Graduate Course on **Computer Security**

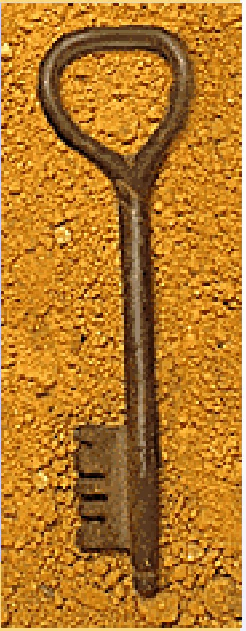
Lecture 1: Information Assurance

Iliano Cervesato iliano@itd.nrl.navy.mil

ITT Industries, Inc @ NRL - Washington DC

http://www.cs.stanford.edu/~iliano/

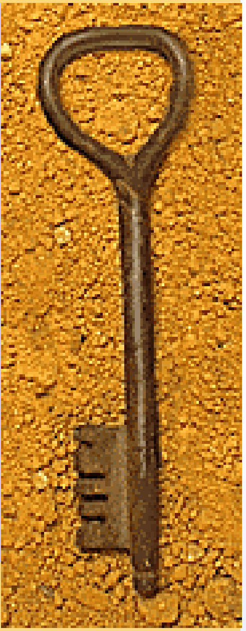
DIMI, Universita' di Udine, Italy December 3, 2001



Outline

- Unintended behaviors
  - Errors and attacks
  - Policies, mechanisms, assurance
- Access Control
  - Governing principles
  - Discretionary AC
  - Mandatory AC
- Information flow
  - Covert channels
  - Stegonography
- Secure execution
  - Safe programs
  - Mobile code

Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

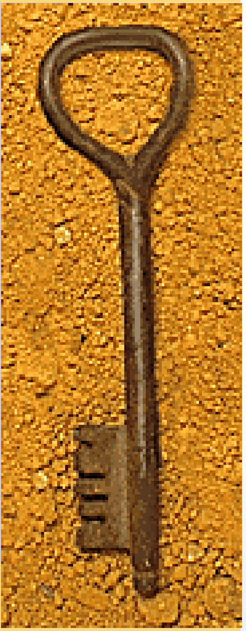
Unintended Behaviors

and remedies

Systems don't meet their functional requirements

- Environmental disruptions
  - ⇒ Fault-tolerant architecture
  - ⇒ Stronger interfaces
- Operator errors
  - ⇒ Education and training
  - ⇒ Better human-computer interfaces
- Poor design/implementation (bugs)
  - ⇒ Languages and tools
  - ⇒ Testing and verification
- Deliberate attacks
  - ⇒ ~~Lower expectations~~
  - ⇒ Security engineering
    - ➢ Theoretical foundations

This course

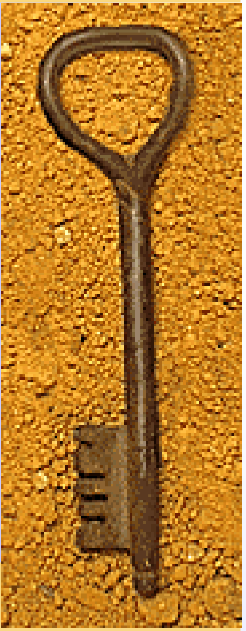


Correctness vs. Security [Mitchell]

- Correctness: satisfy specifications
  - ➢ For reasonable inputs, get reasonable output
- Security: resist attacks
  - ➢ For unreasonable inputs, output not completely disastrous

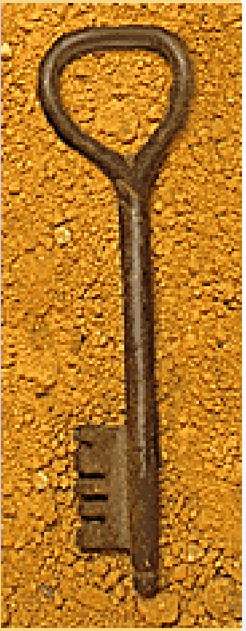
☯

- Main difference
  - ➢ Active interference from the environment



Attack Goals

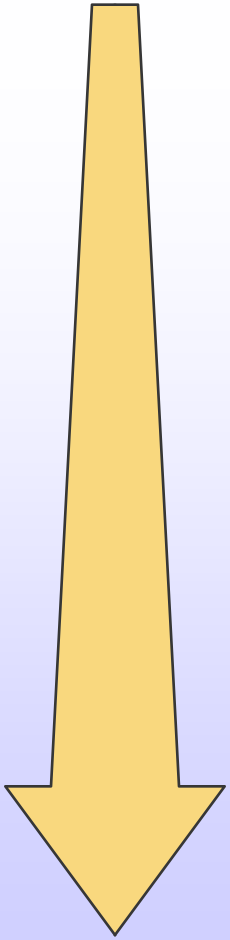
in the physical world	in the electronic world
• Publicity	
➢ Terrorism	➢ Highly contagious viruses
➢ Landing in Red Square	➢ Defacing web pages
• Fraud	
➢ Bank robbery	➢ Credit card number theft
➢ Scams	➢ On-line scams
➢ Plagiarism	➢ Intellectual property theft
• Disruption	
➢ Vandalism	➢ Wiping out data
➢ Obstruction of justice	➢ Denial of service
• Invasion of privacy	
➢ Collection of personal data	➢ Reading private files
➢ Espionage	➢ Surveillance

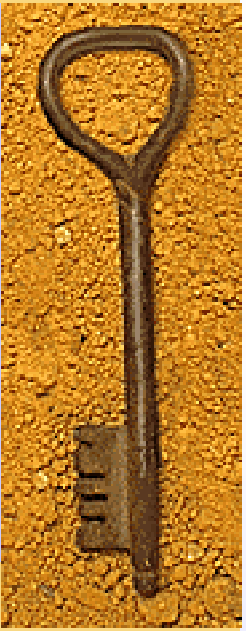


Some Threats

[Defense Science Board]

- Unintended blunders
- Hackers driven by technical challenge
- Disgruntled employees or customers
- Petty criminals
- Organized crime
- Organized terror groups
- Foreign espionage agents
- Information warfare



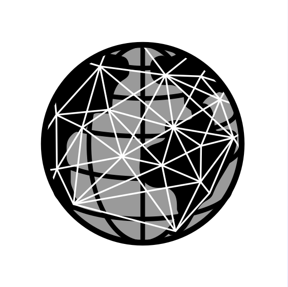


Vulnerable Systems: a Trend

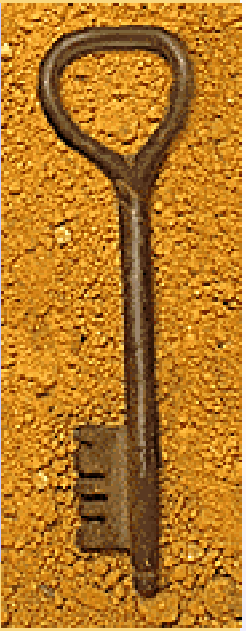
Vulnerability: a weakness that can be exploited to cause damage

Attack : a method to exploit a vulnerability

- The Internet
  - World-Wide connection
  - Distributed: no central design and control
  - Open infrastructures: modems, wireless, DHCP
  - Untrusted software: applets, downloads
  - Unsophisticated users



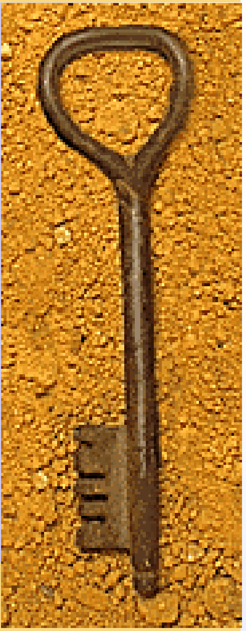
- Security costs
  - Market now, fix bugs later
  - Customers want it, but won't pay for it
- Homogeneity
  - Hardware: x86
  - OS: Windows
  - Applications: COTS



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

The Compromises of Security

- There is no absolute security!
  - Race between attackers and defenders
    - Constant innovation
    - Well-funded, capable, determined attacker succeed
- Costs
  - Relative to target's value
  - Users' inconvenience
  - Users' acceptance
- Detection
  - Rarely possible in real time
  - Works mostly for old threats
- Punishment
  - Hard at a distance
    - No international legislation
    - Poor domestic legislation (DMCA)
  - Perceived "unethical"
    - Freedom of expression
    - Intangibility

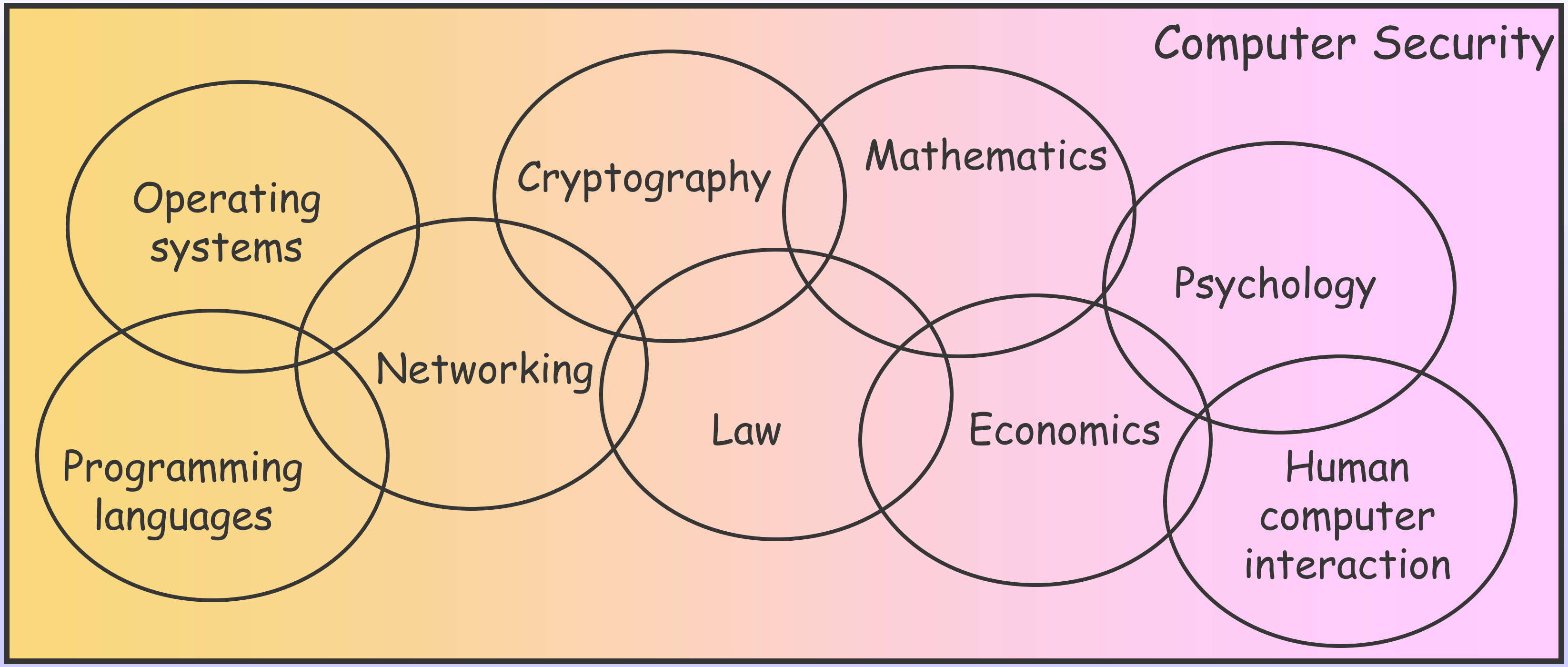


Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Is Cryptography the Solution?

Cryptography is not the same as security

- No crypto in this lecture
- 85% of all CERT advisories cannot be fixed by crypto
- 30-50% of recent security holes from buffer overflow



Computer Security

Operating systems

Cryptography

Mathematics

Psychology

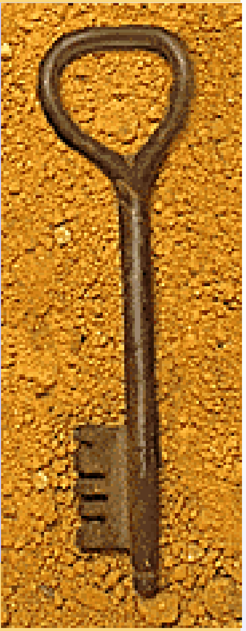
Networking

Law

Economics

Programming languages

Human computer interaction



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Policies, Mechanisms, Assurance

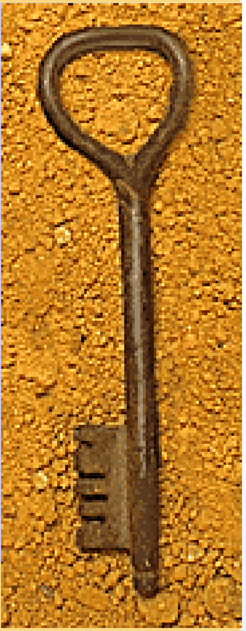
	Systems	Security
What is it supposed to do?	Specifications	Policy
How does it do it?	Implementation	*Mechanisms*
Does it really do it?	Correctness	*Assurance*

Abstraction

- Distinction between
  - Mechanisms
  - Policies

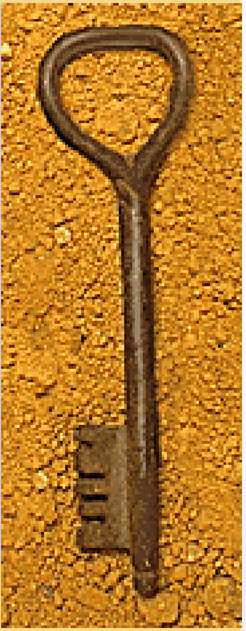
  depends on level of abstraction
- *Assurance* can sort things out
- *Attacker will not politely respect abstraction layers*

Some Security Properties

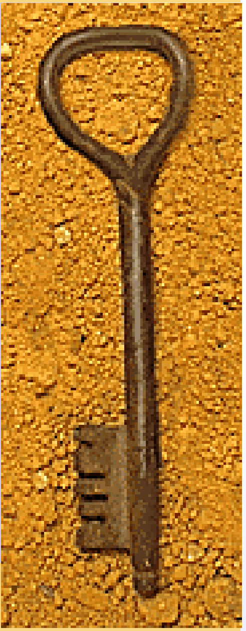


- Integrity: no improper modification
  - Authenticity: integrity of source
  - Non-repudiation: integrity of commitments
  - Accountability: integrity of responsibility
- Secrecy: no improper disclosure
  - Privacy: secrecy of personal data
  - Anonymity: unlinkable secrecy of identity
  - Pseudonimity: linkable secrecy of identity
- Availability: no improper denial of service

Security Policies

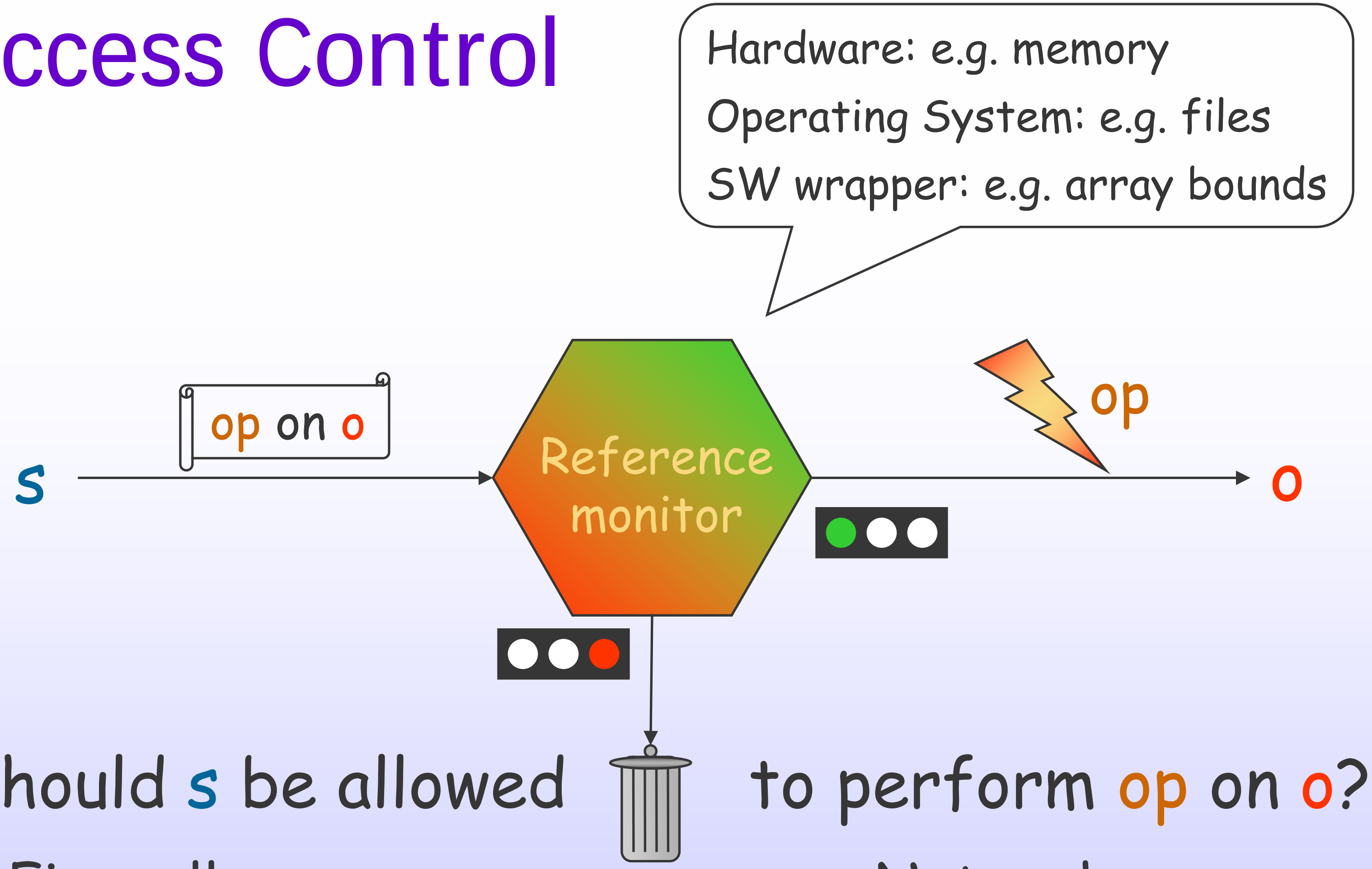


- Collection of security properties
  - Sometimes conflicting
- Application specific
  - E.g., bank:
    - Authenticity of clients at ATM and web
    - Non-repudiation of transactions
    - Integrity of the books
    - Secrecy of client and internal data
    - Availability of alarm system
    - Exclusivity of duties (avoid conflicts of interest)
    - Dual control of sensitive transactions



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Access Control

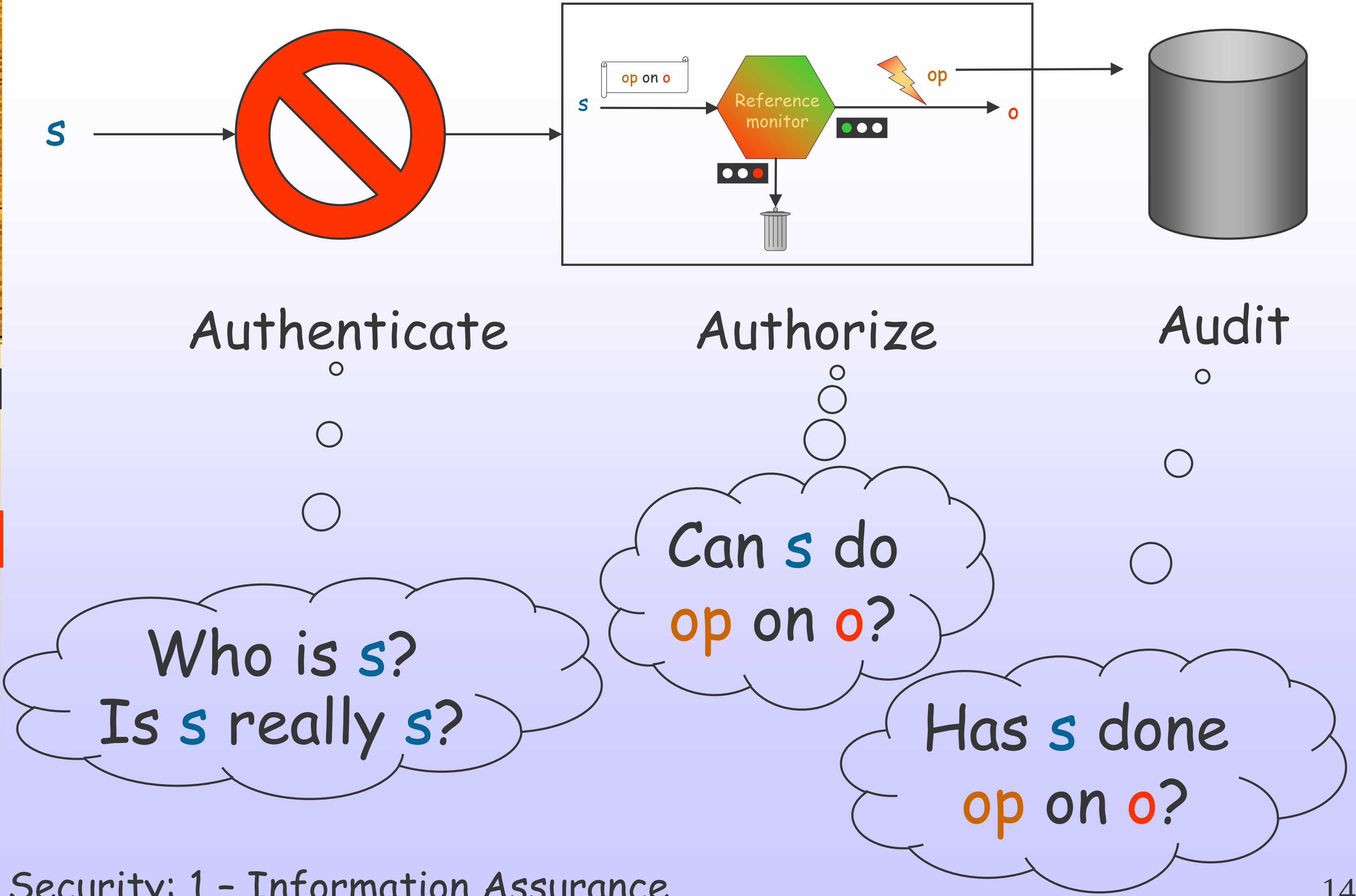


Should s be allowed to perform op on o?

- Firewalls
- Applications
- Middleware
- File system
- Memory management

- Network
- OS
- Hardware

A Bigger Picture: Lampson’s Rule



Authenticate

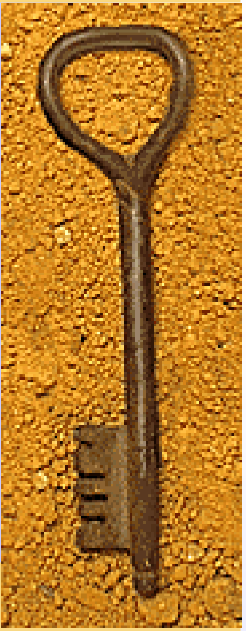
Authorize

Audit

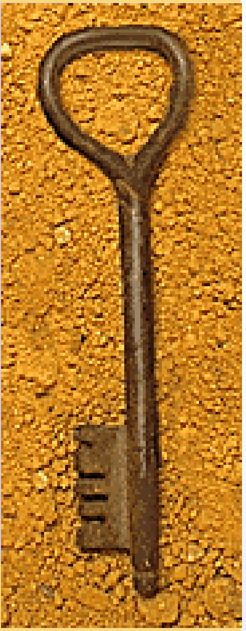
Who is s?
Is s really s?

Can s do
op on o?

Has s done
op on o?



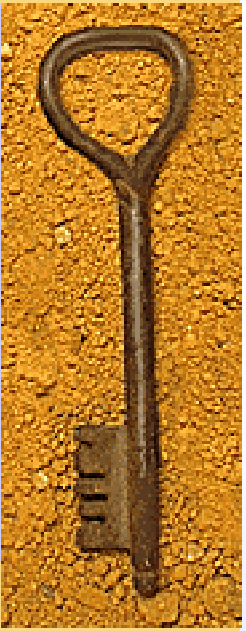
Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code



Governing Principles

- Complete mediation
  - Every access to every object is checked
- Least privilege
  - Do not grant a subject more rights than he needs
- Separation of privileges
  - Avoid conflicts of interests
- Redundancy
  - Diversity of mechanisms
  - Multiple lines of defense
- Non-intrusiveness
  - User acceptance

These can be conflicting requirements



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Mechanisms and Assurance

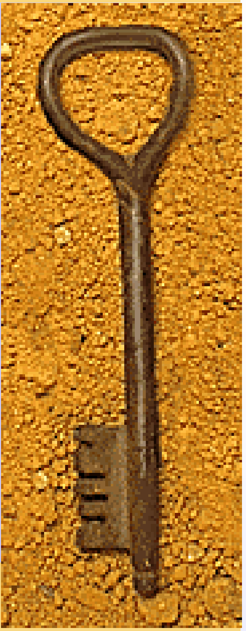
3 main mechanisms

- Discretionary AC
  - Access right belongs to owner
  - Rights can be modified and delegated
- Mandatory AC
  - Access right belongs to resource
  - Rights administered off-band
- Role-Based AC
  - In between

Assurance models

- Mostly dedicated access logics

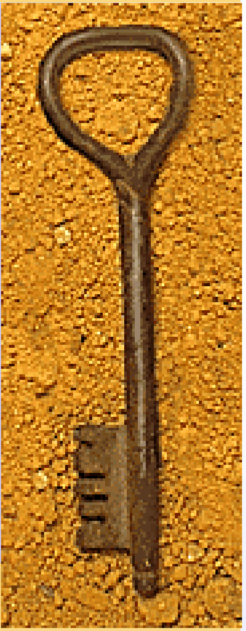
Discretionary Access Control



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

- Subjects
  - Principals who want access
    - Users
    - Programs
    - IP addresses
- Objects
  - Data
  - Resources
- Access rights
  - Operations subjects can perform on objects
  - Privileges
  - Also administrative rights
    - Modify privileges
    - Delegation

Explicit access rules in the form of principal-resource-operation triples



Security

Policies

DAC

MAC

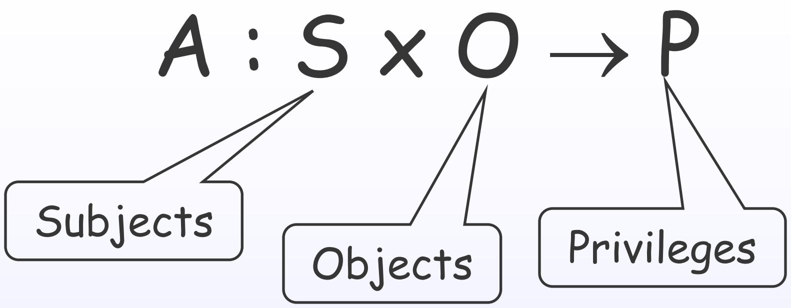
Covert ch.

Stego

Safety

Mobile code

Access Matrix

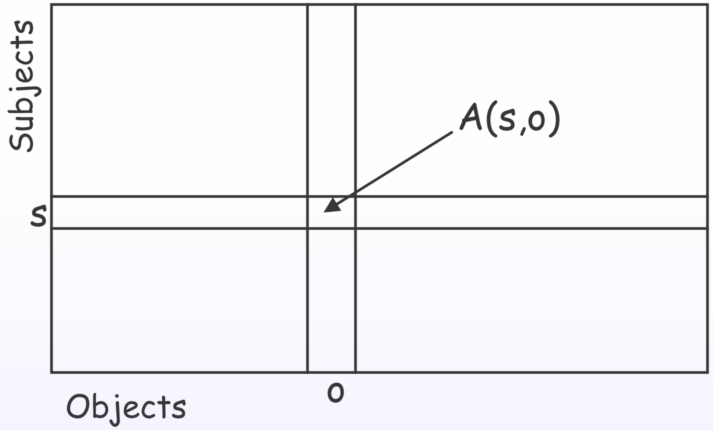


$A : S \times O \rightarrow P$

Subjects

Objects

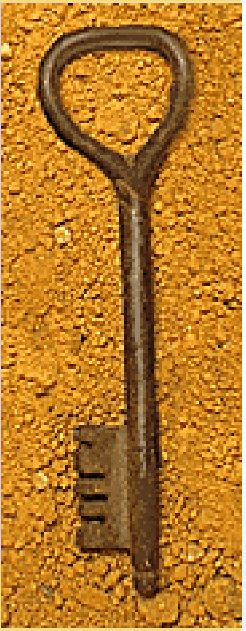
Privileges



Matrix is generally large and sparse

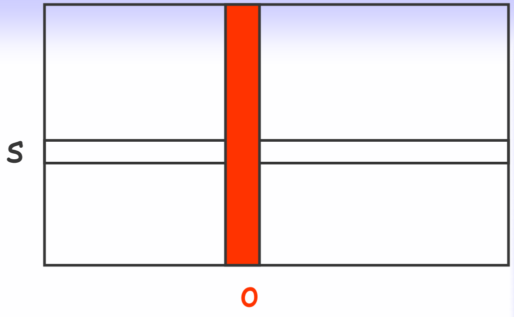
- Store by column: access control lists
  - Files
- Store by row: capability lists
  - Applets, tickets
- Store non-null triples: authorization tables
  - DBMSs

More on this



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Access Control Lists



Implement access matrix by columns

- Lists s's who can access o and for what
- Maintained close to objects
  - E.g., bit permissions of files

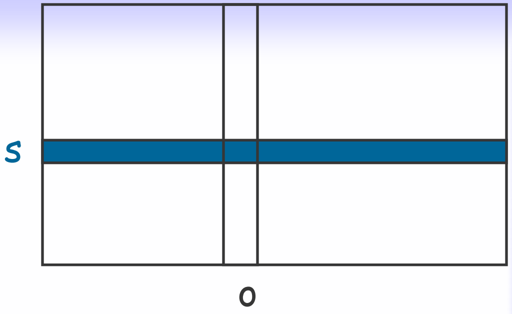
+ Compact
+ Easy per-object review

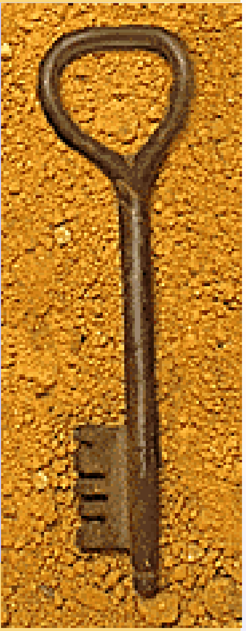
– Revoking a subject can be hard

– Require authentication of subjects

- Useful when
  - Many objects
  - Few subjects or simple grouping

Capability Lists





Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Implement access matrix by rows

- Lists o's that s can access and for what
- Maintained at entry point for s
  - ➢ E.g., when applet is downloaded

+ Capabilities controlled by s

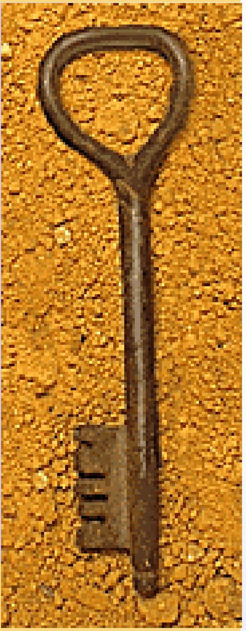
+ Easy to forward and delegate

– Revoking a capability can be hard

– Requires unforgeability of tickets

- Useful when
  - ➢ Delegation is necessary
  - ➢ Holders of privileges are hard to anticipate

Implementing Capabilities

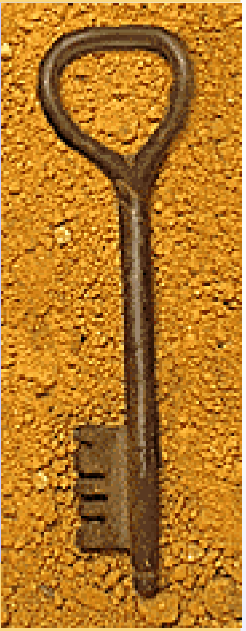


Sophistication to prevent forgery

- Stored in protected address space
- Special tags with hardware support
- As references in typed languages
- Encrypted
- Cryptographic certificate

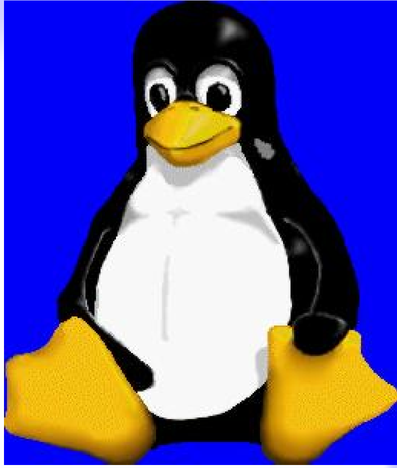
ACLs and capability lists are often combined

- Diversity of mechanism
- Get the best of both worlds



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

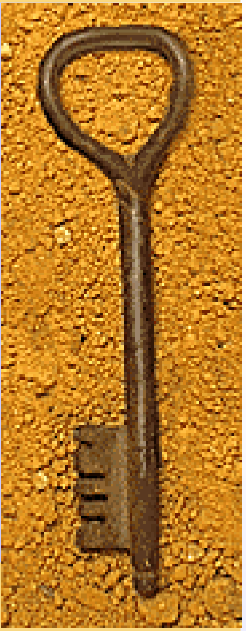
AC in Unix Systems



- Files protected by ACLs
  - Subjects are users (or `root`)
  - File has an owner and a group
  - Operations
    - read, write, execute
    - For user, group, world
    - 9 bits: e.g., `rw-r--r–`
- Rudimentary capability lists
  - `/etc/passwd, /etc/group, /etc/host.deny, /etc/host.allow`
- Programs
  - Run in protected memory
  - With privileges of caller (unless `suid/sgid` set)

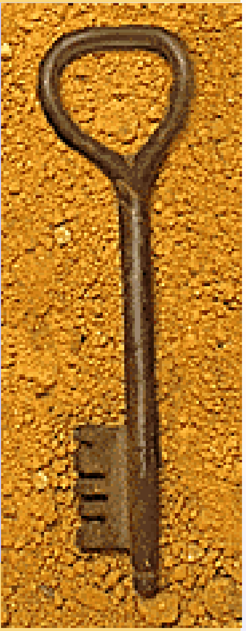
AC in Microsoft Products

- DOS: No AC
  - Single user system
- Windows 95, 98
  - Basic AC
  - No protected memory!
- Windows NT
  - Under the hood, it is Unix
  - But richer:
    - Users organized into domains
    - Easy to obey the principle of least privilege
- Windows 2000, ME, XP
  - Even richer



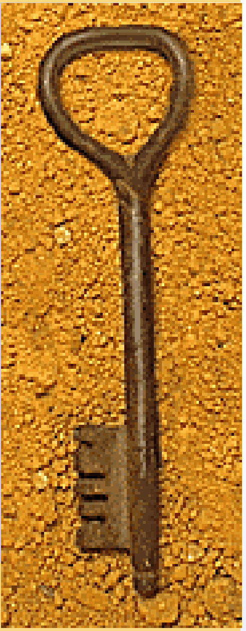
Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

AFS – Andrew File System



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

- Meta file system
  - Transparently connects FSs or NFSs
  - E.g.: $\texttt{/afs/}\underbrace{\texttt{cs.cmu.edu}}_{\text{cell}}\underbrace{\texttt{/user/iliano/.plan}}_{\text{file within cell}}$
- Elaborate ACL mechanism (`rlidwk`)for
  - Directories: `l`ist, `i`nsert, `d`elete, `a`dminister
  - Files: `rwk` similar to Unix `rwx` (yet different)
  - Subjects:
    - Users, possibly remote
    - Groups (controlled by users)
    - Users within groups



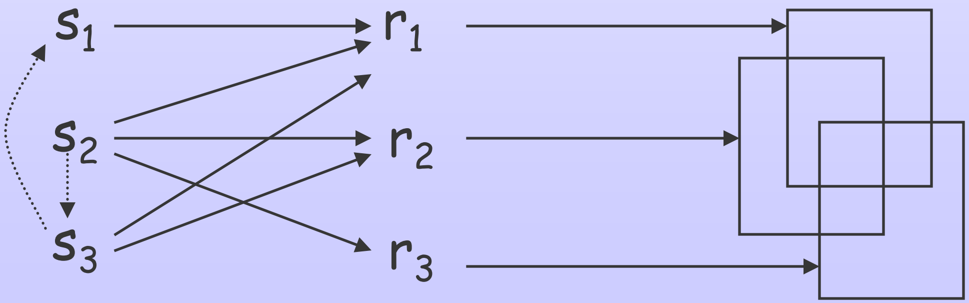
Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Role-Based Access Control

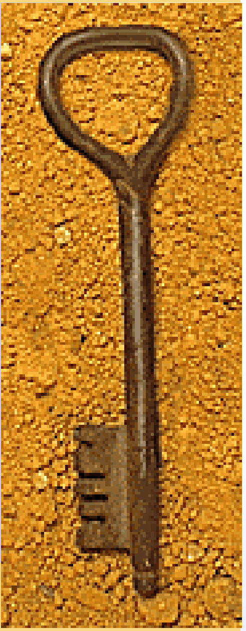
- Subject may need different rights for different activities
  - Tom as system administrator (`root`)
  - Tom as user `tommy`
  - Tom as consultant for bank A
  - Tom as consultant for company B

Roles

- Access to users mediated by roles
  - s in role r has all the privileges of r



Advantages of RBAC

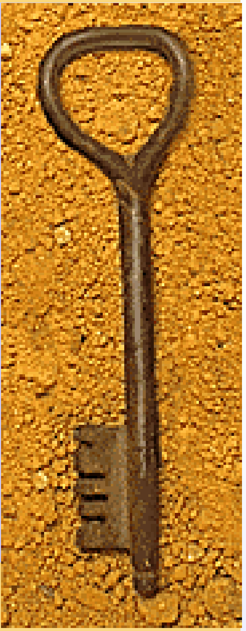


Supports

- Least privilege
- Easy revocation
- Separation of duty
- Role hierarchies
- (Partial) anonymity

Note

- Group: set of users
- Role: set of privileges

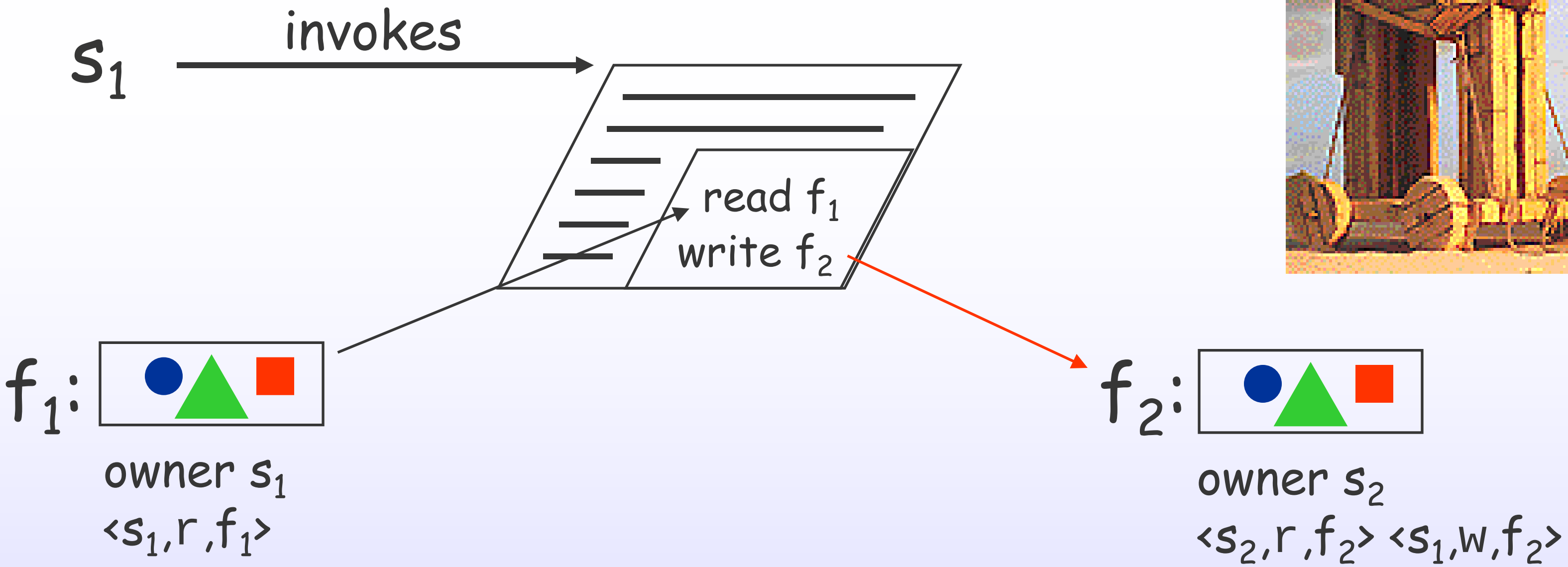


Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Limitations of Discretionary AC

- Vulnerable to Trojan Horses
  - Rogue code acquires privileges through
    - Carelessness/Ignorance of user
    - Delegation mechanism
    - Fact that only direct accesses are regulated
  - Code executed by trusted user is trusted
    - Source of all virus attacks
- No control over released information
  - Access is attribute of subject
  - How about making it attribute of object?
- Leaves security policy to each subject
  - Intrinsically limited to saving subjects from themselves

Trojan Horses



s_1 invokes

read f_1
write f_2

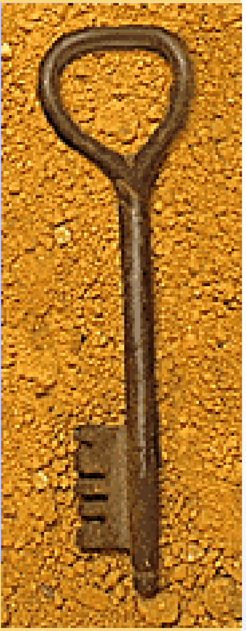
f_1: ● ▲ ■

owner s_1
$<s_1,r,f_1>$

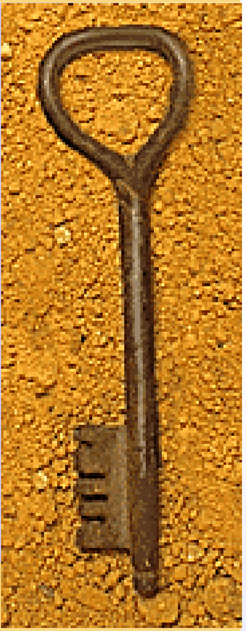
f_2: ● ▲ ■

owner s_2
$<s_2,r,f_2>$ $<s_1,w,f_2>$

- s_2 has gained access to s_1's data
  - s_1 is not aware
  - Discretionary AC policy respected
    - Computer virus downloaded from the net?
    - Network worm that exploited vulnerability?



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Mandatory Access Control

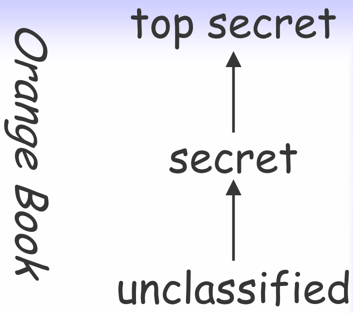
Access decided only based on security level of subject w.r.t. security level of object

Distinguish

- User
  - Trusted, possibly
- Subject
  - Process operating on behalf of the user
  - Untrusted

- Assign security levels to subjects and objects
- System enforces AC policy
  - Users has no control on security level
  - No Trojan horses
- 2 flavors
  - Secrecy based → address information leakage
  - Integrity based → prevent corruption of information

Bell-La Padula Model

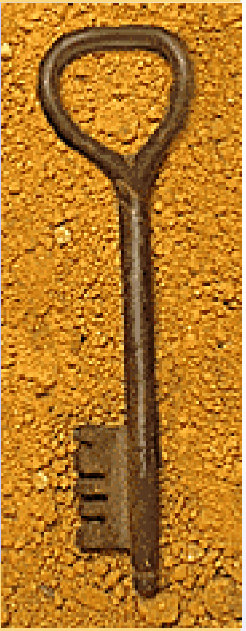


- Classes represent secrecy levels
  - Users' level: clearance
  - Objects' level: sensitivity of information
  - Levels may form a lattice

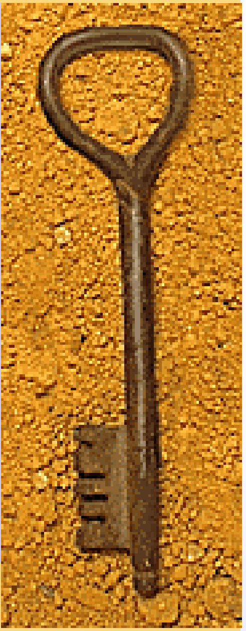
- No write-down
- No read-up

Prevent Trojan Horses

Enforce secrecy

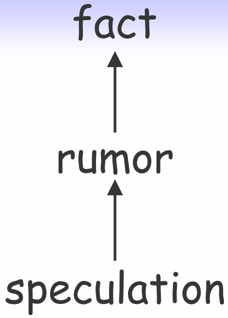


Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Biba Model



- Classes represent integrity levels
  - Users' level: trustworthiness
  - Objects' level: trust in validity of information
  - No write-up
  - No read-down

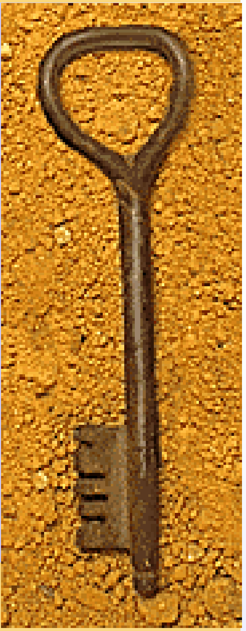
May corrupt good data

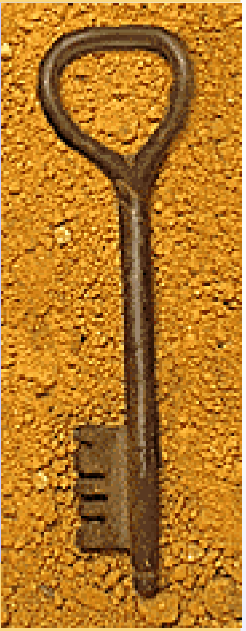
Data may be invalid

- Dual to Bell-La Padula …
- … but not exclusive

Limitations of Mandatory AC

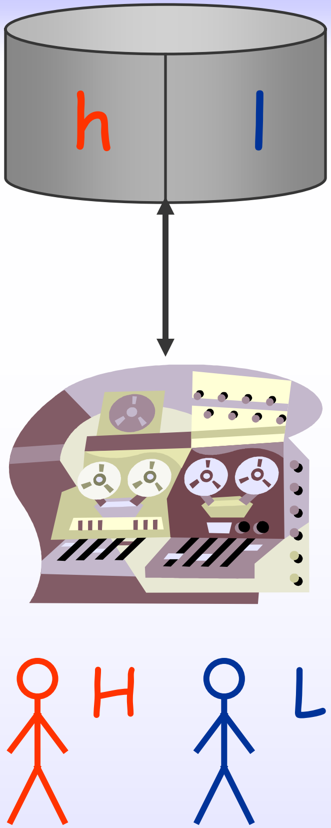
- Vulnerable to *covert channels*
  - Unauthorized downgrading of information
    - High-level user H transmits value of high-level variable h to low level user L
- Popular during era of mainframes, …
  - Computers were expensive
- … but now mostly abandoned
  - Physical separation of sensitive information
    - Reside on independent networks
    - Share at most keyboard and monitor [McLean]



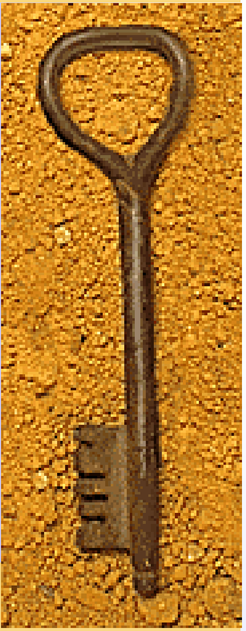


Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

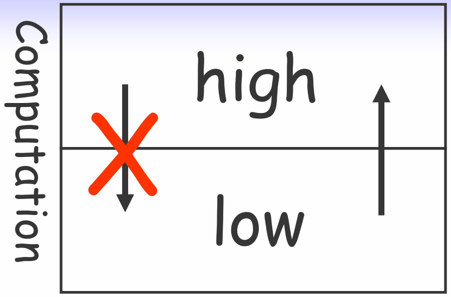
Covert Channels



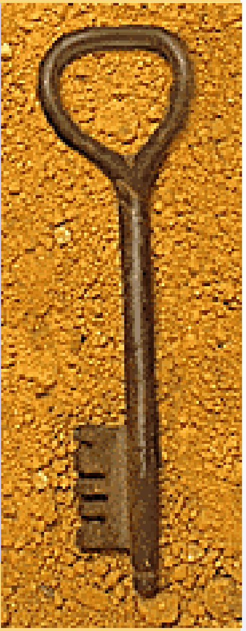
- H: if h then 1 else 0
- H: if h then fill disk
- L: try to write file
- H: if h then heavy computation
- Observed increased pizza delivery when Pentagon on high alert
- Extensions deal with
  - Infinite computation
  - Probabilities
  - Non-determinism, …



Non-Interference

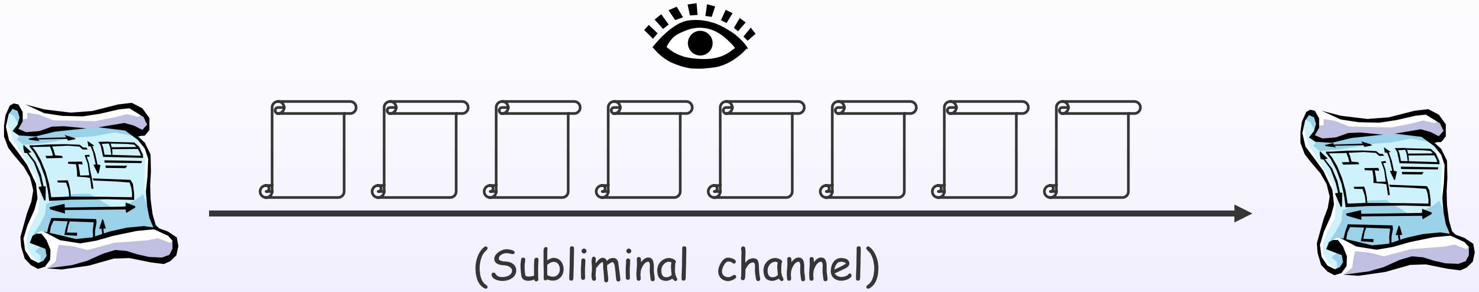


- Restrictions on the *flow of information*
- Effect of H computation not visible to L
  - Value of accessible data
  - Side-effects of H computation
  - Formal definition(s) in process algebra
- Shift of perspective
  - Era of mainframes is gone
  - Physical separation of sensitive information
  - New issues
    - Mobile code



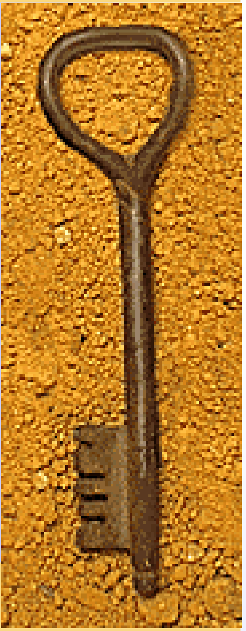
Stegonography

Transmit information undetected



Note

- Cryptography
  - information is hidden but detectable
- Covert channel
  - usually minimal bandwidth



Security

Policies

DAC

MAC

Covert ch.

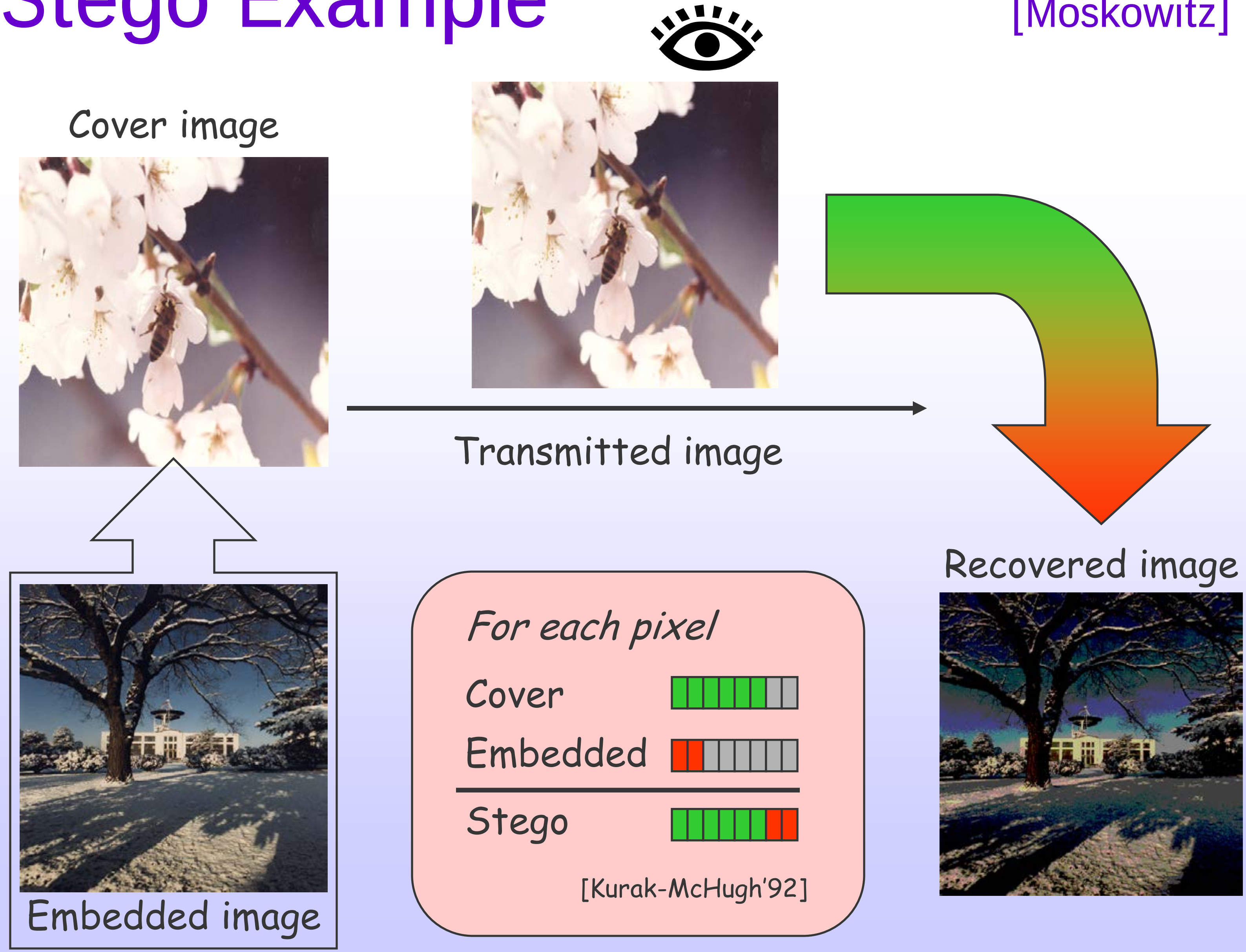
Stego

Safety

Mobile code

Stego Example

[Moskowitz]



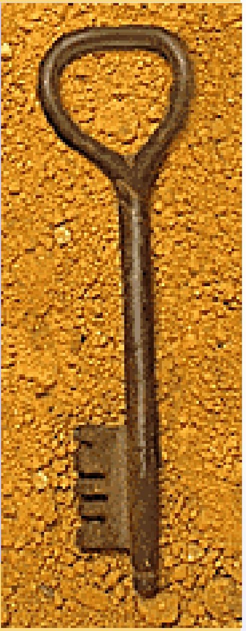
For each pixel

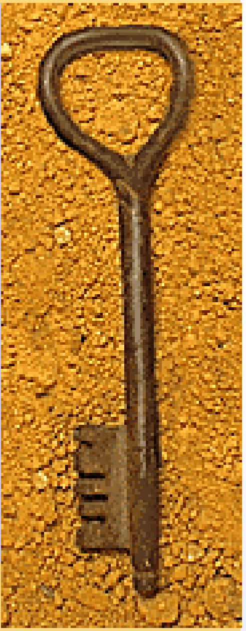
Cover	🟩🟩🟩🟩🟩🟩⬜⬜
Embedded	🟥🟥⬜⬜⬜⬜⬜⬜
Stego	🟩🟩🟩🟩🟩🟩🟥🟥

[Kurak-McHugh'92]

Programs

- … do access resources
- Different from other principals
  - Call chains
    - E.g. applet on browser on OS
  - Rights determined by several principals
    - Writer
    - Installer
    - Owner
    - Principals involved in call chain
  - Failure of access mediation can be truly catastrophic





What can go wrong?

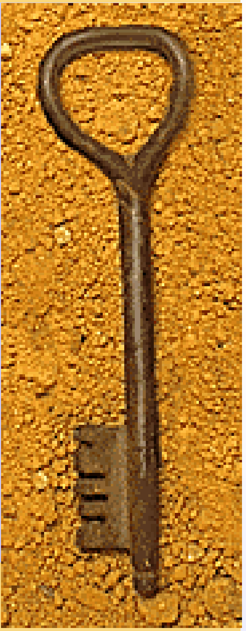
- Incorrect access control set up
  - Infrequent
- Bugs in underlying operations
  - Buggy Trusted Computing Base

HW, SW and set-up info on which the security of the system depends

- Dangerous code is executed
  - Visual Basic scripts in incoming email
    - Especially if title is nice ("*I love you*")
  - Claims to be the right device driver

Educate users
Safer languages for mobile code
Additional in-line reference monitors
Finer delegation of privileges
Signed code
Virus scanners

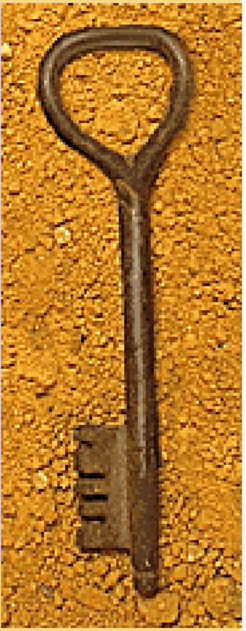
- Access control is circumvented
  - Easiest way to steal a credit card number is to ask for it



Formal Correctness

- Formal specification
  - All behaviors are covered … and provably so
  - Specification is mathematical objects
- Correctness is mathematically proved
  - Operating system components
  - Cryptographic primitives
  - Security protocols
  - Language run-time
- Still relatively rare
  - Expensive and time-consuming
  - Require expertise
  - Not always convincing
  - Not widespread … yet

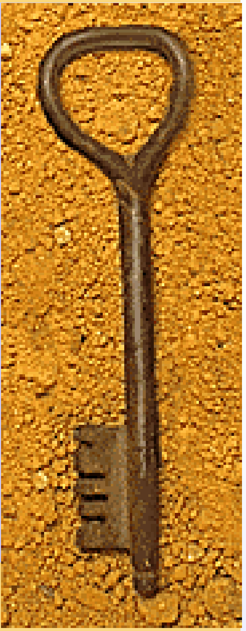
Trusted Computing Base - TCB



HW, SW and setup information on which the security of the system depends

- Should be right
  - Defined precisely
  - Small and simple
    - Windows keeps even printer drivers in kernel!
      - Not trustworthy
  - Specified
  - Tested
  - Verified

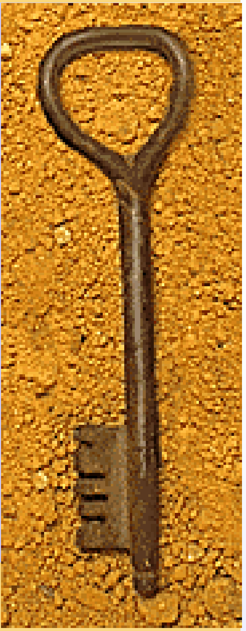
Memory Accesses



- Managed by the OS with HW support
- Till recently: Wild West
- Now
  - Programs confined in protected memory separate from that of other programs or OS
    - No direct access
    - Access only through interfaces
  - Does not prevent program from corrupting its own memory
  - One especially dangerous interface
    - The execution stack

Buffer Overflow

[Martín Abadi]

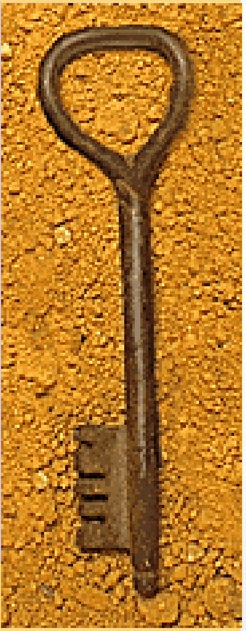


- The tail of a long argument smashes the return address on the stack
- Upon a `return`, control jumps to malicious code

Avoiding buffer overflow

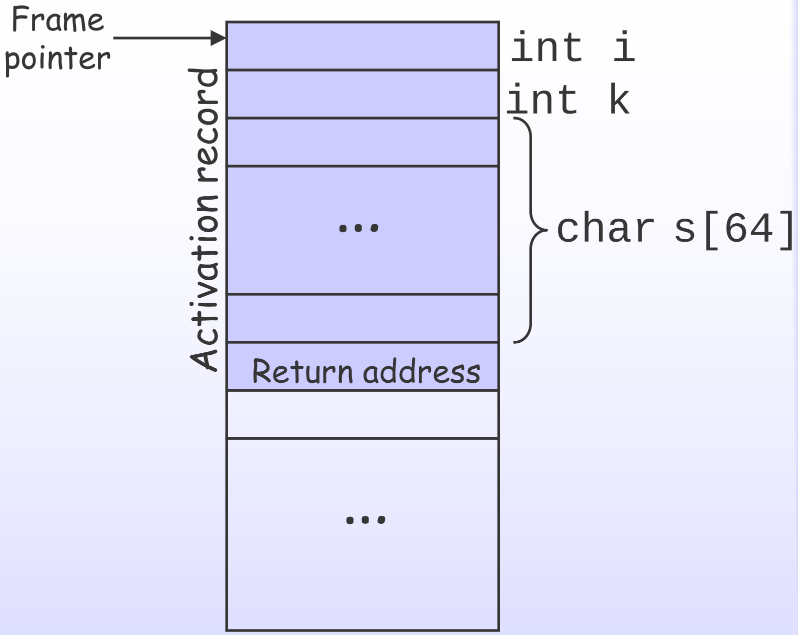
- Separate code and data segments
  - Disallowed code modification
  - Disallowed jumps to data
- Static analysis
- Safe libraries (wrappers)
- Safe programming languages

Example of Buffer Overflow

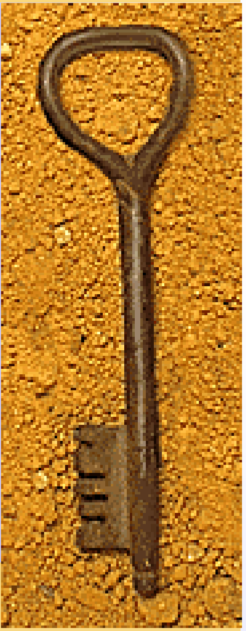


```
gets(s);
```

- Input more than 64 bytes
    - gets just writes it down the stack
- *Bytes 65-68*
    - address of byte 69 on the stack
- *Byte 69*
    - Instructions of malicious code

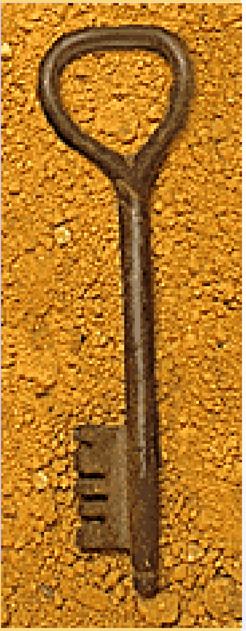


Safe Programs



- Do not crash
  - Even without confinement in protected memory
    - May share address space
  - Cannot corrupt even their own memory
- Cannot access arbitrary addresses
  - Otherwise could easily crash the system
- Can be written in any language, but
  - Some languages allow writing only safe programs
    - Pure Lisp, pure Java, ML
  - Some languages isolate potentially unsafe code
    - Modula-3, Java with native methods, C#
  - Some languages are hopeless
    - Assembly languages, C

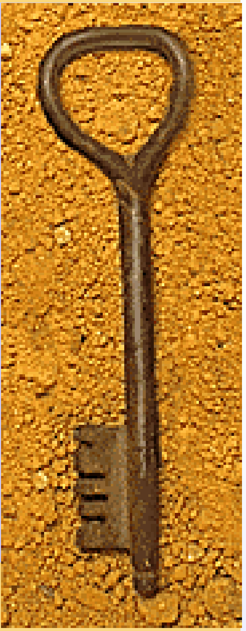
Safe Programming Languages



Allow writing only safe programs

- Front-line of programming language research
  - Precise definitions
    - Either provably safe, or
    - People are refining definitions and proof techniques
  - Tractable theory with sophisticated methods
    - Safety usually ensured by type checking
  - Powerful static analysis techniques
    - Byte-code verification
    - Proof-carrying code
    - Typed assembly language
    - Buffer overflow detection for C
- Word is starting to get out (Java)

What about High-Level Security?



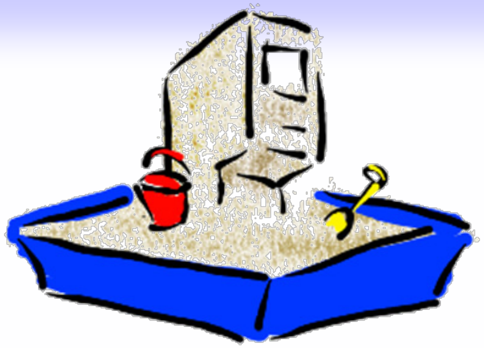
Few programming languages designed for it

- Language descriptions rarely specify security
- Implementations not always secure

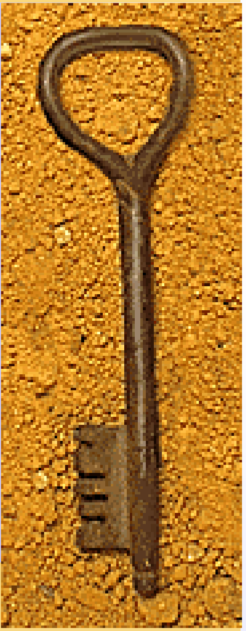
Exception: Java

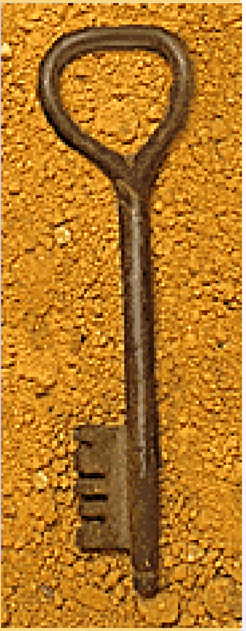
Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Java Security



- Taming mobile code
  - Security manager associated with code at load-time
  - Serves as reference monitor for requests from code
- Java 1.0
  - Local code has full access
  - Remote code confined in sandbox
- Java 1.1
  - Local, trusted and signed code has full access
  - Other remote code confined to sandbox
- Java 1.2
  - Configurable fine-grained security policy for all code
  - Default is sandbox





Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

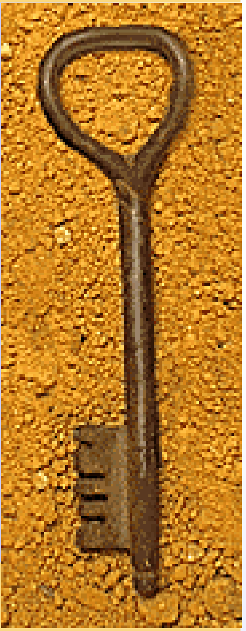
Stack Inspection

- Java run-time components
  - Different provenience → more or less trusted
  - Different permissions for protected resources
  - May call each other
- To access resource
  - Whole execution thread must have permission
  - But … trusted code can take responsibility
    - `BeginPrivilege` overrides inspection of callers
- X Windows attacked by confusing font manager
  - No stack inspection

g calls f on directory d:
→ f and g must have permission to look at d

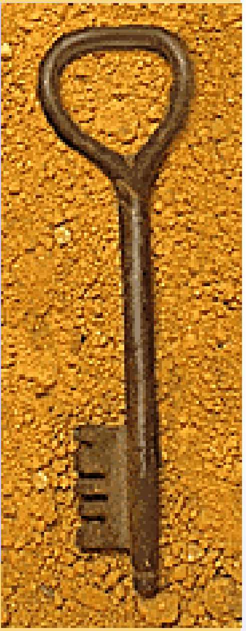
g calls f on public web
f updates log file with each query
f looks in cache and temporary files:
→ only f needs permission to touch log, cache and temporary files
→ f should call `BeginPrivilege`

Readings



- Dieter Gollmann, *Computer Security*, 1999
- Pierangela Samarati, *Access Control: Policies, Models, Architectures, and Mechanisms*, 2000
- M. Abadi, B. Lampson, M. Burrows and E. Wobber, *Authentication in Distributed Systems: Theory and Practice*, 1992
- John McLean, Security Models, 1994
- Luca Cardelli, *Type Systems*, 1997
- D. Wagner, J. Foster, E. Brewer and A. Aiken, A First Step towards Automated Detection of Buffer Overrun Vulnerabilities, 2000
- G. Necula and P. Lee, *Research on Proof-Carrying Code for Untrusted Code Security*, 1997
- Li Gong, *Inside Java 2 Platform Security*, 1999

Exercises for Lecture 1

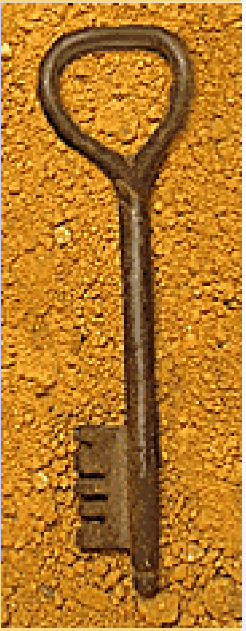


- Write a plausible security policy for a medical network consisting of doctors, hospitals, emergency rooms and insurances
- Group exercise: design a covert channel and try it on a word I'll give one of you next time
- Does an unsafe program always crash?
- What operations make C unsafe?
  - ➢ Exploit them to write a program that crashes

Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code

Next …

- Elements of Cryptography



Security
Policies
DAC
MAC
Covert ch.
Stego
Safety
Mobile code