Security in Operating Systems: Basics
CS 111
Operating Systems
Peter Reiher

Outline

- Basic concepts in computer security
- Access control
- Cryptography

Security: Basic Concepts

- What do we mean by security?
- What is trust?
- Why is security a problem?
  - In particular, a problem with a different nature than, say, performance
  - Or even reliability

What Is Security?

- *Security* is a policy
  - E.g., “no unauthorized user may access this file”
- *Protection* is a mechanism
  - E.g., “the system checks user identity against access permissions”
- Protection mechanisms implement security policies
- We need to understand our goals to properly set our policies
  - And threats to achieving our goals
  - These factors drive which mechanisms we must use

Security Goals

- Confidentiality
  - If it's supposed to be secret, be careful who hears it
- Integrity
  - Don't let someone change something they shouldn't
- Availability
  - Don't let someone stop others from using services
- Exclusivity
  - Don't let someone use something he shouldn't
- Note that we didn't mention "computers" here
  - This classification of security goals is very general

Access Control

- Security could be easy
  - If we didn't want anyone to get access to anything
- The trick is giving access to only the right people
- How do we ensure that a given resource can only be accessed by the proper people?
- The OS plays a major role in enforcing access control

Goals for Access Control

- Complete mediation
- Least privilege
- Useful in a networked environment
- Scalability
- Cost and usability

Common Mechanisms for Access Control in Operating Systems

- Access control lists
  - Like a list of who gets to do something
- Capabilities
  - Like a ring of keys that open different doors
- They have different properties
- And are used by the OS in different ways

The Language of Access Control

- *Subjects* are active entities that want to gain access to something
  - E.g., users or programs
- *Objects* represent things that can be accessed
  - E.g., files, devices, database records
- *Access* is any form of interaction with an object
- An entity can be both subject and object

Access Control Lists

- ACLs
- For each protected object, maintain a single list
- Each list entry specifies a subject who can access the object
  - And the allowable modes of access
- When a subject requests access to a object, check the access control list

An Analogy

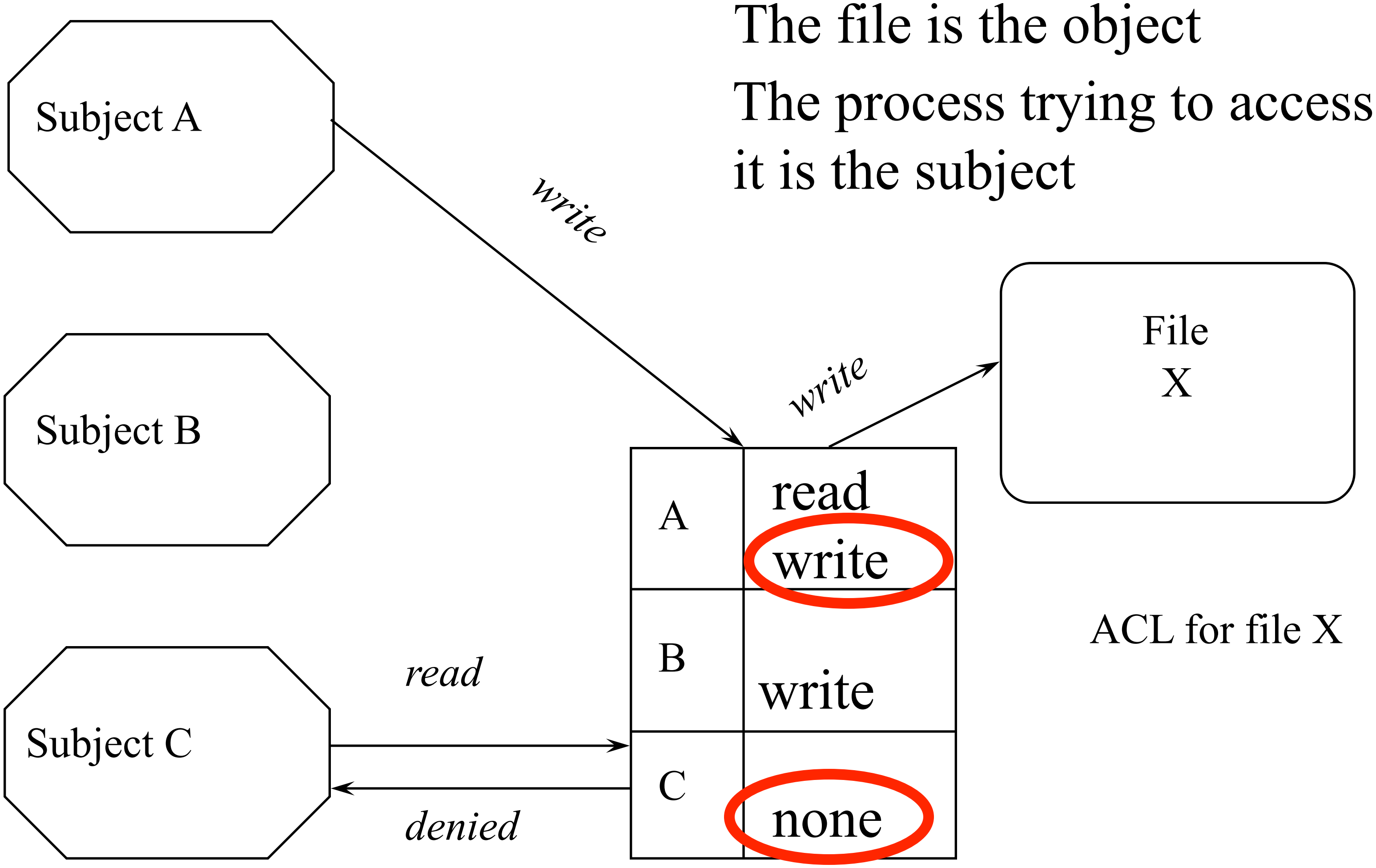
This is an access control list

Joe Hipster

An ACL Protecting a File

The file is the object

The process trying to access it is the subject



A	read write
B	write
C	none

Issues For Access Control Lists

- How do you know the requestor is who he says he is?
- How do you protect the access control list from modification?
- How do you determine what resources a user can access?

Who Is The Requestor?

- Requires authentication
  - At the granularity of the access control list
- For operating systems, commonly that granularity is user
  - But could be process
  - Or something else
- We’ll discuss operating system authentication later

Protecting the ACL

- If entity can change the ACL, all protection disappears
  - Unless the entity is privileged to do so
- ACLs are commonly controlled by the OS
- Changes are made only through specific interfaces
- Allowing checks to be made at the time of the requested change

An Example Use of ACLs: the Unix File System

- An ACL-based method for protecting files
  - Developed in the 1970s
- Still in very wide use today
  - With relatively few modifications
- Per-file ACLs (files are the objects)
- Three subjects on list for each file
    - Owner, group, other
- And three modes
  - Read, write, execute
  - Sometimes these have special meanings

Storing the ACLs

- They can be very small
  - Since there are only three entries
  - Basic ACL is only 9 bits
- Therefore, kept inside the file descriptor
- Makes it easy to find them
  - Since trying to open the file requires the file descriptor, anyway
- Checking this ACL is not much more than a logical AND with the requested access mode

Changing Access Permissions With ACLS

- Mechanically, the OS alone can change an ACL (in most systems)
- But who has the right to ask the OS to do so?
- In simple ACL systems, each object has an owner
  - Only the owner can change the ACL
  - Plus there's often a superuser who can do anything
- In more sophisticated ACL systems, changing an ACL is a mode of access to the object
  - Those with such access can give it to others
  - Or there can even be a meta-mode, which says if someone who can change it can grant that permission to others

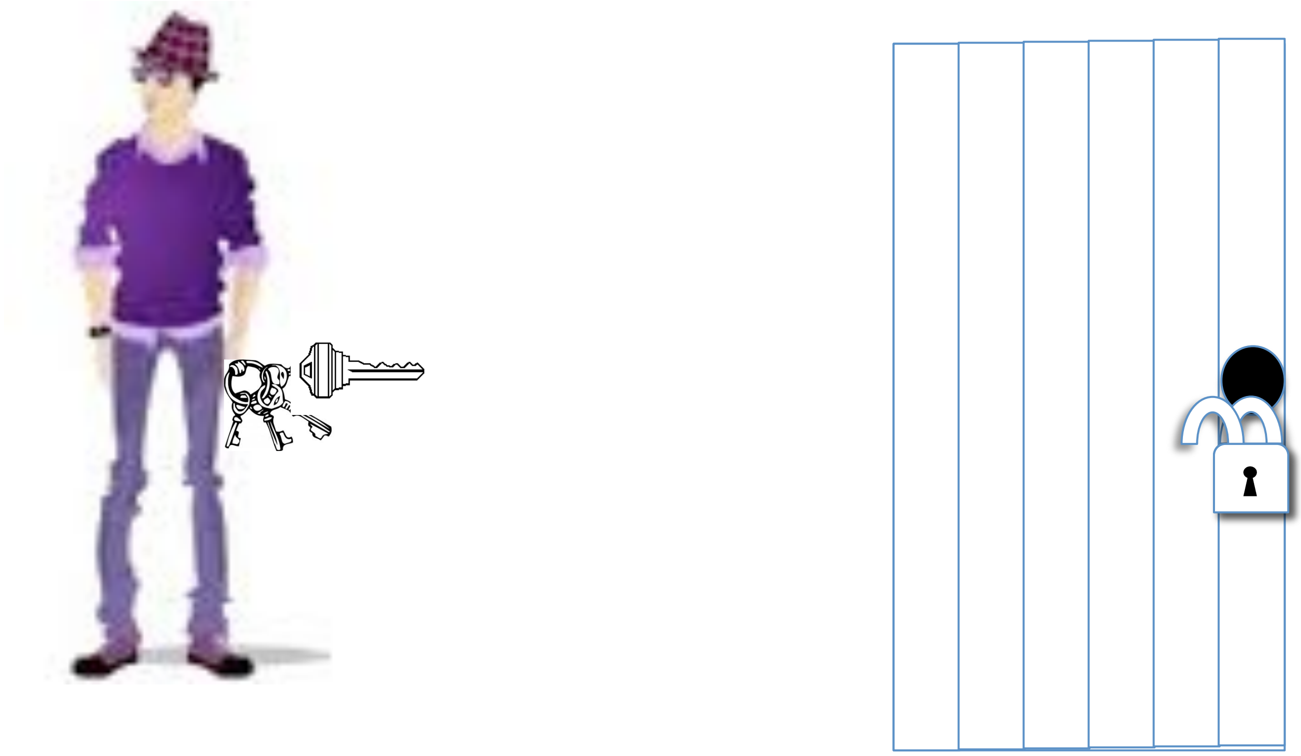
Pros and Cons of ACLs

+ Easy to figure out who can access a resource
+ Easy to revoke or change access permissions
– Hard to figure out what a subject can access
– Changing access rights requires getting to the object

Capabilities

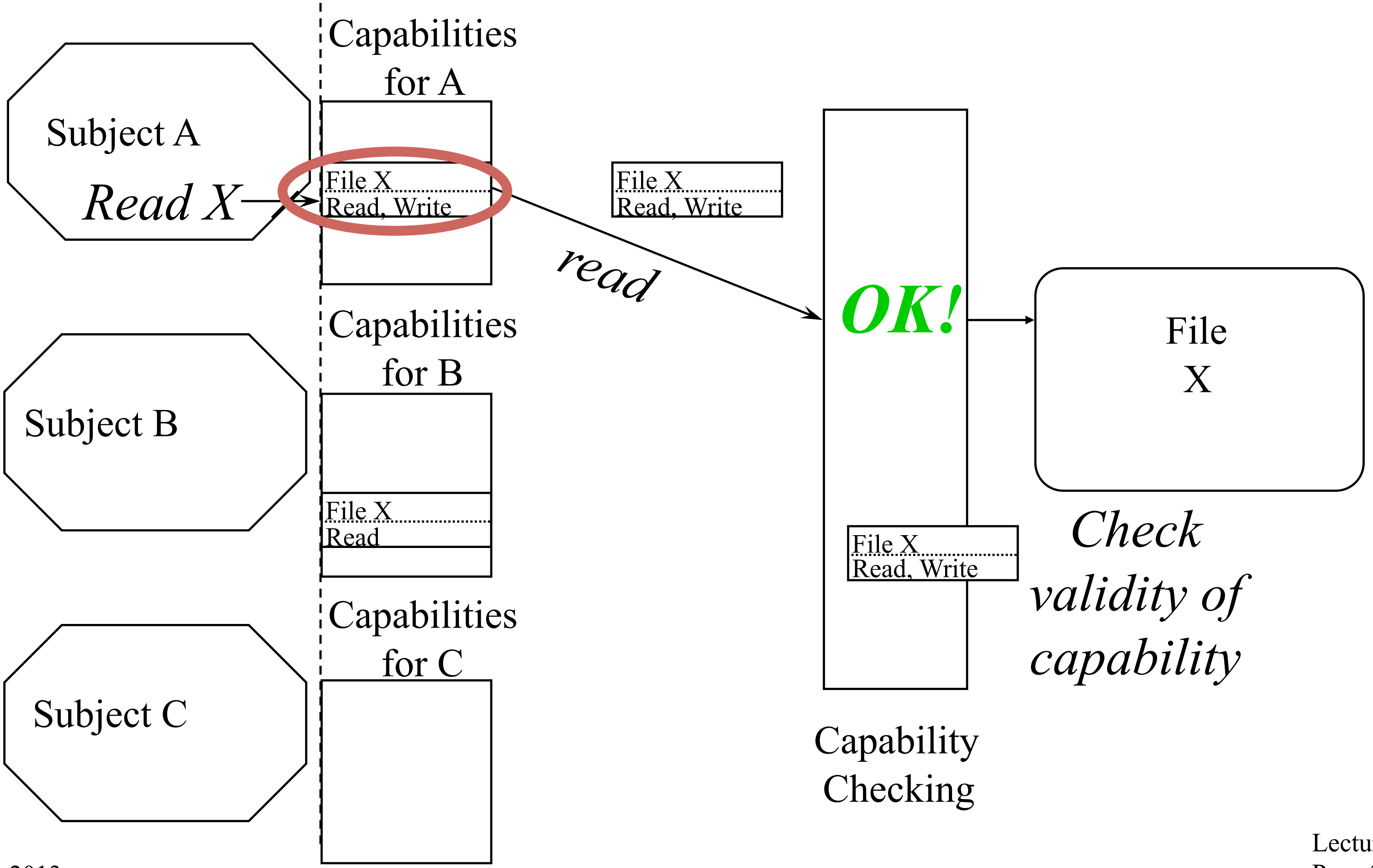
- Each subject keeps a set of data items that specify his allowable accesses
- Essentially, a set of tickets
- To access an object, present the proper capability
- Possession of the capability for an object implies that access is allowed

An Analogy

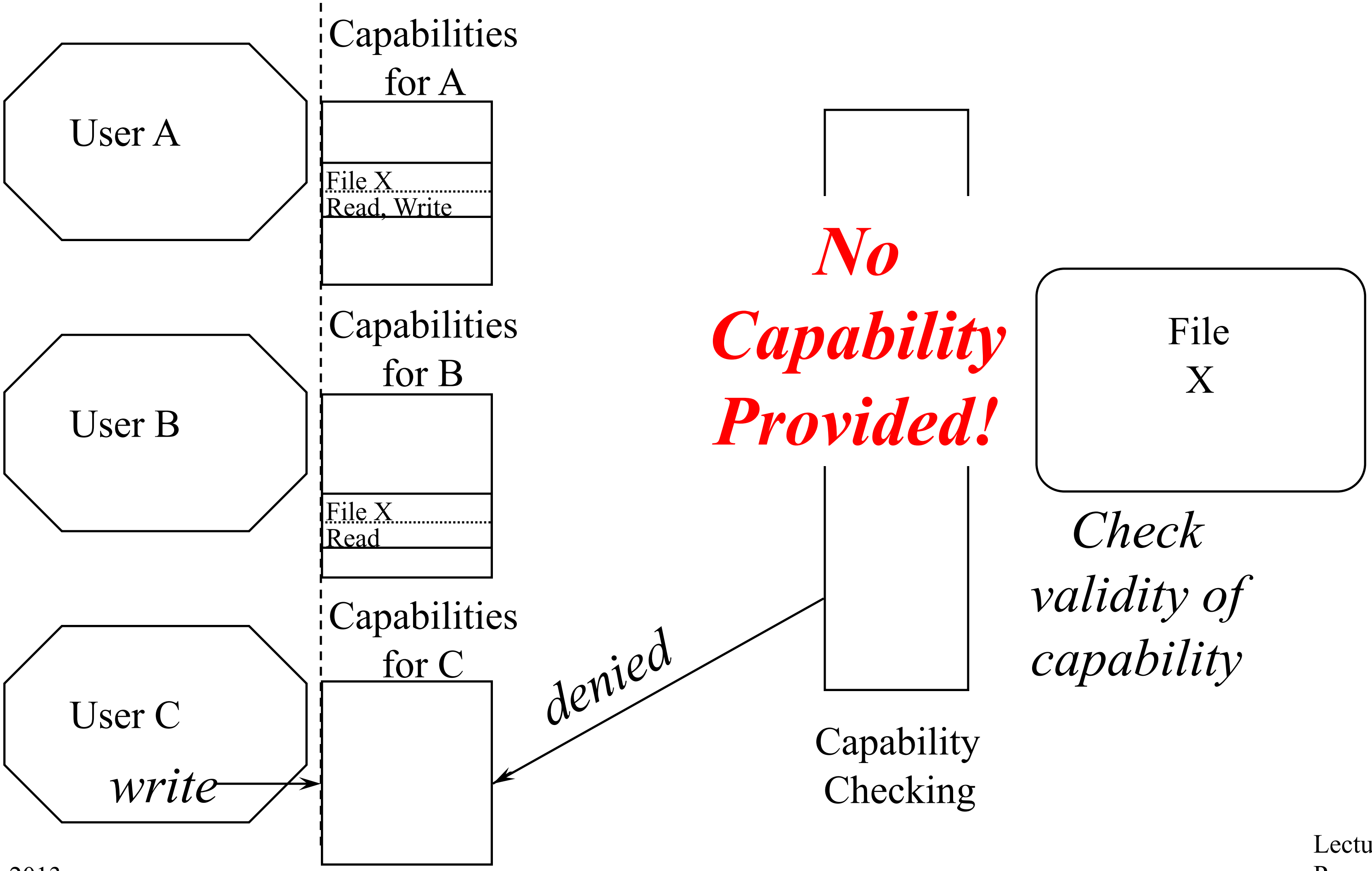


The key is a capability

Capabilities Protecting a File



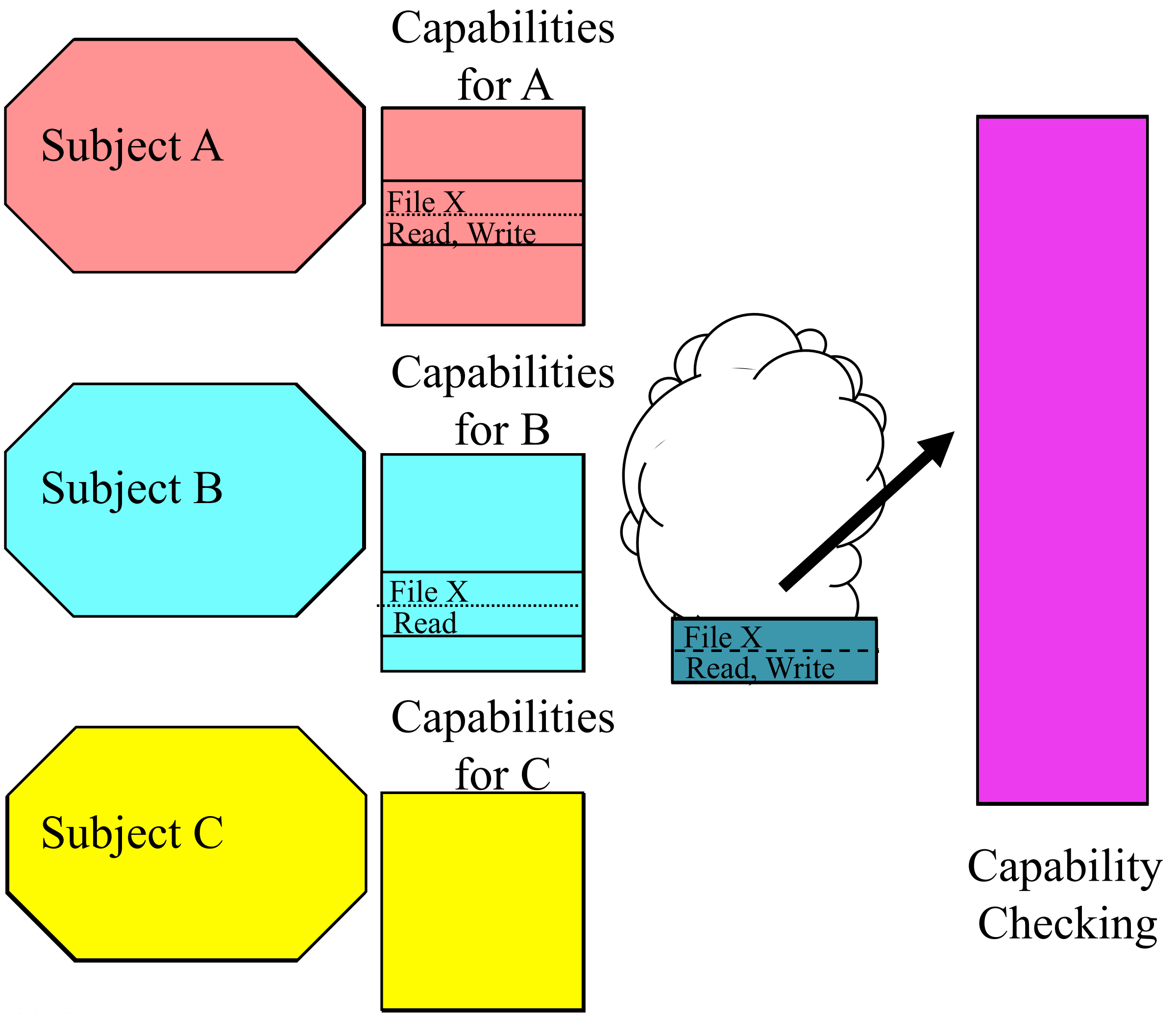
Capabilities Denying Access



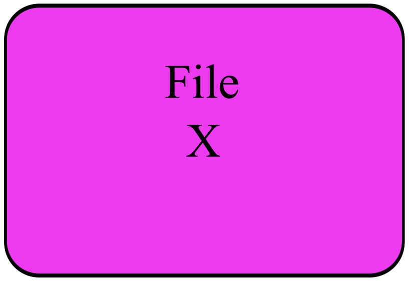
Properties of Capabilities

- Capabilities are essentially a data structure
  - Ultimately, just a collection of bits
- Merely possessing the capability grants access
  - So they must not be forgeable
- How do we ensure unforgeability for a collection of bits?
- One solution:
  - Don’t let the user/process have them
  - Store them in the operating system

Capabilities and Networks



How can we tell if it’s a good capability?



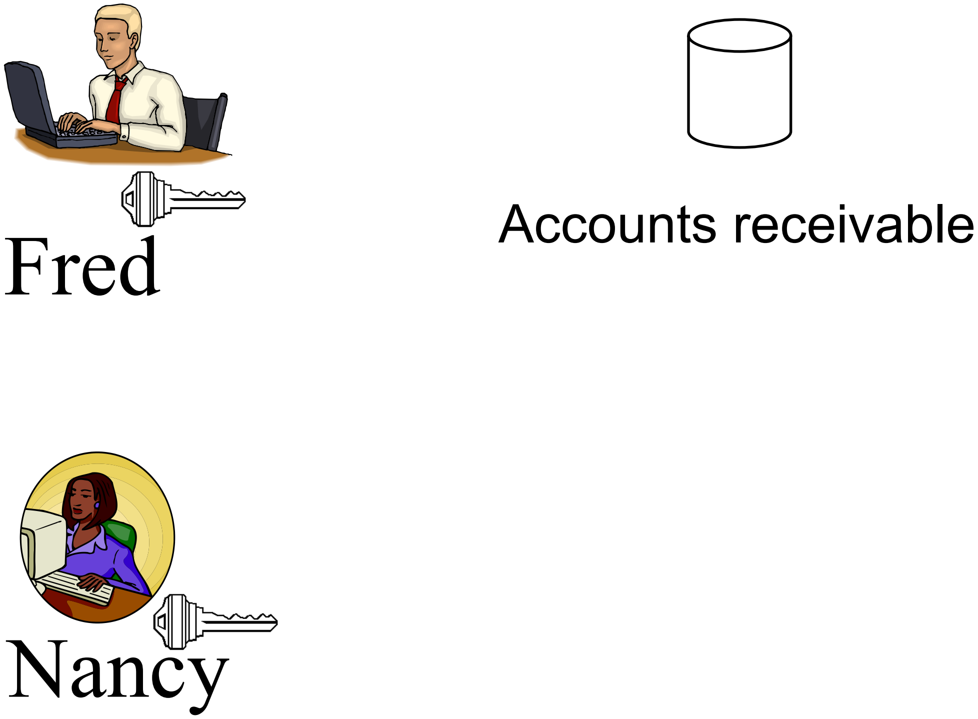
Cryptographic Capabilities

- Create unforgeable capabilities by using cryptography
  - We'll discuss cryptography in detail in the next lecture
- Essentially, a user CANNOT create this capability for himself
- The examining entity can check the validity
- Prevents creation of capabilities from nothing
  - But doesn't prevent copying them

Revoking Capabilities

- A simple problem for capabilities stored in the operating system
  - Just have the OS get rid of it
- Much harder if it's not in the operating system
  - E.g., in a network context
- How do we make the bundle of bits change from valid to invalid?
- Consider the real world problem of a door lock
- If several people have the key, how do we keep one of them out?

Illustrating the Problem



Fred

Accounts receivable

Nancy

How do we take away Fred’s capability?

Without taking away Nancy’s?

Changing Access Permissions With Capabilities

- Essentially, making a copy of the capability and giving it to someone else
- If capabilities are inside the OS, it must approve
- If capabilities are in user/process hands, they just copy the bits and hand out the copy
  - Crypto methods can customize a capability for one user, though
- Capability model often uses a particular type of capability to control creating others
  - Or a mode associated with a capability

Pros and Cons of Capabilities

+ Easy to determine what objects a subject can access
+ Potentially faster than ACLs (in some circumstances)
+ Easy model for transfer of privileges
– Hard to determine who can access an object
– Requires extra mechanism to allow revocation
– In network environment, need cryptographic methods to prevent forgery

OS Use of Access Control

- Operating systems often use both ACLs and capabilities
  - Sometimes for the same resource
- E.g., Unix/Linux uses ACLs for file opens
- That creates a file descriptor with a particular set of access rights
  - E.g., read-only
- The descriptor is essentially a capability

Enforcing Access in an OS

- Protected resources must be inaccessible
  - Hardware protection must be used to ensure this
  - So only the OS can make them accessible to a process
- To get access, issue request to resource manager
  - Resource manager consults access control policy data
- Access may be granted directly
  - Resource manager maps resource into process
- Access may be granted indirectly
  - Resource manager returns a “capability” to process

Direct Access To Resources

- OS checks access control on initial request
- If OK, OS maps it into a process' address space
  - The process manipulates resource with normal instructions
  - Examples: shared data segment or video frame buffer
- Advantages:
  - Access check is performed only once, at grant time
  - Very efficient, process can access resource directly
- Disadvantages:
  - Process may be able to corrupt the resource
  - Access revocation may be awkward
    - You've pulled part of a process' address space out from under it

Indirect Access To Resources

- Resource is not directly mapped into process
  - Process must issue service requests to use resource
  - Access control can be checked on each request
  - Examples: network and IPC connections
- Advantages:
  - Only resource manager actually touches resource
  - Resource manager can ensure integrity of resource
  - Access can be checked, blocked, revoked at any time
    - If revoked, system call can just return error code
- Disadvantages:
  - Overhead of system call every time resource is used