CS 591: Formal Methods in Security and Privacy

Noninterference and Relational Hoare Logic

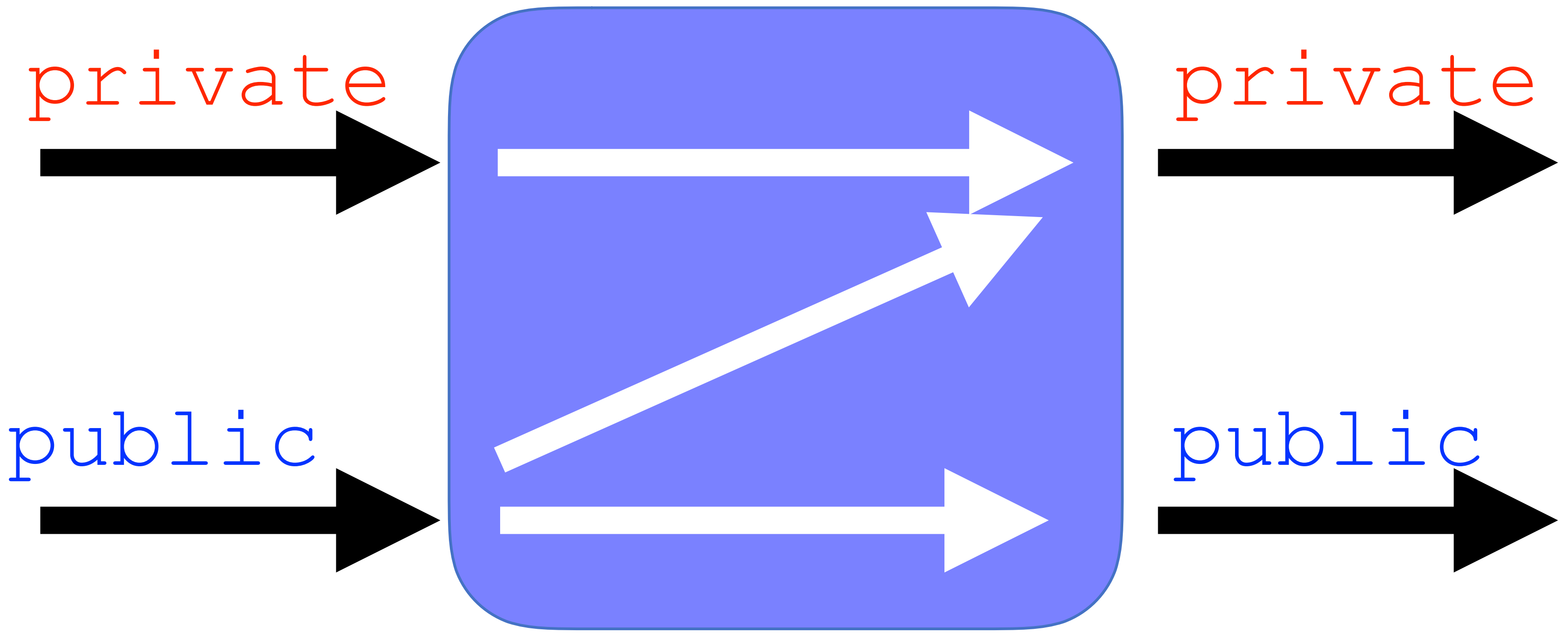
Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

From the previous classes

Information Flow Control

We want to guarantee that confidential inputs do not flow to nonconfidential outputs.



Low equivalence

Two memories m_1 and m_2 are low equivalent if and only if they coincide in the value that they assign to public variables.

In symbols: $m_1 \sim_{low} m_2$

Noninterference

A program `prog` is noninterferent if and only if, whenever we run it on two low equivalent memories m_1 and m_2 we have that:

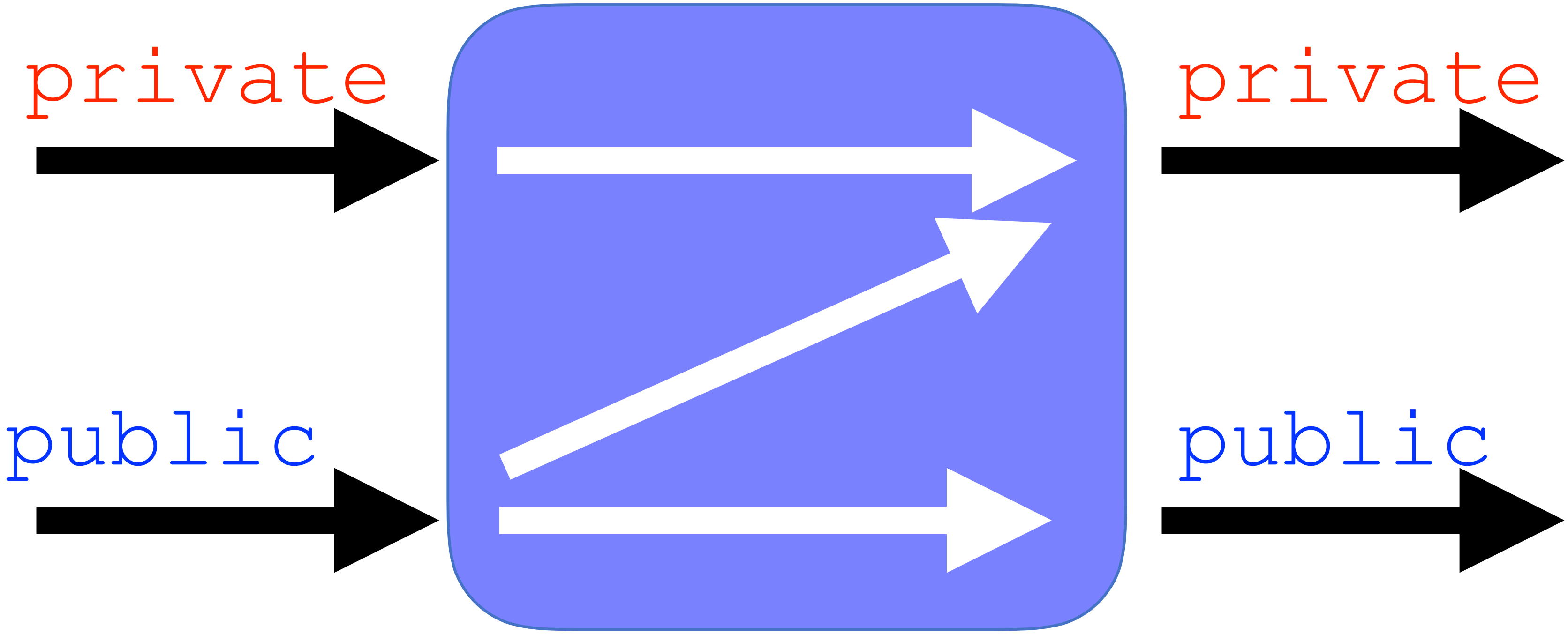
1) Either both terminate or both non-terminate;
2) If they both terminate we obtain two low equivalent memories m_1' and m_2'.

Noninterference

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$:

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$

2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



Does this program satisfy noninterference?

```
x:private
y:public

x:=y
```

Yes

Does this program satisfy noninterference?

```
x:private
y:public


y:=x
```

No

Is this program secure?

```
x:private
y:public

y:=x
y:=5
```

Yes

Does this program satisfy noninterference?

```
x:private
y:public

if x mod 3 = 0 then
  y:=1
else
  y:=0
```

No - an “implicit flow”

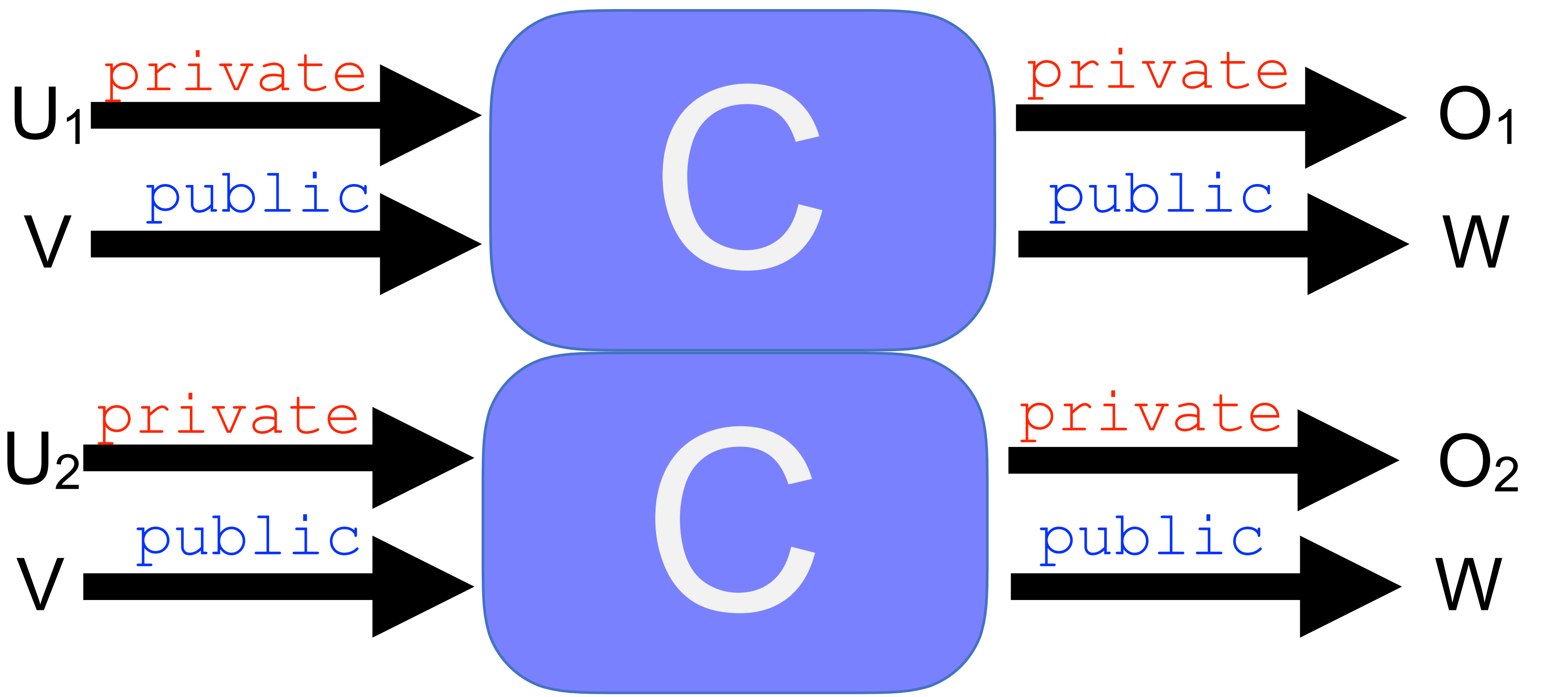
How can we prove our programs noninterferent?

Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$:

1) $\{c\}_{m1}=\perp$ iff $\{c\}_{m2}=\perp$

2) $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Hoare Logic - RHL

Precondition

Program$_1$ ~ Program$_2$

Postcondition

Precondition
(a logical formula)

$$c_1 \sim c_2 : P \Rightarrow Q$$

Program

Program

Postcondition
(a logical formula)

Relational Assertions

$$c_1 \sim c_2 : P \Rightarrow Q$$

Need to talk about variables
of the two memories

$$c_1 \sim c_2 : x\langle 1\rangle \leq x\langle 2\rangle \Rightarrow x\langle 1\rangle \geq x\langle 2\rangle$$

Tags describing which
memory we are referring to.

Validity of Hoare quadruple

We say that the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid if and only if for every pair of memories m_1, m_2 such that $P(m_1, m_2)$ we have:

1) $\{c_1\}_{m1} = \perp$ iff $\{c_2\}_{m2} = \perp$
2) $\{c_1\}_{m1} = m_1'$ and $\{c_2\}_{m2} = m_2'$ implies $Q(m_1', m_2')$.

How do we check this?

Rules of Relational Hoare Logic

Skip

$$\frac{}{\vdash \texttt{skip} \sim \texttt{skip} : P \Rightarrow P}$$

Rules of Relational Hoare Logic

Abort

$$\frac{}{\vdash \texttt{abort} \sim \texttt{abort} : \texttt{true} \Rightarrow \texttt{false}}$$

Rules of Relational Hoare Logic

Assignment

$$\overline{\vdash x_1 := e_1 \sim x_2 := e_2 : P[e_1\langle 1\rangle / x_1\langle 1\rangle, e_2\langle 2\rangle / x_2\langle 2\rangle] \Rightarrow P}$$

What is changed from last class?

Today: More Relational Hoare Logic

Rules of Relational Hoare Logic
Assignment Example

$$\vdash \texttt{x:=x+1} \sim \texttt{y:=y-1}:$$
$$\texttt{(x<1> = -y<2>)}$$
$$\texttt{[(x+1)<1>/x<1>,}$$
$$\texttt{(y-1)<2>/y<2>]} \Rightarrow$$
$$\texttt{x<1> = -y<2>}$$

Rules of Relational Hoare Logic
Assignment Example

$$\frac{}{\vdash \texttt{x:=x+1} \sim \texttt{y:=y-1} : \texttt{(x<1> = -y<2>)[(x<1>+1)/x<1>, (y<2>-1)/y<2>]} \Rightarrow \texttt{x<1> = -y<2>}}$$

Rules of Relational Hoare Logic
Assignment example

$$\vdash \texttt{x:=x+1} \sim \texttt{y:=y-1}:$$
$$\texttt{x<1>+1} = -(\texttt{y<2>-1}) \Rightarrow$$
$$\texttt{x<1>} = -\texttt{y<2>}$$

Rules of Relational Hoare Logic

Composition

$$\frac{\vdash \mathtt{c}_1 \sim \mathtt{c}_2 : P \Rightarrow R \qquad \vdash \mathtt{c}_1' \sim \mathtt{c}_2' : R \Rightarrow S}{\vdash \mathtt{c}_1;\mathtt{c}_1' \sim \mathtt{c}_2;\mathtt{c}_2' : P \Rightarrow S}$$

Rules of Relational Hoare Logic

Consequence

$$\frac{P \Rightarrow S \qquad \vdash c_1 \sim c_2 : S \Rightarrow R \qquad R \Rightarrow Q}{\vdash c_1 \sim c_2 : P \Rightarrow Q}$$

We can weaken P, i.e. replace it by something that is implied by P. In this case S.

We can strengthen Q, i.e. replace it by something that implies Q. In this case R.

Consequence + Assignment

Example

$$x\langle 1\rangle = -y\langle 2\rangle \Rightarrow x\langle 1\rangle + 1 = -(y\langle 2\rangle - 1)$$

$$\vdash x := x+1 \sim y := y-1 :$$
$$x\langle 1\rangle + 1 = -(y\langle 2\rangle - 1) \Rightarrow x\langle 1\rangle = -y\langle 2\rangle$$

$$x\langle 1\rangle = -y\langle 2\rangle \Rightarrow x\langle 1\rangle = -y\langle 2\rangle$$

$$\vdash x := x+1 \sim y := y-1 :$$
$$x\langle 1\rangle = -y\langle 2\rangle \Rightarrow x\langle 1\rangle = -y\langle 2\rangle$$

Rules of Relational Hoare Logic

If-then-else

$$\vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge \neg e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\frac{\vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge e_2\langle 2\rangle \wedge P \Rightarrow Q \qquad \vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge \neg e_2\langle 2\rangle \wedge P \Rightarrow Q}{\vdash \begin{array}{c} \texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2' \end{array} : P \Rightarrow Q}$$

Is this correct?

Rules of Relational Hoare Logic

If-then-else

$$\mathtt{P} \Rightarrow (\mathtt{e}_1\mathtt{<1>} \Leftrightarrow \mathtt{e}_2\mathtt{<2>})$$

$$\vdash \mathtt{c}_1 \sim \mathtt{c}_2 : \mathtt{e}_1\mathtt{<1>} \wedge \mathtt{P} \Rightarrow \mathtt{Q}$$

$$\vdash \mathtt{c}_1' \sim \mathtt{c}_2' : \neg \mathtt{e}_1\mathtt{<1>} \wedge \mathtt{P} \Rightarrow \mathtt{Q}$$

$$\vdash \begin{array}{c} \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{c}_1\ \mathtt{else}\ \mathtt{c}_1' \\ \sim \\ \mathtt{if}\ \mathtt{e}_2\ \mathtt{then}\ \mathtt{c}_2\ \mathtt{else}\ \mathtt{c}_2' \end{array} : \mathtt{P} \Rightarrow \mathtt{Q}$$

Rules of Relational Hoare Logic

While

$$\frac{P \Rightarrow (e_1\langle 1\rangle \Leftrightarrow e_2\langle 2\rangle) \qquad \vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge P \Rightarrow P}{\vdash \begin{array}{c} \texttt{while } e_1 \texttt{ do } c_1 \\ \sim \\ \texttt{while } e_2 \texttt{ do } c_2 \end{array} : P \Rightarrow P \wedge \neg e_1\langle 1\rangle}$$

Invariant

Rules of Relational Hoare-Logic
One-sided Rules

What do we do if our two programs have different forms? There are three pairs of *one-sided* rules.

$$\vdash \begin{array}{c} \texttt{if e then } c_1 \texttt{ else } c_1' \\ \sim \\ c_2 \end{array} : P \Rightarrow Q$$

Rules of Relational Hoare Logic

If-then-else — left

$$\vdash c_1 \sim c_2 : e\langle 1\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1' \sim c_2 : \neg e\langle 1\rangle \wedge P \Rightarrow Q$$

$$\vdash \begin{array}{c} \texttt{if } e \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ c_2 \end{array} : P \Rightarrow Q$$

Rules of Relational Hoare Logic

If-then-else — right

$$\vdash c_1 \sim c_2 : e\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1 \sim c_2' : \neg e\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash \begin{array}{c} c_1 \\ \sim \\ \texttt{if } e \texttt{ then } c_2 \texttt{ else } c_2' \end{array} : P \Rightarrow Q$$

Rules of Relational Hoare Logic

Assignment — left

```
⊢x:=e ~ skip:
  P[e<1>/x<1>] ⇒ P
```

Rules of Relational Hoare Logic

Assignment — right

$$\dfrac{}{\vdash \texttt{skip} \sim \texttt{x:=e} : \texttt{P[e<2>/x<2>]} \Rightarrow \texttt{P}}$$

Also pair of one-sided rules for while — we'll ignore for now

Rules of Relational Hoare Logic
Program Equivalence Rule

$\models \mathtt{P : c_1 \equiv c_2}$ **means** $\mathtt{\{c_1\}_m = \{c_2\}_m}$
for all $\mathtt{m}$ **such that** $\mathtt{P(m)}$

$$\frac{\models \mathtt{P : c_1' \equiv c_1} \qquad \models \mathtt{P : c_2' \equiv c_2} \qquad \mathtt{c_1' \sim c_2' : P \Rightarrow Q}}{\vdash \mathtt{c_1 \sim c_2 : P \Rightarrow Q}}$$

Rules of Relational Hoare Logic
Program Equivalences

$$\models P \ : \ \texttt{skip;c} \equiv \texttt{c}$$

$$\models P \ : \ \texttt{c;skip} \equiv \texttt{c}$$

$$\models P : \texttt{(c1;c2);c3} \equiv \texttt{c1;(c2;c3)}$$

...

Rules of Relational Hoare Logic
Combining Composition and Equivalence

We can combine the Composition and Program Equivalence Rules to split commands where we like:

$$\frac{\vdash \texttt{c}_1\texttt{;c}_2 \sim \texttt{c}_1{}' : P \Rightarrow R \qquad \vdash \texttt{c}_3 \sim \texttt{c}_2{}'\texttt{;c}_3{}' : R \Rightarrow Q}{\vdash \texttt{c}_1\texttt{;c}_2\texttt{;c}_3 \sim \texttt{c}_1{}'\texttt{;c}_2{}'\texttt{;c}_3{}' : P \Rightarrow Q}$$

Rules of Relational Hoare Logic
Combining Composition and Equivalence

$$\vdash \texttt{c}_1 \sim \texttt{skip}: P \Rightarrow R$$

$$\vdash \texttt{c}_2 \sim \texttt{c}_1': R \Rightarrow Q$$

$$\overline{\vdash \texttt{c}_1;\texttt{c}_2 \sim \texttt{skip};\texttt{c}_1': P \Rightarrow Q}$$

$$\overline{\vdash \texttt{c}_1;\texttt{c}_2 \sim \texttt{c}_1': P \Rightarrow Q}$$

Rules of Relational Hoare Logic
Combining Composition and Equivalence

$$\vdash c_1 \sim c_1' : P \Rightarrow R$$

$$\vdash c_2 \sim \texttt{skip} : R \Rightarrow Q$$

$$\vdash c_1;c_2 \sim c_1';\texttt{skip} : P \Rightarrow Q$$

$$\vdash c_1;c_2 \sim c_1' : P \Rightarrow Q$$

Relational Hoare Logic in EasyCrypt

- EasyCrypt's implementation of Relational Hoare Logic has much in common with its implementation of Hoare Logic.
- Look for the pRHL tactics in Section 3.4 of the EasyCrypt Reference Manual (the "p" stands for "probabilistic", but ignore that for now).

In Lab next, we'll look at some noninterference proofs in EasyCrypt