Advanced Tools from Modern Cryptography

Lecture 9

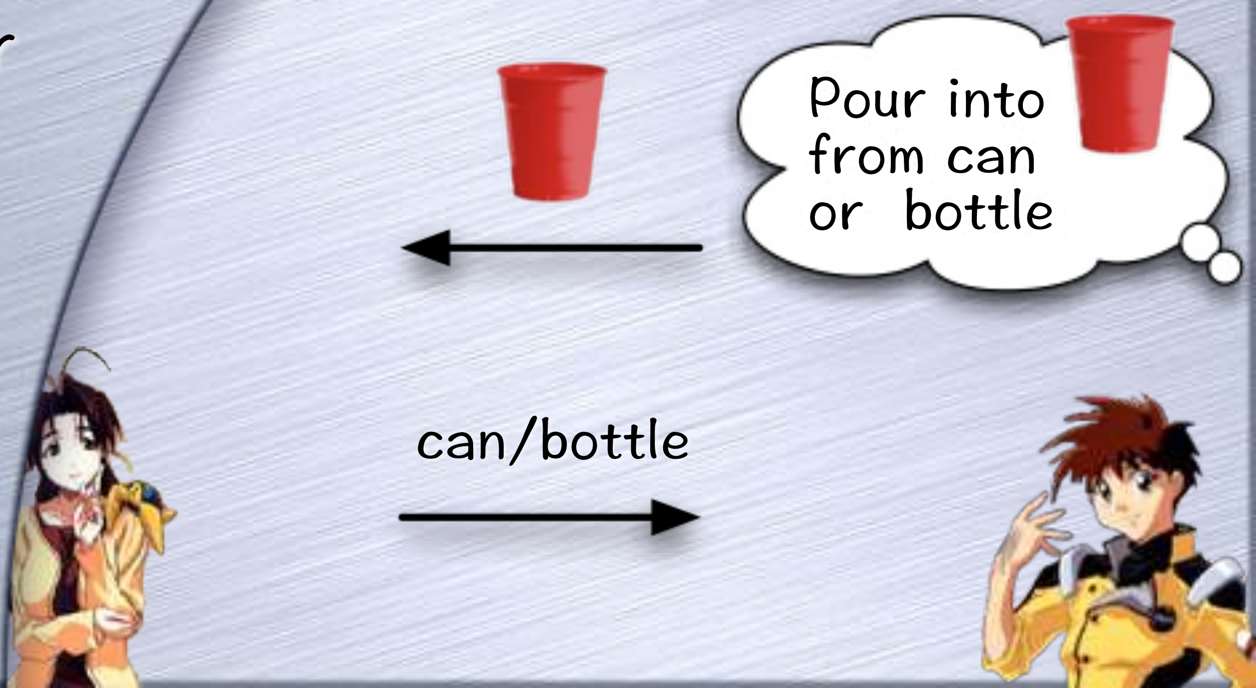
Zero-Knowledge Proofs

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)

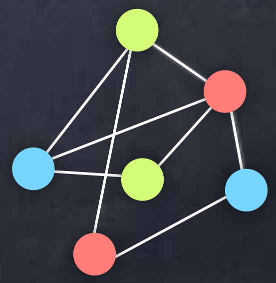
An Example

- **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



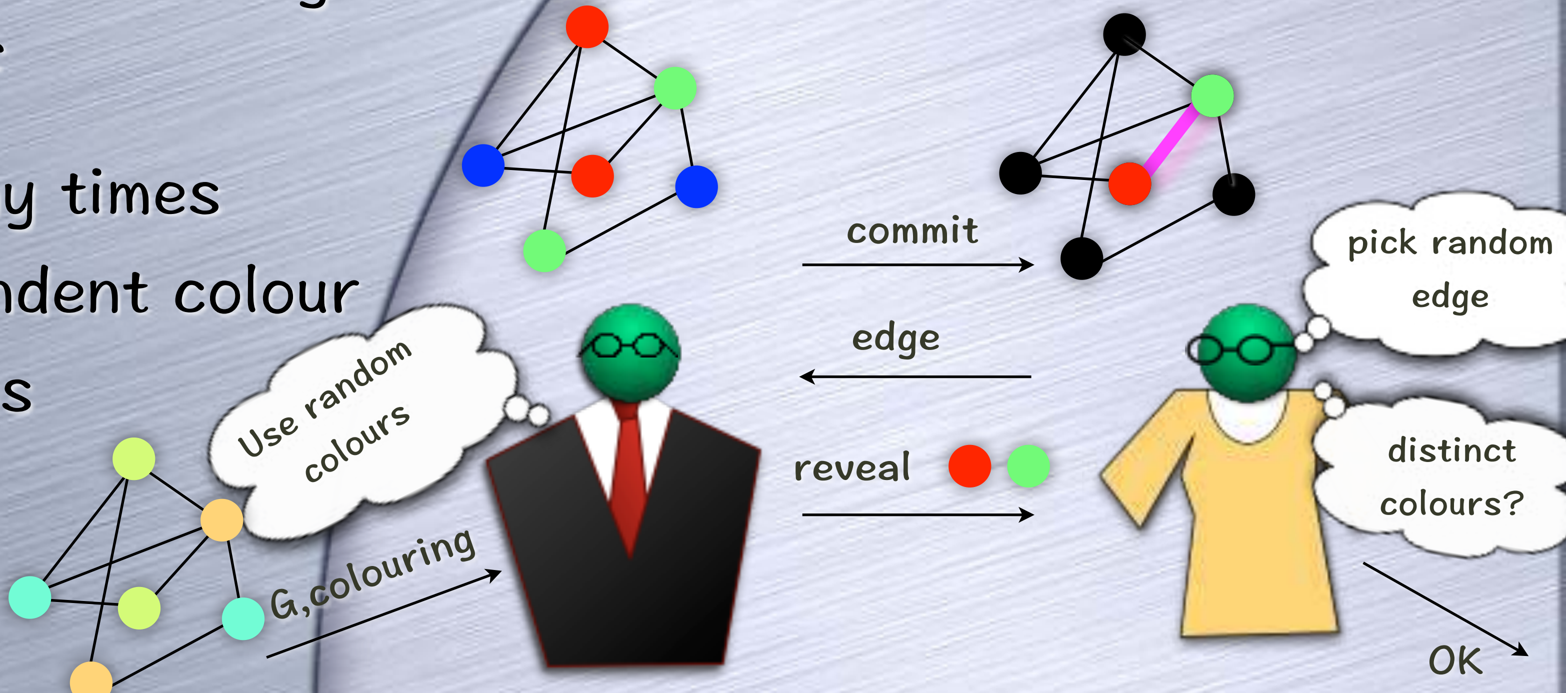
Commitment

- 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). 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

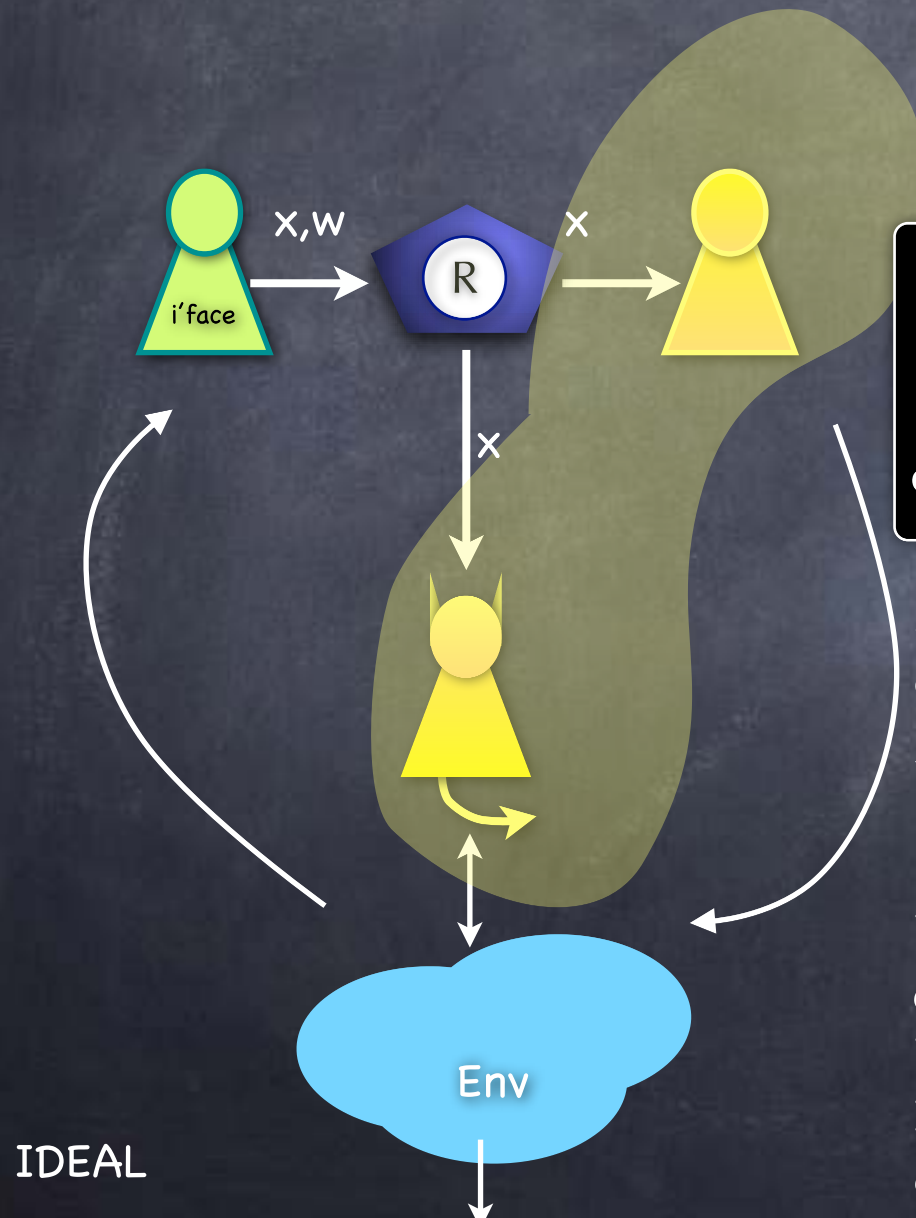


ZK Proofs Vocabulary

- Statements: Of the form "∃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, ∃ 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 | ∃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:

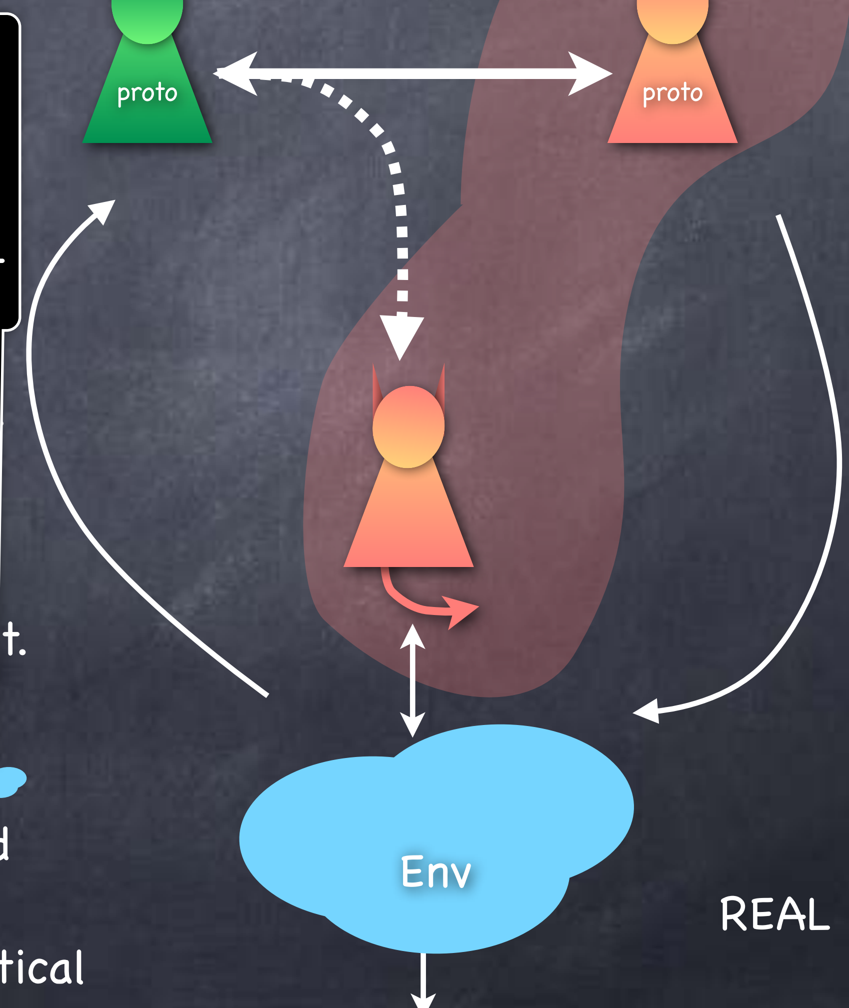
∀ PPT

∃ PPT s.t.

∀ ~~PPT~~

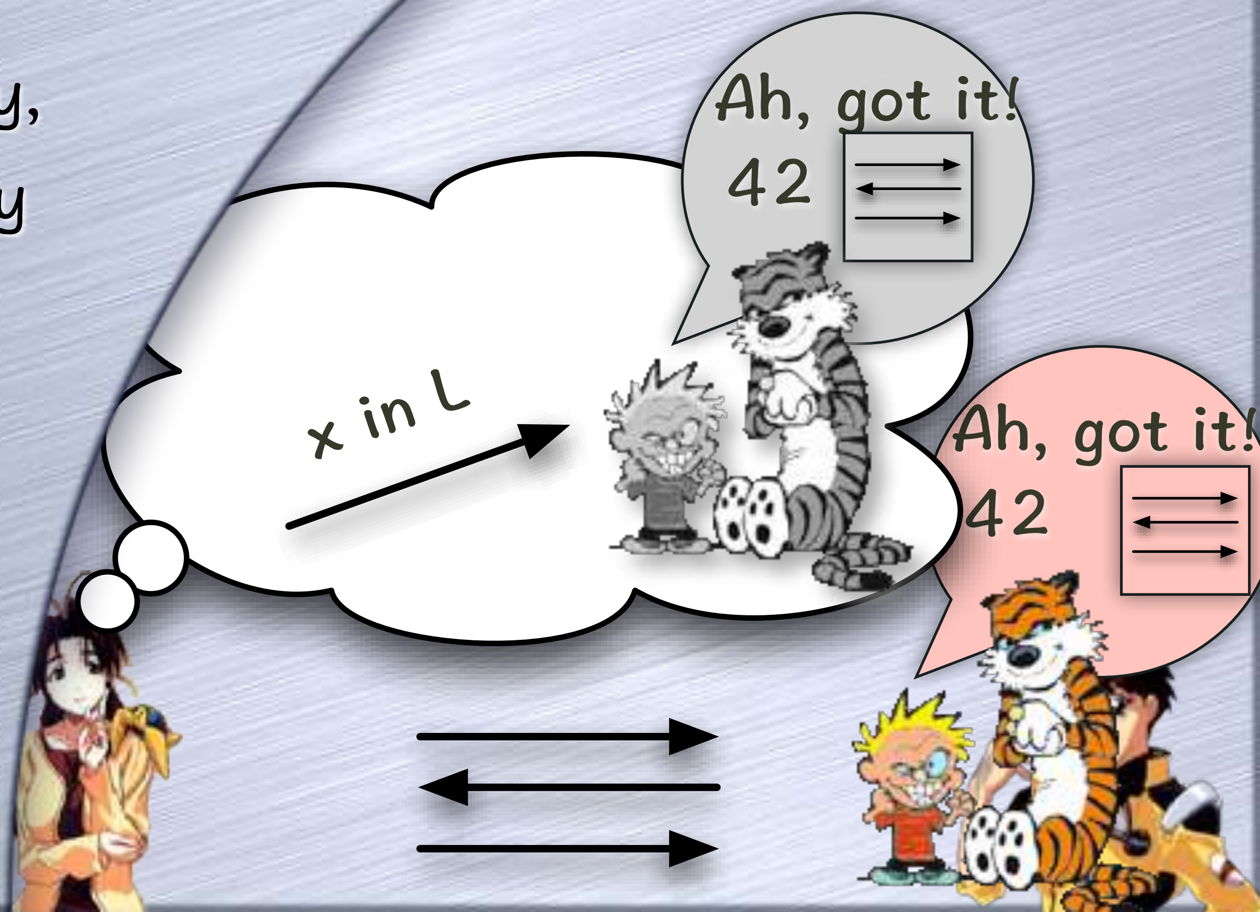
output of

in REAL and IDEAL are almost identical



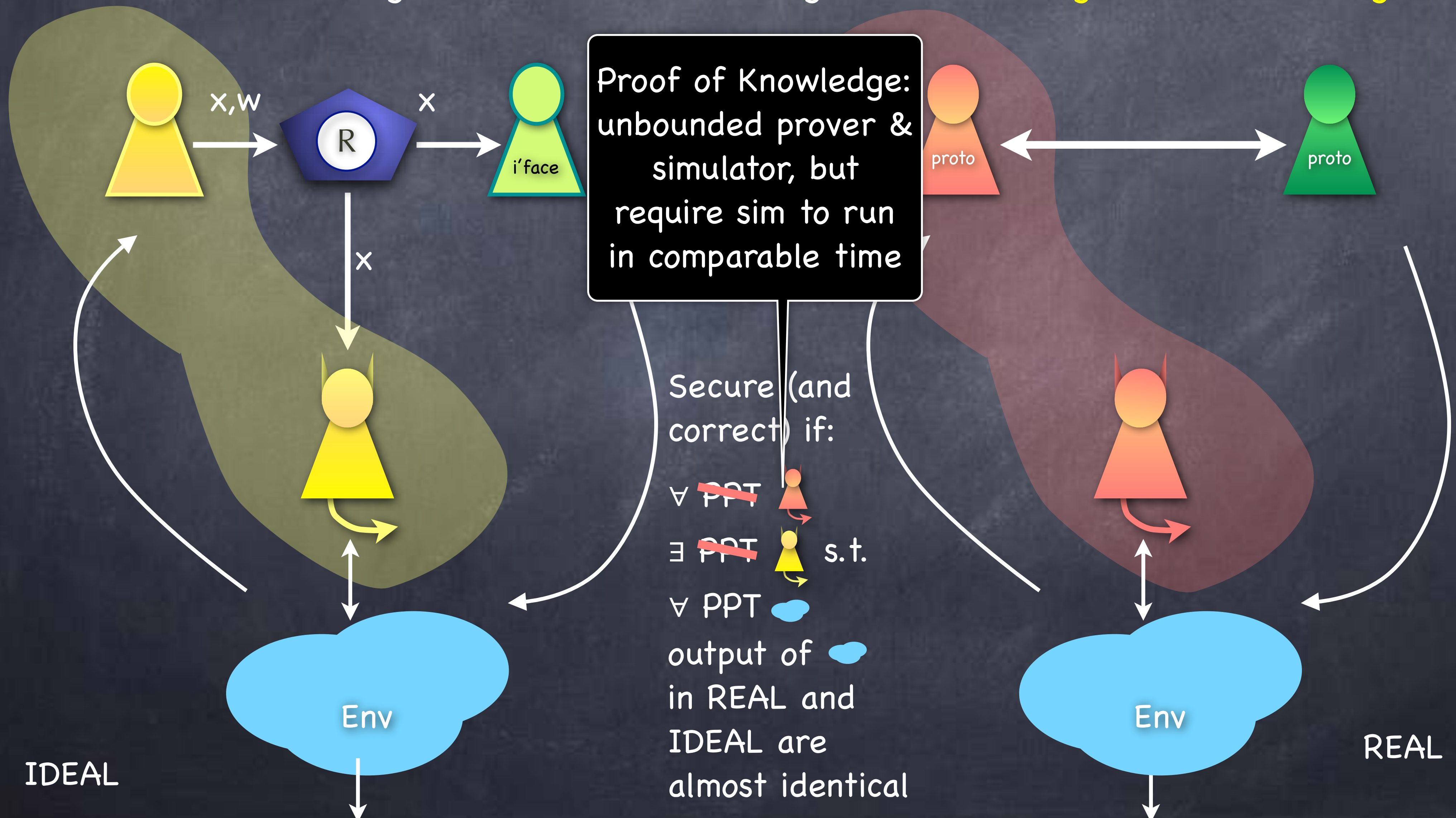
Simplified Picture

- ZK Property:
  - A corrupt verifier's view could have been "simulated"
  - ∀ adversarial strategy, ∃ a simulation strategy which 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

Discrete Logarithm

- In a cyclic group, all elements can be written as $g^0, g^1, \ldots, g^{n-1}$
- Given $D \in G$ and a generator g, $\exists$ unique $d \in [0,n-1]$ s.t. $D = g^d$
  - Discrete logarithm of D w.r.t. g
- In many groups, finding the discrete logarithm is computationally hard
- Many commitment schemes, encryption schemes, collision-resistant hash functions etc. based on the hardness of discrete logarithm and related problems

Honest-Verifier ZK Proofs

- A ZK Proof of knowledge of discrete log of $R=g^r$
  - $P \longrightarrow V$: $U := g^u$
    $V \longrightarrow P$: v
    $P \longrightarrow V$: $w := rv + u$ (modulo order of the group)
    V checks: $g^w = R^v U$
  - Proof of Knowledge:
    - Firstly, $g^w = R^v U \Rightarrow w = rv+u$, where $U = g^u$
    - If after sending U, P could respond to two different values of v: $w_1 = rv_1 + u$ and $w_2 = rv_2 + u$, then can solve for r
  - HVZK: simulation picks w, v first and sets $U = g^w/R^v$

HVZK and Special Soundness

- HVZK: Simulation for honest (passively corrupt) verifier
  - e.g. in PoK of discrete log, simulator picks (v,w) first and computes U (without knowing u). Relies on verifier to pick v independent of U.
- Special soundness: given (U,v,w) and (U,v′,w′) s.t. v≠v′ and both accepted by verifier, can derive a witness
  - e.g. solve r from w=rv+u and w′=rv′+u (given v,w,v′,w′)
  - Proof of knowledge (in stand-alone setting): for each U s.t. prover has significant probability of being able to convince, a simulator can extract r from the prover with overwhelming probability (using "rewinding")
  - Can amplify soundness using parallel repetition: still 3 rounds

Honest-Verifier ZK Proofs

- ZK PoK to prove equality of discrete logs for $((g,R),(C,D))$, i.e., $R = g^r$ and $D = C^r$ [Chaum-Pederson]
- $P \longrightarrow V$: $(U,T) := (g^u, C^u)$
  $V \longrightarrow P$: v
  $P \longrightarrow V$: $w := rv+u$
  V checks: $g^w = R^v U$ and $C^w = D^v T$
- Proof of Knowledge:
  - $g^w = R^v U$, $C^w = D^v T$ $\Longrightarrow$ $w = rv+u = r'v+u'$ where $U=g^u$, $T=g^{u'}$ and $R=g^r$, $D=C^{r'}$
  - If after sending (U,T) P could respond to two different values of v: $rv_1 + u = r'v_1 + u'$ and $rv_2 + u = r'v_2 + u'$, then $r=r'$
- HVZK: simulation picks w, v first and sets $U=g^w/R^v$, $T=C^w/D^v$