

# CS 511: Formal Methods in Security and Privacy

Rest of class preview for projects

Marco Gaboardi  
gaboardi@bu.edu

Alley Stoughton  
stough@bu.edu

# Final Projects

Projects can take different forms depending on the interest of each student but all the projects must have a formal method component.

You may work on your project alone or with others. Groups can be composed by at most two students.

Each group **is expected to meet with us regularly.**

We expect to see you three times to discuss advancements and directions of the project.

We will reserve **Oct 14, Oct 28 and Nov 18** to meet with you.

The deadline for choosing a project is **one week from now (Oct 14).**

# Final Projects

The evaluation of the project will be based on the continuous progress assessed during the regular meetings, a short report, and an in-class presentation.

Presentations: Dec 8 and Dec 10.

# Differential Privacy

# Releasing the mean of Some Data

```
Mean(d : private data) : public real
  i:=0;
  s:=0;
  while (i<size(d))
    s:=s + d[i]
    i:=i+1;
  return (s/i)
```

# $(\epsilon, \delta)$ -Differential Privacy

## Definition

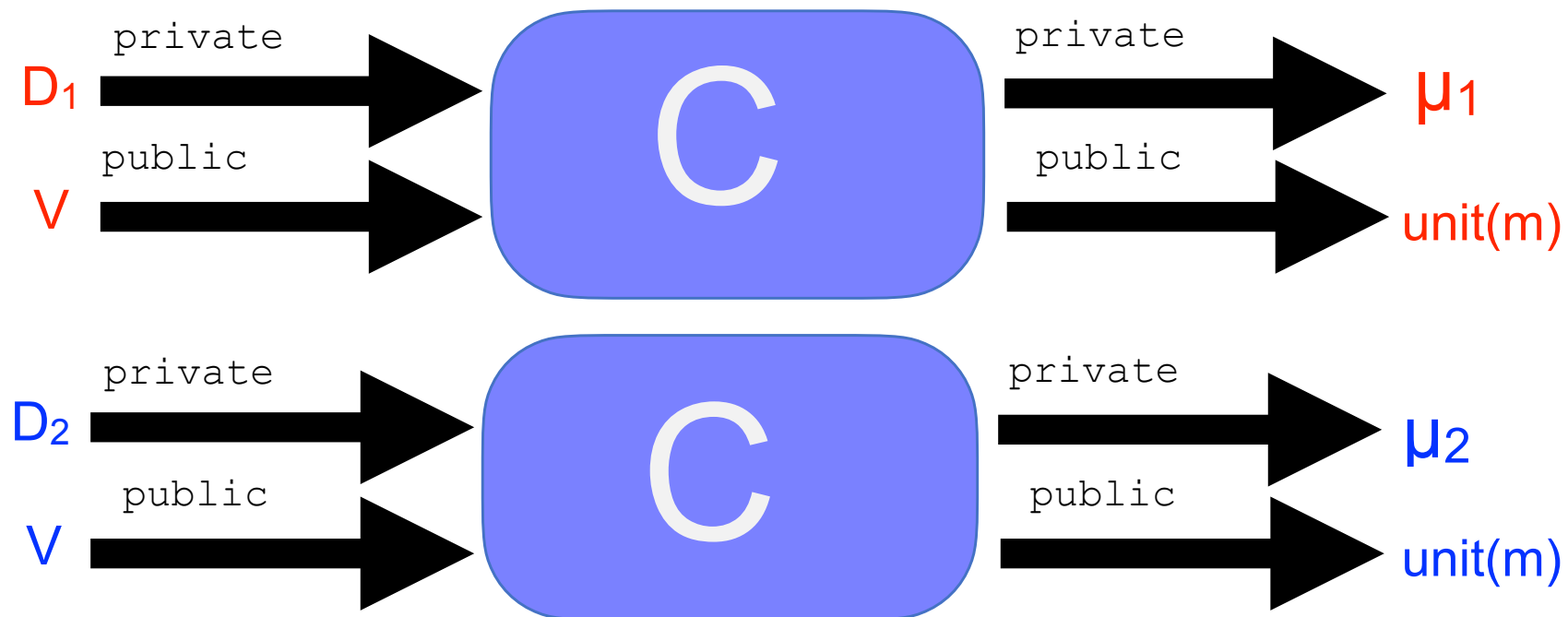
Given  $\epsilon, \delta \geq 0$ , a probabilistic query  $Q: X^n \rightarrow R$  is  $(\epsilon, \delta)$ -differentially private iff  
for all adjacent database  $b_1, b_2$  and for every  $S \subseteq R$ :

$$\Pr[Q(b_1) \in S] \leq \exp(\epsilon) \Pr[Q(b_2) \in S] + \delta$$

# Differential Privacy as a Relational Property

$c$  is **differentially private** if and only if for every  $m_1 \sim m_2$  (extending the notion of adjacency to memories):

$\{c\}_{m_1} = \mu_1$  and  $\{c\}_{m_2} = \mu_2$  implies  $\Delta_\epsilon(\mu_1, \mu_2) \leq \delta$



# Releasing privately the mean of Some Data

```
Mean(d : private data) : public real
  i:=0;
  s:=0;
  while (i<size(d))
    s:=s + d[i]
    i:=i+1;
  z:=$ Laplace(sens/eps,0)
  z:= (s/i)+z
  return z
```

# apRHL

Indistinguishability  
parameter

Precondition  
(a logical formula)

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

Probabilistic  
Program

Probabilistic  
Program

Postcondition  
(a logical formula)

# $R - (\varepsilon, \delta)$ -Coupling

Given two distributions  $\mu_1 \in \mathcal{D}(A)$ , and  $\mu_2 \in \mathcal{D}(B)$ , we have an  $R - (\varepsilon, \delta)$ -coupling between them, for  $R \subseteq A \times B$  and  $0 \leq \delta \leq 1$ ,  $\varepsilon \geq 0$ , if there are two joint distributions  $\mu_L, \mu_R \in \mathcal{D}(A \times B)$  such that:

- 1)  $\pi_1(\mu_L) = \mu_1$  and  $\pi_2(\mu_R) = \mu_2$ ,
- 2) the support of  $\mu_L$  and  $\mu_R$  is contained in  $R$ .

That is, if  $\mu_L(a, b) > 0$ , then  $(a, b) \in R$ ,  
and if  $\mu_R(a, b) > 0$ , then  $(a, b) \in R$ .

- 3)  $\Delta_\varepsilon(\mu_L, \mu_R) \leq \delta$

# Probabilistic Relational Hoare Logic

## Composition

$$\frac{\vdash_{\epsilon_1, \delta_1} C_1 \sim C_2 : P \Rightarrow R \quad \vdash_{\epsilon_2, \delta_2} C_1' \sim C_2' : R \Rightarrow S}{\vdash_{\epsilon_1 + \epsilon_2, \delta_1 + \delta_2} C_1 ; C_1' \sim C_2 ; C_2' : P \Rightarrow S}$$

# Report One-sided Noisy Max

```
ROSNM ( $q_1, \dots, q_k : \text{list data} \rightarrow \mathbb{R}$ ,  
         $b : \text{list data}, \epsilon : \mathbb{R}$ ) : nat  
  i = 1;  
  best = 0;  
  while (i  $\leq$  k){  
    cur =  $q_i(b) + \text{Lap}^+(\epsilon/2)$   
    if (cur > best  $\wedge$  i=1)  
      max = i ;  
      best = cur;  
  }  
  return max;
```

How can we reason about  
the accuracy of these  
mechanisms?

# Estimating the Error

$$\Pr_{r \sim M(D)} [u(q(D), r) \geq \alpha] \leq \beta$$

# Estimating the Error

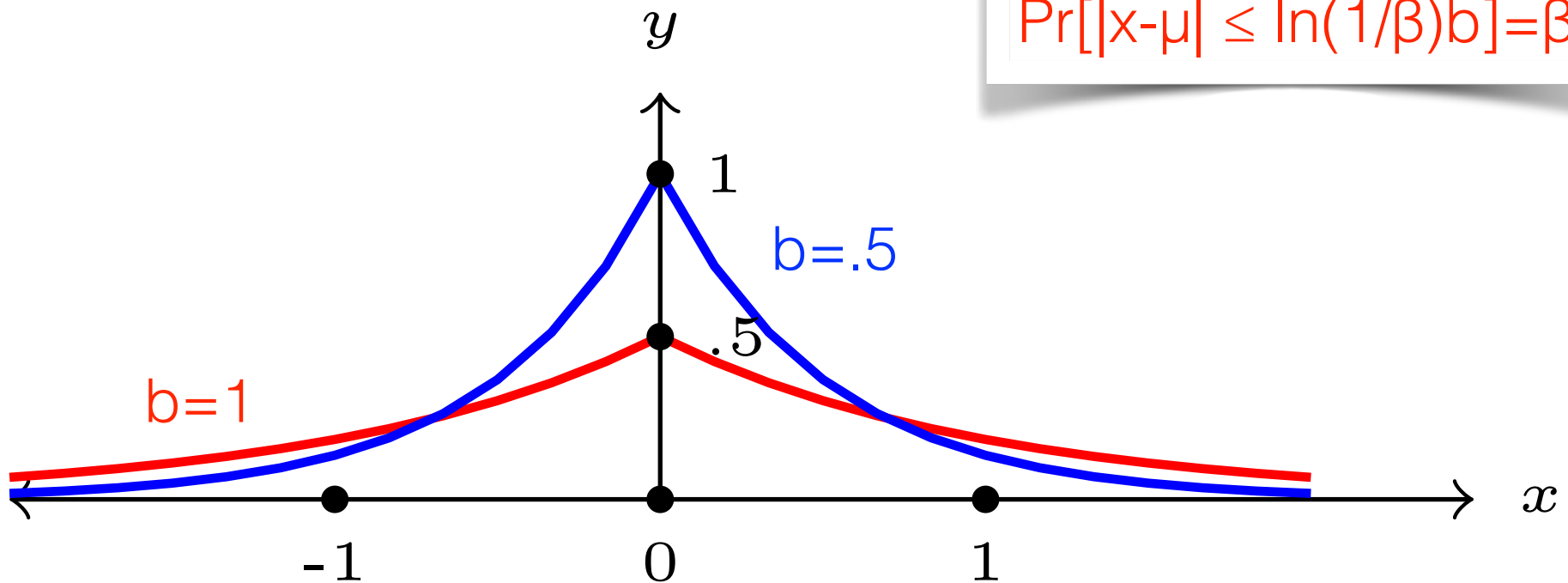
$$\Pr_{r \sim M(D)} [u(q(D), r) \geq \alpha] \leq \beta$$

We could be interested in showing that with probability  $1-\beta$  a mechanism returns a value that is  $\alpha$ -close to the original one.

# Laplace' tail bound

$$\text{Lap}(b, \mu)(X) = \frac{1}{2b} \exp\left(-\frac{|\mu - X|}{b}\right)$$

$$\Pr[|x - \mu| \leq \ln(1/\beta)b] = \beta$$

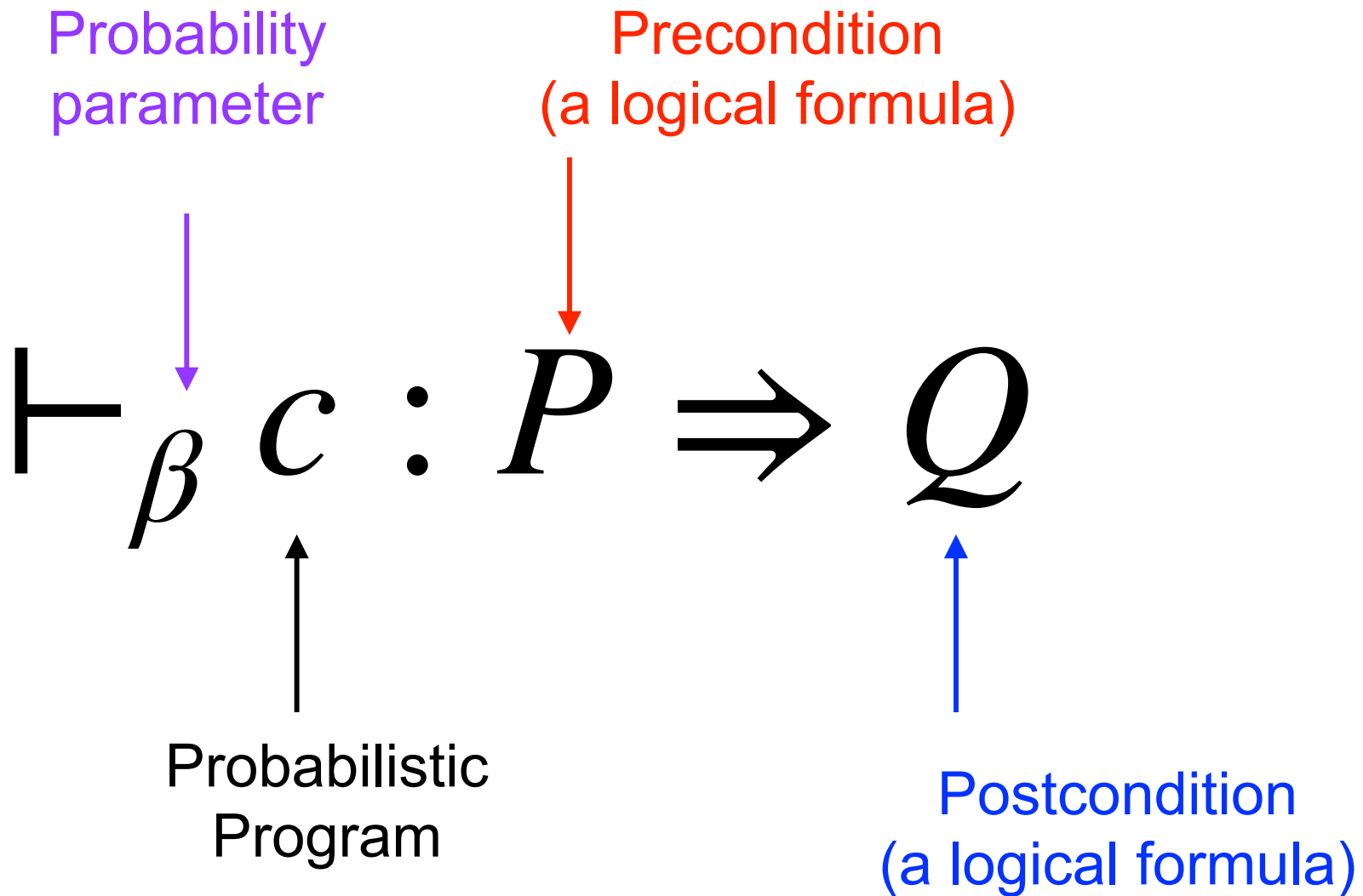


# Union bound

A basic fact of probability theory gives us:

$$\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$$

# uHL



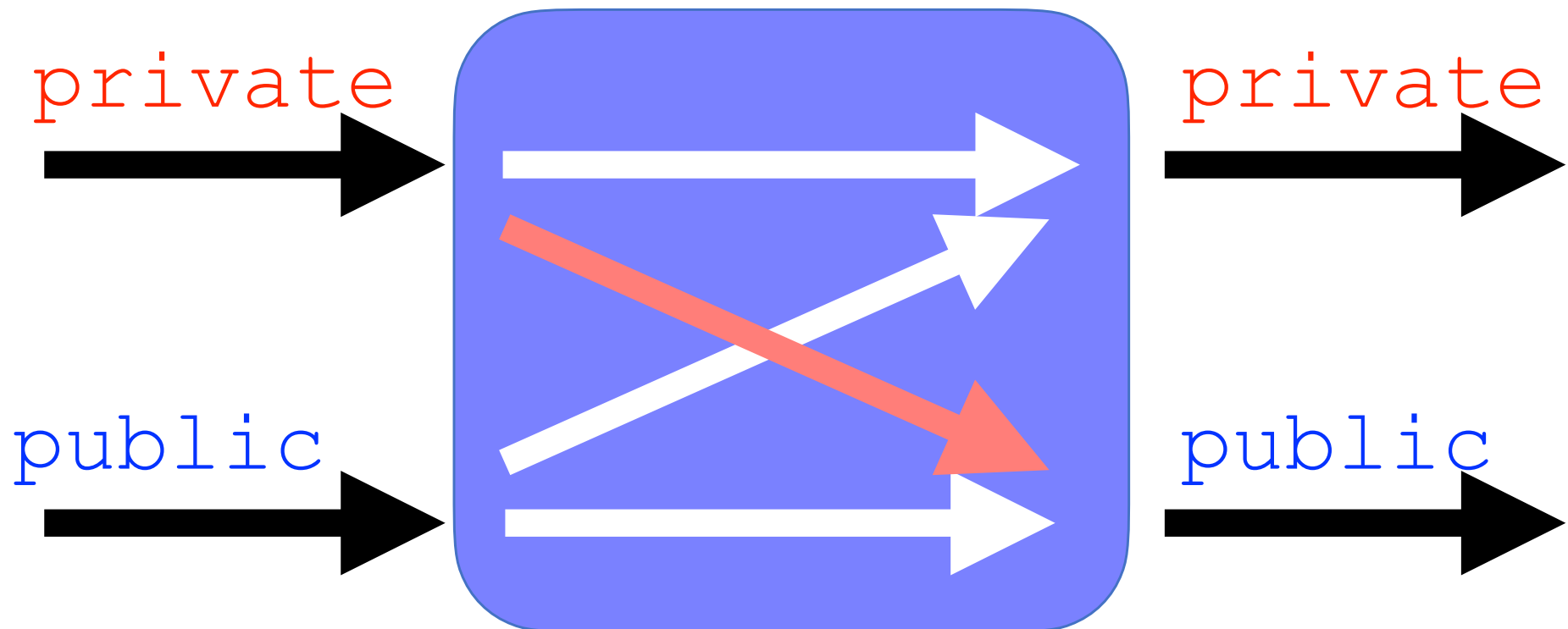
# Unary u-lifting

We say that a subdistribution  $\mu \subseteq D(A)$  is in the unary u-lifting of the predicate  $Q \subseteq A$ , denoted  $Q^u_\beta(\mu)$  if and only if we have:

$$\Pr_{r \sim \mu}[r \notin Q] \leq \beta$$

# Quantitative Information Flow Control

We want to **quantify** the **confidential information** that **leaks** in what is considered **nonconfidential**.



# Guessing Game



- The adversary has some prior  $\pi_R$  on  $R$  and it updates it after seeing  $U$ .

# Information leakage

Information leaked =  
initial uncertainty - remaining uncertainty

- Which could be

$$\text{Leakage}(U) = H(R) - H(R | U)$$

- This is the mutual information between R and U

# Gain function

- Suppose we have a set of secrets **X** and a set of actions **W**, then a gain function  $g$  is a function of type:

$$g : \mathbf{X} \times \mathbf{W} \rightarrow \mathbb{R}$$

- We can think about  $g$  as a scoring function for actions on a secret

# Gain function

- Suppose we have a set of secrets **X** and a set of actions **W**, then a gain function  $g$  is a function of type:

$$g : \mathbf{X} \times \mathbf{W} \rightarrow \mathbb{R}$$

- We can think about  $g$  as a scoring function for actions on a secret

We could have a similar definition based on losses.

# Type Systems for Noninterference

- Instead of proving noninterference in EasyCrypt, type systems for noninterference can be defined
- The goal is to ensure that if a program is well-typed, that it is noninterferent
- If typechecking is decidable, then there will always be noninterferent programs that don't type check

# Type Systems for Noninterference

- A project could study existing type systems and improve on them in some way
- A project could implement some of these type systems, running them on various examples and counterexamples
- A project could translate well-typed programs in some type system into EasyCrypt proofs of noninterference

# Crypto Theory Game-hopping Security Proofs

- We will study the proof of security of a symmetric encryption scheme using the game-hopping (also known as sequence of games) approach, giving an explicit upper bound on the ability of an adversary to break the security of the scheme
- A project could prove the security of another cryptographic scheme

<https://github.com/alleystoughton/EasyTeach/tree/master/encryption>

# Real/Ideal Paradigm Security Proofs

- We will study a *real/ideal paradigm* security proof in which we bound an adversary's ability to distinguish a "real" protocol from a secure-by-construction "ideal" one
- Our example will involve *security via indirection*, where protocol parties use a trusted computing base giving them indirect access to objects
- Real/ideal paradigm projects are possible, possibly using security via indirection

<https://github.com/alleystoughton/GuessingGame>

# Universally Composable Security and EasyUC

- We will give an introduction to EasyUC, a formalization in EasyCrypt of Universally Composable Security, a refinement of the real/ideal paradigm
  - In UC security, the security of a protocol can be applied when proving the security of a bigger protocol using the protocol

<https://github.com/easyuc/EasyUC>

# Universally Composable Security and EasyUC

- EasyUC has a domain specific language -- the UC DSL -- for describing ideal functionalities, protocols and simulators
- A type checker catches many errors
- An interpreter lets one experiment with UC models, ensuring they work as intended
- There is a translator into EasyCrypt, where sequence of games security proofs are carried out
- Various EasyUC-related projects are possible

# Proving Lower Bounds of Computational Problems

- We have a published framework for formalizing proofs establishing lower bounds for computational problems in the *query model*, where an algorithm doesn't have direct access to its data but may only make queries about it
- A project could prove a new lower bound result using this framework

<https://github.com/alleystoughton/AlgorithmicBounds>

# Proving Lower Bounds of Computational Problems

- We also have an example lower bound proof in the *randomized query model*, where we allow for some probability that an algorithm produces the wrong answer
  - The proof uses the crypto game hopping approach
- A project could prove lower bounds for other examples
- A project could develop a more formal framework for such results

<https://github.com/alleystoughton/ProbabilisticOr>