ZK Proofs (cntd.)

Universal Composition

ZK Proofs (cntd.)
Universal Composition

Lecture 16

RECALL

An Example

RECALL

An Example

- **Graph Isomorphism**

RECALL

An Example

- **Graph Isomorphism**
  - (G_0, G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$

RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ

RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol

RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$

RECALL

An Example

- **Graph Isomorphism**
  - (G_0, G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$

$G^* := \pi(G_1)$
(random π)

RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$

$G^* := \pi(G_1)$
(random π)

G^*

RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$

$G^* := \pi(G_1)$
(random π)

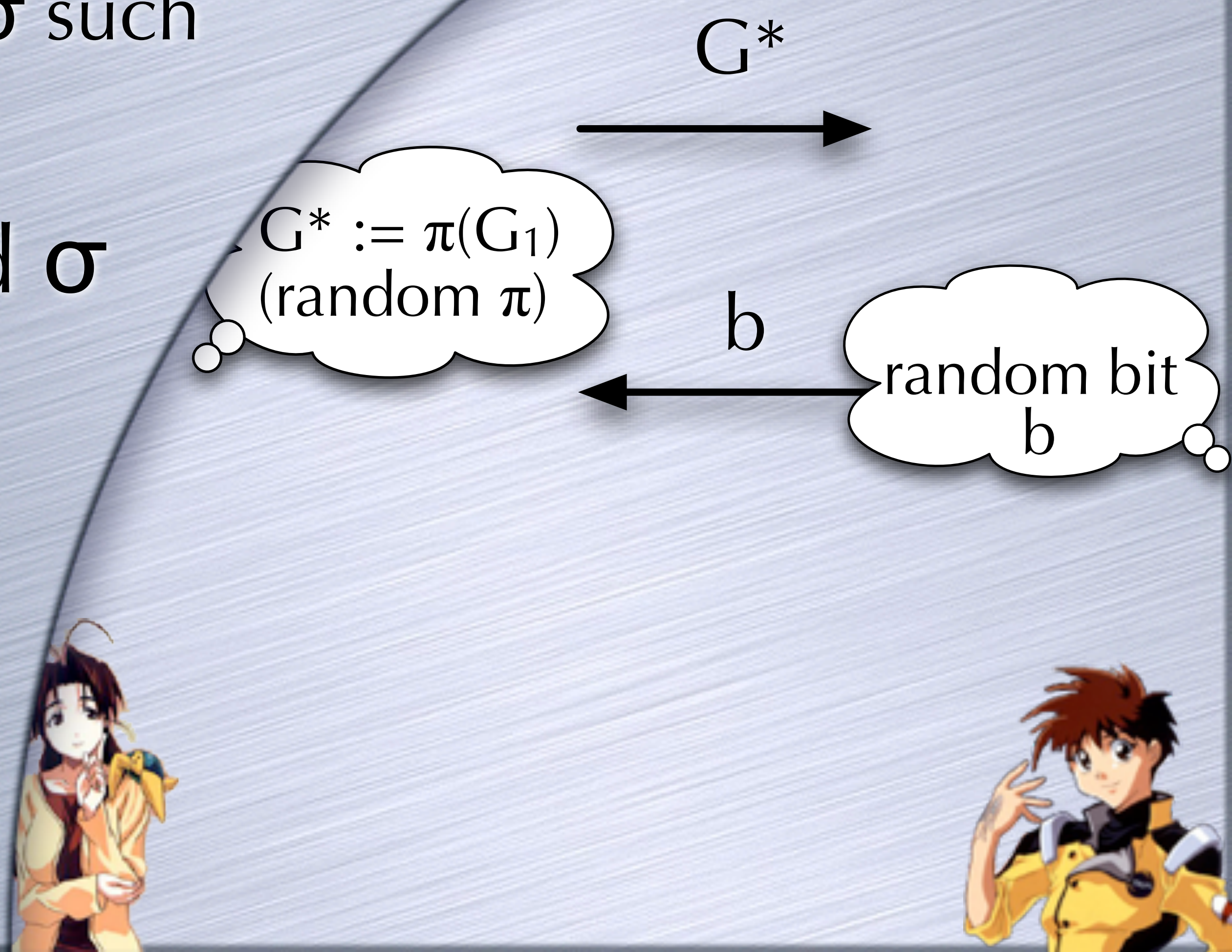
$\xrightarrow{G^*}$

random bit b

RECALL

An Example

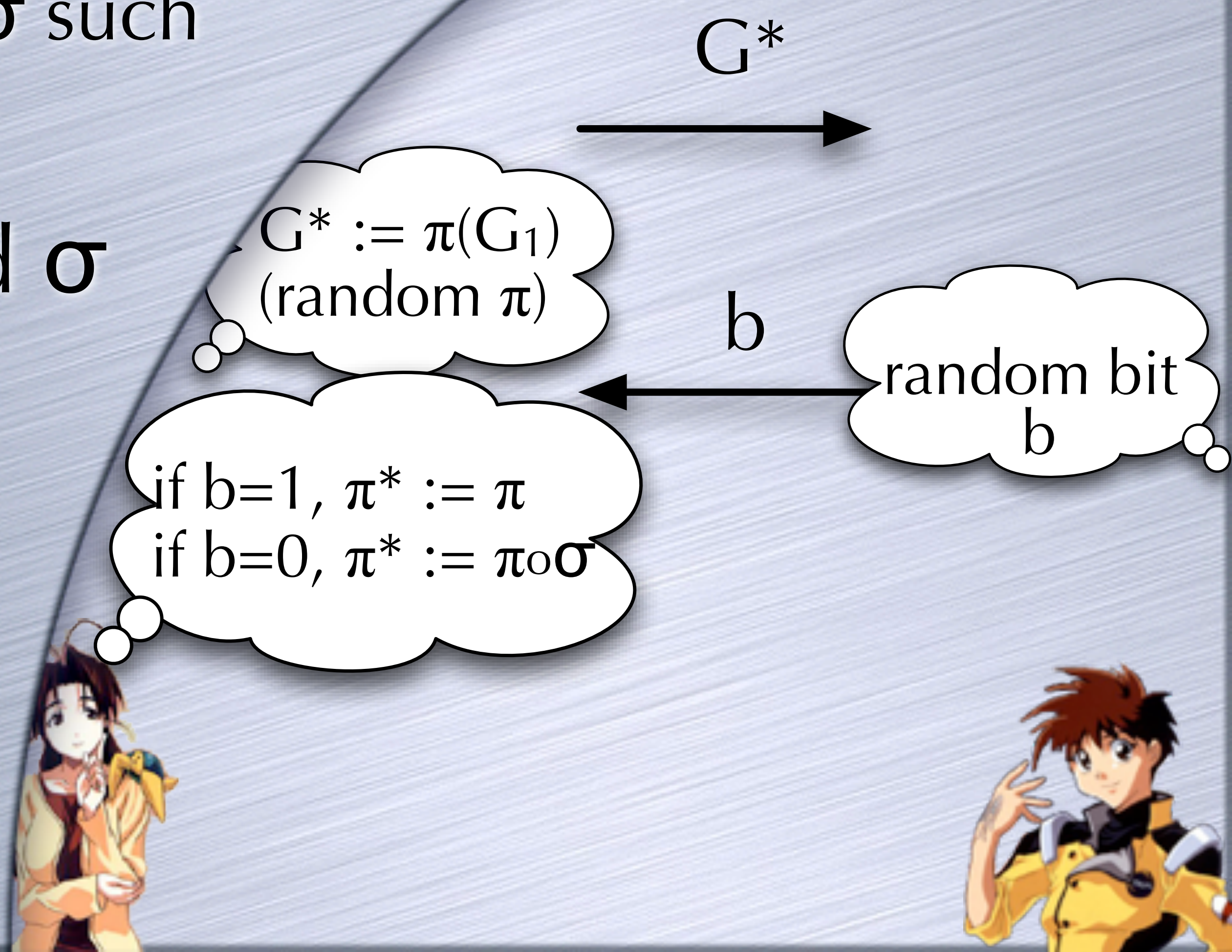
- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$



RECALL

An Example

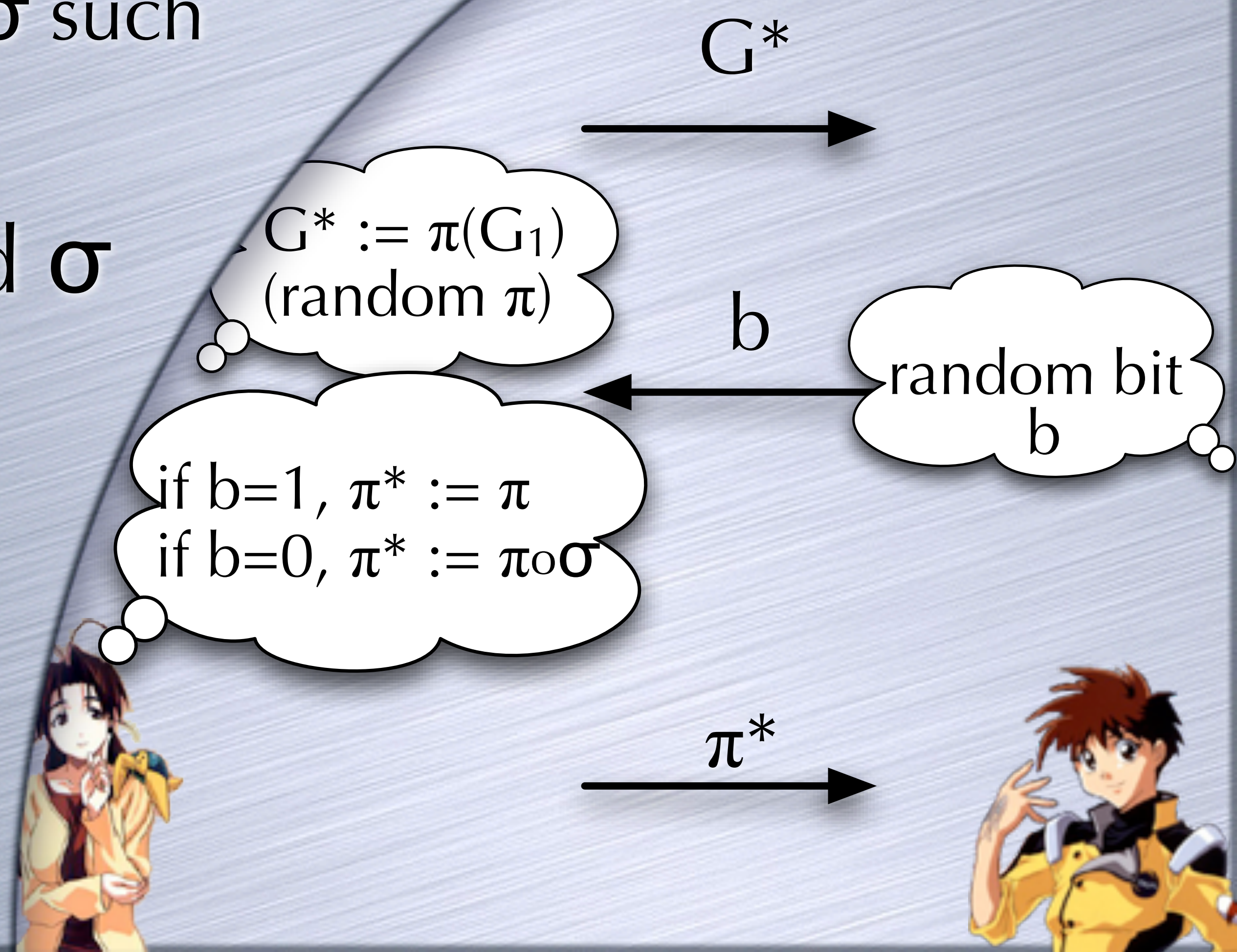
- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$



RECALL

An Example

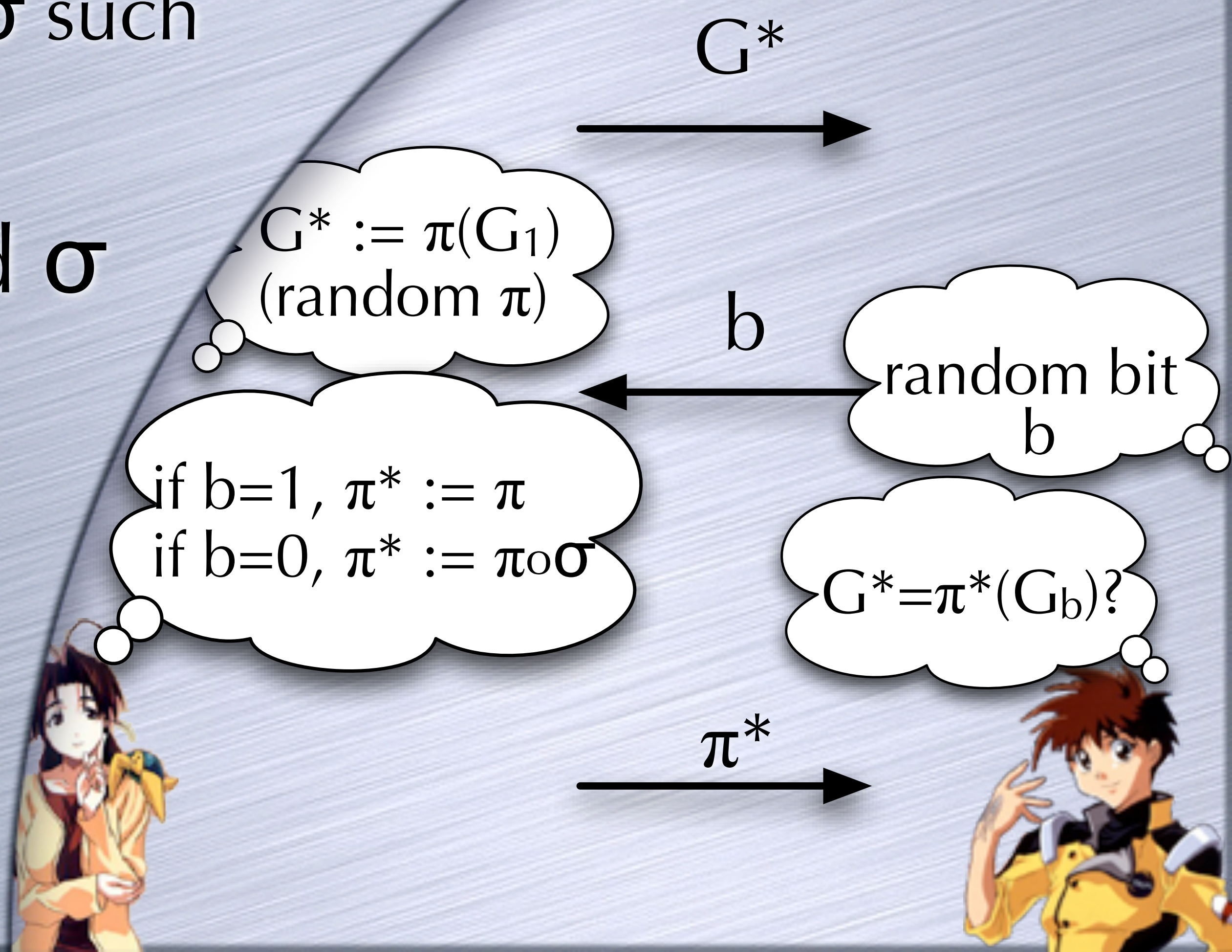
- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$



RECALL

An Example

- **Graph Isomorphism**
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol
  - Bob sees only b, π^* and G^* s.t. $\pi^*(G_b) = G^*$



The Legend of William Tell

A Side Story

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

***Charlie**: How do you know?*

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

***Charlie**: How do you know?*

***Bob**: I just saw him shoot an apple placed on his son's head! See this!*

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

***Charlie**: How do you know?*

***Bob**: I just saw him shoot an apple placed on his son's head! See this!*

***Charlie**: That apple convinced you? Anyone could have made it up!*

***Bob**: But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

Bob: Alice just proved it to me! See this:

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

***Charlie**: How do you know?*

***Bob**: I just saw him shoot an apple placed on his son's head! See this!*

***Charlie**: That apple convinced you? Anyone could have made it up!*

***Bob**: But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

Bob: Alice just proved it to me! See this:

$$G^*, b, \pi^* \text{ s.t. } G^* = \pi^*(G_b)$$

The Legend of William Tell

A Side Story

Bob*: William Tell is a great marksman!*

Charlie*: How do you know?*

Bob*: I just saw him shoot an apple placed on his son's head! See this!*

Charlie*: That apple convinced you? Anyone could have made it up!*

Bob*: But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

Bob: Alice just proved it to me! See this:

$$G^*,\ b,\ \pi^* \text{ s.t. } G^*=\pi^*(G_b)$$

Charlie: That convinced you? Anyone could have made it up!

The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

Bob: Alice just proved it to me! See this:

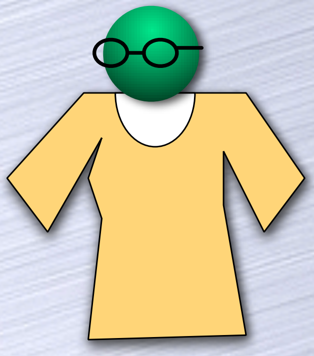
$$G^*,\ b,\ \pi^* \text{ s.t. } G^* = \pi^*(G_b)$$

Charlie: That convinced you? Anyone could have made it up!

Bob: But I picked b at random and she had no trouble answering me...

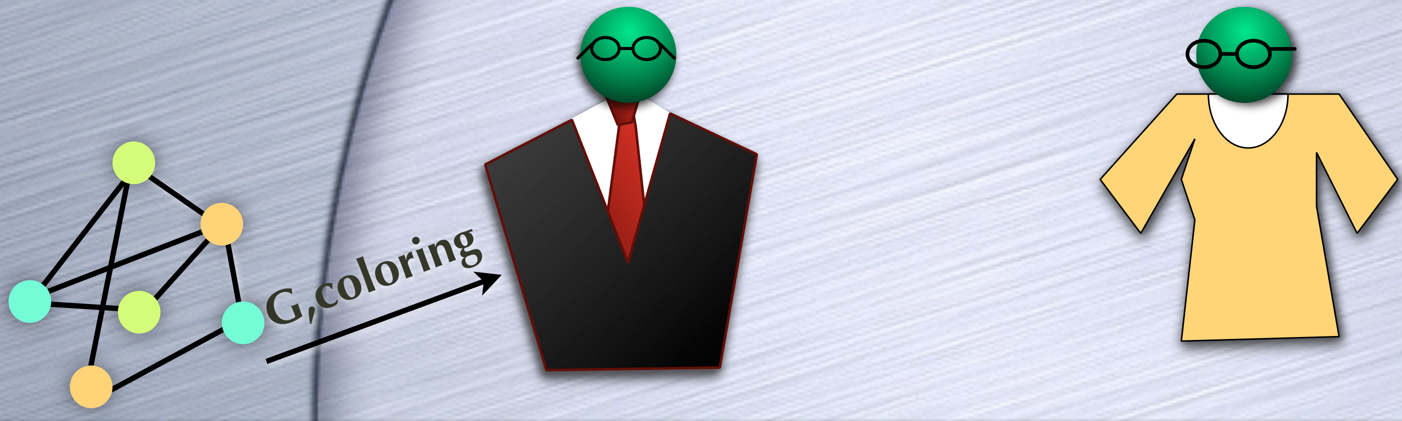
RECALL

A ZK Proof for Graph Colorability



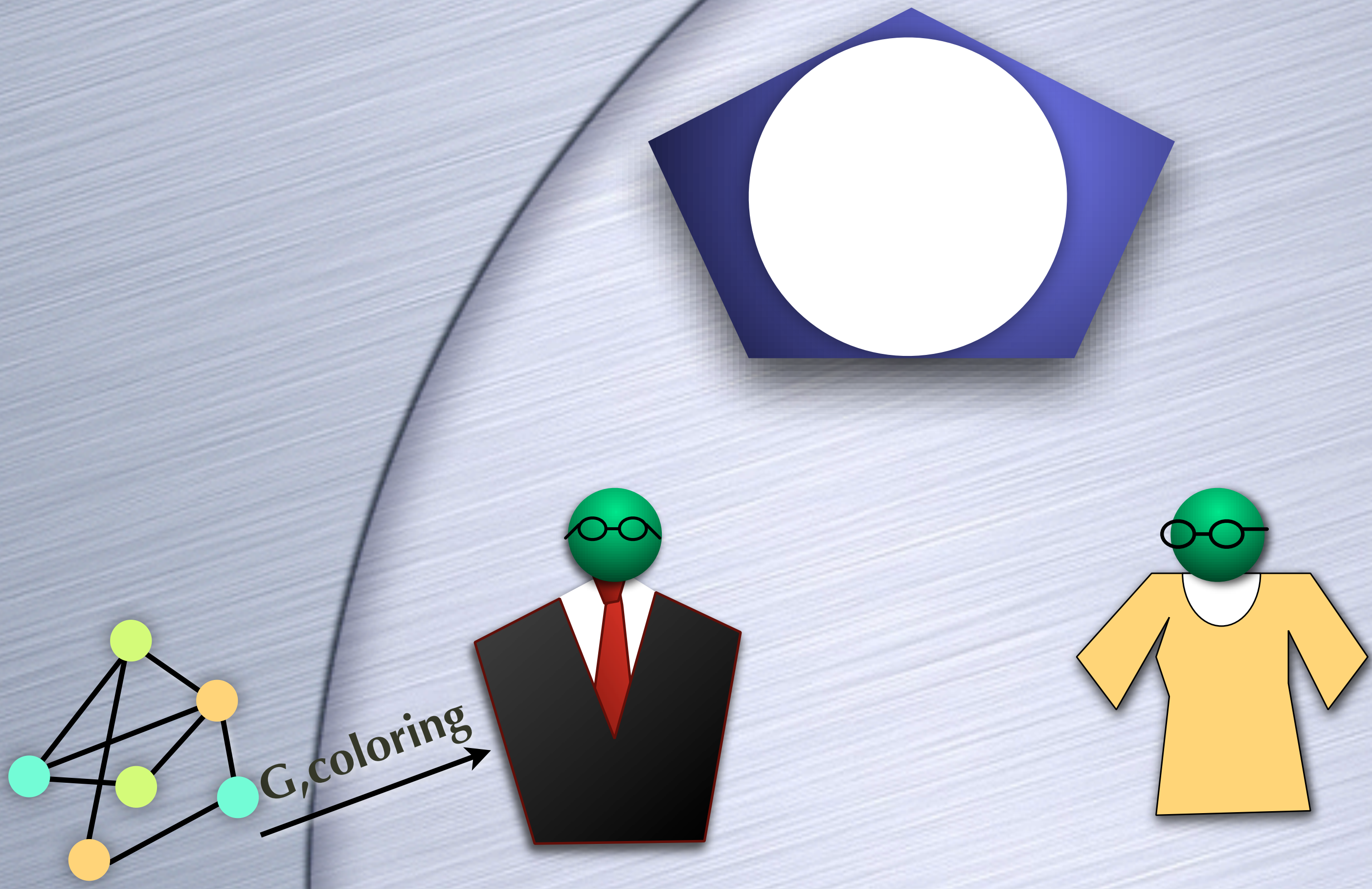
RECALL

A ZK Proof for Graph Colorability



A ZK Proof for Graph Colorability

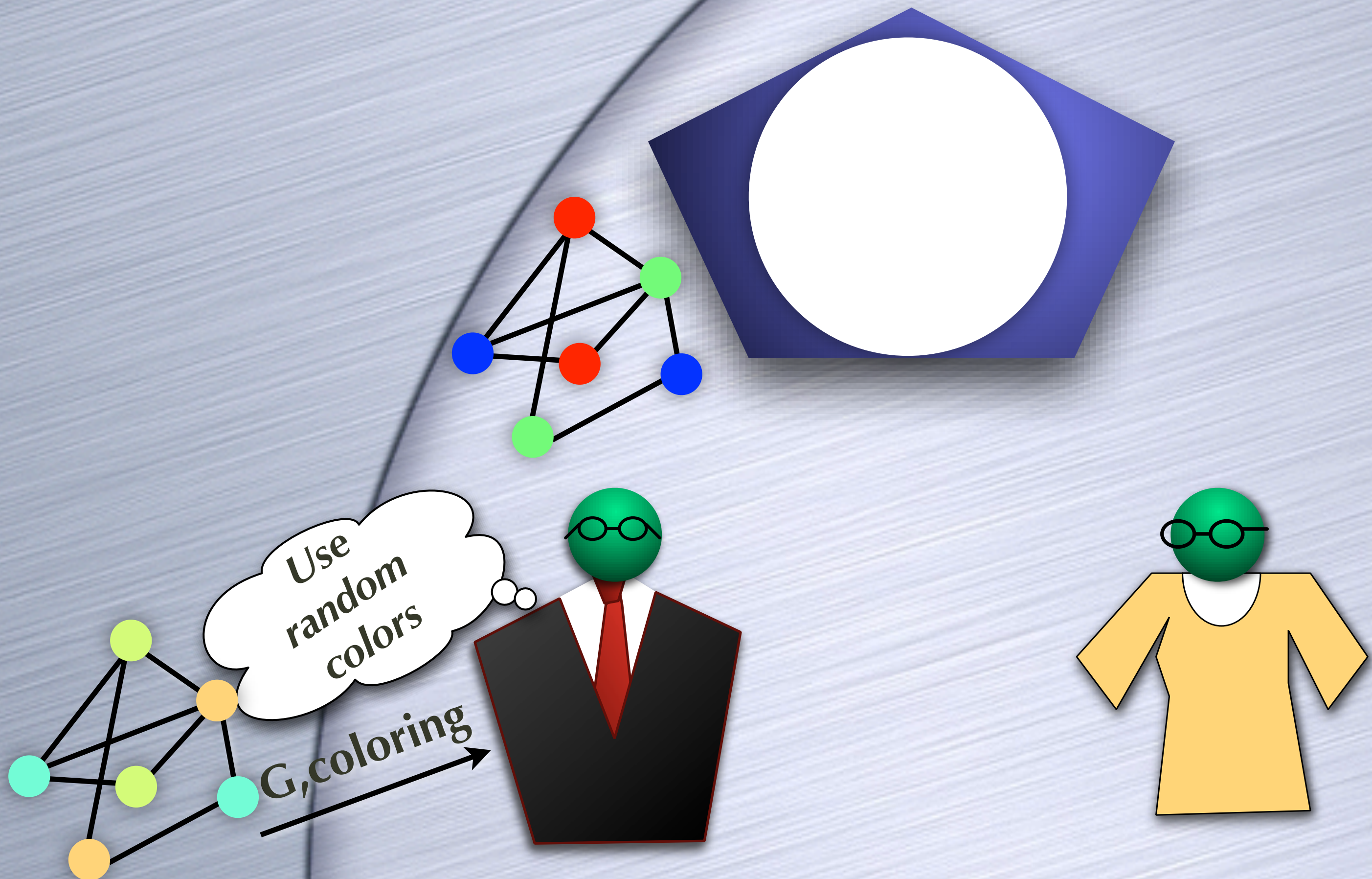
- Uses a commitment protocol as a subroutine



A ZK Proof for Graph Colorability

RECALL

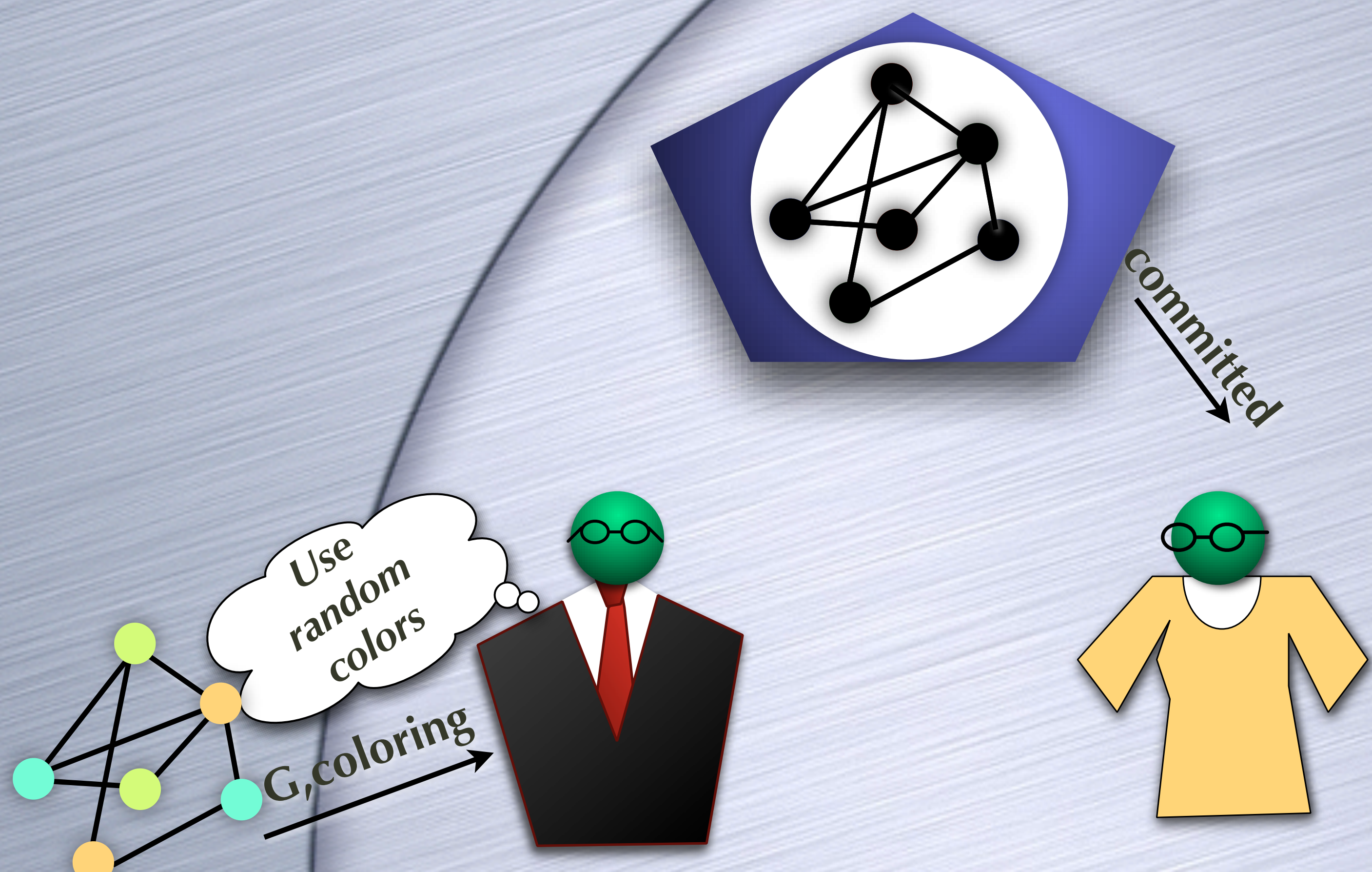
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

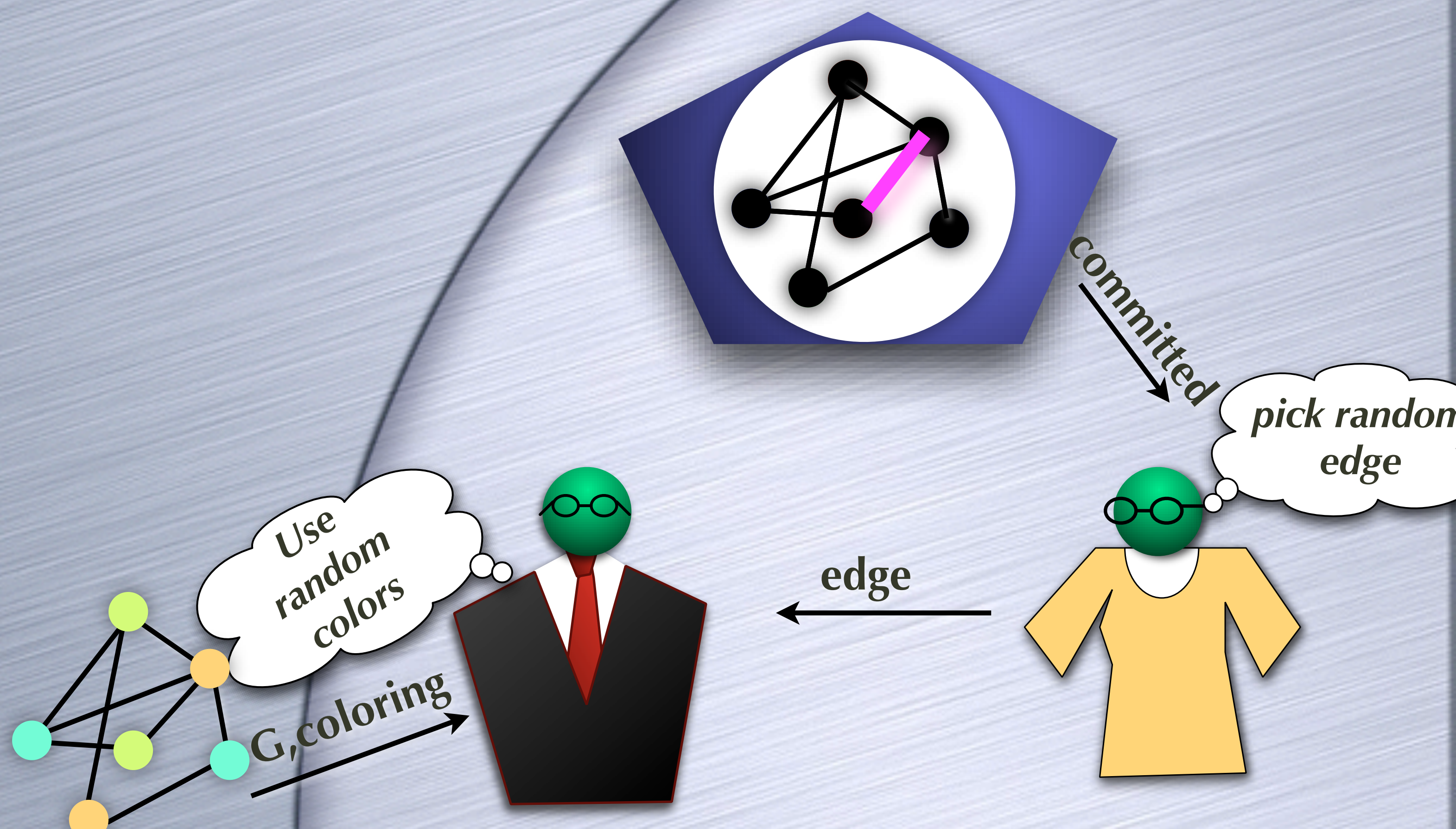
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

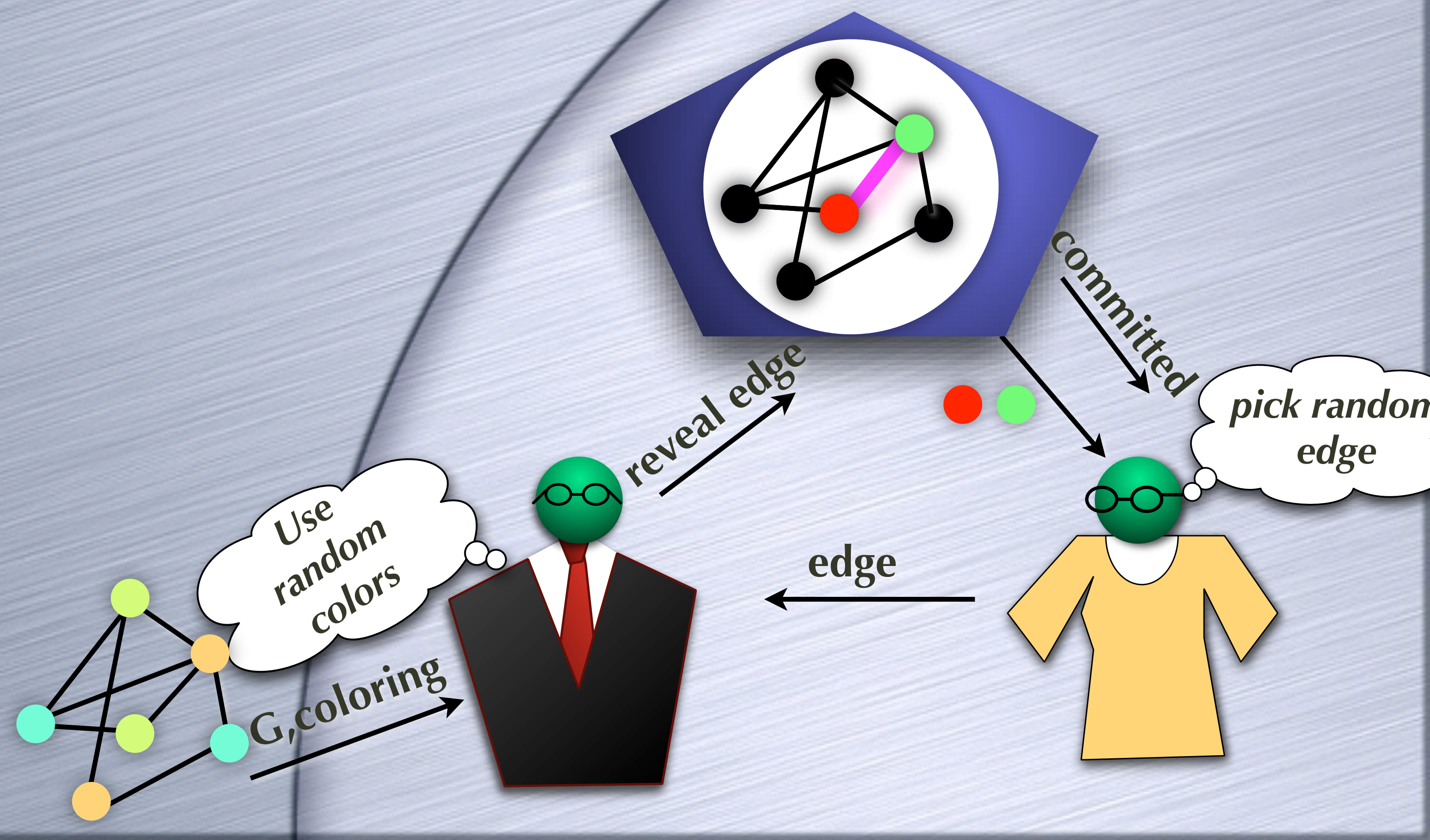
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

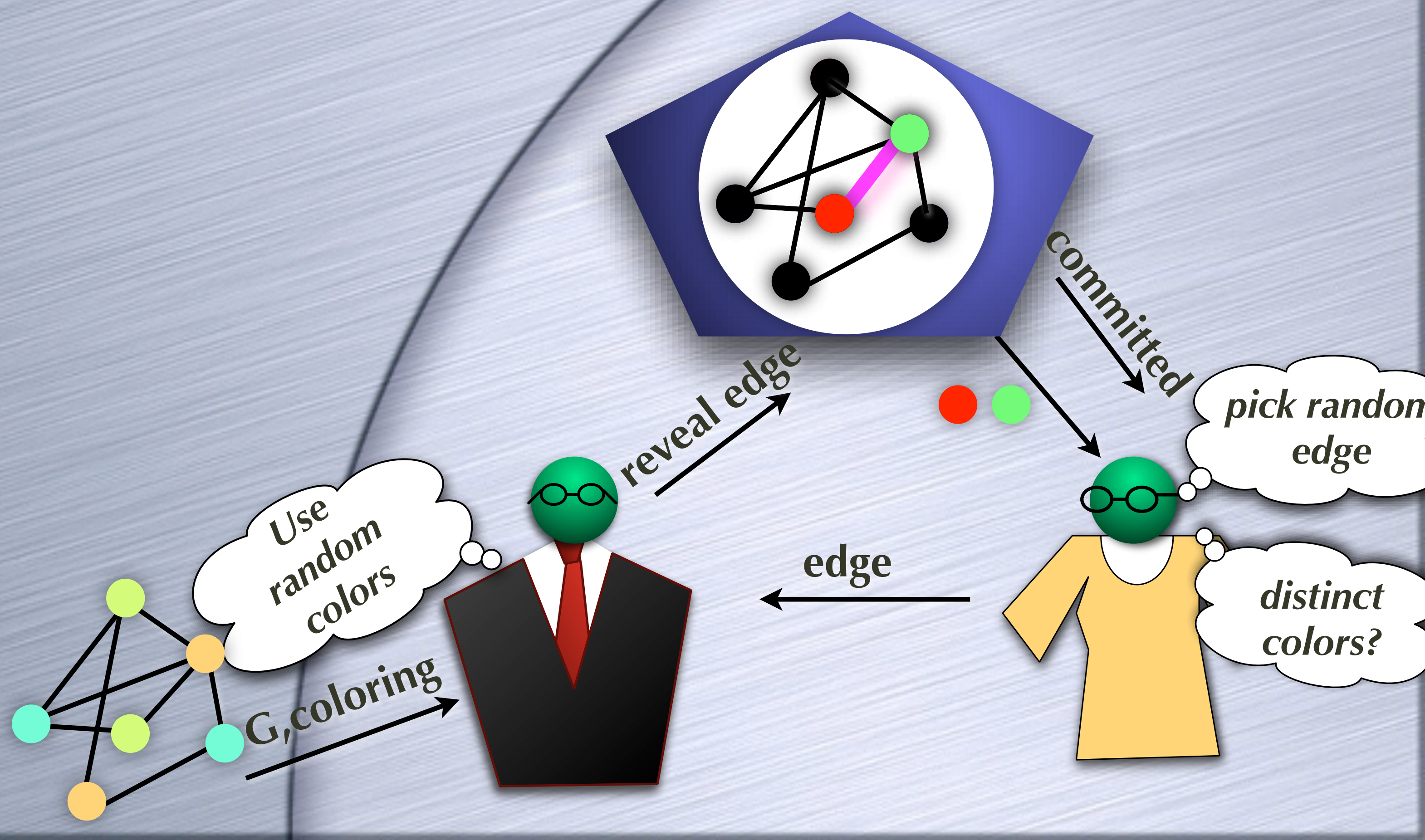
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

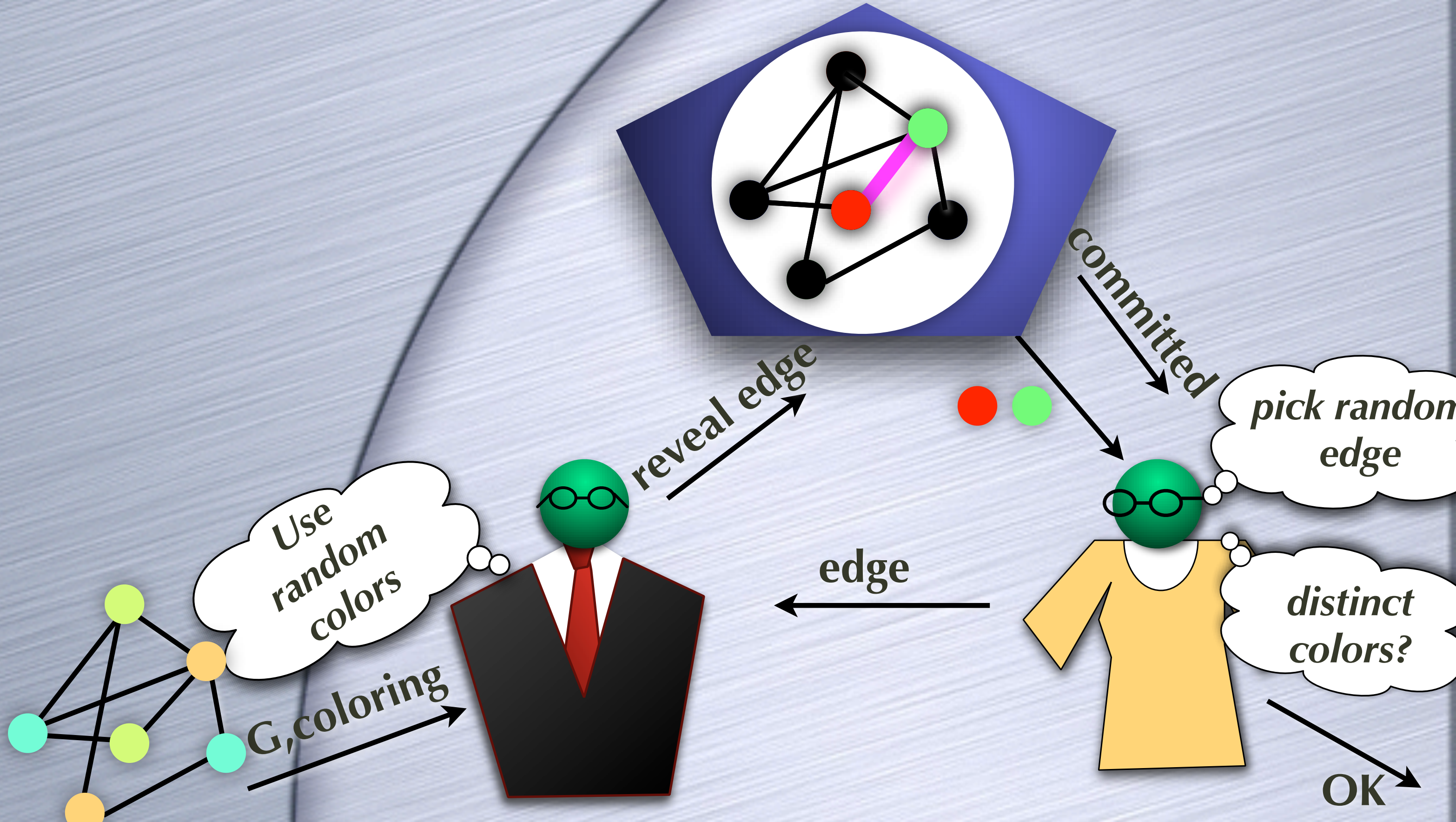
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

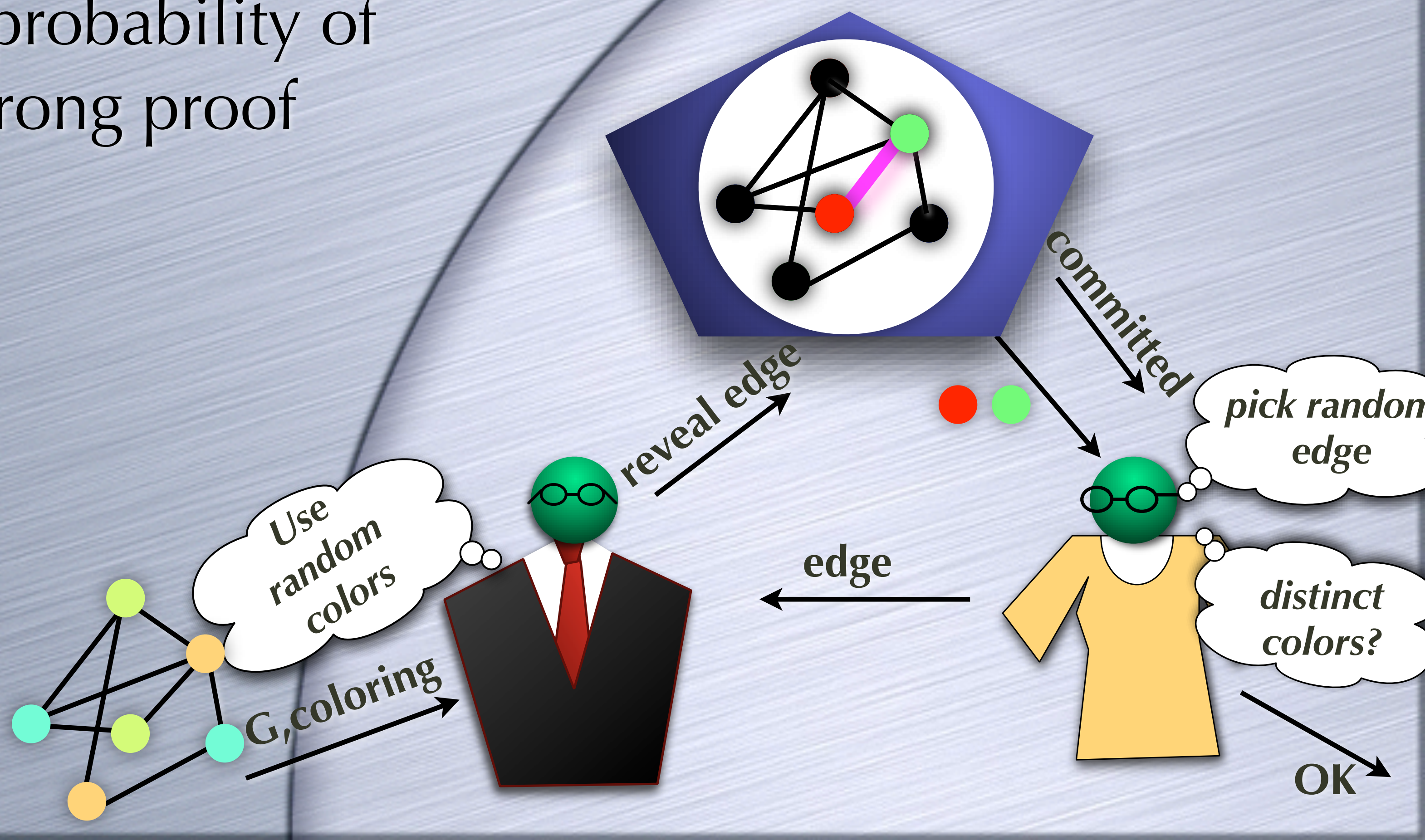
- Uses a commitment protocol as a subroutine



RECALL

A ZK Proof for Graph Colorability

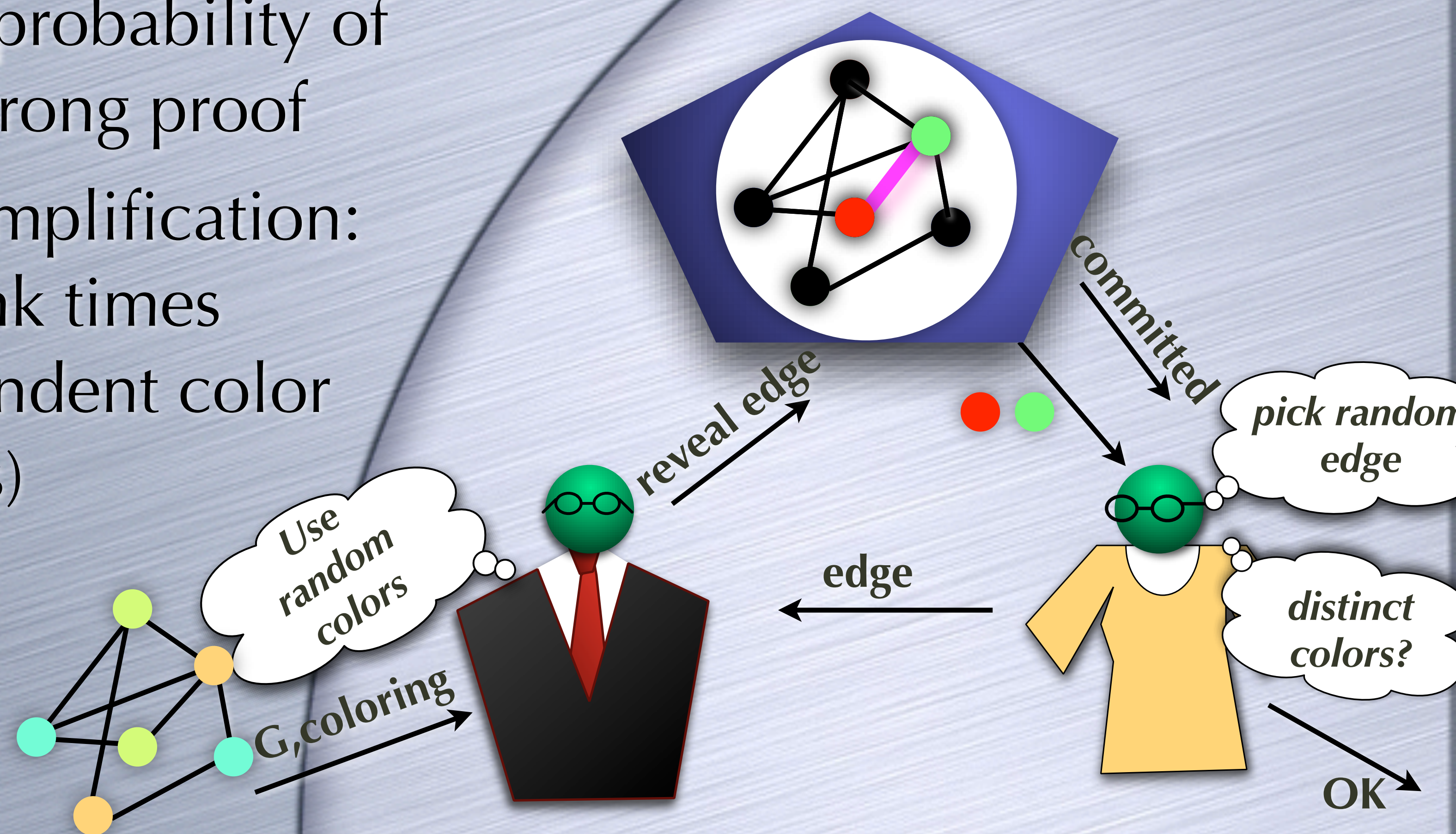
- Uses a commitment protocol as a subroutine
- At least 1/m probability of catching a wrong proof



A ZK Proof for Graph Colorability

RECALL

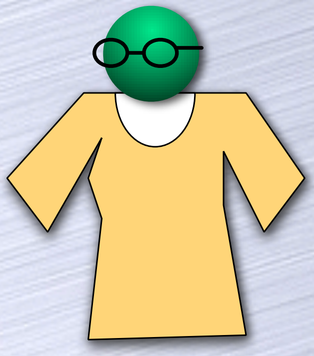
- Uses a commitment protocol as a subroutine
- At least 1/m probability of catching a wrong proof
- Soundness amplification: Repeat say mk times (with independent color permutations)



A Commitment Protocol

A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B

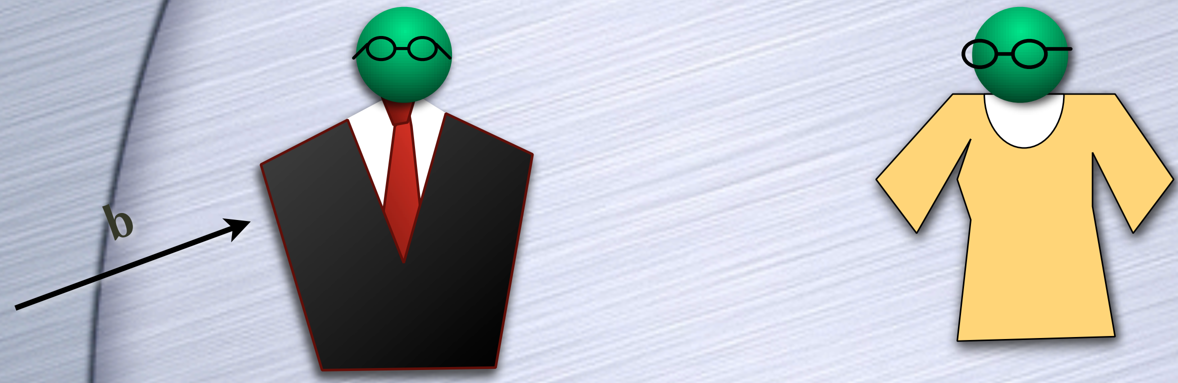


A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding

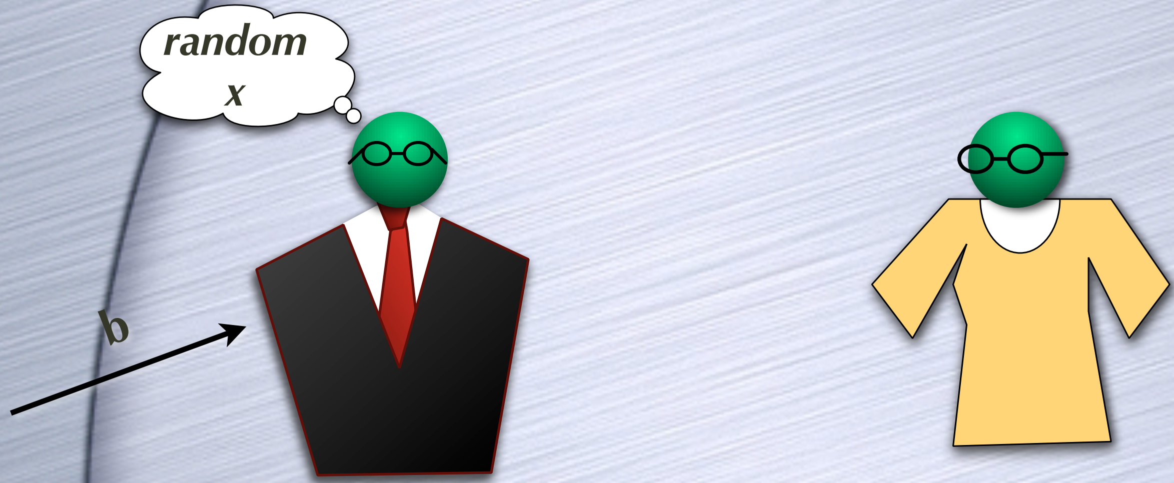
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



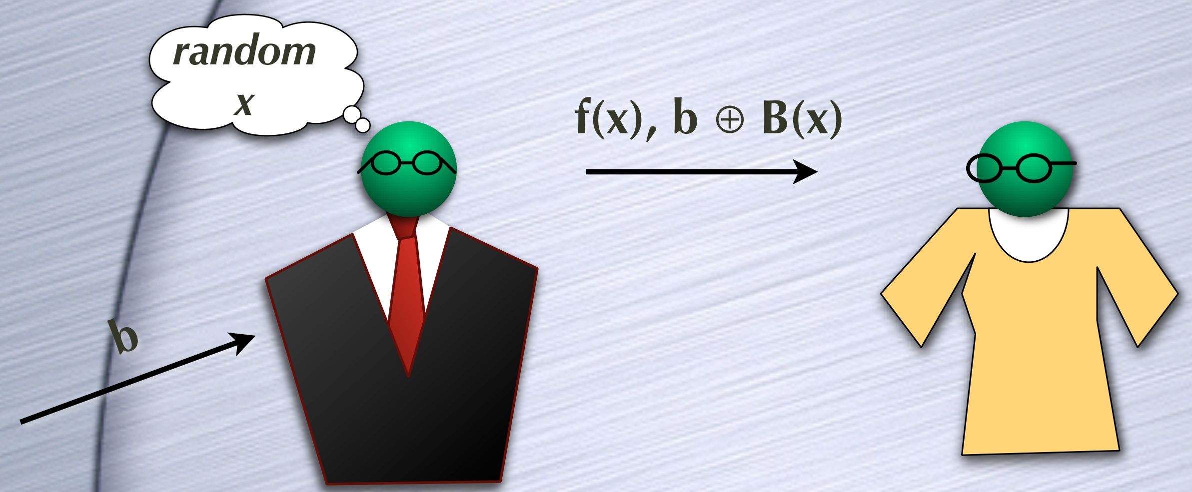
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



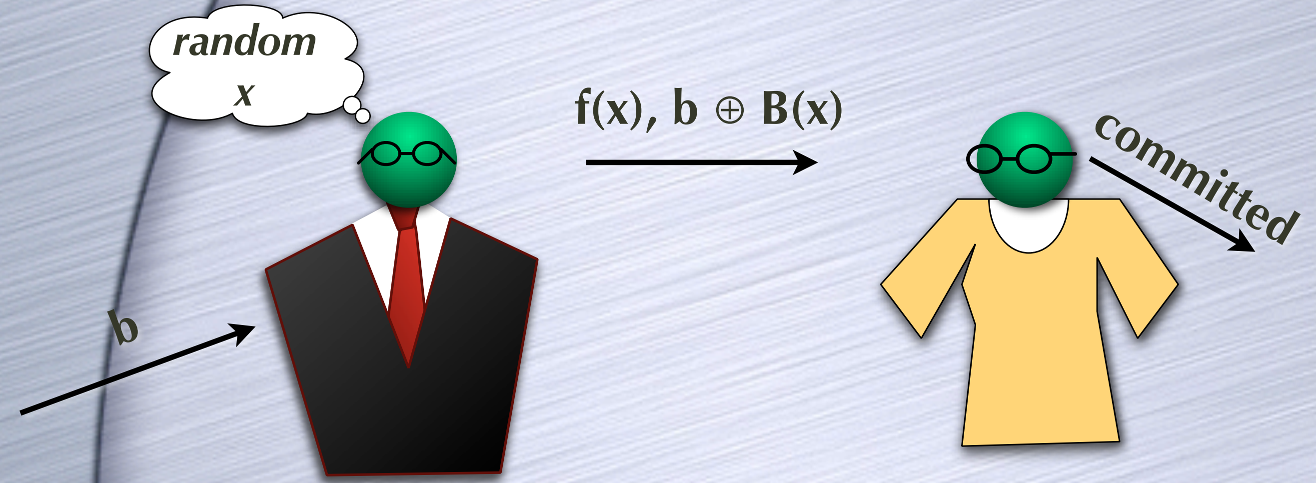
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



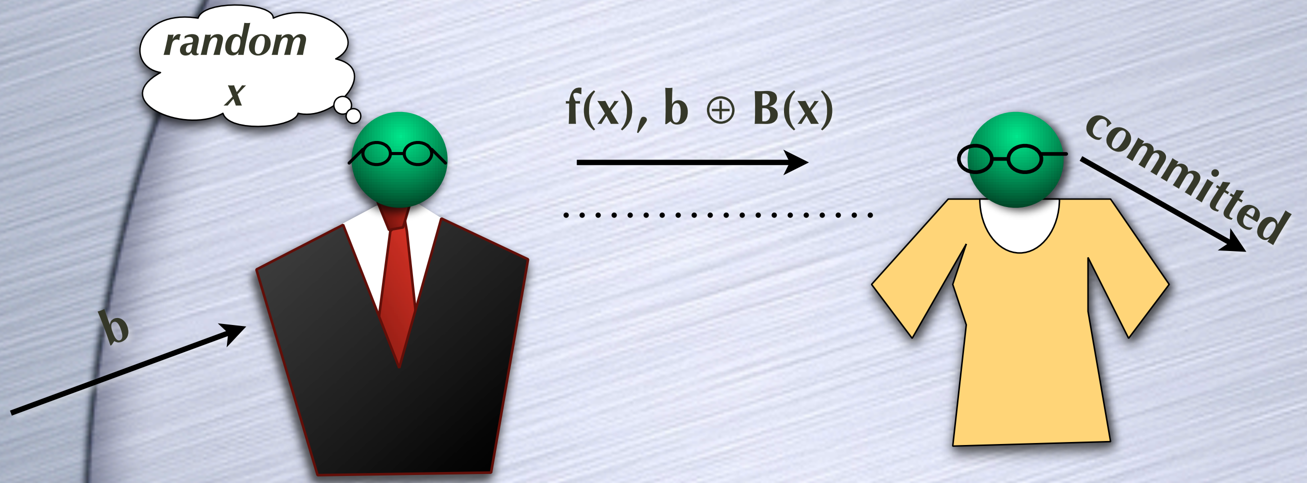
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



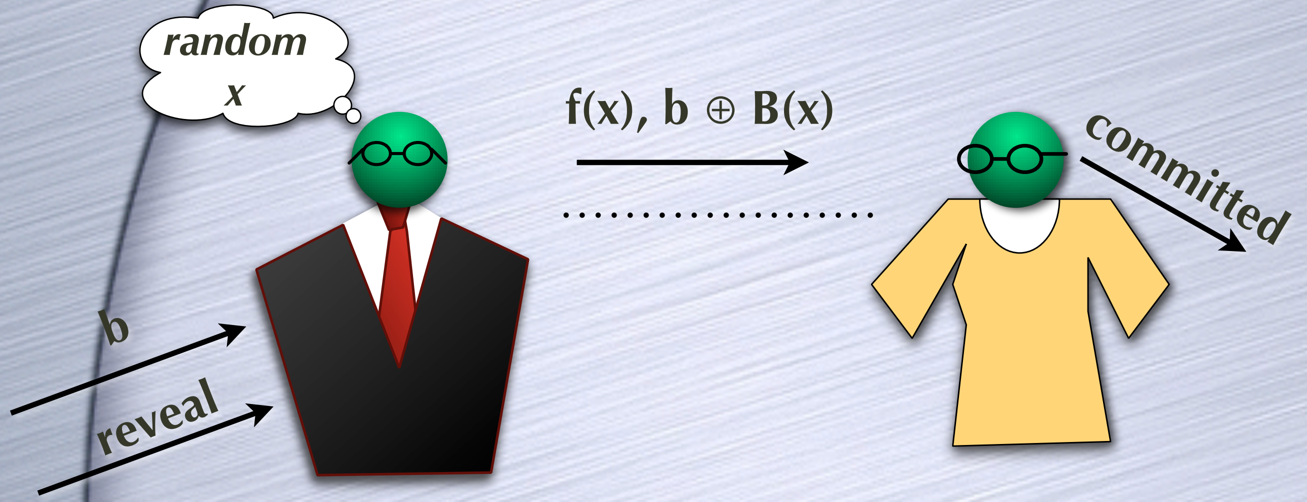
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



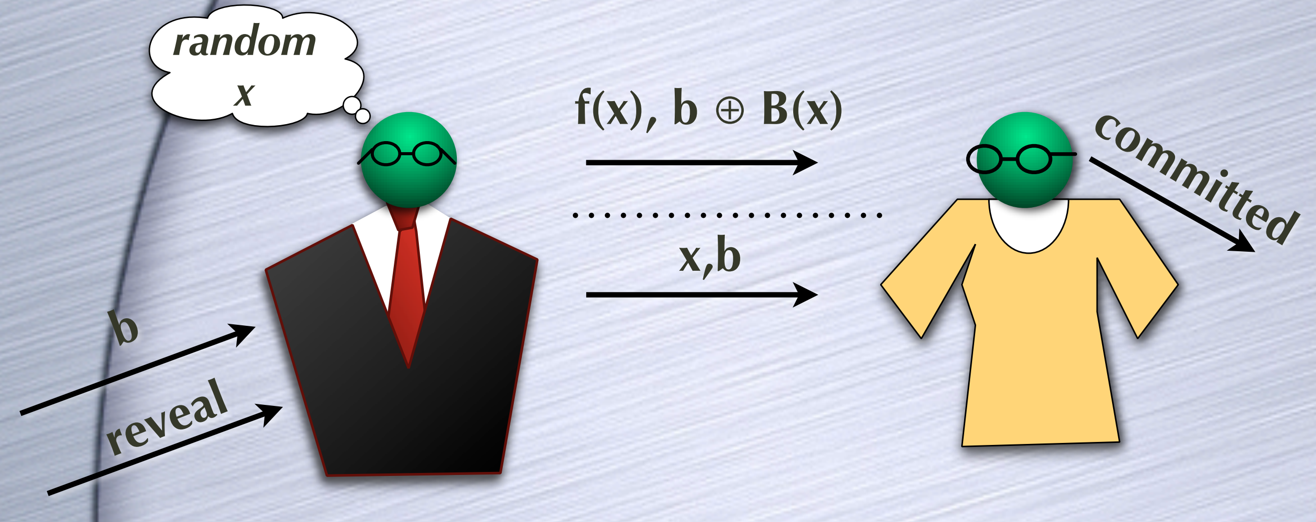
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



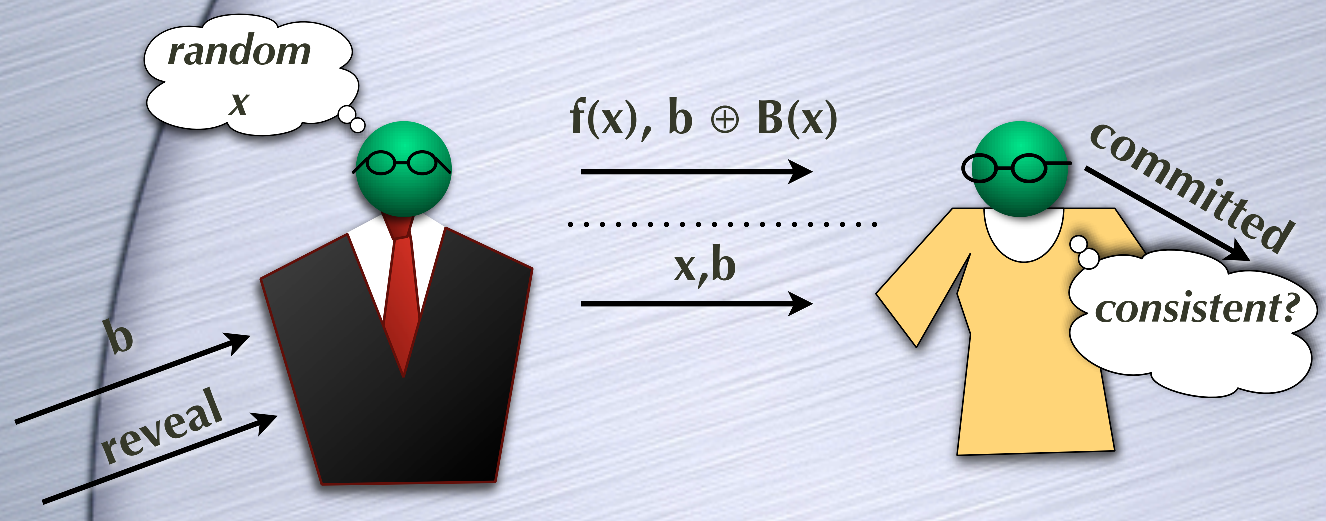
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



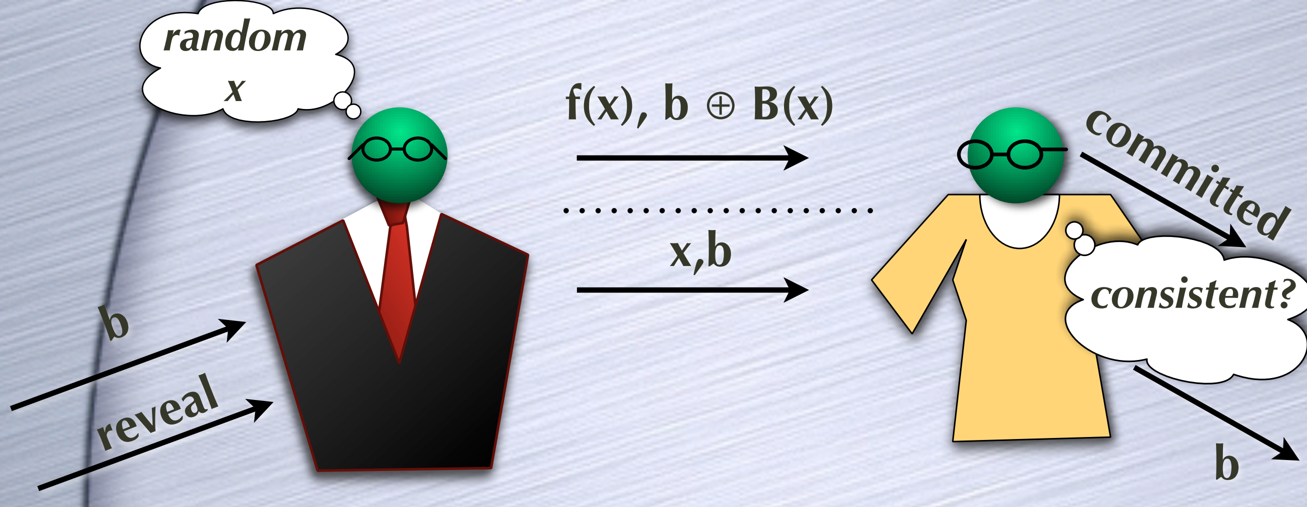
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



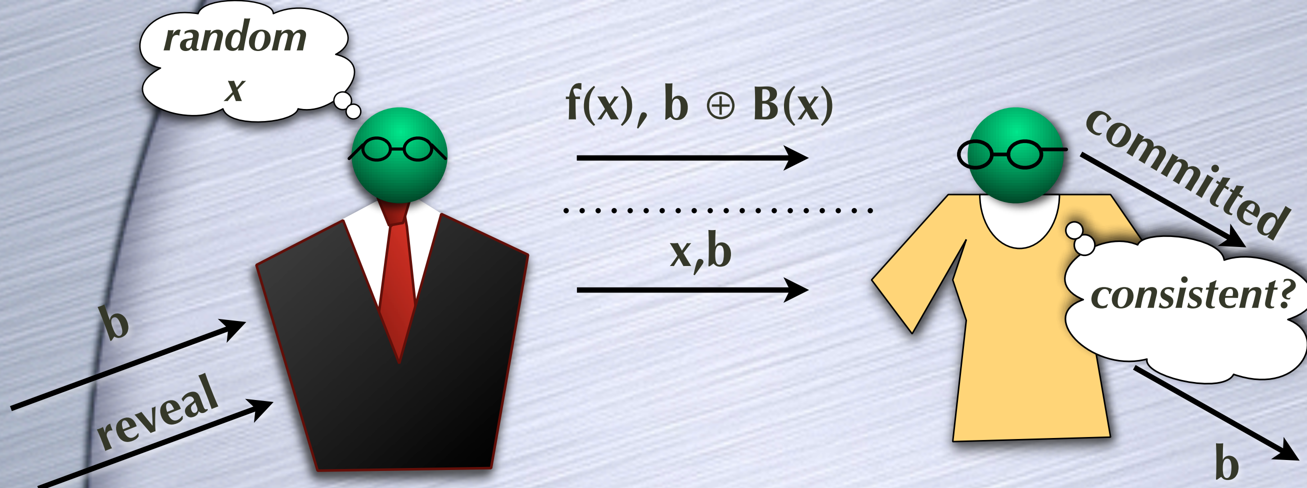
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding



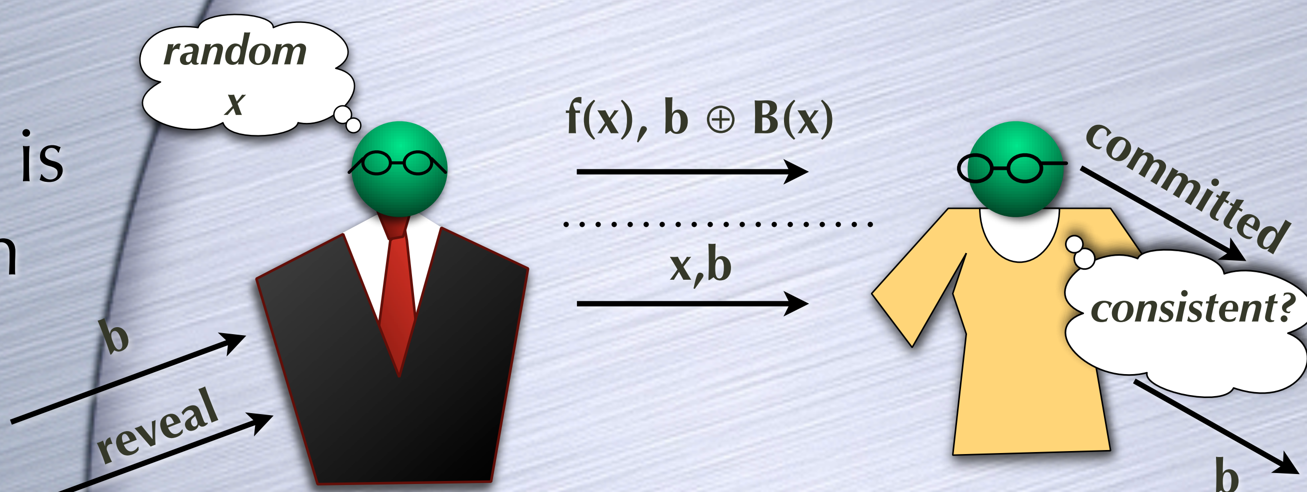
A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding
- Perfectly binding because f is a permutation



A Commitment Protocol

- Using a OWP f and a hardcore predicate for it B
- Satisfies only classical (IND) security, in terms of hiding and binding
- Perfectly binding because f is a permutation
- Hiding because B(x) is pseudorandom given f(x)



ZK Proofs: What for?

ZK Proofs: What for?

- Authentication

ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge

ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols

ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols

ZK Proofs: What for?

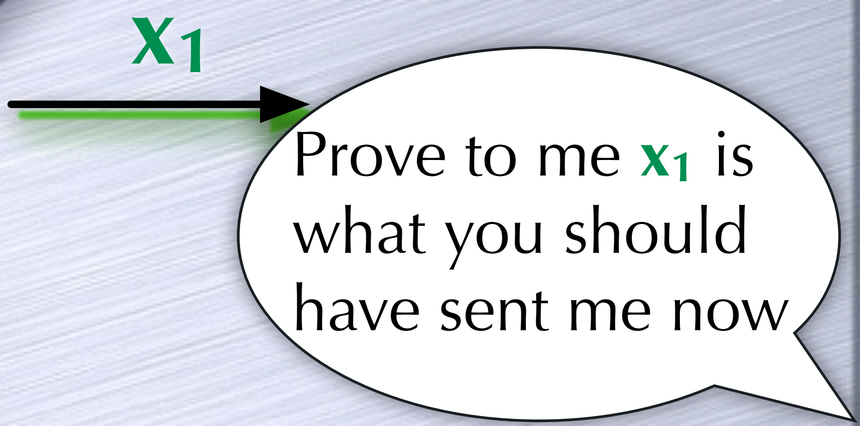
- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed

ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed

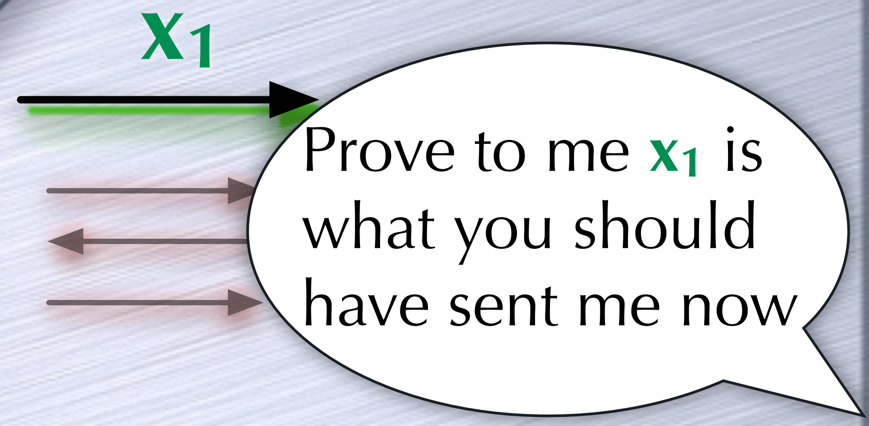
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed



ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed

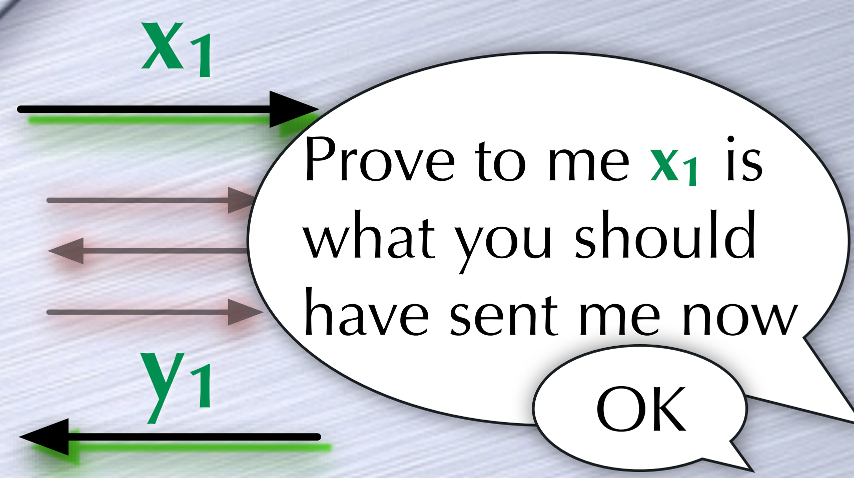


ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed

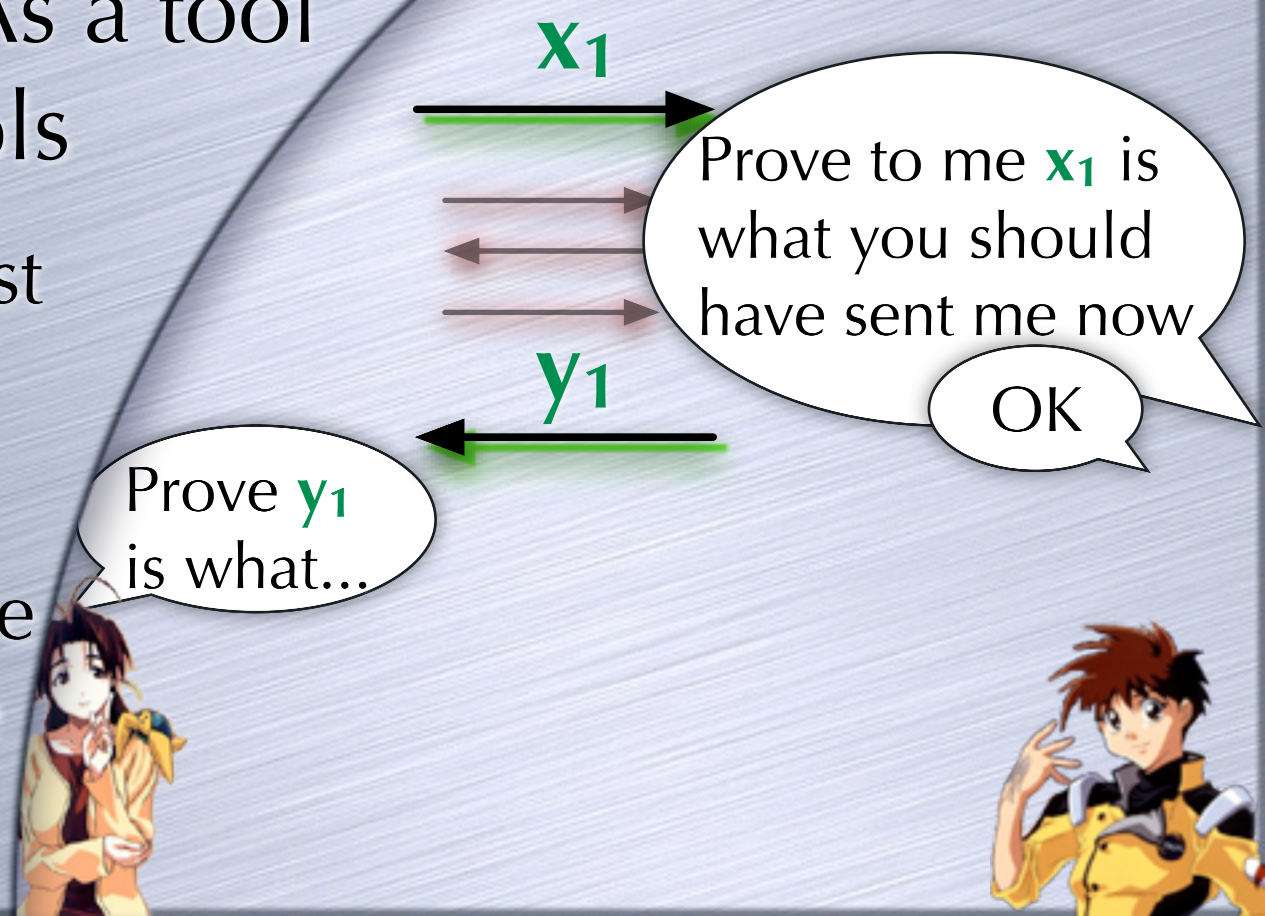
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed



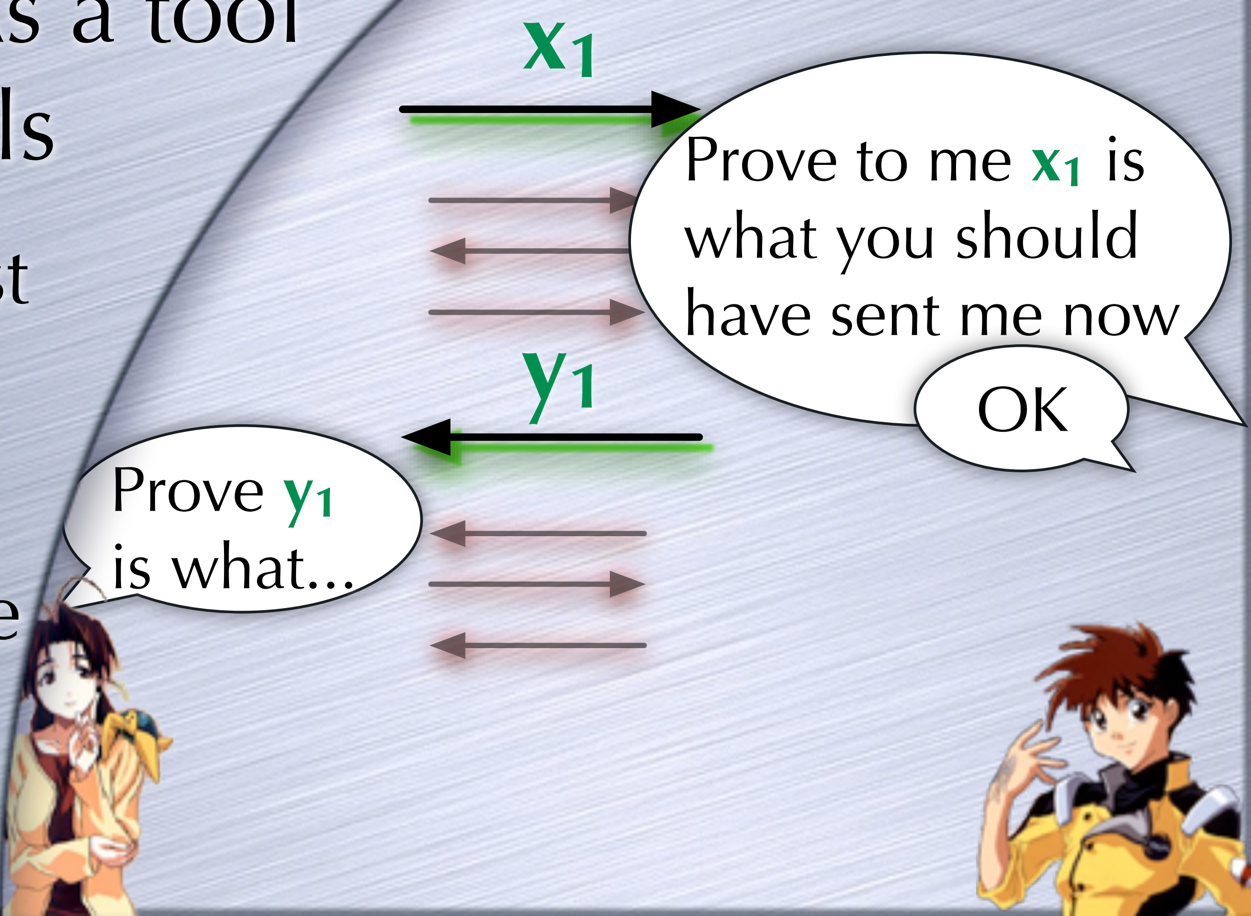
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed



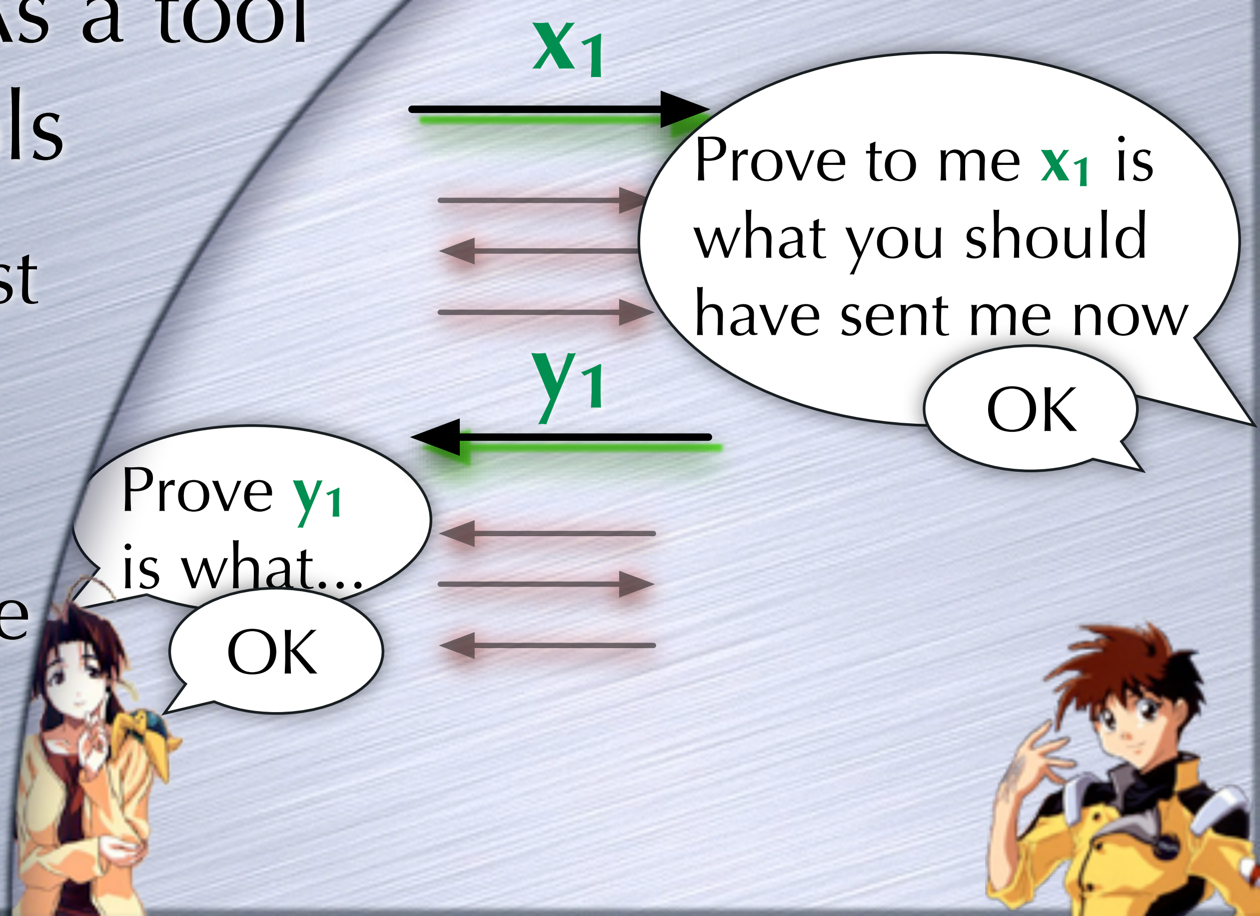
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed



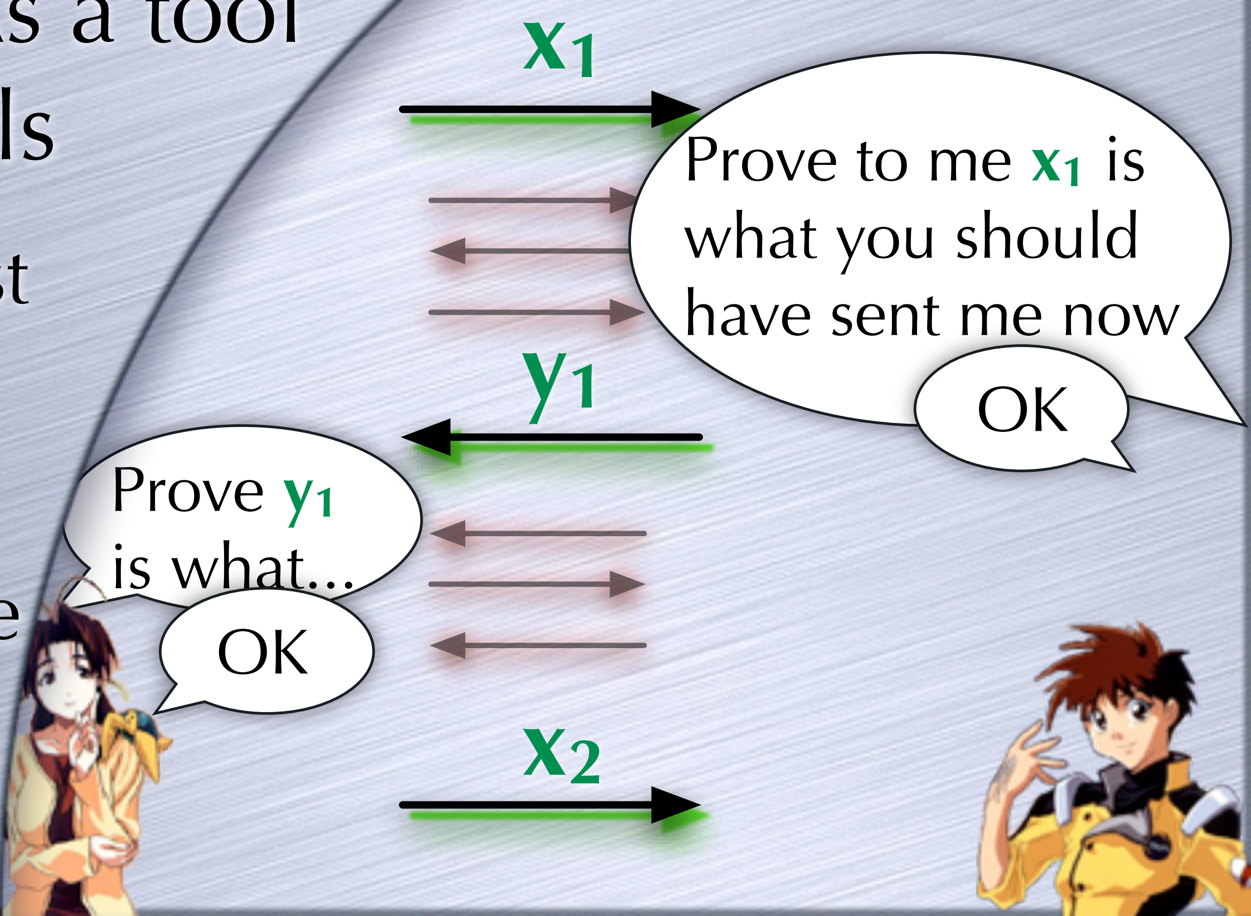
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed



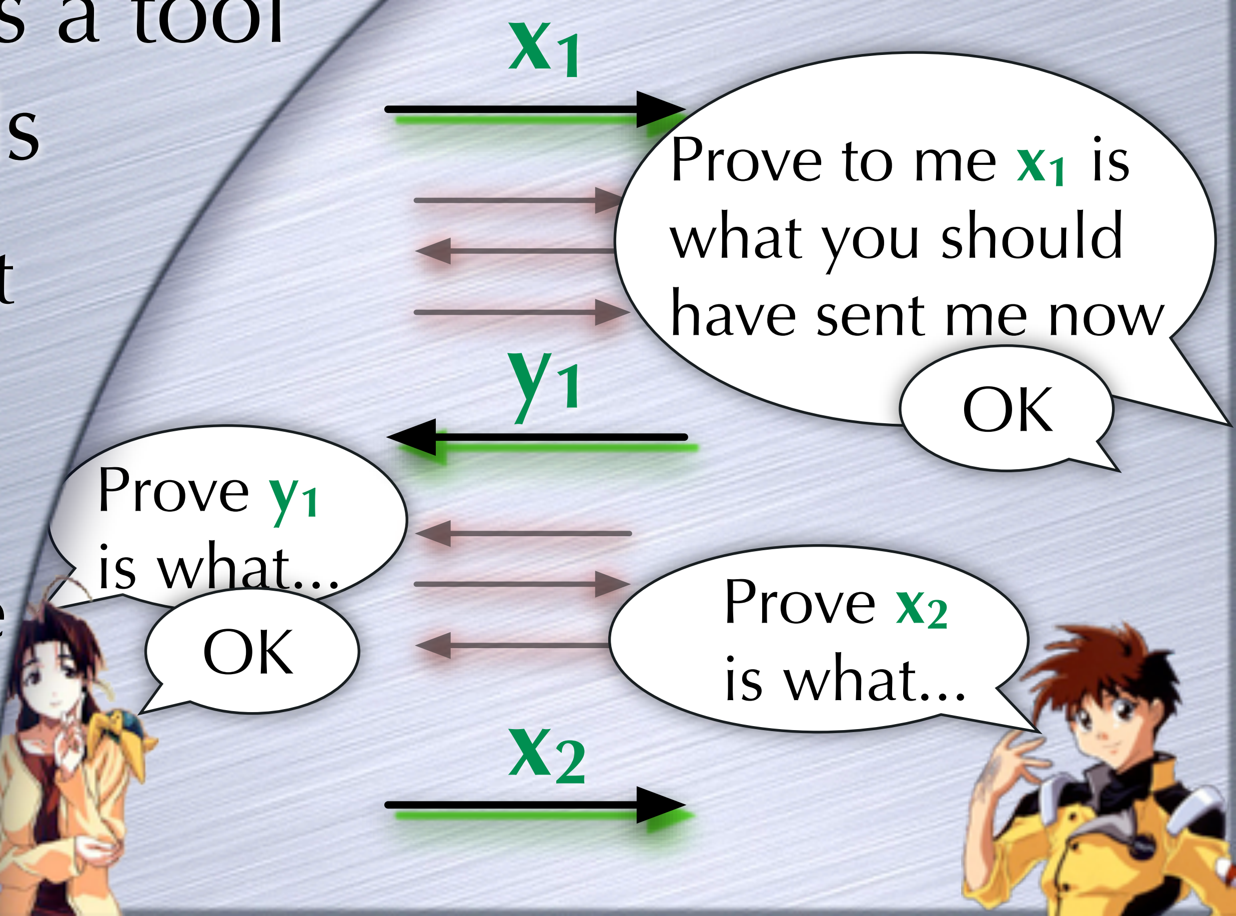
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed

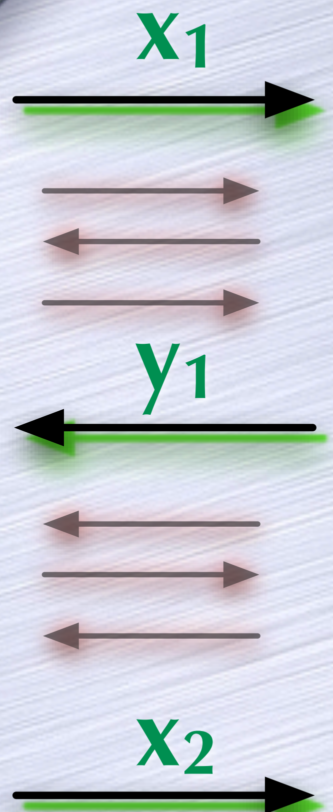


ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed

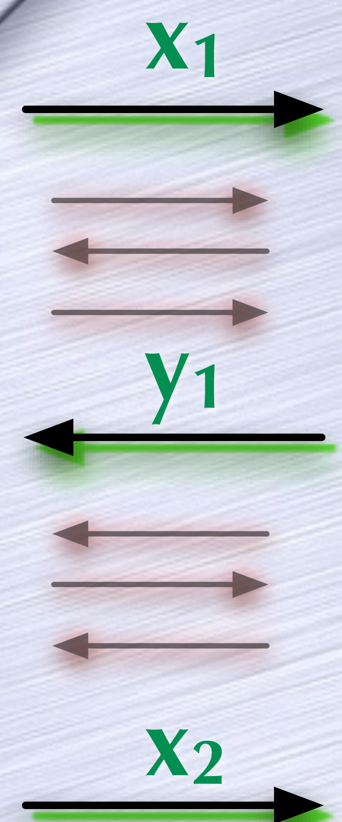


Does it fit in?



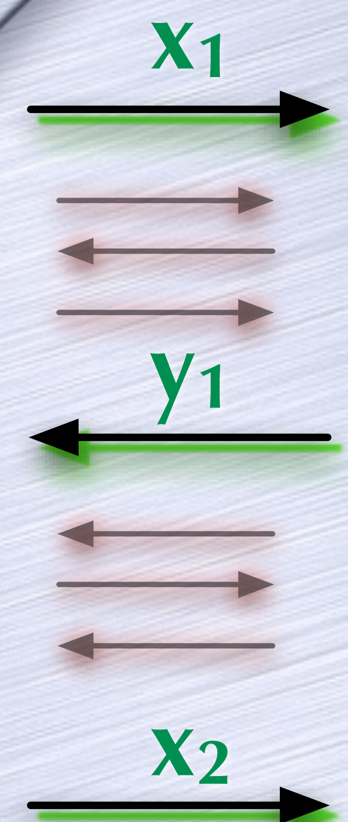
Does it fit in?

- Does the proof stay ZK in the big picture?



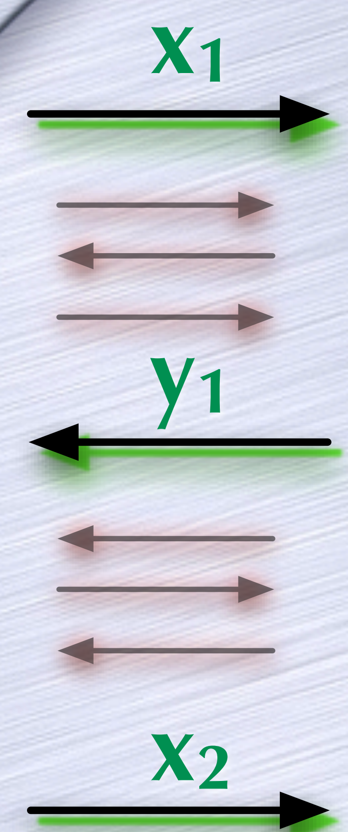
Does it fit in?

- Does the proof stay ZK in the big picture?
  - Composition



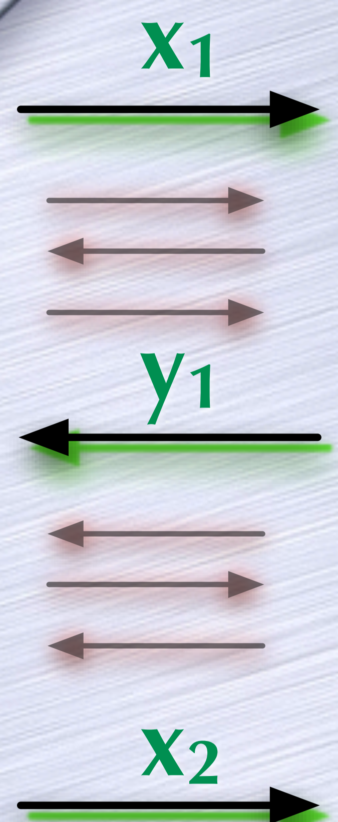
Does it fit in?

- Does the proof stay ZK in the big picture?
  - Composition
    - Several issues: auxiliary information from previous runs, concurrency issues, malleability/man-in-the-middle



Does it fit in?

- Does the proof stay ZK in the big picture?
  - Composition
    - Several issues: auxiliary information from previous runs, concurrency issues, malleability/man-in-the-middle
  - In general, to allow composition more complicated protocols



Non-Interactive ZK

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
    - No soundness: prover can give the simulated proof!

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
  - No soundness: prover can give the simulated proof!
- NIZK: a trusted "common random string" (CRS) is published, and the proof/verification is w.r.t CRS

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
  - No soundness: prover can give the simulated proof!
- NIZK: a trusted "common random string" (CRS) is published, and the proof/verification is w.r.t CRS
  - NIZK property: a simulator can simulate the CRS and the proofs

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
  - No soundness: prover can give the simulated proof!
- NIZK: a trusted "common random string" (CRS) is published, and the proof/verification is w.r.t CRS
  - NIZK property: a simulator can simulate the CRS and the proofs
  - Note: CRS is a part of the proof, but prover is not allowed to choose it (otherwise no soundness)

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
  - No soundness: prover can give the simulated proof!
- NIZK: a trusted "common random string" (CRS) is published, and the proof/verification is w.r.t CRS
  - NIZK property: a simulator can simulate the CRS and the proofs
  - Note: CRS is a part of the proof, but prover is not allowed to choose it (otherwise no soundness)
- NIZK schemes exist for all NP languages (using "enhanced" T-OWP)

Non-Interactive ZK

- Can the prover just give a written proof (no interaction) which any one can verify and can simulate too?
  - No soundness: prover can give the simulated proof!
- NIZK: a trusted "common random string" (CRS) is published, and the proof/verification is w.r.t CRS
  - NIZK property: a simulator can simulate the CRS and the proofs
  - Note: CRS is a part of the proof, but prover is not allowed to choose it (otherwise no soundness)
- NIZK schemes exist for all NP languages (using "enhanced" T-OWP)
  - Also can NIZK-ify some ZK protocols in the RO Model (no CRS)

An IND-security Notion

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.
    - A ZK proof is always WI, but not vice-versa

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.
    - A ZK proof is always WI, but not vice-versa
- WI Proofs used as components inside larger protocols

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.
    - A ZK proof is always WI, but not vice-versa
- WI Proofs used as components inside larger protocols
  - Sometimes with certain other useful properties

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.
    - A ZK proof is always WI, but not vice-versa
- WI Proofs used as components inside larger protocols
  - Sometimes with certain other useful properties
    - e.g. WI-PoK, "Sigma protocols"

An IND-security Notion

- ZK (as opposed to SIM-ZK/ZK-PoK) weakens soundness guarantee
- A weakening of ZK property: Witness Indistinguishability (WI)
  - Adversarial verifier gives (x,w_0,w_1) and prover uses (x,w_b) for a random b. Adversary has negligible advantage in guessing b.
    - A ZK proof is always WI, but not vice-versa
- WI Proofs used as components inside larger protocols
  - Sometimes with certain other useful properties
    - e.g. WI-PoK, "Sigma protocols"
- Defined in standalone setting, but WI property is preserved under "parallel composition"

Composition

Composition Issues

Composition Issues

- Multiple executions provide new opportunities for the hacker

Composition Issues

- Multiple executions provide new opportunities for the hacker

Play the GM's against each other

Will not lose against both!

Composition Issues

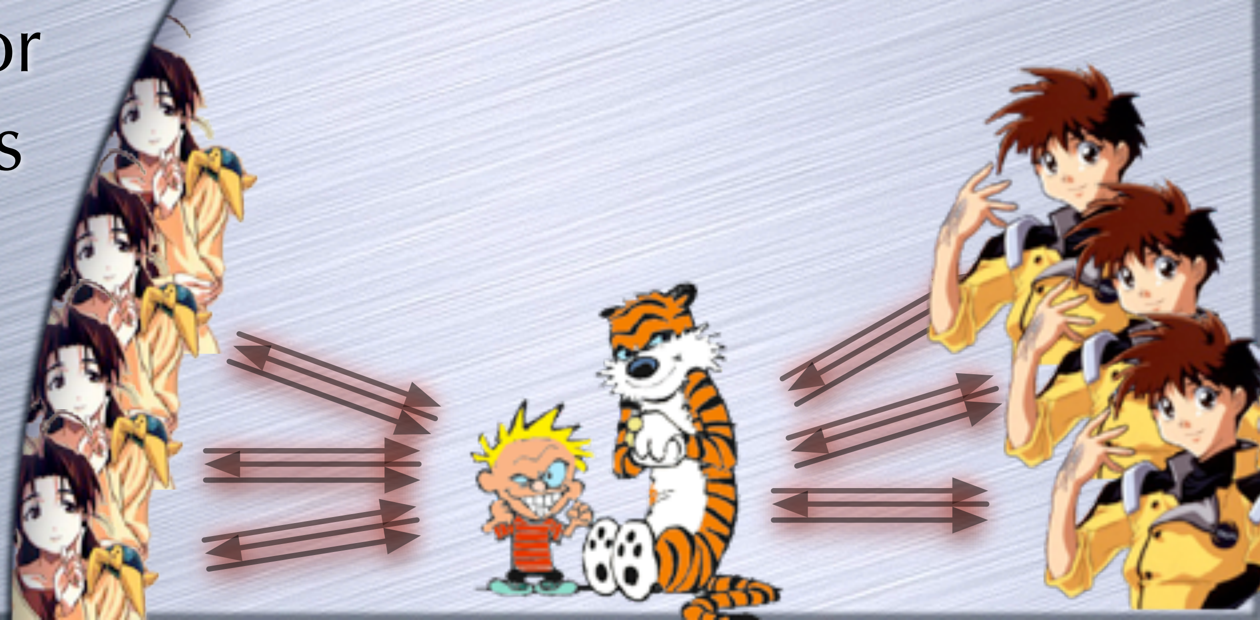
- Multiple executions provide new opportunities for the hacker
  - Person-in-the-middle attack

Play the GM's against each other

Will not lose against both!

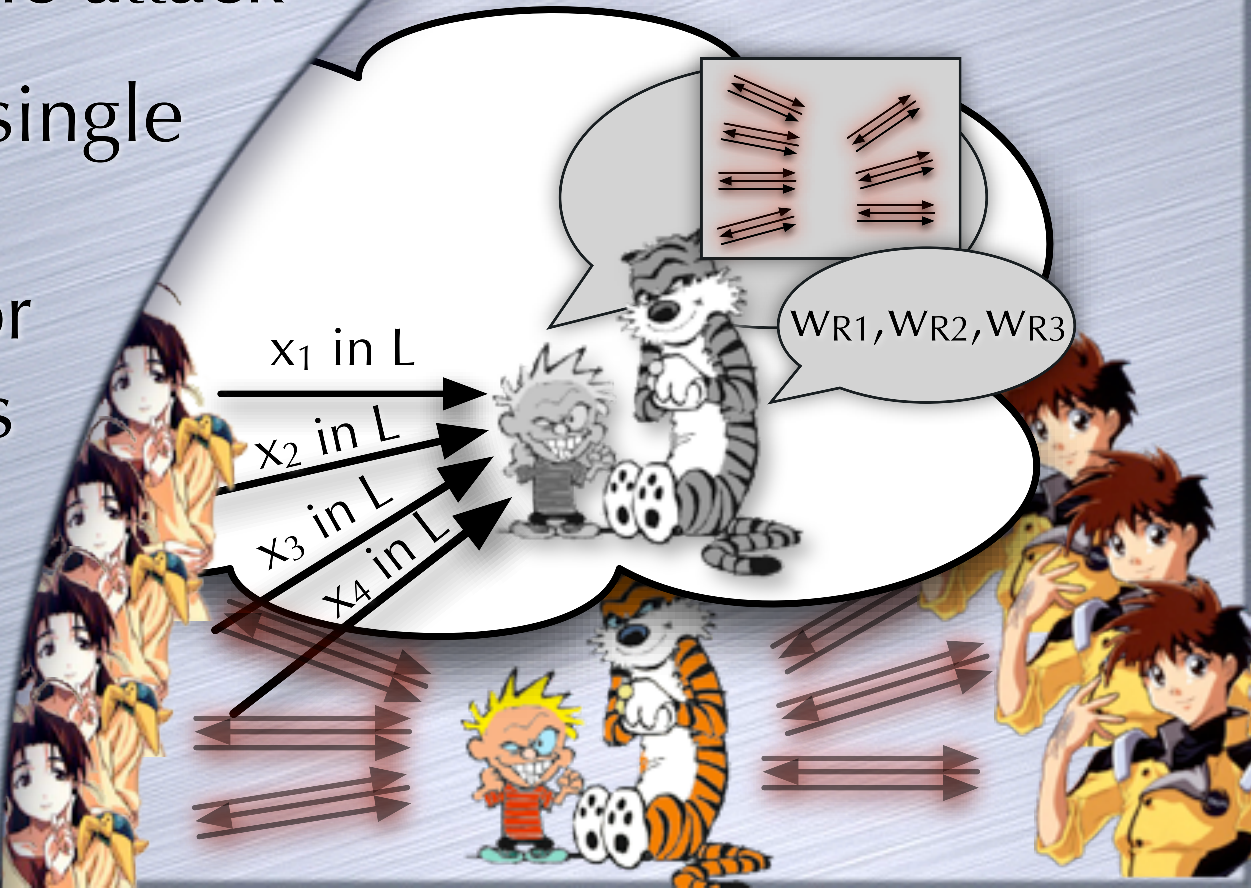
Composition Issues

- Multiple executions provide new opportunities for the hacker
  - Person-in-the-middle attack
  - Simulatability of a single execution doesn't imply simulation for multiple executions



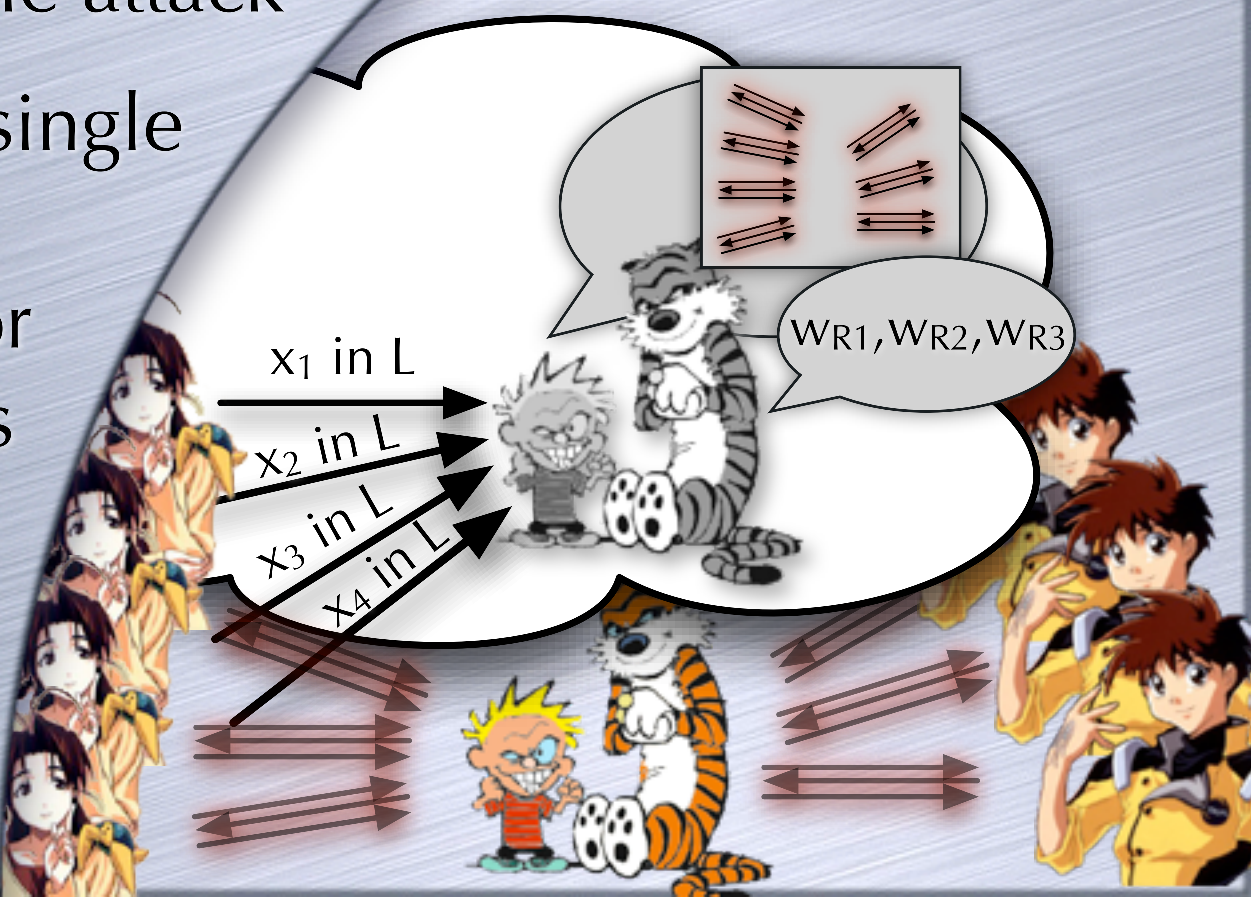
Composition Issues

- Multiple executions provide new opportunities for the hacker
  - Person-in-the-middle attack
  - Simulatability of a single execution doesn't imply simulation for multiple executions



Composition Issues

- Multiple executions provide new opportunities for the hacker
  - Person-in-the-middle attack
  - Simulatability of a single execution doesn't imply simulation for multiple executions
    - Or when run along with other protocols



Universal Composition

Universal Composition

- A security guarantee

Universal Composition

- A security guarantee
  - that can be given for a "composed system"

Universal Composition

- A security guarantee
  - that can be given for a “composed system”
  - such that security for each component separately implies security for the entire system

Universal Composition

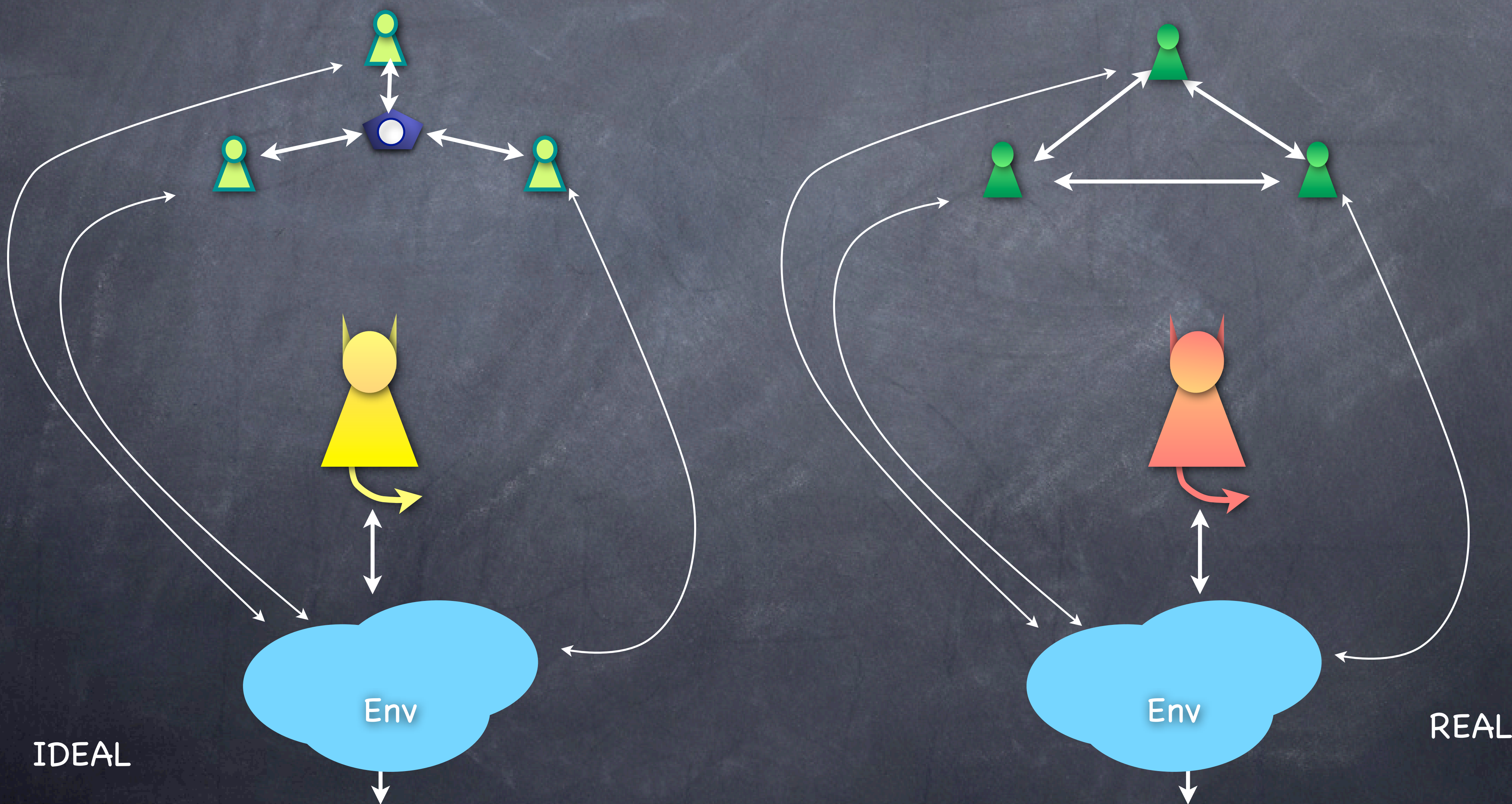
- A security guarantee
  - that can be given for a "composed system"
  - such that security for each component separately implies security for the entire system
  - and is meaningful! (otherwise, "everything is secure" is composable)

Universal Composition

- A security guarantee
  - that can be given for a "composed system"
  - such that security for each component separately implies security for the entire system
  - and is meaningful! (otherwise, "everything is secure" is composable)
    - Will use SIM security

Security

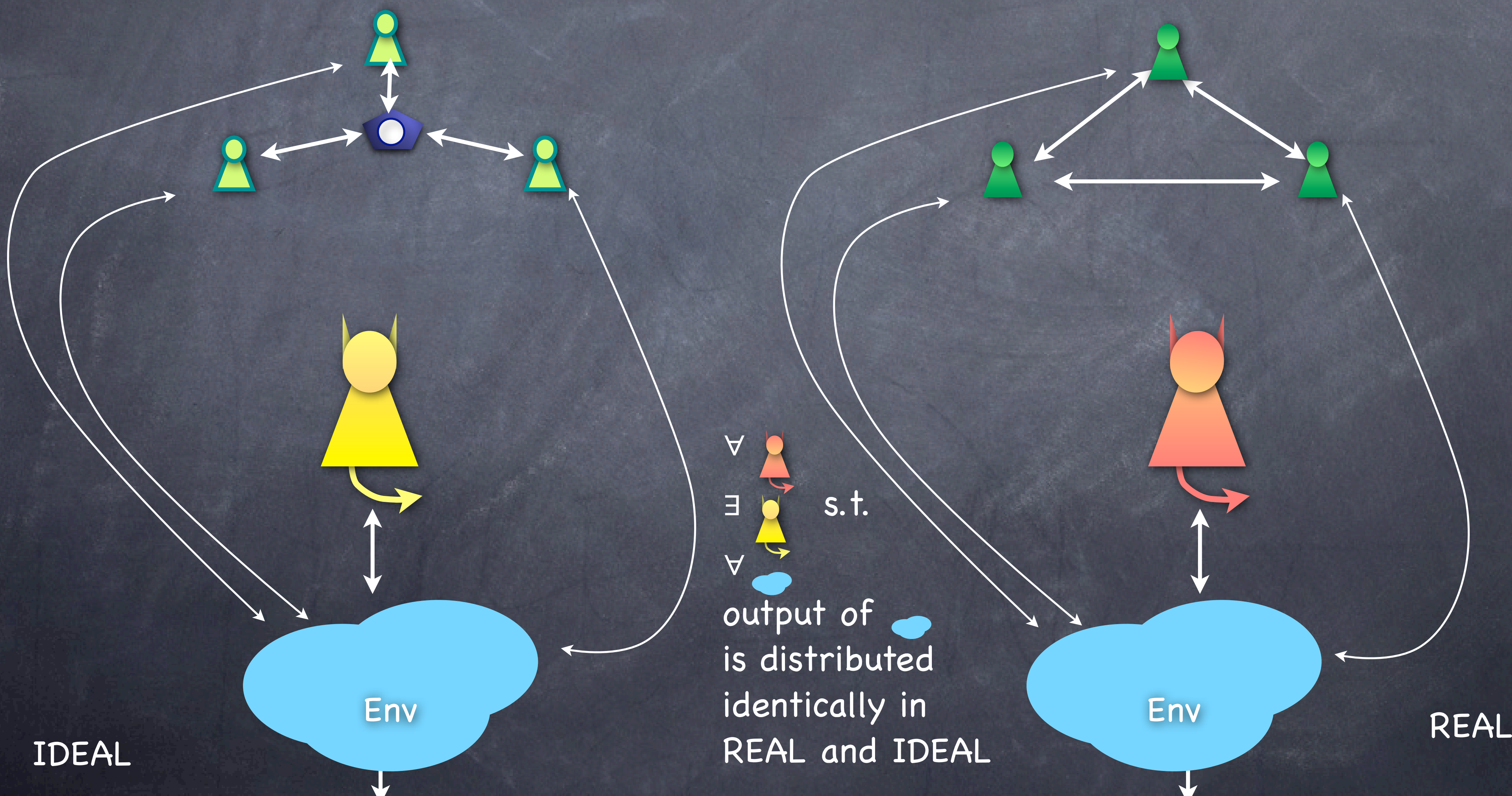
REAL (with protocol) is as secure as IDEAL (with functionality) if:



RECALL

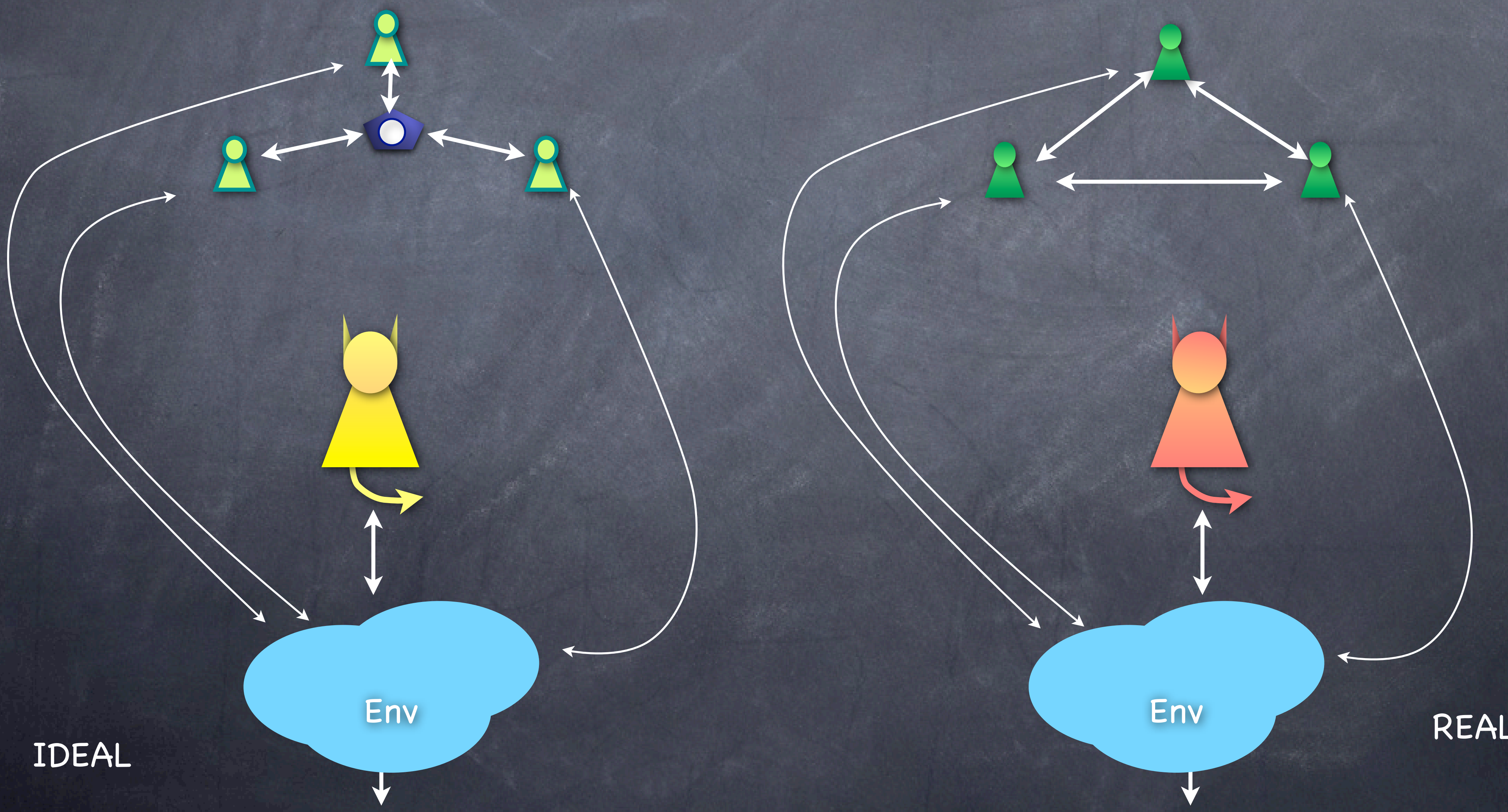
Security

REAL (with protocol) is as secure as IDEAL (with functionality) if:



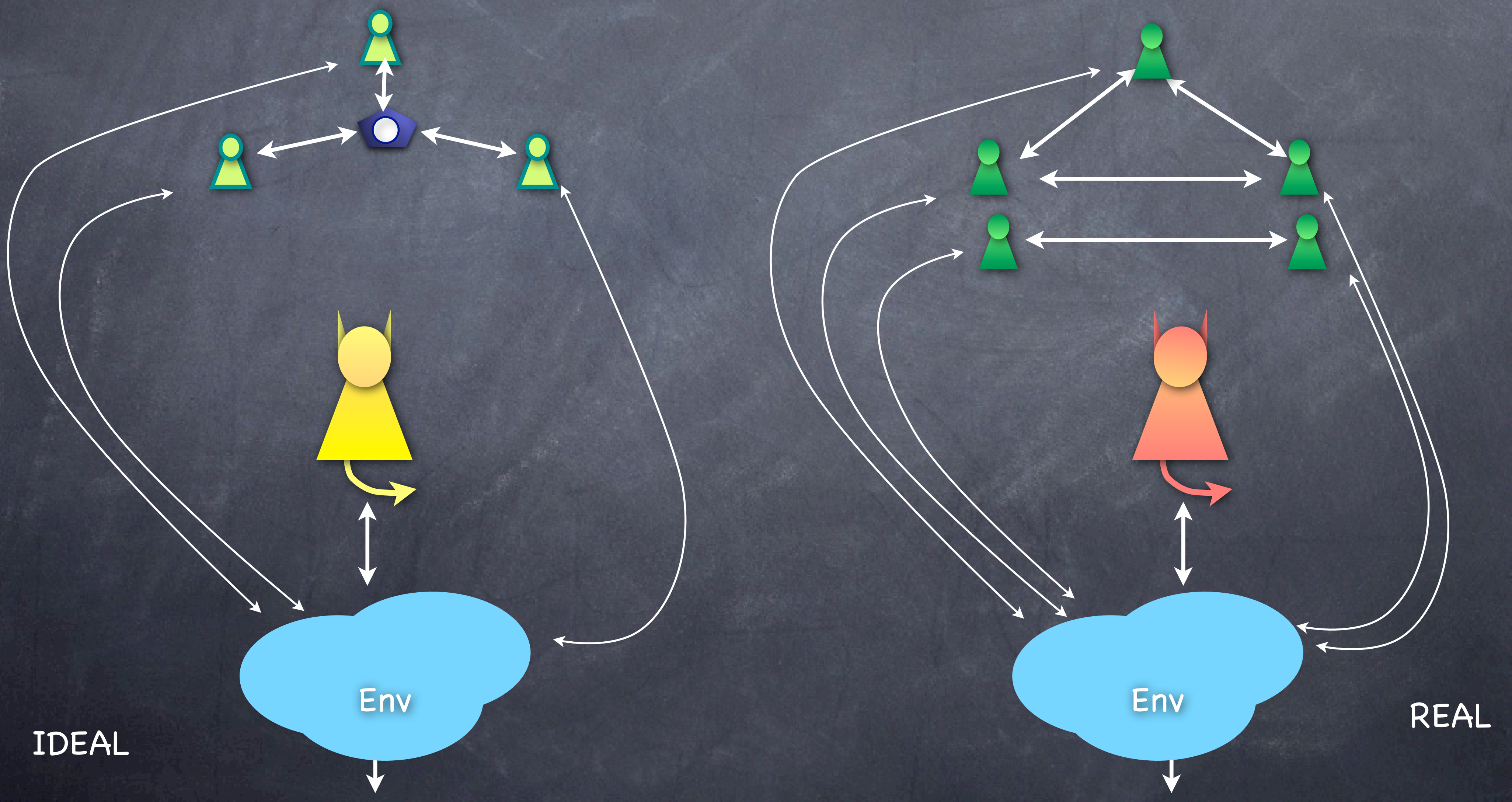
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities



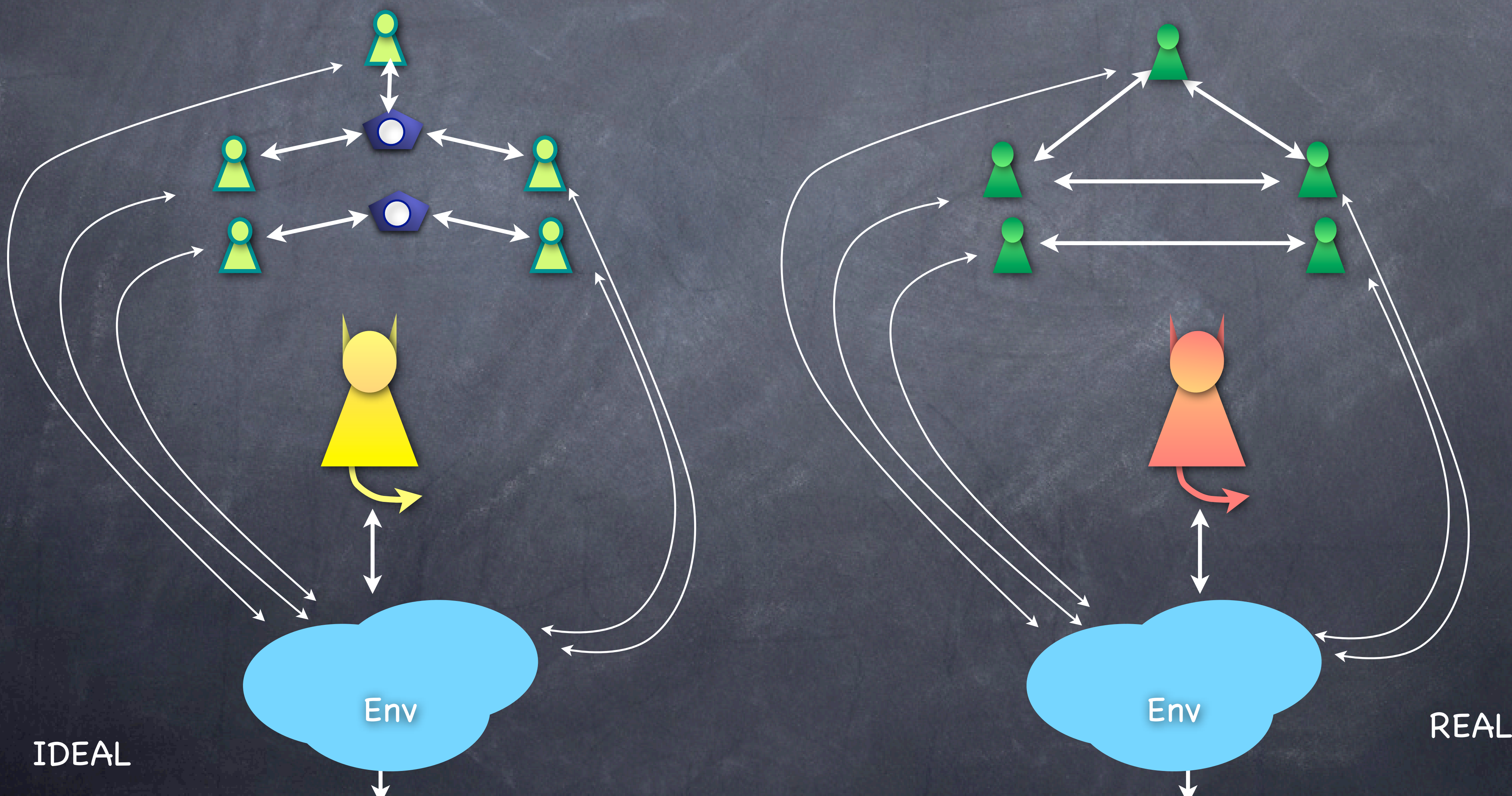
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities



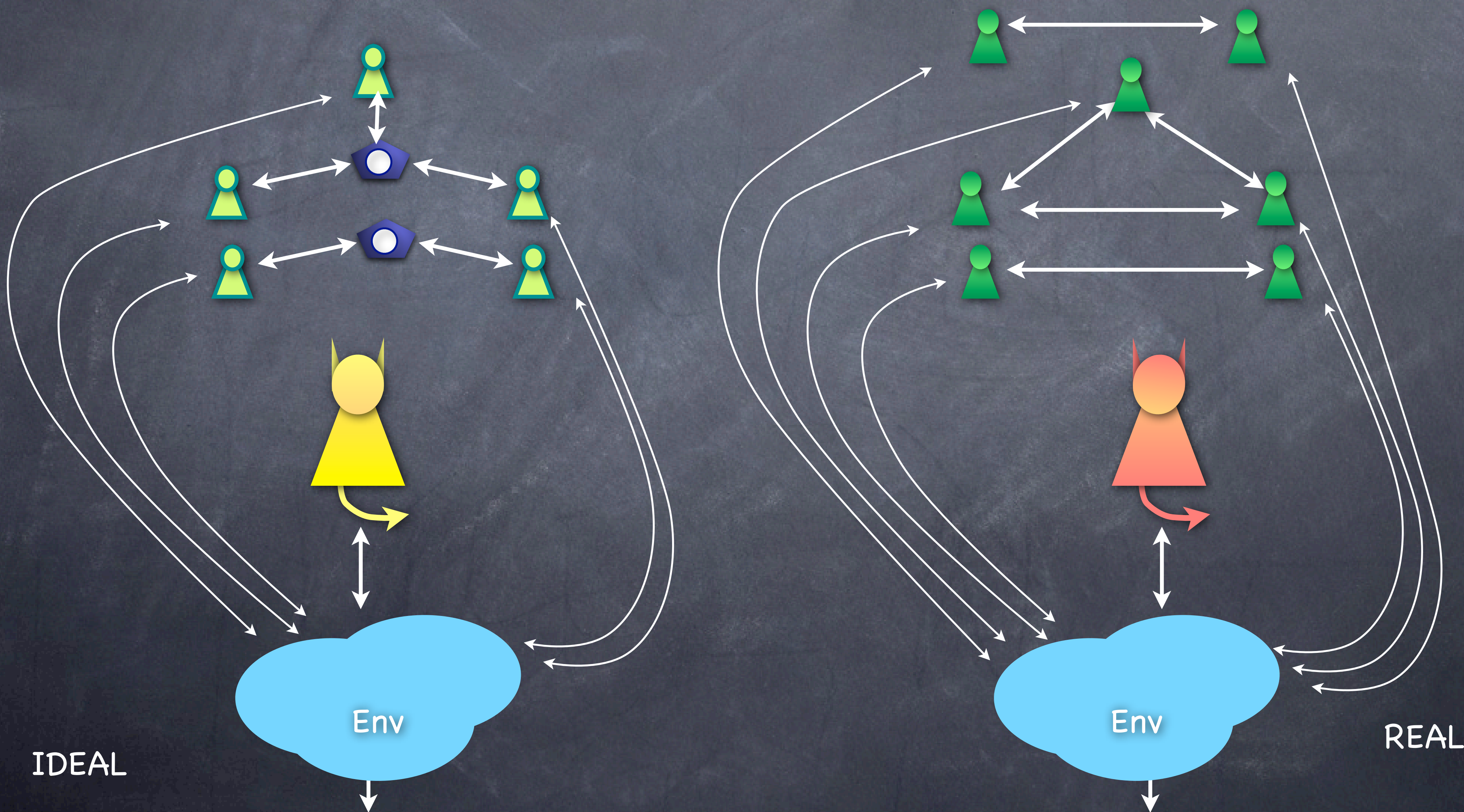
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities



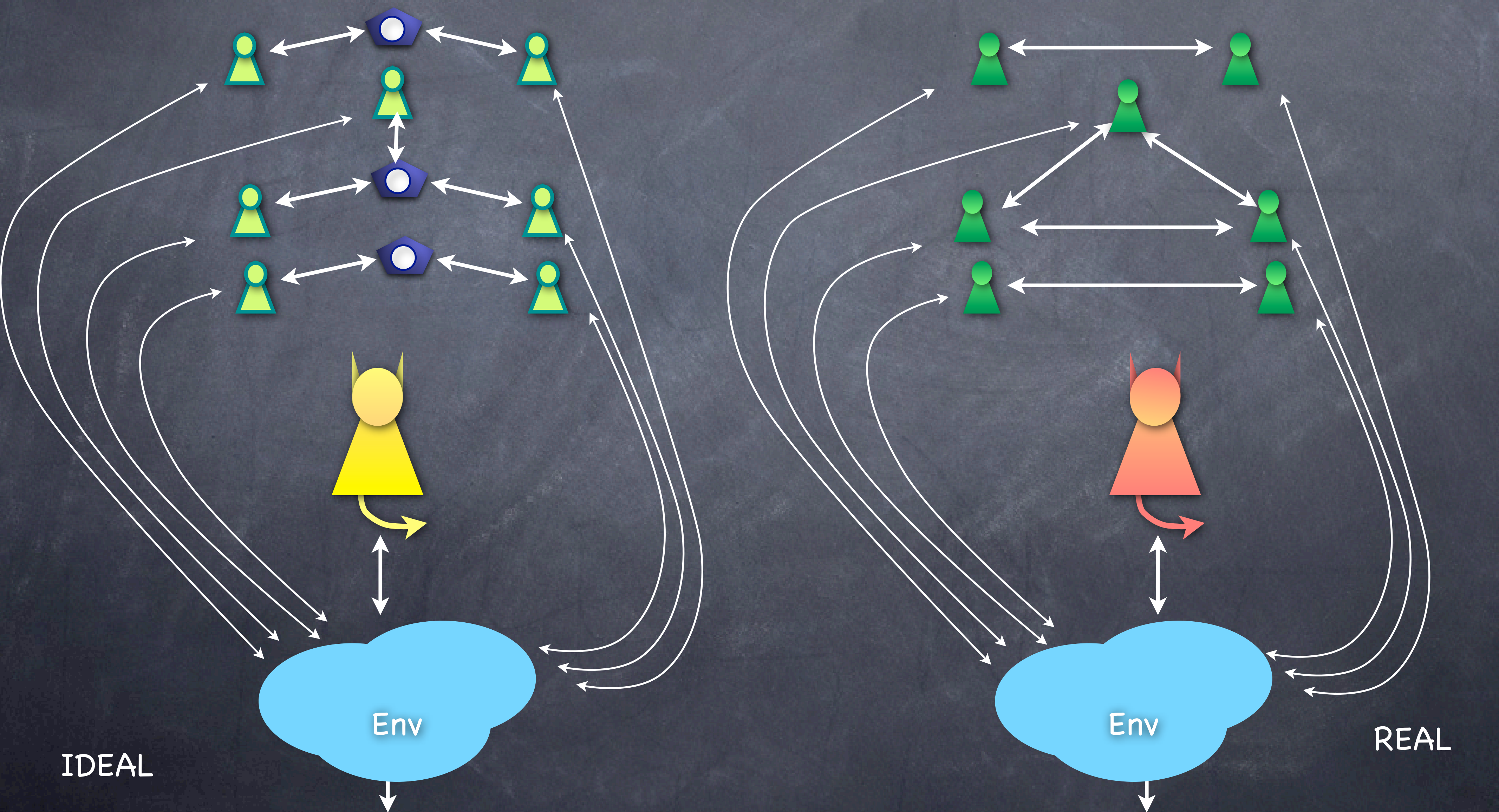
Security of Composed Systems

- Extend to allow a “composed system” with multiple functionalities



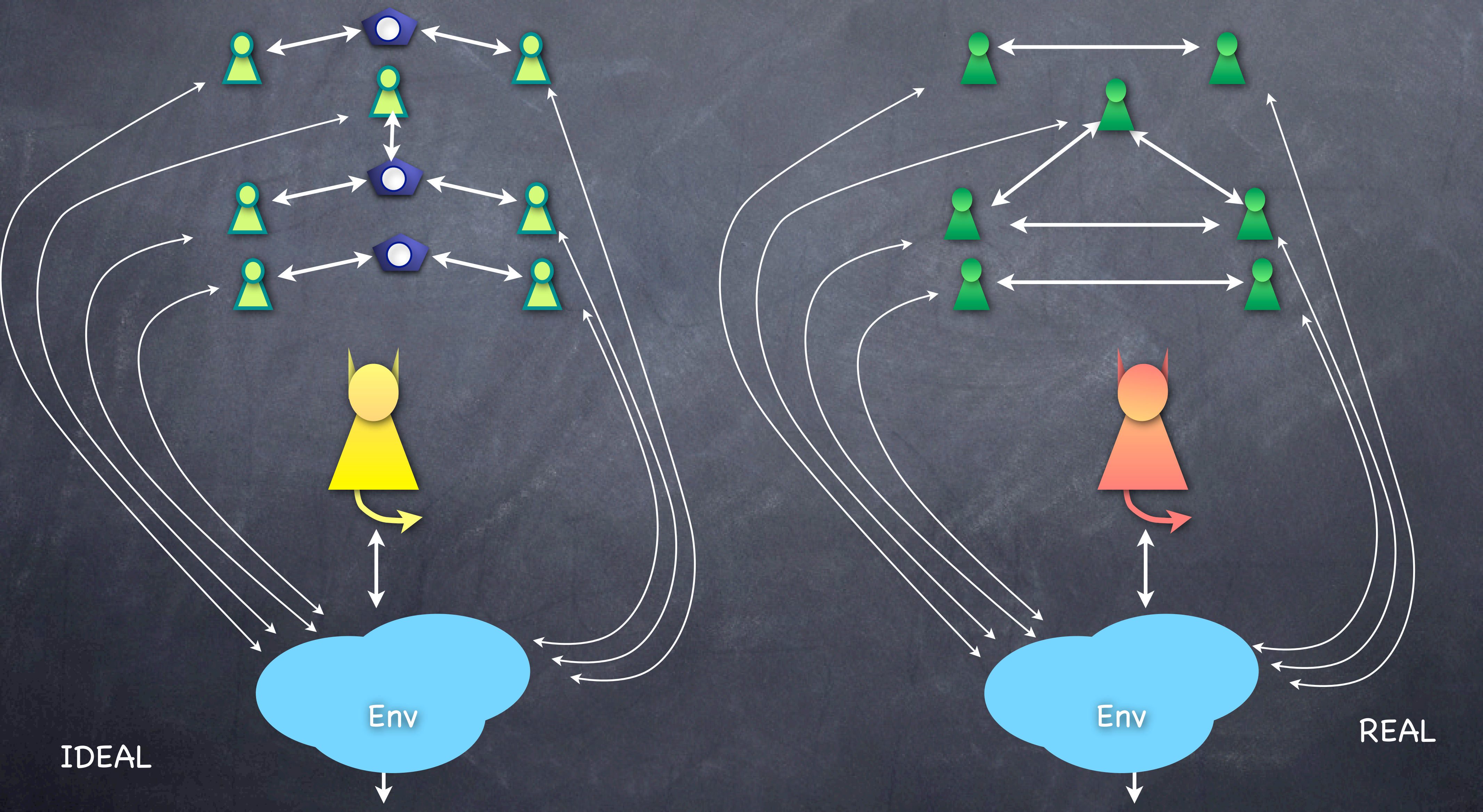
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities



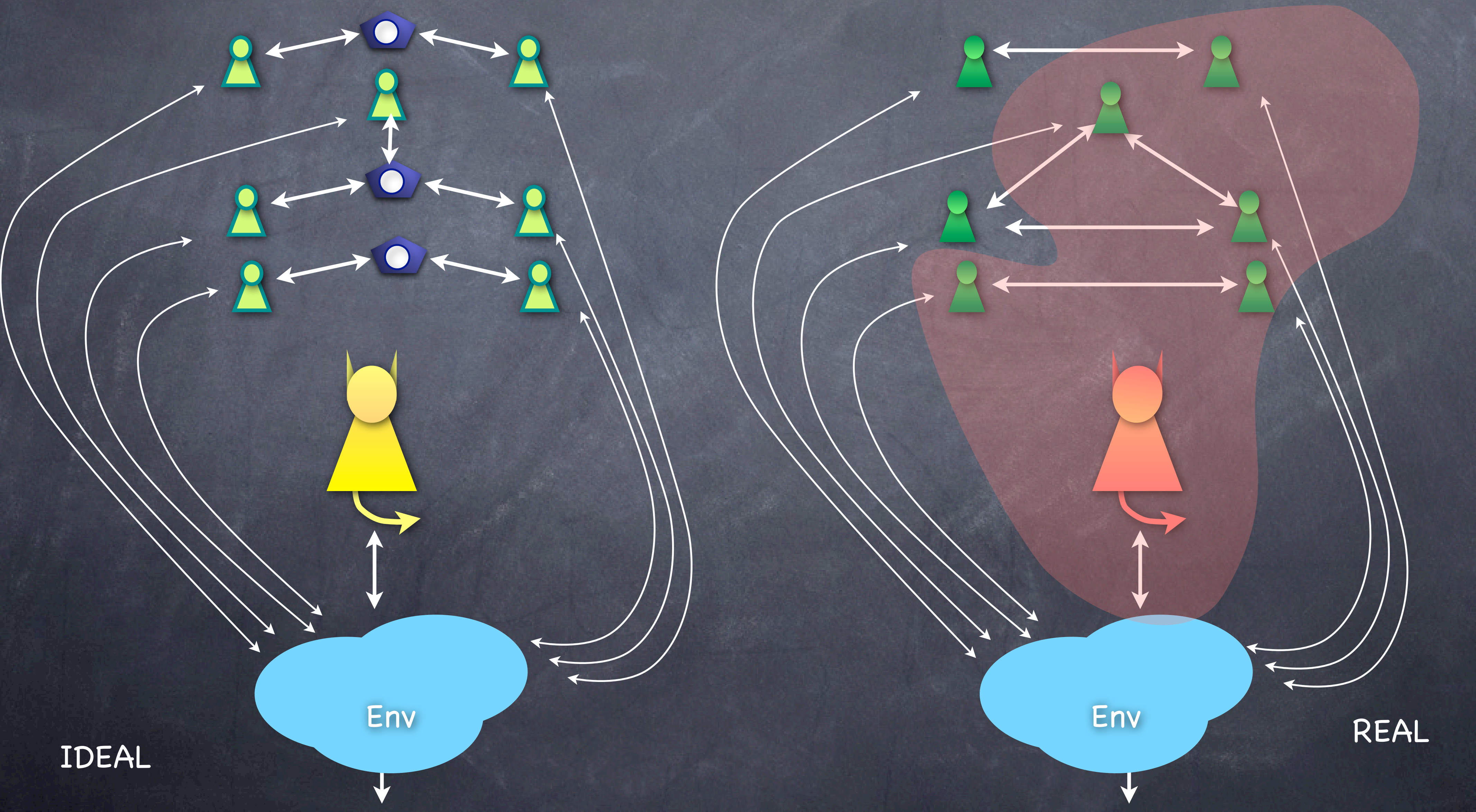
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities
- REAL (with protocols) is as secure as IDEAL (with functionalities) if:



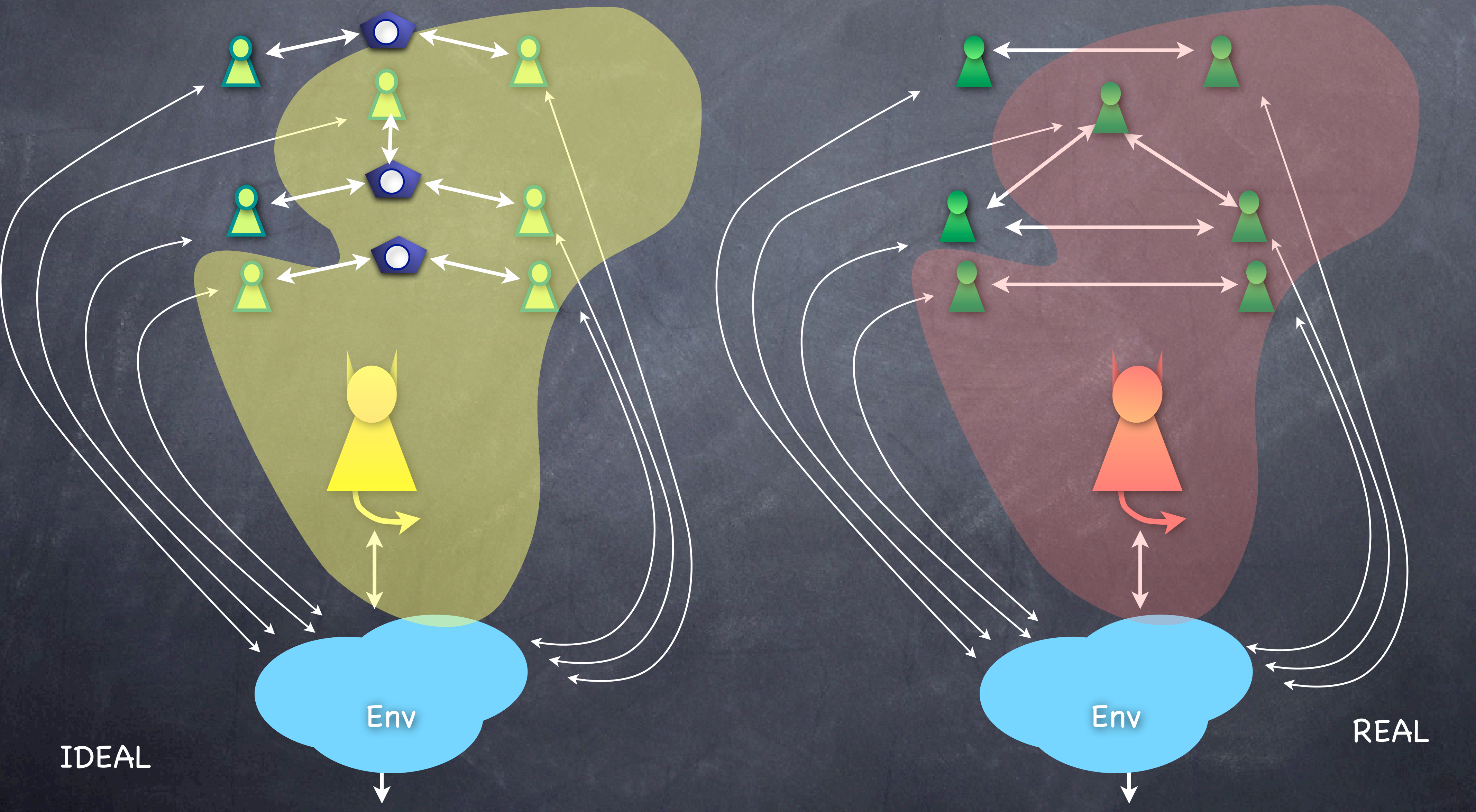
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities
- REAL (with protocols) is as secure as IDEAL (with functionalities) if:



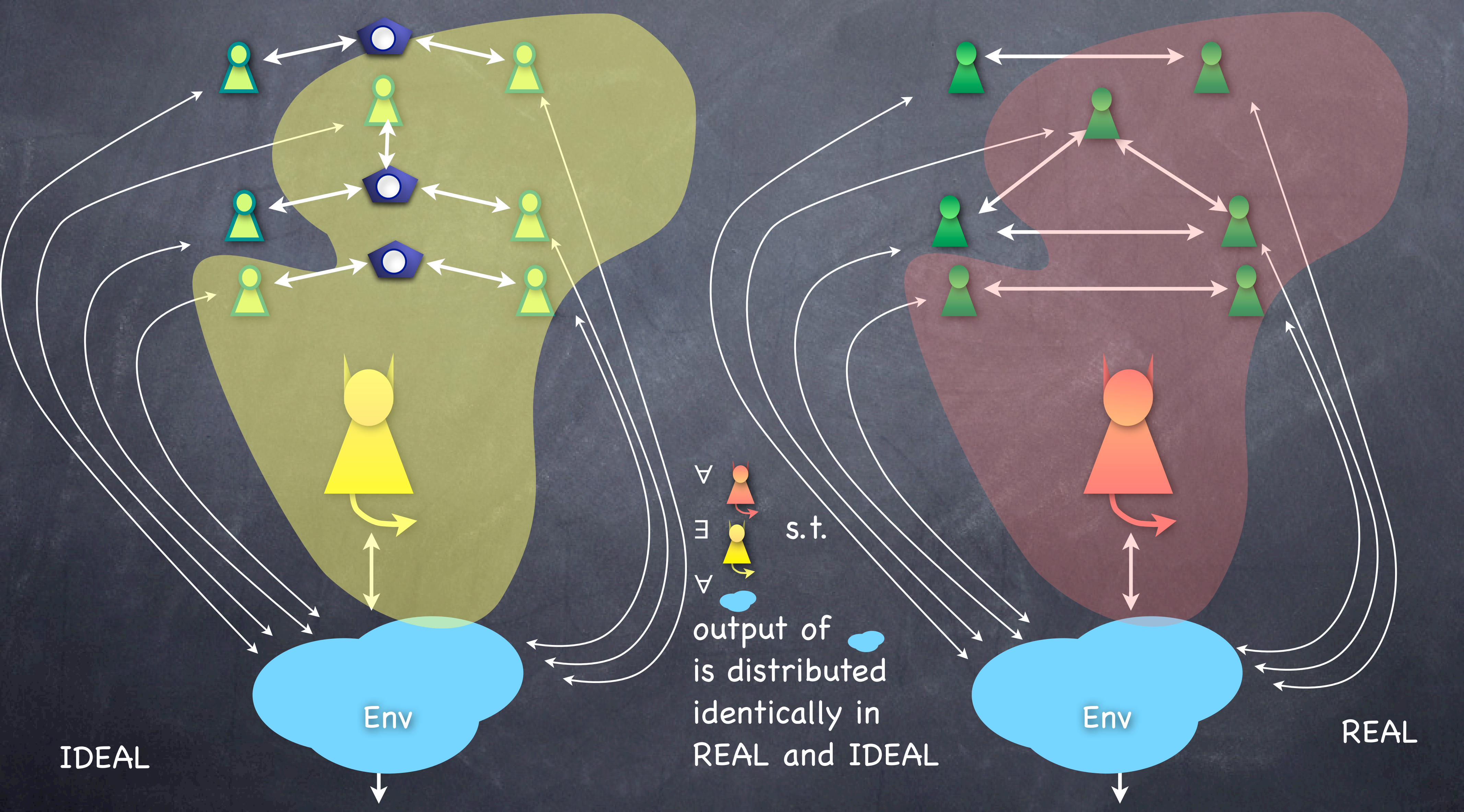
Security of Composed Systems

- Extend to allow a "composed system" with multiple functionalities
- REAL (with protocols) is as secure as IDEAL (with functionalities) if:

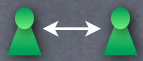
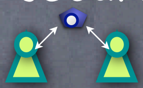


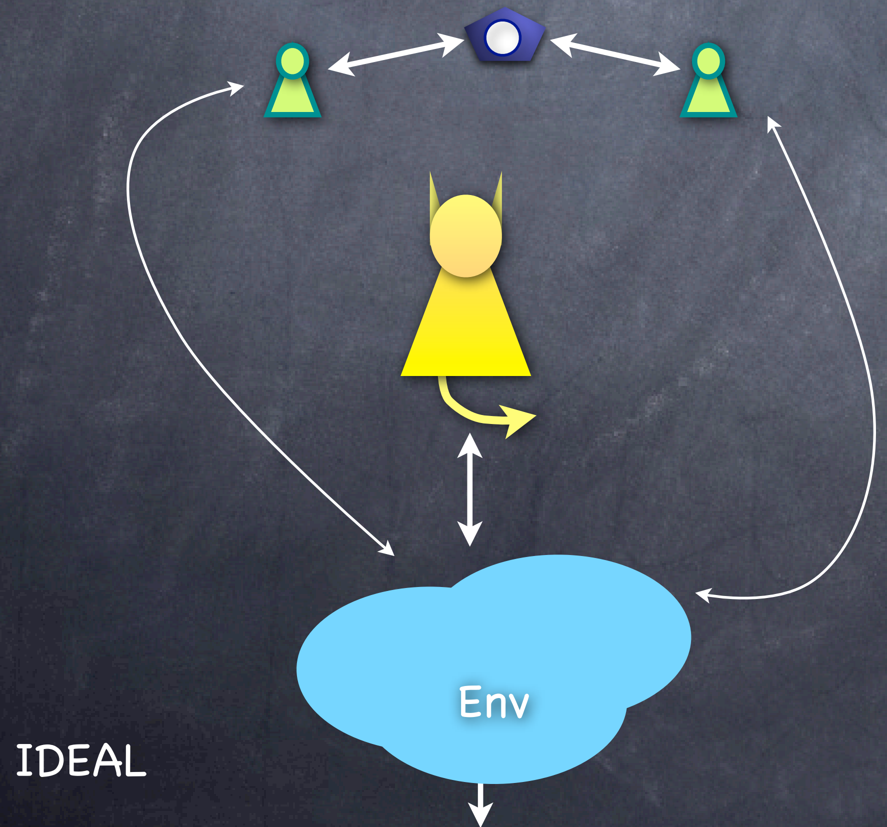
Security of Composed Systems

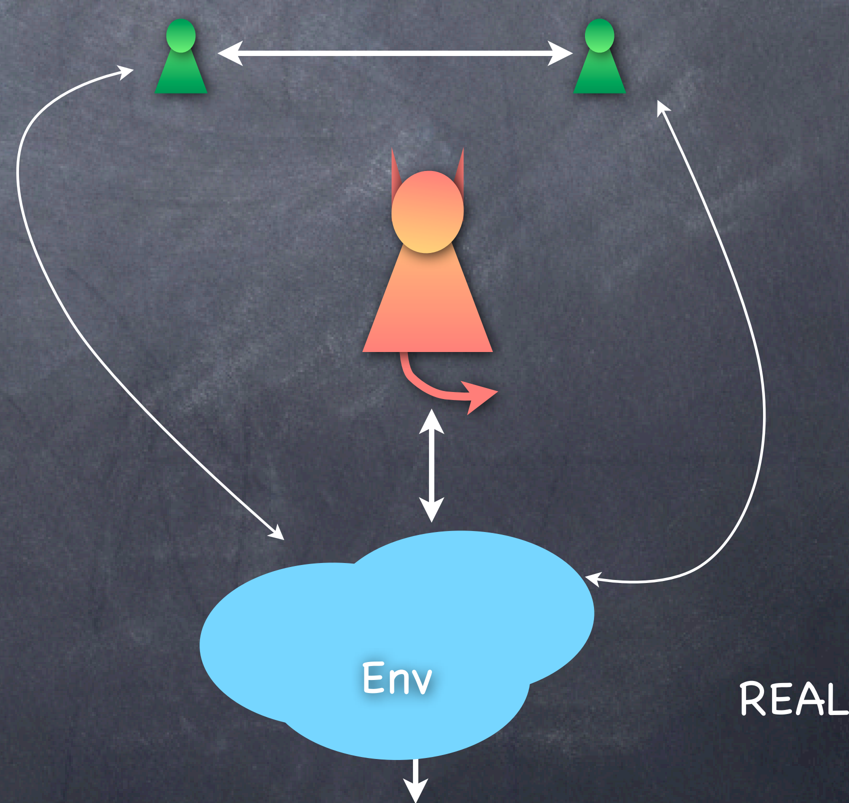
- Extend to allow a "composed system" with multiple functionalities
- REAL (with protocols) is as secure as IDEAL (with functionalities) if:



Universal Composition - 1

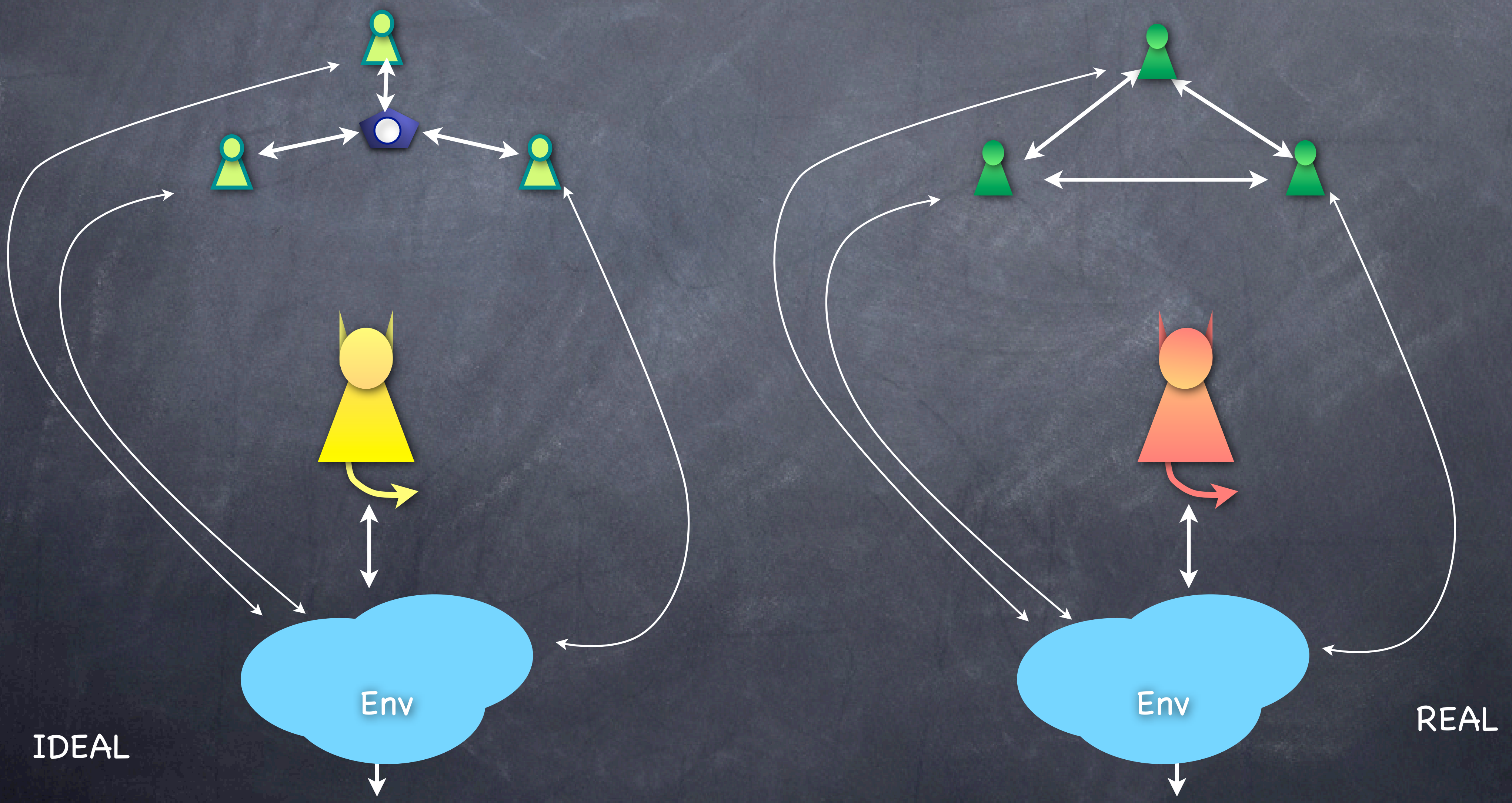
If each protocol secure (i.e.,  is as secure as  etc.)





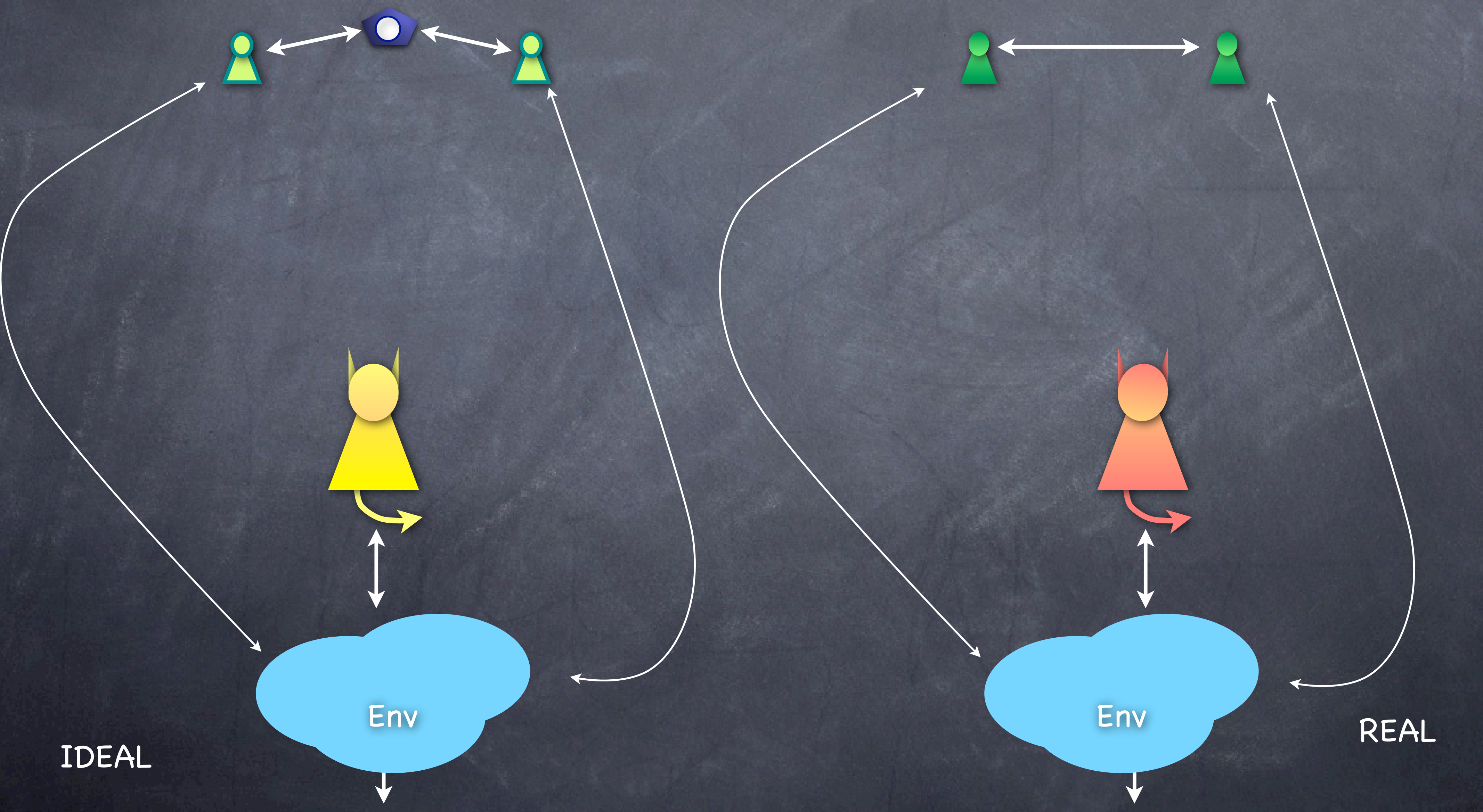
Universal Composition - 1

If each protocol secure (i.e., ↔ is as secure as etc.)



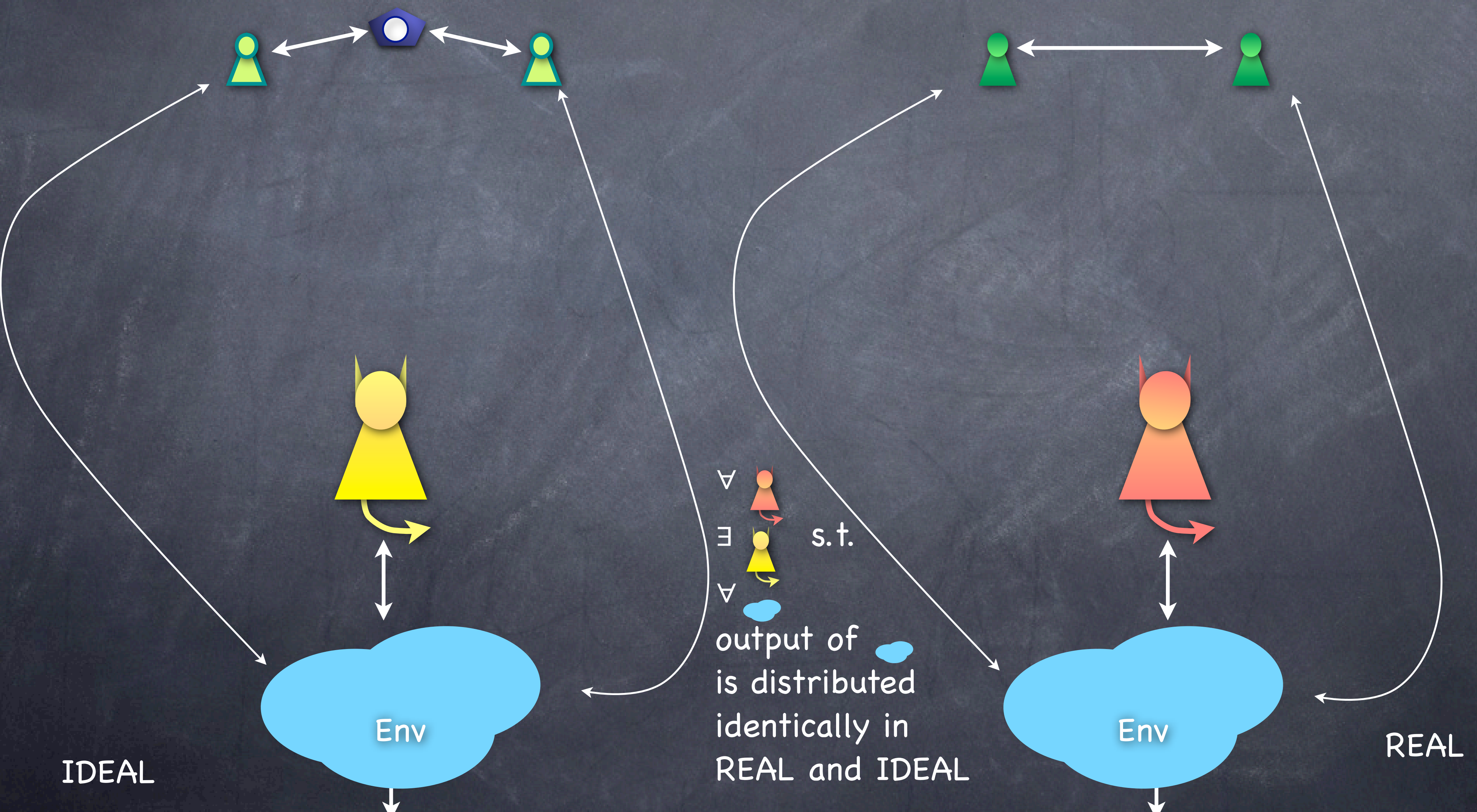
Universal Composition - 1

If each protocol secure (i.e., is as secure as etc.)



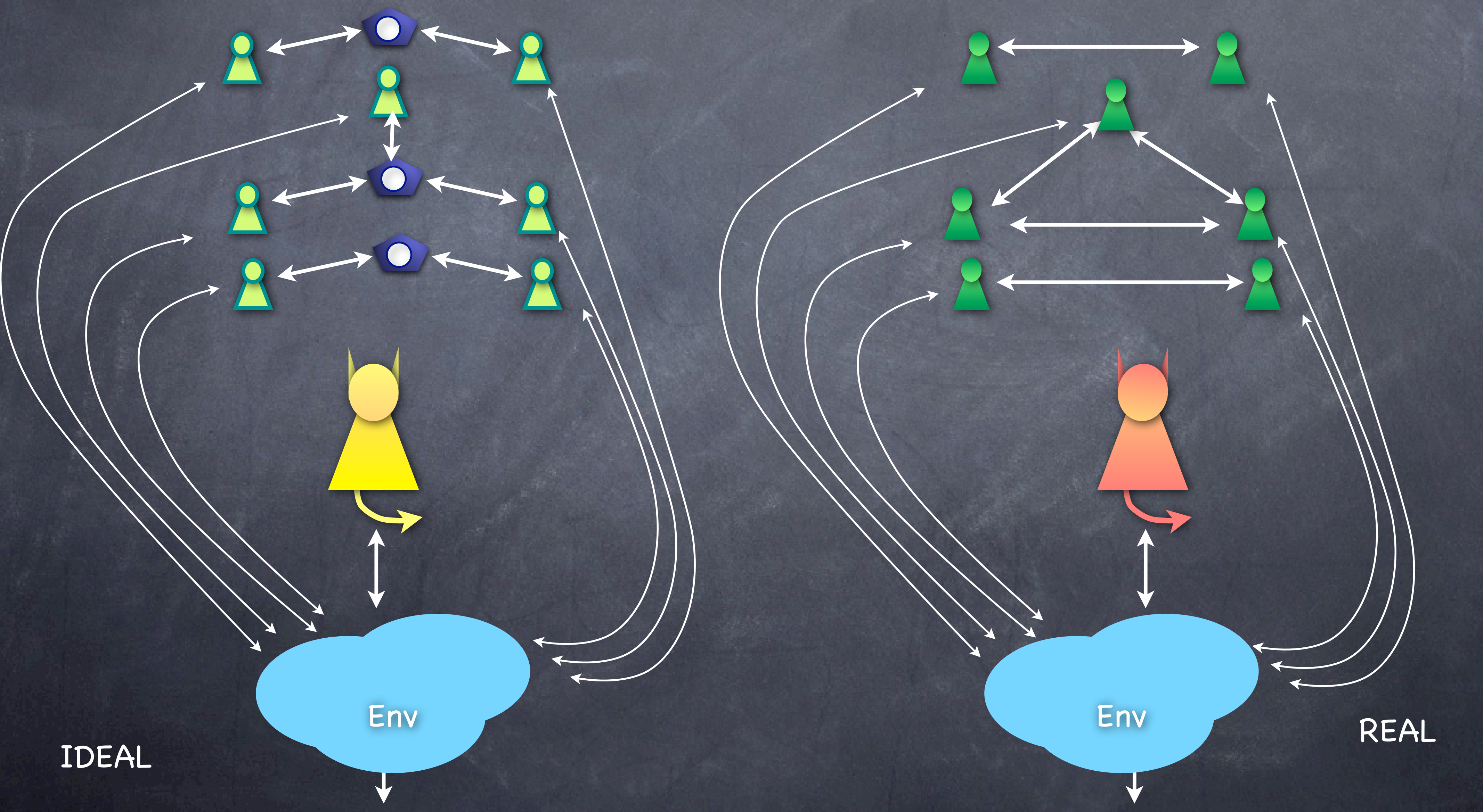
Universal Composition - 1

If each protocol secure (i.e., ↔ is as secure as etc.)



Universal Composition - 1

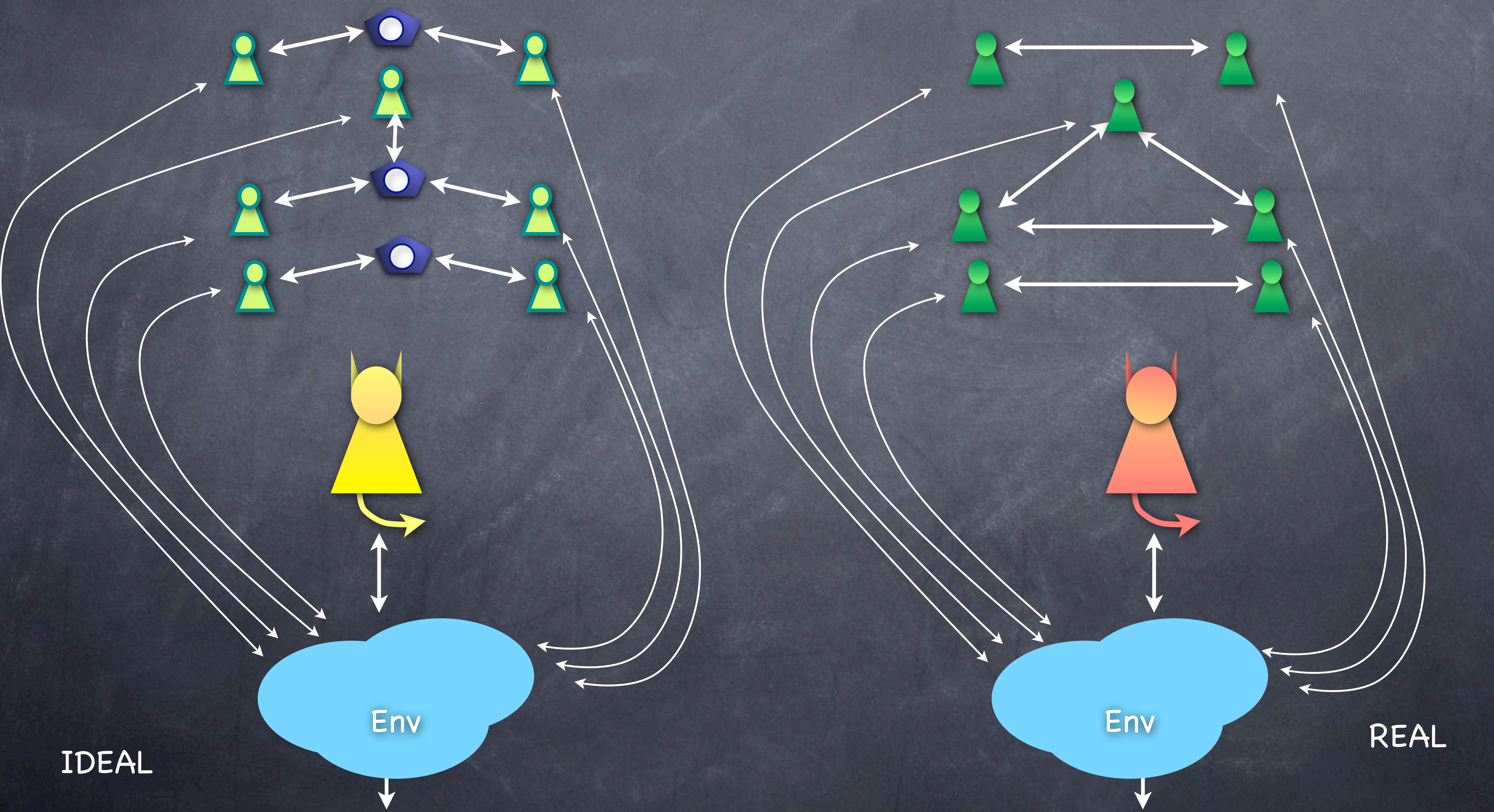
then concurrent sessions are secure too



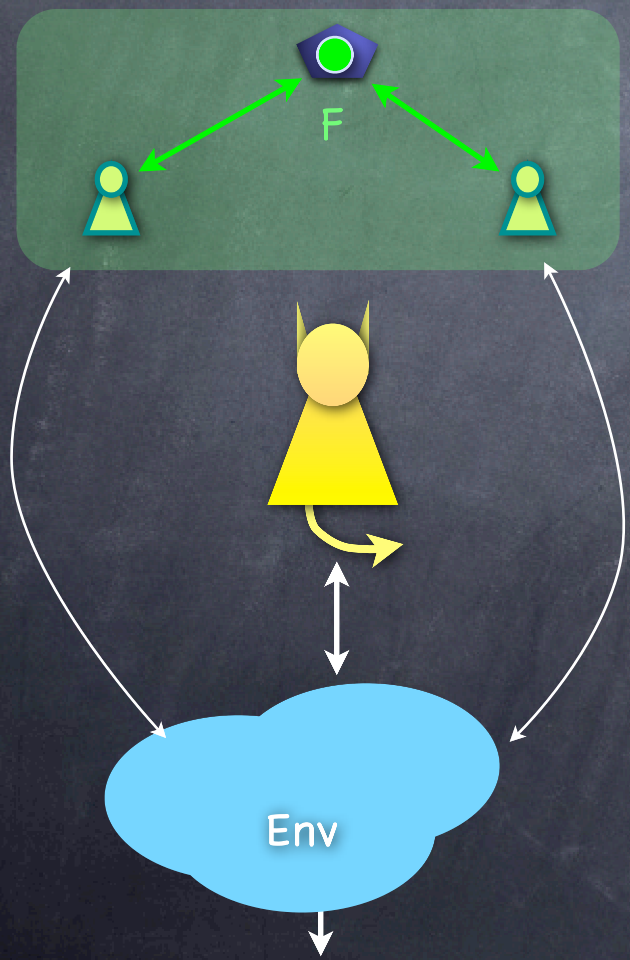
Universal Composition - 1

then concurrent sessions are secure too

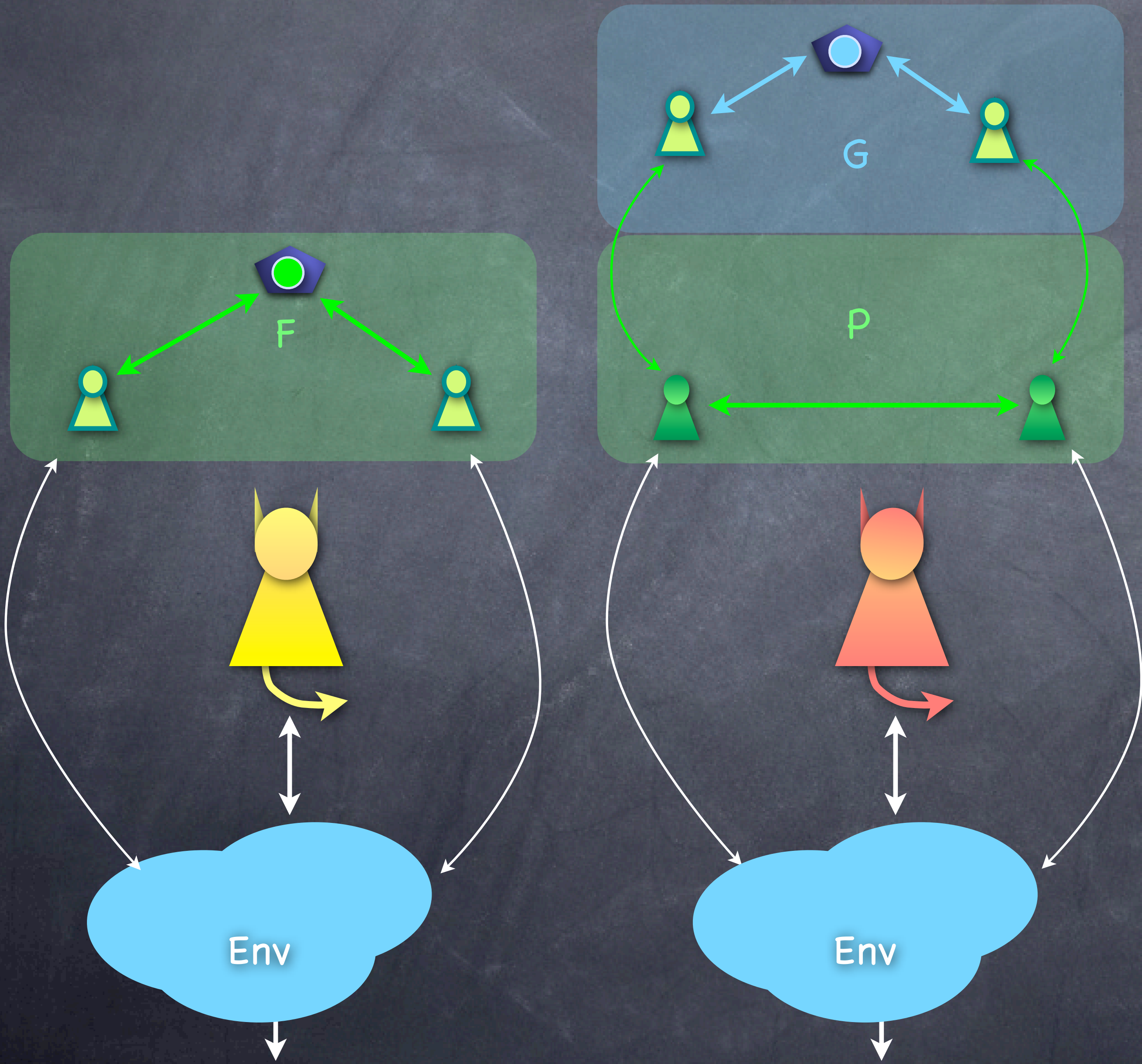
i.e., is as secure as etc.



Universal Composition - 2

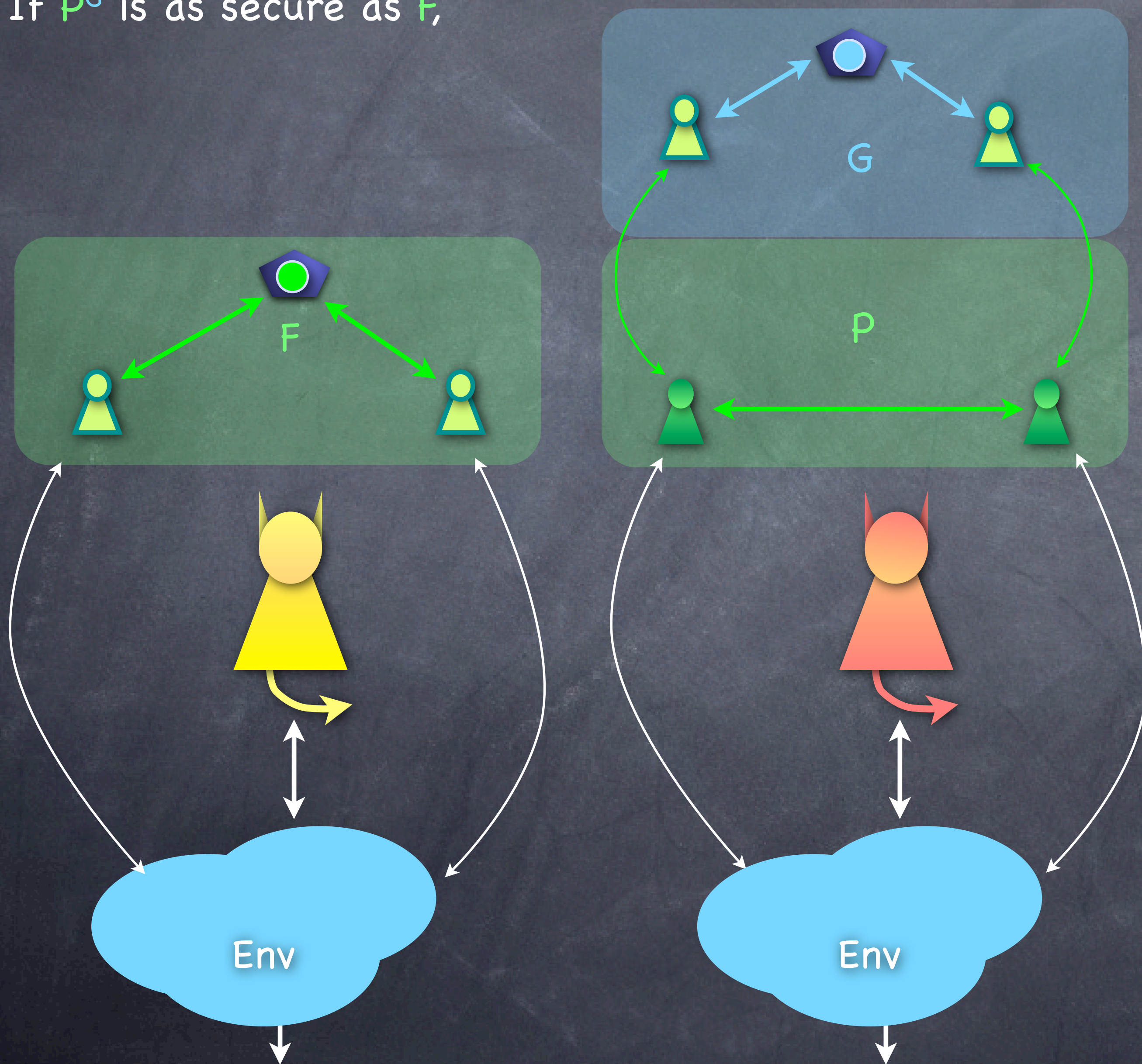


Universal Composition - 2



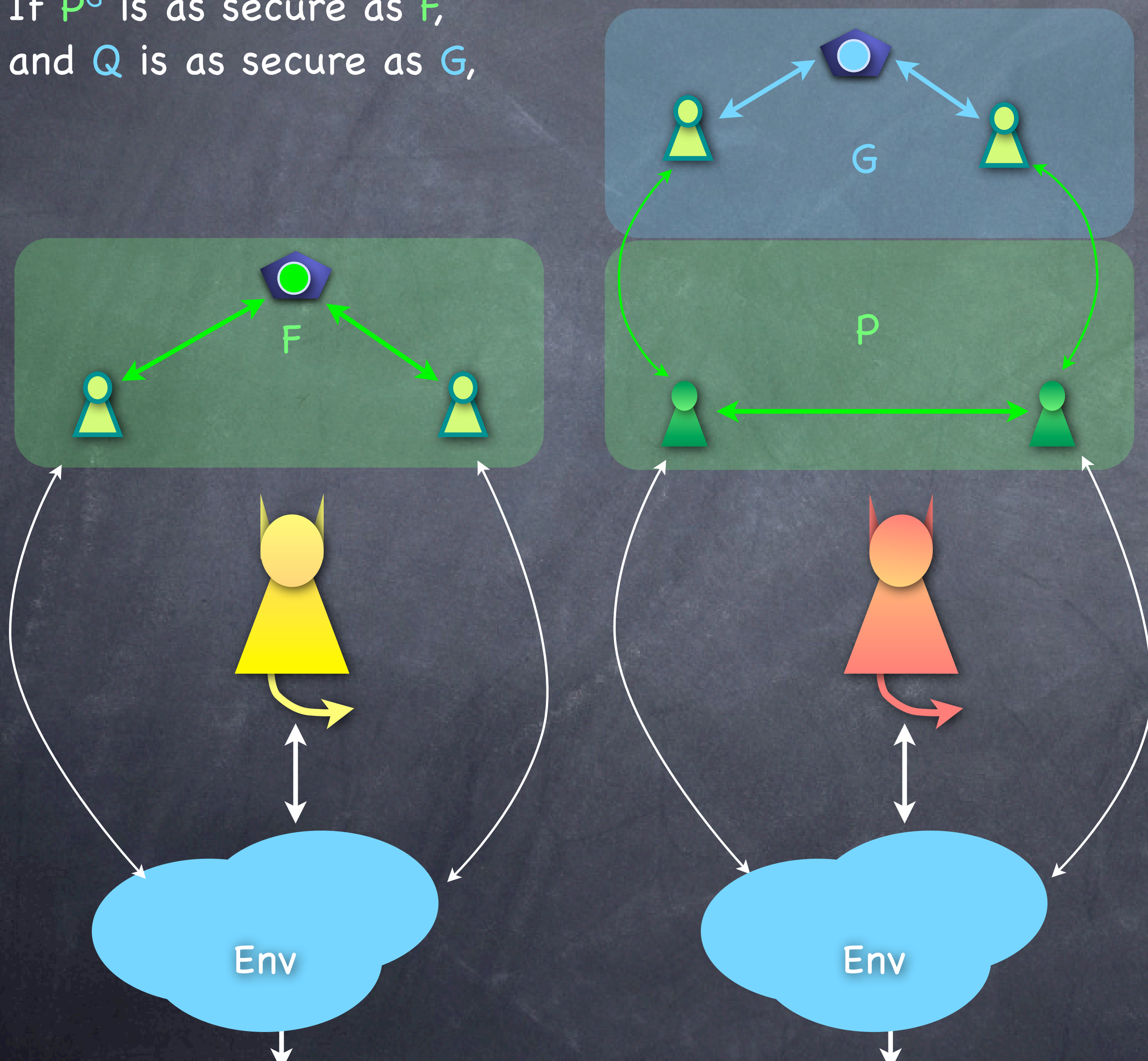
Universal Composition - 2

If P^G is as secure as F,



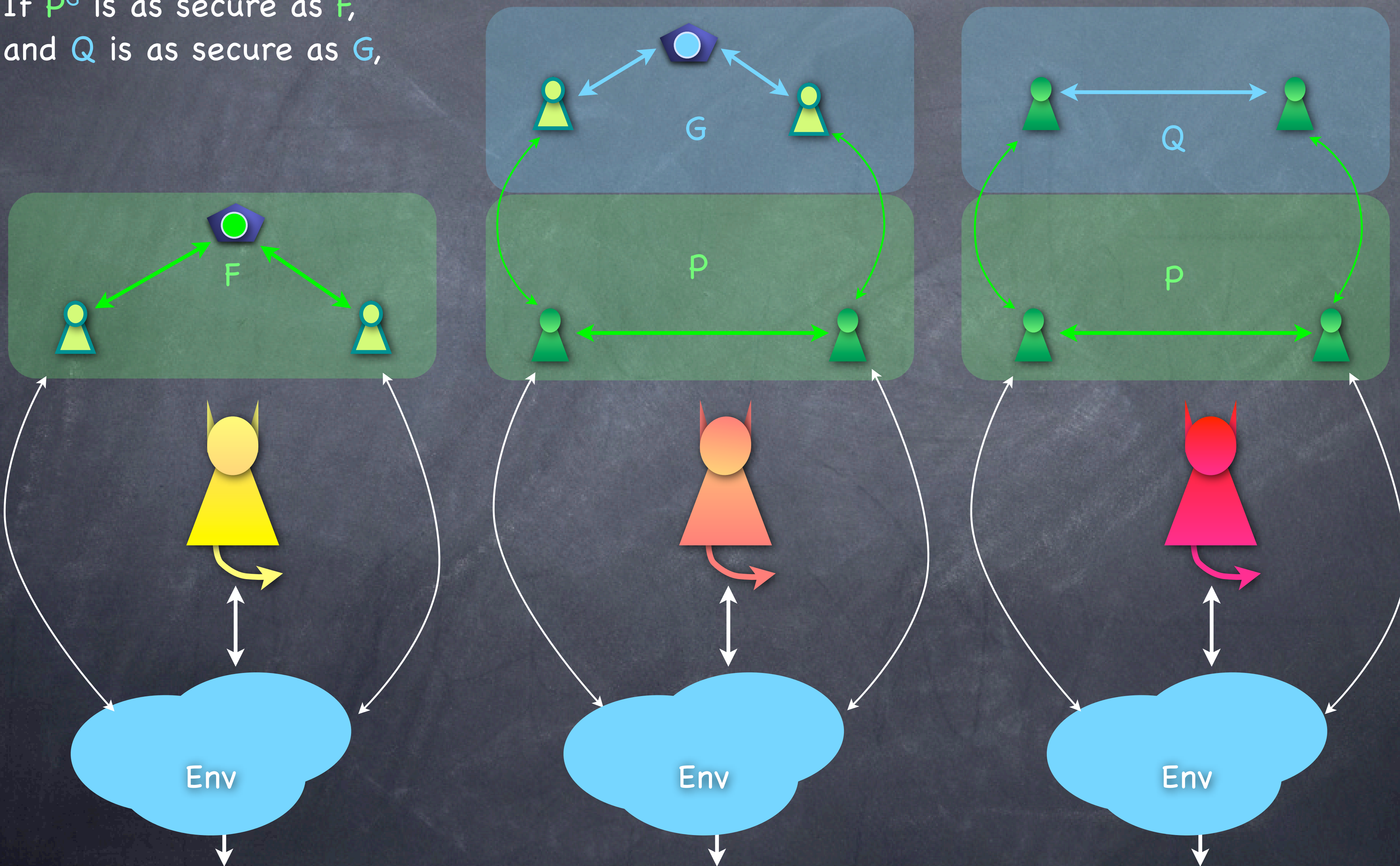
Universal Composition - 2

If P^G is as secure as F,
and Q is as secure as G,



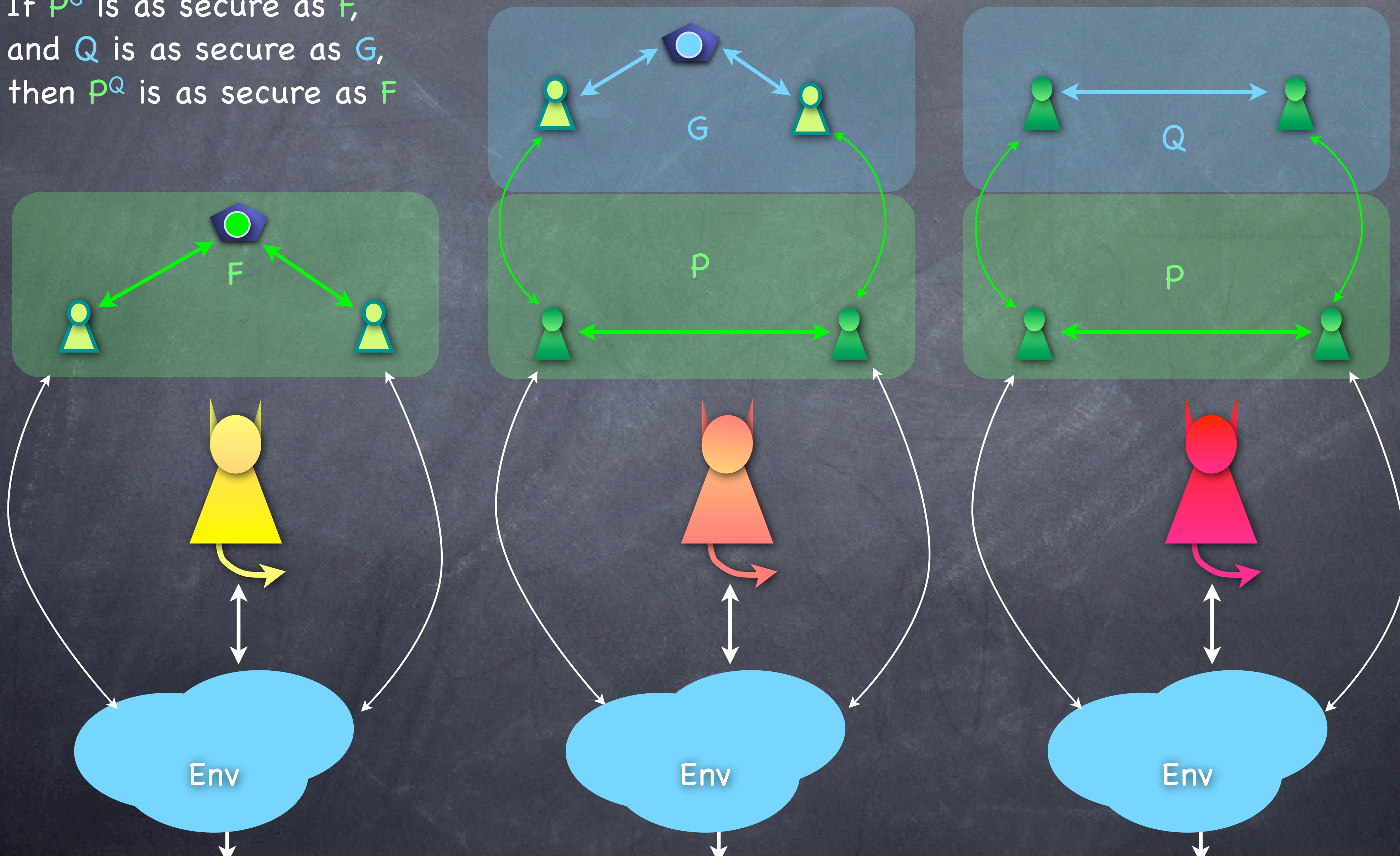
Universal Composition - 2

If P^G is as secure as F,
and Q is as secure as G,



Universal Composition - 2

If P^G is as secure as F,
and Q is as secure as G,
then P^Q is as secure as F



Universal Composition

Universal Composition

- More generally:

Universal Composition

- More generally:
  - Start from world A (think "IDEAL")

Universal Composition

- More generally:
  - Start from world A (think "IDEAL")
    - Repeat (for any poly number of times):

Universal Composition

- More generally:
  - 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

Universal Composition

- More generally:
  - 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")

Universal Composition

- More generally:
  - 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

Universal Composition

- More generally:
  - 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

UC and SIM-security

UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition

UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition
  - Even when considering only one component, other components could be present in the environment

UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition
  - Even when considering only one component, other components could be present in the environment
- Considering an arbitrary environment is anyway necessary for the security guarantee to be useful

UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition
  - Even when considering only one component, other components could be present in the environment
- Considering an arbitrary environment is anyway necessary for the security guarantee to be useful
  - But by itself may not imply universal composition: e.g. with PPT REAL world, unbounded IDEAL (simulator or functionality)

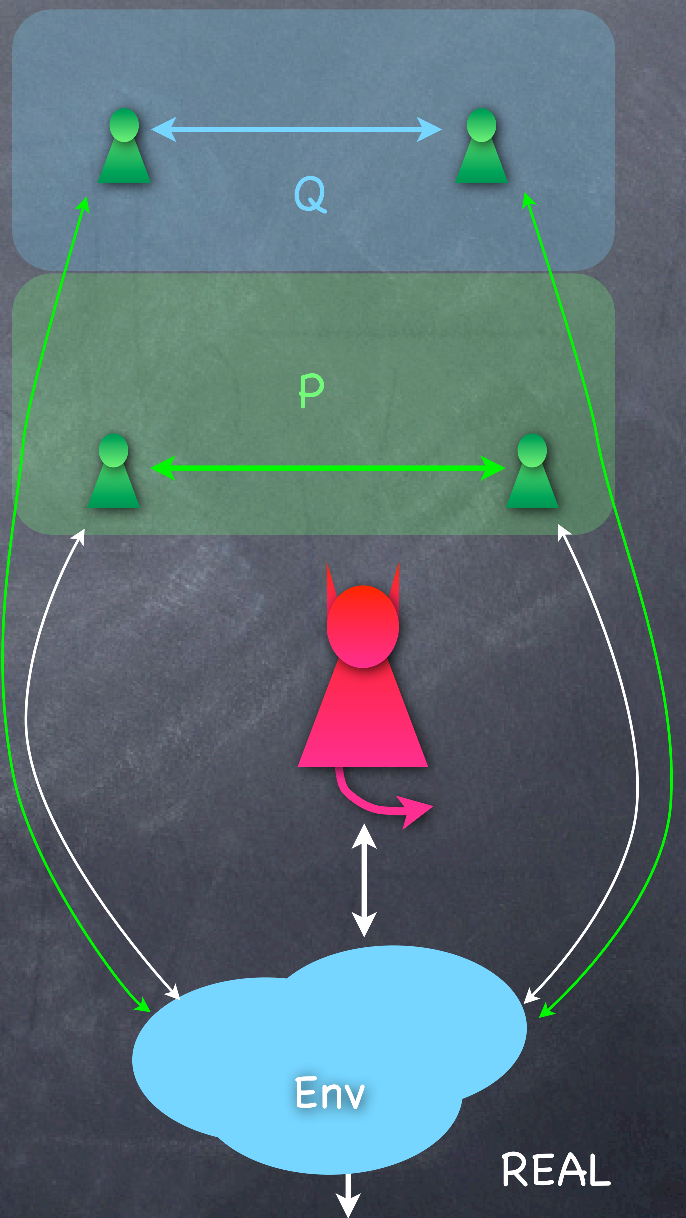
UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition
  - Even when considering only one component, other components could be present in the environment
- Considering an arbitrary environment is anyway necessary for the security guarantee to be useful
  - But by itself may not imply universal composition: e.g. with PPT REAL world, unbounded IDEAL (simulator or functionality)
  - Also, UC by itself does not imply a meaningful security (nor require an environment)

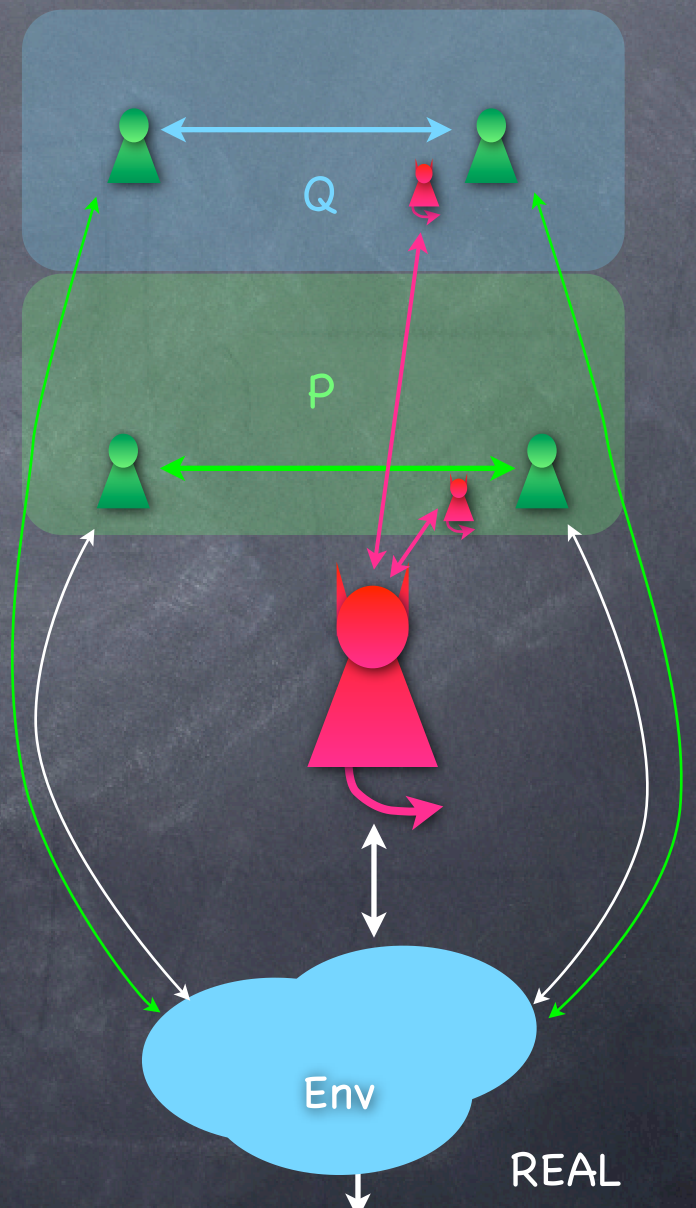
UC and SIM-security

- Key to universal composition is allowing an arbitrary environment in the SIM-security definition
  - Even when considering only one component, other components could be present in the environment
- Considering an arbitrary environment is anyway necessary for the security guarantee to be useful
  - But by itself may not imply universal composition: e.g. with PPT REAL world, unbounded IDEAL (simulator or functionality)
  - Also, UC by itself does not imply a meaningful security (nor require an environment)
    - e.g. Define security of composed system as security of each individual component; Or, define everything secure.

Proving the UC theorem

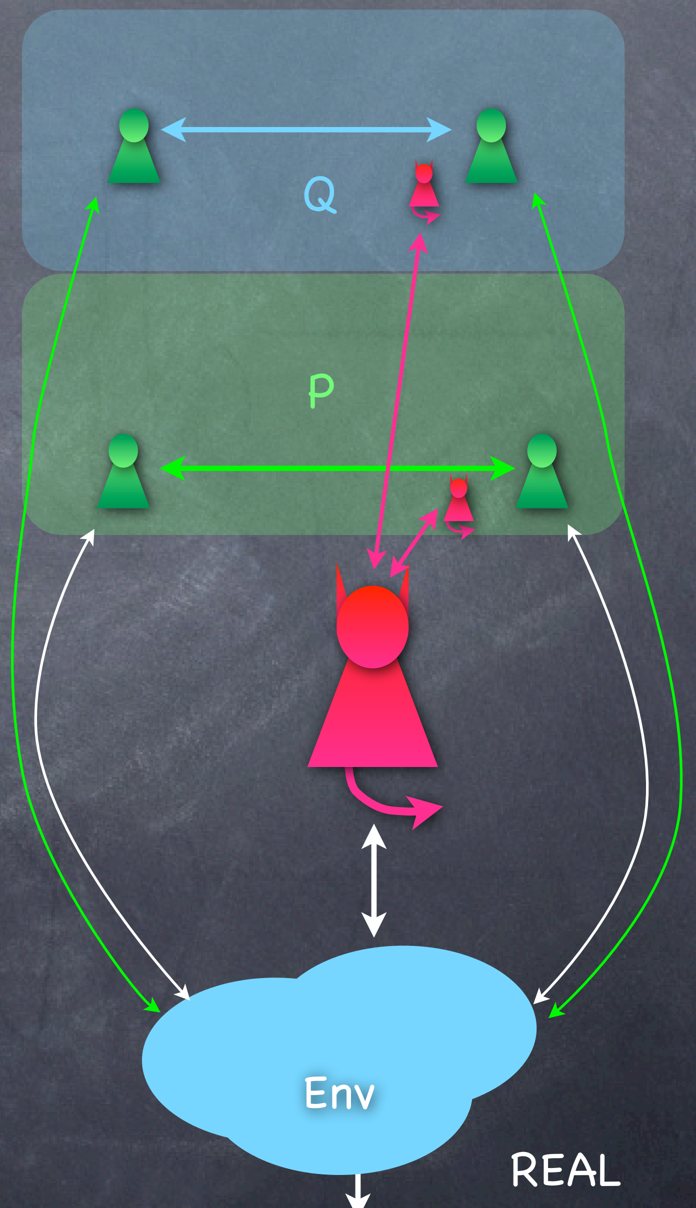


Proving the UC theorem



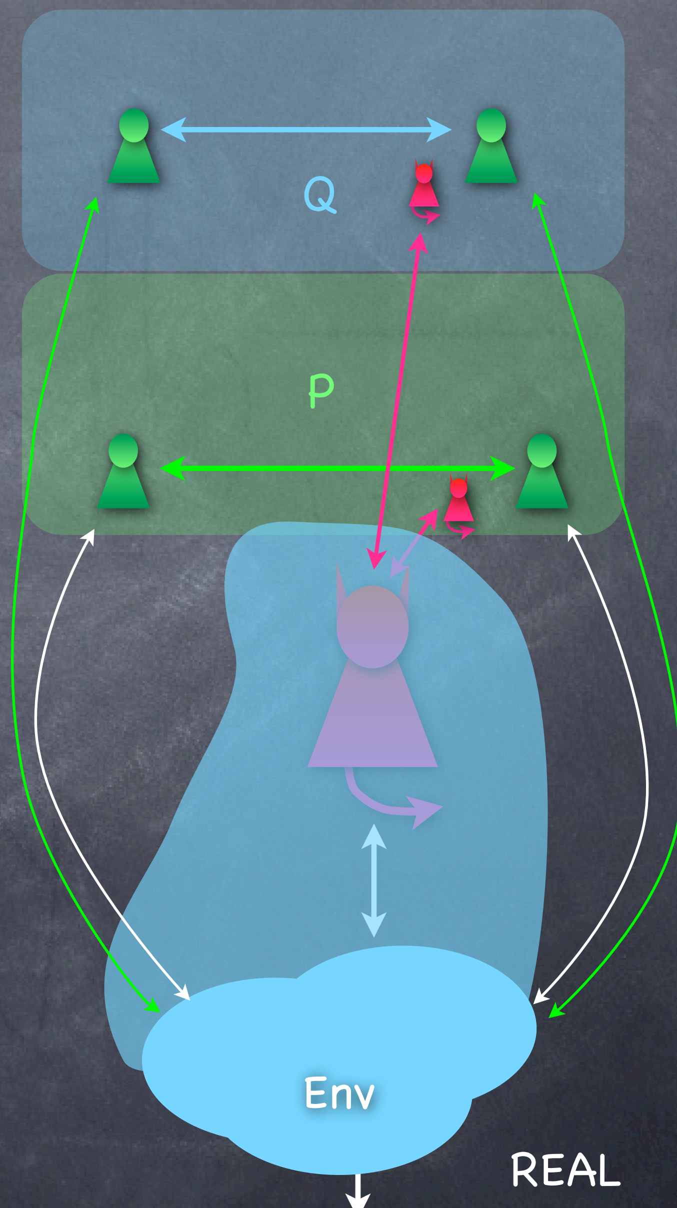
Proving the UC theorem

- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols



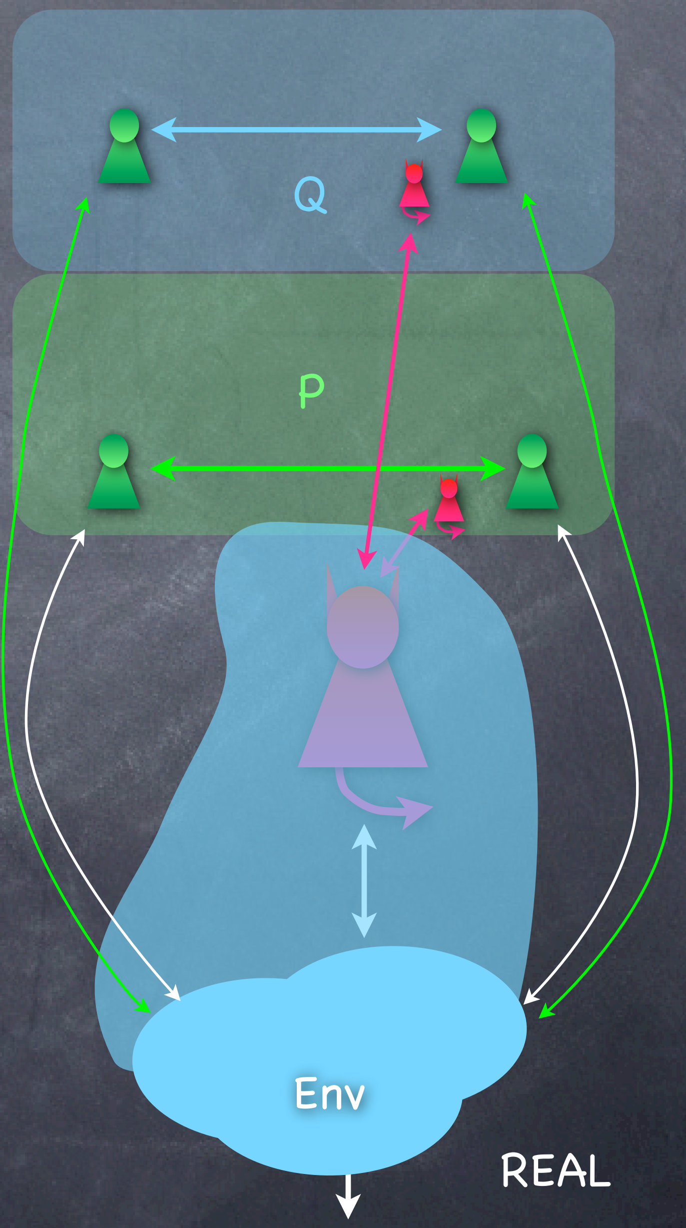
Proving the UC theorem

- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols



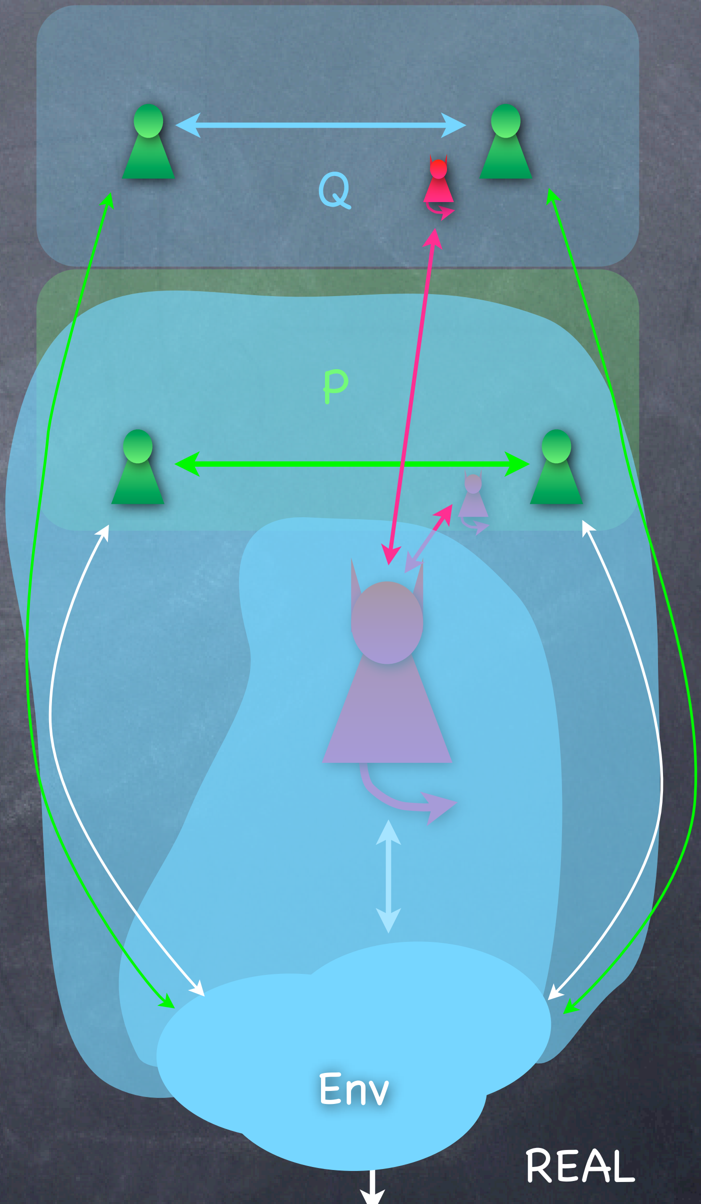
Proving the UC theorem

- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols
- Now consider new environment s.t. only Q (and its adversary) is outside it



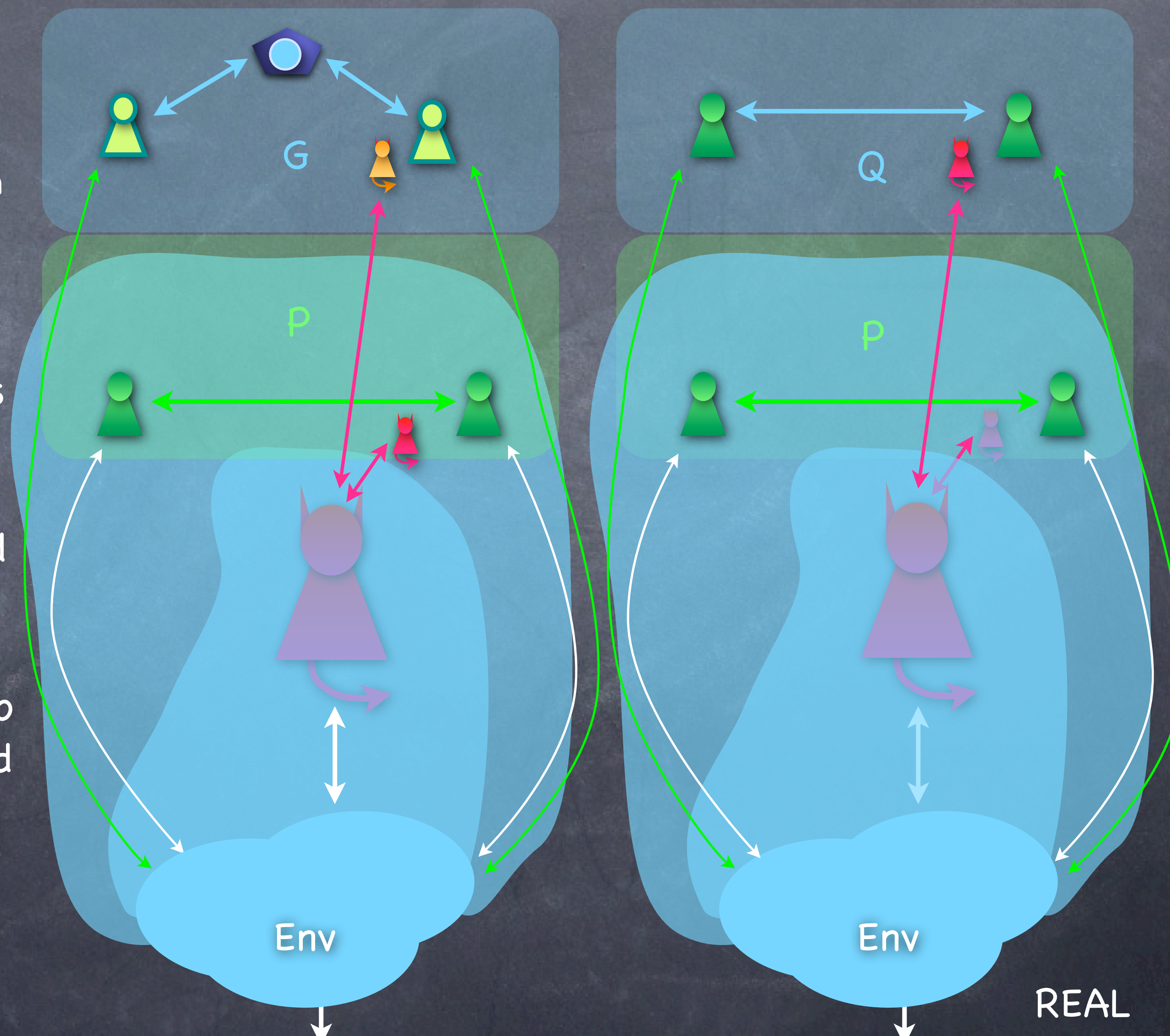
Proving the UC theorem

- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols
- Now consider new environment s.t. only Q (and its adversary) is outside it



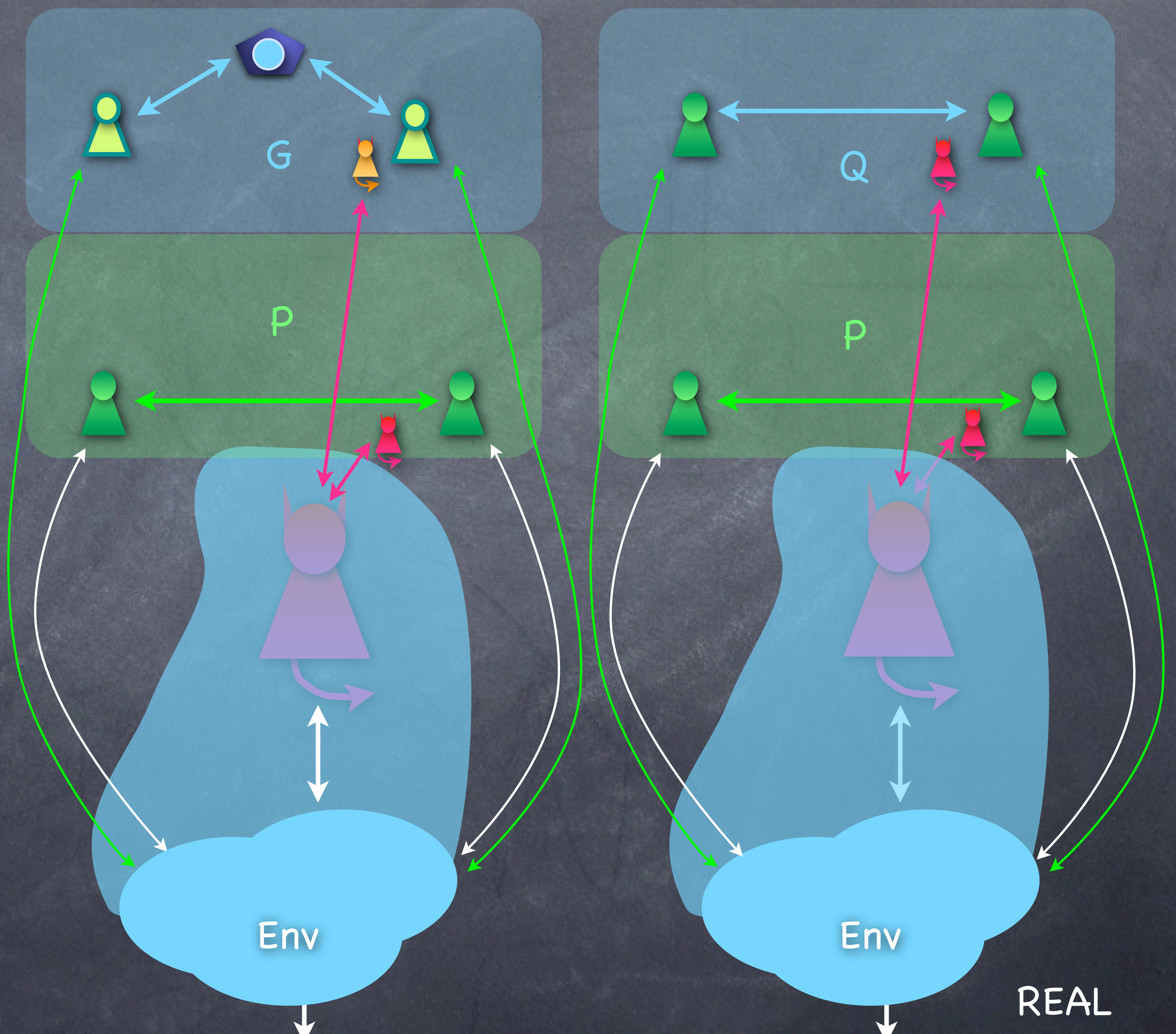
Proving the UC theorem

- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols
- Now consider new environment s.t. only Q (and its adversary) is outside it
- Use "Q is as secure as G" to get a new world with G and a new adversary

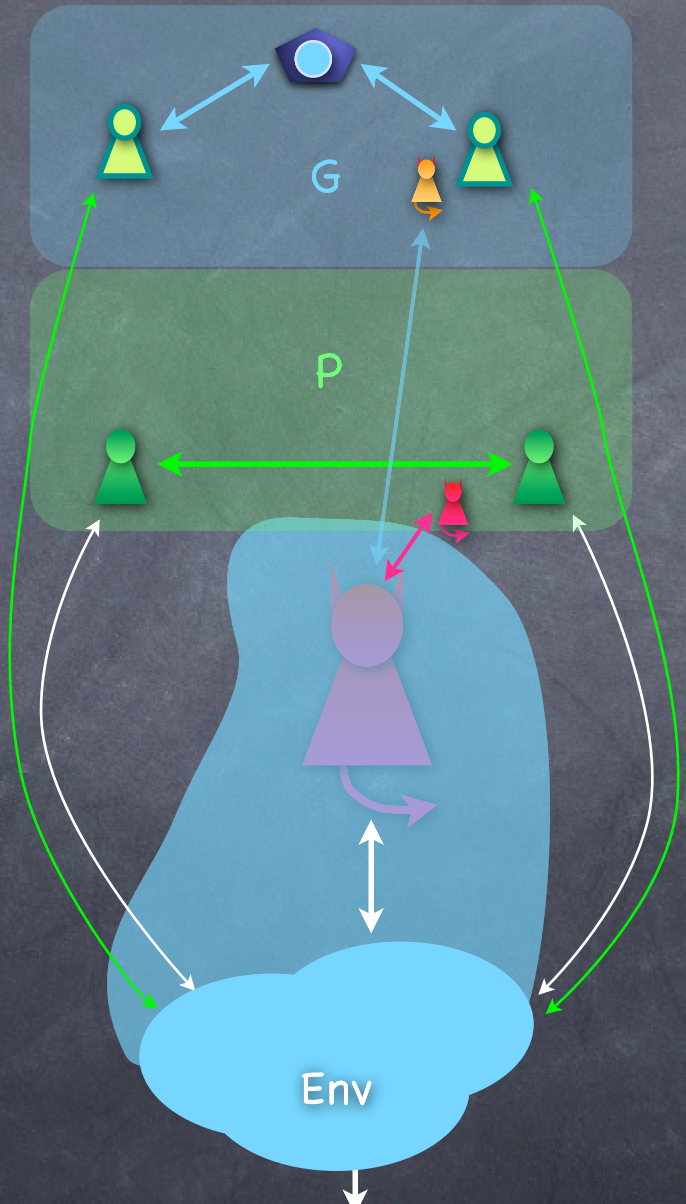


Proving the UC theorem

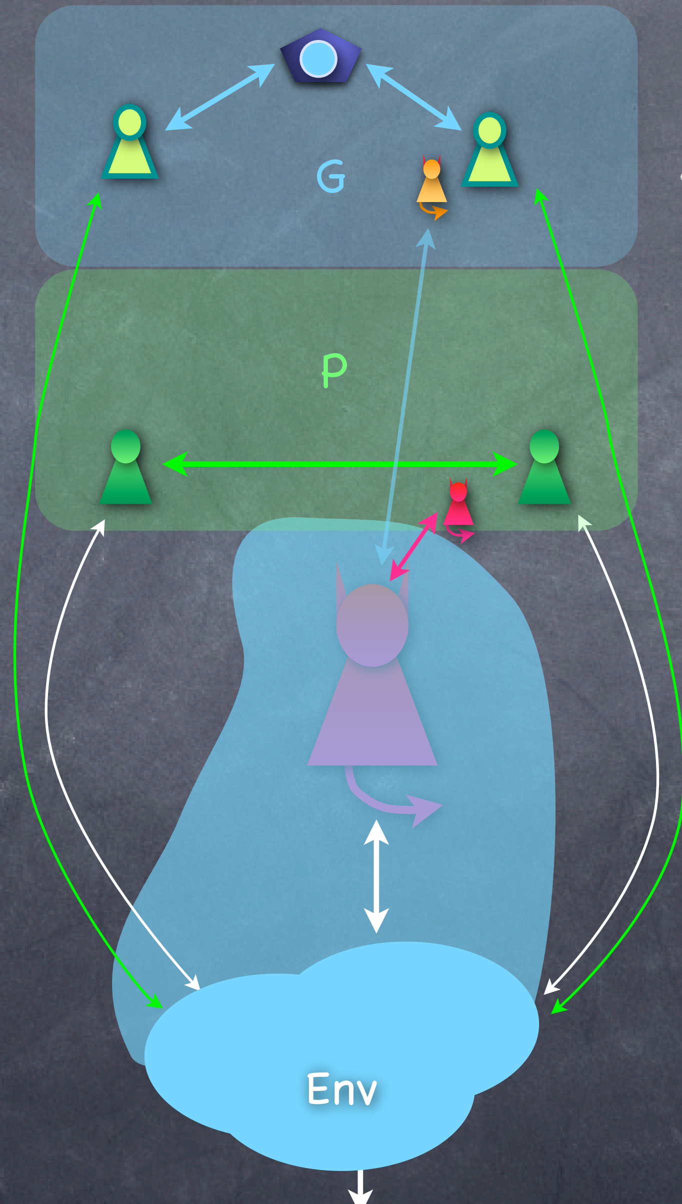
- Consider environment which runs the adversary internally, and depends on "dummy adversaries" to interface with the protocols
- Now consider new environment s.t. only Q (and its adversary) is outside it
- Use "Q is as secure as G" to get a new world with G and a new adversary



Proving the UC theorem

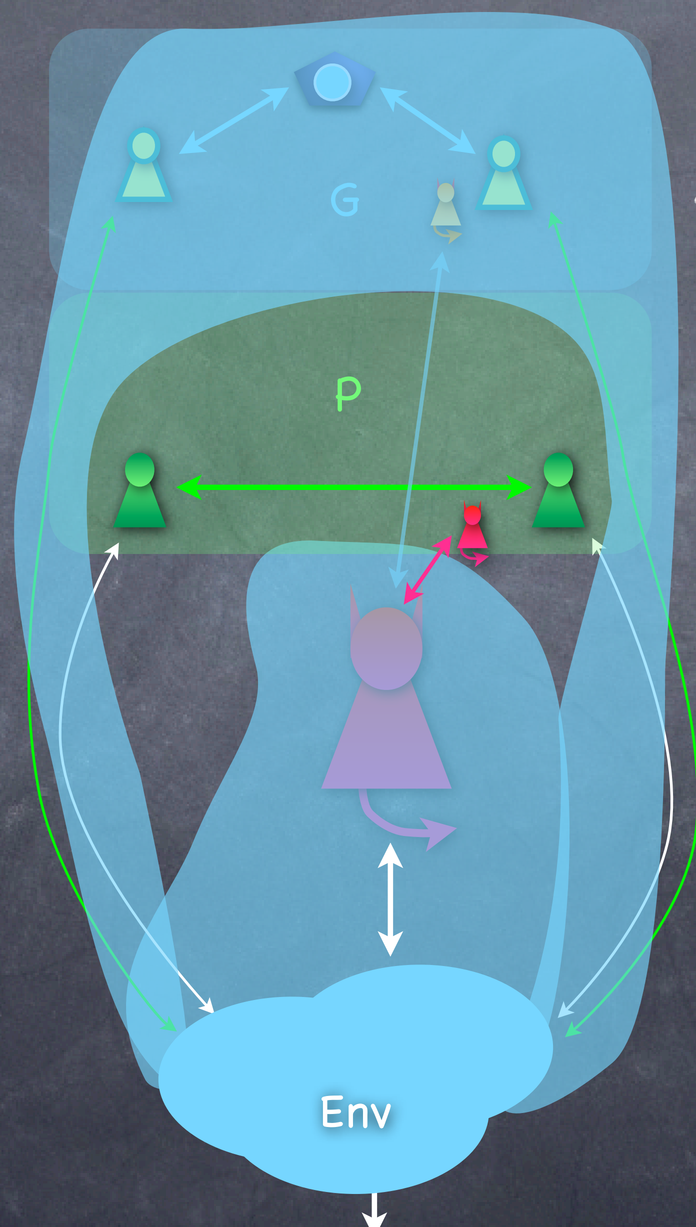


Proving the UC theorem



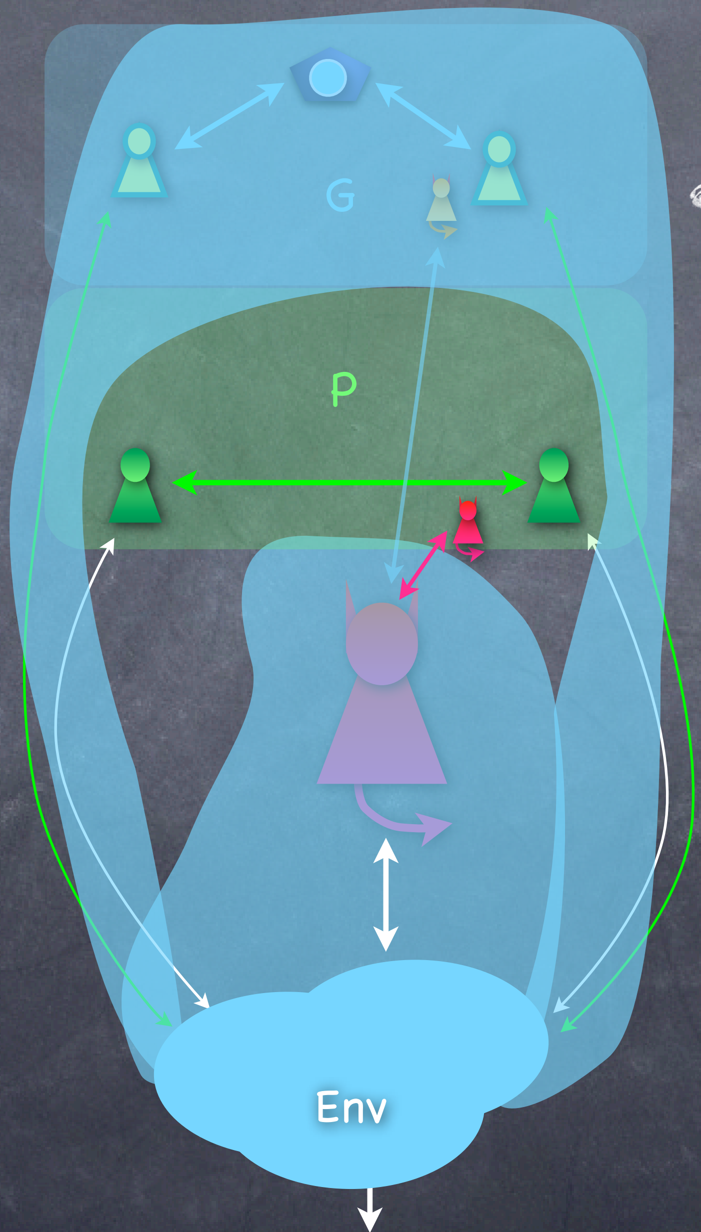
- Now consider new environment s.t. only P (and adversary) is outside it

Proving the UC theorem



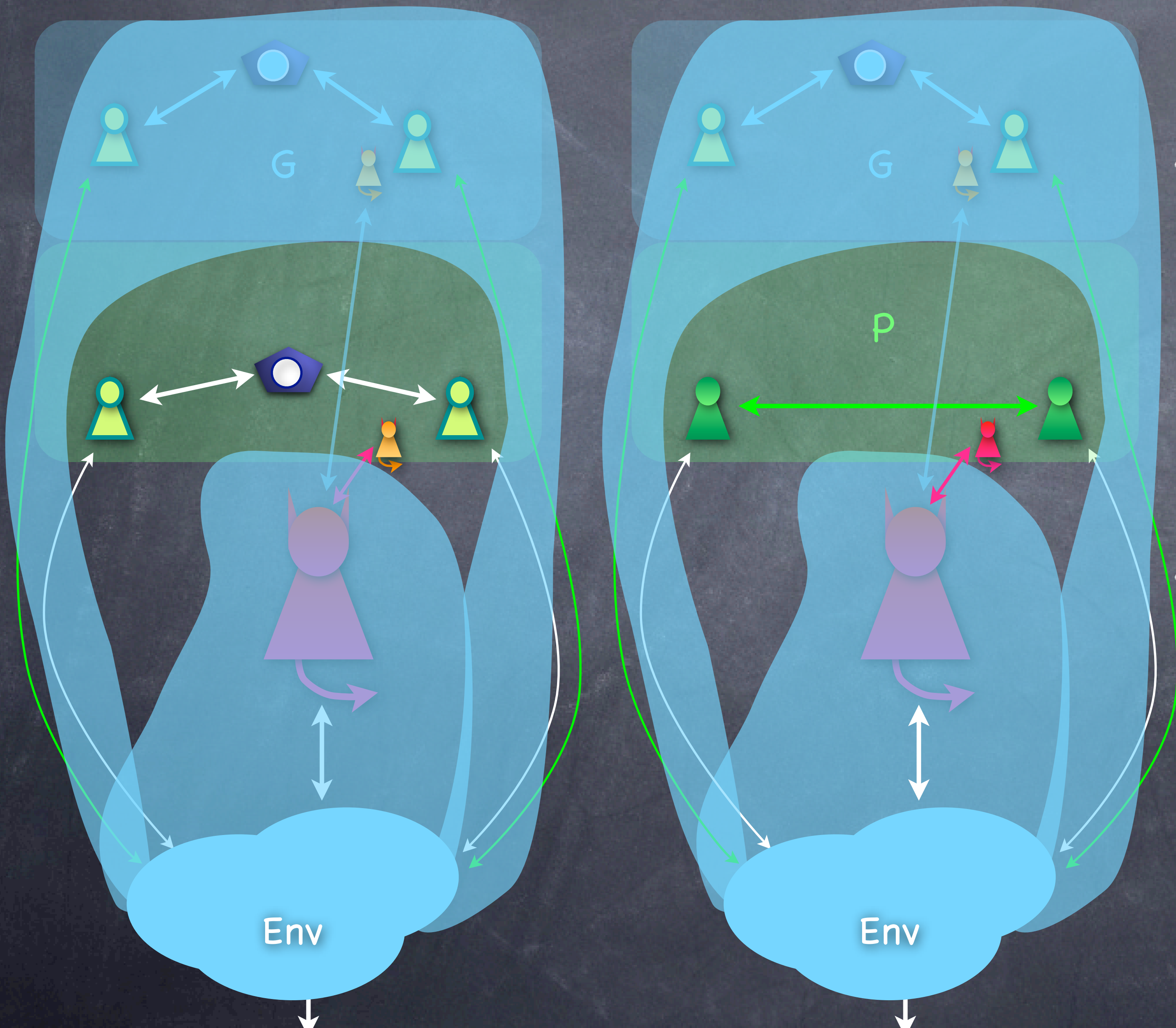
- Now consider new environment s.t. only P (and adversary) is outside it

Proving the UC theorem



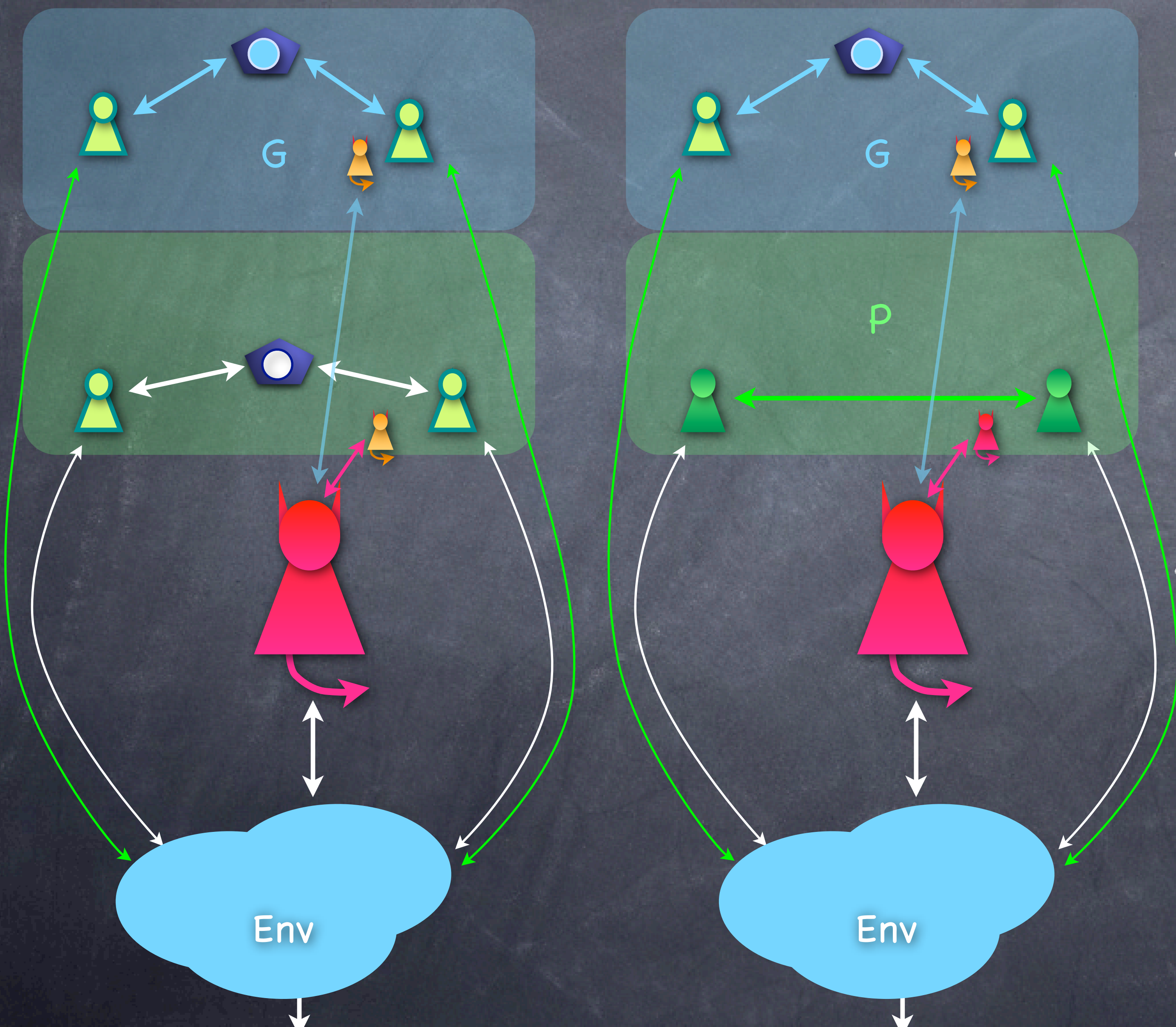
- Now consider new environment s.t. only P (and adversary) is outside it
- Note: G and simulator for Q/G are inside the new environment

Proving the UC theorem



- Now consider new environment s.t. only P (and adversary) is outside it
  - Note: G and simulator for Q/G are inside the new environment
- Use "P is as secure as F" to get a new world with F and a new adversary

Proving the UC theorem



- Now consider new environment s.t. only P (and adversary) is outside it
  - Note: G and simulator for Q/G are inside the new environment
- Use "P is as secure as F" to get a new world with F and a new adversary