Bitcoin Backbone, Consesus, Variable Difficulty

Nikos Leonardos
University of Athens

Bitcoin info

- Bitcoin was the first decentralized cryptocurrency with no need for a trusted central authority.
  - Previous work: Pricing functions of Dwork and Naor [1992], MicroMint of Rivest and Shamir [1996], Hashcash of Back [1997,2002], Szabo's bit gold [1998], Karma by Vishnumurthy, Chandrakumar, Sirer [2003].
- Introduced in the 2008 paper "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto (a pseudonym).
- Released as open-source code in 2009; first block: 9, Jan 2009.
  - Nowadays there are more than than 700,000 blocks.
- The total number of bitcoins will not exceed 21 million and this limit is expected to be reached around 2140.
  - Nowadays there are more than 19 million bitcoins in circulation.
  - The smallest denomination is the satoshi, equal to 10^{-8} bitcoins.

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

- A formal description of the model in which the problem and its solution can be described.
- The properties that a suggested solution should satisfy.
- A formal description of the protocol.
- Proof that Bitcoin backbone indeed has the desired properties.

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.

What is not in the model

- Honest parties losing messages or becoming eclipsed or becoming unable to know the current time.
  - Parties experiencing such issues are factored into the adversary.
- The honest parties' incentives.
  - On the other hand, adversarial parties wish to inflict the worst possible damage independently of utility.
- An adversary with computational power that even on occasion, exceeds that of honest parties.
- Attacks that exploit specific weaknesses of the underlying cryptographic primitives.

  [We will use idealized versions of hash functions and digital signatures].

Hash functions

A cryptographic hash function is a deterministic algorithm

$$H : \{0,1\}^* \to \{0,1\}^\kappa$$

with the following properties.

- Preimage resistance: Given $y \in \{0,1\}^\kappa$ it should be computationally infeasible to compute x such that $H(x) = y$.
- Second-preimage resistance: Given x and $y = H(x)$ it should be computationally infeasible to compute a $x' \neq x$ such that $H(x') = y$.
- Collision resistance: It should be computationally infeasible to compute $x \neq x'$ such that $H(x) = H(x')$.

For a meaningful formal definition one considers cryptographic hash families.

Proof-of-work in the random-oracle model

A moderately hard computational task: Given a hash-function $H(\cdot)$ with range $\{0,1\}^{\kappa}$ and a y, find x such that $H(x,y)$ begins with a lot of zeroes. More generaly, given a target T,

- find x such that $H(x,y) < T$.

Proof-of-work in the random-oracle model

A moderately hard computational task: Given a hash-function $H(\cdot)$ with range $\{0,1\}^{\kappa}$ and a y, find x such that $H(x,y)$ begins with a lot of zeroes. More generaly, given a target T,

- find x such that $H(x,y) < T$.

We'll work in the "random oracle" model. That is, we assume the existence of a hash-function $H(\cdot)$ that operates as follows.

- On a query x, the returned value $H(x)$ is a random number from the range of $H(\cdot)$, unless x has been queried before in which case $H(\cdot)$ is consistent (equal to the previous returned value).

Proof-of-work in the random-oracle model

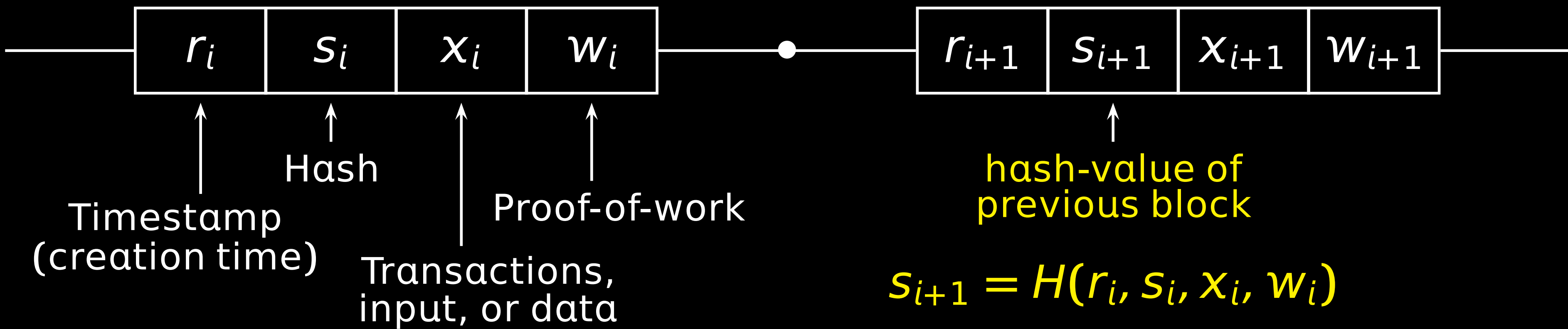
A moderately hard computational task: Given a hash-function $H(\cdot)$ with range $\{0,1\}^\kappa$ and a y, find x such that $H(x,y)$ begins with a lot of zeroes. More generaly, given a target T,

- find x such that $H(x,y) < T$.

We'll work in the "random oracle" model. That is, we assume the existence of a hash-function $H(\cdot)$ that operates as follows.

- On a query x, the returned value $H(x)$ is a random number from the range of $H(\cdot)$, unless x has been queried before in which case $H(\cdot)$ is consistent (equal to the previous returned value).
- A query is successful with probability $\frac{T}{2^\kappa}$, and one needs in expectation $\frac{2^\kappa}{T}$ calls to the oracle $H(\cdot)$ for a proof-of-work.
- Among poly(k) queries, the probability of a collision (two distinct x and x' with $H(x) = H(x')$) is exponentially small in κ.

Bitcoin's data structure: the blockchain



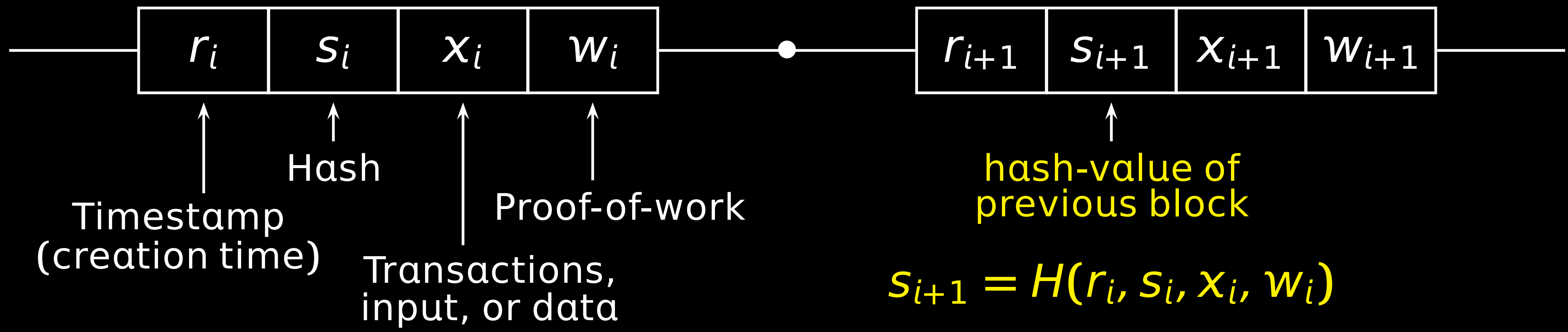
- A block (r, s, x, w) is valid if it has a small hash-value, providing a proof-of-work:

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

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

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

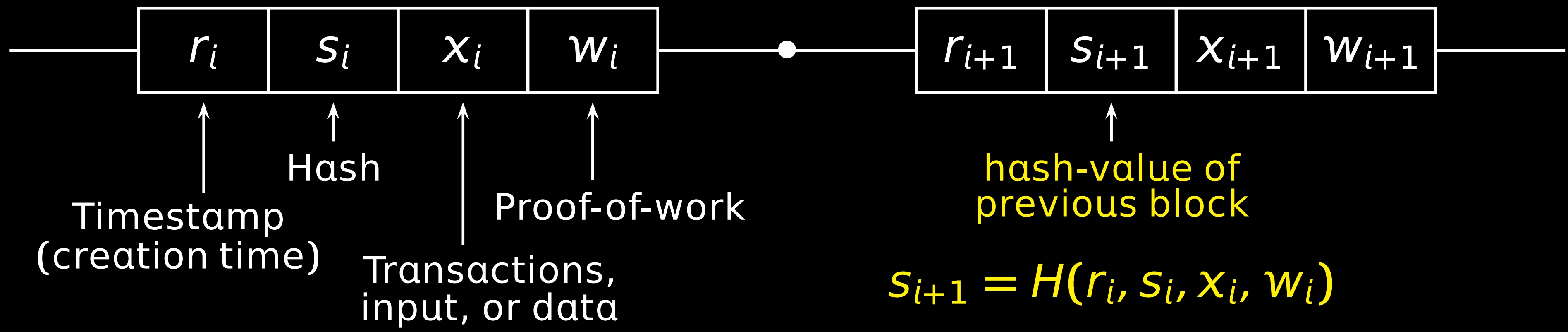
Comments on the blockchain



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

- 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.
- By adjusting the target T we control how hard is computing a block: the lower the target the higher the difficulty, wlog $1/T$.

Transactions on the blockchain



A transaction has the following form:

- "From the output (say 10BTC) of transaction i in block j (which was sent to public pk_0), send 2BTC to pk_1 and 7BTC to pk_2"--- signed with sk_0.
- Fees, coinbase transaction.
- Parties need to agree on which is the j-th block.

Bitcoin backbone: 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

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

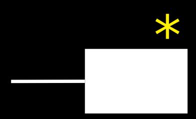
  If successful, let $C \leftarrow C \,\|\, (r, s, x, w)$.

- If $C \neq C_0$ (i.e., you computed or switched-to another (longer) chain), diffuse the new chain C.

An execution example

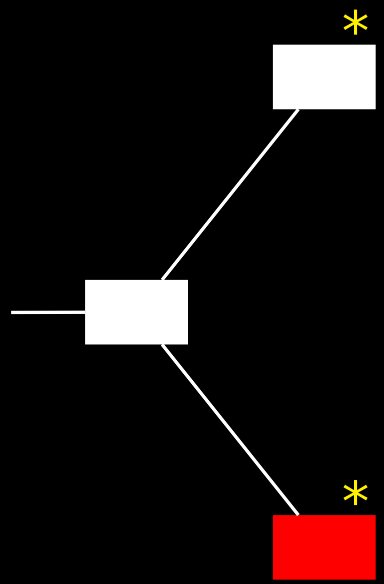
$\emptyset$

An execution example



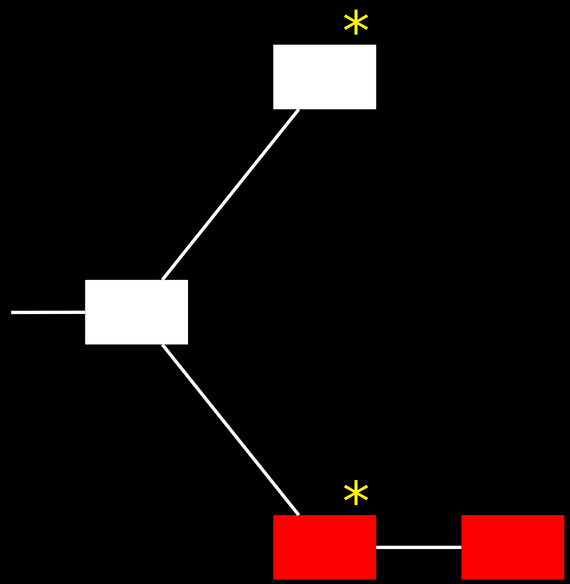
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star ($\ast$) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



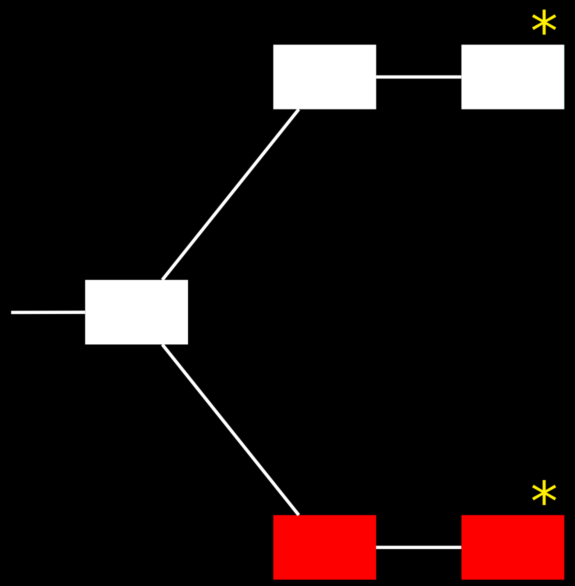
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



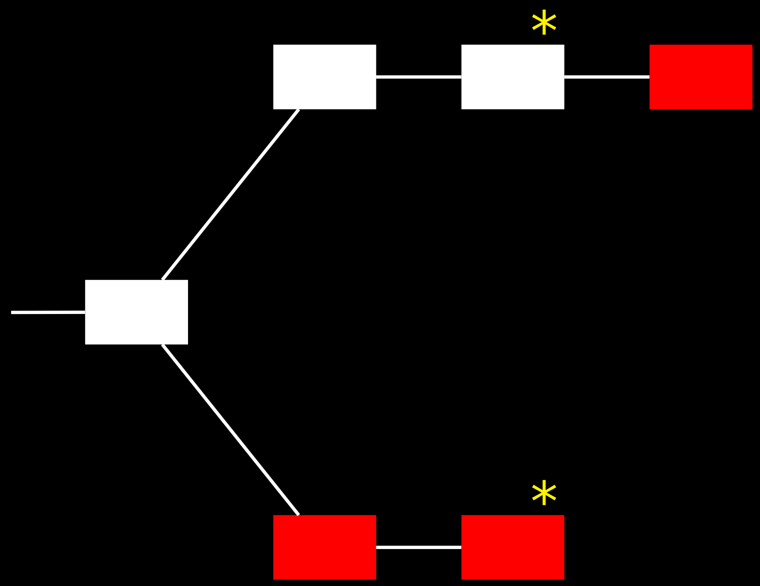
- **White** blocks have been computed by an **honest** party.
- **Red** blocks have been computed by the **adversary**.
- A **star** $(*)$ on a block means that an honest party **has** the chain ending with that block at the given round.

An execution example



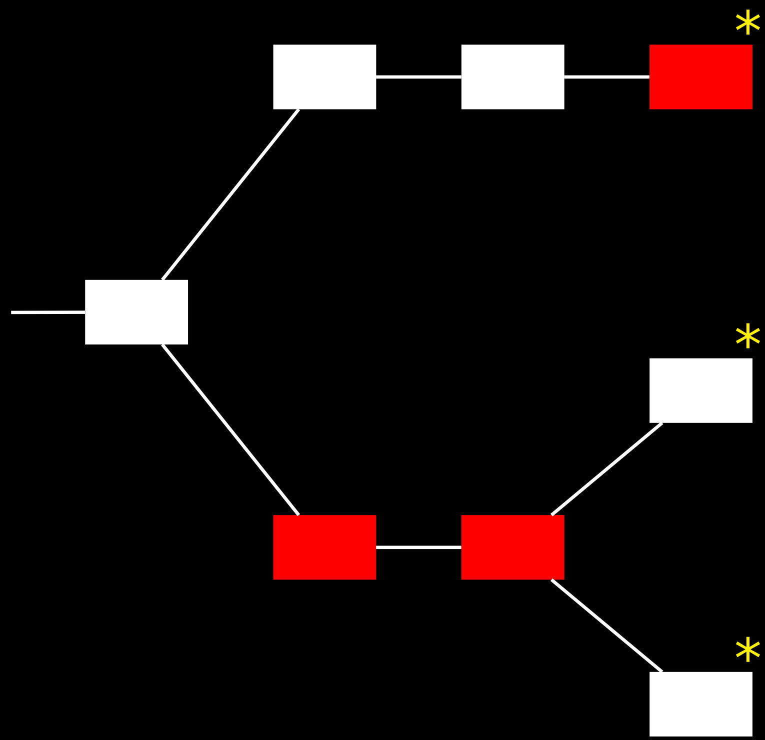
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star ($*$) on a block means that an honest party has the chain ending with that block at the given round.

An execution example



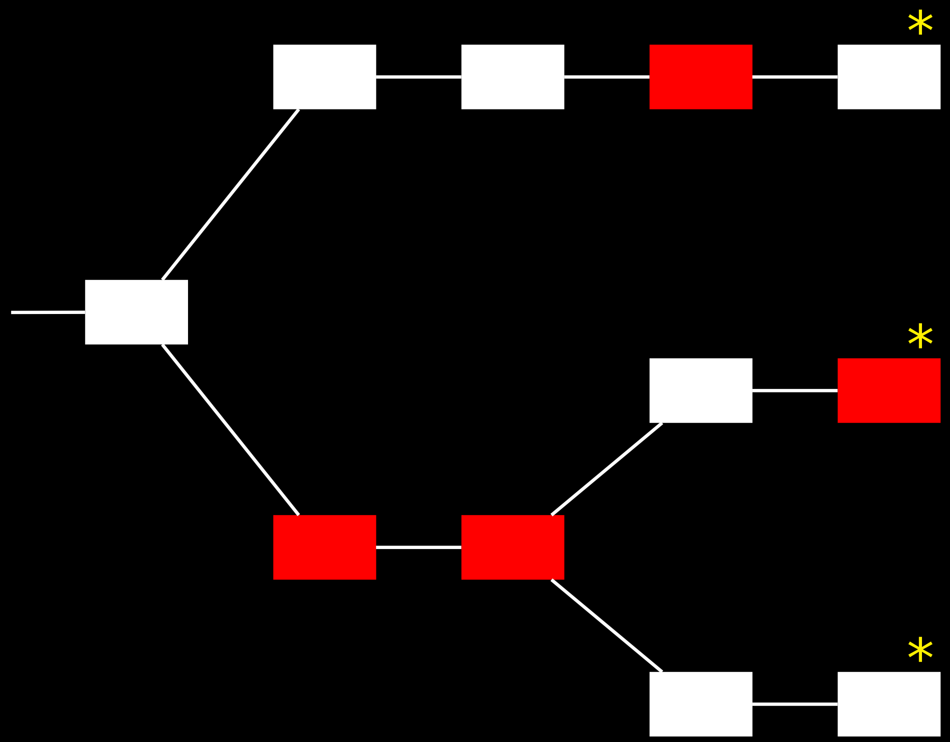
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



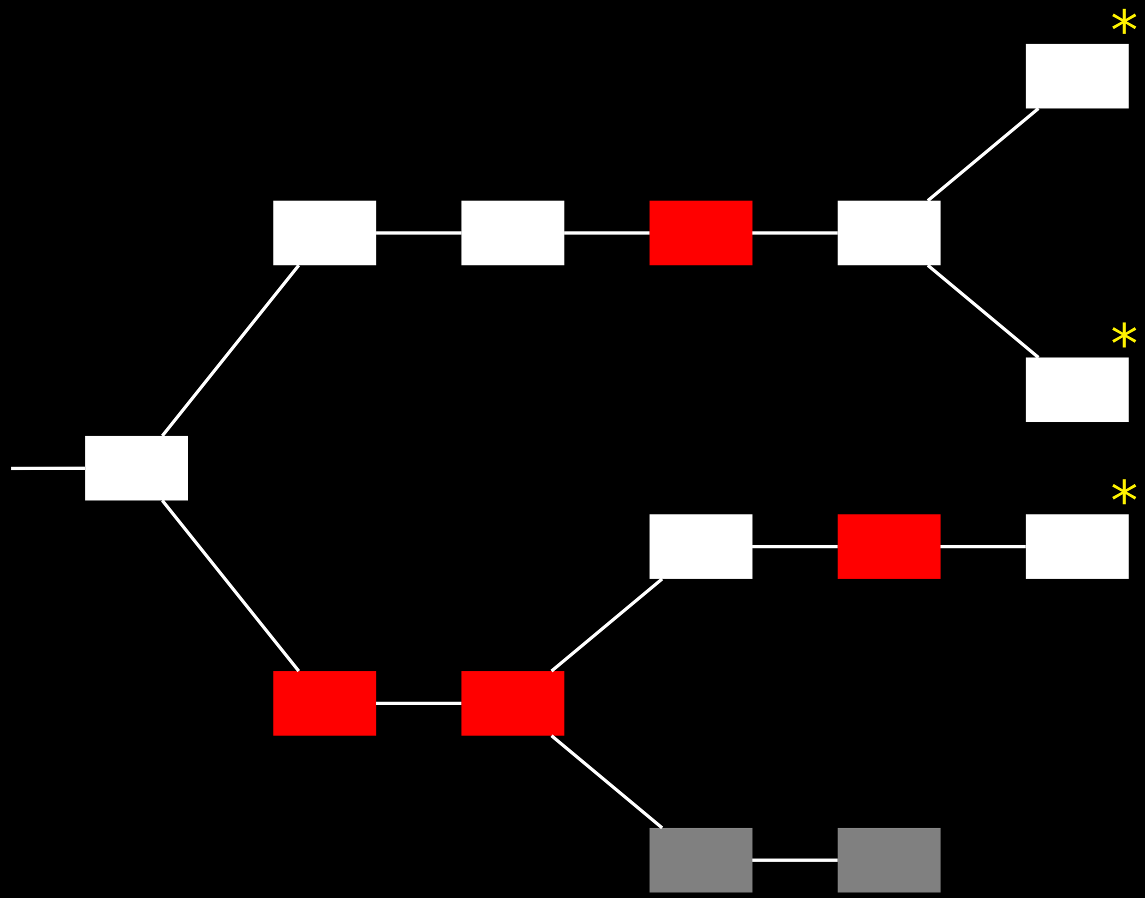
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



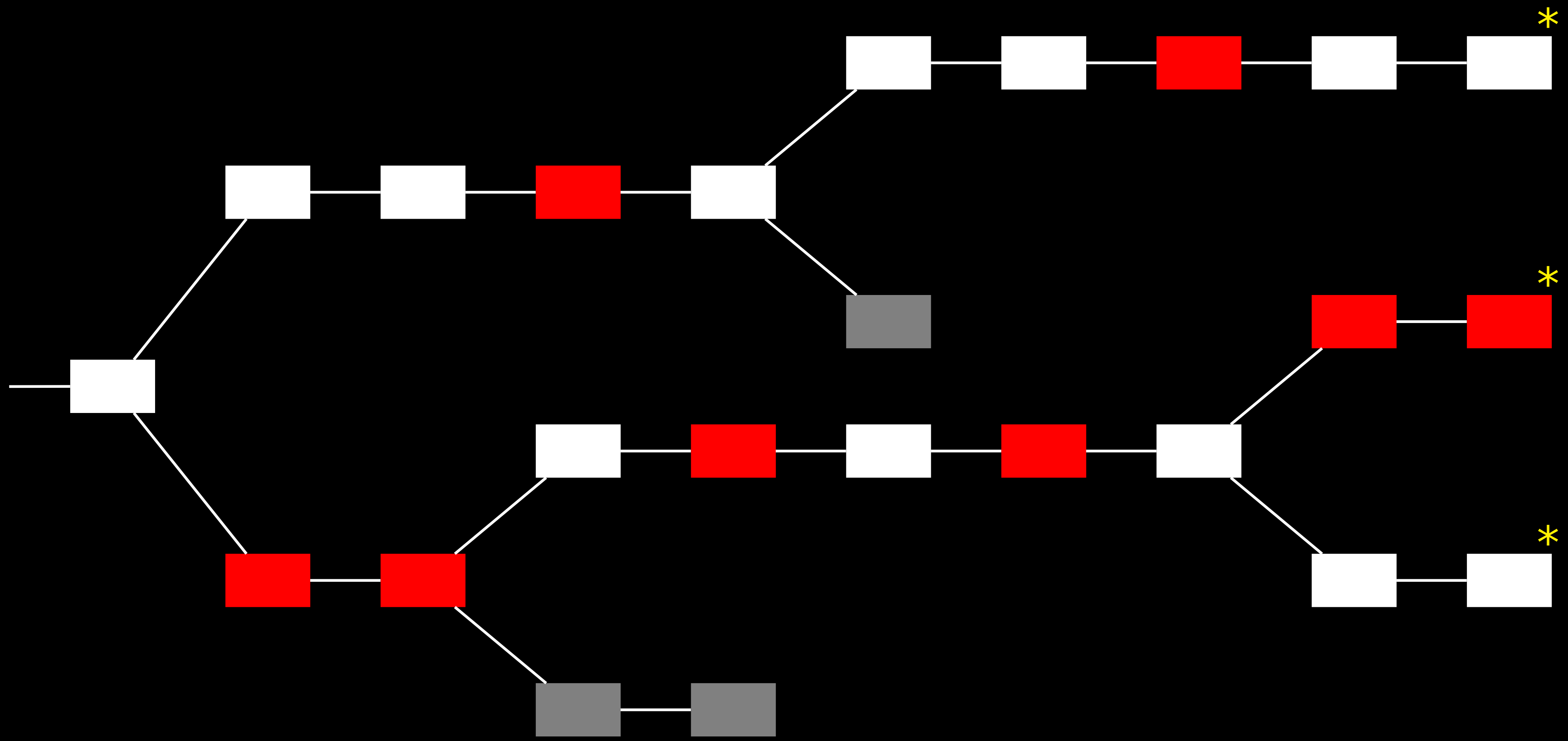
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star $(*)$ on a block means that an honest party has the chain ending with that block at the given round.

An execution example



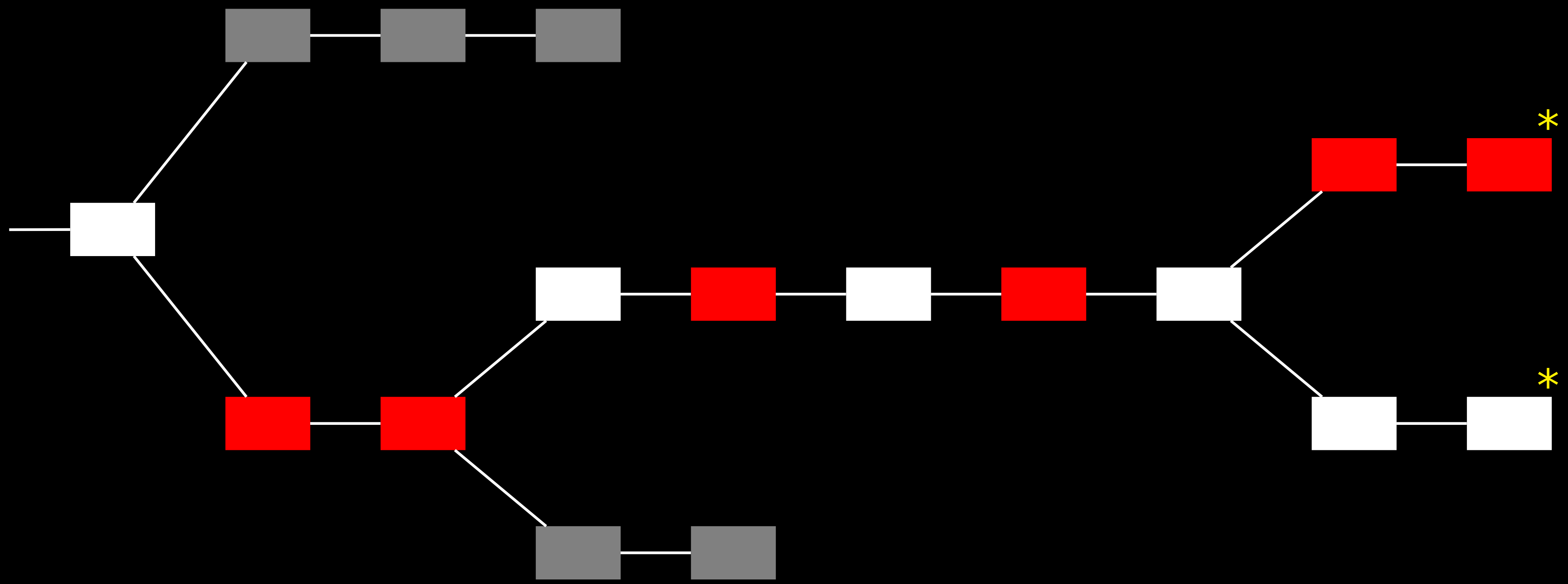
- White blocks have been computed by an honest party.
- Red blocks have been computed by the adversary.
- A star ($*$) on a block means that an honest party has the chain ending with that block at the given round.

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 diffused, it will eventually become confirmed by all honest parties.

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 diffused, 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. (If a party prunes a sufficiently large number of blocks from its chain, then the remaining part is a prefix of any other party's chain.)

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 diffused, 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. (If a party prunes a sufficiently large number of blocks from its chain, then the remaining part is a prefix of any other party's chain.)

Chain-Quality Property. Any sufficiently large segment of an honest party's chain, will contain some blocks computed from honest parties.

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 diffused, 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. (If a party prunes a sufficiently large number of blocks from its chain, then the remaining part is a prefix of any other party's chain.)

Chain-Quality Property. Any sufficiently large segment 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.

Analysis: Random Variables

Successful Round. A round r in which at least one honest party computes a block.

$$X_r = 1 \iff r \text{ is a successful round}$$

$$f := \mathbf{E}[X_r] = 1 - (1-p)^n \approx pn$$

Analysis: Random Variables

Successful Round. A round r in which at least one honest party computes a block.

$$X_r = 1 \iff r \text{ is a successful round}$$

$$f := \mathbf{E}[X_r] = 1 - (1-p)^n \approx pn$$

Uniquely Successful Round. A round r in which exactly one honest party computes a block.

$$Y_r = 1 \iff r \text{ is a uniquly successful round}$$

$$\mathbf{E}[Y_r] = np(1-p)^{n-1} > np(1-pn) \geq f(1-f)$$

Analysis: Random Variables

Successful Round. A round r in which at least one honest party computes a block.

$$X_r = 1 \iff r \text{ is a successful round}$$
$$f := \mathbf{E}[X_r] = 1-(1-p)^n \approx pn$$

Uniquely Successful Round. A round r in which exactly one honest party computes a block.

$$Y_r = 1 \iff r \text{ is a uniquly successful round}$$
$$\mathbf{E}[Y_r] = np(1-p)^{n-1} > np(1-pn) \geq f(1-f)$$

Adversary. For each **query** j,

$$Z_j = 1 \iff \text{the adversary computed a block with his } j\text{-th query}$$
$$\mathbf{E}[Z_r] = \mathbf{E}[Z_1 + \cdots + Z_t] = \mathbf{E}[Z_r] = \mathbf{E}[Z_1] + \cdots + \mathbf{E}[Z_t] = pt$$

Chain-Growth Lemma

Chain-Growth Lemma. *Suppose that at round r an honest party has a chain of length ℓ. Then, by round $s \geq r$, every honest party has adopted a chain of length at least*

$$\ell + X_r + \cdots + X_{s-1}.$$

Chain-Growth Lemma

Chain-Growth Lemma. *Suppose that at round r an honest party has a chain of length ℓ. Then, by round $s \geq r$, every honest party has adopted a chain of length at least*

$$\ell + X_r + \cdots + X_{s-1}.$$

Chernoff Bound. *Suppose $\{X_i : i \in [n]\}$ are mutually independent Boolean random variables, with $\Pr[X_i = 1] = p$, for all $i \in [n]$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = pn$. Then, for any $\delta \in (0, 1]$,*

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2} \quad \text{and} \quad \Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}.$$

Chain-Growth Lemma

Chain-Growth Lemma. *Suppose that at round r an honest party has a chain of length ℓ. Then, by round $s \geq r$, every honest party has adopted a chain of length at least*

$$\ell + X_r + \cdots + X_{s-1}.$$

Chernoff Bound. *Suppose $\{X_i : i \in [n]\}$ are mutually independent Boolean random variables, with $\Pr[X_i = 1] = p$, for all $i \in [n]$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = pn$. Then, for any $\delta \in (0,1]$,*

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2} \quad \text{and} \quad \Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}.$$

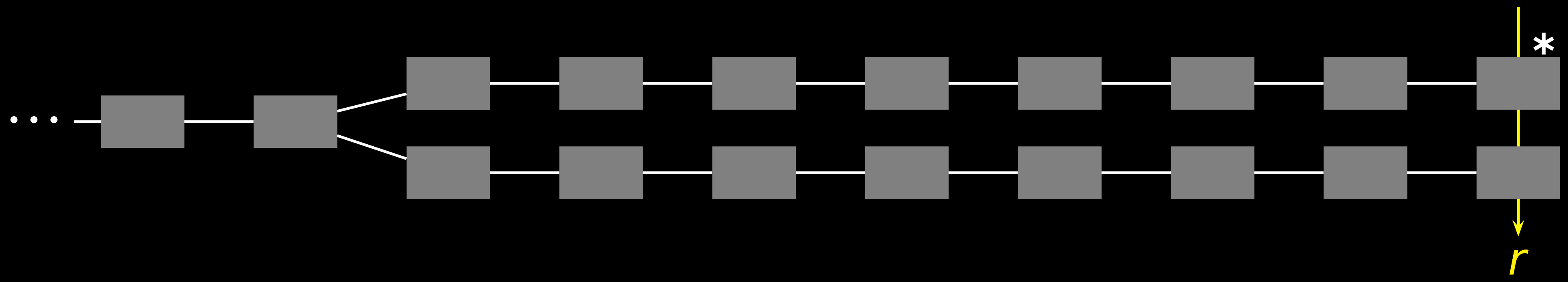
Chain-growth property. *With probability at least $1-e^{-\Omega(\epsilon^2 fs)}$, the chain of any honest party increases by at least*

$$(1-\epsilon)fs \approx (1-\epsilon)pns$$

blocks after s consecutive rounds. ($\mathbf{E}[X_1 + \cdots + X_s] = fs \approx pns$.)

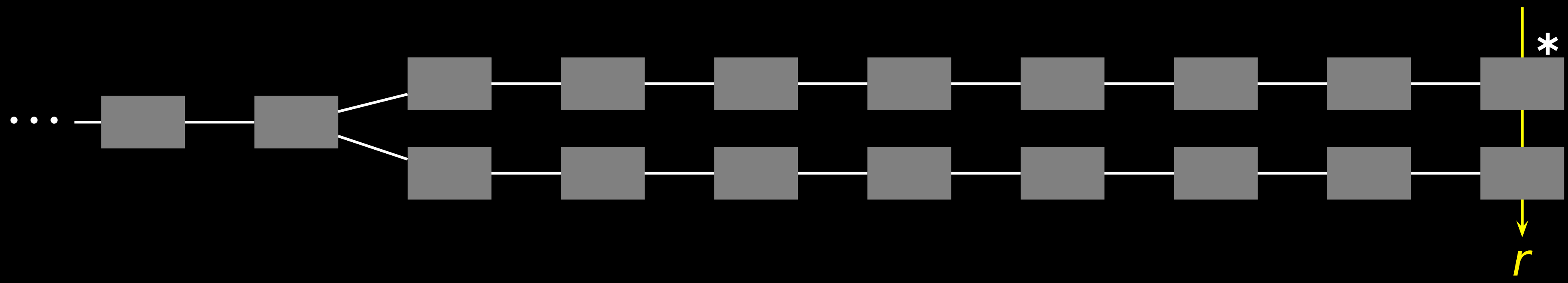
Common-Prefix Lemma

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.)*



Common-Prefix Lemma

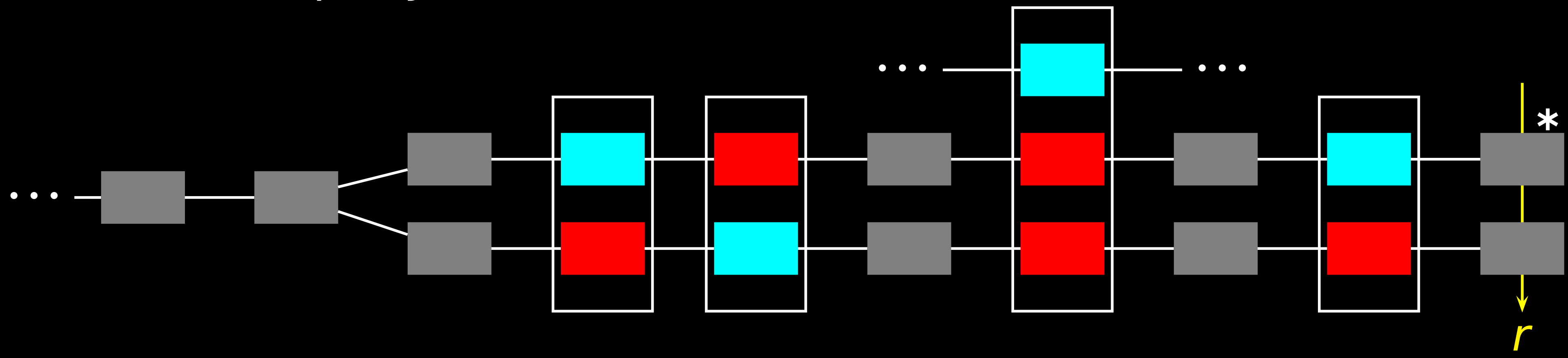
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



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]

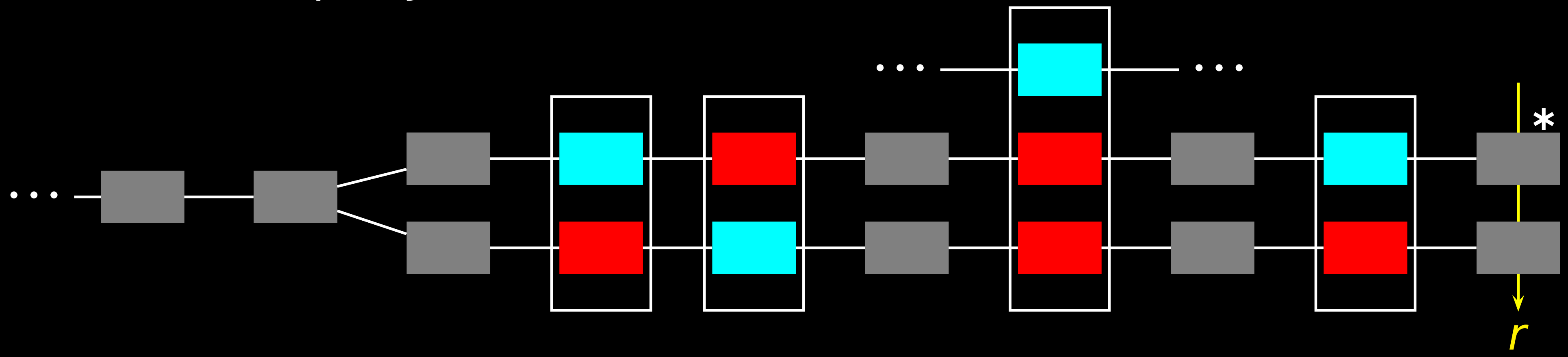
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last* k *blocks, is at most* $e^{-\Omega(k)}$. *(The party with the shortest chain should be honest.)*



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.)*

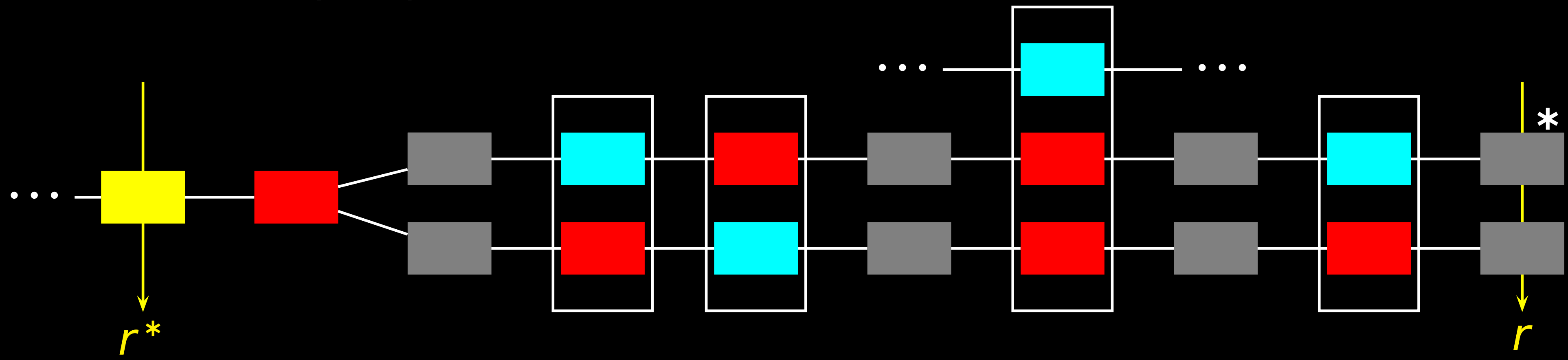


Observation. *Suppose the ℓ-the block of a chain was computed by an honest party in a uniquely successful round. Then any other ℓ-th block has been computed by the adversary.*

Proof. Suppose a block of height ℓ was computed by an honest party at a round u with $Y_u = 1$. If any honest party computed a block of height ℓ at any round $r < u$, then any honest party is trying to extend a chain of length at least ℓ at round u. Similarly for $r > u$.

Proof of the common-prefix lemma [GKL15]

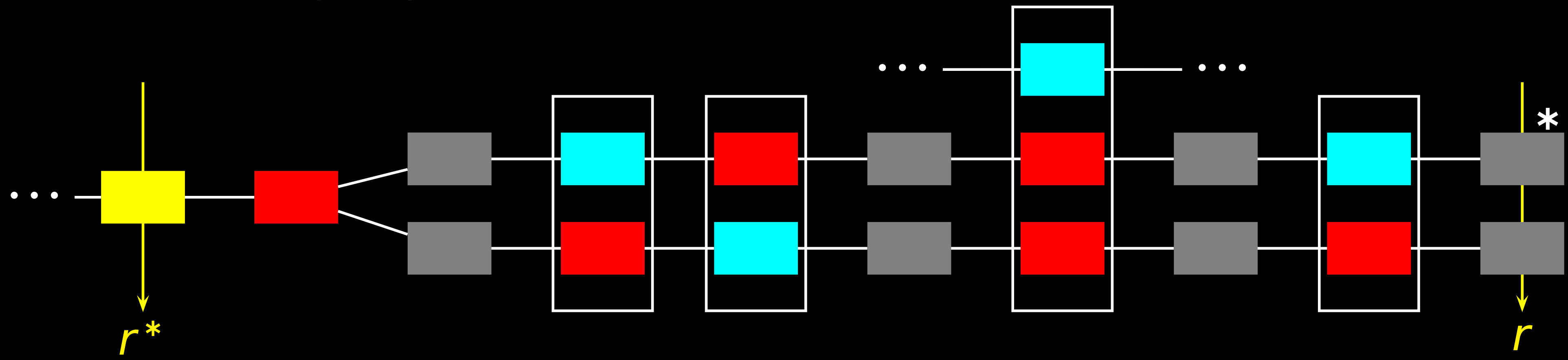
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last* k *blocks, is at most* $e^{-\Omega(k)}$. *(The party with the shortest chain should be honest.)*



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]

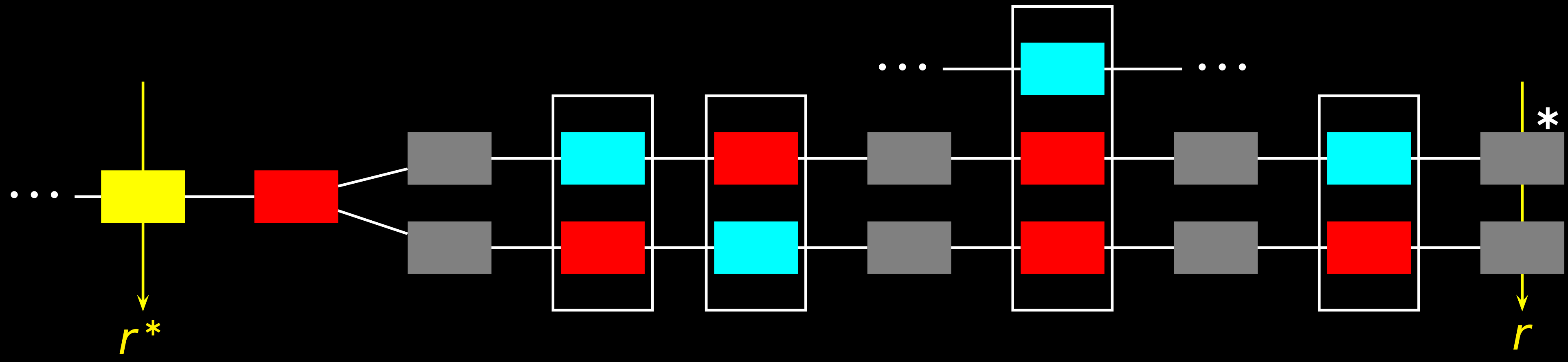
Common-Prefix Lemma. *The probability that at a given round two parties have chains that disagree in the last k blocks, is at most $e^{-\Omega(k)}$. (The party with the shortest chain should be honest.)*



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.)*

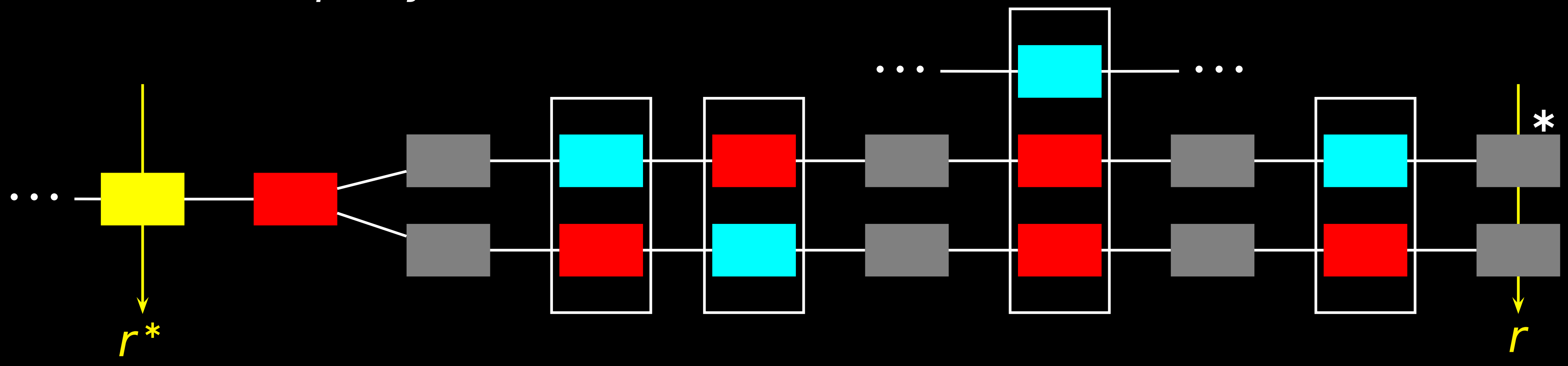


Proof. Let r^* be the last round before the fork that was computed by an honest party. Set $S = \{r^* + 1, \ldots, r - 1\}$. By the Lemma, to every uniquely successful round in S corresponds an adversarial block computed in S. It follows that

Uniqely successful rounds in S $\leq$ Adversarial successes 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 $S = \{r^*+1, ..., r-1\}$. By the Lemma, to every uniquely successful round in S corresponds an adversarial block computed in S. It follows that

Uniqely successful rounds in S	$\leq$	Adversarial successes in S.
$E[\sum Y_i] \approx pn(1-f)\lvert S\rvert$		$E[\sum Z_i] = pt\lvert S\rvert$.

Proof of the common-prefix lemma (cont'd)

Recall that $\mathbf{E}[Y_i] > f(1-f)$. Let $Y(S) = \sum_{r \in S} Y_r$. Then, since $\mathbf{E}[Y(S)] = \sum_{r \in S} f(1-f) = f(1-f)|S|$, by the Chernoff bound,

$$\Pr[Y(S) \leq (1-\epsilon) f(1-f)|S|] = e^{-\Omega(|S|)}.$$

Similarly

$$\Pr[Z(S) \geq (1+\epsilon) pt|S|] = e^{-\Omega(|S|)}.$$

Proof of the common-prefix lemma (cont'd)

Recall that $\mathbf{E}[Y_i] > f(1-f)$. Let $Y(S) = \sum_{r\in S} Y_r$. Then, since $\mathbf{E}[Y(S)] = \sum_{r\in S} f(1-f) = f(1-f)|S|$, by the Chernoff bound,

$$\Pr[Y(S) \leq (1-\epsilon)f(1-f)|S|] = e^{-\Omega(|S|)}.$$

Similarly

$$\Pr[Z(S) \geq (1+\epsilon)pt|S|] = e^{-\Omega(|S|)}.$$

Honest Majority Assumption. $t < (1-\delta)n$ *for* $\delta > 3\epsilon + 3f$.

Proof of the common-prefix lemma (cont'd)

Recall that $\mathbf{E}[Y_i] > f(1-f)$. Let $Y(S) = \sum_{r \in S} Y_r$. Then, since $\mathbf{E}[Y(S)] = \sum_{r \in S} f(1-f) = f(1-f)|S|$, by the Chernoff bound,

$$\Pr[Y(S) \leq (1-\epsilon)f(1-f)|S|] = e^{-\Omega(|S|)}.$$

Similarly

$$\Pr[Z(S) \geq (1+\epsilon)pt|S|] = e^{-\Omega(|S|)}.$$

Honest Majority Assumption. *$t < (1-\delta)n$ for $\delta > 3\epsilon + 3f$.*

Assuming these bad events don't occur (union bound) and the Honest Majority Assumption

$$\begin{aligned}
Z(S) &< (1+\epsilon)pt|S| \\
&< (1+\epsilon)(1-\delta)pn|S| && (t < (1-\delta)n) \\
&< (1+\epsilon)(1-\delta) \cdot \frac{f}{1-f} \cdot |S| && ((1-f)pn < f) \\
&< (1-\epsilon)f|S| && (\delta > 3\epsilon + 3f) \\
&< Y(S)
\end{aligned}$$

$\square$

Chain Quality

Chain Quality. *For any ℓ (sufficiently many) blocks in the chain of an honest party, the ratio of adversarial blocks is at most*

$$(1+\epsilon)\cdot\frac{t}{n}.$$

Chain Quality

Chain Quality. *For any* ℓ *(sufficiently many) blocks in the chain of an honest party, the ratio of adversarial blocks is at most*

$$(1+\epsilon)\cdot\frac{t}{n}.$$

Compare to the ideal ratio $t/(n+t)$.

Chain Quality

Chain Quality. *For any* ℓ *(sufficiently many) blocks in the chain of an honest party, the ratio of adversarial blocks is at most*

$$(1+\epsilon)\cdot\frac{t}{n}.$$

Corollary. *If* $t < (1-\epsilon)n$*, there is at least one honest block among any* ℓ *consecutive blocks in the chain of an honest party.*

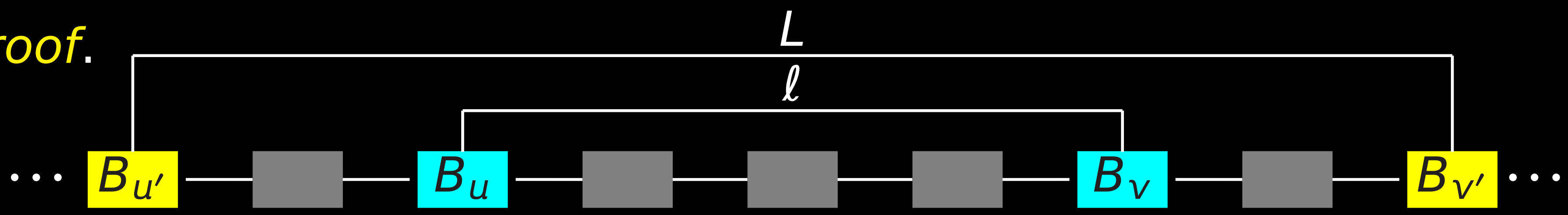
Proof. The ratio of adversarial blocks is less than $(1+\epsilon)(1-\epsilon) < 1$.

Chain Quality

Chain Quality. *For any ℓ (sufficiently many) blocks in the chain of an honest party, the ratio of adversarial blocks is at most*

$$(1+\epsilon)\cdot\frac{t}{n}.$$

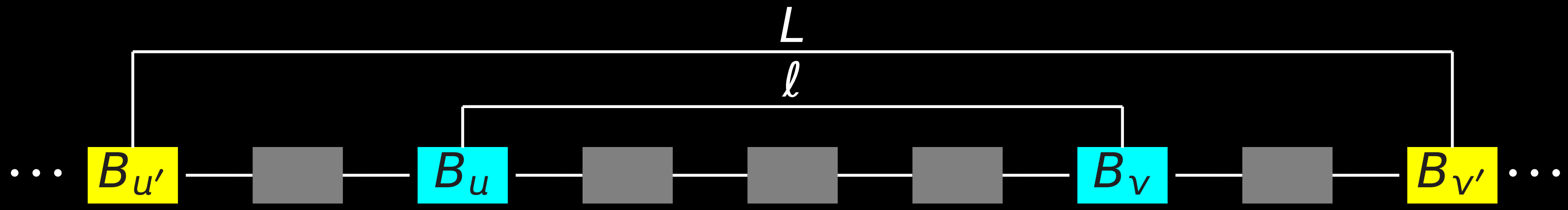
Proof.



- u' is greatest such that $B_{u'}$ was computed by an honest party.
- v' is least such that there exists a round at which an honest party was trying to extend the chain ending at block $B_{v'}$.
- r_1 is the round that $B_{u'}$ was created.
- r_2 first round that an honest party attempts to extend $B_{v'}$.
- $S = \{r : r_1 \leq r < r_2\}$.

Proof of Chain-Quality Property

Proof Cont'd.



We may assume that all the L blocks have been computed during the rounds in the set S.

- The number of successful rounds is at least $X \geq (1 - \frac{\epsilon}{3})pn|S|$.
- The number of adversarial blocks is at most $Z \leq (1 + \frac{\epsilon}{3})pt|S|$.
- Chain growth implies that $L \geq X$.
- The fraction of adversarial blocks is at most

$$\frac{Z}{L} \leq \frac{Z}{X} \leq \frac{1+\frac{\epsilon}{3}}{1-\frac{\epsilon}{3}} \cdot \frac{t}{n} \leq (1+\epsilon) \cdot \frac{t}{n}. \qquad \square$$

Tightness of Chain Quality

Theorem. *There exists an adversary such that, with probability at least* $1 - e^{-\Omega(\epsilon^2 \ell)}$ *($\ell = \Omega(1/\epsilon)$), there will be* ℓ *consecutive blocks in the chain of every honest party in which the fraction of adversarial blocks is at least*

$$\frac{t}{n} - 2\epsilon.$$

Tightness of Chain Quality

Theorem. *There exists an adversary such that, with probability at least* $1 - e^{-\Omega(\epsilon^2 \ell)}$ *($\ell = \Omega(1/\epsilon)$), there will be* ℓ *consecutive blocks in the chain of every honest party in which the fraction of adversarial blocks is at least*

$$\frac{t}{n} - 2\epsilon.$$

A selfish mining attack.

- The adversary keeps on extending a private chain.
- Whenever an honest party finds a solution, the (rushing) adversary releases one block from the private chain.
- If the private chain is depleted the adversary returns to the public chain.

Tightness of Chain Quality

Theorem. *There exists an adversary such that, with probability at least* $1 - e^{-\Omega(\epsilon^2 \ell)}$ *(*$\ell = \Omega(1/\epsilon)$*), there will be* ℓ *consecutive blocks in the chain of every honest party in which the fraction of adversarial blocks is at least*

$$\frac{t}{n} - 2\epsilon.$$

A selfish mining attack.

- The adversary keeps on extending a private chain.
- Whenever an honest party finds a solution, the (rushing) adversary releases one block from the private chain.
- If the private chain is depleted the adversary returns to the public chain.

Assumption. Ties between chains of equal length always favor the adversary.

Analysis of the Selfish Mining Attack

- Consider a set S of at least $\ell/(1-\epsilon)pn$ consecutive rounds.
  - This implies $X(S) \leq \ell$ (recall Chain-Growth Property).
- The number Z of adversarial blocks is at least $\frac{t}{n} \cdot \ell$.
- The number Z' of orphaned adversarial blocks computed in S is at most $\epsilon\ell$ with high probability.
  - k adversarial blocks may be orphaned, only if an honest party computes $k+1$ sequential blocks.
- The number Z'' of adversarial blocks not released in S is at most $\epsilon^2\ell$ with high probability.
  - k adversarial blocks are not released, only if no honest party computed a block in the meantime.

The ratio of adversarial blocks is at least

$$\frac{Z - Z' - Z''}{X} \geq \frac{\frac{t}{n} \cdot \ell - \epsilon\ell - \epsilon^2\ell}{\ell} \geq \frac{t}{n} - 2\epsilon \qquad \square$$

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 $v_i = x$).
- **Termination:** All honest processes should terminate.

Byzantine agreement (consensus)

A set of parties $\{1, \ldots, n\}$, t of which are controlled and coordinated by an adversary. Parties have inputs $x_1, \ldots, x_n \in \{0, 1\}$ and want to decide on outputs $v_1, \ldots, v_n$ 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 $v_i = x$).
- **Termination:** All honest processes should terminate.

Remark. *Note that n here is the total number of parties.*

Byzantine Agreement: Fundamental Results

- 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].
- Bit complexity is $\Omega(nt)$ [Dolev-Reischuk 1985].
- Fully Polynomial: There exists a protocol for all $t < \frac{n}{3}$, that terminates in $t+1$ rounds, and both computation and communication are polynomial in n. [Garay, Moses, "Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds." 1998]

Byzantine Agreement: Toy Proof

When 1 party out of n might be Byzantine, at least 2 rounds are needed.

- Upon receiving $00 \ldots 001$, an honest party should output 0.
  - Because of validity, since party p_n could be Byzantine.
- Upon receiving $00 \ldots 011$, an honest party should output 0.
  - Because party p_{n-1} could be Byzantine, and some parties might have received $00 \ldots 001$ and going to answer 0.
- Upon receiving $00 \ldots 0111$, an honest party should output 0.
  - Because party p_{n-2} could be Byzantine, and some parties might have received $00 \ldots 011$ and going to answer 0.

$$\vdots$$

- Upon receiving $01 \ldots 111$, an honest party should output 0.

Contradiction! Because the first party could be Byzantine.

Consensus: $t < n/2$ necessary (even with crypto)

Proof. On input $0 \ldots 0, 1 \ldots, 1$, where there are $n/2$ zeroes and $n/2$ ones and all parties are honest, the protocol terminates in one of the following three states.

A. All honest parties output 0.

B. All honest parties output 1.

C. Honest parties have mixed outputs.

The adversary chooses a strategy as follows.

- In case A, he corrupts the first half of parties and behaves honestly. Validity fails.
- In case B, he corrupts the second half of parties and behaves honestly. Validity fails.
- In case C, he does not corrupt any party. Agreement fails.

Consensus: $t < n/3$ necessary with delays

$A(mute)$ $B(0)$ $C(0)$	Since B and C have 0 and A might have crashed, at some time t_A parties B and C should terminate with 0.
$A(1)$ $B(mute)$ $C(1)$	Since A and C have 1 and B might have crashed, at some time t_B parties A and C should terminate with 1.
$A(1)$ $B(0)$ $C(*)$	Adversary C talks to A as if he has a 1 and to B as if he has a 0. Meanwhile, he holds messages $A \longleftrightarrow B$ for $t_C > t_A + t_B$ rounds.

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 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 diffuse it).
3) Each party decides on the output equal to the majority of the inputs recorded in the first k blocks.

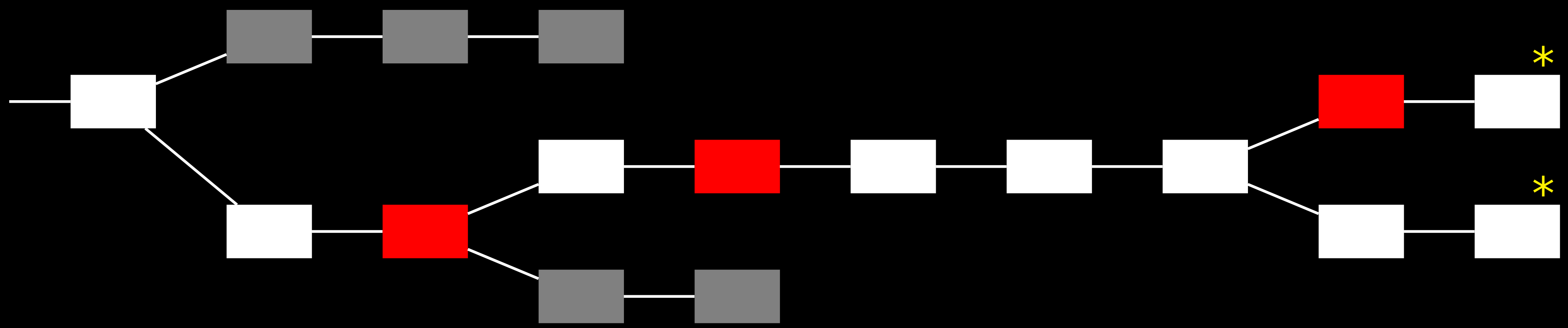
Byzantine Agreement Protocol

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 diffuse it).

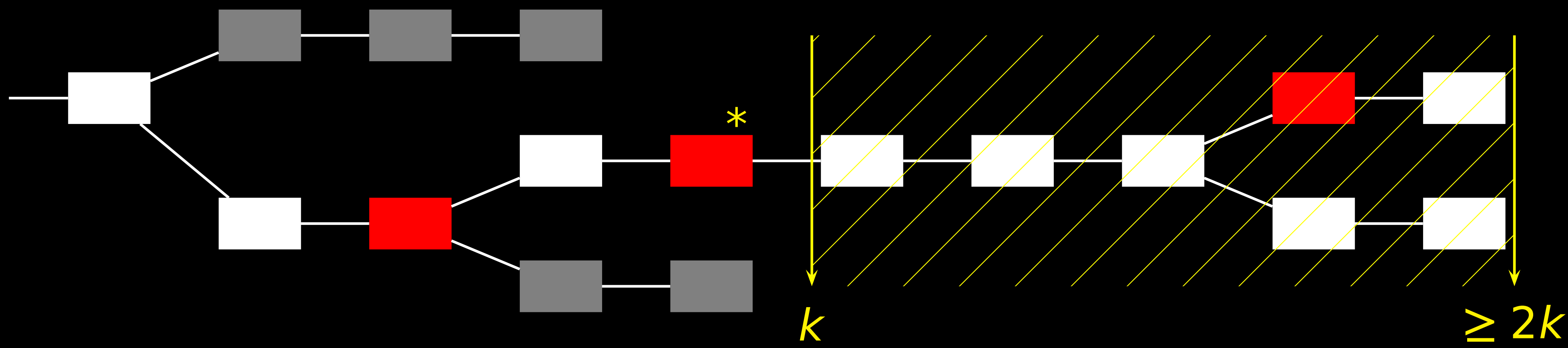
3) Each party decides on the output equal to the majority of the inputs recorded in the first k blocks.



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 diffuse 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.