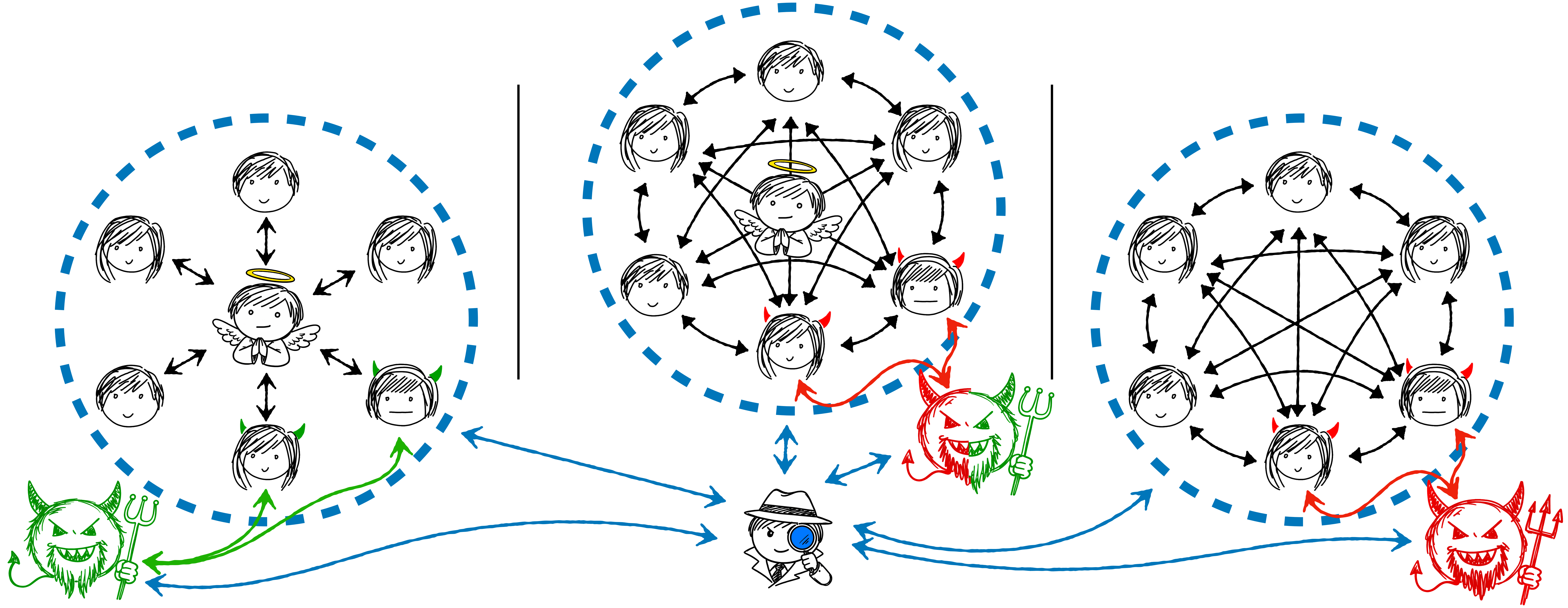
CS4501 Cryptographic Protocols
Lecture 10: GRR Example, Composition in a Perfect World

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Recap: The $\mathcal{F}_{\mathsf{mul}}$-*Hybrid* Model

- *Hybrid Models* are models of the world that *blend* aspects of the Ideal and Real world models. The parties run a non-trivial protocol, like in the real world, but one or more functionalities also participate in the protocol. This is another world on which the Distinguisher can run experiments!



Recap: The $\mathcal{F}_{\mathsf{mul}}$-*Hybrid* Model

- *Hybrid Models* are models of the world that *blend* aspects of the Ideal and Real world models. The parties run a non-trivial protocol, like in the real world, but one or more functionalities also participate in the protocol. This is another world on which the Distinguisher can run experiments!
- The usefulness of hybrid models comes from the *transitivity* of indistinguishability: if no algorithm can distinguish the real world from a hybrid world, and no algorithm can distinguish the same hybrid world from the ideal world, then the real and ideal worlds must also be indistinguishable.
- Looking at it another way, hybrid models allow us to break down protocols *and their security proofs* into self-contained, reusable components rather than constructing and proving them monolithically.

Recap: How to *Use* the $\mathscr{F}_{\text{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathscr{F}_{\text{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathscr{F}_{\text{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathscr{F}_{\text{mul}}$.
4. Prove that security is maintained if we *replace* $\mathscr{F}_{\text{mul}}$ with π_{mul}.

Lemma 1: Let $p > n > t$, let $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be any *randomized* n-ary function, and let C be a circuit that computes f. Assuming synchronicity and secure channels, $\pi_{\text{BGW}}(n, t, p, C)$ perfectly realizes $\mathscr{F}_{\text{SFE}}(n, f, \mathbb{F}_p, \ldots, \mathbb{F}_p)$ in the presence of a semi-honest $\mathscr{A}$ that statically corrupts up to $n - 1$ parties in the $\mathscr{F}_{\text{mul}}$-hybrid model.

Proved in Lecture 8!

Recap: How to *Use* the $\mathcal{F}_{\mathsf{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathcal{F}_{\mathsf{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathcal{F}_{\mathsf{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\mathsf{mul}}$.
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\mathsf{mul}}$ with π_{mul}.

Lemma 2: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels there exists an n-party protocol that perfectly realizes $\mathcal{F}_{\mathsf{mul}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

Lecture 8: why this is nontrivial. Lecture 9: two protocols.

Recap: How to *Use* the $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul}. **Today!**

Perfect MPC Composition Theorem: Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that perfectly realizes $\mathcal{F}_{\text{inner}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

Recap: How to *Use* the $\mathcal{F}_{\mathsf{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathcal{F}_{\mathsf{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathcal{F}_{\mathsf{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\mathsf{mul}}$.
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\mathsf{mul}}$ with π_{mul}.

Putting it together...

Perfect MPC Feasibility Theorem: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels there exists an n-party protocol that perfectly realizes $\mathcal{F}_{\mathsf{SFE}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

3b. Recap: The GRR Protocol

Motivation

Multiplication: we have $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$, and we need $g \leftarrow \mathscr{P}_{p,t,w \cdot w'}$.

We need 2 things from g:

1. To achieve privacy, it must be uniform among polynomials of degree t.
2. To achieve correctness, it must have degree t and $g(0) = w \cdot w'$.

What we have from $\hat{g} = f \cdot f'$:

3. Although $\hat{g}$ isn't uniform and it has the wrong degree, it *does* encode the right value (i.e. $\hat{g}(0) = w \cdot w'$). If we define the n Lagrange bases $\ell_i(x) \forall i \in [n]$ with respect to the x-coordinates $[n]$, then we have

$$w \cdot w' = \boxed{\sum_{i \in [n]} \ell_i(0) \cdot \hat{g}(i)}$$

Observation: looking at it another way, we are trying to find a degree-t Shamir sharing of *this* equation. *Is there an easy way using tools we know?*

Shares of Shares?

Suppose we wanted to securely compute $y = \sum_{i \in [n]} c_i \cdot x_i$ where each party P_i for $i \in [n]$ has input $x_i \in \mathbb{F}_p$ and the c_i are public constants. *How would we do it?*

Using the pieces of the BGW protocol:

1. Every P_i samples $\langle x_i \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$ and sends $\langle x_i \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$
2. Every P_i computes $\langle y \rangle_i = \sum_{j \in [n]} c_j \cdot \langle x_j \rangle_i$. This ls linear, and thus local.
3. The parties reconstruct y from $\langle y \rangle$.

Notice:

- The security of this protocol has nothing to do with the distribution of the x_i values. They can be *completely arbitrary*, and yet $\langle y \rangle$ is a *uniform* sharing.
- If we let $c_i = \ell_i(0)$ and $x_i = \hat{g}(i)$ then $\langle y \rangle$ is a *uniform* degree-t sharing of $\hat{g}(0)$!

Putting it Together: $\pi_{\mathsf{GRR}}(n, t, p)$.

Inputs: Every P_i for $i \in [n]$ has input $\langle w \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w}$ and input $\langle w' \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w'}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w \rangle_i \cdot \langle w' \rangle_i$. Note that $\hat{g} \in \mathscr{P}_{p,2t,w\cdot w'}$.
2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$.
3. Every P_i computes $\langle w \cdot w' \rangle_i = \sum_{j\in[n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$.

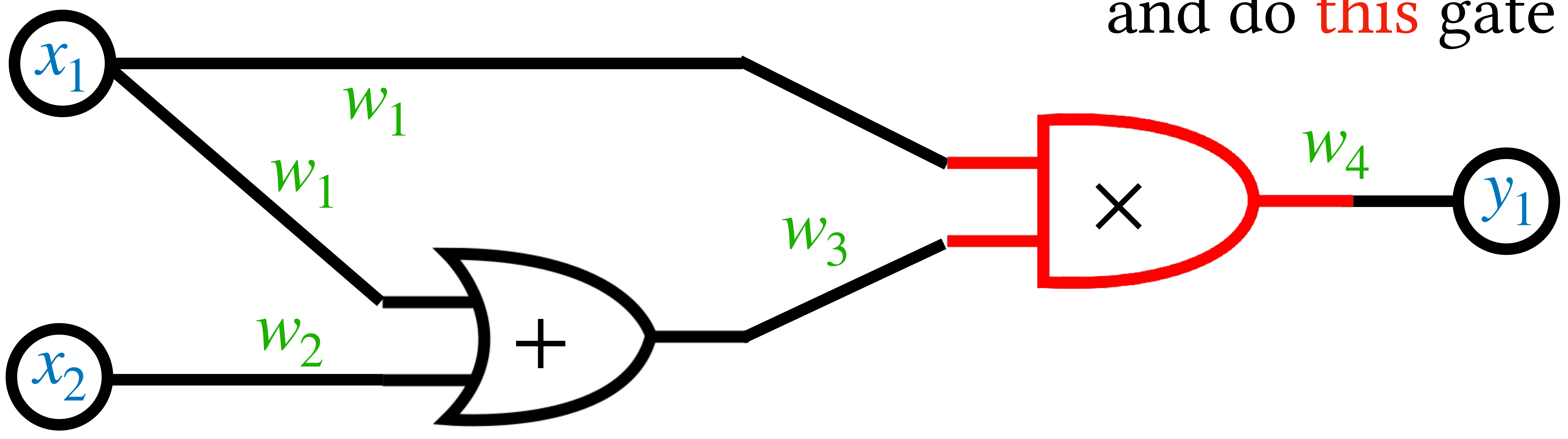
Outputs: Each P_i ends with $\langle w \cdot w' \rangle_i$.

Another Look at $\pi_{\mathsf{GRR}}(n, t, p)$.

	Local Values/Ops	Values Transmitted By						
		P_1	$\dots$	P_i	$\dots$	P_n		
Polynomial	$f(x) \cdot f'(x) =: \hat{g}(x)$							$g(x)$
Encoded Value	$w \cdot w' =: w \cdot w'$	$\ell_1(0) \cdot \hat{g}(1)$		$\cdots + \ell_i(0) \cdot \hat{g}(i)$		$\cdots + \ell_n(0) \cdot \hat{g}(n)$	$=$	$w \cdot w'$
Share of P_1	$\langle w \rangle_1 \cdot \langle w' \rangle_1 =: \hat{g}(1)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_1$		$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_1$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_1$	$=$	$\langle w \cdot w' \rangle_1$
$\vdots$	$\vdots \quad \vdots \quad \vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
Share of P_i	$\langle w \rangle_i \cdot \langle w' \rangle_i =: \hat{g}(i)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_i$		$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_i$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_i$	$=$	$\langle w \cdot w' \rangle_i$
$\vdots$	$\vdots \quad \vdots \quad \vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
Share of P_n	$\langle w \rangle_n \cdot \langle w' \rangle_n =: \hat{g}(n)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_n$		$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_n$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_n$	$=$	$\langle w \cdot w' \rangle_n$

Another Worked Example

- $t = 1$, $n = 3$, $p = 5$, so we have 3 parties (P_1, P_2, P_3), with at least two honest, and they work over $\mathbb{F}_5$. Once again, P_1 will be corrupted.
- Remember our circuit from earlier? We will use the values we assigned to the wires and do this gate as our example.



- Remember that we had $w_1 = 2$, $w_3 = 1$, $w_4 = 2$, $\langle w_1 \rangle = (0,3,1)$, $\langle w_3 \rangle = (0,4,3)$, and $\langle w_4 \rangle = (4,1,3)$.


Sim({1}, {0}, {0}, {4})
sim₁
logic
w₁ = 2
w₃ = 1
view₁
⟨w₁⟩₁ = 0
⟨w₃⟩₁ = 0
view₂
⟨w₁⟩₂ = 3
⟨w₃⟩₂ = 4
view₃
⟨w₁⟩₃ = 1
⟨w₃⟩₃ = 3
π_GRR(3,1,5)

$\mathsf{Sim}(\{1\},\{0\},\{0\},\{4\})$

Inputs: Copy the inputs of the corrupt parties into their views.

sim_1

$\langle w_1\rangle_1 = 0$

$\langle w_3\rangle_1 = 0$

logic

$w_1 = 2$

$w_3 = 1$

view_1

$\langle w_1\rangle_1 = 0$

$\langle w_3\rangle_1 = 0$

view_2

$\langle w_1\rangle_2 = 3$

$\langle w_3\rangle_2 = 4$

view

$\langle w_1\rangle_3 =$

$\langle w_3\rangle_3 =$

sim_1

${}_1\rangle_1 = 0$

${}_3\rangle_1 = 0$

logic

$w_1 = 2$

$w_3 = 1$

$\hat{g}(0) = 2$

view_1

$\langle w_1\rangle_1 = 0$

$\langle w_3\rangle_1 = 0$

$\hat{g}(1) = 0$

view_2

$\langle w_1\rangle_2 = 3$

$\langle w_3\rangle_2 = 4$

$\hat{g}(2) = 2$

view_3

$\langle w_1\rangle_3 = 1$

$\langle w_3\rangle_3 = 3$

$\hat{g}(3) = 3$

$\pi_{\mathsf{GRR}}(3,1,5)$

Inputs: Every P_i for $i \in [n]$ has input $\langle w_1\rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w_1}$ and input $\langle w_3\rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w_3}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w_1\rangle_i \cdot \langle w_3\rangle_i$.

The parties do this deterministic operation locally.

$\mathsf{Sim}(\{1\}, \{0\}, \{0\}, \{4\})$

Inputs: Copy the inputs of the corrupt parties into their views.

1. For every $i \in I$, compute $\hat{g}(i) := \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$ and copy this value into the view of corrupted P_i.

Sim does this deterministic operation.

sim_1	logic	view_1	view_2	view_3
$\langle w_1 \rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$	$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_3 =$
$\langle w_3 \rangle_1 = 0$	$w_3 = 1$	$\langle w_3 \rangle_1 = 0$	$\langle w_3 \rangle_2 = 4$	$\langle w_3 \rangle_3 =$
$\hat{g}(1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) =$

sim$_1$

$\rangle_1 = 0$

$\rangle_1 = 0$

$1) = 0$

logic

$w_1 = 2$

$w_3 = 1$

$\hat{g}(0) = 2$

view$_1$

$\langle w_1 \rangle_1 = 0$

$\langle w_3 \rangle_1 = 0$

$\hat{g}(1) = 0$

$\langle \hat{g}(1) \rangle_1 = 1$

$\langle \hat{g}(1) \rangle_2 = 2$

$\langle \hat{g}(1) \rangle_3 = 3$

view$_2$

$\langle w_1 \rangle_2 = 3$

$\langle w_3 \rangle_2 = 4$

$\hat{g}(2) = 2$

$\langle \hat{g}(1) \rangle_2 = 2$

view$_3$

$\langle w_1 \rangle_3 = 1$

$\langle w_3 \rangle_3 = 3$

$\hat{g}(3) = 3$

$\langle \hat{g}(1) \rangle_3 = 3$

$\pi_{\mathsf{GRR}}(3,1,5)$

Inputs: Every P_i for $i \in [n]$ has input $\langle w_1 \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w_1}$ and input $\langle w_3 \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w_3}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$.

2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \setminus \{i\}$.

P_1 samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + \hat{g}(1)$.

w.p. 1/5, $f_1(x) = 1x + 0 \implies \langle \hat{g}(1) \rangle = (1,2,3)$

sim_1

$\rangle_1 = 0$

$_3\rangle_1 = 0$

$1) = 0$

logic

$w_1 = 2$

$w_3 = 1$

$\hat{g}(0) = 2$

view_1

$\langle w_1 \rangle_1 = 0$

$\langle w_3 \rangle_1 = 0$

$\hat{g}(1) = 0$

$\langle \hat{g}(1) \rangle_1 = 1$

$\langle \hat{g}(1) \rangle_2 = 2$

$\langle \hat{g}(1) \rangle_3 = 3$

$\langle \hat{g}(2) \rangle_1 = 1$

view_2

$\langle w_1 \rangle_2 = 3$

$\langle w_3 \rangle_2 = 4$

$\hat{g}(2) = 2$

$\langle \hat{g}(1) \rangle_2 = 2$

$\langle \hat{g}(2) \rangle_1 = 1$

$\langle \hat{g}(2) \rangle_2 = 0$

$\langle \hat{g}(2) \rangle_3 = 4$

view_3

$\langle w_1 \rangle_3 = 1$

$\langle w_3 \rangle_3 = 3$

$\hat{g}(3) = 3$

$\langle \hat{g}(1) \rangle_3 = 3$

$\langle \hat{g}(2) \rangle_3 = 4$

$\pi_{\mathsf{GRR}}(3,1,5)$

Inputs: Every P_i for $i \in [n]$ has input $\langle w_1 \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w_1}$ and input $\langle w_3 \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w_3}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$.
2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$.

P_2 samples $a_2 \leftarrow \mathbb{F}_5$ and sets $f_2(x) = a_2 \cdot x + \hat{g}(2)$.
w.p. $1/5$, $f_2(x) = 4x + 2 \implies \langle \hat{g}(2) \rangle = (1,0,4)$

sim_1	logic	view_1	view_2	view_3
$\rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$	$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_3 = 1$
$\rangle_1 = 0$	$w_3 = 1$	$\langle w_3 \rangle_1 = 0$	$\langle w_3 \rangle_2 = 4$	$\langle w_3 \rangle_3 = 3$
$1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) = 3$
		$\langle \hat{g}(1) \rangle_1 = 1$		
		$\langle \hat{g}(1) \rangle_2 = 2$	$\langle \hat{g}(1) \rangle_2 = 2$	
		$\langle \hat{g}(1) \rangle_3 = 3$		$\langle \hat{g}(1) \rangle_3 = 3$
		$\langle \hat{g}(2) \rangle_1 = 1$	$\langle \hat{g}(2) \rangle_1 = 1$	
			$\langle \hat{g}(2) \rangle_2 = 0$	
			$\langle \hat{g}(2) \rangle_3 = 4$	$\langle \hat{g}(2) \rangle_3 = 4$
		$\langle \hat{g}(3) \rangle_1 = 4$		$\langle \hat{g}(3) \rangle_1 = 4$
			$\langle \hat{g}(3) \rangle_2 = 0$	$\langle \hat{g}(3) \rangle_2 = 0$
				$\langle \hat{g}(3) \rangle_3 = 1$

$\pi_{\mathsf{GRR}}(3,1,5)$

Inputs: Every P_i for $i \in [n]$ has input $\langle w_1 \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w_1}$ and input $\langle w_3 \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w_3}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$.

2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$.

P_3 samples $a_2 \leftarrow \mathbb{F}_5$ and sets $f_3(x) = a_3 \cdot x + \hat{g}(3)$.
w.p. $1/5$, $f_3(x) = 1x + 3 \implies \langle \hat{g}(3) \rangle = (4,0,1)$

$\mathsf{Sim}(\{1\}, \{0\}, \{0\}, \{4\})$

Inputs: Copy the inputs of the corrupt parties into their views.

1. For every $i \in I$, compute $\hat{g}(i) := \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$ and copy this value into the view of corrupted P_i.
2. For $i \in I$, sample $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$, and for $h \in [n] \backslash I$, sample $\langle \hat{g}(h) \rangle_i \leftarrow \mathbb{F}_p$, subject to the *restriction* that $\langle w_4 \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$

Sim samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + x_1$.
w.p. 1/5, $f_1(x) = 1x + 0 \implies \langle \hat{g}(1) \rangle = (1,2,3)$

Sim samples $\langle \hat{g}(2) \rangle_1 \leftarrow \mathbb{F}_5$. w.p. 1/5, $\langle \hat{g}(2) \rangle_1 = 1$

sim_1	logic	view_1	view_2	view_3
$\langle w_1 \rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$	$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_3 =$
$\langle w_3 \rangle_1 = 0$	$w_3 = 1$	$\langle w_3 \rangle_1 = 0$	$\langle w_3 \rangle_2 = 4$	$\langle w_3 \rangle_3 =$
$\hat{g}(1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) =$
$\langle \hat{g}(1) \rangle_1 = 1$		$\langle \hat{g}(1) \rangle_1 = 1$		
$\langle \hat{g}(1) \rangle_2 = 2$		$\langle \hat{g}(1) \rangle_2 = 2$	$\langle \hat{g}(1) \rangle_2 = 2$	
$\langle \hat{g}(1) \rangle_3 = 3$		$\langle \hat{g}(1) \rangle_3 = 3$		$\langle \hat{g}(1) \rangle_3 =$
$\langle \hat{g}(2) \rangle_1 = 1$		$\langle \hat{g}(2) \rangle_1 = 1$	$\langle \hat{g}(2) \rangle_1 = 1$	
			$\langle \hat{g}(2) \rangle_2 = 0$	
			$\langle \hat{g}(2) \rangle_3 = 4$	$\langle \hat{g}(2) \rangle_3 =$
		$\langle \hat{g}(3) \rangle_1 = 4$		$\langle \hat{g}(3) \rangle_1 =$
			$\langle \hat{g}(3) \rangle_2 = 0$	$\langle \hat{g}(3) \rangle_2 =$
				$\langle \hat{g}(3) \rangle_3 =$

1. For every $i \in I$, compute $\hat{g}(i) := \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$ and copy this value into the view of corrupted P_i.
2. For $i \in I$, sample $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$, and for $h \in [n] \backslash I$, sample $\langle \hat{g}(h) \rangle_i \leftarrow \mathbb{F}_p$, subject to the *restriction* that $\langle w_4 \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$

Sim samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + x_1$.
w.p. 1/5, $f_1(x) = 1x + 0 \implies \langle \hat{g}(1) \rangle = (1,2,3)$

Sim samples $\langle \hat{g}(2) \rangle_1 \leftarrow \mathbb{F}_5$. w.p. 1/5, $\langle \hat{g}(2) \rangle_1 = 1$

We relative to x-coords (0,1,2), we have $\ell_0(3) = 1$, $\ell_1(3) = 2$, $\ell_2(3) = 3$. Recall that $\langle w_4 \rangle_i = \langle \hat{g}(0) \rangle_i$. Interpolating to find the value of the polynomial $\langle \hat{g}(x) \rangle_1$ at $x = 3$ yields:

$$\langle \hat{g}(3) \rangle_1 = \sum_{j \in [0,2]} \ell_j(3) \cdot \langle \hat{g}(j) \rangle_1 = 4$$

sim_1	logic	view_1	view_2	view_3
$\langle w_1 \rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$	$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_3 =$
$\langle w_3 \rangle_1 = 0$	$w_3 = 1$	$\langle w_3 \rangle_1 = 0$	$\langle w_3 \rangle_2 = 4$	$\langle w_3 \rangle_3 =$
$\hat{g}(1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) =$
$\langle \hat{g}(1) \rangle_1 = 1$		$\langle \hat{g}(1) \rangle_1 = 1$		
$\langle \hat{g}(1) \rangle_2 = 2$		$\langle \hat{g}(1) \rangle_2 = 2$	$\langle \hat{g}(1) \rangle_2 = 2$	
$\langle \hat{g}(1) \rangle_3 = 3$		$\langle \hat{g}(1) \rangle_3 = 3$		$\langle \hat{g}(1) \rangle_3 =$
$\langle \hat{g}(2) \rangle_1 = 1$		$\langle \hat{g}(2) \rangle_1 = 1$	$\langle \hat{g}(2) \rangle_1 = 1$	
			$\langle \hat{g}(2) \rangle_2 = 0$	
			$\langle \hat{g}(2) \rangle_3 = 4$	$\langle \hat{g}(2) \rangle_3 =$
$\langle \hat{g}(3) \rangle_1 = 4$		$\langle \hat{g}(3) \rangle_1 = 4$		$\langle \hat{g}(3) \rangle_1 =$
			$\langle \hat{g}(3) \rangle_2 = 0$	$\langle \hat{g}(3) \rangle_2 =$
				$\langle \hat{g}(3) \rangle_3 =$

1. For every $i \in I$, compute $\hat{g}(i) := \langle w_1 \rangle_i \cdot \langle w_3 \rangle_i$ and copy this value into the view of corrupted P_i.
2. For $i \in I$, sample $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$, and for $h \in [n] \backslash I$, sample $\langle \hat{g}(h) \rangle_i \leftarrow \mathbb{F}_p$, subject to the *restriction* that $\langle w_4 \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$
3. Copy the outputs of the corrupt parties into their views.

Sim samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + x_1$.
w.p. 1/5, $f_1(x) = 1x + 0 \implies \langle \hat{g}(1) \rangle = (1,2,3)$

Sim samples $\langle \hat{g}(2) \rangle_1 \leftarrow \mathbb{F}_5$. w.p. 1/5, $\langle \hat{g}(2) \rangle_1 = 1$

We relative to x-coords (0,1,2), we have $\ell_0(3) = 1$, $\ell_1(3) = 2$, $\ell_2(3) = 3$. Recall that $\langle w_4 \rangle_i = \langle \hat{g}(0) \rangle_i$. Interpolating to find the value of the polynomial $\langle \hat{g}(x) \rangle_1$ at $x = 3$ yields:

$$\langle \hat{g}(3) \rangle_1 = \sum_{j \in [0,2]} \ell_j(3) \cdot \langle \hat{g}(j) \rangle_1 = 4$$

sim_1

$\langle w_1 \rangle_1 = 0$
$\langle w_3 \rangle_1 = 0$
$\hat{g}(1) = 0$
$\langle \hat{g}(1) \rangle_1 = 1$
$\langle \hat{g}(1) \rangle_2 = 2$
$\langle \hat{g}(1) \rangle_3 = 3$
$\langle \hat{g}(2) \rangle_1 = 1$
$\langle \hat{g}(3) \rangle_1 = 4$
$\langle w_4 \rangle_1 = 4$

logic

$w_1 = 2$
$w_3 = 1$
$\hat{g}(0) = 2$

view_1

$\langle w_1 \rangle_1 = 0$
$\langle w_3 \rangle_1 = 0$
$\hat{g}(1) = 0$
$\langle \hat{g}(1) \rangle_1 = 1$
$\langle \hat{g}(1) \rangle_2 = 2$
$\langle \hat{g}(1) \rangle_3 = 3$
$\langle \hat{g}(2) \rangle_1 = 1$
$\langle \hat{g}(3) \rangle_1 = 4$

view_2

$\langle w_1 \rangle_2 = 3$
$\langle w_3 \rangle_2 = 4$
$\hat{g}(2) = 2$
$\langle \hat{g}(1) \rangle_2 = 2$
$\langle \hat{g}(2) \rangle_1 = 1$
$\langle \hat{g}(2) \rangle_2 = 0$
$\langle \hat{g}(2) \rangle_3 = 4$
$\langle \hat{g}(3) \rangle_2 = 0$

view_3

$\langle w_1 \rangle_3 =$
$\langle w_3 \rangle_3 =$
$\hat{g}(3) =$
$\langle \hat{g}(1) \rangle_3 =$
$\langle \hat{g}(2) \rangle_3 =$
$\langle \hat{g}(3) \rangle_1 =$
$\langle \hat{g}(3) \rangle_2 =$
$\langle \hat{g}(3) \rangle_3 =$

sim_1	logic	view_1	view_2	view_3
$\rangle_1 = 0$	$w_1 = 2$	$\langle w_1\rangle_1 = 0$	$\langle w_1\rangle_2 = 3$	$\langle w_1\rangle_3 = 1$
$\rangle_1 = 0$	$w_3 = 1$	$\langle w_3\rangle_1 = 0$	$\langle w_3\rangle_2 = 4$	$\langle w_3\rangle_3 = 3$
$1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) = 3$
$))_1 = 1$		$\langle\hat{g}(1)\rangle_1 = 1$		
$))_2 = 2$		$\langle\hat{g}(1)\rangle_2 = 2$	$\langle\hat{g}(1)\rangle_2 = 2$	
$))_3 = 3$		$\langle\hat{g}(1)\rangle_3 = 3$		$\langle\hat{g}(1)\rangle_3 = 3$
$2))_1 = 1$		$\langle\hat{g}(2)\rangle_1 = 1$	$\langle\hat{g}(2)\rangle_1 = 1$	
			$\langle\hat{g}(2)\rangle_2 = 0$	
			$\langle\hat{g}(2)\rangle_3 = 4$	$\langle\hat{g}(2)\rangle_3 = 4$
$3))_1 = 4$		$\langle\hat{g}(3)\rangle_1 = 4$		$\langle\hat{g}(3)\rangle_1 = 4$
			$\langle\hat{g}(3)\rangle_2 = 0$	$\langle\hat{g}(3)\rangle_2 = 0$
				$\langle\hat{g}(3)\rangle_3 = 1$
$4\rangle_1 = 4$	$w_4 = 2$	$\langle w_4\rangle_1 = 4$	$\langle w_4\rangle_2 = 1$	$\langle w_4\rangle_3 = 3$

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w_1\rangle_i \cdot \langle w_3\rangle_i$.

2. Every P_i samples $\langle\hat{g}(i)\rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle\hat{g}(i)\rangle_j$ to P_j for $j \in [n]\backslash\{i\}$.

3. Every P_i computes $\langle w_4\rangle_i = \sum_{j\in[n]} \ell_j(0) \cdot \langle\hat{g}(j)\rangle_i$.

Outputs: Each P_i ends with $\langle w_4\rangle_i$.

We relative to x-coords (1,2,3), we have $\ell_1(0) = 3$, $\ell_2(0) = 2$, $\ell_3(0) = 1$.

P_1 interpolates to find the value of the polynomial $\langle\hat{g}(x)\rangle_1$ at $x = 0$ yielding:

$$\langle w_4\rangle_1 = \langle\hat{g}(0)\rangle_1 = \sum_{j\in[n]} \ell_j(0) \cdot \langle\hat{g}(j)\rangle_1 = 0$$

P_2 and P_3 do equivalently to find their outputs.

Final Views at Experiment End

sim_1	logic	view_1	view_2	view_3
$\langle w_1\rangle_1 = 0$	$w_1 = 2$	$\langle w_1\rangle_1 = 0$	$\langle w_1\rangle_2 = 3$	$\langle w_1\rangle_3 = 1$
$\langle w_3\rangle_1 = 0$	$w_3 = 1$	$\langle w_3\rangle_1 = 0$	$\langle w_3\rangle_2 = 4$	$\langle w_3\rangle_3 = 3$
$\hat{g}(1) = 0$	$\hat{g}(0) = 2$	$\hat{g}(1) = 0$	$\hat{g}(2) = 2$	$\hat{g}(3) = 3$
$\langle \hat{g}(1)\rangle_1 = 1$		$\langle \hat{g}(1)\rangle_1 = 1$		
$\langle \hat{g}(1)\rangle_2 = 2$		$\langle \hat{g}(1)\rangle_2 = 2$	$\langle \hat{g}(1)\rangle_2 = 2$	
$\langle \hat{g}(1)\rangle_3 = 3$		$\langle \hat{g}(1)\rangle_3 = 3$		$\langle \hat{g}(1)\rangle_3 = 3$
$\langle \hat{g}(2)\rangle_1 = 1$		$\langle \hat{g}(2)\rangle_1 = 1$	$\langle \hat{g}(2)\rangle_1 = 1$	
			$\langle \hat{g}(2)\rangle_2 = 0$	
			$\langle \hat{g}(2)\rangle_3 = 4$	$\langle \hat{g}(2)\rangle_3 = 4$
$\langle \hat{g}(3)\rangle_1 = 4$		$\langle \hat{g}(3)\rangle_1 = 4$		$\langle \hat{g}(3)\rangle_1 = 4$
			$\langle \hat{g}(3)\rangle_2 = 0$	$\langle \hat{g}(3)\rangle_2 = 0$
				$\langle \hat{g}(3)\rangle_3 = 1$
$\langle w_4\rangle_1 = 4$	$w_4 = 2$	$\langle w_4\rangle_1 = 4$	$\langle w_4\rangle_2 = 1$	$\langle w_4\rangle_3 = 3$

4. Composition Theorem

The Theorem (again)

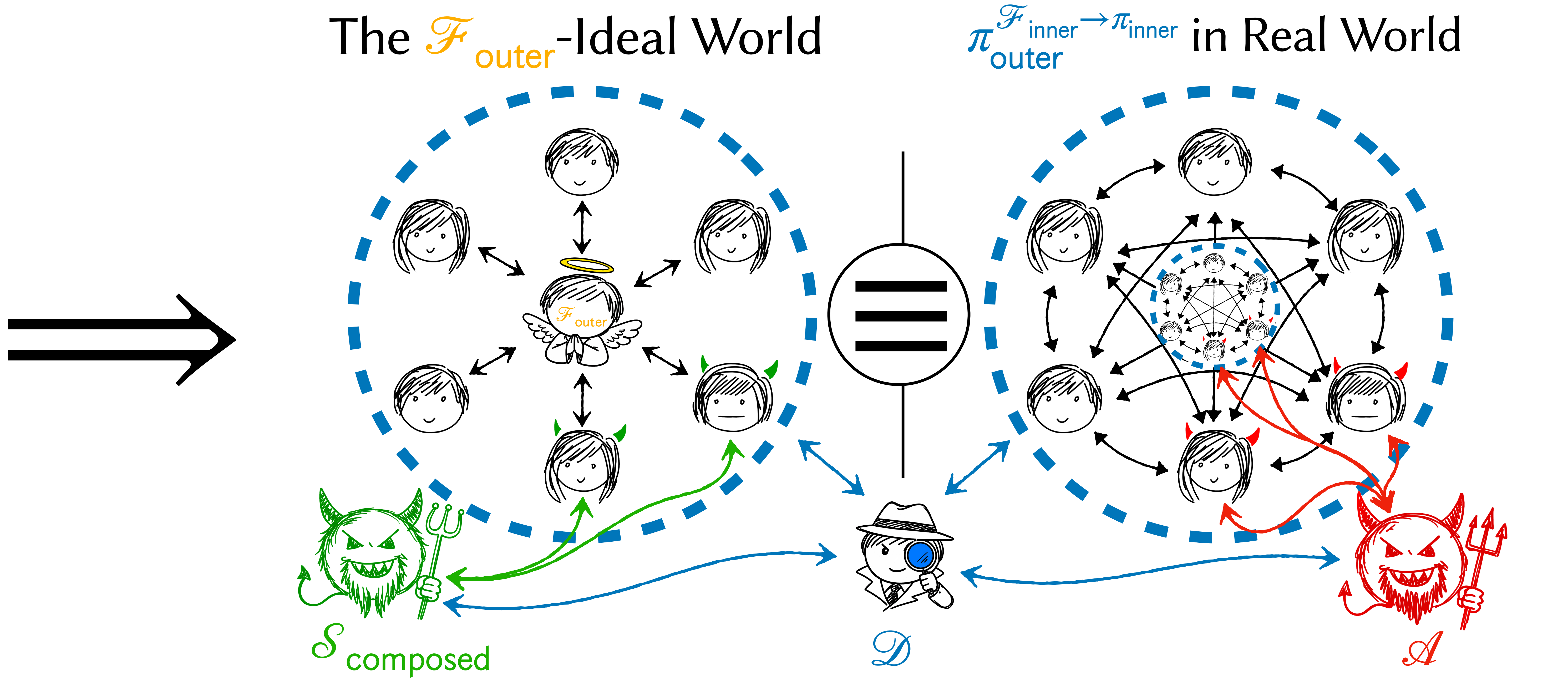
Perfect MPC Composition Theorem: Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that perfectly realizes $\mathcal{F}_{\text{inner}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \rightarrow \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \rightarrow \pi_{\text{inner}}}$ perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

$\Longrightarrow$

The $\mathcal{F}_{outer}$-Ideal World
π_{outer} in $\mathcal{F}_{inner}$-Hybrid Model
The $\mathcal{F}_{inner}$-Ideal World
π_{inner} in Real World
$\equiv$
$\wedge$
$\mathcal{F}_{outer}$
$\mathcal{F}_{inner}$
$\mathcal{S}_{outer}$
$\mathcal{S}_{inner}$
$\mathcal{D}$
$\mathcal{A}$
$\Longrightarrow$
The $\mathcal{F}_{outer}$-Ideal World

The $\mathcal{F}_{\text{outer}}$-Ideal World
π_{outer} in $\mathcal{F}_{\text{inner}}$-Hybrid Model
The $\mathcal{F}_{\text{inner}}$-Ideal World
π_{inner} in Real World
$\mathcal{F}_{\text{outer}}$
$\mathcal{F}_{\text{inner}}$
$\equiv$
$\wedge$
$\mathcal{S}_{\text{outer}}$
$\mathcal{D}$
$\mathcal{A}$
$\mathcal{S}_{\text{inner}}$
$\Longrightarrow$
The $\mathcal{F}_{\text{outer}}$-Ideal World
$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \rightarrow \pi_{\text{inner}}}$ in Real World



- If corrupts P_i in π_{outer} it also corrupts P_i in π_{inner}.
- Need new $\mathcal{S}_{\text{composed}}$ that works for the composed distribution!
- The *only* thing we know is that $\mathcal{S}_{\text{outer}}$ and $\mathcal{S}_{\text{inner}}$ exist. We must use those facts to build $\mathcal{S}_{\text{composed}}$.

Problems we might run into:

- π_{outer} might invoke $\mathscr{F}_{\text{inner}}$ many times. Inputs to $\mathscr{F}_{\text{inner}}$ can be correlated between different invocations, and correlated with other messages in π_{outer}.
- After $\mathscr{F}_{\text{inner}}$ is replaced with π_{inner}, $\mathscr{A}$ can use something it learned in one instance of π_{inner} to help it break security in a *different* instance of π_{inner}. When we proved that π_{inner} realizes $\mathscr{F}_{\text{inner}}$, our proof was "in a vacuum." We considered only one instance running in a universe by itself!
- We do not know anything about how π_{outer} and π_{inner} work. We only know lemmas that π_{outer} realizes $\mathscr{F}_{\text{outer}}$ under $\mathscr{S}_{\text{outer}}$, and π_{inner} that realizes $\mathscr{F}_{\text{inner}}$ under $\mathscr{S}_{\text{inner}}$.
- This means we have to build our proof that $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ realizes $\mathscr{F}_{\text{outer}}$ under $\mathscr{S}_{\text{composed}}$ exclusively out of those two lemmas!

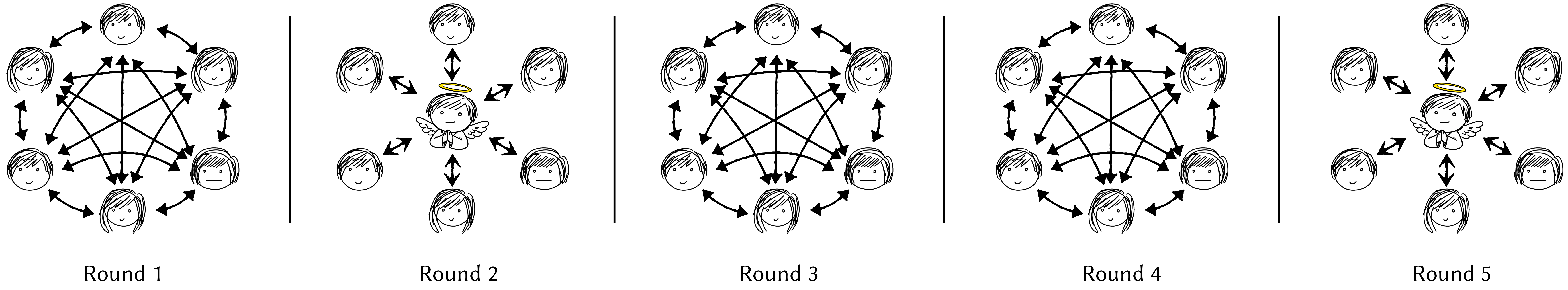
Steps to Achieve Composition

1. Construct the composed protocol $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$.
2. Construct a composed simulator $\mathcal{S}_{\text{composed}}$ for every $\mathcal{A}$ that might attack $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$.
3. Prove that if there is a combination of $(\mathcal{A}, \mathcal{D})$ that can distinguish $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ from $\mathcal{F}_{\text{outer}}$ under $\mathcal{S}_{\text{composed}}$, then at least one of two things must exist:

- a combination of $(\mathcal{A}', \mathcal{D}')$ that can distinguish π_{outer} from $\mathcal{F}_{\text{outer}}$ under $\mathcal{S}_{\text{outer}}$, contradicting the security of π_{outer}.
- a combination of $(\mathcal{A}', \mathcal{D}')$ that can distinguish π_{inner} from $\mathcal{F}_{\text{inner}}$ under $\mathcal{S}_{\text{inner}}$, contradicting the security of π_{inner}.

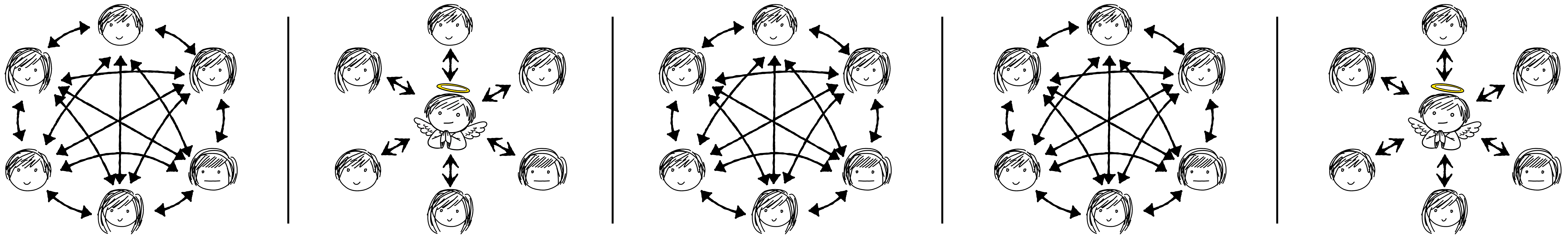
Recall: The $\mathcal{F}_{\text{inner}}$-*Hybrid* Model

- In this class, we assume that in a hybrid model, all communication happens in rounds (i.e. we have synchrony) and that in every round, the parties communicate in one of the following *mutually exclusive* ways:
  - They send messages to one another over secure point-to-point channels, just like in the real world.
  - They send messages to $\mathcal{F}_{\text{inner}}$, and receive a reply, just like in the ideal world.



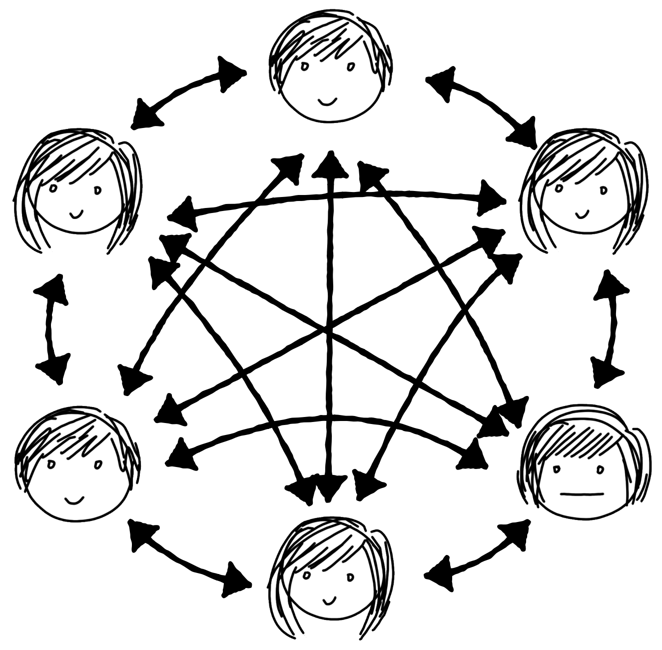
1. Constructing $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$

π_{outer}



Round 1 | Round 2: invoke $\mathcal{F}_{\text{inner}}$ | Round 3 | Round 4 | Round 5: invoke $\mathcal{F}_{\text{inner}}$

$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$

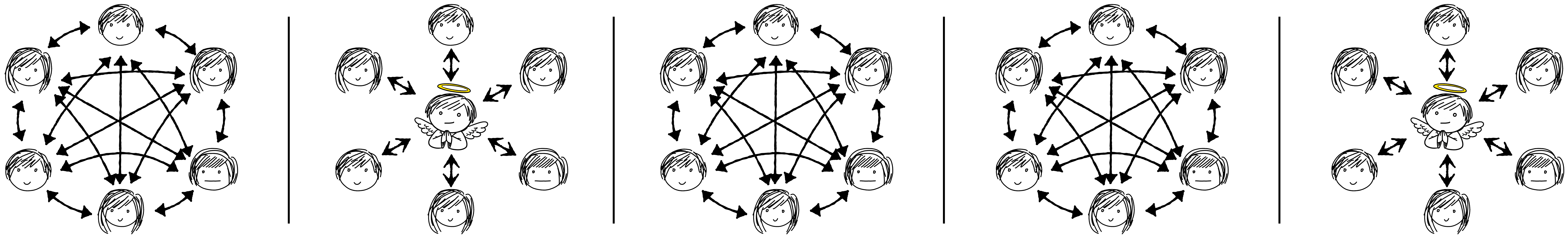


Round 1 of π_{outer}

- At the end of round 1, the parties have inputs to send to $\mathcal{F}_{\text{inner}}$.
- Remember, π_{inner} takes the same inputs as $\mathcal{F}_{\text{inner}}$ and produces the same outputs. We can drop it into place!

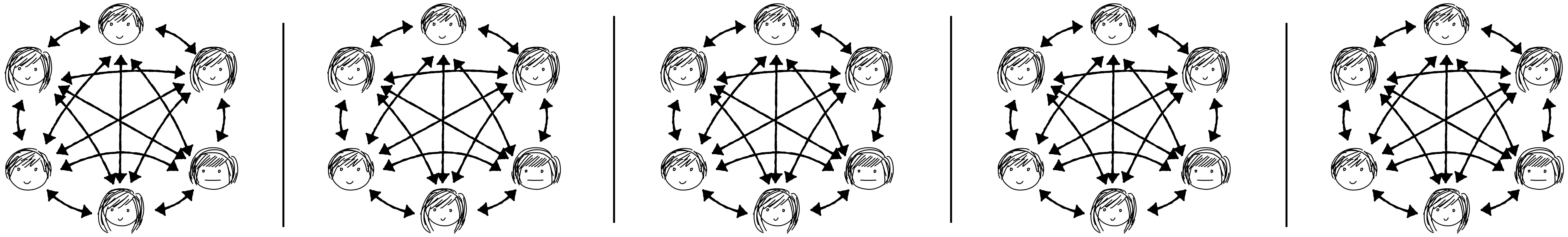
1. Constructing $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$

π_{outer}



Round 1 | Round 2: invoke $\mathcal{F}_{\text{inner}}$ | Round 3 | Round 4 | Round 5: invoke $\mathcal{F}_{\text{inner}}$

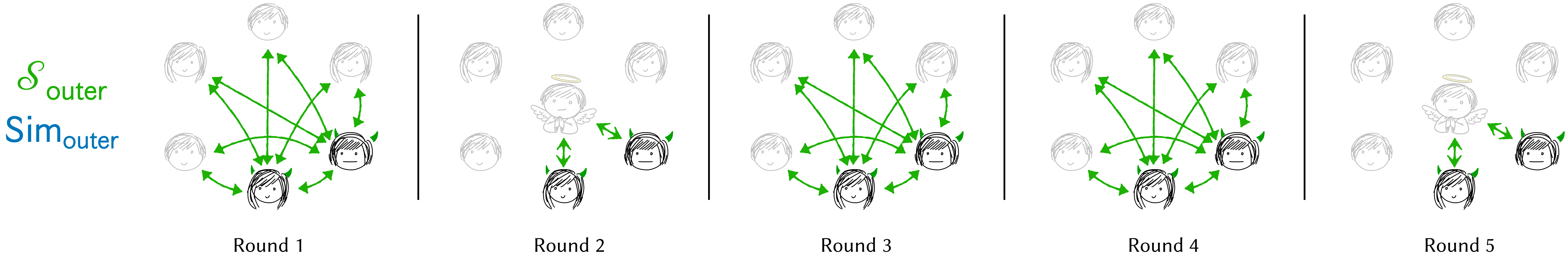
$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$



Round 1 of π_{outer} | Invoke π_{inner} (all rounds) | Round 3 of π_{outer} | Round 4 of π_{outer} | Invoke π_{inner} (all rounds)

A closer look at $\mathcal{S}_{\text{outer}}$

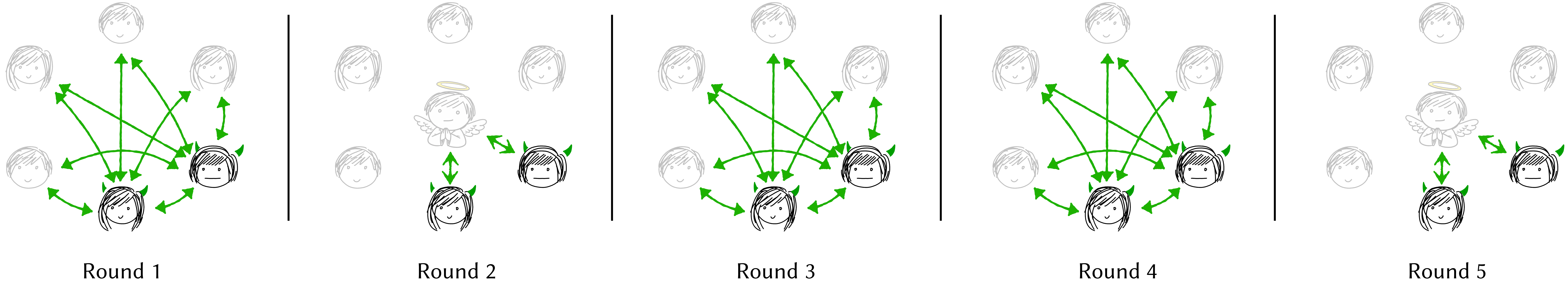
- Remember, when $\mathcal{A}$ is semi-honest, we said it was sufficient for $\mathcal{S}_{\text{outer}}$ to internally run an algorithm $\mathsf{Sim}_{\text{outer}}$ which uses the *inputs* and *outputs* of the corrupt parties in π_{outer} to produce something indistinguishable from the *view* of the corrupt parties in π_{outer}. Note that $\mathcal{S}_{\text{inner}}$ works similarly with $\mathsf{Sim}_{\text{inner}}$.



Constructing $\mathcal{S}_{\text{composed}}$

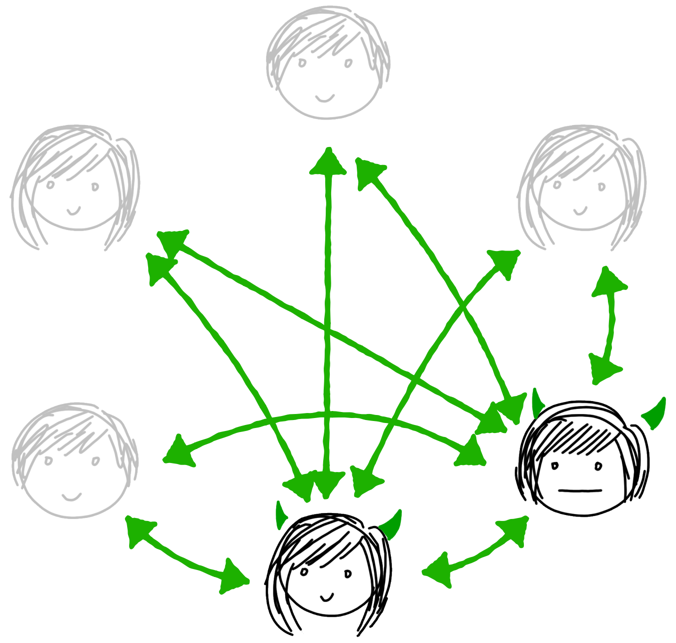
$\mathcal{S}_{\text{outer}}$

$\text{Sim}_{\text{outer}}$



$\mathcal{S}_{\text{composed}}$

$\text{Sim}_{\text{composed}}$



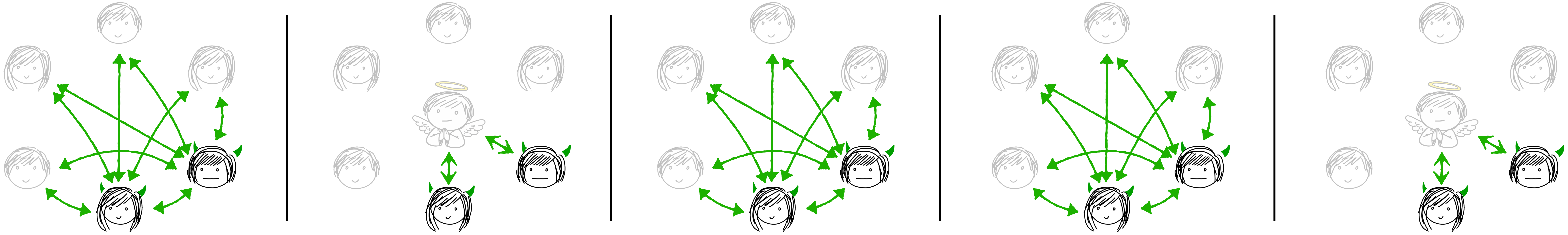
Use $\text{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}

- At the end of round 1, $\mathcal{S}_{\text{composed}}$ knows the inputs that the corrupt parties will send to $\mathcal{F}_{\text{inner}}$. It can use $\text{Sim}_{\text{outer}}$ to get the *outputs* that they will receive *from* $\mathcal{F}_{\text{inner}}$ in round 2.
- However, the protocol $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ uses π_{inner} instead of $\mathcal{F}_{\text{inner}}$.
- Good news: $\text{Sim}_{\text{inner}}$ can simulate π_{inner} using the same IO!

Constructing $\mathcal{S}_{\text{composed}}$

$\mathcal{S}_{\text{outer}}$

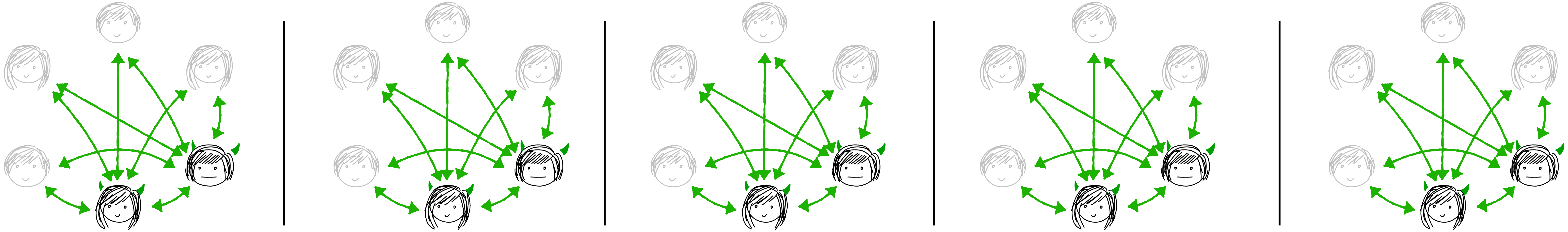
$\mathsf{Sim}_{\text{outer}}$



Round 1 | Round 2 | Round 3 | Round 4 | Round 5

$\mathcal{S}_{\text{composed}}$

$\mathsf{Sim}_{\text{composed}}$



Use $\mathsf{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}

Use $\mathsf{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\mathsf{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Use $\mathsf{Sim}_{\text{outer}}$ to simulate round 3 of π_{outer}

Use $\mathsf{Sim}_{\text{outer}}$ to simulate round 4 of π_{outer}

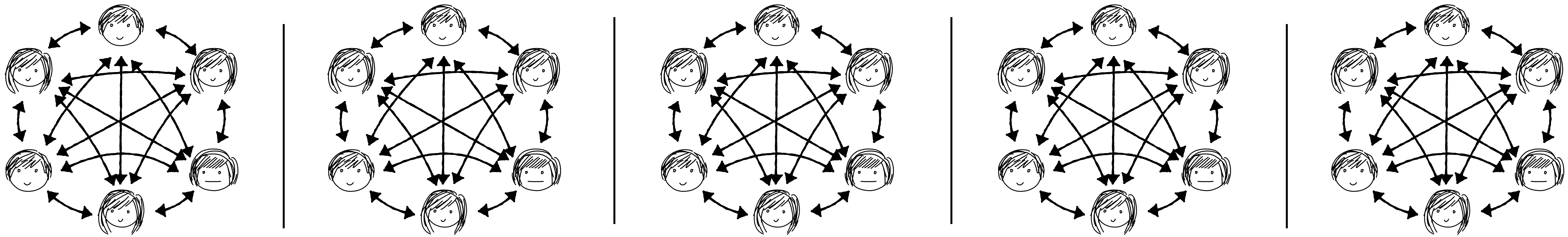
Use $\mathsf{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\mathsf{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Now we have a protocol and simulator...

Real World

$\mathscr{A}$

$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$



Round 1 of π_{outer}

Invoke π_{inner} (all rounds)

Round 3 of π_{outer}

Round 4 of π_{outer}

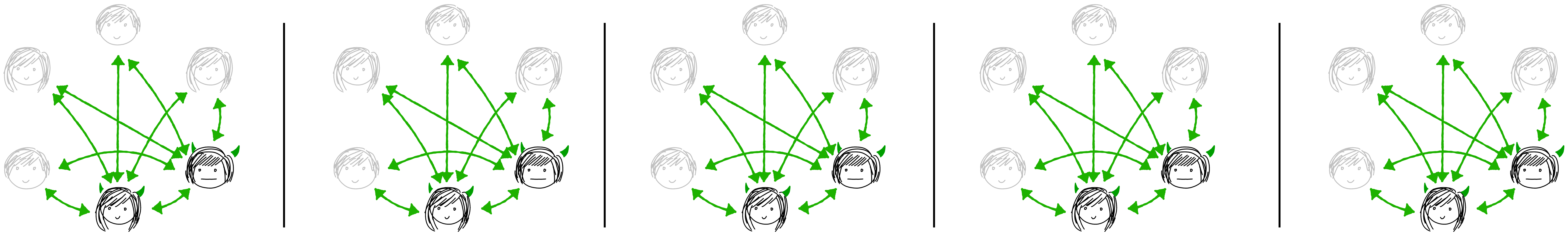
Invoke π_{inner} (all rounds)

Ideal World

$\mathscr{S}_{\text{composed}}$

$\text{Sim}_{\text{composed}}$

$\mathcal{F}_{\text{outer}}$



Use $\text{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}

Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Use $\text{Sim}_{\text{outer}}$ to simulate round 3 of π_{outer}

Use $\text{Sim}_{\text{outer}}$ to simulate round 4 of π_{outer}

Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Now we have a protocol and simulator...

Suppose we had an adversary $\mathscr{A}$ and distinguisher $\mathscr{D}$ that could tell these two distributions apart...

We need to show that the existence of $(\mathscr{A},\mathscr{D})$ contradicts the security of one of the *uncomposed* protocols, π_{outer} and π_{inner}.

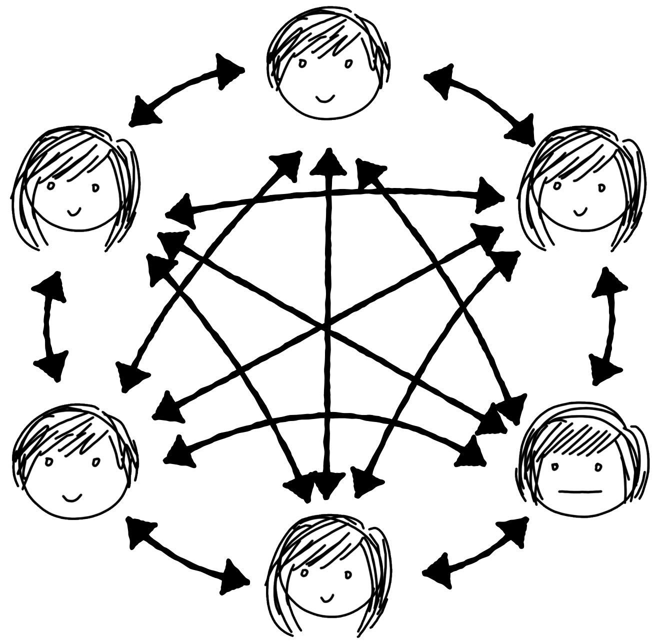
Problem: $\mathscr{A}$ and $\mathscr{D}$ can use information from π_{outer} *and multiple* instances of π_{inner} to distinguish. How can we relate them to a *single* instance of either π_{outer} *or* π_{outer}?

Now we have a protocol and simulator...

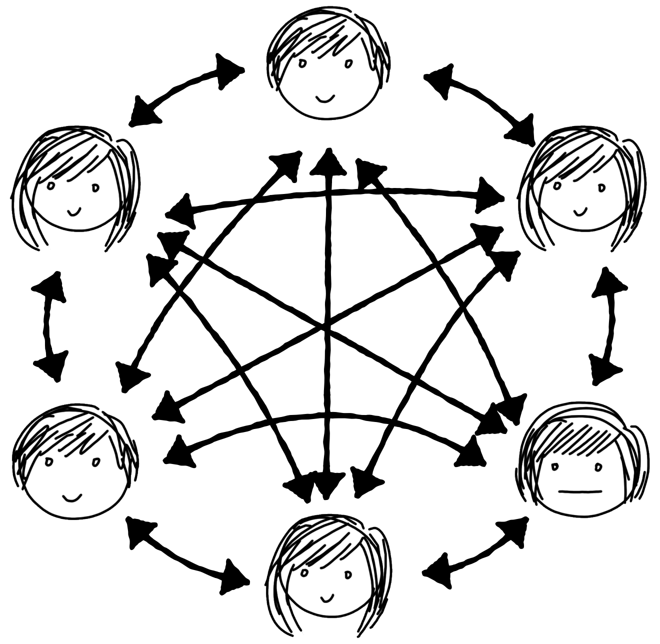
Real World

$\mathcal{A}$

$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}}\to\pi_{\text{inner}}}$



Round 1 of π_{outer}



Invoke π_{inner} (all rounds)

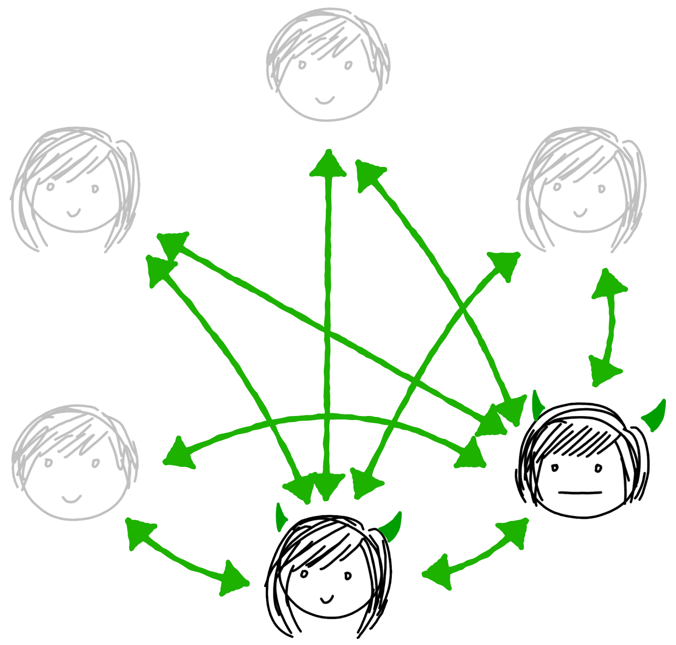
nds)

Ideal World

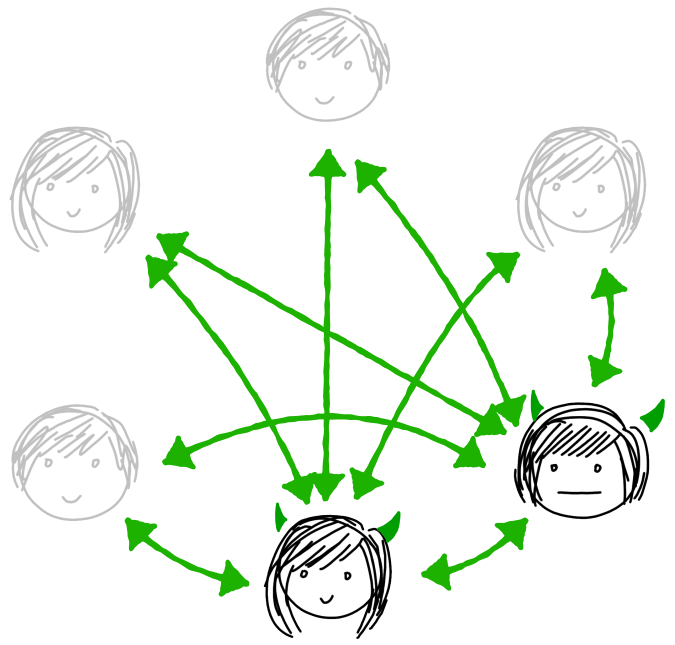
$\mathcal{S}_{\text{composed}}$

$\text{Sim}_{\text{composed}}$

$\mathcal{F}_{\text{outer}}$



Use $\text{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}



Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

round 3 of π_{outer}

round 4 of π_{outer}

ulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

In other words, we changed too much at once!

What if we designed a *sequence* of distributions that *blend* features of $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}}\to\pi_{\text{inner}}}$ and $\text{Sim}_{\text{composed}}$?

Each distribution in the sequence will replace exactly *one* protocol invocation (π_{inner} or π_{outer}) with *one* simulator invocation ($\text{Sim}_{\text{inner}}$ or $\text{Sim}_{\text{outer}}$).

We call these distributions *hybrid distributions*. If we prove that every *neighboring* pair is indistinguishable, then *transitivity* allows us to conclude that the *endpoints* ($\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}}\to\pi_{\text{inner}}}$ and $\text{Sim}_{\text{composed}}$) are indistinguishable too!

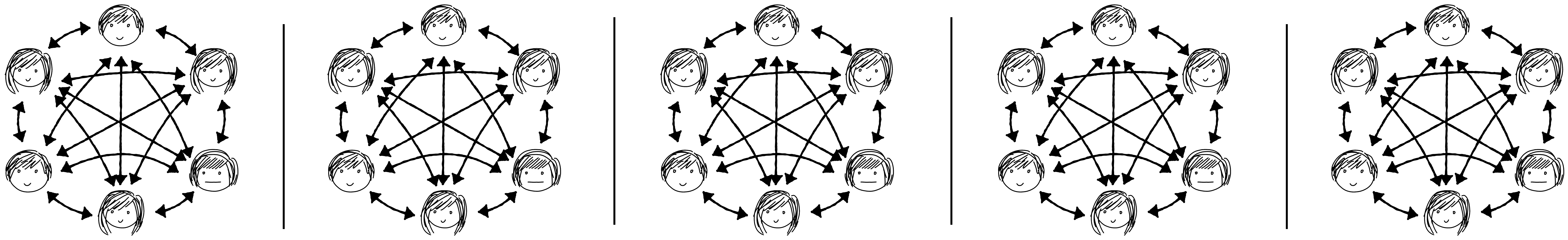
Note: *hybrid distributions* should not be confused with *hybrid models*. We use the former to design protocols and the latter to gradually show indistinguishability in proofs.

We have a protocol and a *hybrid distribution*

Real World

$\mathscr{A}$

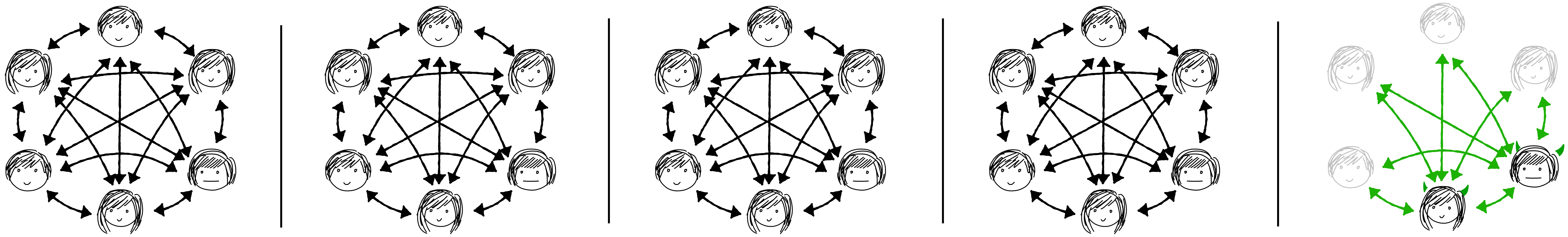
$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$



Round 1 of π_{outer} | Invoke π_{inner} (all rounds) | Round 3 of π_{outer} | Round 4 of π_{outer} | Invoke π_{inner} (all rounds)

Hybrid Dist.

$\mathscr{H}_1$



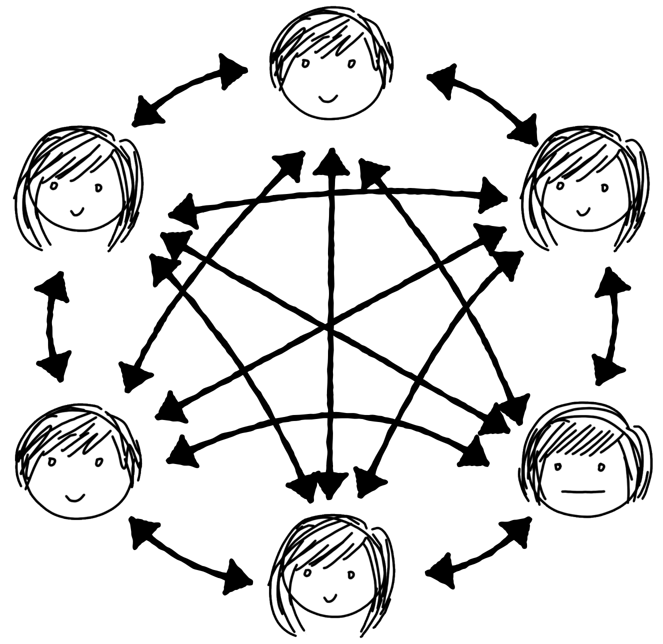
Round 1 of π_{outer} | Invoke π_{inner} (all rounds) | Round 3 of π_{outer} | Round 4 of π_{outer} | Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

We have a protocol and a *hybrid distribution*

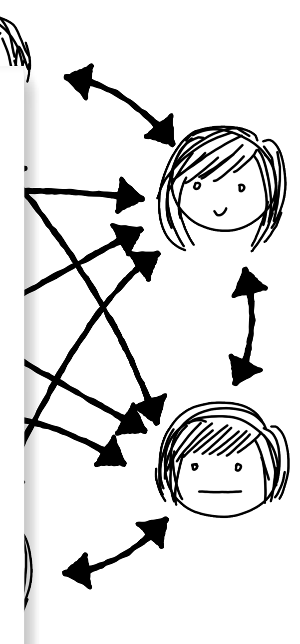
Real World

$\mathcal{A}$

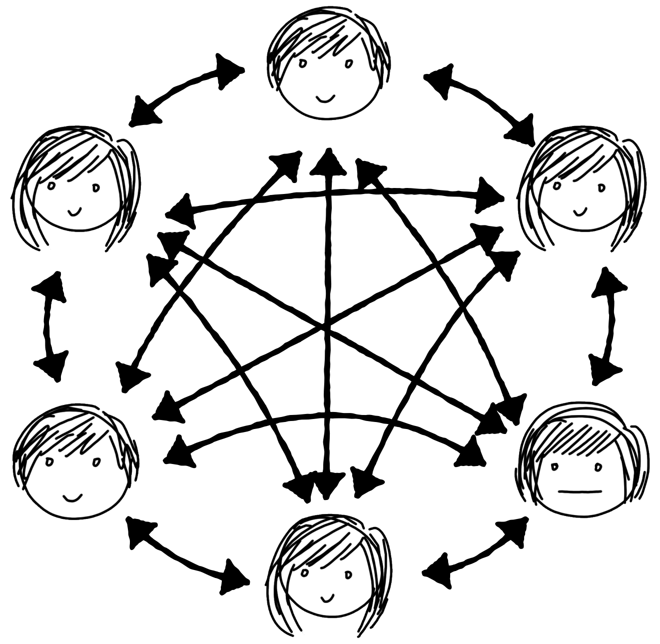
$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$



Round 1 of π_{outer}



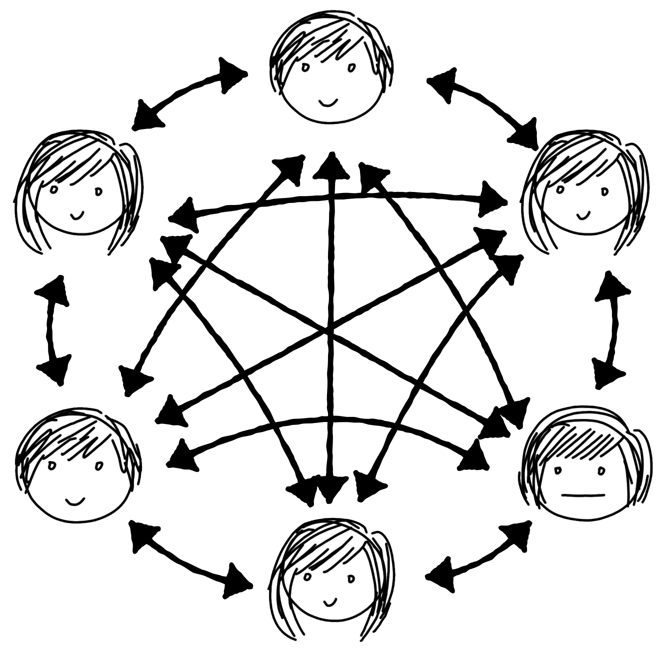
π_{outer}



Invoke π_{inner} (all rounds)

Hybrid Dist.

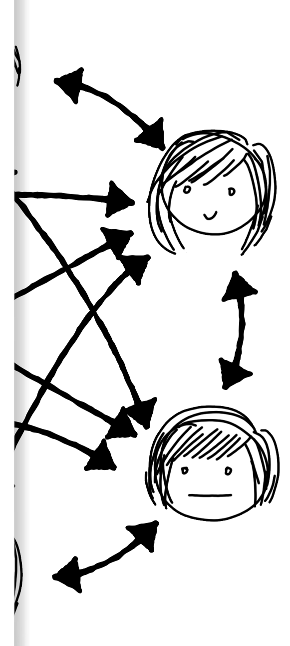
$\mathcal{H}_1$



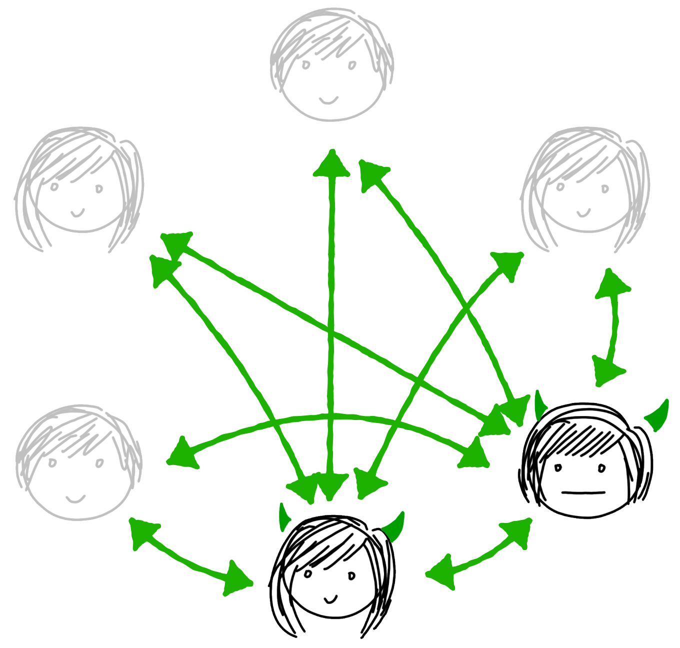
Round 1 of π_{outer}

Invoke π_{inner} (all rounds)

Round 3 of π_{outer}



Round 4 of π_{outer}



Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Remember, everything apart from the real world is a thought experiment.

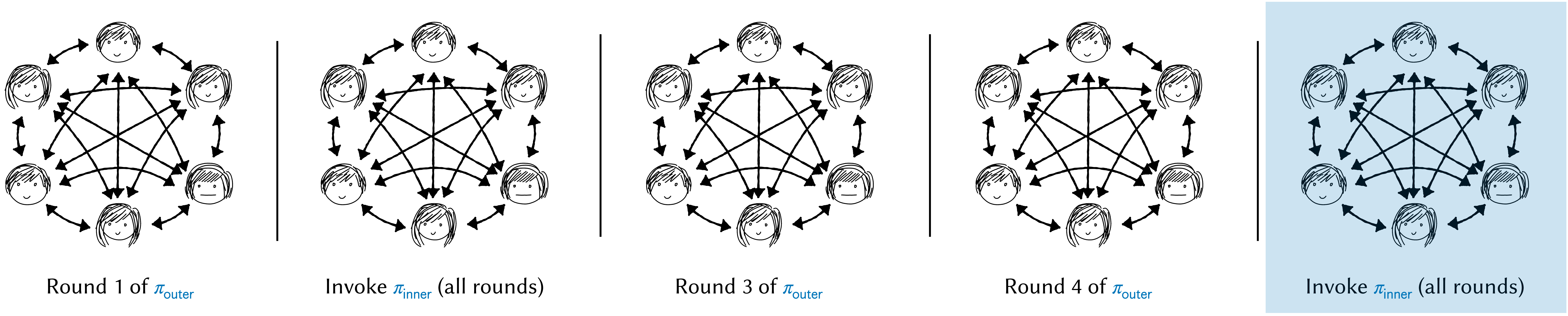
A hybrid distribution is coherent in the sense that it can sampled by an omniscient "game master" who has access to the internal state of all parties.

However, it will generally not be possible to for an adversary to sample a hybrid distribution on its own (like it can with the full simulation in the ideal world) and it will generally not make sense to run a hybrid distribution in the real world (like the protocol)

Real World

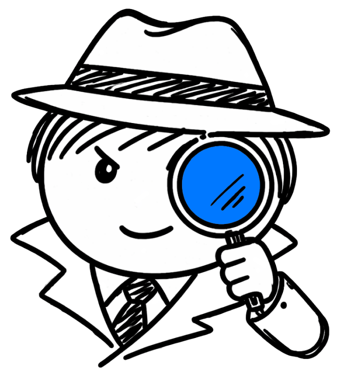
$\mathcal{A}$

$\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$



- Suppose towards contradiction that there exists a distinguisher $\mathcal{D}$ that can tell these two distributions apart.
- i.e. if we label them 0 and 1, suppose that

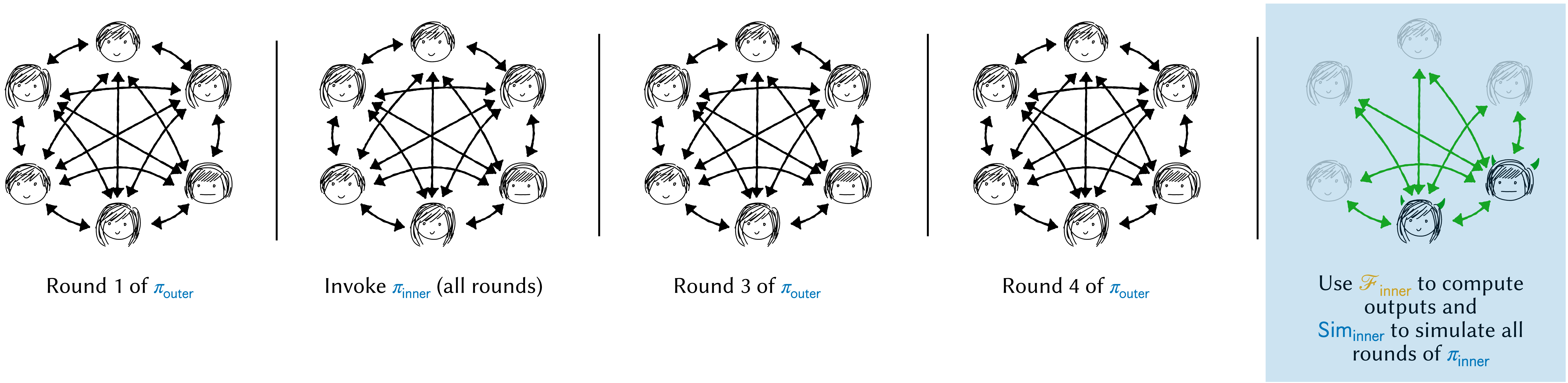
$$\Pr\left[\mathcal{D} \text{ outputs 1 given } \pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}\right] \neq \Pr[\mathcal{D} \text{ outputs 1 given } \mathcal{H}_1]$$



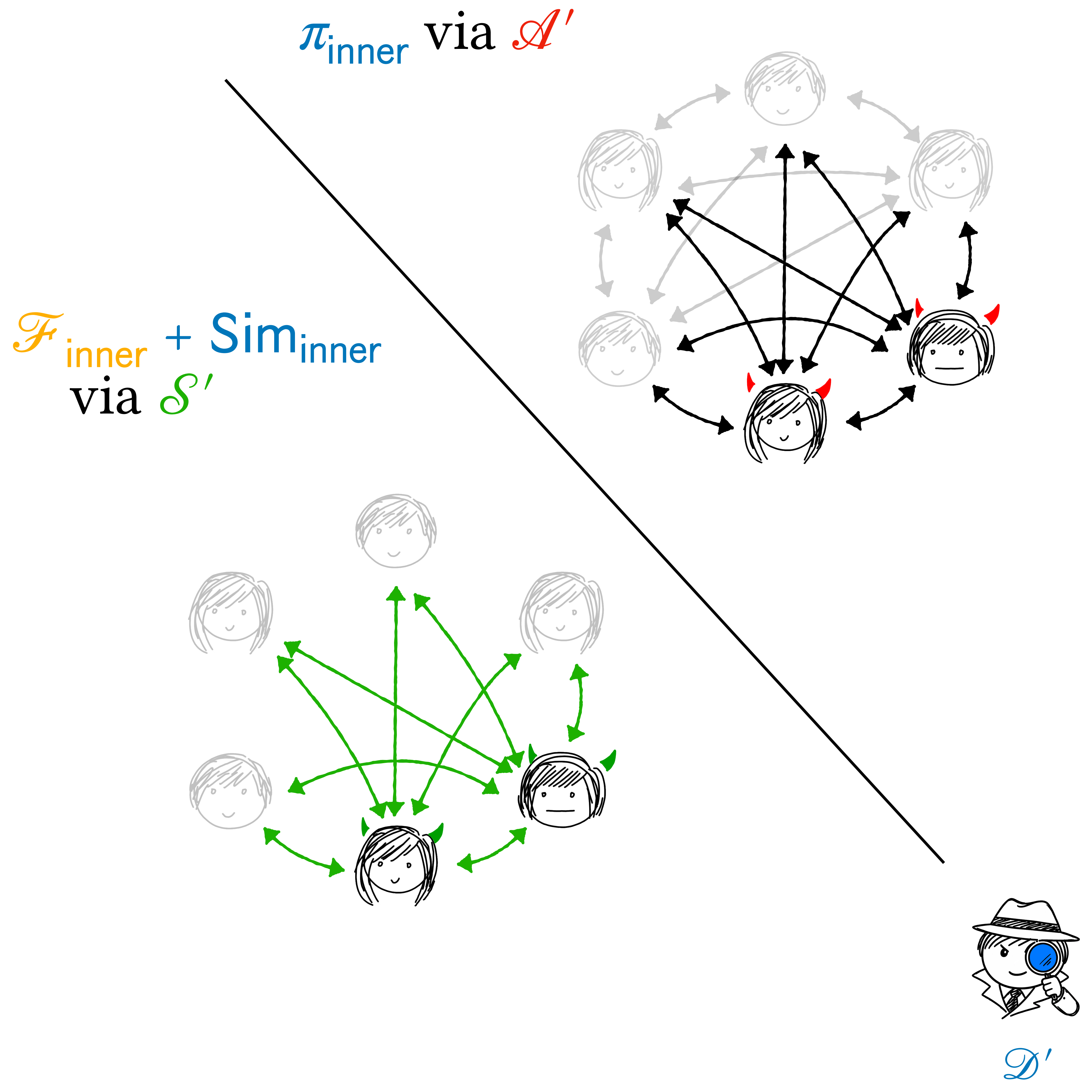
- We want to show that we can use $\mathcal{D}$ to build *another* distinguisher $\mathcal{D}'$ that can distinguish just the corrupt view of π_{inner} from $\mathsf{Sim}_{\text{inner}}$, which contradicts the security of π_{inner}.

Hybrid Dist.

$\mathcal{H}_1$

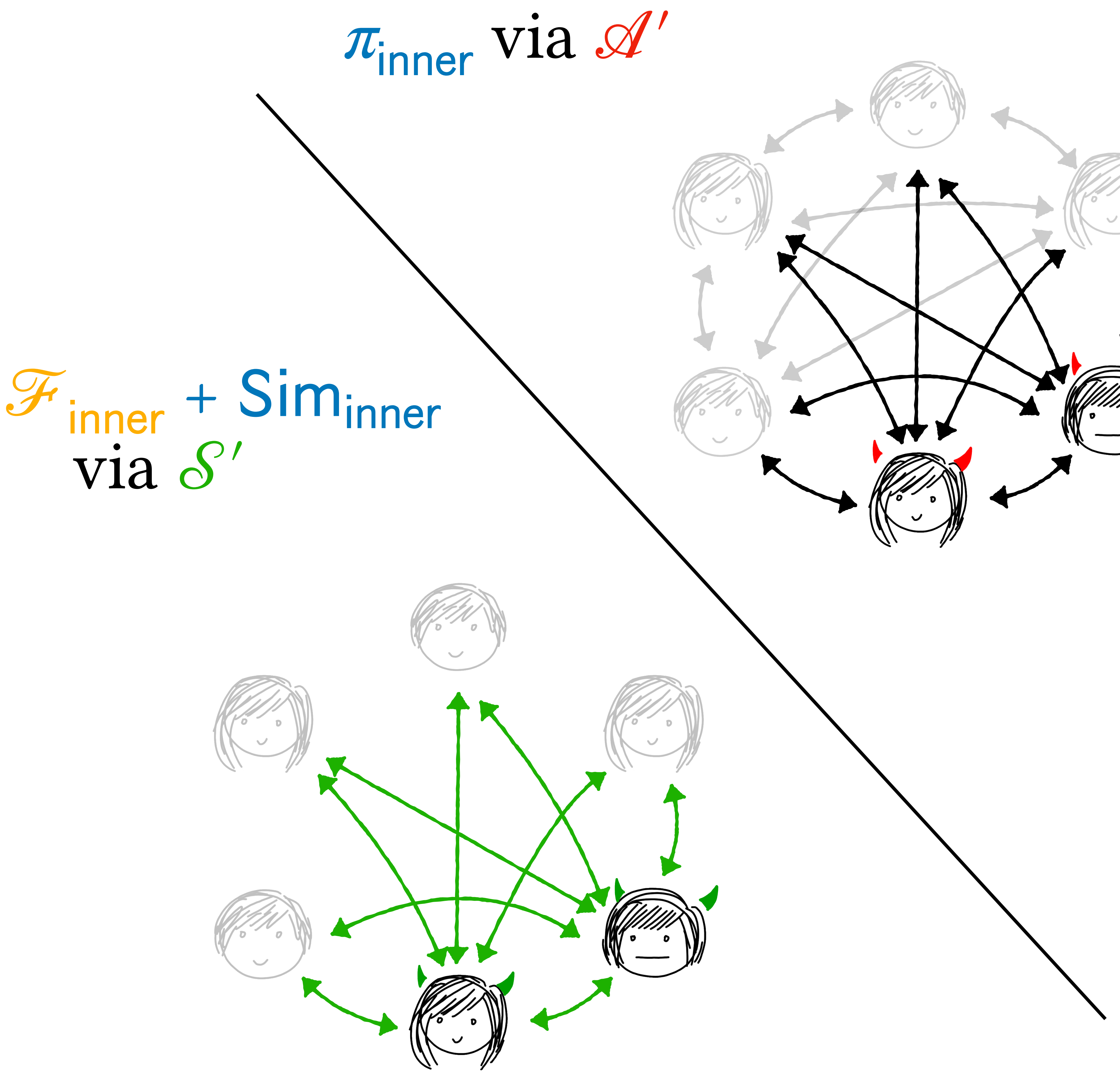


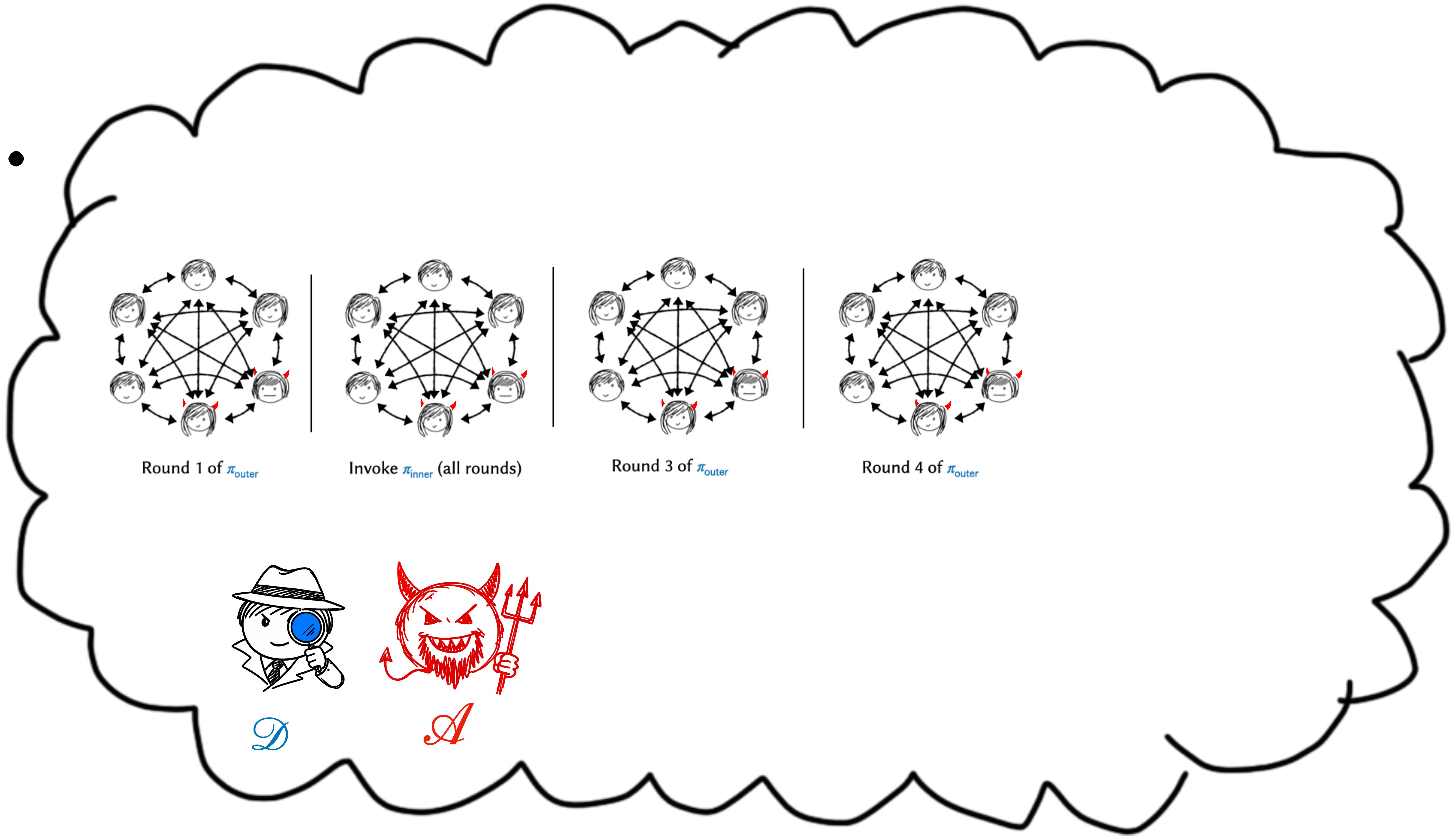
Building $\mathscr{D}'$ using $\mathscr{D}$ (a *Security Reduction*).

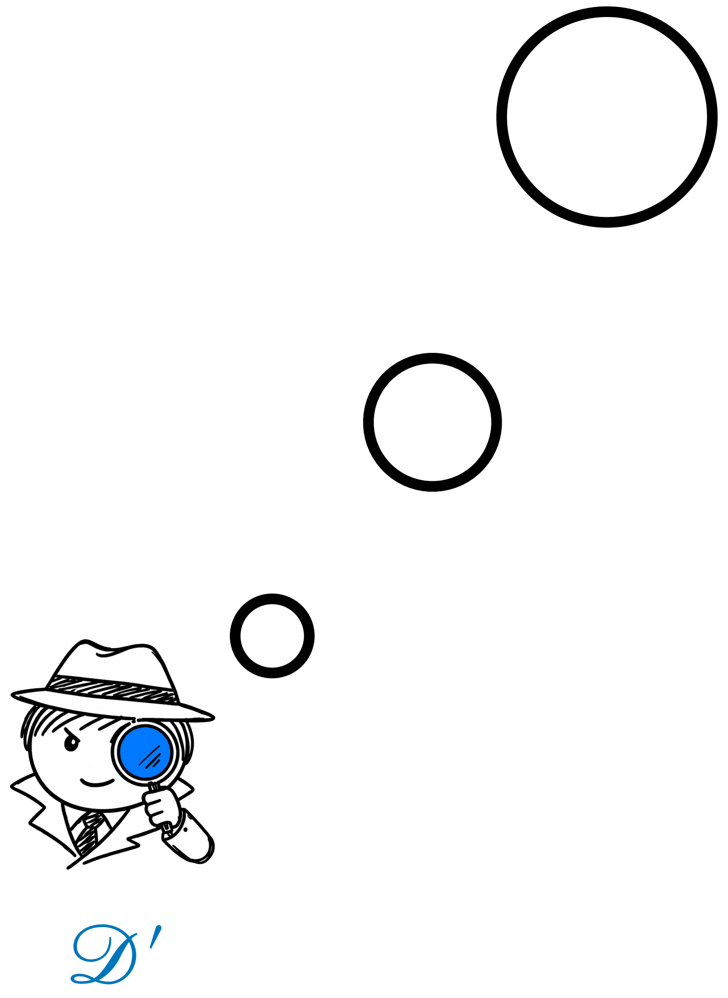


- $\mathscr{D}'$ is exposed to an experiment wherein it is given *either* a real *or* simulated view of the corrupt parties in π_{inner}. Recall that these views are supplied by $\mathscr{A}'$ and $\mathscr{S}'$ respectively. We will assume is collaborating with the $\mathscr{A}'$ that returns everything it can see.
- $\mathscr{D}'$ can run $\mathscr{D}$ "in its head" (i.e. as a subroutine). Since $\mathscr{D}$ only knows how to distinguish $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ from $\mathscr{H}_1$, $\mathscr{D}'$ must construct one of those two.
- $\mathscr{D}'$ also emulates the specific $\mathscr{A}$ that makes $\mathscr{D}$ effective.
- The goal is to make it so that if $\mathscr{D}$ guesses correctly whether it sees $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ or $\mathscr{H}_1$, then $\mathscr{D}'$ guesses correctly whether it sees π_{inner} or $\mathscr{F}_{\text{inner}}$ + $\text{Sim}_{\text{inner}}$.
- The good news is that the only difference between $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ and $\mathscr{H}_1$ is that one call to π_{inner} is replaced by a call to $\mathscr{F}_{\text{inner}}$ + $\text{Sim}_{\text{inner}}$. This is exactly the same as the difference between the two distributions that $\mathscr{D}'$ might be seeing!

Building $\mathcal{D}'$ using $\mathcal{D}$.



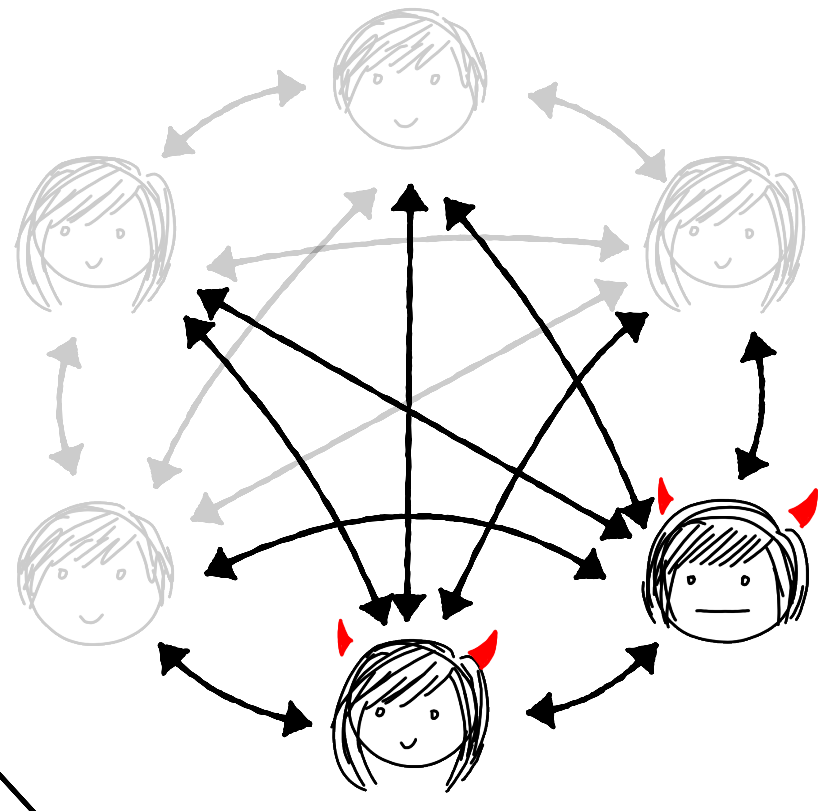




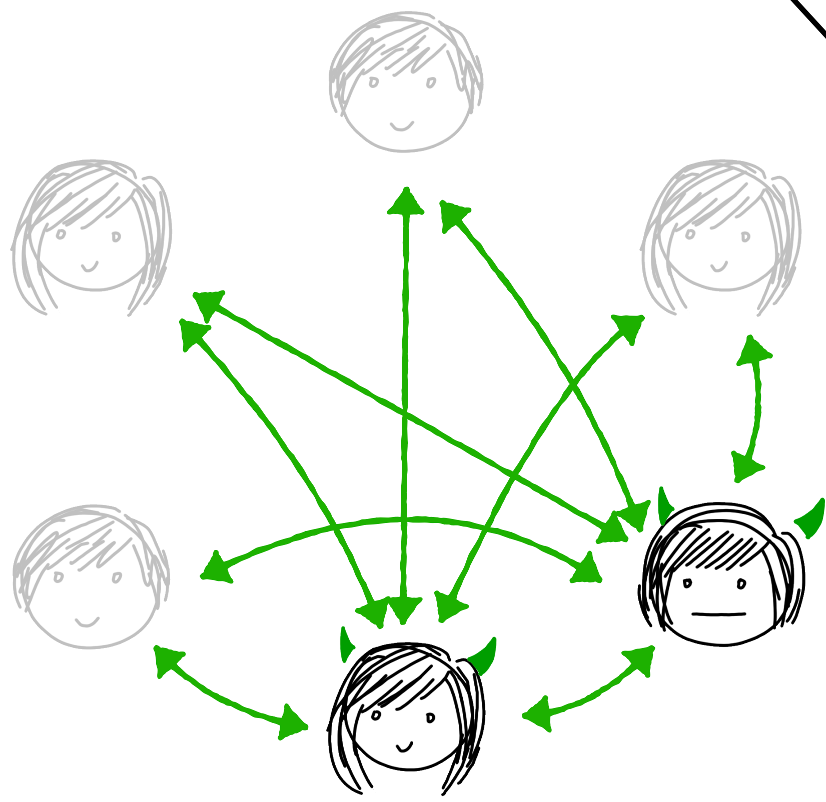
- The initial rounds of $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}}\to\pi_{\text{inner}}}$ and $\mathcal{H}_1$ are identical, so $\mathcal{D}'$ can simply sample them in its head for $\mathcal{D}$ and $\mathcal{A}$.
- Arriving at the last round, $\mathcal{D}'$ knows the inputs of *all parties* for the next step. It also knows which parties the $\mathcal{A}$ working with $\mathcal{D}$ has corrupted in the emulated experiment.

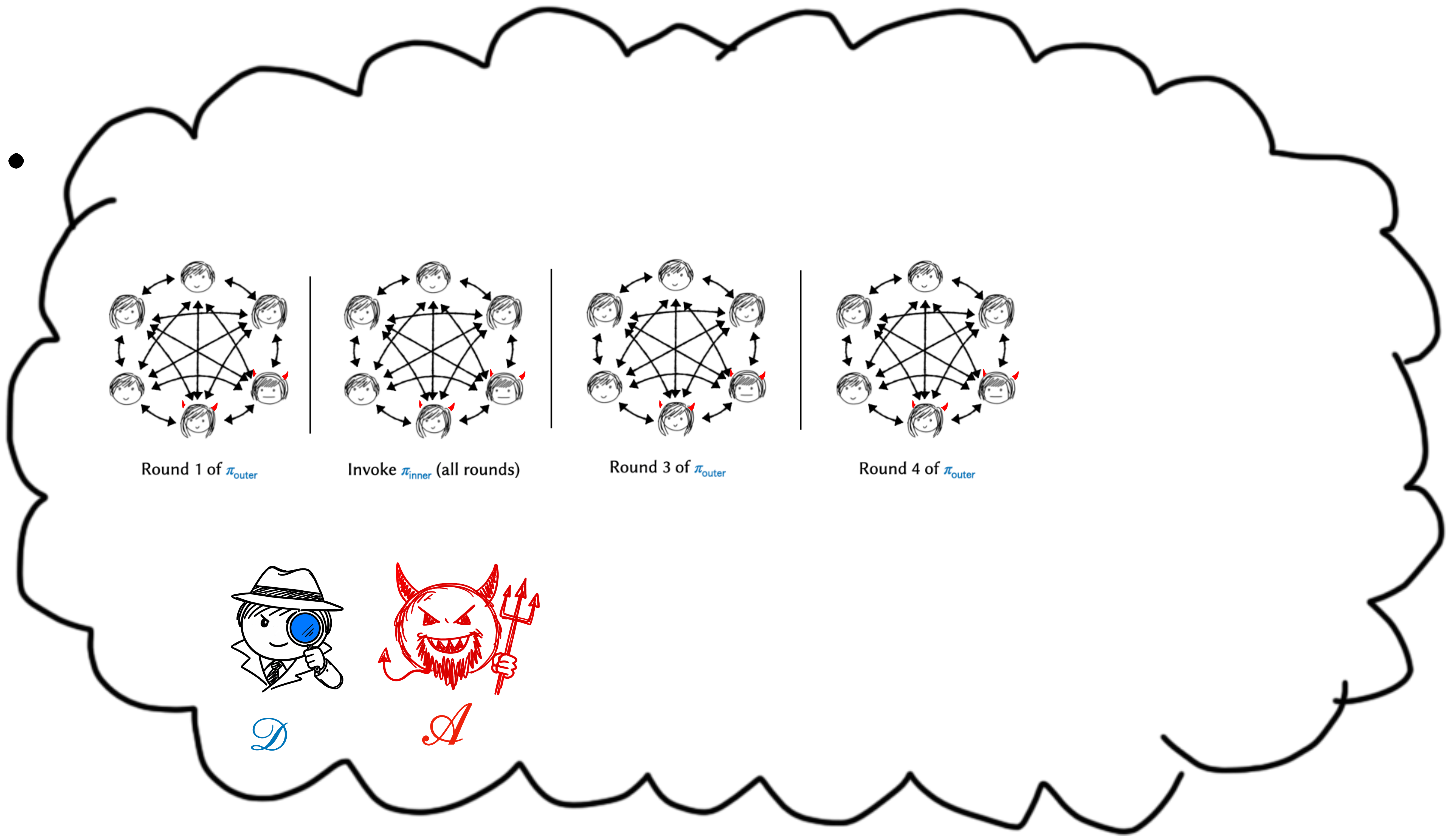
Building $\mathscr{D}'$ using $\mathscr{D}$.

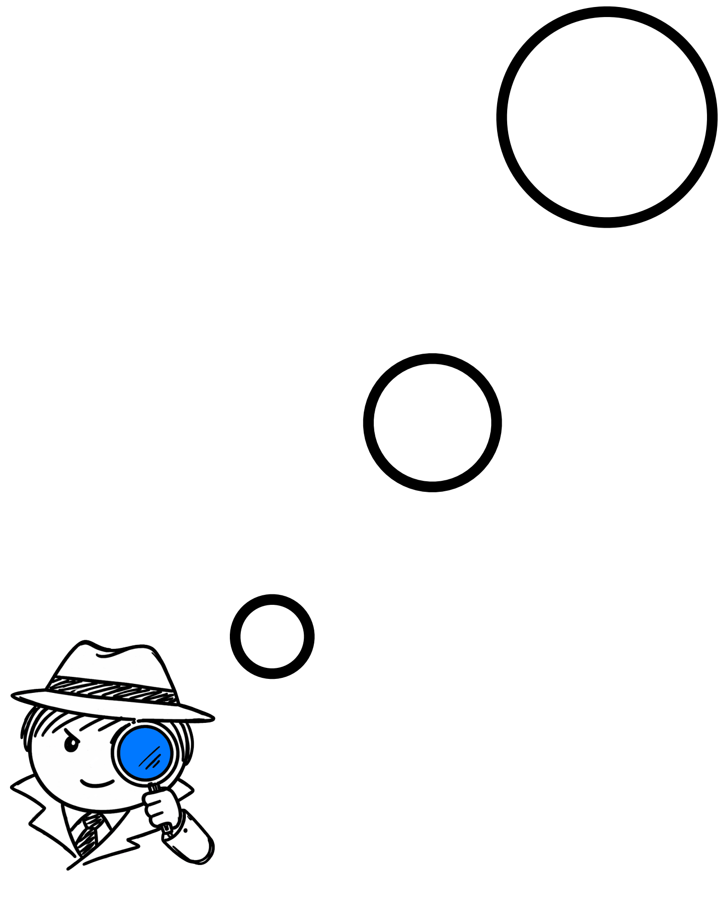
π_{inner} via $\mathscr{A}'$



$\mathscr{F}_{\text{inner}}$ + $\text{Sim}_{\text{inner}}$ via $\mathscr{S}'$

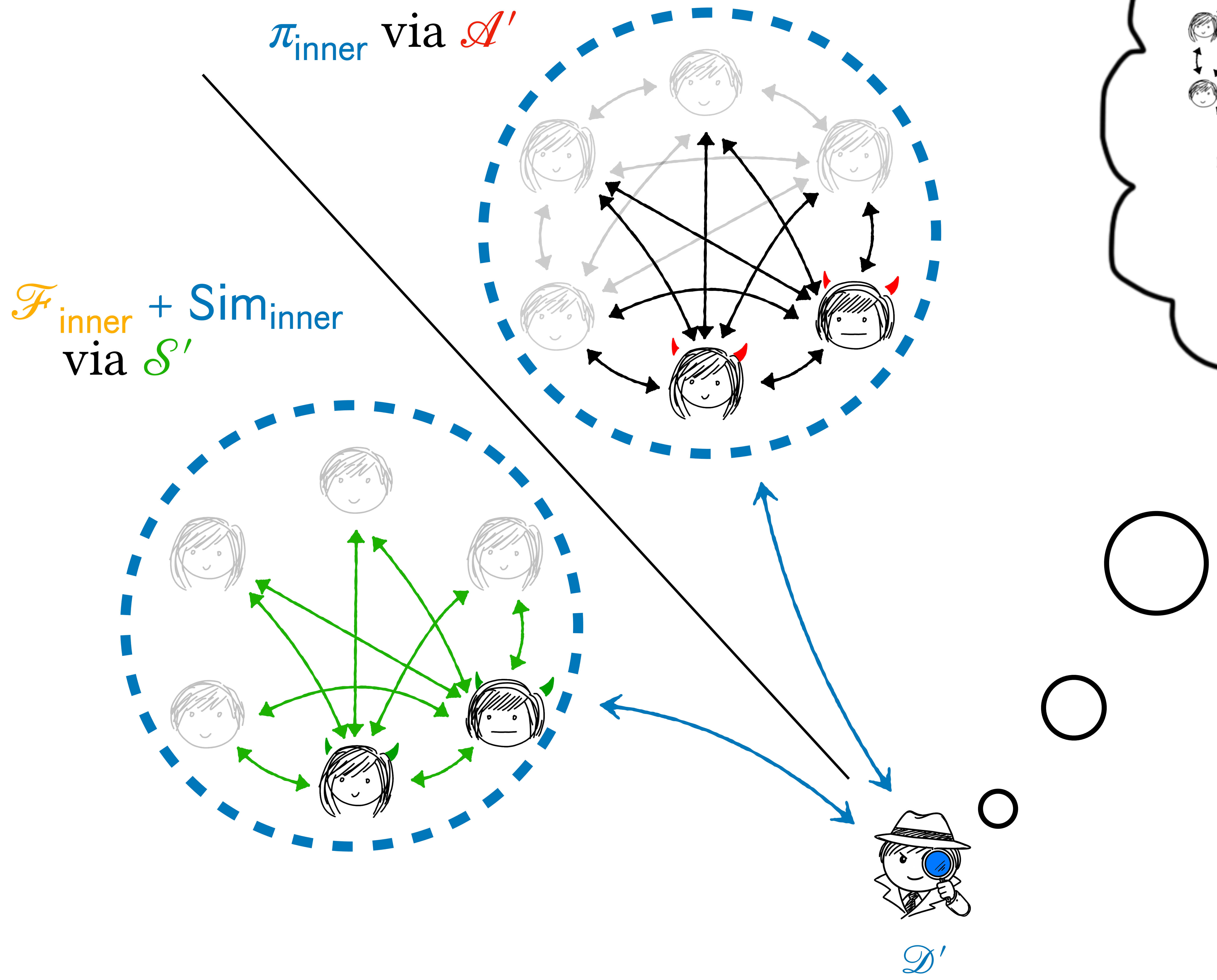


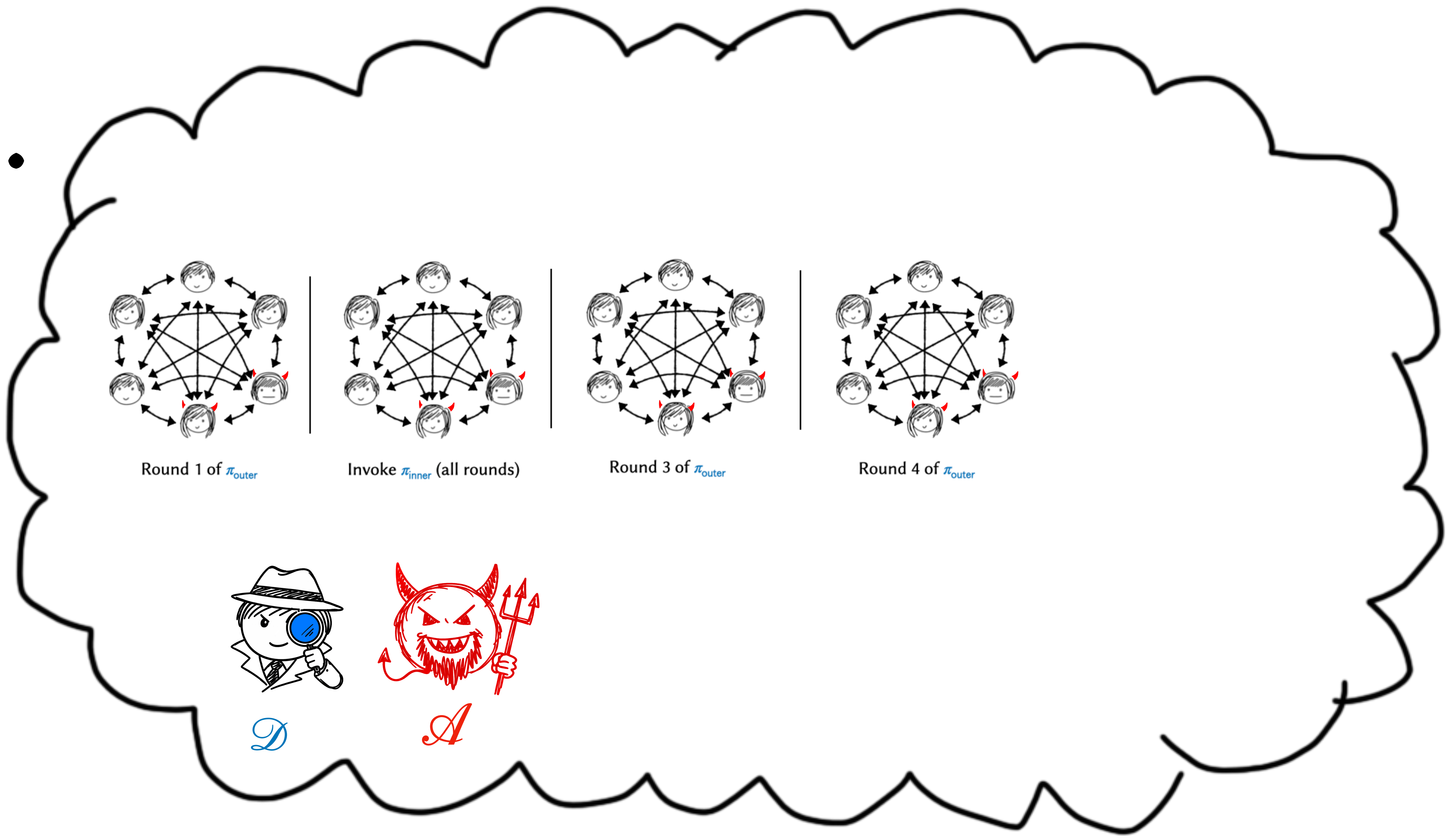




- Since $\mathscr{D}'$ controls the inputs in its own experiment, it can set them to be the ones that it generated in its head using the first rounds of $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}} / \mathscr{H}_1$.
- $\mathscr{D}'$ also instructs its own adversary (either $\mathscr{A}'$ or $\mathscr{S}'$) to corrupt the same parties that $\mathscr{A}$ asked to corrupt in the emulated experiment.

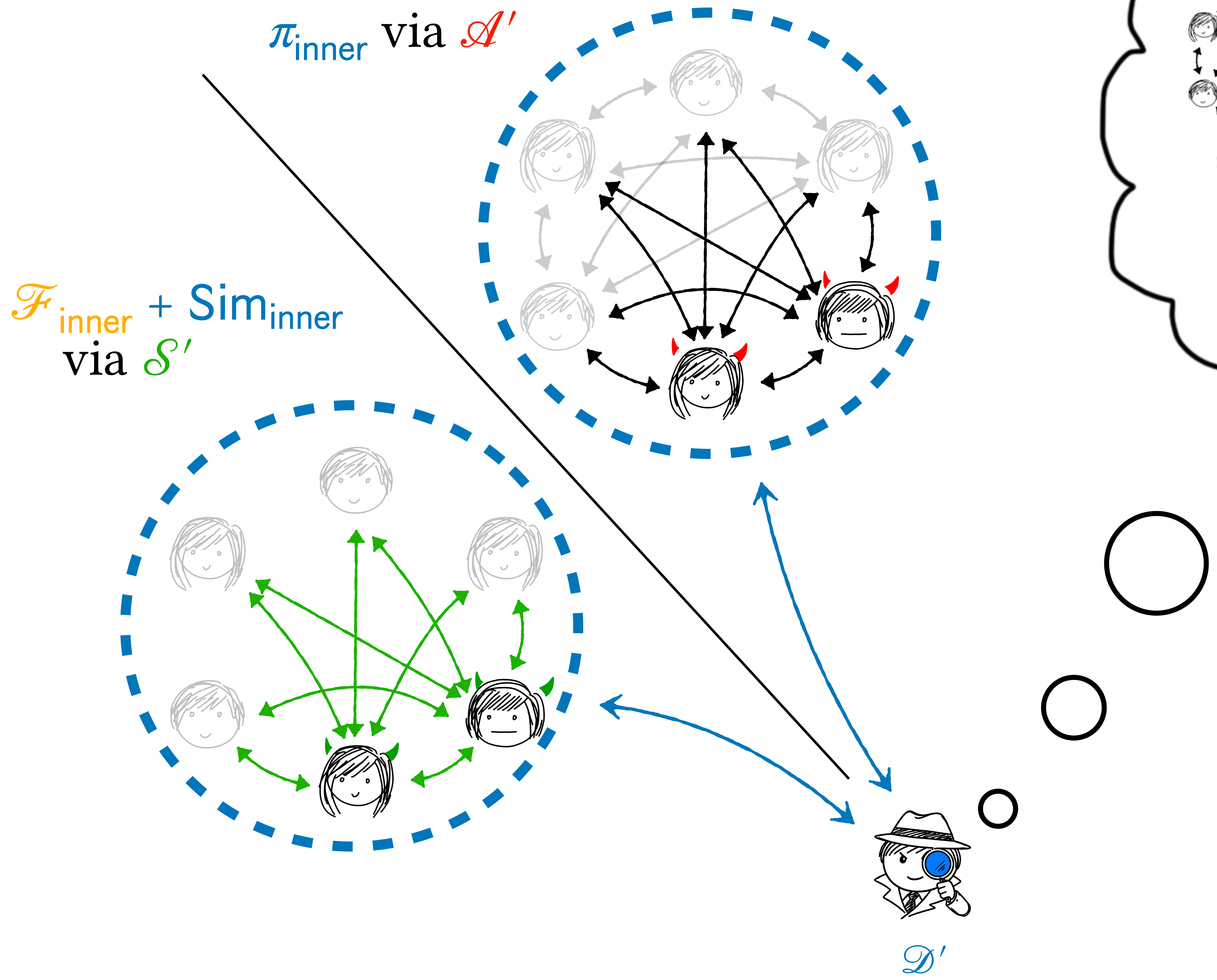
Building $\mathscr{D}'$ using $\mathscr{D}$.

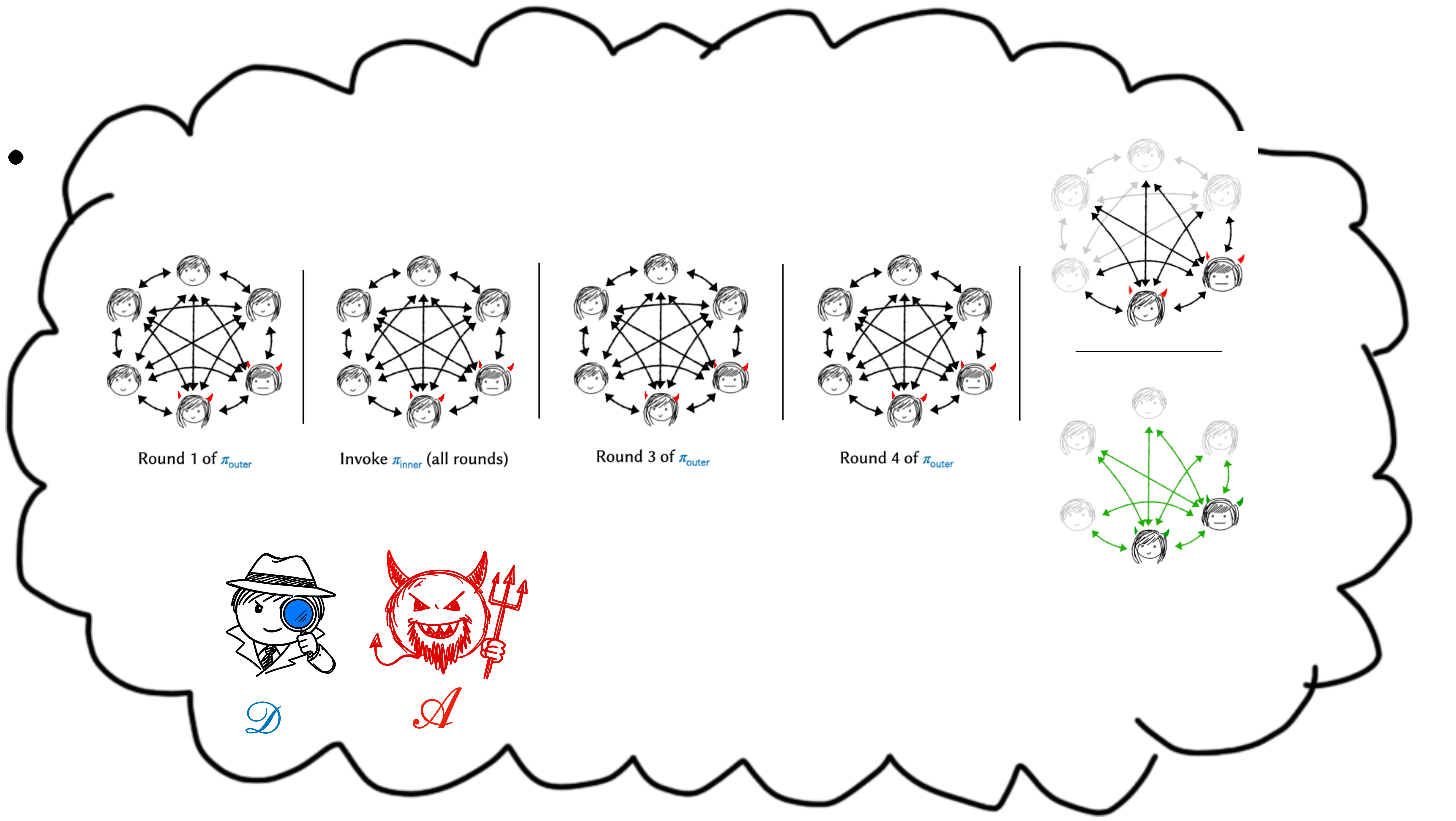




- After doing this, $\mathscr{D}'$ receives a view from *either* $\mathscr{A}'$ or $\mathscr{S}'$ (it does not know which) and it embeds that view back into the experiment in its head for $\mathscr{D}$ and $\mathscr{A}$ as the final round.

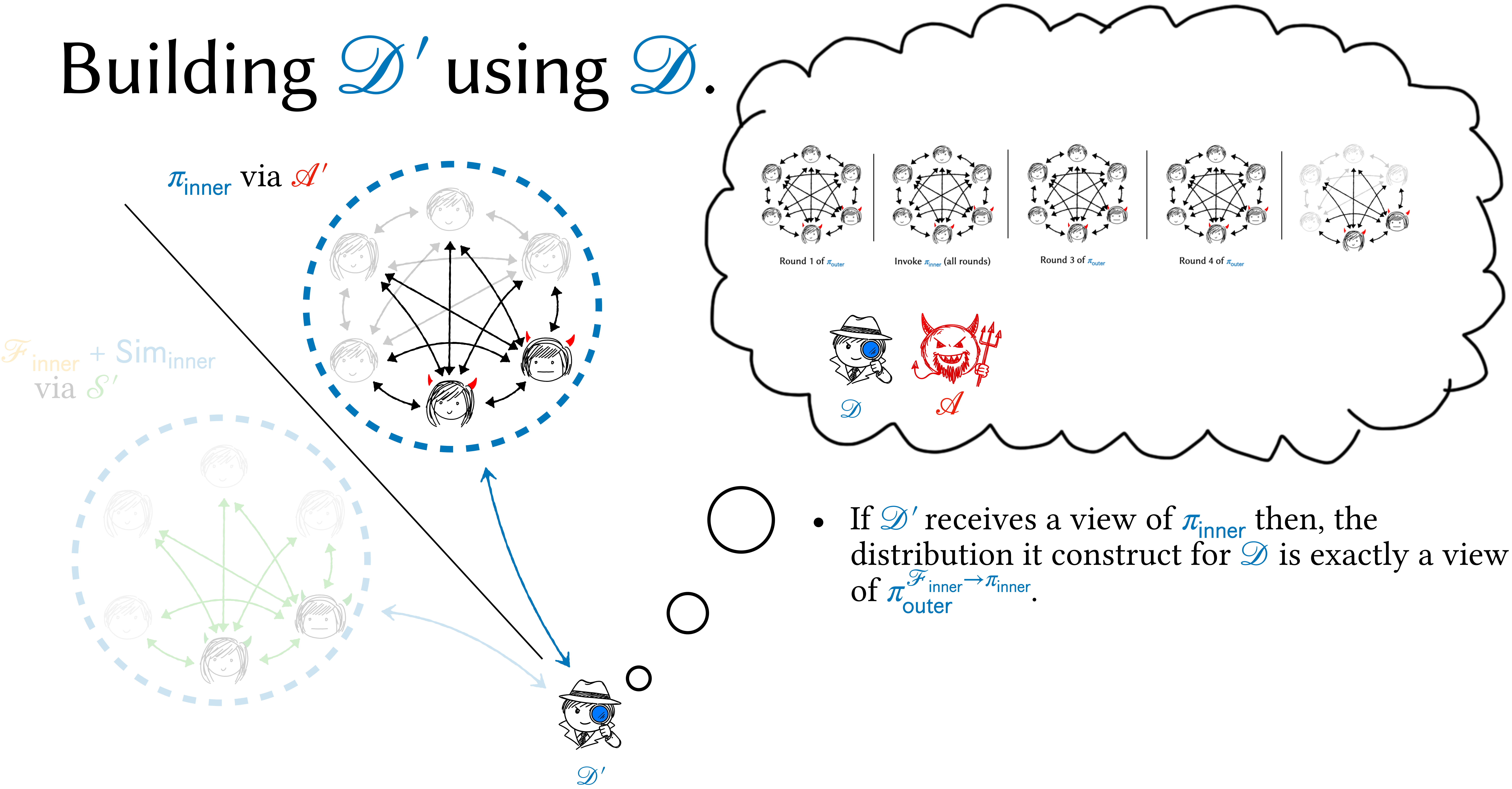
Building $\mathscr{D}'$ using $\mathscr{D}$.





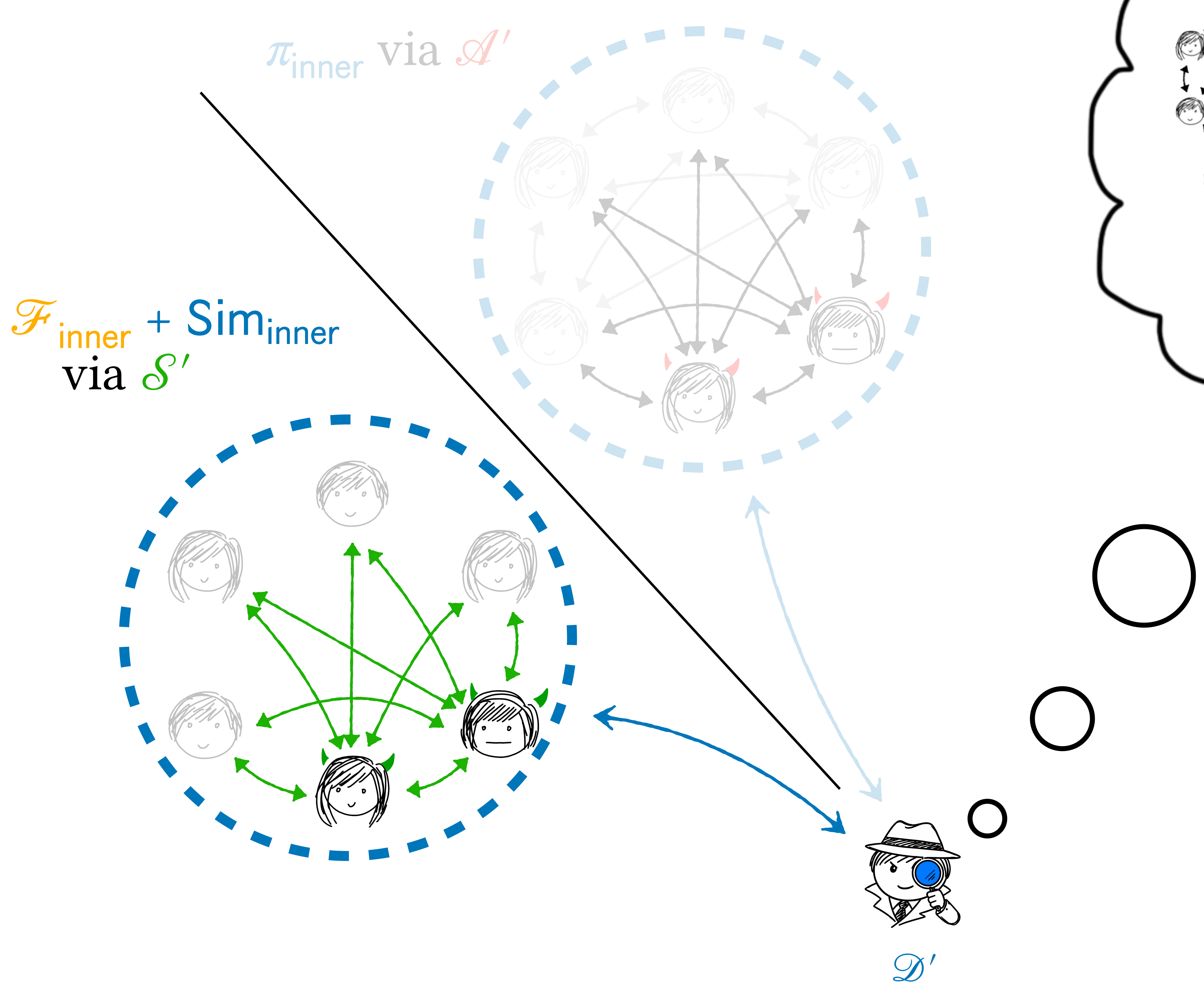
- After doing this, $\mathscr{D}'$ receives a view from *either* $\mathscr{A}'$ or $\mathscr{S}'$ (it does not know which) and it embeds that view back into the experiment in its head for $\mathscr{D}$ and $\mathscr{A}$ as the final round.

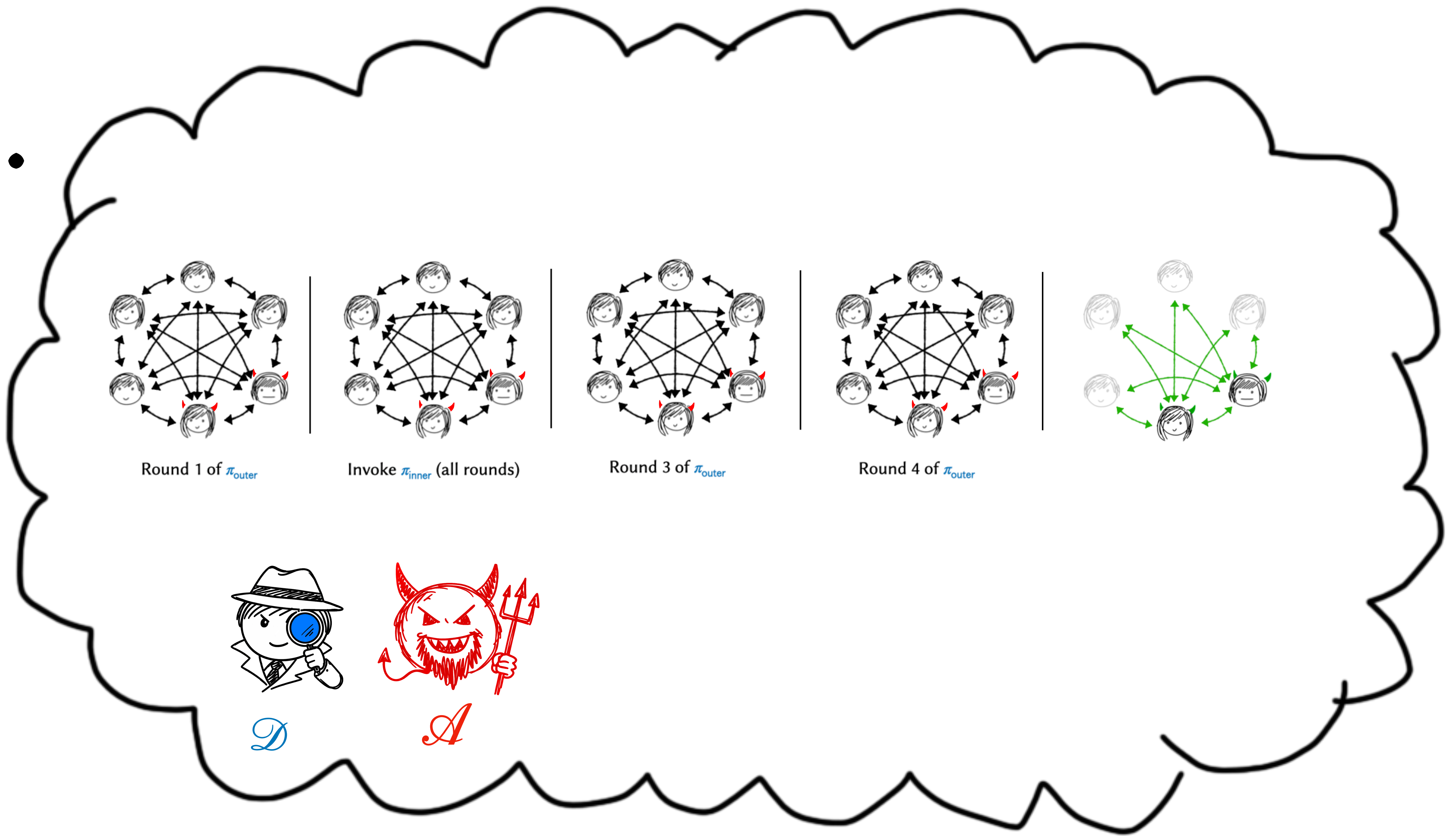
Building $\mathscr{D}'$ using $\mathscr{D}$.



- If $\mathscr{D}'$ receives a view of π_{inner} then, the distribution it construct for $\mathscr{D}$ is exactly a view of $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}}\to\pi_{\text{inner}}}$.

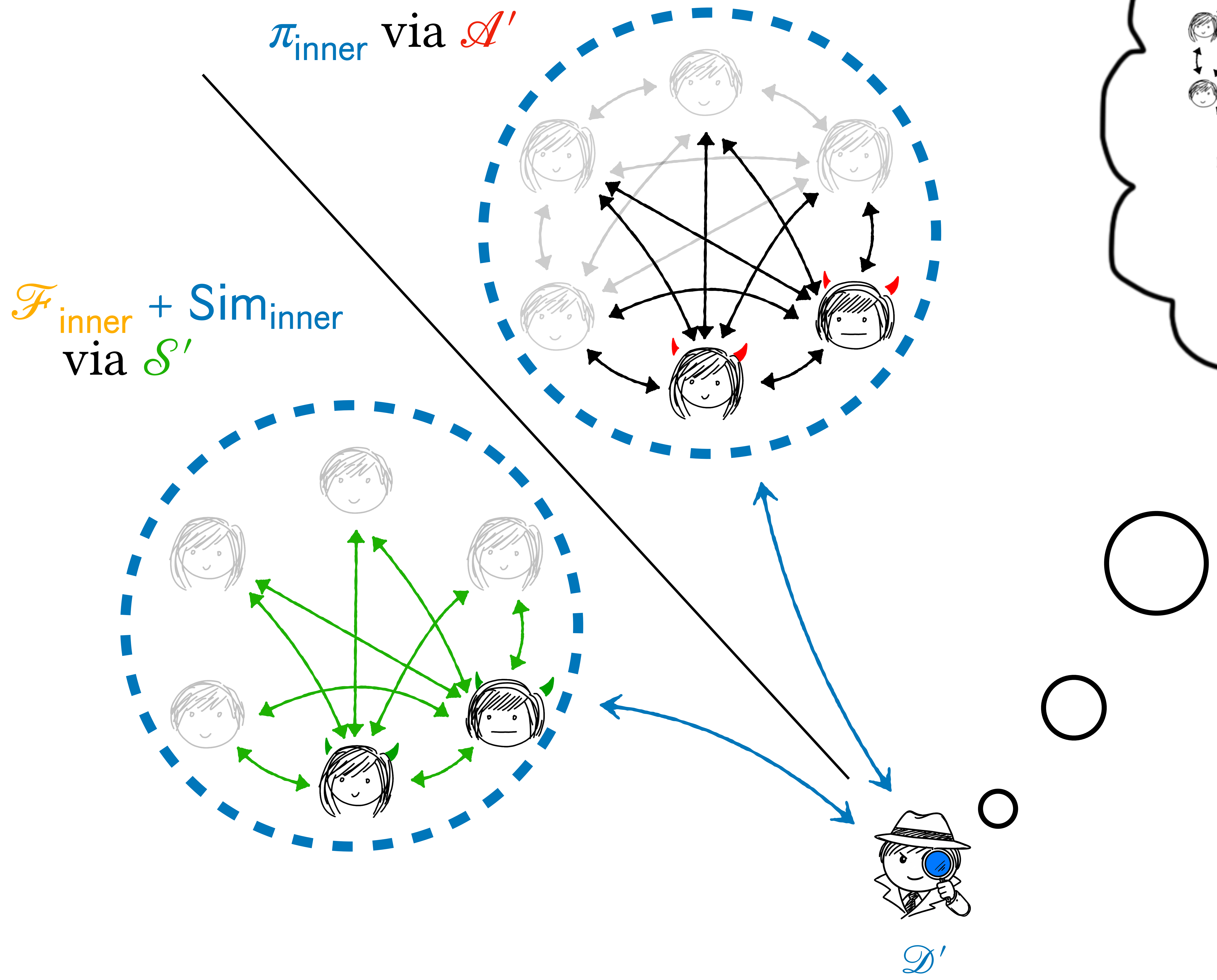
Building $\mathscr{D}'$ using $\mathscr{D}$.

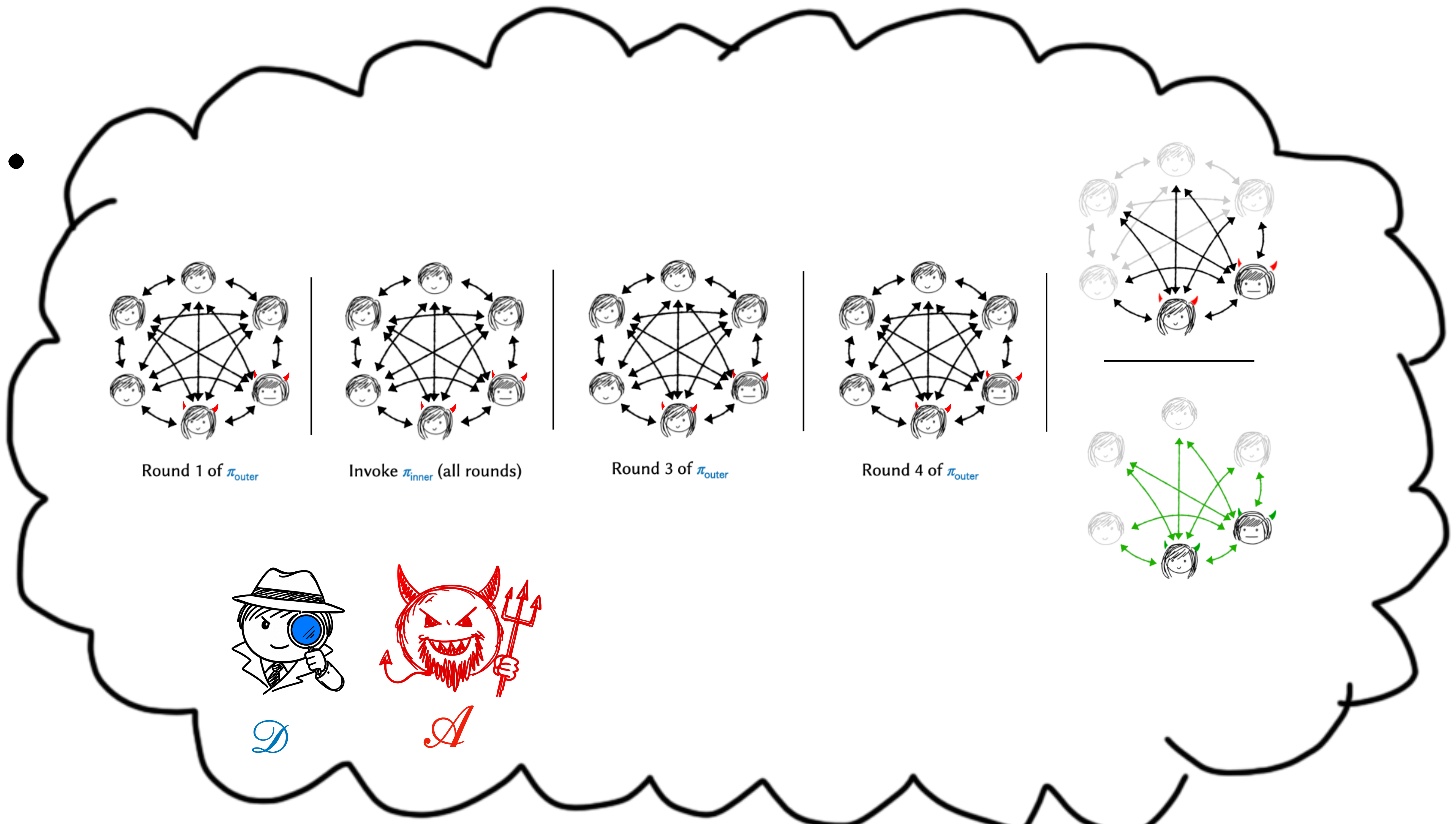




- If $\mathscr{D}'$ receives a view of π_{inner} then, the distribution it constructs for $\mathscr{D}$ is exactly a view of $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$.
- If $\mathscr{D}'$ receives a view of $\mathscr{F}_{\text{inner}}$ + $\text{Sim}_{\text{inner}}$, then the distribution it constructs for $\mathscr{D}$ is exactly a view of $\mathscr{H}_1$.

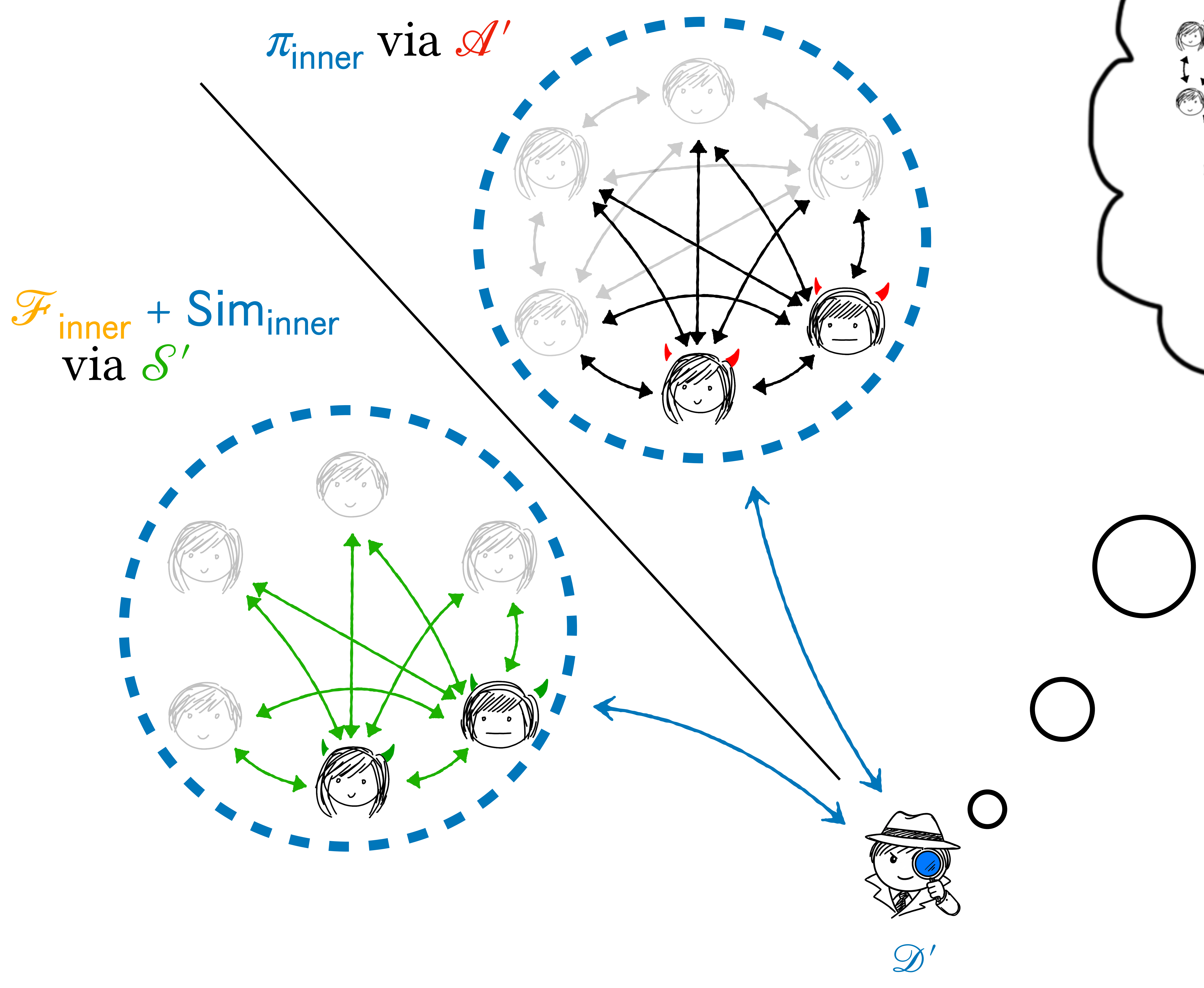
Building $\mathscr{D}'$ using $\mathscr{D}$.

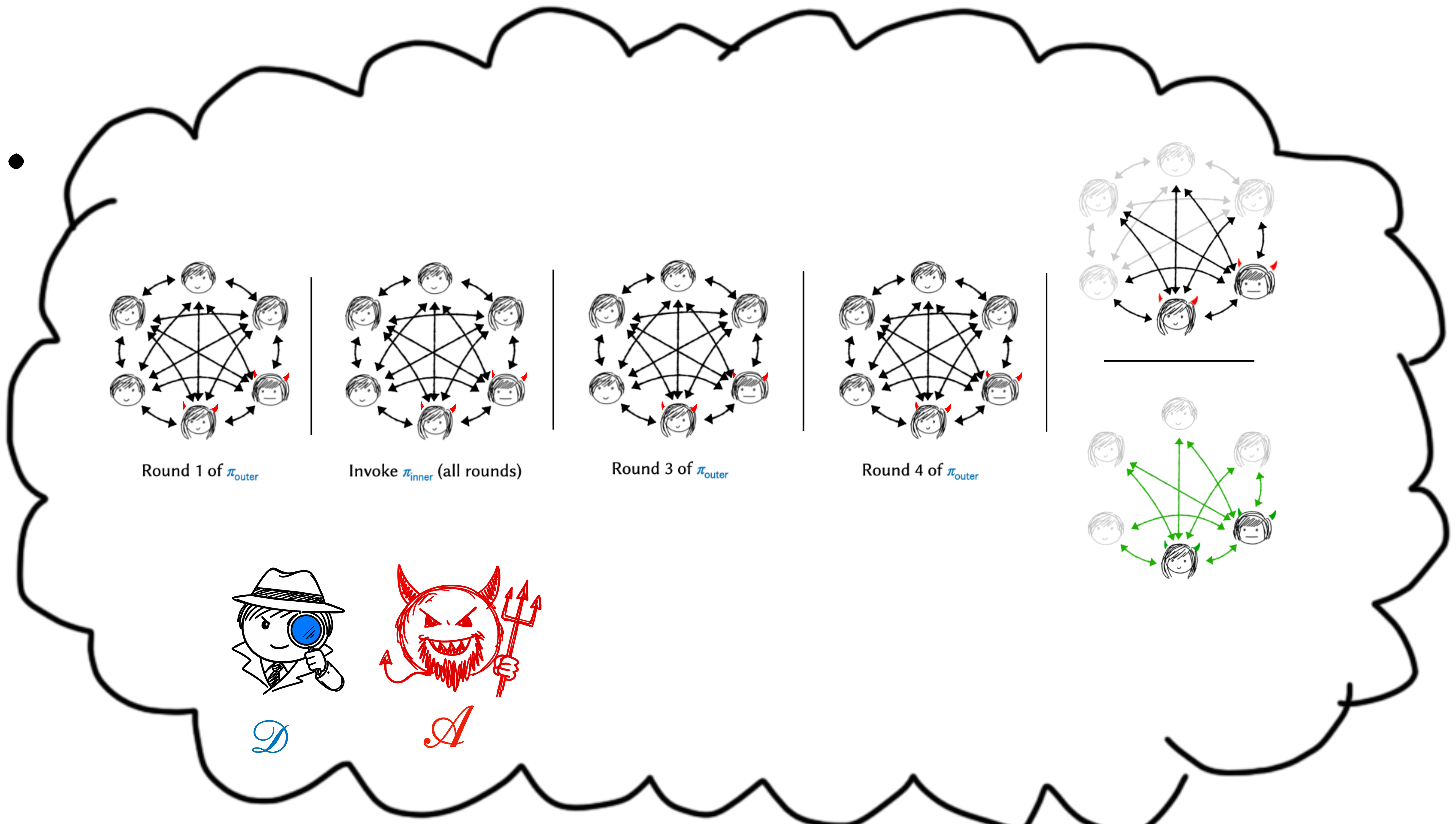




- It is important that the two possible views in the emulated experiment be identically distributed to $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ and $\mathscr{H}_1$, because those are the only views that $\mathscr{D}$ and $\mathscr{A}$ are guaranteed to work correctly on.
- In some sense, we are playing a trick on $\mathscr{D}$ and $\mathscr{A}$. If they detect it, they might refuse to do anything!

Building $\mathscr{D}'$ using $\mathscr{D}$.

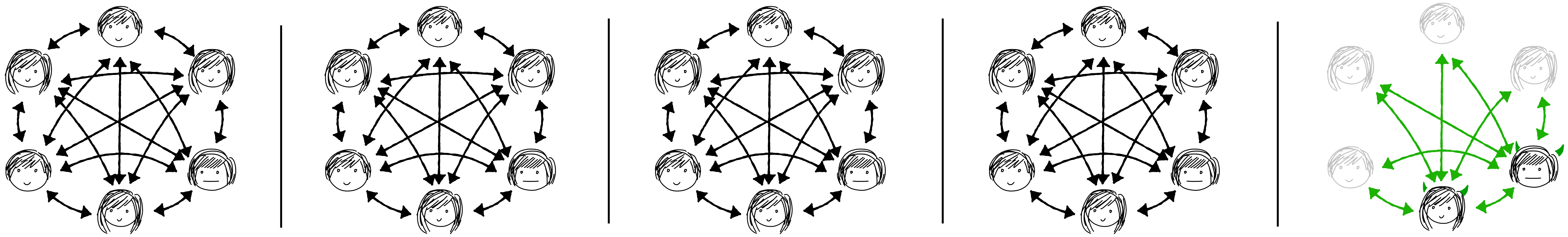




- In this case, the emulated views are perfect, so $\mathscr{D}'$ distinguishes just as effectively as $\mathscr{D}$.
- However, one of our premises was that π_{inner} realizes $\mathscr{F}_{\text{inner}}$. This implies that no effective distinguisher $\mathscr{D}'$ can exist.
- Putting these facts together, no effective distinguisher $\mathscr{D}$ can exist, and thus $\mathscr{H}_1$ and $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ are identically distributed!

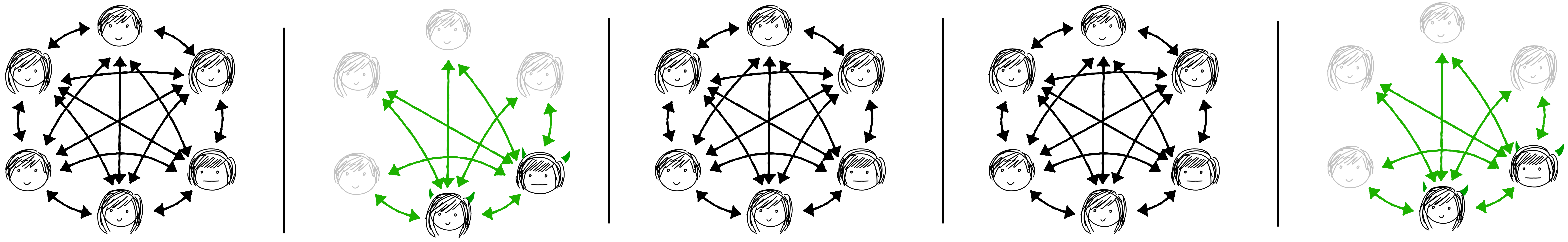
Each hybrid distribution makes *one* change.

Hybrid Dist. $\mathcal{H}_1$



Round 1 of π_{outer} | Invoke π_{inner} (all rounds) | Round 3 of π_{outer} | Round 4 of π_{outer} | Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\mathsf{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Hybrid Dist. $\mathcal{H}_2$

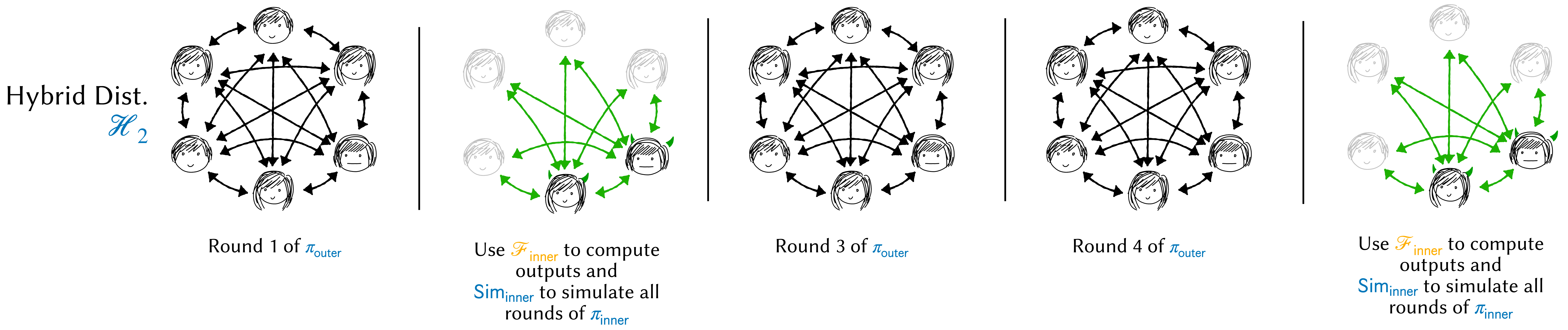


Round 1 of π_{outer} | Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\mathsf{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner} | Round 3 of π_{outer} | Round 4 of π_{outer} | Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\mathsf{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Each hybrid distribution makes *one* change.

The exact same argument that convinced us that $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}} \equiv \mathcal{H}_1$ from the distinguisher's perspective can be repeated to convince us that $\mathcal{H}_1 \equiv \mathcal{H}_2$ from the distinguisher's perspective.

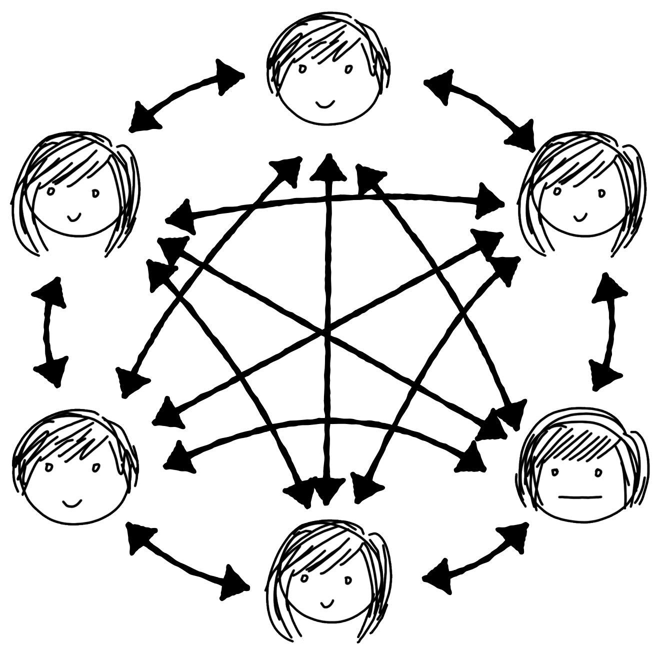
Each hybrid distribution makes *one* change.



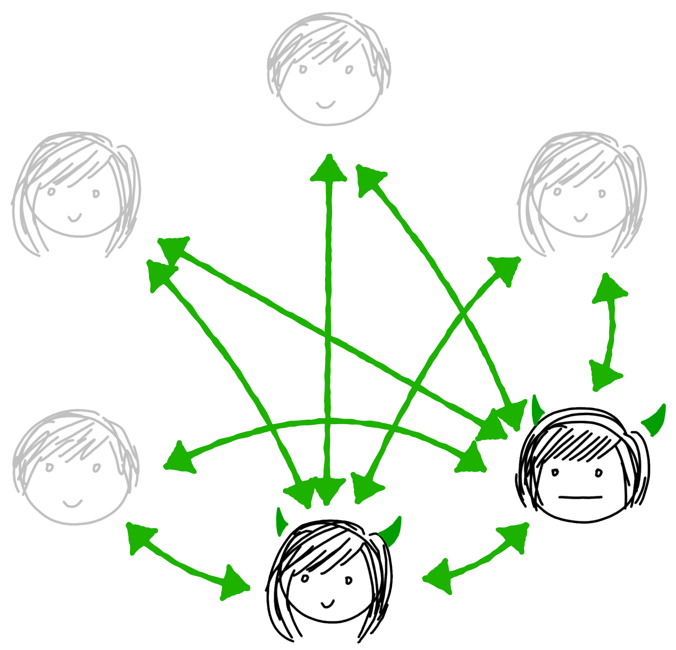
- Now we have replaced all invocations of π_{inner} with simulations that were created using $\mathsf{Sim}_{\text{inner}}$.
- We do not need to know the honest parties internal state to use $\mathsf{Sim}_{\text{inner}}$, though; we only need the internal state of the corrupt parties.
- This means that we can finally replace the outer protocol π_{outer} with a simulation created using $\mathsf{Sim}_{\text{outer}}$!

Each hybrid distribution makes *one* change.

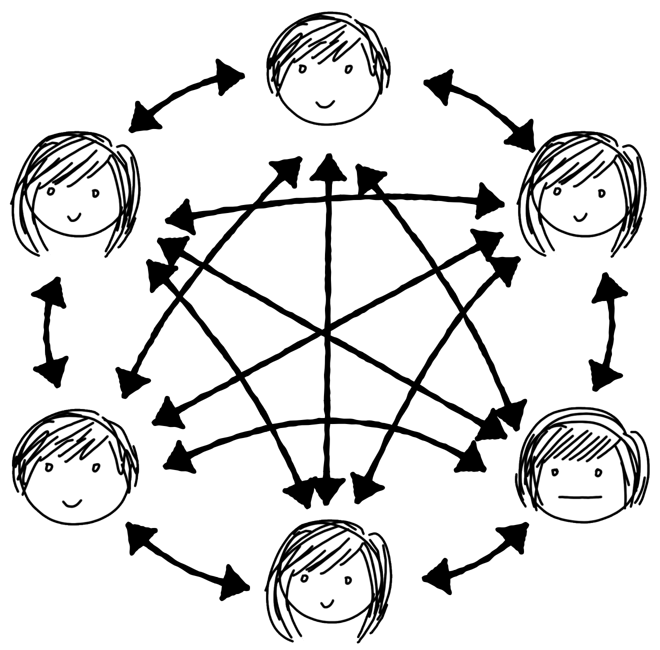
Hybrid Dist. $\mathcal{H}_2$



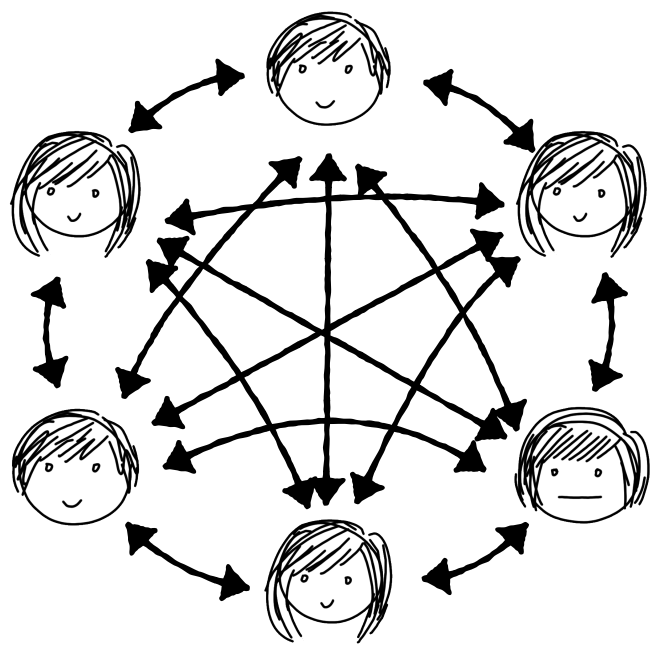
Round 1 of π_{outer}



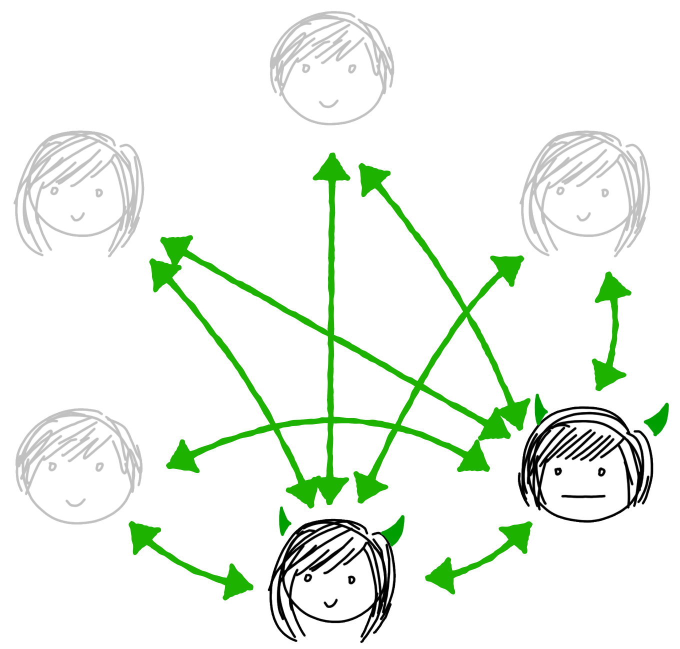
Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}



Round 3 of π_{outer}

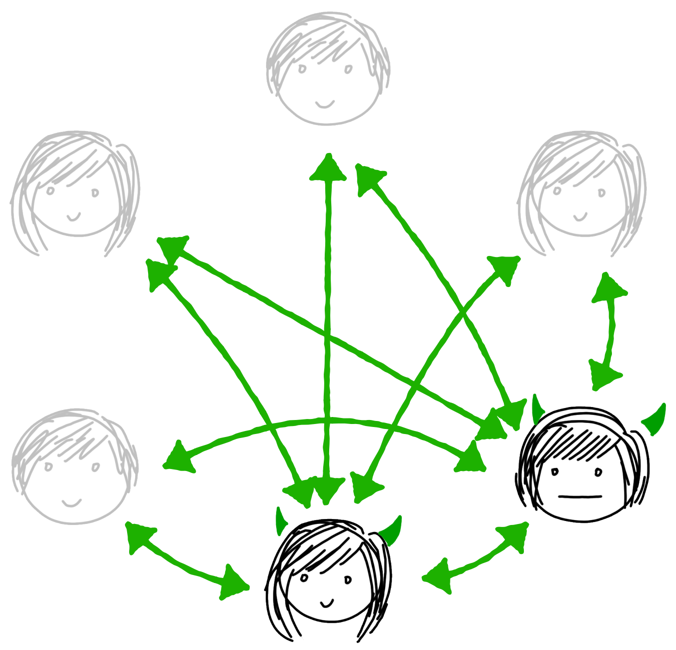


Round 4 of π_{outer}

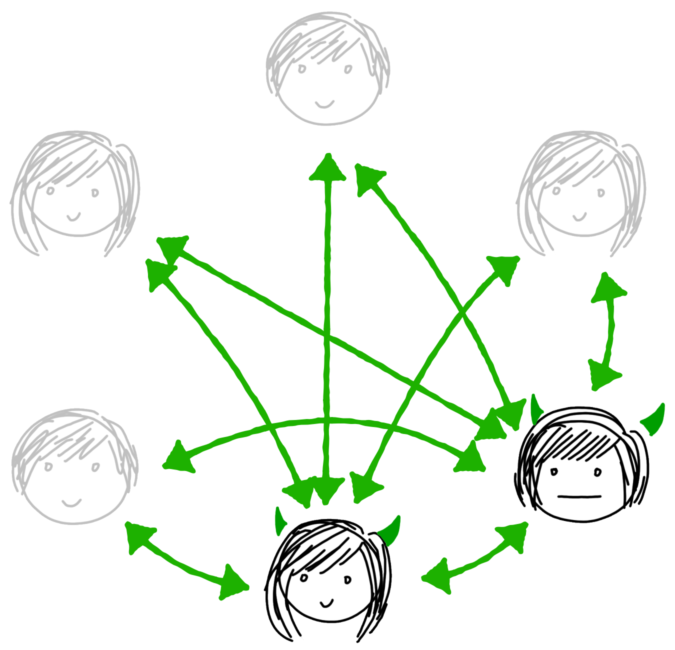


Use $\mathcal{F}_{\text{inner}}$ to compute outputs and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

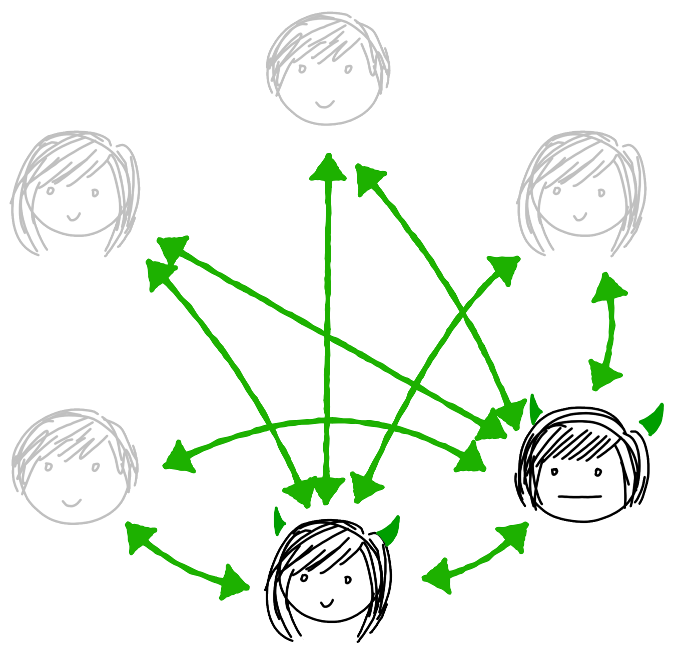
Ideal World $\mathcal{S}_{\text{composed}}$ $\text{Sim}_{\text{composed}}$ $\mathcal{F}_{\text{outer}}$



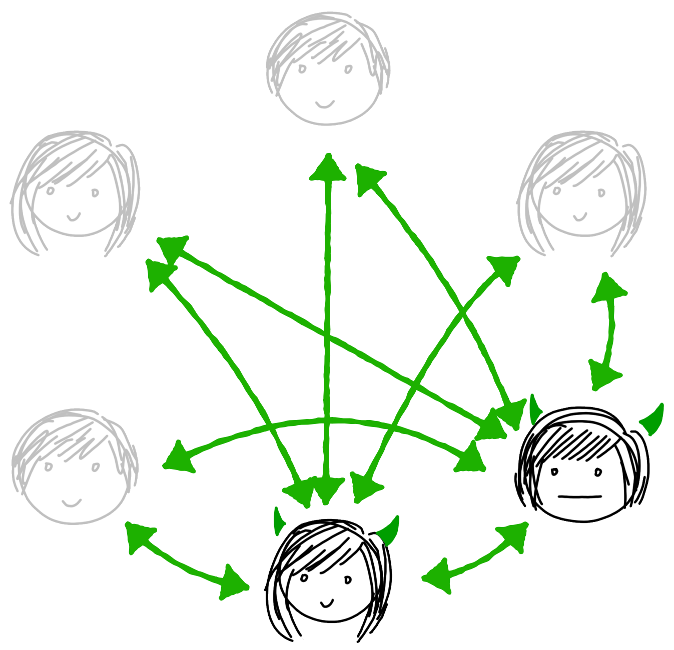
Use $\text{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}



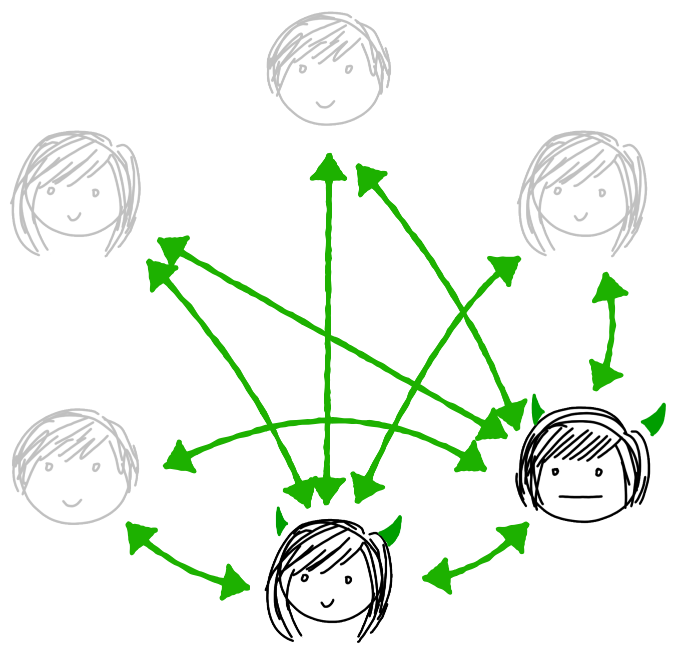
Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}



Use $\text{Sim}_{\text{outer}}$ to simulate round 3 of π_{outer}



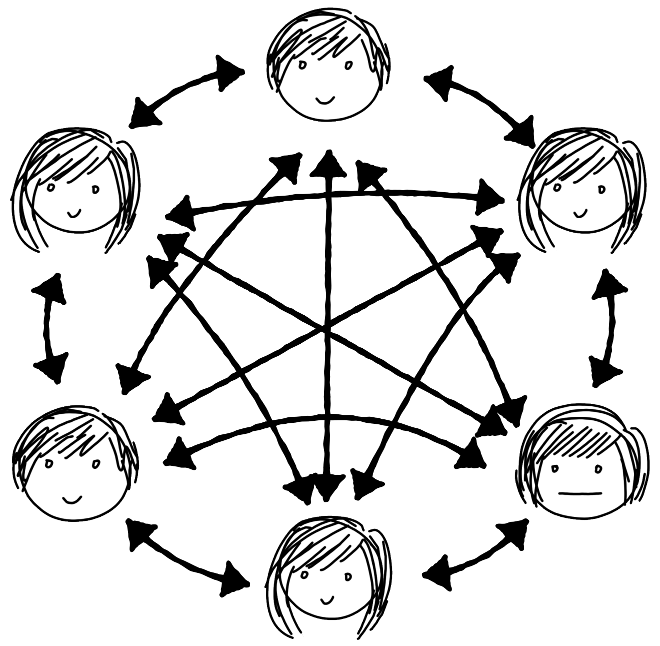
Use $\text{Sim}_{\text{outer}}$ to simulate round 4 of π_{outer}



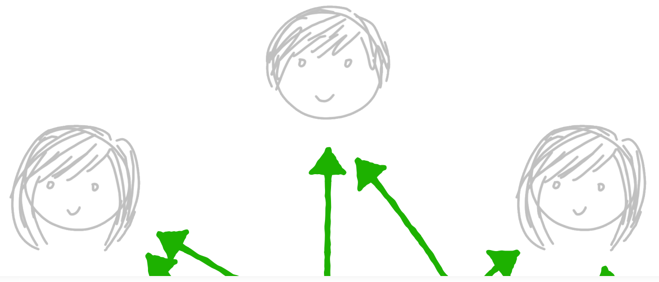
Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

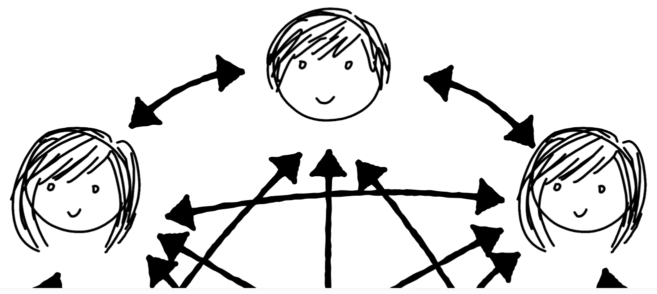
Each hybrid distribution makes *one* change.

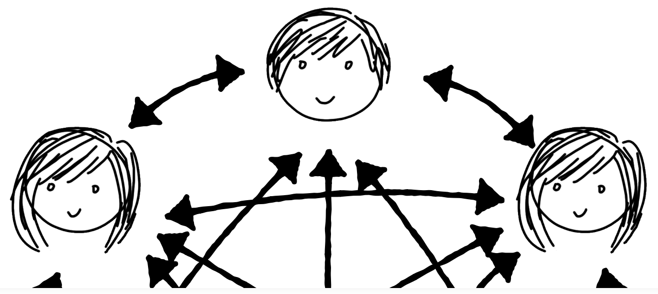
Hybrid Dist. $\mathscr{H}_2$

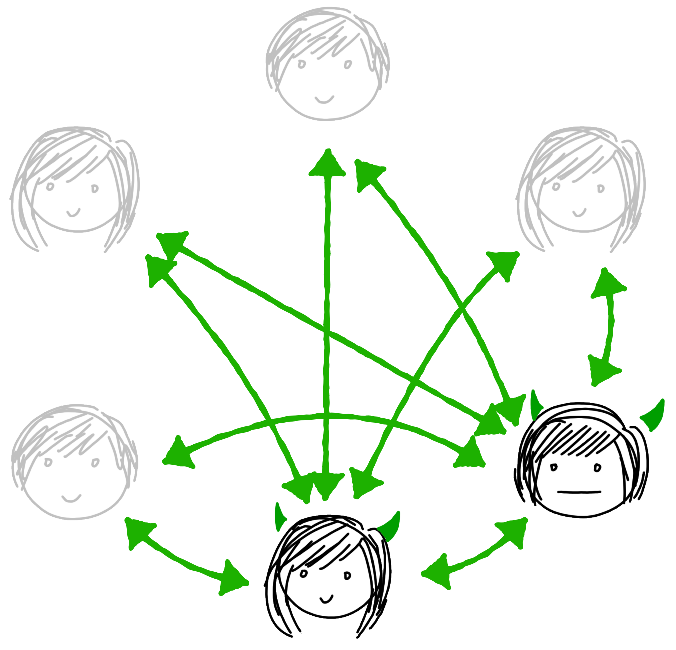


Round 1 of π_{outer}









Use $\mathscr{F}_{\text{inner}}$ to compute outputs and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

We can write another security reduction, but this time we will use any distinguisher $\mathscr{D}$ that can tell $\mathscr{H}_2$ apart from $\text{Sim}_{\text{composed}}$ to build a distinguisher $\mathscr{D}'$ that contradicts the lemma that π_{outer} realizes $\mathscr{F}_{\text{outer}}$.

By contraposition we conclude that $\mathscr{H}_2 \equiv (\text{Sim}_{\text{composed}}, \mathscr{F}_{\text{outer}})$.

Finally, by the transitivity of $\equiv$ we can see that the view of $\mathscr{A}$ in $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is identically distributed to the view produced by $\mathscr{S}_{\text{composed}}$ in an interaction with $\mathscr{F}_{\text{outer}}$.

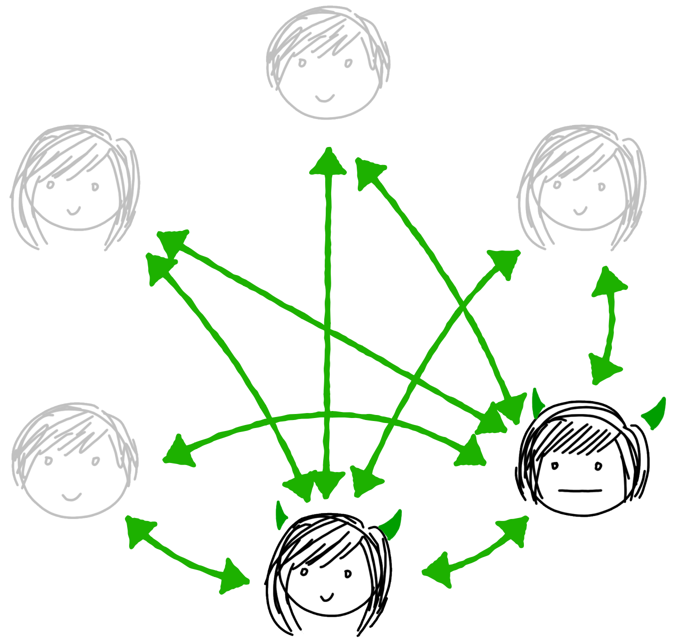
This holds for every $\mathscr{A}$ and $\mathscr{D}$, so we conclude that $\pi_{\text{outer}}^{\mathscr{F}_{\text{inner}} \to \pi_{\text{inner}}}$ realizes $\mathscr{F}_{\text{outer}}$. ∎

Ideal World

$\mathscr{S}_{\text{composed}}$

$\text{Sim}_{\text{composed}}$

$\mathscr{F}_{\text{outer}}$

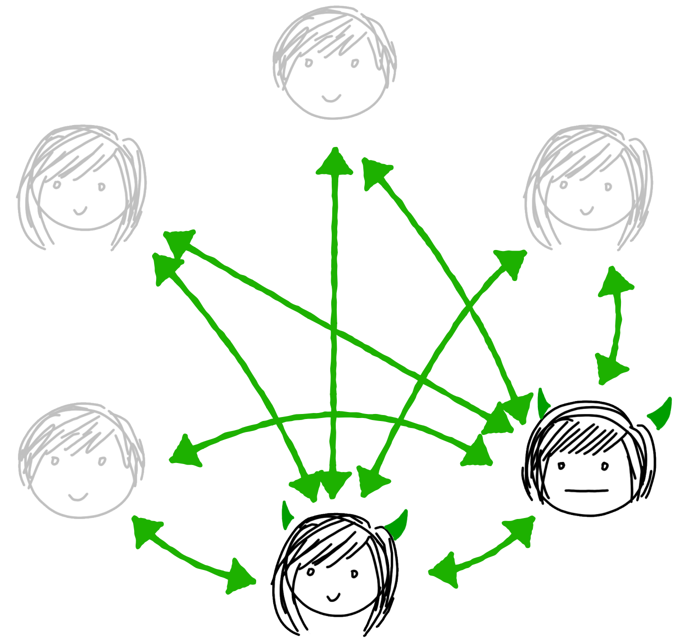


Use $\text{Sim}_{\text{outer}}$ to simulate round 1 of π_{outer}

Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Use $\text{Sim}_{\text{outer}}$ to simulate round 3 of π_{outer}

Use $\text{Sim}_{\text{outer}}$ to simulate round 4 of π_{outer}



Use $\text{Sim}_{\text{outer}}$ to simulate corrupt IO of π_{outer} and $\text{Sim}_{\text{inner}}$ to simulate all rounds of π_{inner}

Transitivity gives us our conclusion!

Take a deep breath.

- I know this was way too much to absorb!
- Translating all these pictures into math would hard for anyone (even a cryptography professor!)
- To make it worse, I played fast and loose with notation.
- Nevertheless I want you to get a sense for how composition works, because it's really interesting!
- There won't be any homework or tests on this :)
- If you want to see a formal version (for a much more restricted class of protocols), come to grad crypto!

The Theorem (again again)

Perfect MPC Composition Theorem: Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that perfectly realizes $\mathcal{F}_{\text{inner}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

This Composition Theorem is Very Limited.

- Only works for protocols that are organized into synchronous rounds, where every round contains communication or invocation of exactly one functionality. No concurrency is allowed. No asynchrony is allowed. The real world is asynchronous and concurrent, though…

- Requires protocols to be *perfectly* secure. Very few deployed protocols are perfectly secure…

- Only works for static, semi-honest adversaries. Real world adversaries could very well cheat…

- This isn't very realistic…

These restrictions can be relaxed… with work.

Universally Composable Security:
A New Paradigm for Cryptographic Protocols*

Ran Canetti†

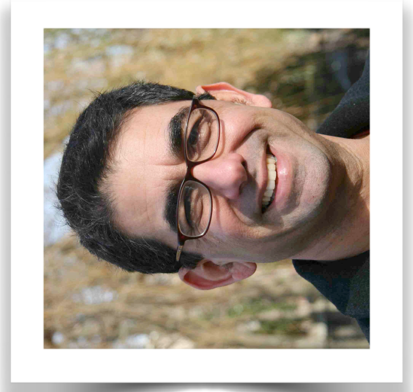
February 11, 2020

Abstract

We present a general framework for describing cryptographic protocols and analyzing their security. The framework allows specifying the security requirements of practically any cryptographic task in a unified and systematic way. Furthermore, in this framework the security of protocols is preserved under a general composition operation, called universal composition.

The proposed framework with its security-preserving composition operation allows for modular design and analysis of complex cryptographic protocols from simpler building blocks. Moreover, within this framework, protocols are guaranteed to maintain their security in any context, even in the presence of an unbounded number of arbitrary protocol sessions that run concurrently in an adversarially controlled manner. This is a useful guarantee, which allows arguing about the security of cryptographic protocols in complex and unpredictable environments such as modern communication networks.

Keywords: cryptographic protocols, security analysis, protocol composition, universal composition.



- *General* composition theorems are hard.
  There have been a handful of attempts.
- This framework/theorem is probably the most popular one. It was originally introduced in 2001. It was finally published in a journal in 2020.
- Those 19 years were spent finding and fixing bugs, and handling new use-cases. The best way to find both is by using it!
- People are still finding and fixing bugs in this framework today! *Why is it so hard?*
- The model description and proofs are *huge*!

These restrictions can be relaxed… with work.

(And it used to be almost twice as long as this!)

The story of protocol composition hasn’t been finished yet. There are still many open questions - and you could help solve them!

In case you are lost in the weeds:

Perfect MPC Feasibility Theorem: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels there exists an n-party protocol that perfectly realizes $\mathcal{F}_{\mathsf{SFE}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

Proof: Direct corollary of Lemmas 1 and 2 and the Composition Theorem. ∎

You just proved that *any* function can be securely computed! Congratulations!

Hopefully some of you are objecting…

"We didn't actually prove the composition theorem.
You just showed us pictures!"

In this case, you can easily *inline* the GRR multiplication protocol and proof into the BGW protocol and proof.

This kind of inlining will be much more difficult in the future.

CS4501 Cryptographic Protocols Lecture 10: GRR Example, Composition in a Perfect World

https://jackdoerner.net/teaching/#2026/Spring/CS4501