

Foundational Aspects of Blockchain Protocols

Juan A. Garay

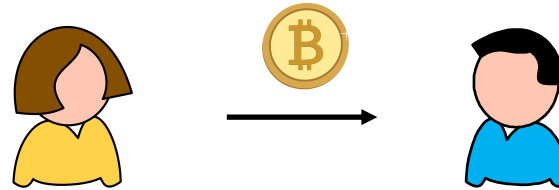
Texas A&M University

garay@cse.tamu.edu

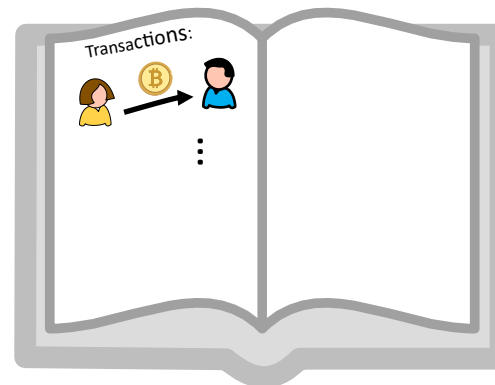
Based on joint work with Aggelos Kiayias (U. of Edinburgh), Nikos Leonardos (U. of Athens) and Giorgos Panagiotakos (U. of Edinburgh)

Motivation

1) The problem: digital money transfer

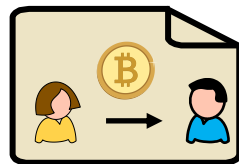
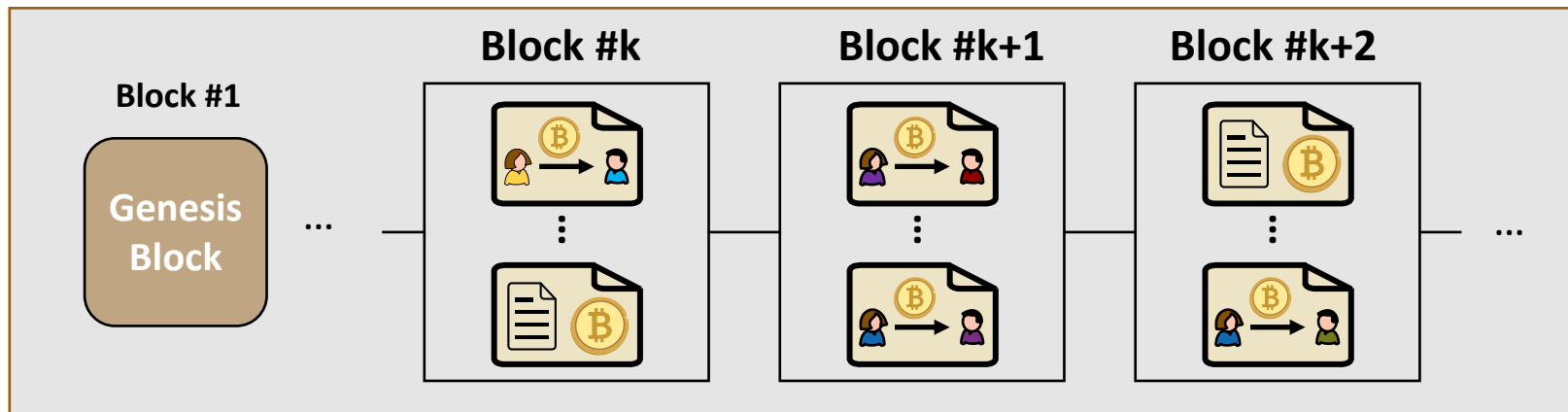


2) The accounting: implement a ledger!



Blockchain Abstractions

A Public Ledger or a Bulletin Board



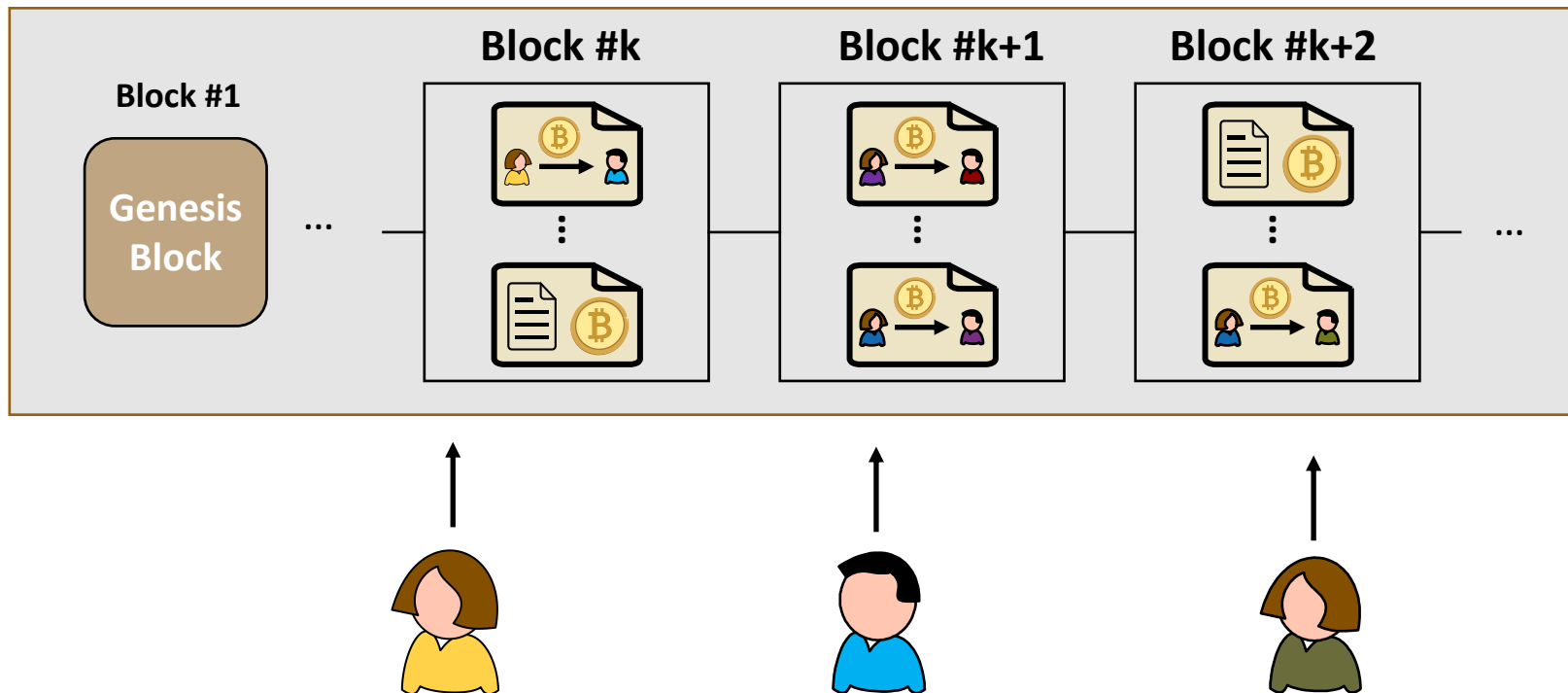
Simple transactions



Complex contracts

Blockchain Abstractions

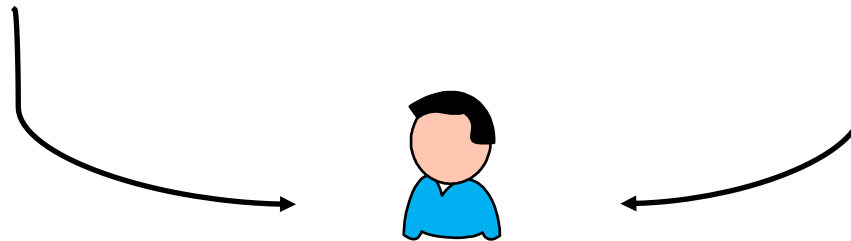
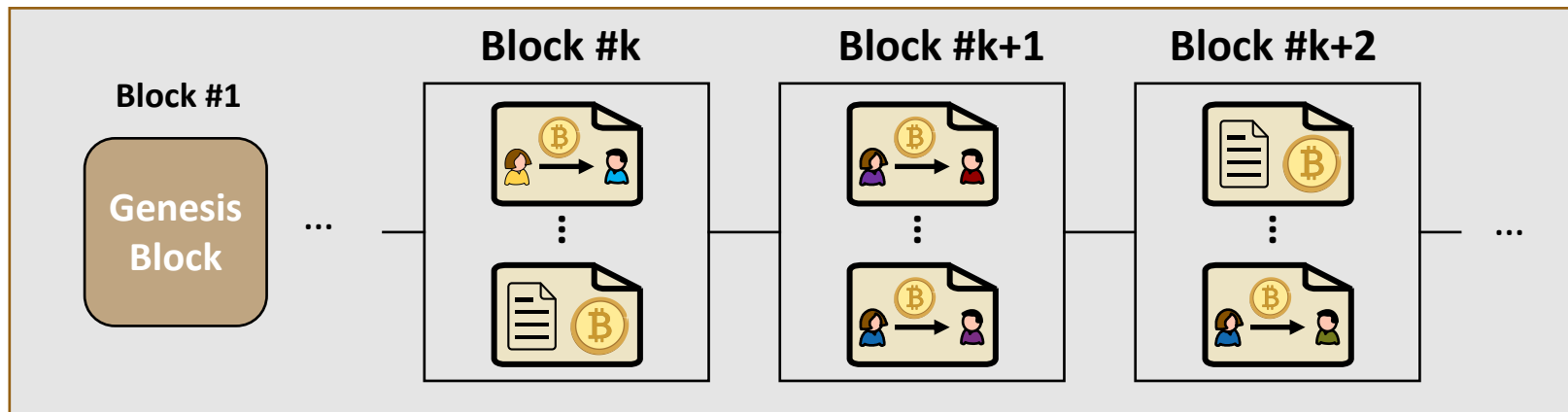
A Public Ledger or a Bulletin Board



1) Content is provided by any user with sufficient funds

Blockchain Abstractions

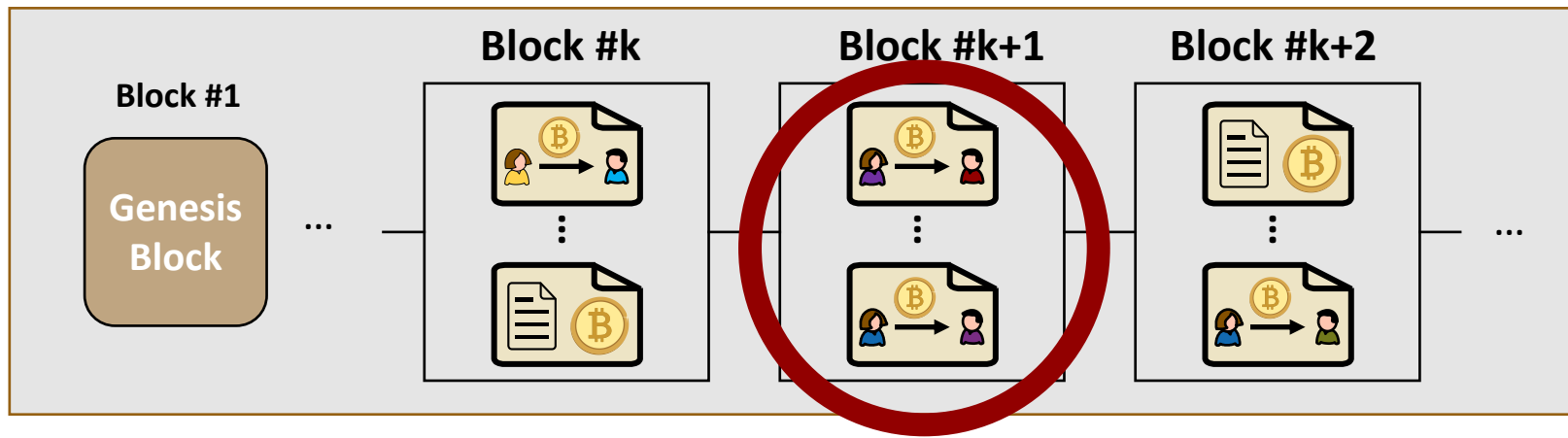
A Public Ledger or a Bulletin Board



2) Anyone can read the current content of the ledger

Blockchain Abstractions

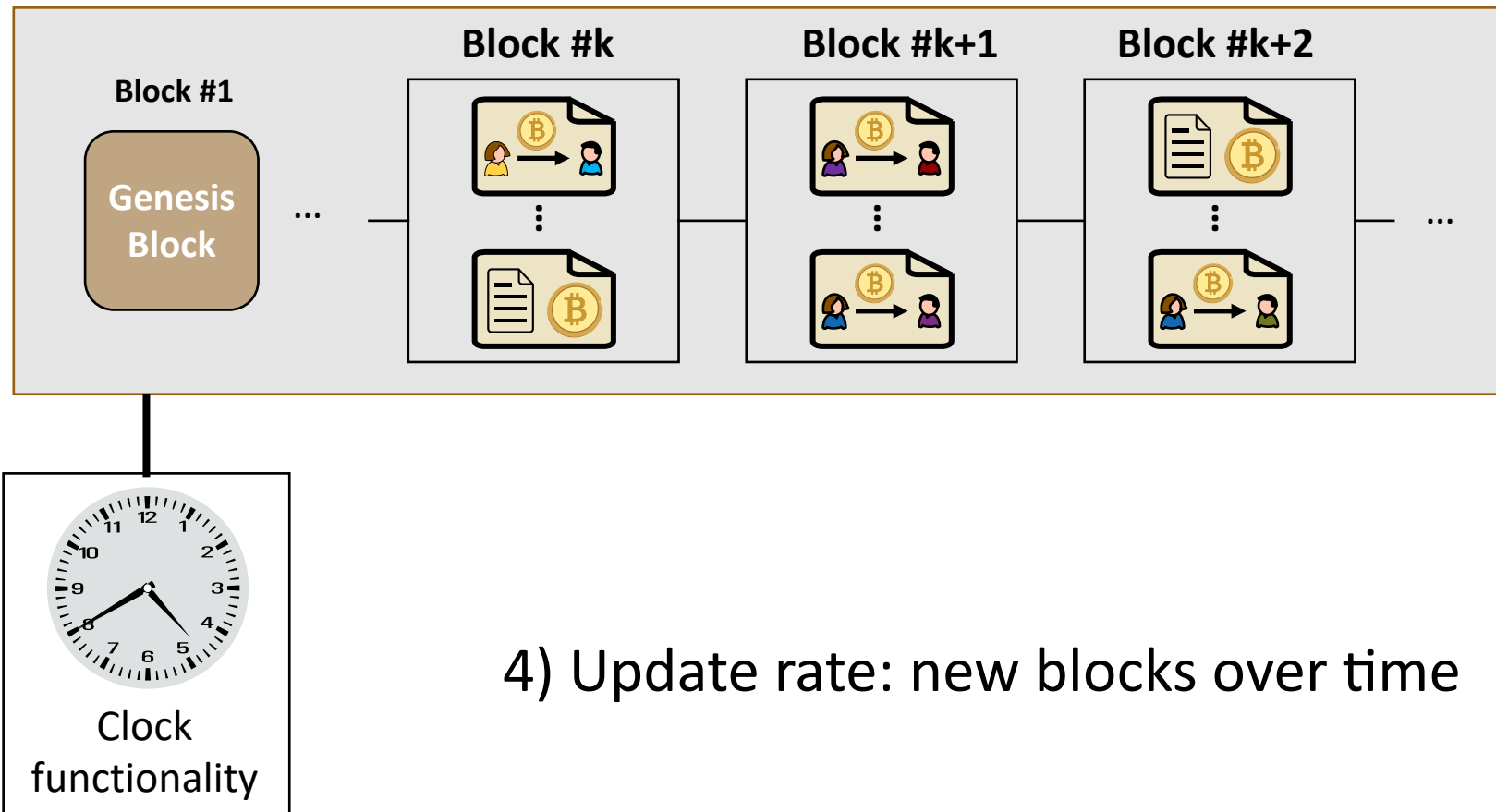
A Public Ledger or a Bulletin Board



3) No modifications of contained blocks possible

Blockchain Abstractions

A Public Ledger or a Bulletin Board



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:
 - i. fees
 - ii. 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:

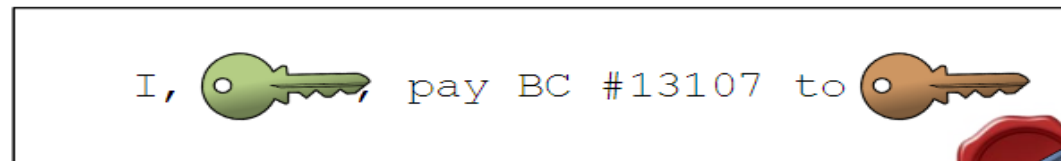
Base58(RIPEMD160(SHA256(PubKey)))

E.g., *37k7toV1Nv4DfmQbmZ8KuZDQCYK9x5KpzP*

- PrivKey used to sign outgoing transactions

- Wallet: many (PrivKey, PubKey)

- Transaction:



Slide credit: Marc Stevens

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.

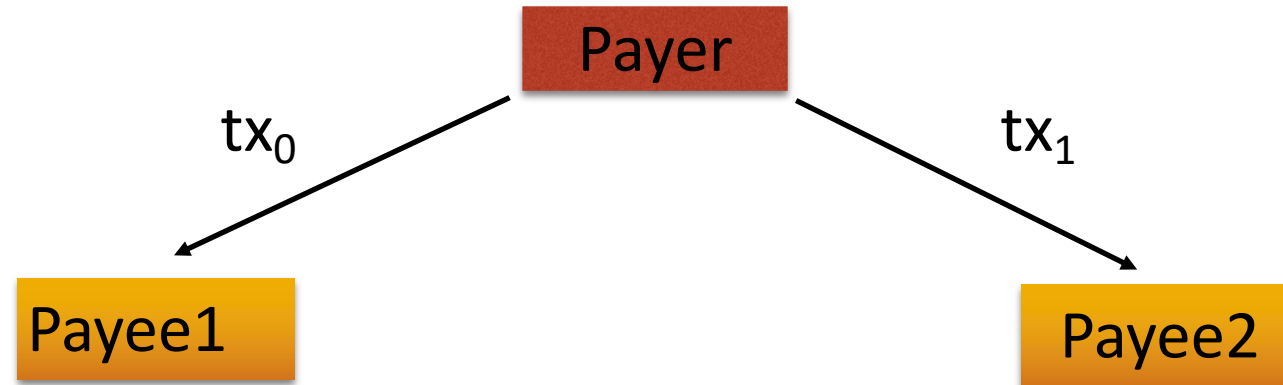
Bob now wants to send 0.4 BTC to Charles. The transaction output from Alice's transaction is now used to fund this new transaction. The transaction creates two new outputs: One with 0.4 BTC that is associated with Charles' address, and one with 0.6 BTC associated with Bob's address (it is the change). 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”
$$\text{Valid}(\tau, tx) \in \{\text{True}, \text{False}\}$$
- 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

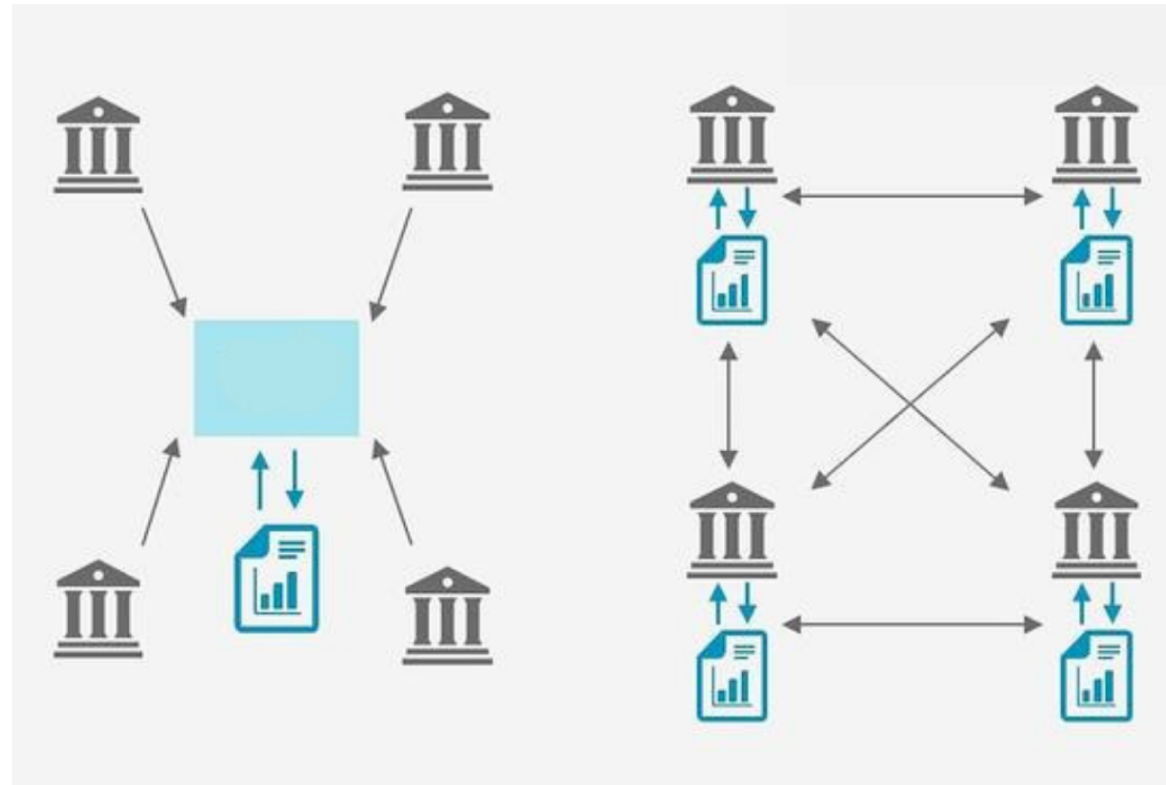


Figure: <http://blogs.wsj.com/cio/2016/02/02/cio-explainer-what-is-blockchain/>

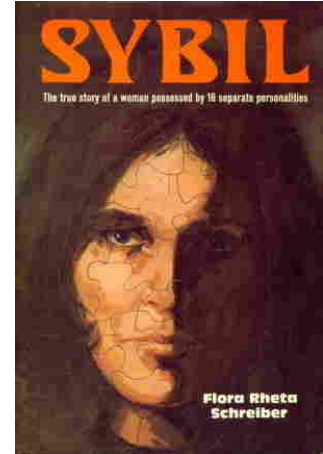
- **Consistency:** Everyone sees the same history
- **Liveness:** Everyone can add new transactions

The “Permissionless” Model [Nak08]

- Strong impossibility results w/o authentication [Oku05, BCLPR05]
- *Sybil* attacks are unavoidable [Dou02,...]
- $\frac{1}{3}$ 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* of computational power
 - **Consistency**: Everyone sees the same history
 - **Liveness**: Everyone can add new transactions



Talk Plan

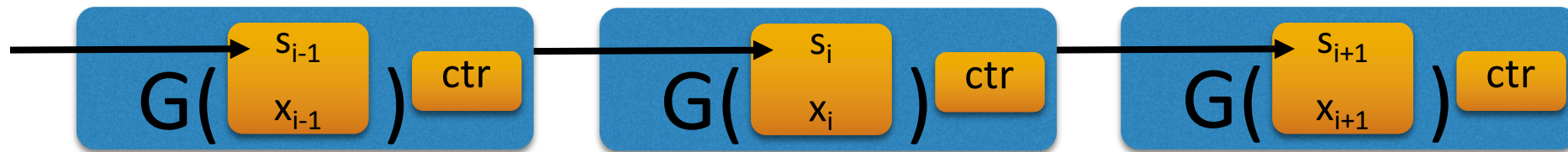
Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - PoS-based blockchains
- References

What Is a Blockchain?



What Is a Blockchain?

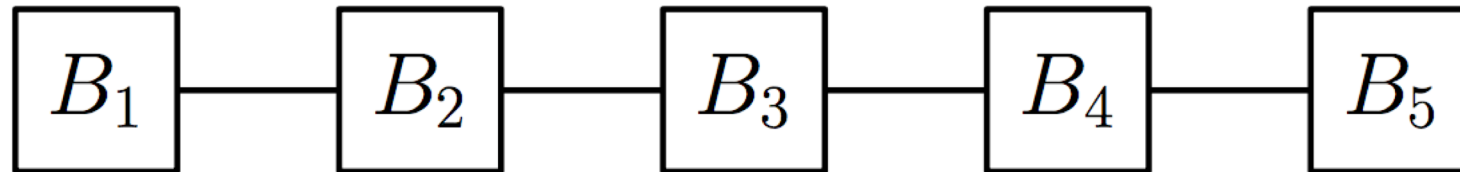
- Parties (“miners”) have to do work in order to install a transaction

What Is a Blockchain?

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

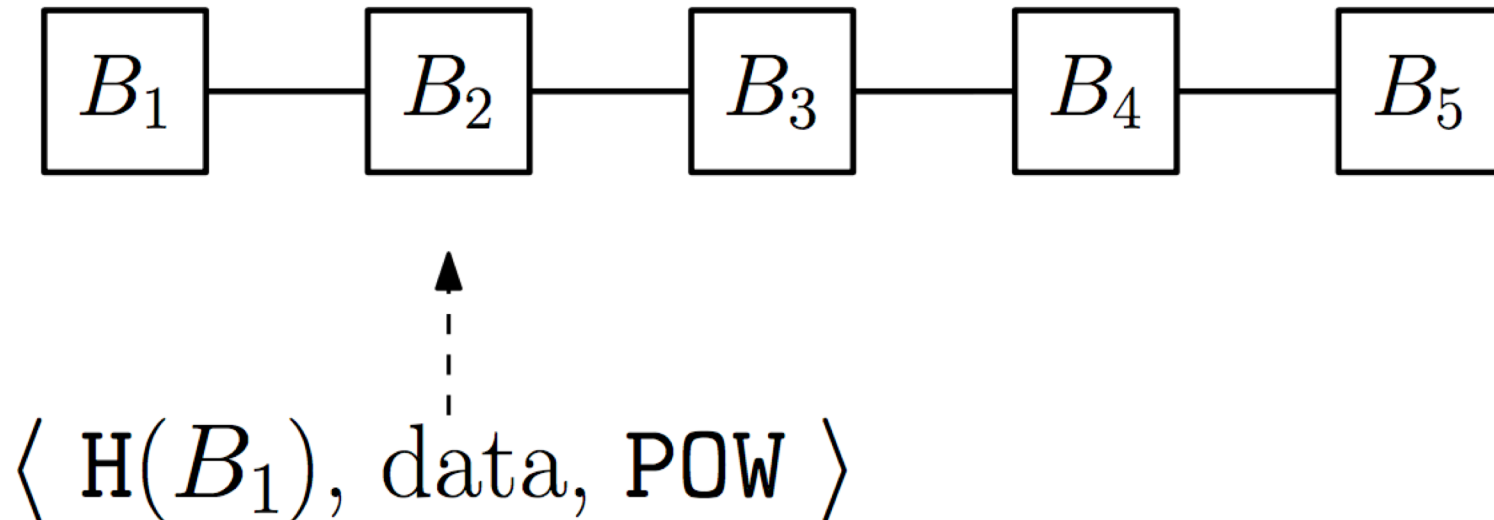
What Is a Blockchain?

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



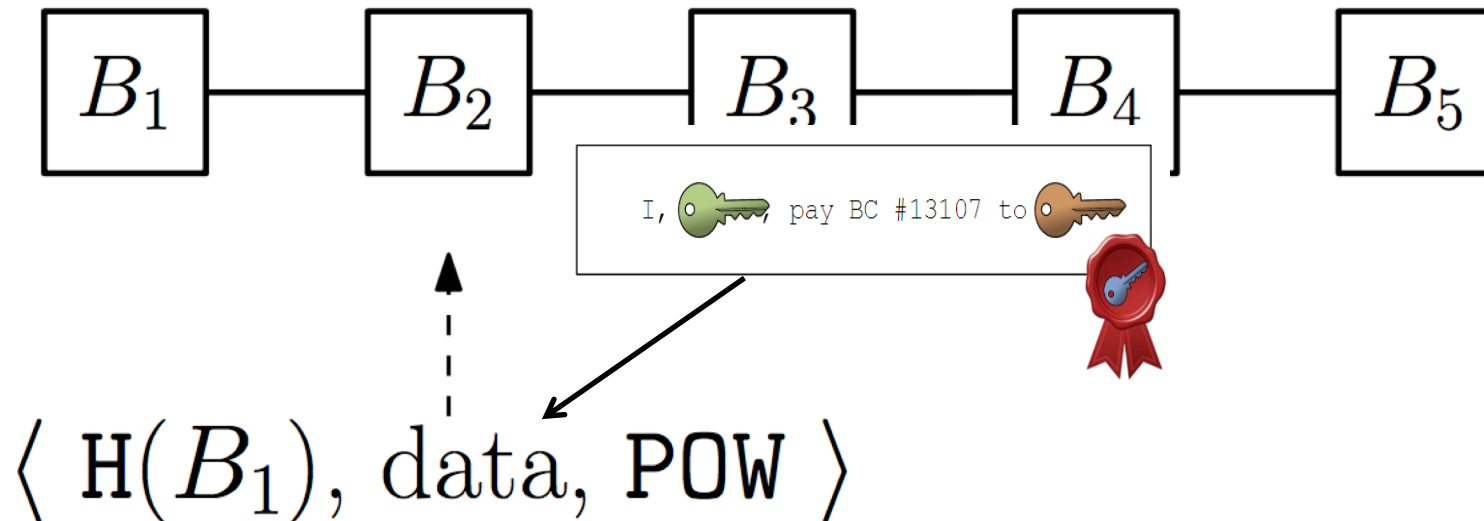
What Is a Blockchain?

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



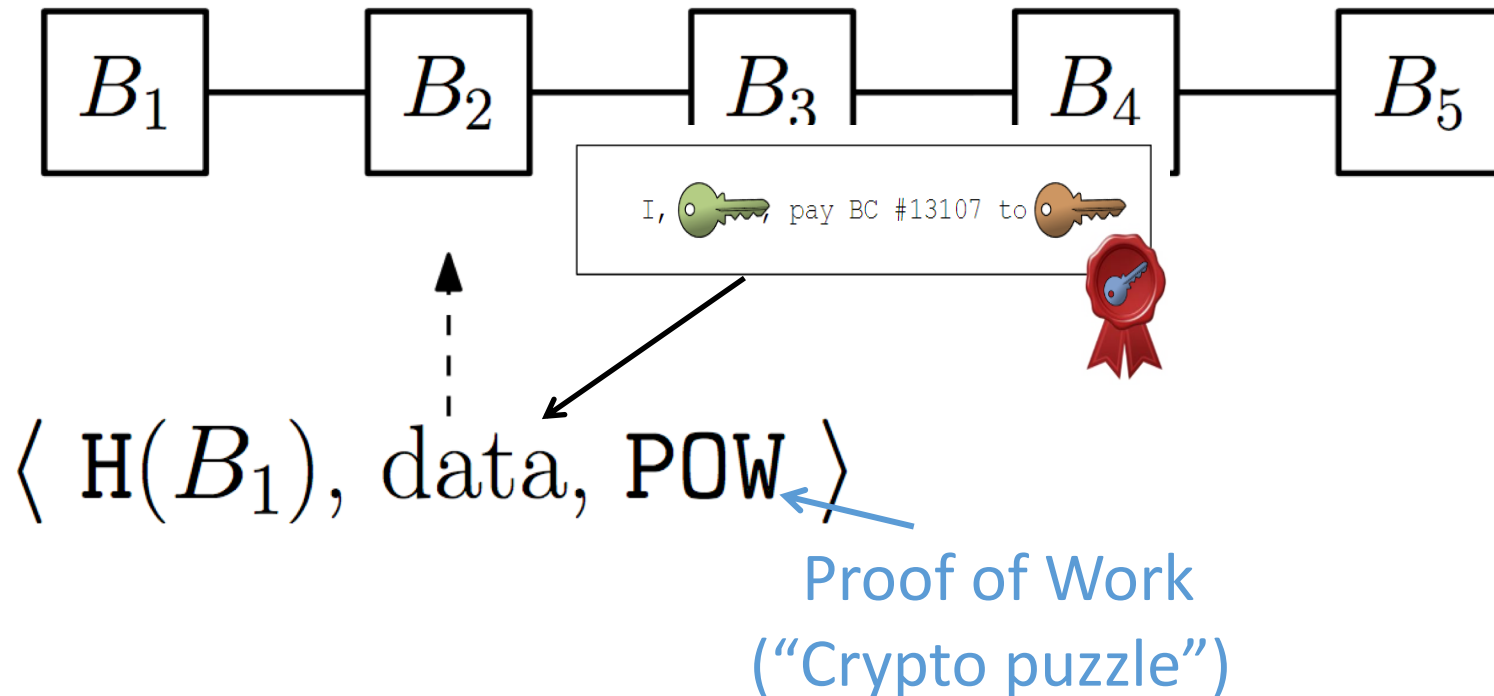
What Is a Blockchain?

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



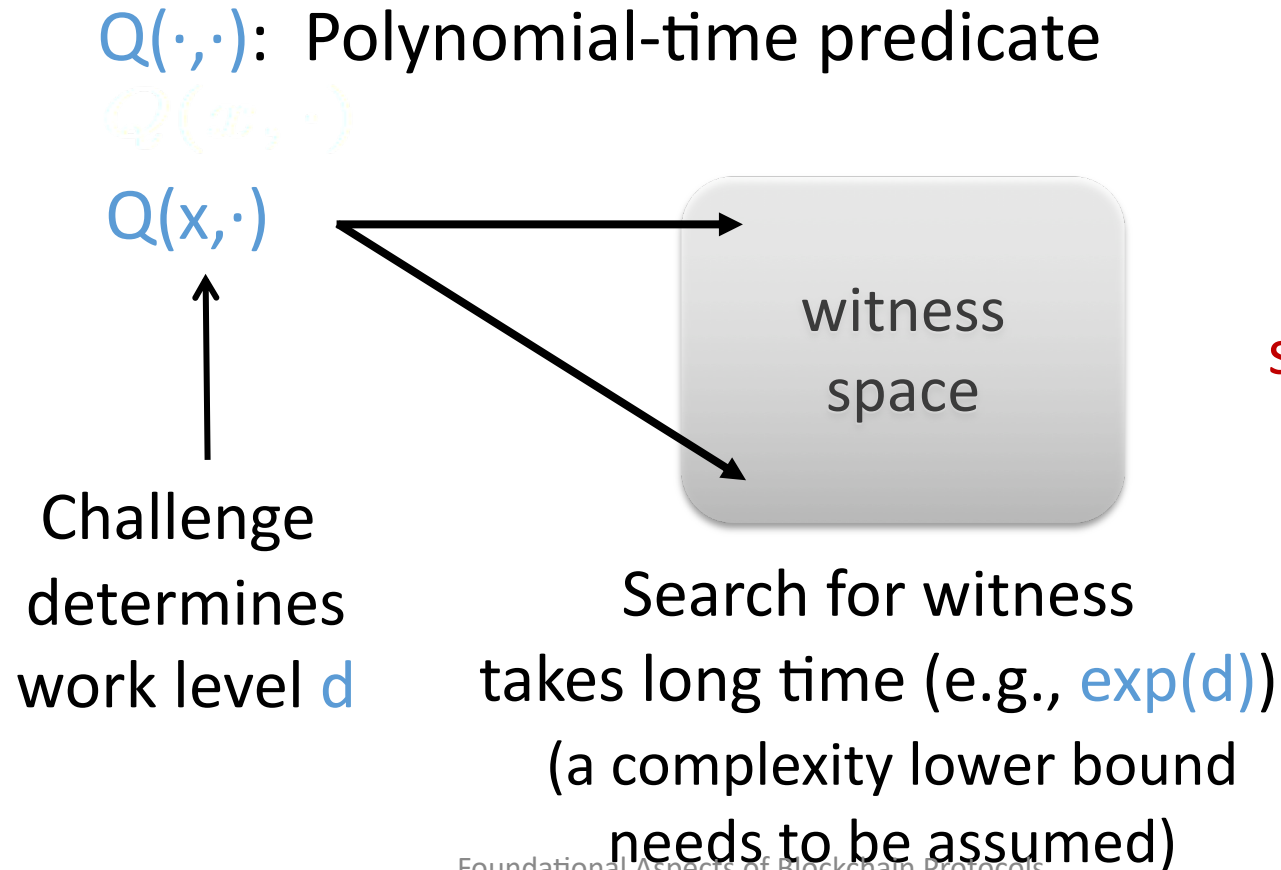
What Is a Blockchain?

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



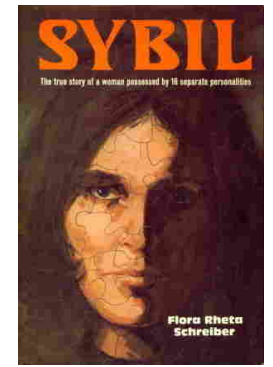
Proofs of Work (aka “Crypto Puzzles”)

- “Moderately hard functions” [DN92,RSW96,Back97,JB99,GMPY06...]



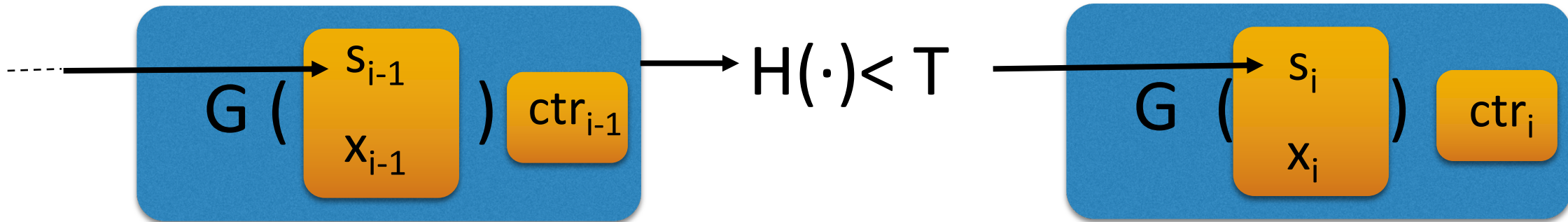
Successful only with some (small) probability!

Verification is easy!



Proofs of Work [DN92,...]

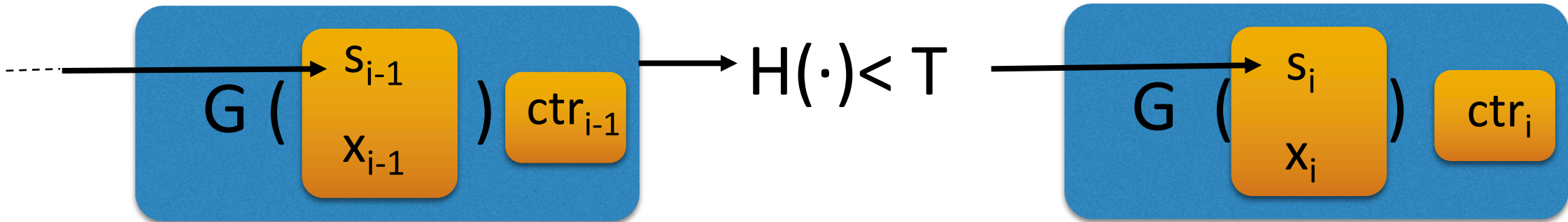
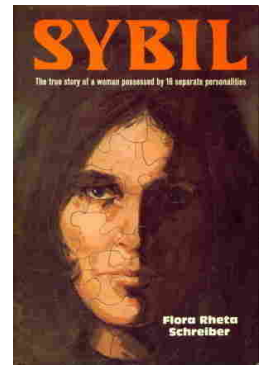
- Spam mitigation, Sybil attacks, denial of service protection, ...
- Most impactful application: Design of blockchain protocols such as Bitcoin



- This talk's focus: PoW
 - Proof-of-* (* = stake, space/memory,...); more later

Proofs of Work [DN92,...]

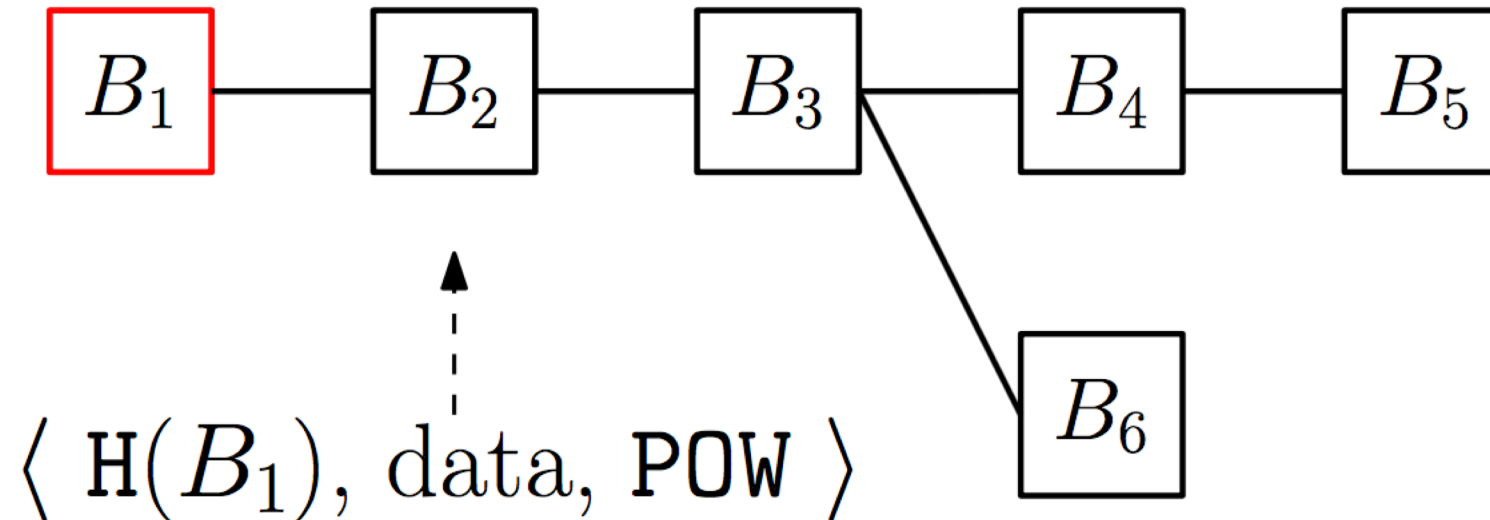
- Spam mitigation, Sybil attacks, denial of service protection, ...
- Most impactful application: Design of blockchain protocols such as Bitcoin



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

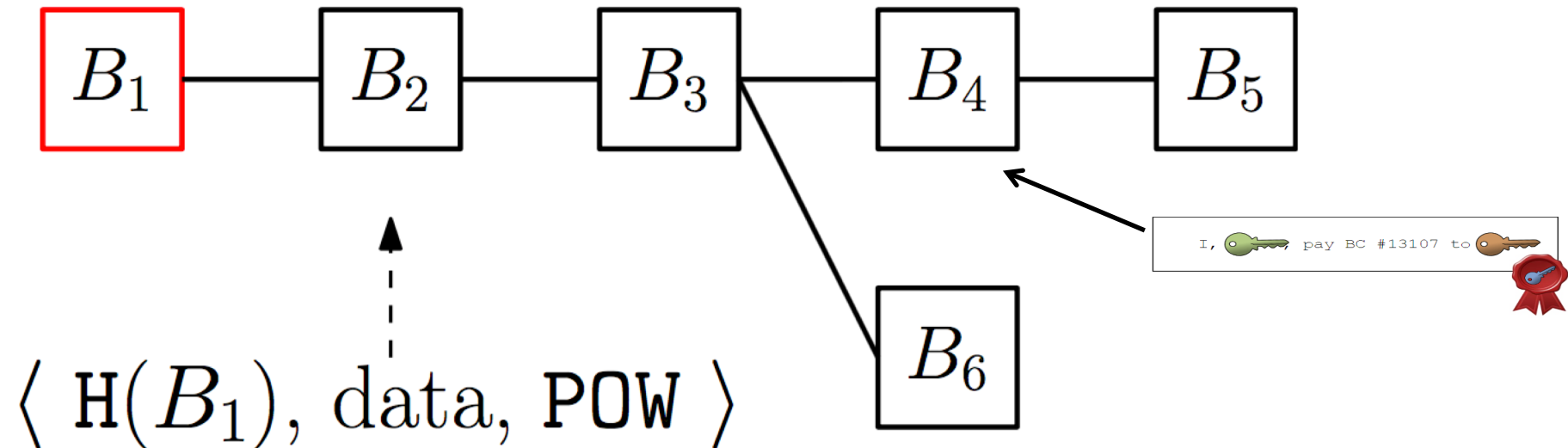
What Is a Blockchain?

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



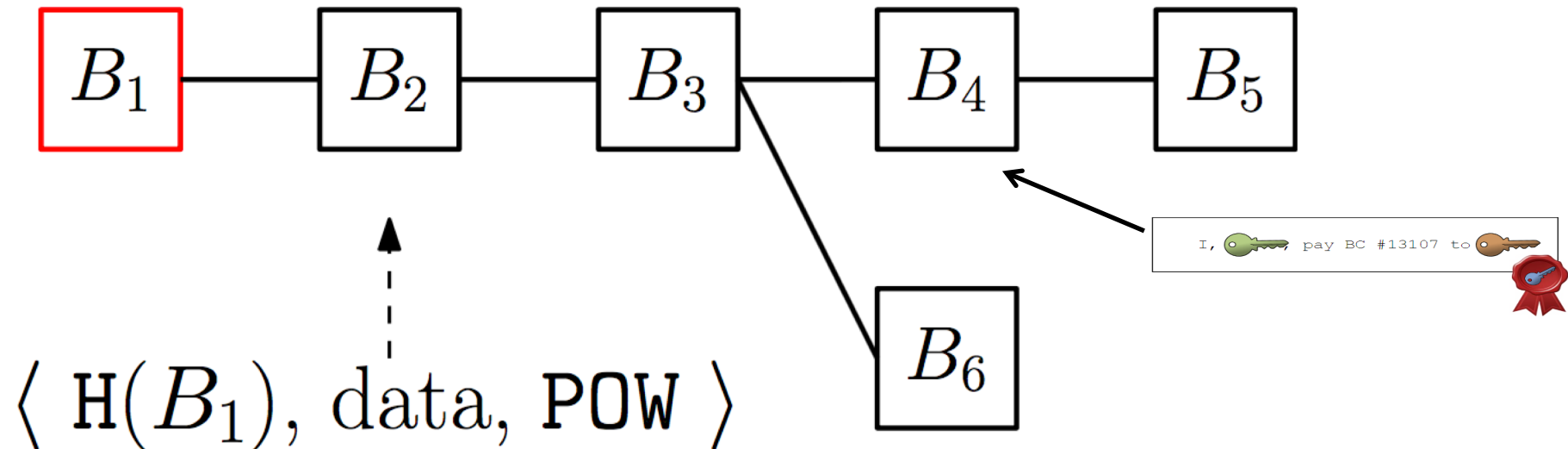
What Is a Blockchain?

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



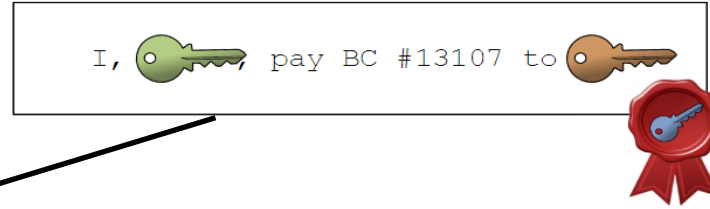
What Is a Blockchain?

- 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”

Using PoWs



- Miners collect a set of transactions

$$\mathbf{tx} = (\mathbf{tx}_1, \mathbf{tx}_2, \dots, \mathbf{tx}_m)$$

- Then do “work”

$\mathbf{ctr} := 0$; **while** $\text{Hash}(\mathbf{ctr}; \text{Hash}(\tau, \mathbf{tx})) > T$ **do** $\mathbf{ctr}++$

T : block’s “target” (*difficulty level*)

($T = 000000000000000000171A8B00$)

- If **while** loop terminates “broadcast” $(\tau, \mathbf{ctr}, \mathbf{tx})$
(new “block”: state, counter, set of transactions)
- This talk’s focus: RO-based PoWs (more later...)

Using POWs (2)

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

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

- Extend the transaction ledger

$$\tau := \tau' \parallel \mathbf{tx}'$$

$$\tau = \tau \parallel \mathbf{tx}'$$

(“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
- 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,...]

Outstanding questions:

- 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

Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - PoS-based blockchains
- References

The Bitcoin *Backbone* Protocol — A Principled Approach



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
 - $V(\cdot)$: *content (of chain) validation predicate*
 - $I(\cdot)$: *input contribution function*
 - $R(\cdot)$: *chain reading function*

→ Distinguish data structure from application layer

Network/Computational Model

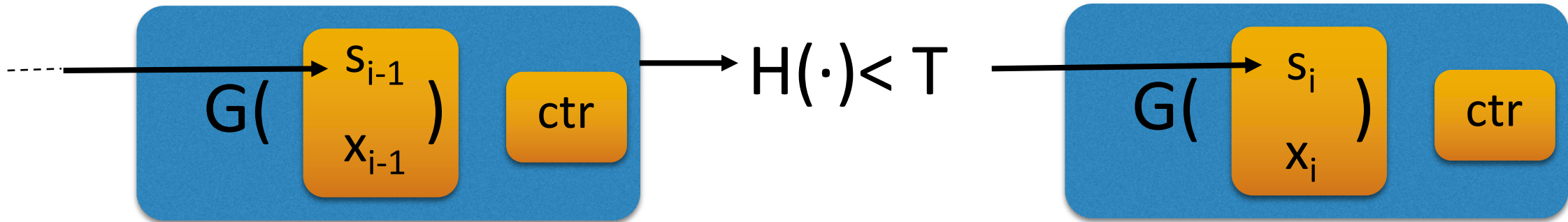
- Protocol executed by *fixed* no. of parties n (not necessarily known to participants); **active/“rushing”/adaptive adversary** controls a subset
 - Security against all possible attacks
- Underlying communication graph not fully connected (P2P); messages delivered through “**diffusion**” mechanism (“**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**
 - 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* [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...

The Bitcoin Backbone (1)

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

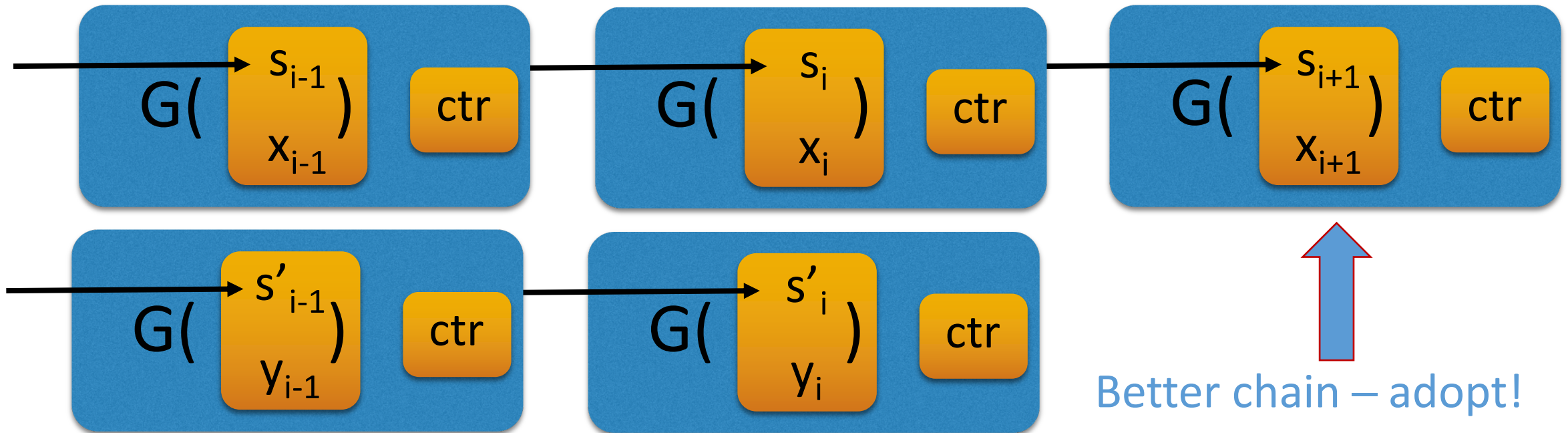


- The *contents* of C satisfy the predicate $V(x_1, \dots, x_i) = \text{true}$

$G($

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:



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

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)

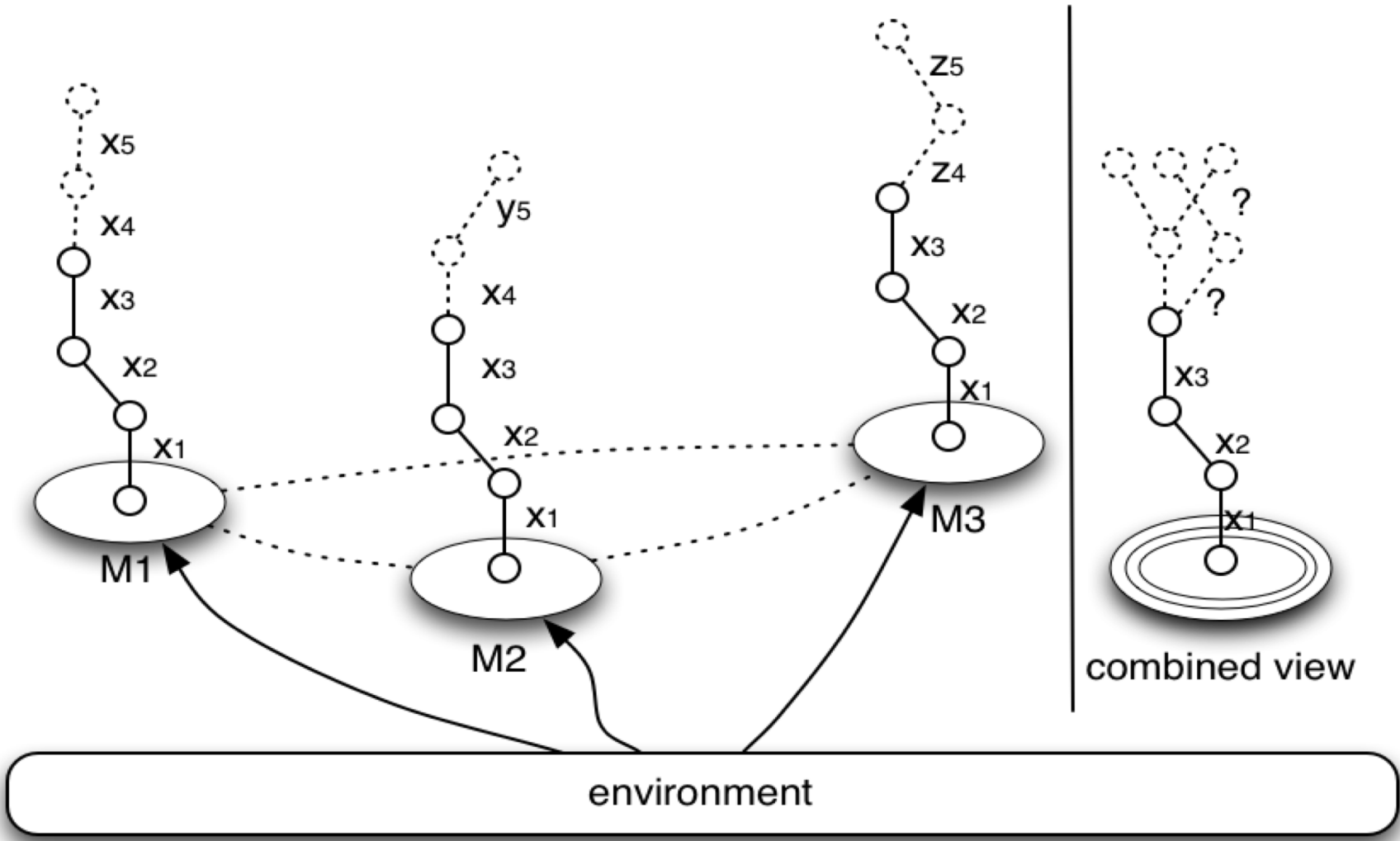
Any (large enough) chunk of an honest player's chain will contain some blocks from honest players

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

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

$$r_1 \geq r_2 \implies \alpha - \alpha^2$$

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

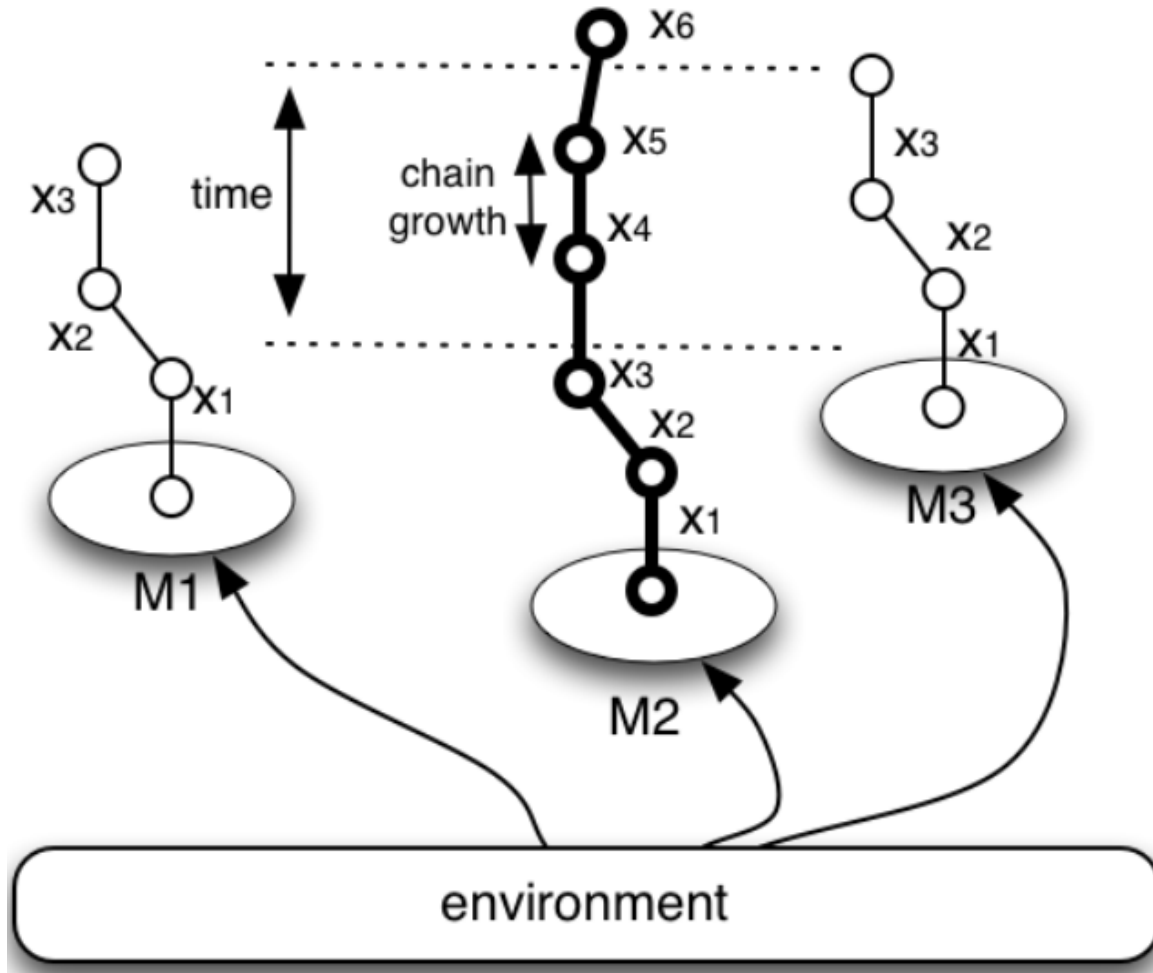
α = Honest parties expected POW solutions in a round

β = Adversary's expected POW solutions in a round

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

$$\gamma > \beta, \quad \gamma^2 = \beta^2 \implies \beta^2 - \gamma^2 = 1 > 0$$

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

α = Honest parties expected POW solutions in a round

β = Adversary's expected POW solutions in a round

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

$$\gamma > \lambda\beta, \lambda = \frac{\beta}{\gamma} < \frac{\beta}{\alpha - \alpha^2} \implies \lambda^2 - 1 > 0$$

Backbone Protocol Properties (3)

- 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

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 parties (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 $H()$, $G()$)

q : Number of POW attempts per round

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

$f = 1 - (1 - p)^{q(n-t)}$: Probability that **at least one** honest party produces a PoW in a round

- **Note:** $f \approx pq(n-t)$ ($f \ll 1$)

Prob. that **exactly one** honest party produces a PoW in a round

$$\geq pq(n-t) (1 - p)^{q(n-t)-1} > pq(n-t) (1 - f)$$

Notation (2)

X_i : Random variable $\in \{0,1\}$, honest party obtains a PoW at round i

Y_i : *Exactly one* honest party obtains a PoW at round i

Z_i : One of the corrupt parties obtains a PoW at round i

α = Honest parties expected PoW solutions in a round

S : Sequence of consecutive rounds

f $X(S): \sum_{i \in S} X_i$

Similarly $Y(S), Z(S)$

$Y(S) \approx \alpha^2$

Notation (3)

S : Sequence of consecutive rounds

$X(S)$: Number of *successful rounds* (honest parties obtain a PoW)

$Y(S)$: Number of *uniquely successful rounds* (one honest party obtains a PoW)

$Z(S)$: 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 $n-t/t > (1+\delta)$ (assume $\lambda = 1$):

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

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

Typical Executions

- Consider a number of rounds polynomial in κ
- An execution is *typical* with parameter ϵ if for any set of rounds S , $|S| = \Omega(\kappa)$:

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

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

$$Z(S) < (1 + \epsilon) E[Z(S)]$$

- No collisions or predictions take place against $H()$

Typical Executions (3)

- **Theorem 10:** Typical executions happen almost always (i.e., with probability $1 - e^{-\Omega(\kappa)}$)

Proof sketch:

- X, Y, Z follow the binomial distribution, so we can show the theorem with overwhelming prob. in κ via a Chernoff bound
- There is no collision or prediction for the hash function – from *random oracle* and *input entropy* assumptions

RECALL

Proof Strategy

1. Define the notion of a *typical execution*
2. Argue that typical executions happen with overwhelming probability
3. Prove CP, CQ, CG

$$f = \alpha + \beta$$

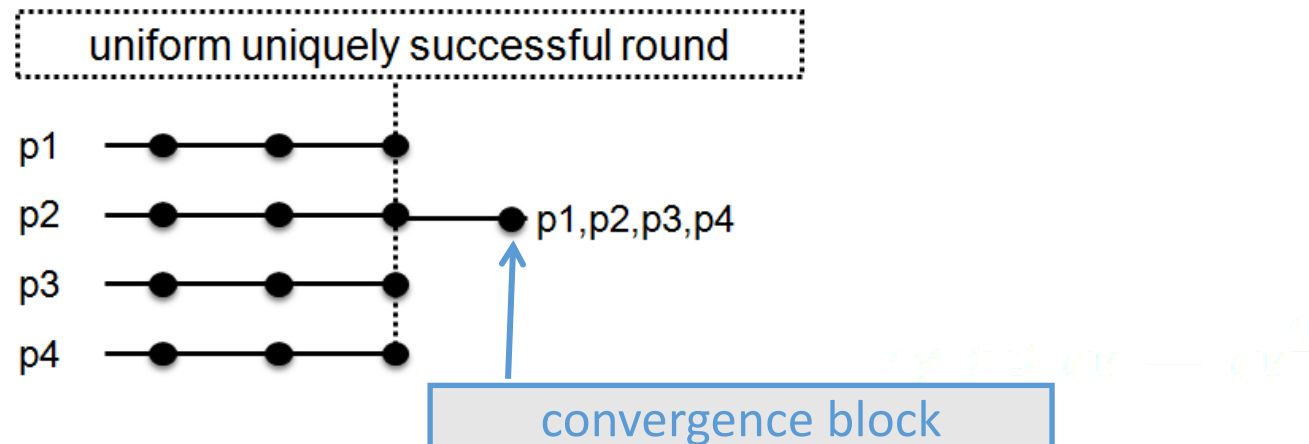
$$f = \alpha + \alpha^2$$

Common Prefix Property

- Recall:

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

- Set $k = \Omega(\kappa)$
- Uniquely successful round*: Round when exactly one honest party generates a POW



Common Prefix Property (2)

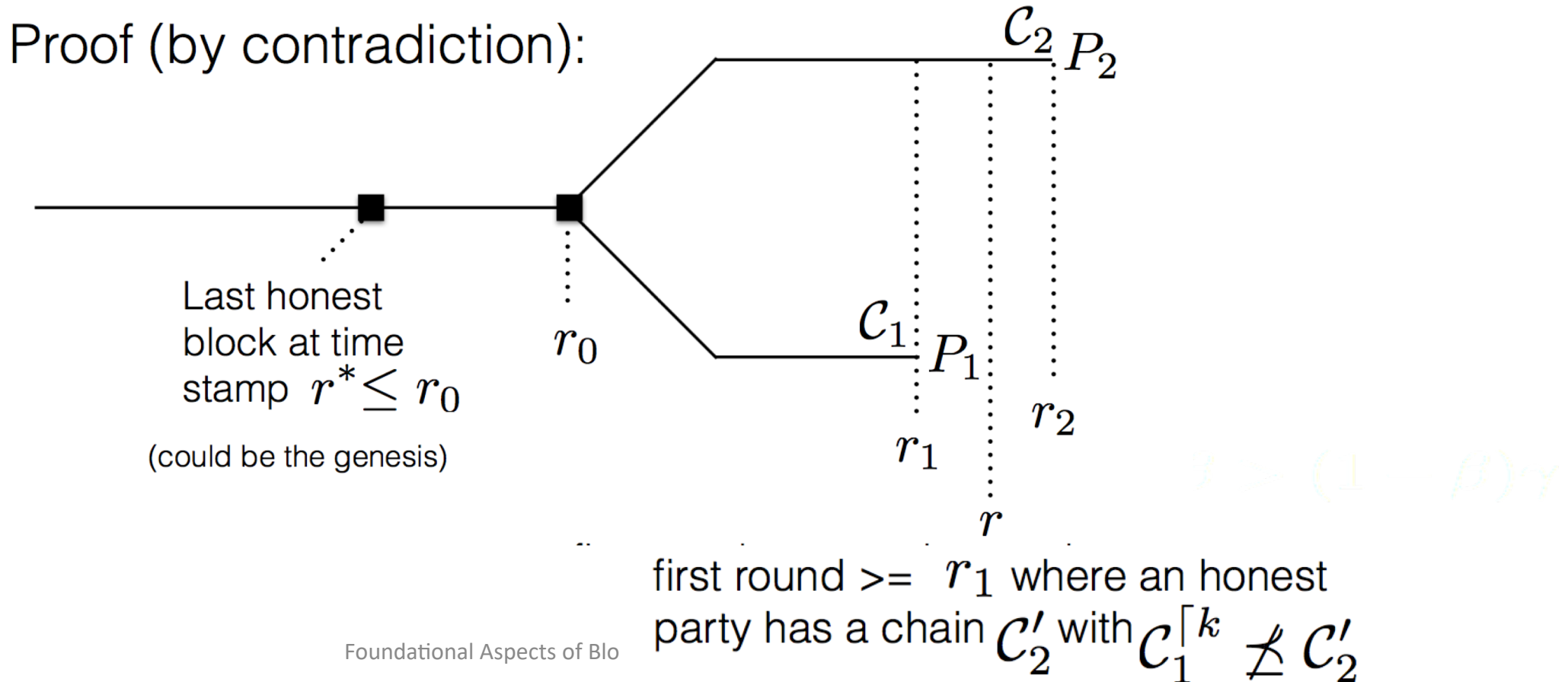
- **Theorem 15:** In a typical execution, the **Common Prefix** property holds with parameter $k = \Omega(\kappa)$, under the Honest Majority assumption ($\delta > 2\epsilon + f$)

$$(1 - \beta)\gamma$$

$$\beta > (1 - \beta)\gamma$$

Common Prefix Property (2)

Theorem 15: In a typical execution, the Common Prefix property holds with parameter $k = \Omega(\kappa)$, under the Honest Majority assumption ($\delta > 2\epsilon + f$)



Common Prefix Property (3)

At round $r - 1$:

All honest parties have a chain \mathcal{C}_i^{r-1} with $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_i^{r-1}$

At the end of round $r - 1$ chain \mathcal{C}'_2 is transmitted for which we know that

$$\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}'_2$$

[by assumption]

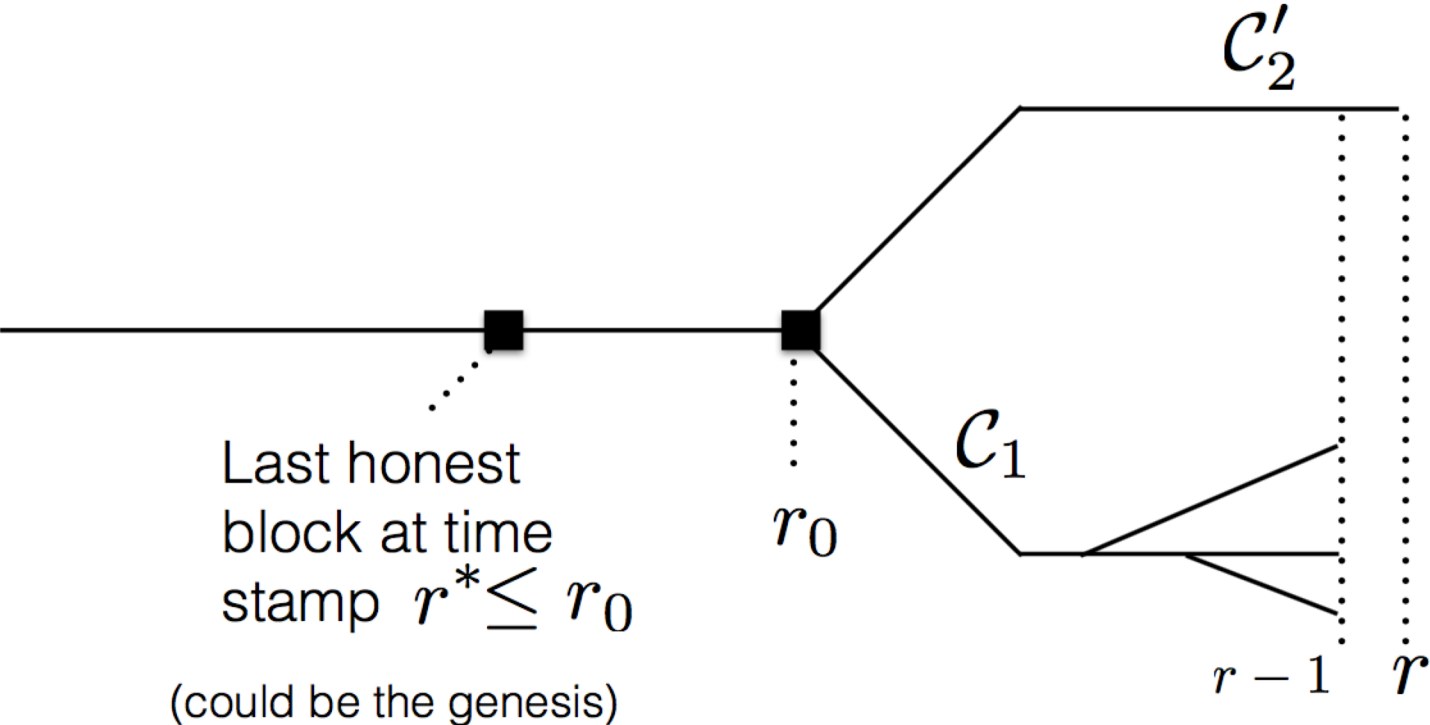
$$|\mathcal{C}'_2| \geq |\mathcal{C}_1|$$

[by the fact that \mathcal{C}'_2 will be accepted at round r by an honest party while at least one honest party at round $r_1 \leq r$ possessed chain \mathcal{C}_1]

(1) (2) (3)

Common Prefix Property (4)

Consider the set of rounds $S = \{r^*+1, \dots, r-1\}$



$(1 - \beta) \gamma$

Common Prefix Property (5)

Consider a *uniquely successful round* in $S = \{r^*+1, \dots, r-1\}$

Lemma: If a block is created in a uniquely successful round at position m in a blockchain, then no other honest party will ever mine at position m in *any* blockchain

Therefore each uniquely successful round in S creates a block that must be matched by another block of the adversary

Lemma: Such adversarial block should also be created within S (by the choice of r^* and typicality)

Common Prefix Property (6)

- It follows that $Z(S) \geq Y(S)$ and $|S| = \Omega(\kappa)$
- By typicality: $Y(S) > (1 - \epsilon) E[Y(S)]$
 $Z(S) < (1 + \epsilon) E[Z(S)]$

Recall:

$$\begin{aligned} E[X(S)] &\approx pq(n-t) |S| \\ E[Y(S)] &> (1-f) E[X(S)] \\ E[Z(S)] &= pqt |S| \\ n-t/t &> \lambda(1+\delta) \end{aligned}$$

$$(1 + \epsilon) pqt |S| \geq (1 - \epsilon)(1 - f) pq(n-t) |S|$$

iff $(1 + \epsilon) / (1 - \epsilon)(1 - f) \geq n-t/t$

Contradiction, as long as

$$(1 + \epsilon) / (1 - \epsilon)(1 - f) < (1 + \delta)$$

which is implied by

$$\delta > 2\epsilon + f$$

QED

Common Prefix: Conclusions

- Using the fact that typical executions happen with overwhelming probability in κ (**Theorem 10**), we have shown:
- **Theorem 15:** In a typical execution, the **Common Prefix** property holds with parameter $k = \Omega(\kappa)$, under the Honest Majority Assumption ($\delta > 2\epsilon + f$)

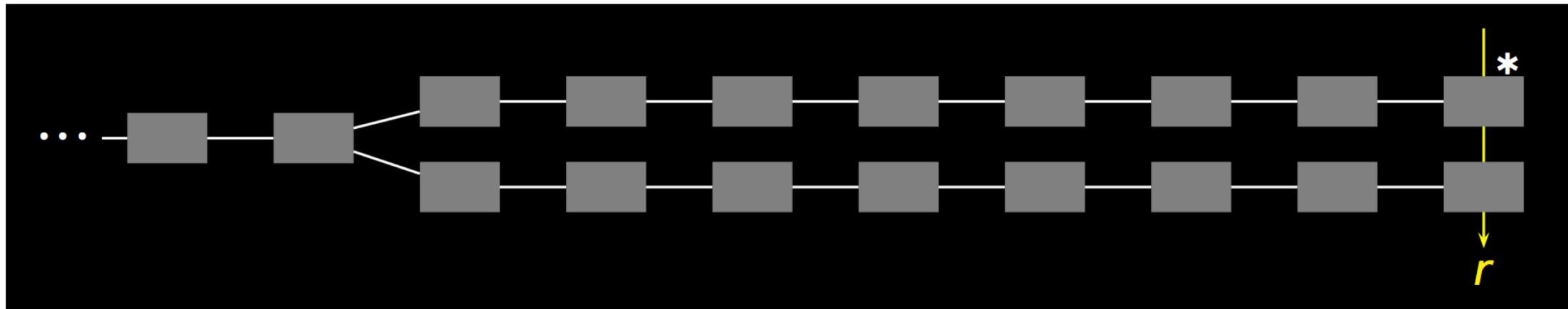
“Excess” of honest parties

Quality of concentration of random variables

Prob. that at least one honest party produces a POW ($f \approx pq(n-t)$)

Common Prefix: Conclusions (2)

- 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 \geq 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

α = Honest parties' expected POW solutions in a round

β = Adversary's expected POW solutions in a round

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

$$\gamma > \lambda \beta, \lambda = \frac{\beta}{\alpha} \implies \lambda \alpha - 1 > 0$$

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

- **Note:** In the absence of an adversary, any set of h honest parties obtain as many blocks as their proportion of hashing power: h/n
- *Ideal chain quality:* If this the case for adversarial parties, i.e., $\mu = t/n$
- *Not achieved* by the Bitcoin backbone protocol

Chain Quality Property (2)

- Consider chain C from an honest party, and l consecutive blocks from that chain
- The CQ coefficient is $\mu = 1/\lambda$ (Recall: $n - t > \lambda(1+\delta) t$)
- **Note:** Proportion of honest blocks $\geq (1 - \mu)k$

$\epsilon =$ Honest parties expected POW solutions in a round

Theorem 16: In a typical execution, the Chain Quality property holds with parameters $\mu = 1/\lambda$ and $k = \Omega(\kappa)$ under the Honest Majority Assumption ($\delta > 2\epsilon + f$)

- 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

Chain Quality Property (3)

Proof: (By contradiction)

- Consider a sequence of blocks $B_u \dots B_v$ in the chain of an honest party with $l = v - u + 1$
- Define an expanded sequence of blocks $B_{u'} \dots B_{v'}$ with $L = v' - u' + 1 \geq l$ occurring during rounds $S = (r_1, \dots, r_2)$ and $\text{OW solutions in a round}$
 1. $B_{u'}$ was produced by an honest party at round r_1
 2. $B_{v'}$ was accepted by an honest party at round r_2
- Let x denote the number of blocks produced by honest parties in the l blocks
- For the sake of contradiction, assume $x < (1 - \mu) l$ ($\leq (1 - \mu) L$)

Chain Quality Property (4)

Lemma: Because of typicality, all the L blocks are computed within $S = (r_1, \dots, r_2)$

Lemma: Because of the choice of S , we have that $L \geq X(S)$ (otherwise no honest party would accept B_v)

Using the above, and $x < (1 - \mu)l$, we have:

$$Z(S) \geq L - x \geq \mu L \geq \mu X(S)$$

$$(1 - \beta)\gamma$$

$$\beta > (1 - \beta)\gamma$$

Chain Quality Property (5)

- It follows that $Z(S) \geq \mu X(S)$ and $|S| = \Omega(\kappa)$
- By typicality: $X(S) > (1 - \epsilon) E[X(S)]$
 $Z(S) < (1 + \epsilon) E[Z(S)]$

Recall:

$$\begin{aligned} E[X(S)] &\approx pq(n-t) |S| \\ E[Y(S)] &> (1-f) E[X(S)] \\ E[Z(S)] &= pqt |S| \\ n-t/t &> \lambda(1+\delta) \end{aligned}$$

$$(1 + \epsilon) pqt |S| \geq \mu (1 - \epsilon) pq(n-t) |S|$$

iff $(1 + \epsilon) / (1 - \epsilon) \geq \mu (n-t) / t$

Contradiction, as long as

$$(1 + \epsilon) / (1 - \epsilon) < (1 + \delta)$$

which is implied by

$$\delta > 2\epsilon + f \quad (\text{Verify!})$$

QED

Chain Quality: Conclusions

- Using the fact that typical executions happen with overwhelming probability in κ (**Theorem 10**), we have shown:
- Theorem 16:** In a typical execution, the **Chain Quality** property holds with parameter parameter $\mu = 1/\lambda$ and $k = \Omega(\kappa)$ under the Honest Majority Assumption **($\delta > 2\epsilon + f$)**

“Excess” of honest parties

Prob. that at least one honest party produces a POW ($f \approx pq(n-t)$)

Quality of concentration of random variables

$$(1 - \beta)\gamma$$

$$\beta > (1 - \beta)\gamma$$

Chain Quality: Conclusions (2)

- Under assumption that ties between chains of equal length always favor the (“rushing”) adversary, Theorem 16 is “tight”
- Adversarial strategy: “selfish mining”
 - When adversary finds a solution (PoW), it keeps it to itself and keeps extending the private chain. Whenever an honest party finds a POW, adversary releases one block from the private chain
- Selfish mining implies that this is tight... Bitcoin’s rewarding mechanism is *not* incentive-compatible

$$(1 - \beta)\gamma$$

$$\beta > (1 - \beta)\gamma$$

RECALL

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*

$$f = \alpha + \beta$$

$$f = \alpha + \alpha^2$$

RECALL

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*

Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - PoS-based blockchains
- References

Applications of the Bitcoin Backbone Protocol

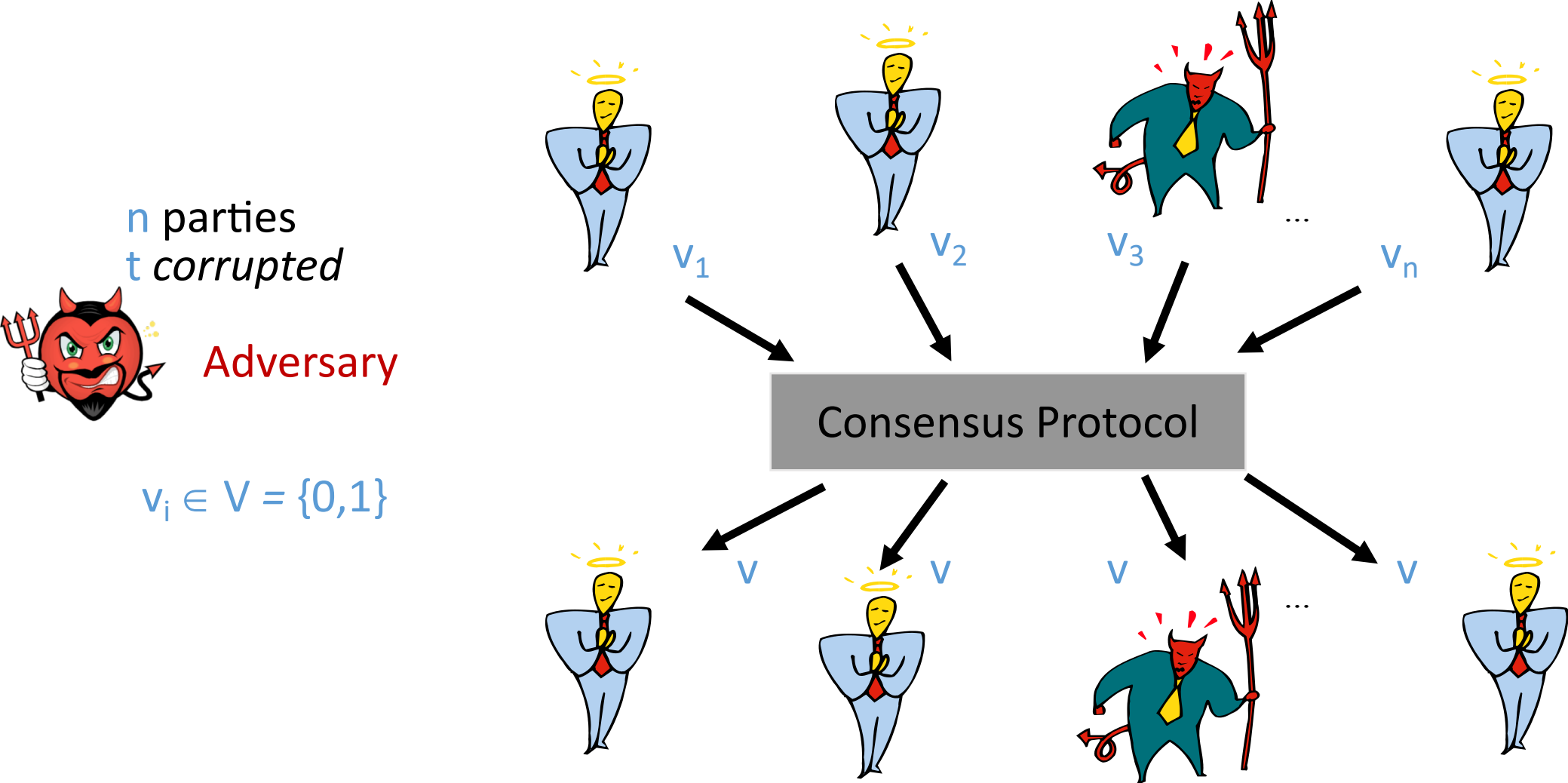
- Consensus
- Robust transaction ledger (Bitcoin)
 - Aka “ledger consensus,” “Nakamoto consensus”



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

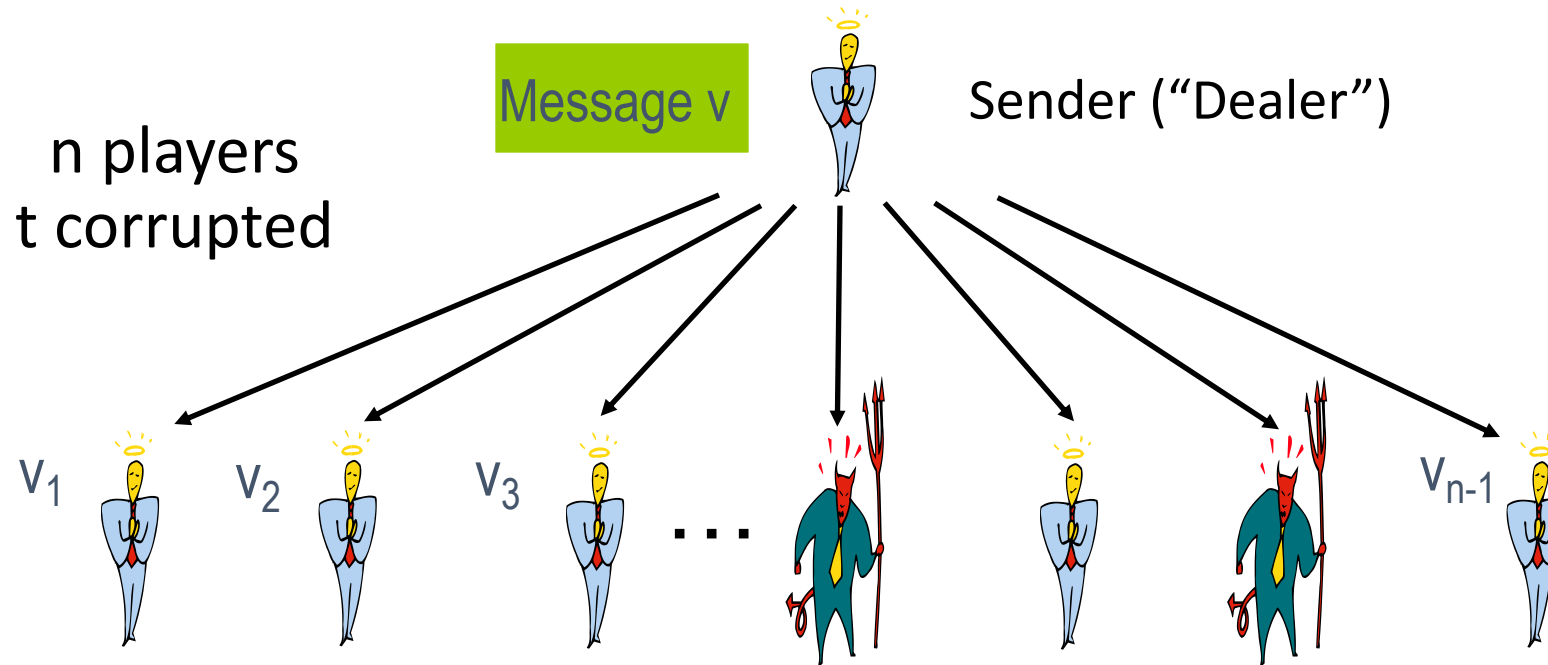
Consensus (Byz. Agreement) [PSL80, LSP82]



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

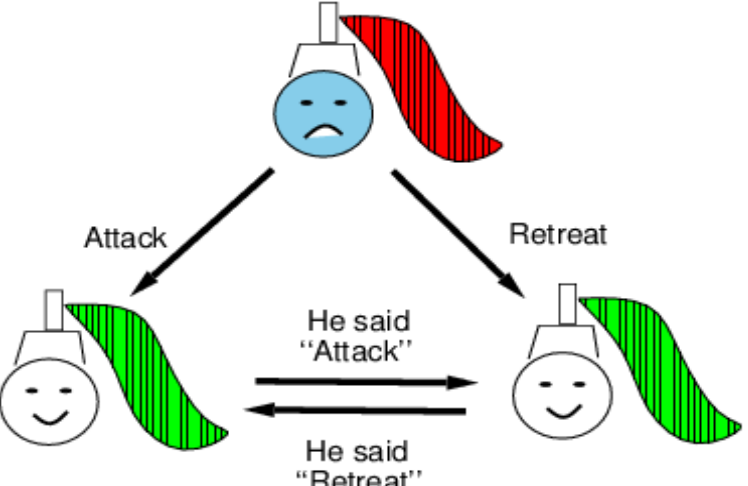
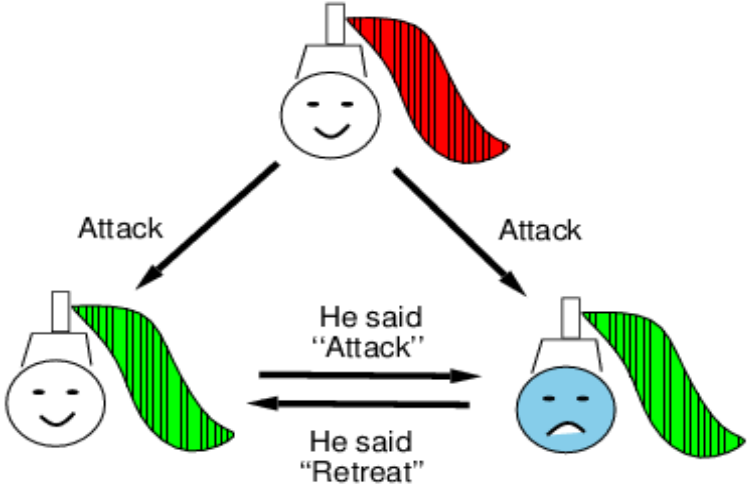
Consensus (Byz. Agreement) [PSL80, LSP82] (2)

- **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 $n \leq 3t$ [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]: $r = t+1$, $\text{poly}(n)$
 - 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
- $r = 1, \dots, t+1$:
 - 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

On the Necessity of an Honest Majority for Consensus

$$t < n/2$$

regardless of the resources available to the parties

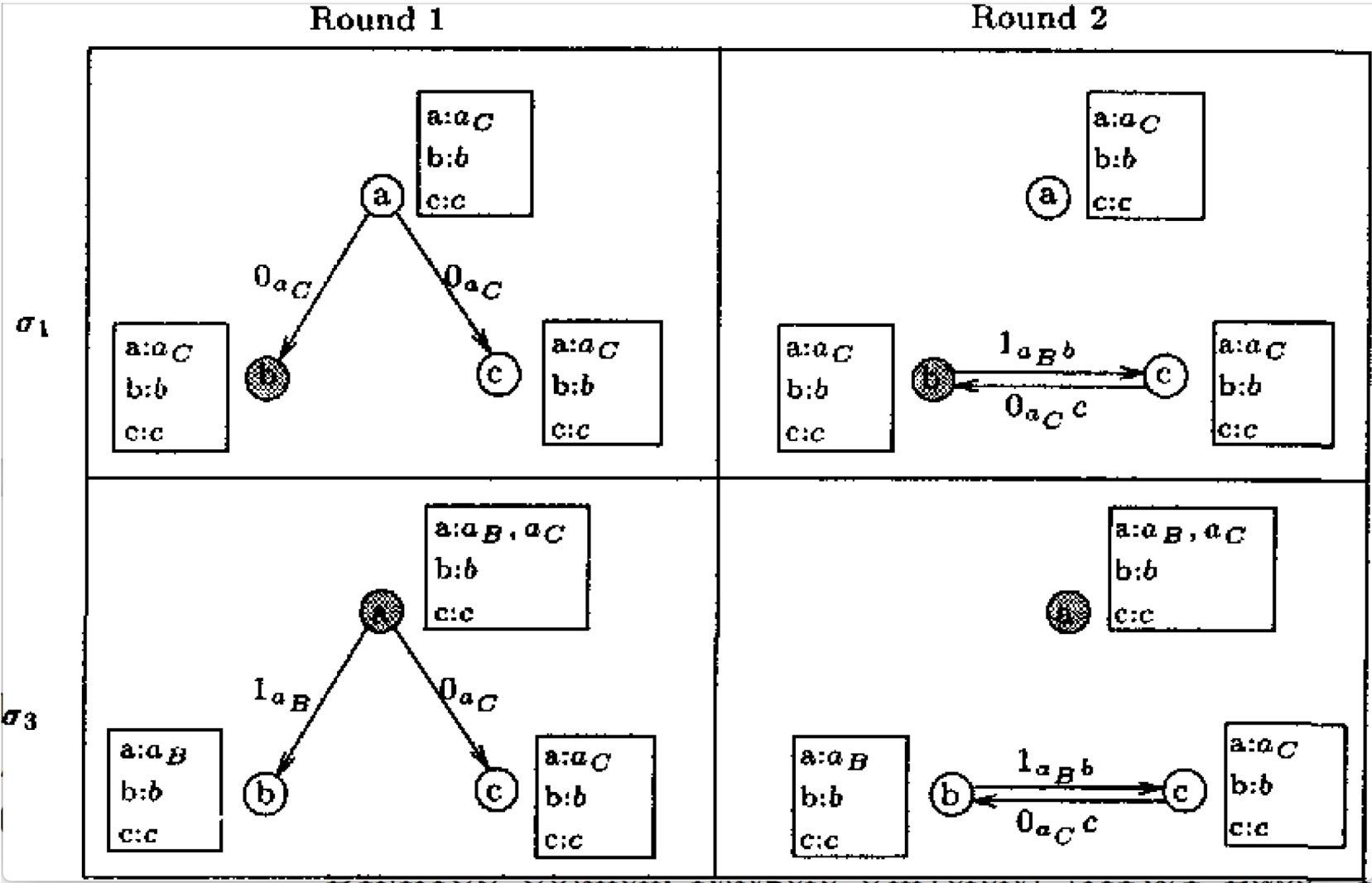
On the Necessity of an Honest Majority for Consensus (2)

- **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 1In 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 $n \leq 3t$

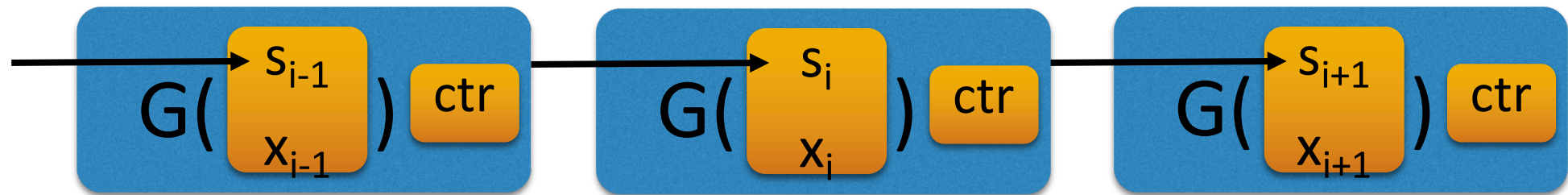
On the Necessity of a PKI (“Private-state Setup”) [Bor96]



Consensus/Broadcast

- One of the most fundamental problems in distributed systems
- 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

Blockchain-based 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)

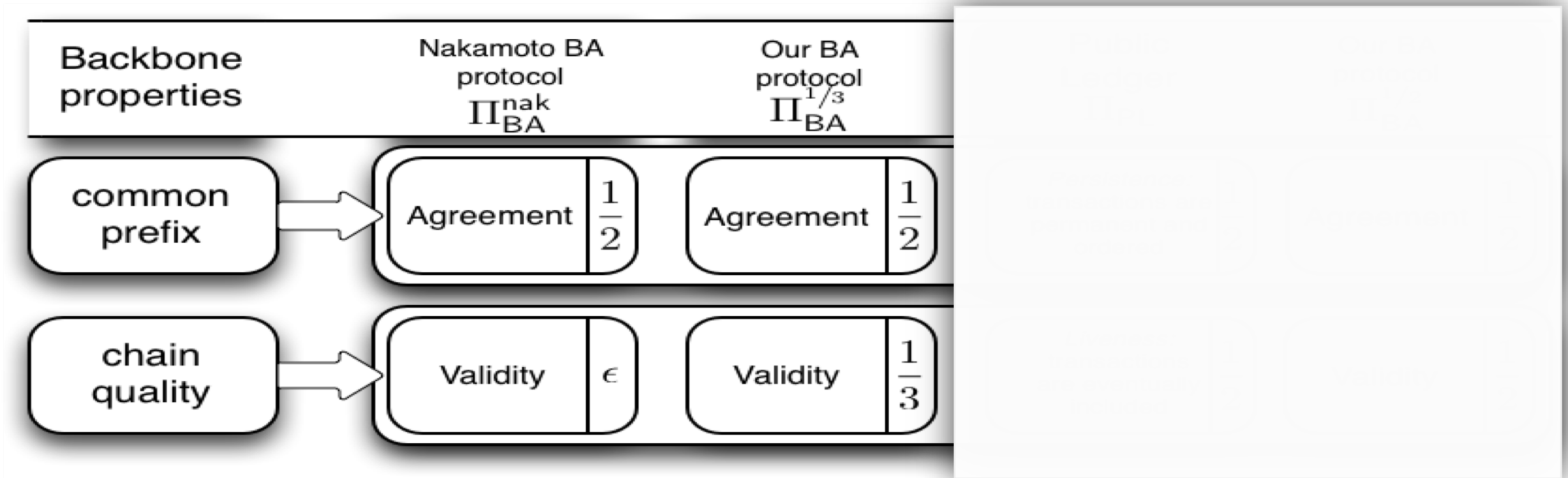
- The n parties start building a blockchain inserting their input
- If a party receives a longer blockchain, it switches to that one and switches its input
- When the blockchain is long enough the party outputs the (unique) value that it contains

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

- 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 $< \frac{1}{3}$ of the parties (tight, due to the *Chain Quality* property)

1/3 Consensus Protocol (2)

- Specification of V, I, R

| | |
|---|--|
| Content validation predicate $V(\cdot)$ | $V(\langle x_1, \dots, x_n \rangle)$ is true if and only if $v_1, \dots, v_n \in \{0, 1\}, \rho_1, \dots, \rho_n \in \{0, 1\}^\kappa$ 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 $V(\langle x_1, \dots, x_n \rangle) = \text{True}$ and $n \geq 2k$, the value $R(\mathcal{C})$ is the majority bit of v_1, \dots, v_k where $x_i = \langle v_i, \rho_i \rangle$; otherwise (i.e., the case $V(\langle x_1, \dots, x_n \rangle) = \text{False}$ 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 $C_1 \upharpoonright^k \leq C_2$ does not hold, in violation of the Common Prefix property.

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*

1/2 Consensus Protocol

- **Idea:** Use POWs to also “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

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

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
$$(H_1(\text{ctr}, G(\text{nonce}, v)) < T) \wedge (\text{ctr} \leq q)$$
- **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

2-for-1 POWs: Composition of POW-based Protocols

$h \leftarrow G(x,s)$
if $H(h,ctr) < T$ then...

$h' \leftarrow G(x',s')$
if $H(h',ctr') < T'$ then...

Given $((x,s), ctr)$
Verify $H(G(h,s), ctr) < T$
Given $((x',s'), ctr')$
Verify $H(G(h',s'), ctr') < T'$

$h \leftarrow G(x,s)$
 $h' \leftarrow G(x',s')$
 $w \leftarrow H(h,h',ctr)$

if $w < T$ then...
if $[w]^R < T'$ then...

Given $((*,*), (s',x') ctr')$
Verify
 $[H(G(*,*), G(s',x'), ctr')]^R < T'$

Not
secure

2-for-1 POWs: Composition of POW-based Protocols

$h \leftarrow G(x,s)$
if $H(h,ctr) < T$ then...

$h' \leftarrow G(x',s')$
if $H(h',ctr') < T'$ then...

Given $((x,s), ctr)$
Verify $H(G(h,s), ctr) < T$
Given $((x',s'), ctr')$
Verify $H(G(h',s'), ctr') < T'$

$h \leftarrow G(x,s)$
 $h' \leftarrow G(x',s')$
 $w \leftarrow H(h,h',ctr)$

if $w < T$ then...
if $[w]^R < T'$ then...

Given $((s,x), (*, *) ctr)$
Verify
 $[H(G(s,x), G(*, *), ctr)] < T$

Not
secure

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 $\frac{1}{2}$ | <i>Persistence:</i> transactions are permanent and ordered $\frac{1}{2}$ | Agreement $\frac{1}{2}$ |
| chain quality | Validity ϵ | Validity $\frac{1}{3}$ | <i>Liveness:</i> transactions are eventually included $\frac{1}{2}$ | Validity $\frac{1}{2}$ |

PoW-based Consensus: A “Common” Approach [GKLP18]

- **Q:** Is the blockchain/PoW-based approach is radically different from the traditional probabilistic protocols?
- $\text{PoW} \cong$ Permissionless signature-like primitive
 - It enables honest-majority consensus in the same way as classical signatures in the traditional setting
- Underlying idea:
 - Parties (*try to*) “broadcast” their inputs (with only probabilistic guarantees in the case of PoWs)
 - When sufficient time has passed for a common view to established, parties output the majority of the received values

RECALL

Applications of the Blockchain

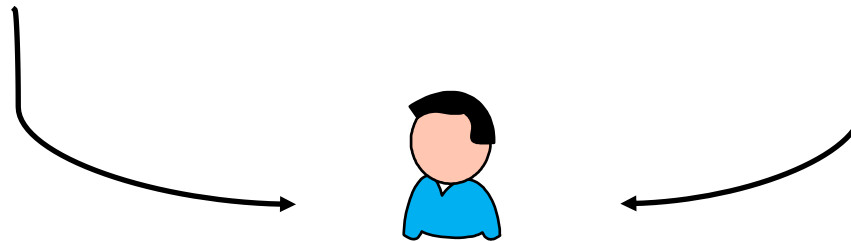
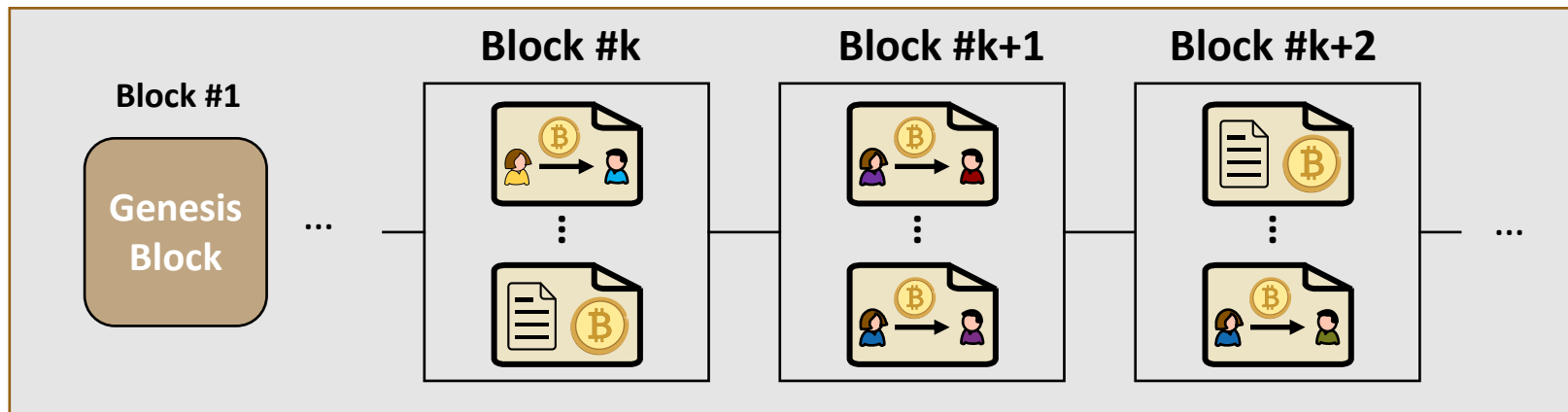
- Consensus
- Robust transaction ledger (Bitcoin)
 - Aka “ledger consensus,” “Nakamoto consensus”



Blockchain Abstractions

RECALL

A Public Ledger or a Bulletin Board

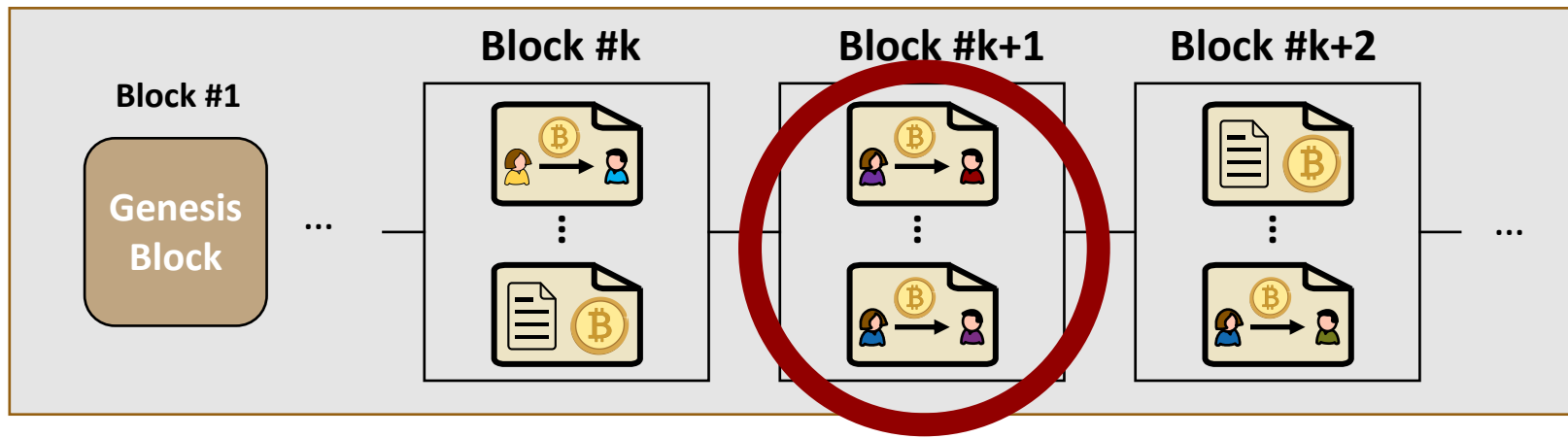


2) Anyone can read the current content of the ledger

Blockchain Abstractions

RECALL

A Public Ledger or a Bulletin Board



3) No modifications of contained blocks possible

Robust Public Transaction Ledgers: Properties (cf. [Sch91])

- **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 $V(\cdot)$ | $V(\langle x_1, \dots, x_m \rangle)$ is true if and only if the vector $\langle x_1, \dots, x_m \rangle$ is a valid ledger, i.e., $\langle x_1, \dots, x_m \rangle \in \mathcal{L}$. |
| Chain reading function $R(\cdot)$ | If $V(\langle x_1, \dots, x_m \rangle) = \text{True}$, the value $R(\mathcal{C})$ is equal to $\langle x_1, \dots, x_m \rangle$; undefined otherwise. |
| Input contribution function $I(\cdot)$ | $I(st, \mathcal{C}, round, \text{INPUT}())$ operates as follows: if the input tape contains (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, $x = \text{tx}_0 x'$ 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

$$f = \alpha + \beta$$

$$g \approx \alpha - \alpha^2$$

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* ($G(\cdot)$: hash function)



α = honest parties expected POW solutions in a round

$$f = \alpha + \beta$$

$$f \approx \alpha - \alpha^2$$

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* ($G(\cdot)$: hash function)
- *Transaction*: $tx = \{a_1, a_2, \dots, a_i\} \rightarrow (\sigma, \{(a'_1, b'_1), (a'_2, b'_2), \dots, (a'_o, b'_o)\})$
 - a_j 's : accounts to be debited; a'_k 's : accounts to be credited; b'_k 's : funds
 - $\sigma = ((vk_1, \sigma_1), (vk_2, \sigma_2), \dots, (vk_i, \sigma_i))$ – verification keys and signatures on message $\{(a'_1, b'_1), (a'_2, b'_2), \dots, (a'_o, b'_o)\}$

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*: $tx = \{a_1, a_2, \dots, a_i\} \rightarrow (\sigma, \{(a'_1, b'_1), (a'_2, b'_2), \dots, (a'_o, b'_o)\})$
 - a_j 's : accounts to be debited; a'_k 's : accounts to be credited; b'_k 's : funds
 - $\sigma = ((vk_1, \sigma_1), (vk_2, \sigma_2), \dots, (vk_i, \sigma_i))$ – verification keys and signatures on message $\{(a'_1, b'_1), (a'_2, b'_2), \dots, (a'_o, b'_o)\}$
- 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}

$$f = cv + \beta$$

$$g = cv + \alpha^2$$

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 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 $\frac{1}{2}$ | <i>Persistence:</i> transactions are permanent and ordered $\frac{1}{2}$ | Agreement $\frac{1}{2}$ |
| chain quality | Validity ϵ | Validity $\frac{1}{3}$ | <i>Liveness:</i> transactions are eventually included $\frac{1}{2}$ | Validity $\frac{1}{2}$ |

RECALL

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 Ledgers [GK18]

- Adaptation of consensus impossibility for dishonest majority
- Uses additional property:
 - *Serializability (“Order”)* [Sch91]: Conflicting transactions can’t appear in the ledger
- High-level intuition:
 - Partition parties into two sets A_b , $b \in \{1,2\}$ of size $n/2$
 - Environment prepares two transactions tx_1, tx_2 that are in conflict (can just use order)
 - By Liveness, the ledger of parties in set A_b contain tx_b
 - By Serializability, transactions in those parties’ ledgers cannot be in tx_{3-b}, tx_b order
⇒ contradiction due to Consistency

Conclusions So Far

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

Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - PoS-based blockchains
- References

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**
 - $t \cdot q$ 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...

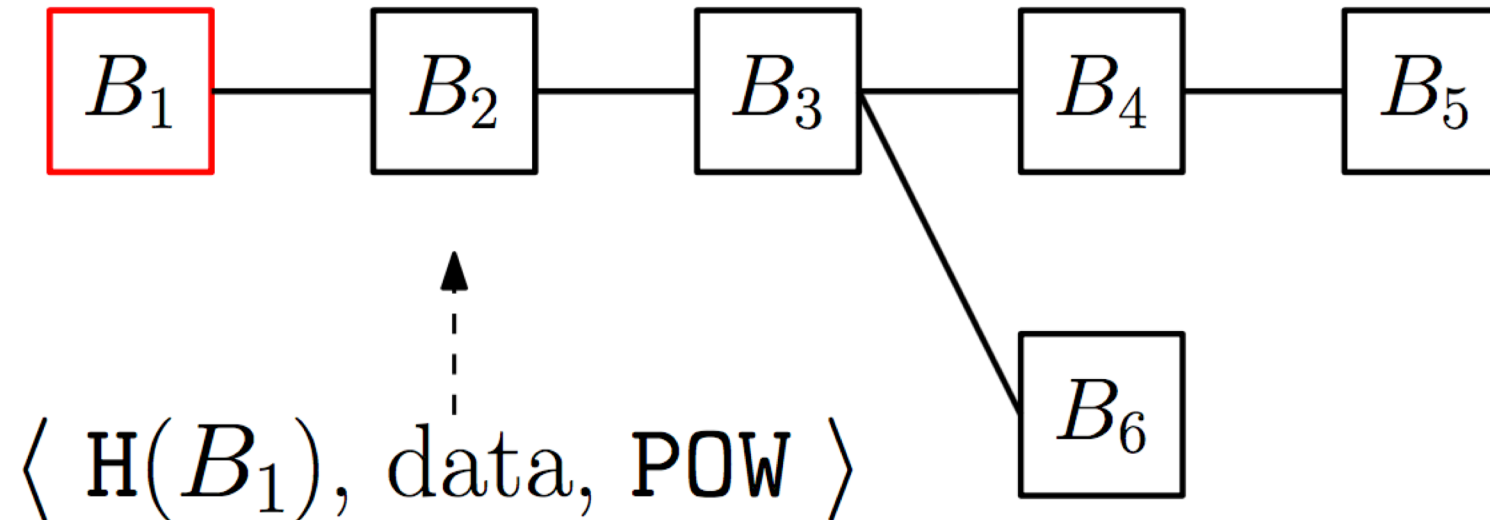
Is a Genesis Block Really Needed?

- “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”



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
 - $t \cdot q$ 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 *Bootstrapped* Backbone Protocol [GKLP18]

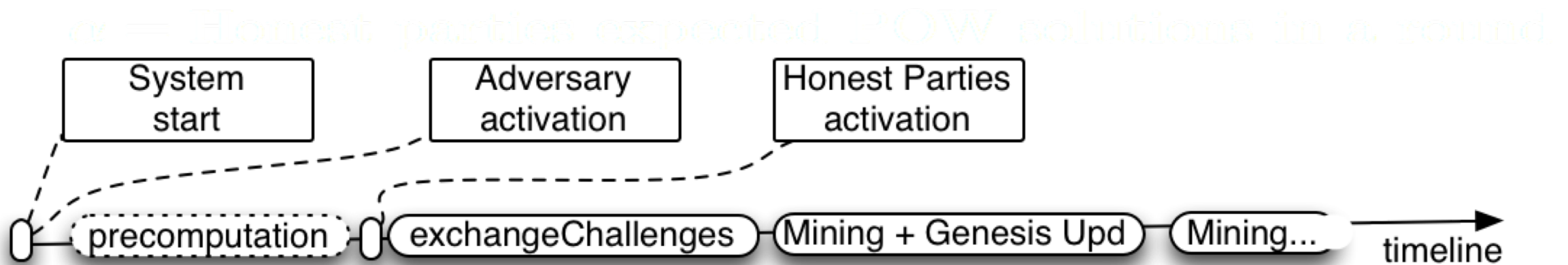
- *No trusted setup* and individual genesis block mining
- Freshness of genesis block impacts chains' total *weight*
- Personalized chain selection rule
- Robustness is achieved after an initial period of protocol stabilization

$$f = \alpha v + \beta$$

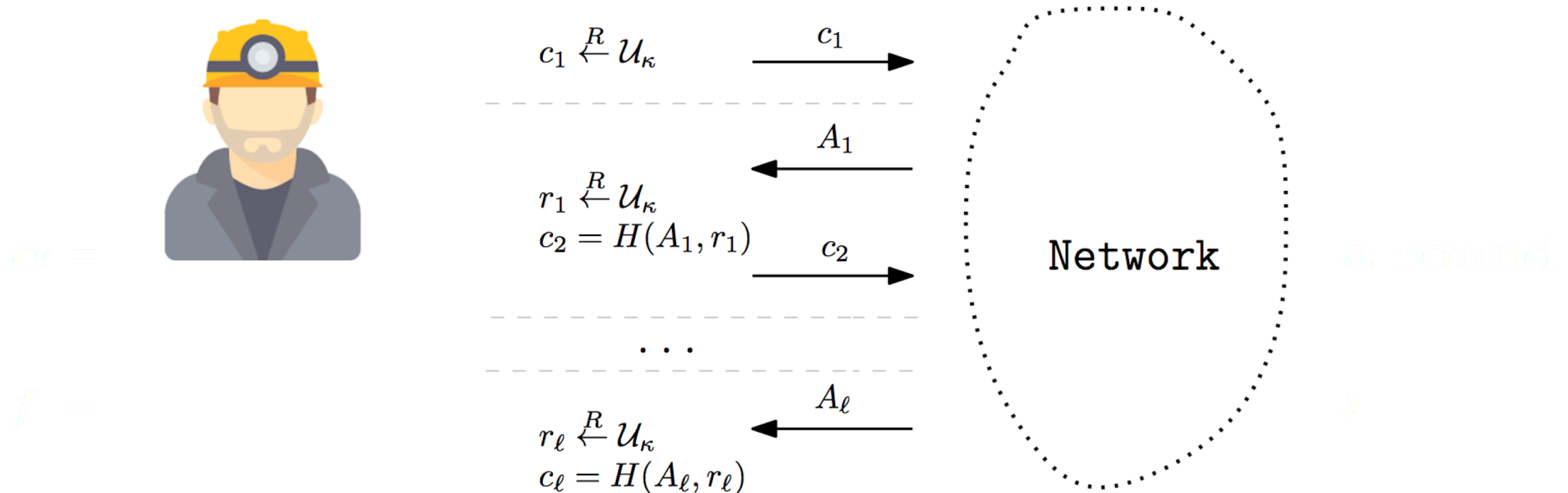
$$g(\beta) = \alpha v - \alpha v^2$$

The *Bootstrapped* Backbone Protocol (2)

- Two phases:
 - Challenge exchange phase ($O(\kappa)$ rounds)
 - Basic backbone functions [GKL15]



The *Bootstrapped* Backbone Protocol: Challenge Exch. Phase

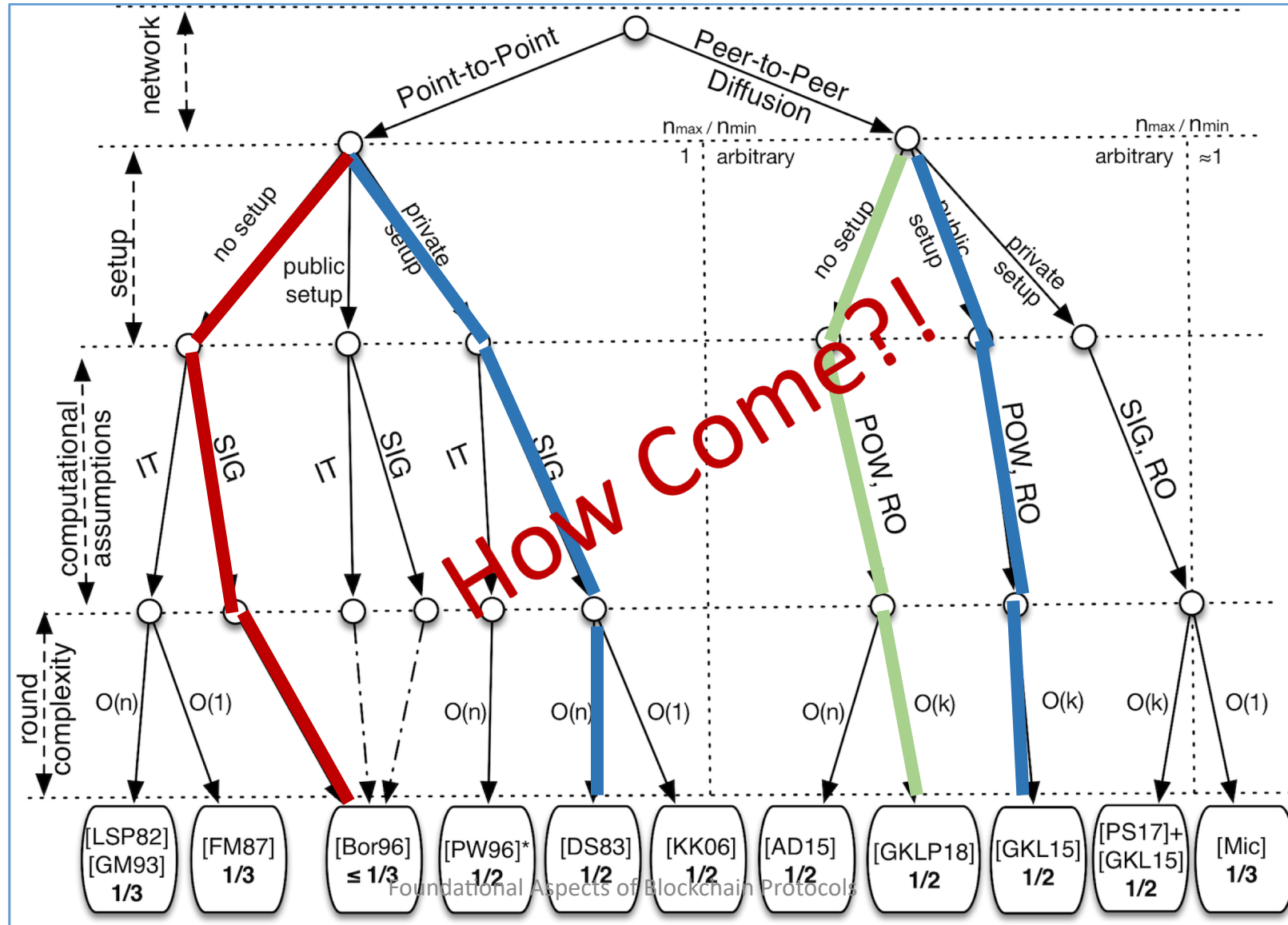


- Each honest party can generate a verifiable unpredictable string
- Disagreement is at most one round

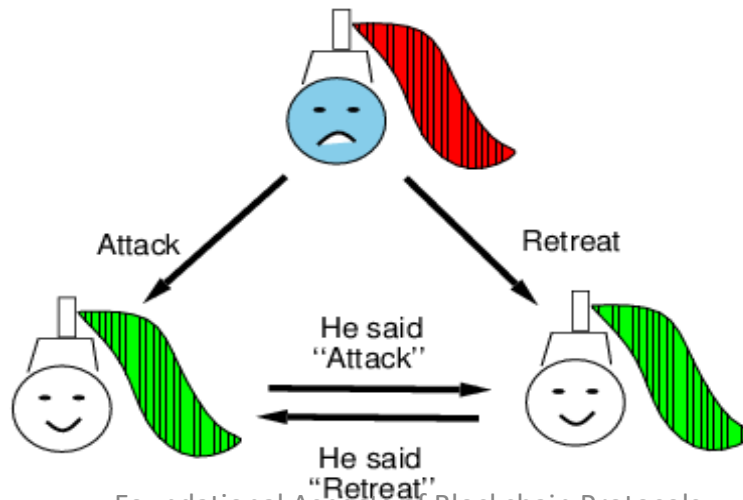
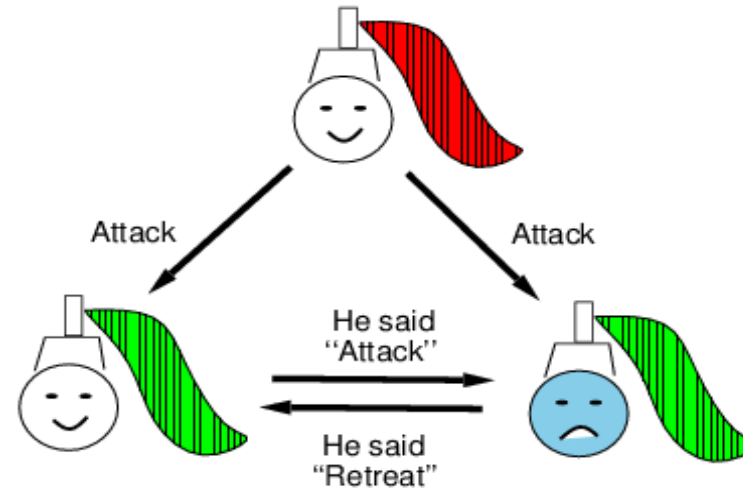
The *Bootstrapped* Backbone Protocol [GKLP18]

- *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**
 - \Rightarrow Can transition from permissionless to permissioned network

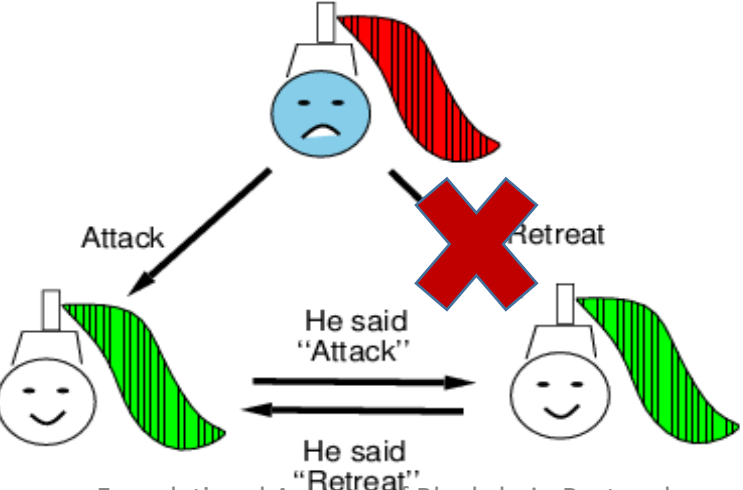
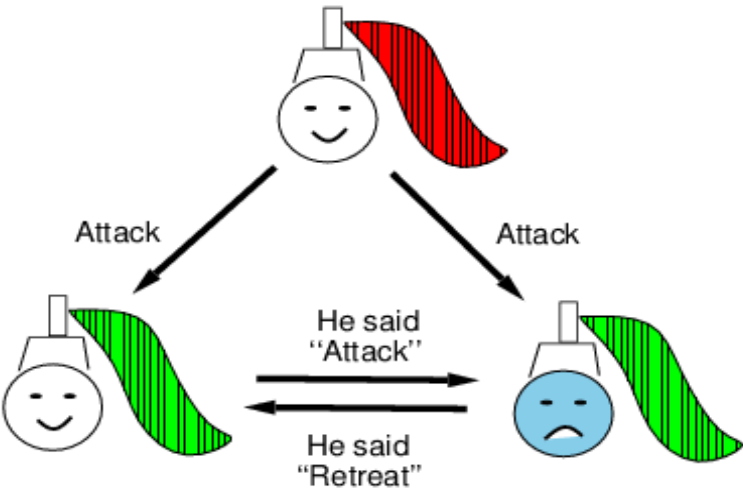
A Consensus Taxonomy [GK18]



Impossibility with $n = 3t$ [PSL80, LSP82]



Resource-restricted Cryptography [GKOPZ19]



Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

Part II

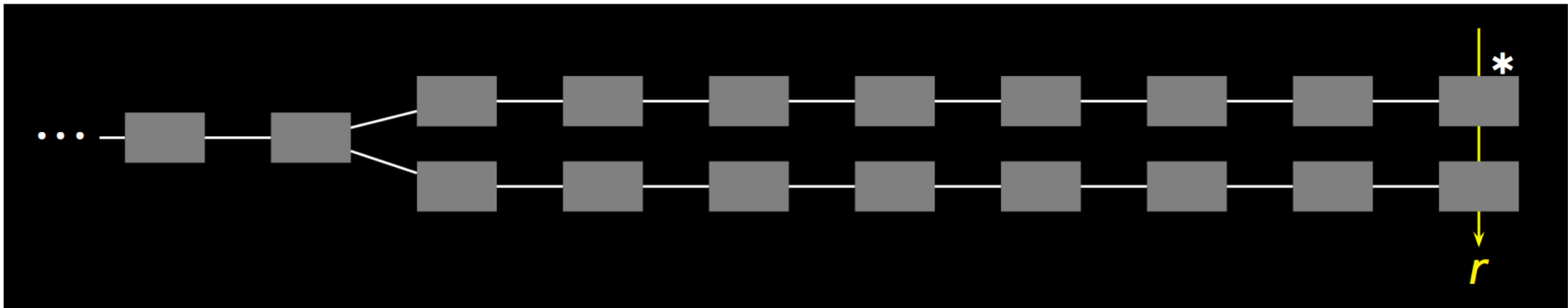
- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- **Not Covered in This Talk**
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - PoS-based blockchains
- References

Blockchains 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: $f = qnT/2^k$
- 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. Analysis extends to Δ -delay model [GKL20]

Common Prefix in the *Static* Setting

- **Common Prefix:** The probability that at a given round two honest parties have chains that disagree is at most $e^{-\Omega(\kappa)}$
- Let $S = (r^*+1, \dots, r-1)$

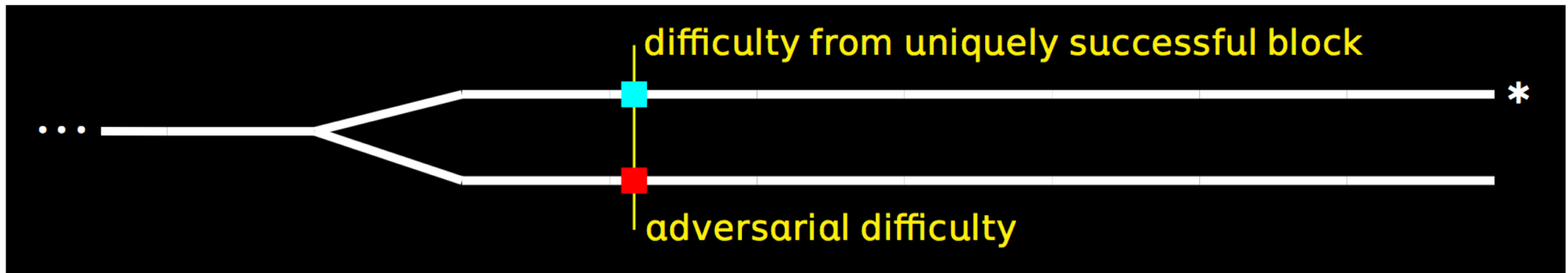


- For every uniquely successful round in S there should be an adversarial block

→ Uniquely successful rounds \leq Adversarial successes

Common Prefix in the *Dynamic* Setting (1)

- **Common Prefix:** The probability that at a given round two honest parties have chains that disagree is at most $e^{-\Omega(\kappa)}$
- Let $S = (r^*+1, \dots, r-1)$



- In sequence S :
 - Difficulty accumulated in uniquely successful rounds \leq Difficulty accumulated by the adversary

Common Prefix in the *Dynamic* Setting (2)

- In sequence S :

→ Difficulty accumulated in uniquely successful rounds \leq Difficulty accumulated by the adversary

- Same statement in static case [GKL15] is “easy,” as we are comparing two **binomial** distributions
- In the dynamic case, success probabilities are random variables that depend on the adversary’s strategy
- Conditional expectations

→ **Martingales**

Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

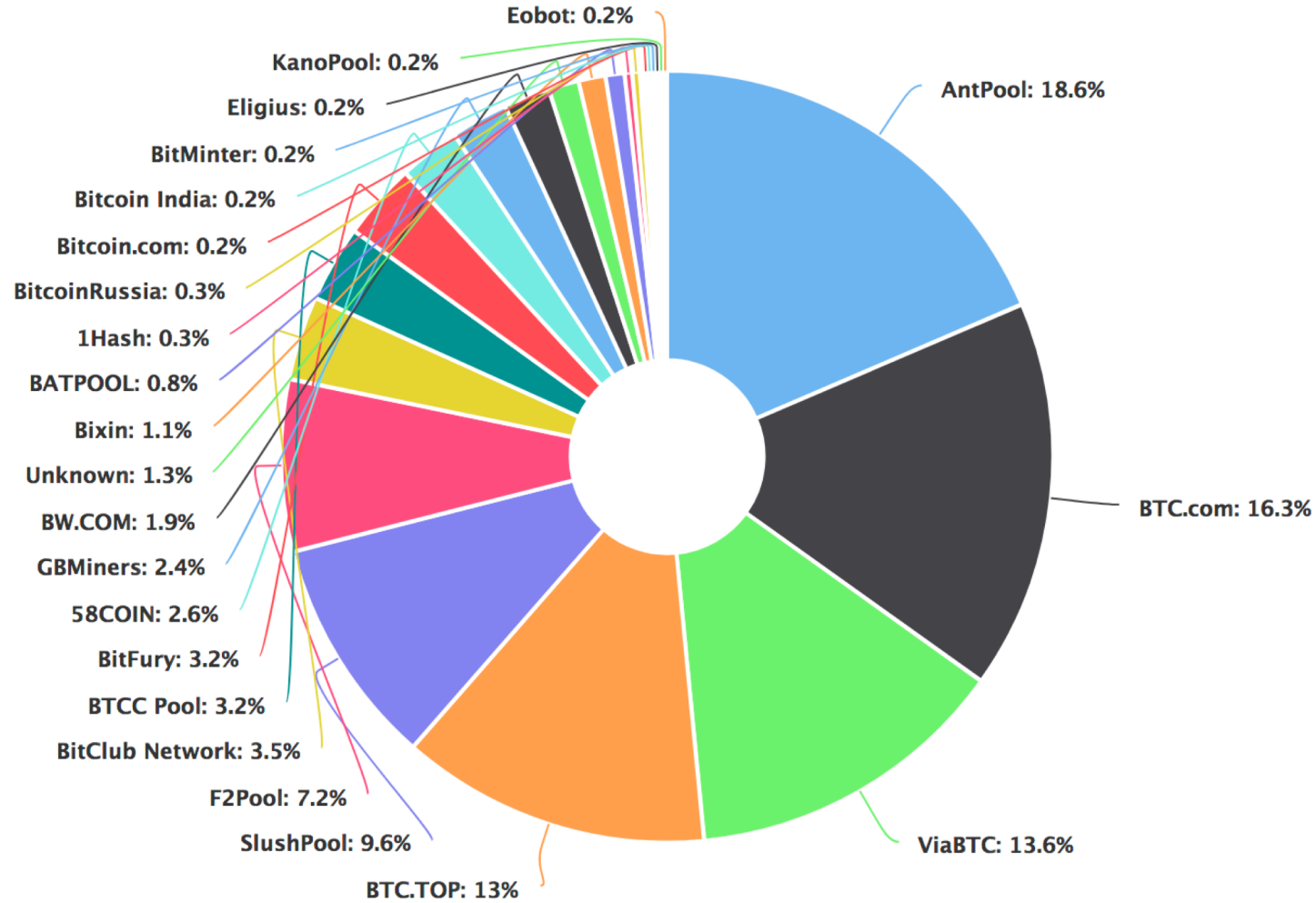
Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - **But why does it work? A Rational Protocol Design analysis**
 - PoS-based blockchains
- References

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

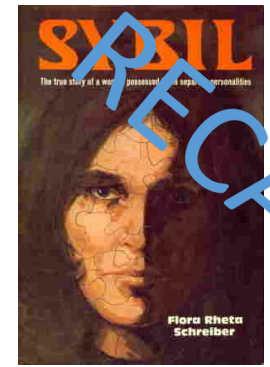
Talk Plan

Part I

- Introduction
- What is a Blockchain?
- A Blockchain Abstraction
 - The Bitcoin *backbone* protocol
- Basic Properties of the Blockchain
 - Common Prefix, Chain Quality, Chain Growth

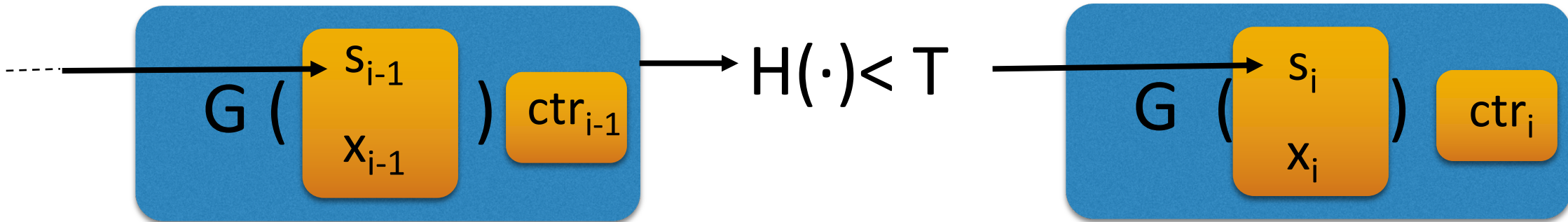
Part II

- Applications
 - Consensus
 - Robust transaction ledger
- Is a Genesis Block Really Needed?
- Not Covered in This Talk
 - Blockchains of variable difficulty
 - But why does it work? A Rational Protocol Design analysis
 - **PoS-based blockchains**
- **References**



Proofs of Work [DN92,...]

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



- This talk's focus: PoW
 - Proof-of-* (* = stake, space/memory,...); more later

Proofs of Stake

- **Stake:** A virtual resource
- A randomized process that takes it into account to *elect* the next eligible miner to produce a block
- Much more energy-efficient protocols due to the removal of PoWs
- PoS approaches:
 - PoS blockchains (NXT [BCNext], Ouroboros [KRDO17,...], Snow White [DPS19])
 - PoS BFT: Adapt classical BFT protocols to work in the PoS setting (Algorand [Mic16, CM19])
- *Not* resource-restricted [GKOPZ19]: “Costless simulation,” “long-range” attacks

Proofs of Stake: Basic Primitives

- *Verifiable Random Functions* [MRS99]
 - Three algorithms: KeyGen, Eval, Verify
 - $\text{KeyGen}(1^k) \rightarrow (\text{vk}, \text{sk})$
 - $\text{Eval}(\text{sk}, m) \rightarrow (y, \pi)$
 - $\text{Verify}(\text{vk}, m, y, \pi) \rightarrow \{0, 1\}$
 - Key properties: Can't be adversarially biased and remains pseudorandom; can be locally evaluated and universally verified
- Key evolving signatures
 - Can update the signing key and keep the verification key
 - Erasure model
- Beacon
 - As stakes shift with transactions blockchains, beacon guarantees fresh randomness enters the system

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, “SoKA Consensus Taxonomy in the Blockchain Era.” To appear in *CT-RSA 2020*. Full version at Cryptology ePrint Archive: Report 2018/754, <http://eprint.iacr.org/2018/754>.

References (2)

- J. Garay, A. Kiayias and G. Panagiotakos, “Consensus from Signatures of Work.” To appear in *CT-RSA 2020*. Full version at Cryptology ePrint Archive: Report 2017/775, <http://eprint.iacr.org/2017/775>
- J. Garay, A. Kiayias and G. Panagiotakos, “Iterated Search Problems and Blockchain Security under Falsifiable Assumptions. Cryptology ePrint Archive: Report 2019/315, <http://eprint.iacr.org/2019/315>
- J. Garay, A. Kiayias, R. Ostrovsky, G. Panagiotakos and V. Zikas, “Resource-Restricted Cryptography: Honest-Majority MPC from a CRS (and No Broadcast).” Cryptology ePrint Archive: Report 2019/1264, <http://eprint.iacr.org/2019/1264>
- J. Garay, A. Kiayias and N. Leonardos, “Chains of Variable Difficulty in the Bounded-Delay Model.” In preparation

Thank You