


Basic Principles of Information Protection

Authors: Jerome H. Saltzer & Michael D. Schroeder

Presentation: Christopher Peery



Parts of this Paper


- ◆ Part I - describes functions, design principles and mechanisms
- ◆ Part II – examines the principles of modern protection and does a comparison of capability systems and access control list systems



Functional Levels of Information Protection


Division of Schemes for protecting information:

- Unprotected Systems
- All-or-nothing Systems
- Controlled Sharing
- User-Programmed Sharing Controls
- Putting strings on information
 - Interesting in the sense that the emphasis is not really on trying to provide absolute security but more on just detecting that something has happened and notifying someone.




Dynamics of use is another consideration.
How does one alter or establish security dynamically

- A system can have static setting in terms of security but how would you deal with the “dynamics of use”
- What kind of semantics would you use?




Design Principles Considerations

- ◆ No complete method to the construction of large general-purpose systems exists yet.
 1. Keep the design simple.
 2. Fail-Safe defaults. Base access decisions on permission rather than exclusion.
 3. Complete mediation. Every access to objects must be checked




4. Open design. Strength should be in the key
5. Least Privilege: Give only as many permissions as needed
6. Work Factor: Make the attacker work hard
 - True strength of any system????



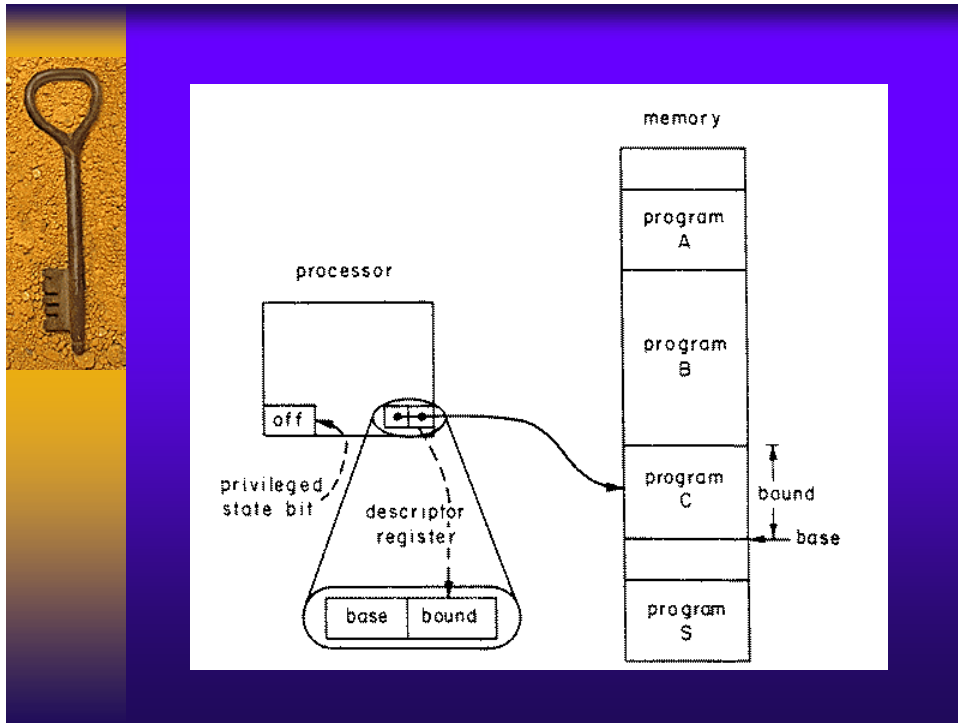
Now some technical implementation Stuff

- ◆ The Basic model :
 - Information is divided on partitions. Access to these partition implies access to the information.
 - The partitions are the fundamental objects to be protected




The Model (cont.)

- ◆ The virtual processors are completely unaware of each other.
- ◆ Each processor has a descriptor register which gives a base and a bound on its intended memory
- ◆ First introduction of the “privileged state” and its bit.
- ◆ Provides protection against each other and against themselves




Authentication Made easy

- ◆ In respect to authentication, objects of protection are the virtual processors
- ◆ Method: Secrecy versus unforgeability (ie password vs. swipe card)
- ◆ PROBLEM: “One way” How will the system authenticate itself to you.



Let us share a little – How to Share information

- ◆ List oriented – user needs ID (should be unforgeable)
- ◆ Ticket oriented – user needs ticket (should be unforgeable)
- ◆ The user is accountable for the actions of a virtual processor. The user is a principal of that processor.
- ◆ All things a principle may use is a domain



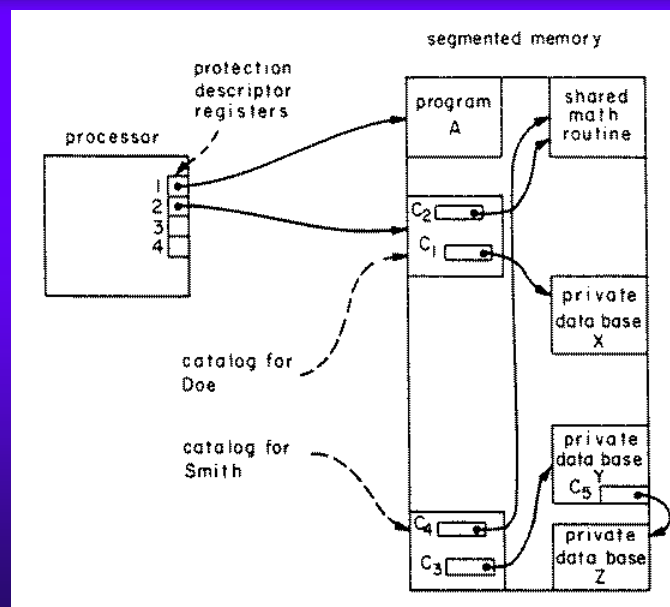
Some more Model Details


- ◆ Address space is universal
- ◆ Any single processor address space is defined by the particular protection descriptor associated with the processor



Using Capabilities


- ◆ All the processor are allowed to load their protection registers themselves
- ◆ This results in two types of objects in memory: protection descriptor values and ordinary data values. A bit is used to distinguish
- ◆ This is known as a “tagged” architecture
- ◆ So the process can load up any protection descriptor that it may need.
- ◆ A catalogue segment can be used to keep track of what may be accessed





Sharing

- ◆ Static Sharing is straight forward. Also dangerous because some else has complete access to your segment.
- ◆ For dynamic authorization, you could set aside a communication segment to prevent this
- ◆ If sharing is done in respect to private communication , this could result in $N*N$ communication segments. To avoid this, implement a “mailbox” segment



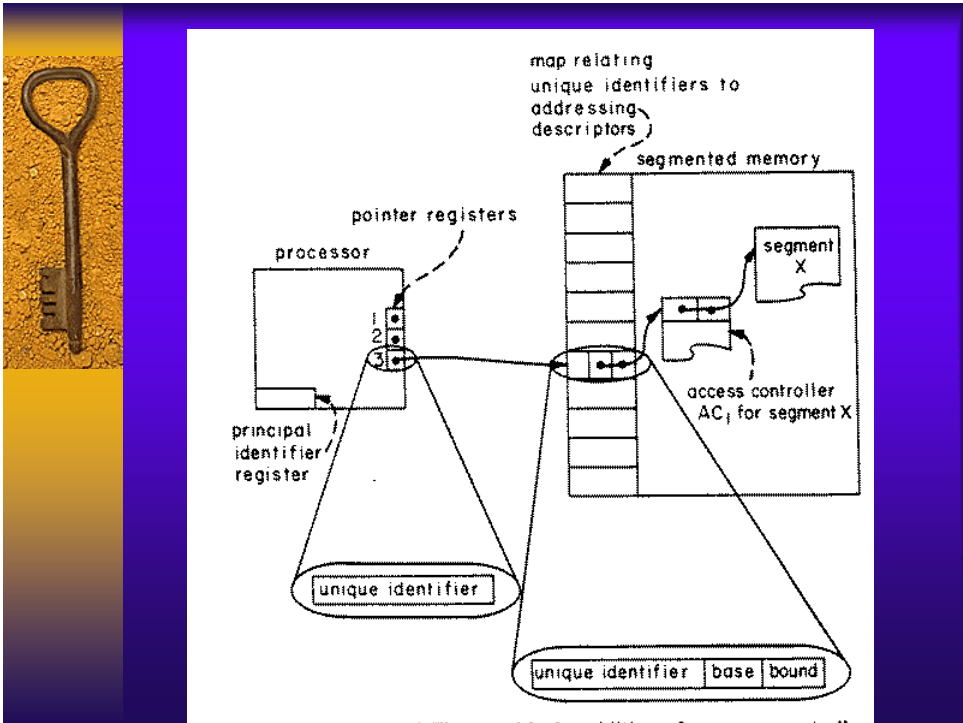
Other issues with Sharing


- ◆ Revocation
- ◆ Propagation
- ◆ Review of access



Access Control System


- ◆ Have access control segments that contain lists of who can access what
- ◆ Each access control segment has a unique identifier that is place in a table.





Changing the Access Controller

- ◆ Possible actions that may be triggered:
 - No action
 - Identifier of principle is logged
 - Change is delayed on day
 - Change is delayed until someone else performs the same action
 - Change is delayed until signal is received from some specific principal



Protected Subsystem

- ◆ This provides a way to encapsulate a collection of programs or data.
- ◆ Access to these programs or data are controlled by fixed entry points

