CS 599: Formal Methods in Security and Privacy

RHL and probabilistic computations

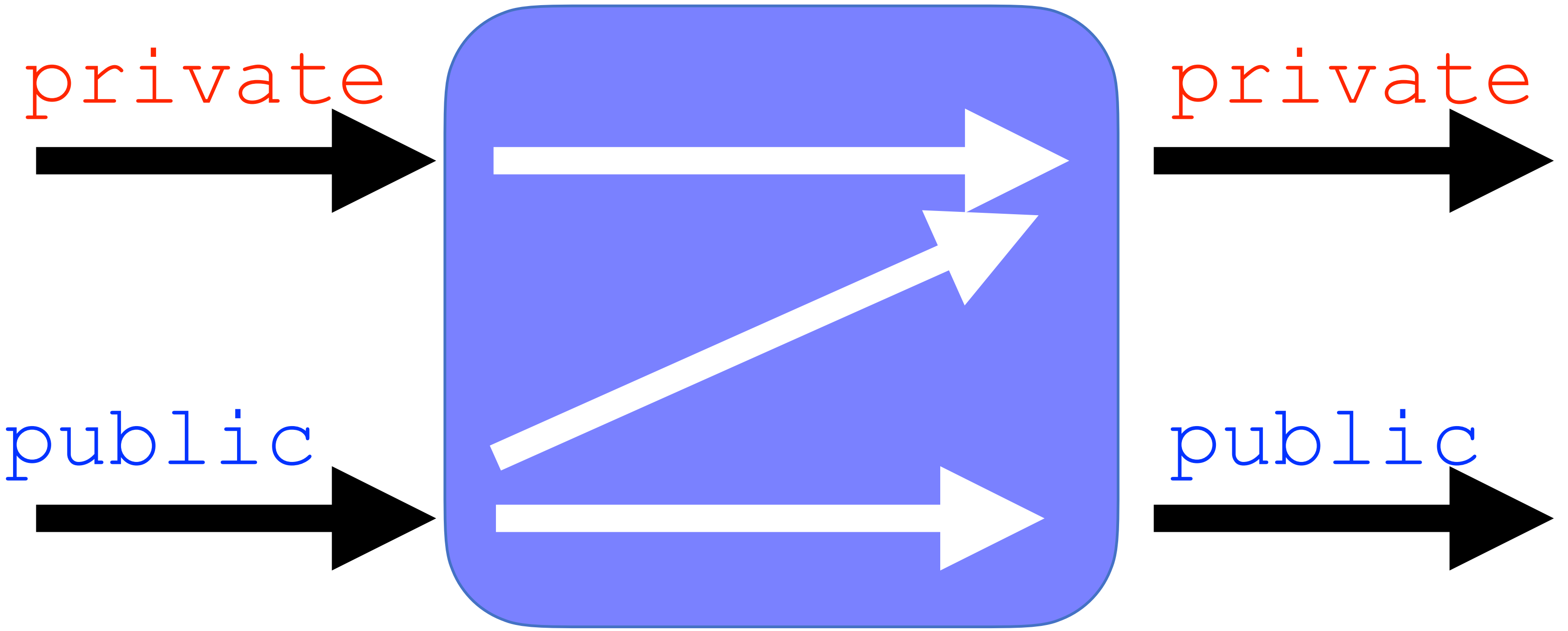
Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

From the previous classes

Information Flow Control

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

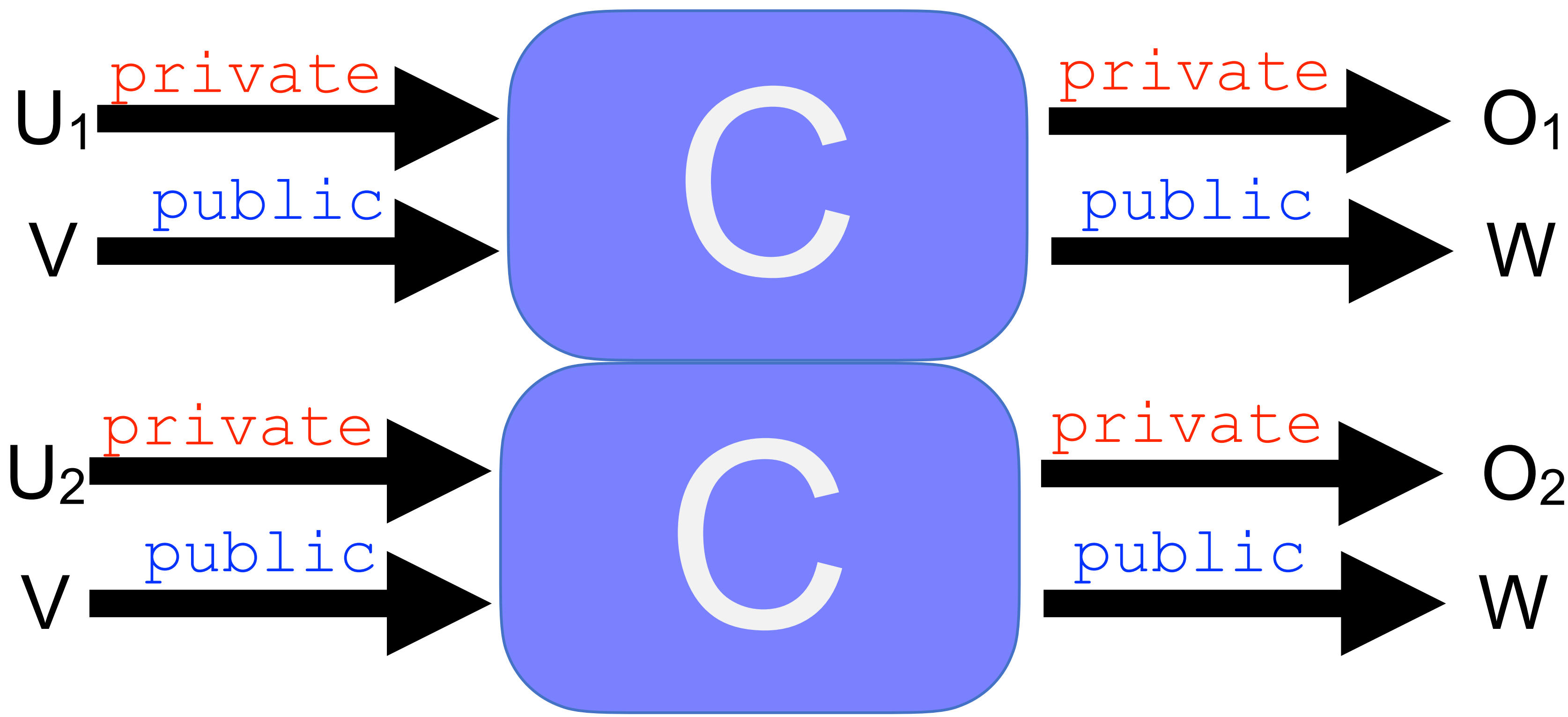


NonInterference

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 $m_1' \sim_{low} m_2'$



Relational Hoare Logic - RHL

Precondition
(a logical formula)

Precondition
$\text{Program}_1 \sim \text{Program}_2$
Postcondition

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

Program

Program

Postcondition
(a logical formula)

More Relational Hoare Logic

How can we prove this?

```
s1:public
s2:private
r:private
i: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
One-sided Rules

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

One pair for if.

Assignment — left

$\vdash$x:=e ~ skip:
P[e<1>/x<1>] ⇒ P

Assignment — right

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

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

Rules of Relational Hoare Logic
Program Equivalence Rule

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

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

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

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

Rules of Relational Hoare Logic
Combining Composition and Equivalence

$$\vdash \mathtt{c_1} \sim \mathtt{c_1'} : \mathtt{P} \Rightarrow \mathtt{R}$$

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

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

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

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).

Soundness

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

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?

Relative Completeness

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

Soundness and completeness with respect to Hoare Logic

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

iff

$$\vdash_{\mathtt{HL}} \mathtt{c_1 ; c_2 : P \Rightarrow Q}$$

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

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
```

$d_1, d_2, \ldots$ probabilistic expressions

Probabilistic Expressions

We extend the language with expression describing probability distributions.

$$\boxed{\texttt{d::= f(}e_1,\ldots,e_n,d_1,\ldots,d_k\texttt{)}}$$

Where `f` is a distribution declaration

Some expression examples

`uniform(`$\{0,1\}^n$`)`   `gaussian(`k,σ`)`   `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?

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,
$supp(\mu) = \{a \in A \mid \mu(a) > 0\}$
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 $\mu \in D(A)$.

Probabilistic Subdistributions

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

We define the probability of an event E⊆A with respect to the subdistribution μ:D(A) as

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

Probabilistic Subdistributions

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

$$\mathbb{P}_{\mu}[\varnothing] = 0$$

$$\mathbb{P}_{\mu}[A] \leq 1$$

$$0 \leq \mathbb{P}_{\mu}[E] \leq 1$$

$E \subseteq F \subseteq A$ implies $\mathbb{P}_{\mu}[E] \leq \mathbb{P}_{\mu}[F]$

$E \subseteq A$ and $F \subseteq A$ implies $\mathbb{P}_{\mu}[E \cup F] \leq \mathbb{P}_{\mu}[E] + \mathbb{P}_{\mu}[F] - \mathbb{P}_{\mu}[E \cap F]$

We will denote by **O** the subdistribution μ defined as constant 0.

Operations over Probabilistic Subdistributions

Let's consider an arbitrary a∈A, we will often use the distribution unit(a) defined as:

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

We can think about unit as a function of type unit:A → 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 let a =μ in M a defined as:

$$\mathbb{P}_{\text{let a =μ in M a}}[E] = \sum_{a \in \text{supp}(\mu)} \mathbb{P}_{\mu}[\{a\}] \cdot \mathbb{P}_{(Ma)}[E]$$

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.

$$\{\mathtt{uniform}(\{0,1\}^n)\}_m \in D(\{0,1\}^n)$$

$$\{\mathtt{gaussian}(k,\sigma)\}_m \in D(\mathtt{Real})$$

Semantics of PWhile Commands

What is the meaning of the following command?

```
k:=$ uniform({0,1}^n); z:= x mod k;
```

We can give the semantics as a function between command, memories and subdistributions over memories.

$$\texttt{Cmd} * \texttt{Mem} \longrightarrow \texttt{D(Mem)}$$

We will denote this relation as:

$$\{c\}_m = \mu$$

Semantics of Commands

This is defined on the structure of commands:

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

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

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

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

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

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

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

Revisiting the 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.

How do we formalize this?

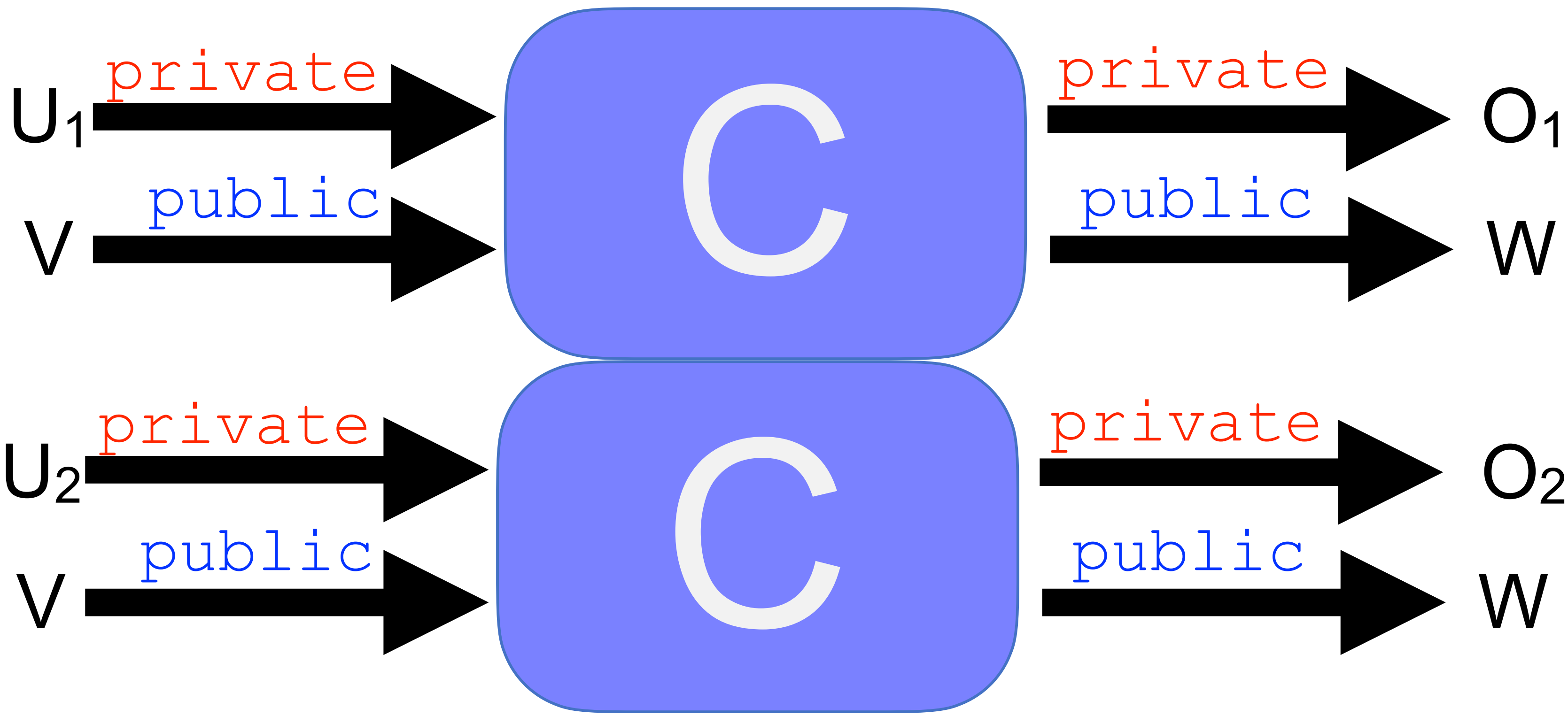
Probabilistic Noninterference

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.

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
```

How can we prove that this is noninterferent?