Formal Reasoning for Security and Privacy

Lecture 3

Marco Gaboardi
Boston University
gaboardi@bu.edu

Formal Semantics

We need to assign a formal meaning to the different components:

Precondition

Program

Postcondition

formal semantics of specification conditions

formal semantics of programs

formal semantics of specification conditions

We also need to describe the rules which combine program and specifications.

Hoare triple

```
Precondition
Program
Postcondition
```

Precondition
(a logical formula)

$$c : P \Rrightarrow Q$$

Program

Postcondition
(a logical formula)

Rules of Hoare Logic:

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

$$\frac{}{\vdash x := e\ :\ P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \qquad \vdash c' : R \Rightarrow Q}{\vdash c; c' :\ P \Rightarrow Q}$$

$$\frac{P \Rightarrow S \qquad \vdash c : S \Rightarrow R \qquad R \Rightarrow Q}{\vdash c :\ P \Rightarrow Q}$$

$$\frac{\vdash c_1 : e \wedge P \Rightarrow Q \qquad \vdash c_2 : \neg e \wedge P \Rightarrow Q}{\vdash \texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2\ :\ P \Rightarrow Q}$$

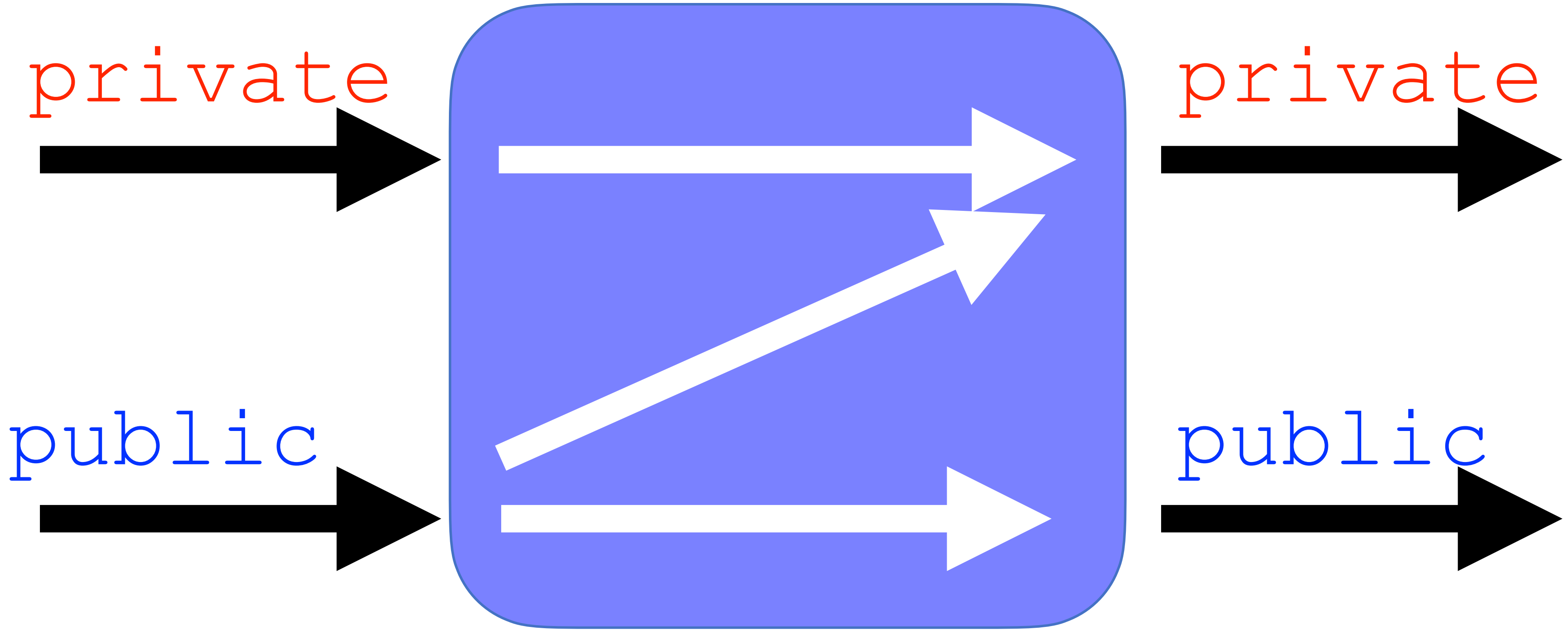
$$\frac{\vdash c\ :\ e \wedge P \Rightarrow P}{\vdash \texttt{while } e \texttt{ do } c\ :\ P \Rightarrow P \wedge \neg e}$$

Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.

Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.



Relational Property

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

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

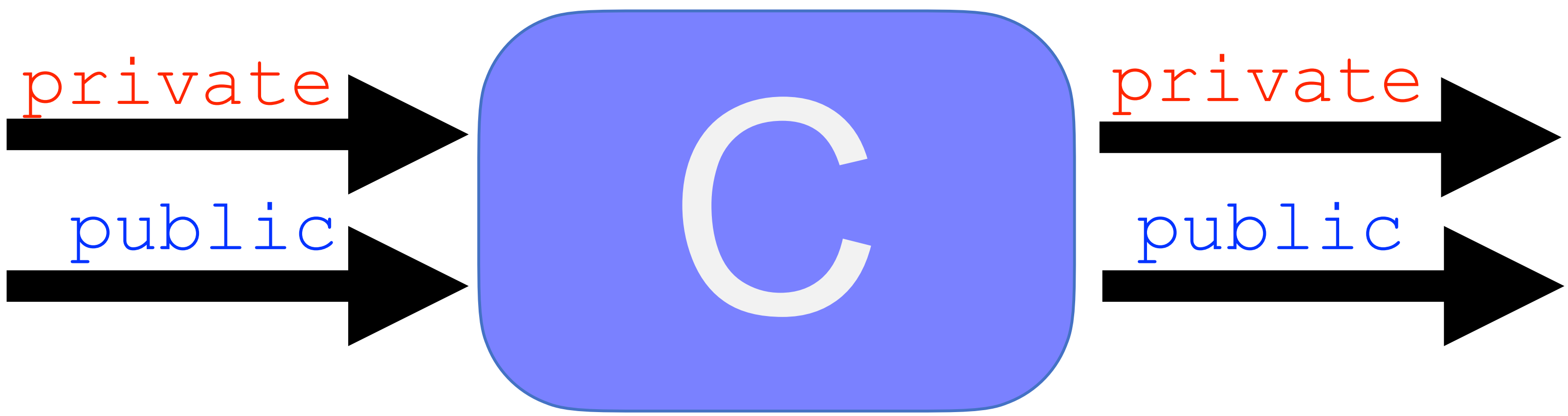
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$

Relational Property

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

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

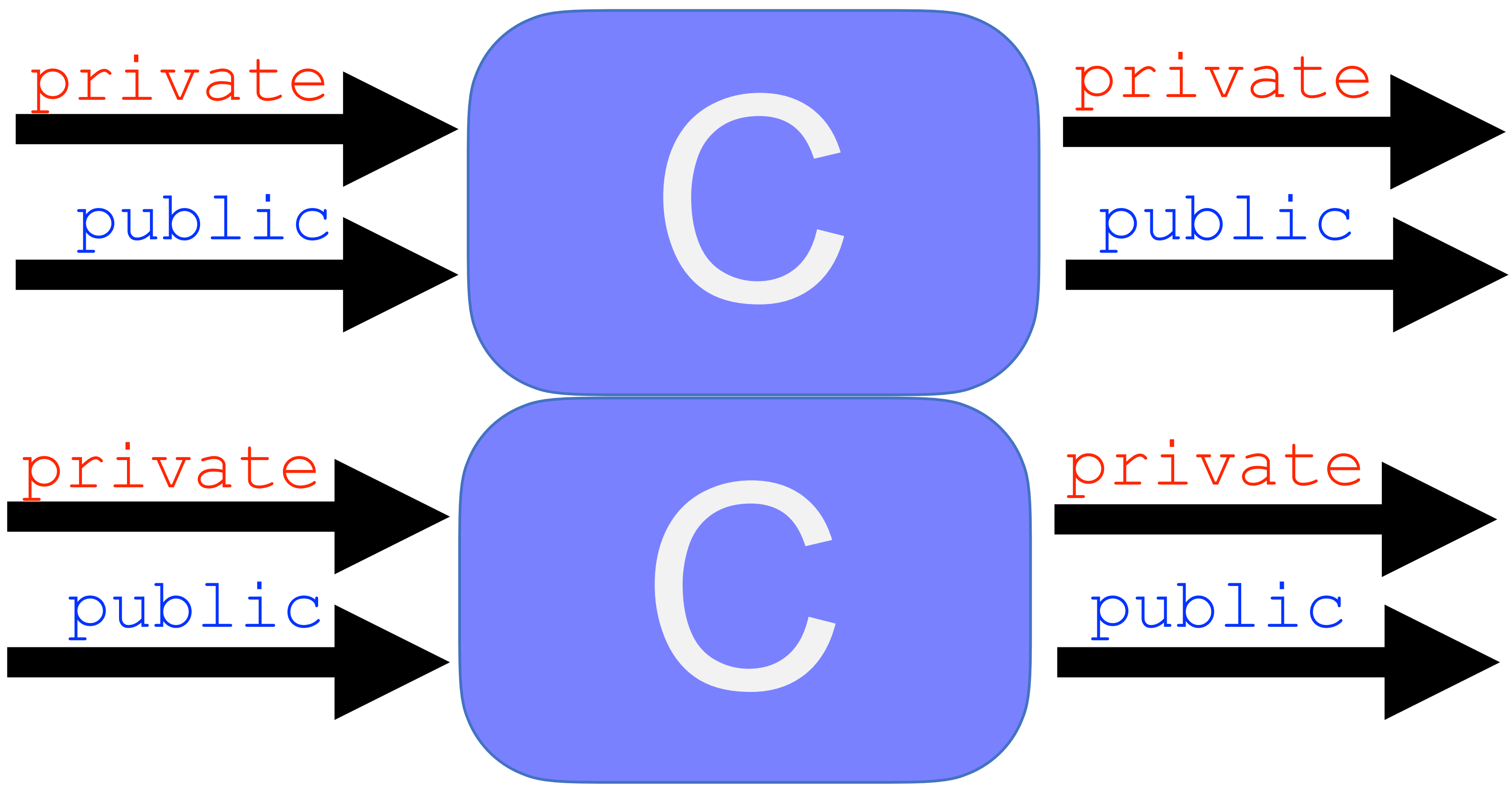
2) $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Property

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

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

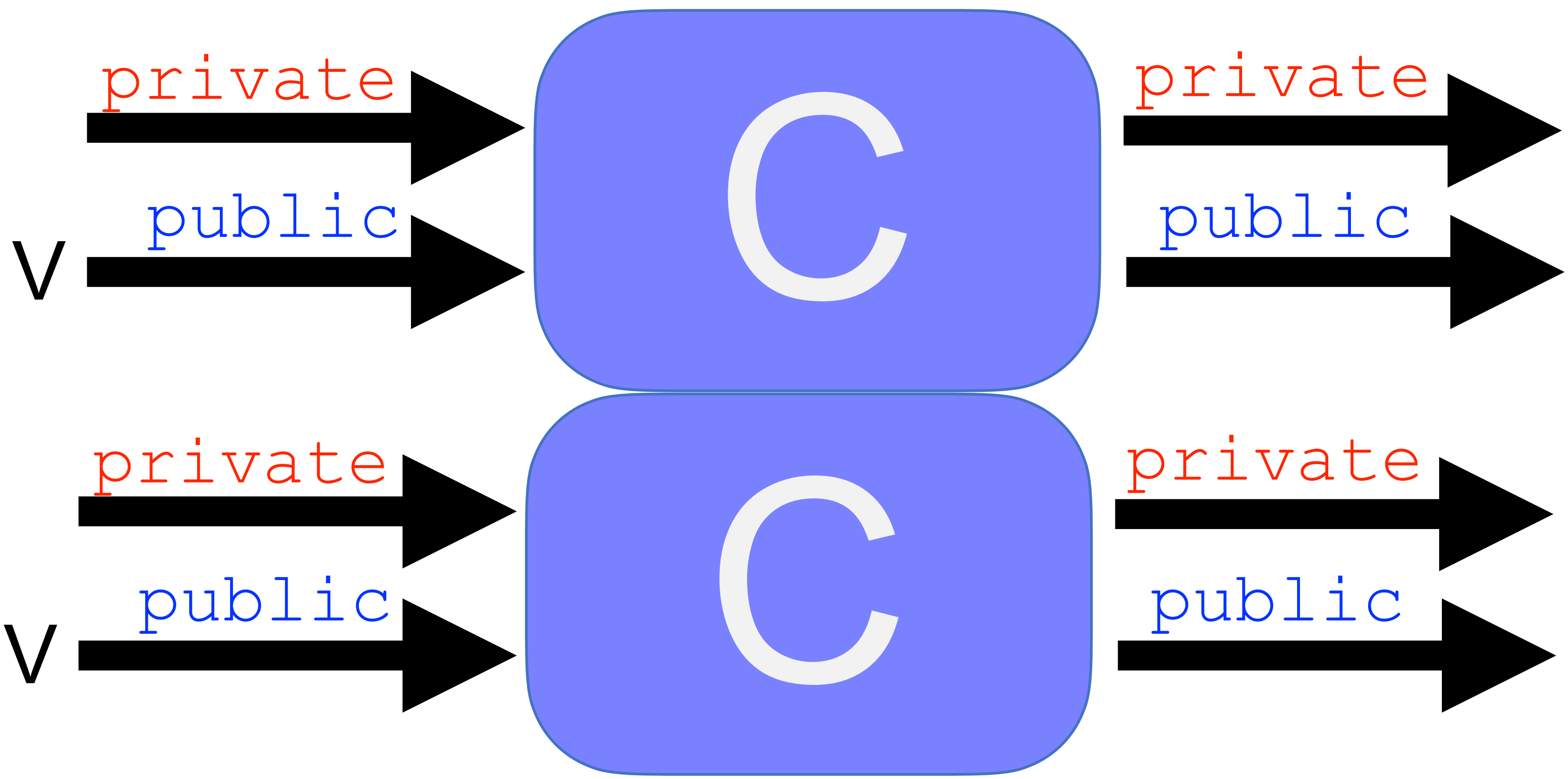


Relational Property

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

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

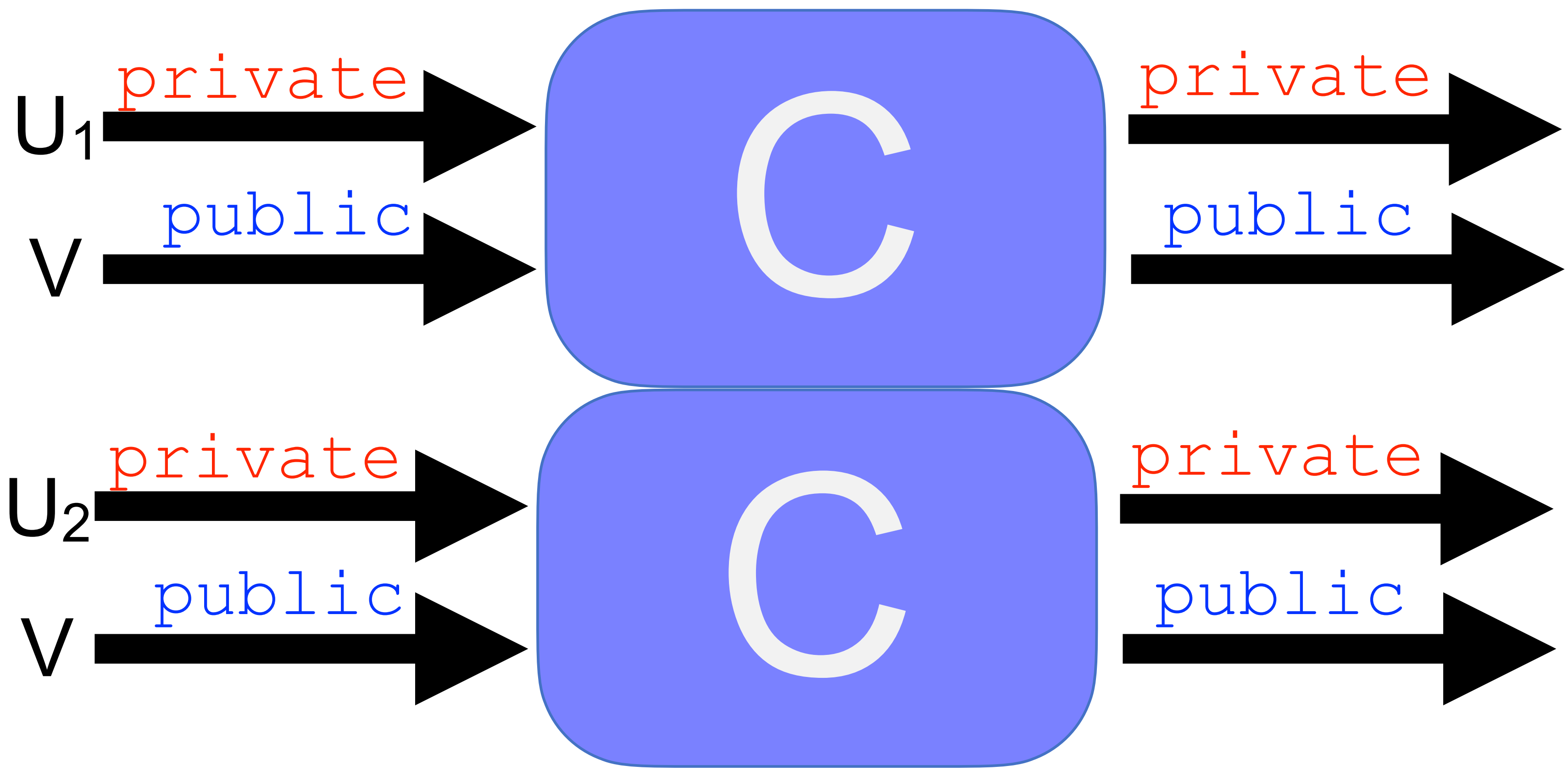
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Property

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

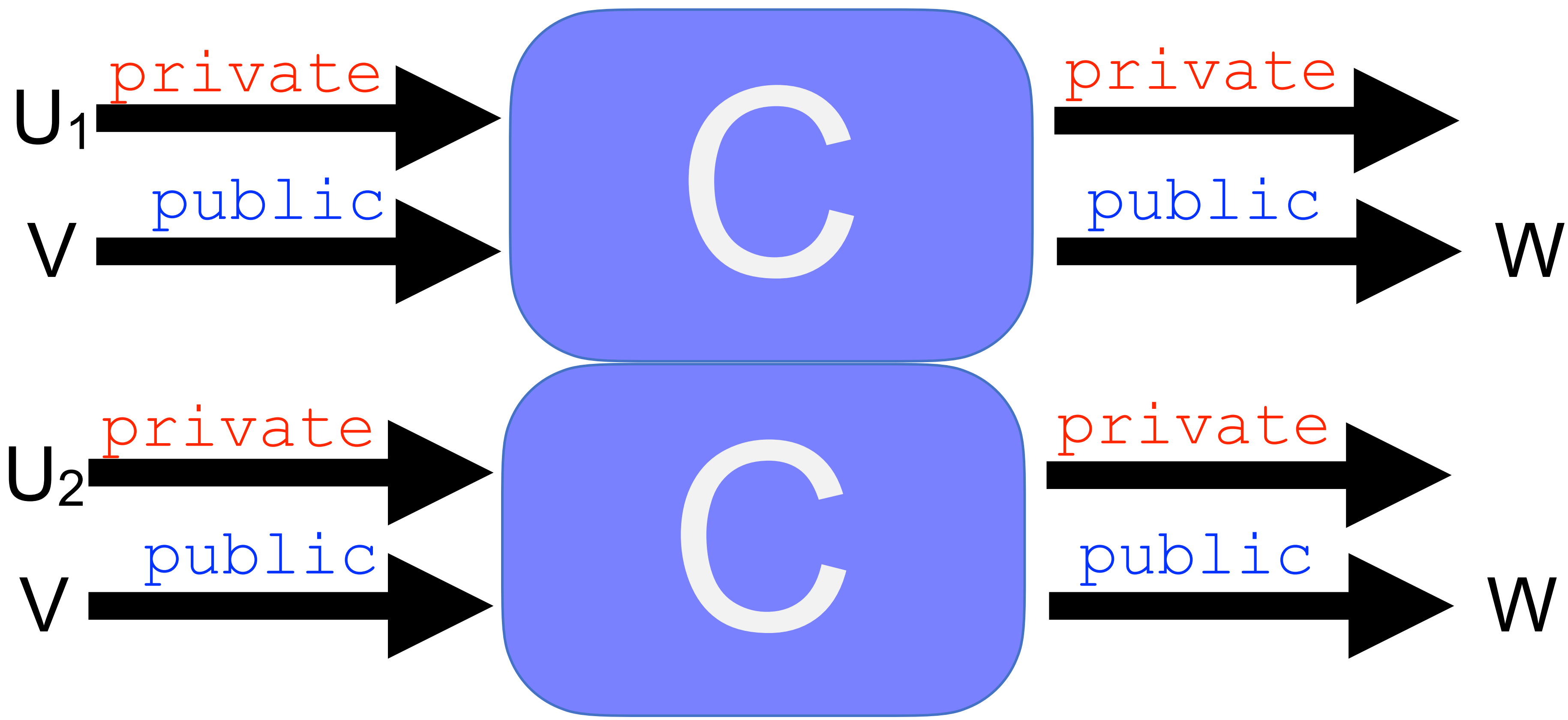
1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Property

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

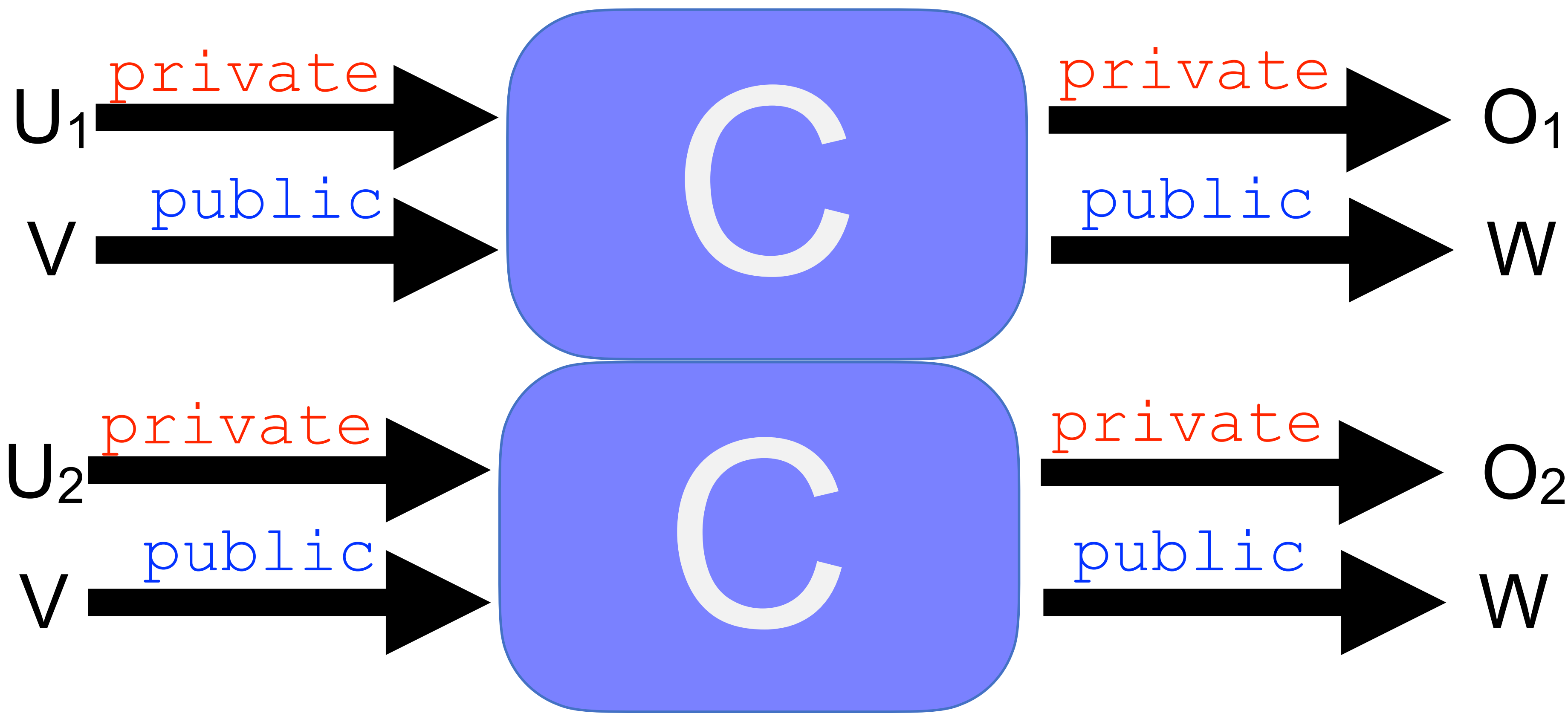
1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Property

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

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

Probabilistic Language

An example

```
OneTimePad(m : private msg) : public msg
  key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

Learning a ciphertext does not change any a priori knowledge about the likelihood of messages.

Probabilistic While (PWhile)

```
c::= abort
   | skip
   | x:= e
   | x:=$ d
   | c;c
   | if e then c else c
   | while e do c
```

$d_1, d_2, \ldots$ probabilistic expressions

Probabilistic Subdistributions

A discrete subdistribution over a set A is a function
$\mu : A \rightarrow [0, 1]$
such that the mass of μ,

$$|\mu| = \sum_{a \in A} \mu(a)$$

verifies $|\mu| \leq 1$.

The support of a discrete subdistribution μ,
$\text{supp}(\mu) = \{a \in A \mid \mu(a) > 0\}$
is necessarily countable, i.e. finite or countably infinite.

We will denote the set of sub-distributions over A by D(A), and say that μ is of type D(A) denoted μ:D(A) if $\mu \in D(A)$.

Semantics of Commands

This is defined on the structure of commands:

$\{\texttt{while e do c}\}_m = \sup_{n \in \texttt{Nat}} \mu_n$

$\mu_n = \texttt{let } m' = \{(\texttt{while}^n \texttt{ e do c})\}_m \texttt{ in } \{\texttt{if e then abort}\}_{m'}$

Semantics of Commands

This is defined on the structure of commands:

$\{\texttt{abort}\}_m = \mathbf{0}$

$\{\texttt{while e do c}\}_m = \sup_{n \in \mathrm{Nat}} \mu_n$

$\mu_n = \texttt{let } m' = \{(\texttt{while}^n \texttt{ e do c})\}_m \texttt{ in } \{\texttt{if e then abort}\}_{m'}$

Semantics of Commands

This is defined on the structure of commands:

$$\{\texttt{abort}\}_m = \mathbf{0}$$

$$\{\texttt{skip}\}_m = \texttt{unit}(m)$$

$$\{\texttt{while e do c}\}_m = \texttt{sup}_{n \in \texttt{Nat}}\ \mu_n$$

$$\mu_n = \texttt{let } m' = \{(\texttt{while}^n \texttt{ e do c})\}_m \texttt{ in } \{\texttt{if e then abort}\}_{m'}$$

Semantics of Commands

This is defined on the structure of commands:

$$\{\texttt{abort}\}_m = \mathbf{0}$$

$$\{\texttt{skip}\}_m = \texttt{unit}(m)$$

$$\{\texttt{x:=e}\}_m = \texttt{unit}(m[x \leftarrow \{e\}_m])$$

$$\{\texttt{while e do c}\}_m = \sup_{n \in Nat} \mu_n$$

$$\mu_n = \texttt{let } m' = \{(\texttt{while}^n \texttt{ e do c})\}_m \texttt{ in } \{\texttt{if e then abort}\}_{m'}$$

Semantics of Commands

This is defined on the structure of commands:

$\{\texttt{abort}\}_m = \mathbf{0}$

$\{\texttt{skip}\}_m = \texttt{unit}(m)$

$\{x{:=}e\}_m = \texttt{unit}(m[x \leftarrow \{e\}_m])$

$\{c;c'\}_m = \texttt{let } m' = \{c\}_m \texttt{ in } \{c'\}_{m'}$

$\{\texttt{while } e \texttt{ do } c\}_m = \sup_{n \in Nat} \mu_n$

$\mu_n = \texttt{let } m' = \{(\texttt{while}^n\ e \texttt{ do } c)\}_m \texttt{ in } \{\texttt{if } e \texttt{ then abort}\}_{m'}$

Semantics of Commands

This is defined on the structure of commands:

$\{abort\}_m = 0$

$\{skip\}_m = unit(m)$

$\{x:=e\}_m = unit(m[x \leftarrow \{e\}_m])$

$\{c;c'\}_m = let\ m' = \{c\}_m\ in\ \{c'\}_{m'}$

$\{if\ e\ then\ c_t\ else\ c_f\}_m = \{c_t\}_m$ If $\{e\}_m = true$

$\{while\ e\ do\ c\}_m = sup_{n \in Nat}\ \mu_n$

$\mu_n = let\ m' = \{(while^n\ e\ do\ c)\}_m\ in\ \{if\ e\ then\ abort\}_{m'}$

Semantics of Commands

This is defined on the structure of commands:

$\{abort\}_m = 0$

$\{skip\}_m = unit(m)$

$\{x{:=}e\}_m = unit(m[x \leftarrow \{e\}_m])$

$\{c;c'\}_m = let\ m'=\{c\}_m\ in\ \{c'\}_{m'}$

$\{if\ e\ then\ c_t\ else\ c_f\}_m = \{c_t\}_m$ If $\{e\}_m = true$

$\{if\ e\ then\ c_t\ else\ c_f\}_m = \{c_f\}_m$ If $\{e\}_m = false$

$\{while\ e\ do\ c\}_m = sup_{n \in Nat}\ \mu_n$

$\mu_n = let\ m'=\{(while^n\ e\ do\ c)\}_m\ in\ \{if\ e\ then\ abort\}_{m'}$

Semantics of Commands

This is defined on the structure of commands:

$\{abort\}_m = 0$

$\{skip\}_m = unit(m)$

$\{x:=e\}_m = unit(m[x \leftarrow \{e\}_m])$

$\{x:=\$\ d\}_m = let\ a=\{d\}_m\ in\ unit(m[x \leftarrow a])$

$\{c;c'\}_m = let\ m'=\{c\}_m\ in\ \{c'\}_{m'}$

$\{if\ e\ then\ c_t\ else\ c_f\}_m = \{c_t\}_m$ If $\{e\}_m = true$

$\{if\ e\ then\ c_t\ else\ c_f\}_m = \{c_f\}_m$ If $\{e\}_m = false$

$\{while\ e\ do\ c\}_m = sup_{n \in Nat}\ \mu_n$

$\mu_n = let\ m'=\{(while^n\ e\ do\ c)\}_m\ in\ \{if\ e\ then\ abort\}_{m'}$

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

Learning a ciphertext does not change any a priori knowledge about the likelihood of messages.

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

Learning a ciphertext does not change any a priori knowledge about the likelihood of messages.

How do we formalize this?

Probabilistic Noninterference

A program `prog` is probabilistically noninterferent if and only if, whenever we run it on two low equivalent memories m_1 and m_2 we have that the probabilistic distributions we get as outputs are the same on public outputs.

Low equivalence on distributions

Two distributions over memories μ_1 and μ_2 are low equivalent if and only if they coincide after projecting out all the private variables.

In symbols: $\mu_1 \sim_{low} \mu_2$

Example: Low equivalence on distributions

Consider memories with x private and y public. The distributions μ_1 and μ_2 defined as

μ_1([x=2,y=0])=2/3, μ_1([x=3,y=1])=1/3

and

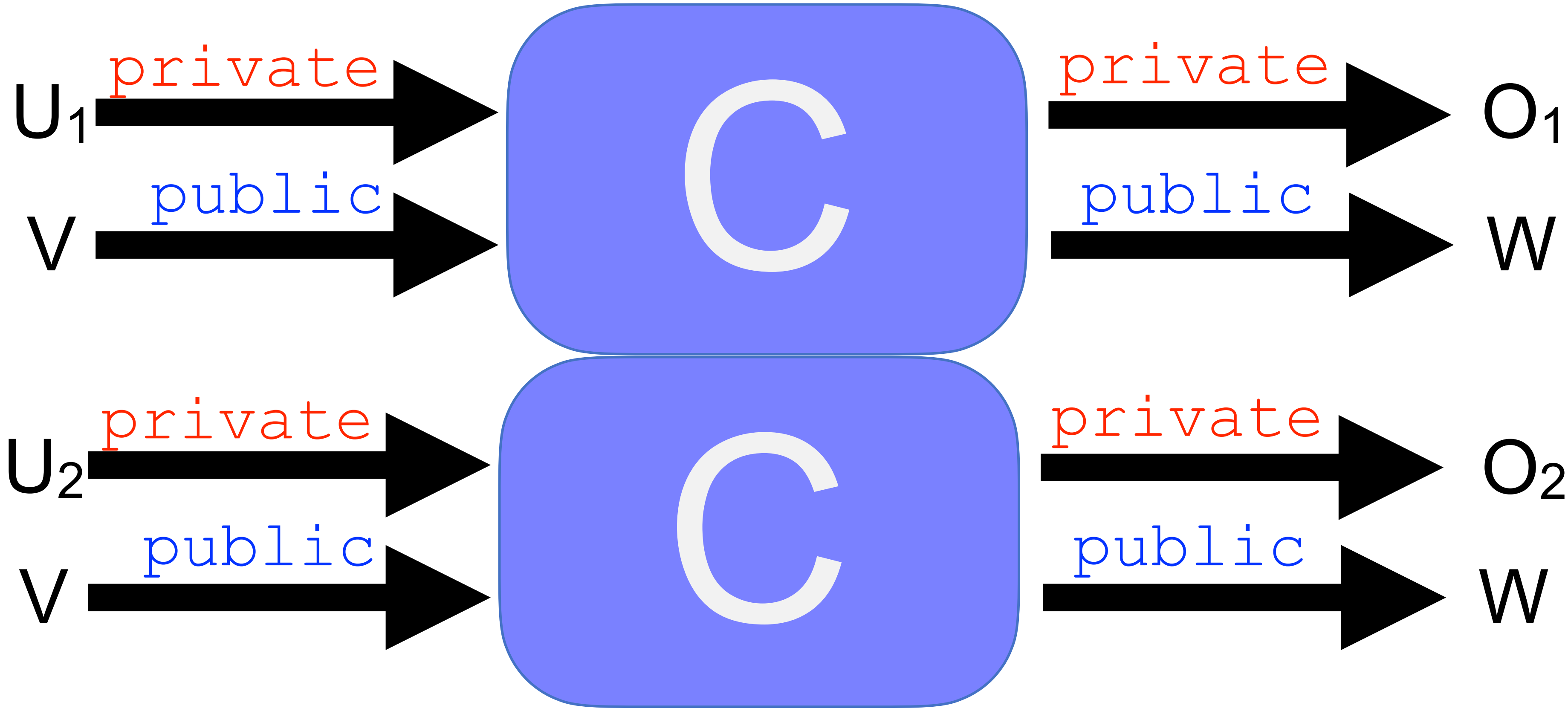
μ_2([x=1,y=0])=1/3, μ_2([x=5,y=0])=1/3, μ_2([x=4,y=1])=1/3

are low equivalent.

Noninterference as a Relational Property

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

$\{c\}_{m1} = \mu_1$ and $\{c\}_{m2} = \mu_2$ implies $\mu_1 \sim_{low} \mu_2$



Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
    cipher := msg xor key;
    return cipher
```

How can we prove that this is noninterferent?

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
    cipher := m xor key;
    return cipher
```

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

m_1 m_2

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}ⁿ);
   cipher := m xor key;
   return cipher
```

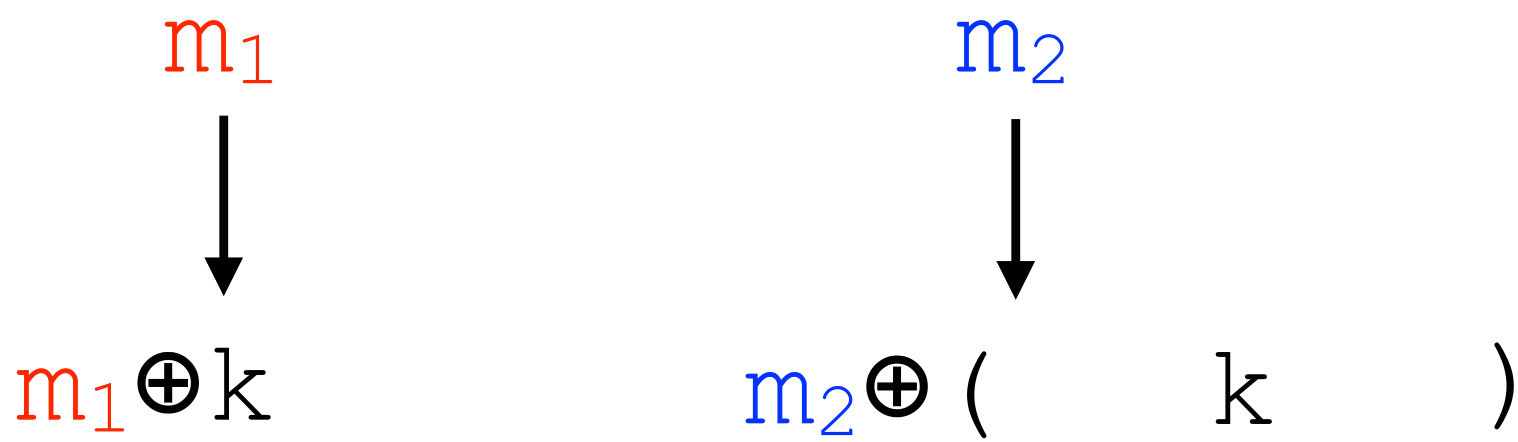
m_1 m_2

↓

$m_1 \oplus k$

Revisiting the example

```
OneTimePad(m : private msg) : public msg
  key :=$ Uniform({0,1}ⁿ);
  cipher := m xor key;
  return cipher
```



We will show it is sound to pick, with some restrictions, a *function* of k as the key for m_2. What could we choose so that the cipher texts are equal?

Revisiting the example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

$m_1 \longrightarrow m_1 \oplus k$

$m_2 \longrightarrow m_2 \oplus (m_1 \oplus k \oplus m_2)$

We will show it is sound to pick, with some restrictions, a *function* of k as the key for m_2. What could we choose so that the cipher texts are equal?

Properties of bitwise xor

$$c \oplus (a \oplus c) = a$$

Properties of bitwise xor

$$c \oplus (a \oplus c) = a$$

Example:

$$100 \oplus (101 \oplus 100) =$$

$$100 \oplus 001 = 101$$

Revisiting the example

```
OneTimePad(m : private msg) : public msg
  key :=$ Uniform({0,1}^n);
  cipher := m xor key;
  return cipher
```

m_1 → $m_1 \oplus k$

m_2 → $m_1 \oplus k$

Applying the property above

Probabilistic Relational Hoare Quadruples

Precondition

Program₁ ~ Program₂

Postcondition

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

Precondition

Probabilistic Program

Probabilistic Program

Postcondition

Validity of Probabilistic 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: $\{c_1\}_{m1} = \mu_1$ and $\{c_2\}_{m2} = \mu_2$ implies $Q(\mu_1, \mu_2)$.

Validity of Probabilistic 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: $\{c_1\}_{m1} = \mu_1$ and $\{c_2\}_{m2} = \mu_2$ implies $Q(\mu_1, \mu_2)$.

Is this correct?!?

Relational Assertions

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

logical formula
over pair of memories
(i.e., relation over memories)

logical formula
over ????

Relational Assertions

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

logical formula
over pair of memories
(i.e., relation over memories)

logical formula
over ????

We need to lift Q to be a relation on distributions, and we do this using the notion of a *coupling* between distributions

Coupling formally

Given two sub-distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(B)$, a coupling between them is a joint sub-distribution $\mu \in D(A \times B)$ whose *marginal sub-distributions* are μ_1 and μ_2, respectively.

$$\pi_1(\mu)(a) = \sum_b \mu(a,b) \qquad \pi_2(\mu)(b) = \sum_a \mu(a,b)$$

$$\pi_1(\mu) = \mu_1 \qquad \pi_2(\mu) = \mu_2$$

R-Coupling

Given two sub-distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(B)$, an R-coupling between them, for $R \subseteq A \times B$, is a joint sub-distribution $\mu \in D(A \times B)$ such that:

1) the marginal sub-distributions of μ are μ_1 and μ_2, respectively,
2) the support of μ is contained in R. That is, if μ`(a,b)>0, then (a,b)`$\in$R.

Coupling Example 1

relation

00	0.25
01	0.25
10	0.25
11	0.25

$\mathtt{k}_1$

$$\mathtt{k}_1 = \mathtt{10} \oplus \mathtt{k}_2 \oplus \mathtt{00}$$

00	0.25
01	0.25
10	0.25
11	0.25

$\mathtt{k}_2$

Coupling Example 1

00	0.25
01	0.25
10	0.25
11	0.25

k_1

relation

$$k_1 = 10 \oplus k_2 \oplus 00$$

k_2

00	0.25
01	0.25
10	0.25
11	0.25

k_2

k_1

	00	01	10	11
00			0.25	
01				0.25
10	0.25			
11		0.25		

Coupling Example 2

00	0.25
01	0.25
10	0.25
11	0.25

k_1

relation

$$k_1 = k_2 \text{ or } k_1 = k_2 \oplus 11$$

00	0.00
01	0.50
10	0.00
11	0.50

k_2

Coupling Example 2

00	0.25
01	0.25
10	0.25
11	0.25

k_1

relation

$$k_1 = k_2 \text{ or } k_1 = k_2 \oplus 11$$

00	0.00
01	0.50
10	0.00
11	0.50

k_2

k_2

k_1

	00	01	10	11
00				0.25
01		0.25		
10		0.25		
11				0.25

Relational lifting of a predicate

We say that two sub-distributions $\mu_1 \in D(A)$ and $\mu_2 \in D(B)$ are in the relational lifting of the relation $R \subseteq A \times B$, denoted μ_1 R* μ_2, if and only if there exists a sub-distribution $\mu \in D(A \times B)$ such that:

1) if μ(a,b)>0, then (a,b)∈R.

2) $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$

Relational lifting of a predicate

We say that two sub-distributions $\mu_1 \in D(A)$ and $\mu_2 \in D(B)$ are in the relational lifting of the relation $R \subseteq A \times B$, denoted $\mu_1 \; R^* \; \mu_2$, if and only if there exists a sub-distribution $\mu \in D(A \times B)$ such that:

1) if $\mu(a,b) > 0$, then $(a,b) \in R$.
2) $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$

I.e., there is an R-coupling for μ_1 and μ_2

Consequences of Lifting

Consequences of Lifting

Suppose E and F are predicates on memories. If we know

$\mu_1 \; (E<1> \; <=> \; F<2>)^{*} \; \mu_2$,

then we can conclude that

$$\Pr_{\mu 1}[E] \qquad \Pr_{\mu 2}[F]$$

Consequences of Lifting

Suppose E and F are predicates on memories. If we know

μ_1 (E<1> <=> F<2>)* μ_2,

then we can conclude that

$$\Pr_{\mu_1}[E] = \Pr_{\mu_2}[F]$$

Consequences of Lifting

Consequences of Lifting

Suppose E and F are predicates on memories. If we know

μ_1 (E<1> => F<2>)* μ_2,

then we can conclude that

$Pr_{\mu_1}[E] \quad Pr_{\mu_2}[F]$

Consequences of Lifting

Suppose E and F are predicates on memories. If we know

μ_1 (E<1> => F<2>)* μ_2,

then we can conclude that

$$\Pr_{\mu 1}[E] \leq \Pr_{\mu 2}[F]$$

Validity of Probabilistic 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: $\{c_1\}_{m1} = \mu_1$ and $\{c_2\}_{m2} = \mu_2$ implies $Q^*(\mu_1, \mu_2)$.

Probabilistic Relational Hoare Logic

Skip

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

Probabilistic Relational Hoare Logic

Assignment

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

Probabilistic Relational Hoare Logic

Composition

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

Probabilistic 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.

Probabilistic Relational Hoare Logic

If-then-else

$$P \Rightarrow (e_1\langle 1\rangle \Leftrightarrow e_2\langle 2\rangle)$$

$$\vdash c_1 \sim c_2 : e_1\langle 1\rangle \land P \Rightarrow Q$$

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

$$\frac{}{\vdash \begin{array}{c}\texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2'\end{array} : P \Rightarrow Q}$$

Probabilistic Relational Hoare Logic

While

$$P \Rightarrow (e_1\langle 1\rangle \Leftrightarrow e_2\langle 2\rangle)$$

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

Probabilistic Relational Hoare Logic

If-then-else - left

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

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

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

Probabilistic Relational Hoare Logic

If-then-else - right

$$\frac{\vdash c_1 \sim c_2 : e\langle 2\rangle \wedge P \Rightarrow Q \qquad \vdash c_1 \sim c_2' : \neg e\langle 2\rangle \wedge P \Rightarrow Q}{\vdash \begin{matrix} c_1 \\ \sim \\ \texttt{if } e \texttt{ then } c_2 \texttt{ else } c_2' \end{matrix} : P \Rightarrow Q}$$

Probabilistic Relational Hoare Logic

Assignment - left

$$\vdash \texttt{x:=e} \sim \texttt{skip}: \texttt{P[e<1>/x<1>]} \Rightarrow \texttt{P}$$

How about the random assignment?

Probabilistic Relational Hoare Logic
Random Assignment

$$\vdash \texttt{x}_1 \texttt{ :=\$ d}_1 \sim \texttt{x}_2 \texttt{ :=\$ d}_2 \texttt{ : ?? } \Rightarrow \texttt{Q}$$

We would like to have:

$$\frac{\texttt{for all } m_1,\ m_2,\ P(m_1,m_2) \Rightarrow \texttt{let } a=\{d_1\}_{m1} \texttt{ in unit}(m_1[x_1\leftarrow a])\ Q^*\ \texttt{let } a=\{d_2\}_{m2} \texttt{ in unit}(m_2[x_2\leftarrow a])}{\vdash x_1 := \$\ d_1 \sim x_2 := \$\ d_2 : P \Rightarrow Q}$$

We would like to have:

$$\frac{\texttt{for all } m_1,\ m_2,\ P(m_1,m_2) \Rightarrow \texttt{let } a=\{d_1\}_{m1} \texttt{ in unit}(m_1[x_1 \leftarrow a])\ Q^*\ \texttt{let } a=\{d_2\}_{m2} \texttt{ in unit}(m_2[x_2 \leftarrow a])}{\vdash x_1 := \$\ d_1 \sim x_2 := \$\ d_2 : P \Rightarrow Q}$$

What is the problem with this rule?

Restricted Probabilistic Expressions

We consider a restricted set of expressions denoting probability distributions.

```
d::= f(d₁,…,dₖ)
```

Where `f` is a distribution declaration

Some expression examples similar to the previous

`uniform({0,1}^128)` `bernoulli(.5)` `laplace(0,1)`

Restricted Probabilistic Expressions

We consider a restricted set of expressions denoting probability distributions.

```
d::= f(d1,…,dk)
```

Where `f` is a distribution declaration

Some expression examples similar to the previous

`uniform({0,1}^128)` `bernoulli(.5)` `laplace(0,1)`

Notice that we don't need a memory anymore to interpret them

Isomorphisms on Sub-distributions

Given two sub-distributions $\mu_1 \in D(A)$ and $\mu_2 \in D(B)$, we say that a mapping $h: A \rightarrow B$ is an *isomorphism* between μ_1 and μ_2 ($h \lhd (\mu_1, \mu_2)$) if and only iff:

1) h is a bijective map between elements in supp(μ_1) and supp(μ_2),
2) for all $a \in A$, $\mu_1(a) = \mu_2(h(a))$

Probabilistic Relational Hoare Logic
Random Assignment

$$P = h \triangleleft (d_1, d_2) \wedge$$
$$\quad \forall v,$$
$$\quad v \in \mathrm{supp}(d_1)$$
$$\quad \Rightarrow$$
$$\quad Q[v/x_1\langle 1\rangle, h(v)/x_2\langle 2\rangle]$$

we let h's definition refer to program variables tagged with <1> or <2>

$$\vdash x_1 :=\$\ d_1 \sim x_2 :=\$\ d_2 : P \Rightarrow Q$$

Back to our example

```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := m xor key;
   return cipher
```

m<1> → m<1>⊕k

m<2> → m<2>⊕(m<1>⊕k⊕m<2>)

Back to our example

d_1=Uniform({0,1}^{n}) d_2=Uniform({0,1}^{n})

$$h(k) = (m\text{<}1\text{>} \oplus k \oplus m\text{<}2\text{>})$$

Is this an isomorphism from d_1 to d_2?

1) Is it bijective between elements in the support of d_1 and d_2?
2) Is it true that for all $v \in \{0,1\}^n$, $d_1(v) = d_2(h(v))$?

Back to our example

d_1=Uniform($\{0,1\}^n$) d_2=Uniform($\{0,1\}^n$)

$$h(k) = (m\langle 1\rangle \oplus k \oplus m\langle 2\rangle)$$

Is this an isomorphism from d_1 to d_2?

1) Is it bijective between elements in the support of d_1 and d_2?
2) Is it true that for all $v \in \{0,1\}^n$, $d_1(v) = d_2(h(v))$?

Yes, it's an isomorphism!

Back to our example

$$h(k)=m<1>\oplus k\oplus m<2>,\quad d_1 = d_2 = Uniform(\{0,1\}^n)$$

$$P = h\lhd(d_1,d_2)\wedge$$

$$\forall v, v\in support(d_1) \Rightarrow$$

$$m<1>\oplus k_1<1>=m<2>\oplus k_2<2>[v/k_1<1>, h(v)/k_2<2>]$$

$$\vdash k_1 := \$ Uniform(\{0,1\}^n) \sim k_2 := \$ Uniform(\{0,1\}^n) :$$

$$P \Rightarrow m<1>\oplus k_1<1>=m<2>\oplus k_2<2>$$

Back to our example

$h(k)=m<1>\oplus k\oplus m<2>,\ d_1 = d_2 = \mathtt{Uniform}(\{0,1\}^n)$

$P = h \lhd (d_1,d_2) \land$

$\forall v, v \in \mathtt{support}(d_1) \Rightarrow$

$m<1>\oplus v = m<2>\oplus(m<1>\oplus v\oplus m<2>)$

$\vdash k_1 := \$\mathtt{Uniform}(\{0,1\}^n) \sim k_2 := \$\mathtt{Uniform}(\{0,1\}^n):$

$P \Rightarrow m<1>\oplus k_1<1> = m<2>\oplus k_2<2>$

Back to our example

$P = h\lhd(d_1,d_2) \wedge \forall v, v \in support(d_1) \Rightarrow$
$m\langle 1\rangle \oplus v = m\langle 2\rangle \oplus (m\langle 1\rangle \oplus v \oplus m\langle 2\rangle)$

$True \Rightarrow P$

$\vdash k_1 := \$Uniform(\{0,1\}^n) \sim k_2 := \$Uniform(\{0,1\}^n) :$
$P \Rightarrow m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle$

$\vdash k_1 := \$Uniform(\{0,1\}^n) \sim k_2 := \$Uniform(\{0,1\}^n) :$
$True \Rightarrow m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle$

Back to our example

$P = h \lhd (d_1, d_2) \wedge \forall v, v \in support(d_1) \Rightarrow m\langle 1\rangle \oplus v = m\langle 2\rangle \oplus (m\langle 1\rangle \oplus v \oplus m\langle 2\rangle)$

$True \Rightarrow P$

$$\vdash k_1 := \$ Uniform(\{0,1\}^n) \sim k_2 := \$ Uniform(\{0,1\}^n) : P \Rightarrow m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle$$

$$\vdash k_1 := \$ Uniform(\{0,1\}^n) \sim k_2 := \$ Uniform(\{0,1\}^n) : True \Rightarrow m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle$$

By Consequence

Soundness

If we can derive $\vdash c_1 \sim c_2 : P \Rightarrow Q$ through the rules of the logic, then the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid.

Completeness?