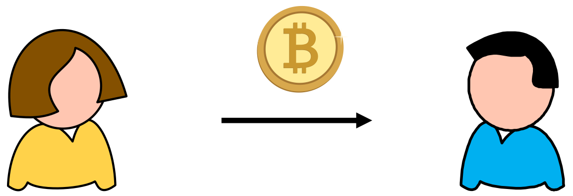
Foundational Aspects of Blockchain Protocols

Juan A. Garay
Texas A&M University
garay@cse.tamu.edu

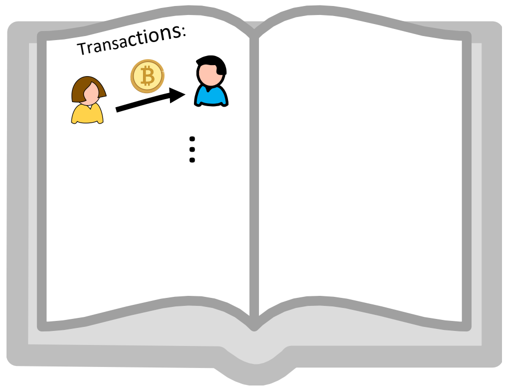
Based on joint work with Aggelos Kiayias (U. of Edinburgh), Nikos Leonardos (U. of Athens) and Giorgos Panagiotakos (U. of Edinburgh)

Motivation

1) The problem: digital money transfer

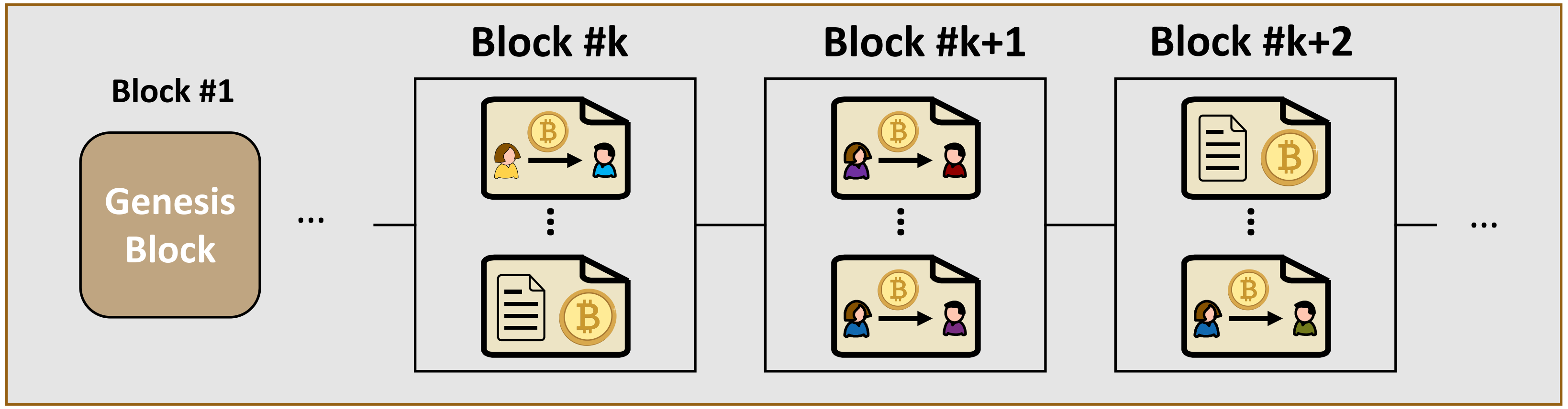


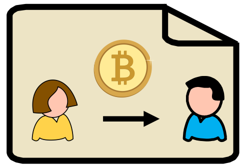
2) The accounting: implement a ledger!



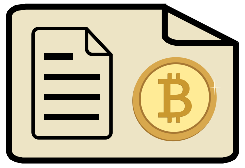
Blockchain Abstractions

A Public Ledger or a Bulletin Board



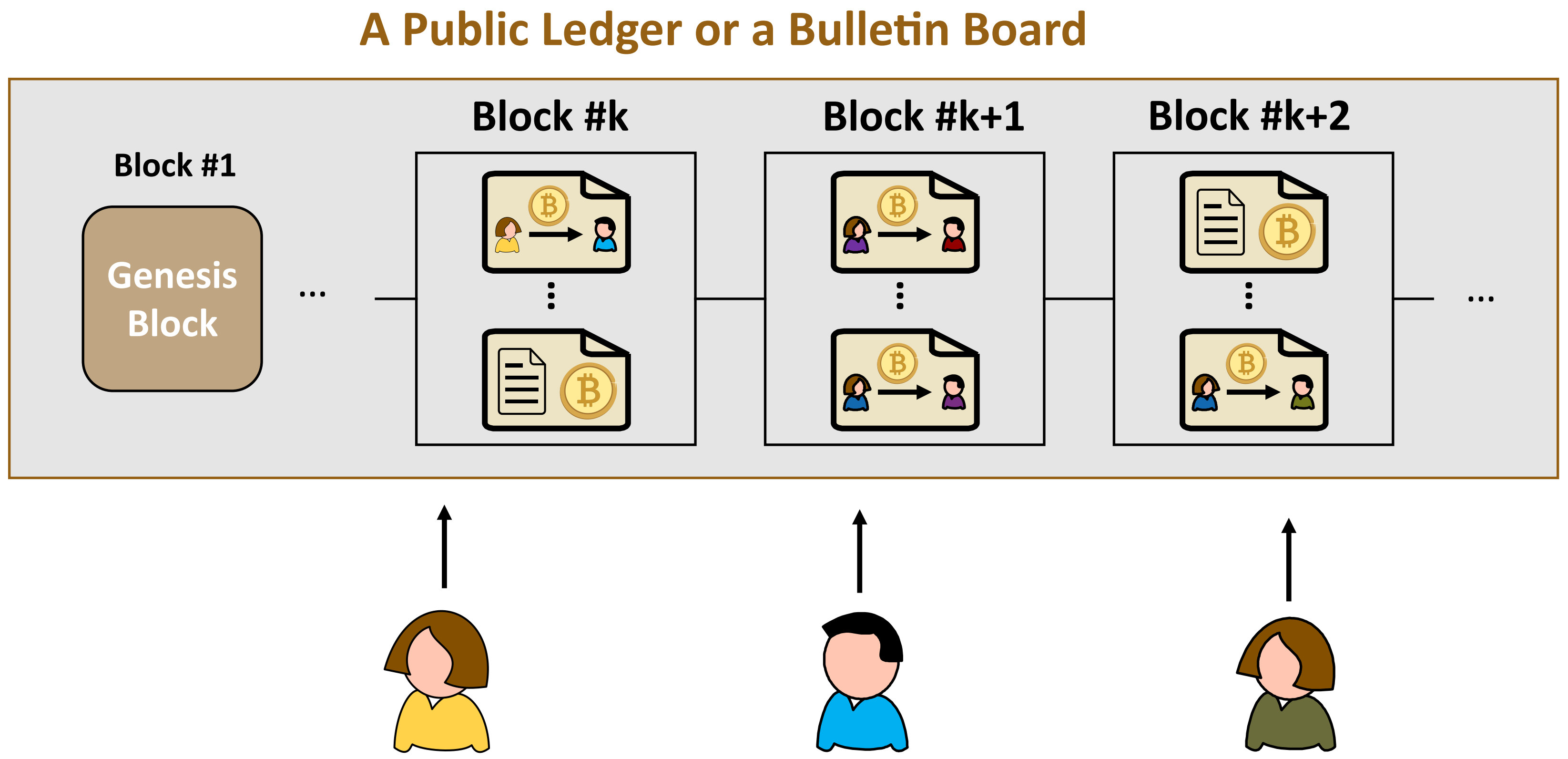


Simple transactions



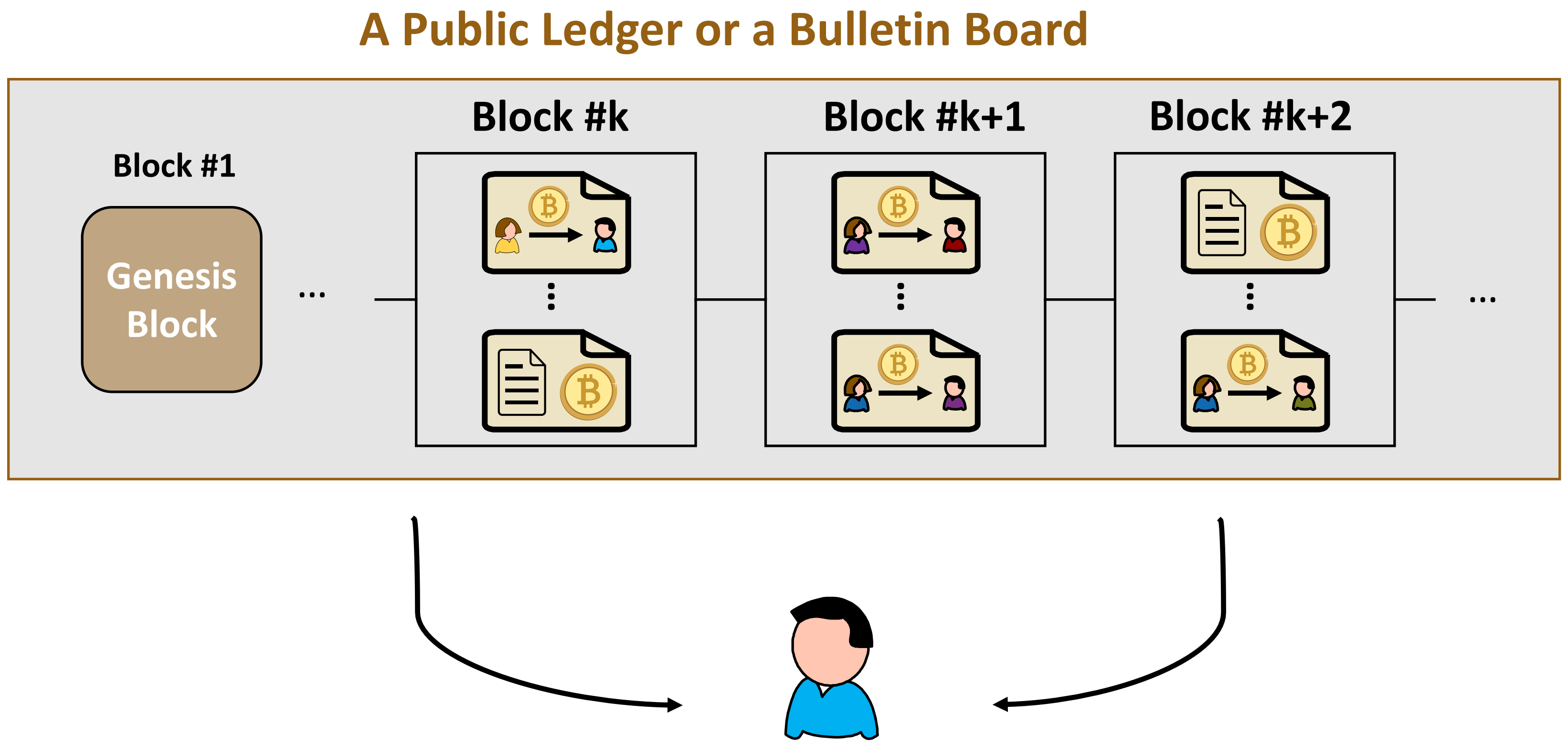
Complex contracts

Blockchain Abstractions



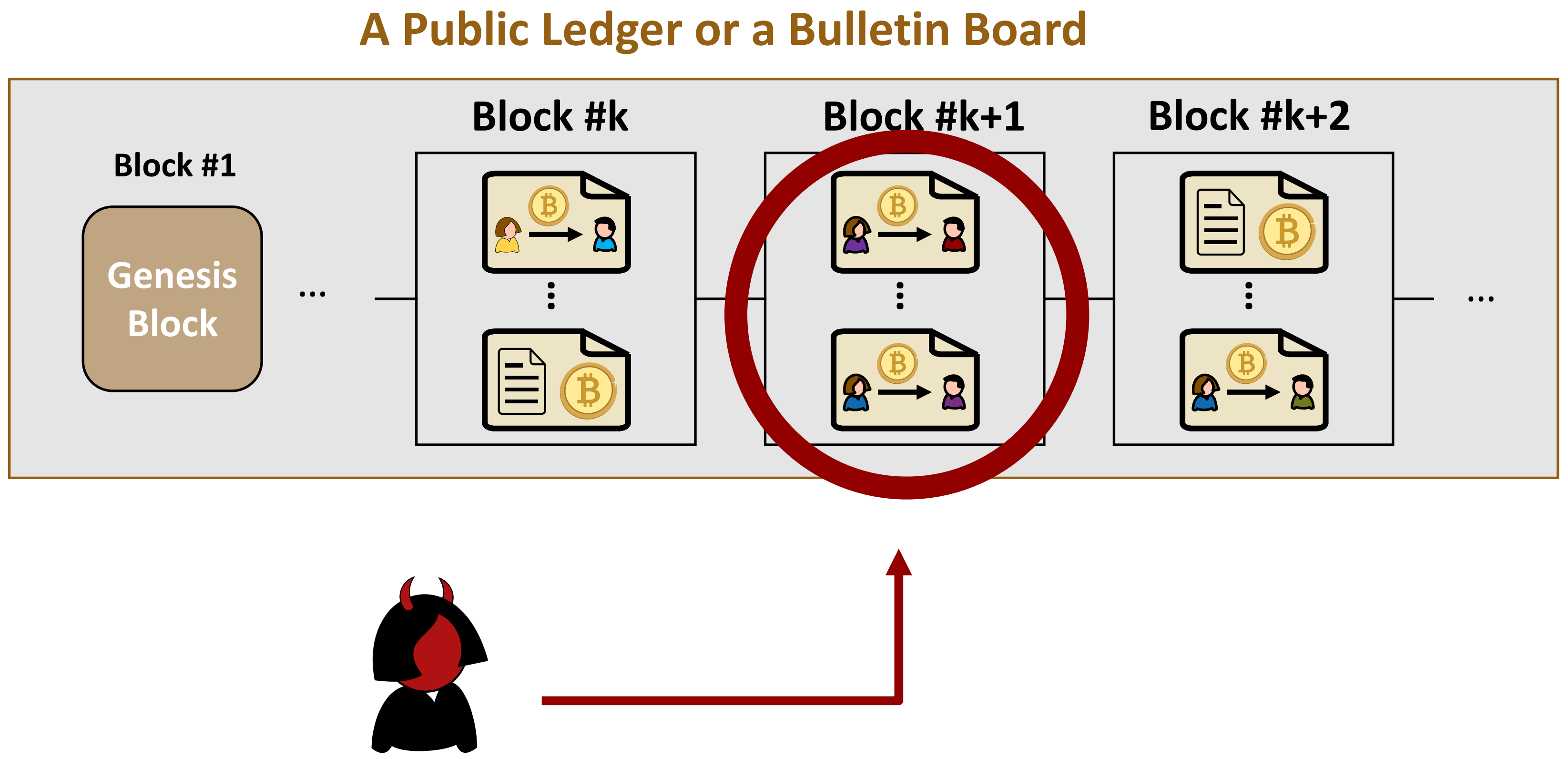
1) Content is provided by any user with sufficient funds

Blockchain Abstractions



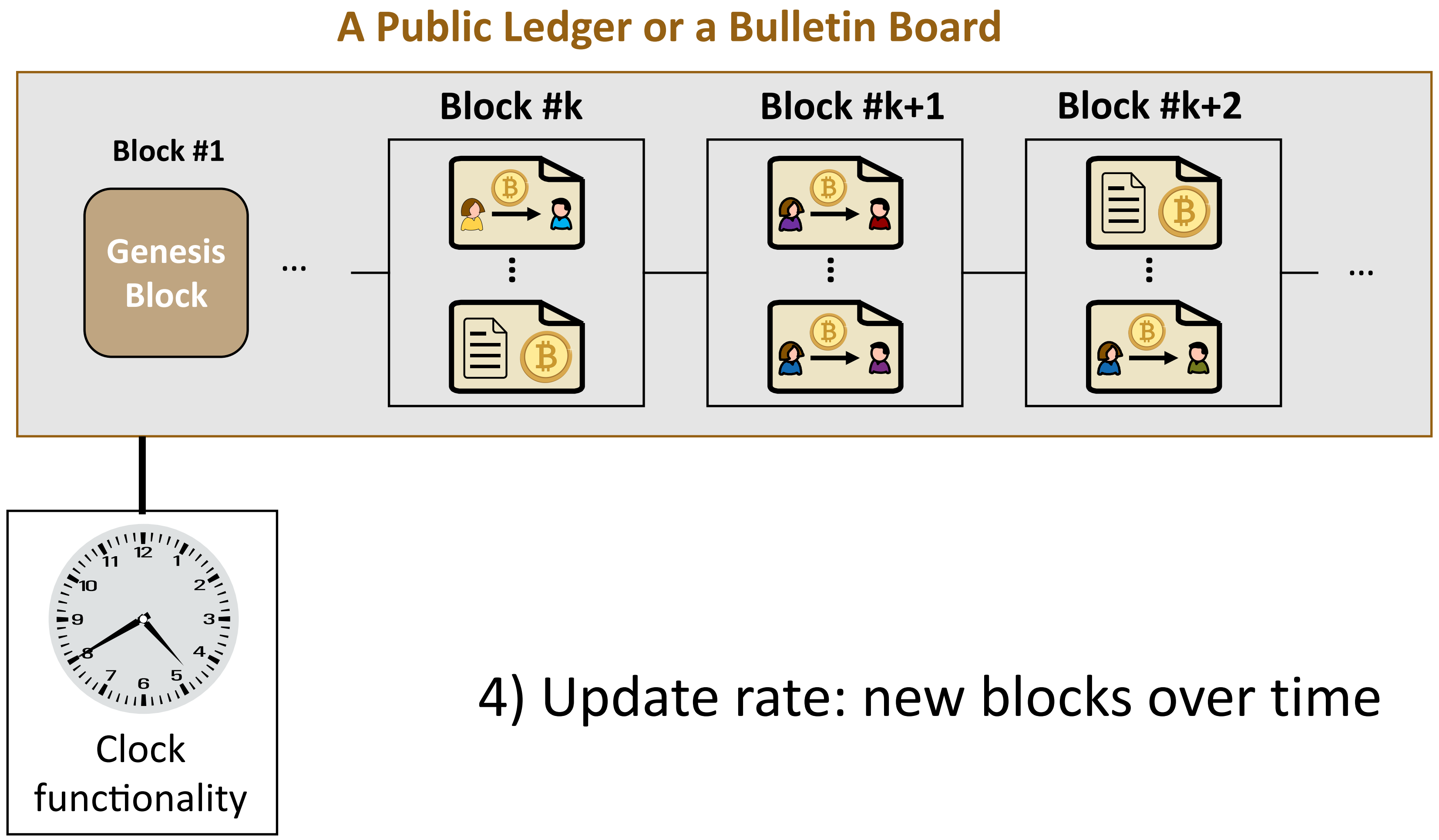
2) Anyone can read the current content of the ledger

Blockchain Abstractions



3) No modifications of contained blocks possible

Blockchain Abstractions



4) Update rate: new blocks over time

Decentralized Payment Systems

- E-cash [Chau82] achieves strong privacy guarantees (untraceability, unlinkability), but it's a centralized approach (bank, national bank/mint)
- First decentralized "*cryptocurrency*" – Bitcoin – proposed in [Nak08]
  - Peer-to-peer payment network
  - No depositor insurance coverage
  - Based on maintaining a public transaction ledger in a distributed manner
- January 2009: the Bitcoin network is created. A number of other cryptocurrencies follow suit
- High impact; a number of other potential applications: contracts, reputation systems, name services, consensus problems, etc.

Bitcoin Parties

Miners

- Do work to maintain the transaction ledger
- Get rewards for their work:
  1. fees
  2. new bitcoins

Payers

- "Broadcast" a transaction stating they send bitcoin
- Rely on security of digital signatures to ensure bitcoins are not stolen

Payees

- Have to generate a Bitcoin address
- Have to verify their address is credited

Bitcoin Address/Account (Security)

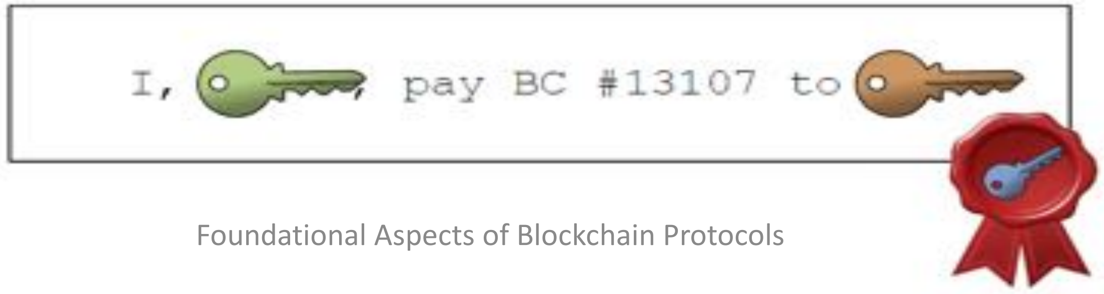
- Based on elliptic curve *secp256k1*
- Account: (PrivKey, PubKey)

- Bitcoin address:

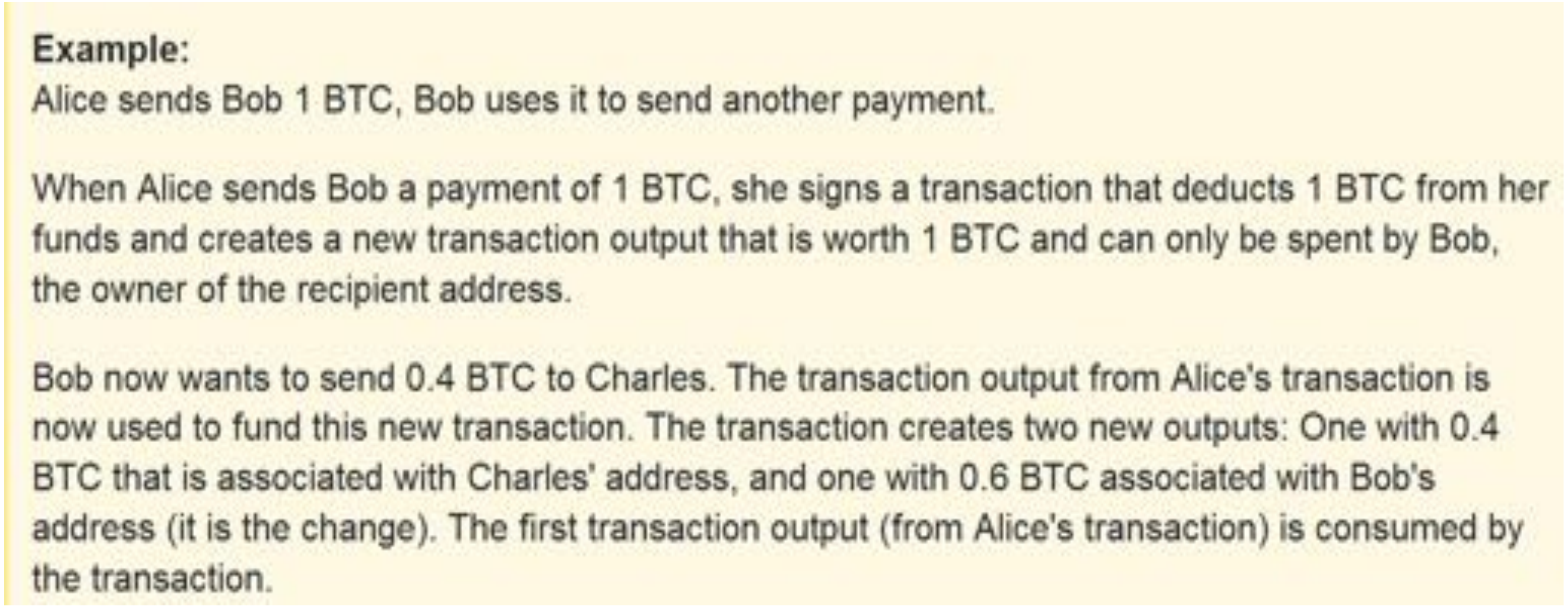
  Base58(RIPEMD160(SHA256(PubKey)))

  E.g., *37k7toV1Nv4DfmQbmZ8KuZDQCYK9x5KpzP*

- PrivKey used to sign outgoing transactions
- Wallet: many (PrivKey, PubKey)
- Transaction:



Bitcoin Transactions



Example:
Alice sends Bob 1 BTC, Bob uses it to send another payment.

When Alice sends Bob a payment of 1 BTC, she signs a transaction that deducts 1 BTC from her funds and creates a new transaction output that is worth 1 BTC and can only be spent by Bob, the owner of the recipient address.

Bob now wants to send 0.4 BTC to Charles. The transaction output from Alice's transaction is now used to fund this new transaction. The transaction creates two new outputs: One with 0.4 BTC that is associated with Charles' address, and one with 0.6 BTC associated with Bob's address (it is the change). The first transaction output (from Alice's transaction) is consumed by the transaction.

Valid Transactions

- Transactions are organized by miners in a *transaction ledger* τ

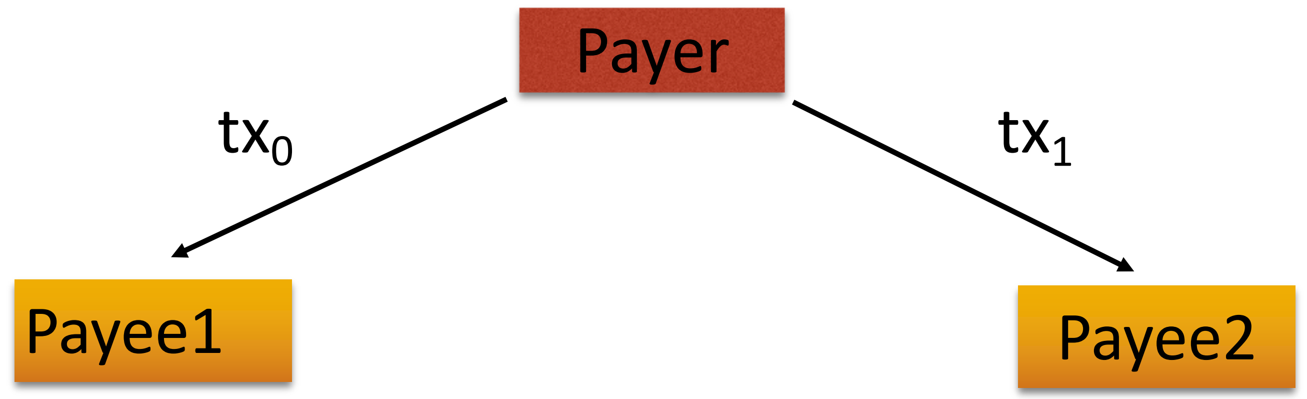
- There is a well-defined public predicate that given a transaction ledger and a transaction decides whether the transaction “makes sense”

$$\mathrm{Valid}(\tau, tx) \in \{\mathrm{True}, \mathrm{False}\}$$

- Each miner will accept a transaction only if it is valid given its *local view* of the ledger

Double-spending Bitcoin

- The "litmus test" for any payment system



- *Double-spending* transactions are inconsistent:

$$tx_b \in \tau \rightarrow \text{Valid}(\tau, tx_{1-b}) = \text{False}$$

- No honest miner will accept an invalid transaction
- As long as miners *agree* on τ double-spending is infeasible

Double-spending Bitcoin (2)

- If *single* miner exists (cf. bank in e-cash), then double-spending is infeasible – but Bitcoin would be guaranteed solely by that entity
- How to facilitate multiple miners while preventing double-spending?
- How to scale this to thousands (…millions?) of users at a global scale and maintain security…
  - …in such a "*permissionless*" model?

The Bitcoin Model

- Not a traditional distributed system
  - Nodes known *a priori* and authenticated
  - Paxos [Lam78], Raft [OO14] , Byzantine Fault Tolerance, Secure Multi-party Computation [Yao82, GMW87, BGW88,…]

The Bitcoin Model

- Not a traditional distributed system
    - Nodes known *a priori* and authenticated
    - Paxos [Lam78], Raft [OO14] , Byzantine Fault Tolerance, Secure Multi-party Computation [Yao82, GMW87, BGW88,…]

- The “permissionless” model
    - Nodes do *not* know each other (not even their exact number!)
    - Nodes come and go
    - *Anyone* can join

The Bitcoin Model

- Not a traditional distributed system
  - Nodes known *a priori* and authenticated
  - Paxos [Lam78], Raft [OO14] , Byzantine Fault Tolerance, Secure Multi-party Computation [Yao82, GMW87, BGW88,…]

- The “permissionless” model
  - Nodes do *not* know each other (not even their exact number!)
  - Nodes come and go
  - *Anyone* can join

- And yet, realize a distributed ledger

Distributed Ledger

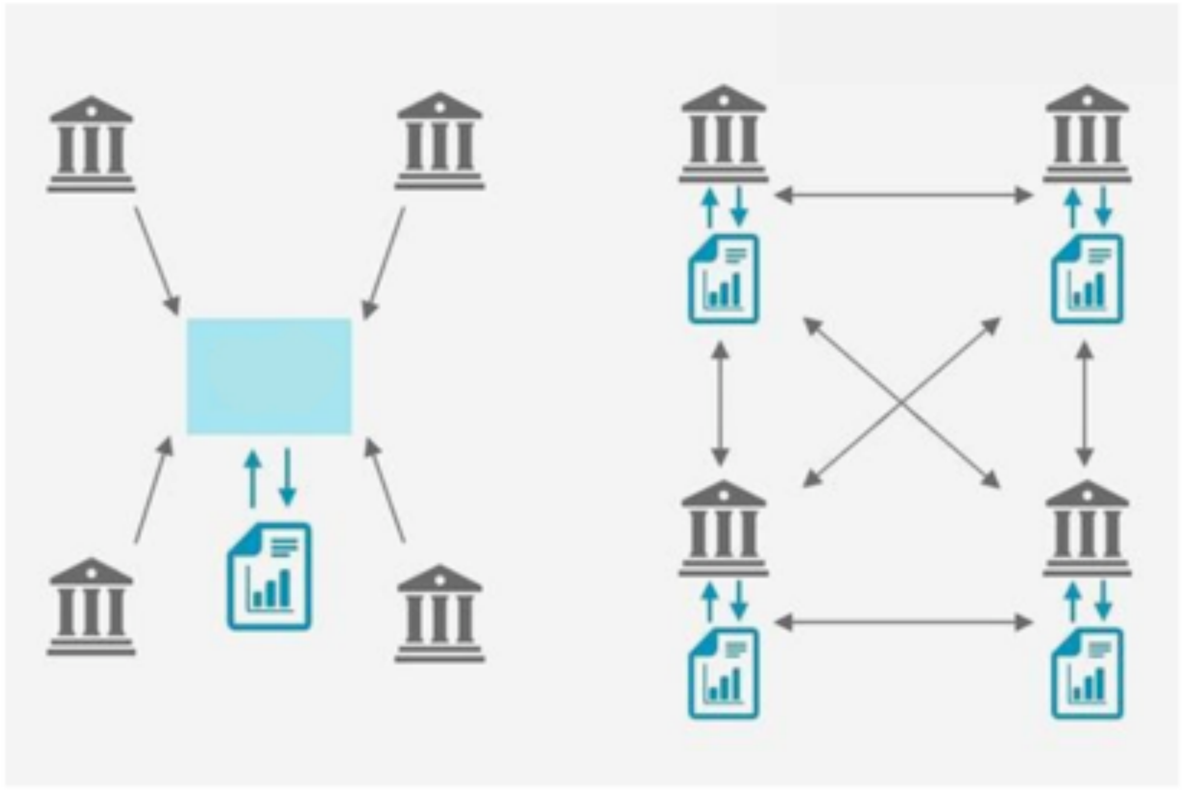
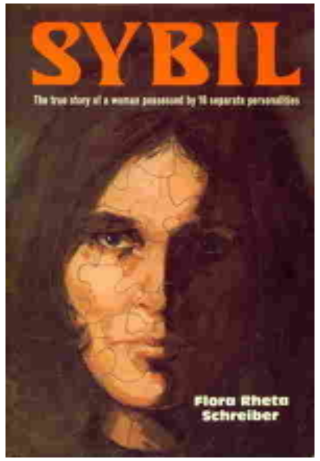


Figure: http://blogs.wsj.com/cio/2016/02/02/cio-explainer-what-is-blockchain/

- Consistency: Everyone sees the same history
- Liveness: Everyone can add new transactions

The “Permissionless” Model [Nak08]

- Strong impossibility results w/o authentication [Oku05, BCLPR05]
- *Sybil* attacks are unavoidable [Dou02,…]
- ⅓ barrier in no. of misbehaving parties [Bor96, Fit03]



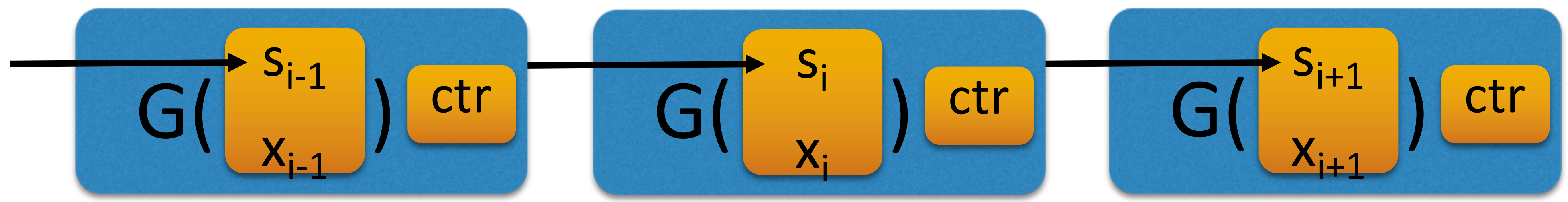
(Informal) Answer:

- The *“blockchain,”* based on Proofs of Work
- **Claim** [Nak08]**:** The blockchain realizes a “distributed ledger,” assuming an *honest majority*
  - Consistency: Everyone sees the same history
  - Liveness: Everyone can add new transactions

Talk Plan

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
  - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
  - Common Prefix, Chain Quality, Chain Growth
- Applications
  - Consensus
  - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
- References

What Is a Blockchain?



What Is a Blockchain?

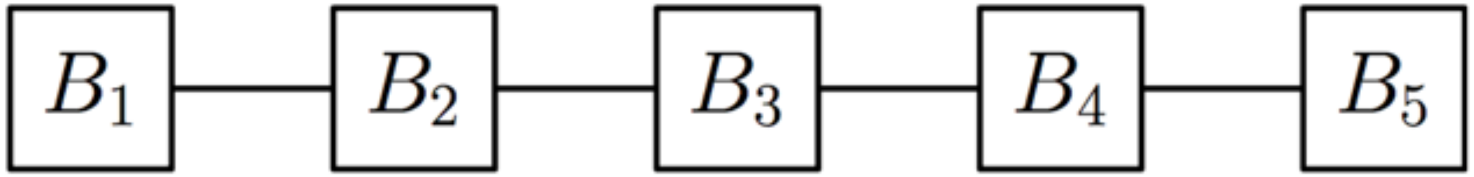
- Parties ("miners") have to do work in order to install a transaction

What Is a Blockchain?

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

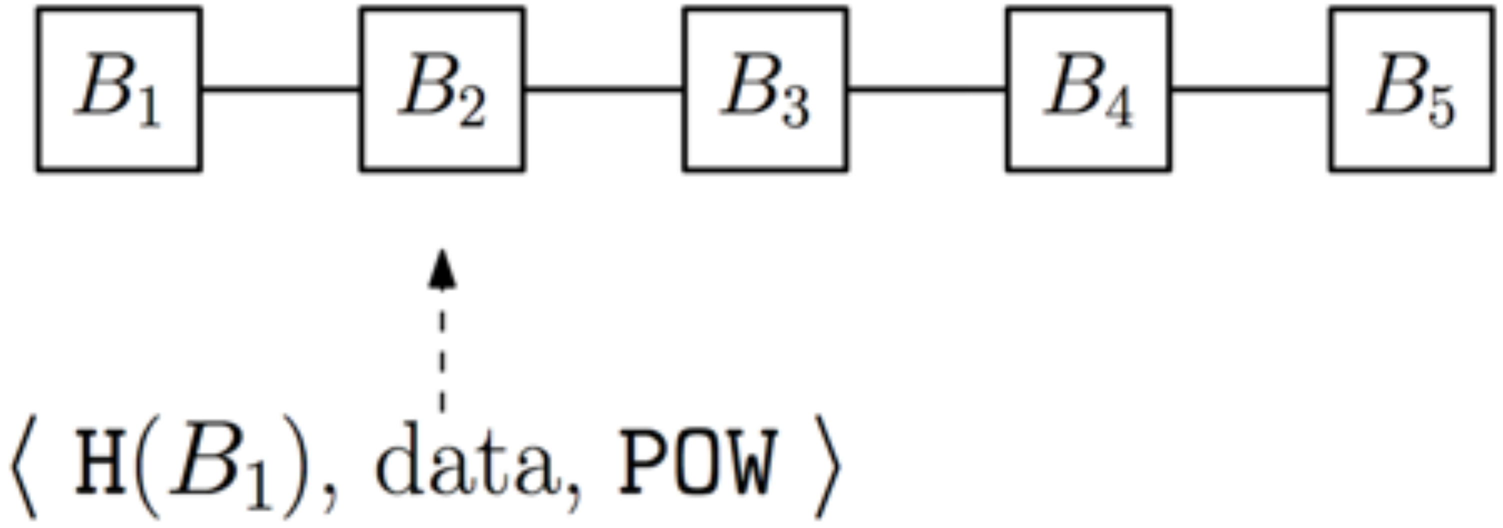
What Is a Blockchain?

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



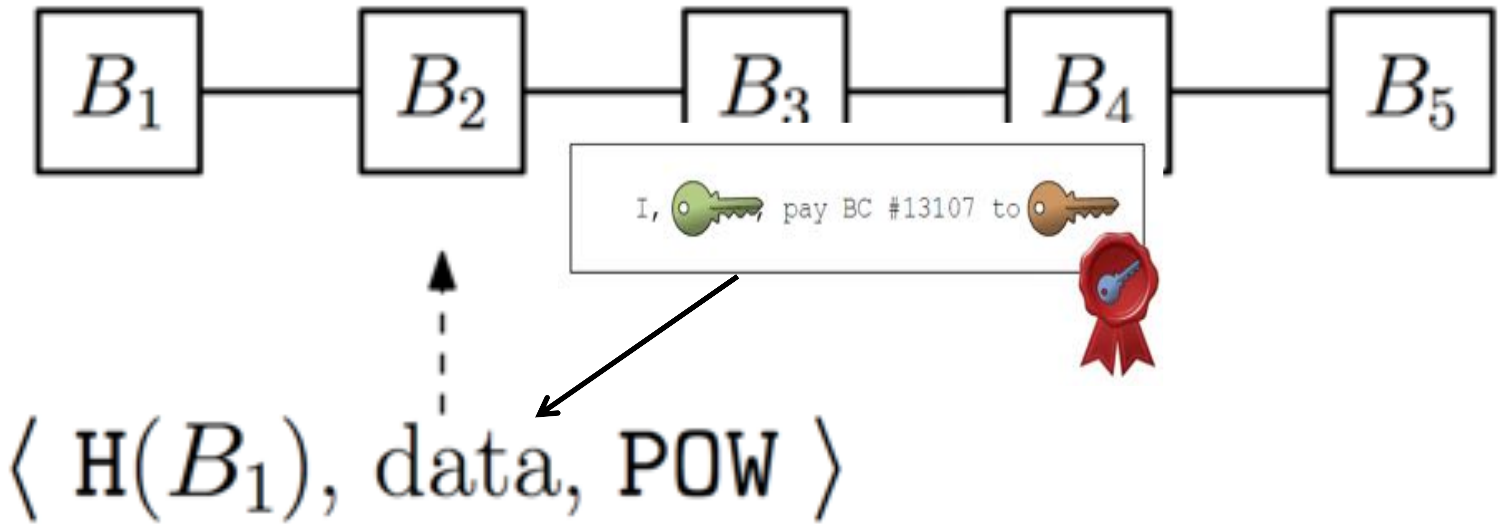
What Is a Blockchain?

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


B_1
B_2
B_3
B_4
B_5
$\langle\ \mathtt{H}(B_1),\ \text{data},\ \mathtt{POW}\ \rangle$

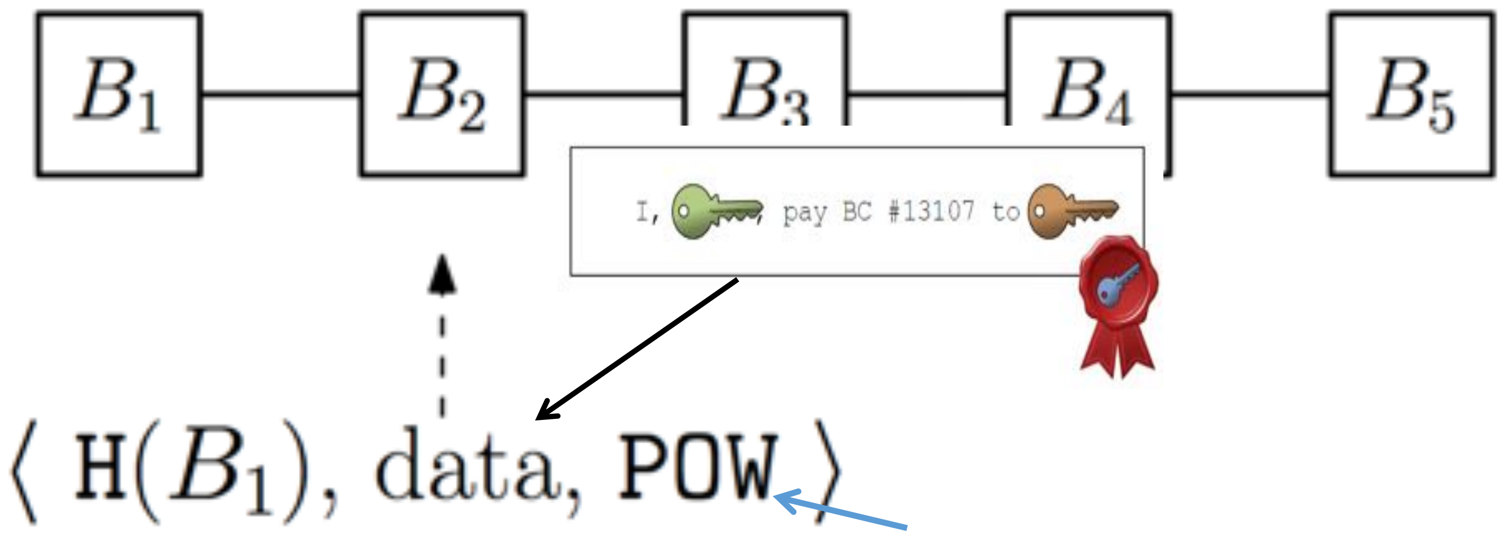
What Is a Blockchain?

- 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

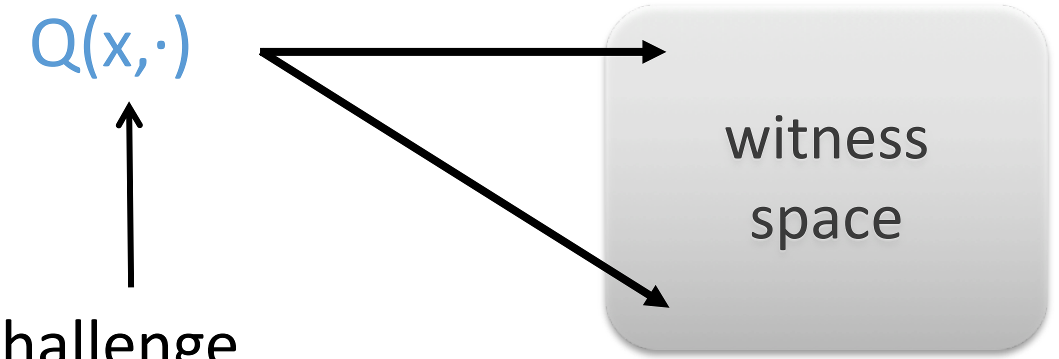


Proof of Work
("Crypto puzzle")

Proofs of Work (aka "Crypto Puzzles")

- "Moderately hard functions" [DN92,RSW96,Back97,JB99,GMPY06…]

Q(·,·): Polynomial-time predicate



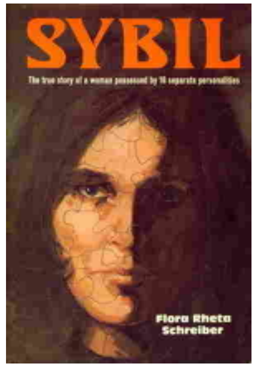
Successful only with some (small) probability!

Challenge determines work level d

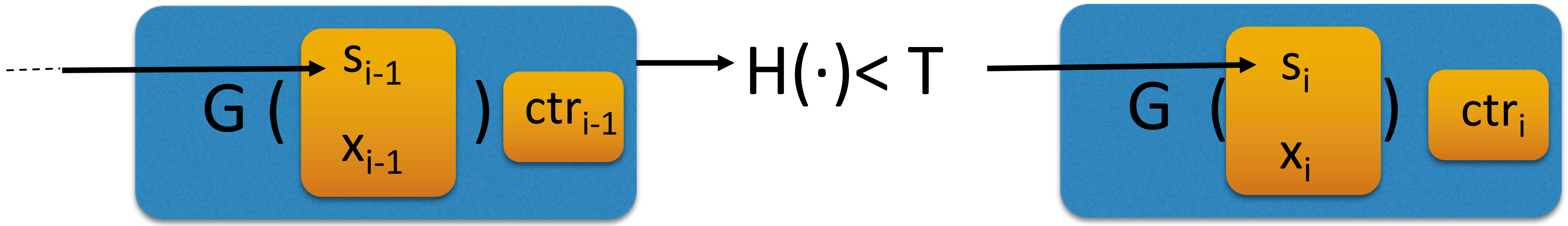
Search for witness takes long time (e.g., exp(d)) (a complexity lower bound needs to be assumed)

Verification is easy!

Proofs of Work [DN92,...]

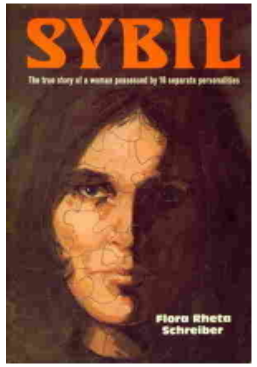

SYBIL
The true story of a woman possessed by 16 separate personalities
Flora Rheta Schreiber

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

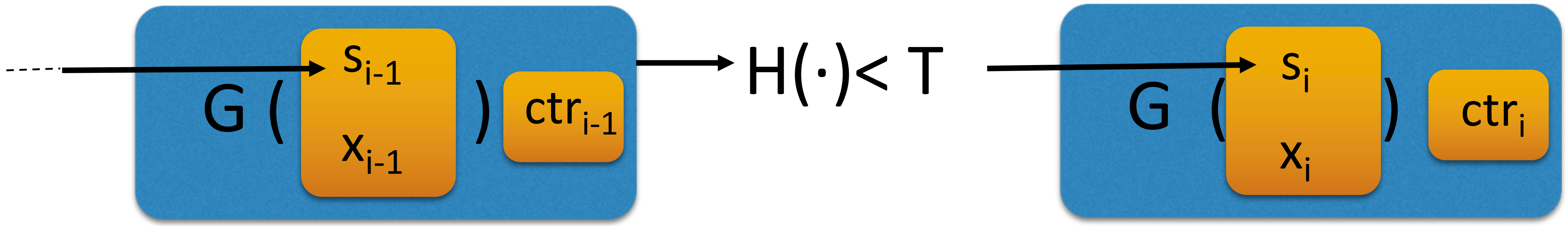


- This talk: PoW
  - Proof-of-* (* = stake, space/memory,...)

Proofs of Work [DN92,…]


SYBIL
The true story of a woman possessed by 16 separate personalities
Flora Rheta Schreiber

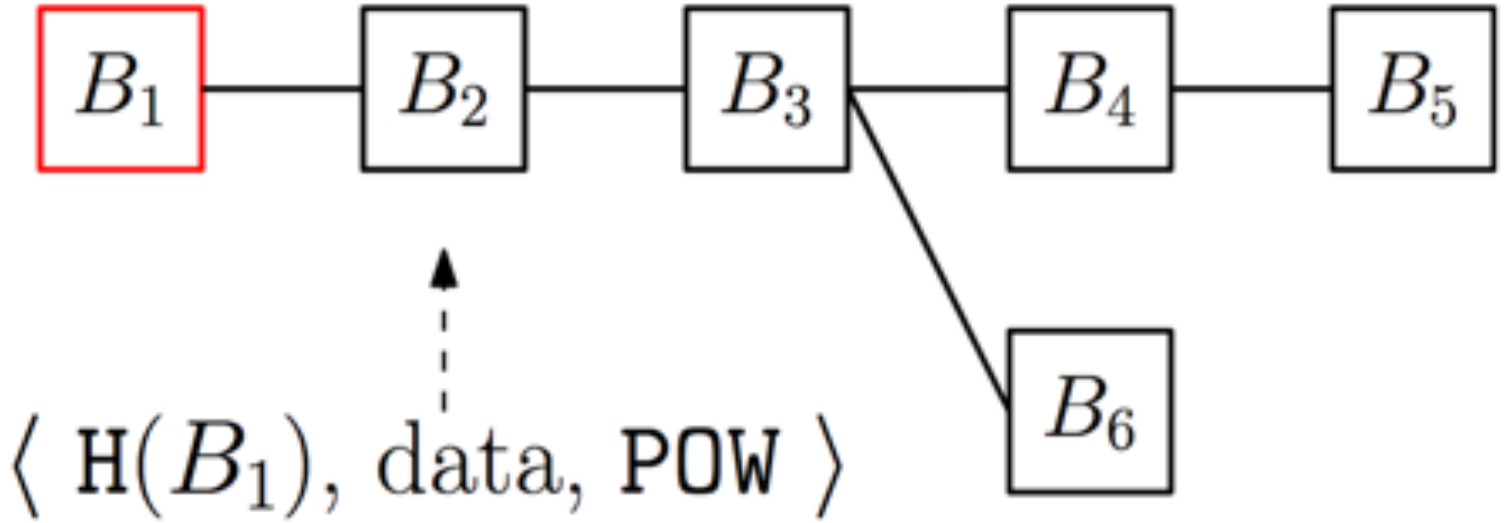
- Spam mitigation, Sybil attacks, denial of service protection, …
- Most impactful application: Design of blockchain protocols such as Bitcoin



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

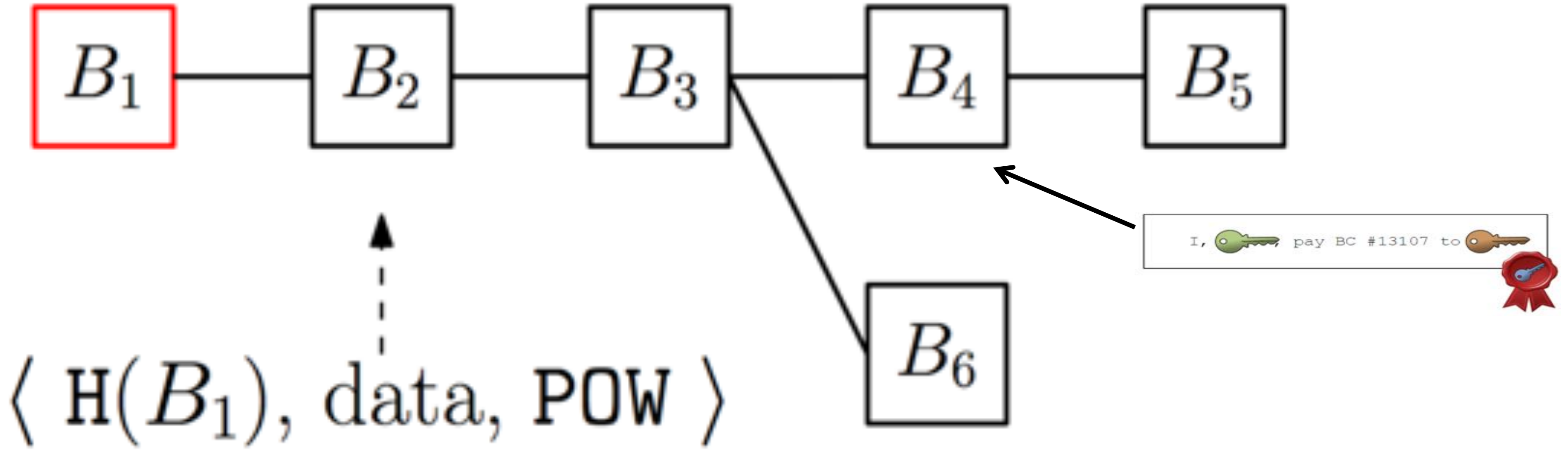
What Is a Blockchain?

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


B_1
B_2
B_3
B_4
B_5
B_6
$\langle\, \mathtt{H}(B_1),\ \text{data},\ \mathtt{POW}\, \rangle$

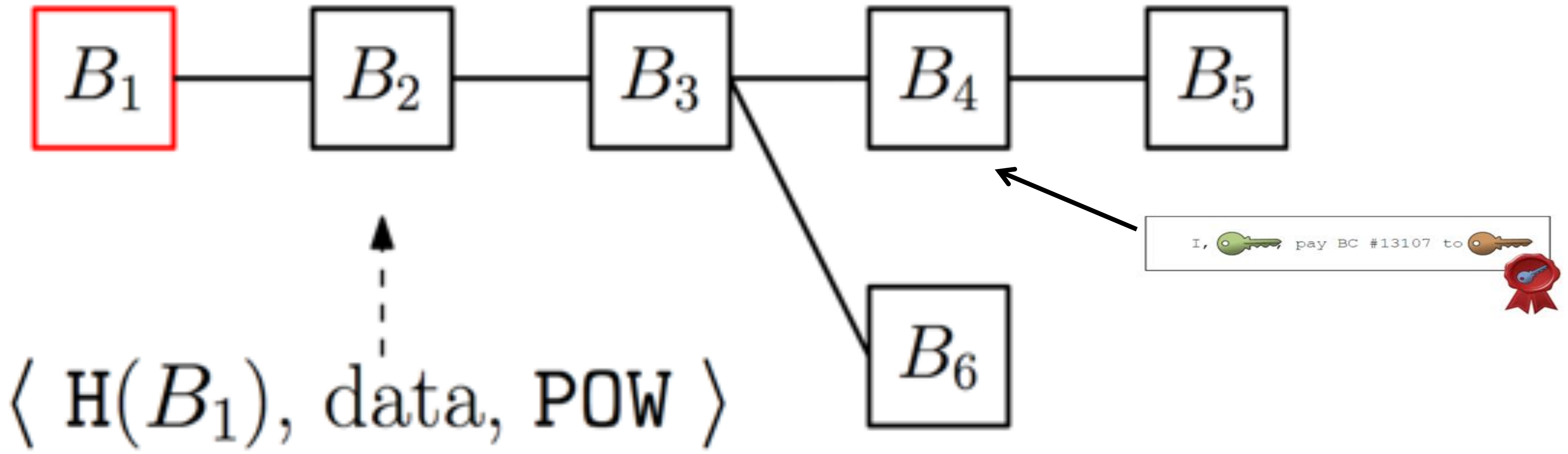
What Is a Blockchain?

- 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?

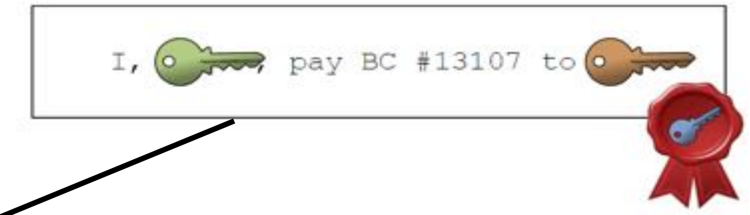
- Parties ("miners") always choose the *longest* chain they received
- If party wants to erase his transaction, he has to find a longer chain!



B_1
B_2
B_3
B_4
B_5
B_6
I, pay BC #13107 to
$\langle\ \mathtt{H}(B_1),\ \text{data},\ \mathtt{POW}\ \rangle$

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

Using POWs



SHA-256(·)

- 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 = 00000000000000171A8B00000000000000000000000000000000000000000000)

- If **while** loop terminates "broadcast" (τ,ctr,**tx**)

  (new "block": state, counter, set of transactions)

Using POWs (2)

- If a vector $(\tau', ctr', \mathbf{tx}')$ is received, check

$$(\tau = \tau') \wedge (\mathrm{Hash}(ctr'; \mathrm{Hash}(\tau, \mathbf{tx}')) \leq T)$$

- Extend the transaction ledger

$$\tau := \tau' \,\|\, \mathbf{tx}'$$

$$\tau := \tau' \,\|\, \mathbf{tx}'$$

("blockchain": sequence of above triples – denoted C)

Longest Chain Wins

- Size (difficulty) *does matter* in Bitcoin:
  - If $(\tau \neq \tau')$ then miners compare their respective sizes in terms of difficulty/number of blocks

$(\tau \neq \tau')$

- Miners' basic rule: If my chain is not smaller, I keep it; else I switch to the new one

Blockchain Protocols (*circa* 2014)

- Bitcoin white paper [Nak08a]
  - Preliminary (limited) analysis
- "The proof-of-work chain is a solution to the *Byzantine Generals* problem" [Nak08b]
- Further analysis: [Decker-Wattenhofer13, Miller-LaViola14, Eyal-Sirer14,…]

Our work:

- What are the ***provable properties*** of this (type of) protocol(s)?
- In what sense does it implement a "robust ledger"?
- What is the connection to the ***consensus*** problem?
- Lay out computational model, trust assumptions and cryptographic tools

A Blockchain Abstraction: The Bitcoin *Backbone* Protocol [GKL15]

The Bitcoin Backbone [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

Network/Computational Model (2)

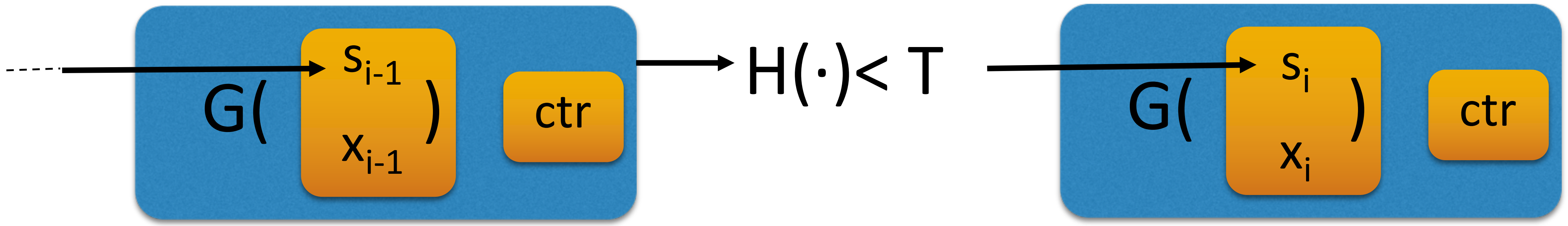
- In each round, each party is allowed q queries to a cryptographic hash function (*random oracle* [BR93])
  - "Flat" version of the world in terms of hashing power
  - t parties controlled by adversary, acting as a malicious mining pool
    $\rightarrow$ $t \cdot q$ queries/round
  - $t < n/2$ corresponds 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* $\mathcal{C}$ of the form:

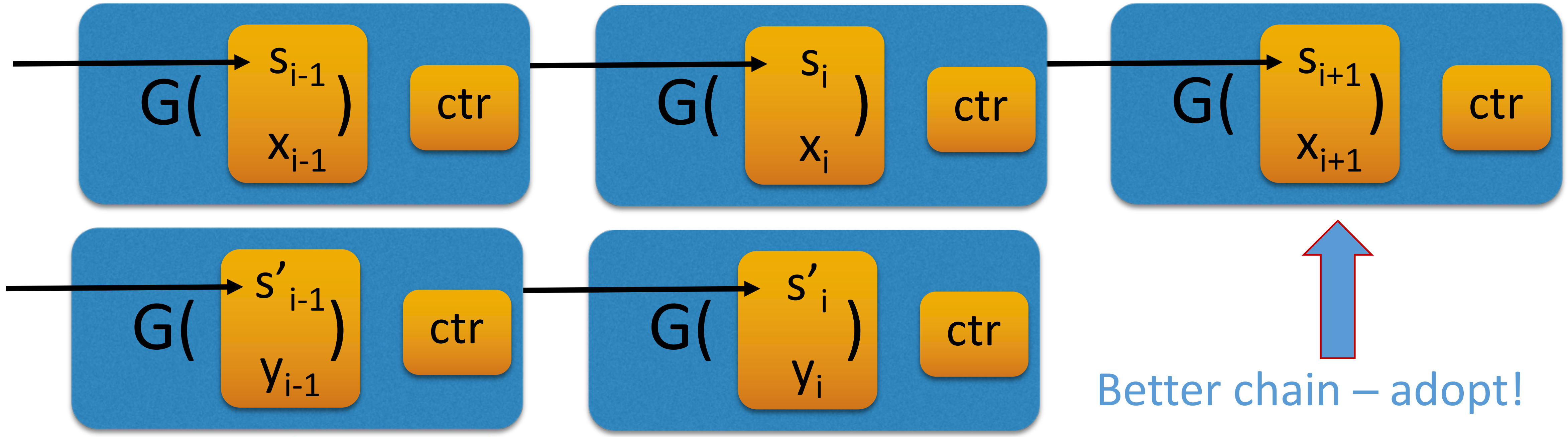


- The *contents* of $\mathcal{C}$ satisfy the predicate $V(x_1,\ldots,x_i) = \text{true}$

The Bitcoin Backbone (2)

- Parameterized by V(), I(), R(), and hash functions G(), H()

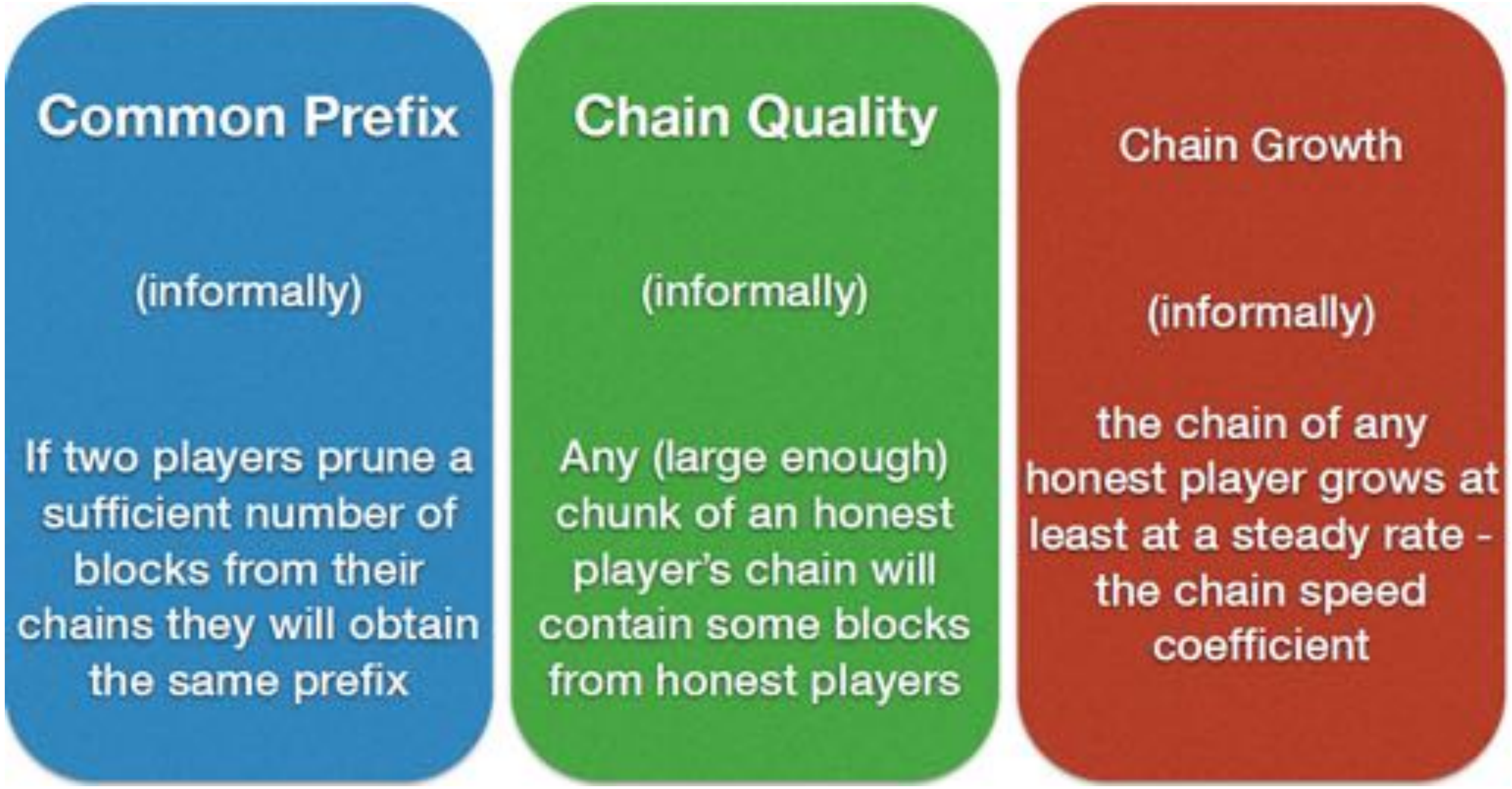
- A player will compare any incoming chains and the local chain wrt their length/difficulty:


G(s_{i-1} x_{i-1}) ctr
G(s_i x_i) ctr
G(s_{i+1} x_{i+1}) ctr
G(s'_{i-1} y_{i-1}) ctr
G(s'_i y_i) ctr
Better chain – adopt!

- Finally, when requested, a player will return $R(x_1,\ldots,x_i)$

Basic Properties of the Blockchain

Backbone Protocol Properties



Common Prefix

(informally)

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

Chain Quality

(informally)

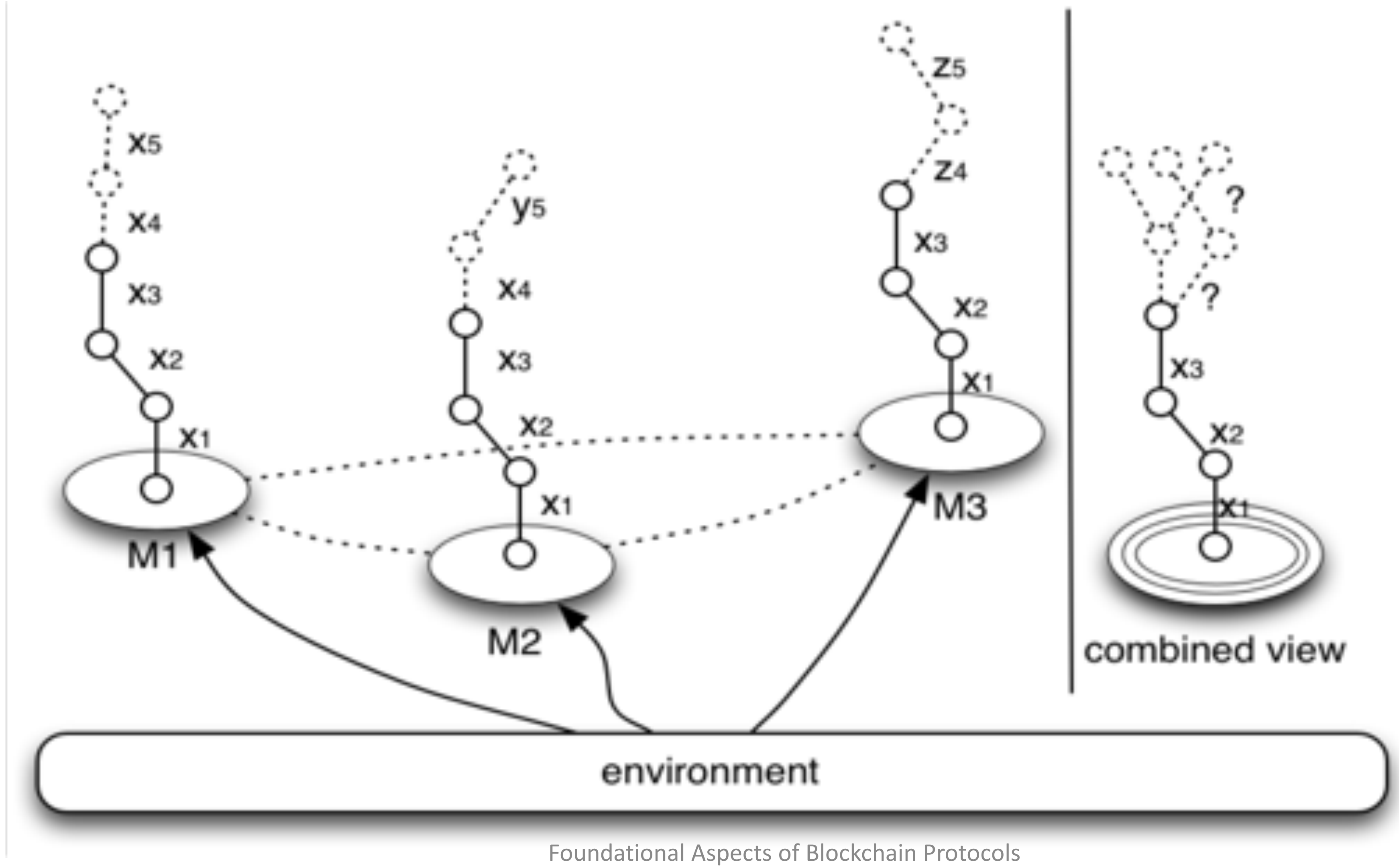
Any (large enough) chunk of an honest player's chain will contain some blocks from honest players

Chain Growth

(informally)

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

Common Prefix: Will Miners Converge?



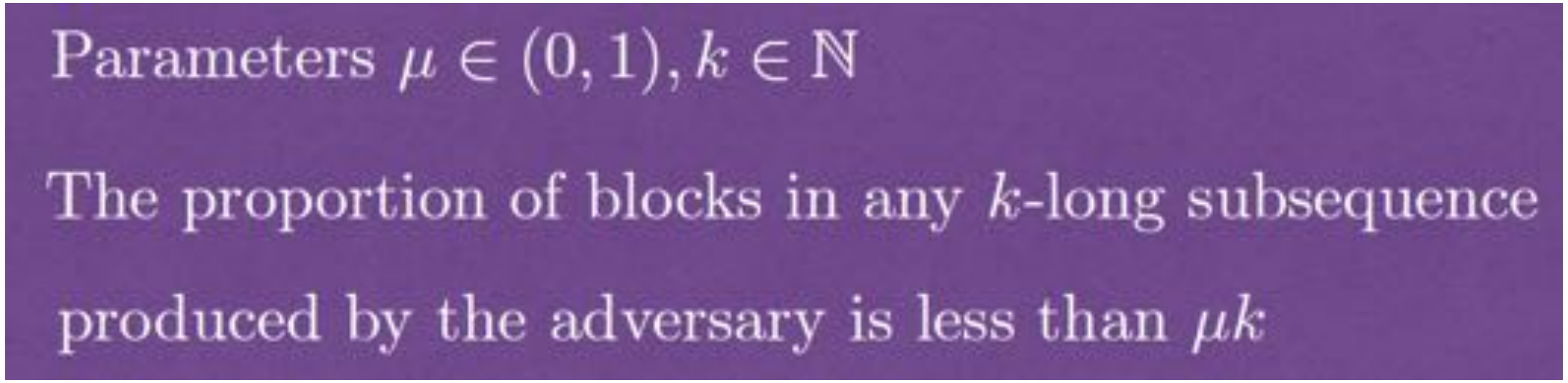
Common Prefix Property

- If two honest parties “prune” sufficiently many blocks from their chains, they get the same prefix

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \ \text{with } \mathcal{C}_1, \mathcal{C}_2 : \ \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

Chain Quality Property

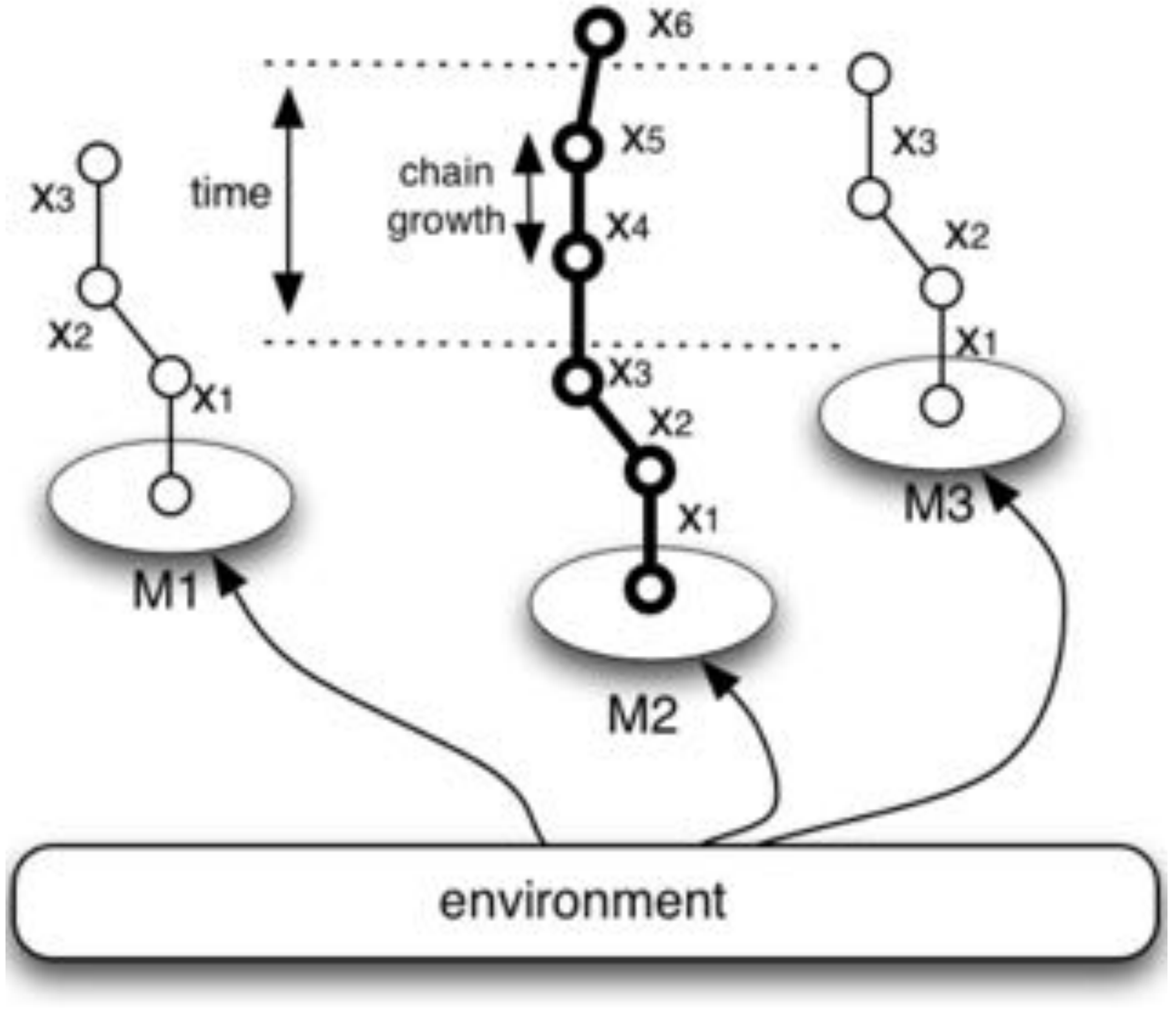
- Will honest blocks be adopted by the honest parties?



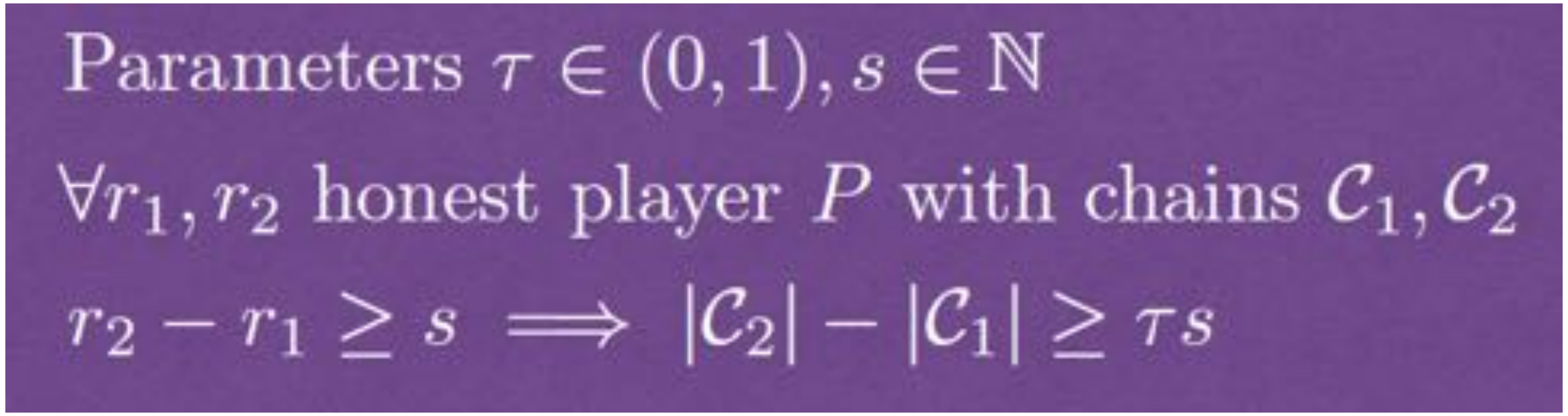
Parameters $\mu \in (0,1), k \in \mathbb{N}$

The proportion of blocks in any k-long subsequence produced by the adversary is less than μk

Chain Growth: Does the Blockchain Grow?



Chain Growth Property


Parameters $\tau \in (0,1), s \in \mathbb{N}$

$\forall r_1, r_2$ honest player P with chains $\mathcal{C}_1, \mathcal{C}_2$

$$r_2 - r_1 \geq s \implies |\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau s$$

Backbone Protocol Properties (3)

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

Proof Strategy

1. Define the notion of *typical execution*
2. Argue that typical executions happen with overwhelming probability
3. Prove CP, CQ, CG
4. (Modularly) Derive application’s properties -- e.g., *Consistency* and *Liveness*

Notation

$n - t, t$: Honest and malicious miners, resp.

$n - t > \lambda(1+\delta)\, t$ (*Honest Majority* assumption, where $\delta > 2\epsilon + f$)

T: "Target" hash value (difficulty level of POW)

κ: Security parameter (range of $H()$)

q: Number of POW attempts per round

$p = T / 2^{\kappa}$

$f = 1 - (1 - p)^{q(n-t)}$: Prob. at least one honest party produces a POW in a round. Observe that $f \approx pq(n-t)$. ($f << 1$)

Notation (2)

S: Sequence of consecutive rounds

X(S): Number of *successful rounds*

Y(S): Number of *uniquely successful rounds*

Z(S): Total no. of *adversarial* POWs computed during S

Expectations:

$E[X(S)] \approx pq(n-t)\ |S|$

$E[Y(S)] > (1-f)\ E[X(S)]$

$E[Z(S)] = pqt\ |S|$

Given that $n-t \,/\, t > \lambda(1+\delta)$:

$E[X(S)] > (1+\delta)\ E[Z(S)]$

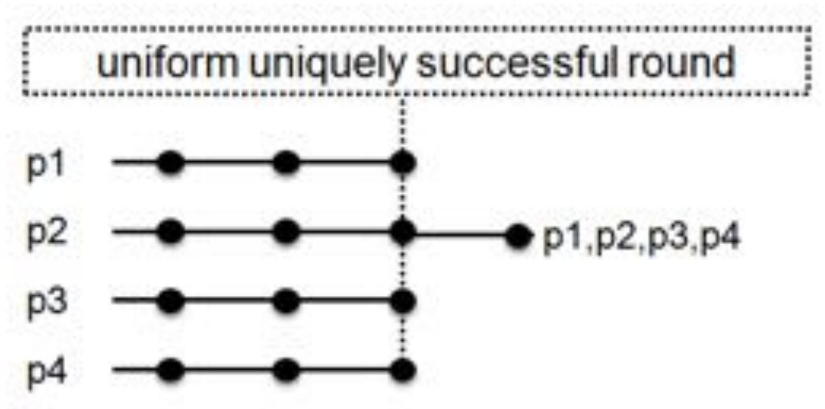
$E[Y(S)] > (1-f)(1+\delta)\ E[Z(S)]$

Common Prefix Property

- Recall:

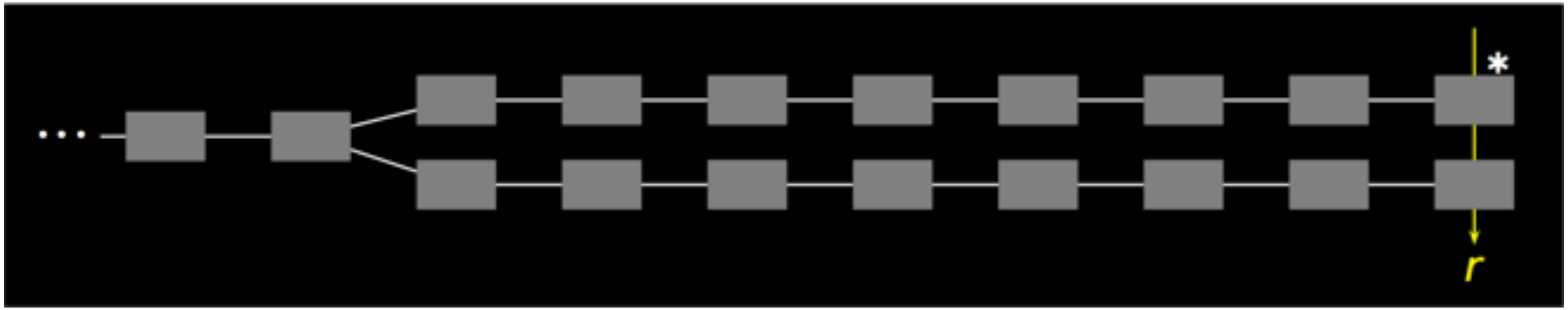
$$\forall r_1, r_2, (r_1 \le r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

- Set k = Ω(κ)
- *Uniquely successful round:* Round when exactly one honest party generates a POW



Common Prefix Property

- **Theorem:** The probability that at a given round two honest parties have chains that disagree is at most $e^{-\Omega(\kappa)}$

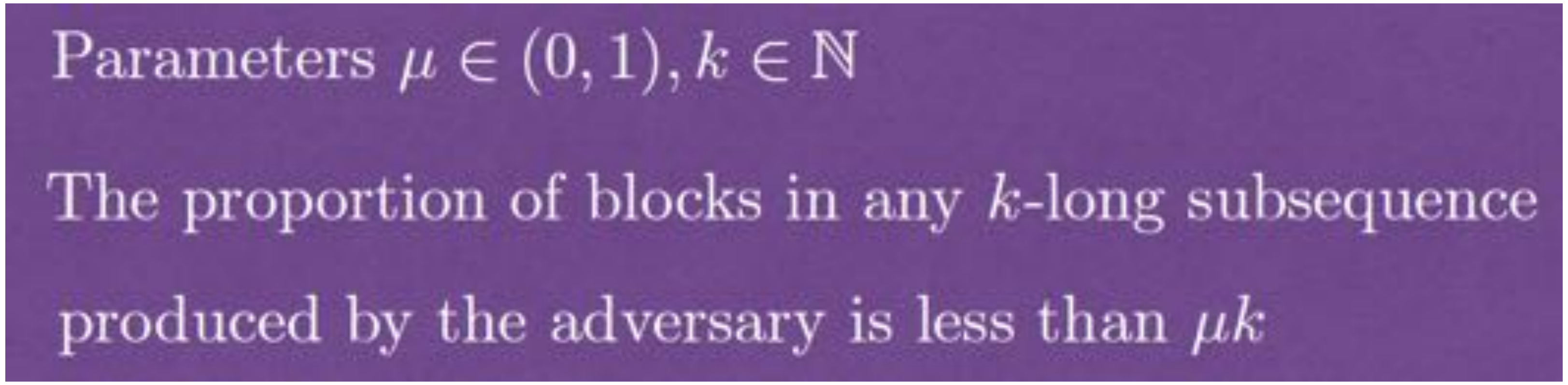


- For every *uniquely successful round* in S there should be an adversarial block

  → Uniquely successful rounds ≥ Adversarial successes

Chain Quality Property

- Will honest blocks be adopted by the honest parties?



Parameters $\mu \in (0,1), k \in \mathbb{N}$

The proportion of blocks in any k-long subsequence produced by the adversary is less than μk

- Recall: $n - t > \lambda(1+\delta)\, t$

- **Theorem:** In a typical execution, the Chain Quality property holds with parameter $\mu = 1/\lambda$

Chain Quality Property (2)

- The theorem is tight
  - There is an adversarial strategy that restricts the honest parties to a ratio of exactly $1- 1/\lambda$
  - The strategy is a type of *selfish mining* strategy [ES14]
    - Malicious miners mine blocks in private attempting to “hit” honest parties’ blocks when they become available

Chain Quality Property (3)

- *Ideal* chain quality: A set of parties with hashing power α may control up to αL blocks in a blockchain of length L

- Our chain quality bound is much more pessimistic: As $\lambda \rightarrow 1$ the adversary can control almost all the blocks

- Selfish mining implies that this is tight… Bitcoin's rewarding mechanism is *not* incentive-compatible

Applications

Applications of the Bitcoin Backbone Protocol

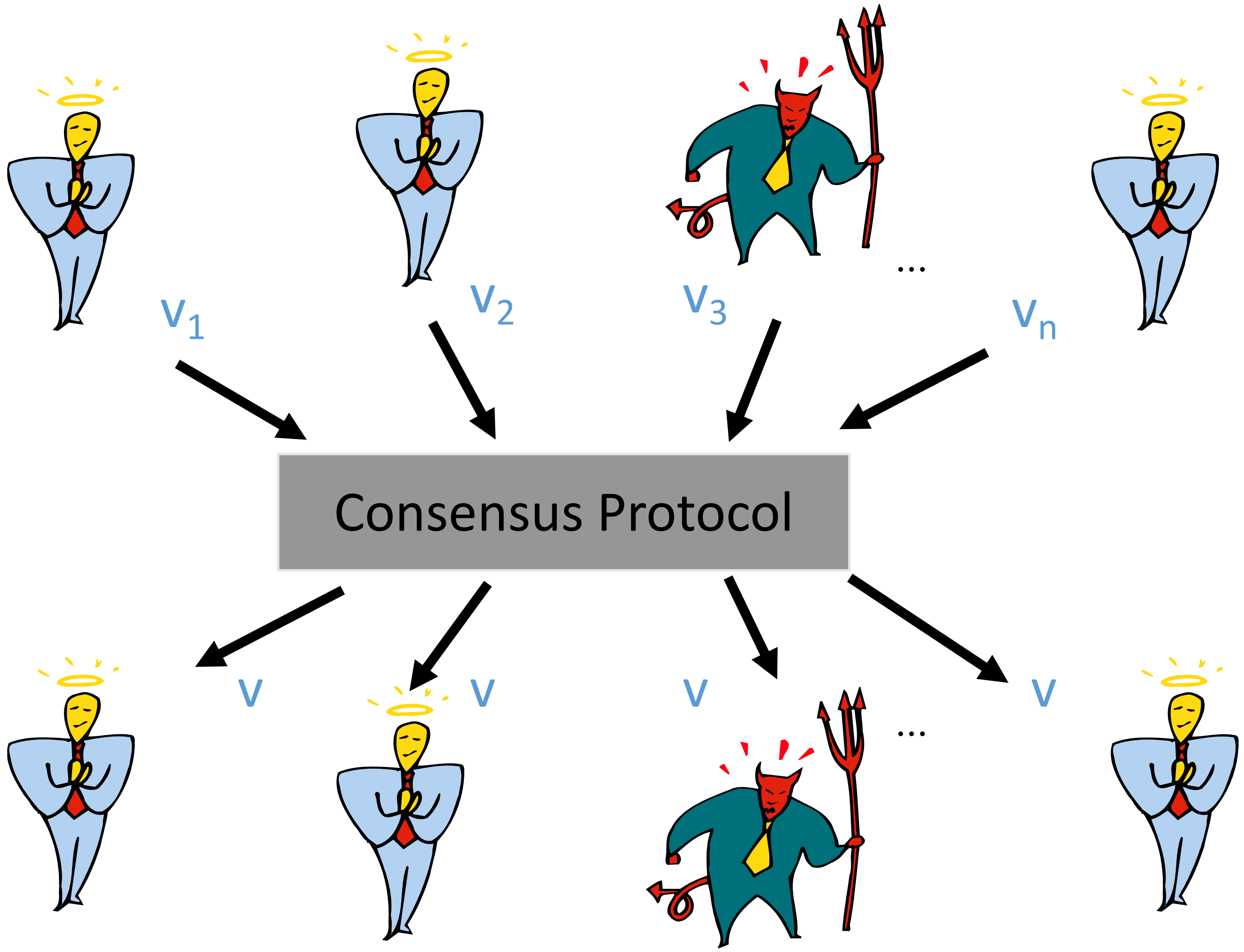
- Consensus
- Robust transaction ledger (Bitcoin)
  - Aka “ledger consensus,” “Nakamoto consensus”

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

- Well-studied problem in distributed systems, fault tolerance and cryptographic protocols

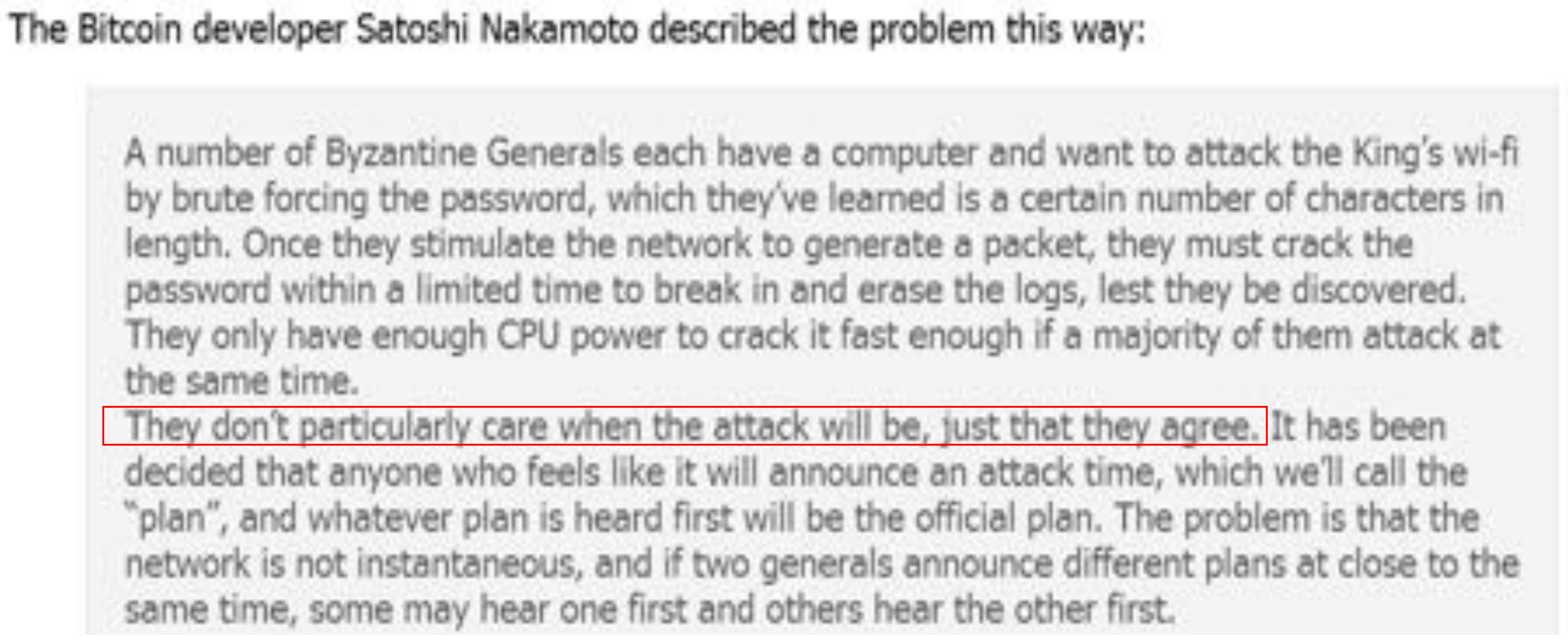
On the Necessity of an Honest Majority

$$t < n/2$$

regardless of the resources available to the parties

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

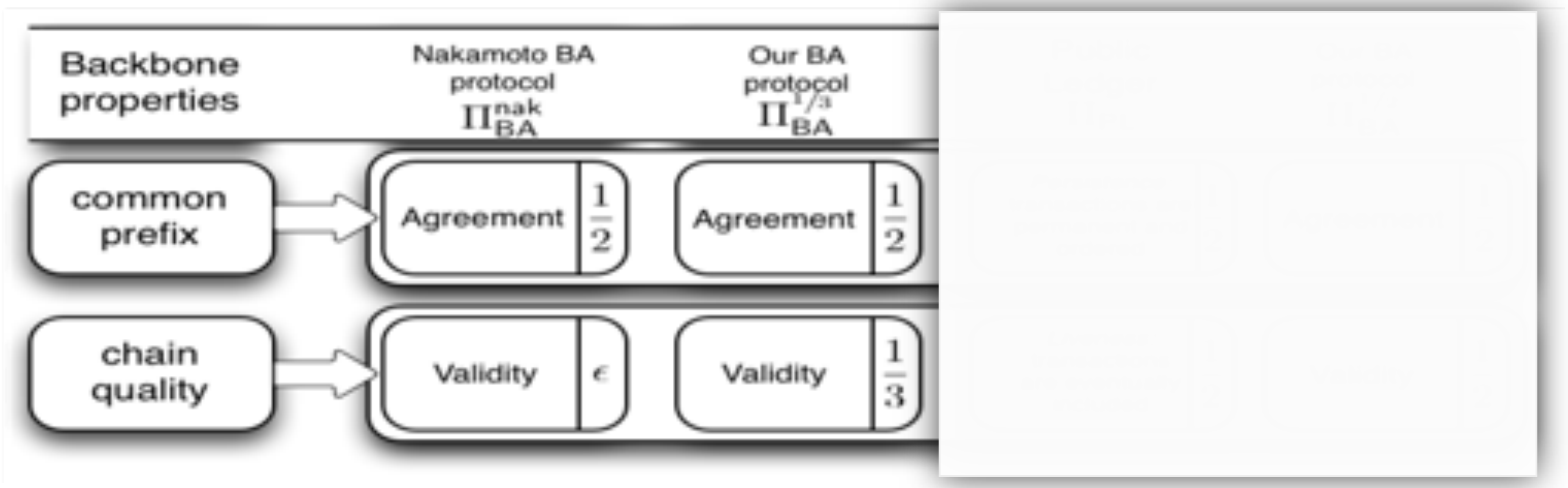
Nakamoto's Consensus Protocol [Nak08b] (3)

- 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



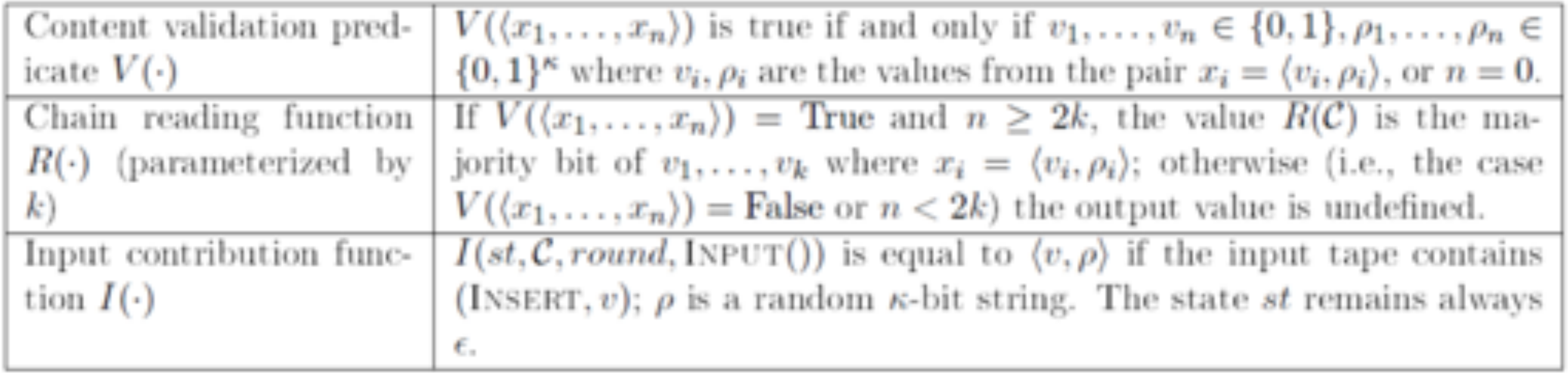
Our First Consensus Protocol

- 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)

1/3 Consensus Protocol (2)

- Specification of V, I, R



Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_n \rangle)$ is true if and only if $v_1, \ldots, v_n \in \{0,1\}, \rho_1, \ldots, \rho_n \in \{0,1\}^\kappa$ where v_i, ρ_i are the values from the pair $x_i = \langle v_i, \rho_i \rangle$, or $n = 0$.
Chain reading function $R(\cdot)$ (parameterized by k)	If $V(\langle x_1, \ldots, x_n \rangle) =$ **True** and $n \geq 2k$, the value $R(\mathcal{C})$ is the majority bit of $v_1, \ldots, v_k$ where $x_i = \langle v_i, \rho_i \rangle$; otherwise (i.e., the case $V(\langle x_1, \ldots, x_n \rangle) =$ **False** or $n < 2k$) the output value is undefined.
Input contribution function $I(\cdot)$	$I(st, \mathcal{C}, round, \text{INPUT}())$ is equal to $\langle v, \rho \rangle$ if the input tape contains (INSERT, v); ρ is a random κ-bit string. The state st remains always ϵ.

1/3 Consensus Protocol (3)

Lemma: Under the Honest Majority Assumption, and assuming $k = O(\kappa)$ and $L = \Omega(\kappa)$, 1/3-consensus protocol satisfies Agreement with probability $1 - e^{-\Omega(\kappa)}$

Proof: It suffices to show that the chains of honest parties begin with the same k blocks. In a typical execution, by Chain Growth, after L rounds, the chain of any honest party has more than 2k blocks.

Disagreement in the first k blocks between the chains of any two honest parties implies that $C_1^{\lceil k} \leq C_2$ does not hold, in violation of the Common Prefix property.

$\boxed{\text{QED}}$

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
- 1/2 can be achieved, using a more elaborate protocol – a technique we call *2-for-1 PoWs*
- **Q:** Is the blockchain-based approach is radically different from the traditional probabilistic protocols?

1/2 Consensus Protocol

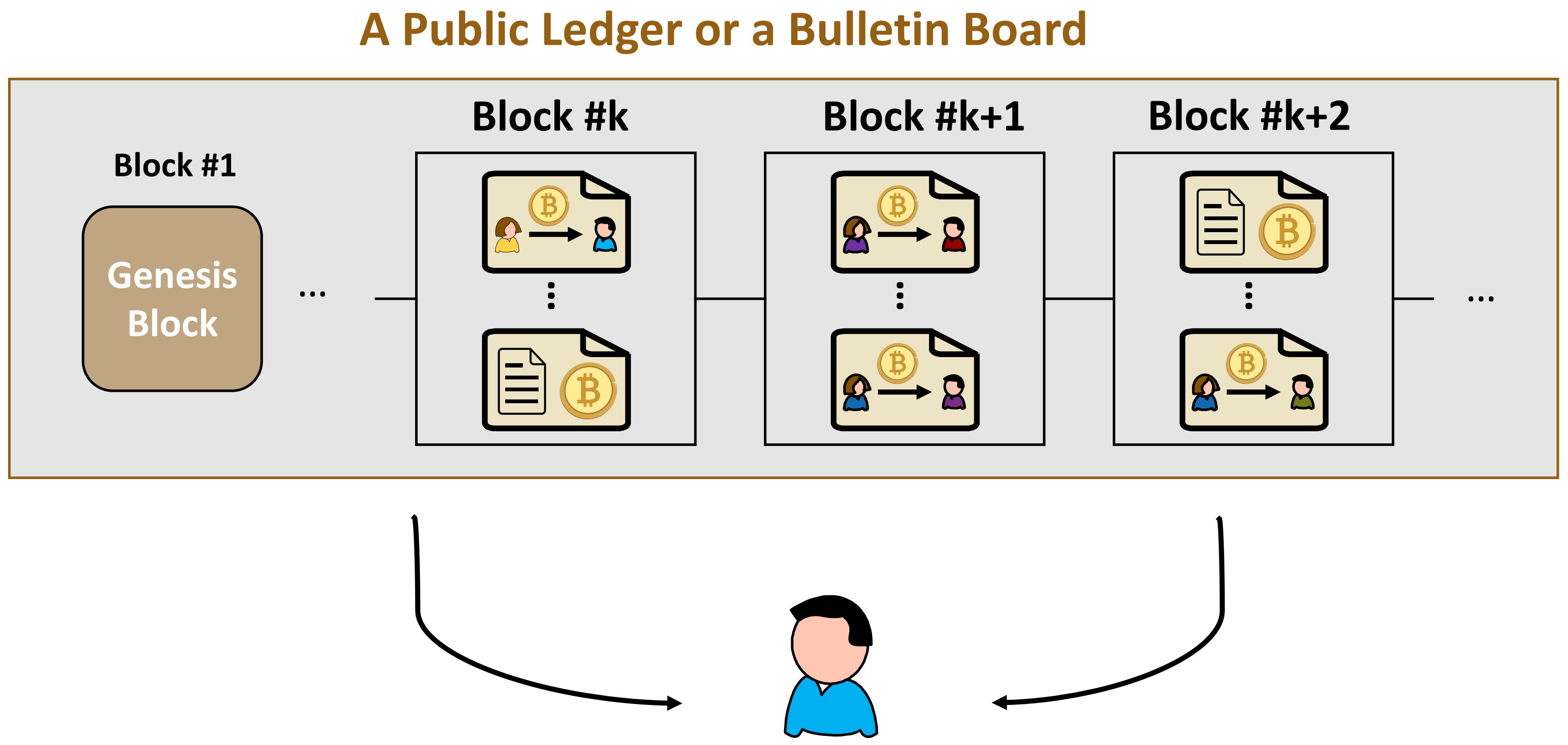
- **Idea:** Use POWs to “mine” transactions, in the same way blocks are mined
- Should guarantee that *number of transactions* is proportional to the hashing power of each party
- Mining:
  - At block level
  - At transaction level

Applications of the Blockchain

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

Blockchain Abstractions

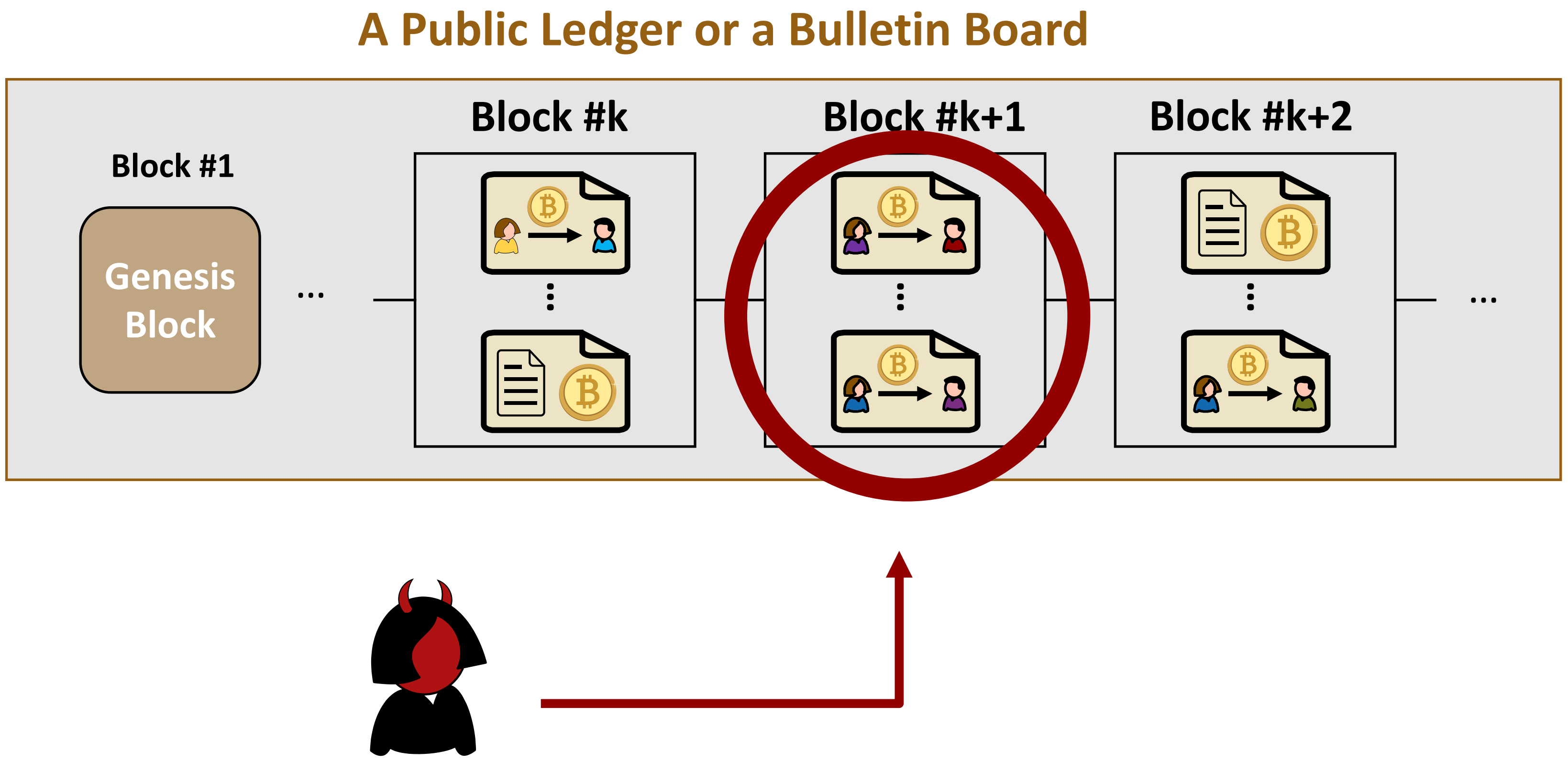
RECALL



2) Anyone can read the current content of the ledger

Blockchain Abstractions

RECALL



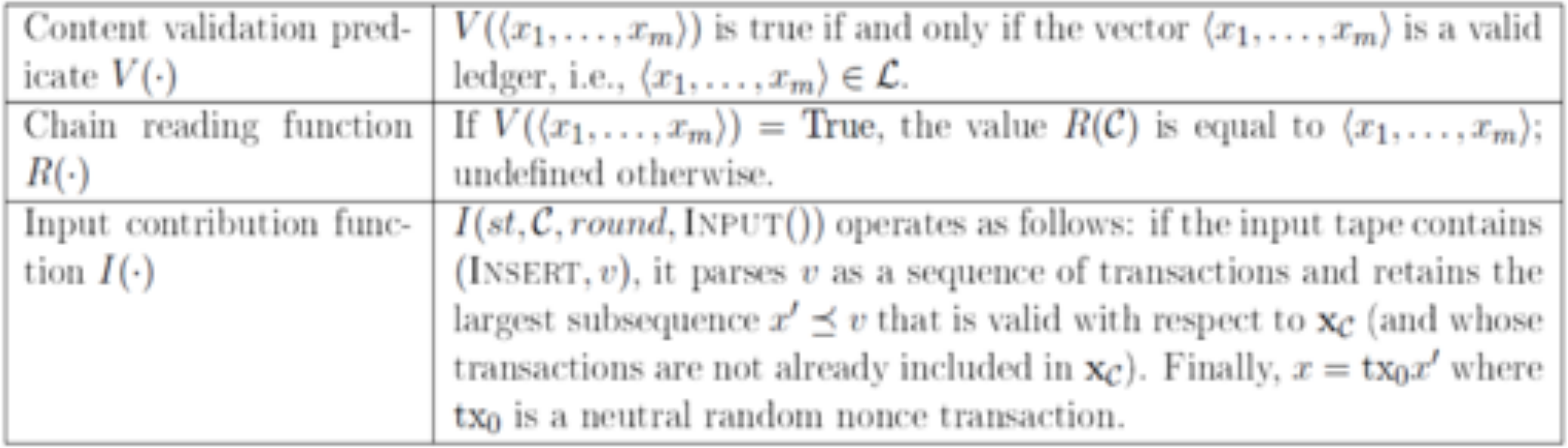
3) No modifications of contained blocks possible

Robust Public Transaction Ledgers: Properties

- **Consistency (Persistence):** Once an honest party ("miner") reports a transaction "deep enough" in the ledger, then all honest parties will report it indefinitely, and at exactly the same position
  - E.g., in Bitcoin, this property implies that credits are final, and that they happened at a certain "time"
  - Consistency is not enough to ensure ledger makes progress
- **Liveness:** Transactions are eventually inserted into the ledger
  - Assuming environment keeps providing them as input for a sufficient number of rounds

The Public Transaction Ledger Protocol

- Specification of V, I, R



Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_m \rangle)$ is true if and only if the vector $\langle x_1, \ldots, x_m \rangle$ is a valid ledger, i.e., $\langle x_1, \ldots, x_m \rangle \in \mathcal{L}$.
Chain reading function $R(\cdot)$	If $V(\langle x_1, \ldots, x_m \rangle) = \text{True}$, the value $R(\mathcal{C})$ is equal to $\langle x_1, \ldots, x_m \rangle$; undefined otherwise.
Input contribution function $I(\cdot)$	$I(st, \mathcal{C}, round, \textsc{Input}())$ operates as follows: if the input tape contains $(\textsc{Insert}, v)$, it parses v as a sequence of transactions and retains the largest subsequence $x' \preceq v$ that is valid with respect to $\mathbf{x}_{\mathcal{C}}$ (and whose transactions are not already included in $\mathbf{x}_{\mathcal{C}}$). Finally, $x = \mathsf{tx}_0 x'$ where tx_0 is a neutral random nonce transaction.

Bitcoin Transactions

- Transactions and accounts defined with respect to a digital signature scheme
    - Three algorithms: (KeyGen, Sign Verify)

α : Honest parties expected POW solutions in a round

$f = \alpha + \beta$

$\gamma = \alpha - \alpha^2$

Bitcoin Transactions

- Transactions and accounts defined with respect to a digital signature scheme
    - Three algorithms: (KeyGen, Sign Verify)
- *Account:* a = (vk, G(vk))
    - G(vk): account's *address* (G(·): hash function)

RECALL

Bitcoin Address/Account (Security)

- Based on elliptic curve *secp256k1*
- Account: (PrivKey, PubKey)

- Bitcoin address:

  Base58(RIPEMD160(SHA256(PubKey)))

  E.g., *37k7toV1Nv4DfmQbmZ8KuZDQCYK9x5KpzP*
- PrivKey used to sign outgoing transactions
- Wallet: many (PrivKey, PubKey)
- Transaction:

I, pay BC #13107 to

Bitcoin Transactions

- Transactions and accounts defined with respect to a digital signature scheme
  - Three algorithms: (KeyGen, Sign Verify)
- *Account:* a = (vk, G(vk))
  - G(vk): account's *address* (G(·): hash function)
- *Transaction:* $tx = \{a_1, a_2, \ldots, a_i\} \rightarrow (\sigma, \{(a'_1, b'_1), (a'_2, b'_2), \ldots (a'_o, b'_o)\})$
  - a_j's : accounts to be debited; a'_k's : accounts to be credited; b'_k's : funds
  - $\sigma = ((vk_1, \sigma_1), (vk_2, \sigma_2), \ldots, (vk_i, \sigma_i))$ – verification keys and signatures on message $\{(a'_1, b'_1), (a'_2, b'_2), \ldots (a'_o, b'_o)\}$

Bitcoin Transactions

- Transactions and accounts defined with respect to a digital signature scheme
  - Three algorithms: (KeyGen, Sign Verify)
- *Account:* $a = (vk, G(vk))$
  - $G(vk)$: account's *address* ($G(\cdot)$: hash function)
- *Transaction:* $tx = \{a_1, a_2,\ldots, a_i\} \rightarrow (\sigma, \{(a'_1,b'_1),(a'_2,b'_2),\ldots(a'_o,b'_o)\})$
  - a_j's : accounts to be debited; a'_k's : accounts to be credited; b'_k's : funds
  - $\sigma = ((vk_1, \sigma_1), (vk_2, \sigma_2),\ldots,(vk_i, \sigma_i))$ – verification keys and signatures on message $\{(a'_1,b'_1),(a'_2,b'_2),\ldots(a'_o,b'_o)\}$
- Entities maintain a number of accounts, and move their balances forward to a new account as they make transactions

Valid Bitcoin Transactions

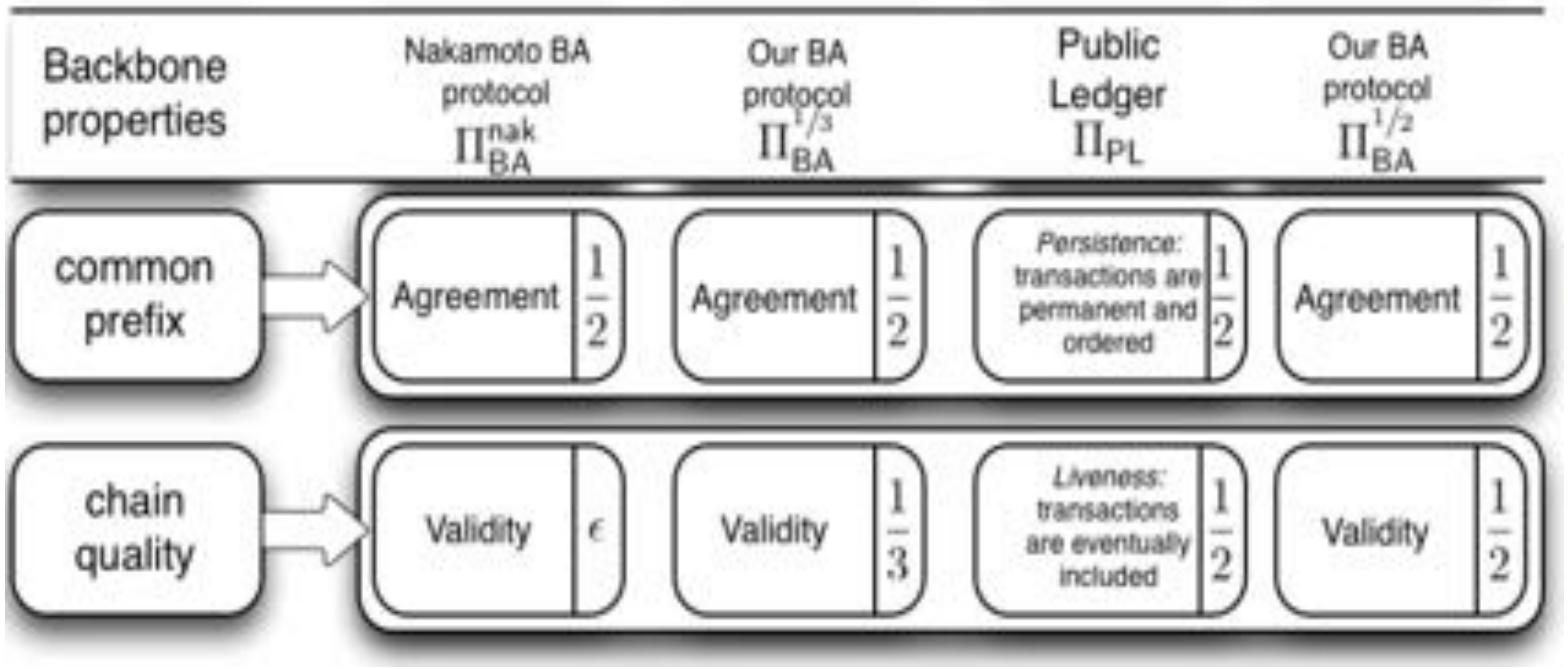
- A transaction tx is *valid* with respect to a Bitcoin ledger $\mathbf{x} = (x_1, x_2, \dots, x_m)$ if
  - All digital signatures verify, and
  - $\sum_{1\dots i} b_j \geq \sum_{1\dots o} b'_j$, where b_j is the balance credited to a_j in latest transaction involving a_j in $\mathbf{x}$

Robust Public Transaction Ledgers: Properties

- **Consistency (Persistence):** Once an honest party ("miner") reports a transaction "deep enough" in the ledger, then all honest parties will report it indefinitely, and at exactly the same position
    - E.g., in Bitcoin, Persistence implies that credits are final, and that they happened at a certain "time"
    - Consistency is not enough to ensure ledger makes progress
- **Liveness:** Transactions are eventually inserted into the ledger
    - Assuming environment keeps providing them as input for a sufficient number of rounds

- Common Prefix → Consistency, Chain Quality + Chain Growth → Liveness

Summary of Applications (2)



Backbone properties	Nakamoto BA protocol $\Pi_{\mathsf{BA}}^{\mathsf{nak}}$	Our BA protocol $\Pi_{\mathsf{BA}}^{1/3}$	Public Ledger Π_{PL}	Our BA protocol $\Pi_{\mathsf{BA}}^{1/2}$
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 $\frac{1}{2}$

Conclusions So Far

- Rigorous cryptographic analysis of the system
  - The Bitcoin *backbone* protocol
  - Bitcoin is secure iff the *majority* of the mining power is honest

Talk Plan

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
  - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
  - Common Prefix, Chain Quality, Chain Growth
- Applications
  - Consensus
  - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
- References

Network/Computational Model (2)

RECALL

- 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
    $\rightarrow$ $t \cdot q$ 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…

Is a Genesis Block Really Needed?

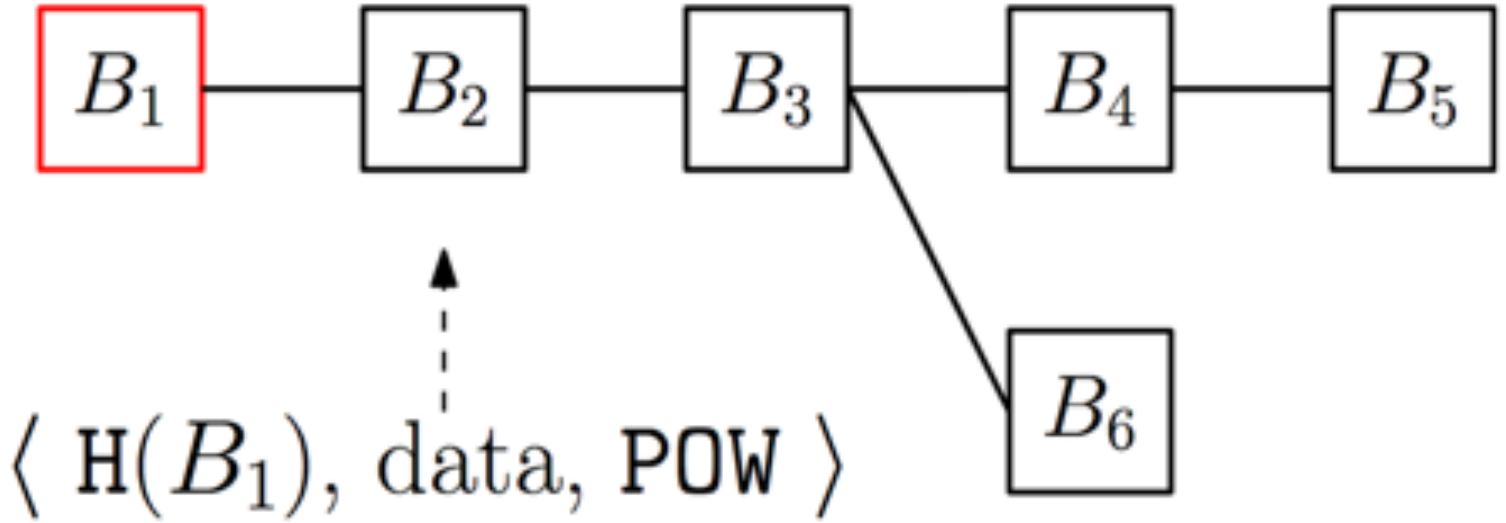
- "Genesis" block
  - First block of the blockchain; must be unpredictable
  - Hardcoded into the software
  - Coinbase parameter contains the following text:

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

THE TIMES
Eat Out from £5
Israel prepares to send tanks and troops into Gaza
Chancellor on brink of second bailout for banks

What Is a Blockchain?

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


B_1
B_2
B_3
B_4
B_5
B_6
$\langle\ \mathtt{H}(B_1),\ \text{data},\ \mathtt{POW}\ \rangle$

Network/Computational Model (2)

RECALL

- 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
    → $t \cdot q$ 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…

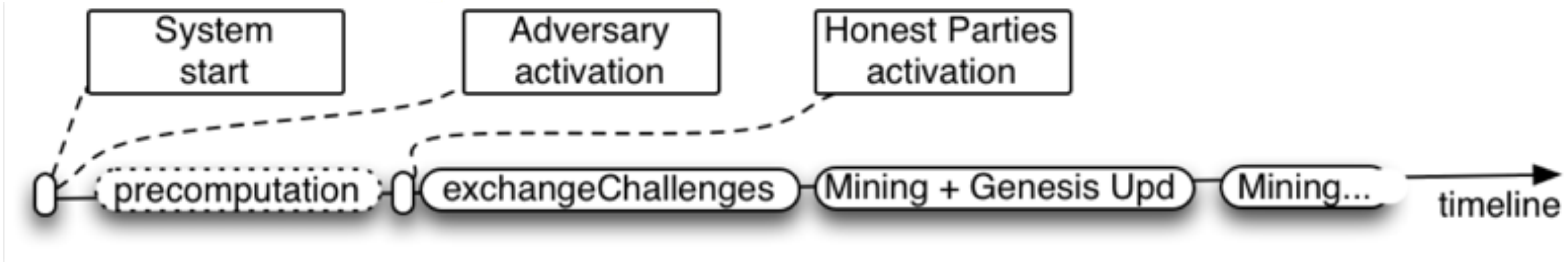
Not needed!

The *Bootstrapped* Backbone Protocol [GKLP18]

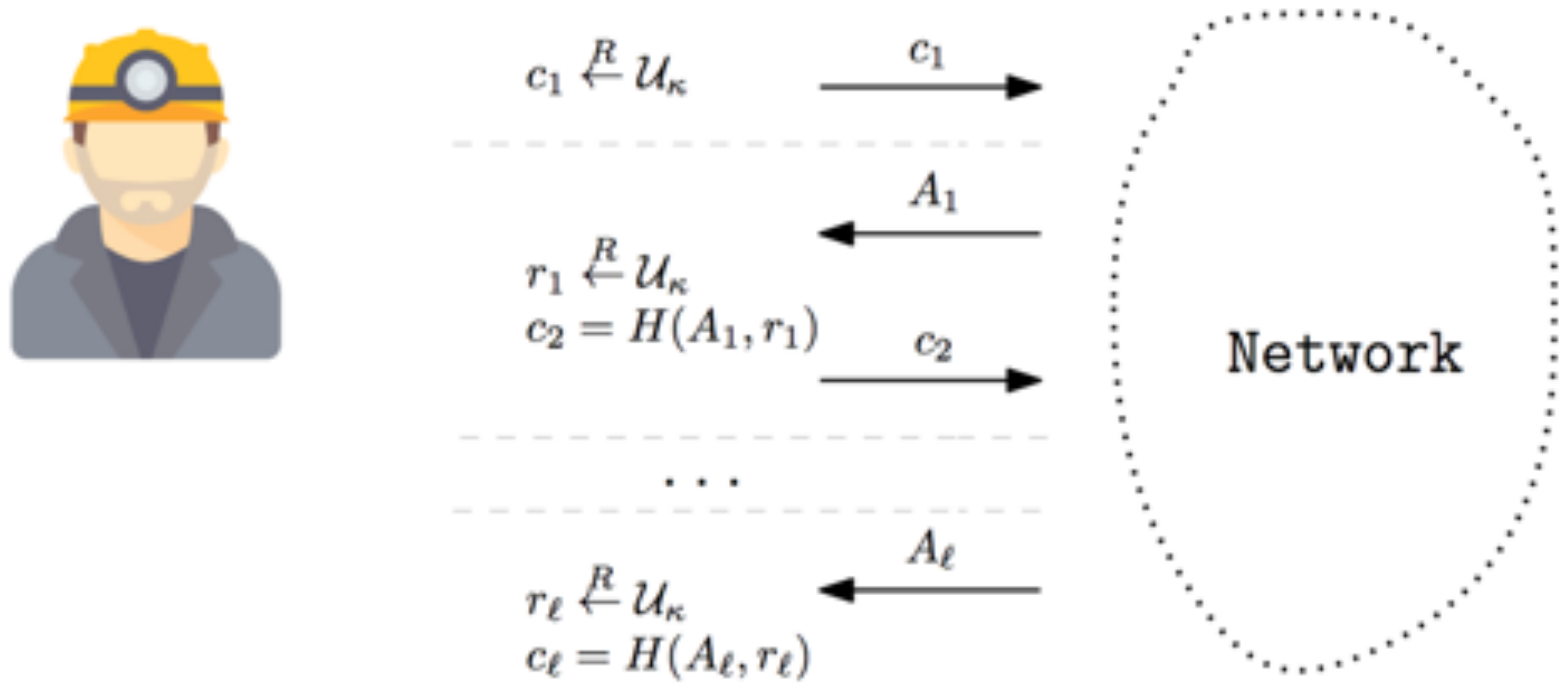
- *No trusted setup* and individual genesis block mining
- Freshness of genesis block impacts chains' total *weight*
- Personalized chain selection rule
- Robustness is achieved after an initial period of protocol stabilization

The *Bootstrapped* Backbone Protocol (2)

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



The *Bootstrapped* Backbone Protocol: Challenge Exch. Phase

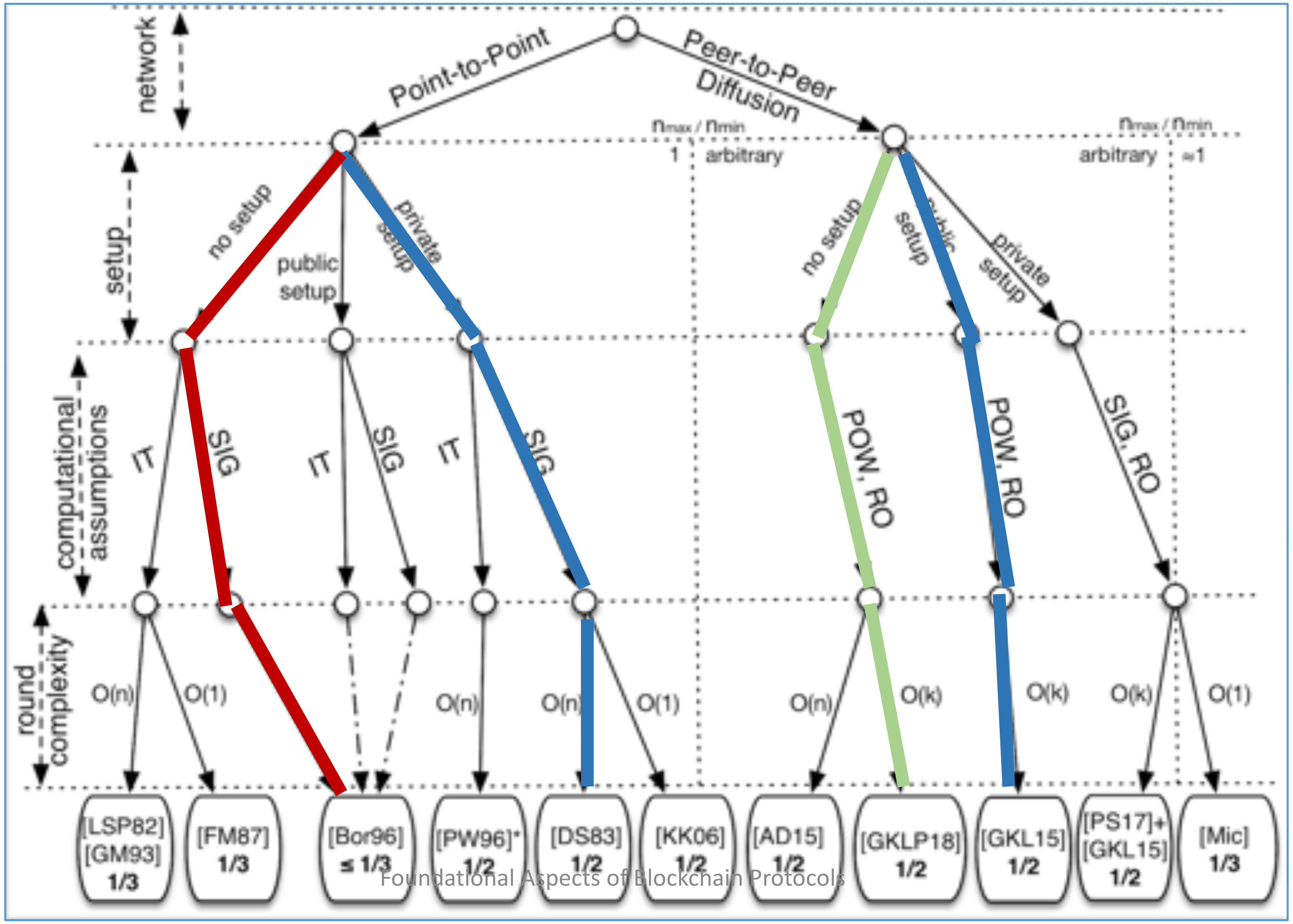

$c_1 \xleftarrow{R} \mathcal{U}_\kappa$
c_1
A_1
$r_1 \xleftarrow{R} \mathcal{U}_\kappa$
$c_2 = H(A_1, r_1)$
c_2
Network
$\dots$
A_ℓ
$r_\ell \xleftarrow{R} \mathcal{U}_\kappa$
$c_\ell = H(A_\ell, r_\ell)$

- Each honest party can generate a verifiable unpredictable string
- Disagreement is at most one round

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
- Consensus and ledger consensus can be solved directly without a trusted setup assuming an honest majority, based on POWs
- Other app's: **PKI setup from scratch**

A Consensus Taxonomy [GK18]



Talk Plan

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
  - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
  - Common Prefix, Chain Quality, Chain Growth
- Applications
  - Consensus
  - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
- References

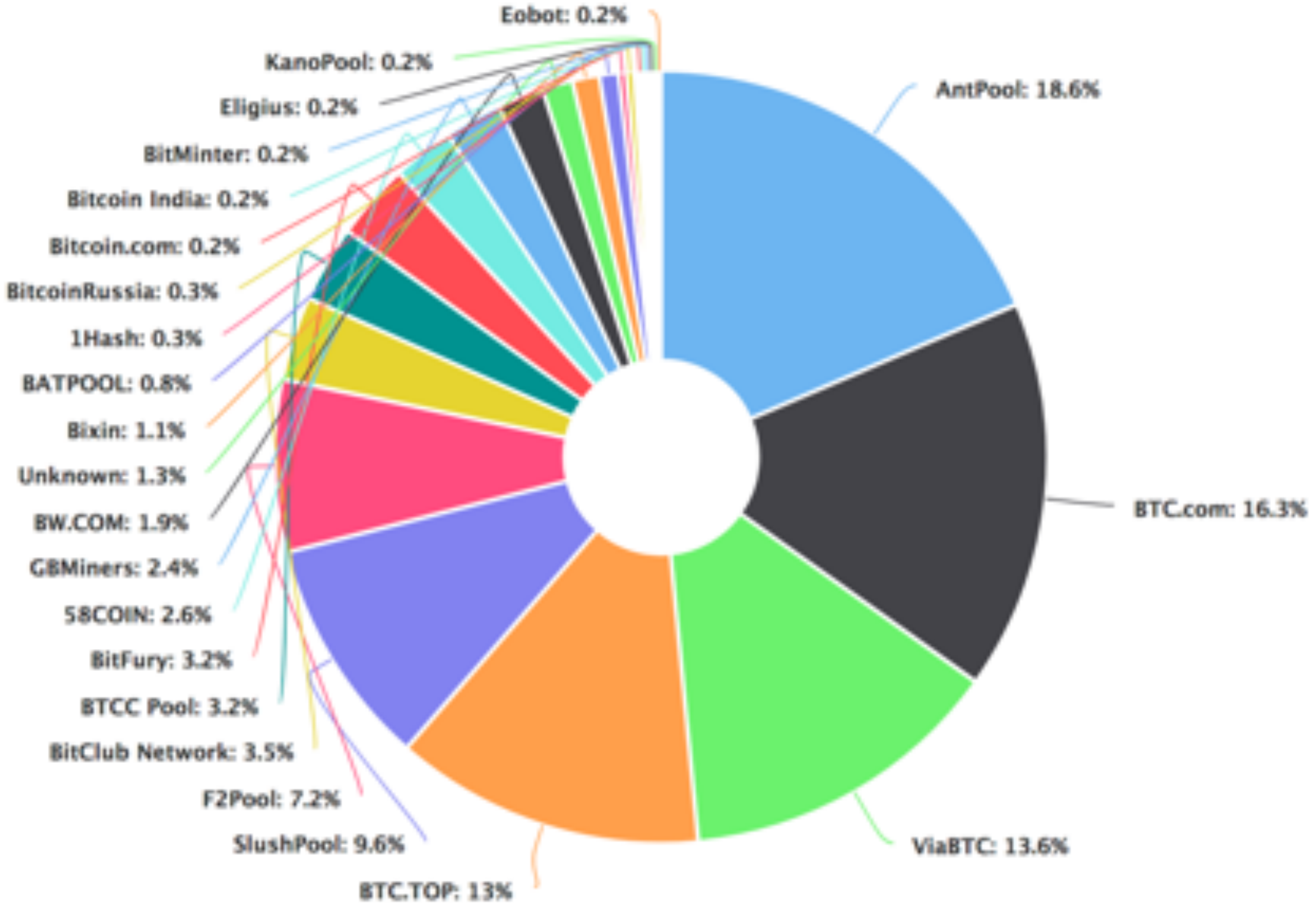
Blockchain of Variable Difficulty [GKL17]

- Bitcoin uses a *target recalculation mechanism*, adjusting POW hardness to accommodate dynamic population of users
- Parties come and go
- Number of parties can increase and decrease, but subject to a constraint
  - $\forall S, |S| = s, \ \max_{r \in S} n_r \leq \gamma \cdot \min_{r \in S} n_r$
- Importance of f ("block production rate")
  - In Bitcoin backbone: $f = qnT/2^{\kappa}$
- In a dynamic environment, recalculate target T to keep f constant
  - $f(n,T) \cong f(n_0,T_0) = f_0$
- We prove security under the assumption that the number of parties won't fluctuate wildly

But Why Does It Work? [BGMTZ18]

- We just saw a rigorous cryptographic analysis of the system
  - The Bitcoin *backbone* protocol
  - Bitcoin is secure iff the *majority* of the mining power is honest

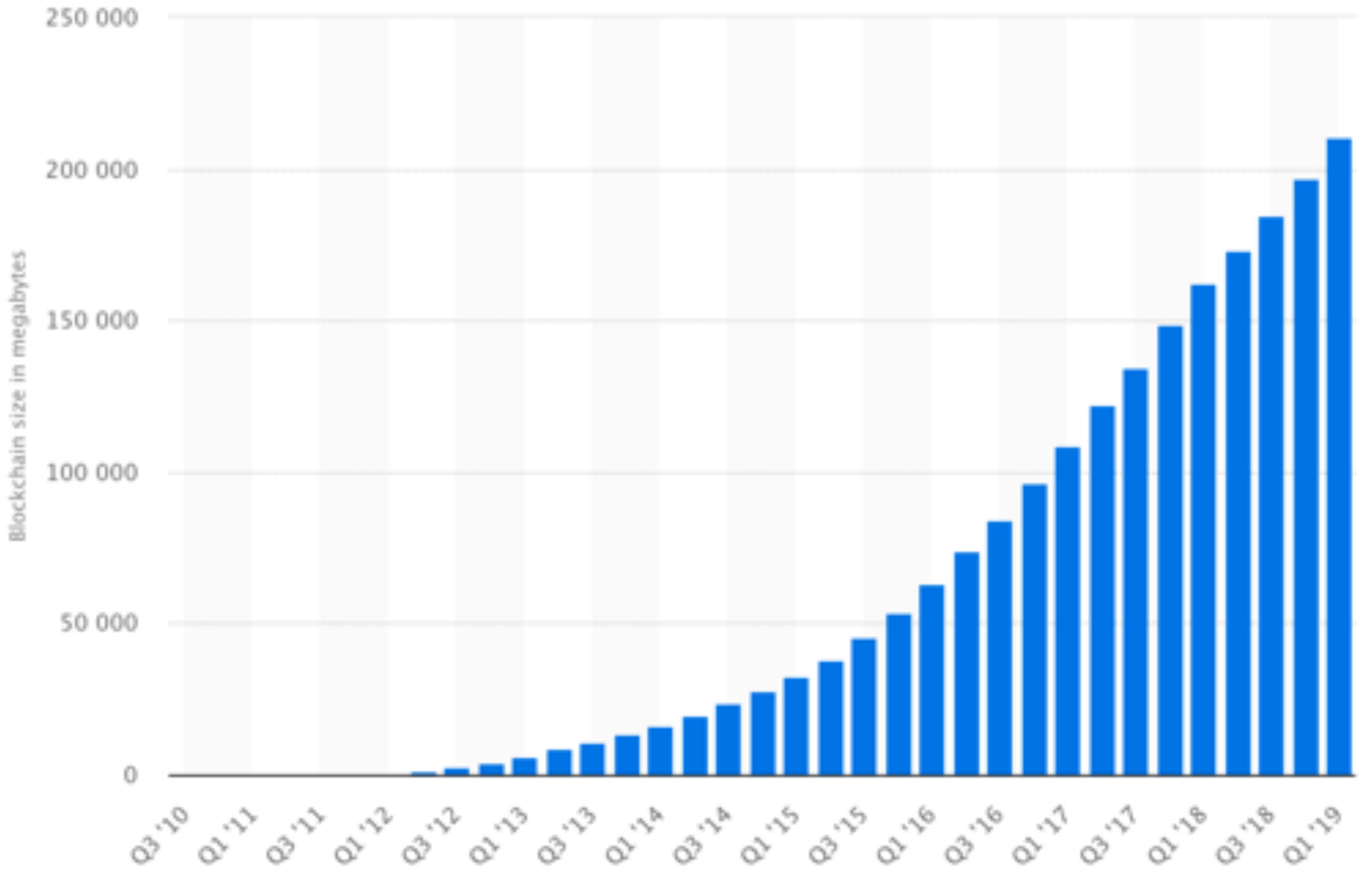
But Why Does It Work? [BGMTZ18]



But Why Does It Work? [BGMTZ18]

- We just saw a rigorous cryptographic analysis of the system
  - The Bitcoin *backbone* protocol
  - Bitcoin is secure iff the *majority* of the mining power is honest
- Bitcoin keeps performing according to spec's, even though mining pools would be able to launch collaborative attacks given the power they control
  - Q_1: How come Bitcoin is not broken using such an attack?
  - Q_2: Why do honest miners keep mining given the plausibility of such attacks?
- We devised a "rational cryptography" framework for capturing the economic forces underlying the tension between honest miners and deviating miners
  - Natural incentives (expected revenue of miners) + Bitcoin's high monetary value can explain that Bitcoin is not attacked in reality
  - *Even though* majority coalitions are in fact possible

Size of the Bitcoin Blockchain (2010—2019, MB)



Improved Communication Proposals

Sublinear SPV (Simple Payment Verification) clients

- SPV: Allows confirmation that transaction has been added to the blockchain, without having to download entire blockchain. E.g., “Transaction tx was in block k”
  - Throw away transactions, keep block headers
  - Bitcoin: 80 × # blocks (~50 MB); Ethereum: 3 GB
- *(NI)PoPoW* (Non-Interactive Proofs of Proof-of-Work) [KMZ17]
  - Optimistically succinct proofs, variety of predicates
  - Additional data structure in block header (skip list)
- *Flyclient* [LKZ17]
  - Additional data structure: Instead of link (hash) to previous block, root of Merkle tree commits to all the nodes
  - Proof that block was part of the chain + proof that tx was part of the block

Improved throughput and confirmation latency [BKTFV18]

- Approach network capacity and propagation delay

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, G. Panagiotakos and N. Leonardos,“Bootstrapping the Blockchain, with Applications to Consensus and Faster PKI Setup.” *PKC 2018.* Full version at Cryptology ePrint Archive: Report 2016/991, http://eprint.iacr.org/2016/991
- J. Garay, A. Kiayias and N. Leonardos, “The Bitcoin Backbone with Chains of Variable Difficulty.” *Crypto 2017.* Full version at Cryptology ePrint Archive: Report 2016/1048, http://eprint.iacr.org/2016/1048
- C. Badertscher, J. Garay, U. Maurer, D. Tschudi and V. Zikas, “But Why Does It Work? A *Rational Protocol Design* Treatment of Bitcoin.” *Eurocrypt 2018.* Full version at Cryptology ePrint Archive: Report 2018/138, https://eprint.iacr.org/2018/138
- J. Garay and A. Kiayias,“A Consensus Taxonomy in the Blockchain Era.” Full version at Cryptology ePrint Archive: Report 2018/754, http://eprint.iacr.org/2018/754. YouTube presentation: https://www.youtube.com/watch?v=mwmFPHnR1Pw

Thank You