The Bitcoin Backbone Protocol with Chains of Variable Difficulty

Nikos Leonardos
University of Athens

joint work with

Juan A. Garay
Texas A&M

and

Aggelos Kiayias
University of Edinburgh, IOHK

Gratefully acknowledging support from ERC project CODAMODA, H2020 project PANORAMIX, IOHK Research

Bitcoin: a solution to two problems

- Bitcoin was the first decentralized cryptocurrency, with no need for a trusted central authority.
- Bitcoin was a fresh solution at an **old, fundamental, and well-studied** problem in distributed computing, the consensus problem.

Bitcoin: a solution to two problems

- Bitcoin was the first decentralized cryptocurrency, with no need for a trusted central authority.
- Bitcoin was a fresh solution at an **old, fundamental, and well-studied** problem in distributed computing, the consensus problem.

Formal analysis

To understand and analyze Bitcoin's core protocol means to supply formal descriptions of the following.

- A model in which a solution to the problem can described.
- The properties that a suggested solution should satisfy.
- Proof that Bitcoin's backbone protocol indeed has the desired properties.

Previous work: The Bitcoin backbone protocol [GKL15]

- First formal analysis of the Bitcoin core protocol.
- Applications on top of the backbone protocol, assuming minority adversarial hashing power.
  - Consensus (blockchain based).
  - Robust transaction ledger (e.g., Bitcoin).

Previous work: The Bitcoin backbone protocol [GKL15]

- First formal analysis of the Bitcoin core protocol.
- Applications on top of the backbone protocol, assuming minority adversarial hashing power.
  - Consensus (blockchain based).
  - Robust transaction ledger (e.g., Bitcoin).
- Follow up work: Model variants and extensions.
  - Additional properties [KP15,PSS17], partial synchrony [PSS17], simulation based security [BMTZ17].

Previous work: The Bitcoin backbone protocol [GKL15]

- First formal analysis of the Bitcoin core protocol.

- Applications on top of the backbone protocol, assuming minority adversarial hashing power.
  - Consensus (blockchain based).
  - Robust transaction ledger (e.g., Bitcoin).

- Follow up work: Model variants and extensions.
  - Additional properties [KP15,PSS17], partial synchrony [PSS17], simulation based security [BMTZ17].

- All of the above work in the static setting, i.e., assume fixed number of participants and a fixed target.

Previous work: The Bitcoin backbone protocol [GKL15]

- First formal analysis of the Bitcoin core protocol.

- Applications on top of the backbone protocol, assuming minority adversarial hashing power.
  - Consensus (blockchain based).
  - Robust transaction ledger (e.g., Bitcoin).

- Follow up work: Model variants and extensions.
  - Additional properties [KP15,PSS17], partial synchrony [PSS17], simulation based security [BMTZ17].

- All of the above work in the static setting, i.e., assume fixed number of participants and a fixed target.

- This is not how Bitcoin works.

  It employs a target recalculation mechanism that adjusts POW hardness and accommodates for dynamic population of users.

This work

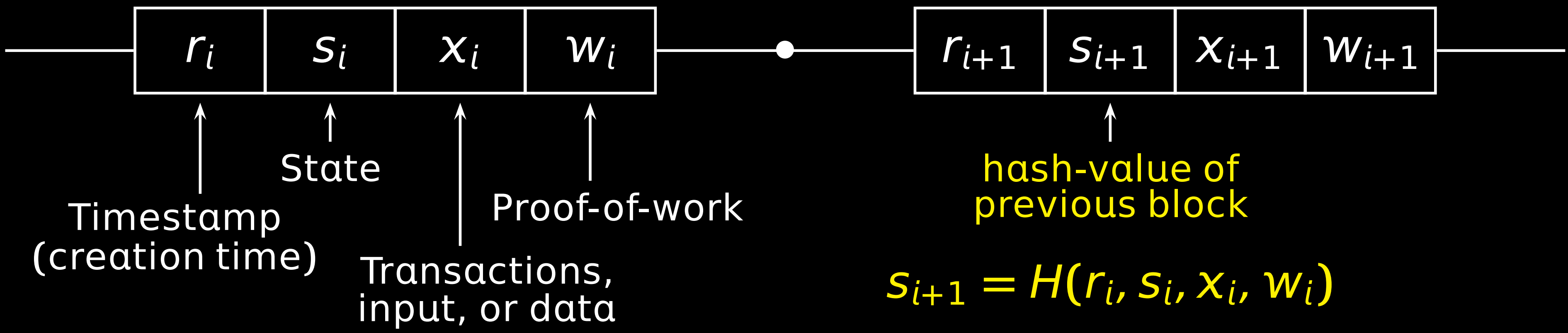
- First formal analysis of Bitcoin's target recalculation function.
- [GKL15] Applications carry over to this setting (consensus, robust transaction ledger).
- New analysis methodology for blockchain protocols in the dynamic setting.

The model

- Synchronous model: time is discrete and divided in rounds.
- A number of honest parties n and an adversary that controls t parties.
  - Honest parties act independently.
  - Parties controlled by the adversary collaborate.
- Parties communicate by broadcasting a message.

  The adversary can:
  - inject messages into a party's incoming messages.
  - reorder a party's incoming messages.
- Anonymous setting: parties cannot associate a message to a sender; they don't even know if two messages come from the same sender.

Bitcoin's data structure: the blockchain



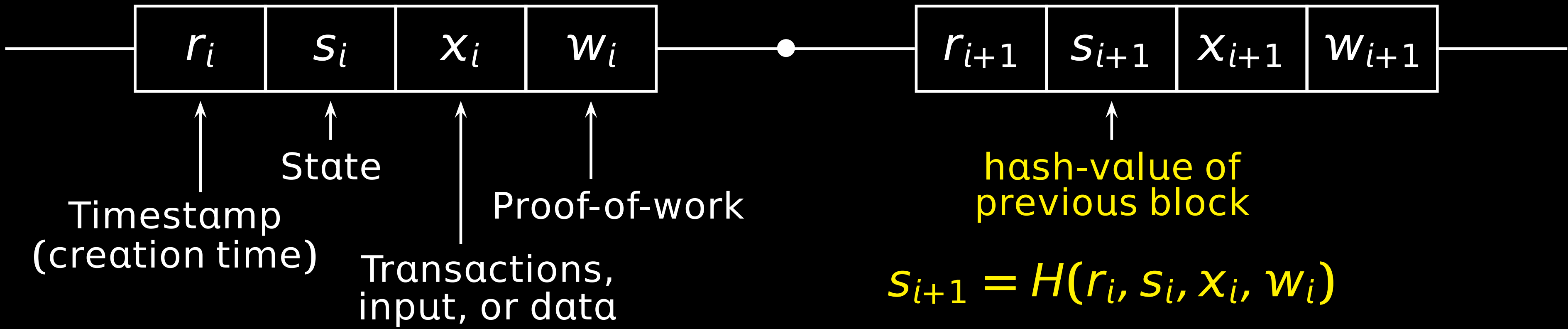
- A block (r, s, x, w) is valid if it has a small hash-value, providing a proof-of-work:

$$H(r, s, x, w) < T.$$

- A chain is valid if all its blocks provide a proof-of-work and each block extends the previous one:

$$\text{for each } i, \quad s_{i+1} = H(r_i, s_i, x_i, w_i) \ \text{ and } \ r_{i+1} > r_i.$$

Comments on the blockchain



$$s_{i+1} = H(r_i, s_i, x_i, w_i)$$

- To alter the contents of a block and preserve the length of the chain the adversary either has to discover a collision in $H(\cdot)$ or compute all the subsequent blocks.
  - Thus the adversary *cannot* delete, copy, inject, or predict blocks.
- The hash function is modeled as a random oracle.
- By adjusting the target T we control how hard is computing a block: the lower the target the higher the difficulty, wlog $1/T$.

A distributed randomized algorithm

In each round r, each party with a chain C_0 performs the following:

- Receive from the network (block)chains $C_1, C_2, \ldots$
- Choose the first longest chain C among the valid ones in $\{C_0, C_1, C_2, \ldots\}$. (Order matters*.)
- Try to extend the longest chain C.

  This is modeled by a Bernoulli trial with a probability of success that depends on the target T.

  — Suppose its last block is the i-th one and equal to (r_i, s_i, x_i, w_i) with $s = H(r_i, s_i, x_i, w_i)$. Find $w \in \{1, 2, \ldots, q\}$ such that

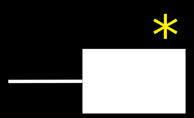
  $$H(r, s, x, w) < T.$$

  If successful, let $C \leftarrow C \,\|\, (r, s, x, w)$.
- If $C \neq C_0$ (i.e., you computed or switched-to another (longer) chain), broadcast the new chain C.

An execution example

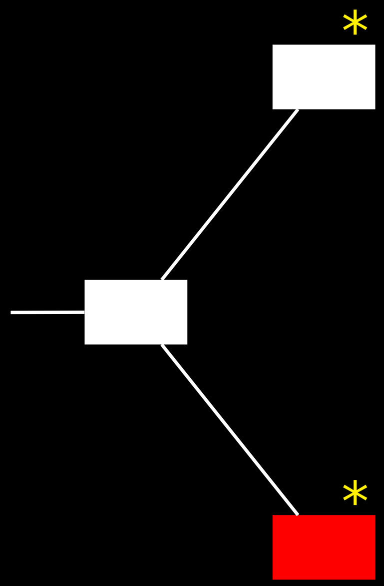
$\emptyset$

An execution example



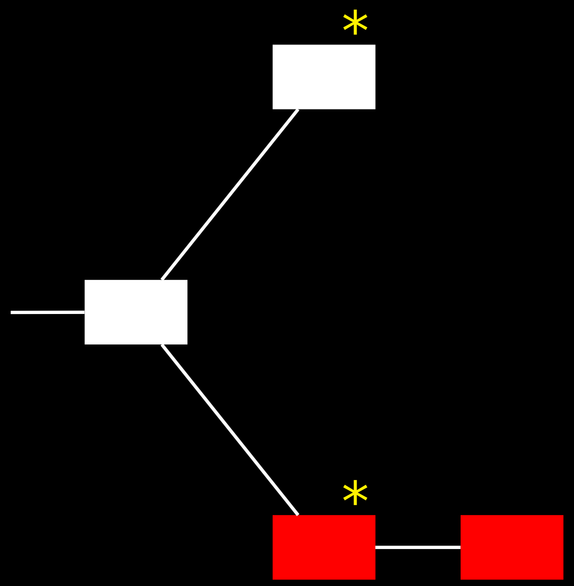
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



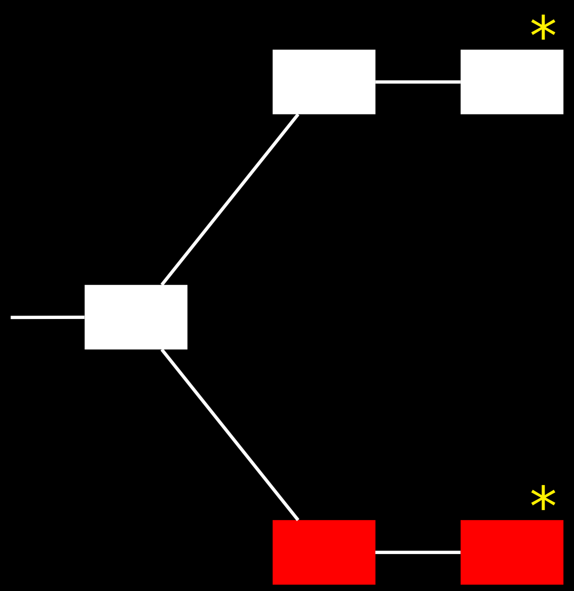
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



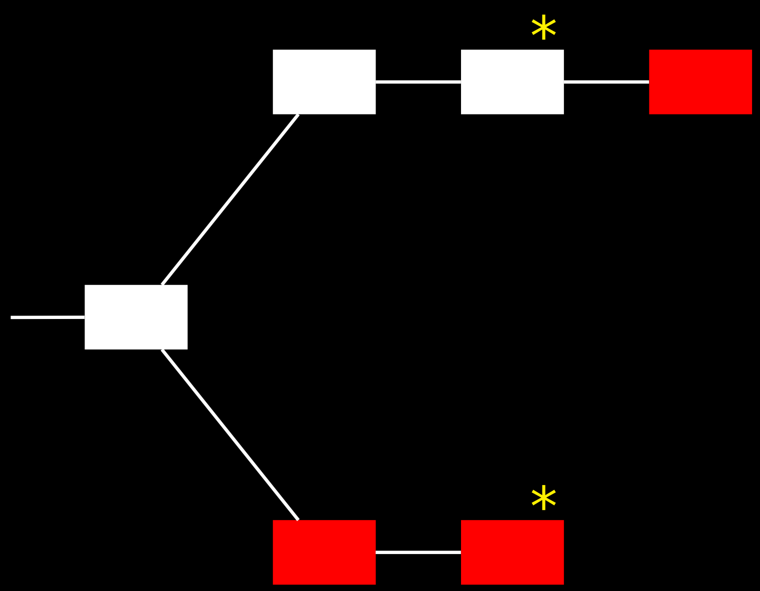
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



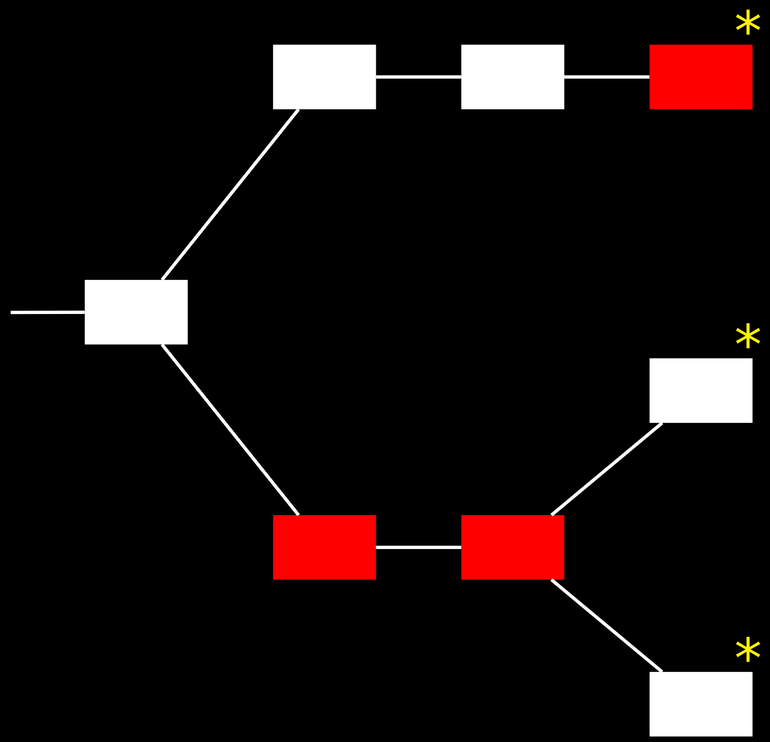
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



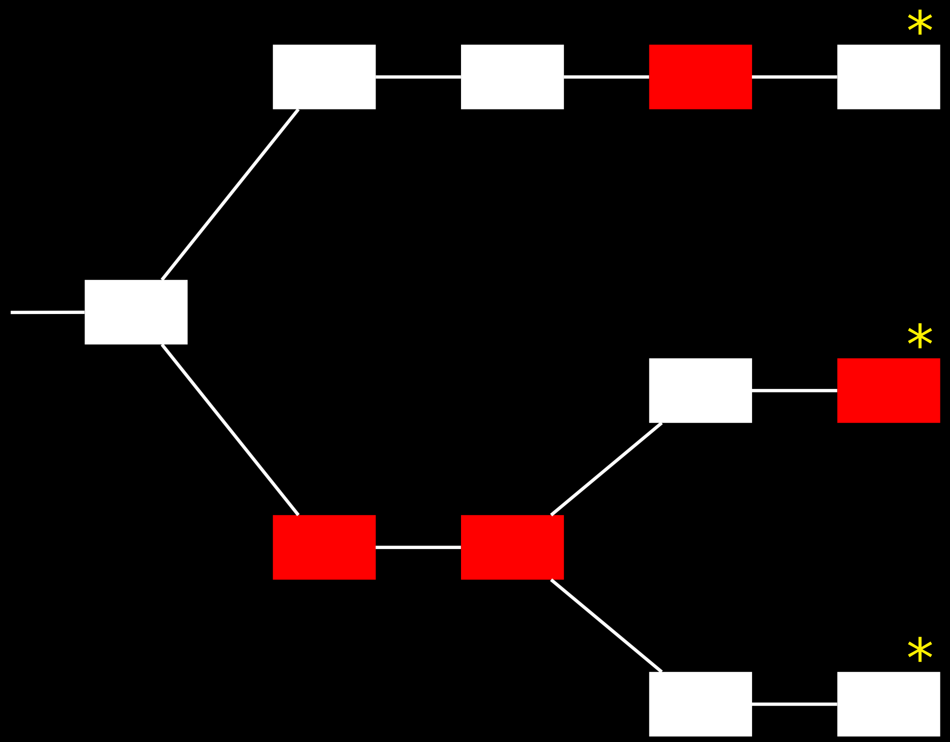
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



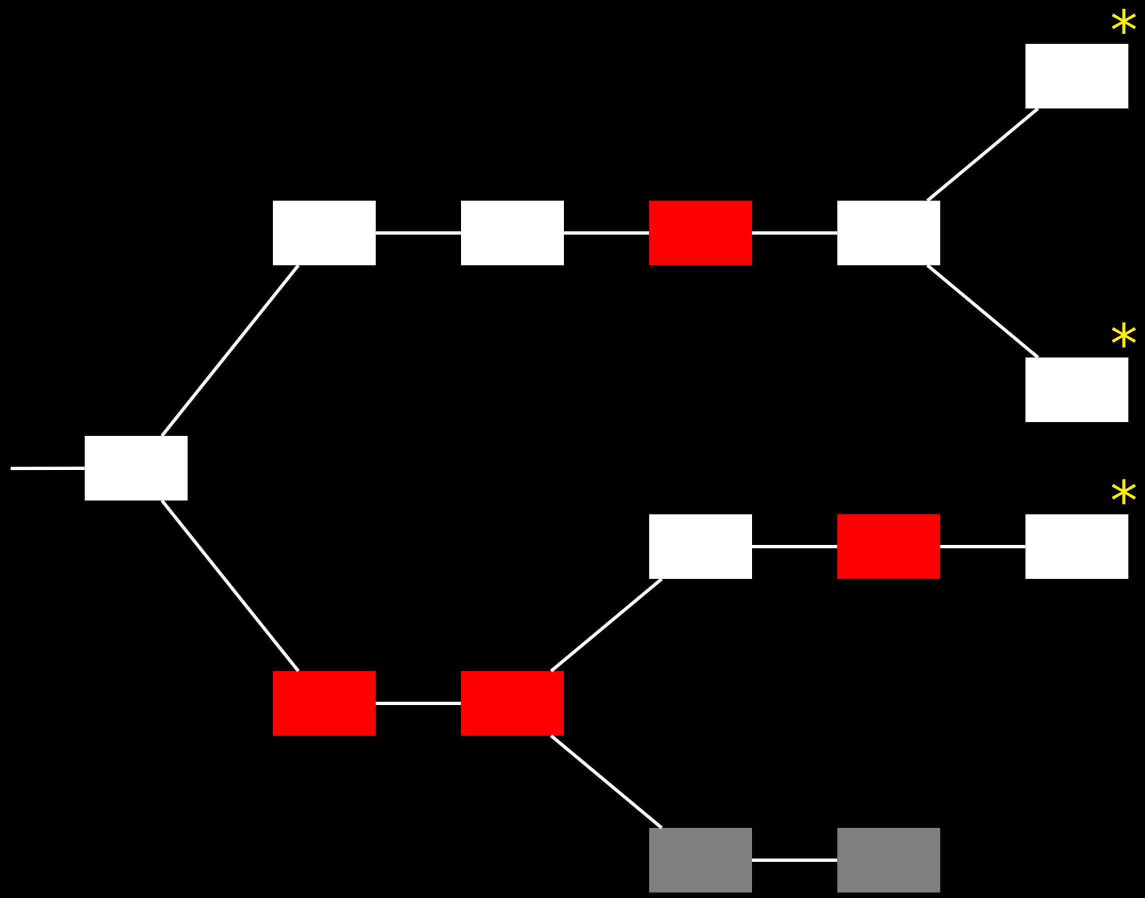
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star ($*$) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



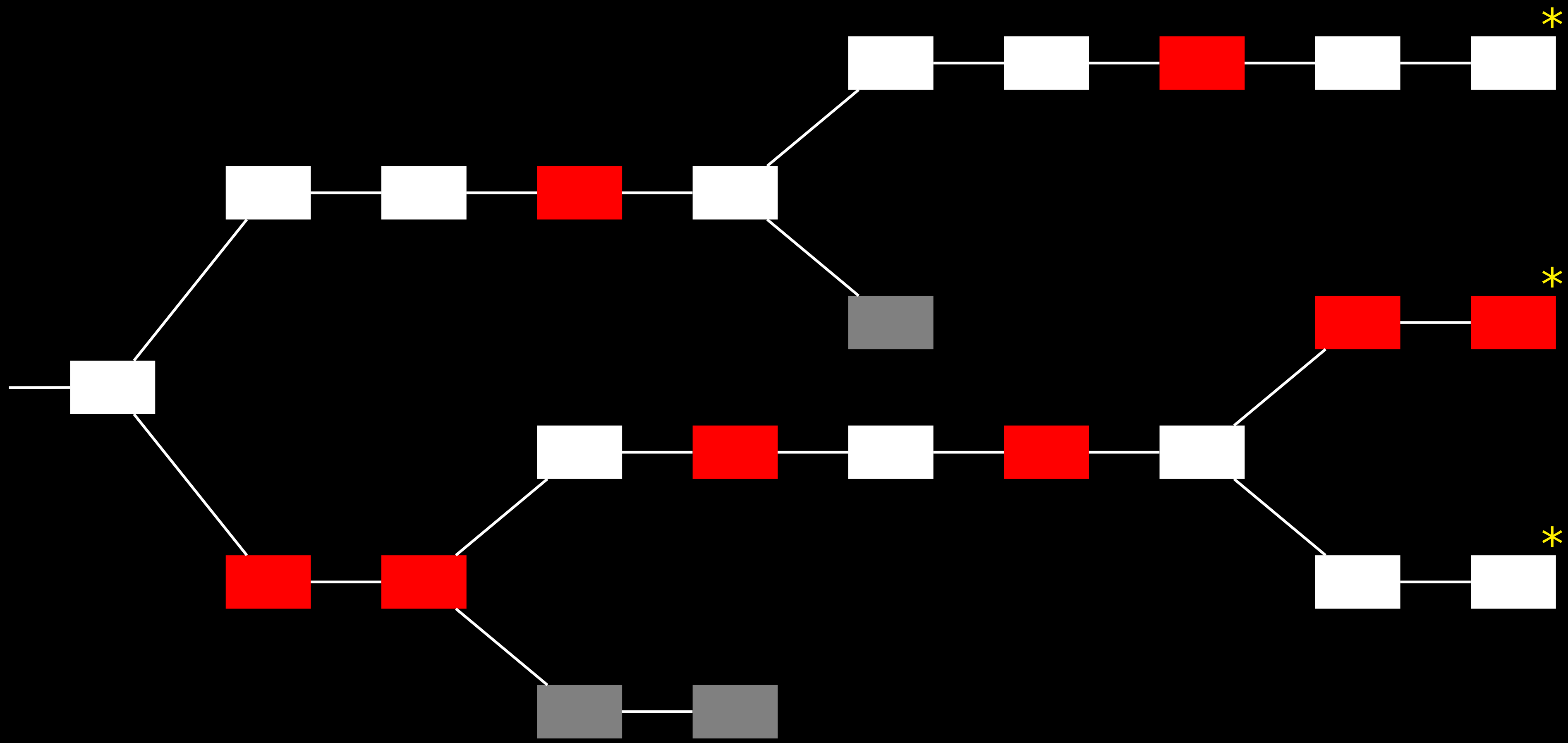
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



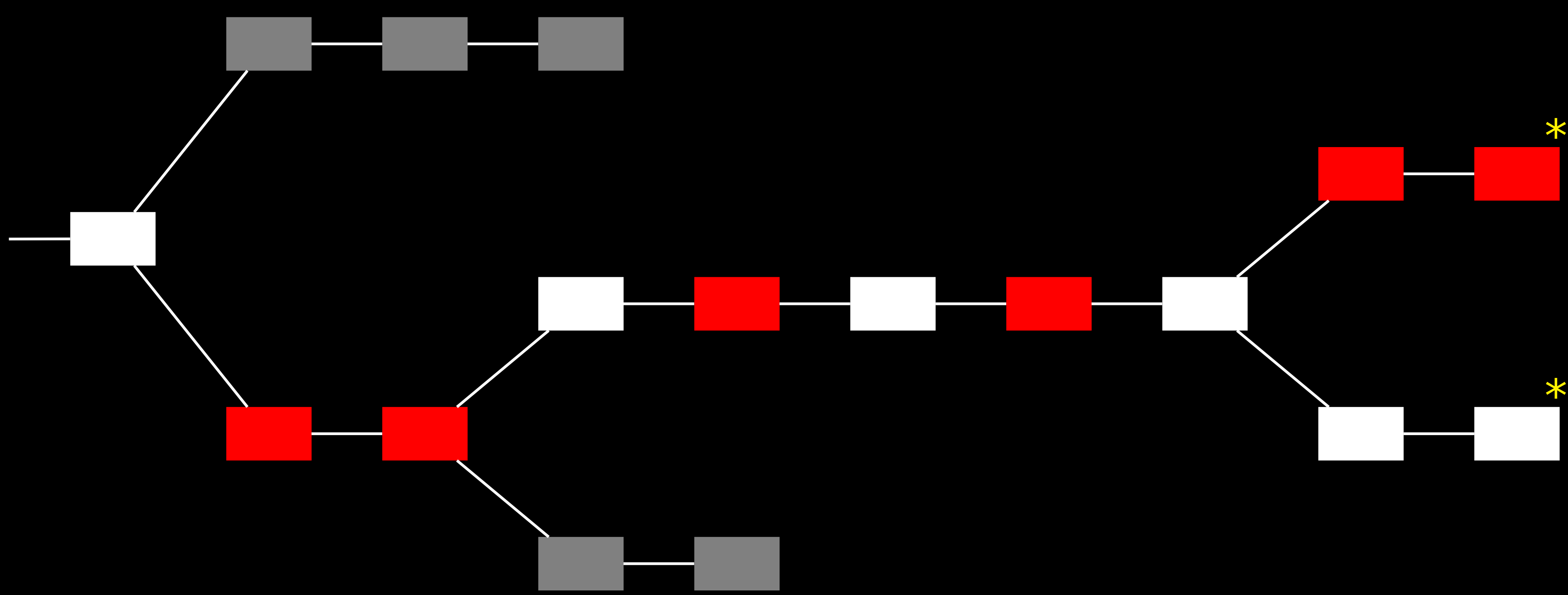
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star (∗) on a block means that an honest party has the chain ending with that block at the given round.

Properties of the transaction ledger

Persistence. If a transaction is confirmed by an honest party, no honest party will ever disagree about the position of that transaction in the ledger.

Properties of the transaction ledger

Persistence. If a transaction is confirmed by an honest party, no honest party will ever disagree about the position of that transaction in the ledger.

Liveness. If a transaction is broadcast, it will eventually become confirmed by all honest parties.

Properties of the blockchain

Common-Prefix Property. Any two honest parties' chains have a large common prefix.

(Specifically, if one party prunes a sufficiently large number of blocks from its chain, it obtains a prefix of the other party's chain.)

Properties of the blockchain

Common-Prefix Property. Any two honest parties' chains have a large common prefix.

(Specifically, if one party prunes a sufficiently large number of blocks from its chain, it obtains a prefix of the other party's chain.)

Chain quality property. Any large enough chunk of an honest party's chain, will contain some blocks computed from honest parties.

Properties of the blockchain

Common-Prefix Property. Any two honest parties' chains have a large common prefix.

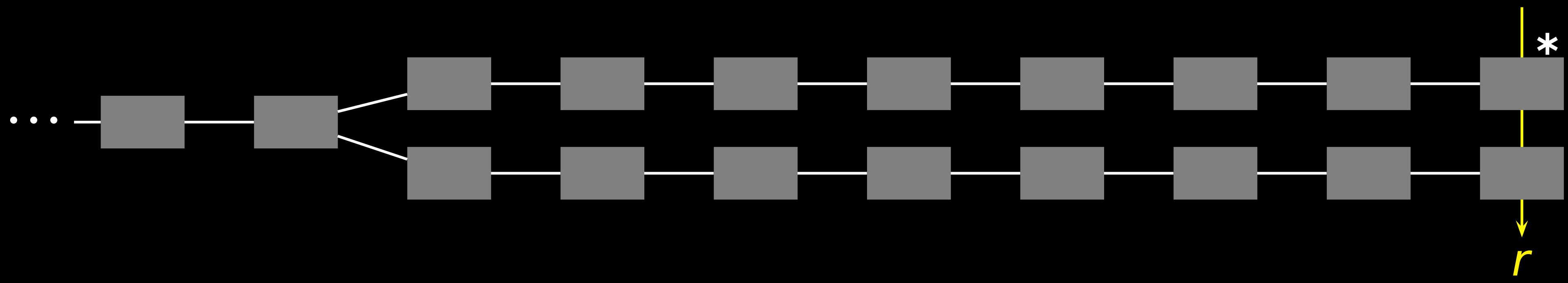
(Specifically, if one party prunes a sufficiently large number of blocks from its chain, it obtains a prefix of the other party's chain.)

Chain quality property. Any large enough chunk of an honest party's chain, will contain some blocks computed from honest parties.

Chain growth property. The chain of any honest party grows at least at a steady rate.

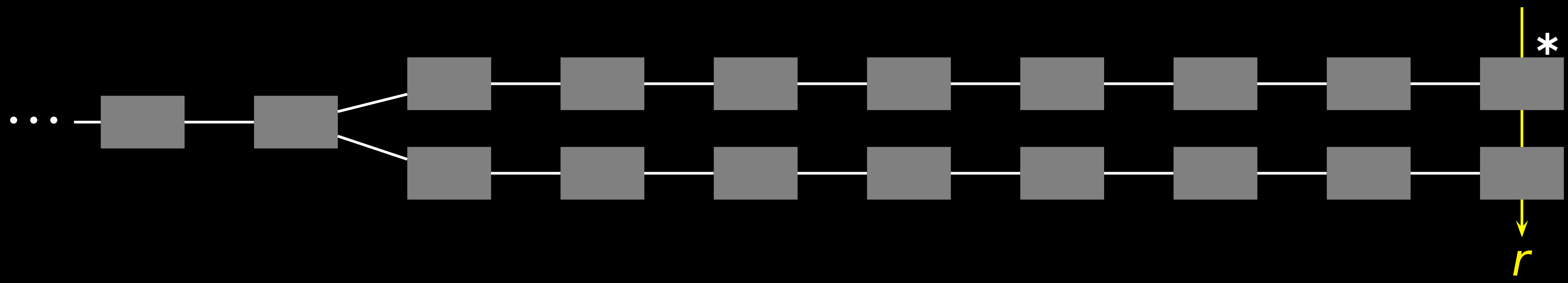
Proof of the common-prefix lemma [GKL15]

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last* k *blocks, is at most* $e^{-\Omega(k)}$. *(The party with the shortest chain should be honest.)*



Proof of the common-prefix lemma [GKL15]

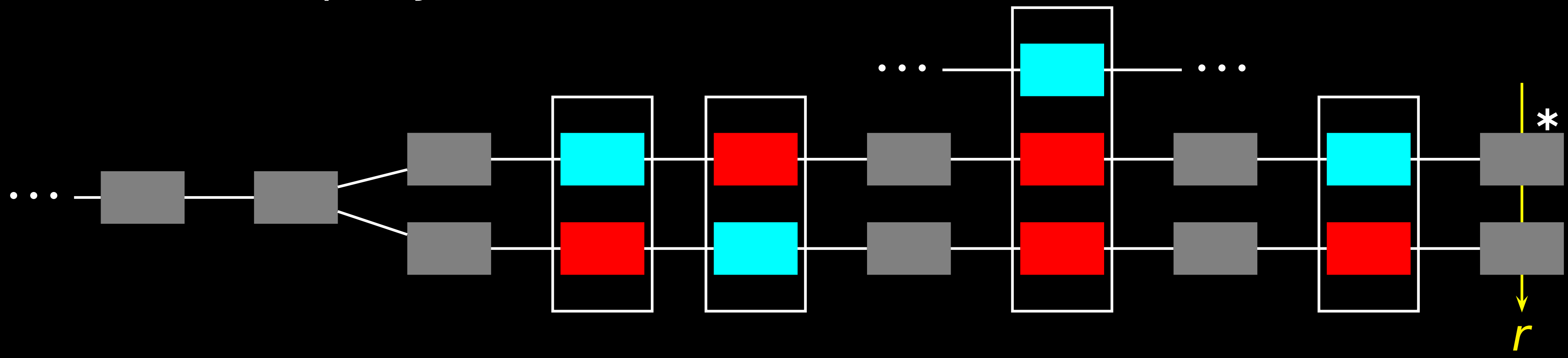
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



Observation. *Suppose the ℓ-the block of a chain was computed by an honest party in a uniquely successful round. Then any other ℓ-th block has been computed by the adversary.*

Proof of the common-prefix lemma [GKL15]

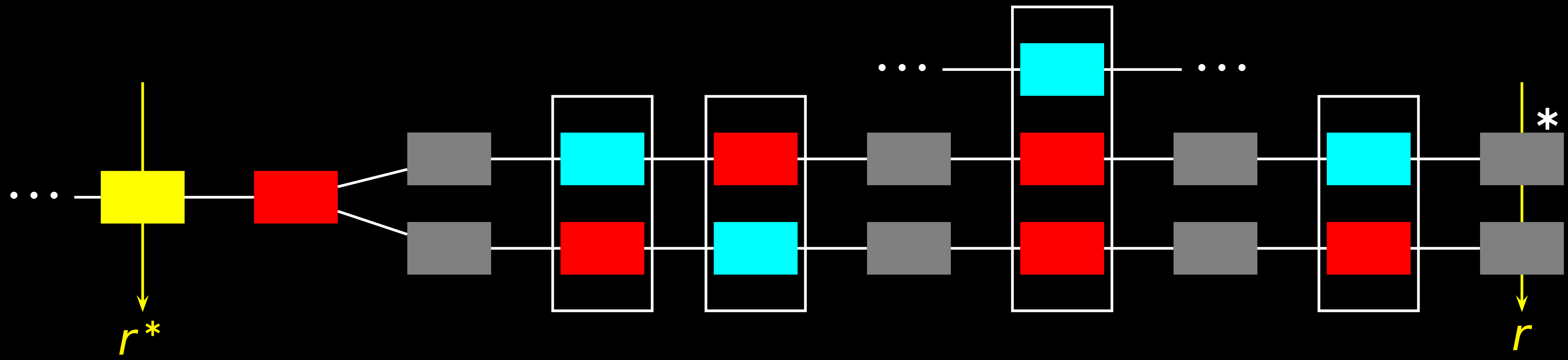
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



Observation. *Suppose the ℓ-the block of a chain was computed by an honest party in a uniquely successful round. Then any other ℓ-th block has been computed by the adversary.*

Proof of the common-prefix lemma [GKL15]

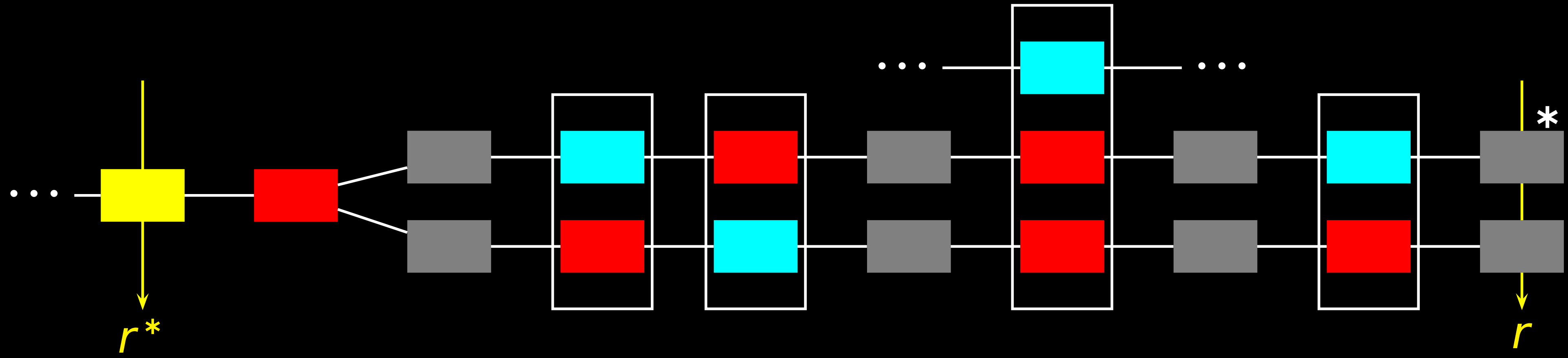
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



Proof. Let r^* be the last round before the fork that was computed by an honest party. Set $S = \{r^* + 1, \ldots, r - 1\}$.

Proof of the common-prefix lemma [GKL15]

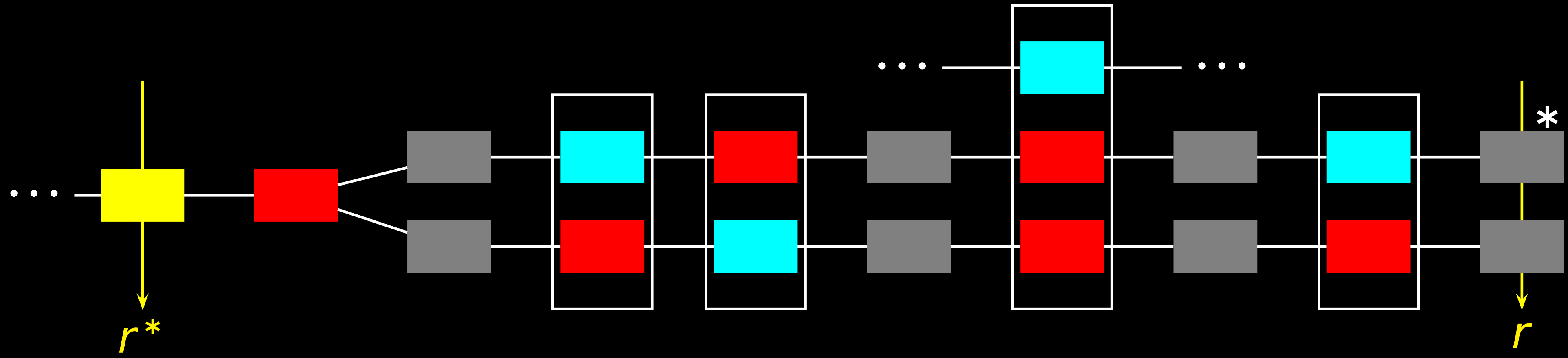
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



Proof. Let r^* be the last round before the fork that was computed by an honest party. Set $S = \{r^* + 1, \ldots, r - 1\}$. By the Lemma, to every uniquely successful round in S corresponds an adversarial block computed in S.

Proof of the common-prefix lemma [GKL15]

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



Proof. Let r^* be the last round before the fork that was computed by an honest party. Set $S = \{r^* + 1, \ldots, r - 1\}$. By the Lemma, to every uniquely successful round in S corresponds an adversarial block computed in S. It follows that

$$\text{Uniqely successful rounds in } S \leq \text{Adversarial successes in } S.$$

□

Dynamic execution: number of parties is changing

- The proof in the static case (fixed number of parties) breaks.
  - As block-production rate goes to 1, persistence breaks.
  - As block-production rate goes to 0, liveness is hurt.

Dynamic execution: number of parties is changing

- The proof in the static case (fixed number of parties) breaks.
  - As block-production rate goes to 1, persistence breaks.
  - As block-production rate goes to 0, liveness is hurt.

- Actually, Bitcoin strives to maintain constant block-production rate of about 1 block per 10 mins.

Dynamic execution: number of parties is changing

- The proof in the static case (fixed number of parties) breaks.
  - As block-production rate goes to 1, persistence breaks.
  - As block-production rate goes to 0, liveness is hurt.
- Actually, Bitcoin strives to maintain constant block-production rate of about 1 block per 10 mins.
- The difficulty of producing a block can be calibrated by changing the target T.

  Note that we want to use this in a distributed manner.

  Bitcoin achieves (approximately) constant rate by having the target of the to-be-computed block determined (locally) by a fixed number of previous blocks.

Dynamic execution: number of parties is changing

- The proof in the static case (fixed number of parties) breaks.
  - As block-production rate goes to 1, persistence breaks.
  - As block-production rate goes to 0, liveness is hurt.

- Actually, Bitcoin strives to maintain constant block-production rate of about 1 block per 10 mins.

- The difficulty of producing a block can be calibrated by changing the target T.

  Note that we want to use this in a distributed manner.

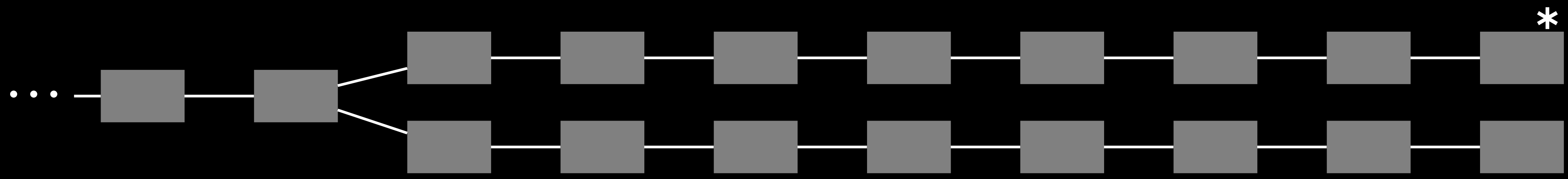
  Bitcoin achieves (approximately) constant rate by having the target of the to-be-computed block determined (locally) by a fixed number of previous blocks.

- Each block now is associated with a target T and difficulty $\frac{1}{T}$.

  Parties now follow the heaviest chain.

The common-prefix lemma in the dynamic case

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



The common-prefix lemma in the dynamic case

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last* k *blocks, is at most* $e^{-\Omega(k)}$. *(The party with the shortest chain should be honest.)*

The common-prefix lemma in the dynamic case

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*

Observation. *Suppose difficulty d of a chain belongs to a block that was computed by an honest party in a uniquely successful round. Then any other block that contains difficulty d has been computed by the adversary.*

The common-prefix lemma in the dynamic case

Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*

Observation. *Suppose difficulty d of a chain belongs to a block that was computed by an honest party in a uniquely successful round. Then any other block that contains difficulty d has been computed by the adversary.*

Difficulty accumulated in uniqely successful rounds in a set S $\leq$ Difficulty accumulated by the adversary during rounds in S □

The common-prefix lemma in the dynamic case

Difficulty accumulated in uniqely successful rounds in a set S	$\leq$	Difficulty accumulated by the adversary during rounds in S

- Same statement in static case [GKL15] is easy, as we are comparing two binomials.
- In the dynamic case, as prove, we have two martingales where success probabilities are random variables depending on the strategy of the adversary.

Naive target recalculation

- The target is recalculated every m blocks.

  Bitcoin uses $m = 2016$ and calls the period between two recalculation points an epoch.

  If one wants to extend a chain of length λm, first determines T by the last m blocks.

Naive target recalculation

- The target is recalculated every m blocks.

  Bitcoin uses $m = 2016$ and calls the period between two recalculation points an epoch.

  If one wants to extend a chain of length λm, first determines T by the last m blocks.

- Informally, if the last m blocks were calculated quickly, then increase difficulty (decrease T), otherwise decrease difficulty (increase T).

Naive target recalculation

- The target is recalculated every m blocks.

  Bitcoin uses $m = 2016$ and calls the period between two recalculation points an epoch.

  If one wants to extend a chain of length λm, first determines T by the last m blocks.

- Informally, if the last m blocks were calculated quickly, then increase difficulty (decrease T), otherwise decrease difficulty (increase T).

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set

  $$T' = \frac{\Delta}{m/f} \cdot T, \qquad (f = \text{block-production rate}).$$

  This is justified because for small f the relation between f and T is approximately linear.

Bahack's difficulty raising attack

- Suppose that at some round r the honest parties have a chain of length λm.
- The adversary builds the next epoch all by himself with fake timestamps, resulting in huge difficulty for the next epoch.
- His strategy is to set T' so small, so that if he computes the 1st block (a superblock of difficulty $\frac{1}{T'}$) of the next epoch fast (say half the expected time), he obtains a chain heavier than the chain the honest parties are expected to have by that time.
- This works with constant probability!

Bahack's difficulty raising attack

- Suppose that at some round r the honest parties have a chain of length λm.
- The adversary builds the next epoch all by himself with fake timestamps, resulting in huge difficulty for the next epoch.
- His strategy is to set T' so small, so that if he computes the 1st block (a superblock of difficulty $\frac{1}{T'}$) of the next epoch fast (say half the expected time), he obtains a chain heavier than the chain the honest parties are expected to have by that time.
- This works with constant probability!

But, Nakamoto knew this!!!

Bitcoin's target recalculation

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set

$$T' = \frac{\Delta}{m/f} \cdot T,$$

Bitcoin's target recalculation

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set

$$T' = \frac{\Delta}{m/f} \cdot T,$$

unless $T' < T/4$ or $T' > 4T$, in which case set $T' = T/4$ or $T' = 4T$ accordingly.

Bitcoin's target recalculation

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set

$$T' = \frac{\Delta}{m/f} \cdot T,$$

unless $T' < T/4$ or $T' > 4T$, in which case set $T' = T/4$ or $T' = 4T$ accordingly.

- If the number of parties keeps increasing by a large factor per epoch, then target recalculation won't catch up.

Bitcoin's target recalculation

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set

$$T' = \frac{\Delta}{m/f} \cdot T,$$

unless $T' < T/4$ or $T' > 4T$, in which case set $T' = T/4$ or $T' = 4T$ accordingly.

- If the number of parties keeps increasing by a large factor per epoch, then target recalculation won't catch up.

- Can we prove security under the assumption that the number of parties does not fluctuate wildly?

Bitcoin's target recalculation

- Suppose the last m blocks were computed in Δ rounds for target T. If we want to have m blocks in every $\frac{m}{f}$ rounds, set
$$T' = \frac{\Delta}{m/f} \cdot T,$$
unless $T' < T/4$ or $T' > 4T$, in which case set $T' = T/4$ or $T' = 4T$ accordingly.

- If the number of parties keeps increasing by a large factor per epoch, then target recalculation won't catch up.

- Can we prove security under the assumption that the number of parties does not fluctuate wildly?

Theorem. *If, for appropriate constants s and λ,*
$$\forall r, r' \qquad |r - r'| \leq s \quad \Longrightarrow \quad \frac{n_r}{\lambda} \leq n_{r'} \leq \lambda n_r,$$
then common prefix and chain quality hold (assuming adversarial minority and appropriate initialization).

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

If a chain C is adopted by an honest party, then C:

- was never abandoned by honest parties for $\Omega(m/f)$ rounds,

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

If a chain $\mathcal{C}$ is adopted by an honest party, then $\mathcal{C}$:

- was never abandoned by honest parties for $\Omega(m/f)$ rounds,
- is $O(m/f)$-accurate—each of its blocks has a timestamp that is $O(m/f)$ rounds away from its real creation time,

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

If a chain C is adopted by an honest party, then C:

- was never abandoned by honest parties for $\Omega(m/f)$ rounds,
- is $O(m/f)$-accurate—each of its blocks has a timestamp that is $O(m/f)$ rounds away from its real creation time,
- has “very good” recalculation points,

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

If a chain C is adopted by an honest party, then C:

- was never abandoned by honest parties for $\Omega(m/f)$ rounds,
- is $O(m/f)$-accurate—each of its blocks has a timestamp that is $O(m/f)$ rounds away from its real creation time,
- has "very good" recalculation points,
- has blocks with "good" targets.

Proof roadmap (high level)

Assuming the execution begins with good initial parameters—i.e., in the beginning the block-production rate is very close to the (desired) f—we show that with high probability the following hold.

If a chain C is adopted by an honest party, then C:

- was never abandoned by honest parties for $\Omega(m/f)$ rounds,
- is $O(m/f)$-accurate—each of its blocks has a timestamp that is $O(m/f)$ rounds away from its real creation time,
- has "very good" recalculation points,
- has blocks with "good" targets.

Theorem. *Every block in a chain that is ever adopted by an honest party, has "accurate" timestamp and "good" target.*

Typical executions

- We define the notion typical execution. Informally, the execution is typical if for any set of consecutive rounds the number of successful queries did not deviate to far from its expectation.

Typical executions

- We define the notion typical execution. Informally, the execution is typical if for any set of consecutive rounds the number of successful queries did not deviate to far from its expectation.
- Having defined the typical set and proved that almost all executions are typical, we can then study its properties without considering probability at all.

Typical executions

- We define the notion typical execution. Informally, the execution is typical if for any set of consecutive rounds the number of successful queries did not deviate to far from its expectation.
- Having defined the typical set and proved that almost all executions are typical, we can then study its properties without considering probability at all.
- For example, in the static case, we may say that an execution is typical if for any set S of $\Omega(k)$ consecutive rounds, the number of blocks computed is $(1 \pm \delta)f|S|$.

Typical executions

- We define the notion typical execution. Informally, the execution is typical if for any set of consecutive rounds the number of successful queries did not deviate to far from its expectation.

- Having defined the typical set and proved that almost all executions are typical, we can then study its properties without considering probability at all.

- For example, in the static case, we may say that an execution is typical if for any set S of $\Omega(k)$ consecutive rounds, the number of blocks computed is $(1 \pm \delta)f|S|$.

- How to do this in the dynamic case? Consider the following stochastic procedure.
  - In the beginning of each round i, the adversary chooses p_i.
  - We gain $\frac{1}{p_i}$ with prob p_i or lose $\frac{1}{1-p_i}$ with prob $1-p_i$.

  Consider an adversary that is deterministic and adaptive.

Concentration bounds

Theorem [McDiarmid, Concentration]. *Let* $X_0, X_1, \ldots$ *be a martingale with respect to the sequence* $Y_0, Y_1, \ldots$. *For* $n \geq 0$, *let*

$$V = \sum_{1 \leq i \leq n} \mathrm{Var}(X_i - X_{i-1} | Y_0, .., Y_{i-1}) \text{ and } b = \max_{1 \leq i \leq n} \sup(X_i - X_{i-1} | Y_0, .., Y_{i-1}),$$

where sup is taken over all possible assignments to $Y_0, \ldots, Y_{i-1}$. *Then, for any* $t, v \geq 0$,

$$\Pr\left[X_n \geq X_0 + t \wedge V \leq v\right] \leq \exp\left\{-\frac{t^2}{2v + 2bt/3}\right\}.$$

Concentration bounds

Theorem [McDiarmid, Concentration]. *Let* $X_0, X_1, \ldots$ *be a martingale with respect to the sequence* $Y_0, Y_1, \ldots$. *For* $n \geq 0$, *let*

$$V = \sum_{1 \leq i \leq n} \mathrm{Var}(X_i - X_{i-1} | Y_0, .., Y_{i-1}) \text{ and } b = \max_{1 \leq i \leq n} \sup(X_i - X_{i-1} | Y_0, .., Y_{i-1}),$$

where sup is taken over all possible assignments to $Y_0, \ldots, Y_{i-1}$. *Then, for any* $t, v \geq 0$,

$$\Pr\Big[X_n \geq X_0 + t \wedge V \leq v\Big] \leq \exp\Big\{-\frac{t^2}{2v + 2bt/3}\Big\}.$$

Proof application: Show that if an execution begins with good initial parameters (in particular, $V \leq v$) and at some point deviates from the desired block-production rate, then concentration was violated while $V \leq v$.

Future/ongoing work

- Improve analysis tightness.
  - Real world Bitcoin parameterization outside our current bounds.

Future/ongoing work

- Improve analysis tightness.
  - Real world Bitcoin parameterization outside our current bounds.
- Partial synchrony.
  - Similar strategy as in the new version of [GKL15] (see eprint) might work.

Future/ongoing work

- Improve analysis tightness.
  - Real world Bitcoin parameterization outside our current bounds.
- Partial synchrony.
  - Similar strategy as in the new version of [GKL15] (see eprint) might work.
- Is Bitcoin's target recalculation best possible?

Thank you for listening

Nakamoto's insight

Re: Bitcoin P2P e-cash paper

Satoshi Nakamoto | Thu, 13 Nov 2008 19:34:25 -0800

James A. Donald wrote:
> It is not sufficient that everyone knows X. We also
> need everyone to know that everyone knows X, and that
> everyone knows that everyone knows that everyone knows X
> - which, as in the Byzantine Generals problem, is the
> classic hard problem of distributed data processing.

The proof-of-work chain is a solution to the Byzantine Generals' Problem. I'll try to rephrase it in that context.

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, otherwise they will be discovered and get in trouble. 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 all agree. It has been decided that anyone who feels like it will announce a time, and whatever time is heard first will be the official attack time. The problem is

https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html

Byzantine agreement (consensus)

A set of parties $\{1, ..., n\}$, t of which are controlled and coordinated by an adversary. Parties have inputs $x_1, ..., x_n \in \{0, 1\}$ and want to decide on outputs $v_1, ..., v_n$ so that the following conditions are satisfied.

- **Agreement:** All honest parties decide on the same value (i.e., if i and j are honest, then $v_i = v_j$).
- **Validity:** If all honest parties have the same input value x, then all honest parties decide x (i.e., if i is honest, then $v_i = x$).

Remarks and classical results for consensus

- One of the classical problems in distributed computing, a variant of which was first introduced in "Reaching Agreement in the Presence of Faults" [Pease-Shostak-Lamport 1980].
- Requires $n > 3t$, unless cryptography is used [PSL].
- Even with cryptographic tools, at least $t + 1$ rounds are needed [Fischer-Lynch and Dolev-Strong 1982].
- In an asynchronous or anonymous network no deterministic protocol exists [Fischer-Lynch-Paterson 1985]. But possible with probability 1 [Ben-Or 1983]. (Rounds are expected to be exponential in t, but if $t = O(\sqrt{n})$ constant.)
- Bit complexity is $\Omega(nt)$ [Dolev-Reischuk 1985].

Byzantine Agreement Protocol

Theorem [GKL2015]. *Assuming* $t < n/3$*, the following protocol terminates after* $\Theta(k)$ *rounds in expectation and solves consensus with probability at least* $1 - e^{-\Omega(k)}$.

Byzantine Agreement Protocol

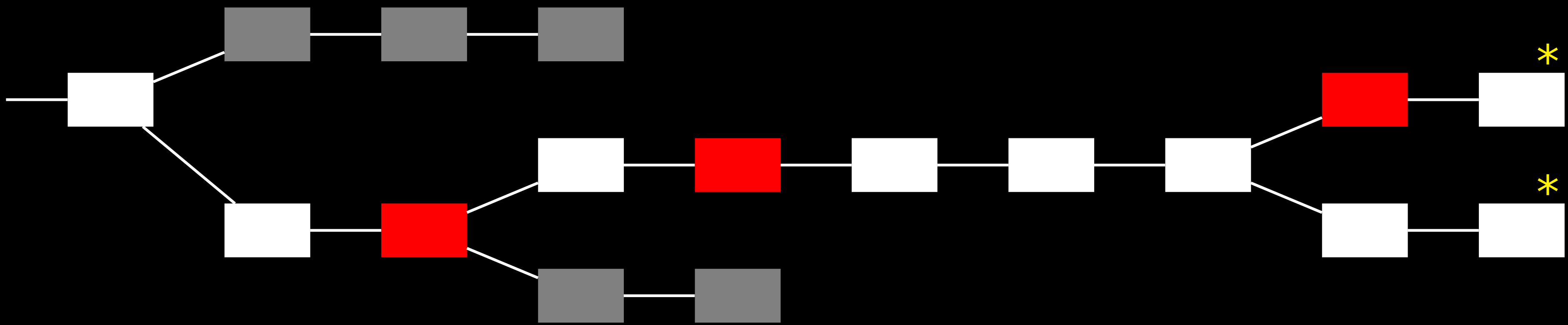
Theorem [GKL2015]. *Assuming $t < n/3$, the following protocol terminates after $\Theta(k)$ rounds in expectation and solves consensus with probability at least $1 - e^{-\Omega(k)}$.*

1) Parties run the Bitcoin protocol, putting their own input-bit in every block they compute.
2) When they obtain a chain with length $\geq 2k$ they halt (after they broadcast it).
3) Each party decides on the output equal to the majority of the inputs recorded in the first k blocks.

Byzantine Agreement Protocol

Theorem [GKL2015]. *Assuming $t < n/3$, the following protocol terminates after $\Theta(k)$ rounds in expectation and solves consensus with probability at least $1 - e^{-\Omega(k)}$.*

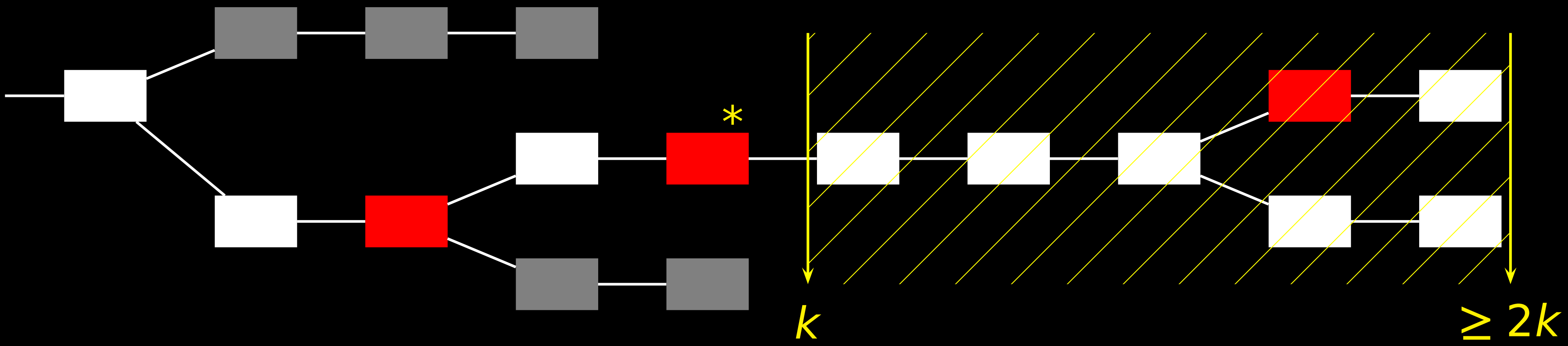
1) Parties run the Bitcoin protocol, putting their own input-bit in every block they compute.
2) When they obtain a chain with length $\geq 2k$ they halt (after they broadcast it).
3) Each party decides on the output equal to the majority of the inputs recorded in the first k blocks.



Byzantine Agreement Protocol

Theorem [GKL2015]. *Assuming $t < n/3$, the following protocol terminates after $\Theta(k)$ rounds in expectation and solves consensus with probability at least $1 - e^{-\Omega(k)}$.*

1) Parties run the Bitcoin protocol, putting their own input-bit in every block they compute.
2) When they obtain a chain with length $\geq 2k$ they halt (after they broadcast it).
3) Each party decides on the output equal to the majority of the inputs recorded in the first k blocks.



Proof of Agreement and Validity

- By the common-prefix property, if the adversary has less than half of the total computational power, Agreement is satisfied with high probability.

  This is because every honest party will output the majority of the input-bits included in the common prefix of their (possibly different) chains. (Consider the first time an honest party has a chain of length at least $2k$.)

Proof of Agreement and Validity

- By the common-prefix property, if the adversary has less than half of the total computational power, Agreement is satisfied with high probability.

  This is because every honest party will output the majority of the input-bits included in the common prefix of their (possibly different) chains. (Consider the first time an honest party has a chain of length at least $2k$.)

- By the chain-quality property, if the adversary has less than one third of the total computational power, Validity is satisfied with high probability.

  This is because out of the k bits of the common prefix, the adversary has computed less than half of them. Therefore, if all the honest parties have the same input x, the majority of the bits in the common prefix will be x.