Advanced Tools from Modern Cryptography

Lecture 10

MPC: GMW Paradigm. Composition.

MPC: Story So Far

- Security against passive corruption
  - "Basic GMW" using OT, Yao's Garbled Circuits using OT, "Passive-BGW" with honest majority
- Security against active corruption (no honest majority)
  - ZK proofs
  - GMW paradigm

GMW Paradigm

- Run a passive-secure protocol Π, but let each party "verify" that the others are following the protocol correctly
  - Correctly: pick arbitrary inputs and arbitrary randomness first, but then follow the specified program
- Need to prove that each message was correctly computed, right when it is sent
  - If proof required only at the end, too late!
- Proving $\exists$ input, rand, s.t. next-message_Π (input,rand,messages) equals the message being sent
  - Should use the same input and randomness through out!
  - ZK proofs not enough

Commit & Prove

- To prove ∃ input, rand, s.t. next-message$_\Pi$(input,rand,messages) equals the message being sent
- Commit-and-Prove functionality: F_{CaP}
  - Alice sends v to F_{CaP}, which sends "committed" to Bob
  - Subsequently, for i=1,2,... Alice sends a function f_i (represented as a circuit) to F_{CaP}, which sends $(f_i,f_i(v))$ to Bob
    - More generally, Alice sends (f_i,w_i) and F_{CaP} sends $(f_i,f_i(v,w_i))$ to Bob (i.e., without revealing w_i)
  - Note: same v used in all rounds
- Could "securely implement" F_{CaP} using a "plain" commitment of v (i.e., not using F_{com}), and proving statements about it using F_{ZK}
  - Or can adapt the MPC-in-the-head protocol for F_{ZK} using F_{OT} instead of F_{Com}

GMW Paradigm

- Run a passive-secure protocol Π, but let each party "verify" that the others are following the protocol correctly
  - Correctly: pick arbitrary inputs and arbitrary randomness first, but then follow the specified program
- Each party proves using F_{CaP} that each message was correctly computed, for the same committed inputs and randomness
  - f_i defined so that $f_i(v) = 1$ iff Π produces message m_i on input/randomness v for the proving party, given the transcript so far
    - All communication in Π assumed to be over public channels

Composition

- We built an active-secure protocol using access to ideal F_{CaP} functionality
  - Is it OK to "replace" it by a secure protocol for F_{CaP}?
  - More generally, can we replace an ideal functionality running in an arbitrary environment with a secure protocol?
  - Depends on the exact definition of security!
    - Looking ahead: OK for both UC security and passive security
    - **Not OK for standalone security**

An example

- An auction, with Alice and Bob bidding:
  - A bid is an integer in the range [0,100]
  - Alice can bid only even integers and Bob odd integers
  - Person with the higher bid wins
- Goal: find out the winning bid (winner & amount) without revealing anything more about the losing bid (beyond what is revealed by the winning bid)
  - F_{max} : Output the higher bid to both parties (Domains are disjoint)

An example

- Secure protocol:
  - Count down from 100
  - At each even round Alice announces whether her bid equals the current count; at each odd round Bob does the same
  - Stop if a party says yes
- Dutch flower auction

Perfect Standalone Security
But doesn't compose!

Attack on Dutch Flower Auction

- Alice and Bob are taking part in two auctions
- Alice's goal: ensure that Bob wins at least one auction with some bid z, and the winning bid in the other auction $\in \{z,z-1\}$
- Easy in the protocol: run the two protocols lockstep. Wait till Bob says yes in one. Done if Bob says yes in the other simultaneously. Else Alice will say yes in the next round.
- Why is this an attack?
    - Impossible to ensure this in IDEAL!

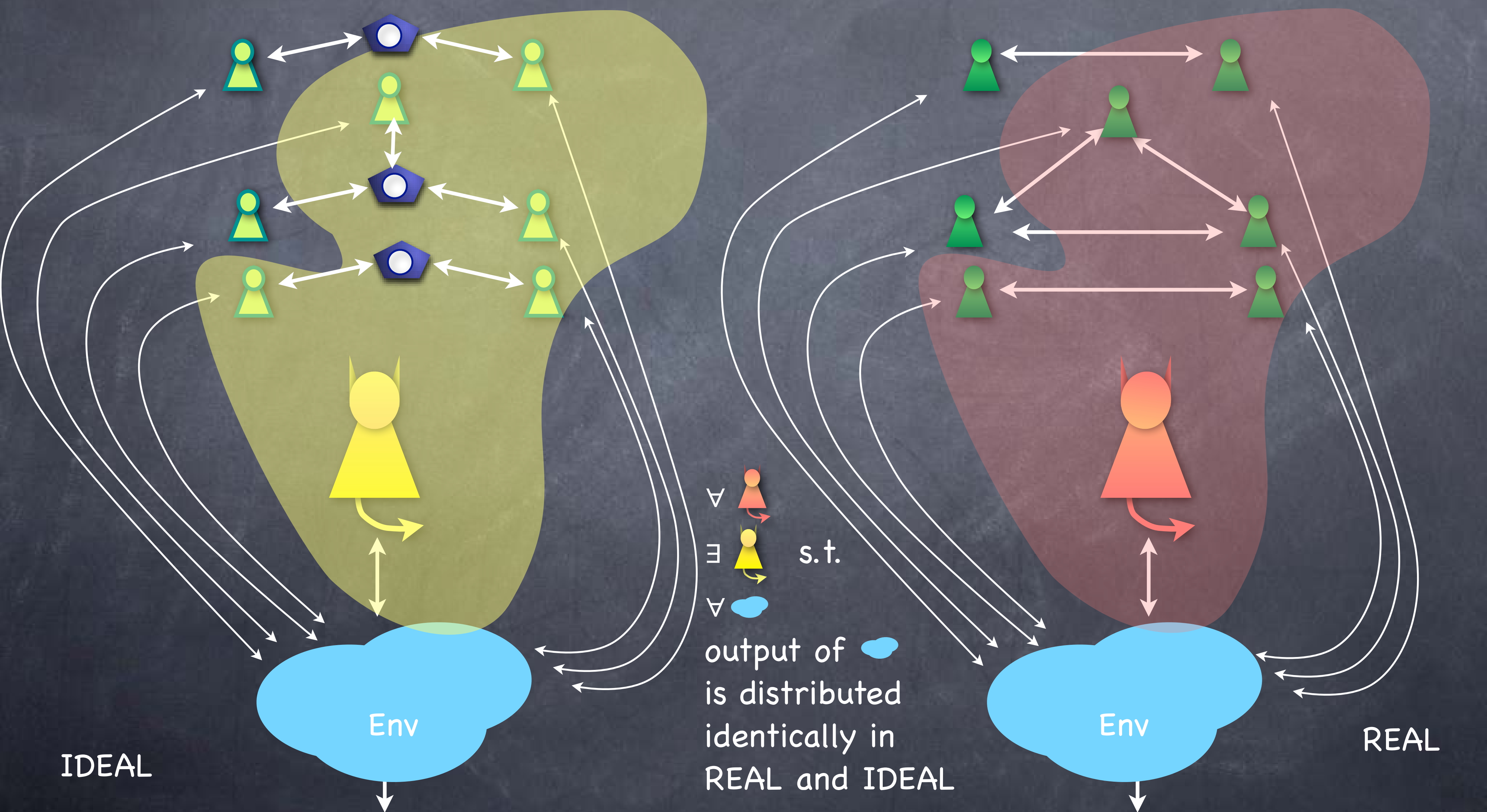
Attack on Dutch Flower Auction

- Alice's goal: ensure that Bob wins at least one auction with some bid z, and the winning bid in the other auction $\in \{z, z-1\}$
- Impossible to ensure this in IDEAL!
- Alice can get a result in one session, before running the other. But what should she submit as her input x in the first one?
  - Trouble if $x \neq 0$, because she could win (i.e., $z-1=x$) and Bob's input in the other session may be $\neq x+1$
  - Trouble if $x=0$, because Bob could win with input 1 (i.e., $z=1$) and in the other session his input > 1

Composition Issues

- Standalone security definition does not ensure security when composed
- Different modes of composition
  - Sequential composition: protocols executed one after the other. Adversary communicates with the environment between executions.
  - Concurrent composition: multiple sessions (typically of the same protocol) are active at the same time, and the adversary can coordinate its actions across the sessions

Concurrent Executions

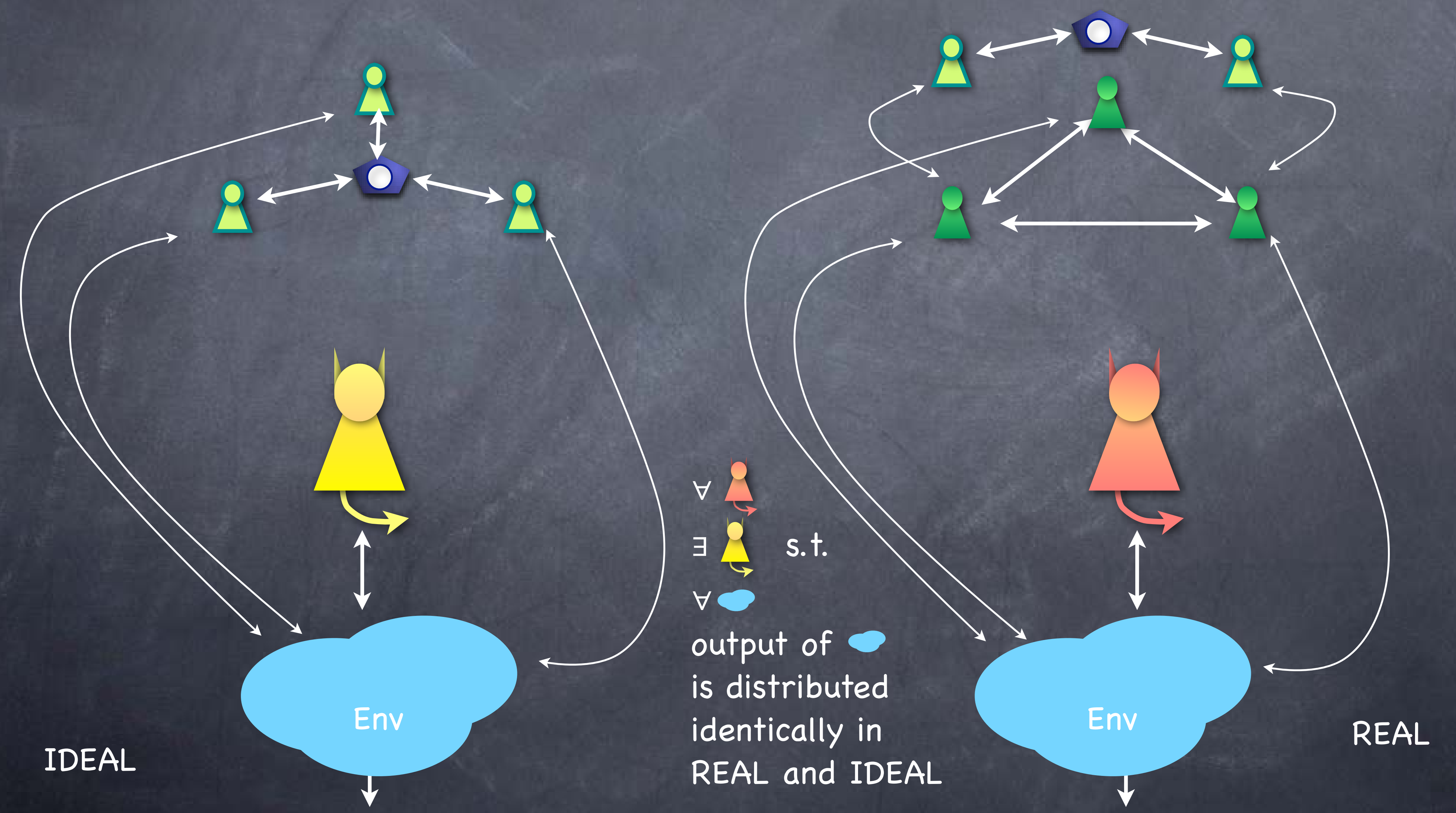


Composition Issues

- Standalone security definition does not ensure security when composed
- Different modes of composition
  - Sequential composition: protocols executed one after the other. Adversary communicates with the environment between executions.
  - Concurrent composition: multiple sessions (typically of the same protocol) are active at the same time, and the adversary can coordinate its actions across the sessions
    - Also, subroutine calls

Subroutines

A "REAL" protocol in which parties access (another) IDEAL protocol

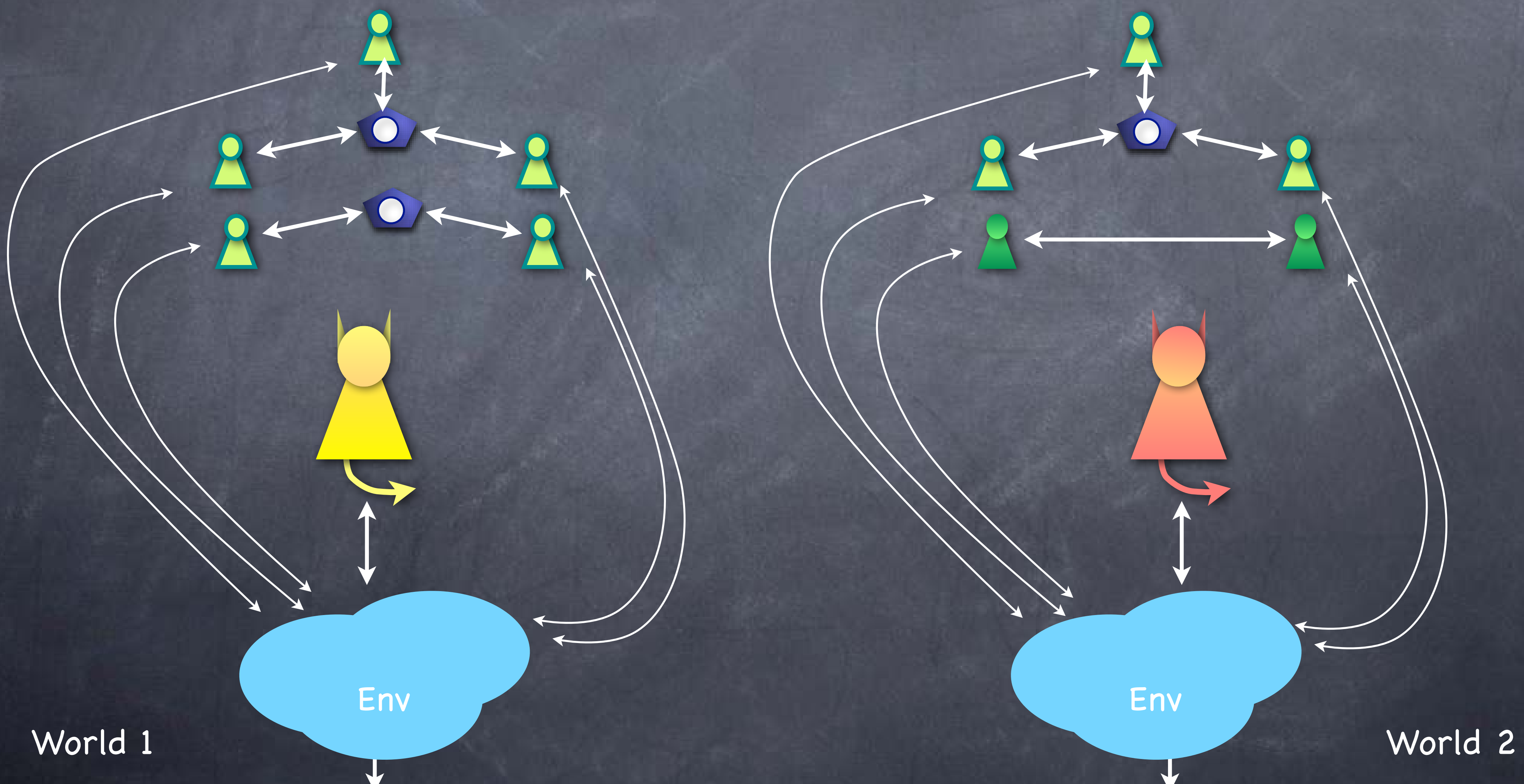


Composition Issues

- Standalone security definition does not ensure security when composed
- Different modes of composition
    - Sequential composition: protocols executed one after the other. Adversary communicates with the environment between executions.
    - Concurrent composition: multiple sessions (typically of the same protocol) are active at the same time, and the adversary can coordinate its actions across the sessions
        - Also, subroutine calls
    - Universal composition: Executed in an arbitrary environment which may include other protocol sessions (possibly calling this session as a subroutine). Live communication between environment and adversary.

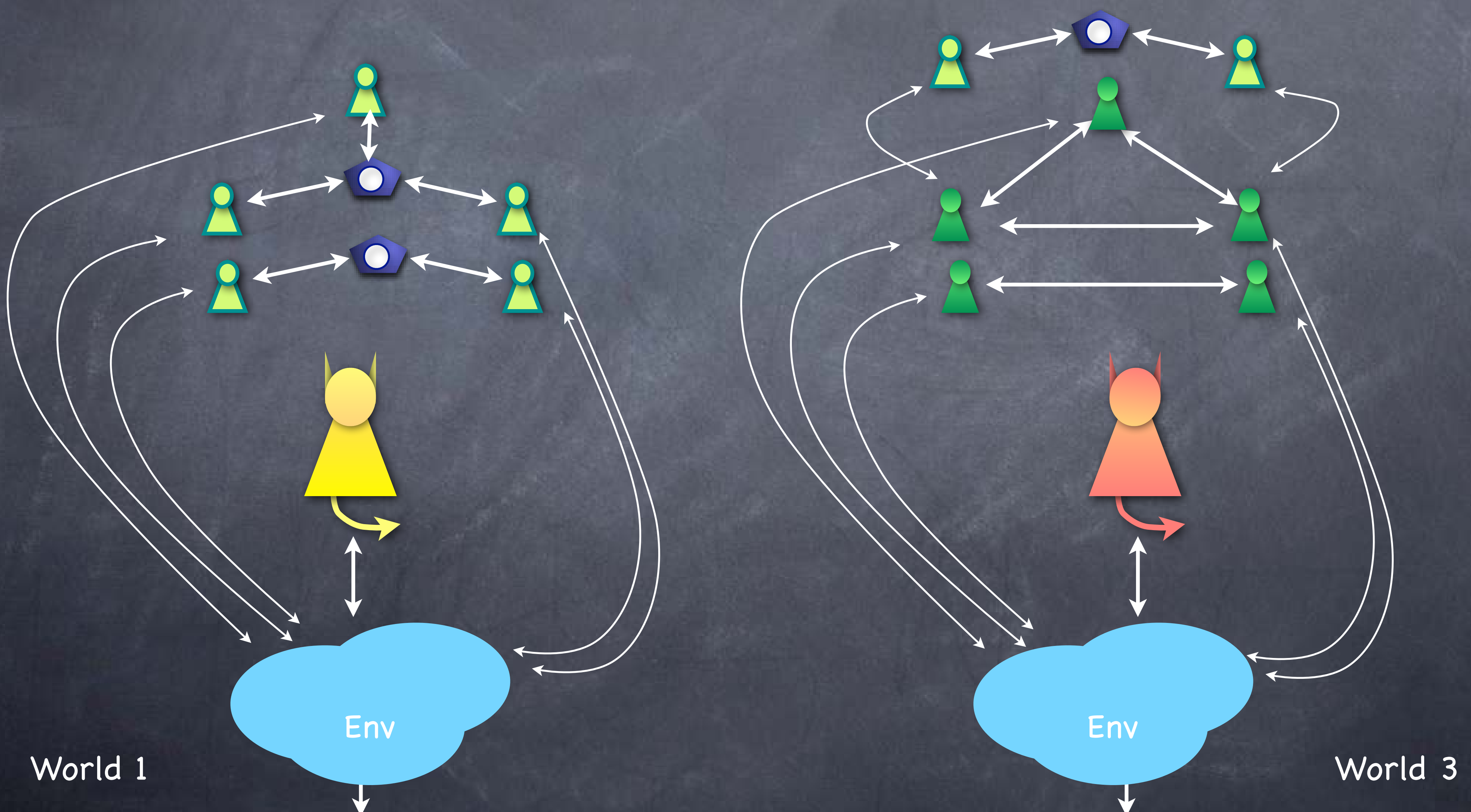
Universal Composition

Replace protocol with which is as secure, etc.

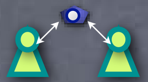


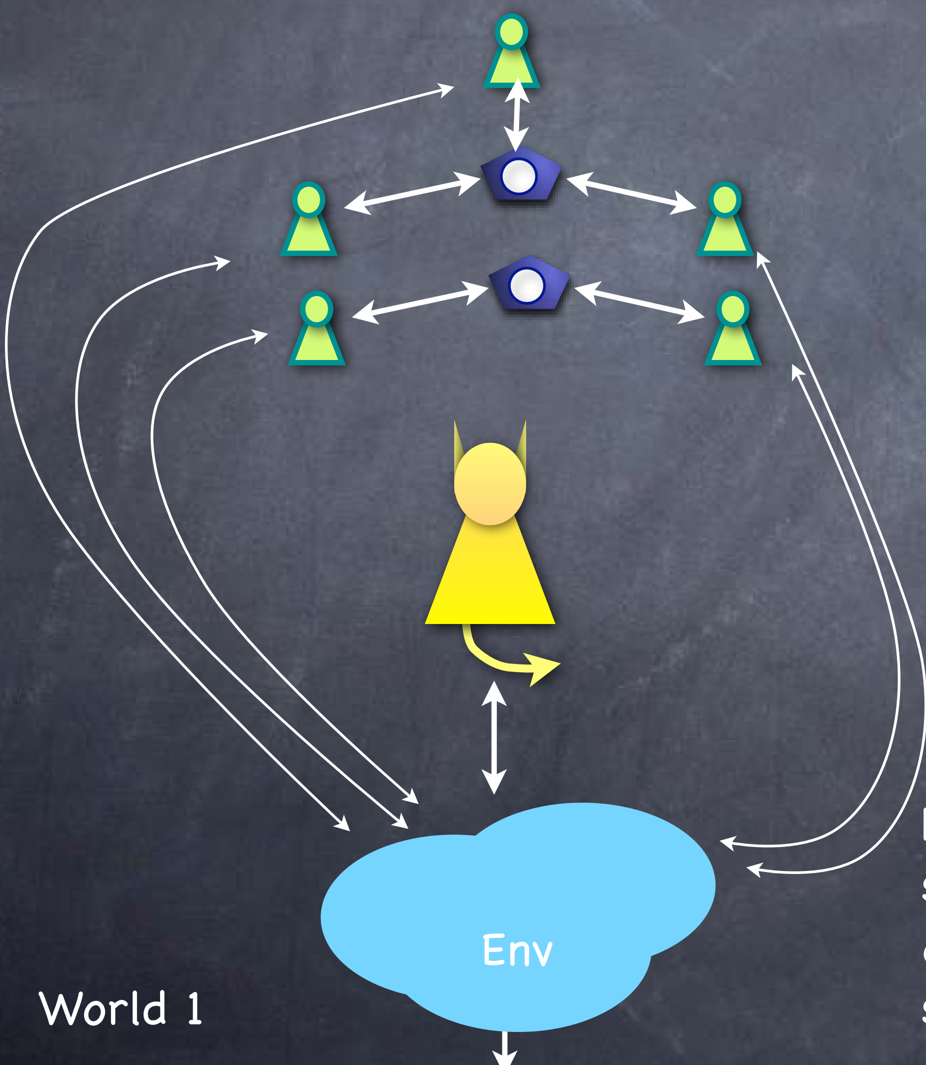
Universal Composition

Replace protocol with which is as secure, etc.



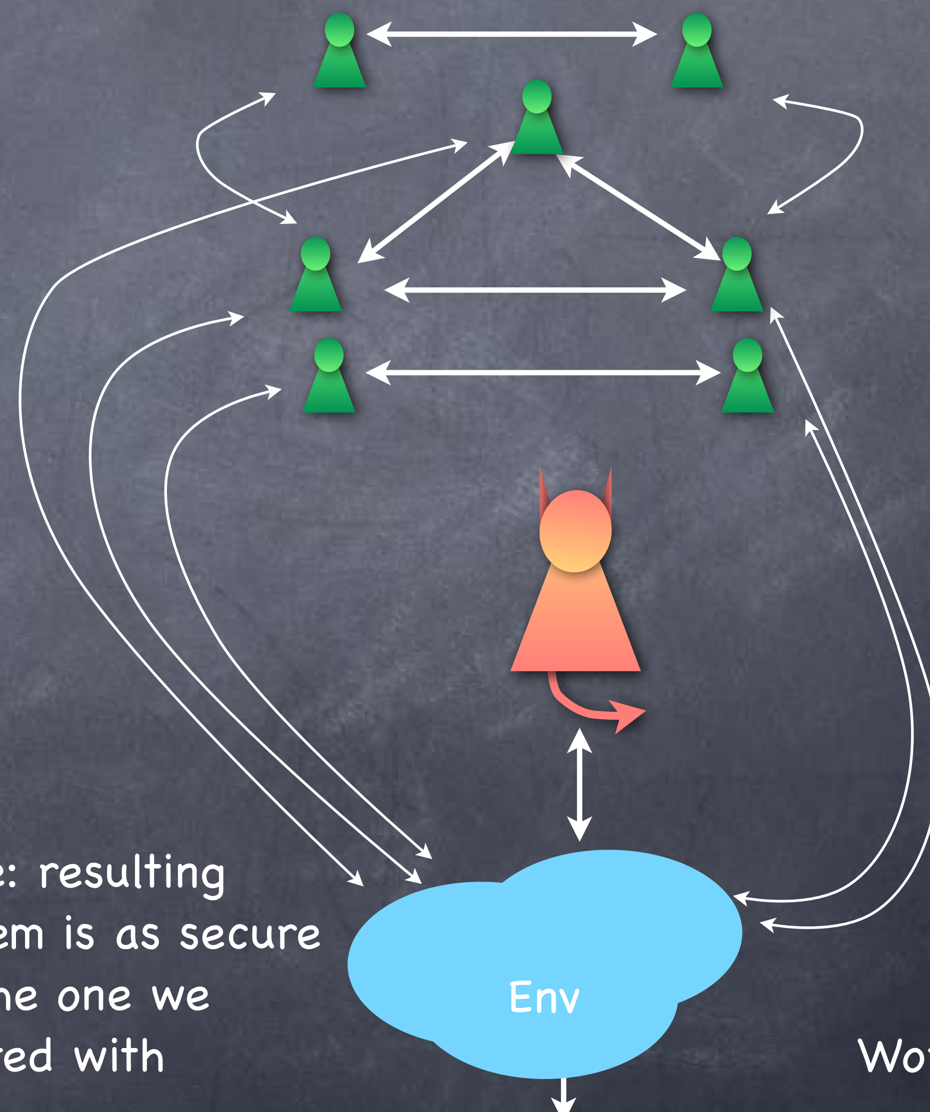
Universal Composition

Replace protocol  with  which is as secure, etc.



World 1

Hope: resulting system is as secure as the one we started with



World 4

Universal Composition

- Start from world A (think "IDEAL")
  - Repeat (for any poly number of times):
    - For some 2 "protocols" (that possibly make use of ideal functionalities) I and R such that R is as secure as I, substitute an I-session by an R-session
  - Say we obtain world B (think "REAL")
  - UC Theorem: Then world B is as secure as world A
- Gives a modular implementation of the IDEAL world