RSAConference2020
San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

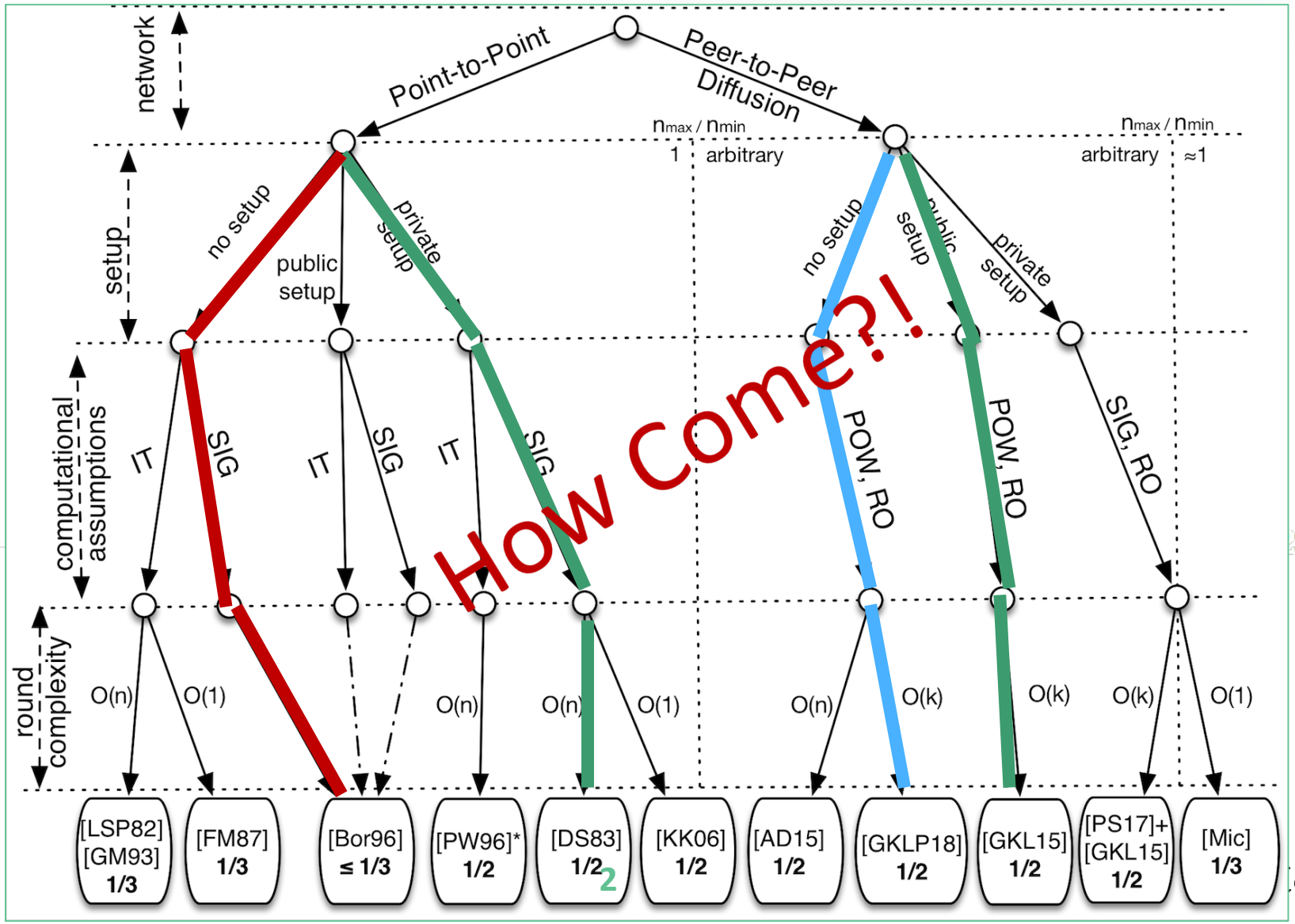
SESSION ID: CRYP-W11

SoK: A Consensus Taxonomy in the Blockchain Era

Juan Garay and Aggelos Kiayias

Texas A&M University and University of Edinburgh & IOHK

A Consensus Taxonomy



Consensus/Broadcast

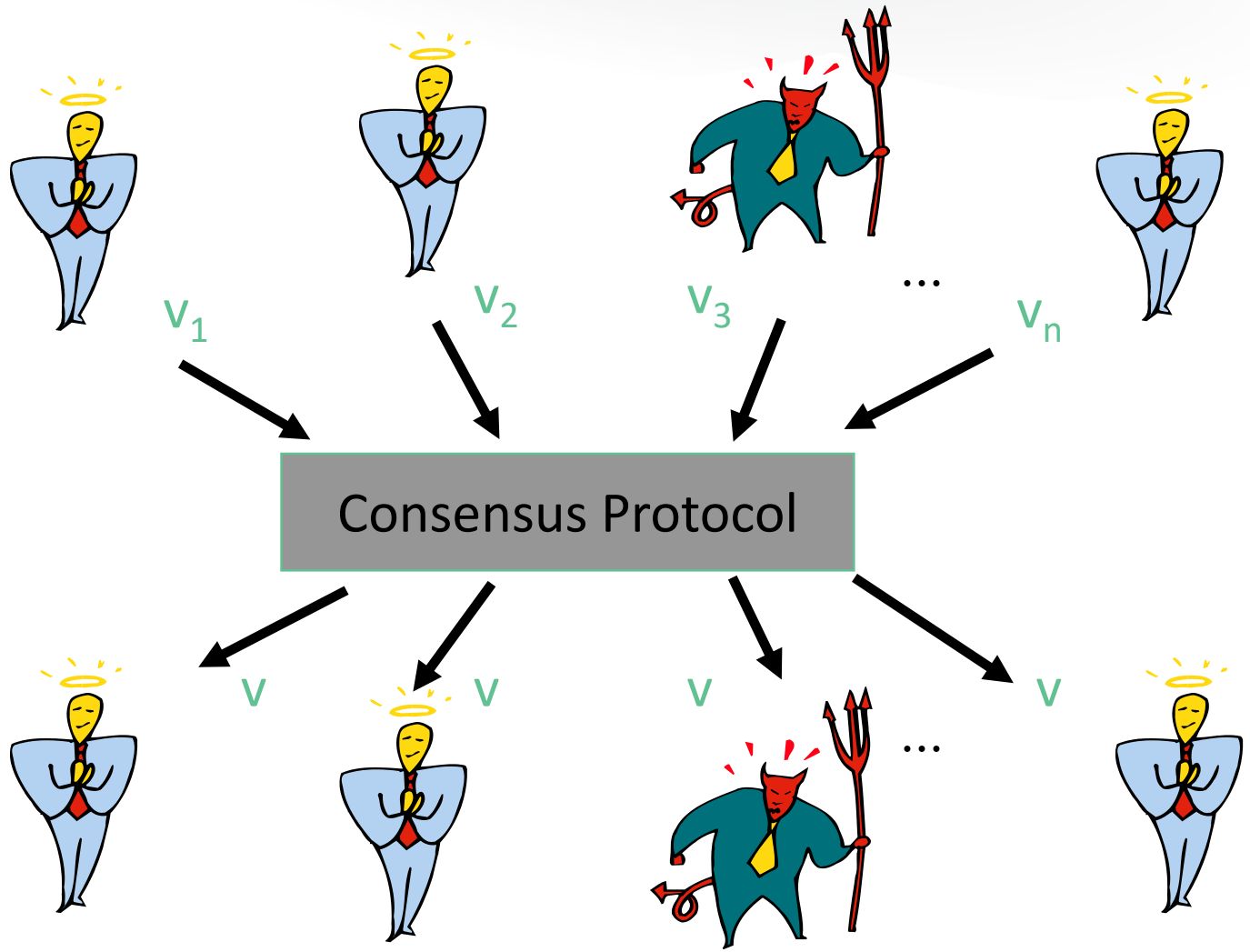
- One of the most fundamental problems in distributed computing
- Important role in cryptographic protocols
- Renewed interest with the advent of blockchain protocols like Bitcoin
  - New protocol paradigms
  - Wider research community
  - Applications expanded to novel settings

Consensus (Byz. Agreement) [PSL80, LSP82]

n parties
t *corrupted*

Adversary

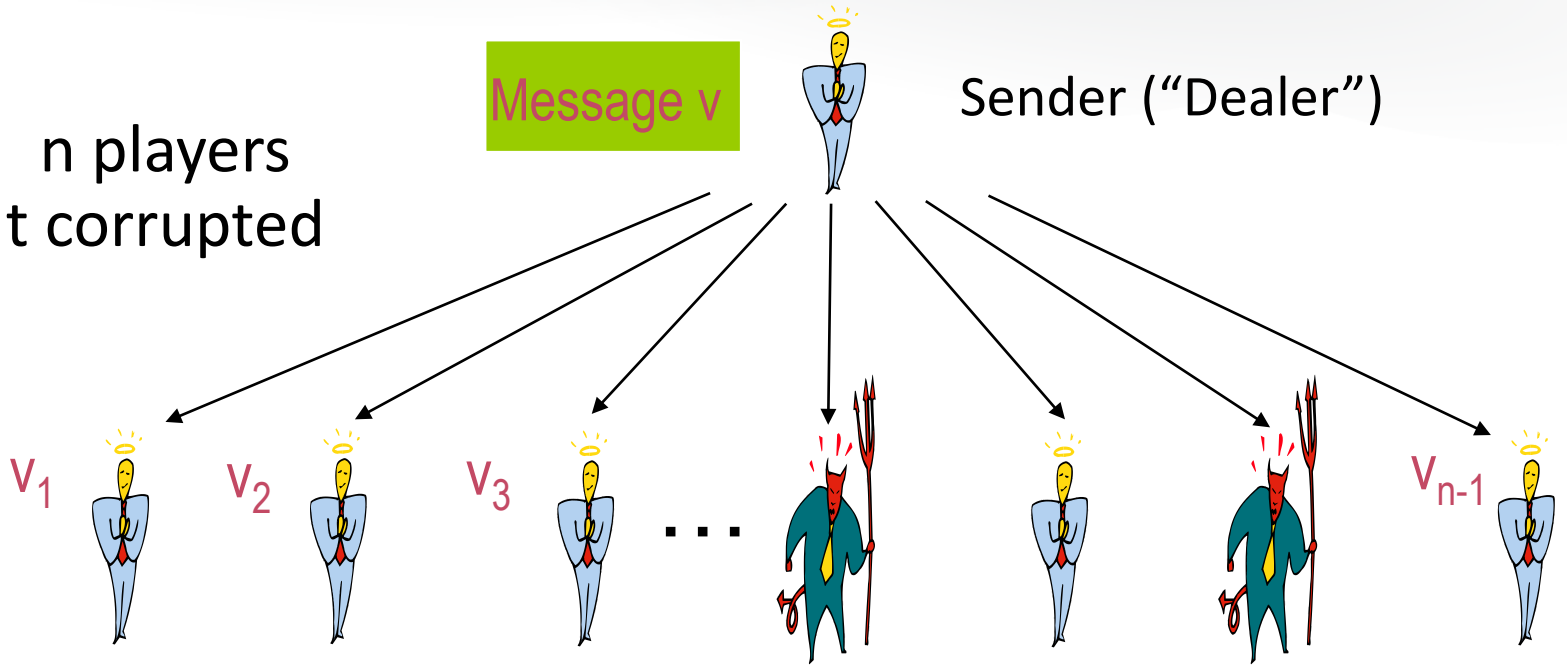
$v_i \in V = \{0,1\}$



Consensus (Byz. Agreement) [PSL80, LSP82] (2)

- **Consensus:** 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: Parties eventually terminate
- Single-sender/source version ("Byzantine Generals," Broadcast)
  - Validity: If sender is honest, then all honest parties output his value

Broadcast (aka *Byz. Generals*) [PSL80, LSP82]



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

Consensus (Byz. Agreement) [PSL80, LSP82] (2)

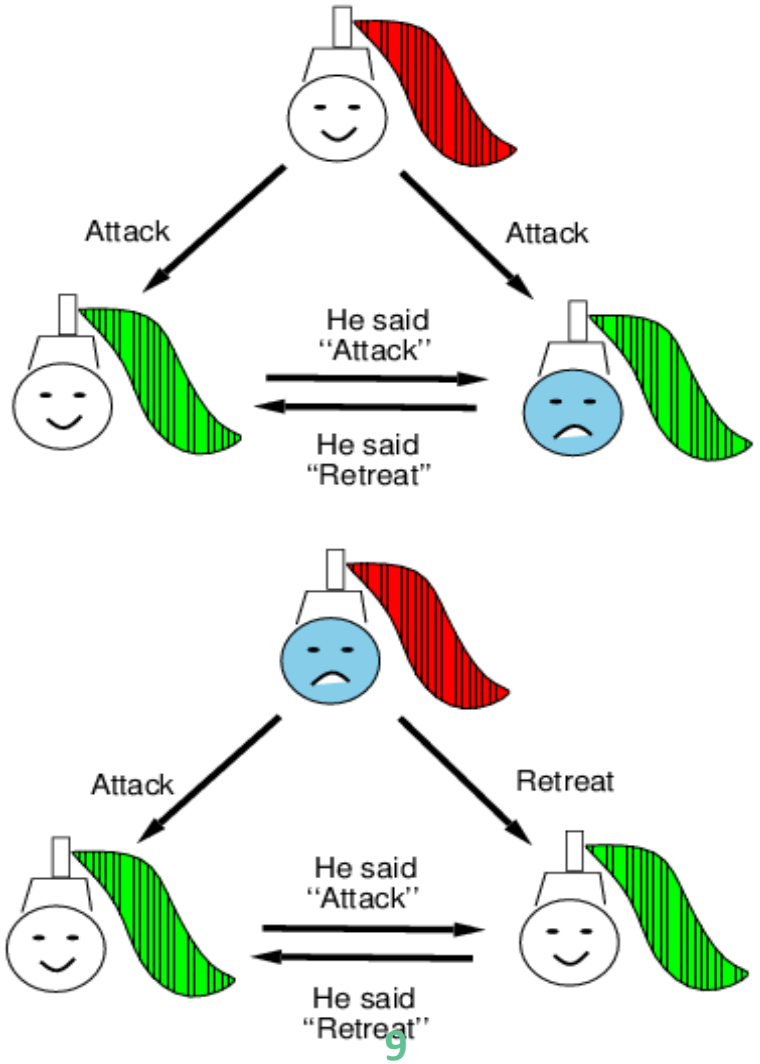
- **Consensus:** n parties start with an initial value v_i
  - Agreement: All honest parties output the same value
  - Validity: If all honest parties start with the same input (say, v), then they output this value
  - Termination: Parties eventually terminate
- Single-sender/source version ("Byzantine Generals," Broadcast)
  - Validity: If sender is honest, then all honest parties output his value

- Fundamental problems in fault-tolerant distributed computing and cryptographic protocols (secure multi-party computation – MPC)

Complexity Measures

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

Impossibility of Consensus with n ≤ 3t [PSL80, LSP82]



Complexity Measures

- Rounds: $r = t+1$ [LSP82, FL82]
- Resiliency:
  - Unconditional setting: $n > 3t$ [LSP82, GM92]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Consensus: $n > 2t$ [DS82, Fit03]

Complexity Measures

- Rounds: $r = t+1$ [LSP82, FL82]
- Resiliency:
  - Unconditional setting: $n > 3t$ [LSP82, GM92]
  - Cryptographic setting:
    - Broadcast: $n > t$ [LSP82, DS82]
    - Consensus: $n > 2t$ [DS82, Fit03]
- Message/Bit complexity: $m = \Omega(n^2)$ [DR85, CW92, BGP92,…]

Cryptographic ("Authenticated") Consensus Protocols

- **Setup:** Public-key infrastructure (PKI)
- **Assumption:** Digital signatures secure against adaptive chosen-message attacks [GMR88]
- [DS82]: r = t+1, poly(n)
  - Broadcast: n > t
  - Consensus: n > 2t

Cryptographic ("Authenticated") Consensus Protocols (2)

[DS82] *broadcast* protocol (informal) ($n > t$):

- Source signs its input value and sends to all parties
- $r = 1,\ldots,t+1$:
  - 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

Talk Plan

- Introduction (Consensus/Broadcast)
- Consensus Lower Bounds
- Blockchain-based Consensus
  - Blockchain basics
  - A blockchain abstraction: The Bitcoin backbone protocol
  - Is a genesis block really needed?: Consensus from scratch
- A Consensus Taxonomy
- A Ledger Consensus Taxonomy
- References

On the Necessity of an Honest Majority for Consensus

$$t < n/2$$

regardless of the resources available to the parties

On the Necessity of an Honest Majority for Consensus (2)

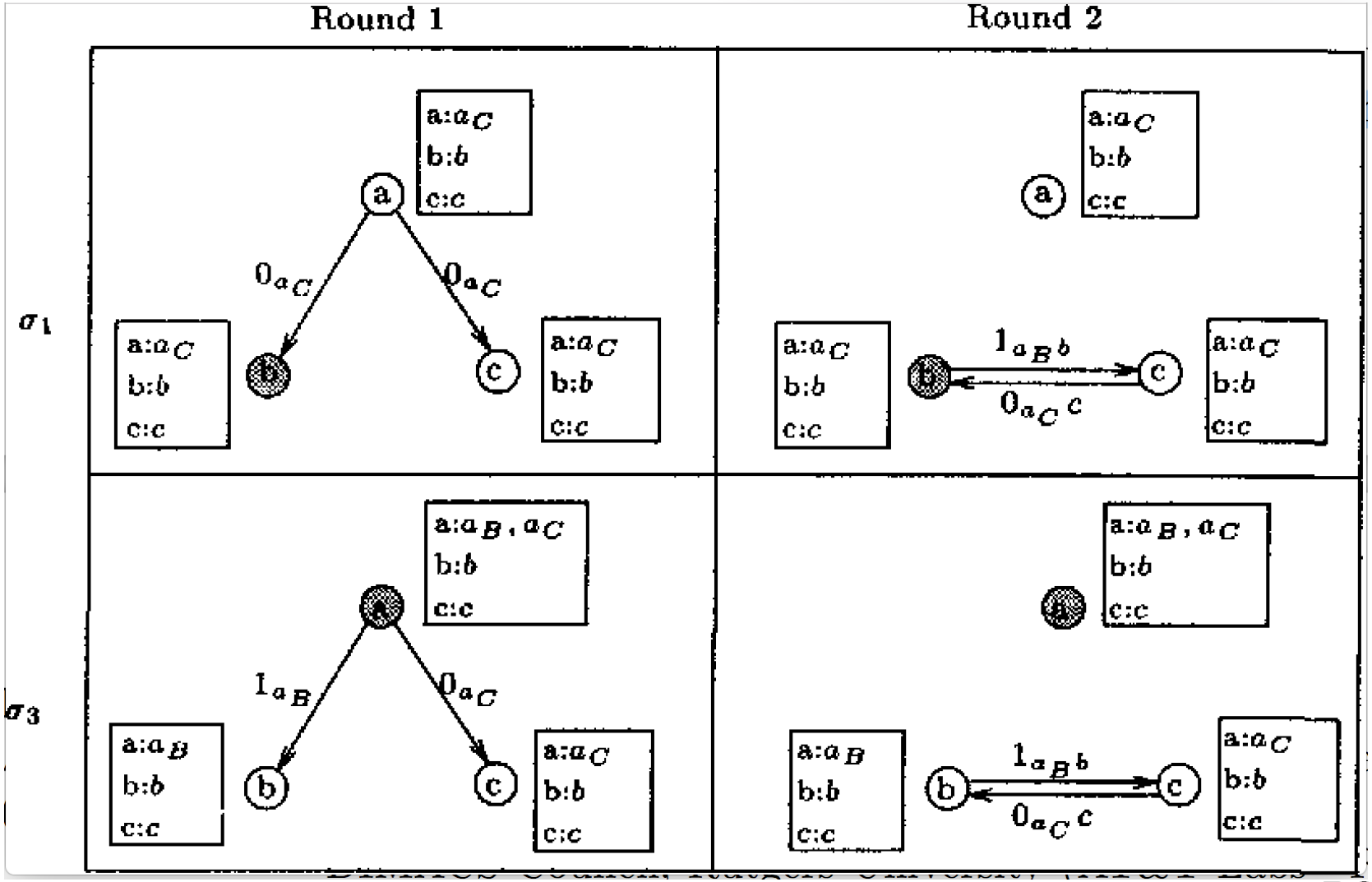
- **Scenario** [Fit03]: n parties equally divided with respect to their initial values $\in \{0,1\}$. Adv. corrupts $\emptyset$, P_0 and P_1 uniformly at random:
  1. With 1/3 prob. adversary corrupts no one
  2. With 1/3 prob. adversary corrupts parties with input 0
  3. With 1/3 prob. adversary corrupts parties with input 1

  In any case, the corrupted parties follow the protocol
- Case 1 requires honest parties to converge to common output (Agreement)
- Case 2 & 3: Honest parties should output 0 (resp., 1) (Validity)
- But three cases are indistinguishable in the view of the honest parties

On the Necessity of a PKI ("Private-state Setup") [Bor96]

- ~~**Setup:** Public-key infrastructure (PKI)~~
- **Assumption:** Digital signatures secure against adaptive chosen-message attacks [GMR88]
- Broadcast/Consensus not possible when $n \leq 3t$

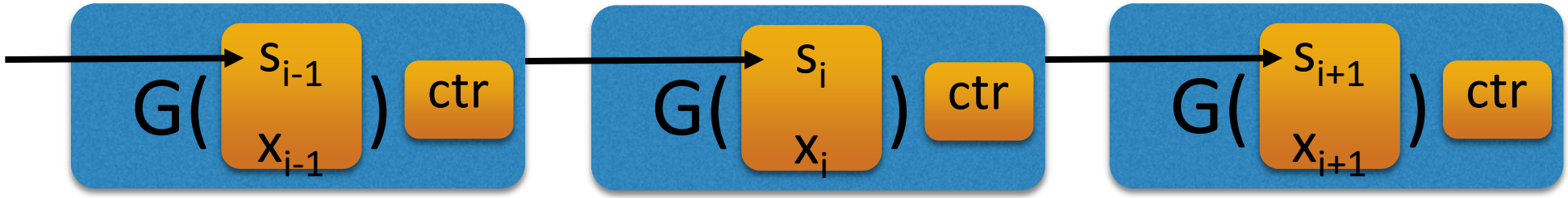
On the Necessity of a PKI ("Private-state Setup") [Bor96]



Talk Plan

- Introduction (Consensus/Broadcast)
- Consensus Lower Bounds
- Blockchain-based Consensus
  - Blockchain basics
  - A blockchain abstraction: The Bitcoin backbone protocol
  - Is a genesis block really needed?: Consensus from scratch
- A Consensus Taxonomy
- A Ledger Consensus Taxonomy
- References

Blockchain-based Consensus



What Is a Blockchain?

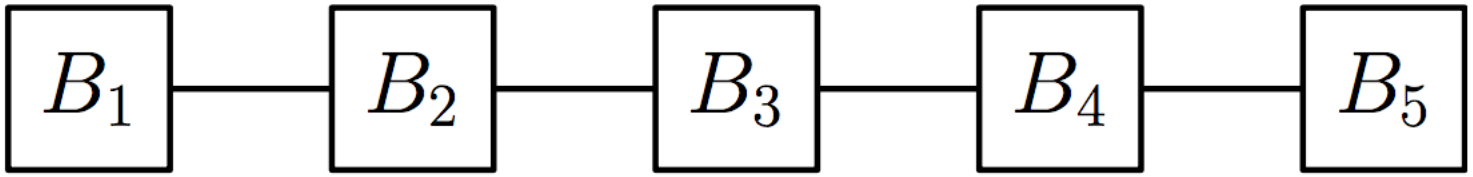
- Parties (“miners”) have to do work in order to install a transaction

What Is a Blockchain?

- Parties (“miners”) have to do work in order to install a transaction
- Transactions are organized in chains of blocks

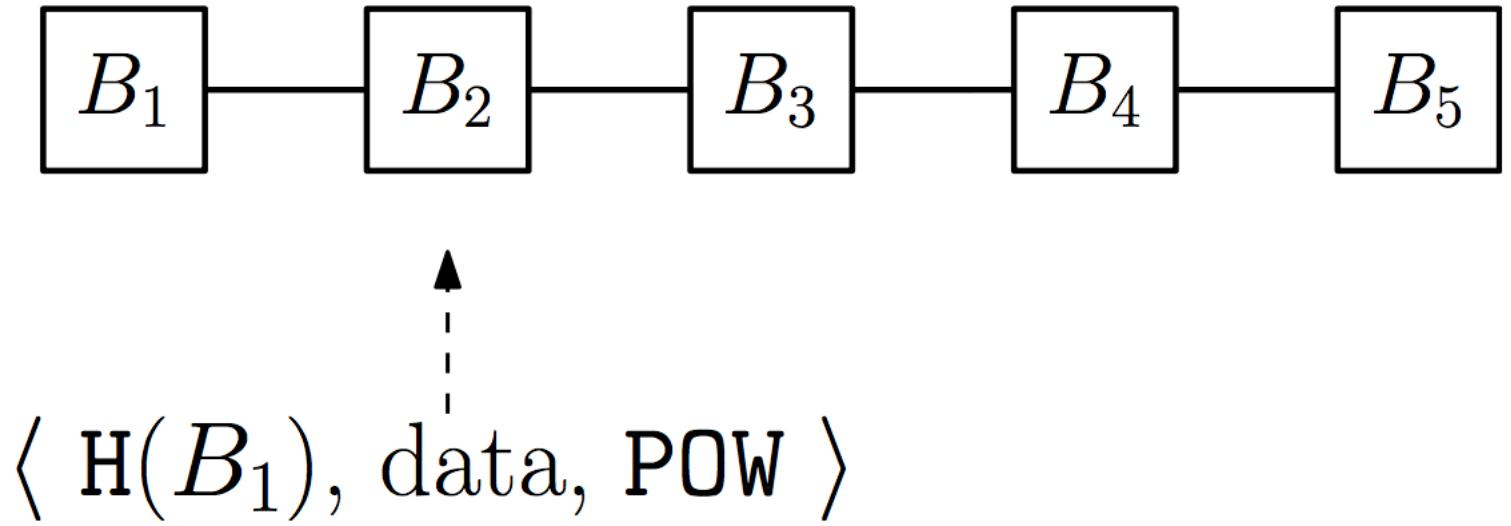
What Is a Blockchain?

- Parties ("miners") have to do work in order to install a transaction
- Transactions are organized in chains of blocks



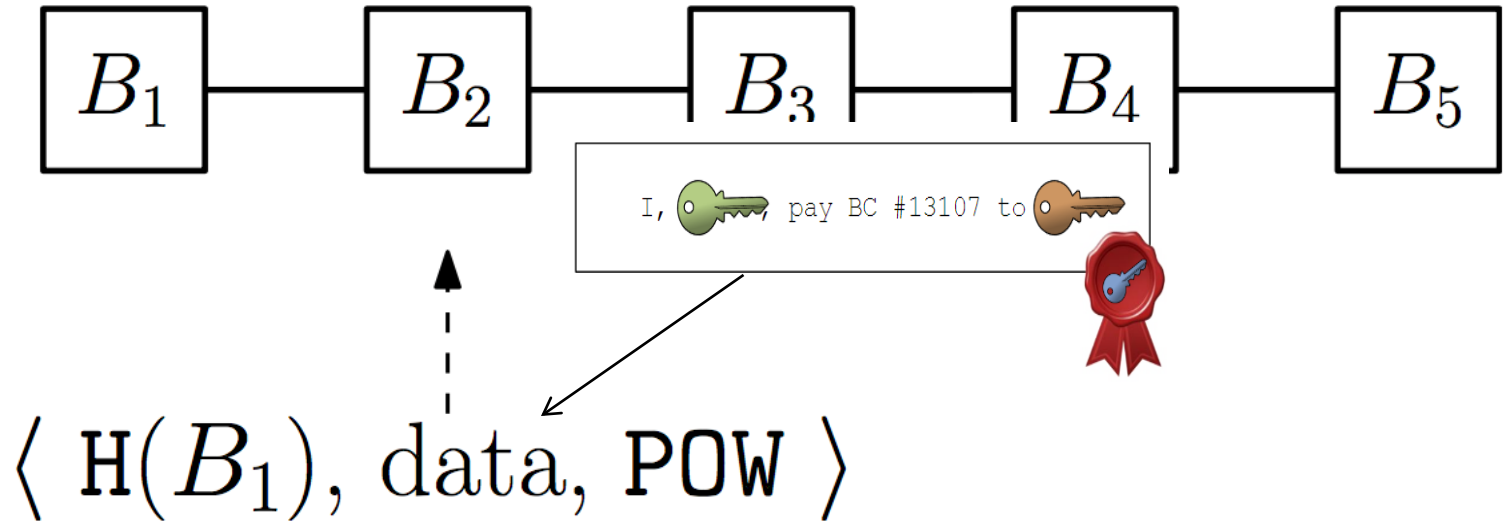
What Is a Blockchain?

- Parties ("miners") have to do work in order to install a transaction
- Transactions are organized in chains of blocks



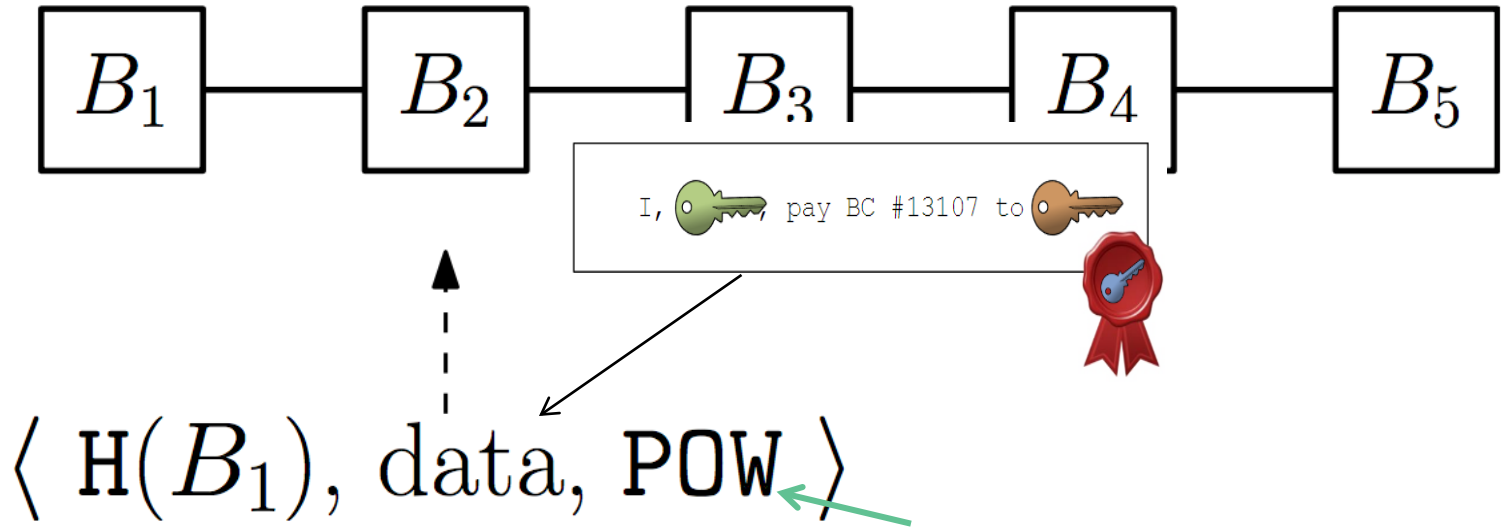
What Is a Blockchain?

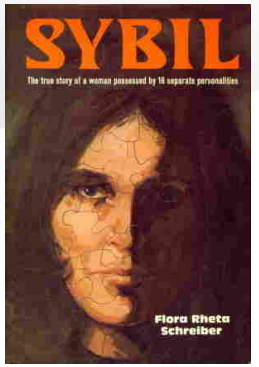
- Parties ("miners") have to do work in order to install a transaction
- Transactions are organized in chains of blocks



What Is a Blockchain?

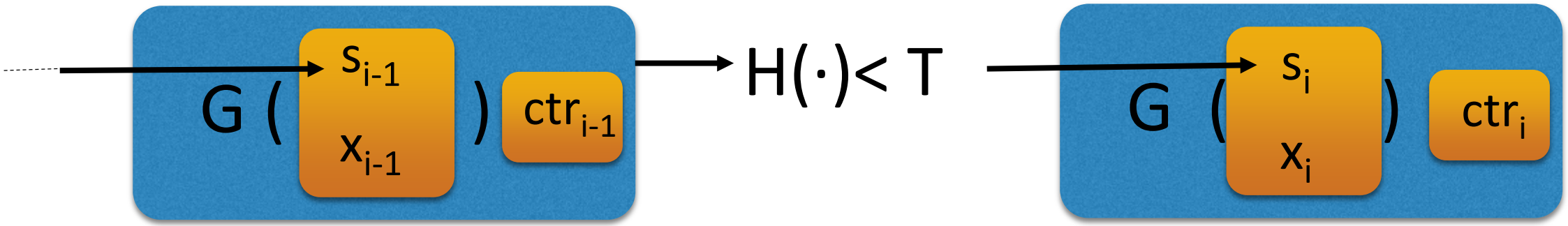
- Parties ("miners") have to do work in order to install a transaction
- Transactions are organized in chains of blocks





Proofs of Work [DN92,…]

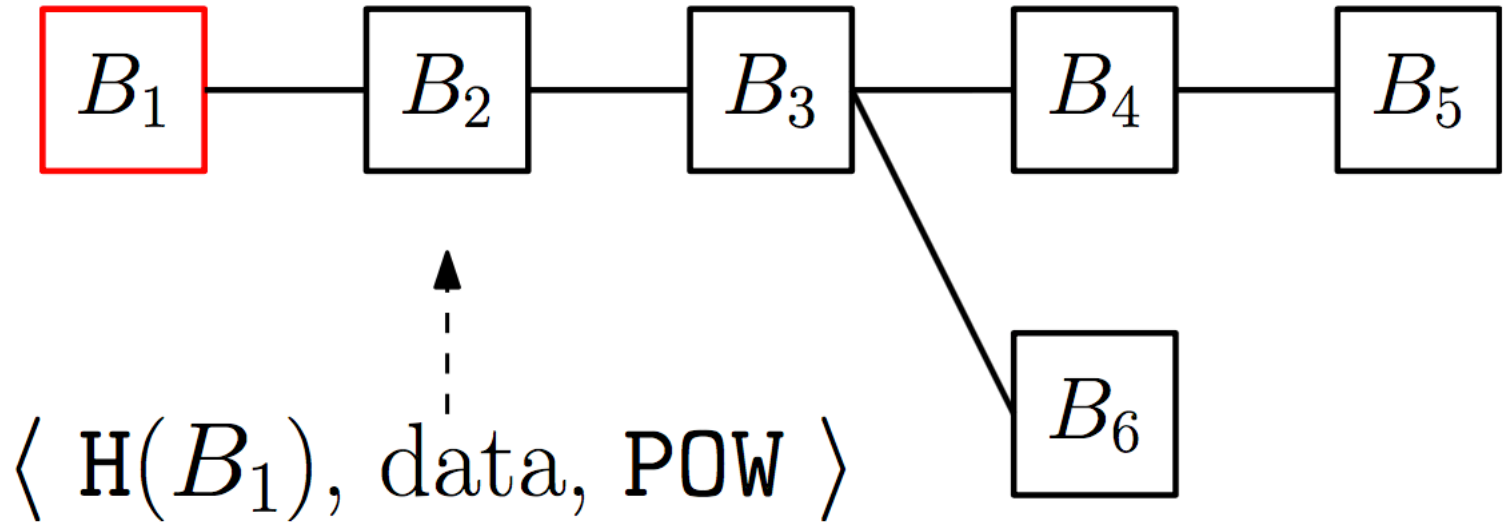
- Spam mitigation, Sybil attacks, denial of service protection, …
- Most impactful application: Design of blockchain protocols such as Bitcoin



$$\text{Hash}(ctr_{i-1}; \text{Hash}(s_{i-1}, x_{i-1})) < T$$

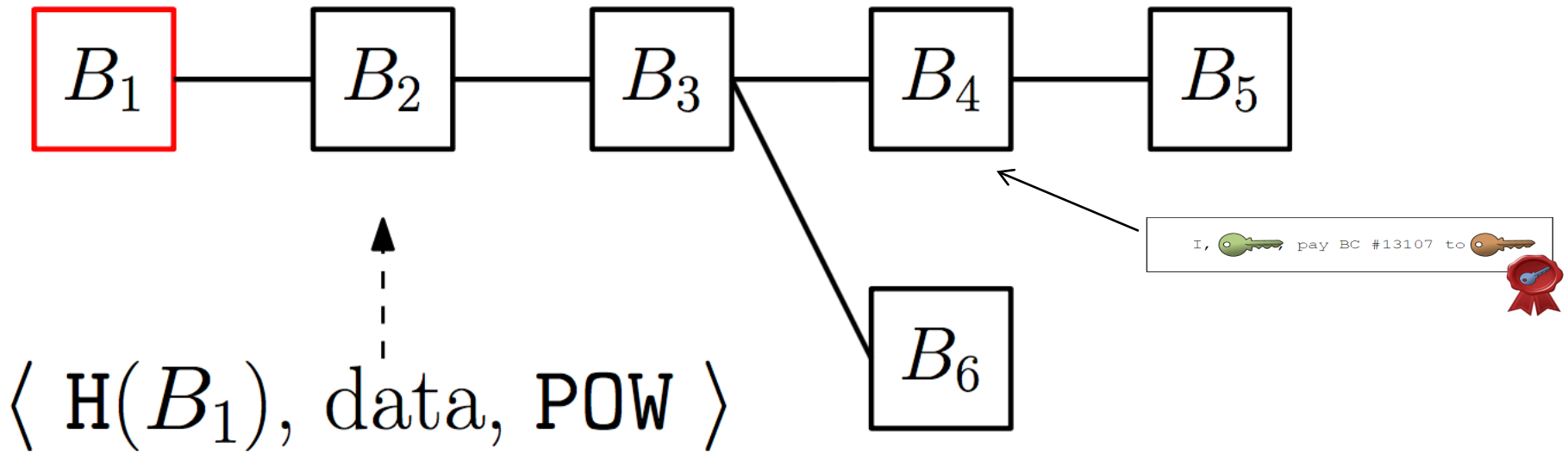
What Is a Blockchain?

- Parties ("miners") always choose the *longest* chain they received



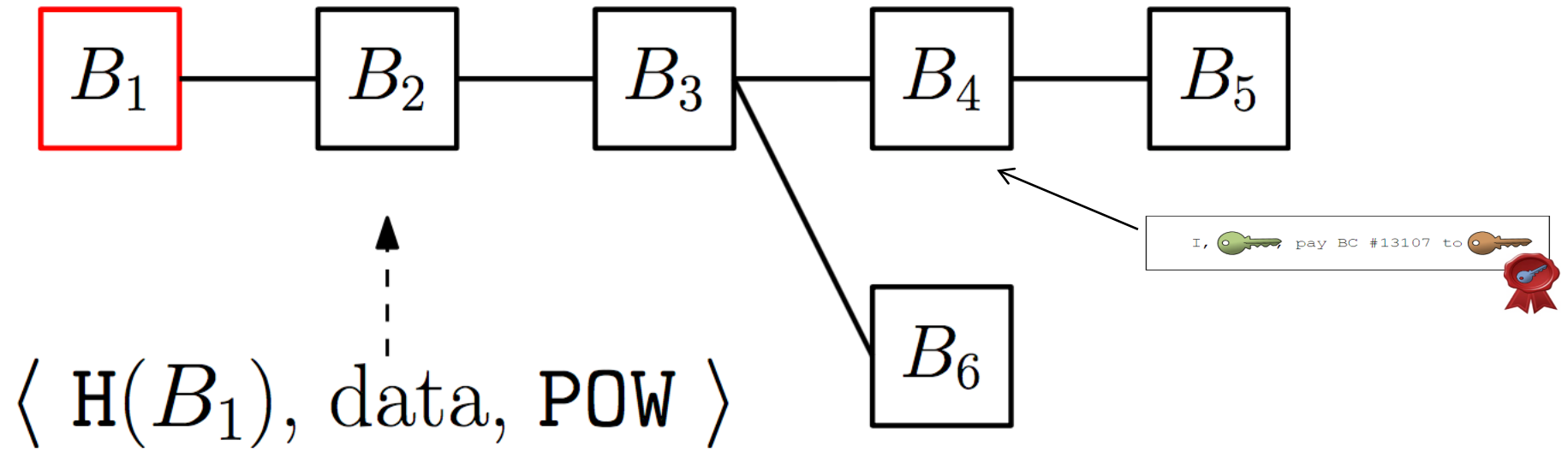
What Is a Blockchain?

- Parties ("miners") always choose the *longest* chain they received
- If party wants to erase his transaction, he has to find a longer chain!



What Is a Blockchain?

- Parties ("miners") always choose the *longest* chain they received
- If party wants to erase his transaction, he has to find a longer chain!



If transaction is "sufficiently deep," he cannot do this unless he has "majority hashing power"

A Blockchain Abstraction: The Bitcoin Backbone Protocol [GKL15]

- Analysis of Bitcoin in a *general adversarial* model
  - Arbitrary attacks: E.g., send inconsistent messages, "selfish mining," etc.
- We extract, formally describe, and analyze the **core** of the Bitcoin protocol – the *Bitcoin backbone*
- Protocol parameterized by three application-specific external functions
  - V(·): *content (of chain) validation predicate*
  - I(·): *input contribution function*
  - R(·): *chain reading function*

→ Distinguish data structure from application layer

Network/Computational Model

- Protocol executed by *fixed* no. of parties n (not necessarily known to participants); active/"rushing"/adaptive adversary controls a subset
  - Security against *all* possible attacks
- Underlying communication graph not fully connected (P2P); messages delivered through "diffusion" mechanism ("broadcast")
- Parties *cannot* authenticate each other; adversary can "spoof" *source* of message, but *can't disconnect* honest parties
- Assume time is divided in *rounds*; within each round all messages are delivered
  - Important in terms of Bitcoin's inherent assumption regarding the players' ability to produce POWs
  - Wlog — analysis extends to *partially synchronous* networks ("bounded-delay" model) [DLS88, PSS17]

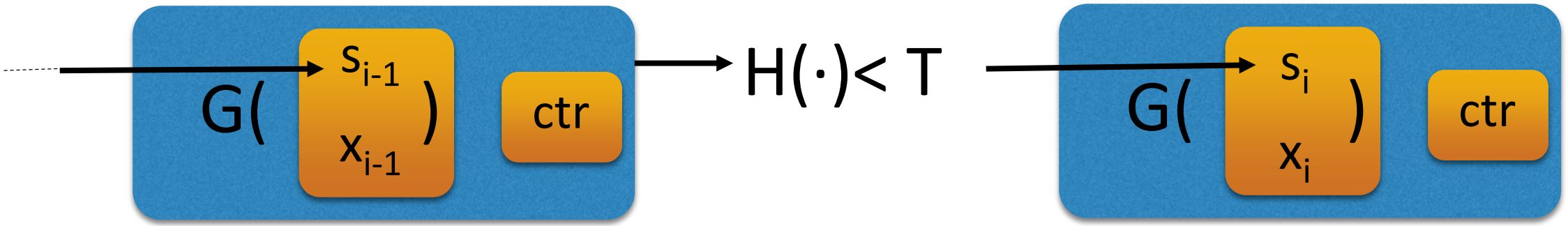
Network/Computational Model (2)

- In each round, each party is allowed q queries to a cryptographic hash function (*random oracle*)
  - "Flat" version of the world in terms of hashing power
  - t parties controlled by adversary, acting as a malicious mining pool
    → $t \cdot q$ queries/round
  - $t < n/2$ corresponds to adv. controlling strictly less of the system's total "hashing power"
  - Worse for honest parties (uncoordinated, decentralized)

- **Trusted set-up:** Unpredictable "genesis" block
  - More later…

The Bitcoin Backbone (1)

- Parameterized by V(), I(), R(), and hash functions G(), H()
  - Players have a *state* C of the form:



  - The *contents* of C satisfy the predicate $V(x_1,\ldots,x_i)$ = true

Basic Properties of the Blockchain [GKL15]

Common Prefix

(informally)

If two players prune a sufficient number of blocks from their chains they will obtain the same prefix

Chain Quality

(informally)

Any (large enough) chunk of an honest player's chain will contain some blocks from honest players

Chain Growth

(informally)

the chain of any honest player grows at least at a steady rate - the chain speed coefficient

From Blockchain Properties to Applications

- Determine the parameters for which above properties (Common Prefix, Chain Quality, Chain Growth) hold with overwhelming probability (in the security parameter)
- Then show how an application's properties can be proven (in a black-box manner) using these properties

Applications of the Bitcoin Backbone Protocol

- Consensus
- Robust transaction ledger (Bitcoin)
  - Aka “ledger consensus,” “Nakamoto consensus”

Nakamoto's Consensus Protocol [Nak08b]

- "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 [Nak08b] (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

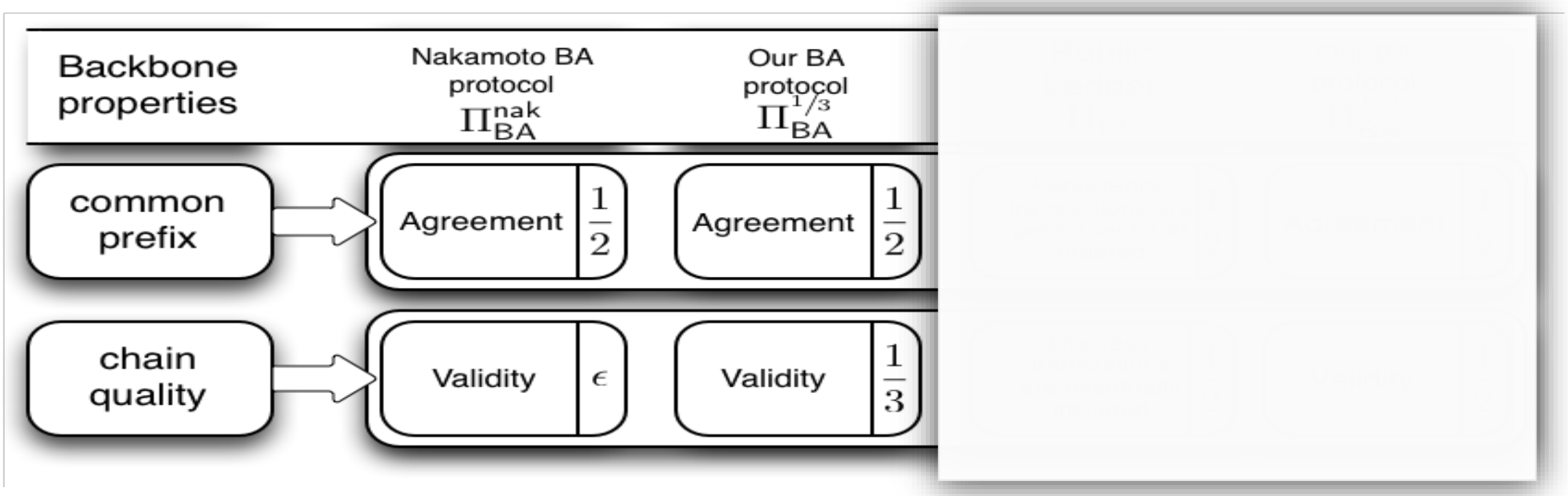
Nakamoto's Consensus Protocol [Nak08b] (3)

- 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

Summary of Applications



Our First Consensus Protocol [GKL15]

- 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

- We get:
  - Agreement from the Common Prefix property
  - Validity as long as adv. controls < ⅓ of the parties (tight, due to the Chain Quality property)

Observations

- Based on the basic Bitcoin backbone properties – CP, CQ, CG – we obtained a *probabilistic solution* for the consensus problem tolerating a 1/3 fraction of corrupted parties
- 1/3 is *suboptimal*
  - **Main obstacle:** The blockchain (backbone) protocol does not provide sufficient *chain quality*
  - We cannot guarantee we have enough blocks originating from honest parties $(1-\beta)\gamma$
- 1/2 can be achieved, using a more elaborate protocol – a technique we call *2-for-1 PoWs*

1/2 Consensus Protocol [GKL15]

- **Idea:** Use POWs to “mine” transactions, in the same way blocks are mined
- Should guarantee that *number of transactions* is proportional to the hashing power of each party
- Mining:
  - At block level
  - At transaction level

1/2 Consensus Protocol (2)

Beware!

Given that POWs would be used for two different tasks, how do we prevent the adversary from shifting his computer power from to the other?

1/2 Consensus Protocol (3)

High-level description:

- Operation: In each round, parties run *two protocols in parallel*:
  - Transaction ledger protocol
    - q queries to oracle H_0

1/2 Consensus Protocol (3)

High-level description:

- Operation: In each round, parties run *two* protocols in parallel:
  - Transaction ledger protocol (cf. Lecture 8)
    - q queries to oracle H_0
  - A *transaction production* protocol, which continuously generates transactions satisfying

$$(H_1(ctr, G(nonce, v)) < T) \wedge (ctr \leq q)$$

1/2 Consensus Protocol (3)

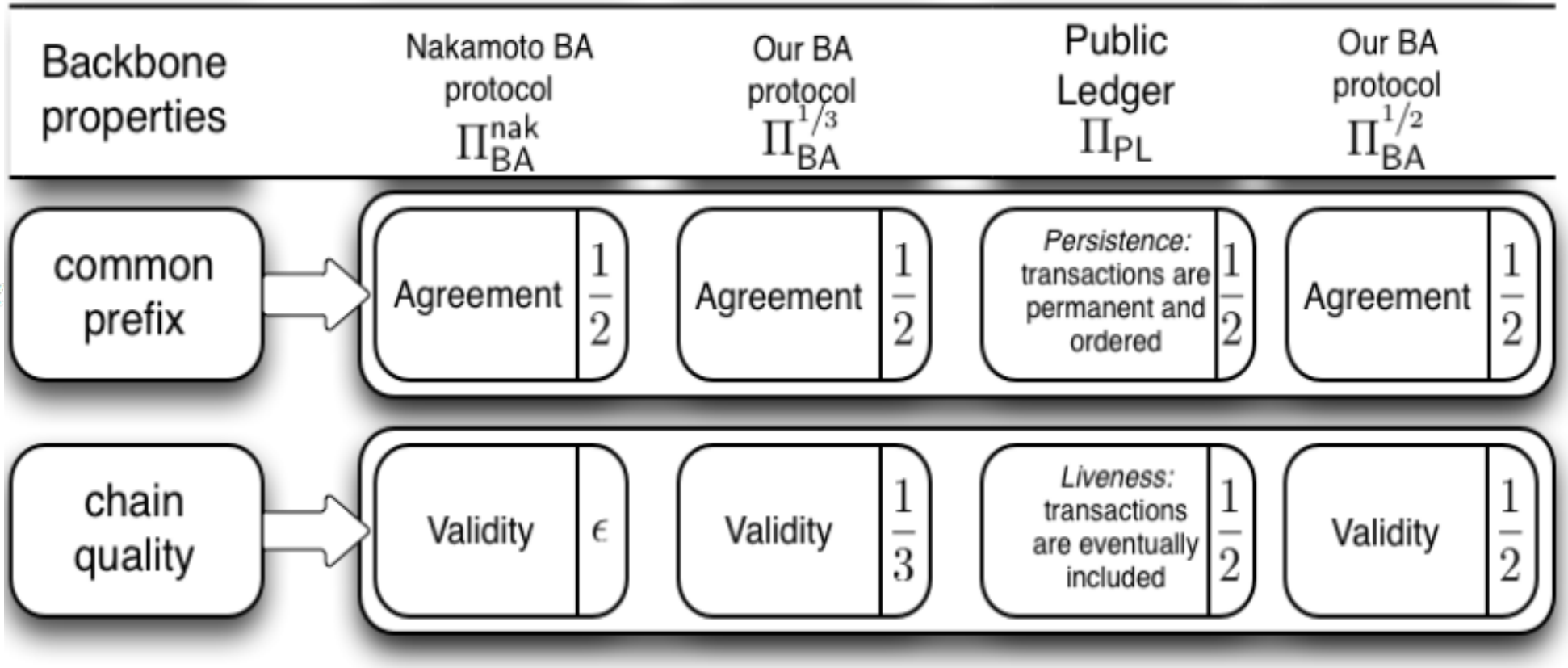
High-level description:

- Operation: In each round, parties run *two* protocols in parallel:
  - Transaction ledger protocol
    - q queries to oracle H_0
  - A *transaction production* protocol, which continuously generates transactions satisfying

$$(H_1(ctr, G(nonce, v)) < T) \wedge (ctr \leq q)$$

- Termination and output: After round L, a party collects all the unique transactions that are present in the first O(k) blocks and returns the majority value (bit) from the bits occurring in these transactions
  - **Note:** Uniqueness takes nonce into account

Summary of Applications (2)



Backbone properties	Nakamoto BA protocol Π_{BA}^{nak}	Our BA protocol $\Pi_{BA}^{1/3}$	Public Ledger Π_{PL}	Our BA protocol $\Pi_{BA}^{1/2}$
common prefix	Agreement $\frac{1}{2}$	Agreement $\frac{1}{2}$	*Persistence:* transactions are permanent and ordered $\frac{1}{2}$	Agreement $\frac{1}{2}$
chain quality	Validity ϵ	Validity $\frac{1}{3}$	*Liveness:* transactions are eventually included $\frac{1}{2}$	Validity $\frac{1}{2}$

Talk Plan

- Introduction (Consensus/Broadcast)
- Consensus Lower Bounds
- Blockchain-based Consensus
  - Blockchain basics
  - A blockchain abstraction: The Bitcoin backbone protocol
  - Is a genesis block really needed?: Consensus from scratch
- A Consensus Taxonomy
- A Ledger Consensus Taxonomy
- References

Is a Genesis Block Really Needed?

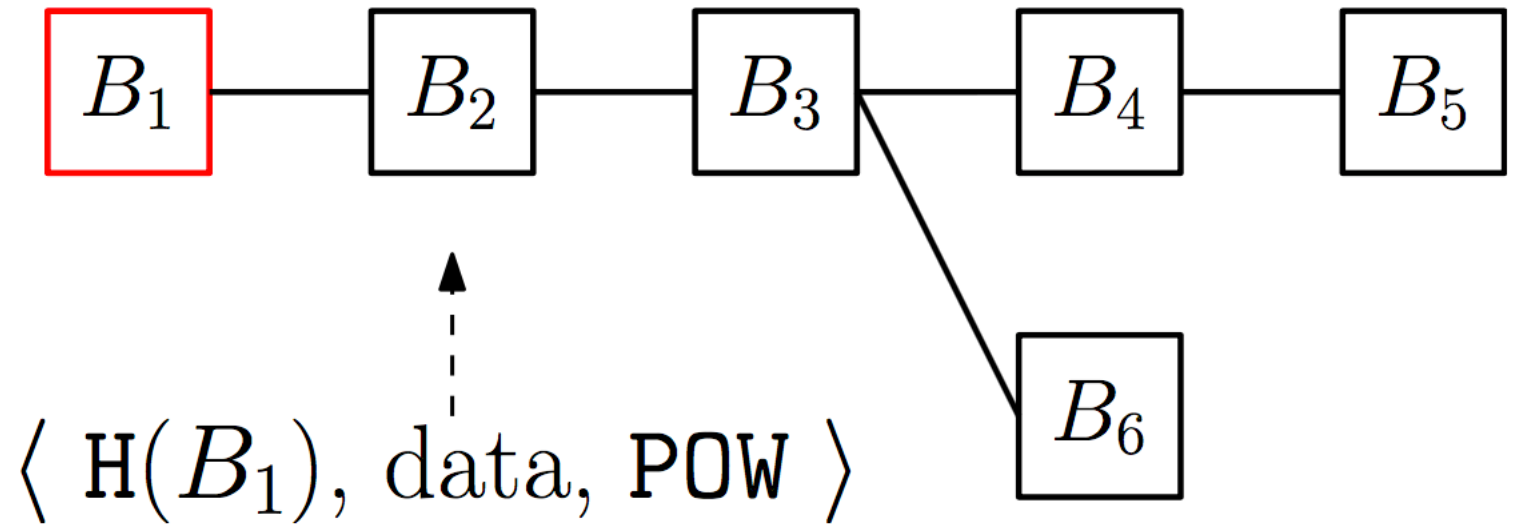
- "Genesis" block
  - First block of the blockchain; must be unpredictable
  - Hardcoded into the software
  - Coinbase parameter contains the following text:

"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"

RECALL

What Is a Blockchain?

- Parties ("miners") always choose the *longest* chain they received



RECALL

Network/Computational Model (2)

- In each round, each party is allowed q queries to a cryptographic hash function (*random oracle* [BR93])
  - "Flat" version of the world in terms of hashing power
  - t parties controlled by adversary, acting as a malicious mining pool
    → t·q queries/round
  - t < n/2 corresponds to adv. controlling strictly less of the system's total "hashing power"
  - Worse for honest parties (uncoordinated, decentralized)

- ~~**Trusted set-up:** Unpredictable "genesis" block~~
  - More later…

Not needed!

The *Bootstrapped* Backbone Protocol [GKLP18]

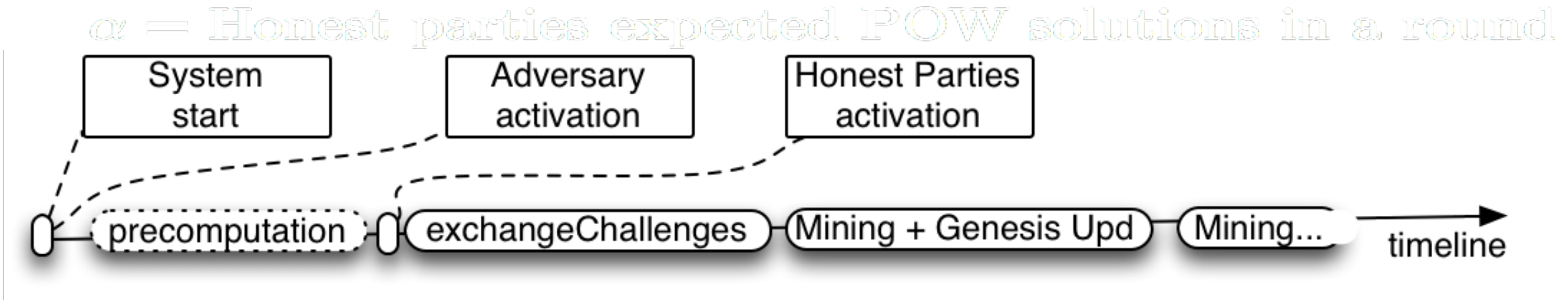
- *No trusted setup* and individual genesis block mining
- Freshness of genesis block impacts chains' total *weight*
- Personalized chain selection rule
- Robustness is achieved after an initial period of protocol stabilization

$$f = \alpha + \beta$$

$$\gamma \approx \alpha - \alpha^2$$

The *Bootstrapped* Backbone Protocol [GKLP18] (2)

- Two phases:
  1. Challenge exchange phase (O(κ) rounds)
  2. Basic backbone functions [GKL15]

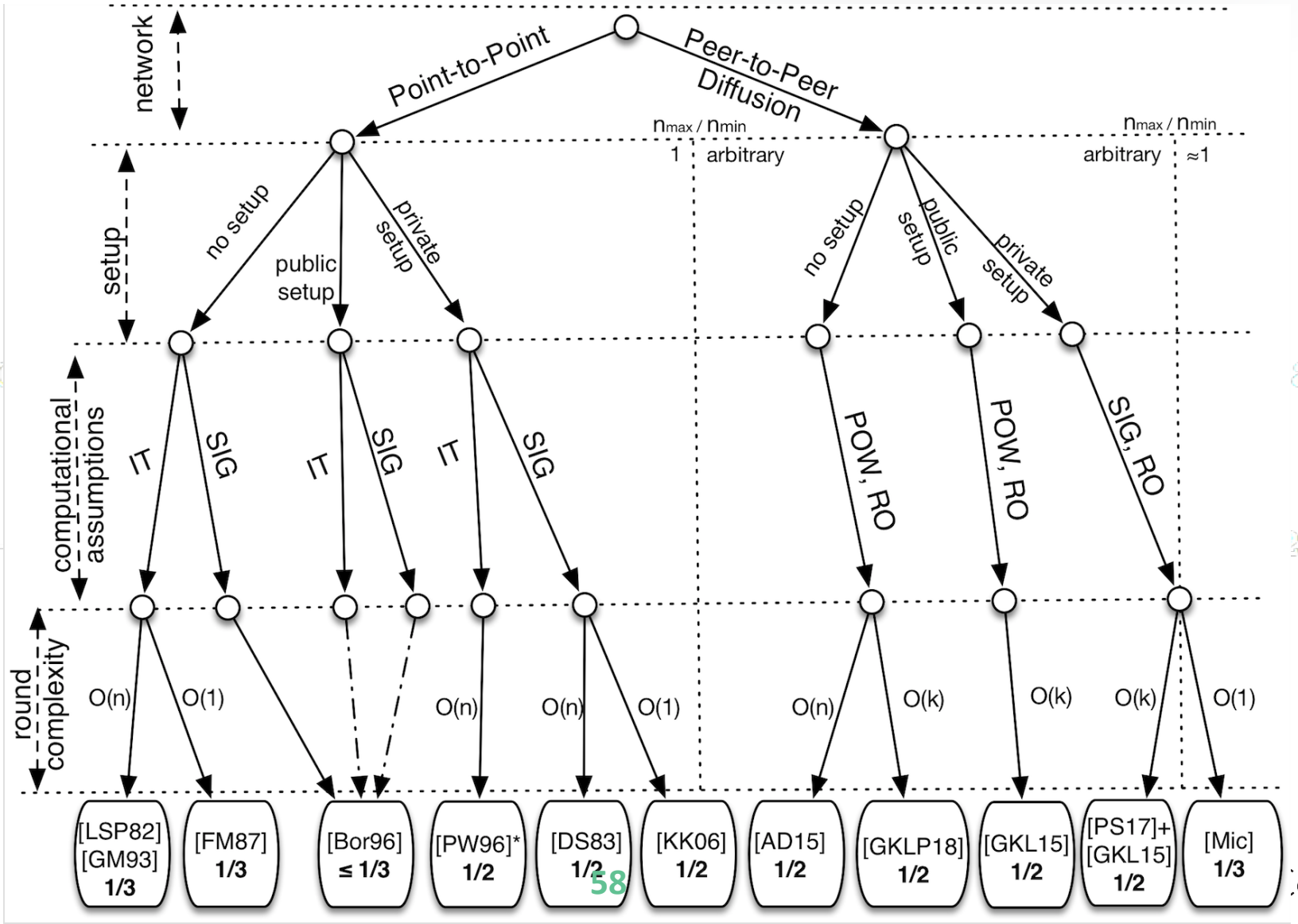


- ⇒ Can achieve consensus with t < n/2 and *no trusted setup*

Talk Plan

- Introduction (Consensus/Broadcast)
- Consensus Lower Bounds
- Blockchain-based Consensus
  - Blockchain basics
  - A blockchain abstraction: The Bitcoin backbone protocol
  - Is a genesis block really needed?: Consensus from scratch
- A Consensus Taxonomy
- A Ledger Consensus Taxonomy
- References

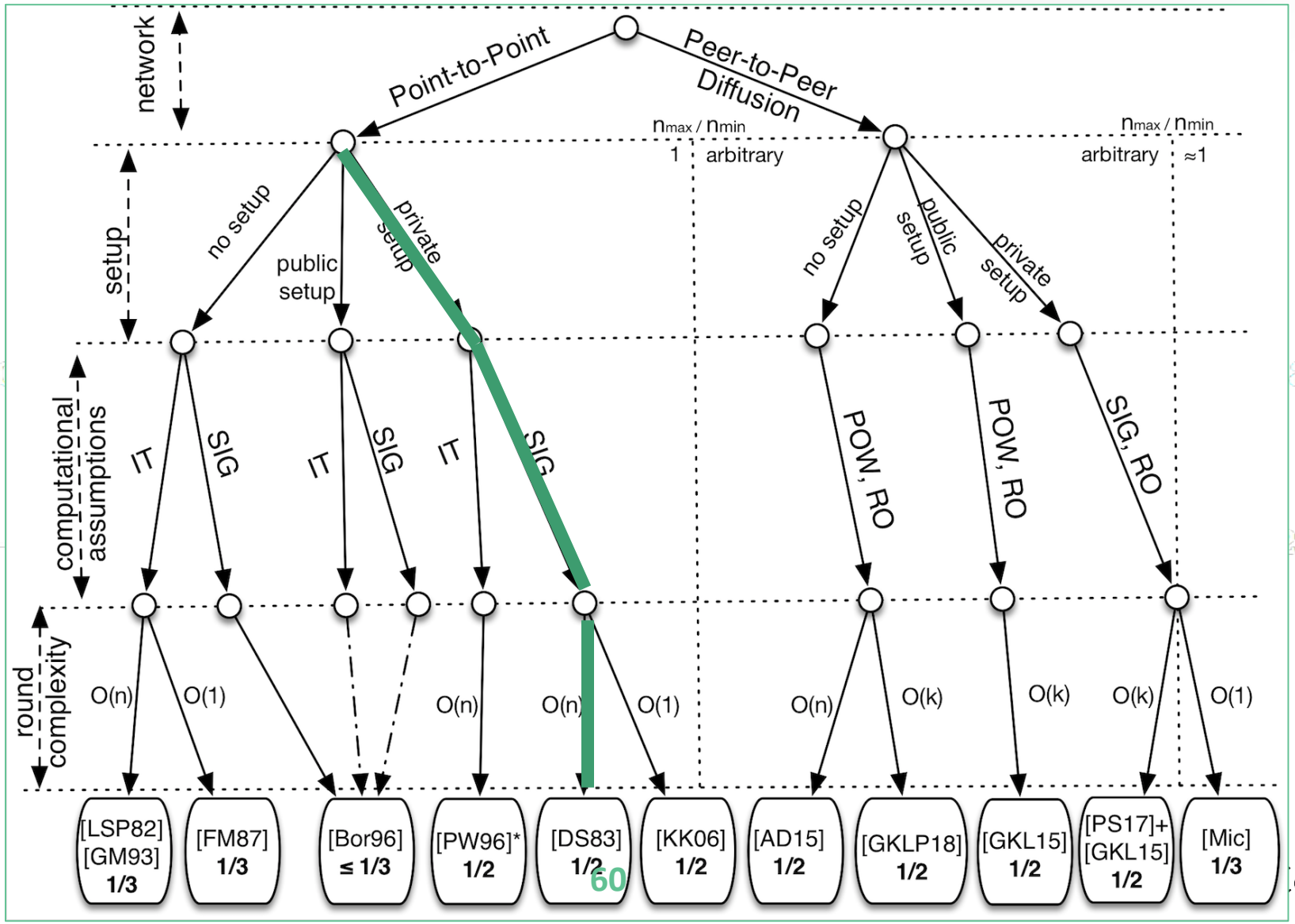
A Consensus Taxonomy



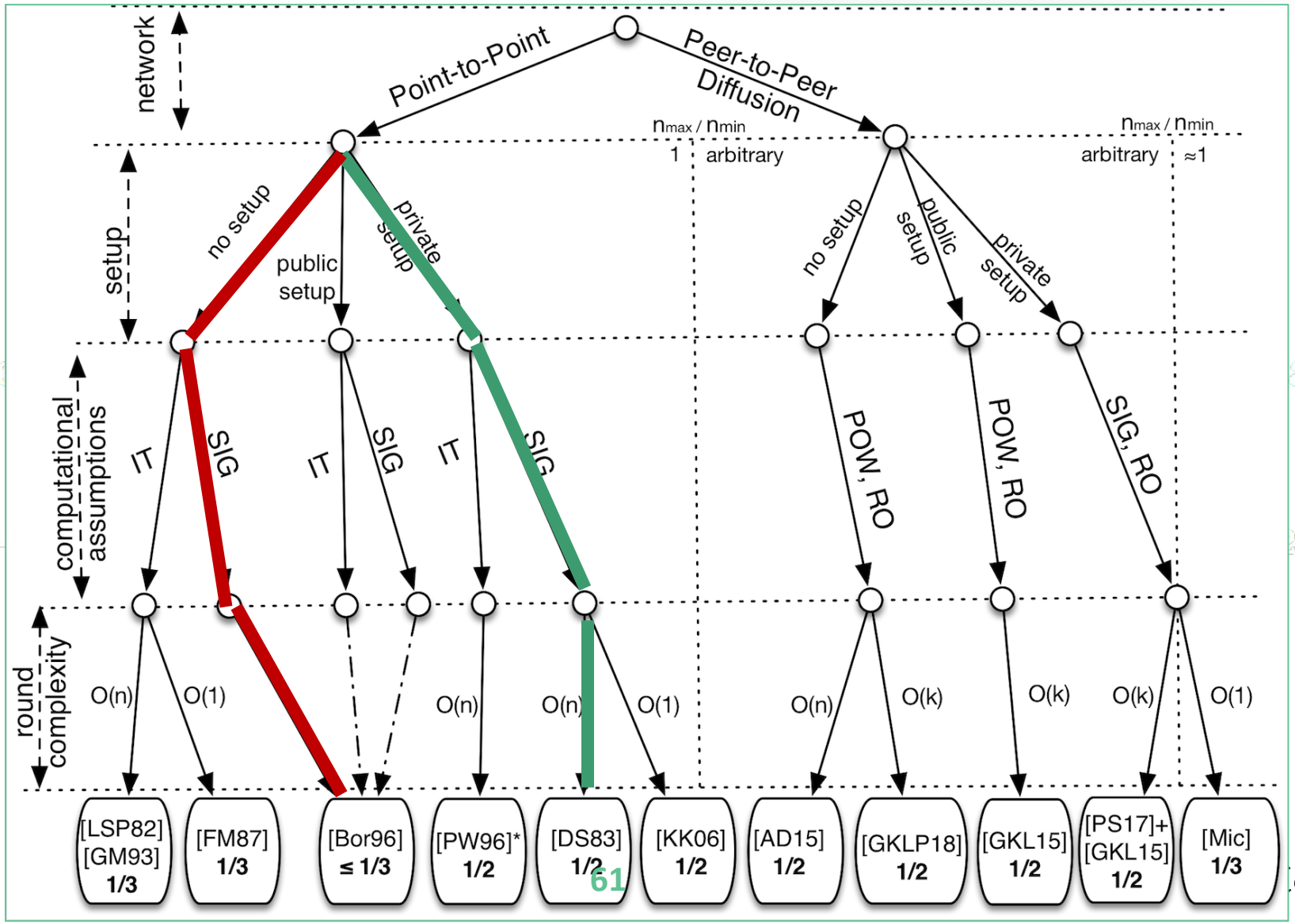
Assumptions/Resources

- **Network:**
  - Communication primitives: Point-to-point (RMT/SMT), peer-to-peer (“Diffuse”)
    - Diffuse ≤ RMT
  - Synchrony
- **Setup:**
  - No setup
  - Private-state setup (e.g., PKI)
  - Public-state setup (e.g., CRS)
- **Computational:**
  - Information-theoretic security
  - Computational security: OWF, PoW, RO
- **Participation tolerance:** n_{max} / n_{min}

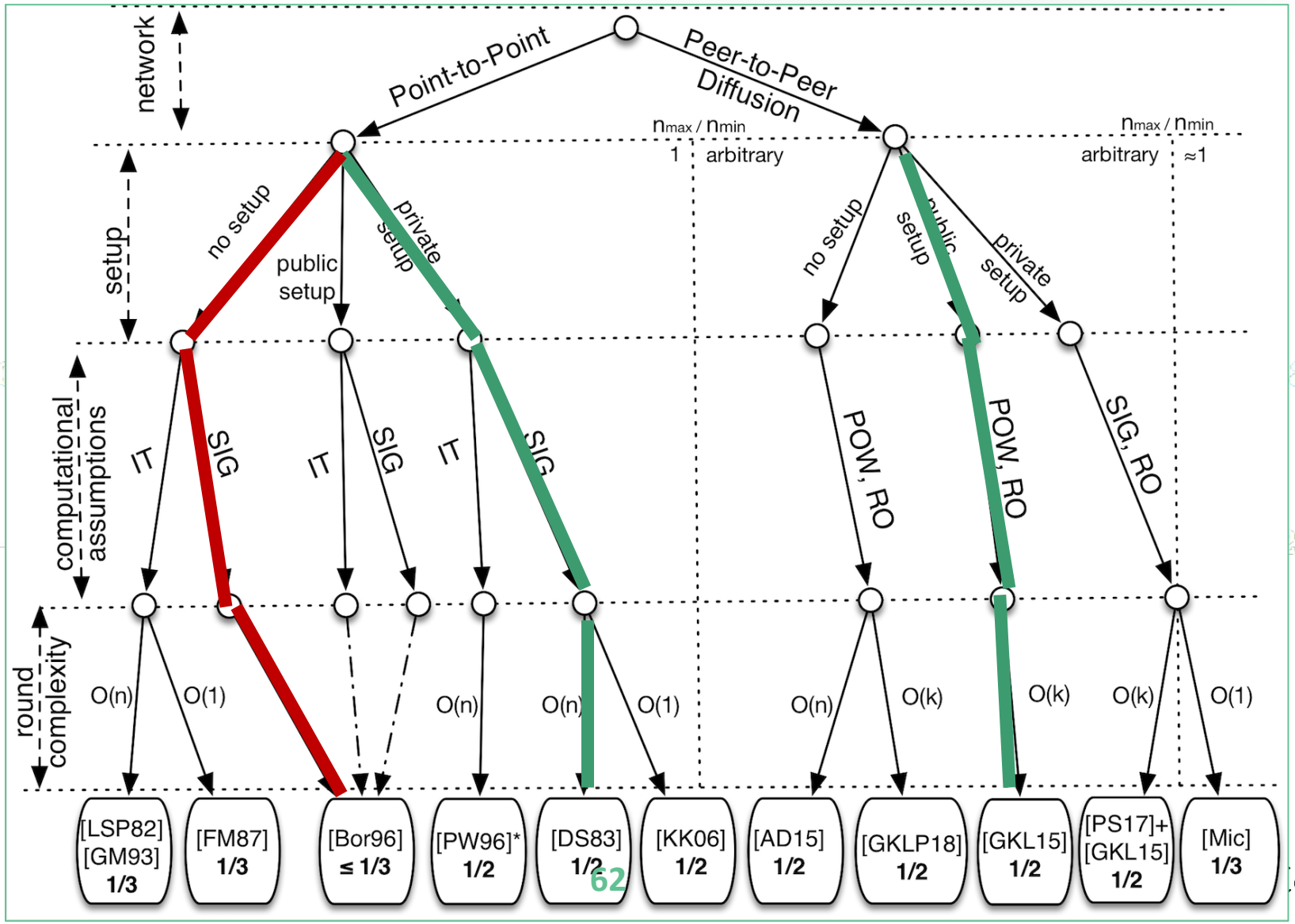
A Consensus Taxonomy



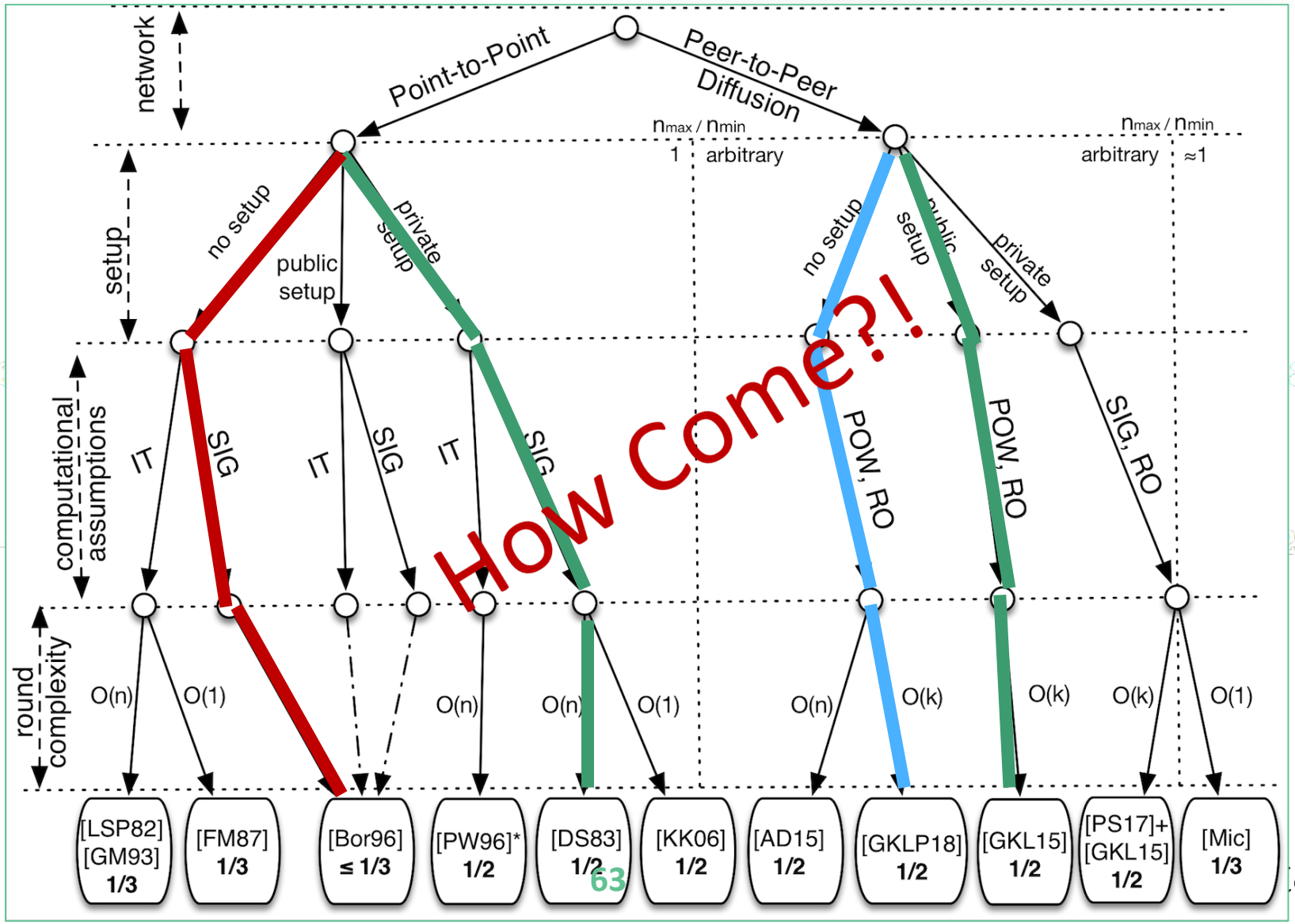
A Consensus Taxonomy



A Consensus Taxonomy

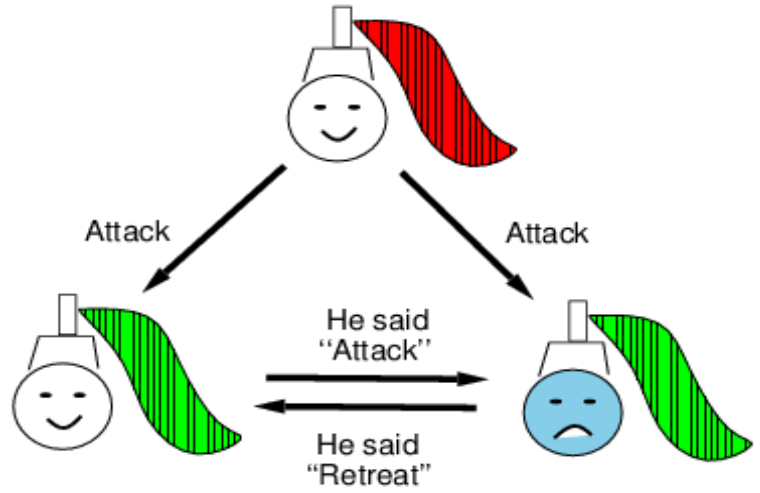


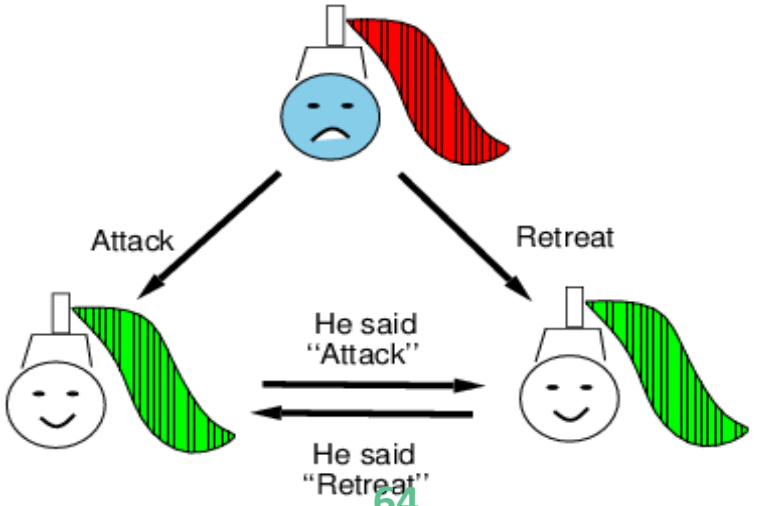
A Consensus Taxonomy



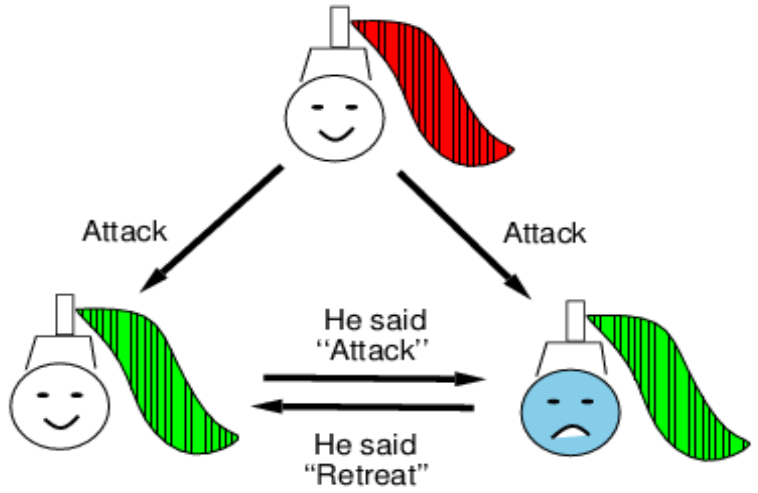
RECALL

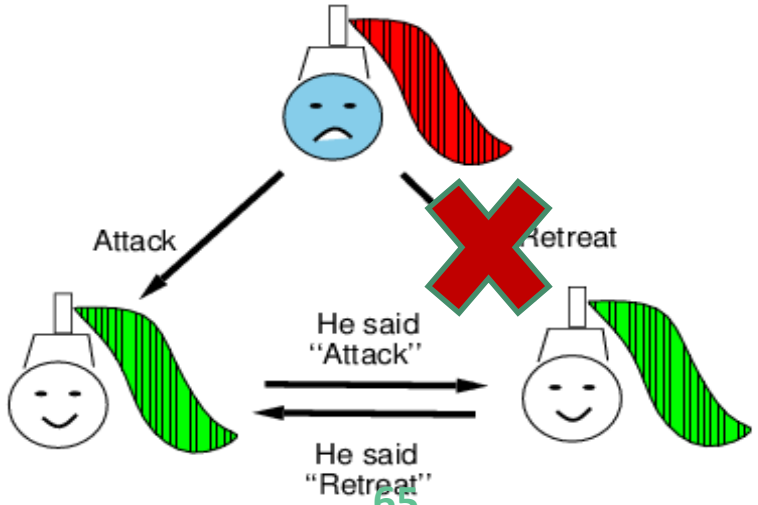
Impossibility with n = 3t [PSL80, LSP82]





Resource-restricted Cryptography [GKOPZ20]





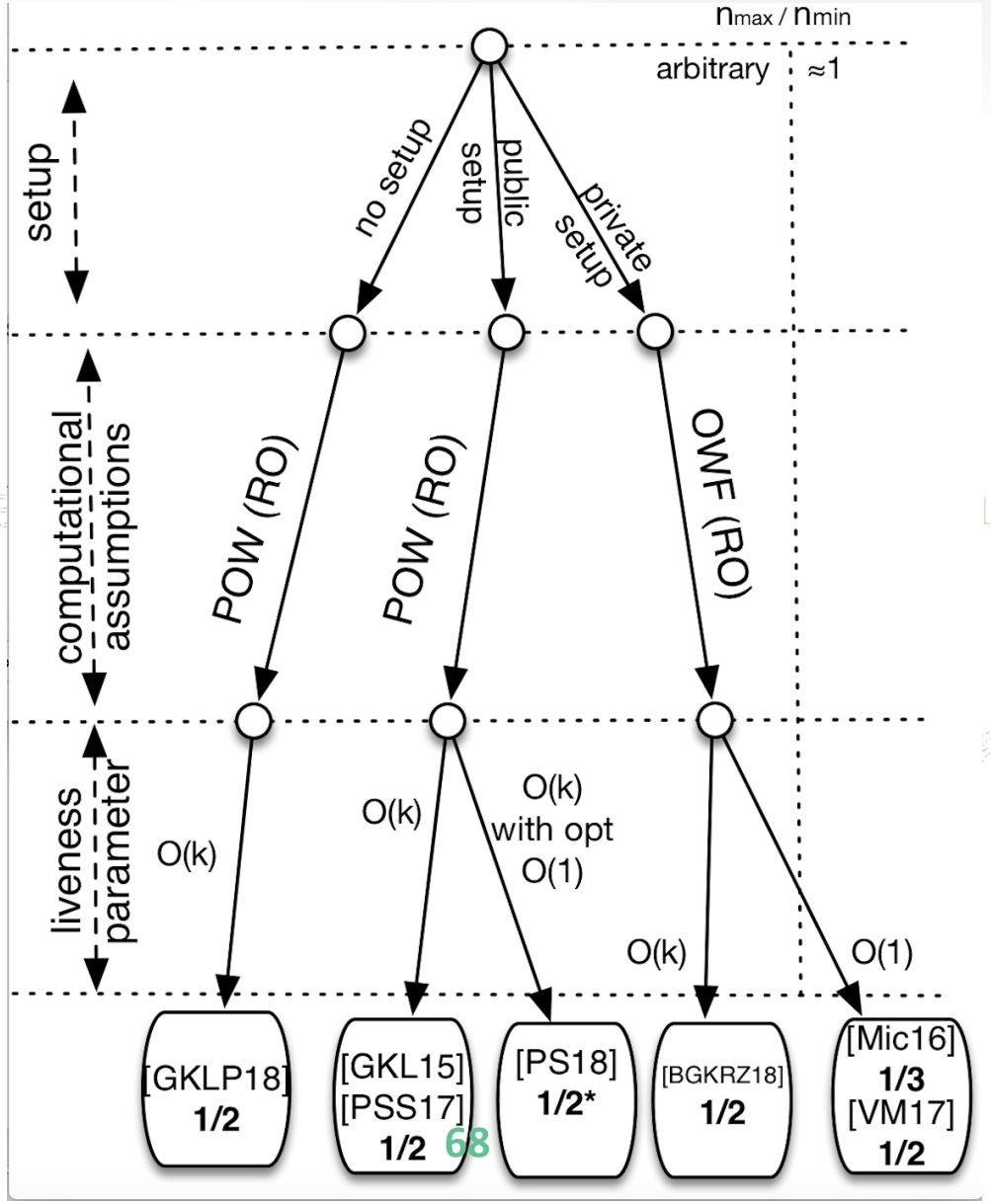
Talk Plan

- Introduction (Consensus/Broadcast)
- Consensus Lower Bounds
- Blockchain-based Consensus
  - Blockchain basics
  - A blockchain abstraction: The Bitcoin backbone protocol
  - Is a genesis block really needed?: Consensus from scratch
- A Consensus Taxonomy
- A Ledger Consensus Taxonomy
- References

Ledger Consensus (aka Nakamoto Consensus)

- State machine replication [Sch90]
- Set of servers operate continuously, accepting and incorporating inputs (transactions) from clients in a data structure called the "ledger"
- **Consistency (Persistence):** Once an honest party ("miner") reports a transaction ("deep enough") in the ledger, then all honest parties will report it indefinitely
- **Liveness:** Transactions are eventually inserted into the ledger

A Ledger Consensus Taxonomy



References

- J. Garay, A. Kiayias and N. Leonardos, “The Bitcoin Backbone Protocol: Analysis and Applications.” *Eurocrypt 2015.* Full version at Cryptology ePrint Archive: Report 2014/765, http://eprint.iacr.org/2014/765
- J. Garay, A. Kiayias and G. Panagiotakos, “Consensus from Signatures of Work.” *CT-RSA 2020*. Cryptology ePrint Archive 2017/775, https://eprint.iacr.org/2017/775
- J. Garay, A. Kiayias, G. Panagiotakos and N. Leonardos,“Bootstrapping the Blockchain, with Applications to Consensus and Faster PKI Setup.” *PKC 2018.* Cryptology ePrint Archive: Report 2016/991, http://eprint.iacr.org/2016/991
- J. Garay, A. Kiayias, R. Ostrovsky, G. Panagiotakos and V. Zikas, “Resource-Restricted Cryptography: Revisiting MPC Bounds in the Proof-of-Work Era.” *Eurocrypt 2020* (to appear). Cryptology ePrint Archive 2019/1264, https://eprint.iacr.org/2019/1264
- J. Garay, A. Kiayias and G. Panagiotakos, “Blockchains from Non-Idealized Hash Functions.” Cryptology ePrint Archive 2019/315, https://eprint.iacr.org/2019/315


Thank You