Advanced Tools from Modern Cryptography

Lecture 8

Computational Security: Indistinguishability, Simulation

Security Definitions

- So far: Perfect secrecy
  - Achieved in Shamir secret-sharing, passive BGW and passive GMW (given a trusted party for OT)
- But for 2PC using Yao's Garbled circuit (even given a trusted party for OT) security only against computationally bounded adversary
  - We haven't defined such security yet!
- <u>Plan</u>
  - Computational Indistinguishability
  - Simulation-based security

Because, the obvious definition obtained by replacing perfect secrecy by computational secrecy turns out to be weak

Recall

Indistinguishability

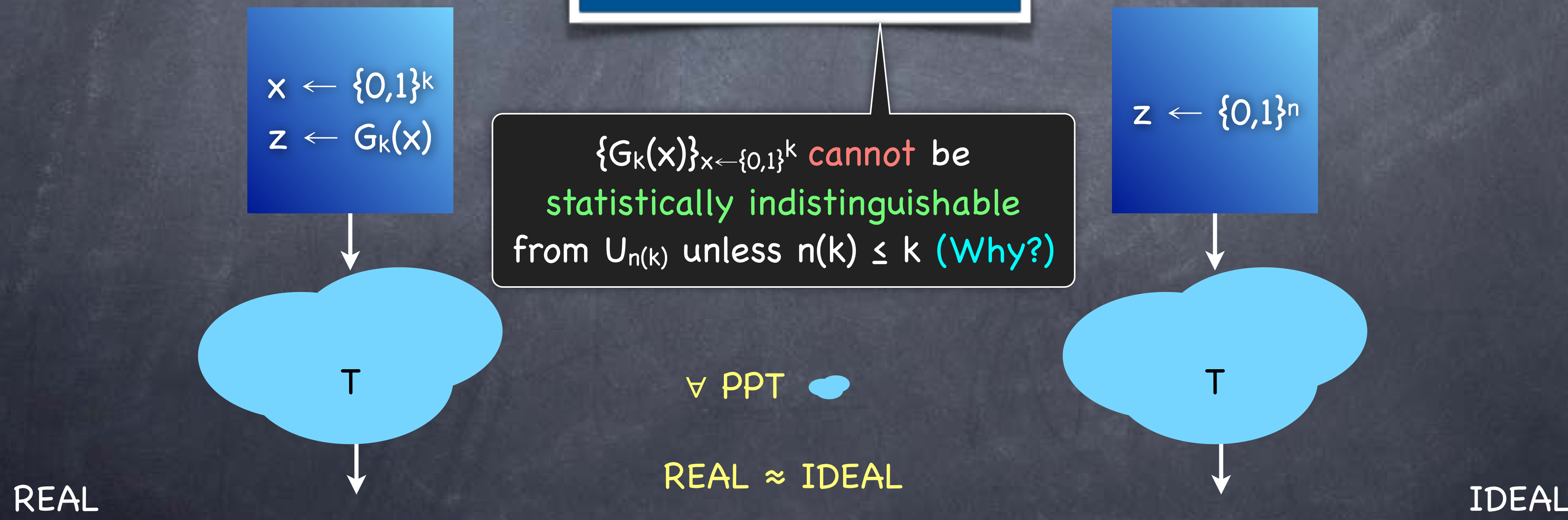
$A_k \approx B_k$

- Distribution ensembles $\{A_k\}$, $\{B_k\}$ **computationally indistinguishable** if $\forall$ Probabilistic Polynomial Time tests T, $\exists$ negligible $\nu(k)$ s.t.

$$\left| \Pr_{x \leftarrow A_k}[T(x)=1] - \Pr_{x \leftarrow B_k}[T(x)=1] \right| \leq \nu(k)$$

Example: Pseudorandomness Generator (PRG)

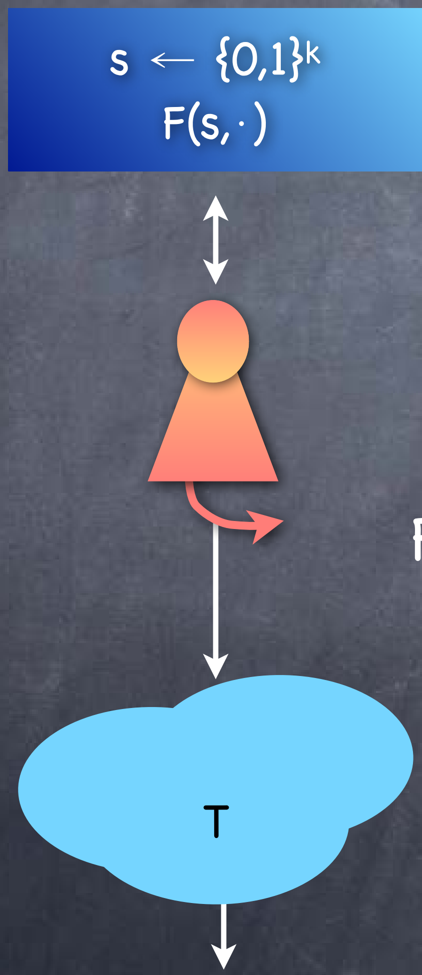
- Takes a short seed and (deterministically) outputs a long string
  - $G_k: \{0,1\}^k \to \{0,1\}^{n(k)}$ where $n(k) > k$
- Security definition: $\{G_k(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{n(k)}$



Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
  - Allows "random-access" (instead of just sequential access)
    - A function F(s;i) outputs the i^{th} block of the pseudorandom string corresponding to seed s
    - Exponentially many blocks (i.e., large domain for i)
- Pseudorandom Function
  - Need to define pseudorandomness for a function (not a string)
  - Idea: the view of an adversary <u>arbitrarily interacting with the function</u> is indistinguishable from its view when interacting with a random function

Pseudorandom Function (PRF)



REAL

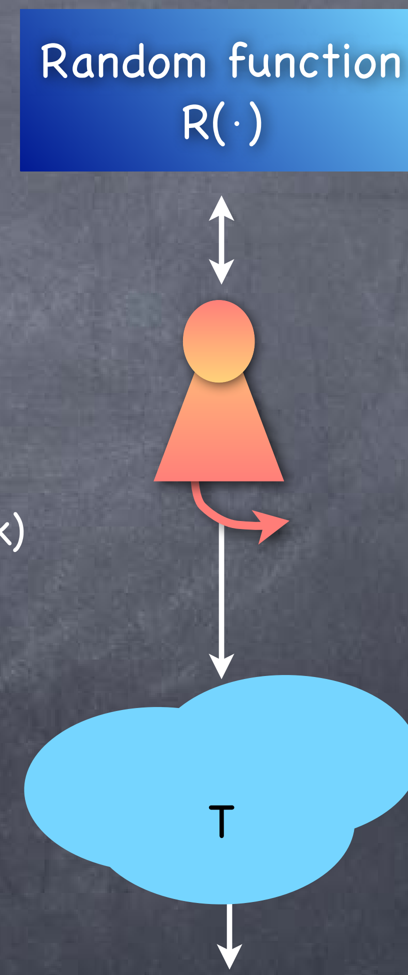
$F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}$

is a PRF if

∀ PPT 

∀ PPT 

REAL ≈ IDEAL



IDEAL

Security for MPC

- Recall: For passive security, secrecy is all the matters
- For a 2-party functionality f, with only Bob getting the output, perfect secrecy against corrupt Bob:
  i.e., $\forall$ x, x′, y s.t., $f(x,y) = f(x',y)$, $\text{view}_{\text{Bob}}(x,y) = \text{view}_{\text{Bob}}(x',y)$
  - In particular, if (y, f(x,y)) uniquely determines x (i.e., if $f(x',y)=f(x,y) \Rightarrow x'=x$), then OK for view to reveal x
- In the computational setting, just replace = with ≈ ?
  - We should ask for more!
  - E.g., f is a decryption algorithm, with key x and ciphertext y
  - Often, a (long enough) ciphertext and message uniquely determines the key
    - But not OK to reveal the key to Bob!

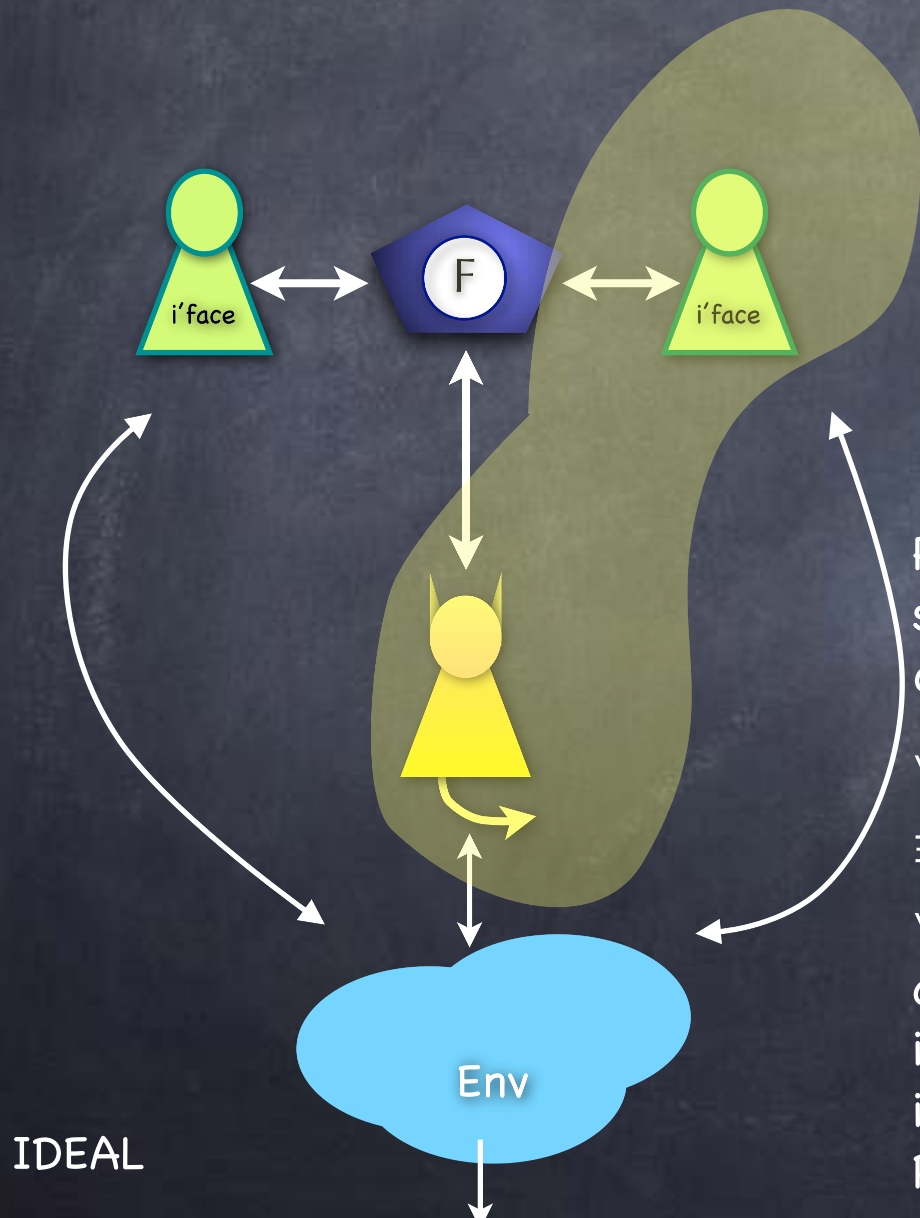
Makes sense only for the view, not f

Because, uniquely determines ≠ reveals!

Security for MPC

- Compare the protocol execution with an "ideal" execution involving an incorruptible trusted party
  - Trusted party collects all inputs, carries out all computation and delivers the outputs (over private channels)
  - Ideal is the best we can hope for
- If anything that could "go wrong" with the protocol execution could happen with the ideal execution too, then it is not the protocol's fault
  - Applies to active,as well as passive corruption
  - Applies to computational as well as information-theoretic security

Simulation-Based Security



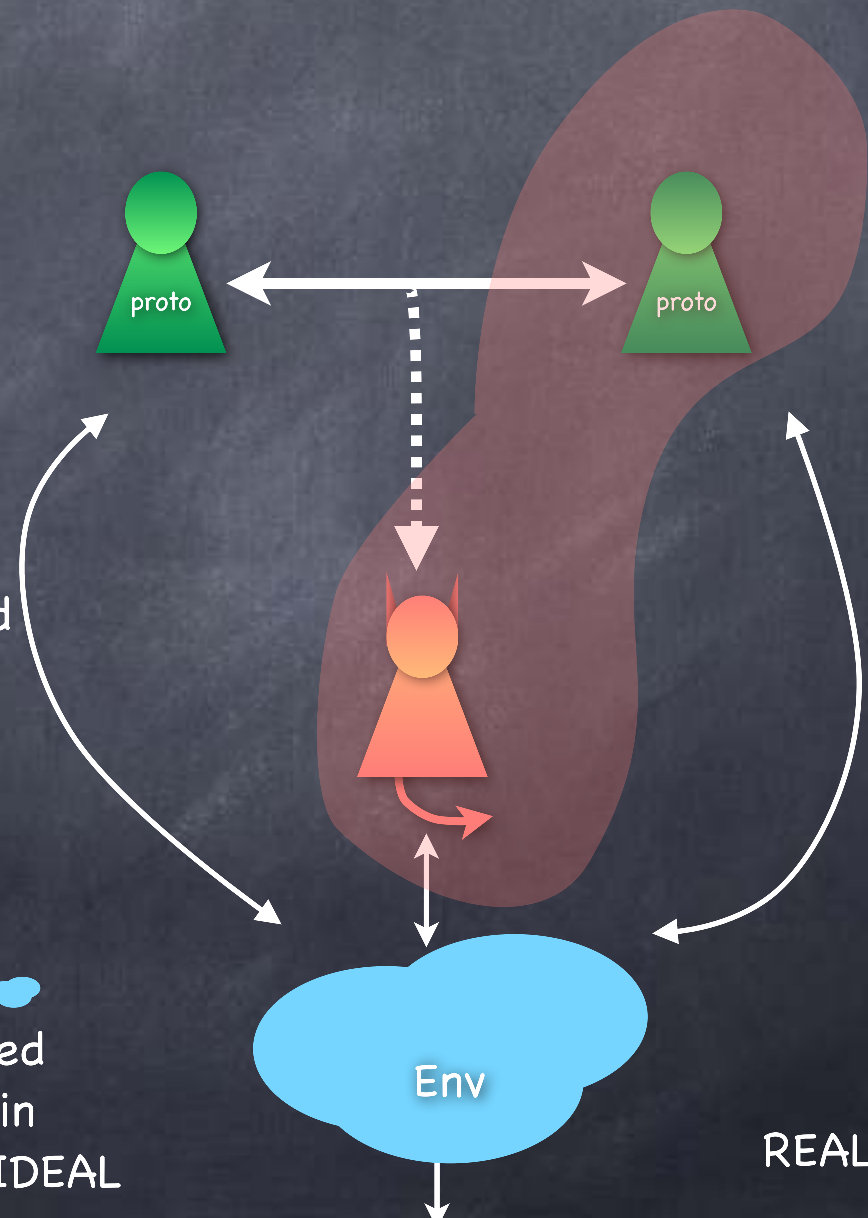
Protocol is secure (and correct) if:

∀ [adversary]

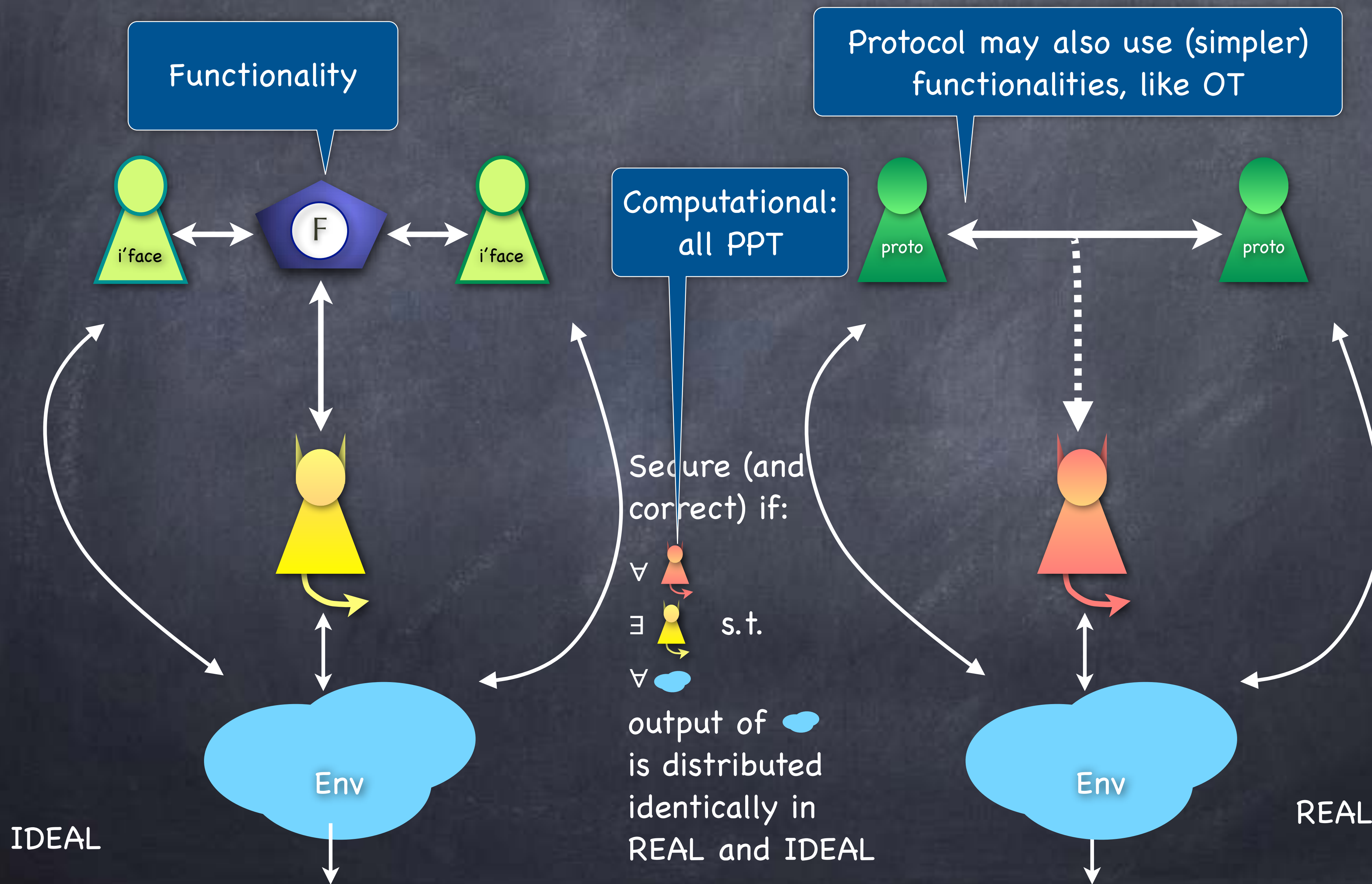
∃ [simulator] s.t.

∀ [environment]

output of [environment] is distributed identically in REAL and IDEAL



Simulation-Based Security



Variants of Security

- Same definitional framework can be used to define various levels of security!
  - Passive adversary: corrupt parties stick to the protocol
    - Will require corrupt parties in the ideal world also to use the correct inputs/outputs
  - Universally Composable security: Active adversary interacting with the environment arbitrarily
  - Standalone security: environment is not "live." Interacts with the adversary before and after (but not during) the protocol
  - Super-PPT simulation: meaningful when the "security" of ideal world is information-theoretic
- Aside: Non-simulation-based security definitions for MPC are also useful for intermediate tools, but often too subtle for final applications

Trust Issues Considered

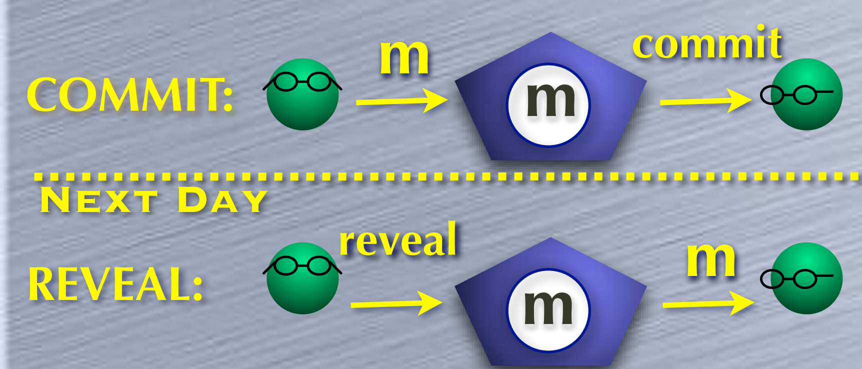
- Protocol may leak a party's secrets
  - Clearly an issue -- even for passive corruption
- Protocol may give adversary illegitimate influence on the outcome
  - Say in poker, if adversary can influence hands dealt
  - An issue even when no secrecy requirements
    - e.g., Exchanging inputs
- Simulation-based security covers these concerns
  - Because the ideal trusted party would allow neither

Example: Coin-Tossing

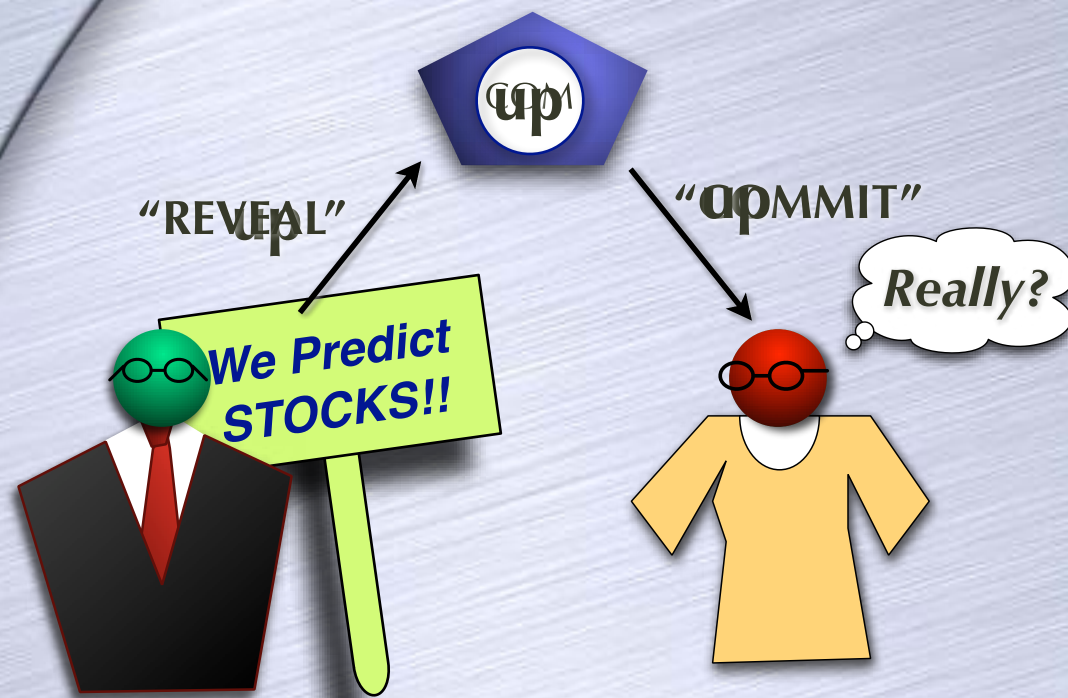
- Functionality F_{coin} samples a uniform random bit and sends it to all parties
- Security against passive corruption is trivial (Why?)
- Fact: Impossible to (even stand-alone) securely realise against computationally unbounded active adversaries
- Protocol for stand-alone security against PPT adversaries using commitment
  - If given ideal commitment functionality, information-theoretic security

Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding



IDEAL World
30 Day Free Trial



Example: Coin-Tossing

- A (fully) secure 2-party protocol for coin-tossing, given an ideal commitment functionality F_{com}
- Alice sends a bit a to F_{com}. (Bob gets "committed" from F_{com})
- Bob sends a bit b to Alice
- Alice sends "open" to F_{com}. (Bob gets a from F_{com})
- Both output $c=a\oplus b$
- Simulator:
  - Will get a bit c from F_{coin}. Needs to simulate the corrupt party's view in the protocol, including the interaction with F_{com}
  - If Alice corrupt: Get a from Alice. Send $b = a\oplus c$.
  - If Bob corrupt: Send "committed". Get b. Send $a = b\oplus c$.
- Perfect simulation: Environment + Adversary's view is identically distributed in REAL and IDEAL (verify!), and hence so is Environment's output