CS 591: Formal Methods in Security and Privacy

Approximate probabilistic relational Hoare Logic

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Q&A

To increase interactivity, I will ask more question to each one of you.

It is not a test, you can always answer “pass!”

Projects

Everyone should have a project now.

Please, don’t hesitate in contacting us if there is some issue with your project.

Recording

This is a reminder that we will record the class and we will post the link on Piazza.

This is also a reminder to myself to start recording!

From the previous classes

An example

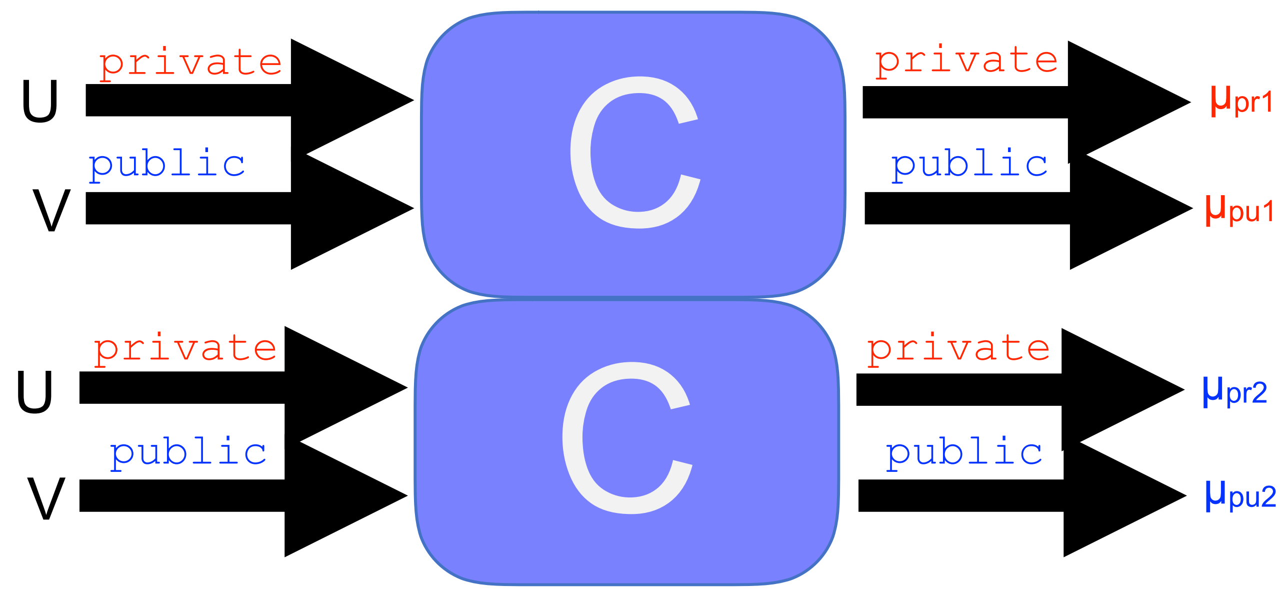
```
OneTimePad(m : private msg) : public msg
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

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

Probabilistic Noninterference as a Relational Property

c is probabilistically 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
```

m_1 m_2

$m_1 \oplus k$

Revisiting the example

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

m_1 m_2

$$\begin{array}{c} m_1 \\ \downarrow \\ m_1 \oplus k \end{array}$$

Suppose we can now chose the key for m_2. What could we choose?

Revisiting the example

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

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

Suppose we can now chose the key for m_2. What could we choose?

Revisiting the example

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

$$
\begin{tikzcd}
m_1 \arrow[d] & m_2 \arrow[d] \\
m_1 \oplus k & m_1 \oplus k
\end{tikzcd}
$$

Probabilistic Relational Hoare Quadruples

Precondition

Program$_1$ ~ Program$_2$

Postcondition

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

Precondition (a logical formula)

Probabilistic Program

Probabilistic Program

Postcondition (a logical formula)

R-Coupling

Given two 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 distribution $\mu \in D(A \times B)$ such that:

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

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 $Q^*(\mu_1, \mu_2)$.

Consequences of Coupling

Given the following pRHL judgment

$$\vdash c_1 \sim c_2 : \text{True} \Rightarrow Q$$

We have that:

if $Q \Rightarrow (R\langle 1\rangle \iff S\langle 2\rangle)$, then $\Pr[c_1 : R] = \Pr[c_2 : S]$

if $Q \Rightarrow (R\langle 1\rangle \Rightarrow S\langle 2\rangle)$, then $\Pr[c_1 : R] \leq \Pr[c_2 : S]$

A sufficient condition for R-Coupling

Given two distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(B)$, and a relation $\mathtt{R} \subseteq A\mathrm{x}B$, if there is a mapping $h:A \rightarrow B$ such that:

1) h is a bijective map between elements in supp(μ_1) and supp(μ_2),
2) for every $a \in$supp(μ_1), $(a,h(a)) \in \mathtt{R}$
3) $\mathtt{Pr}_{\mathtt{x} \sim \mu 1}[\mathtt{x=a}] = \mathtt{Pr}_{\mathtt{x} \sim \mu 2}[\mathtt{x=h(a)}]$

Then, there is an R-coupling between μ_1 and μ_2.
We write $h \lhd (\mu_1, \mu_2)$ in this case.

Probabilistic Relational Hoare Logic
Random Assignment

$$\frac{\begin{array}{l} h \lhd (\{d_1\},\{d_2\}) \\ P = \forall v, v \in supp(\{d_1\}) \\ \Rightarrow Q[v/x_1\langle 1\rangle, h(v)/x_2\langle 2\rangle] \end{array}}{\vdash x_1 := \$\ d_1 \sim x_2 := \$\ d_2 : P \Rightarrow Q}$$

Back to our example

$$h(k) = (m\langle 1\rangle \oplus k \oplus m\langle 2\rangle) \lhd (\{d_1\}, \{d_2\})$$

$$P = \forall k, k \in \{0,1\}^n$$

$$\Rightarrow\ m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle [v/k_1\langle 1\rangle, h(v)/k_2\langle 2\rangle] =$$

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

Back to our example

$$h(k) = (m\langle 1\rangle \oplus k \oplus m\langle 2\rangle) \lhd (\{d_1\}, \{d_2\})$$

$$P = \forall k, k \in \{0,1\}^n$$

$$\Rightarrow \; m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle [v/k_1\langle 1\rangle, h(v)/k_2\langle 2\rangle] =$$

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

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

$$True \;\Rightarrow\; m\langle 1\rangle \oplus k_1\langle 1\rangle = m\langle 2\rangle \oplus k_2\langle 2\rangle$$

Using the assignment rule, we can conclude.

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?

Today:
approximate probabilistic noninterference

One time pad

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

What are the drawbacks of one time pads?

A more realistic example

```
StreamCipher(m : private msg[n]) : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

What guarantees do we want from a PRG?

Properties of PRG

We would like the PRG to increase the number of random bits but also to guarantee the result to be (almost) random.

We can express this as:

$$\boxed{\text{PRG}: \{0,1\}^k \rightarrow \{0,1\}^n \text{ for } n>k}$$

$$\boxed{\text{PRG}(\text{Uniform}(\{0,1\}^k) \approx \text{Uniform}(\{0,1\}^n}$$

How can we measure the similarity between the result of PRG and the uniform distribution?

Statistical distance

We say that two distributions $\mu_1, \mu_2 \in D(A)$, are at statistical distance δ if and only if:

$$\Delta(\mu 1,\mu 2)=\max_{E\subseteq A} \mid Pr_{x\sim\mu 1}[x\in E]-Pr_{x\sim\mu 2}[x\in E] \mid = \delta$$

Properties of PRG

We would like the PRG to increase the number of random bits but also to guarantee the result to be (almost) random.

We can express this as:

$$\boxed{\mathrm{PRG}: \{0,1\}^k \rightarrow \{0,1\}^n \text{ for } n>k}$$

$$\boxed{\Delta(\mathrm{PRG}(\mathrm{Uniform}(\{0,1\}^k),\mathrm{Uniform}(\{0,1\}^n) \leq 2^{-n}}$$

In fact this is a too strong requirement - usually we require that every polynomial time adversary cannot distinguish the two distributions in statistical distance

How can we prove this secure?

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

~

```
StreamCipher(m : private msg[n])
             : public msg[n]
   pkey :=$ PRG(Uniform({0,1}ᵏ));
   cipher := msg xor pkey;
   return cipher
```

How can we prove this secure?

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

~

```
StreamCipher(m : private msg[n])
             : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

m

m

How can we prove this secure?

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

~

```
StreamCipher(m : private msg[n])
              : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

m

$m \oplus k$

m

How can we prove this secure?

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

~

```
StreamCipher(m : private msg[n])
              : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

$$m \longrightarrow m \oplus k$$

$$m \longrightarrow m \oplus pk$$

How can we prove this secure?

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

~

```
StreamCipher(m : private msg[n])
            : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

m   m

$$\Delta(\, m \oplus k \;,\; m \oplus pk \,) \le \delta$$

How to reason formally about this formally?

Approximate Probabilistic Relational Hoare Logic

Indistinguishability parameter

Precondition (a logical formula)

$$\vdash_{\delta} c_1 \sim c_2 : P \Rightarrow Q$$

Probabilistic Program

Probabilistic Program

Postcondition (a logical formula)

How can we define validity?

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 $Q^*(\mu_1, \mu_2)$.