Broadcast and Consensus in Cryptographic Protocols

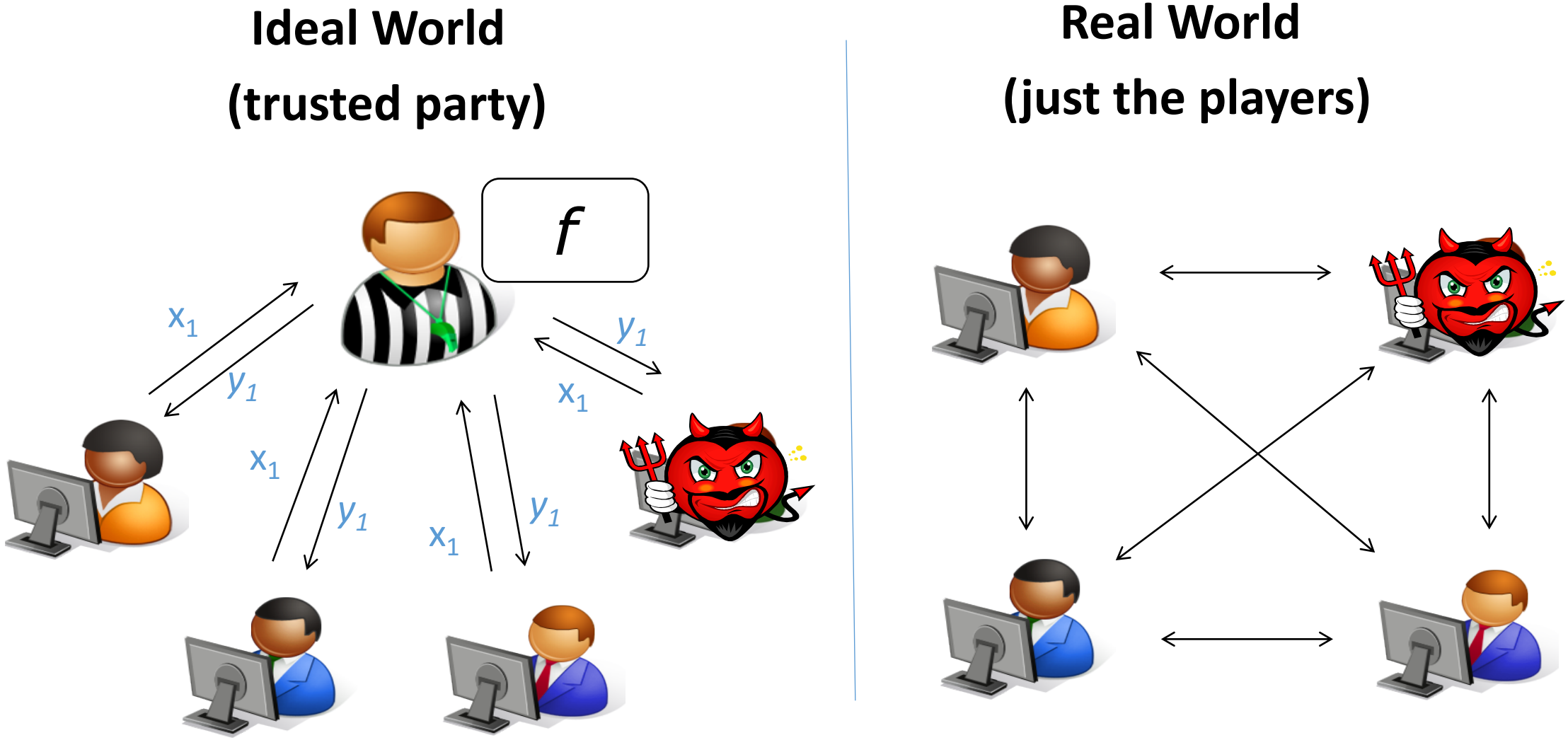
Juan A. Garay
Texas A&M University
garay@tamu.edu
URL://engineering.tamu.edu/cse/people/garay-juan

IISc-IACR School on Cryptology

Secure Multiparty Computation (MPC)

- **Secure multi-party computation** (MPC) [GMW'87] :
  - n parties $\{P_1, P_2, \ldots, P_n\}$, t *corrupted*; each P_i holds a private input x_i
  - One public function $f(x_1,x_2,\ldots,x_n)$
  - All want to learn $y = f(x_1,x_2,\ldots,x_n)$ (*Correctness*)
  - Nobody wants to disclose his private input (*Privacy*)
- **Secure 2-party computation** (2PC) [Yao'82] : n=2
- *Computationally secure* MPC (2PC)

Secure Multiparty Computation (MPC) (2)

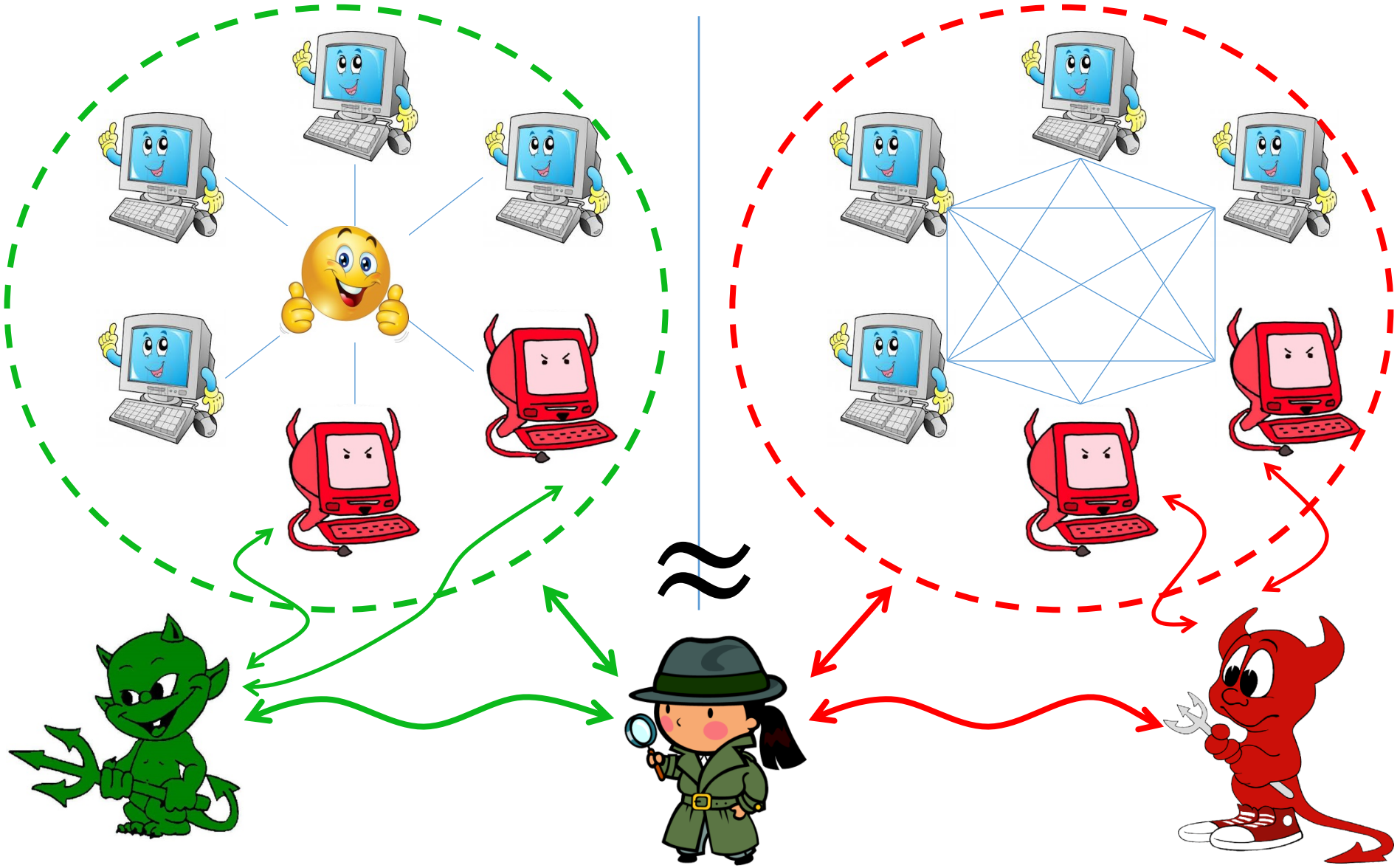


The Trusted-Party Paradigm [GMW87]

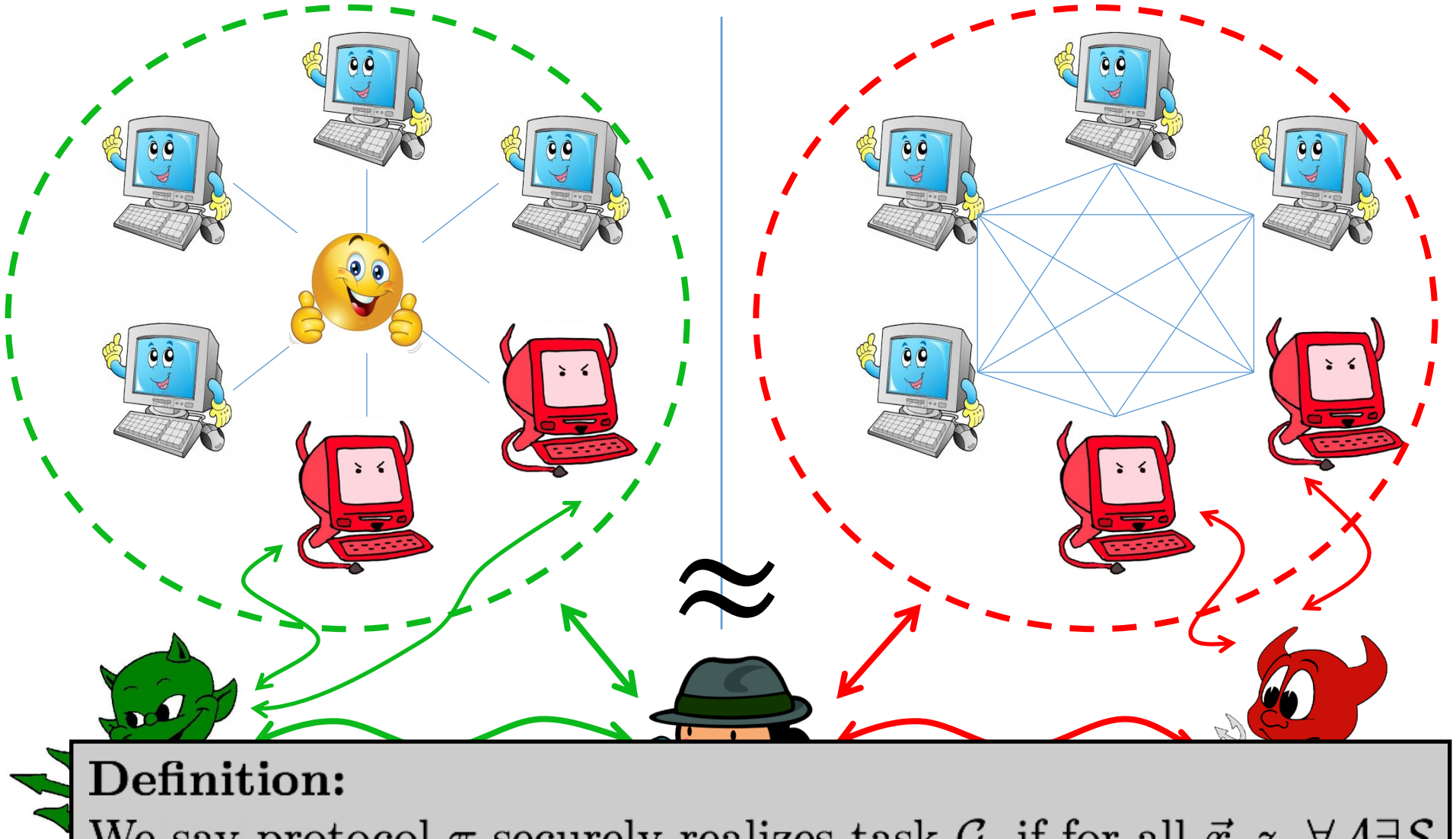
- *'A protocol is secure for some task if it "emulates" an "ideal process" where the parties hand their inputs to a "trusted party," who locally computes the desired outputs and hands them back to the parties.'*

- (Aka the "simulation paradigm.")

Simulation-based Security



Simulation-based Security



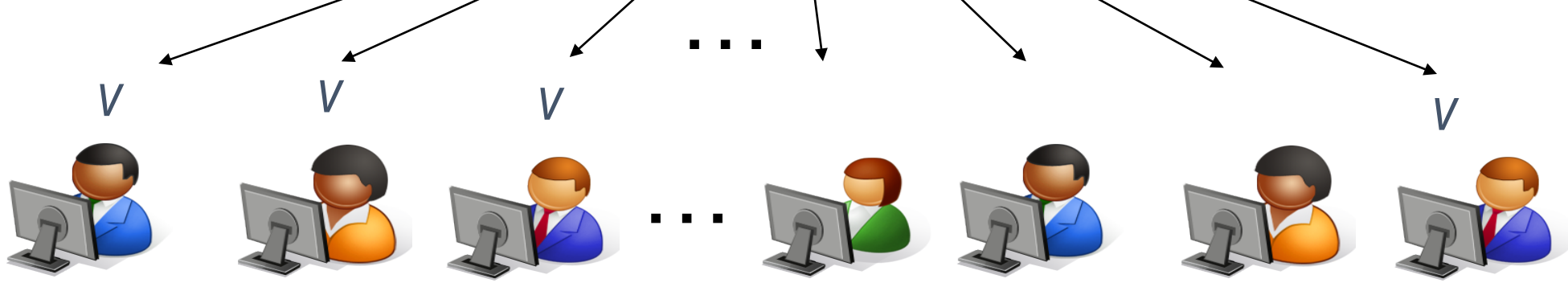
Definition:
We say protocol π securely realizes task $\mathcal{G}$, if for all $\vec{x}, z$, $\forall \mathcal{A} \exists \mathcal{S}$ such that $\{\mathsf{REAL}_{\pi,\mathcal{A}}\}_{\vec{x},z} \approx \{\mathsf{IDEAL}_{\mathcal{G},\mathcal{S}}\}_{\vec{x},z}$

Broadcast Functionality ("Channel") [PSL'80]

n players

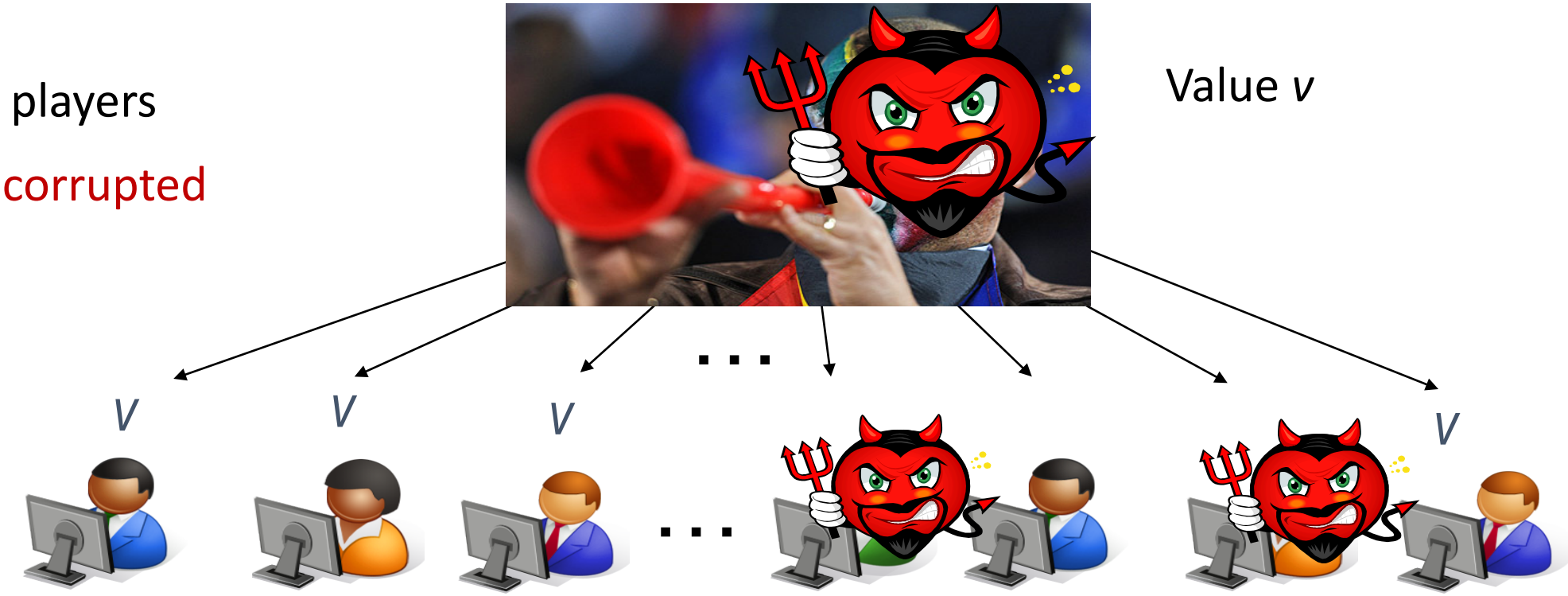
t corrupted

Value *v*



- If source is honest, $v_i = v$ (Validity)
- $v_i = v_j$ (Agreement)

Broadcast Functionality ("Channel") [PSL'80]



- If source is honest, $v_i = v$ (Validity)
- $v_i = v_j$ (Agreement)

MPC: Assumptions on Number of Parties

Unconditionally secure MPC typically assumes:

- For $t < n/3$ [BGW'88, CCD'88]:
  - Secure (private and authentic) pairwise channels
  - Broadcast channel—but it may be realized by Byzantine agreement protocol
- For $t < n/2$ [RB'89]:
  - Secure (private and authentic) pairwise channels
  - **Physical *broadcast channel*** (no protocol exists!)

An MPC Protocol for f

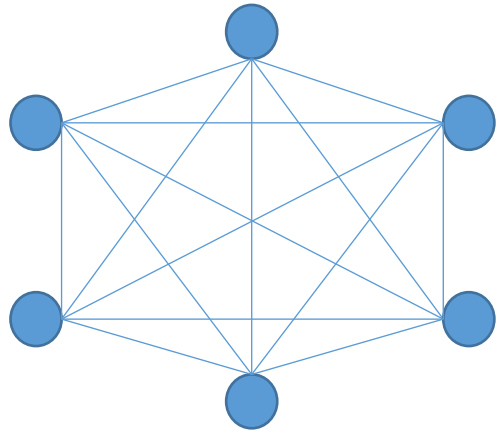
The "*share-compute-reveal*" paradigm:

1. *Share phase*: Each P_i "commits" to his input (using Verifiable Secret Sharing [VSS])
2. *Compute phase*: Shared inputs are used to evaluate an arithmetic circuit C gate-by-gate. (Typically a *linear* VSS scheme is used.)
3. *Reveal phase*: At C 's output gate, parties possess a verifiable sharing of $f(x_1,x_2,\ldots,x_n)$; parties publicly reconstruct this value

- **Multiplication gate:** Most expensive part of MPC protocol – typically requires broadcast channel

Network/Communication Models

- Point-to-point model
  - Secure (private) channels between the parties
    (*Secure Message Transmission*)
- Broadcast model
  - Additional *broadcast channel*
- Synchronous communication
  - Bounded delay
  - Global clock
  - Protocol proceeds in rounds
  - Guaranteed termination



Instantiating Broadcast Channel

Broadcast

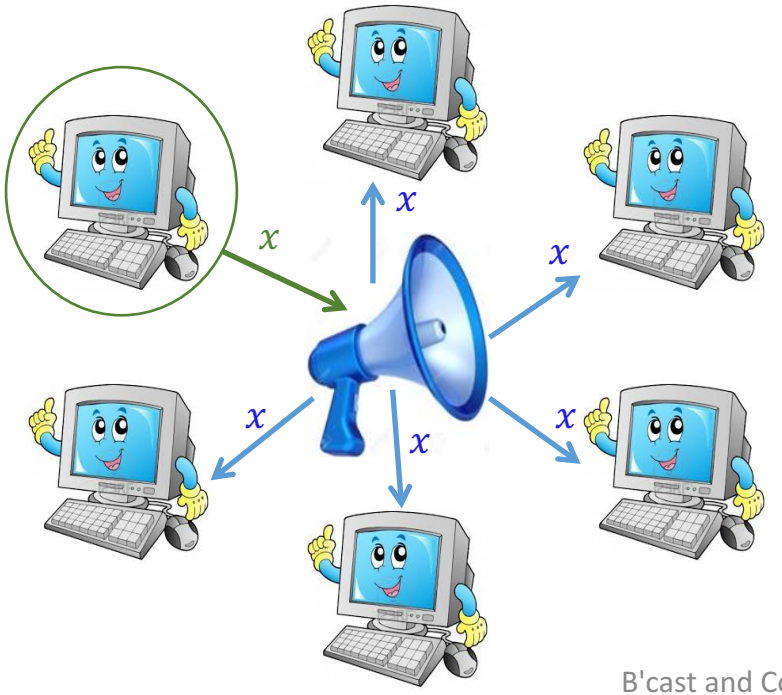
Sender with input x

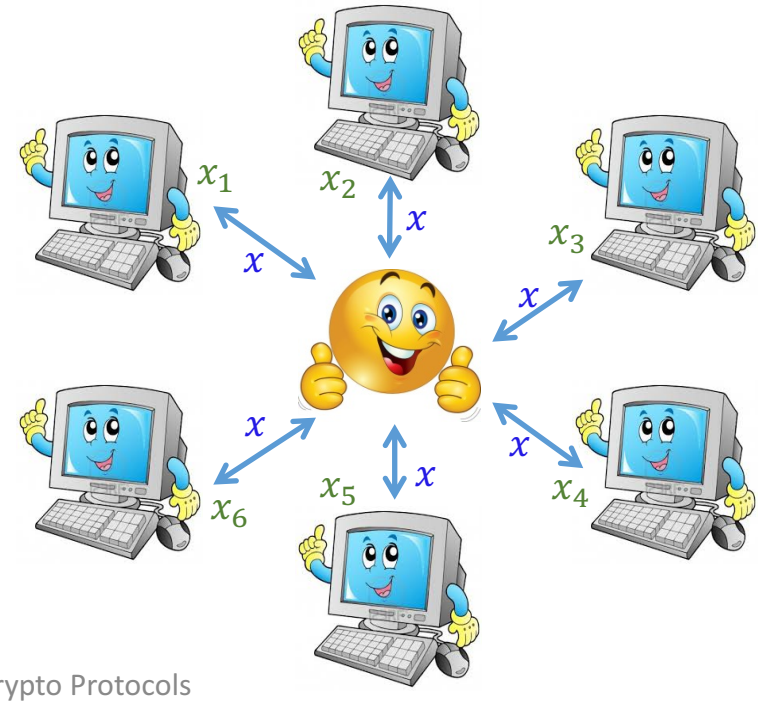
- **Agreement**: All honest parties output the same value
- **Validity**: If the sender is honest, the common output is x

Byzantine agreement

Each P_i has input x_i

- **Agreement**: All honest parties output the same value
- **Validity**: If all honest parties have the same input x, the common output is x





Instantiating Broadcast Channel

Broadcast

Sender with input x

- **Agreement**: All hon
  output the same va
- **Validity**: If the sender is honest, the common output is x

Real-world security definition (property-based)

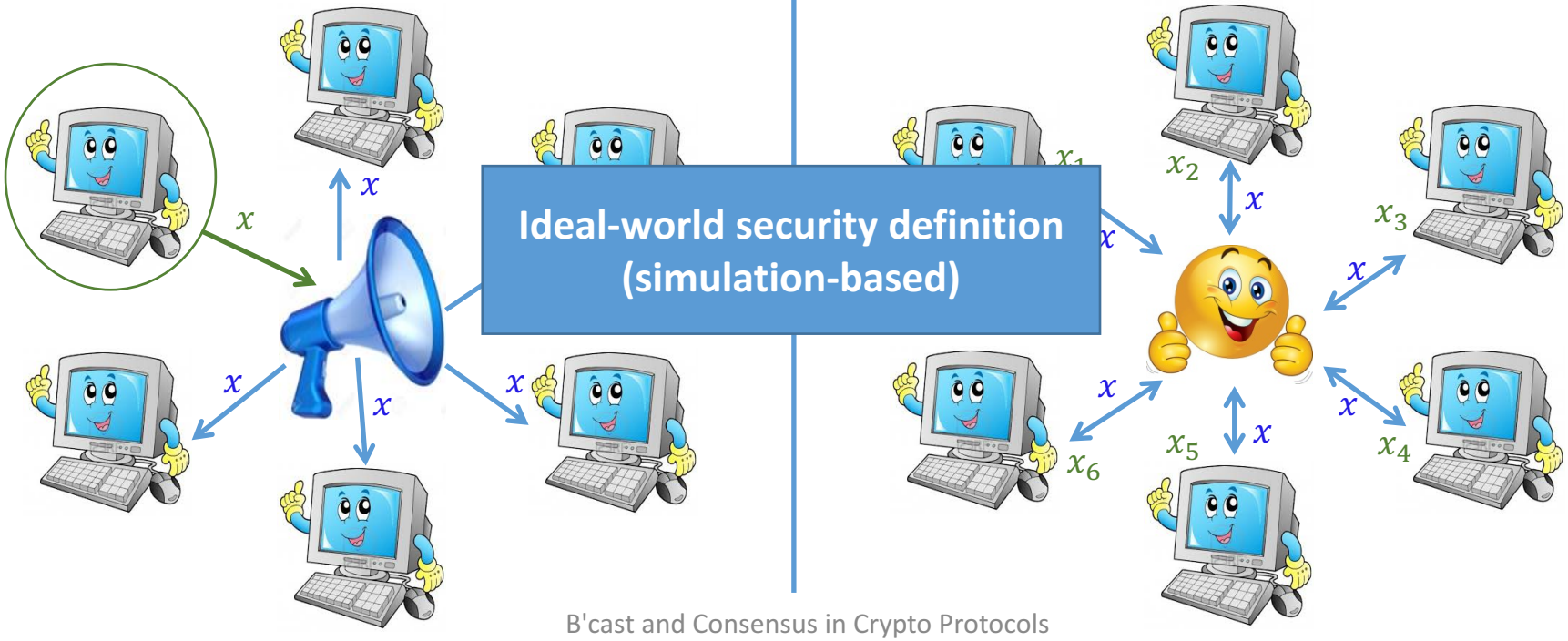
Byzantine agreement

input x_i

nt: All honest parties
e same value

- **Validity**: If all honest parties have the same input x, the common output is x

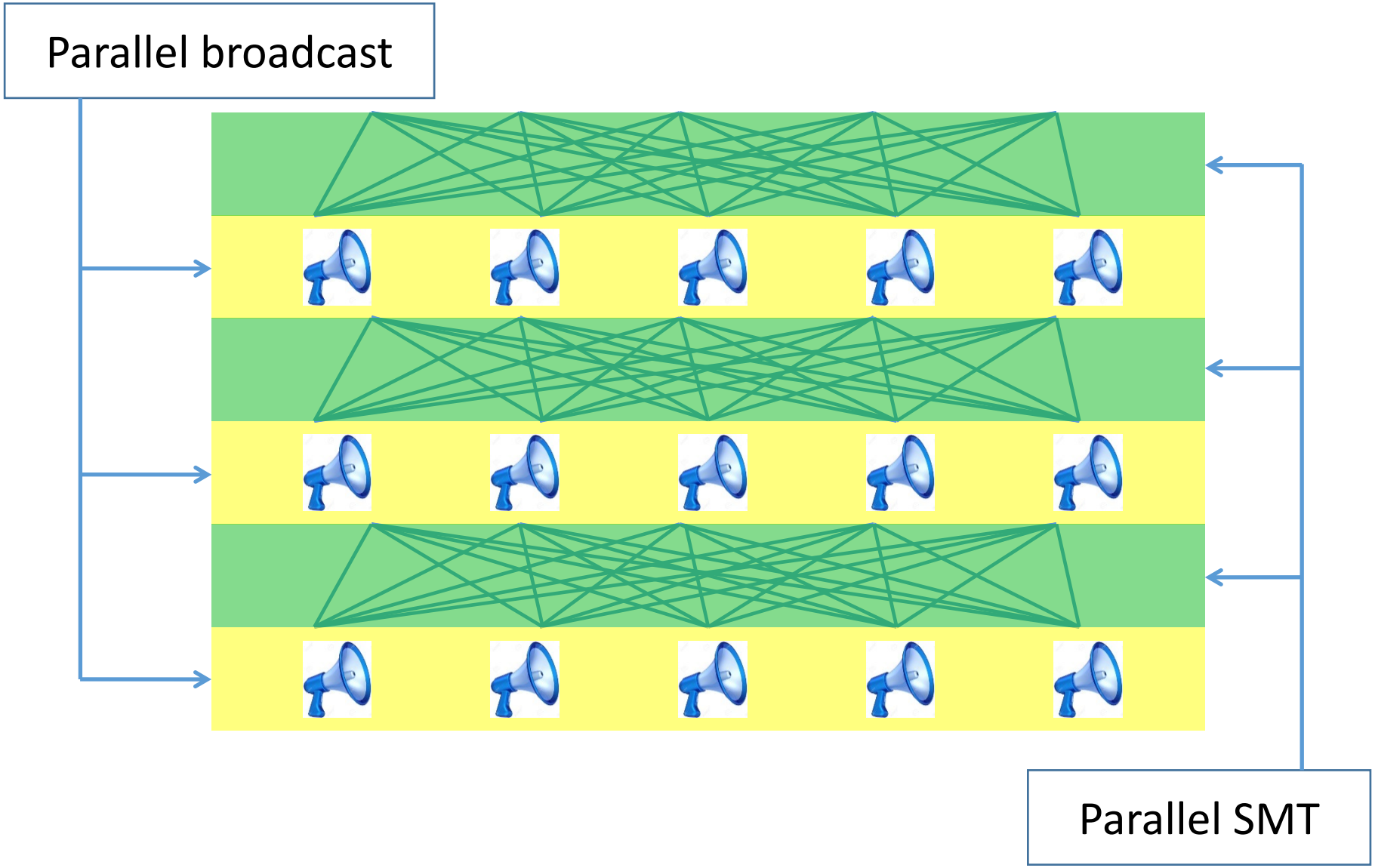
Ideal-world security definition (simulation-based)



Feasibility of MPC with (Instantiated) Broadcast

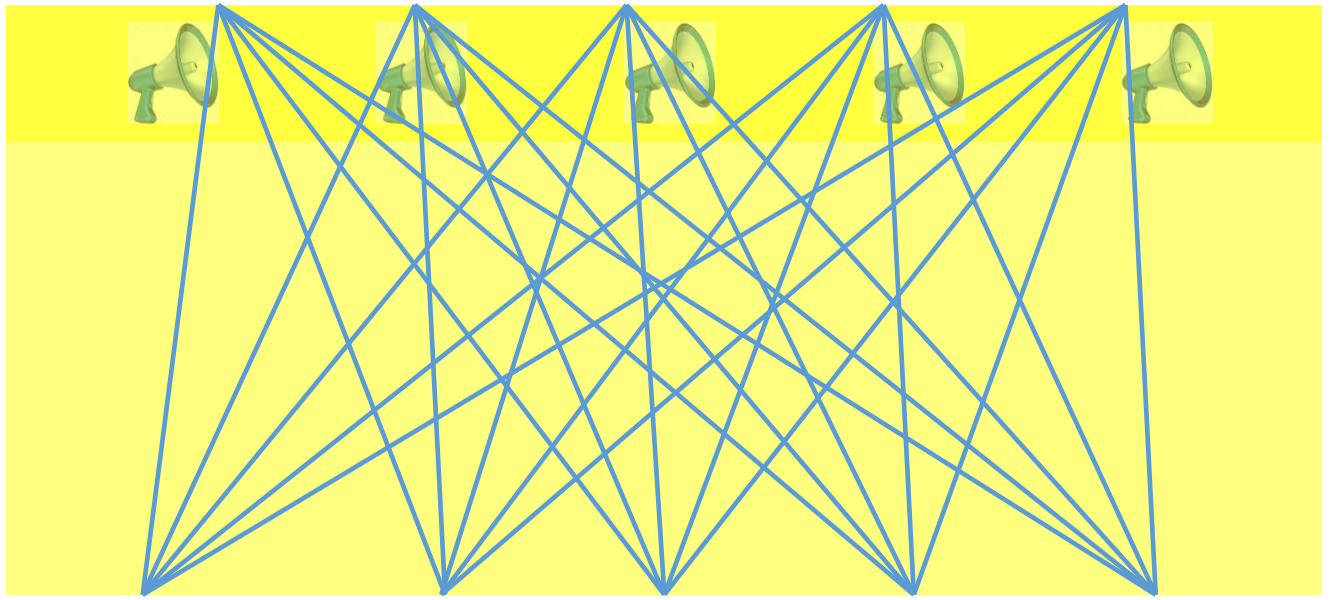
- Classical result [BGW'88]
  - "Share-compute-reveal" paradigm
  - Perfect, adaptively secure for $t < n/3$
  - Concurrently composable
  - $O(d)$ rounds, $O(d)$ broadcasts
- Improving communication complexity
  - E.g., player-elimination framework [HMP'00] [HM'01] [BH'06] [HN'06] [DN'07] [BH'08] [BFO'12]
  - $O(d + n)$ rounds
- Improving round complexity
  - $O(d)$ rounds, 1 broadcast [KKK'07]

Protocols with (Instantiated) Broadcast



Deterministic Broadcast/Consensus Protocols

- Perfect and adaptive security for $t < n/3$
  [BGP'89] [GM'93] [HZ'10]
- "Deterministic termination" (DT) – single output round
- Compose nicely
- However, they require $O(n)$ rounds – this is inherent [FL'82]



Lecture's Roadmap

I. Introduction

II. Brief Recap
- B'cast/consensus definitions, models, protocols

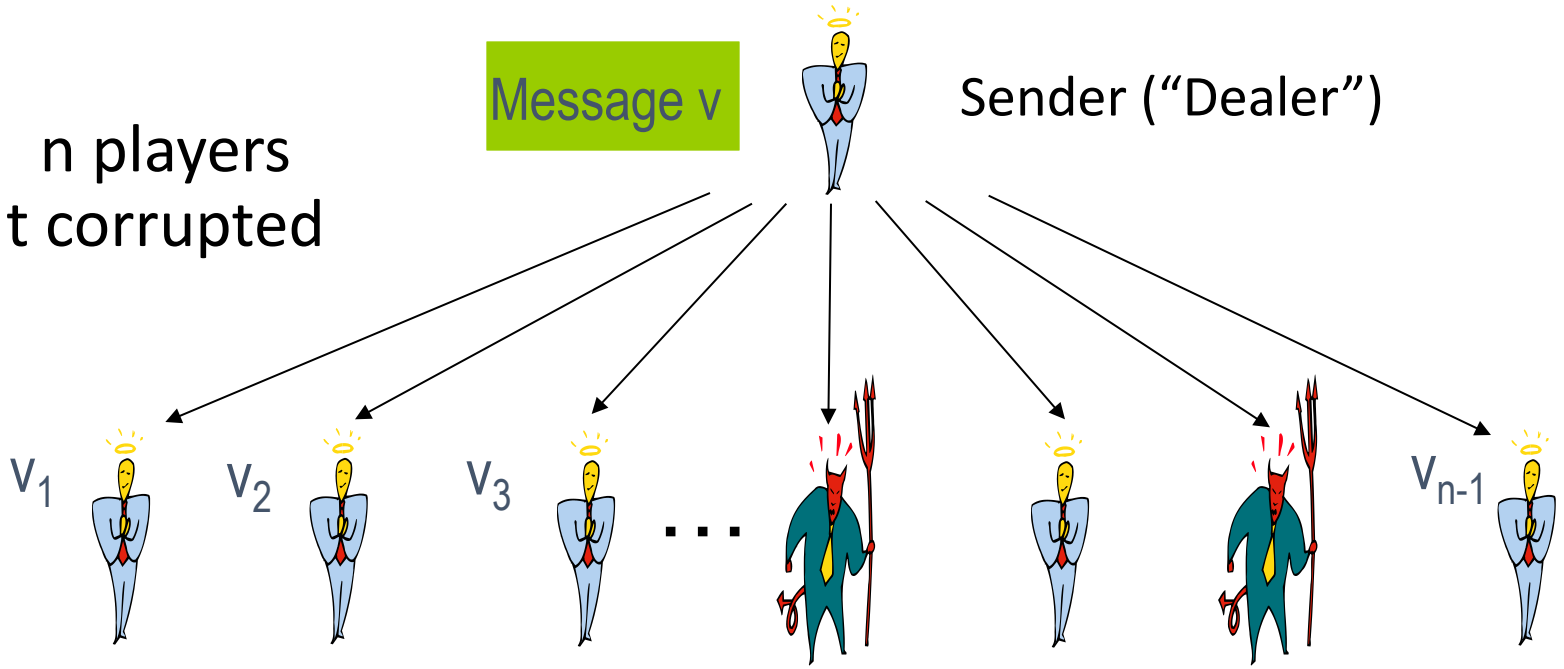
III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

Broadcast (aka *Byzantine Generals*) [PSL80, LSP82]



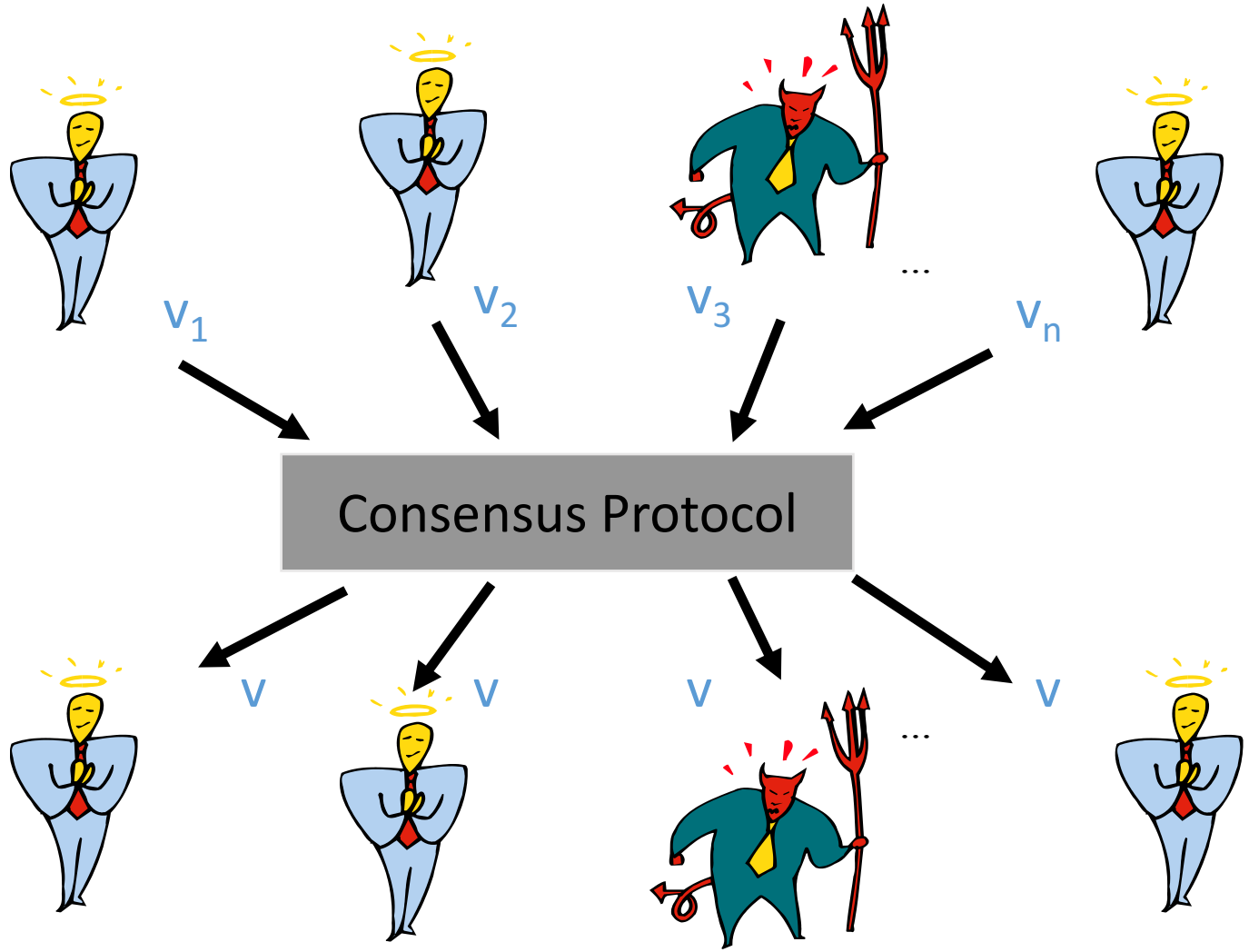
- Validity: If dealer is honest, $v_i = v$
- Agreement: $v_i = v_j$
- Termination: Every player eventually outputs a value

Consensus (aka *Byzantine Agreement*) [PSL80, LSP82]

n parties
t *corrupted*

Adversary

$v_i \in V = \{0,1\}$



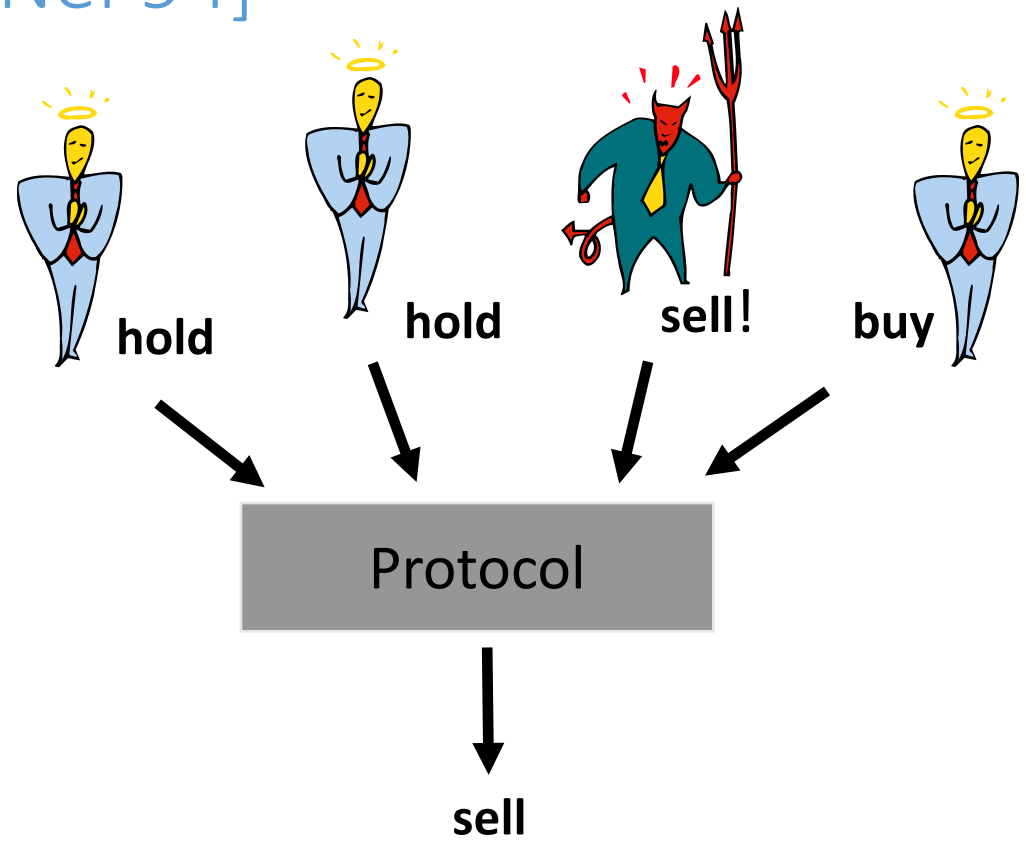
Consensus (aka *Byzantine agreement*) [PSL80, LSP82]

- **The Consensus Problem:** n parties start with an initial value v_i
  - Agreement: All honest parties output the same value
  - Validity: If all honest parties start with the same input (say, v), then they output this value
  - Termination: All honest parties eventually terminate

Strong Consensus [Nei'94]

m > 2

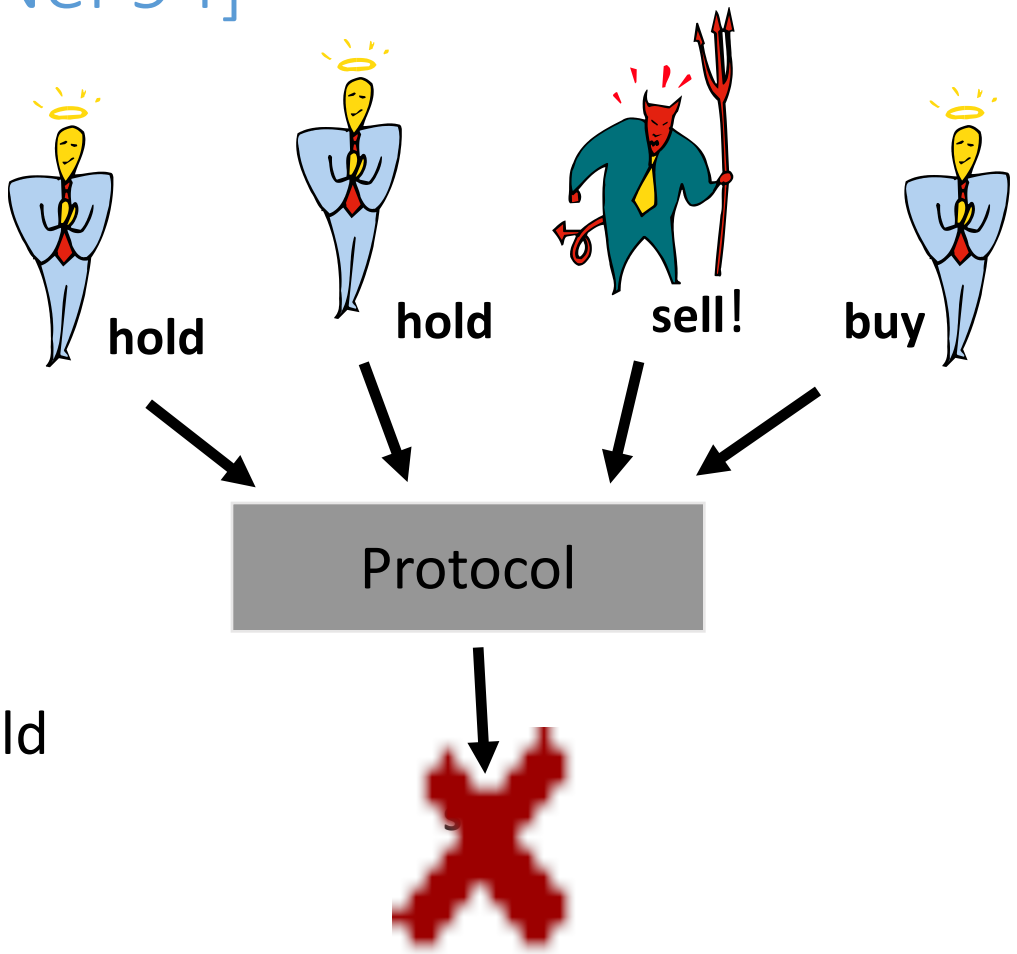
V = {sell,buy,hold}



Strong Consensus [Nei'94]

m > 2

V = {sell,buy,hold}



Decision value held by *good* party!

Strong Consensus

Termination: All good players decide on a value

Agreement: If two good players decide on v and w, resp., then v = w

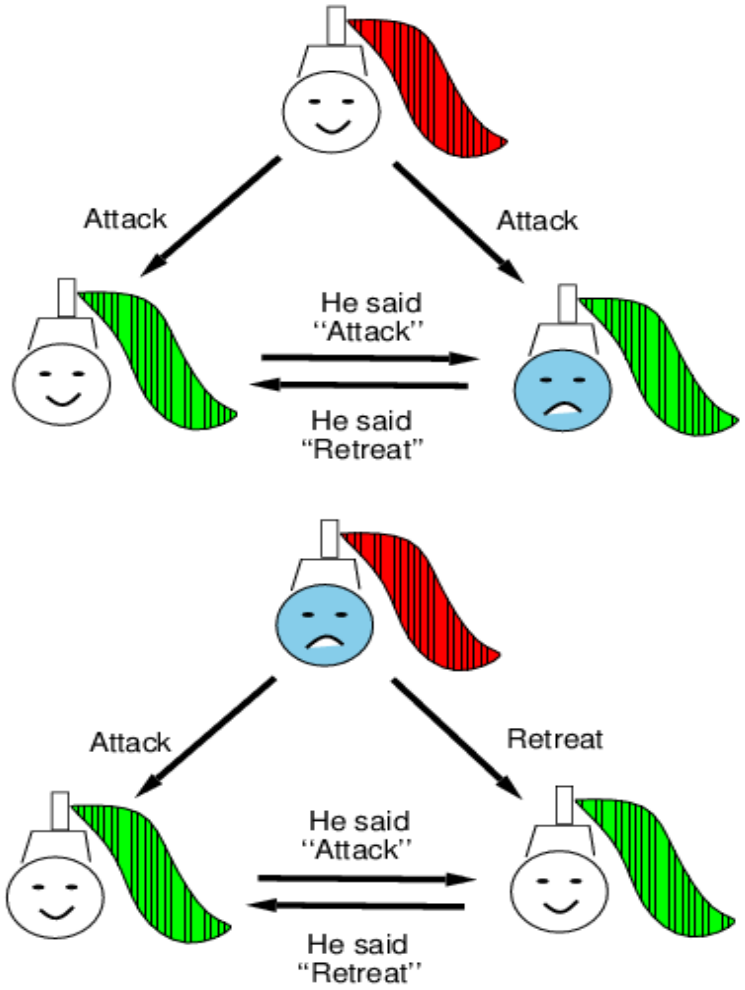
Validity: If all good players have the same initial value, v, then all good players decide on v

Strong Validity: If the good players decide on v, then v is the initial value of some good player

Standard Model(s) (Setup + Assumptions)

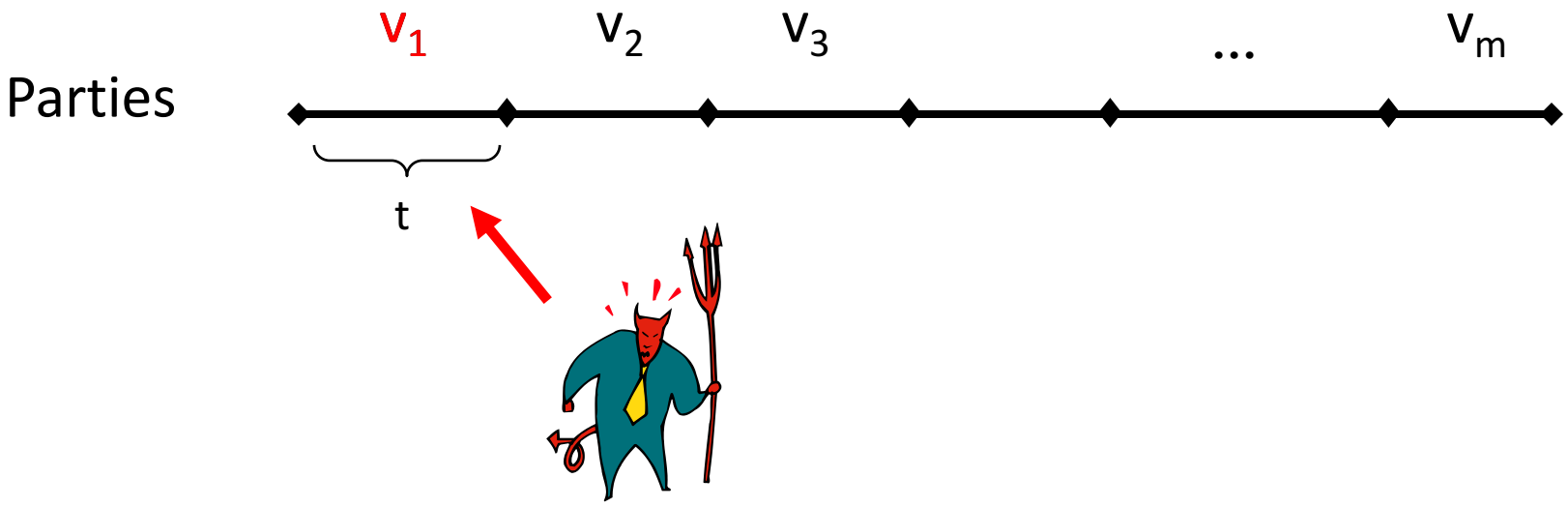
- **Channels:** Authenticated point-to-point
- **Network communication:** Synchronous; *rushing* adversary
- **Adversary's computational power:**
  - Unbounded ("unconditional," information-theoretic security)
  - Polynomial time (in security parameter; cryptographic, "authenticated")

Impossibility of B'cast (Consensus) with n = 3t [PSL80, LSP82]



Impossibility of *Strong* Consensus [Nei'93,FG'03]

$$n > \max(3,m)t$$



Complexity Measures

- **Rounds:** $r = t+1$ [LSP82, FL82]
- **Resiliency:**
  - Unconditional setting: $n > 3t$ [LSP82]

Complexity Measures

- **Rounds:** $r = t+1$ [LSP82, FL82]
- **Resiliency:**
  - Unconditional setting: $n > 3t$ [LSP82]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Agreement: $n > 2t$ [Fit03]

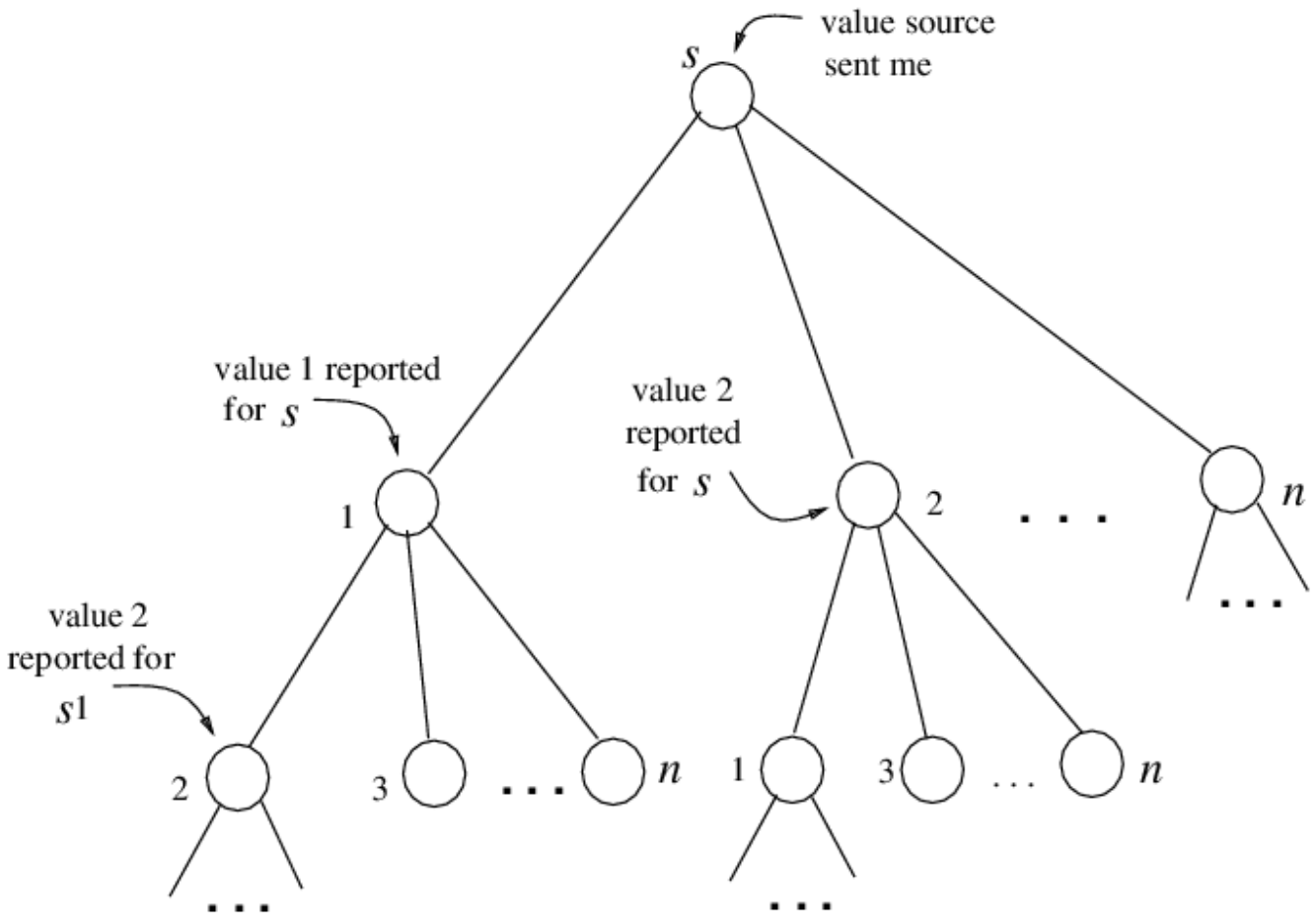
Complexity Measures

- **Rounds:** $r = t+1$ [LSP82, FL82]
- **Resiliency:**
  - Unconditional setting: $n > 3t$ [LSP82]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Agreement: $n > 2t$ [Fit03]
- **Message/Bit complexity:** $m = \Omega(n^2)$ [DR85,BGP92,CW92]

Unconditional Broadcast/Consensus Protocols

- **Network:** Point-to-point authenticated channels
- [LSP82]: $n > 3t$, $r = t+1$, $\exp(n)$

LSP's Protocol ("EIG" [BDDS87])



t+1

Unconditional Consensus Protocols (2)

- **Network:** Point-to-point authenticated channels
- [LSP82]: $n > 3t$, $r = t+1$, $\exp(n)$

Unconditional Consensus Protocols

- **Network:** Point-to-point authenticated channels
- [LSP82]: n > 3t, r = t+1, exp(n)
- [GM93]: n > 3t, r = t+1, poly(n)
  - [BG91,AD15]: r = min(t+1, f+2) (optimal *early stopping* [DRS90])

Unconditional Consensus Protocols

- **Network:** Point-to-point authenticated channels
- [LSP82]: $n > 3t$, $r = t+1$, $\exp(n)$
- [GM93]: $n > 3t$, $r = t+1$, $\text{poly}(n)$
  - [BG91,AD15]: $r = \min(t+1, f+2)$ (optimal *early stopping* [DRS90])
- If $r = O(t)$, much simpler protocols
  - E.g., the "Phase King" paradigm[BG89]

Authenticated Consensus Protocols

- **Setup:** Public-key infrastructure (PKI)
- **Assumption:** Digital signatures secure against adaptive chosen-message attacks [GMR88]
- [DS82]: $n > t$, $r = t+1$, poly(n)

Authenticated Consensus Protocols (2)

- [DS82] protocol (informal):
  - Source signs its input value and sends to all parties
  - r = 1,…,t+1:
    - If any value $v_i \in V = \{0,1\}$ has been *newly* added to a set of accepted values, sign it and send value and signatures to everybody
    - If a value/signatures message is received by any party containing valid signatures by at least r distinct players including the sender, then accept the value and update signatures
  - If only one accepted value, then the party outputs that value; otherwise a default value

Randomized Consensus Protocols

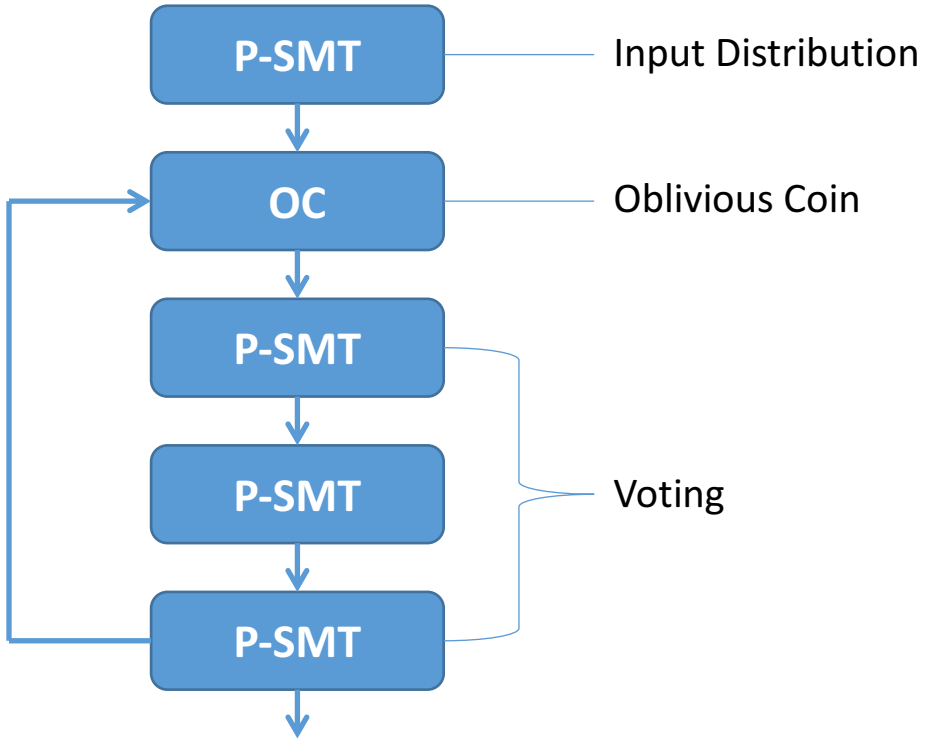
- [BO83, Rab83]: Introduction of randomization to distributed algorithms (2015 Dijkstra Prize)
- Expected *constant* no. of rounds; probabilistic, non-simultaneous termination [DRS90]
- Consensus reduces to access to "common coin" [Rab83]
- [FM88]: Common coin from "scratch"
  - [KK06]: Common coin in the cryptographic setting

Randomized Consensus Protocols

- [BO83, Rab83]: Introduction of randomization to distributed algorithms (2015 Dijkstra Prize)
- Expected *constant* no. of rounds; probabilistic, non-simultaneous termination [DRS90]
- Consensus reduces to access to "common coin" [Rab83]
- [FM88]: Common coin from "scratch"
  - [KK06]: Common coin in the cryptographic setting
- "Probabilistic termination" broadcast/consensus protocols [CCGV16]

Randomized Consensus Protocols (2)

Example: Protocol π_{RBA} (based on [FM'88])



Randomized Consensus Protocols (3)

[FM'88] in more detail:

- Proceeds in phases until termination
- In each phase each party has an input bit
  - If all honest parties start the phase with the same bit, they terminate at the end of the phase
  - Otherwise, with probability $p > 0$ all honest parties agree on the same bit at the end of the phase (and terminate in the next phase)
  - With probability $1 - p$
    - No agreement at the end of the phase, or
    - the adversary makes some of the honest parties terminate; the remaining parties will terminate in the next phase

Randomized Consensus Protocols (4)

- [FM'88] has *Probabilistic Termination* (PT):
  - Expected $O(1)$ rounds
  - No guaranteed termination: statistical security (for PPT parties)
  - No simultaneous termination: honest parties might terminate at different rounds [DRS'90]
  - All honest parties terminate in a constant window
- Extends to multi-valued BA [TC'84]
  - Two additional rounds
- Perfect security [GP'90]
  - Best of both worlds

Randomized Consensus Protocols (6)

Two issues:

1. Running time of *parallel* randomized b'cast/consensus protocols?
2. Composition
   - All PT broadcast protocols are proven secure using a property-based definition
   - Composition theorems require simulation-based proofs

Instantiating Broadcast Channel

RECALL

Broadcast

Sender with input x

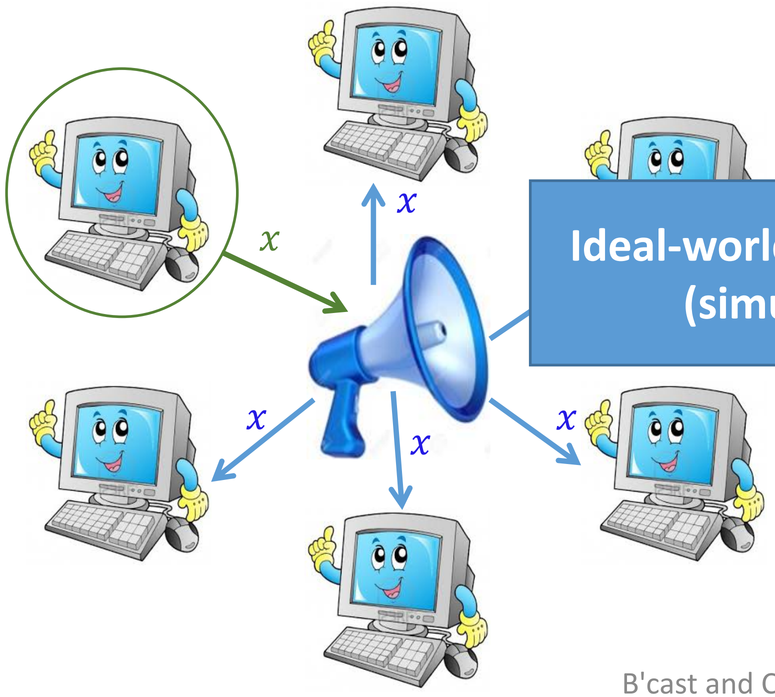
- **Agreement**: all hon
output the same va
- **Validity**: if the sender is honest, the common output is x

Byzantine agreement

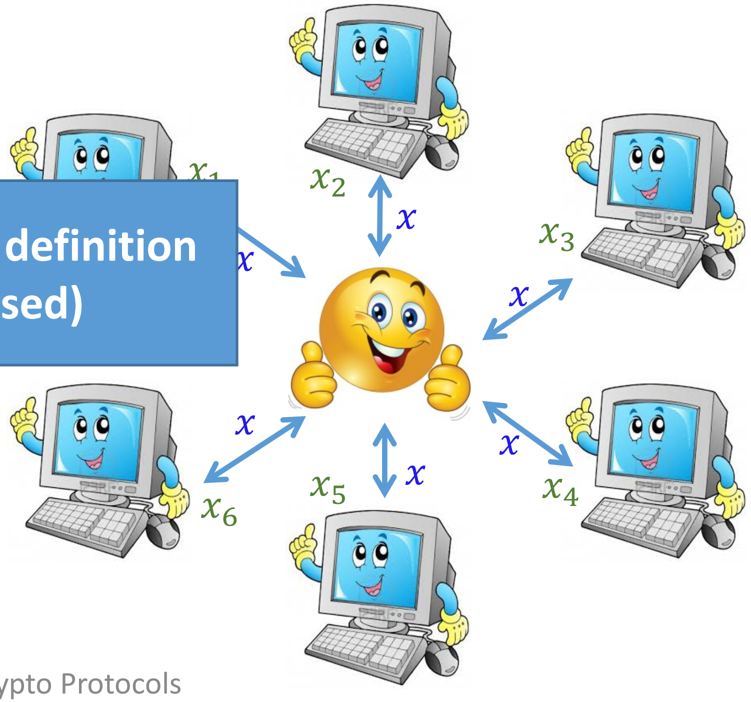
input x_i

- nt: all honest parties
he same value
- **Validity**: if all honest parties have the same input x, the common output is x

Real-world security definition (property-based)



Ideal-world security definition (simulation-based)



Randomized Consensus Protocols (6)

Two issues:

1. Running time of parallel randomized protocols?
2. Composition
   - All PT broadcast protocols are proven secure using a property-based definition
   - Composition theorems require simulation-based proofs

- **Next:** A framework for designing and analyzing PT protocols [CCGZ'16]

Lecture's Roadmap

I. Introduction

II. Brief Recap
- B'cast/consensus definitions, models, protocols

III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

Issue 1

Given: Protocol with *expected* O(1) running time

Issue 1

Given: Protocol with *expected* O(1) running time
(e.g., geometric distribution)

Issue 1

Given: Protocol with *expected* O(1) running time
What's the expected running time of *n* parallel instances?

Issue 1

Given: Protocol with *expected* O(1) running time

What's the expected running time of n parallel instances?

$\Theta(\log n)$ rounds

Issue 1

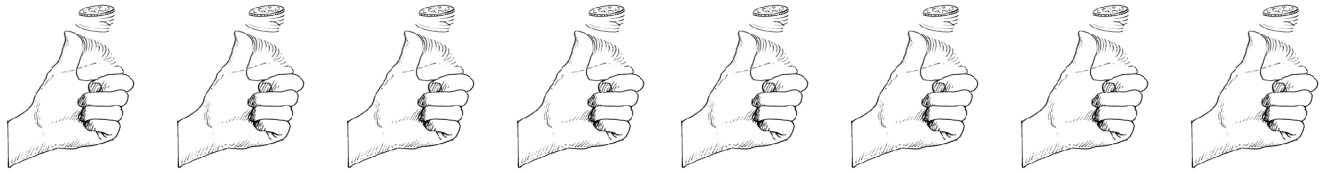
Given: Protocol with *expected* O(1) running time
What's the expected running time of n parallel instances?

$\Theta(\log n)$ rounds

Example: Coin flipping

- Stand-alone coin flip: Pr(*heads*) = ½
  Output is *heads* in expected 2 rounds

- Flipping in parallel n coins, each coin until *heads*
  Expected $\log n$ rounds



Issue 1 (2)

- The mathematical expectation of the maximum of *n* random variables does not necessarily equal the maximum of their expectations [BE'03,Eis'08]
- Fast implementations of broadcast protocols run in expected O(1) time
  → parallel executions no longer constant (nor fixed)
  → non-simultaneous termination
- Composition — how to simulate probabilistic termination?

Composition

- All PT broadcast protocols are proven secure using a property-based definition
- Composition theorems require simulation-based proofs
- [KMTZ'13] defined a UC-based framework for synchronous DT protocols
- PT protocols are very delicate – many subtle issues not captured by [KMTZ'13]
- **Next:** A framework for designing and analyzing PT protocols [CCGZ'16]

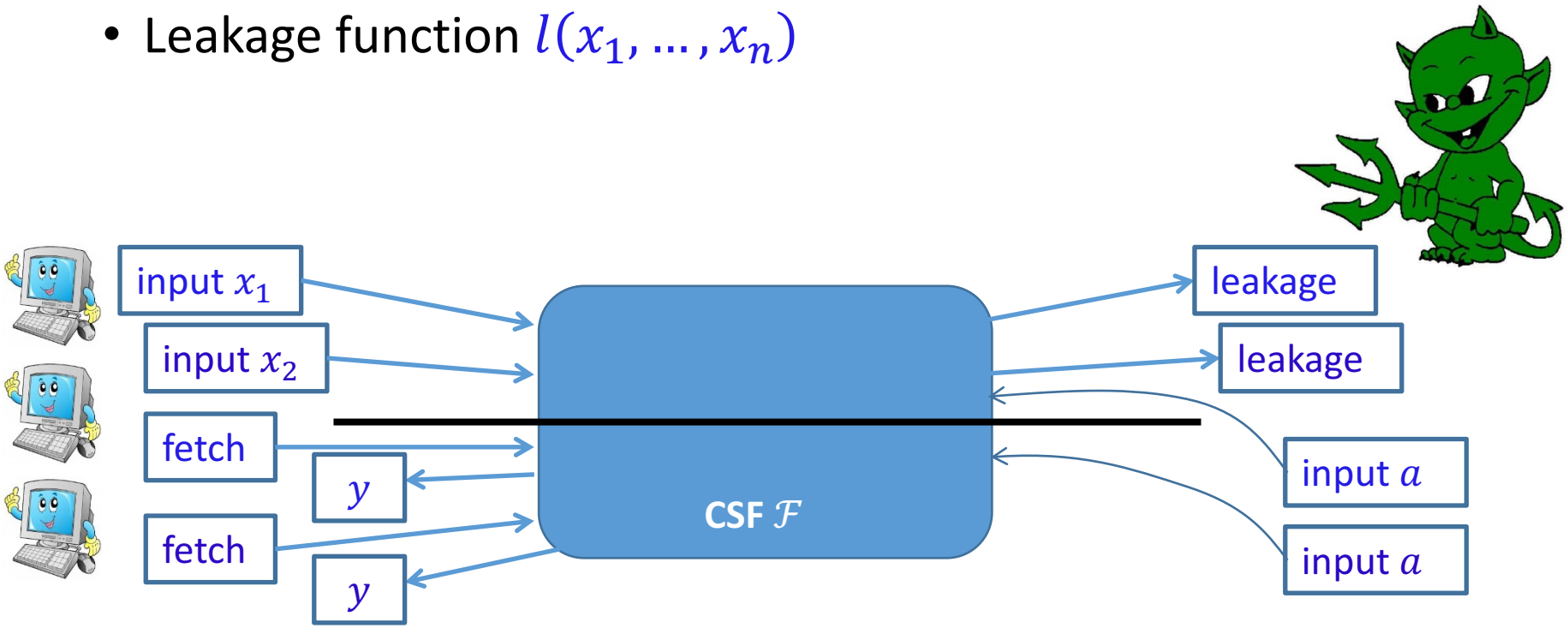
The PT Framework
Part I: The Basics

Synchronous Protocols in UC

- The environment can observe in which round parties terminate [KTMZ'13]
- One-round functionalities hide the round complexity
- In [KTMZ'13] each ideal functionality is parameterized by number of rounds
- Parties continuously request output and receive at the last round
- ⇒ Parties in ideal world receive output at same round as in protocol execution in the real world

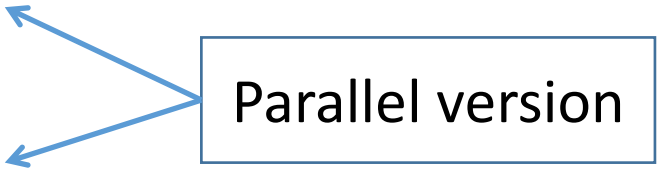
Canonical Synchronous Functionality

- Separate the function from the round structure
- A CSF consists of input round and output round
- Parameterized by
  - (Randomized) function $f(x_1, \ldots, x_n, a)$
  - Leakage function $l(x_1, \ldots, x_n)$



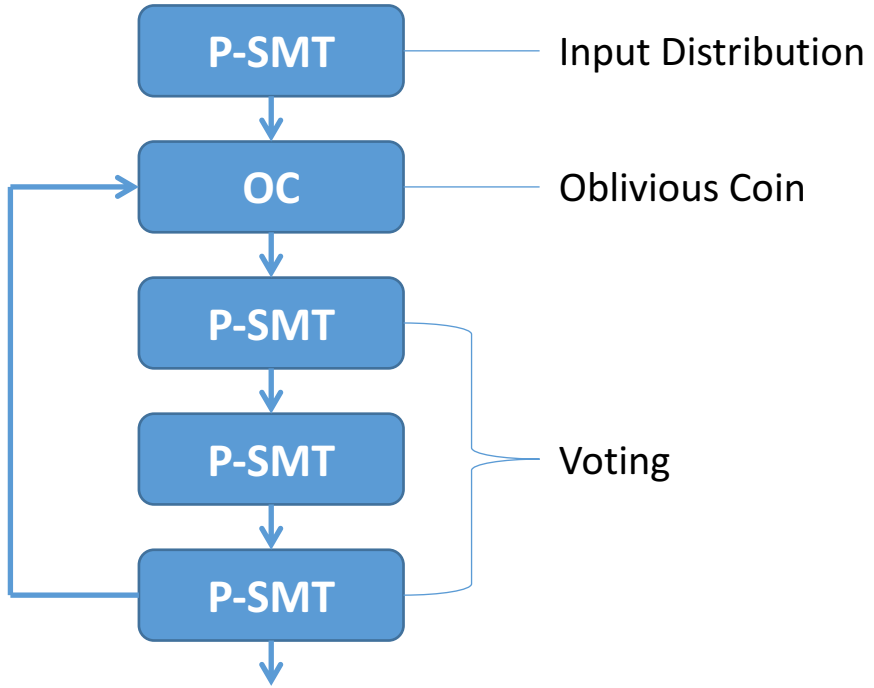
CSF Examples

- **SMT**: P_i sends x_i to P_j
  - $f(x_1, \dots, x_n, a) = (y_1, \dots, y_n)$, s.t. $y_j = x_i$ and $y_k = \lambda$ $(k \neq j)$
  - $l(x_1, \dots, x_n) = \begin{cases} |x_i| \text{ if } P_j \text{ honest} \\ x_i \quad \text{if } P_j \text{ corrupted} \end{cases}$
- **Broadcast**: P_i broadcasts x_i
  - $f(x_1, \dots, x_n, a) = (x_i, \dots, x_i)$
  - $l(x_1, \dots, x_n) = |x_i|$
- **SFE**: parties compute a function g
  - $f(x_1, \dots, x_n, a) = g(x_1, \dots, x_n)$
  - $l(x_1, \dots, x_n) = (|x_1|, \dots, |x_n|)$
- **BA**:
  - $f(x_1, \dots, x_n, a) = \begin{cases} y \text{ if at least } n - t \text{ inputs are } y \\ a \text{ otherwise} \end{cases}$
  - $l(x_1, \dots, x_n) = (x_1, \dots, x_n)$



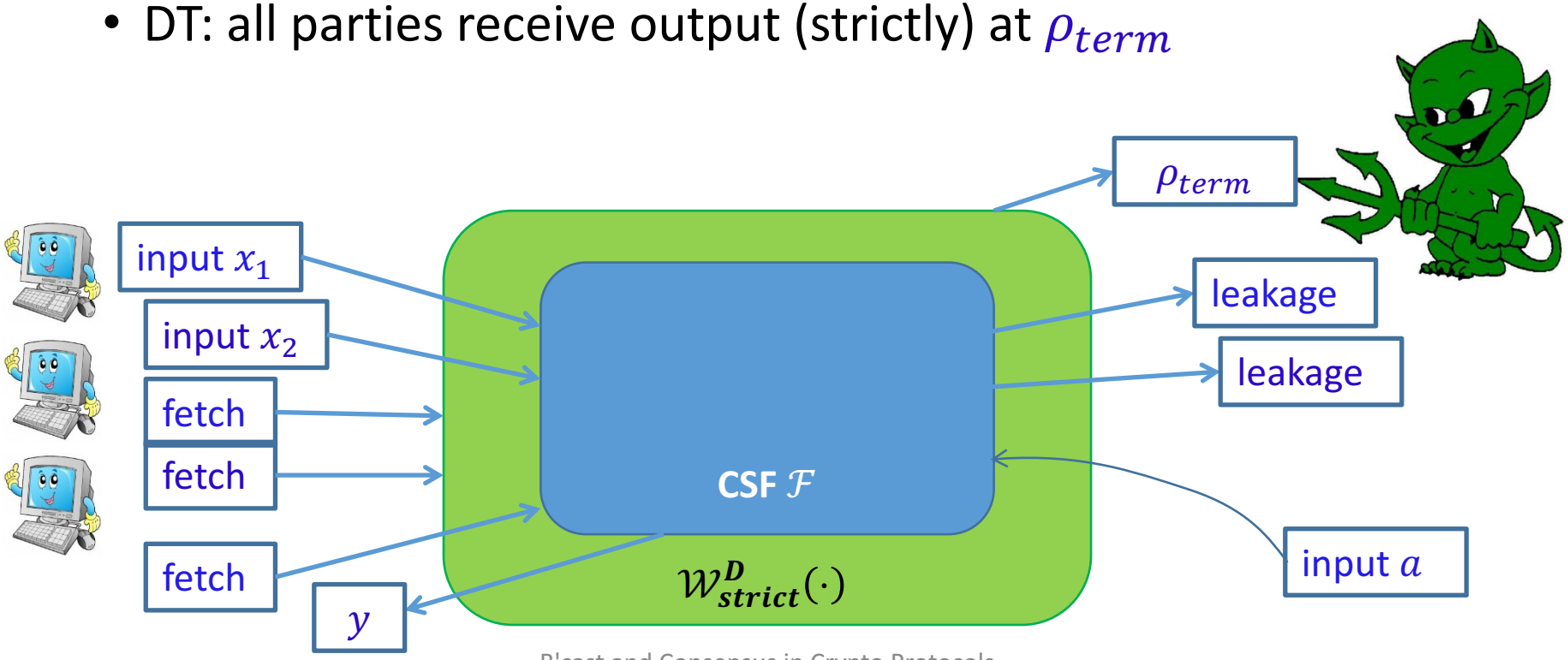
Synchronous Normal Form (SNF)

- SNF protocol:
  - In each round exactly one ideal functionality is called (as in stand-alone)
  - All hybrids are (2-round) CSFs
- Example: Protocol π_{RBA} (based on [FM'87])



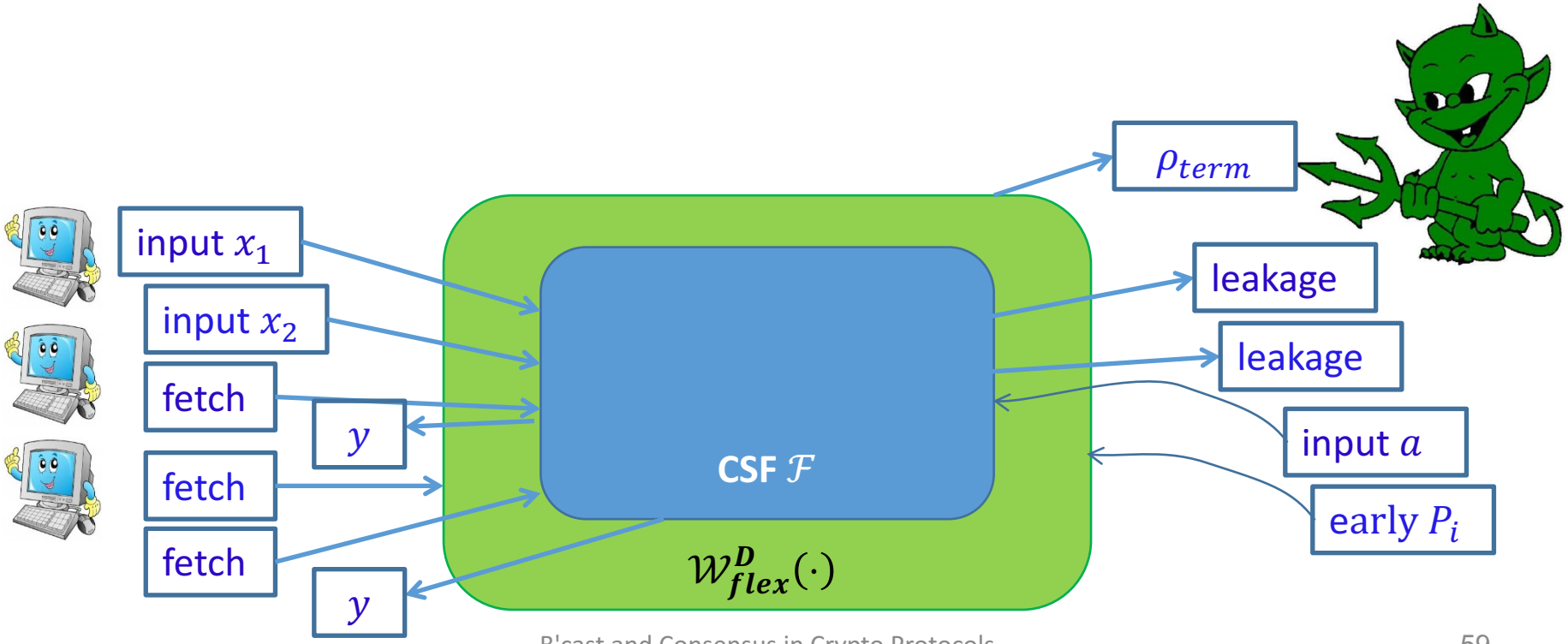
Extending Rounds (DT)

- Most functionalities cannot be implemented by two-round protocols
- Wrap the CSFs with *round-extension* wrappers
  - Sample a termination round $\rho_{term} \leftarrow D$
  - DT: all parties receive output (strictly) at ρ_{term}



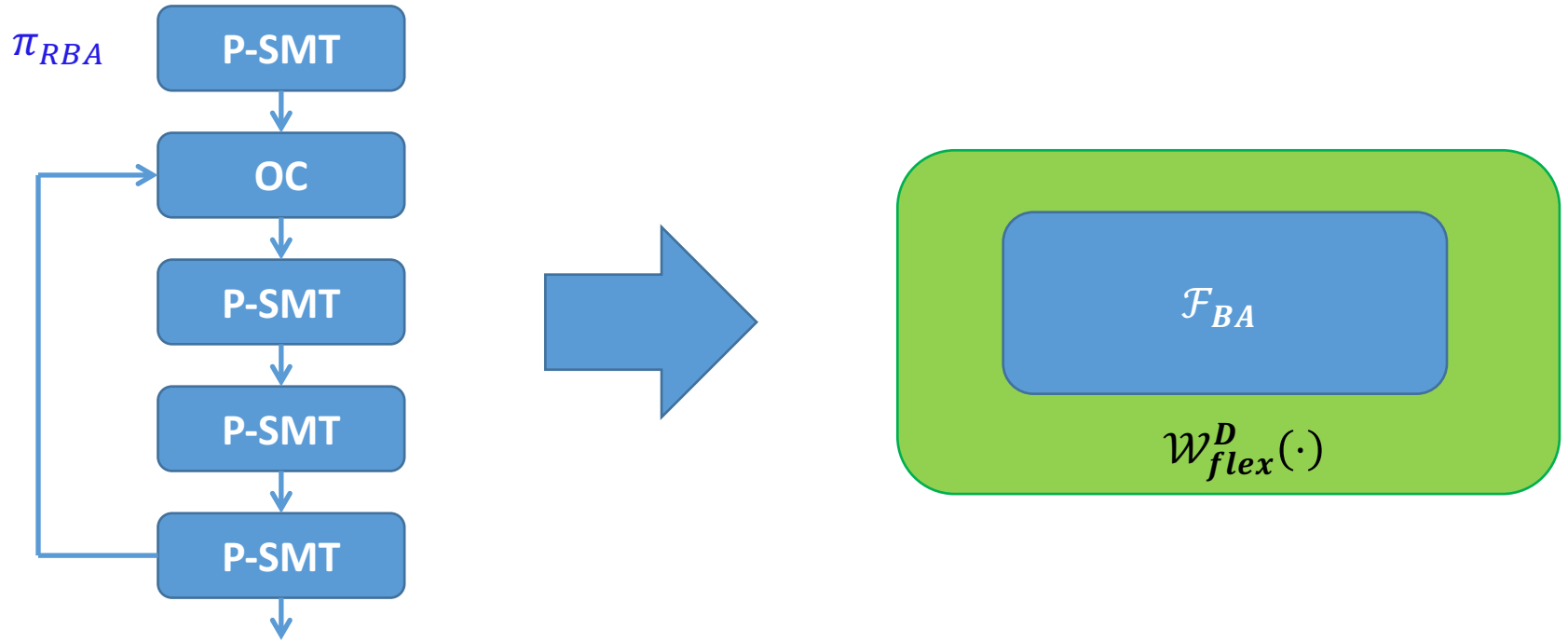
Extending Rounds (PT)

- PT: ρ_{term} is an upper bound
  - Sample a termination round $\rho_{term} \leftarrow D$
  - All parties receive output <u>by</u> ρ_{term} (flexible)
  - $\mathcal{A}$ can instruct early delivery for P_i at any round



Where Do We Stand?

Thm: Protocol π_{RBA} implements $\mathcal{W}_{flex}^{D}(\mathcal{F}_{BA})$ in the $(\mathcal{F}_{PSMT}, \mathcal{F}_{OC})$-hybrid model, for $t < n/3$, assuming all parties start at the same round

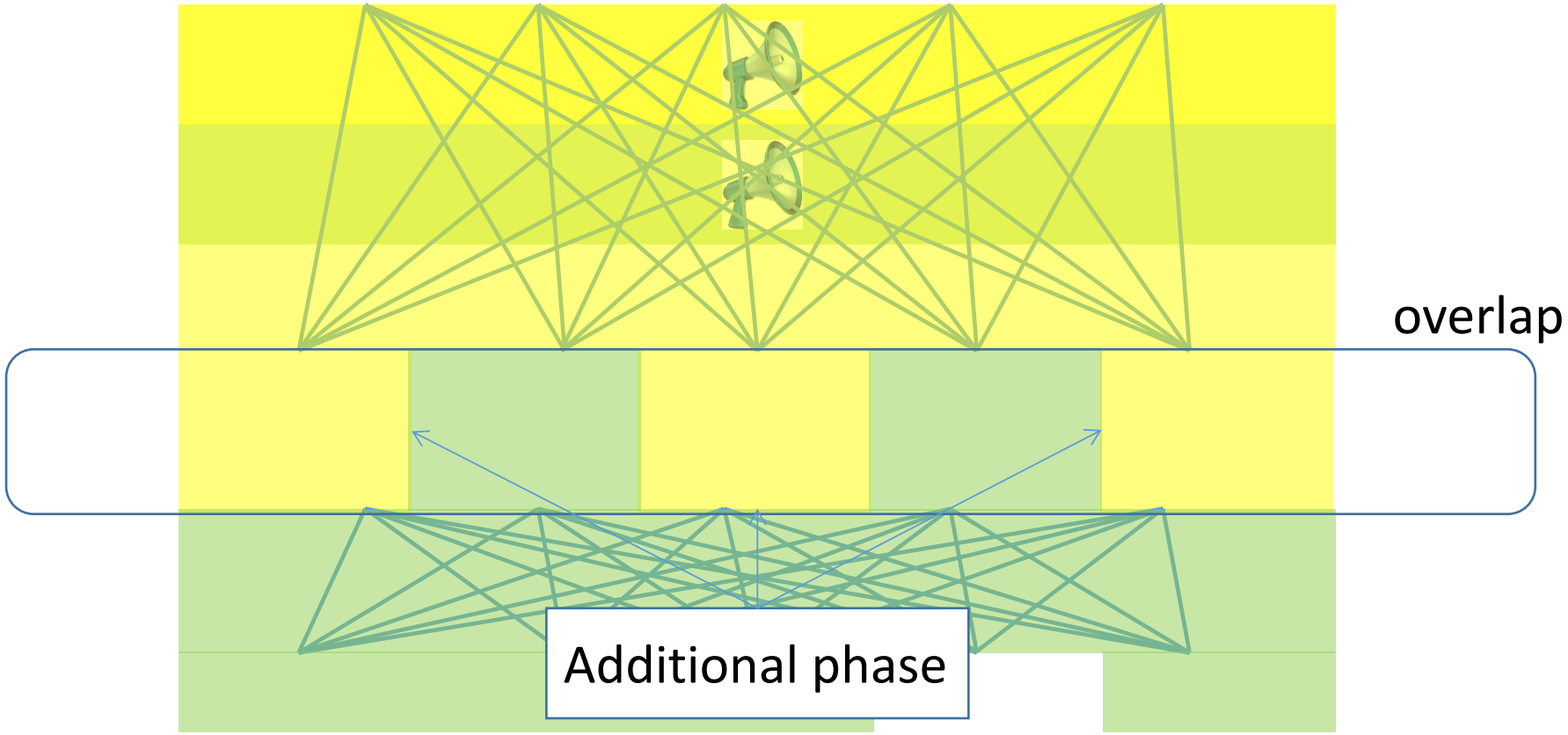


The PT Framework
Part II: Dealing with “Slack”

Problem: Sequential Composition

New execution starts **after all** parties finished previous one

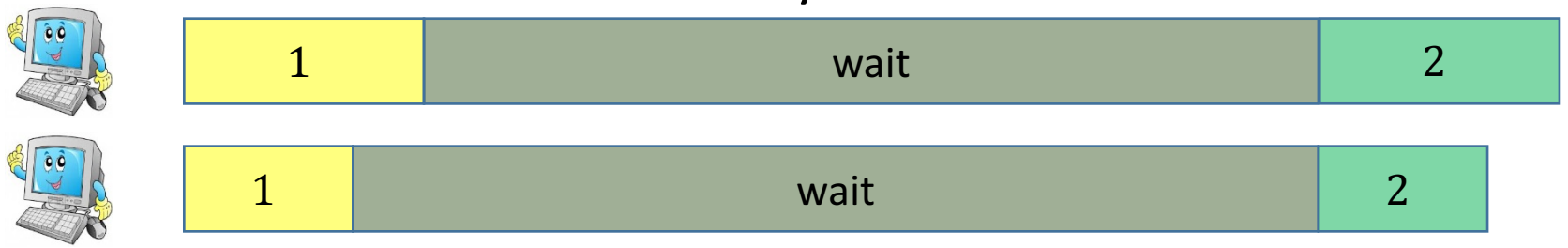
With PT protocols, fast parties start new execution **before** slow parties finished previous execution



Sequential Composition: Solutions

Goal: ℓ sequential executions of expected $O(1)$ rounds protocols in expected $O(\ell)$ rounds

- Naïve solution #1: wait until re-synchronized



- Naïve solution #2:
  Expand each round to $2c + 1$ rounds

Explained in 3 slides

  - Execution 1, start slack $c_1 = c$, expansion factor $2c_1 + 1$
  - Execution 2, slack $c_2 = c(2c_1 + 1)$, factor $2c_2 + 1$
  - Execution 3, slack $c_3 = c(2c_2 + 1)$, factor $2c_3 + 1$
  - After i executions, slack $c(2c_{i-1} + 1) = O(2^{i-1}c^i)$

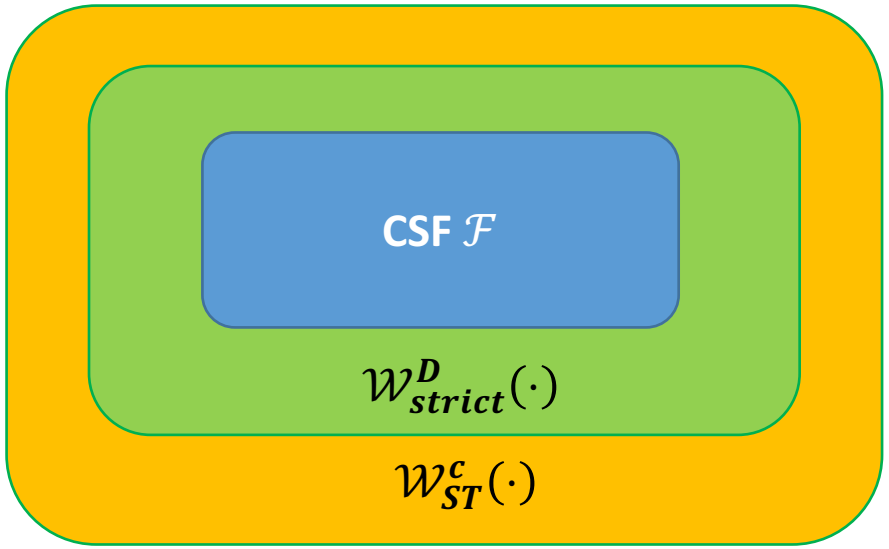
Sequential Composition: Solutions (2)

Goal: ℓ sequential executions of expected $O(1)$ rounds protocols in expected $O(\ell)$ rounds

- [LLR'02] – adding re-synchronization points
  - Statistical security (inherent)
  - Static corruptions
  - Property-based security
- [BE'03] [KK'06]
  - Simpler solutions, partial proofs (no simulation)
- PT framework: A generic compiler for PT protocols
  - Supports non-simultaneous start of the protocol
  - Reduces the slackness to 1
  - Simulation-based security – a composition theorem

“Slack” Tolerance

- **Main idea:** Make the overlap meaningless by adding “dummy” rounds
  - Assume slack of c rounds
  - Extend each round to $3c + 1$ rounds
  - Messages of P_i are queued and forwarded in cycles of $3c + 1$
- DT functionalities: wrap $\mathcal{W}^{D}_{strict}(\mathcal{F})$ with $\mathcal{W}^{c}_{ST}(\cdot)$
  - Each party runs the same number of rounds
  - The slack remains the same

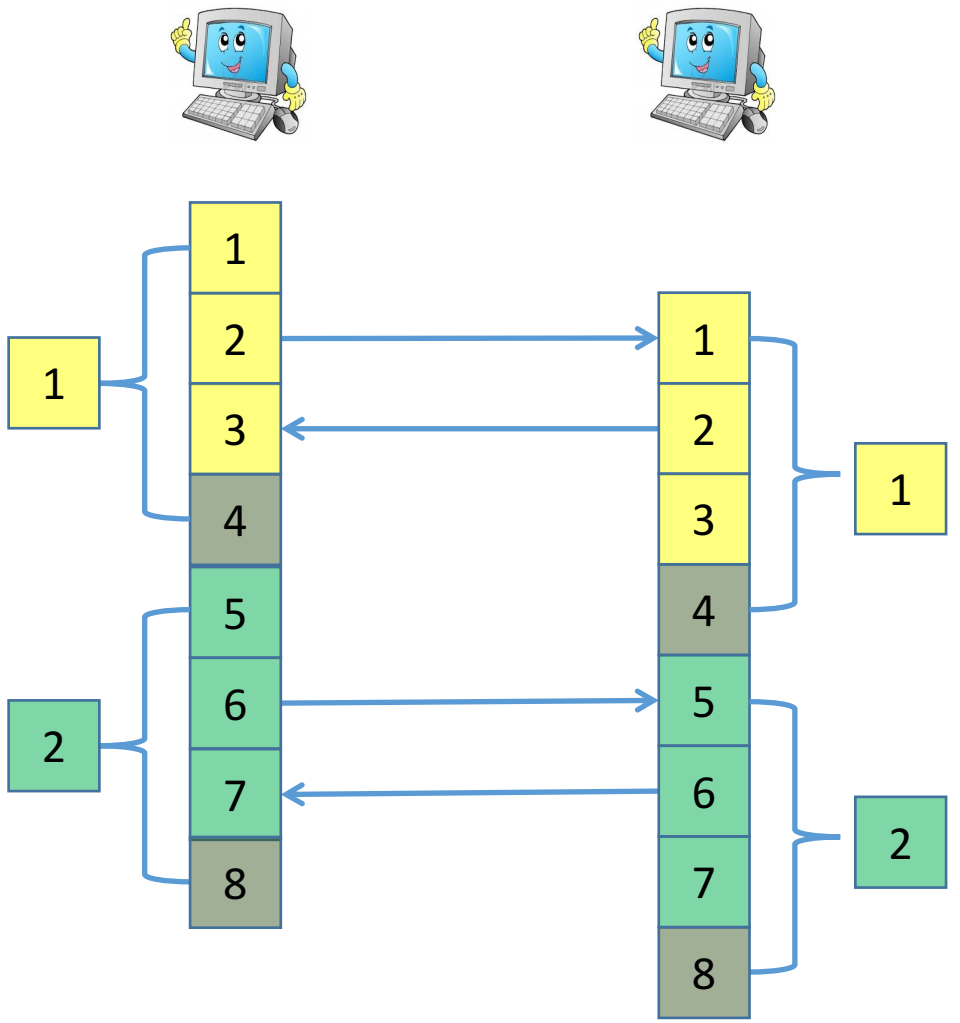


Non-Simultaneous Start

Each round extends to $3c + 1$ rounds:

- Listen for $2c + 1$ rounds
- Send in round $c + 1$
- Wait (without listening) for c rounds

Example: PSMT ($c = 1$)



Concurrent Composition

Round r messages
after round $r - 1$
before round $r + 1$

Each party proceeds in a locally sequential manner

Slack Tolerance and Reduction (PT)

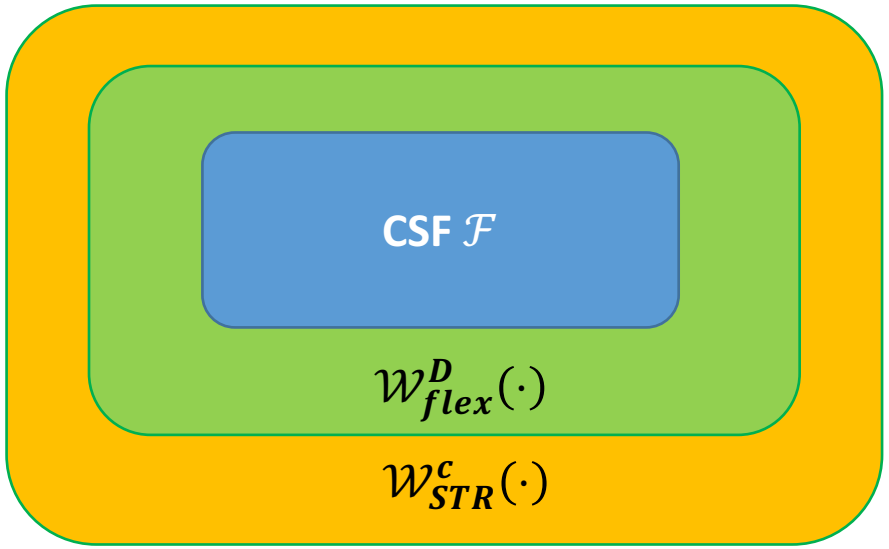
Hybrids introduce additional slack, rounds might blow-up

Use slack-reduction techniques [Bracha'84]

- Upon receiving output v, send (ok, v) to all the parties
- Upon receiving $t + 1$ messages (ok, v), accepts v
- Upon receiving $n - t$ messages (ok, v), terminates

Wrap $\mathcal{W}^{D}_{flex}(\mathcal{F})$ with $\mathcal{W}^{c}_{STR}(\cdot)$

Applies to <u>public-output</u> functionalities



Composition Theorem (Informal)

Denote $\mathcal{W}_{DT}^{c,D}(\mathcal{F}) = \mathcal{W}_{ST}^{c}\left(\mathcal{W}_{strict}^{D}(\mathcal{F})\right)$

$\mathcal{W}_{PT}^{c,D}(\mathcal{F}) = \mathcal{W}_{STR}^{c}\left(\mathcal{W}_{flex}^{D}(\mathcal{F})\right)$

Thm: Let $c \geq 0$ and $t < n/3$ (adaptive & perfect security)

Let π be an SNF protocol implementing a wrapped CSF $\mathcal{W}_{flex}^{D}(\mathcal{F})$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_\ell, \mathcal{F}'_1, \ldots, \mathcal{F}'_m)$-hybrid model, assuming all parties start at the same round

Then, $\mathrm{Comp}^{c}(\pi)$ implements $\mathcal{W}_{PT}^{c,\tilde{D}}(\mathcal{F})$ in the

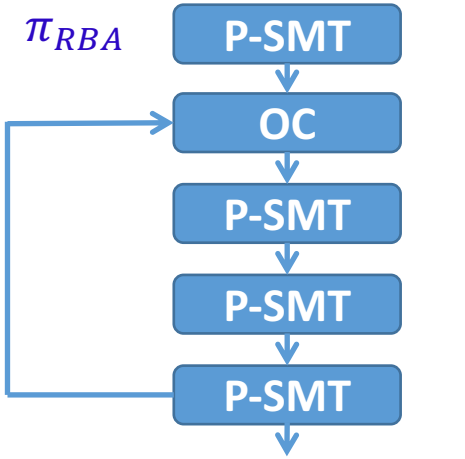
$$\left(\mathcal{W}_{PT}^{c,D_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{PT}^{c,D_\ell}(\mathcal{F}_\ell), \mathcal{W}_{DT}^{c,D'_1}(\mathcal{F}'_1), \ldots, \mathcal{W}_{DT}^{c,D'_m}(\mathcal{F}'_m)\right)$$

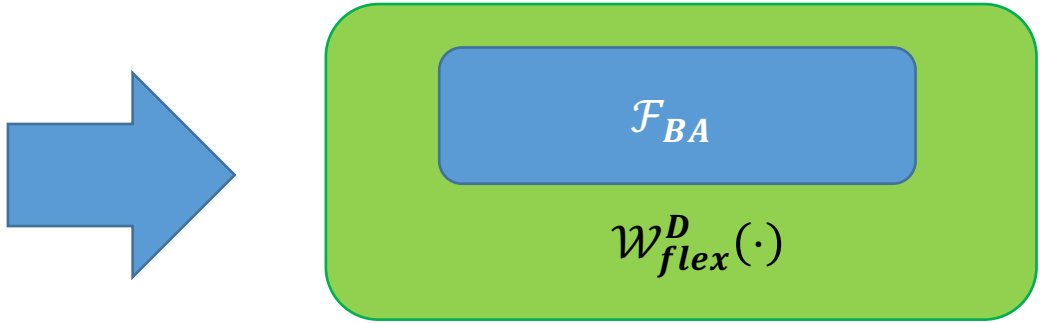
-hybrid model, assuming all parties start within c rounds

If each D_i (D'_i) has constant expectation then $\mathrm{Comp}^{c}(\pi)$ has (asymptotically) same round complexity as π, in expectation

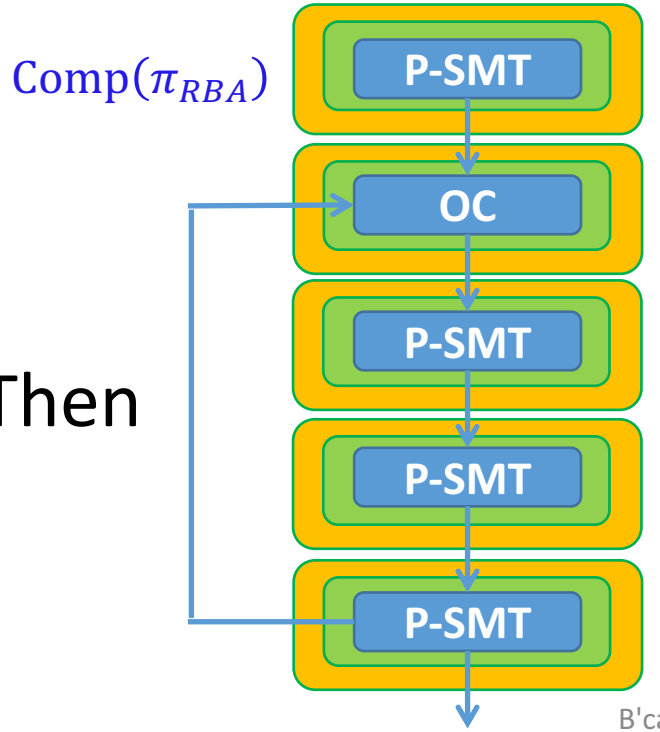
Corollary

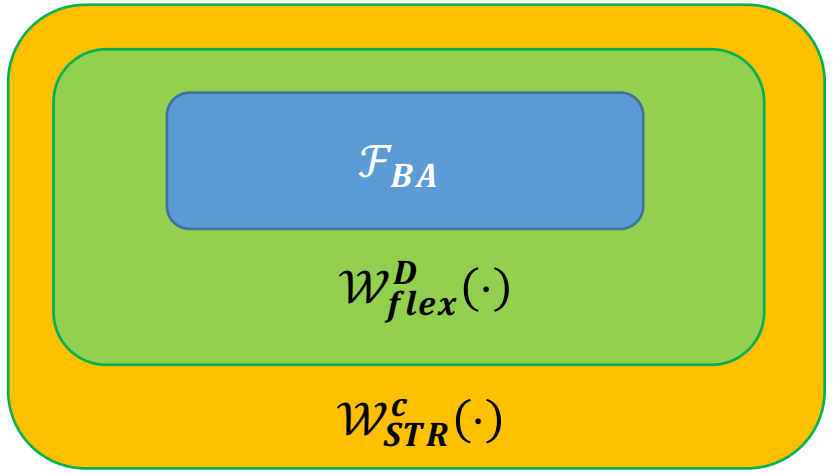
If





Then

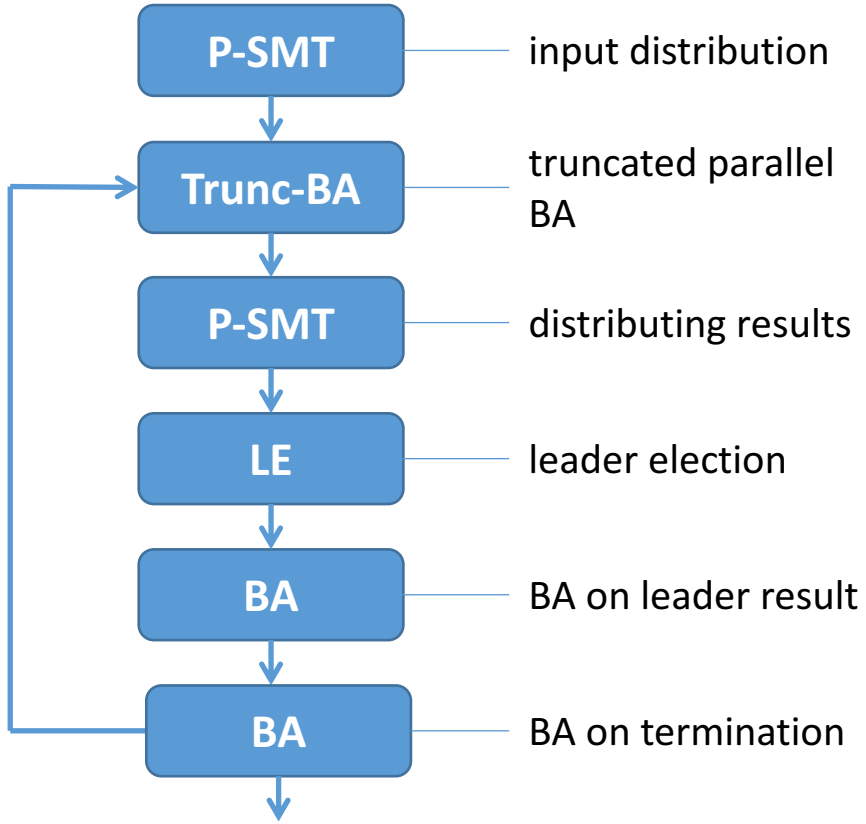




Applications of the PT Framework

Parallel Broadcast

- Running [FM'88] n times in parallel requires expected $\Theta(\log n)$ rounds
- Parallel broadcast in expected $O(1)$ [BE'03]
  - First round: each P_i distributes its input x_i
  - Proceeds in phases until termination



Parallel Broadcast (2)

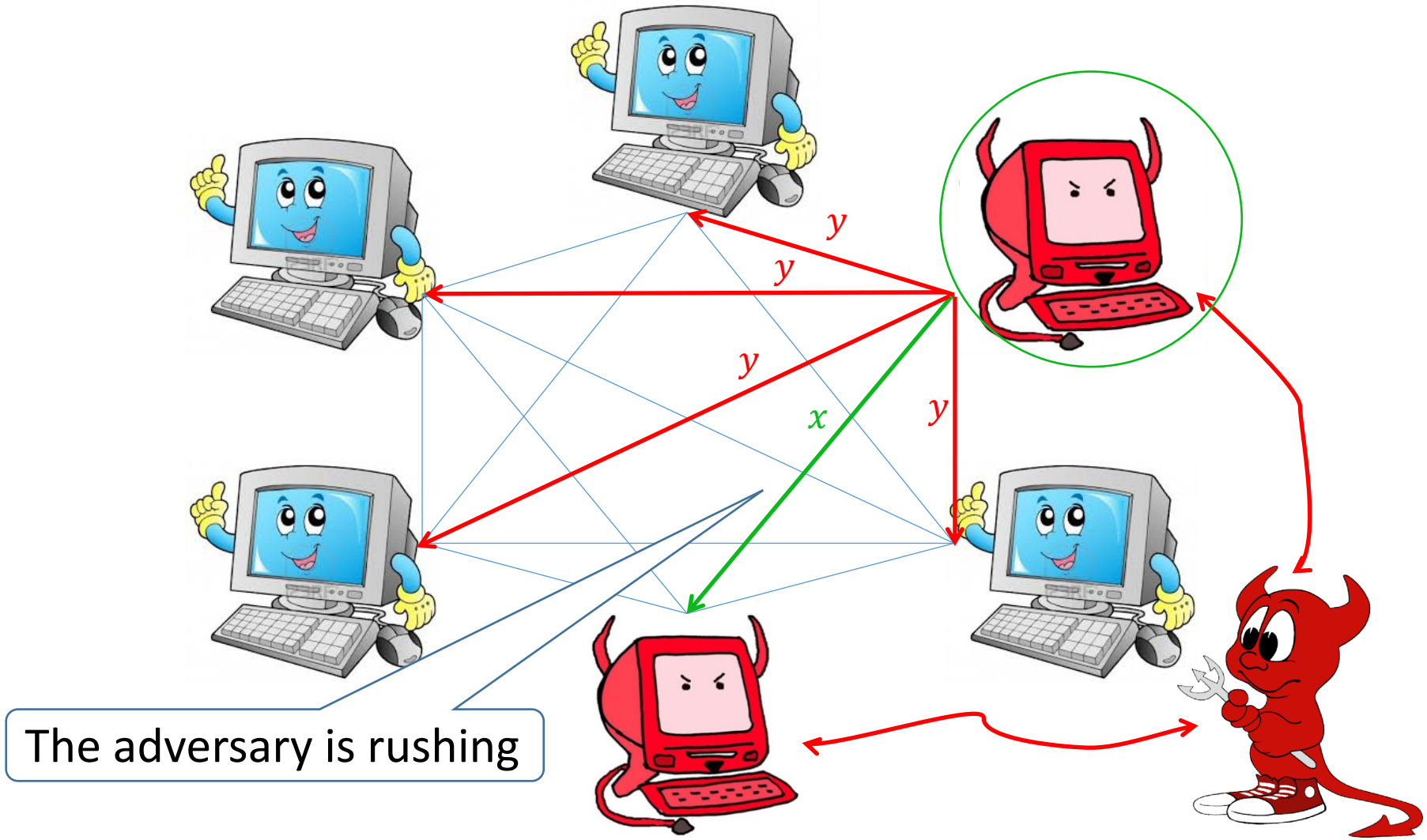
Thm [BE'03]: For appropriate parameters the protocol computes parallel broadcast in expected $O(1)$ rounds

Two issues:

1) No guaranteed termination: statistical security.
   We achieve perfect security (cf. [GP'90])
   - Run at most T phases
   - If not terminated, run a deterministic protocol
2) Adaptive security according to property-based definition (not simulation)

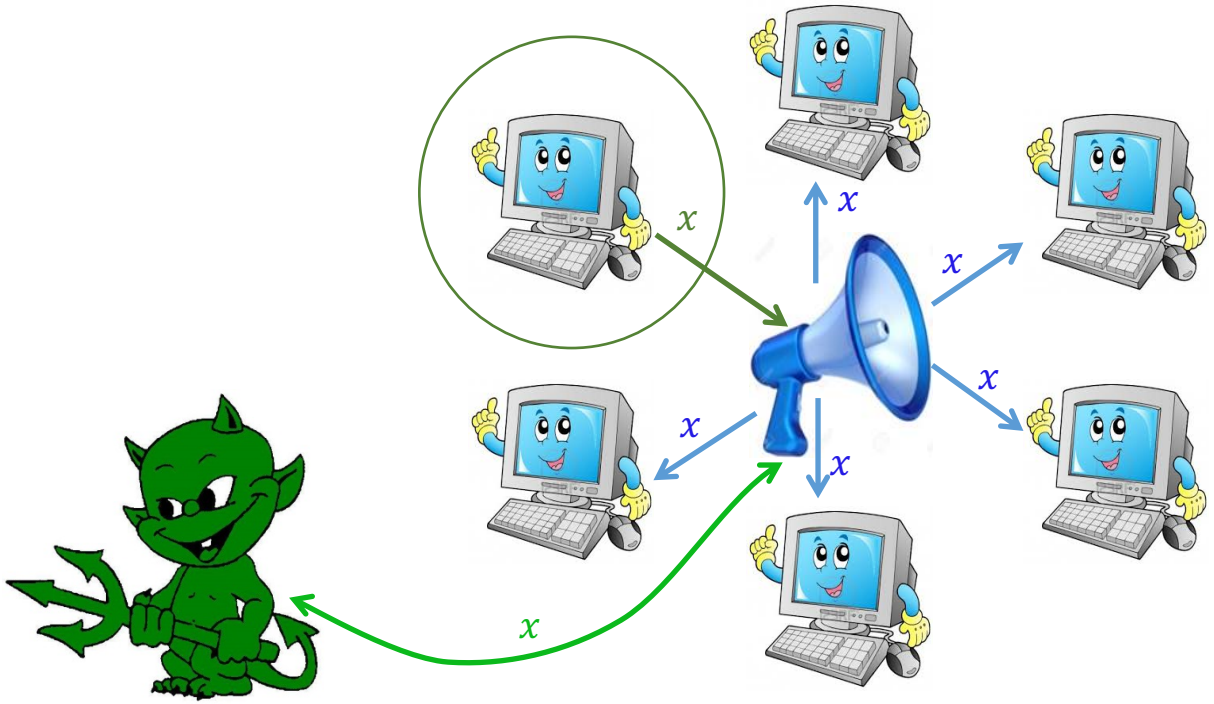
Attack on [BE'03]

Round 1: each party P_i distributes its input x_i



Attack on [BE'03]

- The adversary can corrupt an honest party and change its input x after the protocol started
- This behavior cannot be simulated in the ideal world (as in [HZ'10])



Unfair Broadcast

The ideal adversary is allowed to corrupt the sender and change its input – before any party received it

Def: Unfair broadcast for sender P_i is CSF with

- $f(x_1, \dots, x_n, a) = (x_i, \dots, x_i)$
- $l(x_1, \dots, x_n) = x_i$

The difference from broadcast is the leakage function

Thm: Protocol [BE'03] implements $\mathcal{W}_{flex}(\mathcal{F}_{U-PBC})$ in the $(\mathcal{F}_{PSMT}, \mathcal{F}_{LE}, \mathcal{F}_{BA}, \mathcal{F}_{Trunc-BA})$-hybrid model, for $t < n/3$, assuming all parties start at the same round

Unfair Parallel Bcast ⇒ Parallel Bcast

Before P_i distributes x_i, it commits to its input

1) Each party secret shares its input using $(t+1)$-out-of-n secret sharing
2) Each party broadcasts all the shares it received using an unfair parallel broadcast channel
3) Reconstruct and output the values

Intuition: In round 1 $\mathcal{A}$ only learns random shares

In round 2 $\mathcal{A}$ can change only $t < n/3$ shares

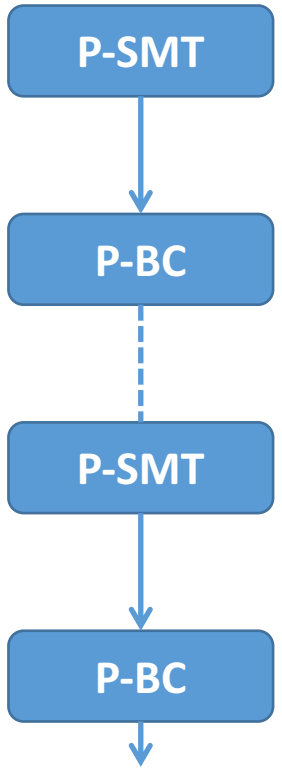
⇒ Inputs of parties that are honest in round 1 (before $\mathcal{A}$ learns anything) are reconstructed properly

Thm: $\mathcal{W}^{D}_{flex}(\mathcal{F}_{PBC})$ can be implemented in the $(\mathcal{F}_{PSMT}, \mathcal{F}_{U-PBC})$-hybrid model, for $t < n/3$, assuming all parties start at the same round

SFE with Expected $O(d)$ Rounds

Thm: Protocol [BGW'88] implements $\mathcal{W}_{flex}^{D}(\mathcal{F}_{SFE})$ in the $(\mathcal{F}_{PSMT}, \mathcal{F}_{PBC})$-hybrid model in $O(d)$ rounds, assuming all parties start at same round

Thm: Let $c \geq 0$.
$\mathcal{W}_{PT}(\mathcal{F}_{SFE})$ can be implemented in the $\left(\mathcal{W}_{DT}(\mathcal{F}_{PSMT}), \mathcal{W}_{PT}(\mathcal{F}_{PBC})\right)$-hybrid model in expected $O(d)$ rounds, assuming all parties start within c rounds



PT Summary

We considered the composability of cryptographic protocols with probabilistic termination

- "PT framework" for designing cryptographic protocols in stand-alone fashion and compiler to fast composition in the UC framework
- Perfect, adaptively secure protocols in the P2P model
  1) BA with expected $O(1)$ rounds
  2) Parallel broadcast with expected $O(1)$ rounds
  3) SFE with expected $O(d)$ rounds

Lecture's Roadmap

I. Introduction
II. Brief Recap
- B'cast/consensus definitions, models, protocols

III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

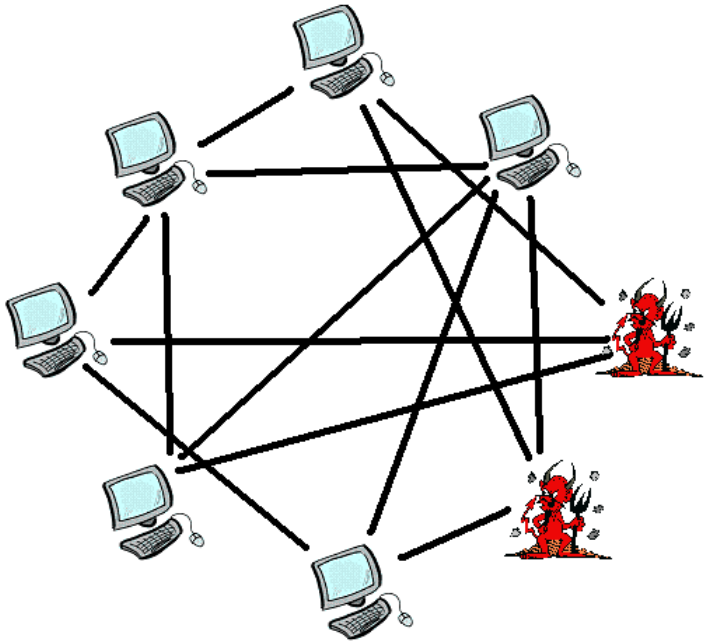
IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

Almost-Everywhere B'cast/Agreement [DPPU'86]

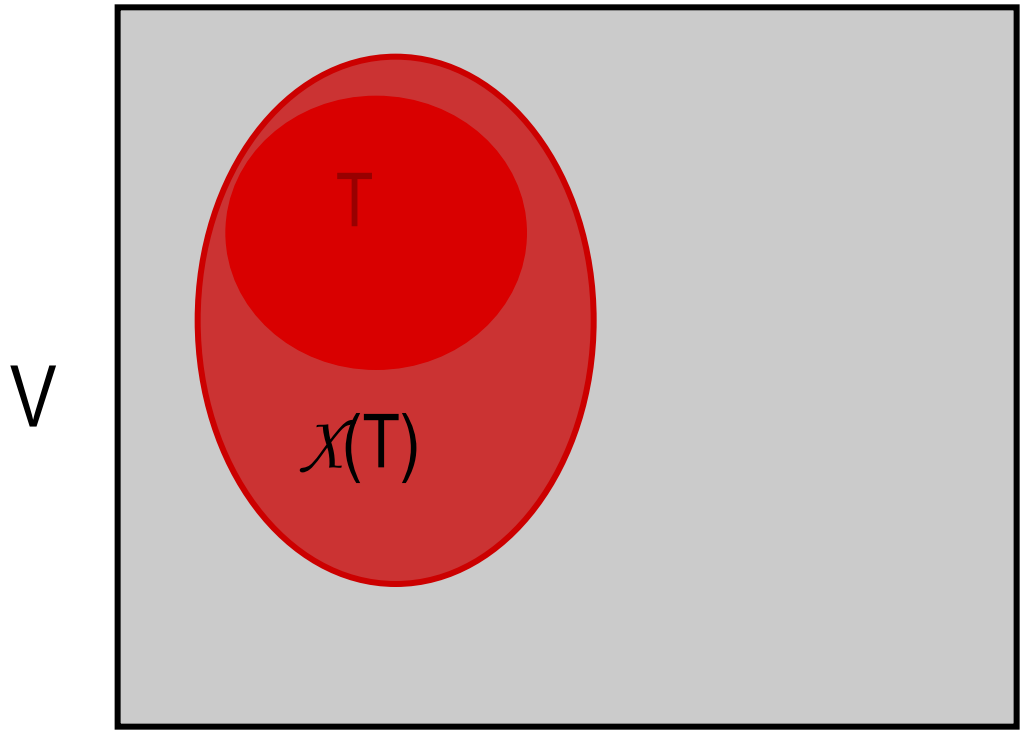
- Unconditionally secure b'cast/consensus:
  - Possible iff < 1/3 of parties are corrupt [LSP'82]
  - Authentic (private) point-to-point channels sufficient…



… but what if **only some of the nodes** are connected?

Almost-Everywhere Agreement (Consensus) [DPPU86]

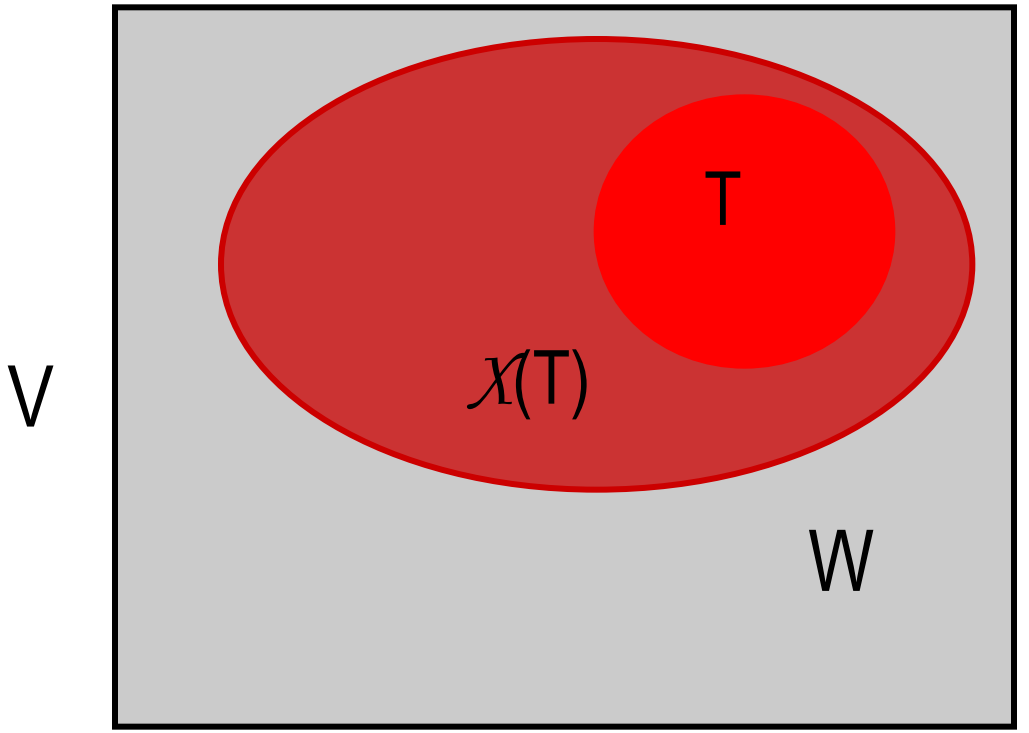
- "Give up" some of the players; guarantee agreement for a large fraction of them
- Adv. *implicitly* corrupts by corrupting sufficiently many neighbors
- $G_n = (V,E)$



Almost-Everywhere Agreement (Consensus) [DPPU86]

- "Give up" some of the players; guarantee agreement for a large fraction of them
- Adv. *implicitly* corrupts by corrupting sufficiently many neighbors
- $G_n = (V,E)$

- W: *Privileged* set



Almost-Everywhere Agreement (Consensus) [DPPU86] (2)

- $G = (V,E)$, $|T| = t$, $\mathcal{P} = 2^V$
- $\mathcal{X}: \mathcal{P}^{(\leq t)} \rightarrow \mathcal{P}$
  1. $T_1 \subseteq T_2 \Rightarrow \mathcal{X}(T_1) \subseteq \mathcal{X}(T_2)$
  2. $T \subseteq \mathcal{X}(T)$

  $X = \max_T \{|\mathcal{X}(T)|\}$

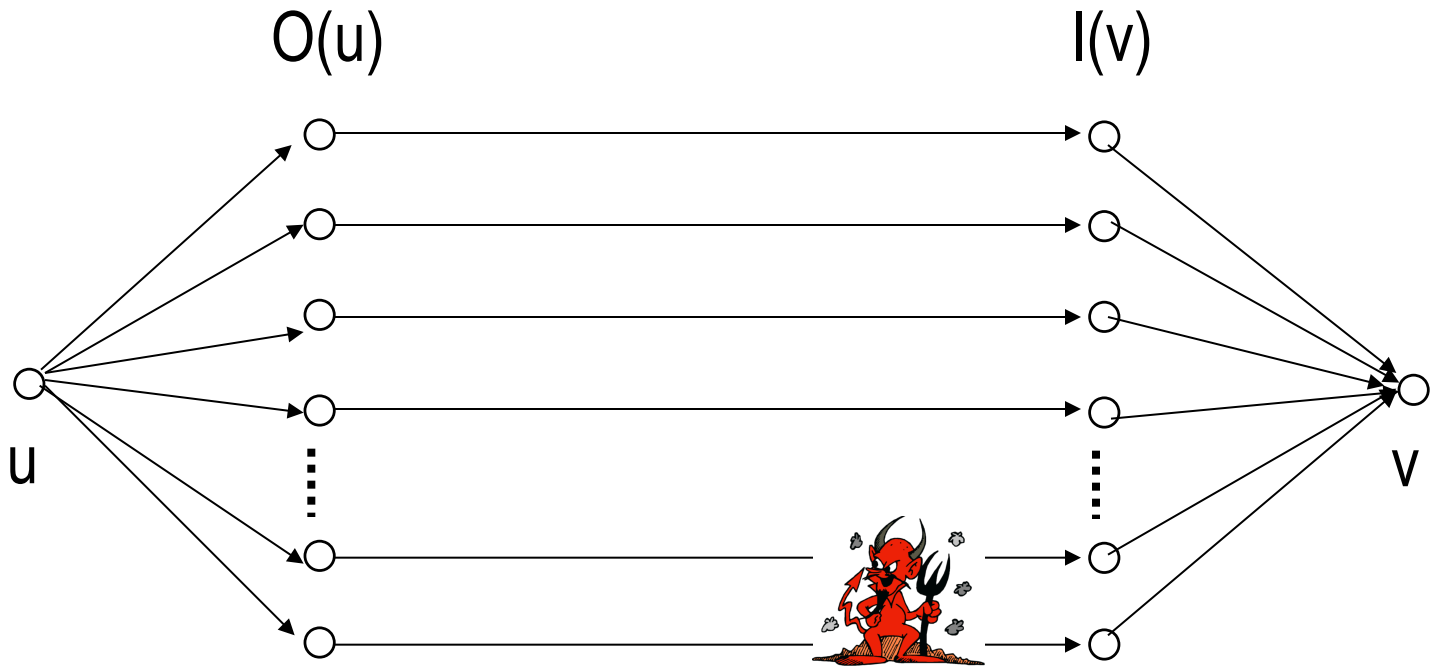
  Protocol π achieves **X-agreement** if $\forall T \exists W, |W| \geq n - X$, s.t. all parties in W are able to reach agreement

- Fully connected network: $\mathcal{X}(T) = T$

Almost-Everywhere Agreement (Consensus) [DPPU86] (3)

- **Transmission scheme** to simulate sending of a message between any two nodes
- If nodes $\in$ W (= V – $\mathcal{X}$(T)), then simulation is faithful
- $\Rightarrow$ Possible to simulate BA protocol for fully connected networks treating processors in $\mathcal{X}$(T)) as faulty
- "Almost-everywhere *broadcast*"

Transmission Scheme [DPPU86]



- v takes *majority* of received copies

X-Agreement on Classes of Networks

- **Objective:** Large sets T, "small" $\mathcal{X}(T)$
- G_n of *constant* degree (butterfly, expander graphs),
  $t = O(n/\log n) \rightarrow O(t)$-agreement [DPPU86]
- G_n of degree $O(n^{\varepsilon})$, $t = O(n) \rightarrow O(t)$-agreement [DPPU86]

Comb. Characterization of Fault-tolerant Networks

Theorem [DPPU86]: Let G_n be a communication graph and $T_i \in \mathcal{P}^{(\leq t)}$. There exists a protocol π such that processors in W_i reach agreement if and only if for every pair of processors $u,v \in W_i \cap W_j$, the set $T_i \cup T_j$ *does not disconnect* u from v in G_n.

X-Agreement on Classes of Networks

- **Objective:** Large sets T, "small" $\mathcal{X}(T)$
- G_n of *constant* degree (butterfly, expander graphs),
  $t = O(n/\log n) \rightarrow O(t)$-agreement [DPPU86]
- G_n of degree $O(n^{\varepsilon})$, $t = O(n) \rightarrow O(t)$-agreement [DPPU86]
- G_n of *constant* degree, $t = O(n) \rightarrow O(t)$-agreement* [Upf92]

  * Inefficient

  Most communication paths between pairs of nodes in W contain one corrupted node

X-Agreement on Classes of Networks

- **Objective:** Large sets T, "small" $\mathcal{X}(T)$
- G_n of *constant* degree (butterfly, expander graphs),
  $t = O(n/\log n) \rightarrow O(t)$-agreement [DPPU86]
- G_n of degree $O(n^{\varepsilon})$, $t = O(n) \rightarrow O(t)$-agreement [DPPU86]
- G_n of *constant* degree, $t = O(n) \rightarrow O(t)$-agreement* [Upf92]

  * Inefficient

  Most communication paths between pairs of nodes in W contain one corrupted node
- G_n of *logarithmic* degree, $t = O(n) \rightarrow O(t)$-agreement [CGO09]

O(n)-Agreement on Logarithmic-Degree Graphs [CGO09]

- We want O(t)-agreement with t = O(n) on graph with degree O(log n/log log n)
- **Idea:** Recursively apply [Upf92] on graphs of size O(log n), plus other tools (*Differential* agreement [FG03])

X-Agreement (MPC) on Classes of Networks (2)

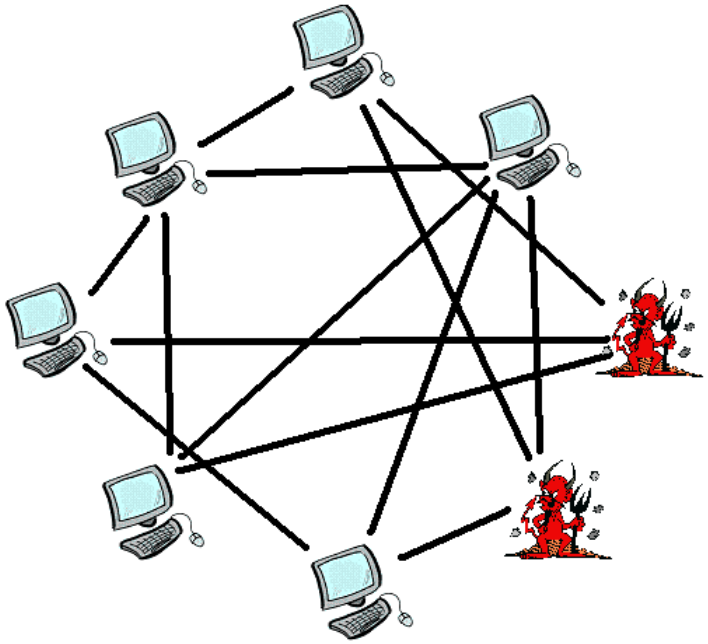
GOAL: Minimize degree and number of doomed nodes while maximizing number of corruptions handled

	Degree of graph ⬇ d	# Corrupted nodes ⬆ t	# Doomed nodes ⬇ x	Running time of algorithm
[DPPU86]	$\mathcal{O}(1)$	$\mathcal{O}(\frac{n}{\log n})$	$\mathcal{O}(t)$	Polynomial
[DPPU86]	$\mathcal{O}(n^{\epsilon})$	$\mathcal{O}(n)$	$\mathcal{O}(t)$	Polynomial
[Upfal92]	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(t)$	Exponential
[CGO10]	$\mathcal{O}(\log^{q} n)$	$\mathcal{O}(n)$	$\mathcal{O}(\frac{t}{\log n})$	Polynomial

Constants $0 < \epsilon < 1 < q$

Almost-Everywhere MPC [GO'08]

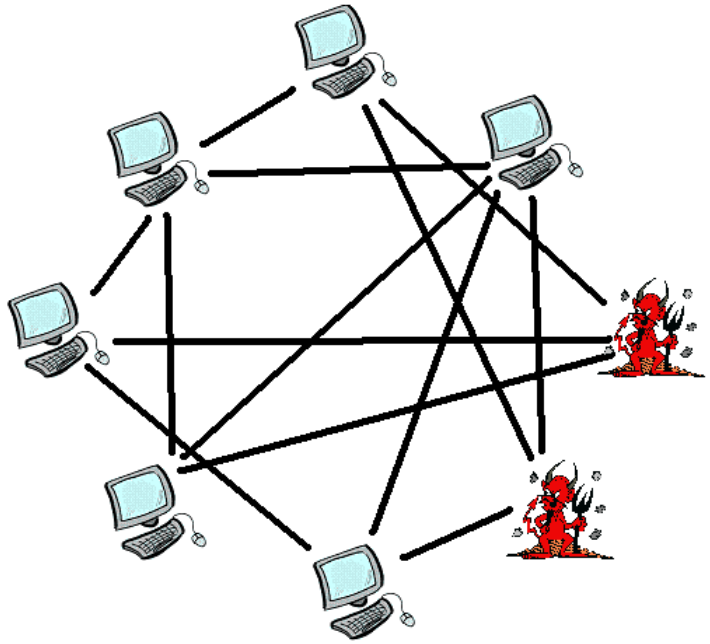
- Unconditionally secure MPC:
  - Possible iff < 1/3 of parties are corrupt [BGW'88, CCD'88]
  - Private point-to-point channels sufficient…



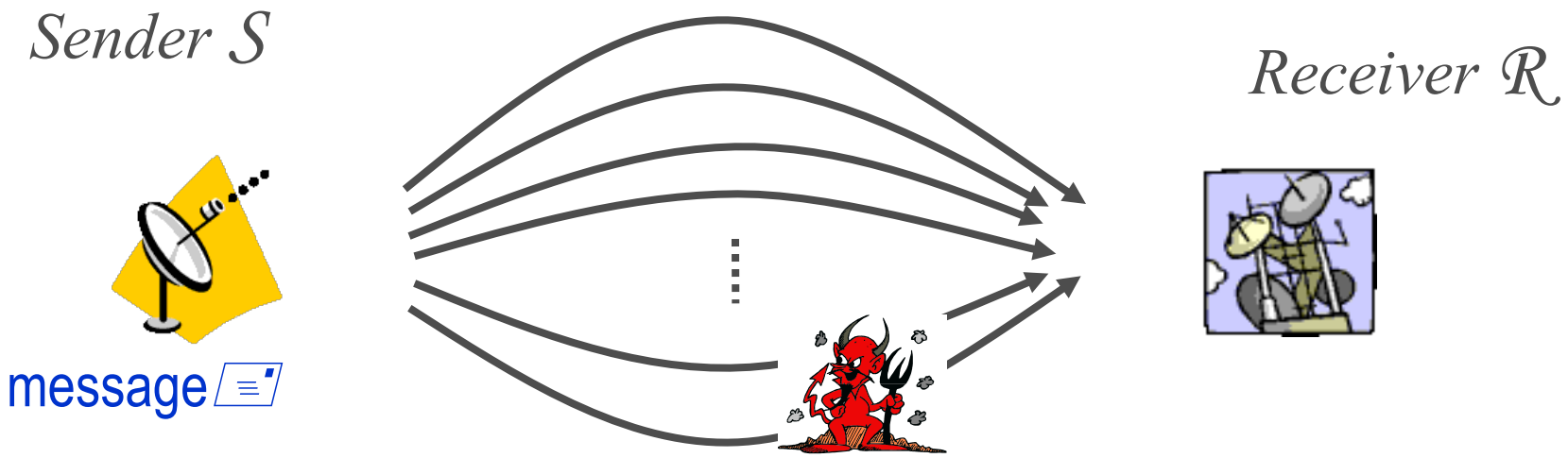
… but what if **only some of the nodes** are connected?

Almost-Everywhere MPC [GO'08] (2)

- **Idea!**: Simulate *private* p2p channels using SMT protocol
  - Easy with connectivity at least 2t+1



Secure Message Transmission (SMT) [DDWY'93]

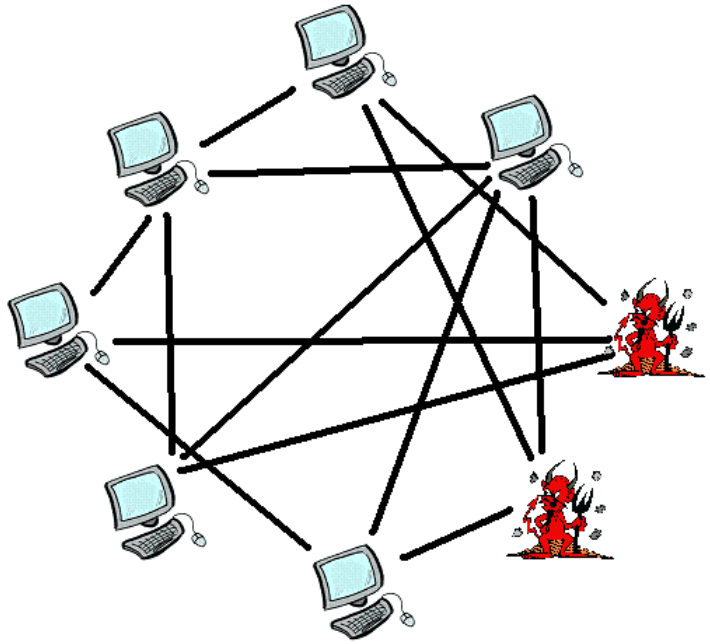


Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- ***t*** wires (actively) corrupted by adversary $\mathcal{A}$

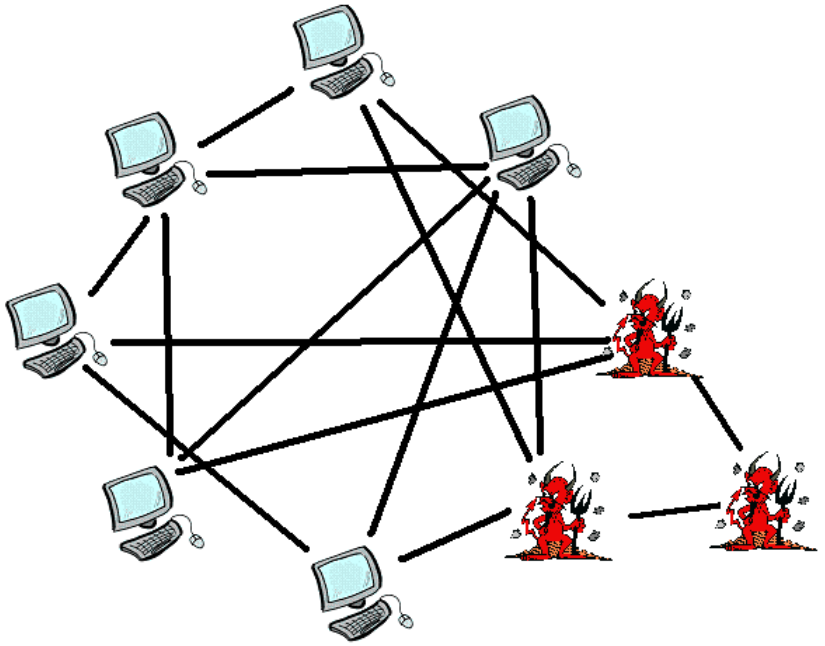
Almost-Everywhere MPC [GO'08] (3)

- **Idea!**: Simulate *private* p2p channels using SMT protocol
  - Easy with connectivity at least 2t+1
  - … Can we do **better**?



SMT-PD to the Rescue!

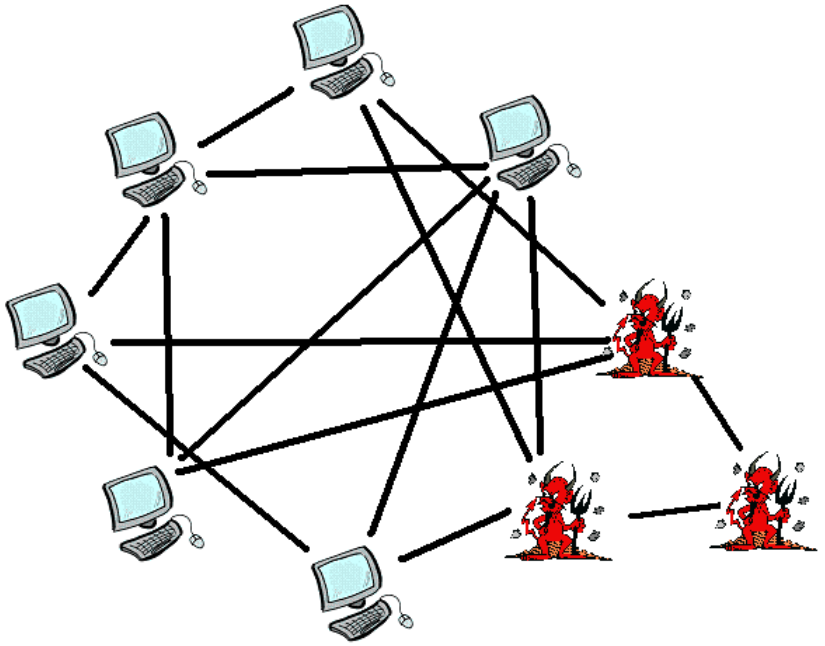
- Yes! Can even get **constant** connectivity (!) [GO'08]
  - …some of the good guys might be totally cut off from the others…



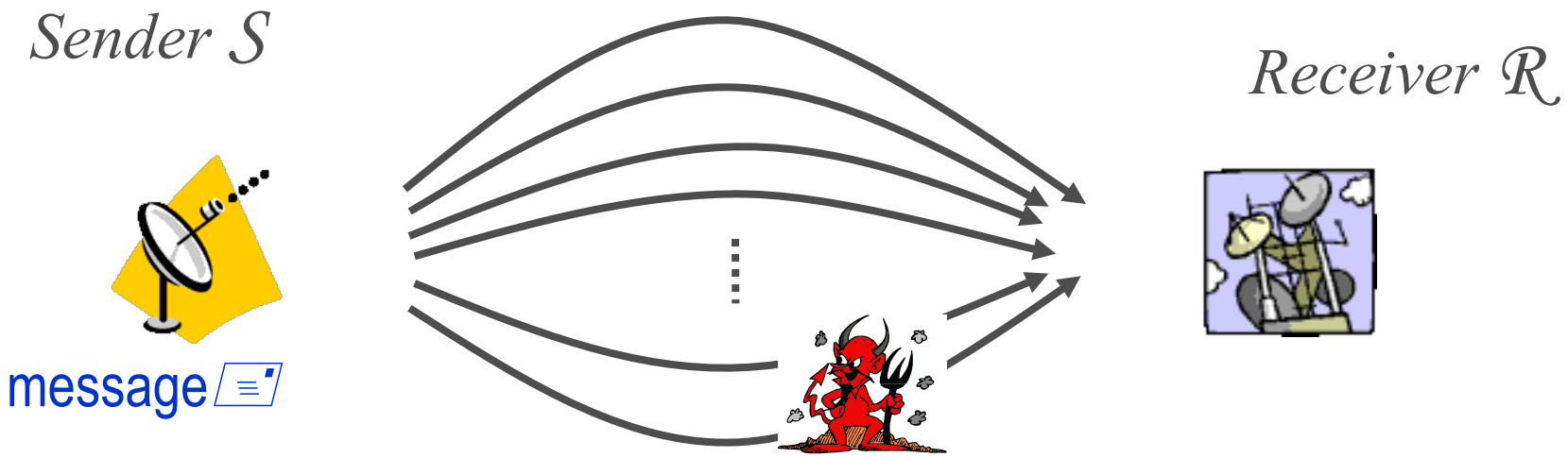
- So we give up on correctness and privacy for these poor lost souls ($\mathcal{X}(\mathsf{T})$)

SMT-PD to the Rescue! (2)

- Idea!: Simulate private p2p channels between nodes using *SMT-PD* protocol
  - Possible even if just one good path!



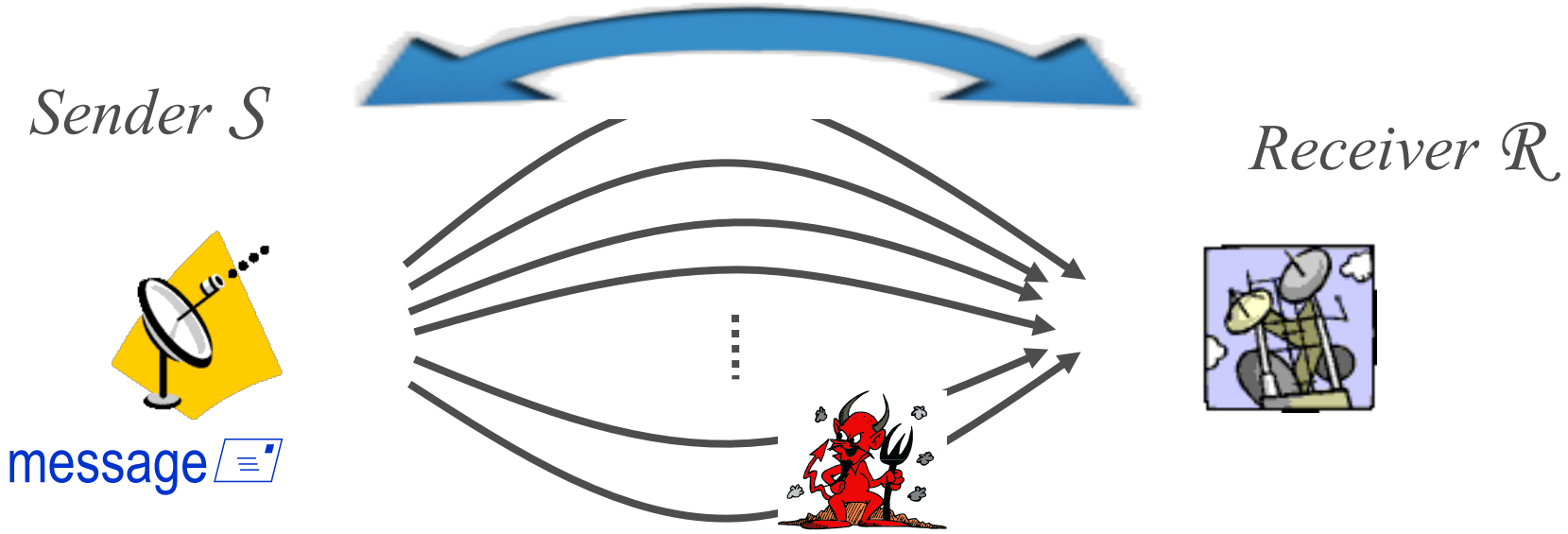
SMT *by Public Discussion* (SMT-PD) [GO'08,GGO'10]



Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- ***t*** wires (actively) corrupted by adversary $\mathcal{A}$

SMT *by Public Discussion* (SMT-PD) [GO'08,GGO'10]

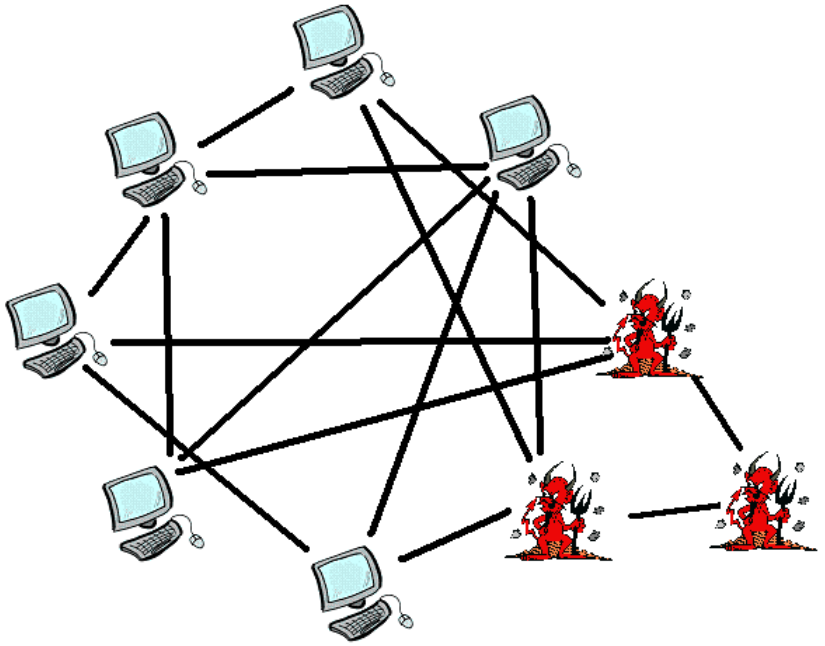


Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- ***t*** wires (actively) corrupted by adversary $\mathcal{A}$
- … plus an (authentic and reliable) *public channel*

SMT-PD to the Rescue! (3)

- Idea!: Simulate private p2p channels between nodes using *SMT-PD* protocol
  - Possible even if just one good path

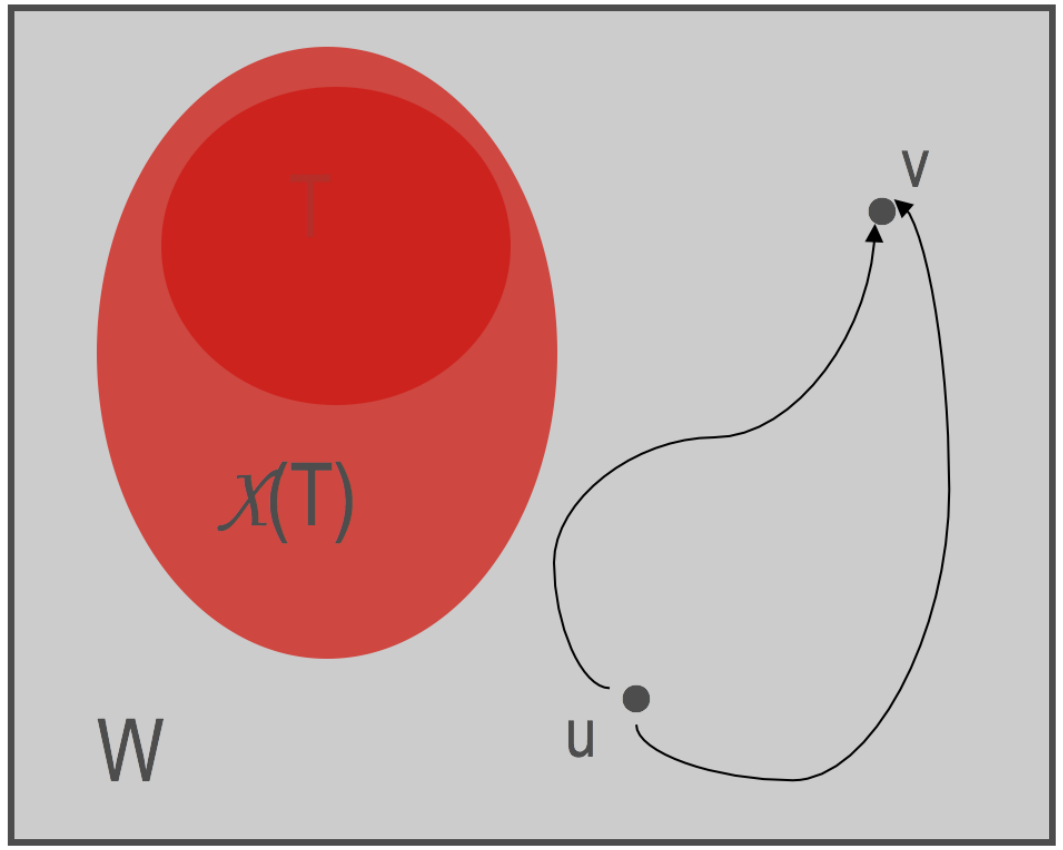


- Use almost-everywhere broadcast to implement public channel

Almost-Everywhere MPC [GO'08] (3)

- How?
  1. AE-broadcast in W
  2. Multiple *paths* between u,v ∈ W, some not corrupted
- **Thm:** MPC possible in G_n iff $n > 3 \cdot |\mathcal{X}(T)|$

Intuition: 1 + 2 = SMT-PD!



Lecture's Roadmap

I. Introduction
II. Brief Recap
- B'cast/consensus definitions, models, protocols

III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

Complexity Measures

RECALL

- **Rounds:** $r = t+1$ [LSP82, FL82]
- **Resiliency:**
  - Unconditional setting: $n > 3t$ [LSP82]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Agreement: $n > 2t$ [Fit03]
- **Message/Bit complexity:** $m = \Omega(n^2)$ [DR85,BGP92,CW92]

Complexity Measures

RECALL

- **Rounds:** $r = t+1$ [LSP82, FL82]
- **Resiliency:**
  - Unconditional setting: $n > 3t$ [LSP82]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Agreement: $n > 2t$ [Fit03]
- **Message/Bit complexity:** $m = \Omega(n^2)$ [DR85,BGP92,CW92]

Authenticated Consensus Protocols (2)

RECALL

- [DS82] protocol (informal):
  - Source signs its input value and sends to all parties
  - r = 1,…,t+1:
    - If any value $v_i \in V = \{0,1\}$ has been *newly* added to a set of accepted values, sign it and send value and signatures to everybody
    - If a value/signatures message is received by any party containing valid signatures by at least r distinct players including the sender, then accept the value and update signatures
  - If only one accepted value, then the party outputs that value; otherwise a default value

IT-Authenticated Broadcast/Consensus

- Use information-theoretic authentication instead of digital signatures
- Additional trusted setup or assumption
  1. Physical broadcast available at onset of the computation [PW'96]
     - Based on *anonymous channels* [Cha'88], which in turn uses VSS
  2. *Secret* Key Infrastructure (SKI)

Pseudosignatures [PW'96]

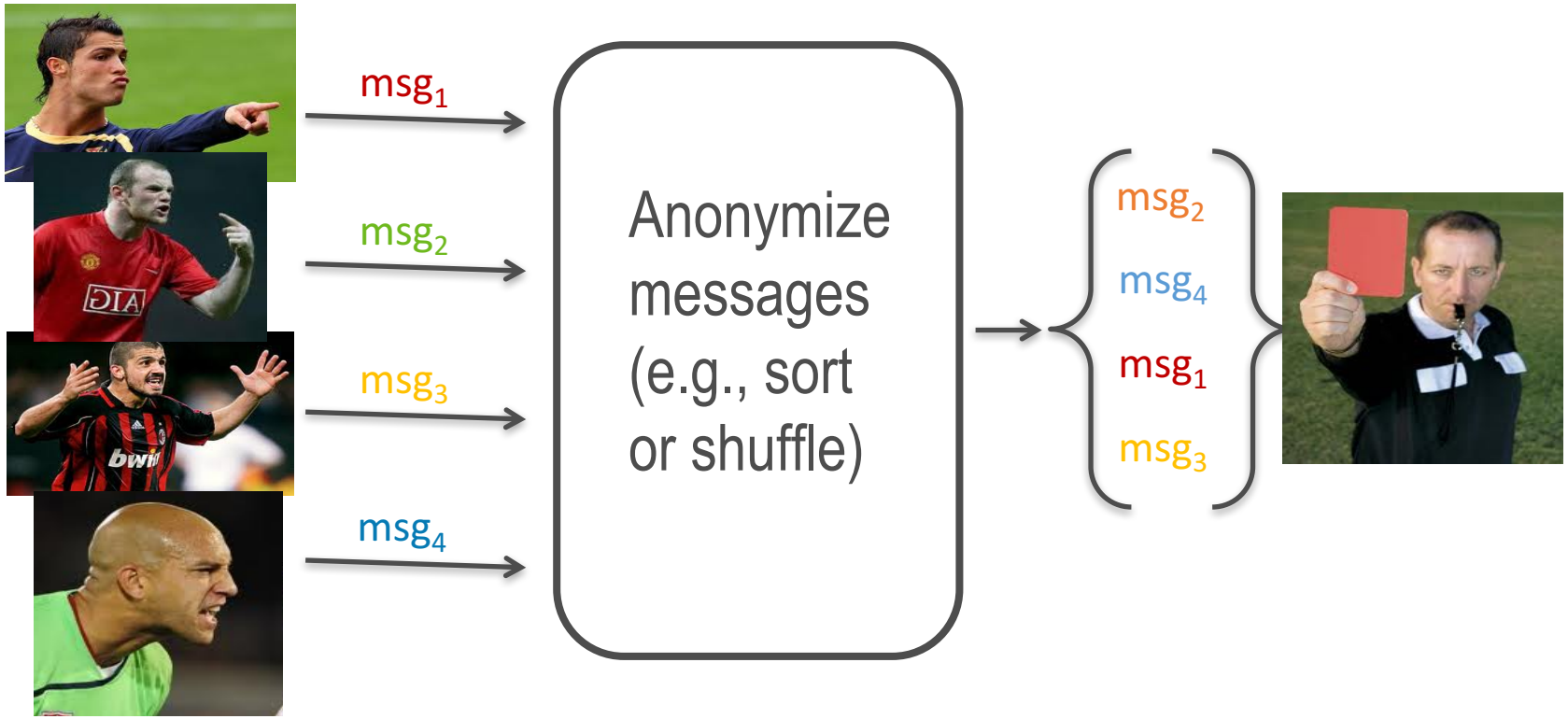
- *Information-theoretic* signature scheme
  - For a fixed-in-advance set of players
  - Verification **keys** are kept secret
  - Uses MACs
  - Needs physical broadcast setup
  - Only *bounded transferability* of signatures

- Once we have them, we can implement *authenticated* broadcast protocol (e.g., [DS83,KK06]; tolerate $n > t$)

  → No more physical broadcasts required!

Pseudosignatures [PW'96] (2)

- Based on **sender-anonymous channel** [Cha'88,GGOR'14]]:



Anonymous Channel: Security Requirements

- Even a cheating Receiver learns no more about honest senders' inputs than the multiset of them (Anonymity)
- Honest Receiver correctly gets all honest messages (Correctness)
- Cheating players have *zero information* on value of honest players' messages, for honest Receiver (Privacy)
- Cheating players' messages are independent of honest players' messages, for honest Receiver (Non-Malleability)

From Anonymous Channel to Pseudosignatures

- Every party sends random keys, anonymously, to Signer.
  Repeat the process "several" (say, p) times.
- Signer receives p ***signature blocks*** of keys
  $B_1 = ((a_{11},b_{11}),\ldots, (a_{1n},b_{1n}))$, …, $B_p = ((a_{p1},b_{p1}),\ldots, (a_{pn},b_{pn}))$

  $\text{Signature}(M) = (a_{11}M \oplus b_{11}, \ldots, a_{1n}M \oplus b_{1n}),$
  $(a_{21}M \oplus b_{21}, \ldots, a_{2n}M \oplus b_{2n}),$
  …
  $(a_{p1}M \oplus b_{p1}, \ldots, a_{pn}M \oplus b_{pn}).$
- 1st verifier: Given (M, σ), verify *all* blocks have correct $aM \oplus b$
- 2nd verifier: Verify *most* blocks have correct $aM \oplus b$
- 3rd verifier: …a *fair number*…

Constant-Round Anonymous Channel [GGOR'14]

- Anonymous channel protocol for $t < n/2$, using only black-box access to a linear VSS protocol
- Protocol is constant-round, and uses no additional broadcast rounds beyond those required by VSS
  - Physical broadcast used by VSS
- *Broadcast complexity*: $B_{share} + B_{rec}$
- Using [GGOR'13] VSS protocol: $B_{share} = 2$, $B_{rec} = 0$

$\rightarrow$ Physical broadcast only used twice

Lecture's Roadmap

I. Introduction
II. Brief Recap
- B'cast/consensus definitions, models, protocols

III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

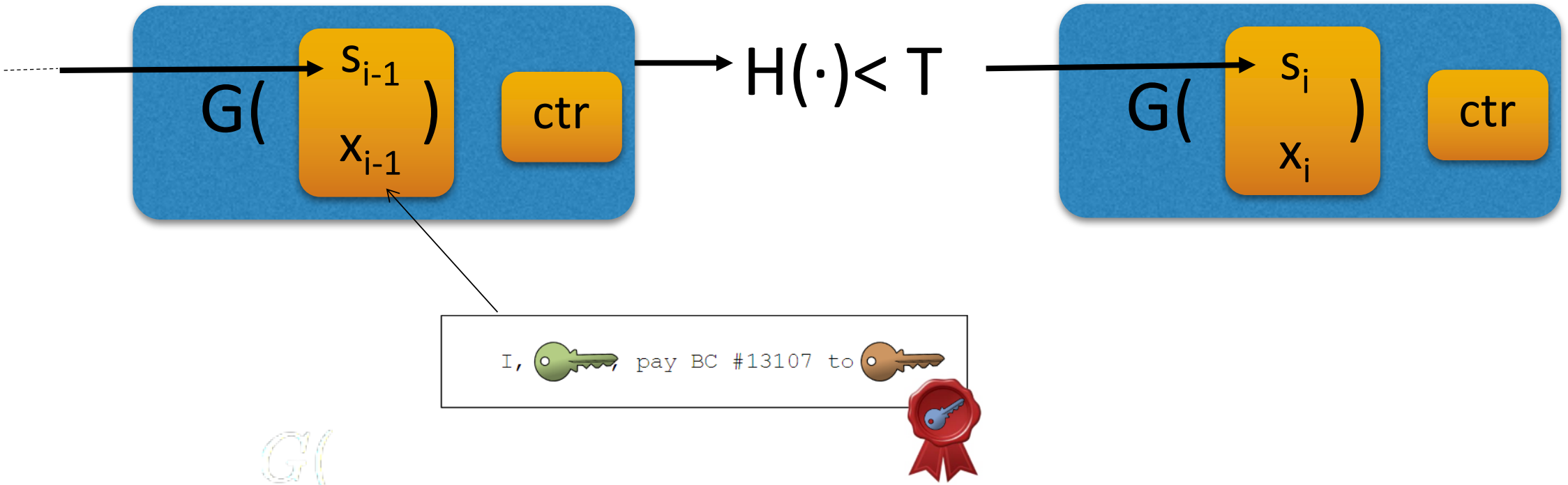
Blockchain-based Consensus

- Consensus in the standard setting
  - Information-theoretic setting: $t < 1/3\ n$ [PSL80, LSP82, GM93]
  - Cryptographic setting:
    - Assuming a Public Key Infrastructure (PKI): $t < 1/2\ n$ [DS83]
    - No PKI: $t < 1/3\ n$ [Bor96, Fit03]

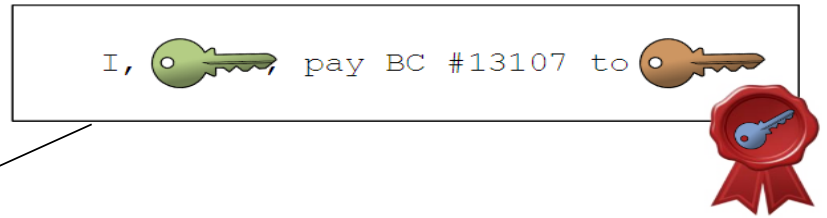
- Consensus in a non-standard (blockchain) setting
  - Open, peer-to-peer
  - No authenticated, point-to-point channels
    - "No port awareness": deterministic consensus is impossible [Oku05]
  - Use POWs to establish PKI, then run standard consensus protocol [AJK05, AD15]
  - *POW-based* consensus protocols [GKL'15,GKLP'18]

What Is a Blockchain?

- Parties ("miners") have to do work in order to install a transaction
- Parties have a *state* C of the form:



Using Proofs of Work (POWs



- Miners collect a set of transactions

  $\mathbf{tx} = (tx_1, tx_2, \ldots, tx_m)$

- Then do "work"

  ctr := 0; **while** Hash(ctr; Hash(τ,**tx**)) > T **do** ctr++

  T: block's "target" (*difficulty level*)

  (T = 0000000000000000171A8B0000000000000000000000000000000000000000)

- If **while** loop terminates "broadcast" (τ,ctr,**tx**) (new "block": state, counter, set of transactions)

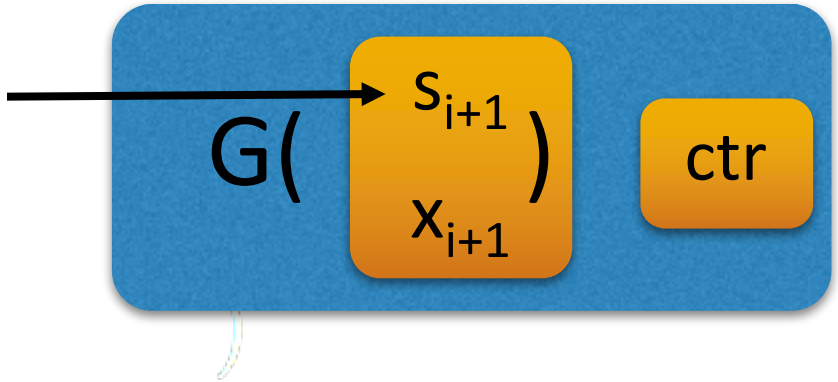
SHA-256(·)

What Is a Blockchain? (2)

- Within a round, players receive inputs from the environment and the network and process them:

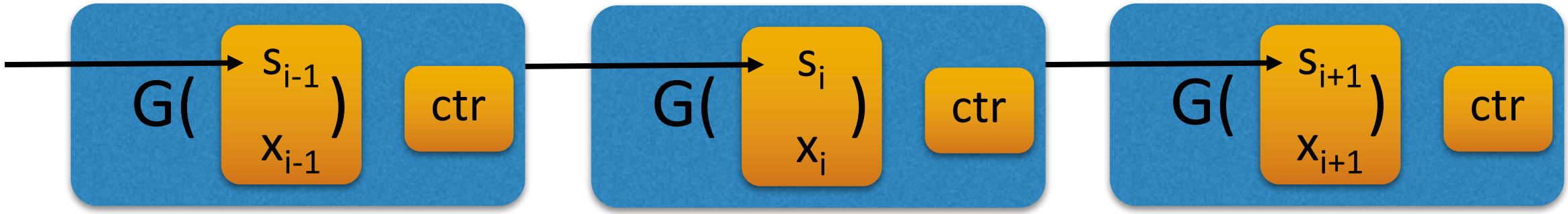
$$x_{i+1} := I(\ldots \text{all local info} \ldots)$$

- Then they use their q queries to H() to obtain a new block by trying ctr = 0,1,2,…



What Is a Blockchain? (3)

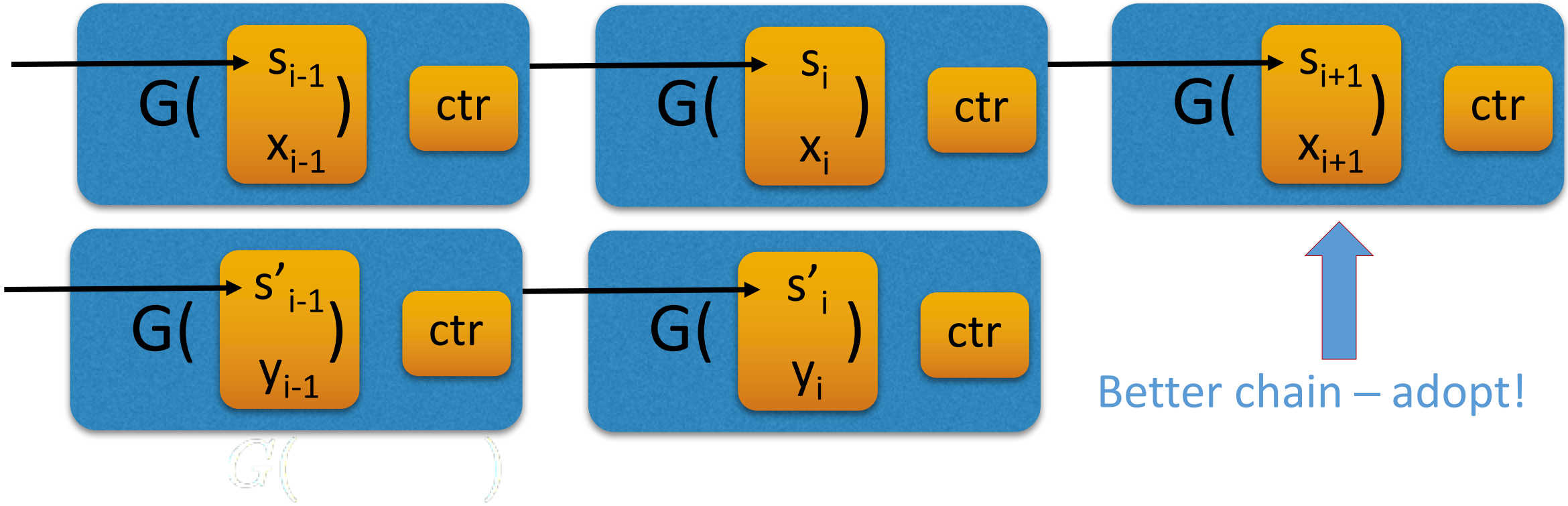
- If a player solves a POW, it extends C:



- The new C is propagated to all players via the (unreliable/ anonymous) diffusion mechanism ("multicast")

What Is a Blockchain? (4)

- A party will compare any incoming chains and the local chain wrt their length/difficulty:



Nakamoto's Consensus Protocol [Nak'08]

- "The proof-of-work chain is a solution to the Byzantine Generals' Problem…"

The Bitcoin developer Satoshi Nakamoto described the problem this way:

> A number of Byzantine Generals each have a computer and want to attack the King's wi-fi by brute forcing the password, which they've learned is a certain number of characters in length. Once they stimulate the network to generate a packet, they must crack the password within a limited time to break in and erase the logs, lest they be discovered. They only have enough CPU power to crack it fast enough if a majority of them attack at the same time.
> They don't particularly care when the attack will be, just that they agree. It has been decided that anyone who feels like it will announce an attack time, which we'll call the "plan", and whatever plan is heard first will be the official plan. The problem is that the network is not instantaneous, and if two generals announce different plans at close to the same time, some may hear one first and others hear the other first.

Nakamoto's Consensus Protocol [Nak'08] (2)

- The *n* parties start building a blockchain inserting their input
- If a party receives a longer blockchain, it switches to that one and switches its input
- When the blockchain is long enough the party outputs the (unique) value that it contains

Nakamoto's Consensus Protocol [Nak'08] (3)

- The *n* parties start building a blockchain inserting their input
- If a party receives a longer blockchain, it switches to that one and switches its input
- When the blockchain is long enough the party outputs the (unique) value that it contains

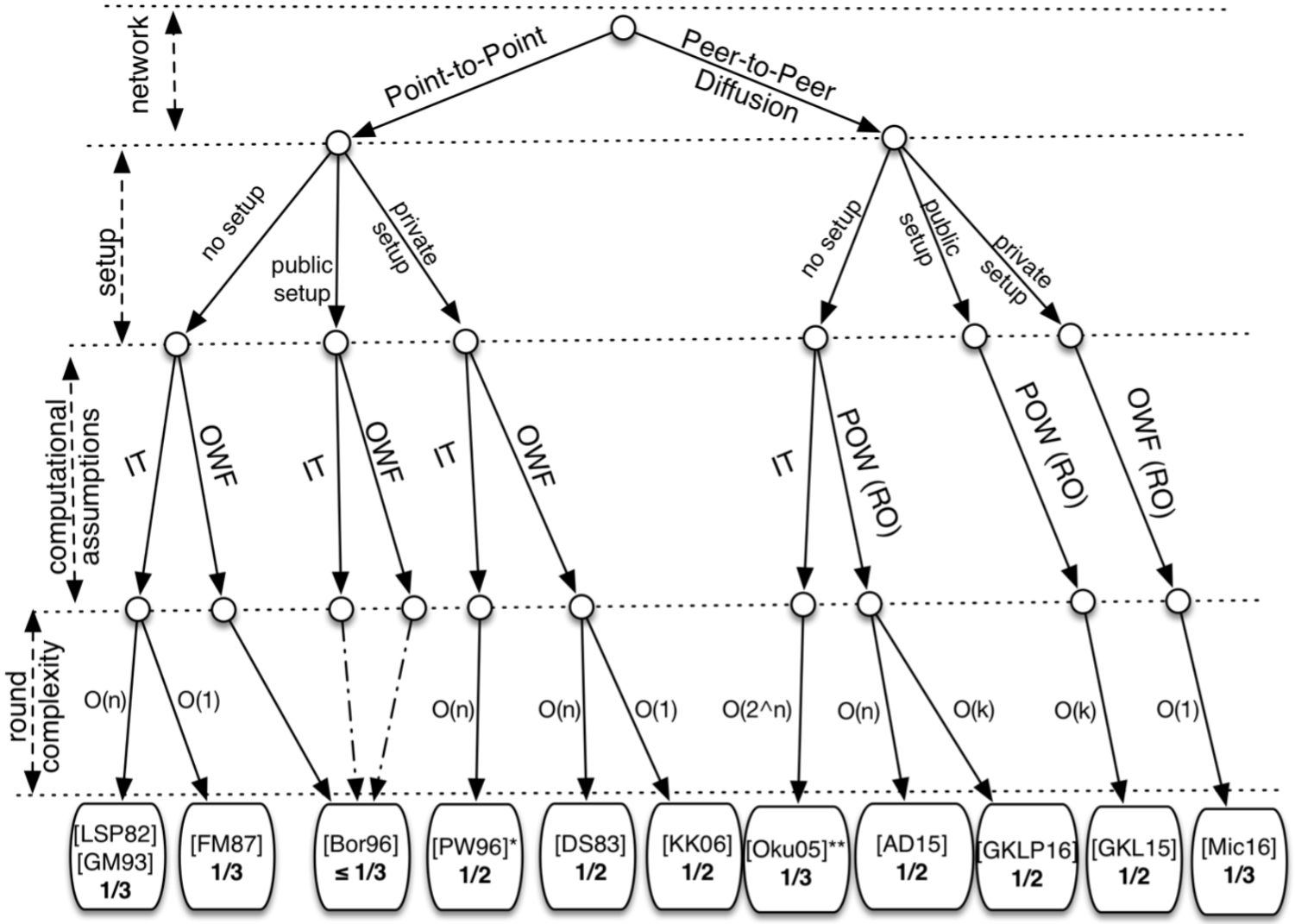
- **Issue:** If adv. finds a solution first, then honest parties will extend adv.'s solution and switch to adv.'s input → protocol doesn't guarantee Validity with overwhelming prob.

→ "Nakamoto consensus" doesn't solve consensus

Blockchain-based Consensus Protocol [GKL'15]

- The *n* parties start building a blockchain inserting their inputs
- If a party receives a longer blockchain switches to that one but *keeps the same input*
- Once the blockchain is long enough (2k) the parties prune the last k blocks and output the *majority value* in the prefix
- Tolerates $t < n/3$ corruptions
  - More elaborate protocol achieves $t < n/2$
- Assumes (public) trusted setup: Genesis block
  - [GKLP'18]: No setup; genesis block not needed

A “Consensus Taxonomy” [GK’18]



Lecture's Summary

I. Introduction

II. Brief Recap
- B'cast/consensus definitions, models, protocols

III. Prob. Termination and Composability of B'cast/Consensus Protocols
- The *Probabilistic Termination* framework
- Applications: UC-secure (parallel) b'cast (resp. SFE) in exp. constant (resp., O(d)) rounds

IV. B'cast/Consensus on Sparse Networks
- AE-b'cast/agreement, AE-MPC (AE: "Almost-Everywhere")

V. "IT-authenticated" B'cast/Consensus
- Information-theoretic *pseudosignatures*

VI. Blockchain-based Consensus
- A "consensus taxonomy"

Broadcast and Consensus in Cryptographic Protocols

Juan A. Garay
Texas A&M University
garay@tamu.edu
URL://engineering.tamu.edu/cse/people/garay-juan

IISc-IACR School on Cryptology