# Secure Multi-Party Computation

Giulia Fanti

Fall 2019

Based on slides by Vitaly Shmatikov

# Administrative

- HW4 due on Friday, 11:59 pm
- Additional OH on Friday (Sruti)
  - Regular location and time
- Final project
  - Presentations last week of class: Mon.  Dec. 2 and Wed. Dec. 4
    - Sign up here: https://docs.google.com/spreadsheets/d/1ylz1MWLtlAJvxUkpTAT0fVtqKabFQXanGh3wqo1g-tc/
    - PLEASE ADD YOUR CANVAS GROUP NUMBER
  - Final writeup due on Dec. 11, 11:59 pm EDT

# In-class Quiz

- On Canvas

# Last time: Hidden Services

**Dr. Neal Krawetz**
@hackerfactor

Just noticed that my Tor hidden service has been under a DDoS for days -- and I never noticed. Someone is seriously trying to take it offline. Hundreds of rendezvous points negotiated per minute. (Zero impact on my server.)

8:33 pm · 15 Nov 2019 · TweetDeck

Explain this tweet

# More explanation

**Dr. Neal Krawetz** @hackerfactor · 15h

I think it's an attempt to take down access to the @internetarchive. I provide the Tor onion service for IA, and it's that service which is under a DDoS. He's using a newer version of the same technique that "Eddie" (i.e., Russian-based attackers) used.

1 5 19

**Dr. Neal Krawetz** @hackerfactor · 12h

What a coincidence... One Tor relay has been requested as a rendezvous node nearly 3x more than any other. And it has an uptime of only a few hours before the attack began. Blocking it reduced DDoS by 70%.

Hey @torproject Here's a hostile relay: metrics.torproject.org/rs.html#search...

6

# Today’s material: Secure Multi-Party Computation

What is it?

- How do we define security?

Examples

- Oblivious transfer
- Garbled circuits

- Focus on computational security

# Secure Multi-Party Computation

- Framework for computation between parties who do not trust each other
- Example: elections
  - N parties, each one has a "Yes" or "No" vote
  - Goal: determine the majority vote, without revealing how other people voted
- Example: auctions
  - Each bidder makes an offer
  - Goal: determine whose offer won without revealing losing offers

**Verifiable Sealed-Bid Auction on the Ethereum Blockchain**

Hisham S. Galal and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Quebéc, Canada

**Trustee: Full Privacy Preserving Vickrey Auction on top of Ethereum**

Hisham S. Galal and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Quebéc, Canada

# More Examples

- Example: distributed data mining
  - Two companies want to compare their datasets without revealing them
    - For example, compute the intersection of two customer lists
- Example: database privacy
  - Evaluate a query on the database without revealing the query to the database owner
  - Evaluate a statistical query without revealing the values of individual entries

**Google open-sources cryptographic tool to keep data sets private**

by RAVIE LAKSHMANAN — 5 months ago in SECURITY

# A Couple of Observations

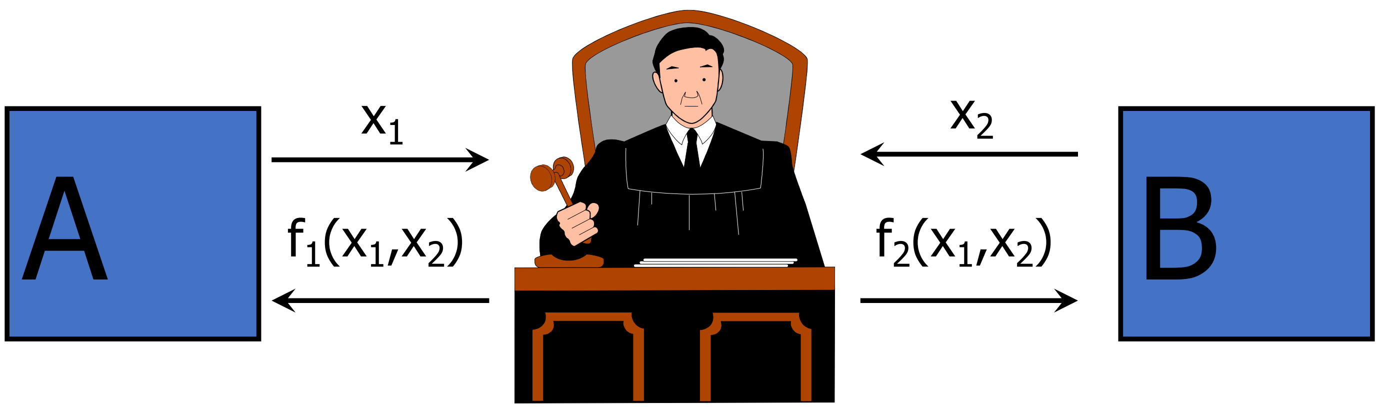
- We are dealing with distributed multi-party protocols
  - "Protocol" describes how parties are supposed to exchange messages on the network
- All of these tasks can be easily computed by a trusted third party
  - Secure multi-party computation aims to achieve the same result without involving a trusted third party

# How to Define Security?

- Must be mathematically rigorous
- Must capture all realistic attacks that a malicious participant may try to stage
- Should be “abstract”
  - Based on the desired “functionality” of the protocol, not a specific protocol
  - Goal: define security for an entire class of protocols

# Ideal Model

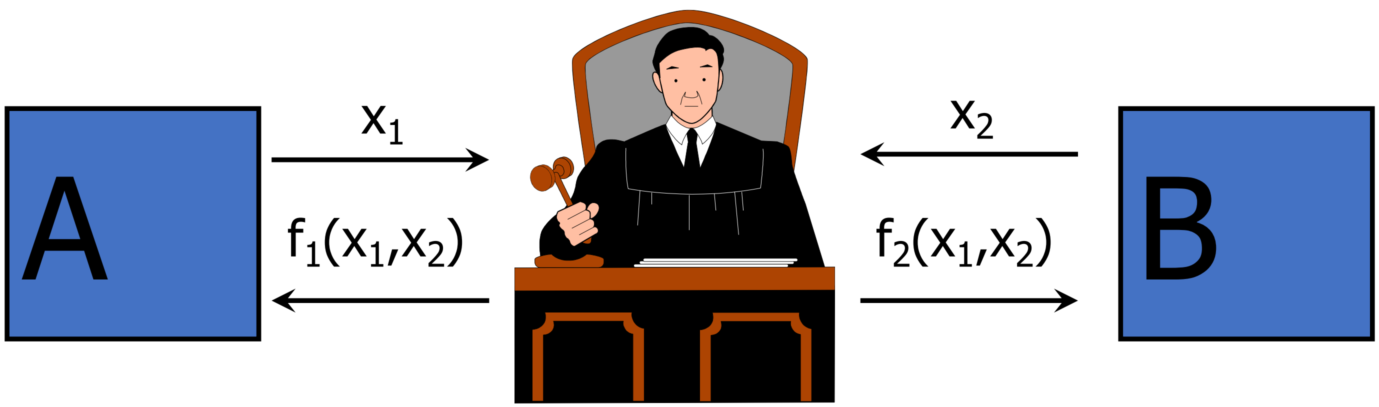
- Intuitively, we want the protocol to behave "as if" a trusted third party collected the parties' inputs and computed the desired functionality
  - Computation in the ideal model is secure by definition!



# In other words…

- A protocol is secure if it emulates an ideal setting where the parties hand their inputs to a "trusted party," who locally computes the desired outputs and hands them back to the parties

[Goldreich-Micali-Wigderson 1987]



# Adversary Models

- Some participants may be dishonest (corrupt)
  - If all were honest, we would not need secure multi-party computation
- Semi-honest (aka passive; honest-but-curious)
  - Follows protocol, but tries to learn more from received messages than he would learn in the ideal model
- Malicious
  - Deviates from the protocol in arbitrary ways, lies about his inputs, may quit at any point
- For now, focus on semi-honest adversaries and two-party protocols

# Correctness and Security

- How do we argue that the real protocol "emulates" the ideal protocol?
- Correctness
  - All honest participants should receive the correct result of evaluating functionality f
  - Because a trusted third party would compute f correctly
- Security
  - All corrupt participants should learn no more from the protocol than what they would learn in the ideal model
  - What does a corrupt participant learn in ideal model?
  - His own input and the result of evaluating f

# Simulation

- Corrupt participant's view of the protocol = record of messages sent and received
  - In the ideal world, this view consists simply of his input and the result of evaluating f
- How to argue that real protocol does not leak more useful information than ideal-world view?
- Key idea: simulation
  - If real-world view (i.e., messages received in the real protocol) can be simulated with access only to the ideal-world view, then real-world protocol is secure
  - Simulation must be <u>indistinguishable</u> from real view

# Terminology

- Distance between probability distributions A and B

$$\text{dist}(A,B) = \frac{1}{2}\sum_{x} |\Pr(A = x) - \Pr(B = x)|$$

- Probability ensemble $A_i$ is a set of discrete probability distributions
    - Index i ranges over some set $I$

- Function f(n) is negligible if it is asymptotically smaller than the inverse of any polynomial

$$\forall c \in \mathbb{N}, \exists m \;\; s.t. |f(n)| < \frac{1}{n^c} \;\; \forall n > m$$

# Indistinguishability Notions

- Distribution ensembles $A_i$ and $B_i$ are equal if $\text{dist}(A_i, B_i) = 0$

- Distribution ensembles $A_i$ and $B_i$ are statistically close if $\text{dist}(A_i, B_i)$ is a negligible function of $i$

- Distribution ensembles $A_i$ and $B_i$ are computationally indistinguishable ($A_i \approx B_i$) if, for any probabilistic polynomial-time algorithm $D$,

$$|\Pr(D(A_i) = 1) - \Pr(D(B_i) = 1)|$$

is a negligible function of $i$

# Ideal World

- Trusted party computes $y = f(x_A, x_B)$, sends result to each party.

Suppose $f(\quad) = (b, \emptyset)$, where $b$ random bit.

$f(\quad)_A = b$

Trusted Third Party

$f(\quad)_B = \emptyset$

Bob learns nothing about $b$.

# Real World

- Propose a protocol $\pi$ to implement functionality in the real world.

1. Alice draws $b'$ randomly.

$$\xrightarrow{\quad b' \quad}$$

$y_A = b'$

$y_B = \emptyset$

Intuitively, does this protocol securely implement the desired functionality $f$?

# SMC Definition

- Protocol $\pi$ for computing $f(XA, XB)$ between A and B is secure if there exist efficient simulator algorithms $S_A$ and $S_B$ such that for all input pairs $(x_A, x_B)$ :
- Correctness: $(y_A, y_B) \approx f(x_A, x_B)$
- Security:
  - Let $\mathrm{Real}_\pi\ (\mathrm{x_A}, \mathrm{x_B}) = \{\mathrm{view_A}, \mathrm{view_B}\}, (y_A, y_B)$ denote the output after running $\pi$ honestly
  - Let $\mathrm{Ideal}_f(x_A, x_B) = \{\mathrm{sim_A}(x_A, y_A), \mathrm{sim_B}(x_B, y_B)\}, (y_A, y_B)$

  - A protocol $\pi$ securely realizes $f$ if $\mathrm{Real}_\pi\ (\mathrm{x_A}, \mathrm{x_B}) \approx \mathrm{Ideal}_f(x_A, x_B)$

# Let's look at our definition

Ideal | Real

- Correctness

$$f(\quad) = (b, \emptyset) \qquad (y_A = b', y_B = \emptyset)$$

- Security

$$(\mathrm{sim}_A(\emptyset, b), \mathrm{sim}_B(\emptyset, \emptyset), b, \emptyset) \qquad (b', b', b', \emptyset)$$

$\mathrm{Ideal}_f(x_A, x_B) =$
$\{\mathrm{sim}_{\mathrm{A}}(x_A, y_A), \mathrm{sim}_{\mathrm{B}}(x_B, y_B)\}, (y_A, y_B)$

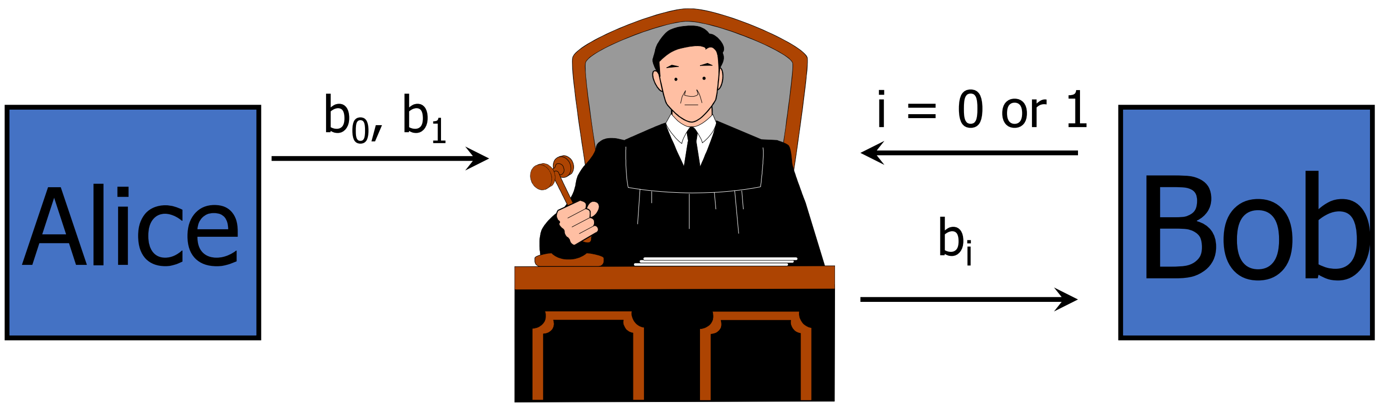
$\mathrm{Real}_\pi\ (\mathrm{x_A}, \mathrm{x_B}) =$
$\{\mathrm{view_A}, \mathrm{view_B}\}, (y_A, y_B)$

These two joint distributions are distinguishable!

# Oblivious Transfer (OT)

[Rabin 1981]

- Fundamental SMC primitive



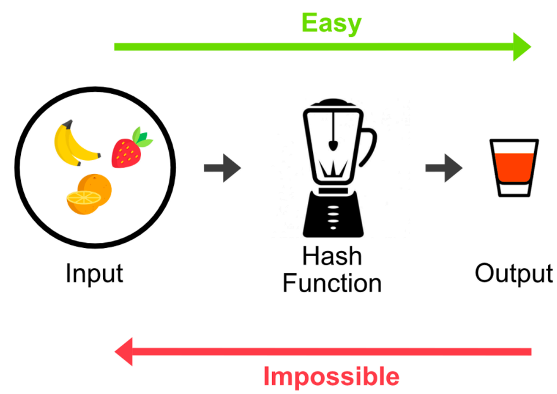

Alice inputs two bits, Bob inputs the index of one of Alice's bits
Bob learns his chosen bit, Alice learns nothing

  - Alice does not learn which bit Bob has chosen
  - Bob does not learn the value of the bit that he did not choose

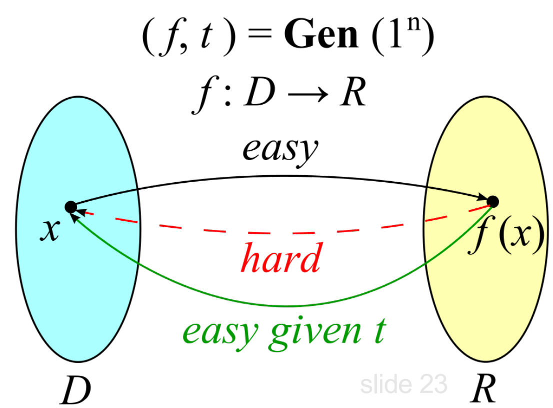
Generalizes to bitstrings, M instead of 2, etc.

# One-Way Trapdoor Functions

- Intuition: one-way functions are easy to compute, but hard to invert (skip formal definition)
    - We will be interested in one-way permutations




- Intution: one-way trapdoor functions are one-way functions that are easy to invert given some extra information called the <u>trapdoor</u>



# Euler's Theorem

- THM: If $a$ and $n$ are relatively prime, and $\phi(n)$ is Euler's totient function (# of numbers that are relatively prime with n), then

$$a^{\phi(n)} \equiv 1 \text{ mod n.}$$

So if $r \equiv 1 \; mod \; \phi(n)$, then $r = k \cdot \phi(n) + 1$. We have

$$\begin{aligned} a^r \text{mod n} &= a^{1+k\phi(n)} \text{mod n} \\ &\equiv a \cdot \left(a^k\right)^{\phi(n)} \text{mod n} \\ &\equiv a \text{ mod n} \end{aligned}$$

# One-way Trapdoor Function: Example

- Example: if $n = pq$ where $p$ and $q$ are large primes and $e$ is relatively prime to φ(n), $f_{e,n}(m) = m^e \bmod n$ is easy to compute, but it is believed to be hard to invert

- Given the trapdoor $d$ s.t. $de \equiv 1 \bmod \phi(n)$, $f_{e,n}(m)$ is easy to invert because $f_{e,n}(m)^d \equiv (m^e)^d \bmod n \equiv m \bmod n$

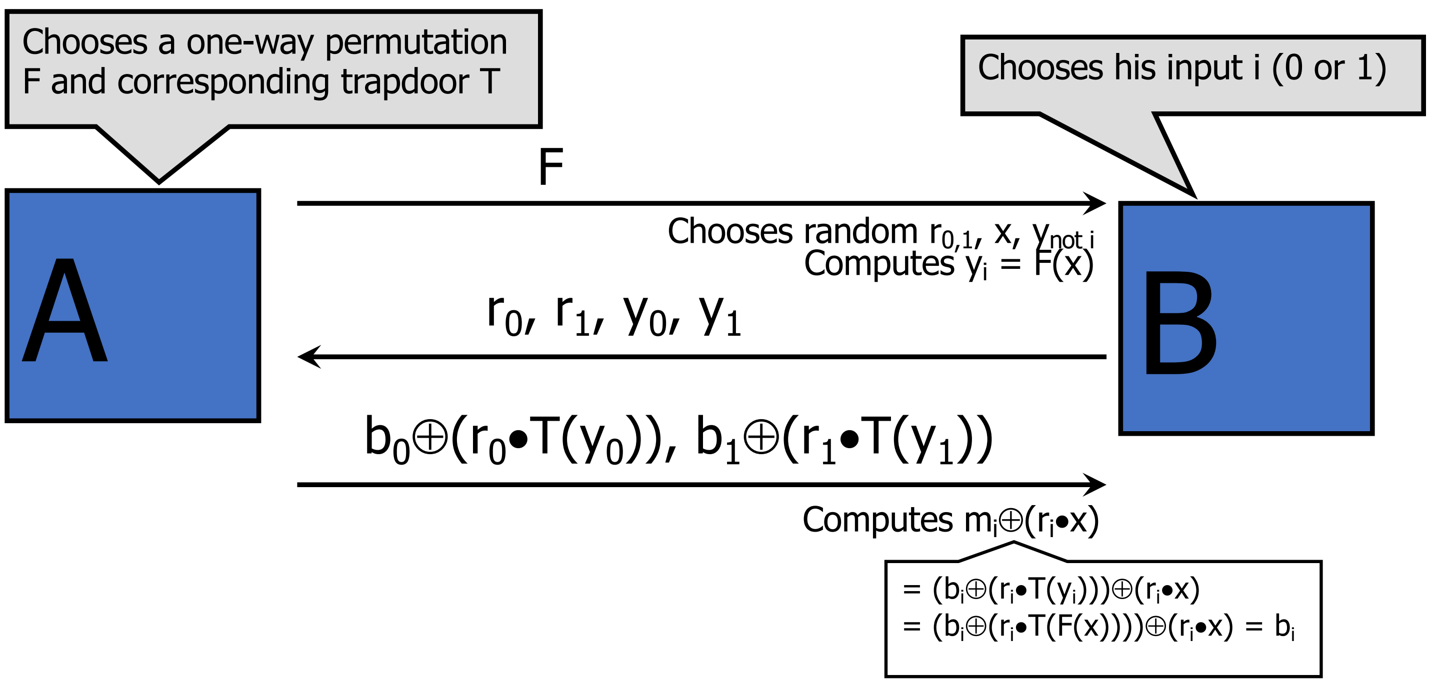
- Why?

# Hard-Core Predicates

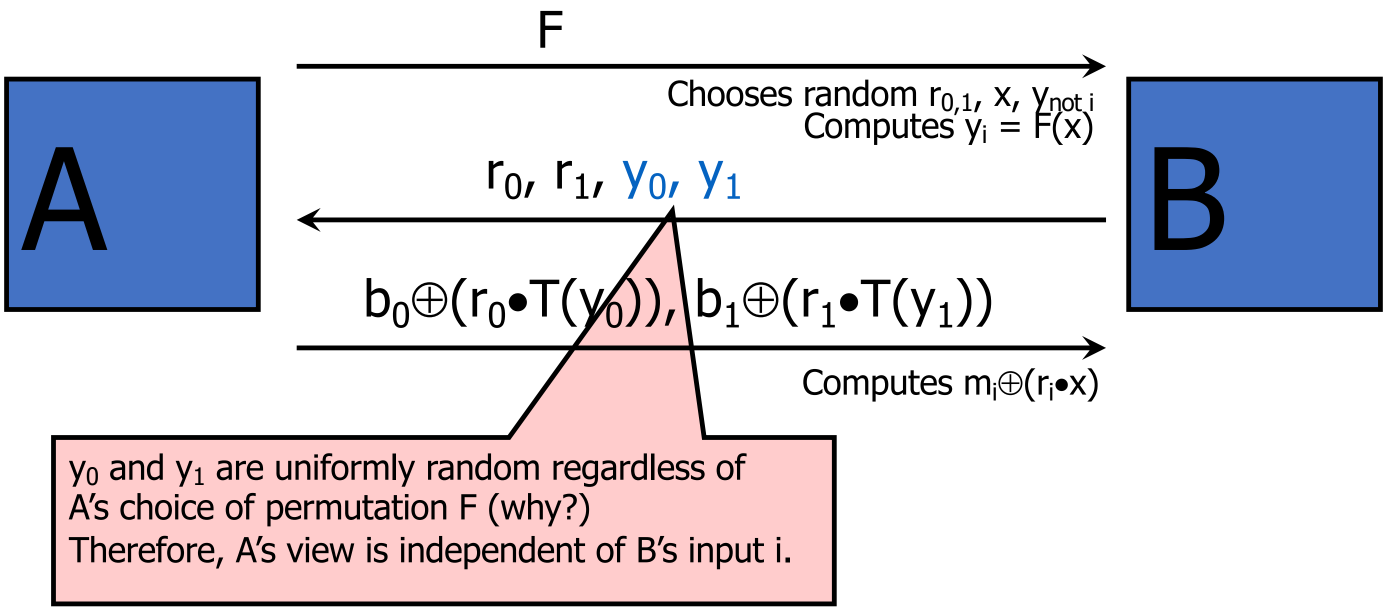
- Let $f$: $S \to S$ be a one-way function on some set $S$
- $B$: $S \to \{0,1\}$ is a hard-core predicate for $f$ if
  - there is a bit of information about $x$ such that learning this bit from $f(x)$ is as hard as inverting $f$
  - $B(x)$ is easy to compute given $x \in S$
  - If an algorithm, given only $f(x)$, computes $B(x)$ correctly with prob $>$ ½ $+$ $\varepsilon$, it can be used to invert $f(x)$ easily
- Goldreich-Levin theorem
  - $B(x, r) = r \bullet x$ is a hard-core predicate for $g(x, r) \ = \ (f(x), r)$
    - $f(x)$ is any one-way function, $r \bullet x = (r_1 x_1) \oplus \ldots \oplus (r_n x_n)$

# Oblivious Transfer Protocol

- Assume the existence of some family of one-way trapdoor permutations

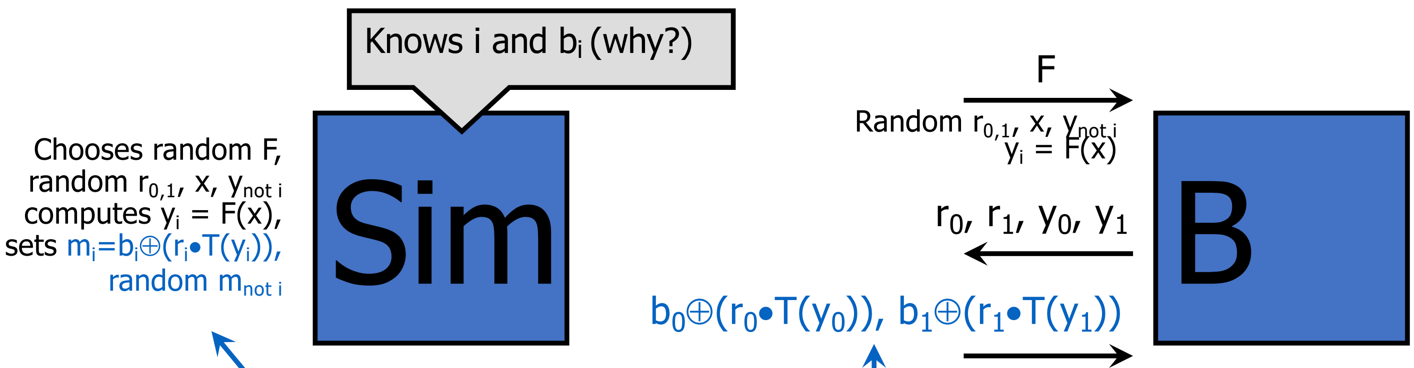


# Proof of Security for B



# Proof of Security for A (Sketch)

- Need to build a simulator whose output is indistinguishable from B's view of the protocol




The only difference between simulation and real protocol:
In simulation, $m_{\text{not } i}$ is random (why?)
In real protocol, $m_{\text{not } i} = b_{\text{not } i} \oplus (r_{\text{not } i} \bullet T(y_{\text{not } i}))$

# Proof of Security for A (Cont'd)

- Why is it computationally infeasible to distinguish random m and $m' = b \oplus (r \bullet T(y))$?
  - b is some bit, r and y are random, T is the trapdoor of a one-way trapdoor permutation
- $(r \bullet x)$ is a hard-core bit for $g(x,r)=(F(x),r)$
  - This means that $(r \bullet x)$ is hard to compute given $F(x)$
- If B can distinguish m and $m' = b \oplus (r \bullet x')$ given only $y=F(x')$, we obtain a contradiction with the fact that $(r \bullet x')$ is a hard-core bit
  - Proof omitted