Lecture 19: Information Flow

CS 5430 4/9/2018

Where we were…

- **Authentication:** mechanisms that bind principals to actions

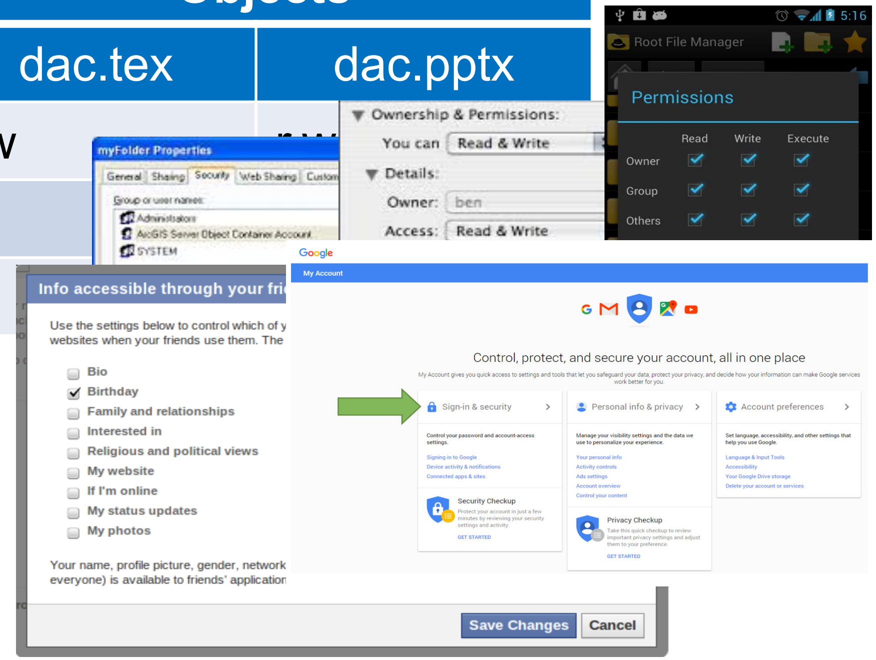
- **Authorization:** mechanisms that govern whether actions are permitted

- **Audit:** mechanisms that record and review actions

Access Control Policy

- An **access control policy** specifies which of the **operations** associated with any given **object** each **subject** is authorized to perform
- Expressed as a relation *Auth*:

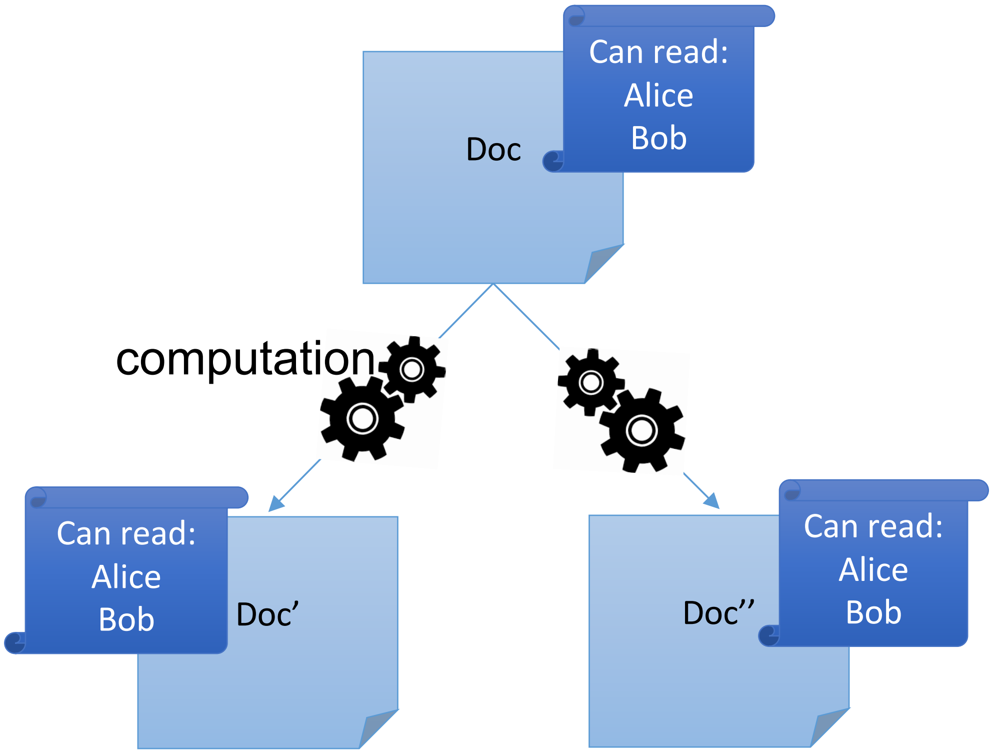
Auth		Objects	
		dac.tex	dac.pptx
subject	ebirrell	r,w	r,w
	clarkson	r	
	student		



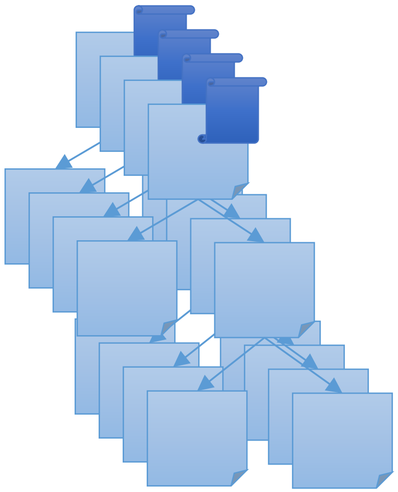
Who defines Policies?

- Discretionary access control (DAC)
  - **Philosophy:** users have the *discretion* to specify policy themselves
  - Commonly, information belongs to the **owner** of object
  - Access control lists, privilege lists, capabilities
- Mandatory access control (MAC)
  - **Philosophy:** central authority *mandates* policy
  - Information belongs to the authority, not to the individual users
  - MLS and BLP, Chinese wall, Clark-Wilson, etc.

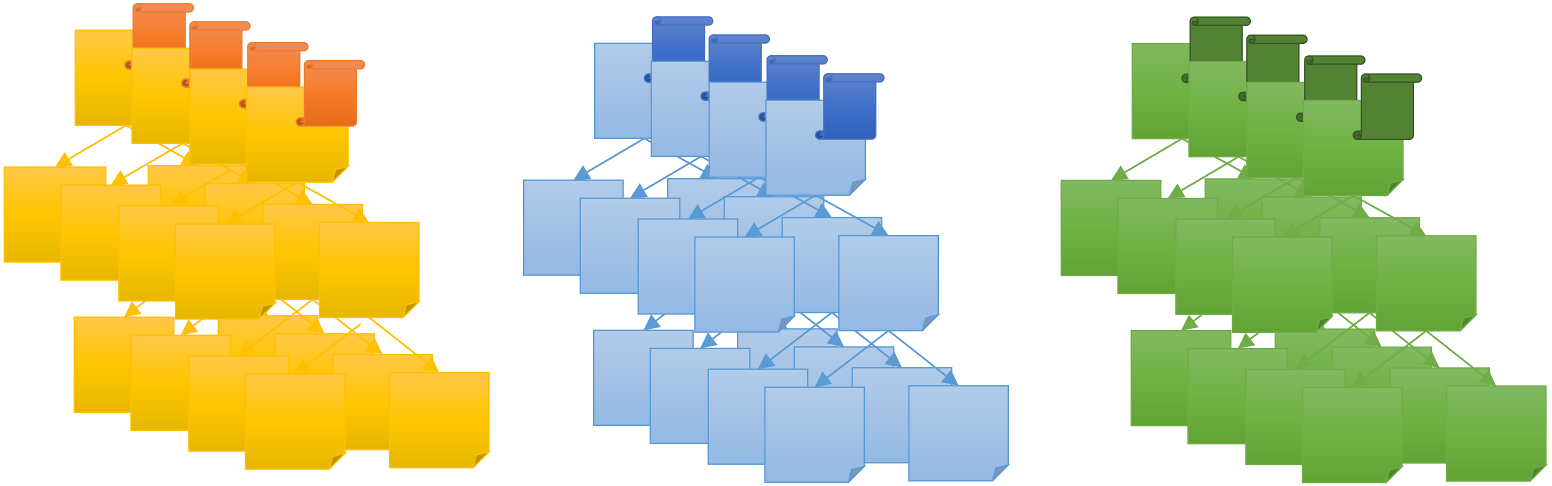
Access control for computed data



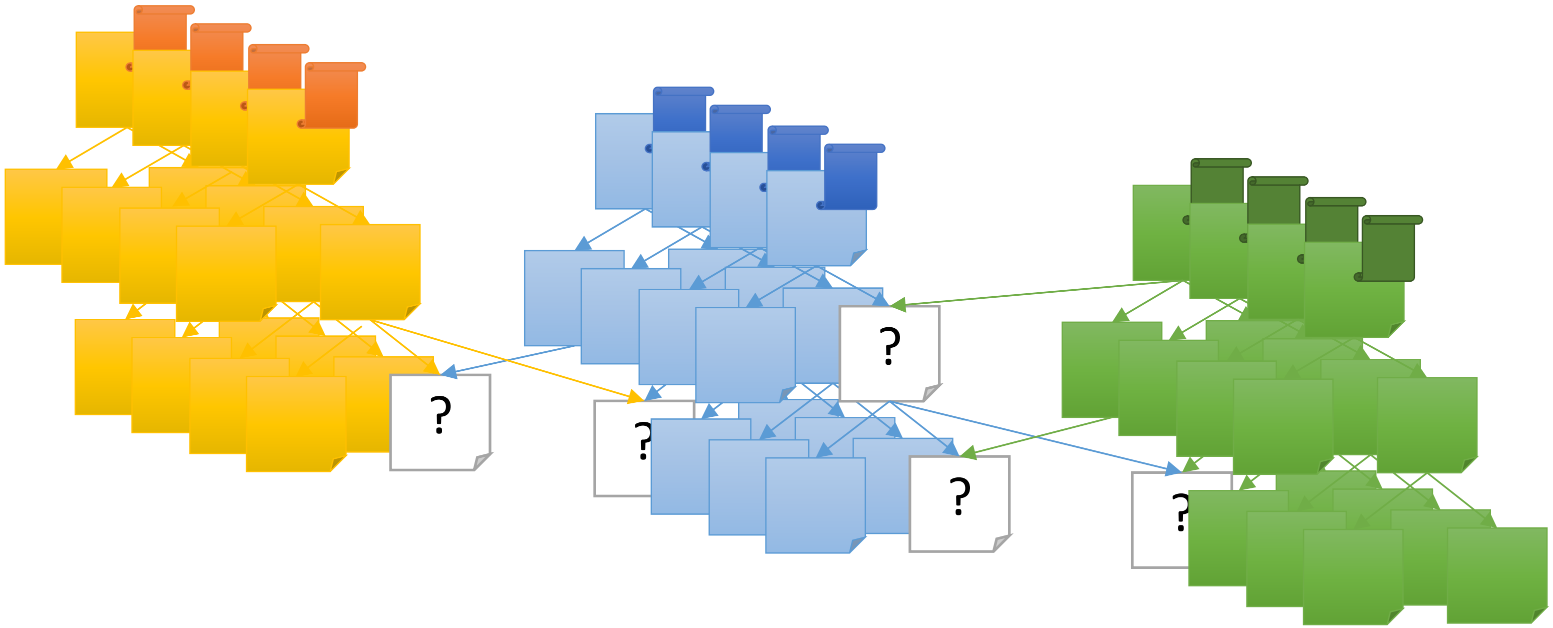
Scaling to many pieces of data…



Scaling to many users…



Scaling to many interactions…



Information Flows between Principals

- Channel: means to communicate information
- Storage channel: written by one program and read by another
- Legitimate channel: intended for communication between programs
- Covert channel: not intended for information transfer yet exploitable for that purpose

Sometimes, we really want to restrict access to information

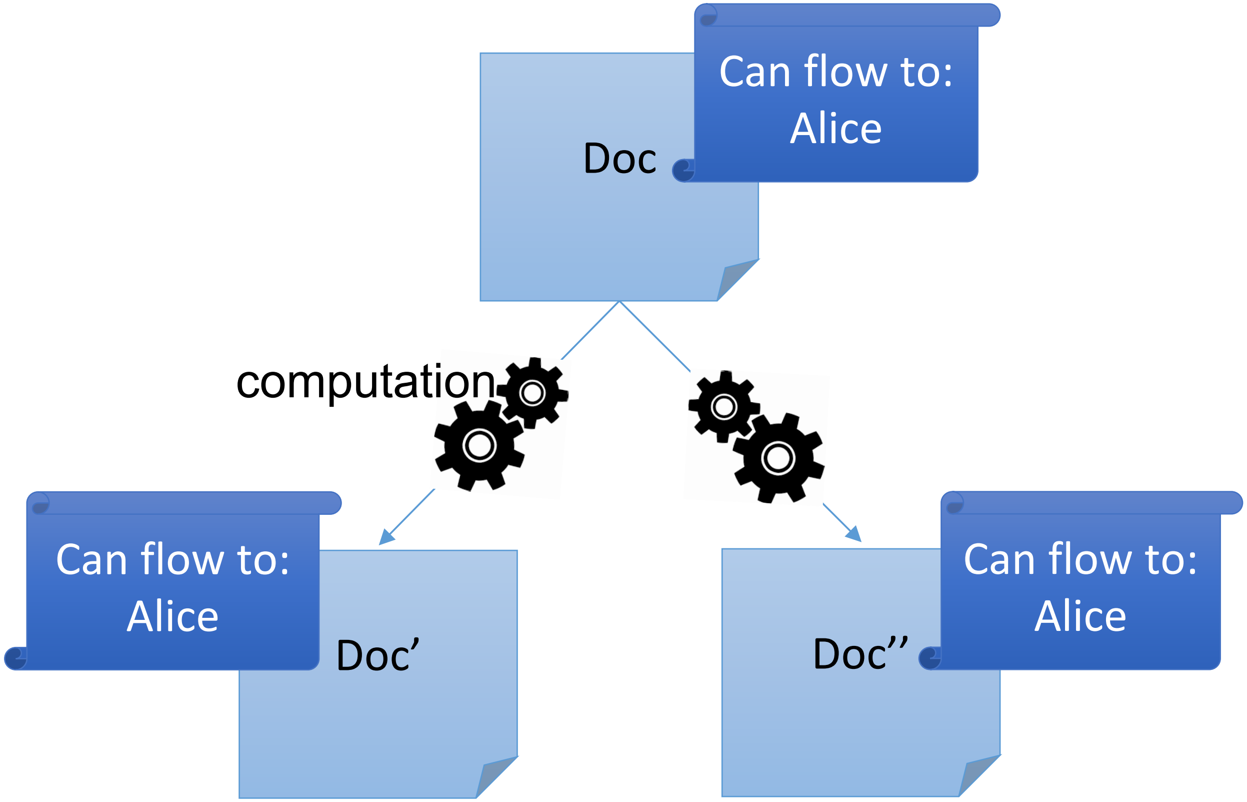
Information Flow (IF) Policies

- Focus on **information** not objects
- An IF policy specifies **restrictions** on the associated data, and on all its derived data.
- IF policy for confidentiality:
  - Value v *and all its derived values* are allowed to be read only by Alice

Different from the access control policy:
Value v is allowed to be read at most by Alice.

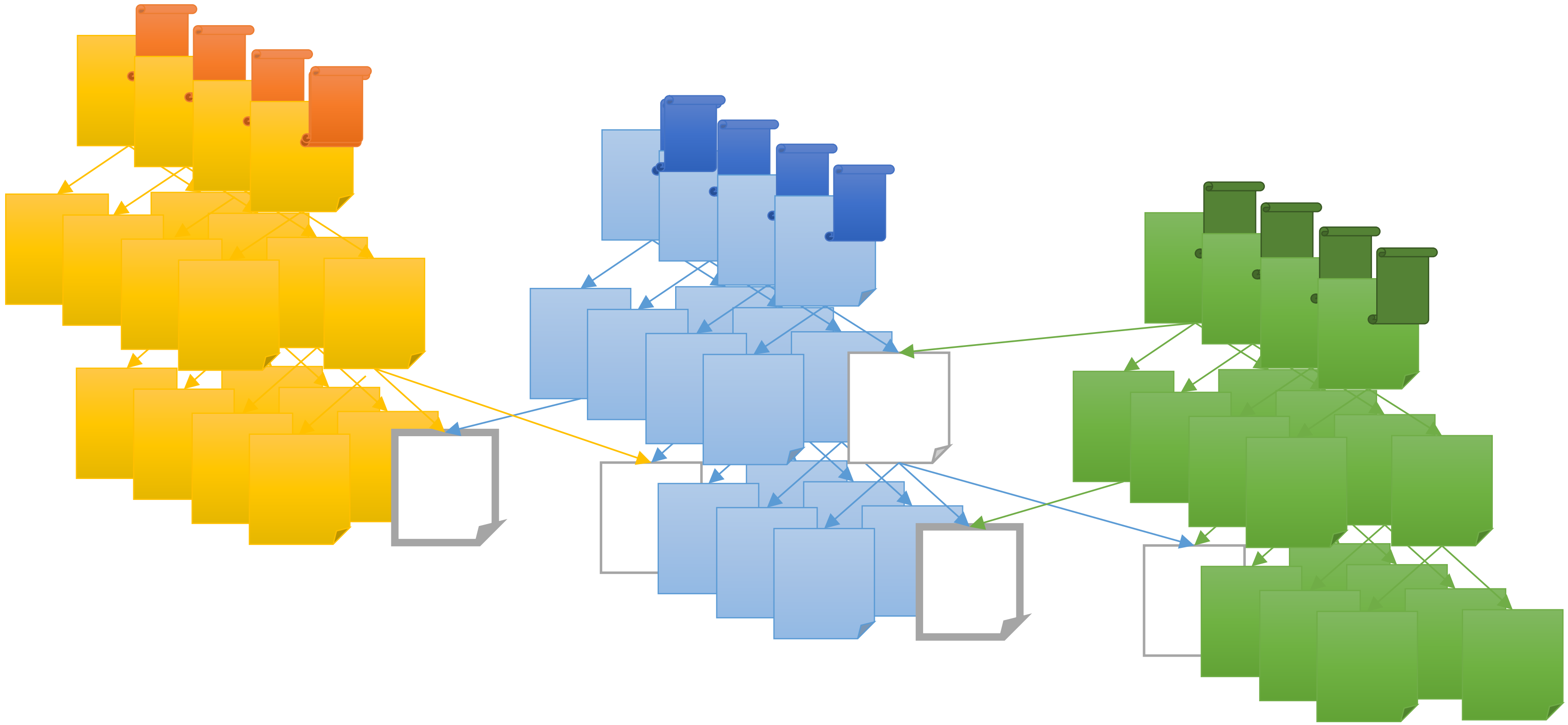
- The enforcement mechanism **automatically** deduces the restrictions for derived data.

Information flow policies

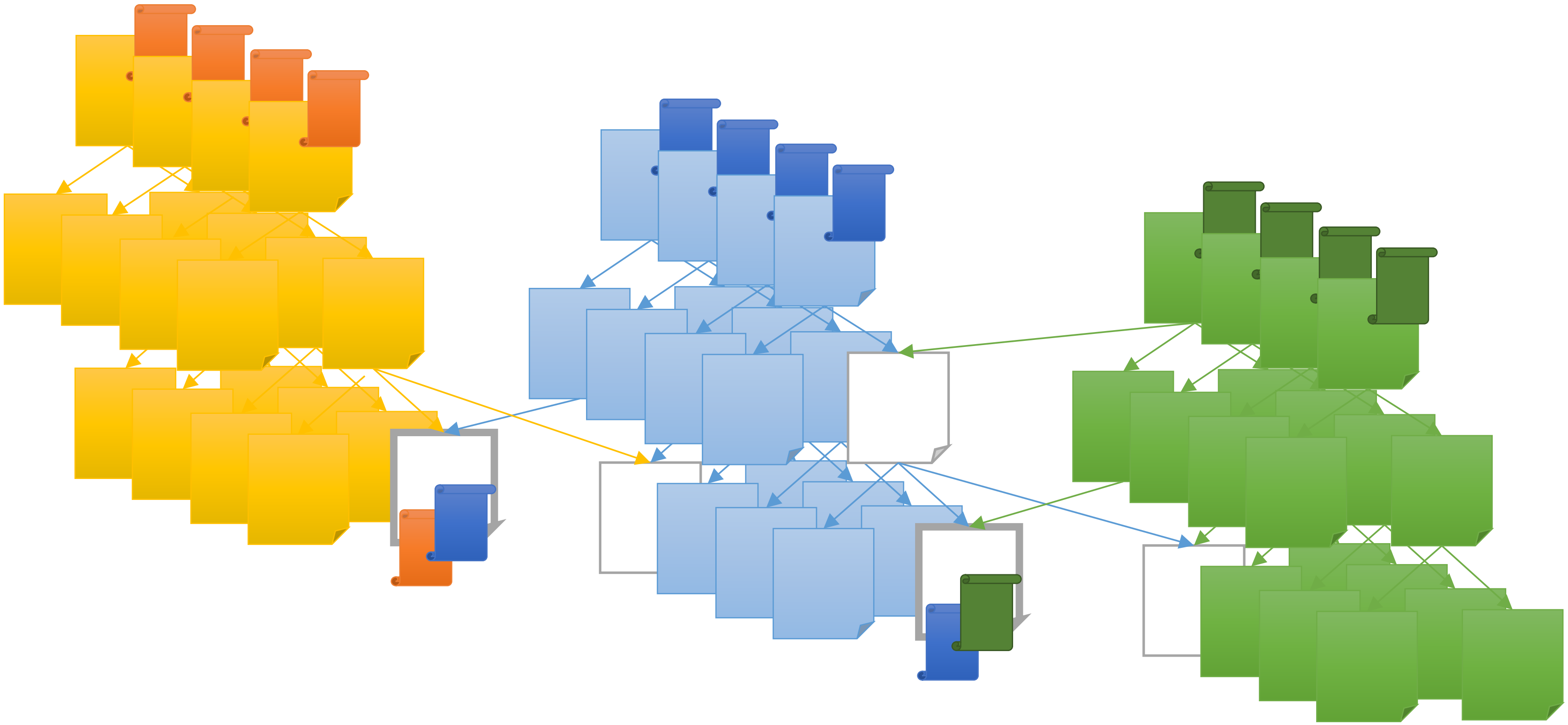


Automatic deduction of policies!

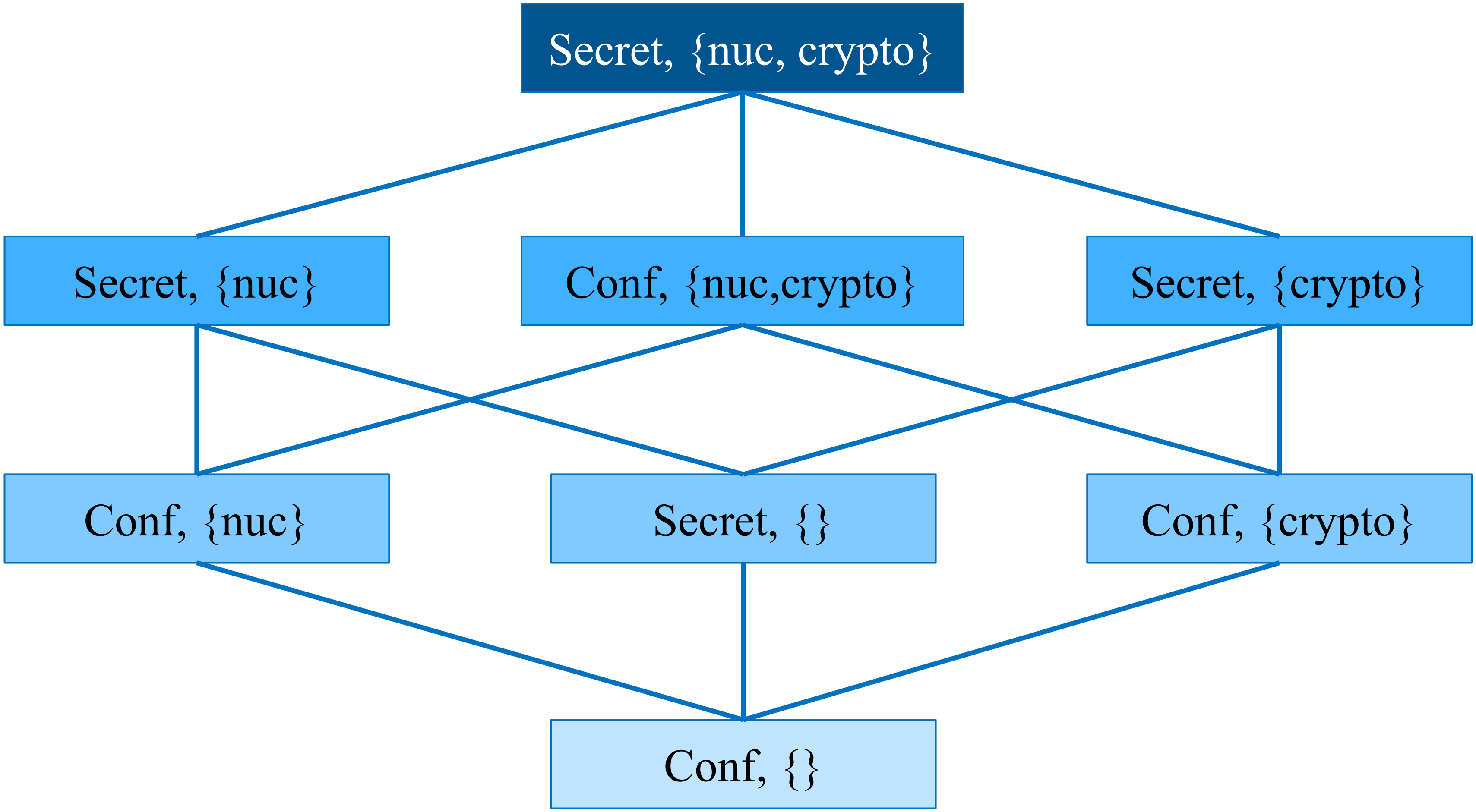
Scaling to many interactions…



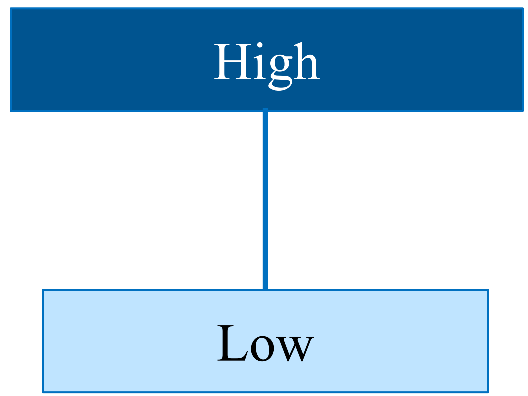
Scaling to many interactions…



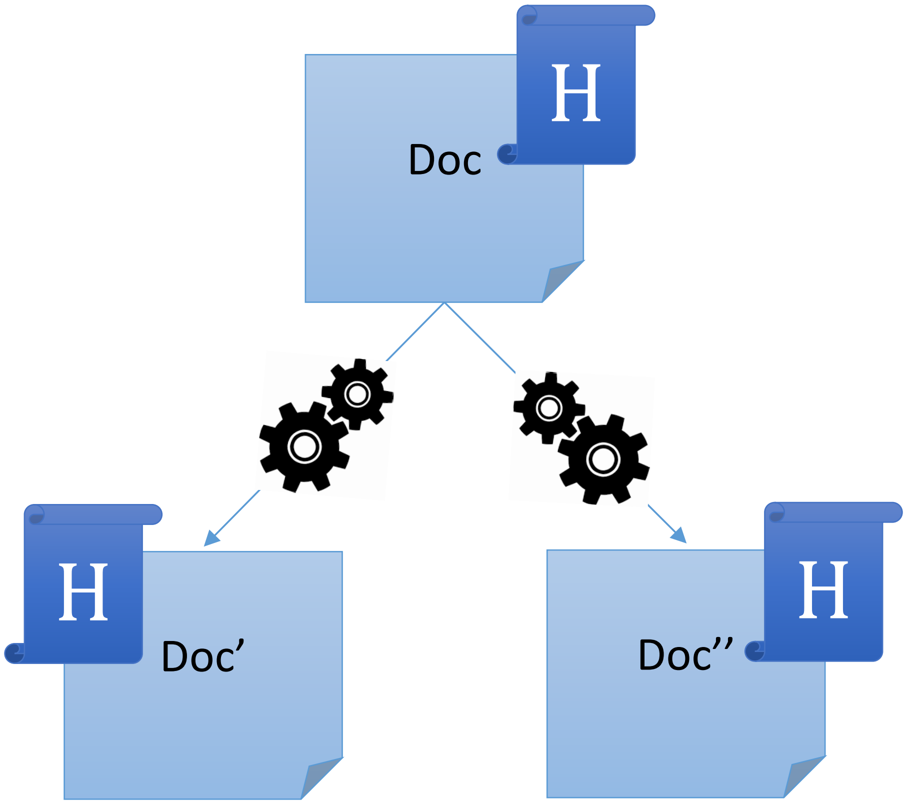
Labels represent policies



Labels represent policies



Labels represent policies



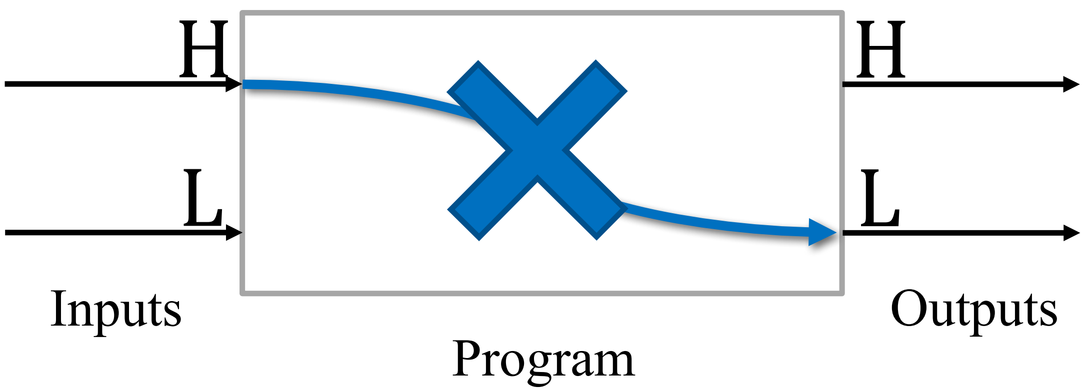
Policy Granularity

- Objects can be system principles (files, programs, sockets…)
- Objects can be program variables

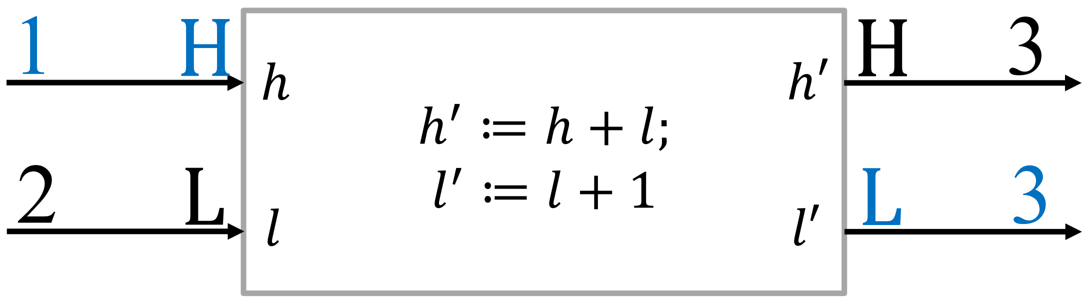
Noninterference
[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

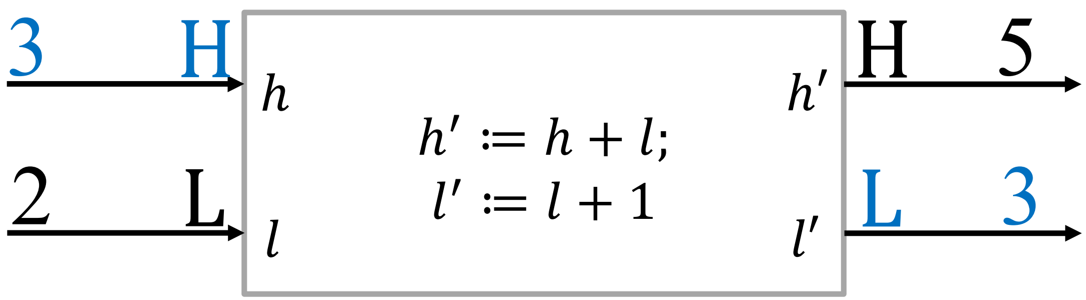
- Changes on H inputs should not cause changes on L outputs.



Noninterference: Example

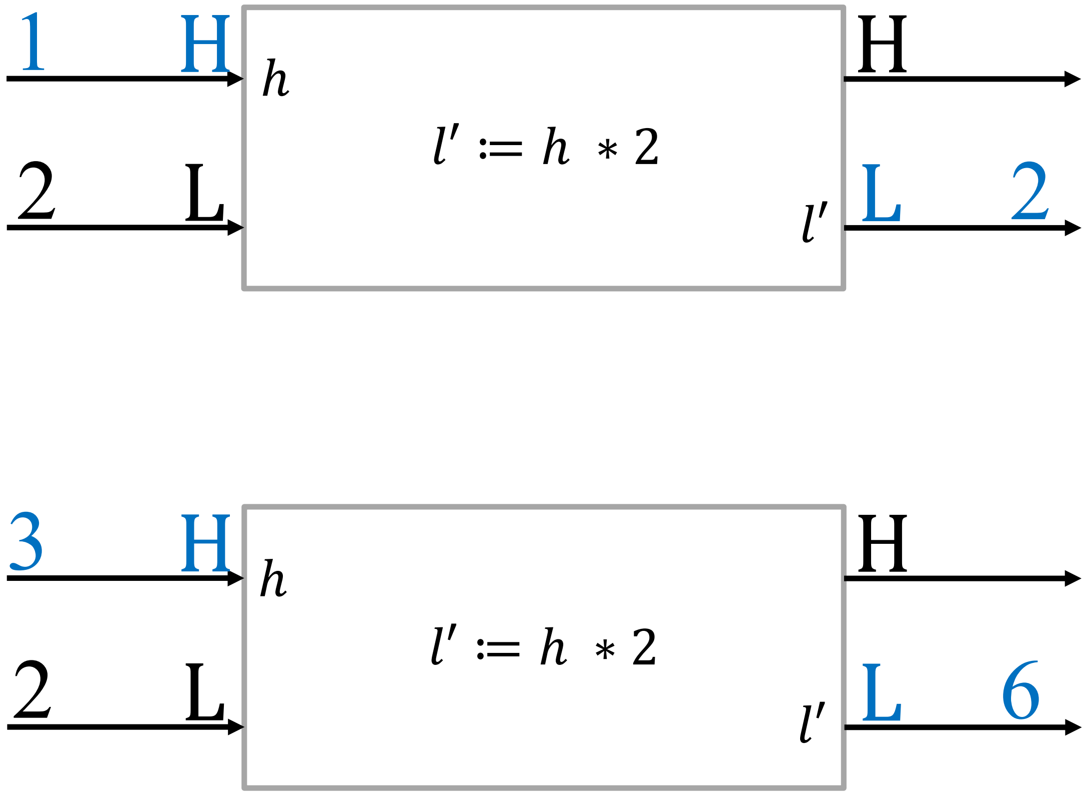


☺



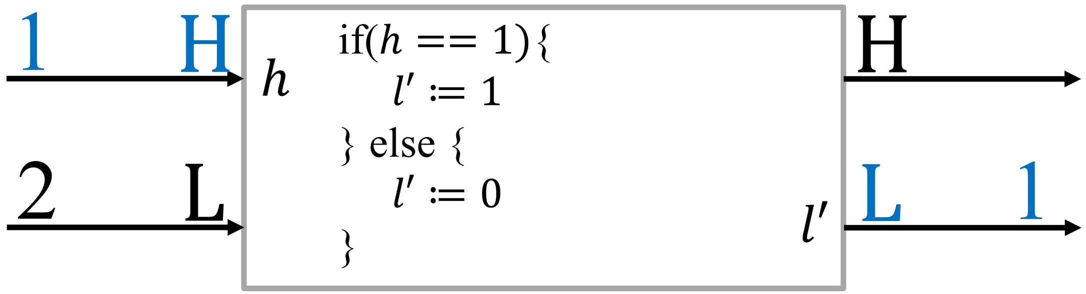
The program satisfies noninterference!

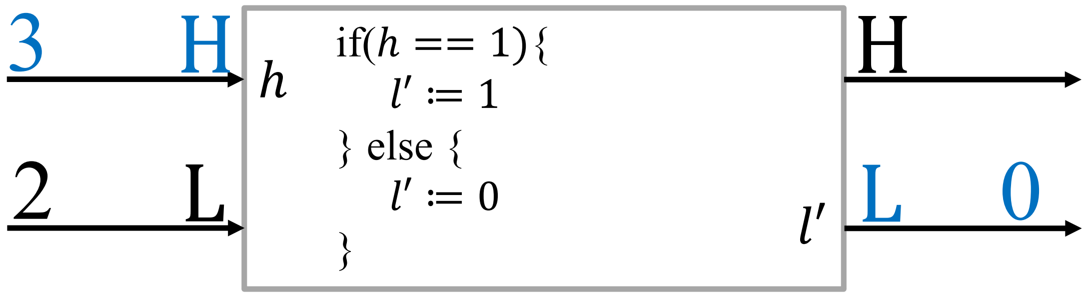
Noninterference: Example



The program does not satisfy noninterference!

Noninterference: Example





The program does not satisfy noninterference!

Noninterference

- Consider a program C.
- Consider two memories M_1 and M_2, such that
  - they agree on values of variables tagged with L:
  - $M_1 =_{\mathrm{L}} M_2$.

  M_1 and M_2 may not agree on values of variables tagged with H.

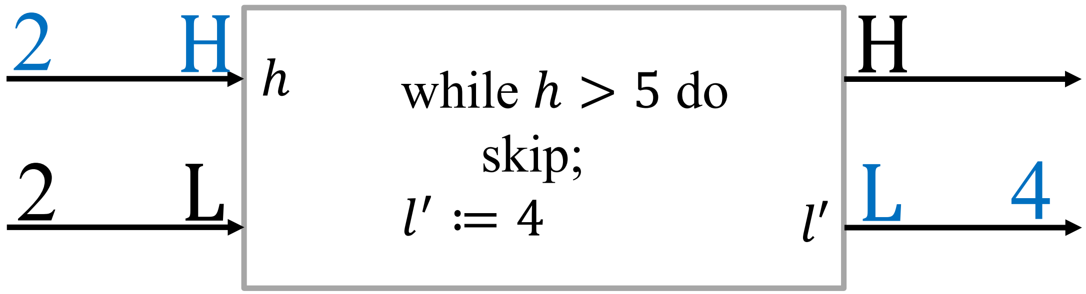
- $C(M_i)$ are the observations produced by executing C to termination on initial memory M_i:
  - final outputs, or
  - intermediate and final outputs.
- Then, observations tagged with L should be the same:
  - $C(M_1) =_{\mathrm{L}} C(M_2)$.

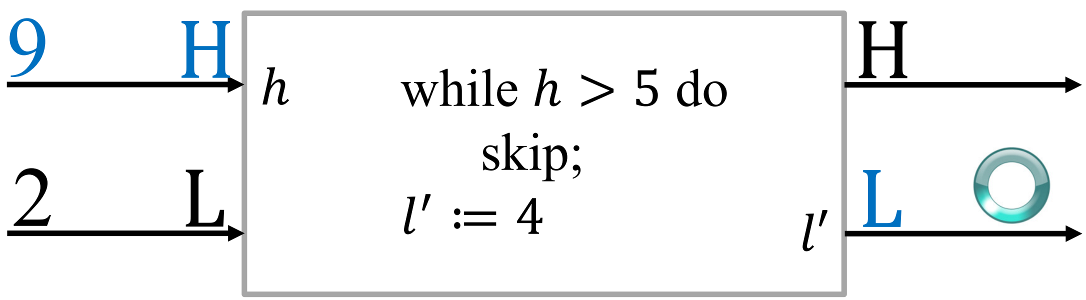
Noninterference

For a program C and a mapping from variables to labels in $\{\mathrm{L}, \mathrm{H}\}$:

$$\forall M_1, M_2: \text{ if } M_1 =_{\mathrm{L}} M_2, \text{ then } C(M_1) =_{\mathrm{L}} C(M_2).$$

Less restrictive than necessary…





Termination sensitive noninterference

- If
  - $M_1 =_\mathrm{L} M_2$,
- then
  - $\boldsymbol{C}$ **terminates on** $\boldsymbol{M_1}$ **iff** $\boldsymbol{C}$ **terminates on** $\boldsymbol{M_2}$**, and**
  - $C(M_1) =_\mathrm{L} C(M_2)$.

Less restrictive than necessary…

Required to be L.

L L L

m := Match(students; grades)

Wanted to be H!

More restrictive than necessary…

Required to be H.

$$\overset{H}{x} := maj(\overset{H}{v_1}, \overset{H}{v_2}, \ldots, \overset{H}{v_n})$$

Wanted to be L!

More restrictive than necessary…

Required to be H.

H H H

$$x := \mathrm{Enc}(\ v;\ k\ )$$

Wanted to be L!

Declassification

- **What**: specify what information may be declassified
  - e.g., LastFourDigits(credit card number) should be low
  - Partial Equivalence Relation (PER) Model
- **Who**: specify who may declassify information
  - e.g., high object owner can write to low objects
  - Decentralized Label Model
- **Where**: specify which pieces of code may declassify
  - e.g., encryption function can write to low objects
  - Intransitive Noninterference, Reactive Noninterference
- **When**: specify when information may be declassified
  - e.g., software key may be shared after payment has been received

Enforcement Mechanisms

- taint-tracking
- runtime monitoring
- type checking