in protection descriptor register 2. Note also that the former function of the privileged state bit has been accomplished by protecting the capabilities. The privileged state bit also has other uses and will be reintroduced later.

Example: Access Control List System

- List-based
- Each memory segment paired with an *access controller*
- Access controllers contain addressing descriptors (i.e., memory addresses) and *access control lists*
- Access control lists determine who can access the memory

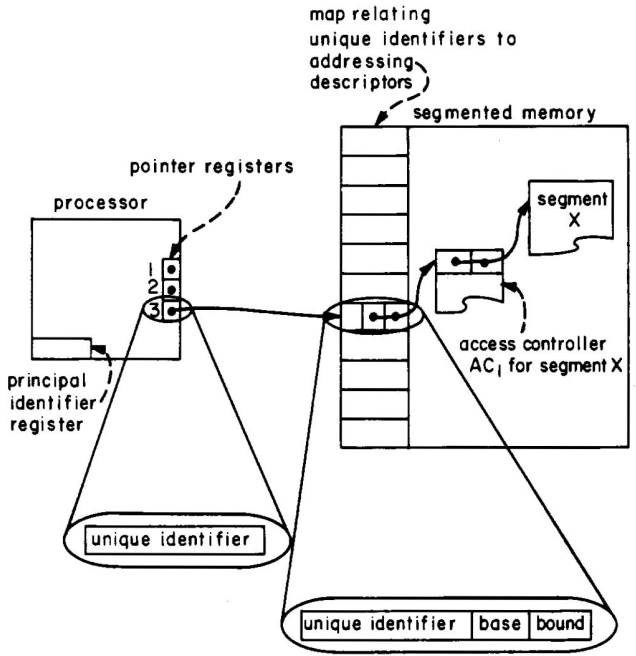


Fig. 9. A revision of Fig. 5, with the addition of an access controller as an indirect address to be used on all references by the processor to the memory. Since the access controller contains permission bits, they no longer need appear in the processor registers, which have been renamed "pointer" registers. Note that the privileged state bit of the processor has been replaced with a principal identifier register.

Example: Protected Subsystem

- Programmable system
- Protected subsystem includes program and memory
- Only accessible by specific entry points, with differing permissions

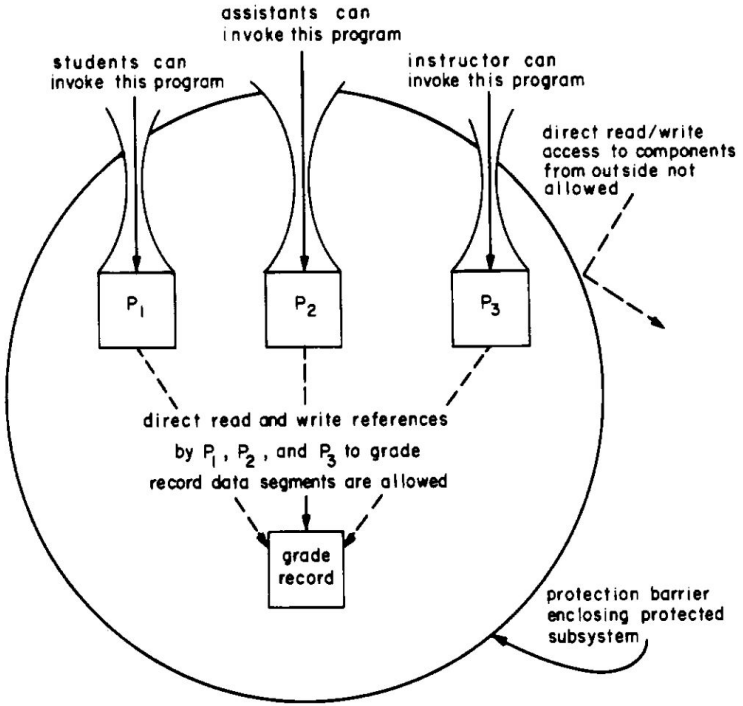


Fig. 14. A protected subsystem to implement the grade-keeping system described in the text. P_1, which can be invoked by all students in the subject, is programmed to return the caller's grade for a particular assignment or the distribution of all grades for an assignment. P_2, which can be invoked by the teaching assistants for the subject, is programmed to allow the addition of new grades to the record but to prevent changing a grade once it is entered. P_3, which can be invoked only by the instructor, is programmed to read or write on request any data in the grade record.

Computer Security in Real World

Butler W. Lampson, 2004

Challenges for Security in Real World

- **Attack from anywhere**: any person on the internet can attack you
- **Sharing with anyone:** you may want to communicate with any user
- **Automated infections:** viruses can spread from your system to many others
- **Hostile code:** especially downloaded from the internet
- **Hostile environment**: public WiFis, physical steal
- **Hostile hosts**: *piracy*

Mental Model of Security in Real World

> What do we want from secure computer systems? Here is a reasonable goal:
>
> *Computers are as secure as real world systems, and people believe it.*

Security is about **value**, **locks**, and **punishments:**

- **Intruder's trade-off:**
  (chance of success)*(reward) **vs** (chance of being caught)*punishment

- **Company's trade-off:**
  (chance of hacking)*(absolute losses) **vs** (cost and inconvenience of a security system)

Terminology

Broader Community	Meaning	Computer Security
Specification	What is it supposed to do?	Policy
Implementation	How does it do it?	Mechanism
Correctness	Does it really work?	Assurance

Policy: Specifying Security

- **Secrecy**: controlling who gets to read the information
- **Integrity**: controlling how information is used
- **Availability**: providing prompt access to information
- **Accountability**: knowing who has had access to information

Examples:

- MIlitary – secrecy
- News Service – ?
- Baking Portal –?

Mechanism: Implementing Security

Sources of Threat

Programs

People

Communications

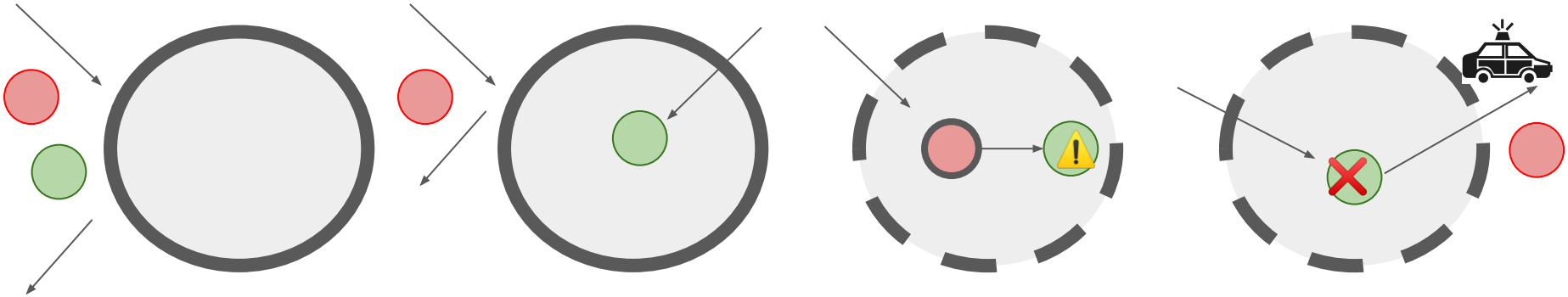
Defensive Strategies

Keep everybody out

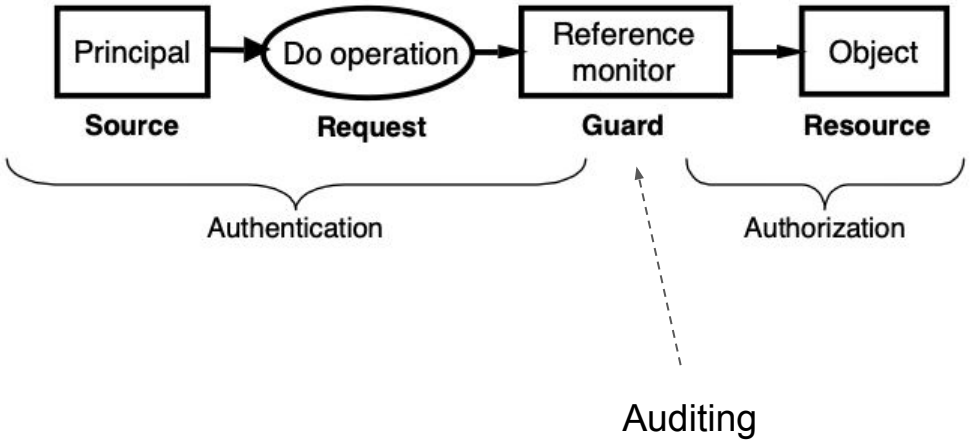
Keep bad guys out

Keep bad guys
from doing damage

Catch bad guys
If the damage is done



Access Control Model



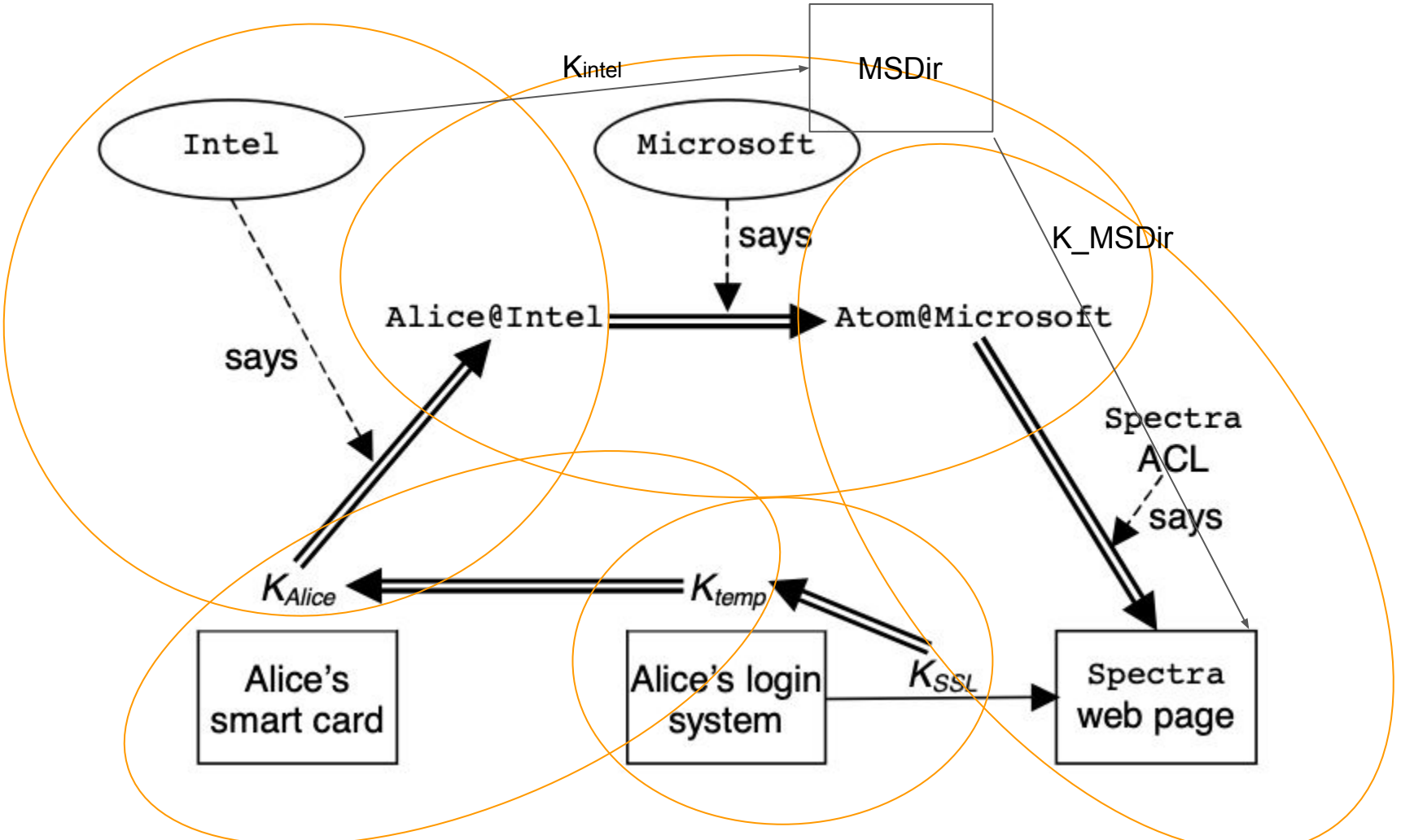
Distributed End-To-End Access Control

P => Q:

- Principal P “speaks for” the principal Q
  - e.g. UW NetID *aksh* speaks for “Aleksei Sholokhov @ UW”
- Q is responsible for everything that P does

Axioms:

- Q can say P => Q – right to grant its own authority
- P $\overset{deleg}{\Rightarrow}$ P/N – parent has authority over children
  - P can say K => P/N
  - E.g. “UW” can say “aksh => Aleksei Sholokhov @ UW”


Kintel
MSDir
Intel
Microsoft
says
K_MSDir
Alice@Intel
Atom@Microsoft
says
Spectra
ACL
says
KAlice
Ktemp
KSSL
Alice's
smart card
Alice's login
system
Spectra
web page

Variations and Implementation Details

- Handling bytes
  - It is important to keep the channels secured end-to-end
- Collecting evidence for each link of responsibility
  - Push – client gathers the evidence and hands it to the object that it wants to access
    - Authentication
  - Pull – the object queries client and/or other databases to collect evidence
    - Authorization
- Summarizing evidence: compress P => Q => R into P => R
  - Handy but hard to revoke access
- Compound principals:
  - Alice AND Bob – both principals need to make action to proceed
  - Alice OR FlakyProgram – FlakyProgram can only access what it's explicitly allowed to.

Discussion

1) Present a famous security breach in a computer system, noting what part of the system was targeted and what was interesting about the attack approach.

- Meltdown and Spectre (2018)
- Pegasus zero-click iPhone attacks
- Java log4j vulnerability
- Row hammer attack

2) Select one of the functional levels of information processing mentioned in the Saltzer paper and give an example not already cited in the paper.

- Modern OS permission systems (controlled sharing)
- DRM (putting strings on information)

3) Consider the example given in the Saltzer paper about the shared math library (shown in Fig 2/6) that uses descriptor based protection but with the users allowed to write to the shared space. If in the context of one of the systems we discussed (eg concurrency in databases), some of the participants could be compromised, what steps can we take to ensure data can be trusted? What implications do the proposed solution have on the system in terms of availability, ease of use/management, etc? How about the use case for protection groups (Fig 10) where a group can be arbitrarily large?

4) What is an example of a case where the statement “You can solve every problem with another level of indirection, except for the problem of too many levels of indirection” applies in computer security or computer science in general?

- Disentangling different modules of a processing pipeline
- Two groups of workers communicating via a master node
- Indirection via SQL