Scaling Bitcoin Securely

Aggelos Kiayias
University of Edinburgh

based on joint work with Juan Garay, Nikos Leonardos, Giorgos Panagiotakos

Analyzing the Bitcoin Protocol

- Nakamoto : adversary vs. honest player working on a chain perform a random walk.

- Assuming honest-majority the adversary cannot "catch" the honest players.

- Nakamoto's analysis can be easily seen to be limited:

  - the adversary can be more creative than just mining in private until he obtains a longer chain. E.g., it can broadcast conflicting chains to different sets of honest miners in order to split their mining power.

The Bitcoin Backbone : analysis and applications

[Eurocrypt 2015, joint work with J. Garay, N. Leonardos]

- Formal model.
  - Instead of arguing security against specific attacks argue security against all possible attackers in the model.
  - State general properties that should be satisfied.

- The bitcoin *backbone* :
  the generic blockchain protocol derived from bitcoin

GKL Model

- A general framework for arguing formally about bitcoin-like protocols.
  - in the tradition of synchronous distributed systems modeling.
  - Stand alone, synchronous execution.
  - Static number of parties.
  - Extensible to the dynamic / composition setting.

The model : *q*-bounded synchronous setting

- Synchronous operation: time is divided in rounds.

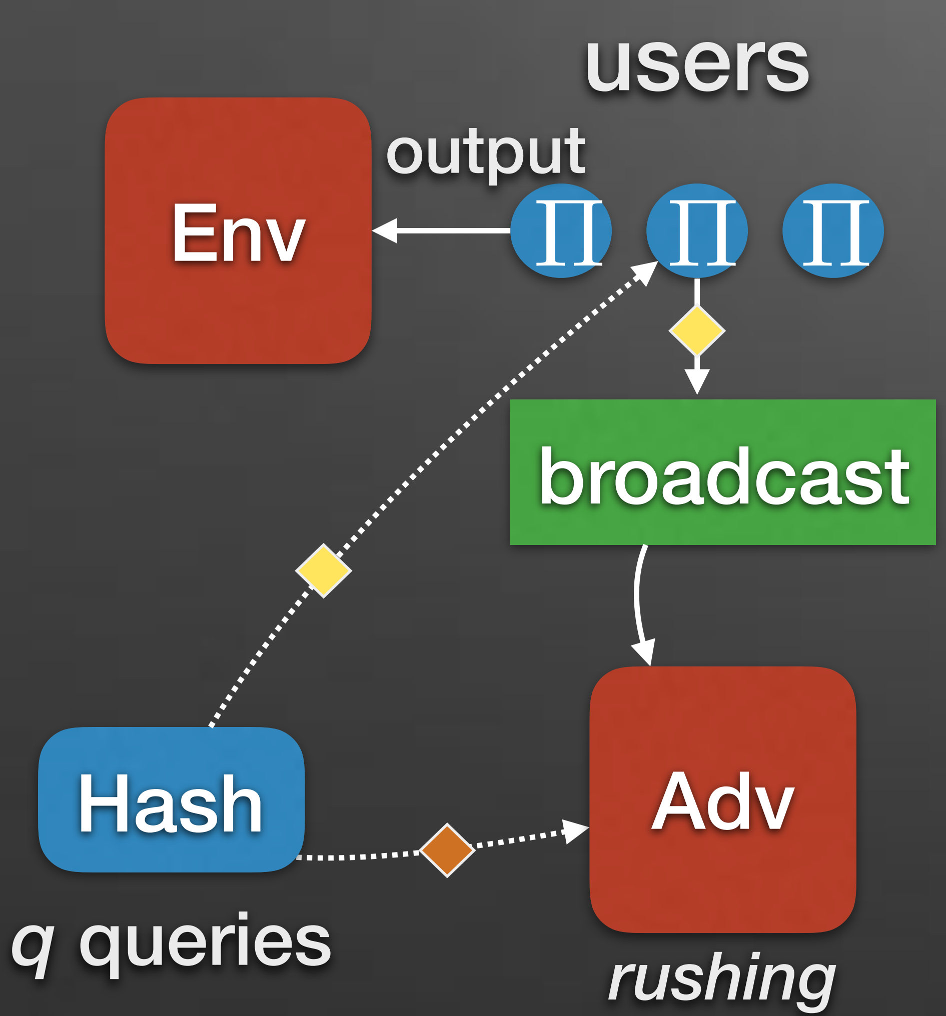
  n parties, *t* of which controlled by the adversary.

- In each round
  each player is allowed *q* queries to a hash function
- messages are sent through a diffusion mechanism

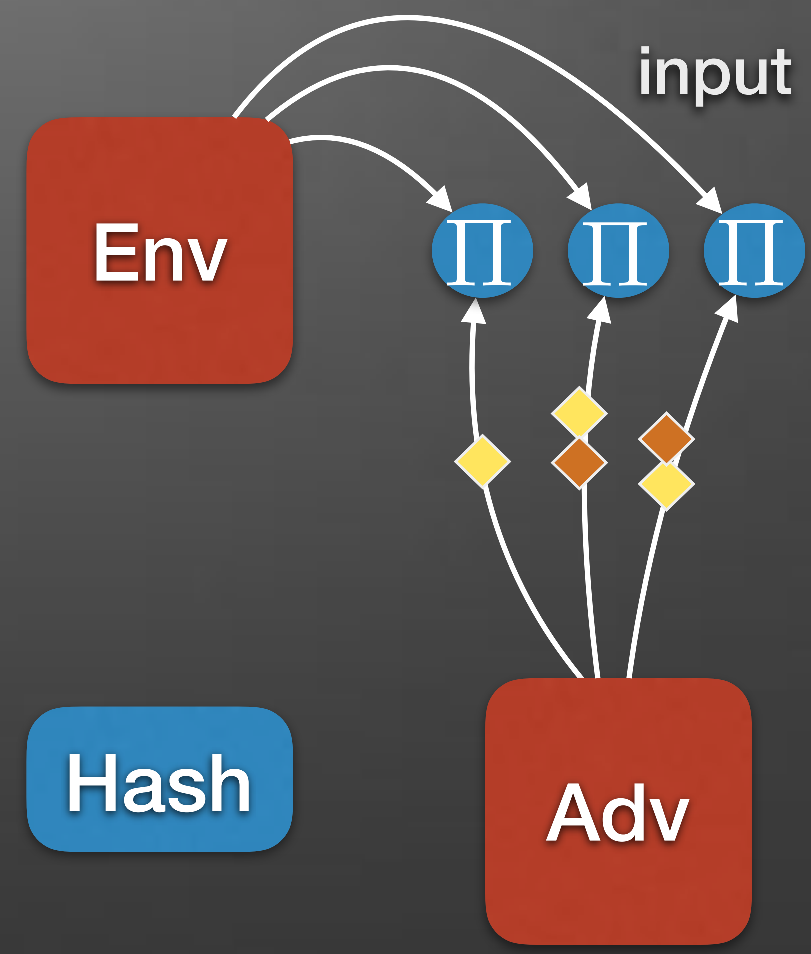
- Adversary may :
  1. spoof messages
  2. generate arbitrary number of messages

Round structure

end of round *i*



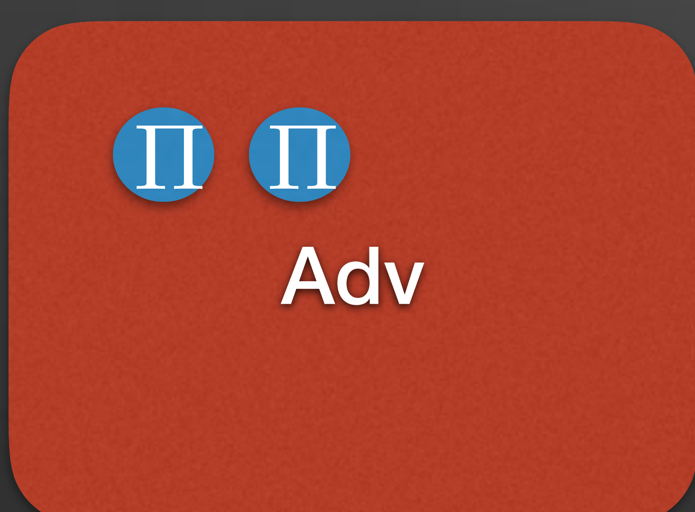
beginning of round *i+1*



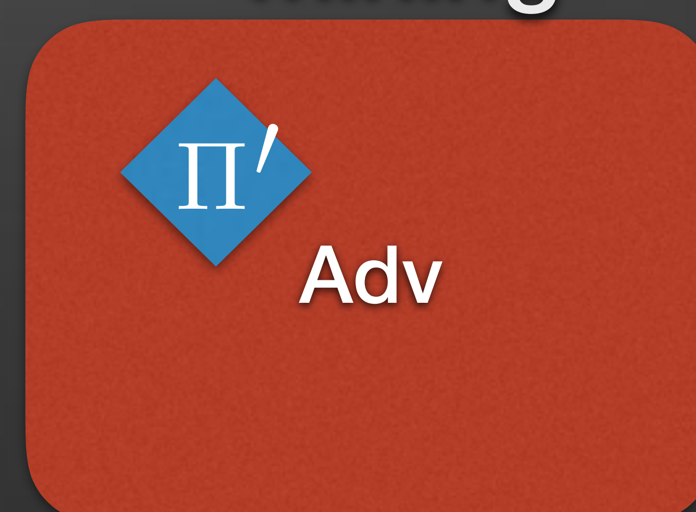
On the generality of the model

- We quantify over all possible adversaries. <u>This includes</u>

some parties receiving only some of the messages



a large mining pool that is performing some type of selfish mining



<u>Or any combination thereof</u>

On the generality of the model

- There are *n-t* honest parties each one receiving *q* queries to the hash function per round.

- The adversary is able to control *t* parties acting as a malicious mining pool.

  - A "flat" version of the world in terms of hashing power.

  - It is worse for honest parties to be separated (they have to pay the price of being decentralized).

Modeling the hash function

- Hash Function = [Random oracle]
  - State = Table T
  - Given any query *x* look up T for pair of the form (x,y)
    - If it does not exist sample y from {0,1}^λ and store (x,y) to T
    - Return y

λ = security parameter

Execution & View

3 PPT machines

protocol Π

adversary $\mathcal{A}$

environment $\mathcal{Z}$

n parties

$\text{VIEW}^{\Pi}_{\mathcal{A},\mathcal{Z}}(1^{\lambda})$ concatenation of the view of each party at each round

random variable with support :

1. coins of $\mathcal{A}, \mathcal{Z}, n$ copies of Π
2. Random oracle

Property of a protocol

fix
a protocol Π
a number of parties *n, t* of which controlled by adversary
a predicate Q

We say that the protocol has property Q with error ϵ if and only if

$$\forall \mathcal{A}\ \forall \mathcal{Z}\ \mathsf{Prob}[Q(\mathsf{VIEW}^{\Pi}_{\mathcal{A},\mathcal{Z}}(1^{\lambda})] \geq 1-\epsilon$$

typically : $\epsilon = \mathsf{negl}(\lambda)$

Sanity check: why use the bitcoin protocol?

Classical results in distributed systems :

Lamport, Shostak Pease '80

- No authentication infrastructure *n,t* are unknown

hence known consensus algorithms cannot be applied

Sanity check: why use the bitcoin protocol?

Classical results in cryptography :

Goldreich Micali Wigderson 1987
any function can be securely computed by *n* parties.
Is this applicable to the bitcoin setting ?

- No authentication infrastructure *n*,*t* are unknown

hence “secure MPC” cannot be applied

Precursors from a consensus point of view

- Aspnes-Jackson-Krishnamourthy 2005. Suggest use of POW to establish PKI (from which one may obtain broadcast (the byzantine generals) and then consensus)

- Okun 2005. Defines anonymous consensus (but no POW - no efficient algorithm).

Bitcoin Backbone

- A precise algorithmic description of the core of the bitcoin protocol that isolates its consensus characteristics in a precise manner (while it abstracts away the transactional aspects)

The Bitcoin Backbone (1)

parameterized by $V(\cdot), I(\cdot), R(\cdot)$
and $G(\cdot), H(\cdot)$ hash functions

- players have a state $\mathcal{C}$ in the form of a "blockchain":

$\mathcal{C}$ satisfies the predicate

$$V(\mathcal{C}) = \text{true}$$

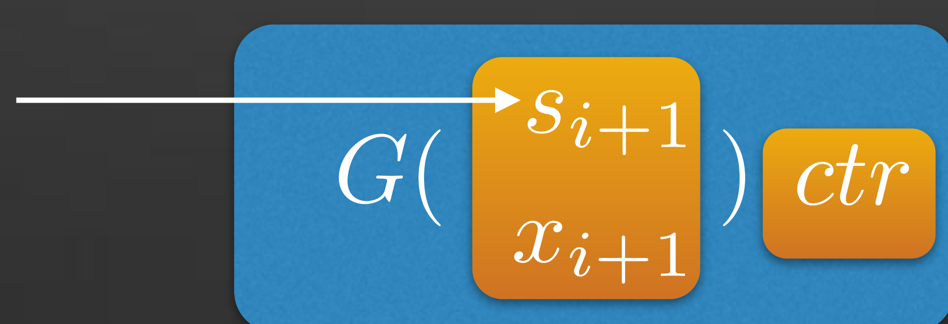
The Bitcoin Backbone (2)

parameterized by $V(\cdot), I(\cdot), R(\cdot)$
and $G(\cdot), H(\cdot)$ hash functions

- Within a round, players obtain (INSERT, x) symbols from the environment and network and process them

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

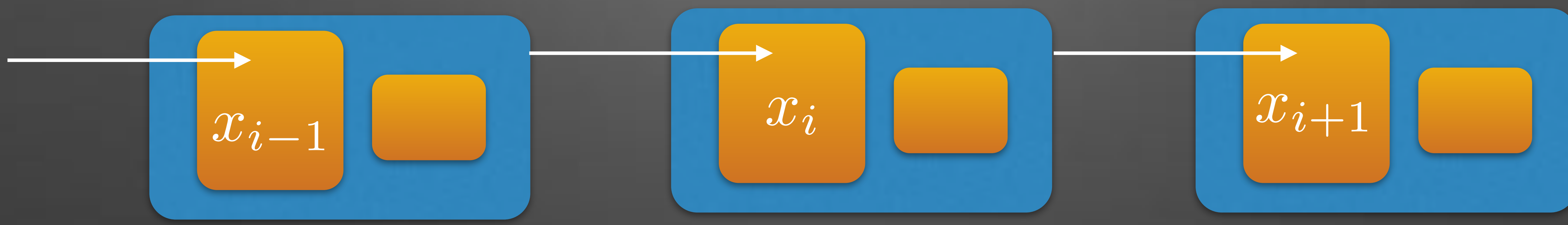
- Then they use their *q* queries to $H(\cdot)$ to obtain a new block by trying $ctr = 0, 1, 2, \ldots$



The Bitcoin Backbone (3)

parameterized by $V(\cdot), R(\cdot), I(\cdot)$

- If a player finds a new block it extends $\mathcal{C}$

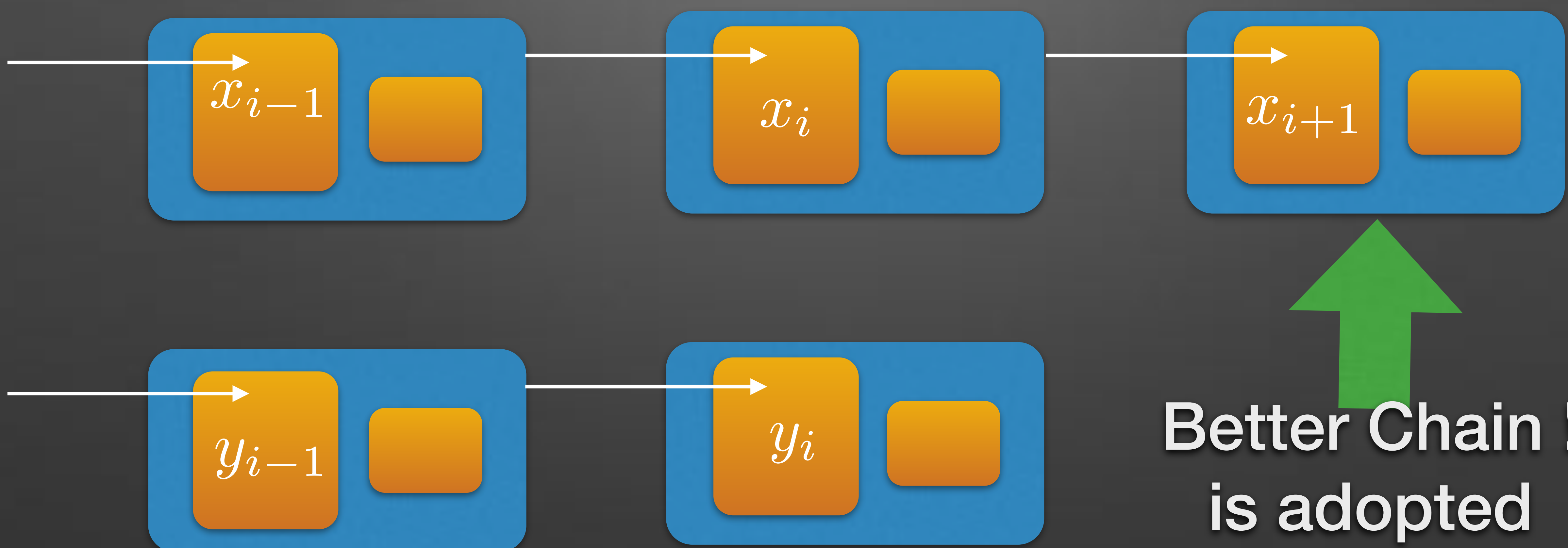


- The new $\mathcal{C}$ is propagated to all players via the (unreliable/anonymous) broadcast

The Bitcoin Backbone (4)

parameterized by $V(\cdot), R(\cdot), I(\cdot)$

- A player will compare any incoming chains and the local chain w.r.t. their length/difficulty



- Finally a player given a (Read) symbol it will return

$$R(x_1, x_2, \ldots, x_{i+1})$$

Input entropy

$$H(ctr, G(s, x)) < D$$

- Simplifying assumption: I(.) chooses a random nonce as part of *x.*
- Subsequently, function *G* maps the random nonces to their hashes.

the parties choose the same random nonce twice, has probability <= $\binom{q_{\text{total}}}{2} 2^{-\lambda}$

G(.) maps those values to the same one (collision) <= $\binom{q_{\text{total}}}{2} 2^{-\lambda}$

Pseudocode : Validate

function validate($\mathcal{C}$)
  $b \leftarrow V(\mathbf{x}_{\mathcal{C}}) \wedge (\mathcal{C} \neq \varepsilon)$
  if $b =$ True **then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
    $\langle s, x, ctr \rangle \leftarrow$ head($\mathcal{C}$)
    $s' \leftarrow H(ctr, G(s, x))$
    repeat
      $\langle s, x, ctr \rangle \leftarrow$ head($\mathcal{C}$)
      if $\mathsf{validblock}_q^D(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$ **then**
        $s' \leftarrow s$ ▷ Retain hash value
        $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove the head from $\mathcal{C}$
      else
        $b \leftarrow$ False
      end if
    until $(\mathcal{C} = \varepsilon) \vee (b =$ False$)$
  end if
  return (b)
end function

Pseudocode : POW

function pow(x, $\mathcal{C}$)
    if $\mathcal{C} = \varepsilon$ **then** ▷ Determine proof of work instance
        $s \leftarrow 0$
    else
        $\langle s', x', ctr' \rangle \leftarrow \mathrm{head}(\mathcal{C})$
        $s \leftarrow H(ctr', G(s', x'))$
    end if
    $ctr \leftarrow 1$
    $B \leftarrow \varepsilon$
    $h \leftarrow G(s, x)$
    while $(ctr \leq q)$ **do**
        if $(H(ctr, h) < D)$ **then** ▷ This $H(\cdot)$ invocation subject to the q-bound
            $B \leftarrow \langle s, x, ctr \rangle$
            break
        end if
        $ctr \leftarrow ctr + 1$
    end while
    $\mathcal{C} \leftarrow \mathcal{C}B$ ▷ Extend chain
    return $\mathcal{C}$
end function

Pseudocode : main loop

$\mathcal{C} \leftarrow \varepsilon$
$st \leftarrow \varepsilon$
$round \leftarrow 0$
while TRUE **do**
    $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C},$ any chain $\mathcal{C}'$ found in RECEIVE())
    $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round,$ INPUT(), RECEIVE()) ▷ Determine the x-value.
    $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
    if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
        BROADCAST($\mathcal{C}$)
    end if
    $round \leftarrow round + 1$
    if INPUT() contains READ **then**
        write $R(\mathcal{C})$ to OUTPUT()
    end if
end while

Requirements.

Input Validity : function I(.) produces inputs satisfiable by V(.)
Input Entropy : function I(.) will not produce the same x value wit
overwhelming probability

Let's prove a property!

During any period from round *r* to $s > r + \lambda$ the chain of an honest party will grow by at least $0.9\gamma\lambda$ blocks

where $\alpha = pq(n-t)$

$p = \frac{D}{2^\lambda}$ probability a single query to be successful

D = corresponds to difficulty of producing a block

with error $\epsilon = \text{negl}(\lambda)$

Proof - Step 1

- Two honest parties, a,b, submit a query to the RO.
    - Let A, B be the events that the respective party finds a hash value less than difficulty threshold *D*.
    - Conditioning on the event that the G(.) values of the two parties are *distinct (no G collision - no repetition of x-values)*, the events A, B are independent.

Proof - Step 2

Given independence :

The probability at least one honest party finds a solution in a single round:

$$1-(1-p)^{q(n-t)} \geq 1-e^{-\alpha} \geq \gamma = \alpha - \alpha^2$$

we call this a

“successful round”

Proof - Step 3

Define a random variable X_i

$$X_i = \begin{cases} 1 & i\text{-th round is successful} \\ 0 & \text{otherwise} \end{cases}$$

Facts

$$\forall i \ \mathsf{Prob}[X_i] \geq \alpha - \alpha^2$$

$$i \neq j \rightarrow$$

$$\mathsf{Prob}[X_i = 1 \wedge X_j = 1] = \mathsf{Prob}[X_i = 1] \cdot \mathsf{Prob}[X_j = 1]$$

Proof - Step 4

- Lemma. At any round *r*, consider an honest party with a chain of length L. By round *s >= r* every honest party has adopted a chain of length at least

$$L + \sum_{i=r}^{s-1} X_i$$

Proof. By induction on *s-r = i*

Base. *i = 0*

Indeed, if the party has a chain of length *L*>0 at round *r,* this means that at a previous round it has broadcasted it. It follows that other honest parties by round *r* either have adopted either this one (or an equally long chain).

Proof - Step 5

Induction Step. Suppose it holds for *i,* we show for *i*+1. By round *s*-1 every honest party has received a chain of length

$$L + \sum_{i=r}^{s-2} X_i$$

if $X_{s-1} = 0$ the result follows immediately

if $X_{s-1} = 1$ we have that *s*-1 is a successful round thus at the end of the round at least one honest party broadcasts a chain of length

$$1 + L + \sum_{i=r}^{s-2} X_i = \sum_{i=r}^{s-1} X_i$$

Proof - Step 6

$$X = \sum_{i=r}^{s} X_i$$ is a Binomial distribution

$$\mu = E[X] \geq \gamma(s-r)$$

Tail bounds for Binomial distribution (Chernoff)

$$\forall \delta \in (0,1] \qquad \text{Prob}[X \leq (1-\delta)\mu] \leq e^{-\delta^2 \mu/2}$$

Corollary. $\text{Prob}[X \leq (1-\delta)\gamma(s-r)] \leq e^{-\delta^2 \gamma(s-r)/2}$

Proof - Step 7

- It follows that from round *r* to round *s* all honest parties will grow their chain by $$\sum_{i=r}^{s-1} X_i$$

  which is at least

  $$(1-\delta)\gamma(s-r-1) \geq 0.9\gamma\lambda$$

  with probability $1 - e^{-\delta^2\gamma(s-r-1)/2} \geq 1 - e^{-0.005\gamma\lambda}$

We set $\delta = 0.1$

$s > r + \lambda$

QED

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)

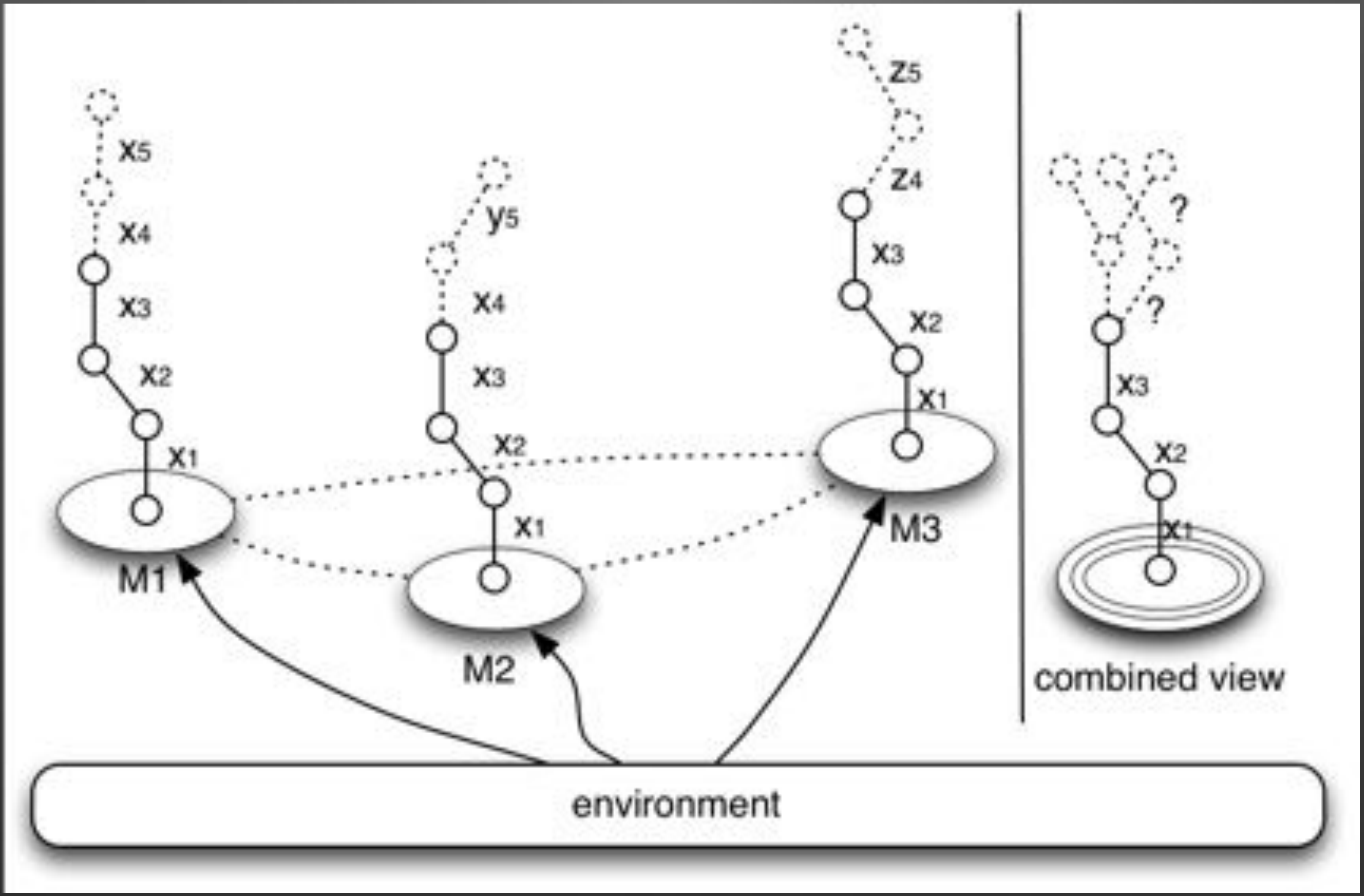
Any (large enough) chunk of an honest player's chain will contain some blocks from honest players

Chain Growth

(informally)

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

CP : will players converge?



Common prefix property

- (Common-prefix) no matter the strategy of the adversary, the chains of two honest parties will fork in the last *k* blocks with probability at most $e^{-\Omega(k)}$

Assuming: $(\alpha - \alpha^2) > \dfrac{f + \sqrt{f^2 + 4}}{2} \cdot \beta$

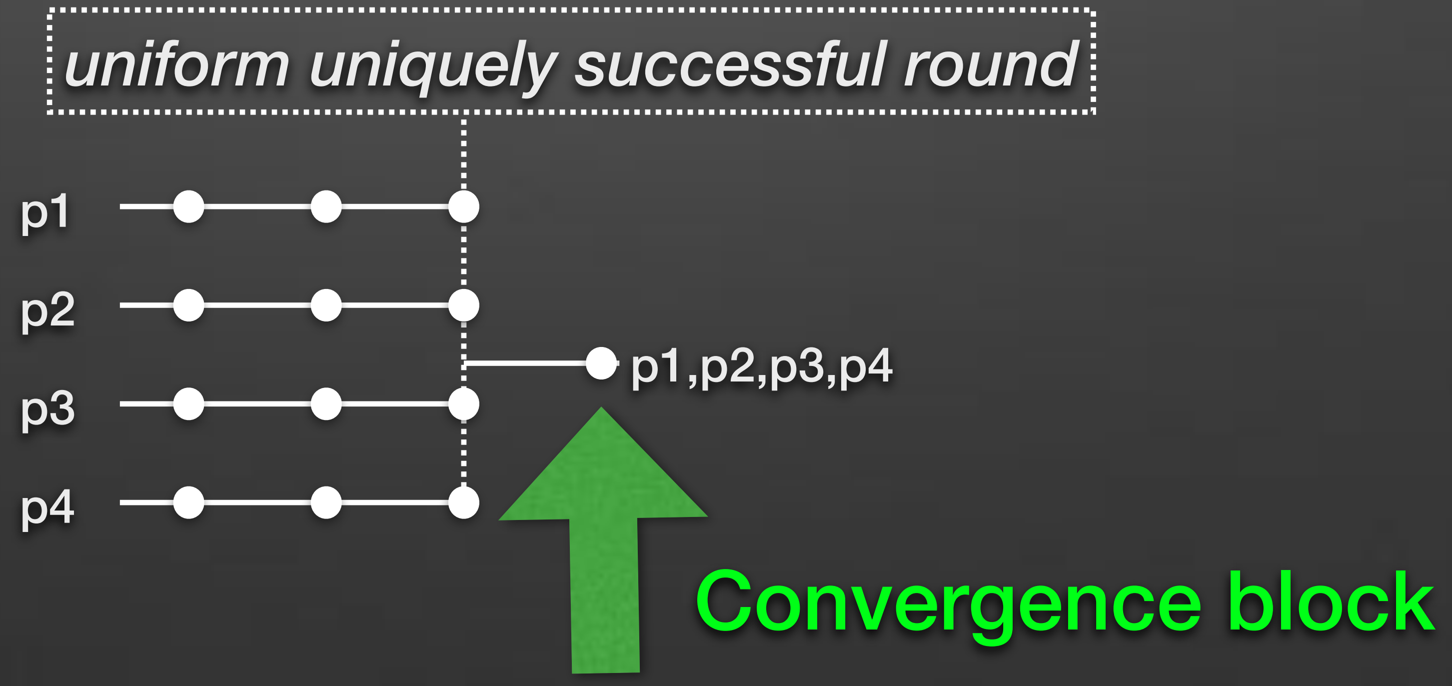
α : (as before) $= pq(n-t)$

β : expected adversarial POW's per round $= pqt$

$f = \alpha + \beta = pqn$

Common prefix property (2)

- Common-prefix theorem: (proof idea)
  - *Uniform round:* is a round where all honest parties invoke a POW with a chain of the same length.
  - *Uniquely successful round:* is a round when exactly one honest party is successful.



Common prefix property (3)

- Common-prefix theorem: (proof idea, cont.)
  - *Uniform uniquely successful rounds* allow parties to produce convergence blocks.
  - A fork that spans such convergence blocks may only exist if an adversary produces one POW for each.
  - The rate of such rounds is $(1-\beta)\gamma$

.. therefore in order for the adversary to maintain a fork for a certain length it should be $\beta > (1-\beta)\gamma$

Common prefix property (4)

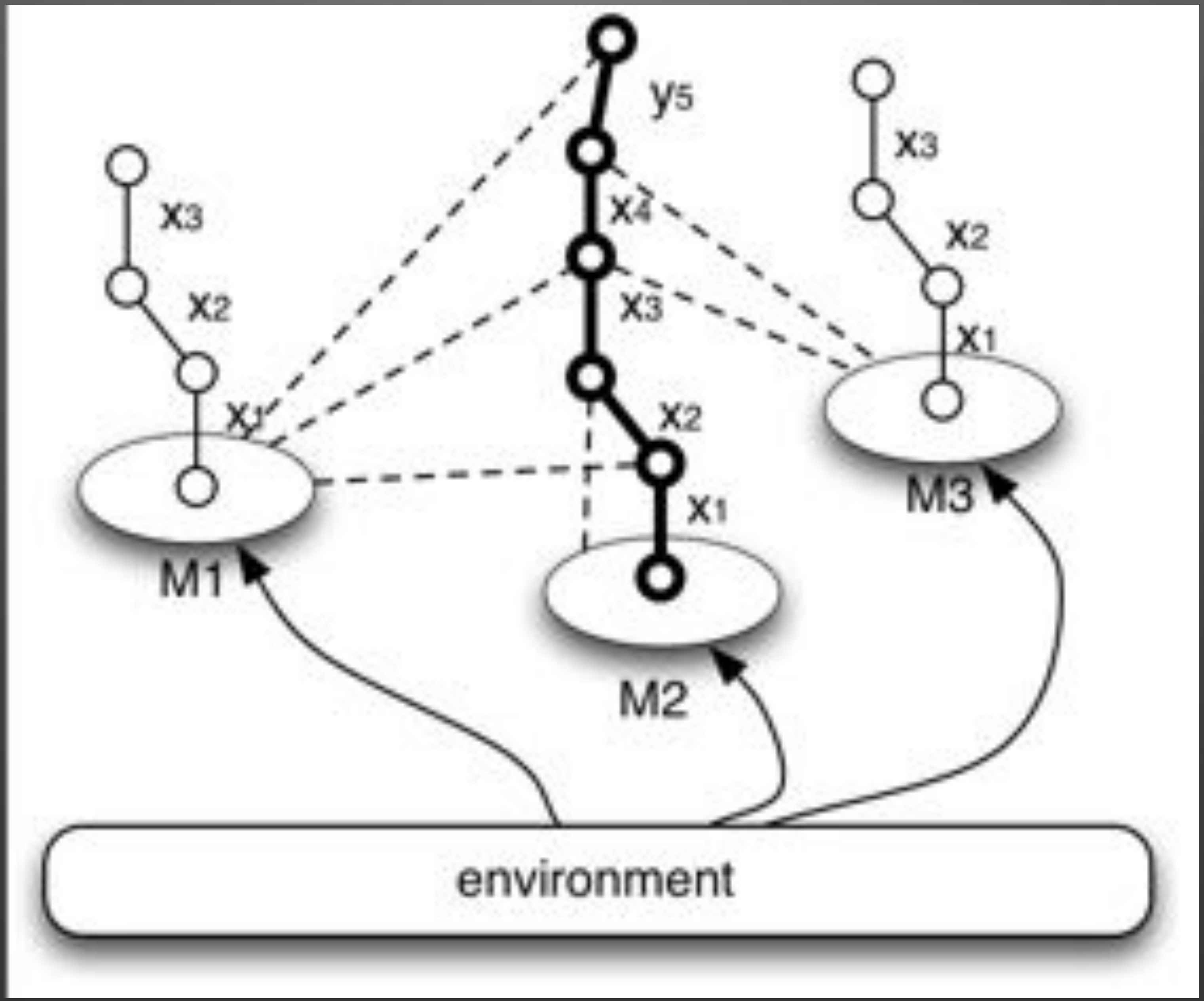
- Common-prefix theorem: (proof idea, cont.)

In order for the adversary to maintain a fork for a certain length it should be $\beta > (1-\beta)\gamma$

which we can derive from our assumption $\lambda^2 - f\lambda - 1 \geq 0$

Now if $f \to 0$ we can let $\lambda \to 1$ (adversarial tolerance up to 50%)
(fast information propagation)

CQ : will honest blocks be adopted by honest players?



Chain Quality Property (1)

- (Chain quality) any sequence of ℓ blocks in an honest party's chain will contain $1 - \frac{1}{\lambda}$ honest blocks with probability $1 - e^{-\Omega(\ell)}$

assuming: $(\alpha - \alpha^2) > \lambda\beta$ for $\lambda \geq \frac{f + \sqrt{f^2 + 4}}{2}$

α : expected honest POW's per round

β : expected adversarial POW's per round

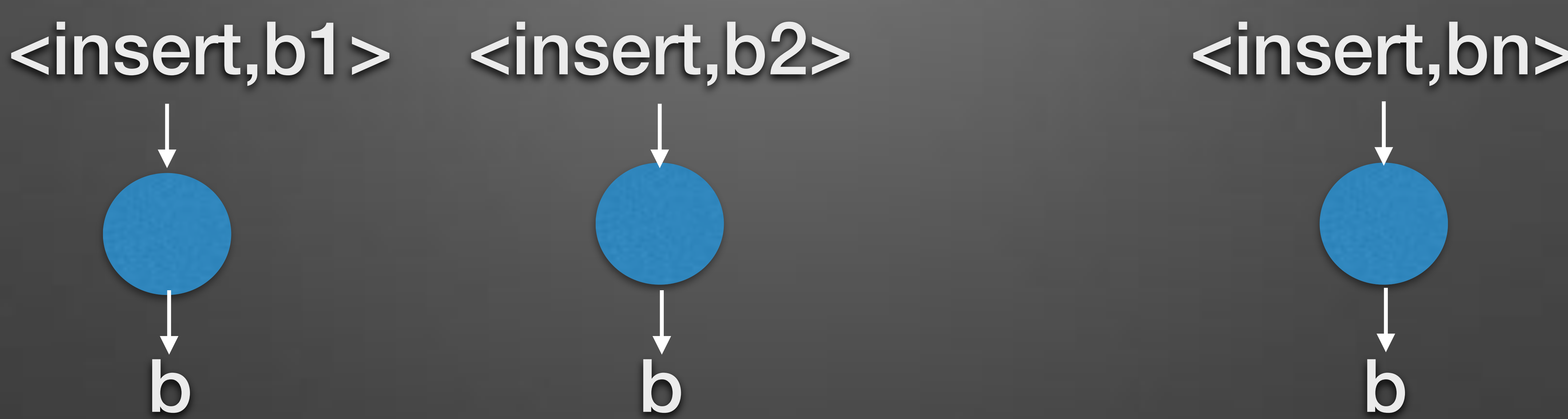
$f = \alpha + \beta$

Chain Quality Property (2)

- we show our result is tight:
  - there is an adversarial strategy that restricts the honest parties to an exactly ratio of $1 - \frac{1}{\lambda}$
  - The strategy is a type of *selfish mining* (ES14)
    - Malicious miners mine blocks in private attempting to "hit" honest parties' blocks when they become available.

The consensus problem

implementing consensus:



Agreement = all parties output the same value

Validity = if all honest parties have the same insert bit, then this matches the output

Termination = all honest parties terminate

Applying the backbone for consensus

Nakamoto consensus protocol

Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_n \rangle)$ is true if and only if it holds that $v_1 = \ldots = v_n \in \{0,1\}, \rho_1, \ldots, \rho_n \in \{0,1\}^\kappa$ where $x_i = \langle v_i, \rho_i \rangle$.
Chain reading function $R(\cdot)$ (parameterized by k)	If $V(x_\mathcal{C}) = \text{True}$ and $\text{len}(\mathcal{C}) \geq k$, the value of $R(\mathcal{C})$ is the (unique) value v that is present in each block of $\mathcal{C}$, while it is undefined if $V(x_\mathcal{C}) = \text{False}$ or $\text{len}(\mathcal{C}) < k$.
Input contribution function $I(\cdot)$	If $\mathcal{C} = \emptyset$ and (INSERT, v) is in the input tape then $I(st, \mathcal{C}, round, \text{INPUT}())$ is equal to $\langle v, \rho \rangle$ where $\rho \in \{0,1\}^\kappa$ is a random value; otherwise (i.e., the case $\mathcal{C} \neq \emptyset$), it is equal to $\langle v, \rho \rangle$ where v is the unique $v \in \{0,1\}$ value that is present in $\mathcal{C}$ and $\rho \in \{0,1\}^\kappa$ is a random value. The state st always remains ϵ.

Agreement works —
Validity only with constant probability

Applying the backbone for consensus

a 1/3 consensus protocol

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 $x_i = \langle v_i, \rho_i \rangle$.
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.

Agreement — Validity works but only 1/3

Robust Transaction Ledgers

- They are protocols that satisfy two properties:

  Persistence: parameter *k*. If an honest party reports a transaction tx *k* blocks deep, then tx will be always reported, in the same position, by all honest parties.

  Liveness: parameters *u*, *k*. If all honest parties attempt to insert the transaction *tx* in the ledger, then after u rounds, an honest party will report it *k* blocks deep in the ledger.

transaction processing time : *u* as a function of *k*

Applying the backbone for a transaction ledger

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 $V(\langle x_1, \ldots, x_m \rangle) =$ 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 Persistence & Liveness

Intuitively, persistence will follow from agreement and liveness from chain quality

- It can be shown that

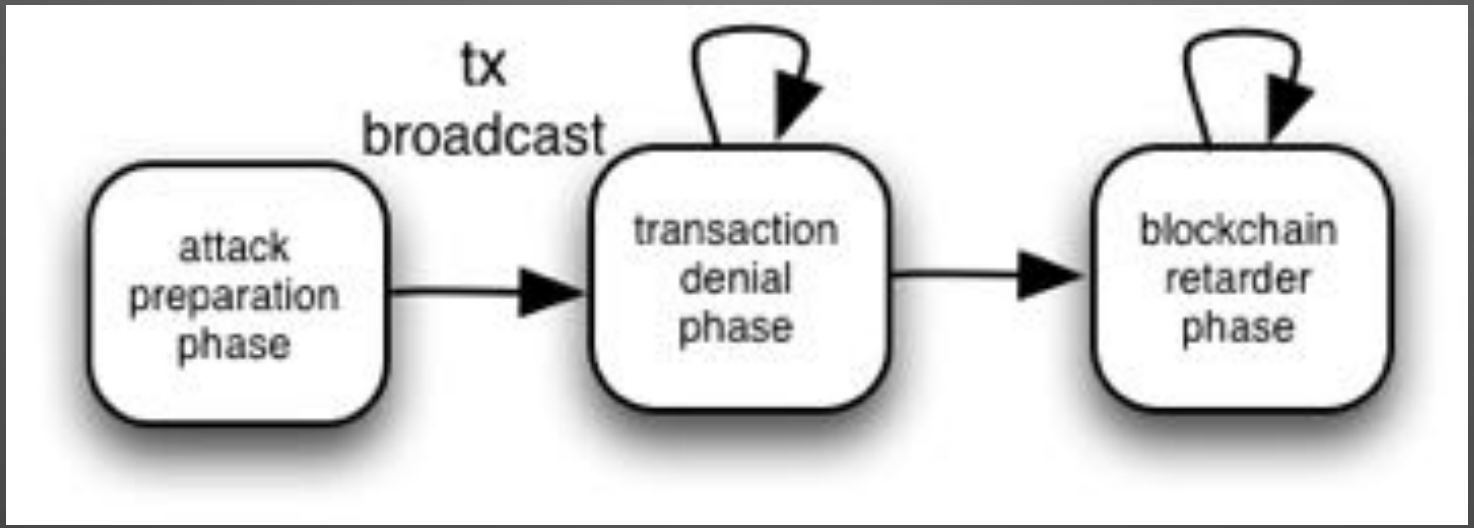
$$u = 2k/(1-\delta)(\alpha - \alpha^2)$$

with probability

$$1 - e^{-\Omega(\delta^2 k)}$$

Liveness Attacks

joint work with G. Panagiotakos



Attacker:

1. performs block-withholding releasing blocks when honest parties get ahead.
2. when the transaction appears it continues block-withholding but does not ever switch back to the main chain.

Applying the backbone for consensus

can we get to 1/2 consensus?

<u>Main obstacle (intuitively).</u>
the blockchain itself does not provide high enough validity.

This is due to low chain quality: we cannot guarantee that we have enough blocks stemming from honest parties in the blockchain

Overcoming the validity problem [GKL15]

- The *n* parties build a ledger but now generate transactions based on POW that contain their inputs.
- Once the blockchain is long enough the parties' prune the last *k* blocks and output the majority of the values drawn from the set of transactions in the ledger.

Beware! given that POW's are used for two different tasks how do we prevent the attacker from shifting its hashing power from the one to the other?

2-for-1 POWs

Composition of POW-based protocols

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

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

given $((s,x), ctr)$
verify:
$$H(G(s,x), ctr) < T$$
given $((s',x'), ctr')$
verify:
$$H(G(s',x'), ctr') < T'$$

Not Secure

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

if $[w]^{\mathsf{R}} < T \ldots$

given
$$((*,*),(s',x'), ctr')$$
verify:
$$[H(G(*,*), G(s',x'), ctr')]^{\mathsf{R}} < T'$$

The Dynamic Setting

- Consider the sequence:

$$\{n_r\}_{r\in\mathbb{N}} \quad n_1, n_2, \ldots$$

Generalized environment : in each round some parties are activated

When a party is activated it sends a special join message. It joins by the next round and starts mining. The number of mining parties in each round is n_r

Block Difficulty

- The probability of winning a block is determined by the target value *D*.

- Bitcoin uses SHA256 as the hashing algorithm.

    - Solving the challenge requires an expected number of $2^{256}/D$ steps.

difficulty calibration: aims at producing a block per 10 minutes. Each 2016 blocks, timestamps are taken into account and target *D* is calibrated.

Security in the dynamic setting

- Are basic properties (common prefix, chain quality) maintained in the dynamic setting?
- Not for arbitrary $\{n_r\}_{r\in\mathbb{N}}$
- The calibration mechanism itself can be attacked.
  - [Bahack'13]

Blockchain protocol variants

- Randomized Select.
- GHOST
- Bitcoin-NG
- Lightning Network

Randomized select

[ES14]

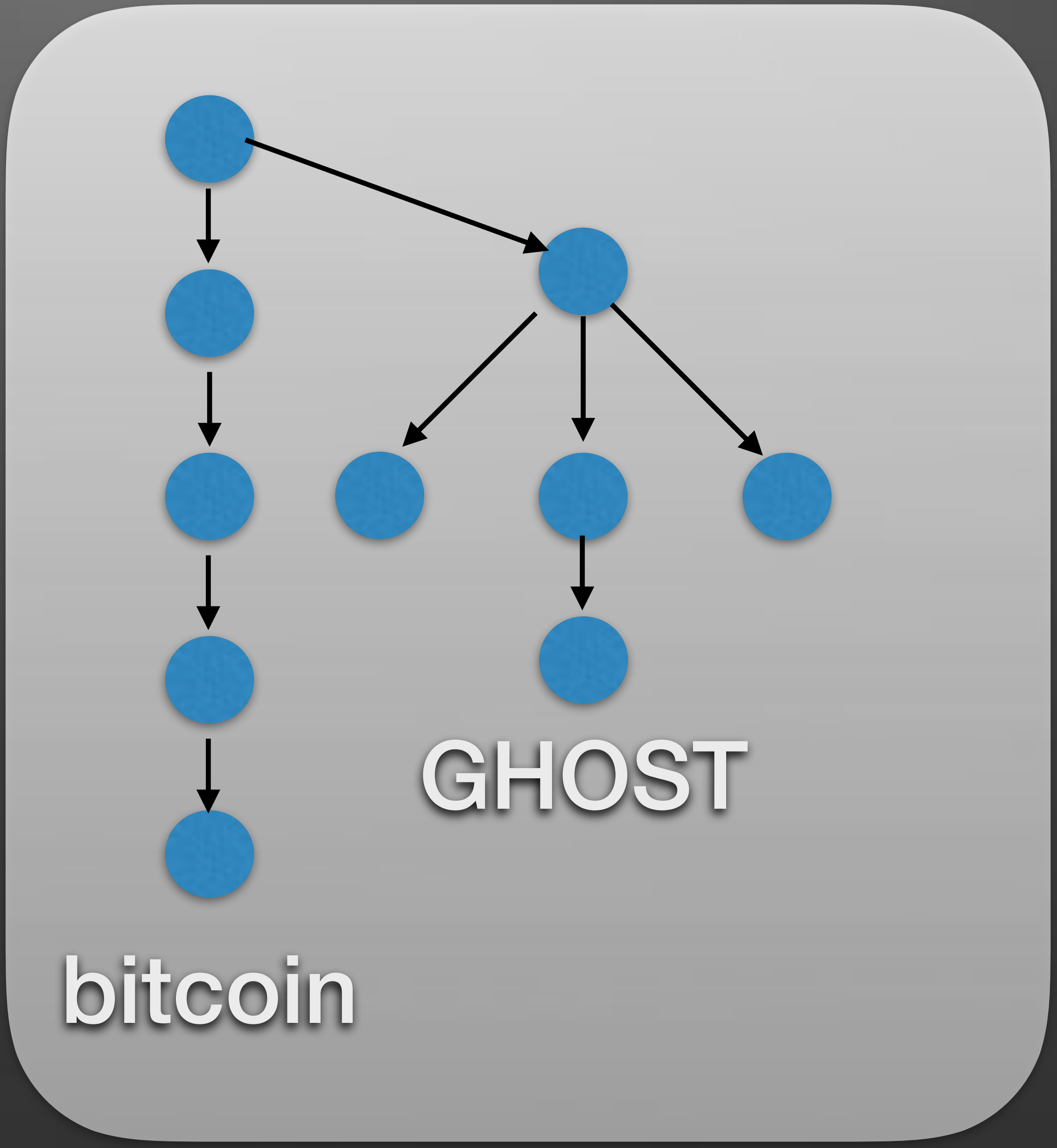
- A suggestion to modify the chain selection rule:
- Nodes choose randomly between chains received very close to each other.
- Intention : neutralize attacks in the selfish mining domain where the attacker persistently knocks out honest blocks (due to e.g., network propagation superiority).
- This makes a difference in the analysis in our framework (as our rushing adversary has by default network propagation superiority).

GHOST

[SZ13]

- A suggestion to modify the chain selection rule:

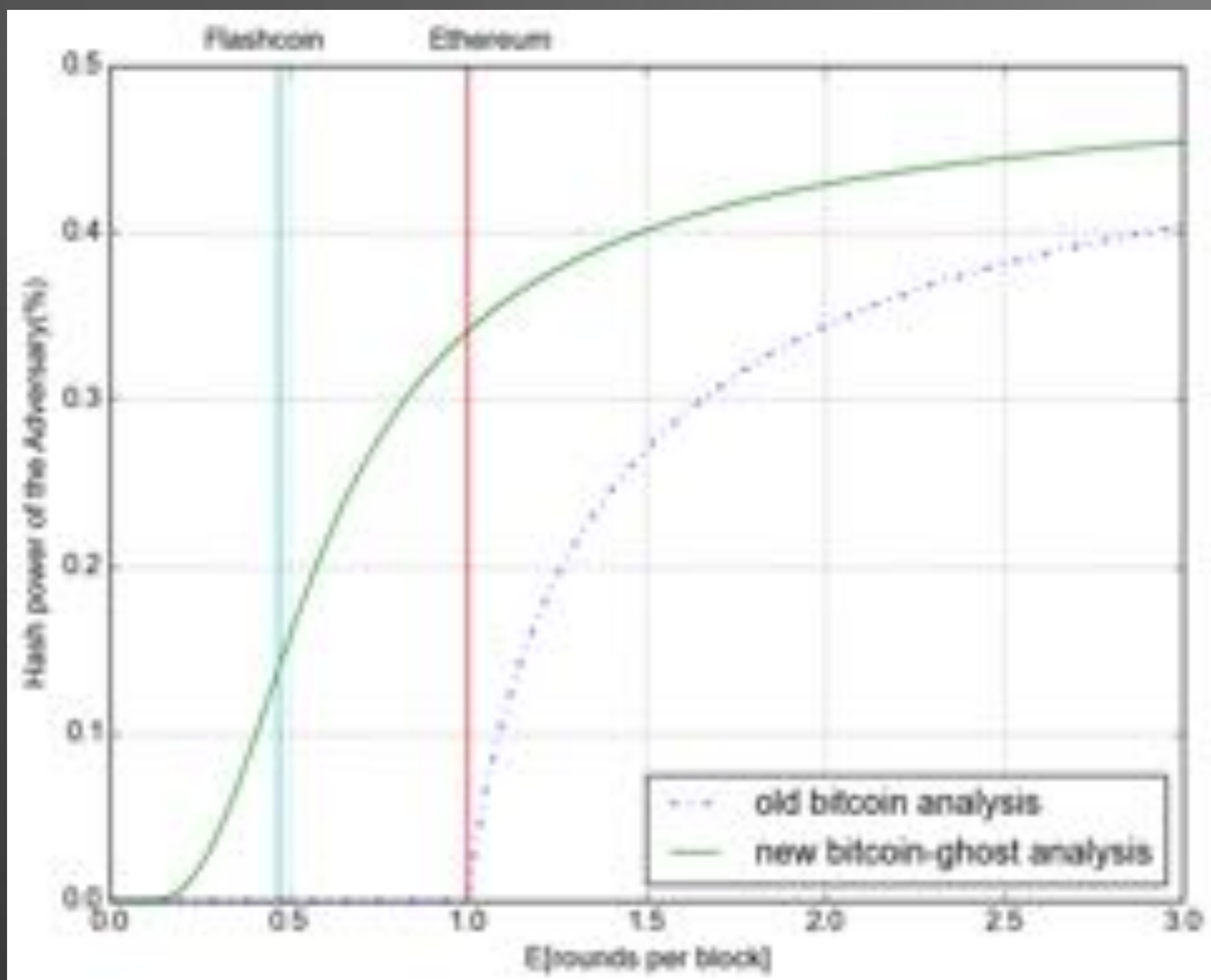
GHOST =
greedy heaviest
observed subtree

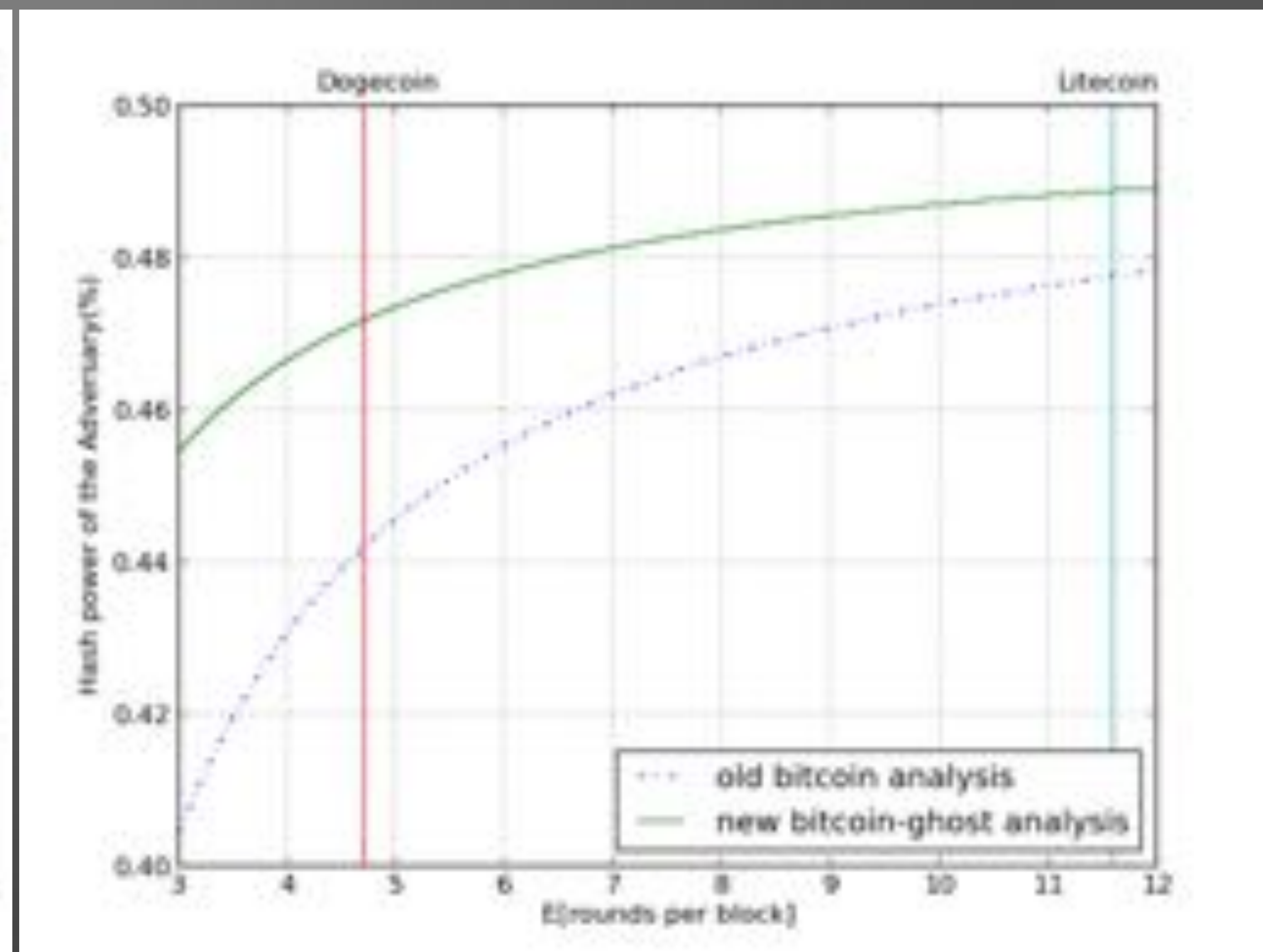


Scalability

- How fast can we run blockchain protocols before security breaks down?

Speed Security Tradeoffs





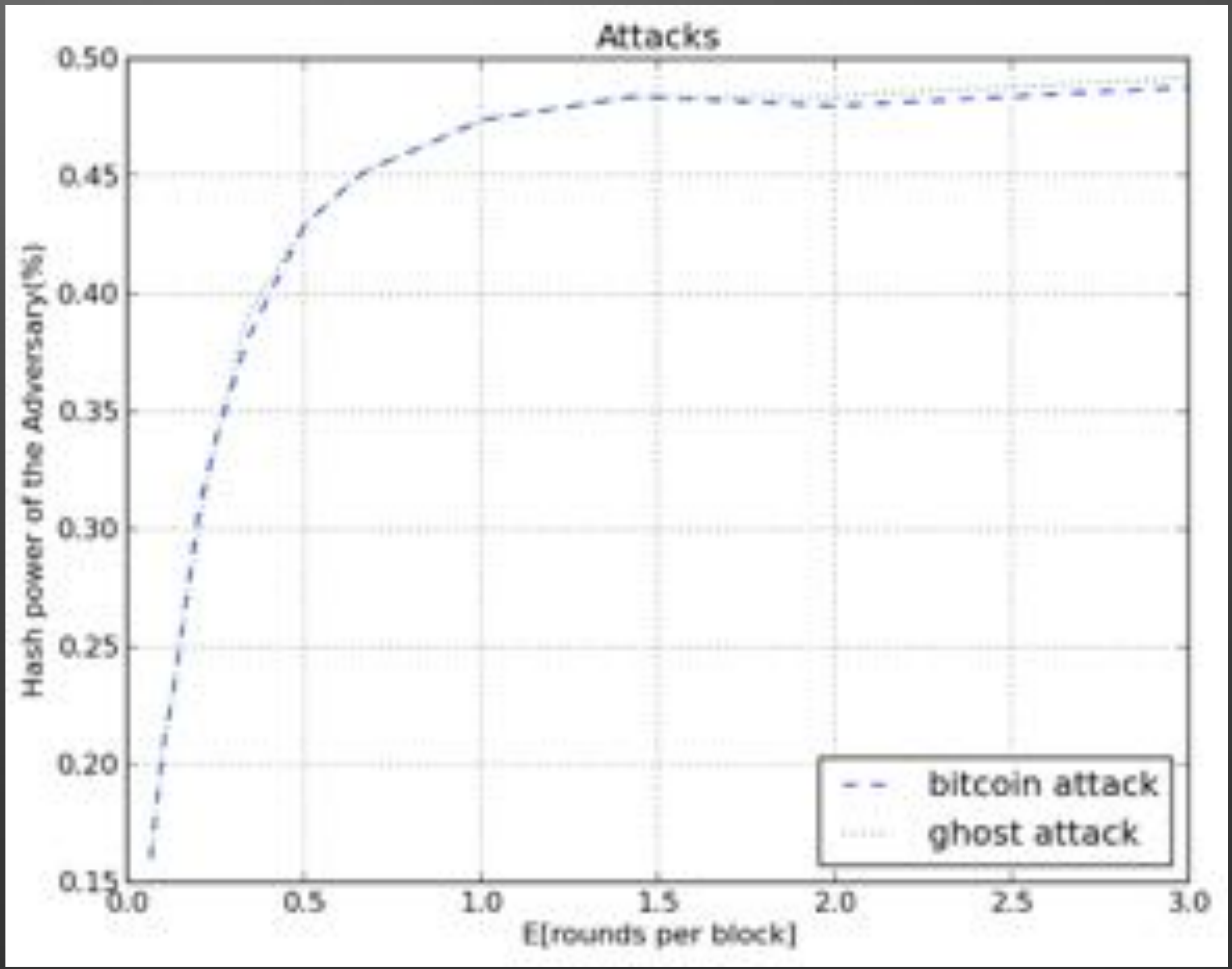
Bitcoin is far to the right in these diagrams

old bitcoin analysis is [GKL15]
new bitcoin-ghost analysis is from http://eprint.iacr.org/2015/1019

Round is taken to be around 12 sec

Attacks on CP property

[strategy :
maintain a fork
as much as
possible trying
to divide honest
parties between
the two branches]



Challenges

- Can we make the bitcoin backbone incentive compatible?
- What is the best way to extend the bitcoin ledger functionality to enable advanced applications that require complex transactions (e.g., contracts, automatic rewards etc.)?
- Can we devise blockchain protocols with better performance characteristics (while maintaining all security properties)? can we prove optimality of protocols (e.g., liveness)?
- How to reduce the energy required for ledger maintenance - - Alternative approaches to POW, e.g., proof of stake.

Post-Doc Opening

on blockchain systems

at University of Edinburgh

Aggelos Kiayias

akiayias@inf.ed.ac.uk