Securing Computation Workshop, Simons Institute Berkeley, June 12, 2015

How Fair is Your Protocol?

A Utility-based Approach to Protocol Optimality

Juan Garay (Yahoo Labs)
Jonathan Katz (UMD)
Björn Tackmann (UCSD)
Vassilis Zikas (ETH Zürich)

[PODC 2015]

Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.

Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.
2. They toss a coin $i^* \in \{1,2\}$.

Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.
2. They toss a coin $i^* \in \{1,2\}$.
3. $p_{(3-i^*)}$ opens its commitment to p_{i^*}

Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.
2. They toss a coin $i^* \in \{1,2\}$.
3. $p_{(3-i^*)}$ opens its commitment to p_{i^*}
4. p_{i^*} opens its commitment to $p_{(3-i^*)}$

Two Coin-Toss Protocols

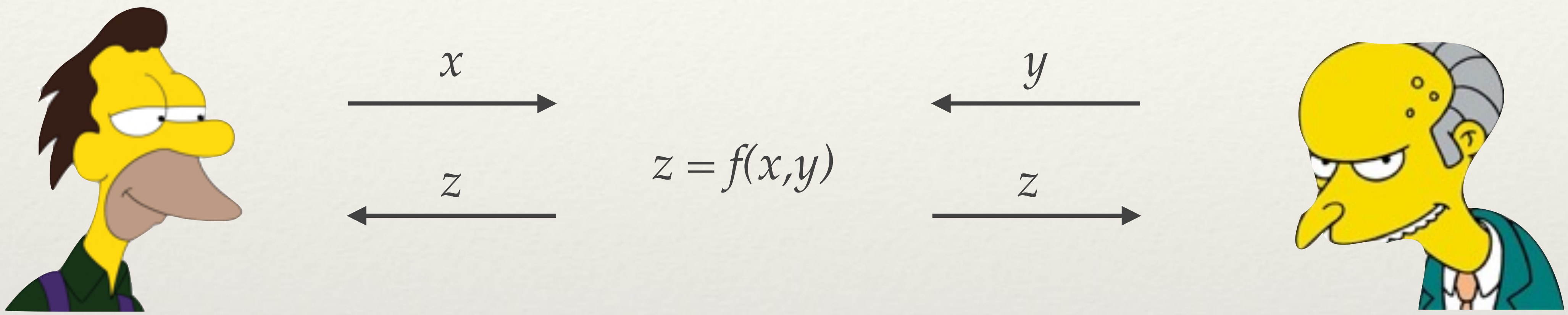
Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.
2. They toss a coin $i^* \in \{1,2\}$.
3. $p_{(3-i^*)}$ opens its commitment to p_{i^*}
4. p_{i^*} opens its commitment to $p_{(3-i^*)}$
5. The result is the XOR.

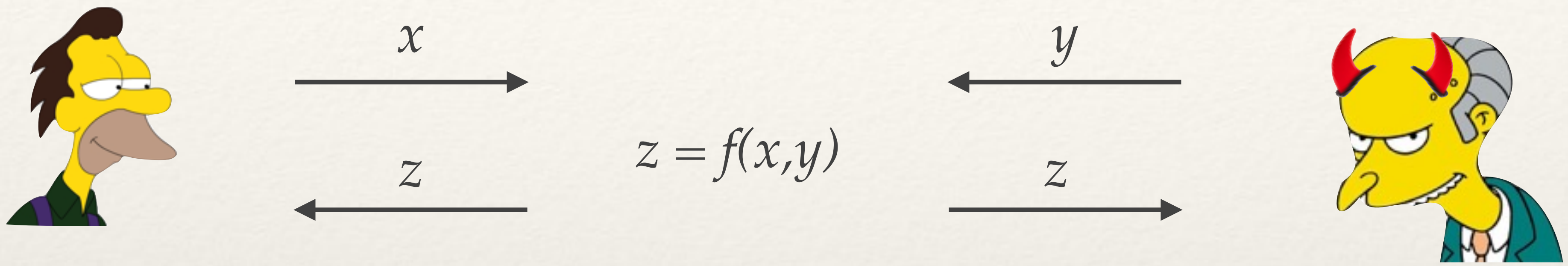
Fairness in SFE



Fairness:

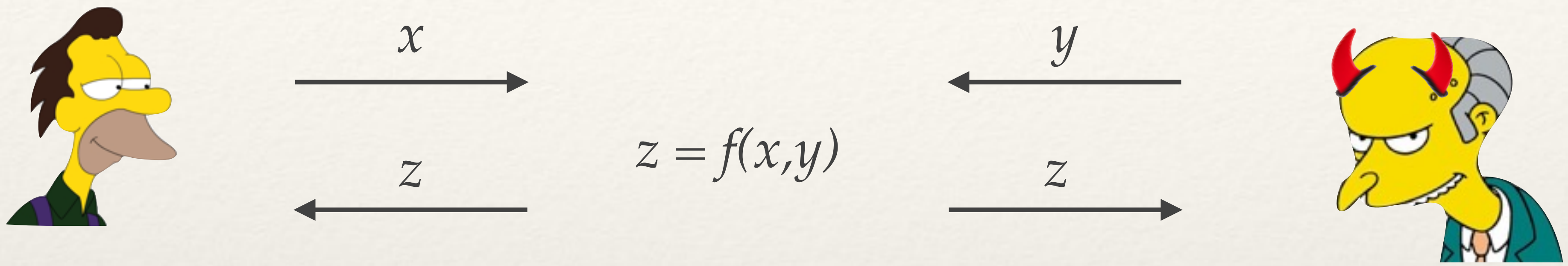
- "if one party learns the output, the other party also learns it,"
- generally impossible in 2PC [Cleve, STOC'86].

Fairness in SFE



	does not get z		does not get z
	does not get z		gets z
	gets z		does not get z
	gets z		gets z

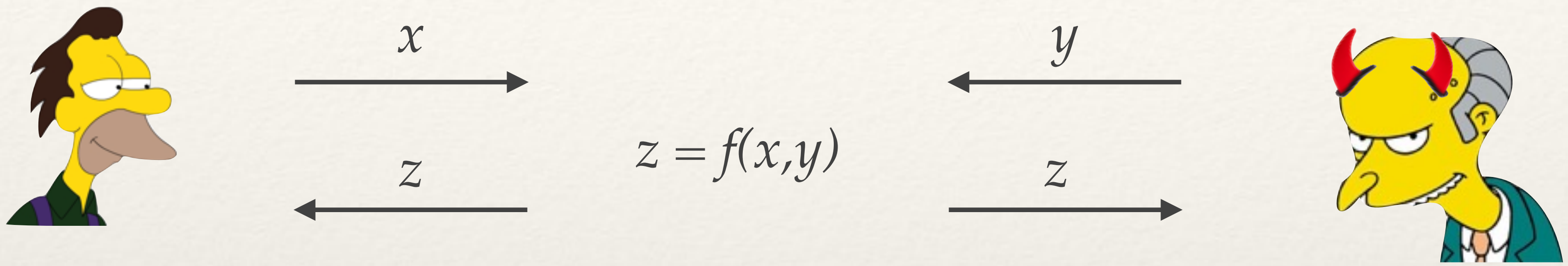
Fairness in SFE



Possible outcomes (intuitively):

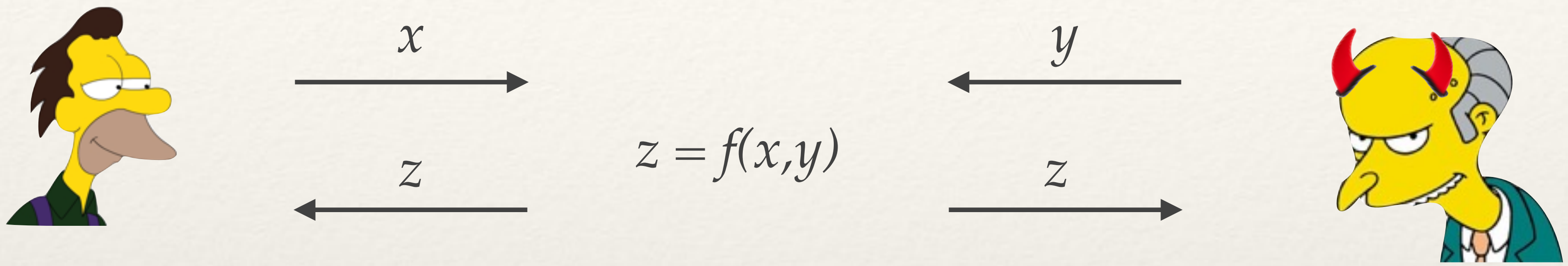
	does not get z		does not get z
	does not get z		gets z
	gets z		does not get z
	gets z		gets z

Fairness in SFE



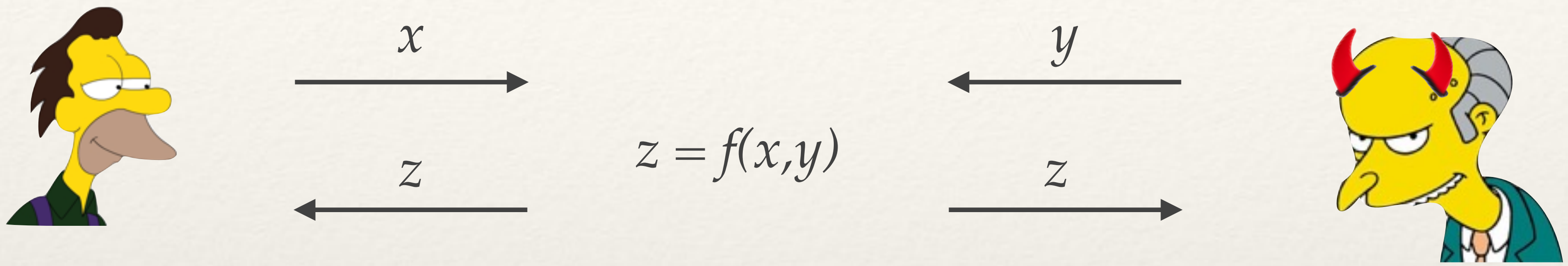
Possible outcomes (intuitively):				"Fairness"
	does not get z		does not get z	good
	does not get z		gets z	bad
	gets z		does not get z	good
	gets z		gets z	good

Fairness in SFE



Possible outcomes (intuitively):				"Fairness"	Utility
	does not get z		does not get z	good	γ_{00}
	does not get z		gets z	bad	γ_{10}
	gets z		does not get z	good	γ_{01}
	gets z		gets z	good	γ_{11}

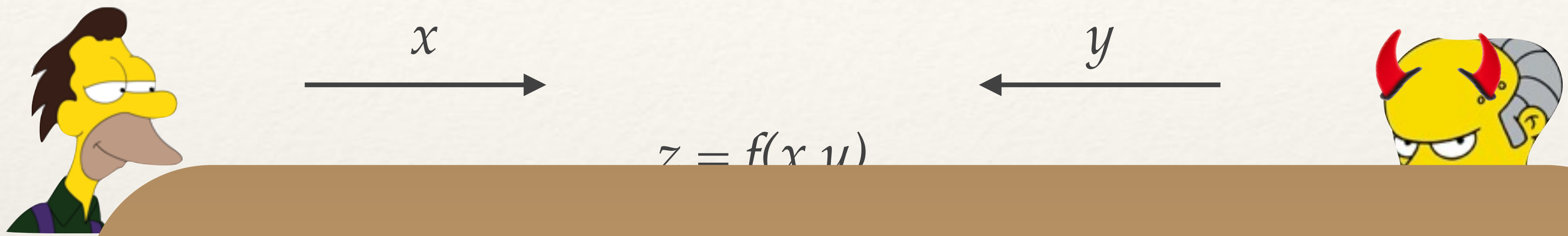
Fairness in SFE



Possible outcomes (intuitively):				"Fairness"	Utility
	does not get z		does not get z	good	γ_{00}
	does not get z		gets z	bad	γ_{10}
	gets z		does not get z	good	γ_{01}
	gets z		gets z	good	γ_{11}

Natural conditions: $\gamma_{01} < \gamma_{00}, \gamma_{11}$ and $\gamma_{00}, \gamma_{11} < \gamma_{10}$

Fairness in SFE



Protocol comparison and optimality:

- the utilities for the individual outcomes define an expected payoff for each adversarial strategy,
- a protocol is *better* (fairer) if the expected payoff of the *best* adversarial strategy is smaller.

	Utility
	γ_{00}
	γ_{10}
	γ_{01}
	γ_{11}

Natural conditions: $\gamma_{01} < \gamma_{00}, \gamma_{11}$ and $\gamma_{00}, \gamma_{11} < \gamma_{10}$

Other Relaxed Notions of Fairness

- "Gradual Release"-type approaches [Goldwasser-Levin, 1990; Garay-MacKenzie-Prabhakaran-Yang, 2005; …]
- Rational fairness [Asharov-Canetti-Hazay, 2011]
- *1/p*-Security [Gordon-Katz, 2010; …]

Rational Protocol Design

Protocol π →

← Adversary strategy for π

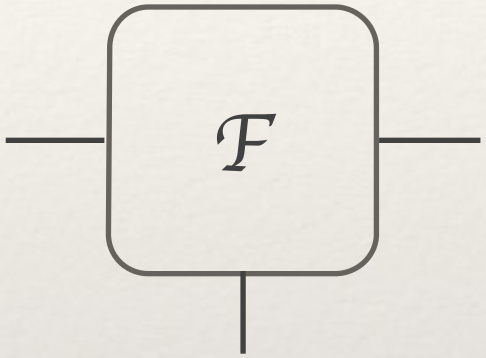
Protocol Designer

Attacker

- Two-move "meta" game,
- zero-sum: $u_D = -u_A$,
- ε-subgame-perfect equilibrium.

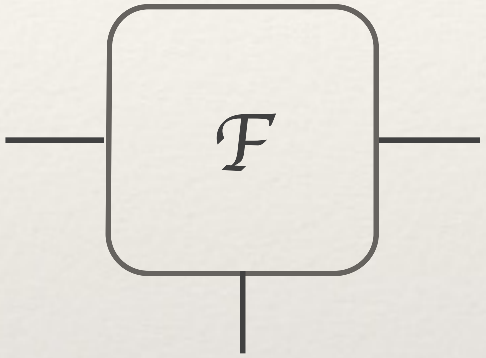
[Garay-Katz-Maurer-T-Zikas, FOCS 2013]

Rational Protocol Design



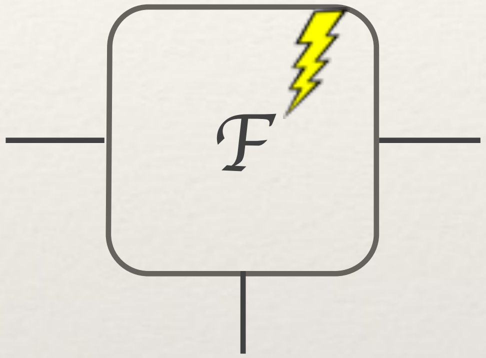
Rational Protocol Design

Step 1: Relax functionality



Rational Protocol Design

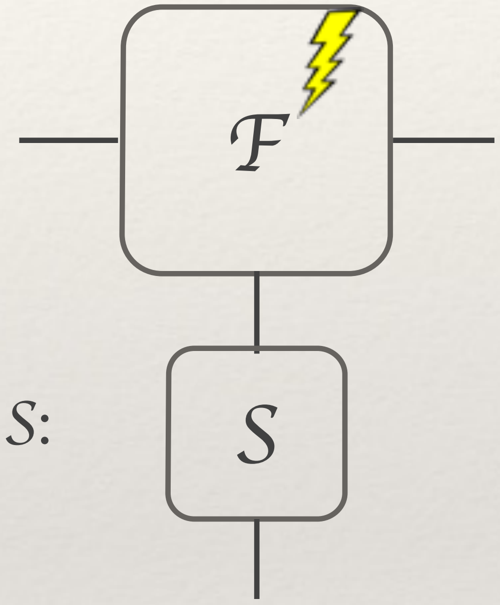
Step 1: Relax functionality



Rational Protocol Design

Step 1: Relax functionality

s.t. ∃ $\mathcal{S}$:

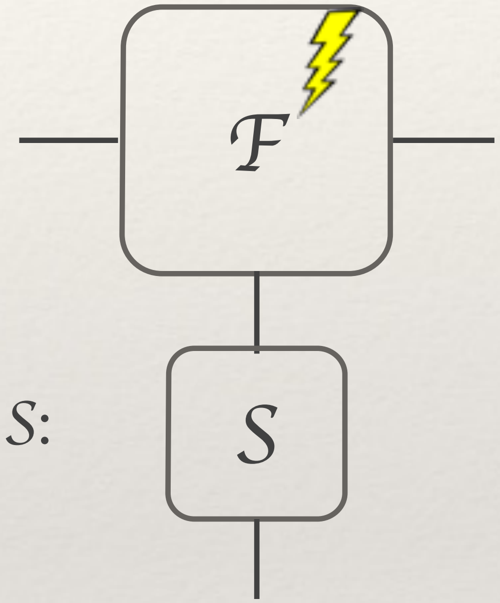


Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

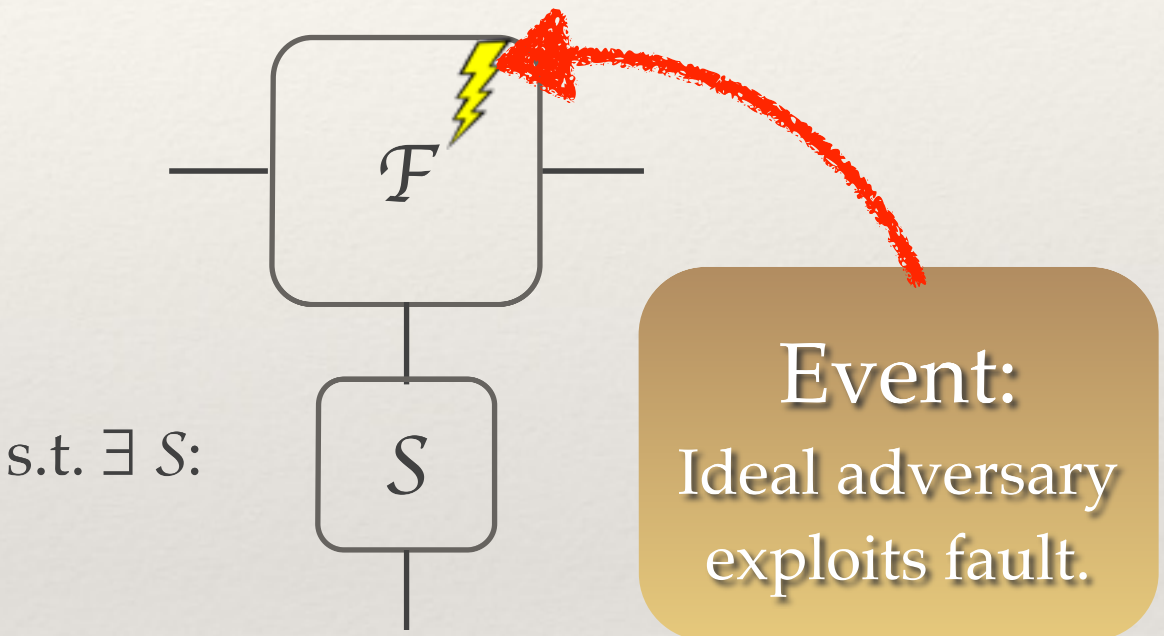
s.t. $\exists\ \mathcal{S}$:



Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

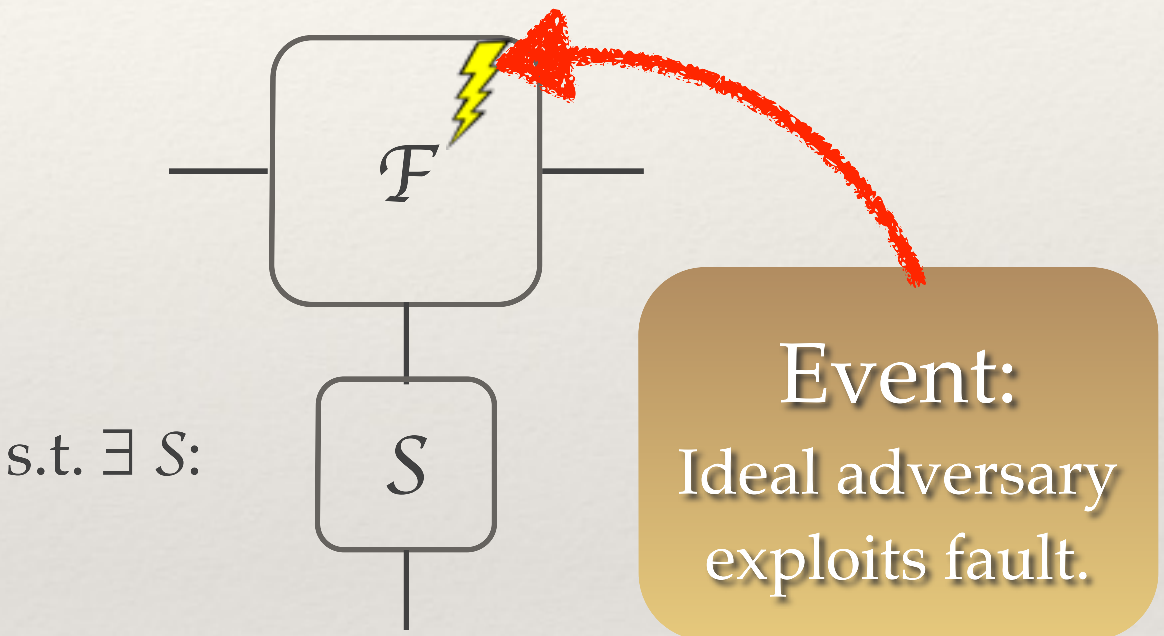


Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

Step 3: Define payoff

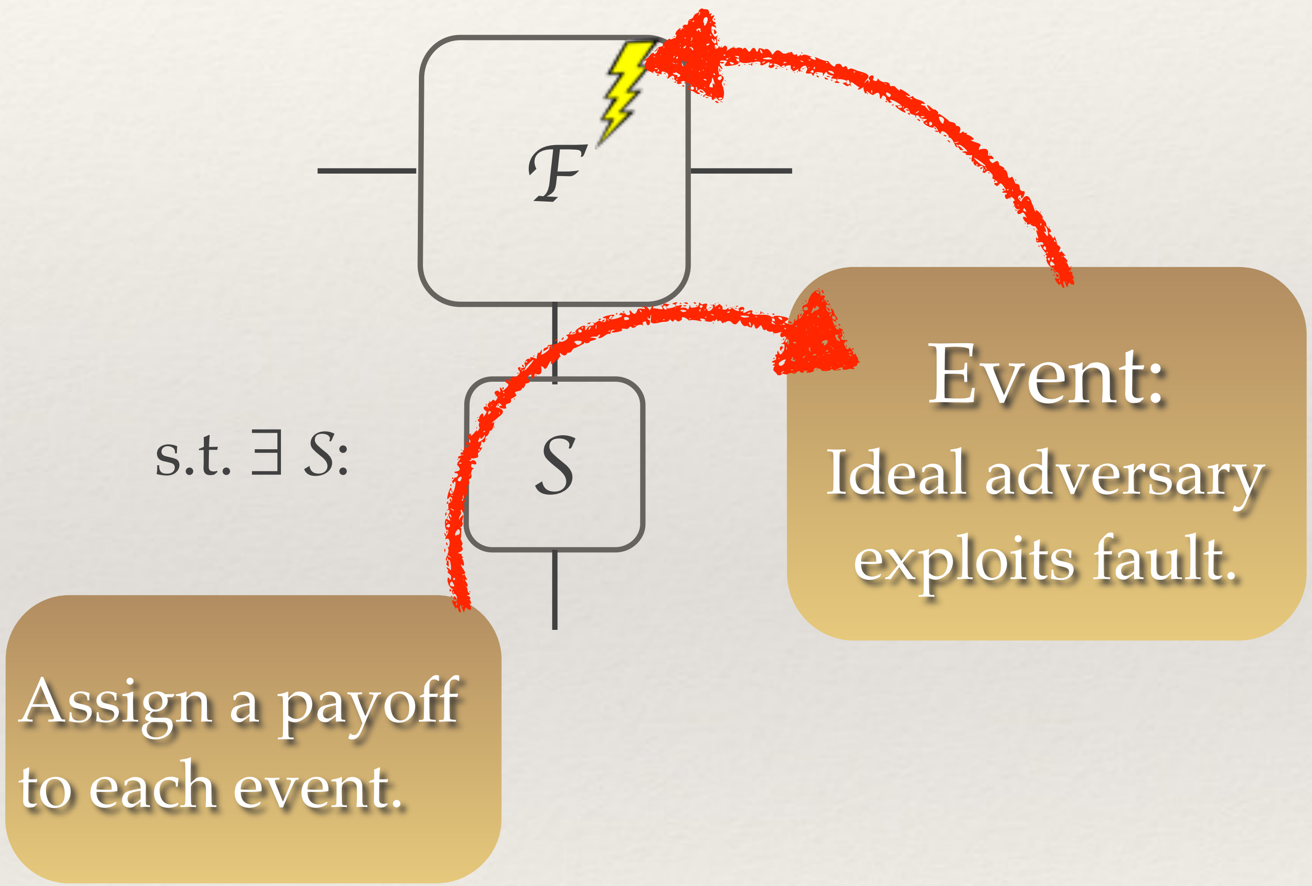


Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

Step 3: Define payoff

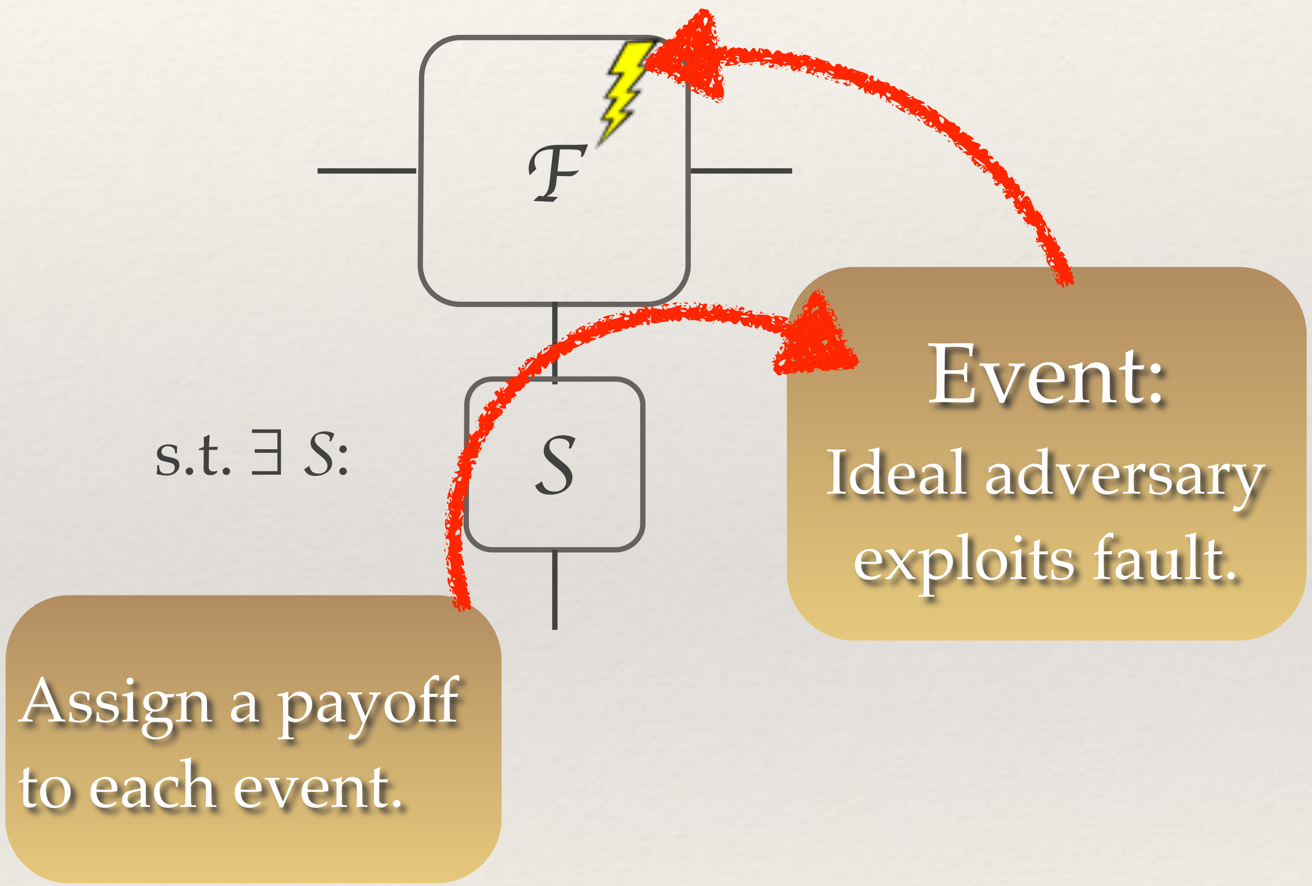


Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

Step 3: Define payoff



$$\text{Payoff}(\mathcal{A}) = \min_{\text{"good" } \mathcal{S}} \text{payoff}(\mathcal{S})$$

Defining Fairness (1)

x →

← z

$\mathcal{F}_{\text{fair-sfe}}$

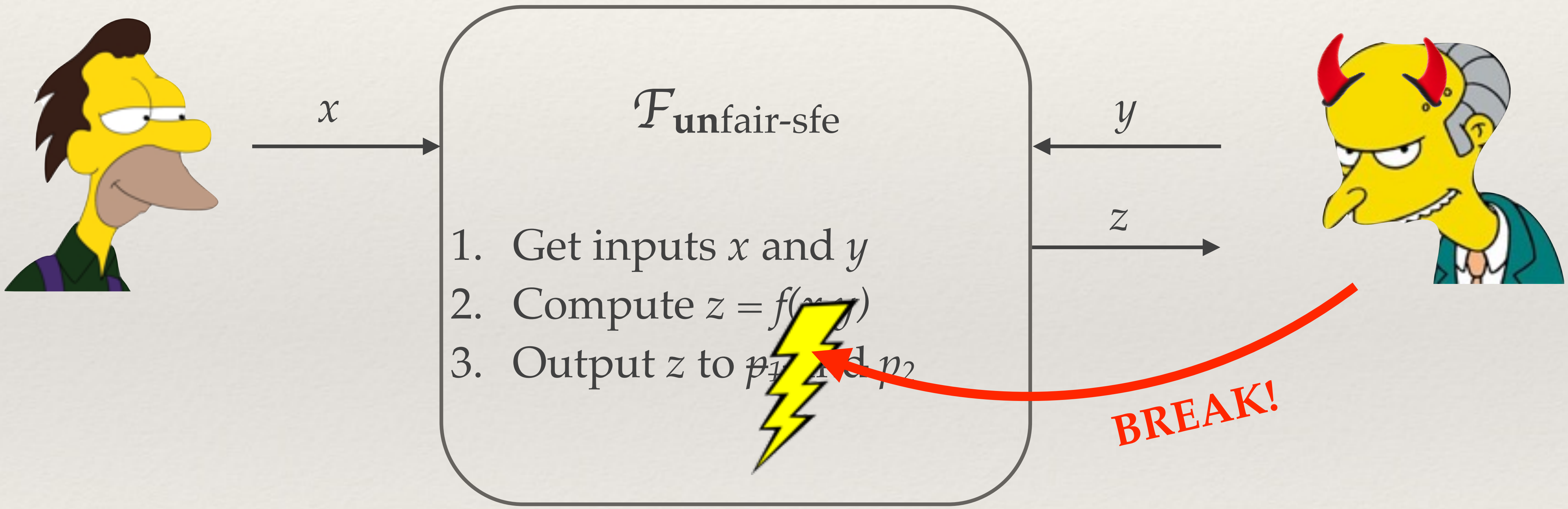
1. Get inputs x and y
2. Compute $z = f(x,y)$
3. Possibly: Output z to p_1 and p_2

← y

z →

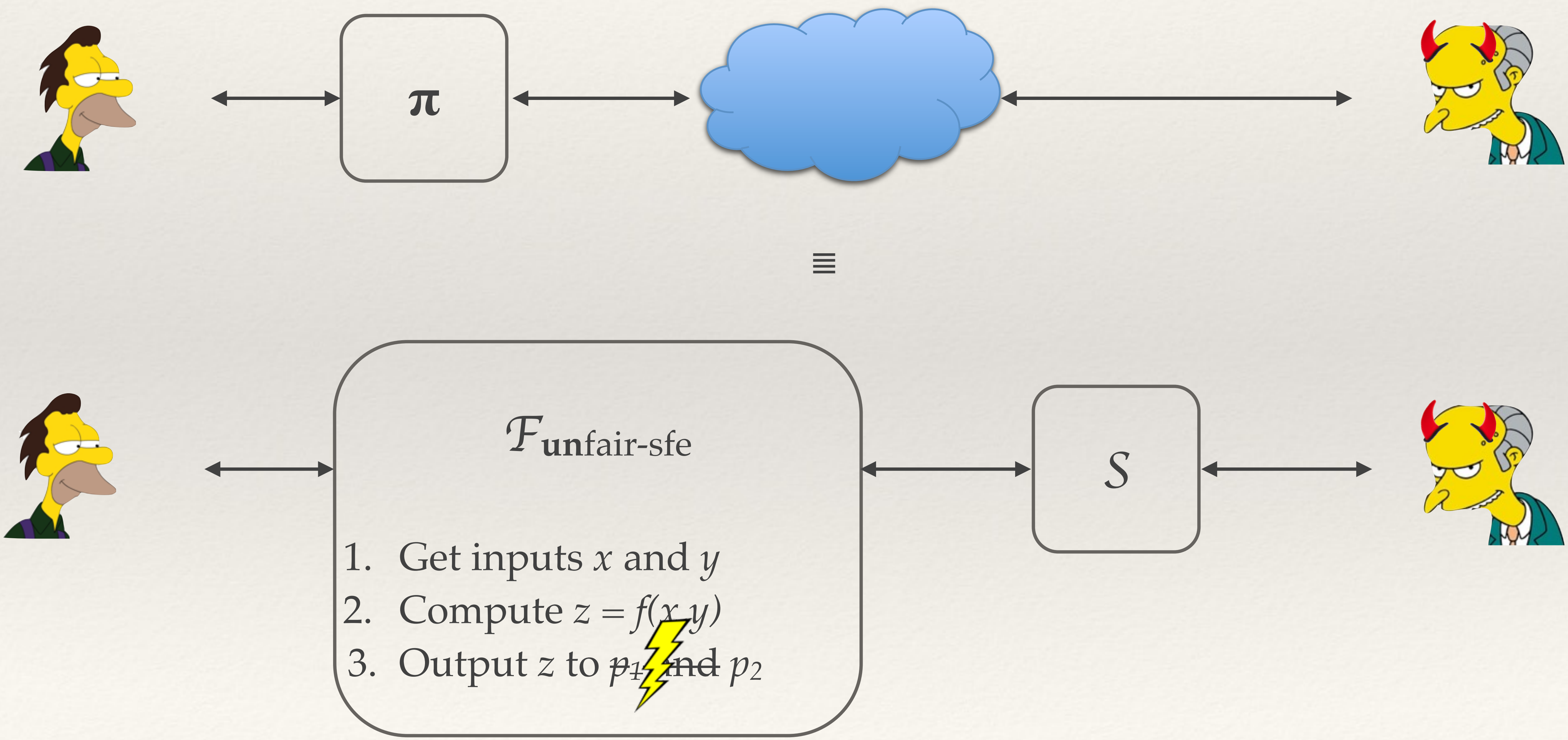
Defining Fairness (2)

Step 1:



Defining Fairness (3)

The protocol $\boldsymbol{\pi}$ realizes $\mathcal{F}_{\textbf{un}\text{fair-sfe}}$, i.e., there is S:



Defining Fairness (4)

Step 2: Define events in the *ideal* execution:

(a) Neither party gets the output: E_{00}, payoff γ_{00}

(b) Only honest party gets the output: E_{01}, payoff γ_{01}

(c) Only corrupted party gets the output: E_{10}, payoff γ_{10}

(d) Both parties get the output: E_{11}, payoff γ_{11}

Natural conditions: $\gamma_{01} < \gamma_{00}, \gamma_{11}$ and $\gamma_{00}, \gamma_{11} < \gamma_{10}$

Defining Fairness (5)

Step 3: Define the expected payoff for each S:

$$\text{payoff}(S) = \sum_{i,j \in \{0,1\}} \Pr(E_{ij}) \cdot \gamma_{ij}$$

The payoff of an adversary is the expected payoff of the *best* simulator:

$$\text{Payoff}(\mathcal{A}) = \min_{\text{"good" } S} \text{payoff}(S)$$

Optimal Protocol for Two Party SFE

- The protocol achieves $\frac{\gamma_{10} + \gamma_{11}}{2}$.
- This is optimal (see next slide).

1. In an *unfair* SFE:
   (a) choose $i^* \in \{1,2\}$
   (b) compute a sharing of the output value
   (c) output i^* and one share to each party
2. in case of abort, restart with default input for other party
3. $p_{(3-i^*)}$ sends its share to p_{i^*}
4. p_{i^*} sends its share to $p_{(3-i^*)}$

Optimal Protocol for Two Party SFE

- The protocol achieves $\frac{\gamma_{10} + \gamma_{11}}{2}$.
- This is optimal (see next slide).

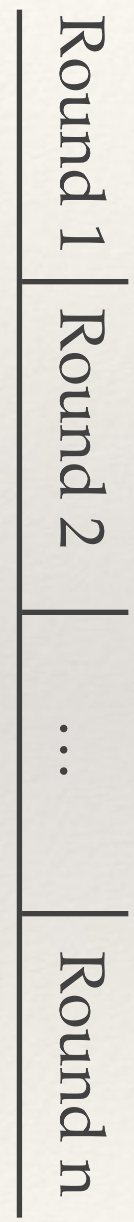
Proof idea:

- secure w/o fairness (based on underlying SFE and repeat before leaking output)
- the simulator chooses i^* uniformly at random

1. In an *unfair* SFE:
   (a) choose $i^* \in \{1,2\}$
   (b) compute a sharing of the output value
   (c) output i^* and one share to each party
2. in case of abort, restart with default input for other party
3. $p_{(3-i^*)}$ sends its share to p_{i^*}
4. p_{i^*} sends its share to $p_{(3-i^*)}$

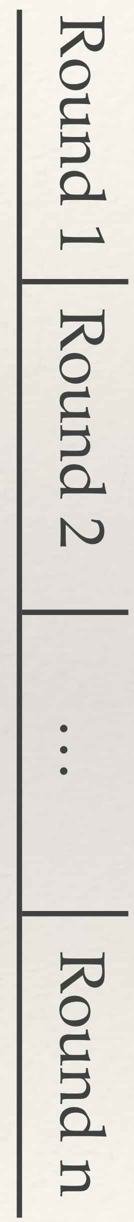
Optimal Protocol for Two Party SFE

There exist functions such that…



Optimal Protocol for Two Party SFE

There exist functions such that…



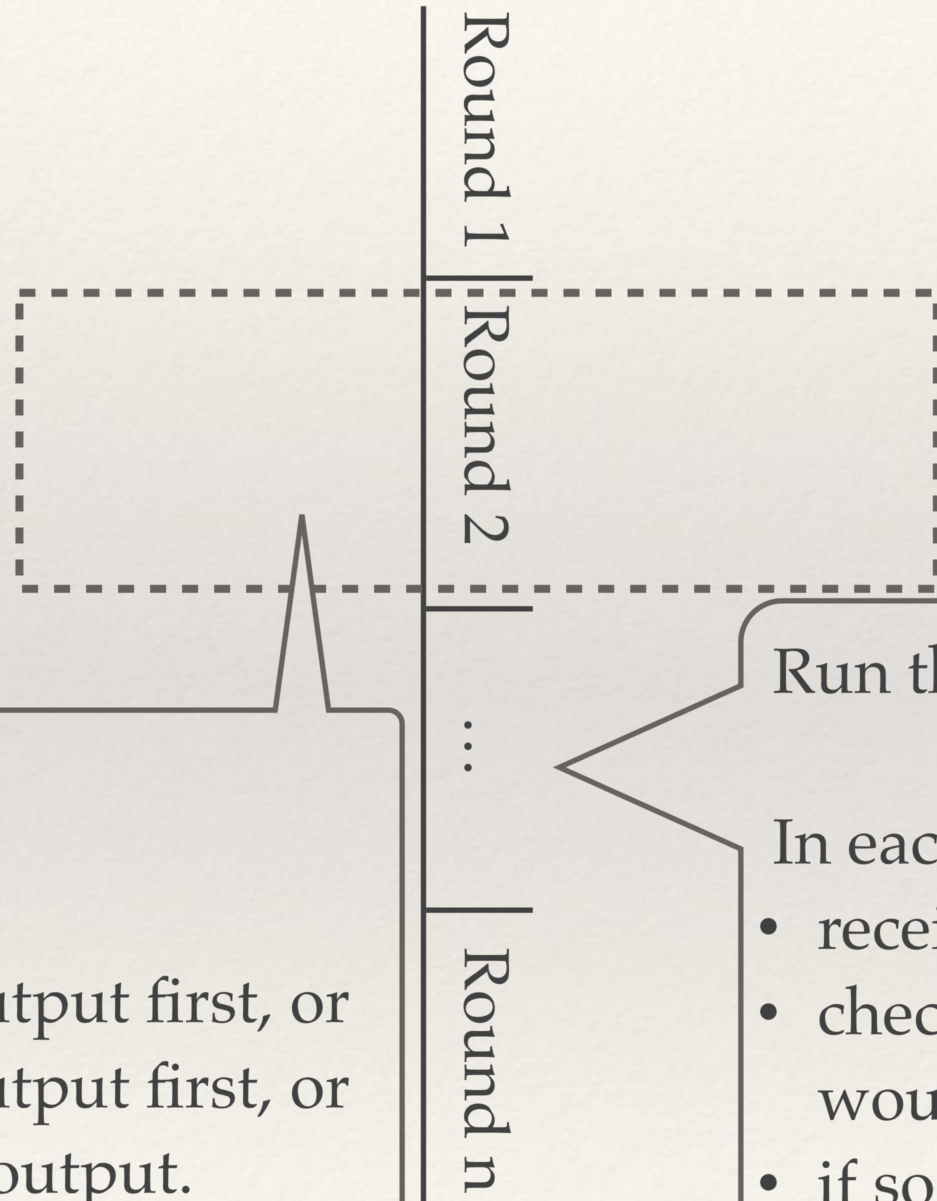
Run the honest protocol as follows.

In each round:

- receive the honest party's message,
- check whether the honest protocol would generate output,
- if so, then abort,
- otherwise, send the honestly computed message for this round

Optimal Protocol for Two Party SFE

There exist functions such that…



In each round:

- p_1 receives the output first, or
- p_2 receives the output first, or
- both receive the output.

Run the honest protocol as follows.

In each round:

- receive the honest party's message,
- check whether the honest protocol would generate output,
- if so, then abort,
- otherwise, send the honestly computed message for this round

Optimal Protocol for Two Party SFE

There exist functions such that…

Round 1

Proof idea:

- a protocol can be improved by never outputting to both in the same round
- at least one party is *first* with probability at least 1/2

In each round:

- p_1 receives the output first, or
- p_2 receives the output first, or
- both receive the output.

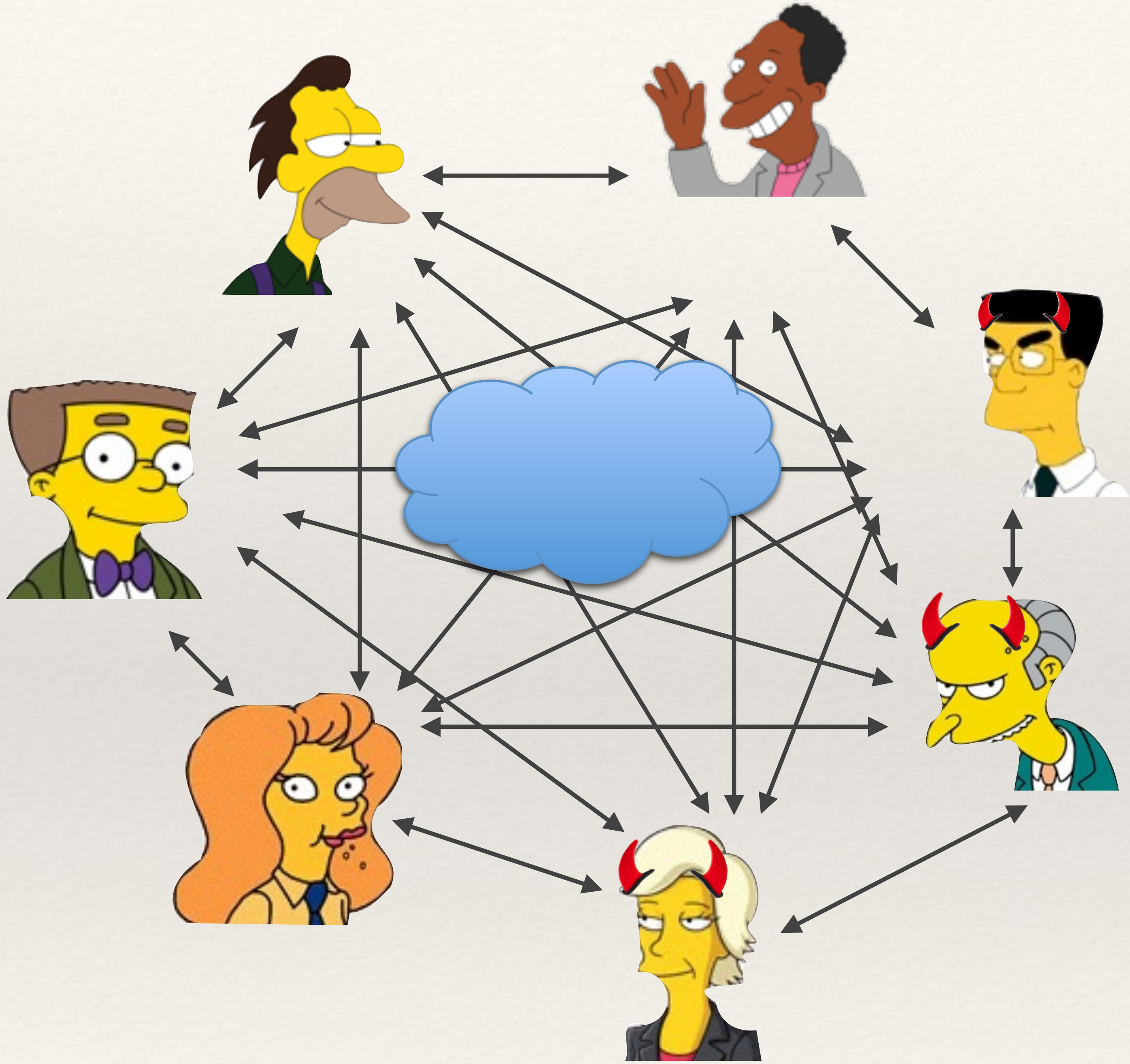
ound n

…ocol as follows.

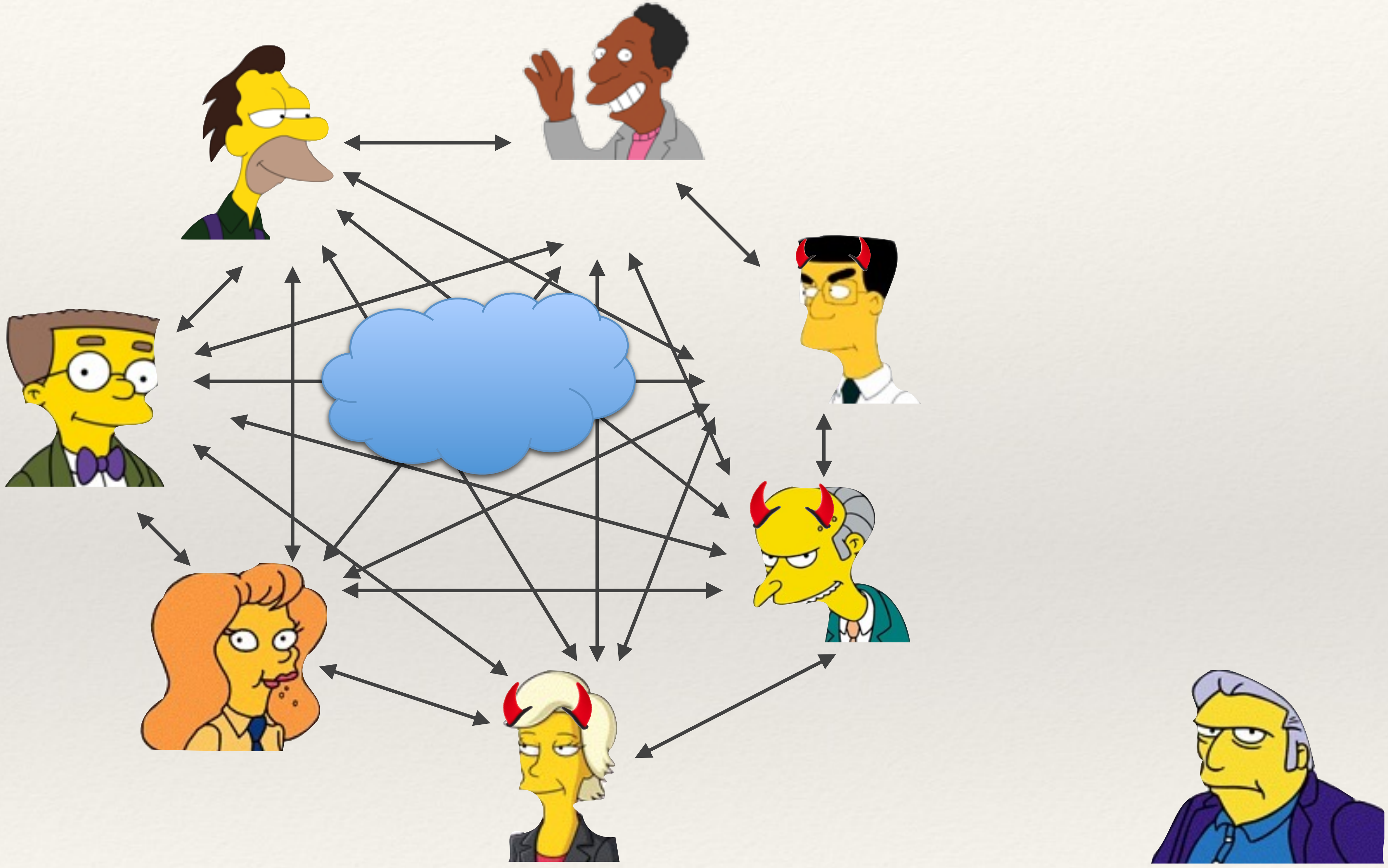
… party's message,

- check whether the honest protocol would generate output,
- if so, then abort,
- otherwise, send the honestly computed message for this round

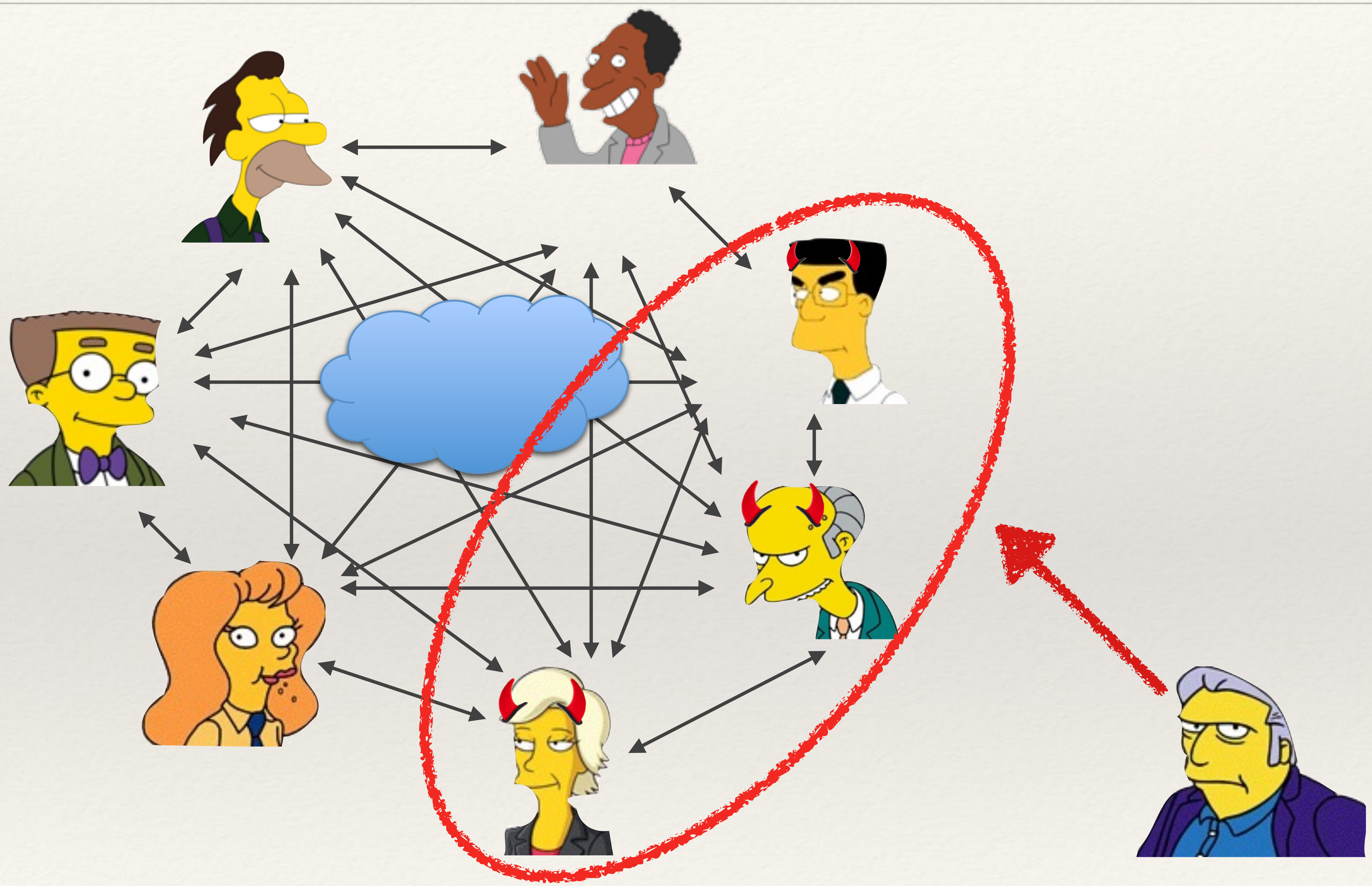
The Multi-Party Case



The Multi-Party Case



The Multi-Party Case



The Multi-Party Case

$\mathcal{F}_{\text{fair-sfe}}$

1. Get inputs $x_1, x_2, \ldots, x_n$
2. Compute $z = f(x_1, x_2, \ldots, x_n)$
3. Possibly: Output z to $p_1, p_2, \ldots, p_n$

The Multi-Party Case

$\mathcal{F}_{\text{fair-sfe}}$

1. Get inputs $x_1, x_2, \ldots, x_n$
2. Compute $z = f(x_1, x_2, \ldots, x_n)$
3. Possibly: Output z to $p_2, \ldots, p_n$

BREAK!

Give output only to me!

Multi-Party Fairness

Step 2: Define events in the *ideal* execution:

A. No party gets the output: E_{00}, payoff γ_{00}

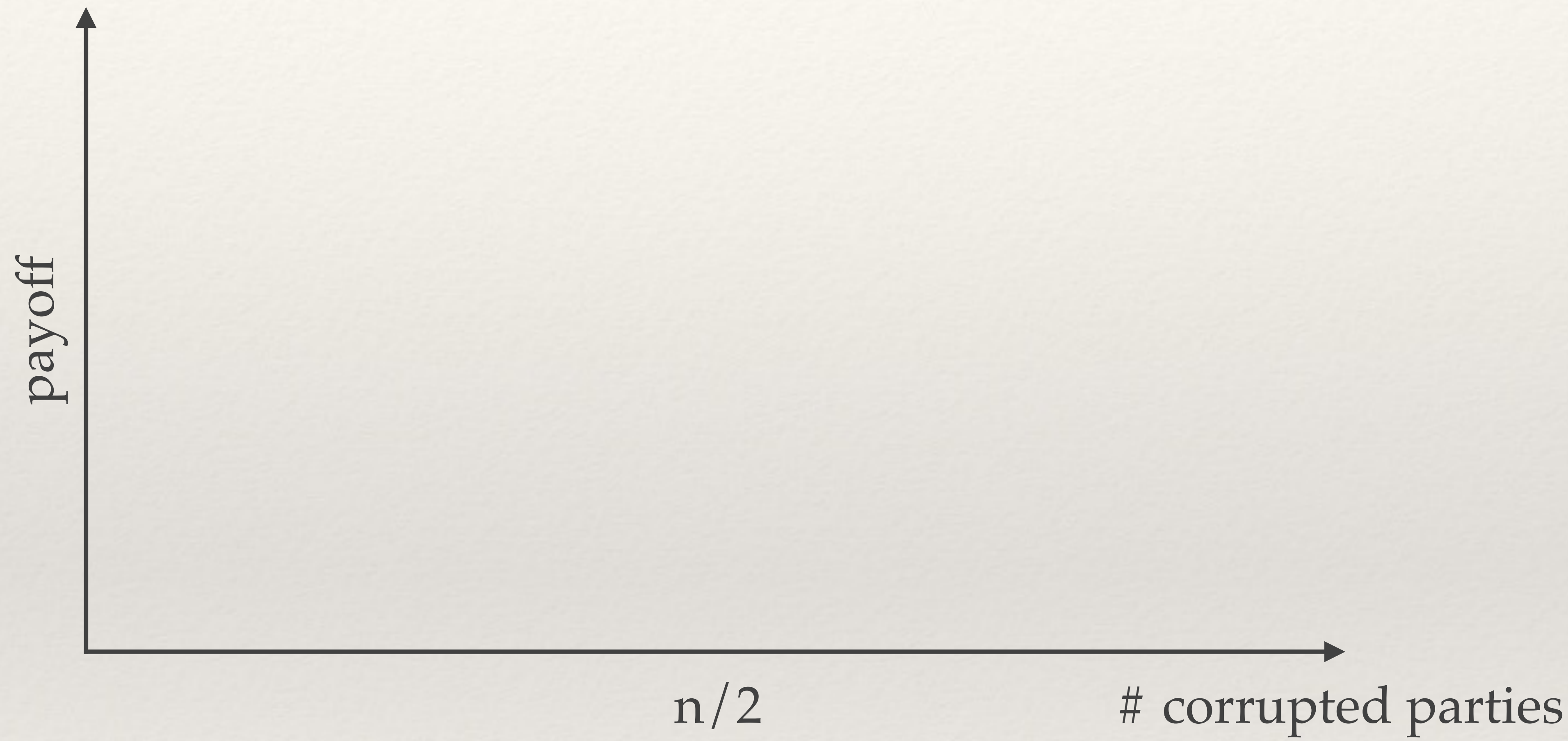
B. Exactly all honest parties get the output: E_{01}, payoff γ_{01}

C. Not all honest parties, but some corrupted party gets the output: E_{10}, payoff γ_{10}

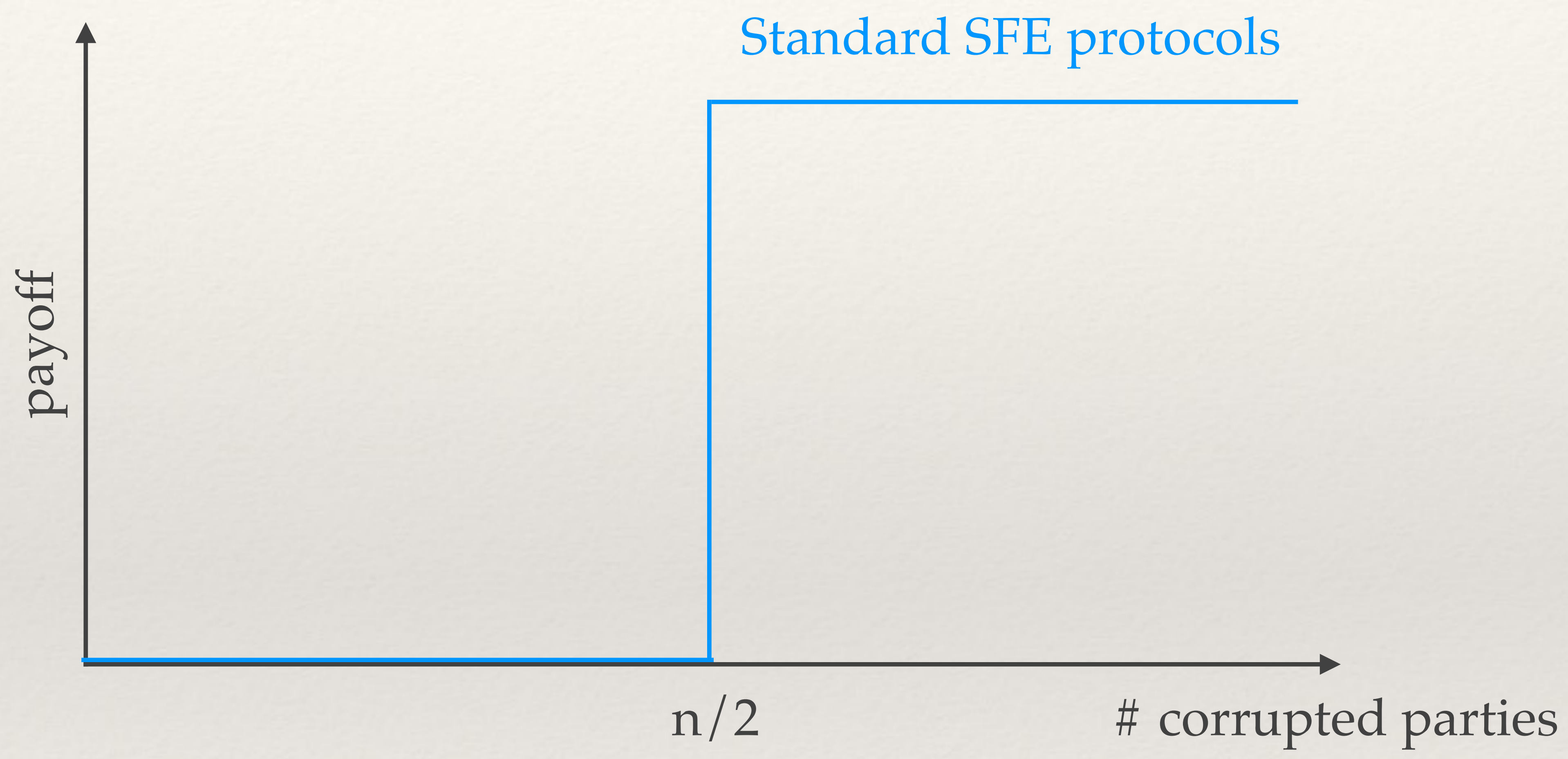
D. All honest parties and some corrupted party get the output: E_{11}, payoff γ_{11}

Here: stronger condition $\gamma_{01} < \gamma_{00} < \gamma_{11} < \gamma_{10}$.

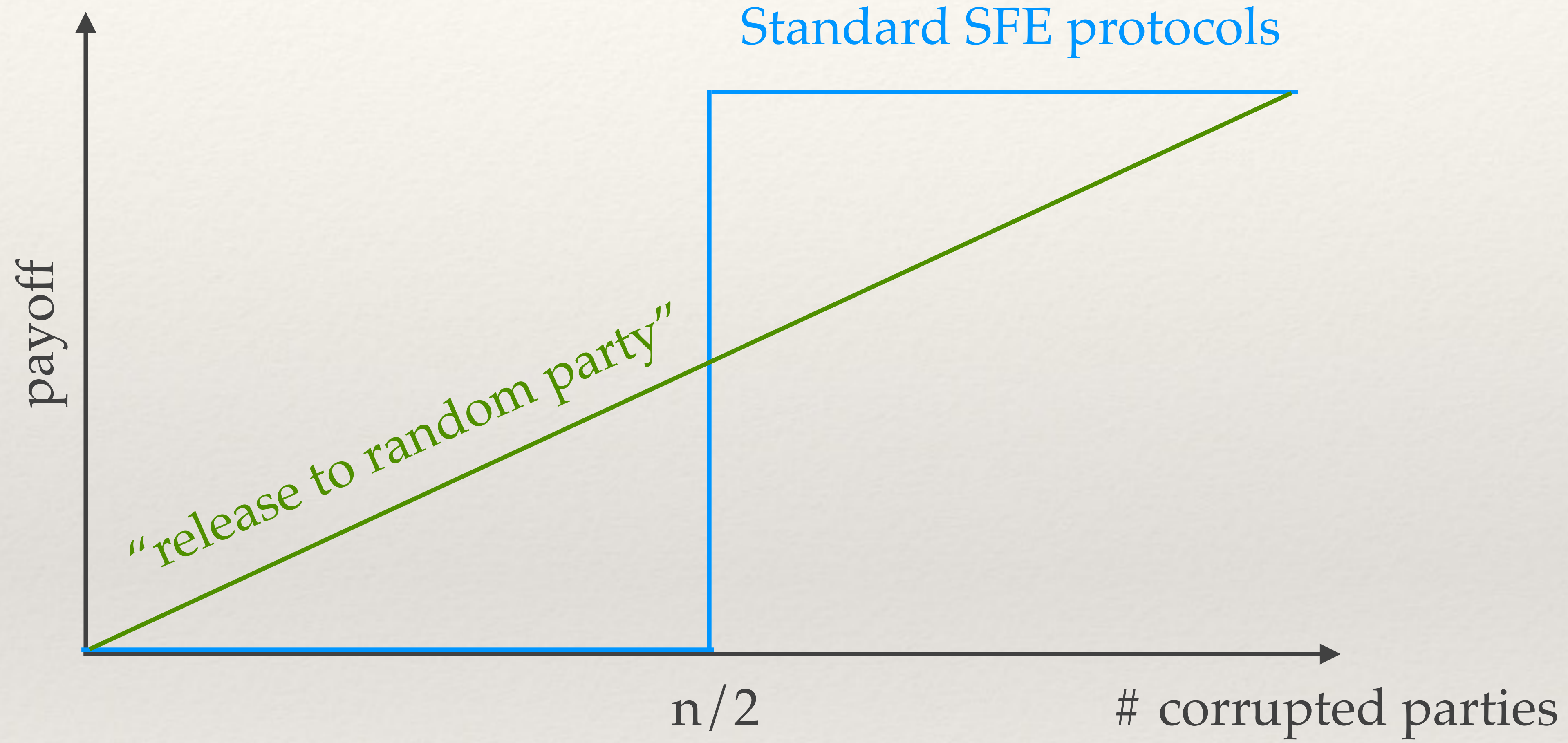
Multi-Party Fairness



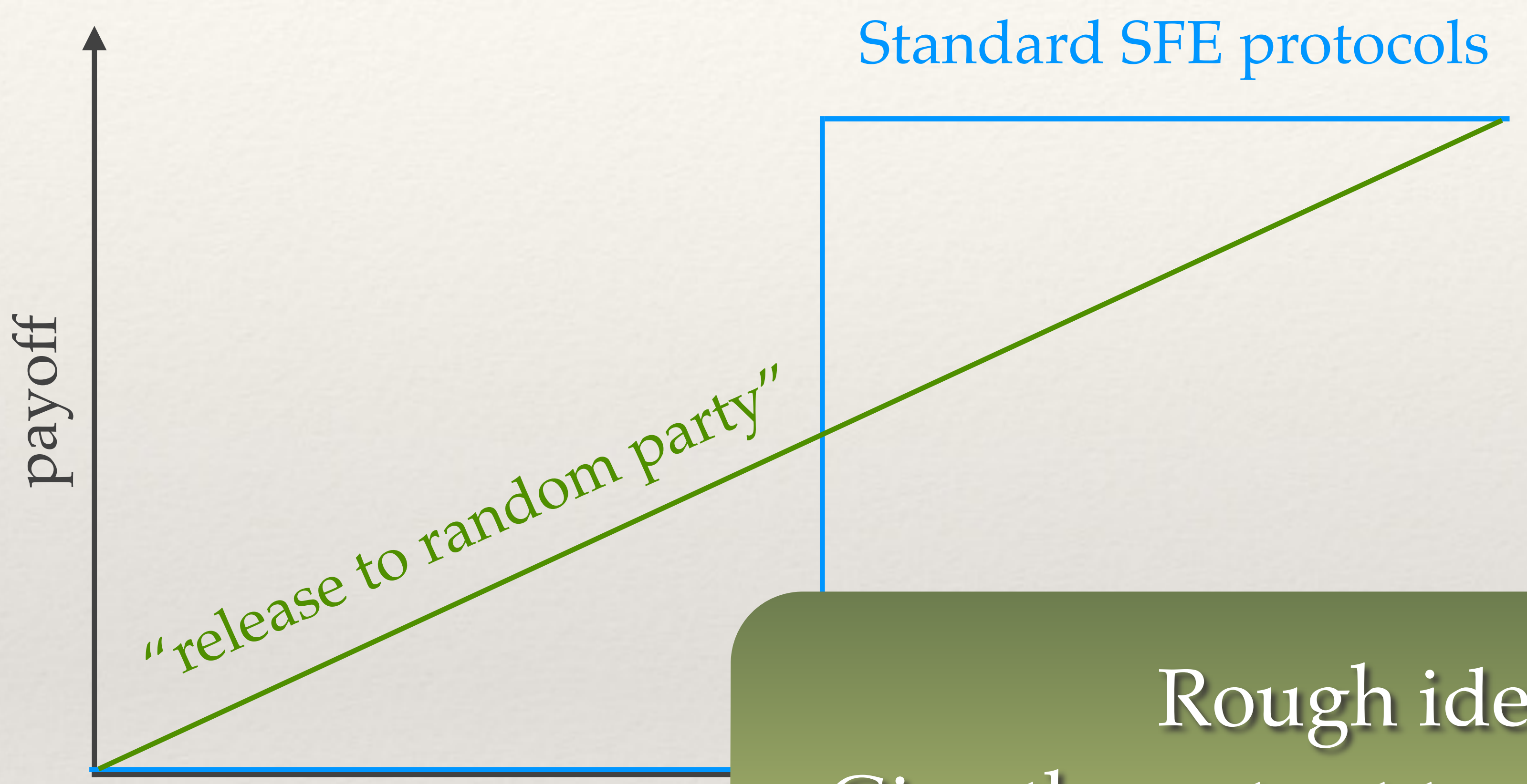
Multi-Party Fairness



Multi-Party Fairness



Multi-Party Fairness



Rough idea:
Give the output to some party,
let him distribute.

Other Relaxed Notions of Fairness

- "Gradual Release"-type approaches [Goldwasser-Levin, 1990; Garay-MacKenzie-Prabhakaran-Yang, 2005, …]
- Rational fairness [Asharov-Canetti-Hazay, 2011]
  - Not closely related, after all…
- *1/p*-Security [Gordon-Katz, 2010]
  - Similar (quantitative) guarantee,
  - protocols for functions with small domain or range,
  - formally more relaxed definition.

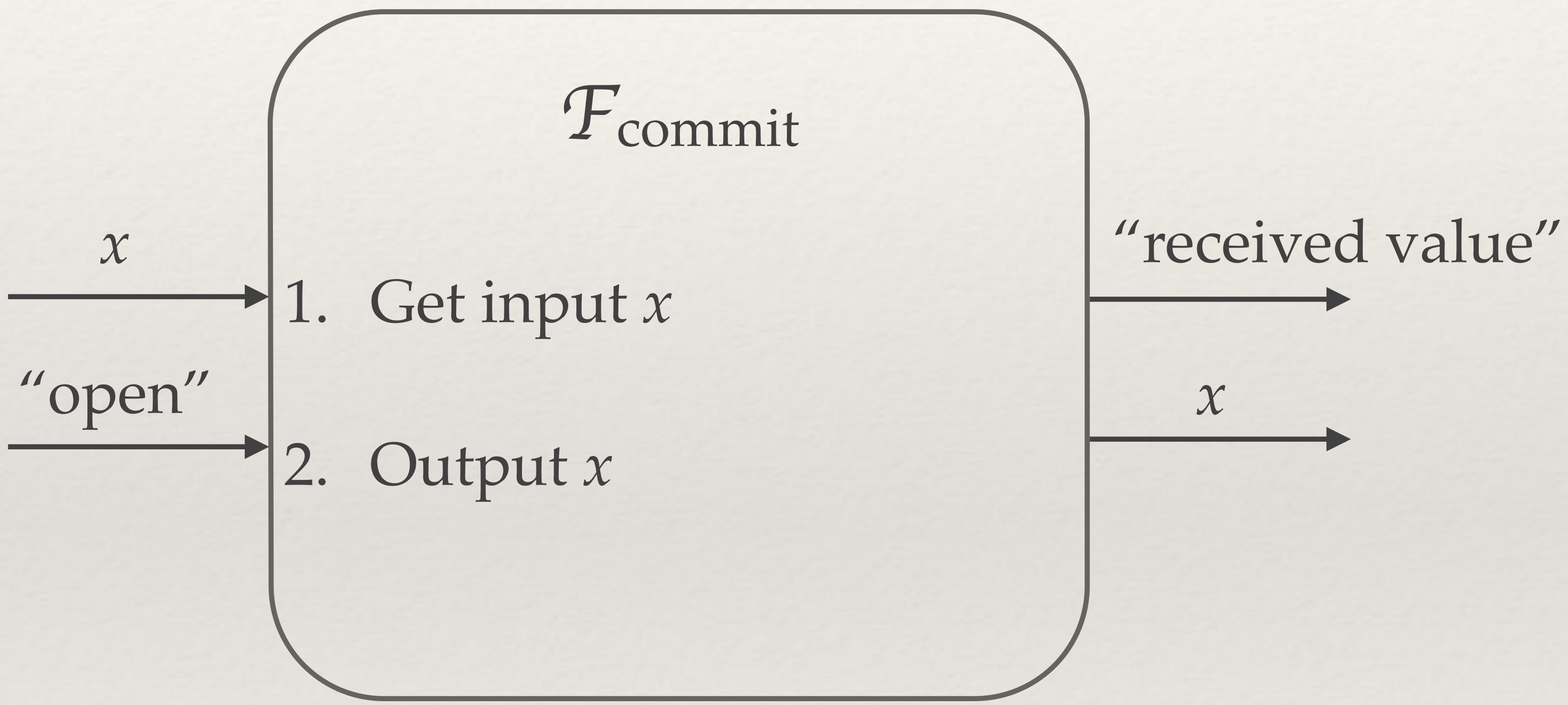
Rational Protocol Design

- General framework (beyond fairness),
- supports composition (via the underlying framework),
- generalizes to reactive functionalities (follow-up).

cf. Ranjit's talk

Rational Protocol Design

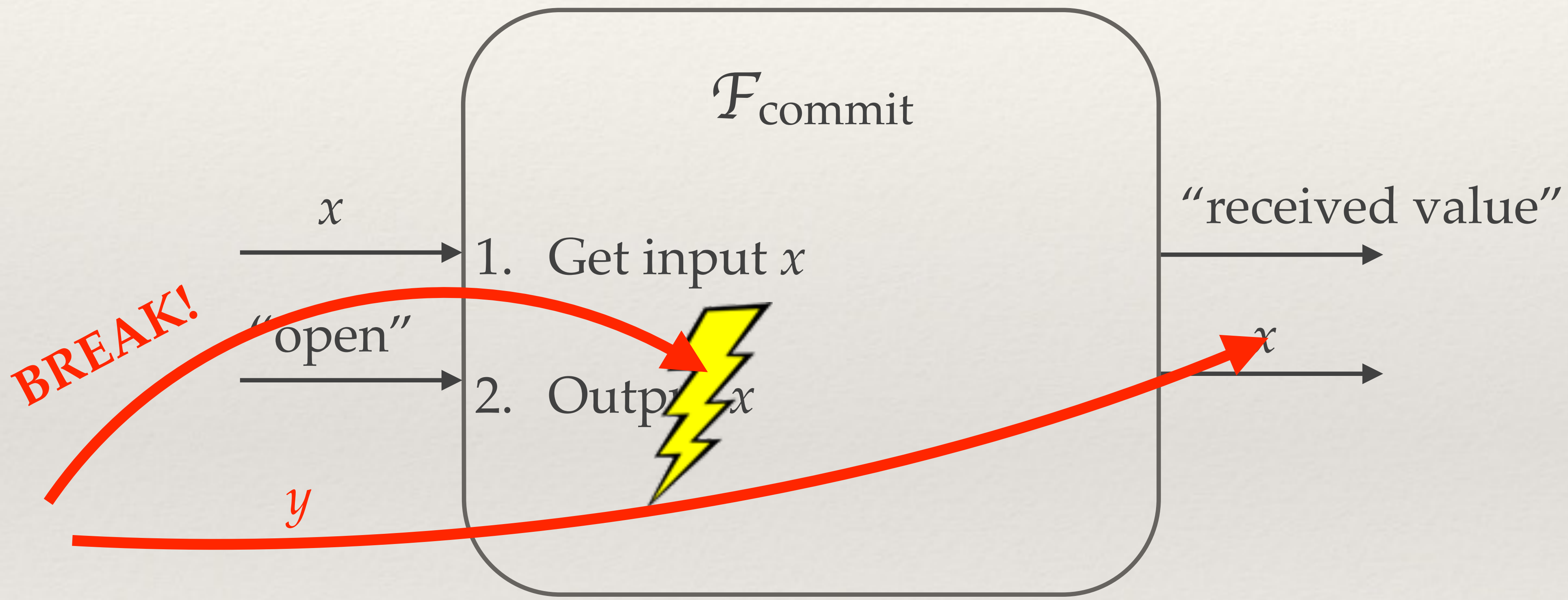
"Rational" commitment*:



* as mentioned by Rosario on Monday.

Rational Protocol Design

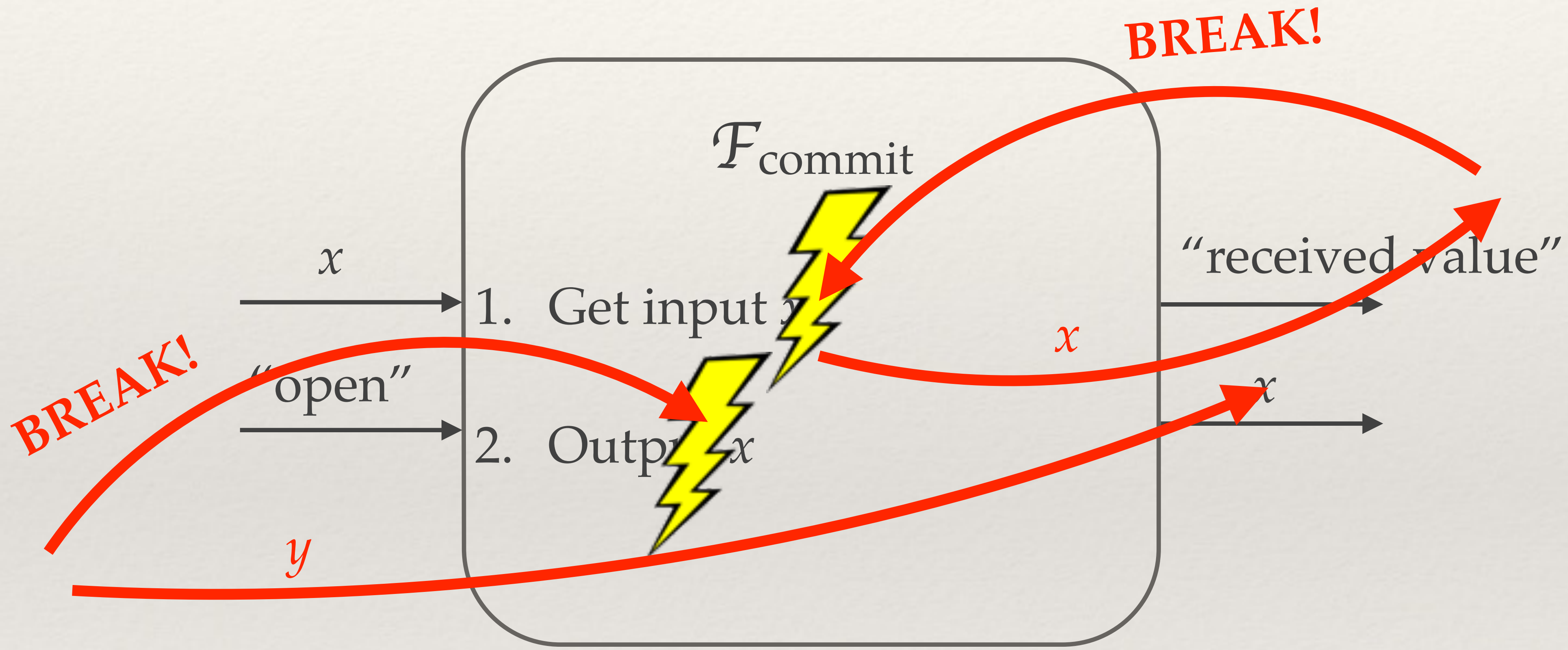
"Rational" commitment*:



* as mentioned by Rosario on Monday.

Rational Protocol Design

"Rational" commitment*:



* as mentioned by Rosario on Monday.

Summary

- RPD is a general framework capturing incentives,
- idea: build the best protocol w.r.t. the incentives,
- we showed optimal protocols for fairness in SFE.
- Follow-up: Reactive functionalities.