

# Tools for Security

- Physical security
- Access control
- Encryption
- Authentication
- Encapsulation
- Intrusion detection
- Common sense

# Physical Security

- Lock up your computer
  - Actually, sometimes a good answer
- But what about networking?
  - Networks poke a hole in the locked door
- Hard to prevent legitimate holder of a computer from using it as he wants
  - E.g., smart phone jailbreaks
- In any case, lack of physical security often makes other measures pointless

# Access Controls

- Only let authorized parties access the system
- A lot trickier than it sounds
- Particularly in a network environment
- Once data is outside your system, how can you continue to control it?
  - Again, of concern in network environments

# Encryption

- Algorithms to hide the content of data or communications
- Only those knowing a secret can decrypt the protection
- One of the most important tools in computer security
  - But not a panacea
- Covered in more detail later in class

# Authentication

- Methods of ensuring that someone is who they say they are
- Vital for access control
- But also vital for many other purposes
- Often (but not always) based on encryption

# Encapsulation

- Methods of allowing outsiders limited access to your resources
- Let them use or access some things
  - But not everything
- Simple, in concept
- Extremely challenging, in practice

# Intrusion Detection

- All security methods sometimes fail
- When they do, notice that something is wrong
- And take steps to correct the problem
- Reactive, not preventative
  - But it's unrealistic to believe any prevention is certain
- Must be automatic to be really useful

# Common Sense

- A lot of problems arise because people don't like to think
- The best security tools generally fail if people use them badly
- If the easiest way in is to fool people, that's what attackers will do



# 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
  - And at the right time and circumstances
- How do we ensure that a given resource can only be accessed when it should be?

# Goals for Access Control

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

# Access Control Mechanisms

- Access control lists
- Capabilities
- Access control matrices
  - Theoretical concept we won't discuss in detail
- Role based access control

# 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

# Mandatory vs. Discretionary Access Control

- Mandatory access control is dictated by the underlying system
  - Individual users can't override it
  - Even for their own data
- Discretionary access control is under command of the user
  - System enforces what they choose
  - More common than mandatory

# Access Control Lists

- For each protected resource, maintain a single list
- Each list entry specifies a user who can access the resource
  - And the allowable modes of access
- When a user requests access to a resource, check the access control list (ACL)

# ACL Objects and Subjects

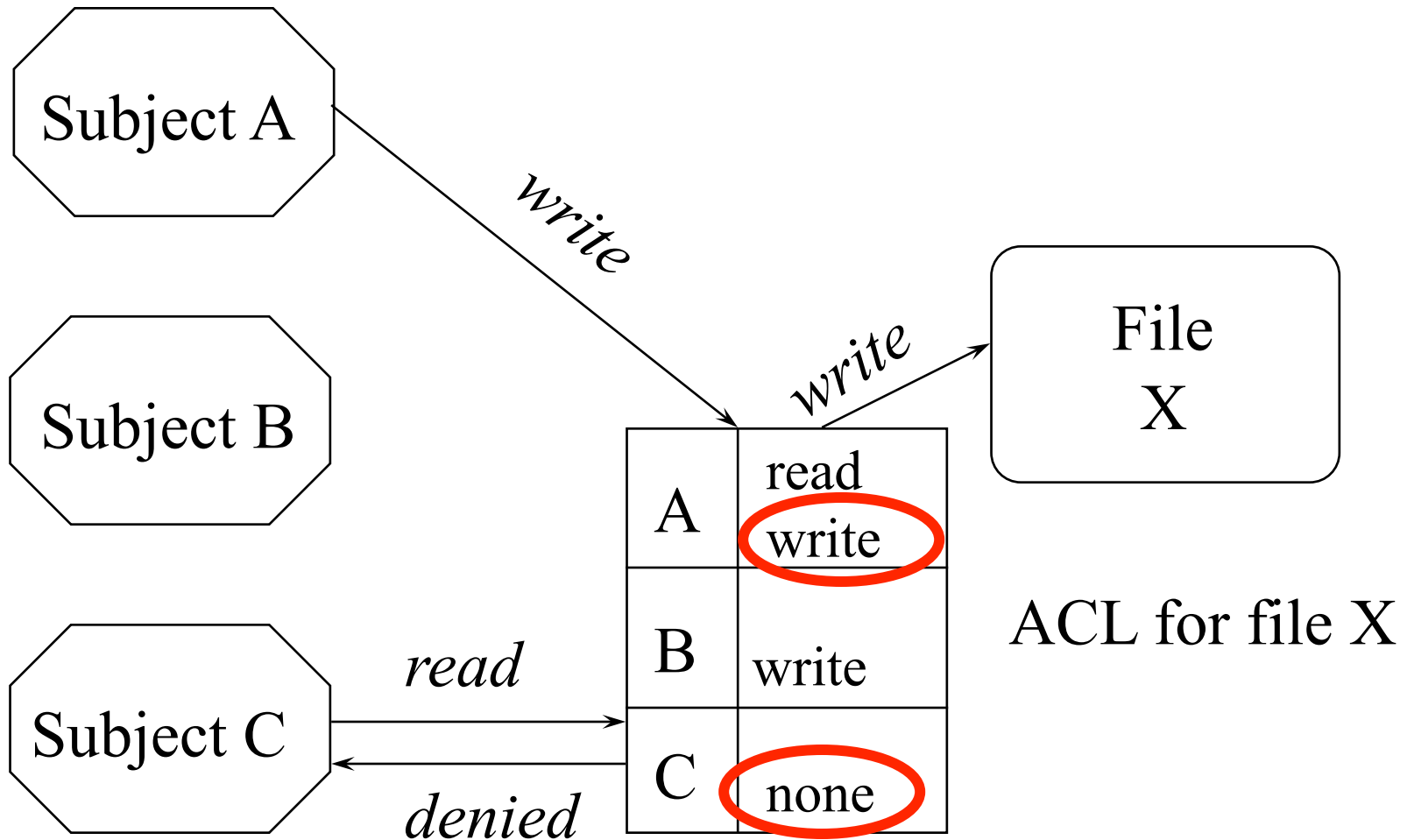
- In ACL terminology, the resources being protected are *objects*
- The entities attempting to access them are *subjects*
  - Allowing finer granularity of control than per-user

# ACL Example

- An operating system example:
  - Using ACLs to protect a file
- User (Subject) A is allowed to read and write to the file
- User (Subject) B may only read from it
- User (Subject) C may not access it



# An ACL Protecting a File



# Issues for Access Control Lists

- How do you know that 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?
- Generally issues for OS design

# 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

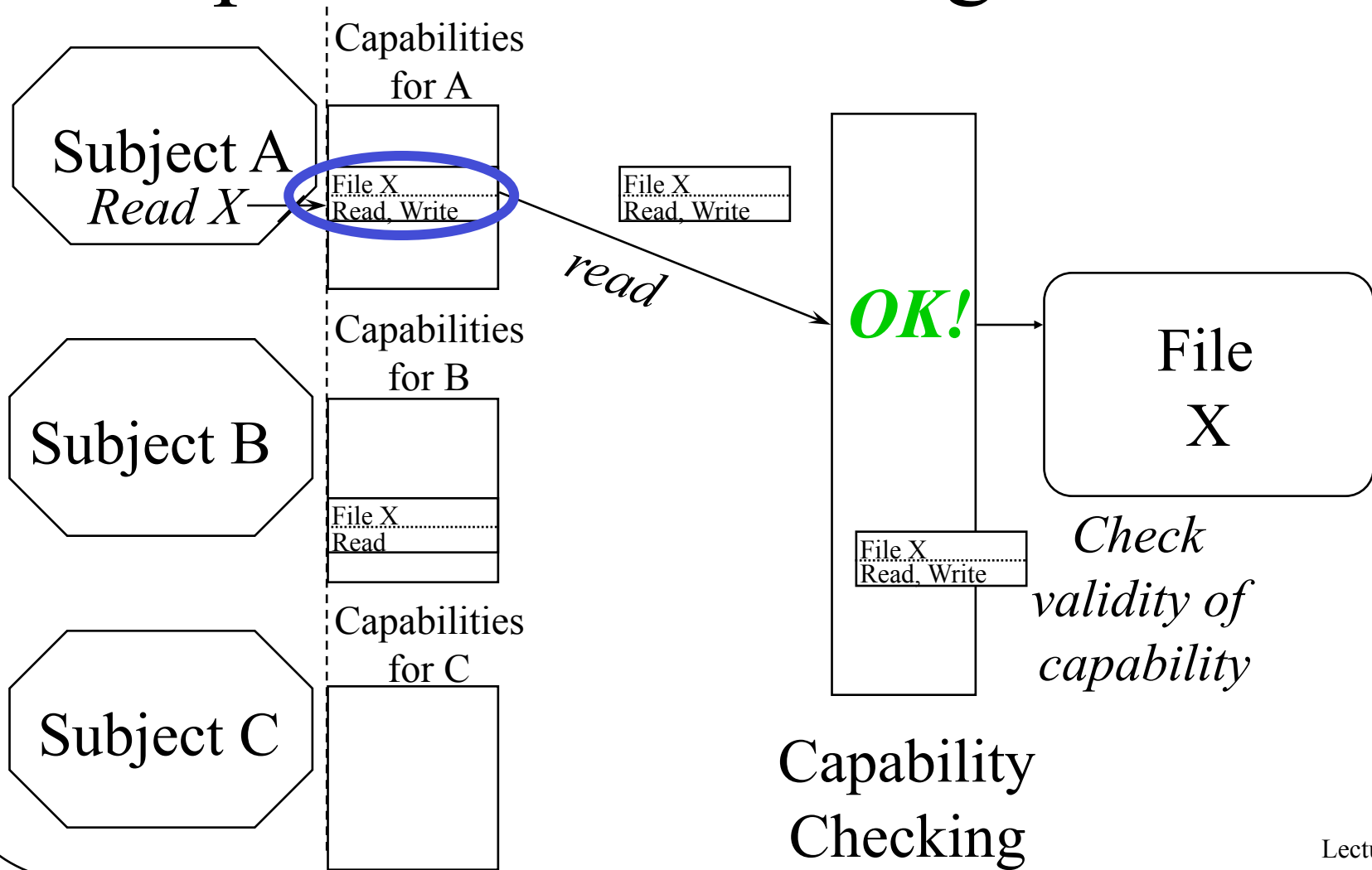
# Capabilities

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

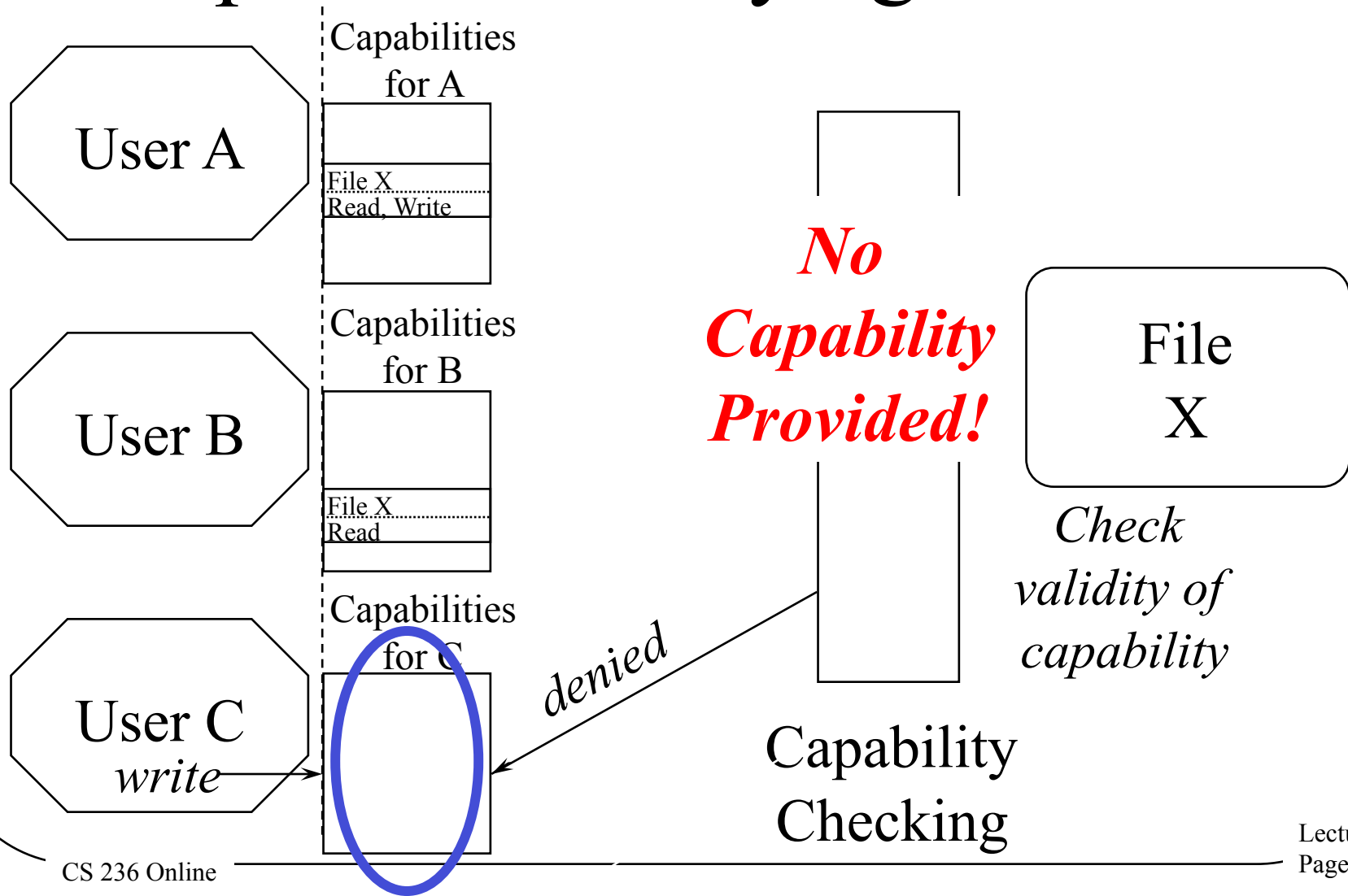
# Properties of Capabilities

- Must be unforgeable
  - In single machine, keep capabilities under control of OS
  - What about in a networked system?
- In most systems, some capabilities allow creation of other capabilities
  - Process can pass a restricted set of capabilities to a subprocess

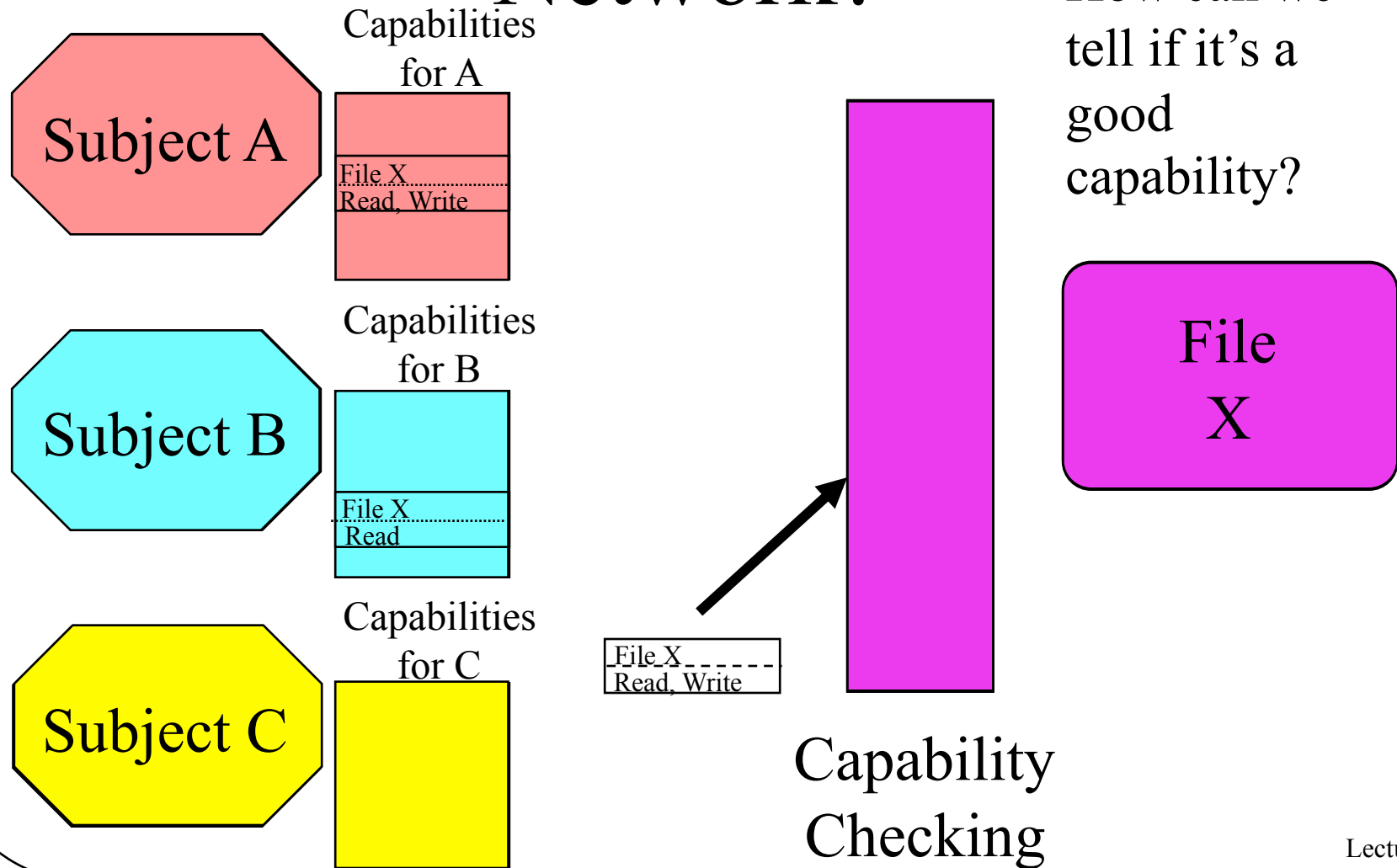
# Capabilities Protecting a File



# Capabilities Denying Access

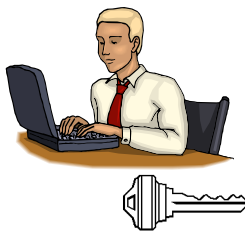


# How Will This Work in a Network?

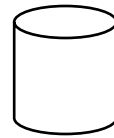




# Revoking Capabilities



Fred



Accounts  
receivable

How do we take  
away Fred's  
capability?



Nancy

Without taking  
away Nancy's?

# Options for Revoking Capabilities

- Destroy the capability
  - How do you find it?
- Revoke on use
  - Requires checking on use
- Generation numbers
  - Requires updating non-revoked capabilities

# Pros and Cons of Capabilities

- + Easy to determine what 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

# Distributed Access Control

- ACLs still work OK
  - Provided you have a global namespace for subjects
  - And no one can masquerade
- Capabilities are more problematic
  - Security relies on unforgeability
  - Provided by cryptographic methods
  - Prevents forging, not copying

# Role Based Access Control

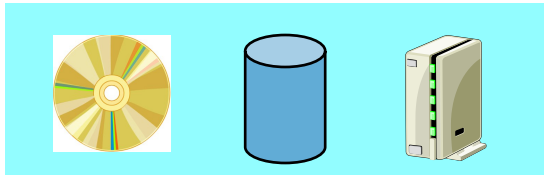
- An enhancement to ACLs or capabilities
- Each user has certain roles he can take while using the system
- At any given time, the user is performing a certain role
- Give the user access to only those things that are required to fulfill that role
- Available in some form in most modern operating systems

# A Simple Example

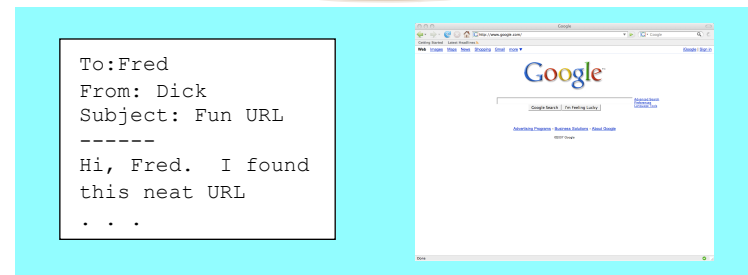
Fred is a system administrator



But Fred is also a normal user



Fred should operate under one role while doing system administration



And another role while doing normal stuff

# Continuing With Our Example

He decides to upgrade the C++ compiler



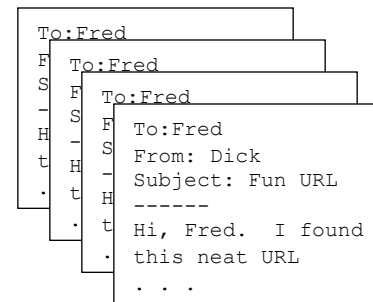
Fred logs on as “fred”

He reads his email

So he changes his role to “sysadmin”

Then he has the privileges to upgrade the compiler

But may have lost the privileges to read “fred’s” email



***Result: Evil malware in fred’s email can’t “upgrade” the compiler***

# Changing Roles

- Role based access control only helps if changing roles isn't trivial
  - Otherwise, the malicious code merely changes roles before doing anything else
- Typically requires providing some secure form of authentication
  - Which proves you have the right to change roles
  - Usually passwords, but other methods possible



# Practical Limitations on Role Based Access Control

- Number of roles per user
- Problems of disjoint role privileges
- System administration overheads
- Generally, these cause usability and management problems

# Reference Monitors

- Whatever form it takes, access control must be instantiated in actual code
  - Which checks if a given attempt to reference an object should be allowed
- That code is called a *reference monitor*
- Obviously, good reference monitors are critical for system security

# Desirable Properties of Reference Monitors

- Correctness
- Proper placement
- Efficiency
- Simplicity
- Flexibility