Lecture 20: Information Flow

CS 181S

Fall 2020

Where we were…

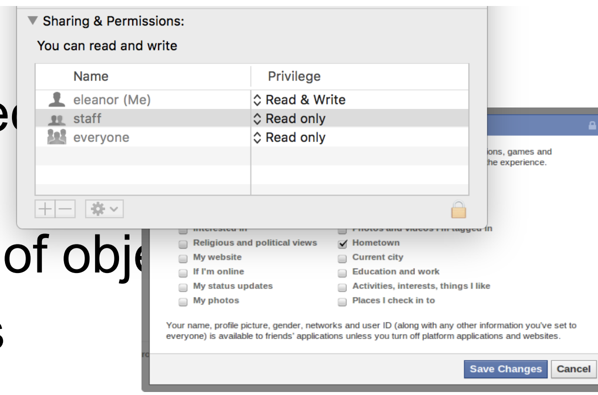
- **Authentication:** mechanisms that bind principals to actions

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

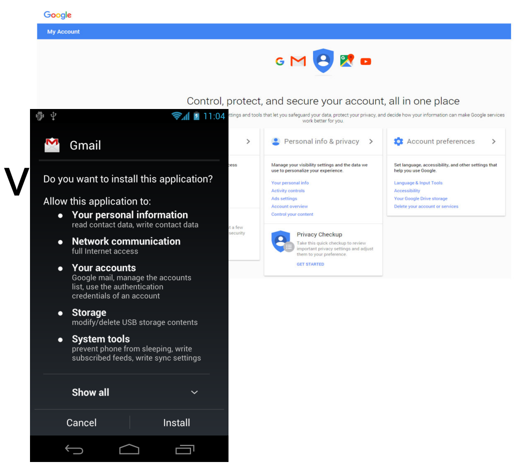
- **Audit:** mechanisms that record and review actions

Who defines Policies?

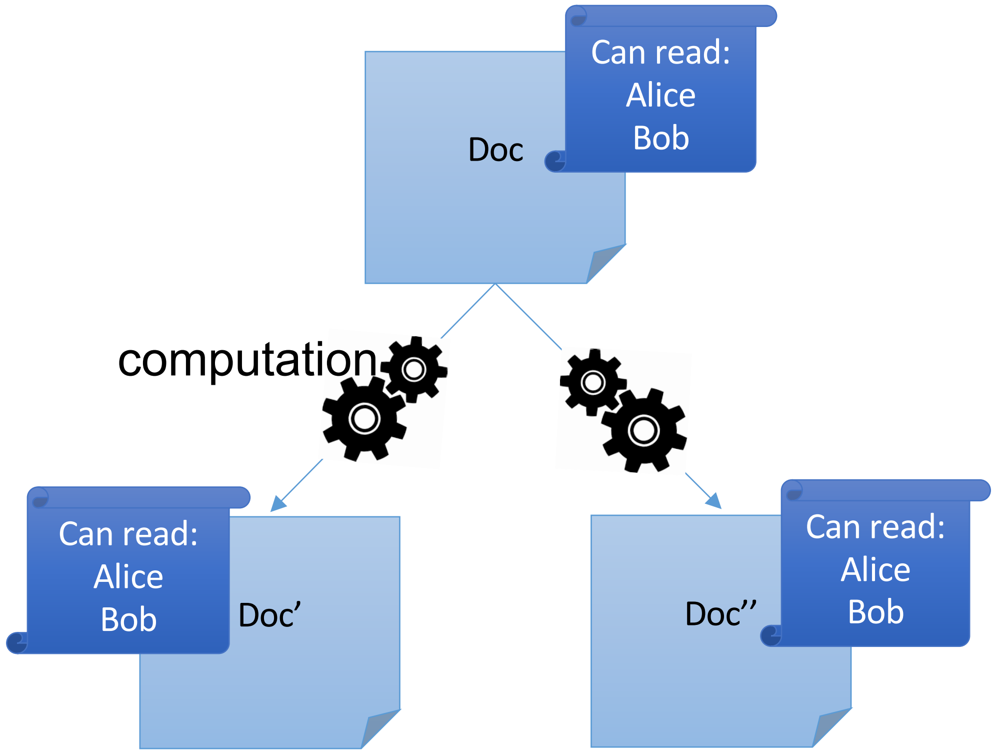
- Discretionary access control (DAC)
  - **Philosophy:** users have the *discretion* to spe themselves
  - Commonly, information belongs to the **owner** of obj
  - Access control lists, privilege lists, capabilities



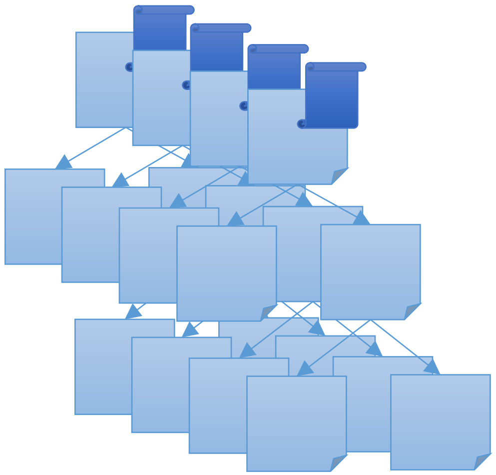
- Mandatory access control (MAC)
  - **Philosophy:** central authority *mandates* policy
  - Information belongs to the authority, not to the indiv
  - MLS and BLP, Chinese wall, Clark-Wilson, etc.



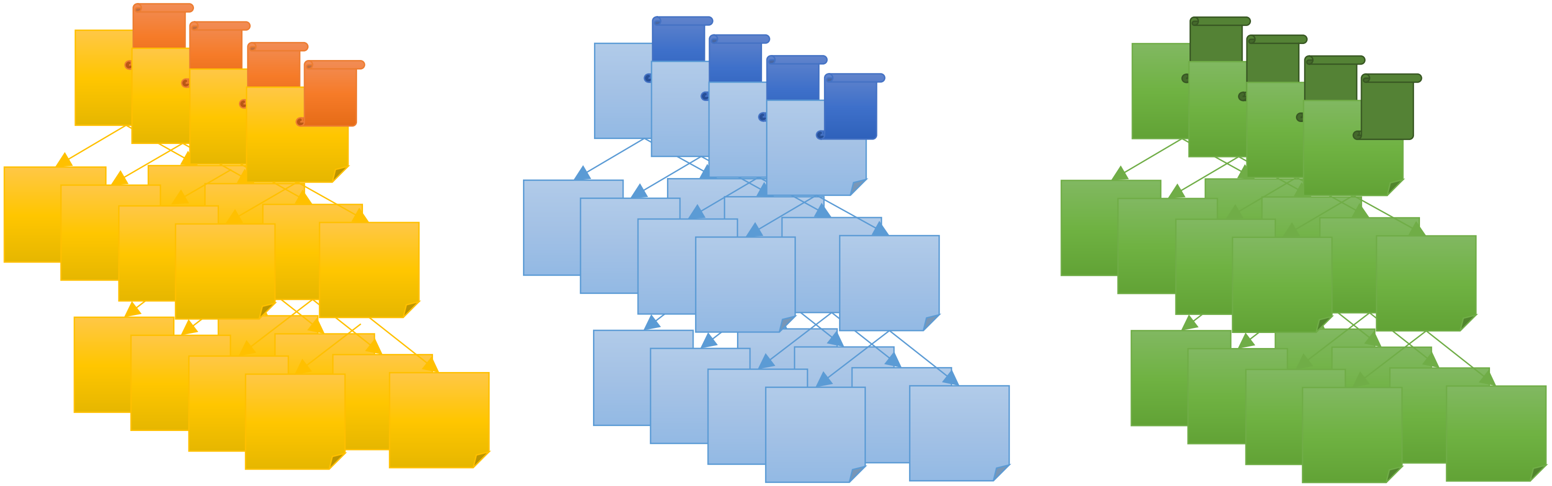
Access control for computed data



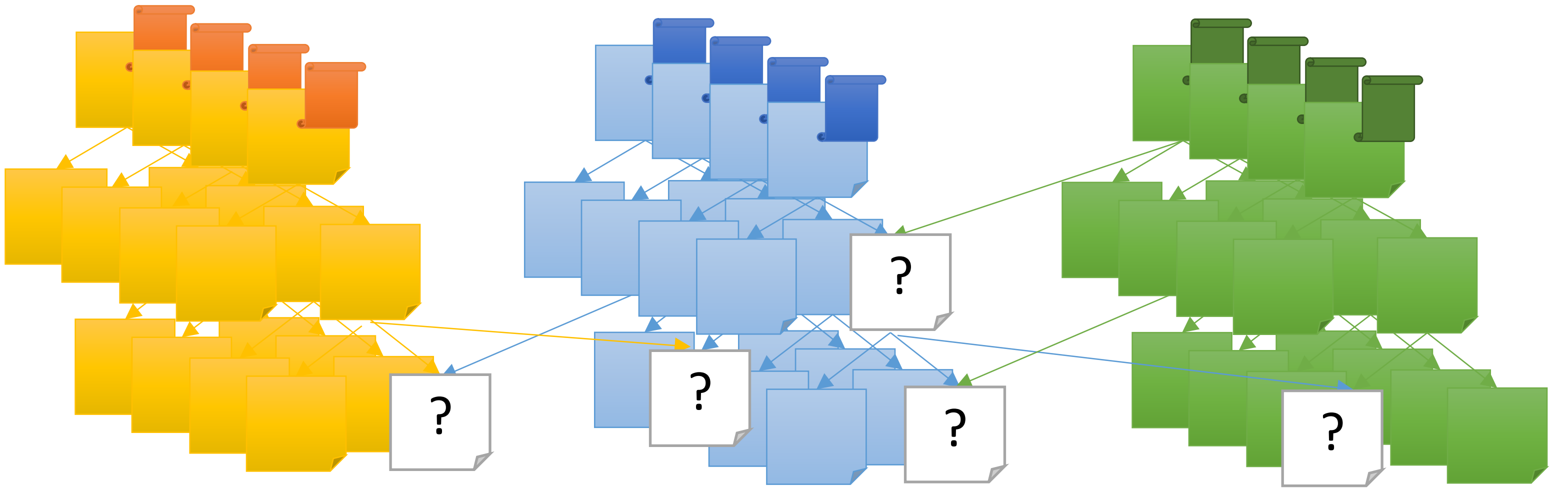
Scaling to many pieces of data…



Scaling to many users…

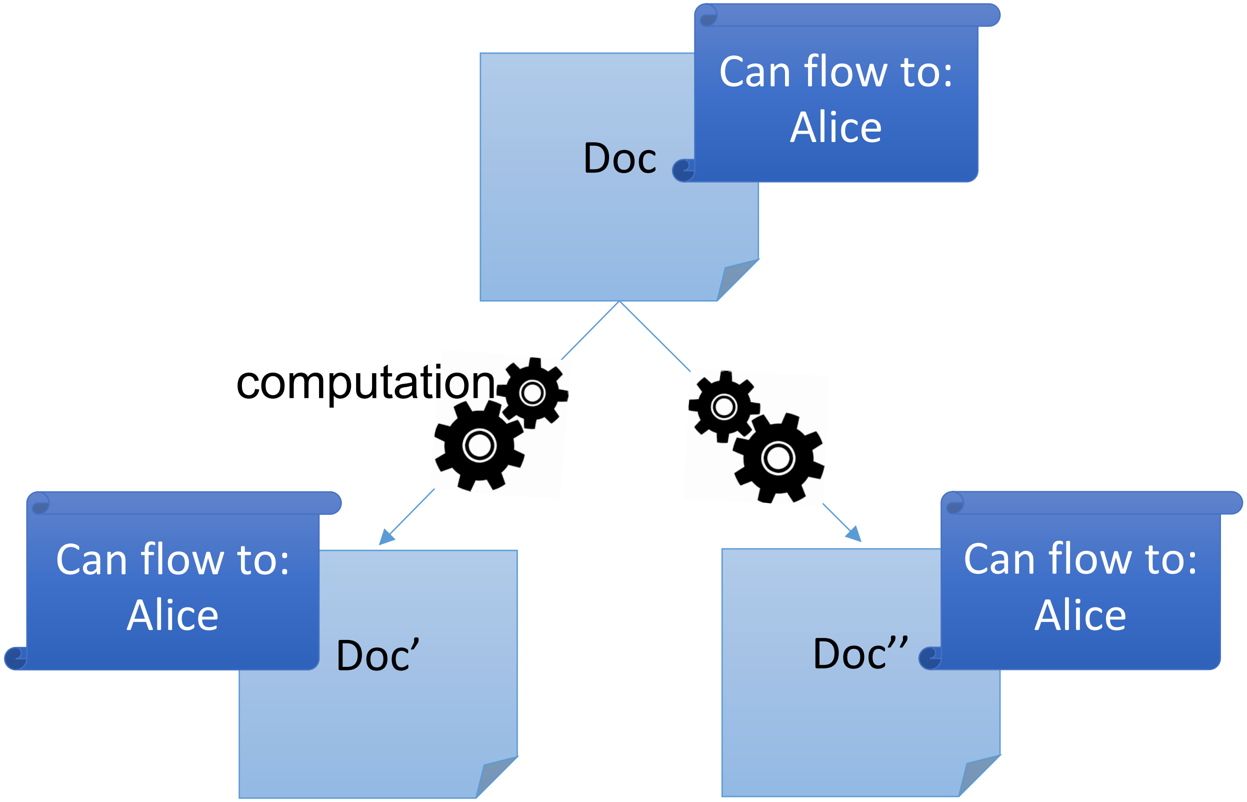


Scaling to many interactions…



Need to assign restrictions in an automatic way.

Information flow policies



Automatic deduction of policies!

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

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

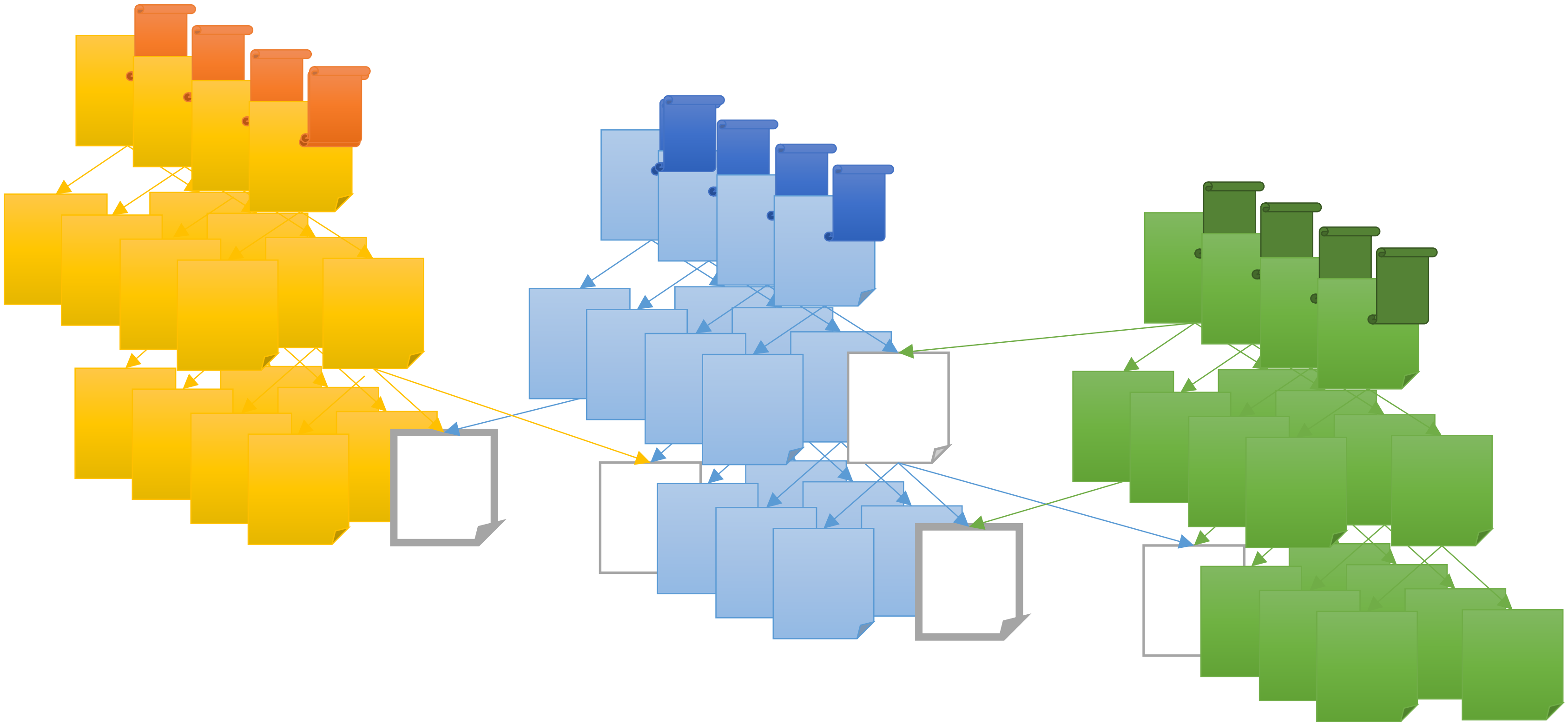
Different from the access control policy:
Value v is allowed to be read at most by Alice.

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

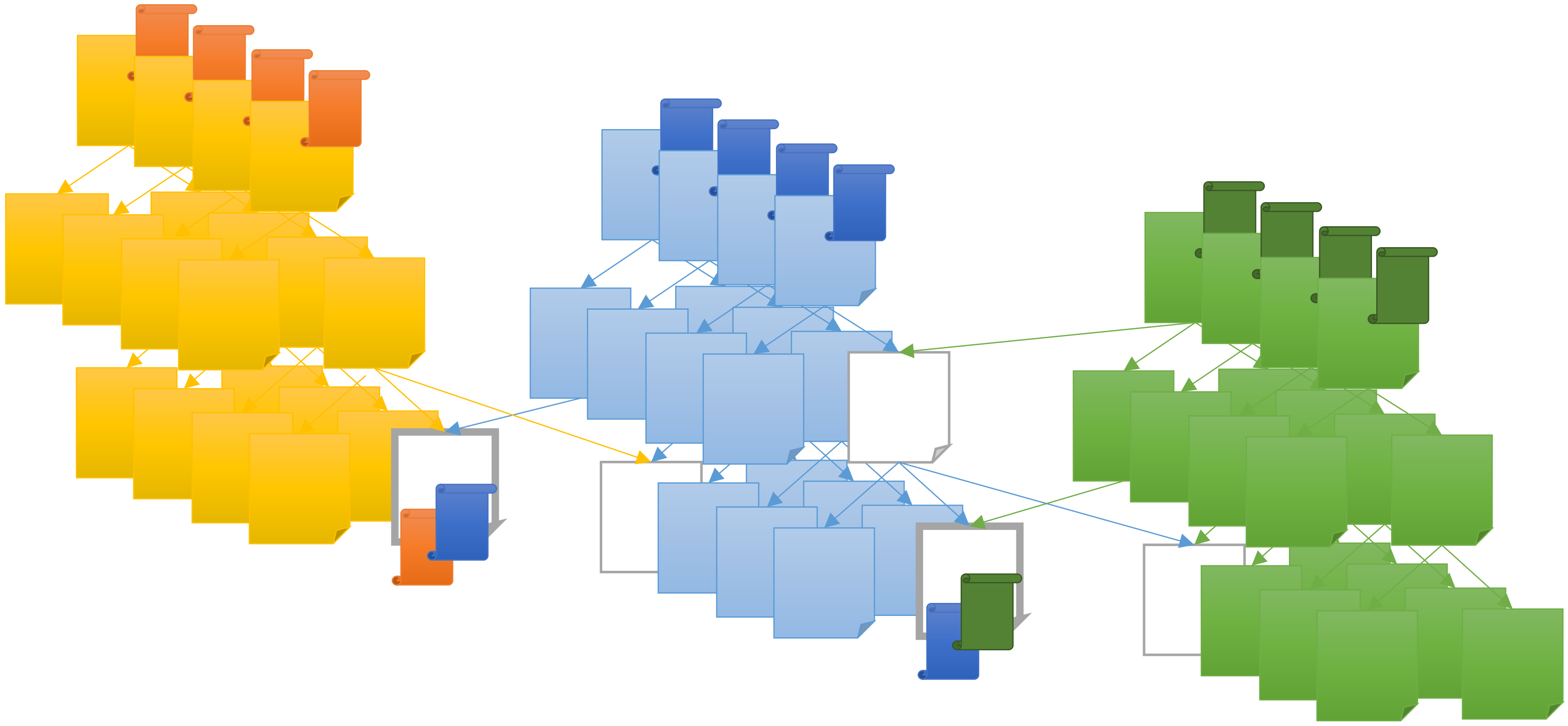
Policy Granularity

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

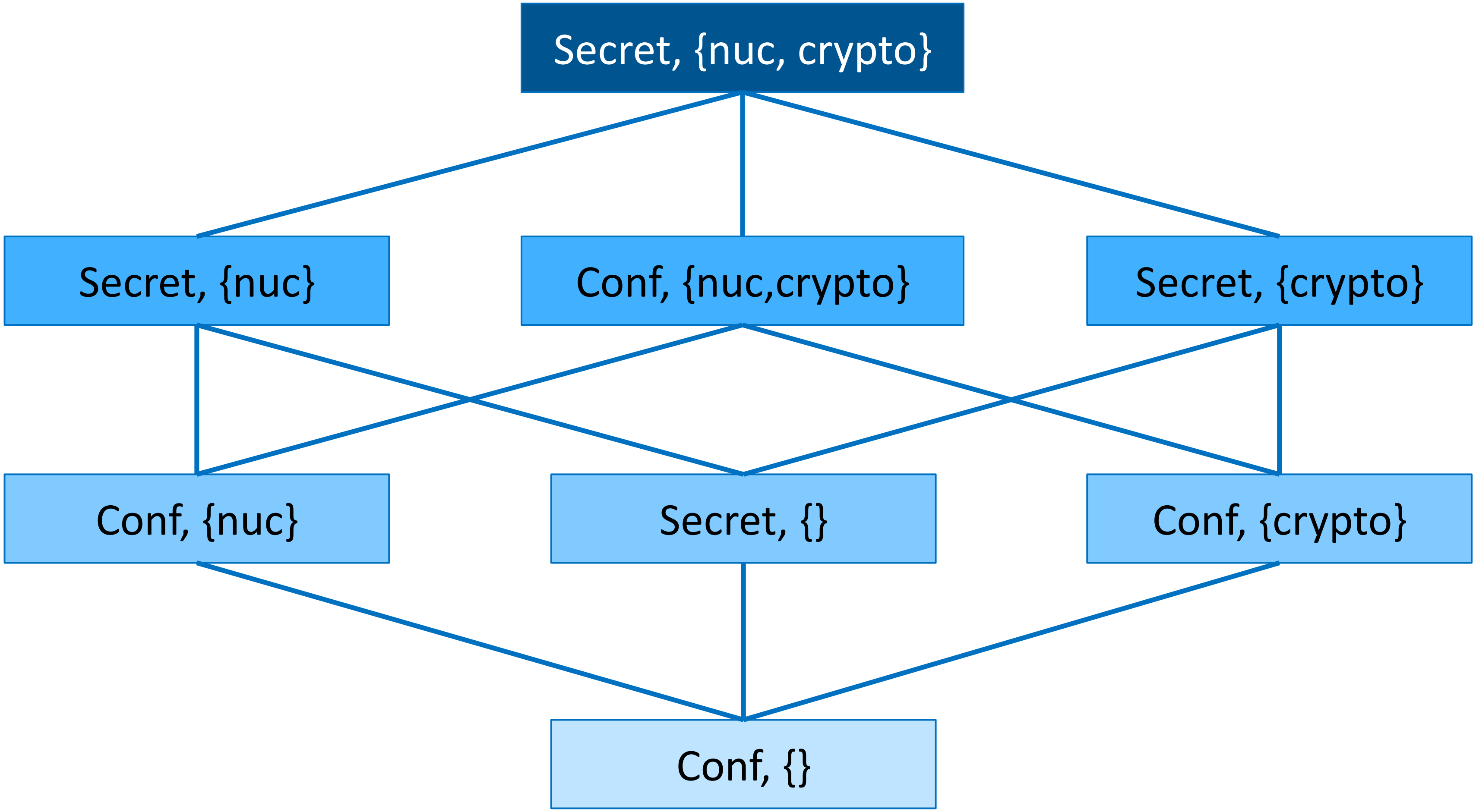
Scaling to many interactions…



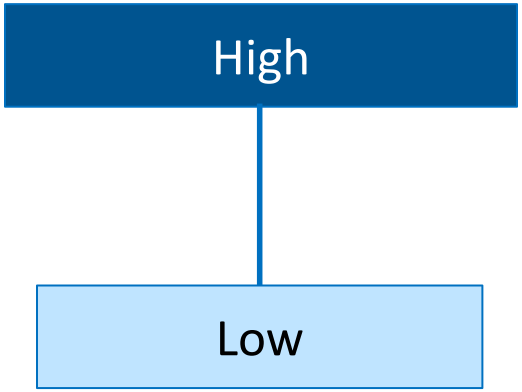
Scaling to many interactions…



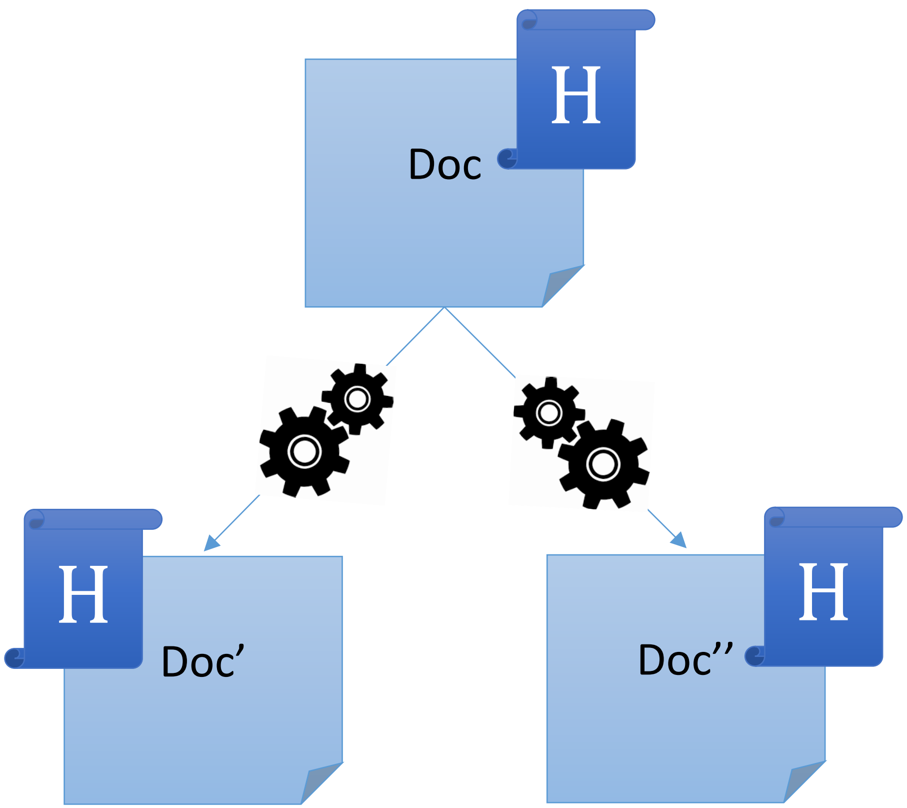
Labels represent policies



Labels represent policies



Labels represent policies

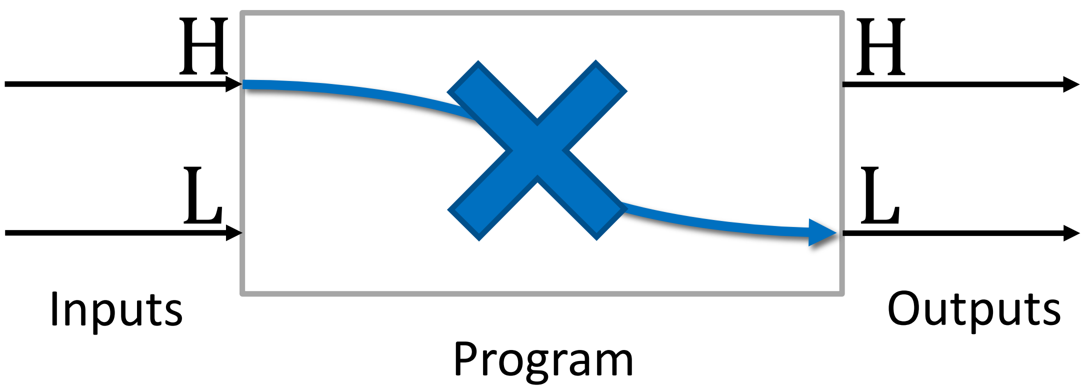


Noninterference

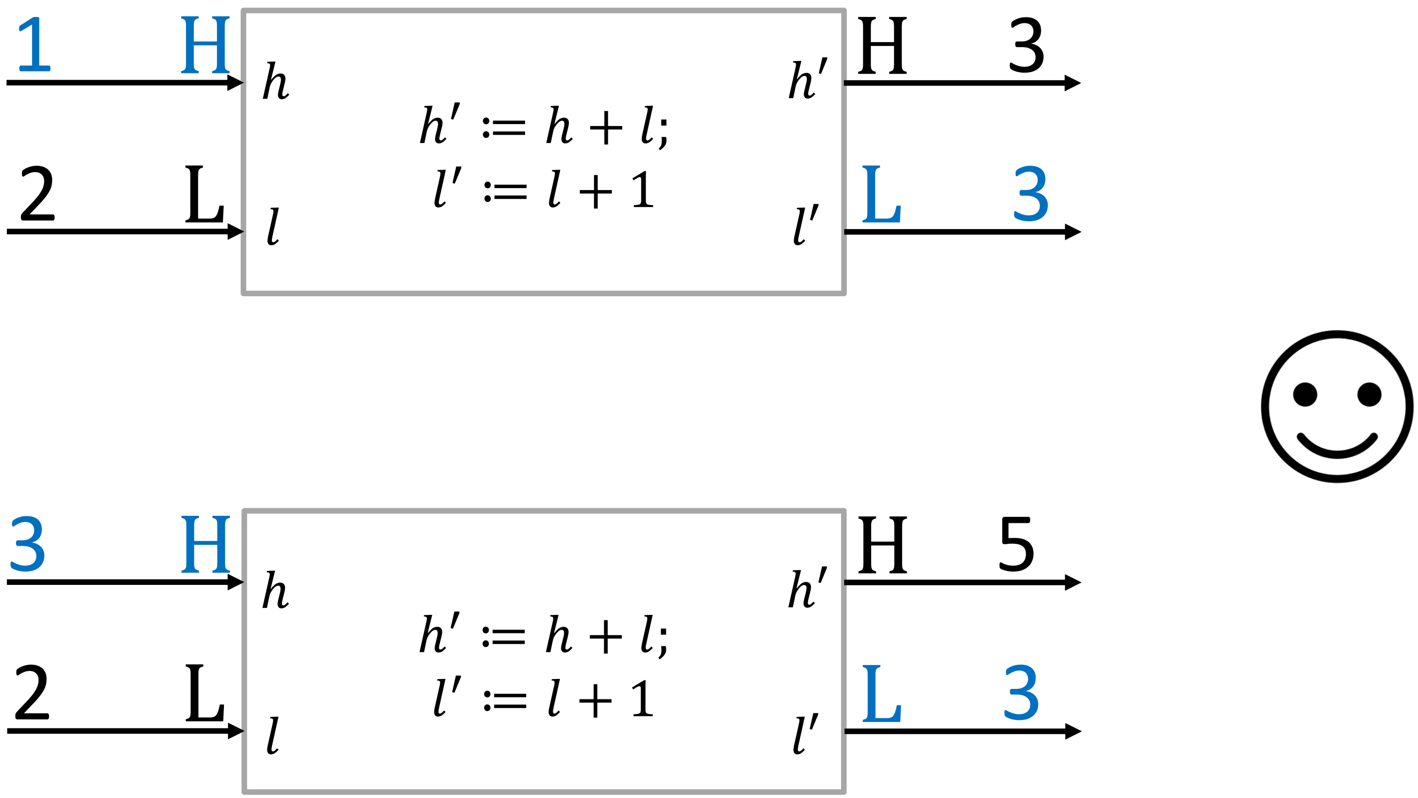
[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

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

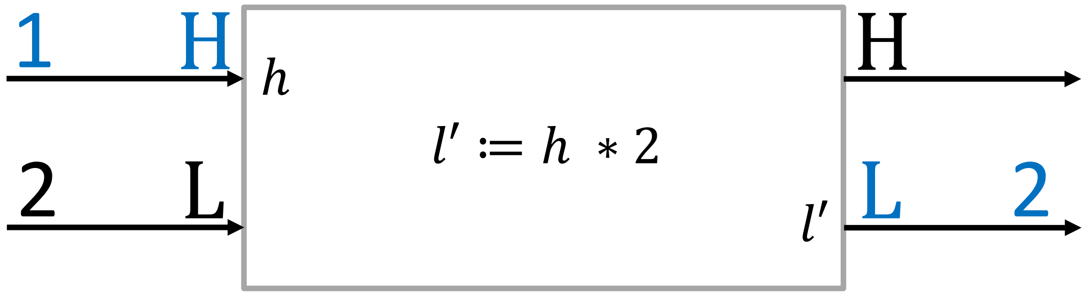


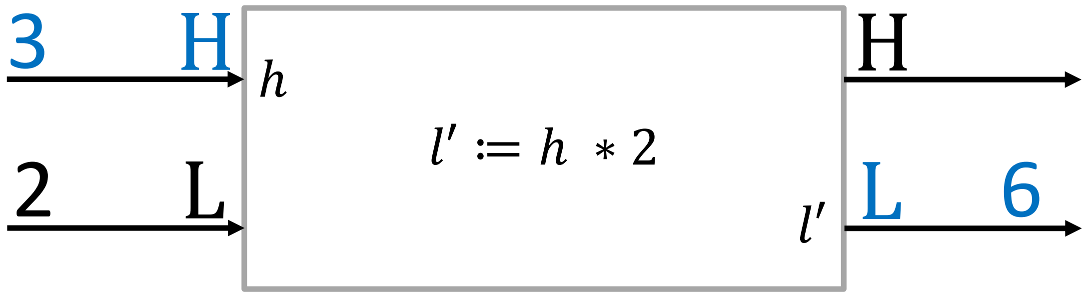
Noninterference: Example



The program satisfies noninterference!

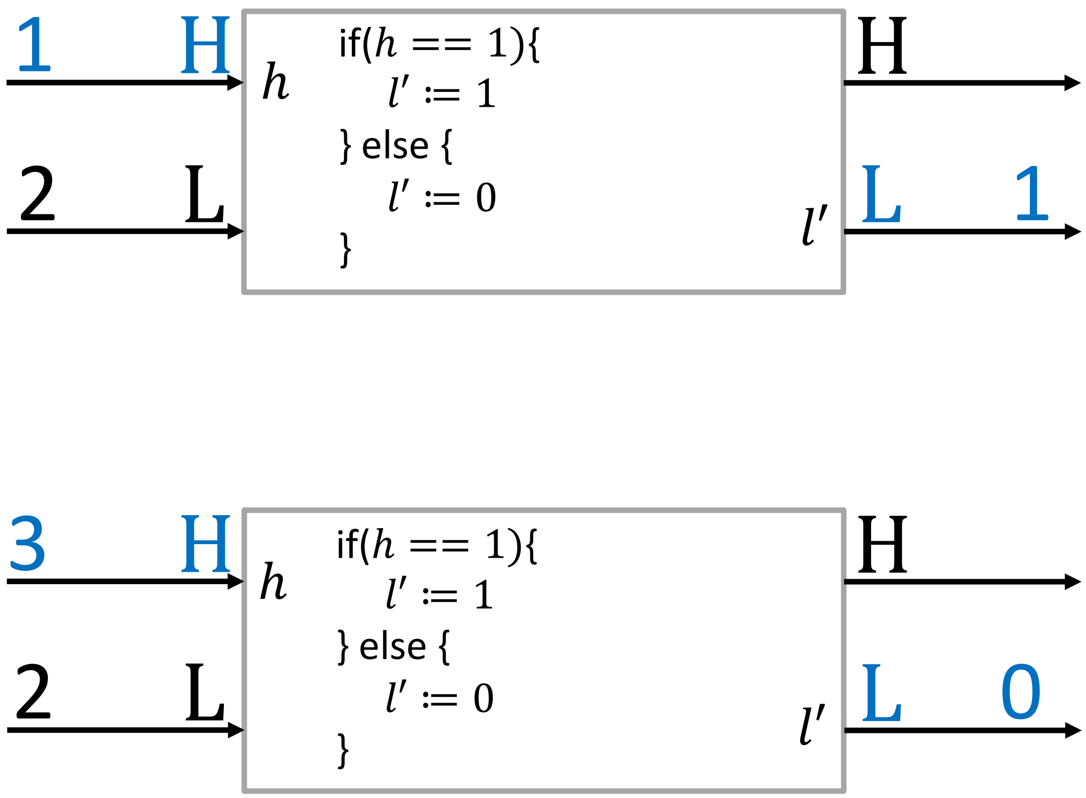
Noninterference: Example





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 might 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 $\{\mathrm{L}, \mathrm{H}\}$:

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

Exercise 1: Noninterference

- P outputs (H_O, L_O) where $H_O = H_I || L_I$ and $L_O = L_I$
  - || denotes string concatenation.

- P outputs L_O where $L_o = \begin{cases} L_I & \text{if } H_I \text{ is even} \\ L_I || L_I & \text{if } H_I \text{ is odd} \end{cases}$

Enforcement Mechanisms

- Static Information Flow Control:
  - type checking

- Dynamic Information Flow Control:
  - taint-tracking
  - runtime monitoring

A simple programming language

```
e ::= x | n | e1+e2 | ...

c ::= x = e
    | if e then c1 else c2
    | while e do c
    | c1; c2
```

Typing rules for expressions

Judgement $\Gamma \vdash \mathbf{e} : \ell$

According to mapping Γ, expression $\mathbf{e}$ has type (i.e., label) ℓ.

Constant: $\Gamma \vdash \mathbf{n} : \bot$

Variable: $\Gamma \vdash \mathbf{x} : \Gamma(\mathbf{x})$

Expression: $\Gamma \vdash \mathbf{e+e'} : \ell \sqcup \ell'$

if $\Gamma \vdash \mathbf{e} : \ell$

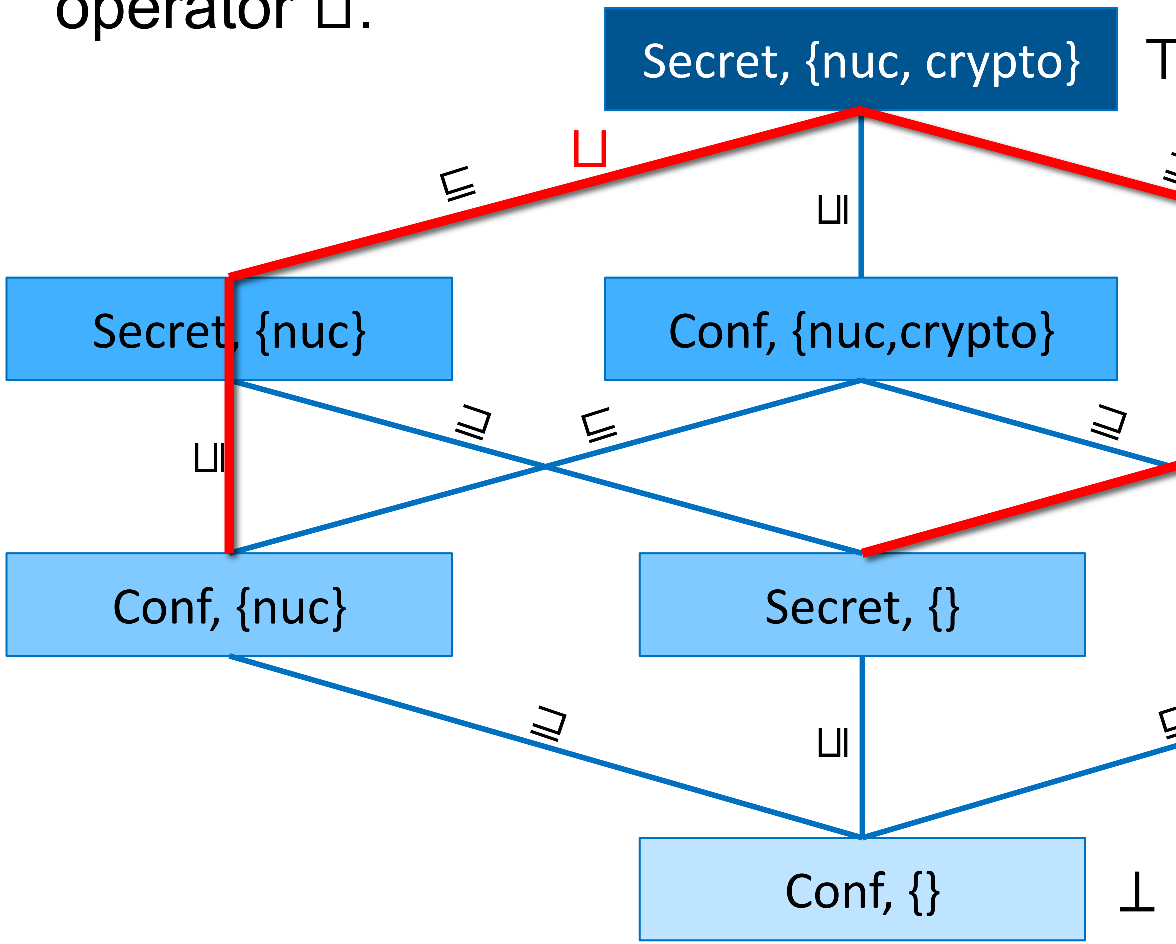
and $\Gamma \vdash \mathbf{e'} : \ell'$

Operator for combining labels

- For each ℓ and ℓ', there should exist label $\ell \sqcup \ell'$, such that:
  - $\ell \sqsubseteq \ell \sqcup \ell'$, $\ell' \sqsubseteq \ell \sqcup \ell'$, and
  - if $\ell \sqsubseteq \ell''$ and $\ell' \sqsubseteq \ell''$, then $\ell \sqcup \ell' \sqsubseteq \ell''$.
- $\ell \sqcup \ell'$ is called the **join** of ℓ and ℓ'.
- Operator $\sqcup$ is associative and commutative.

Lattice of labels

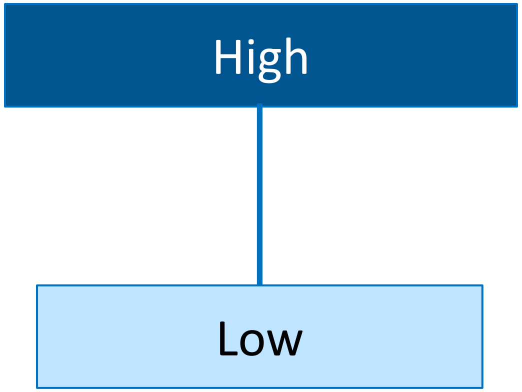
- The set of labels and relation ⊑ define a lattice, with join operator ⊔.



Exercise 2: Join

- What are the following labels (H or L)?

1. $H \sqcup H$
2. $H \sqcup L$
3. $L \sqcup H$
4. $L \sqcup L$



Typing rules for commands

Judgement $\Gamma, ctx \vdash$ **c**

According to mapping Γ, and context label ctx, command **c** is type correct (⇒ satisfies noninterference)

```
e ::= x | n | e1+e2 | ...

c ::= x = e
    | if e then c1 else c2
    | while e do c
    | c1; c2
```

Exercise 3: Checking an assignment

```
x = y
```

Examples for confidentiality

Γ(**x**) is L.
Γ(**y**) is L.
Does this assignment satisfy NI?

Γ(**x**) is H.
Γ(**y**) is L.
Does this assignment satisfy NI?

Γ(**x**) is L.
Γ(**y**) is H.
Does this assignment satisfy NI?

Checking an assignment

Assignments cause **explicit** information flows.

$$\mathtt{x = y}$$

It satisfies NI, if $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$.

Checking an assignment

$$\mathbf{x = y}$$

It satisfies NI, if $\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})$.

<u>MLS for confidentiality</u>

"no read up":

S may read O iff Label(O) ⊑ Label (S)

"no write down":

S may write O' iff Label(S) ⊑ Label (O')

Checking an assignment

$$\mathbf{x} = \mathbf{y}$$

It satisfies NI, if $\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})$.

<u>MLS for confidentiality</u>

“no read up”:

C may read **y** iff Label(**y**) ⊑ Label (C)

“no write down”:

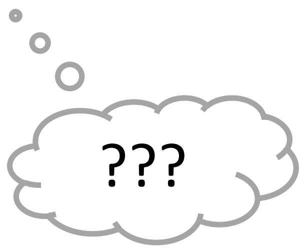
C may write **x** iff Label(C) ⊑ Label (**x**)

Checking an assignment

$$\mathtt{x = y + z}$$

It satisfies NI, if $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$ and $\Gamma(\mathtt{z}) \sqsubseteq \Gamma(\mathtt{x})$.

It satisfies NI, if $\Gamma(\mathtt{y+z}) \sqsubseteq \Gamma(\mathtt{x})$.



Checking an assignment

$$\mathbf{x} = \mathbf{y} + \mathbf{z}$$

It satisfies NI, if $\Gamma(\mathbf{y}) \sqcup \Gamma(\mathbf{z}) \sqsubseteq \Gamma(\mathbf{x})$.

Exercise 4: Checking an if-statement

```
if z > 0 then
    x = 1
else
    x = 0
```

Examples for confidentiality

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{z})$ is L.
Does this if-statement satisfy NI?

$\Gamma(\mathbf{x})$ is H.
$\Gamma(\mathbf{z})$ is L.
Does this if-statement satisfy NI?

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{z})$ is H.
Does this if-statement satisfy NI?

Checking an if-statement

```
if z > 0 then
    x = 1
else
    x = 0
```

Conditional commands (e.g., if-statements and while-statements) cause **implicit** information flows.

Context

```
if z > 0 then
    x = 1
else
    x = 0
```

They reveal information about **z**>0.

Introduce a context label ctx

Its ctx is $\Gamma(\mathbf{z})$.

Context

```
if z > 0 then
    x = 1
else
    x = 0
```

Check if

$ctx \sqcup \Gamma(\mathbf{e}) \sqsubseteq \Gamma(\mathbf{x})$.

Implicit flow

Explicit flow

Introduce a context label ctx

Its ctx is $\Gamma(\mathtt{z > 0})$.

Assignment rule

$\Gamma, ctx \vdash$ **`x:=e`**

`if` $\Gamma \vdash$ **`e`** $: \ell$

`and` $\ell \sqcup ctx \sqsubseteq \Gamma$**`(x)`**

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \ell \sqcup ctx \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, ctx \vdash \mathbf{x{:=}e}}$$

If-rule

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathbf{if\ e\ then\ c1\ else\ c2}}$$

Static type system

Assignment-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \ell \sqcup ctx \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, ctx \vdash \mathbf{x := e}}$$

If-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathbf{if\ e\ then\ c1\ else\ c2}}$$

While-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c}}{\Gamma, ctx \vdash \mathbf{while\ e\ do\ c}}$$

Sequence-Rule:

$$\frac{\Gamma, ctx \vdash \mathbf{c1} \qquad \Gamma, ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathbf{c1 ; c2}}$$

Soundness of type system

$$\Gamma, ctx \vdash \mathbf{c} \quad \Rightarrow \quad \mathbf{c} \text{ satisfies NI}$$

Exercise 5: Feedback

1. Rate how well you think this recorded lecture worked
   1. Better than an in-person class
   2. About as well as an in-person class
   3. Less well than an in-person class, but you still learned something
   4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have particular questions you would like me to address class?

4. Do you have any other comments or feedback?