CS 591: Formal Methods in Security and Privacy
Probabilistic computations

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**.
Noninterference as a Relational Property

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

1) \( \{c\}_{m_1} = \bot \) iff \( \{c\}_{m_2} = \bot \)

2) \( \{c\}_{m_1} = m_1' \) and \( \{c\}_{m_2} = m_2' \) implies \( m_1' \sim_{\text{low}} m_2' \)
Relational Hoare Quadruples

\[ c_1 \sim c_2 : P \Rightarrow Q \]

Precondition (a logical formula)

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) \( \{c_1\}_{m_1} = \bot \iff \{c_2\}_{m_2} = \bot \)

2) \( \{c_1\}_{m_1} = m_1' \) and \( \{c_2\}_{m_2} = m_2' \) implies \( Q(m_1', m_2') \).

How do we check this?
Which rules do we need to prove this?

\[
x: \text{private} \\
y: \text{public} \\
y := x; y := 5
\]

\[
\quad\Rightarrow\quad =_{\text{low}} \implies =_{\text{low}}
\]
Rules of Relational Hoare Logic

Assignment

\[ \vdash x_1 := e_1 \sim x_2 := e_2 : \]
\[ P[e_1<1>/x_1<1>, e_2<2>/x_2<2>] \Rightarrow P \]
Rules of Relational Hoare Logic

Consequence

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 $R$. 

\[
\begin{align*}
P & \Rightarrow S \\
\vdash C_1 \sim C_2 : S & \Rightarrow R \\
R & \Rightarrow Q \\
\vdash C_1 \sim C_2 : P & \Rightarrow Q
\end{align*}
\]
Rules of Relational Hoare Logic
Composition

\[ \vdash c_1 \sim c_2 : P \Rightarrow R \quad \vdash c_1' \sim c_2' : R \Rightarrow S \]

\[ \vdash c_1 ; c_1' \sim c_2 ; c_2' : P \Rightarrow S \]
Which rules do we need to prove this?

x: private
y: public

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

\[ := \text{low} \Rightarrow := \text{low} \]
Rules of Relational Hoare Logic

If-then-else

\[ P \Rightarrow (e_1 < 1> \iff e_2 < 2>) \]

\[ \vdash c_1 \sim c_2 : e_1 < 1> \land P \Rightarrow Q \]

\[ \vdash c_1' \sim c_2' : \neg e_1 < 1> \land P \Rightarrow Q \]

if \( e_1 \) then \( c_1 \) else \( c_1' \)

\[ \vdash \sim : P \Rightarrow Q \]

if \( e_2 \) then \( c_2 \) else \( c_2' \)
Which rules do we need to prove this?

s1: public
s2: private
r: private
i: public
n: public

proc Compare (s1:list[n] bool, s2:list[n] bool)
i:=0;
r:=0;
while i< n do
    if not(s1[i]=s2[i]) then
        r:=1
    i:=i+1
:n>0 /
\ =low ⇒ =low
Rules of Relational Hoare Logic
While

\[ P \Rightarrow (e_1<1> \iff e_2<2>) \]

\[ \vdash \neg c_1 \land c_2 : e_1<1> \land P \Rightarrow P \]

\[ \vdash \neg \text{while } e_1 \text{ do } c_1 \]

\[ \vdash \neg \text{while } e_2 \text{ do } c_2 \]

Invariant
Rules of Relational Hoare Logic
If-then-else - left

\[ \vdash c_1 \sim c_2 : e{<1} \land P \implies Q \]
\[ \vdash c_1' \sim c_2 : \neg e{<1} \land P \implies Q \]

\[ \vdash \quad \sim \quad : P \Rightarrow Q \]

\[
\begin{align*}
\text{if } e \text{ then } c_1 \text{ else } c_1' \\
\text{C}_2
\end{align*}
\]
Rules of Relational Hoare Logic

If-then-else - right

\[ \vdash c_1 \sim c_2' : \neg e \langle 2 \rangle \land P \Rightarrow Q \]

\[ \vdash c_1 \sim c_2 : e \langle 2 \rangle \land P \Rightarrow Q \]

\[ \vdash c_1 \sim c_2' : \neg e \langle 2 \rangle \land P \Rightarrow Q \]

\[ \vdash \quad c_1 \quad \sim \quad : P \Rightarrow Q \]

\[ \quad \text{if } e \text{ then } c_2 \text{ else } c_2' \]
Rules of Relational Hoare Logic
Assignment - left

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

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

If a quadruple $c_1 \sim c_2 : P \implies Q$ is valid, and we have an oracle to derive all the true statements of the form $P \implies S$ and of the form $R \implies Q$, then we can derive $\vdash c_1 \sim c_2 : P \implies Q$ through the rules of the logic.
Soundness and completeness with respect to Hoare Logic

\[ \vdash_{\text{RHL}} c_1 \sim c_2 : P \Rightarrow Q \]

iff

\[ \vdash_{\text{HL}} c_1 ; c_2 : P \Rightarrow Q \]

Under the assumption that we can partition the memory adequately, and that we have termination.
Possible projects

In Easycrypt
• Look at how to guarantee trace-based noninterference.
• Look at how to guarantee side-channel free noninterference.
• Look at the relations between self-composition and relational logic.

Not related to Easycrypt
• Look at type systems for non-interference.
• Look at other methods for relational reasoning
• Look at declassification
Today: Probabilistic Language
An example

```
OneTimePad(m : private msg) : public msg
key ::= Uniform({0,1}^n);
cipher ::= msg 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, ... probabilistic expressions
Probabilistic Expressions

We extend the language with expression describing probability distributions.

$$d ::= f(e_1, \ldots, e_n, d_1, \ldots, d_k)$$

Where $f$ is a distribution declaration

Some expression examples

uniform($\{0,1\}^n$)  gaussian($k,\sigma$)  laplace($k,b$)
Semantics of Probabilistic Expressions

We would like to define it on the structure:

\[ \{ f(e_1, \ldots, e_n, d_1, \ldots, d_k) \}_m = \{ f \}(\{ e_1 \}_m, \ldots, \{ e_n \}_m, \{ d_1 \}_m, \ldots, \{ d_k \}_m) \]

but is the result just a value?
A discrete subdistribution over a set $A$ is a function $\mu : A \rightarrow [0, 1]$ such that the mass of $\mu$, 

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

verifies $|\mu| \leq 1$.

The support of a discrete subdistribution $\mu$, $\text{supp}(\mu) = \{a \in A \mid \mu(a) > 0\}$ 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 $\mu$ is of type $D(A)$ denoted $\mu:D(A)$ if $\mu \in D(A)$. 
Probabilistic Subdistributions

We call a subdistribution with mass exactly 1, a distribution.

We define the probability of an event \( E \subseteq A \) with respect to the subdistribution \( \mu : \mathcal{D}(A) \) as

\[
\mathbb{P}_\mu[E] = \sum_{a \in E} \mu(a)
\]
Probabilistic Subdistributions

Let's consider $\mu \in \mathcal{D}(\mathcal{A})$, and $E \subseteq \mathcal{A}$, we have the following properties

$$P_\mu[\emptyset] = 0$$

$$P_\mu[A] \leq 1$$

$$0 \leq P_\mu[E] \leq 1$$

$E \subseteq F \subseteq \mathcal{A}$ implies $P_\mu[E] \leq P_\mu[F]$.

$E \subseteq \mathcal{A}$ and $F \subseteq \mathcal{A}$ implies $P_\mu[E \cup F] \leq P_\mu[E] + P_\mu[F] - P_\mu[E \cap F]$.

We will denote by $\mathbf{O}$ the subdistribution $\mu$ defined as constant 0.
Operations over Probabilistic Subdistributions

Let’s consider an arbitrary $a \in A$, we will often use the distribution $\text{unit}(a)$ defined as:

$$\mathbb{P}_{\text{unit}(a)}[\{b\}] = \begin{cases} 1 \text{ if } a=b \\ 0 \text{ otherwise} \end{cases}$$

We can think about $\text{unit}$ as a function of type $\text{unit}: A \rightarrow D(A)$
Operations over Probabilistic Subdistributions

Let’s consider a distribution $\mu \in D(A)$, and a function $M : A \rightarrow D(B)$ then we can define their composition by means of an expression $\text{let } a = \mu \text{ in } M(a)$ defined as:

$$\mathbb{P}_{\text{let } a = \mu \text{ in } M(a)}[E] = \sum_{a \in \text{supp}(\mu)} \mathbb{P}_\mu[\{a\}] \cdot \mathbb{P}_{(Ma)}[E]$$
Semantics of Probabilistic Expressions - revisited

We would like to define it on the structure:

\[ \{ f(e_1, \ldots, e_n, d_1, \ldots, d_k) \}_m = \{ f \}(\{ e_1 \}_m, \ldots, \{ e_n \}_m, \{ d_1 \}_m, \ldots, \{ d_k \}_m) \]

With input a memory \( m \) and output a subdistribution \( \mu \in D(A) \) over the corresponding type \( A \). E.g.

\[ \{ \text{uniform}({0,1}^n) \}_m \in D({0,1}^n) \]

\[ \{ \text{gaussian}(k, \sigma) \}_m \in D(\text{Real}) \]