Advanced Tools from Modern Cryptography

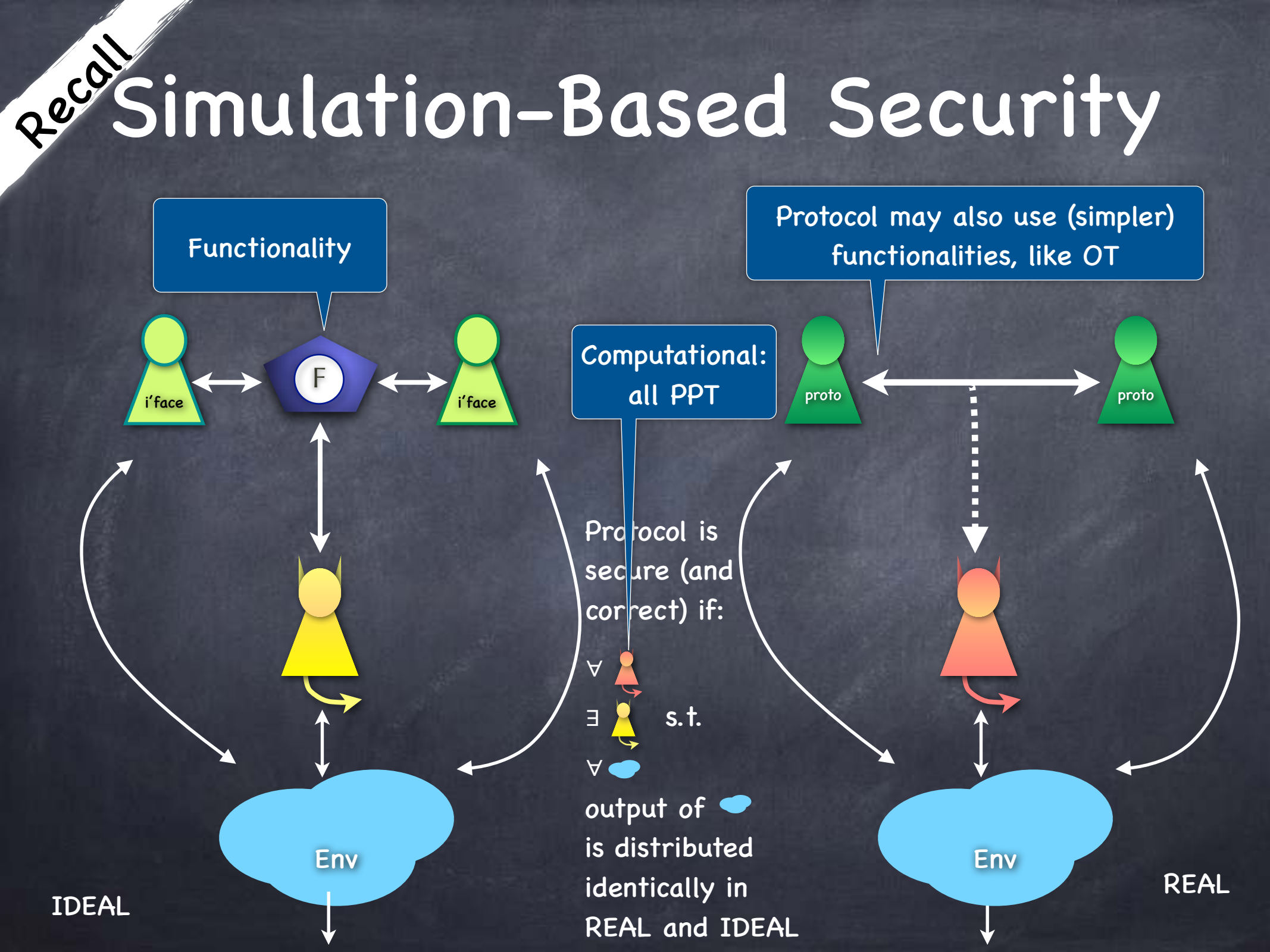
Lecture 9

Simulation (ctd.)

Zero-Knowledge Proofs

Recall

Simulation-Based Security



Recall

Example: Coin-Tossing

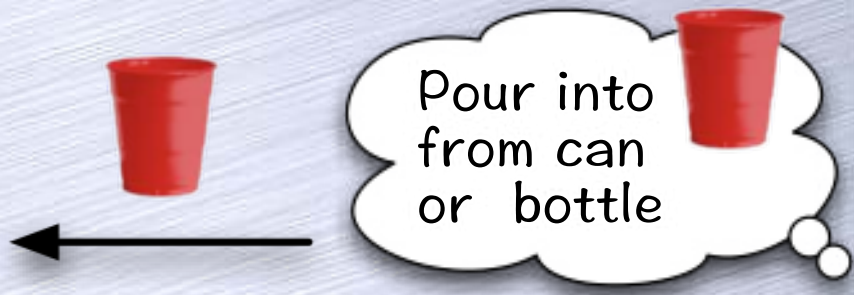
- A (fully) secure 2-party protocol for coin-tossing, given an ideal commitment functionality F_{com}
  - Alice sends $a \in \{0,1\}$ to F_{com}. (Bob gets "committed" from F_{com})
  - Bob sends $b \in \{0,1\}$ to Alice
  - Alice sends "open" to F_{com}. (Bob gets a from F_{com})
  - Both output $c=a \oplus b$
- 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 $b = a \oplus c$.
    (Block output if Alice doesn't send "open" to F_{com}.)
  - 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

Zero-Knowledge Proof

- In cryptographic settings, often need to be able to verify various claims
  - e.g., 3 encryptions A,B,C are of values a,b,c s.t. a=b+c
  - Proof 1: Reveal a,b,c and how they get encrypted into A,B,C
  - Proof 2: Without revealing anything at all about a,b,c except the fact that a=b+c ?
    - Zero-Knowledge Proof!
- Important application to secure multi-party computation: to upgrade the security of MPC protocols from security against passive corruption to security against active corruption
  - (Next time)

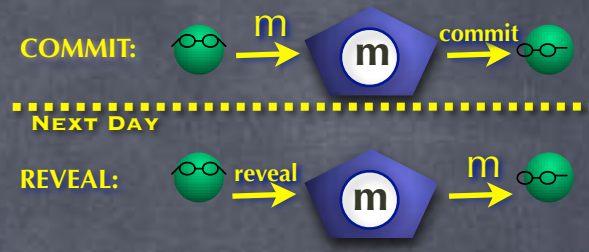
An Example

- Coke in bottle or can
  - Prover claims: coke in bottle and coke in can are different
- ZK proof:
  - prover tells whether cup was filled from can or bottle
  - repeat till verifier is convinced

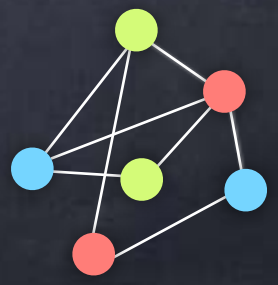


can/bottle

Commitment

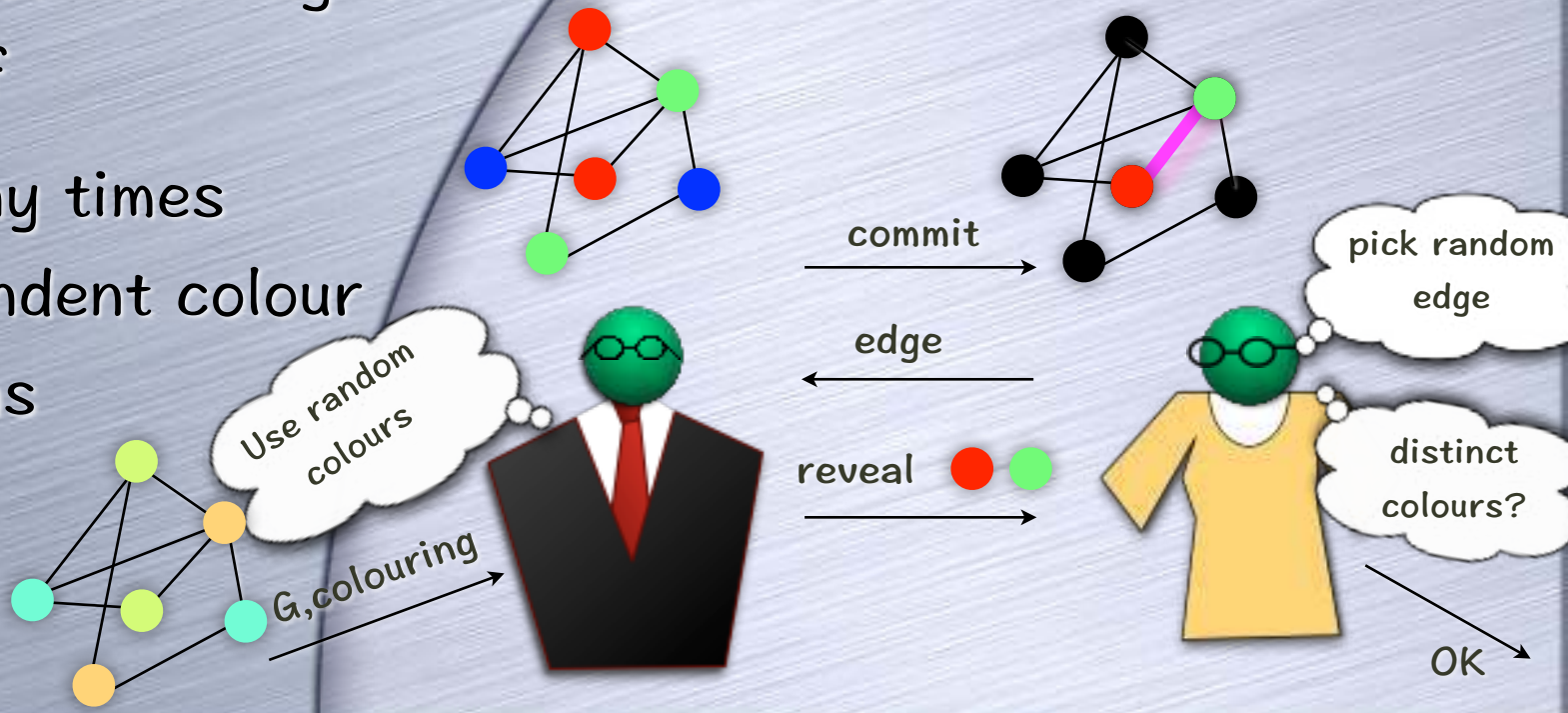


- Recall the functionality of **Commitment**:
  - Committing to a value: Alice puts the message in a box, locks it, and sends the locked box to Bob, who learns nothing about the message
  - Revealing a value: Alice sends the key to Bob. At this point she can't influence the message that Bob will get on opening the box.
- Implementation in the Random Oracle Model: Commit(x) = H(x,r) where r is a long enough random string, and H is a random hash function (available as an oracle) with a long enough output. To reveal, send (x,r).
  - ⚠ ROM is a heuristic model: Can do provably impossible tasks in this model!
- An Example: To prove that the nodes of a graph can be coloured with at most 3 colours, so that adjacent nodes have different colours



A ZK Proof for Graph Colourability

- Uses a commitment protocol as a subroutine
- At least 1/#edges probability of catching a wrong proof
- Repeat many times with independent colour permutations

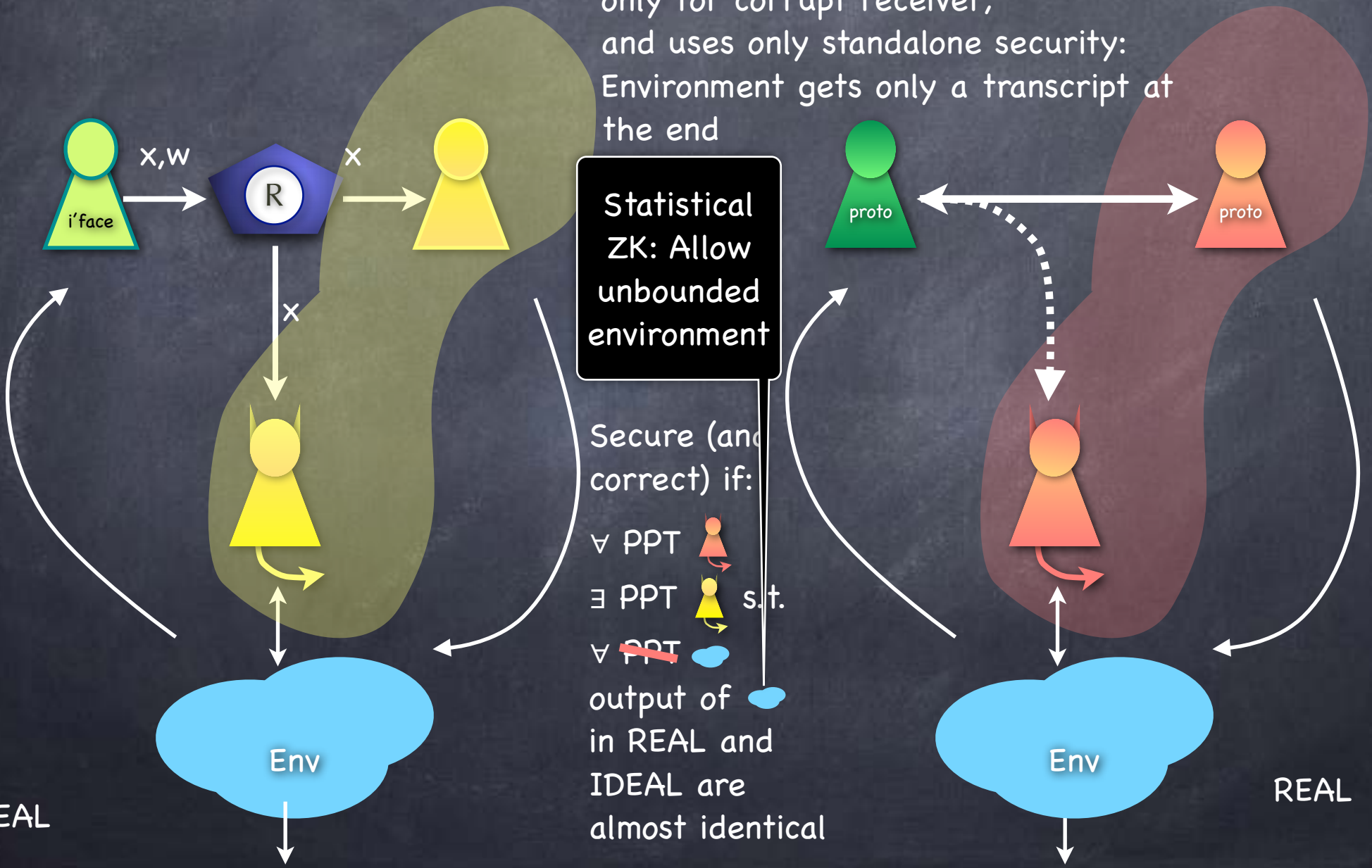


ZK Proofs Vocabulary

- Statements: Of the form "$\exists w$ s.t. relation R(x,w) holds", where R defines a class of statements, and x specifies the particular statement (which is a common input to prover and verifier)
  - e.g., Given a graph G, $\exists$ a colouring ϕ s.t. Valid(G,ϕ) holds
  - The relation R can be efficiently verified (polynomial time in size of x)
    - Set L = { x | $\exists w$ R(x,w) holds } is a language in NP
  - w is called a "witness" for $x \in L$
- **Completeness**: If prover & verifier are honest, for all $x \in L$, and prover given a valid witness w, verifier will always accept
- **Soundness**: If $x \notin L$, no matter what a cheating prover does, an honest verifier will reject (except with negligible probability)
  - Proof-of-Knowledge: A stronger soundness notion
- **Zero-Knowledge**: A (corrupt) verifier's view can be simulated (honest prover, $x \in L$)
- Soundness can be required to hold even against computationally unbounded provers
  - ZK Argument system: Like a ZK proof system, but soundness only against PPT adversaries

ZK Property

Classical definition uses simulation only for corrupt receiver;
and uses only standalone security:
Environment gets only a transcript at the end



Statistical ZK: Allow unbounded environment

Secure (and correct) if:

∀ PPT

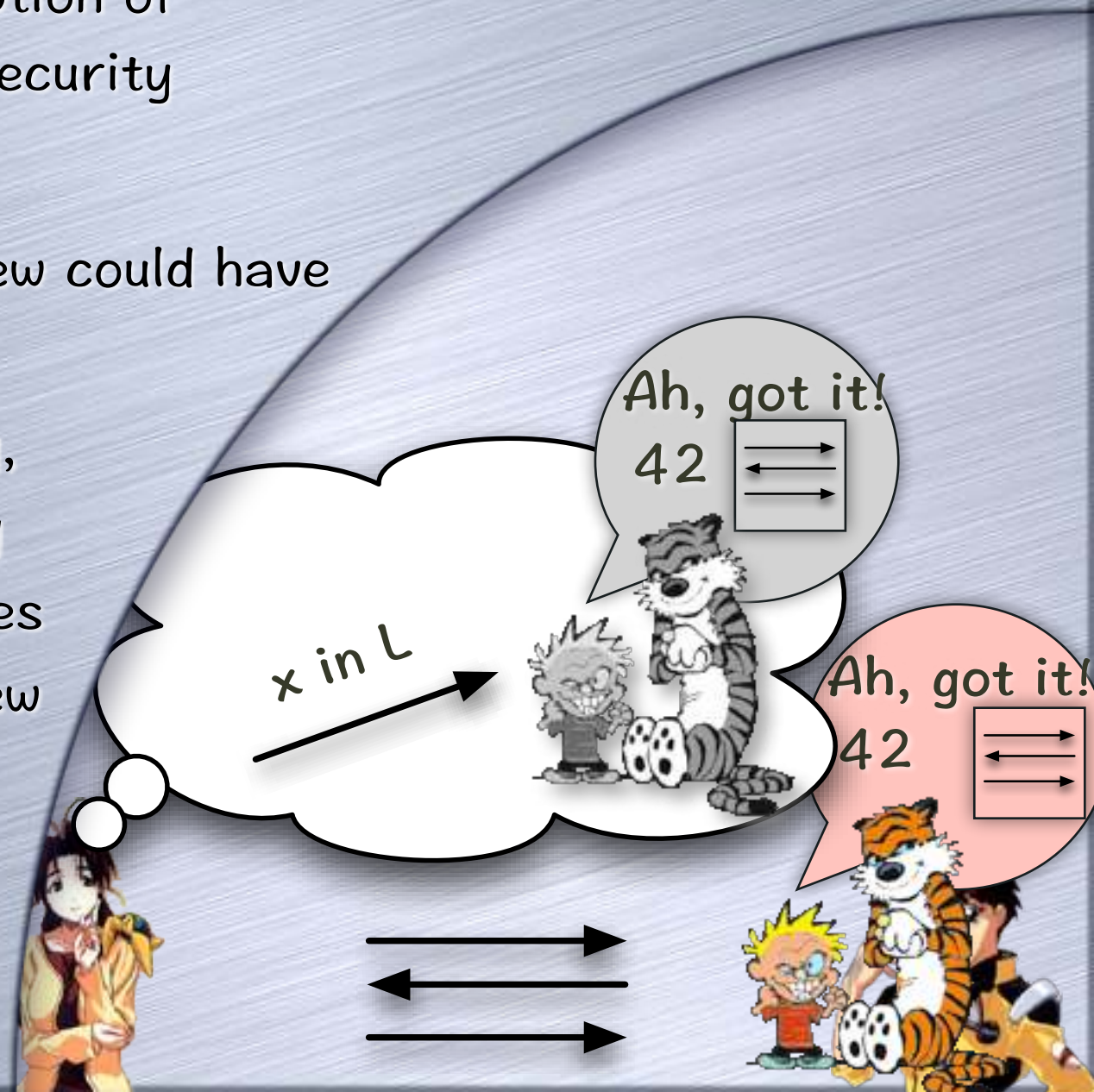
∃ PPT s.t.

∀ ~~PPT~~

output of in REAL and IDEAL are almost identical

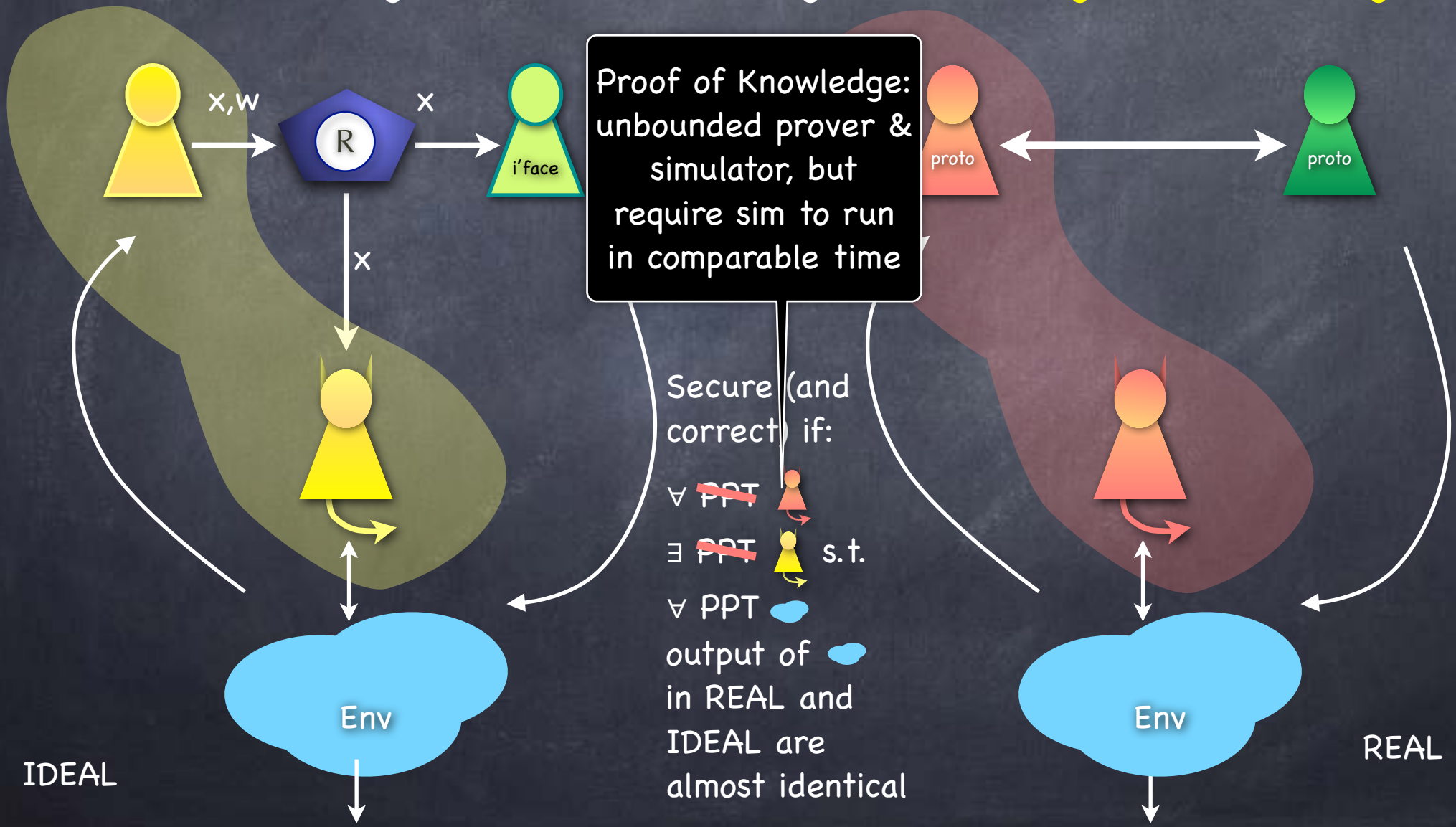
In Other Pictures…

- Simulation only for corruption of verifier and stand-alone security
- ZK Property:
  - A corrupt verifier's view could have been "simulated"
  - ∀ adversarial strategy, ∃ a simulation strategy which, ∀x ∈ L, produces an indistinguishable view
- Completeness and soundness defined separately



Two-Sided Simulation

- Require simulation also when prover is corrupt
  - Then simulator is a witness extractor
- Adding this (in standalone setting) makes it an Argument of Knowledge



Some ZK Proof Techniques

- Classic protocols for NP complete problems
  - e.g., graph 3 colorability (with standalone-secure commitment, instantiated using, say, one-way permutations)
  - Any NP language L has a ZK proof system via reduction to an NP complete problem
- More generally, by committing to a "probabilistically checkable proof"
  - Can improve the communication efficiency
- More efficient protocols for specific NP languages (avoiding the overhead of reduction to NP complete languages)
  - e.g., Proof of equality of discrete logs (coming up)
- Using MPC as a robust encoding
  - "MPC-in-the-head" (later)
- Non-interactive variants (later)
  - Often in the random-oracle model