- Take Home Midterm (Released Thursday)
- No Class on Thursday
- Office Hours moved to 3-5PM

Advanced Cryptography
CS 655

Week 8:

- SCRYPT (wrapup)
- Proof of Sequential Work/Proof of Space

Scrypt is maximally memory-hard

Joël Alwen
IST Austria

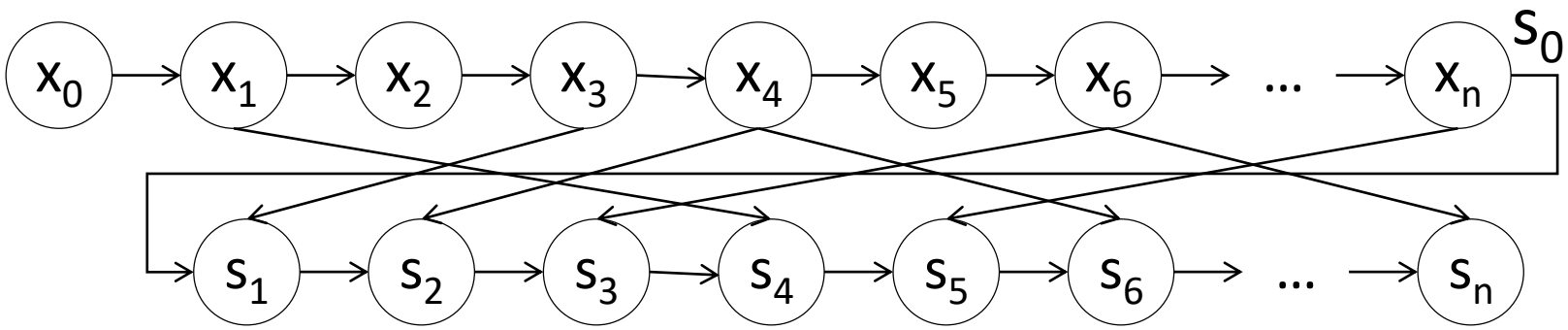
Binyi Chen
UCSB

Krzysztof Pietrzak
IST Austria

Leonid Reyzin
Boston U.
(work done at IST Austria)

Stefano Tessaro
UCSB

[Percival 2009]: scrypt



$H: \{0,1\}^* \rightarrow \{0,1\}^w$ random oracle

Input: x_0

Repeat n times: $x_i = H(x_{i-1})$

$s_0 = x_n$

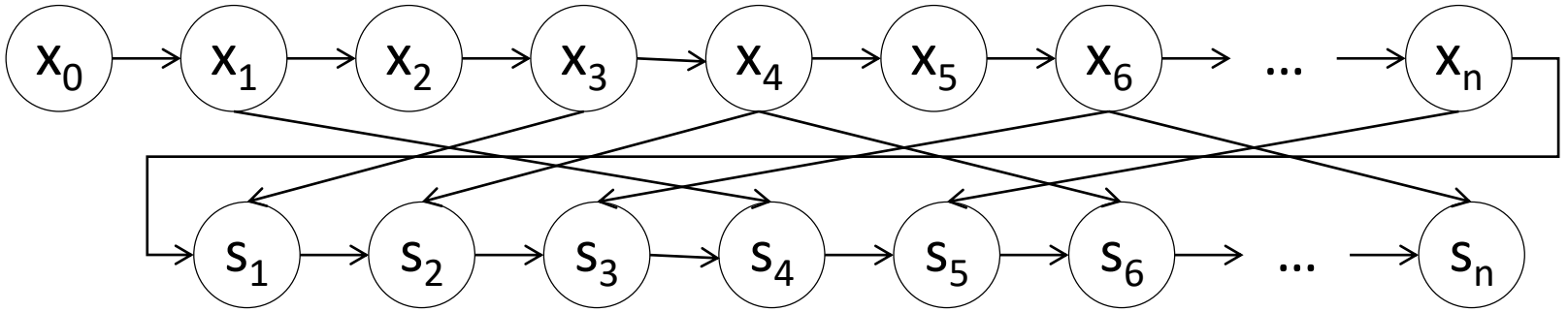
Repeat n times: $s_i = H(s_{i-1} \oplus x_j)$ for $\mathbf{j = s_{i-1} \bmod n}$

Output: s_n

Data-Dependent Memory Access
➔ Pebbling Attacks Don't apply

Our Result

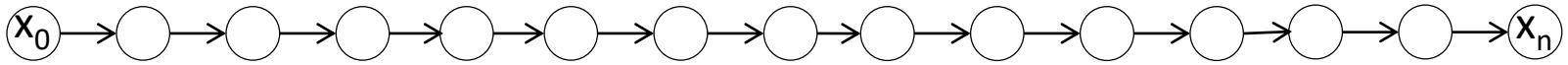
H: $\{0,1\}^* \rightarrow \{0,1\}^w$ random oracle



Theorem: in the parallel RO model, cc(scrypt) = $\Omega(n^2)$

The first ever construction works!

How quickly can you play this game?



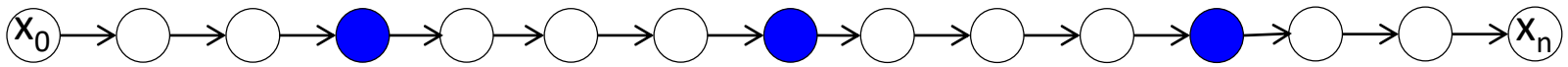
You have x_0 and whatever storage you want

I give you uniform challenge c from 1 to n

You return x_c

If you store nothing but x_0: n/2 H-queries per challenge

How quickly can you play this game?



You have x_0 and whatever storage you want

I give you uniform challenge c from 1 to n

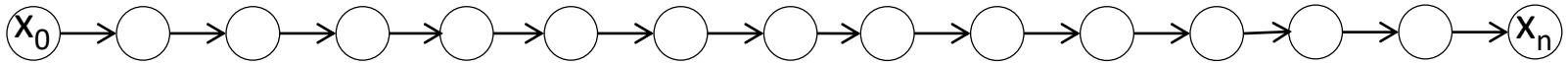
You return x_c

If you store nothing but x_0: n/2 H-queries per challenge

If you store p hash values: n/(2p) H-queries per challenge

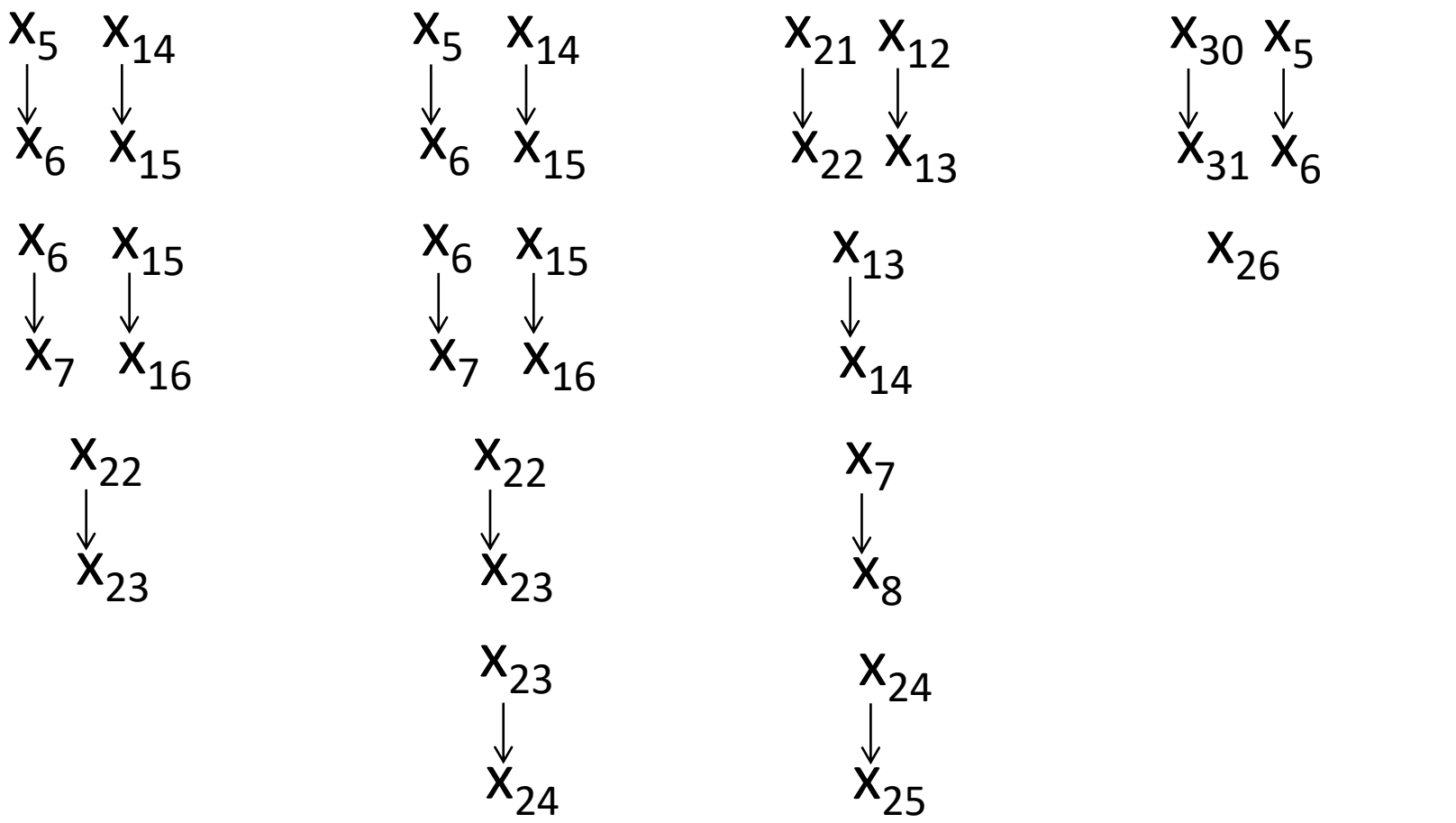
If you store something other than hash values?

Extracting labels from A's memory

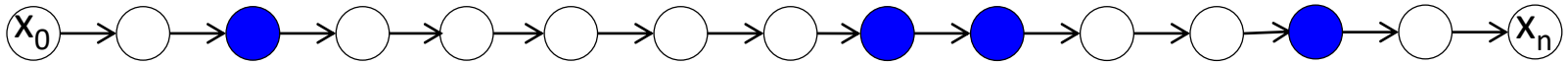


Imagine: run A on every possible challenge and record queries

… c=23 c=24 c=25 c=26 …



memory pw $\Rightarrow$ time $\geq n/(2p)$



Mark blue any label whose earliest appearance is not from H

…	c=23	c=24	c=25	c=26	…
	x_5 x_{14}	x_5 x_{14}	x_{21} x_{12}	x_{30} x_5	
	$\downarrow$ $\downarrow$	$\downarrow$ $\downarrow$	$\downarrow$ $\downarrow$	$\downarrow$ $\downarrow$	
	x_6 x_{15}	x_6 x_{15}	x_{22} x_{13}	x_{31} x_6	
	x_6 x_{15}	x_6 x_{15}	x_{13}	x_{26}	
	$\downarrow$ $\downarrow$	$\downarrow$ $\downarrow$	$\downarrow$		
	x_7 x_{16}	x_7 x_{16}	x_{14}		

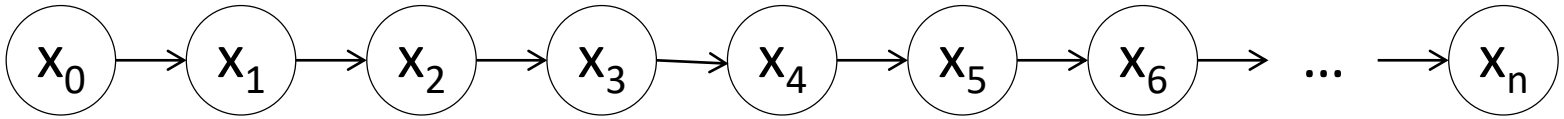
Lemma 1: all blue labels can be extracted from memory of A without querying H (so |blue set| $\leq$ p)

Lemma 2: Time to answer c $\geq$ distance from nearest blue

Conclusion: storage pw $\Rightarrow$ time $\geq n/(2p)$

How to go from this…

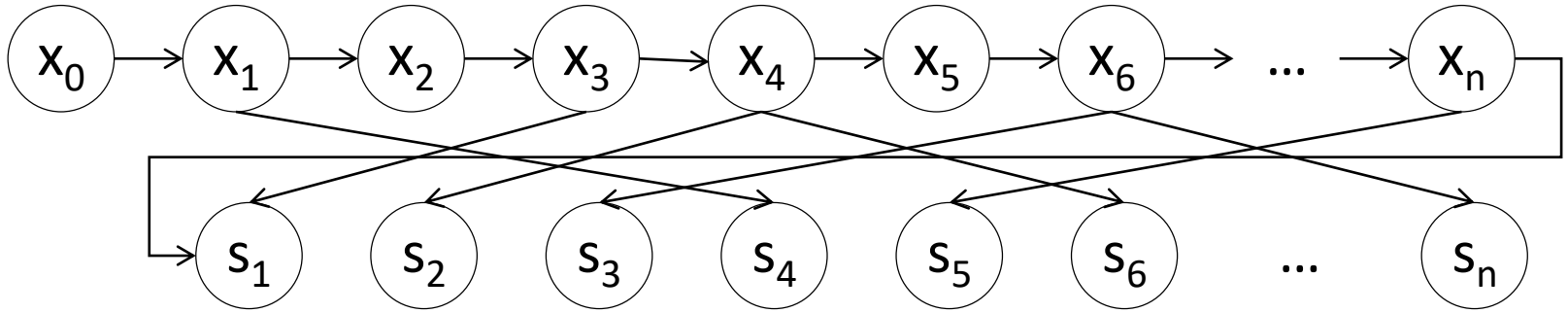
H: $\{0,1\}^* \rightarrow \{0,1\}^w$ random oracle



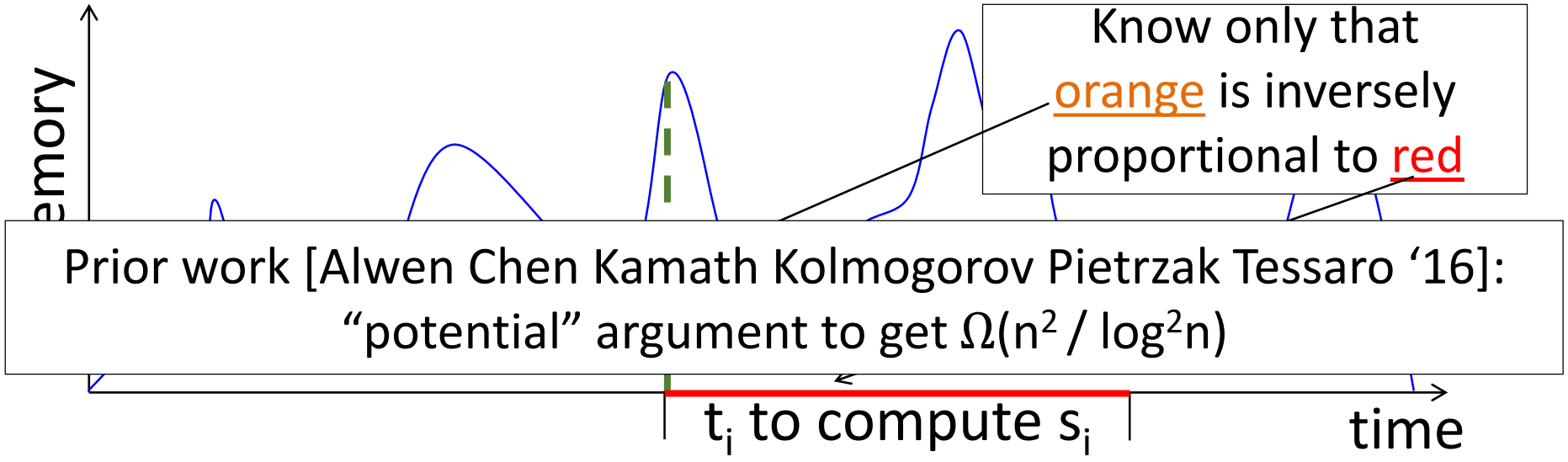
Single random challenge: memory $\geq \frac{nw}{2} \bullet \frac{1}{\text{time}}$

… to cc(n challenges)

H: $\{0,1\}^* \to \{0,1\}^w$ random oracle

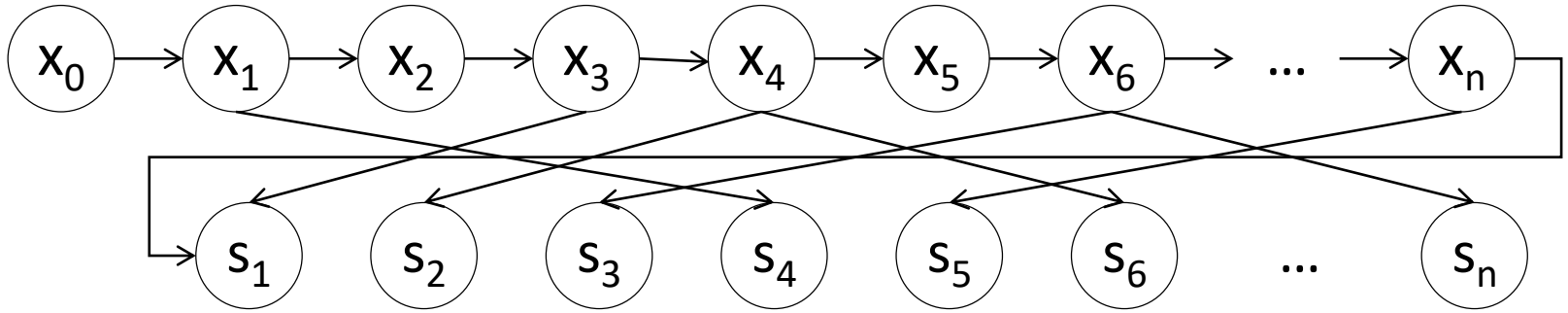


Single random challenge: memory $\geq \frac{nw}{2} \bullet \frac{1}{\text{time}}$

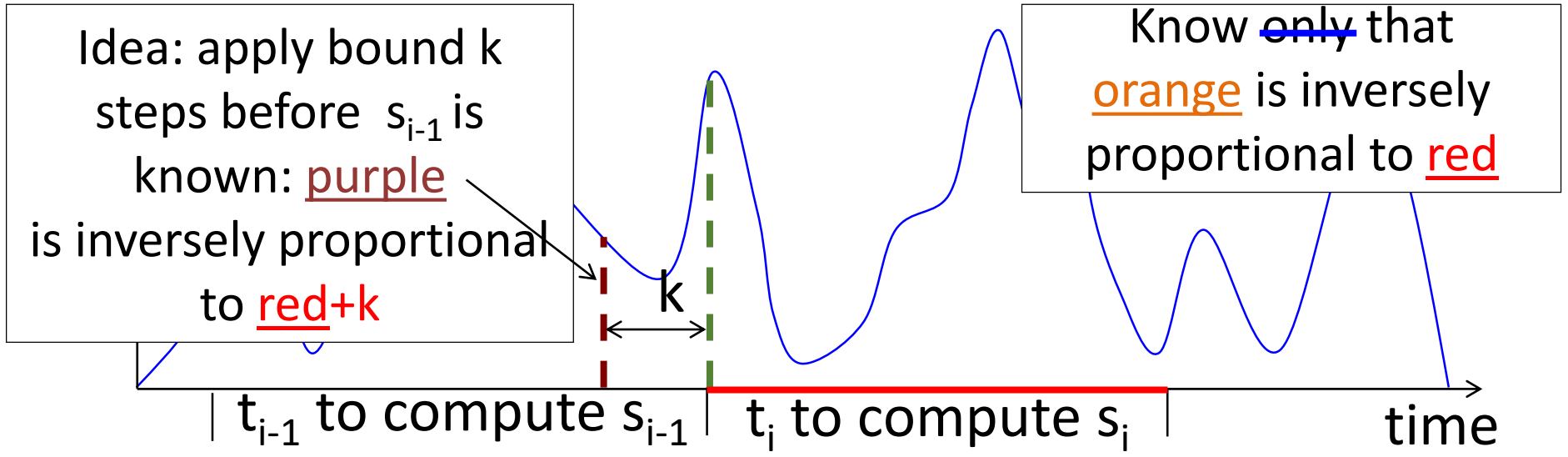


… to cc(n challenges)

H: $\{0,1\}^* \rightarrow \{0,1\}^w$ random oracle

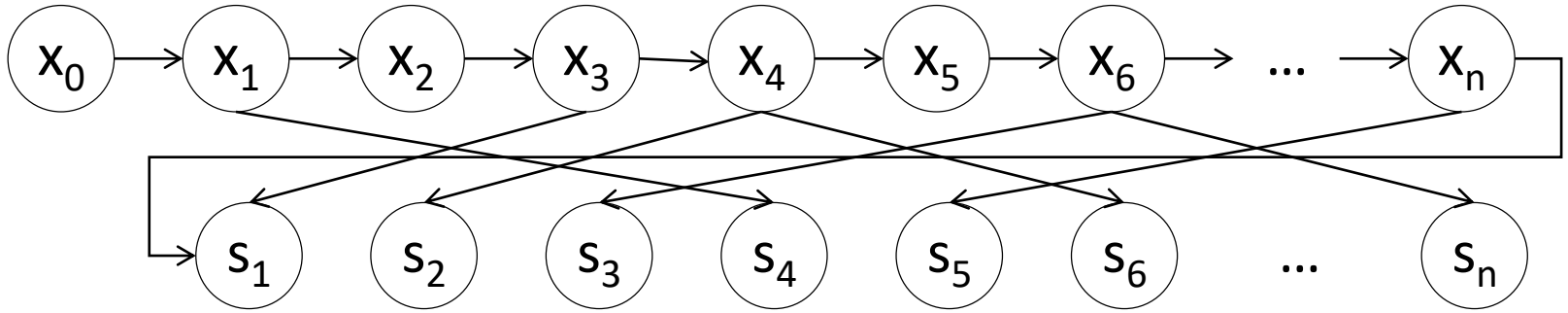


Single random challenge: memory $\geq \frac{nw}{2} \bullet \frac{1}{\text{time}}$

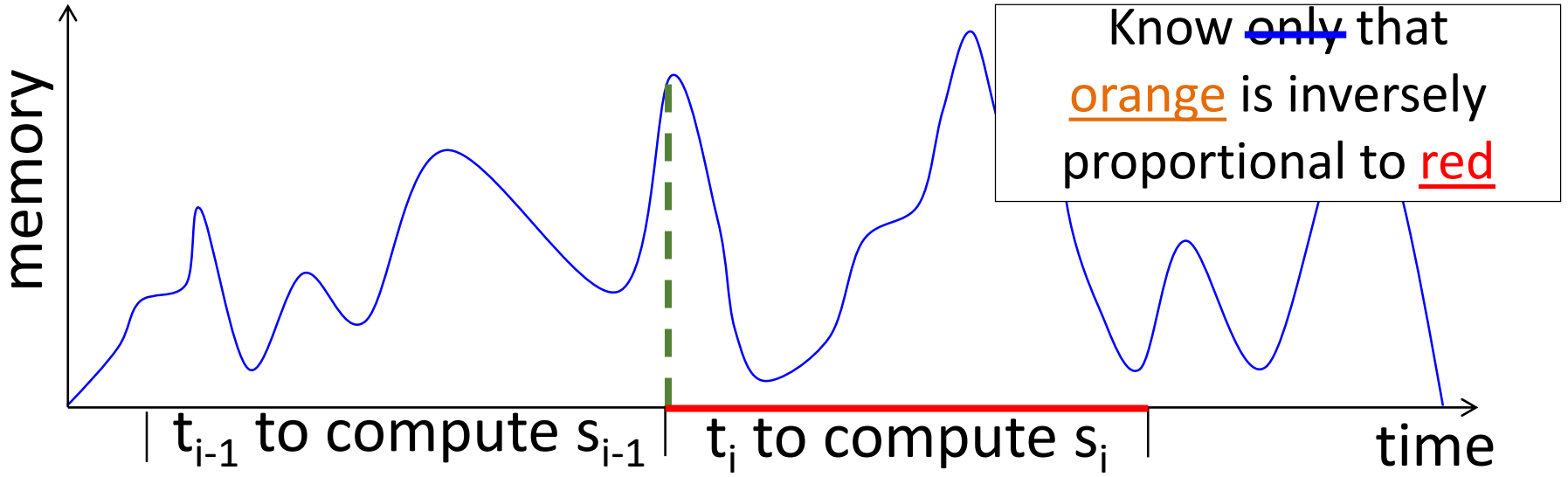


… to cc(n challenges)

H: $\{0,1\}^* \rightarrow \{0,1\}^w$ random oracle

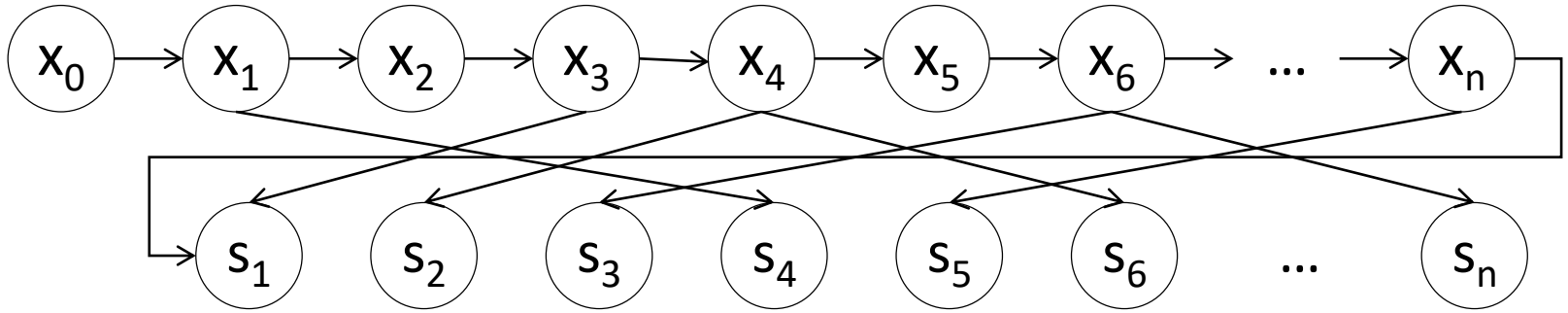


Single random challenge: memory $\geq \frac{nw}{2} \bullet \frac{1}{\text{time}}$

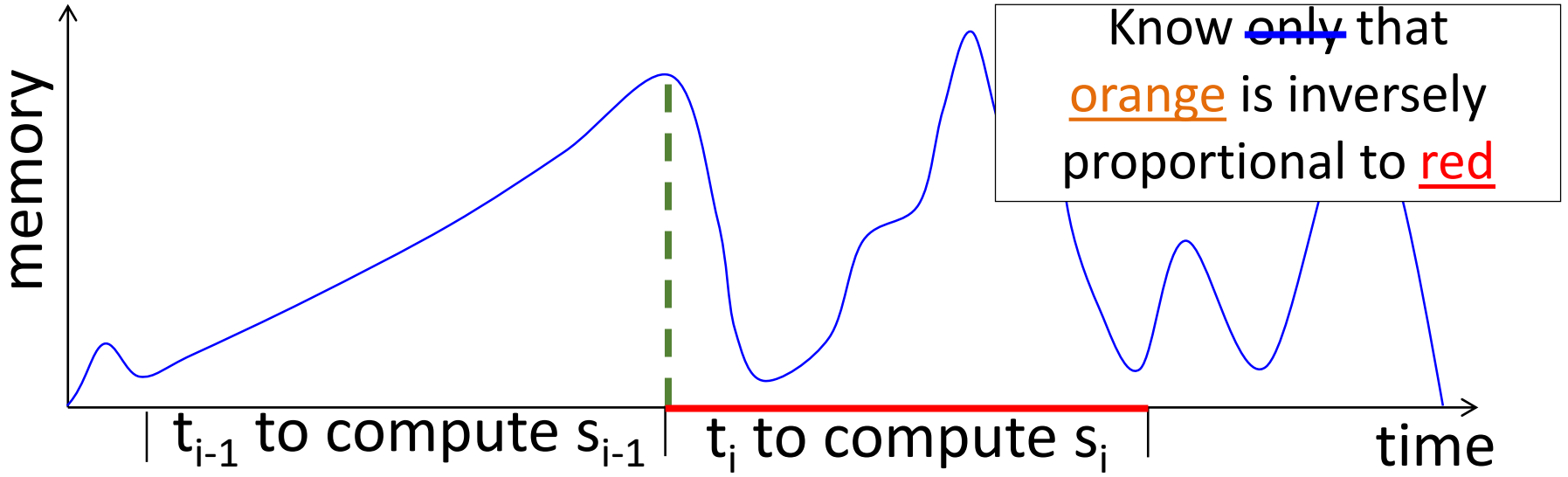


… to cc(n challenges)

H: $\{0,1\}^* \rightarrow \{0,1\}^w$ random oracle



Single random challenge: memory $\geq \frac{nw}{2} \bullet \frac{1}{\text{time}}$

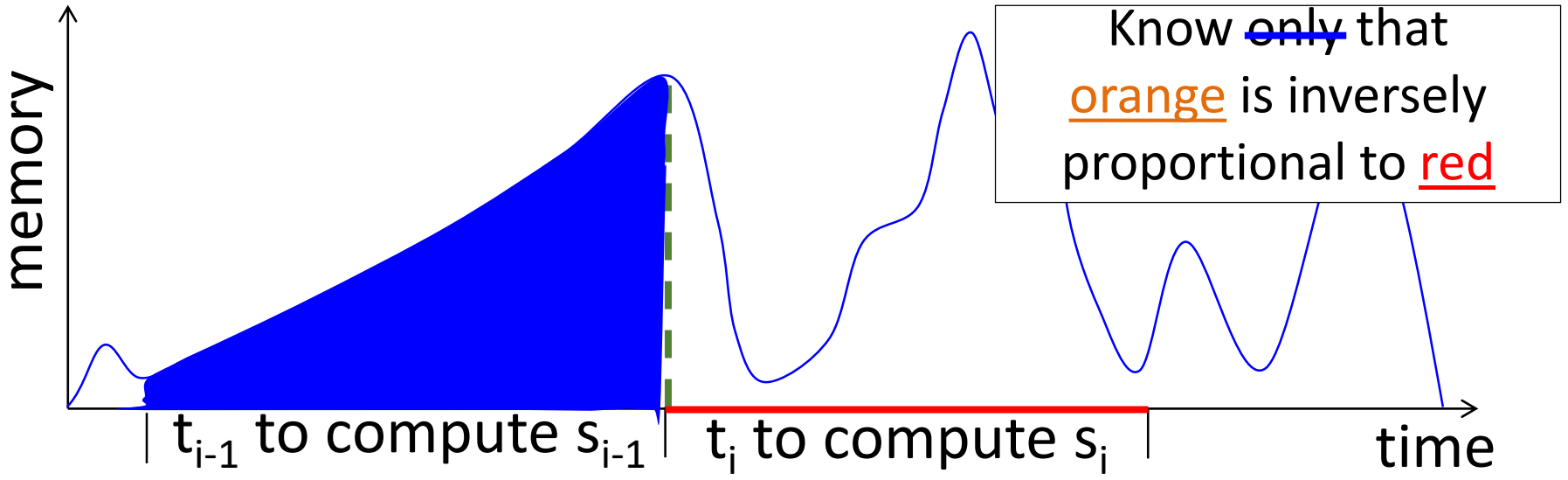


… to cc(n challenges)

Adding up memory used during previous challenge:

$$\frac{nw}{2}\left(\frac{1}{t_i}+\frac{1}{t_i+1}+\cdots+\frac{1}{t_i+t_{i-1}}\right)\geq\frac{nw}{2}\left(\ln\left(t_i+t_{i-1}\right)-\ln t_i\right)$$

Single random challenge: memory $\geq \frac{nw}{2}\bullet\frac{1}{\text{time}}$



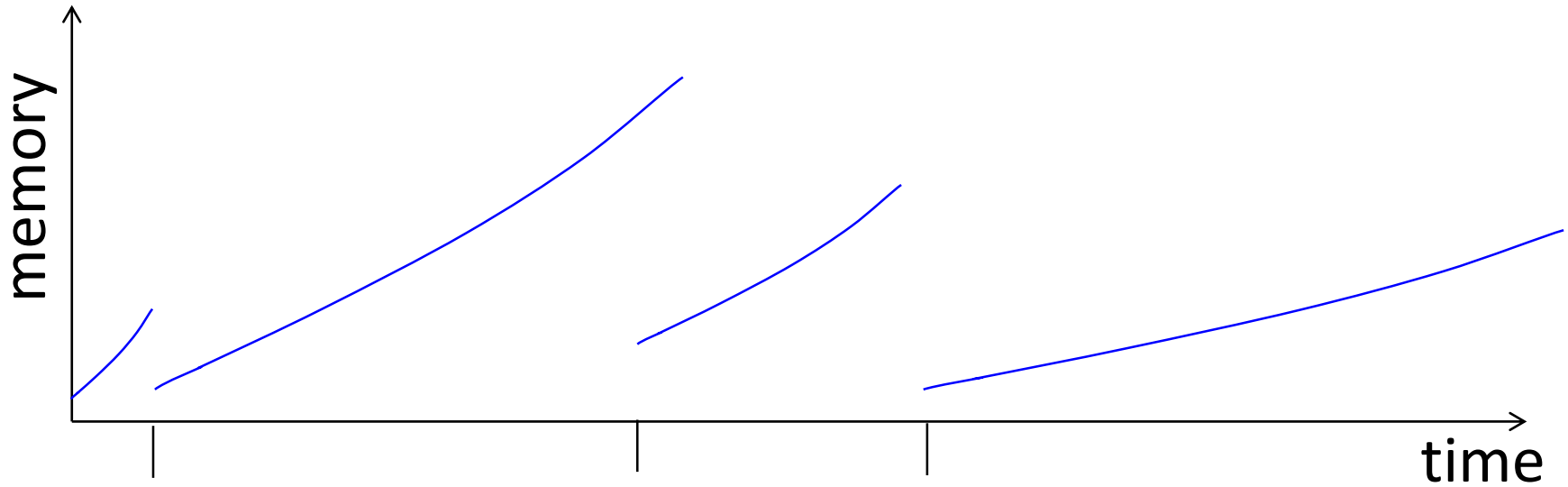
… to cc(n challenges)

Adding up memory used during previous challenge:

$$\frac{nw}{2}\left(\frac{1}{t_i}+\frac{1}{t_i+1}+\cdots+\frac{1}{t_i+t_{i-1}}\right)\geq\frac{nw}{2}\left(\ln\left(t_i+t_{i-1}\right)-\ln t_i\right)$$

Adding up over all challenges i from 1 to n:

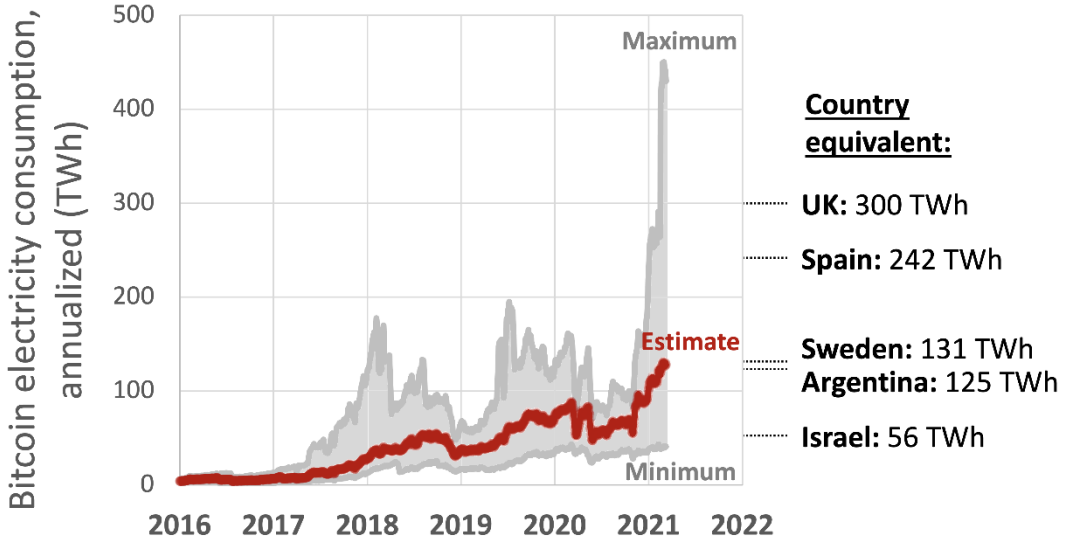
$$\tfrac{1}{2}nw\left(\ln\left(t_1+t_2\right)-\ln t_2+\ln\left(t_2+t_3\right)-\ln t_3+\ldots+\ln\left(t_{n-1}+t_n\right)-\ln t_n\right)$$
$$\geq \tfrac{1}{2}nw\,(n\ln 2)\geq\Omega(n^2w)$$



Mining Bitcoin (Proofs of Work)

Proofs of Space

- Proof of Work
  - Energy Intensive
  - Non-Egalitarian
    - Original Vision for Bitcoin: anyone can mine with idle cycles on PC
- Alternatives:
  - Proof of Stake (Democratic?)
  - Proof of Space
- Proof of Space Applications:
  - Distributed Consensus
  - Proofs of (Replicated Storage)



Technical Ingredient #2

Definition: A DAG G_n^{ε} is ε-extremely depth robust if it is (e,d)-depth-robust *for all* $e + d \leq (1 - \varepsilon)n$.

- **[NEW]** ε-extremely depth robust DAG $D_n^{\varepsilon}$with indegree O(log (n))
  - Construction: similar to [EGS75]
  - Many technical details to work out (see paper)

Useful Observation: *Any* subgraph of $D_n^{\varepsilon}[S]$ of size $|S| > \varepsilon n$ must contain a path of length $|S| - \varepsilon n$

Proof: Otherwise DAG D_n^{ε} is not (e, d)-depth robust for $\mathrm{d} = |S| - \varepsilon n$ and $e = |V \setminus S| = n - |S|$. Contradiction, D_n^{ε} is ε-extremely depth robust and

$$e + d = n - \varepsilon n \leq (1 - \varepsilon)n.$$

Let's Play an (Extreme) Pebbling Game

Definition: A DAG G_n^{ε} is ε-extremely depth robust if it is (e,d)-depth-robust *for all* $e + d \leq (1 - \varepsilon)n$.

Let G be a ε-extremely depth robust graph with 4N nodes.

You can place S pebbles on the graph (anywhere)

Challenger asks you to place a pebble on node 3N + c for a random challenge $1 \leq c \leq N$.

How fast can you respond to the challenge (in expectation)?

Let's Play a (Pebbling) Game

Definition: A DAG G_n^{ε} is ε-extremely depth robust if it is (e,d)-depth-robust *for all* $e + d \leq (1 - \varepsilon)n$.

Observation 1: there is a directed path P of length $4\mathrm{N} - \mathrm{S} - 4\varepsilon N$.

Observation 2: At least $\mathrm{N} - \mathrm{S} - 4\varepsilon N$ nodes in $[3N + 1, 4N]$ have depth at least $3\mathrm{N} - \mathrm{S} - 4\varepsilon N \geq N$ (assume $\mathrm{S} \leq \mathrm{N}$ and $4\varepsilon \leq 1$)

Observation 3: With probability at least $1 - \frac{\mathrm{S}}{\mathrm{N}} - 4\varepsilon$ we will take N rounds to respond to the challenge.

Non-Pebbling Game

Definition: A DAG G_n^{ε} is ε-extremely depth robust if it is (e,d)-depth-robust *for all* $e + d \leq (1 - \varepsilon)n$.

Let G be a ε-extremely depth robust graph with 4N nodes.

Let $L_1, \ldots, L_{4N}$ denote the labels of the graph G using random oracle $H(.)$
e.g., if $\text{parents}(v) = (u, w)$ then $L_v = H(Lu, L_w)$

You can store $S\lambda$ bits in memory

Challenger picks a random challenge $1 \leq c \leq N$ and asks you for label L_{3N+c} for.

How fast can you respond to the challenge (in expectation)?

Non-Pebbling Game

Definition: A DAG G_n^{ε} is ε-extremely depth robust if it is (e,d)-depth-robust *for all* $e + d \leq (1 - \varepsilon)n$.

Let $L_1, \ldots, L_{4N}$ denote the labels of the graph G using random oracle $H(.)$
e.g., if $\text{parents}(v) = (u, w)$ then $L_v = H(Lu, L_w)$

You can store $S\lambda$ bits in memory

Challenger picks a random challenge $1 \leq c \leq N$ and asks you for label L_{3N+c} for.

Extractor Argument: Can PROM algorithm into an equivalent pebbling strategy with $S(1 + o(1))$ pebbles.

Proof of Space

- Prover wants to convince verify that s/he has allocated N blocks of space e.g., storing labels $L_{3N+1}, \ldots, L_{4N}$

- Cheating prover may try to store $S < N\lambda(1 - 4\varepsilon)$ bits
  - Pebbling Reduction: Cheating prover cannot respond to
- Verifier can periodically challenge prover for label L_{3N+c} and the expects response quickly
  - With probability $1 - \frac{S}{N} - 4\varepsilon$ cheater cannot respond to a random challenge quickly
  - Multiple Challenges: Amplify probability cheating prover is caught
- **Question:** Does the verifier need to store all of the labels too?

Proof of Space

- Prover wants to convince verify that s/he has allocated N blocks of space e.g., storing labels $L_{3N+1}, \ldots, L_{4N}$
- **Question:** Does the verifier need to store all of the labels too?
- **Attempt 1:** Prover generates Merkle-Tree commitment to all labels $L_1, \ldots, L_{4N}$ and sends root ϕ to the verifier.
- **Problem?** What if the prover commits to the wrong labels e.g., labels that are easy to compress ?

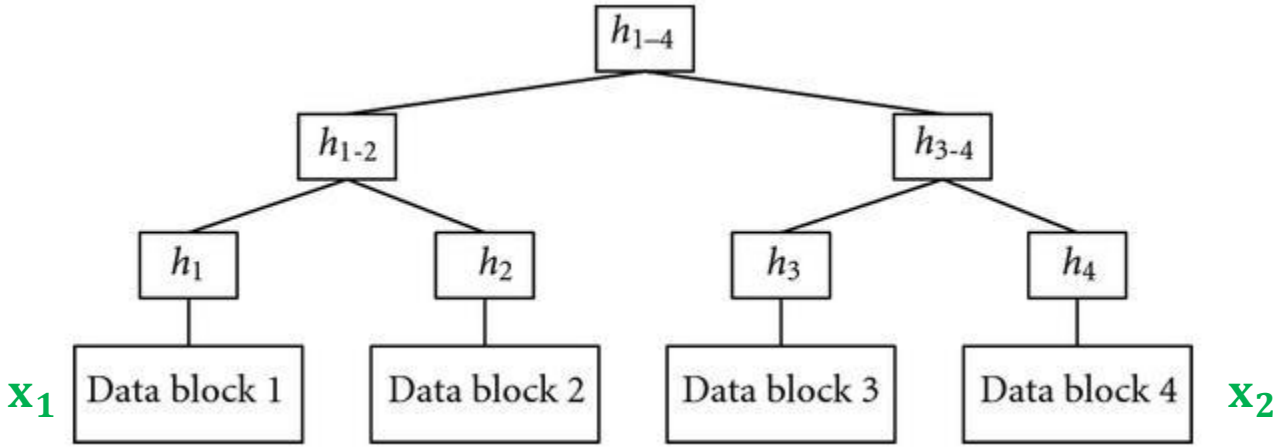
Merkle Trees

$$\mathbf{MT^s}(x) := \mathrm{h^s}(x)$$

$$\mathbf{MT^s}(\mathbf{x_1}, \ldots, \mathbf{x_{2^i}}) :=$$

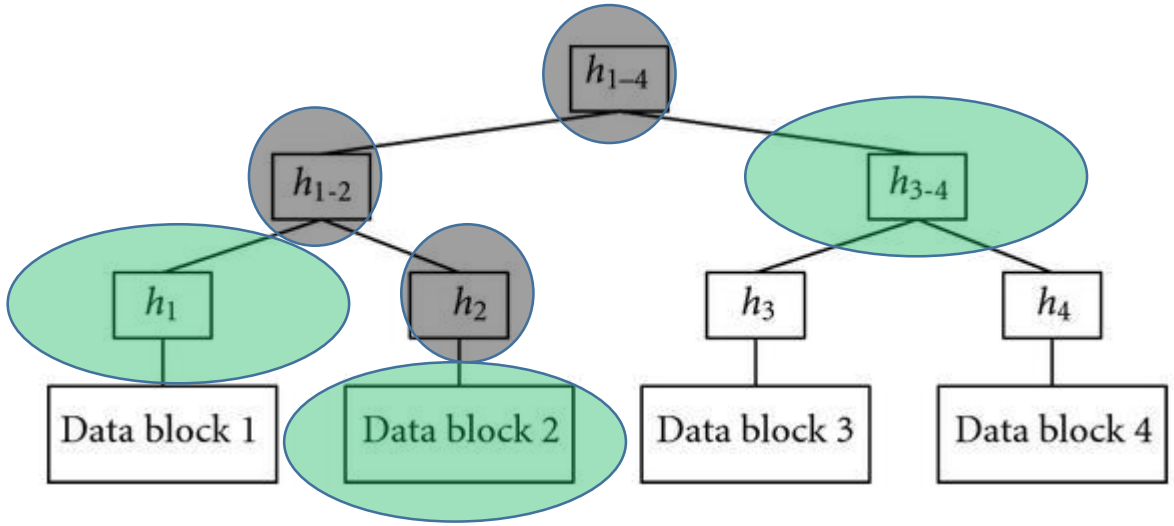
$$\mathrm{h^s}\left(\mathbf{MT^s}(\mathbf{x_1}, \ldots, \mathbf{x_{2^{i-1}}}), \mathbf{MT^s}(\mathbf{x_{2^{i-1}+1}}, \ldots, \mathbf{x_{2^i}})\right)$$

Theorem: Let (Gen, $\mathrm{h^s}$) be a collision resistant hash function then $\mathbf{MT^s}$ is collision resistant.



Merkle Trees

- **Proof of Correctness for data block 2**



- **Verify that root matches**
- **Proof consists of just log(n) hashes**
  - **Verifier only needs to permanently store only one hash value**

Proof of Space

- **Question:** Does the verifier need to store all of the labels too?

- **Solution:** Prover generates Merkle-Tree commitment to all labels $L_1, \ldots., L_{4N}$ and sends root ϕ to the verifier.
  - Verifier responds by picking k random nodes $1 \leq c_1, \ldots, c_k \leq 4N$
  - For each challenge c_i prover must reveal labels for node c_i and the labels for $\text{parents}(c_i) = \{v_1, \ldots, v_t\}$
  - Verifier validates Merkle Tree openings and checks that the labels are consistent e.g., $L_{c_i} = H(L_{v_1}, \ldots, L_{v_t})$

Proof of Space

- **Solution:** Prover generates Merkle-Tree commitment to all labels $L_1, \ldots, L_{4N}$ and sends root ϕ to the verifier.

- Suppose that εN labels are locally inconsistent

  ➔ cheater avoids detection with probability at most $\left(1 - \frac{\varepsilon}{4}\right)^k$

  ➔ Can make this probability negligible by setting $k = \frac{4\lambda}{\varepsilon}$

$$\left(1 - \frac{\varepsilon}{4}\right)^{\frac{4\lambda}{\varepsilon}} \leq e^{-\lambda}$$

Proof of Space

- **Solution:** Prover generates Merkle-Tree commitment to all labels $L_1, \ldots., L_{4N}$ and sends root ϕ to the verifier.

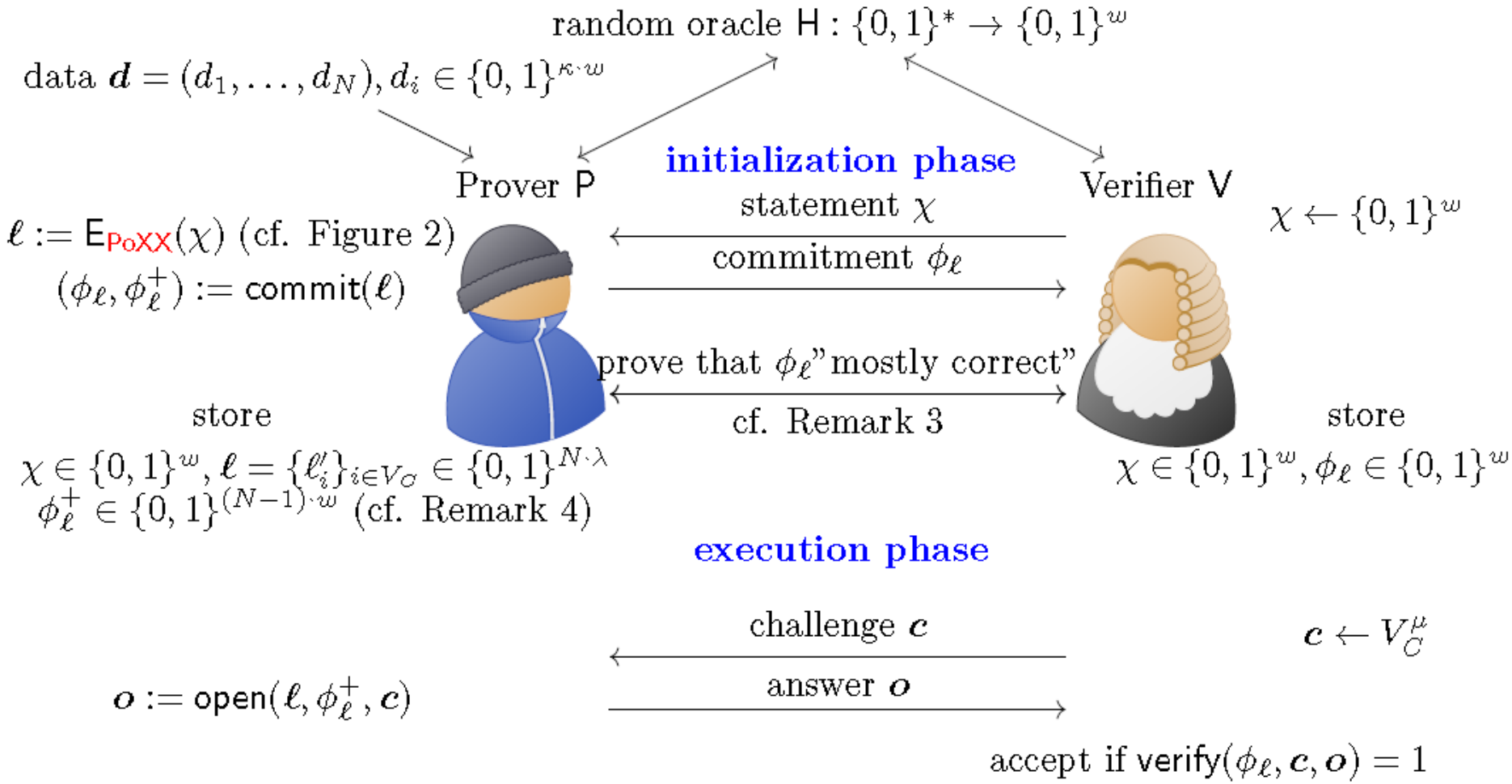
- Suppose that εN labels are locally inconsistent

- **Revisit Pebbling Argument:** Give the attacker S pebbles + allow the attacker to delete εN nodes from the graph (inconsistent labels)
- **Intuition:** with probability at least $\left(1 - \frac{\mathrm{S}}{\mathrm{N}} - 4\varepsilon\right) - \varepsilon = 1 - \frac{\mathrm{S}}{\mathrm{N}} - 5\varepsilon$ cheater cannot respond to a random challenge quickly

Proof of Space

- Prover wants to convince verify that s/he has allocated N blocks of space e.g., storing labels $L_{3N+1}, \ldots, L_{4N}$
- **Question:** Does the verifier need to store all of the labels too?
- **Solutions:** Prover generates Merkle-Tree commitment to all labels $L_1, \ldots, L_{4N}$ and sends root ϕ to the verifier.
  - Verifier responds by picking k random nodes $1 \leq c_1, \ldots, c_k \leq 4N$
  - For each challenge c_i prover must reveal labels for node c_i and the labels for $\text{parents}(c_i) = \{v_1, \ldots, v_t\}$
  - Verifier validates Merkle Tree openings and checks that the label is consistent e.g., $L_{c_i} = H(L_{v_1}, \ldots, L_{v_t})$

Proof of Space



Proof of Catalytic Space

- **Catalyst:** substance that increases rate of chemical reaction without itself undergoing any permanent chemical change

- **Idea:** You have stored large 1TB dataset which you do not access regularly, but you should not delete.

- **Proof of Catalytic Space:** Can participate in proof of work without permanently erasing dataset i.e., dataset can be recovered

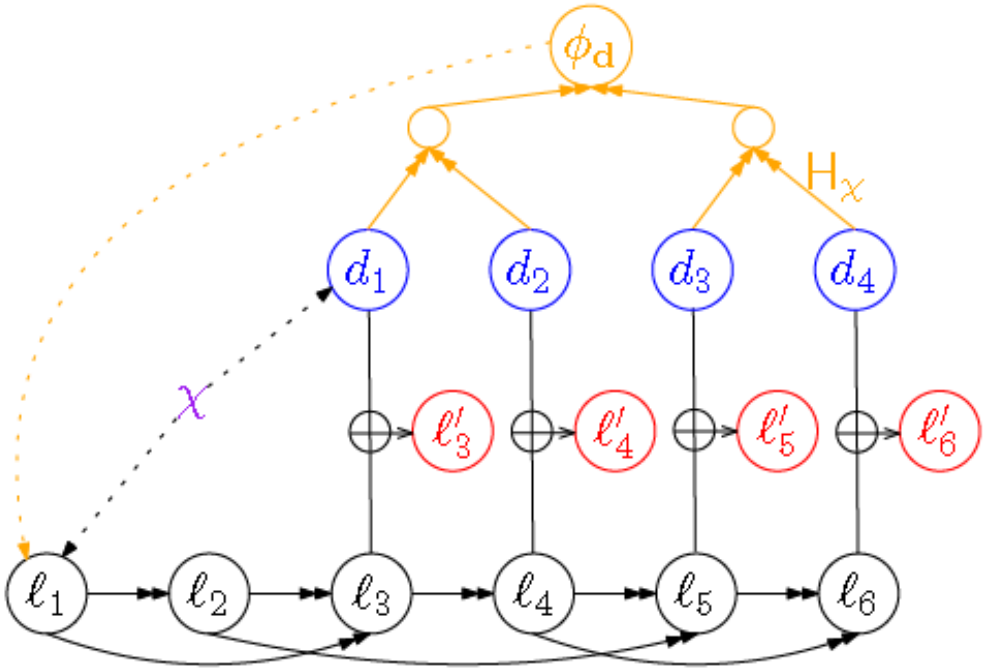
Proof of Catalytic Space

- Initial Input/Nonce: χ
- File: $d = (d_1, \ldots, d_N)$
- Merkle-Tree Commitment to d

Using $H_\chi(x) := H(\chi, x)$

Honest Prover Stores Red Labels

Can (slowly) recover file d
given red labels + nonce χ



Simple Proofs of Sequential Work

Bram Cohen

Krzysztof Pietrzak

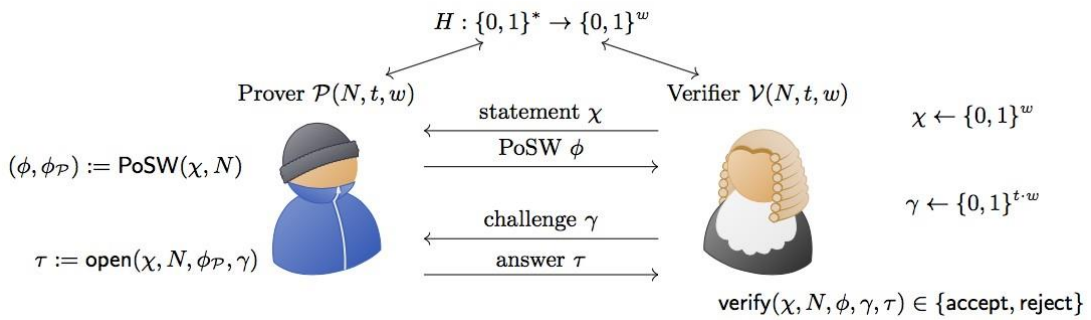
Eurocrypt 2018, Tel Aviv, May 1st 2018

Outline

- What Proofs of Sequential Work Sketch
- How of Construction & Proof
- Why Sustainable Blockchains

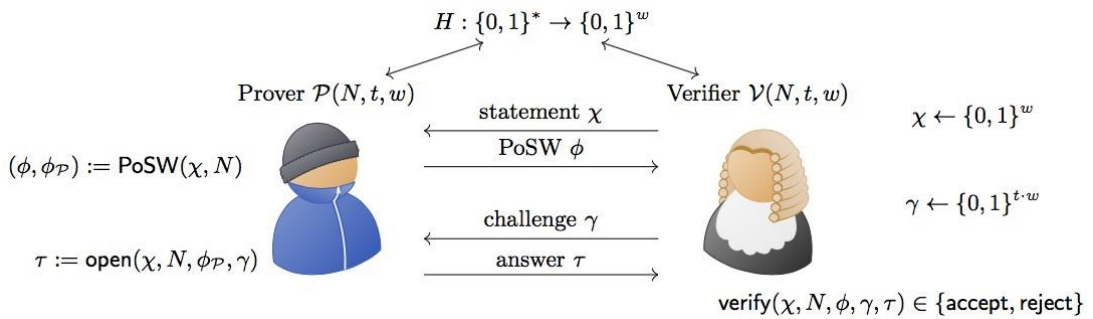
Outline

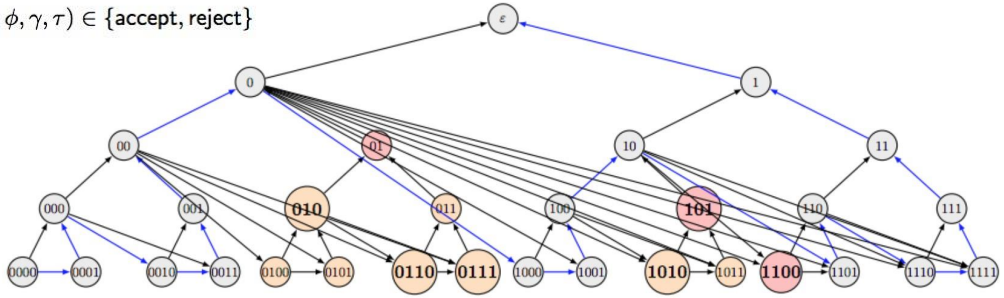
- What Proofs of Sequential Work Sketch
- How of Construction & Proof
- Why Sustainable Blockchains



Outline

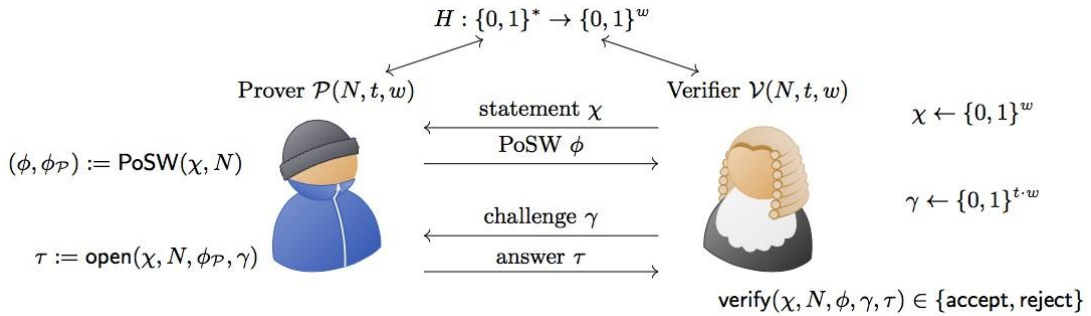
- What Proofs of Sequential Work Sketch
- How of Construction & Proof
- Why Sustainable Blockchains

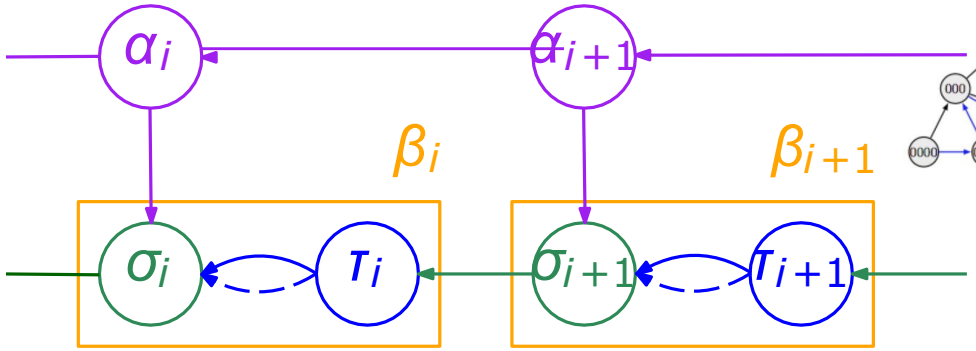


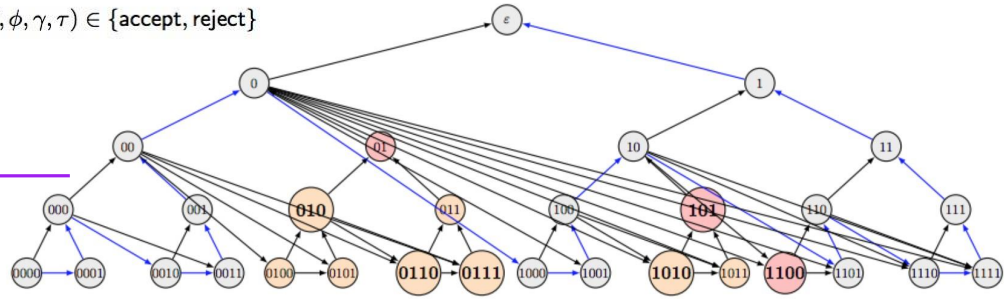


Outline

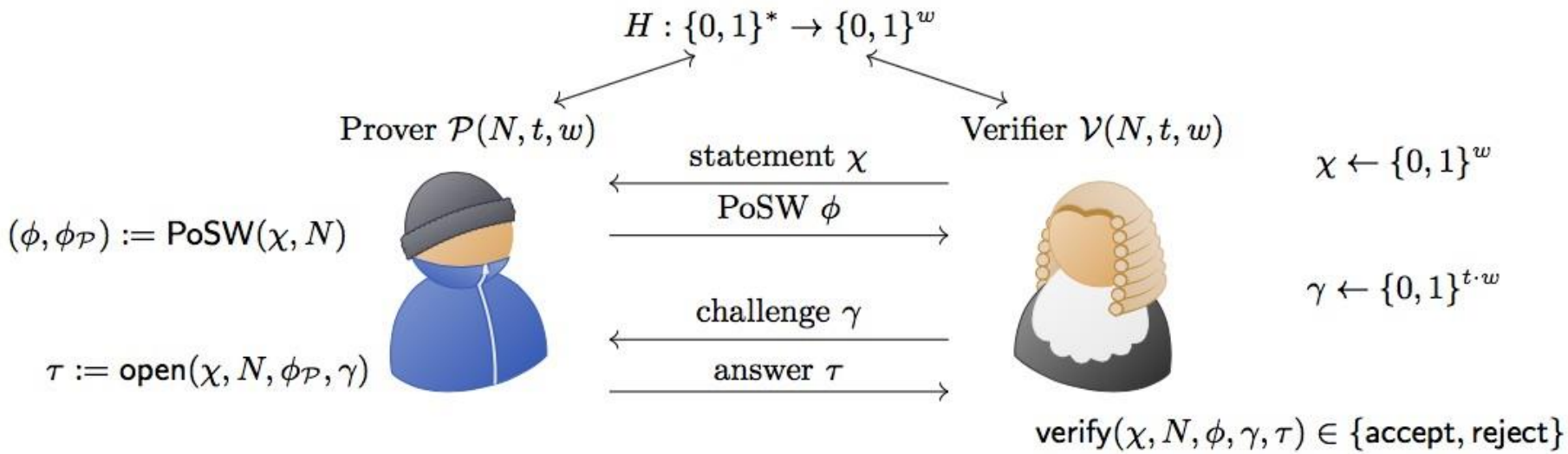
- What Proofs of Sequential Work Sketch
- How of Construction & Proof
- Why Sustainable Blockchains







Proofs of Sequential Work



Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

puzzle: $(N = p \cdot q, x, T)$, solution: $x^{2^T} \bmod N$

solution computed with two exponentiation given p, q:

$e \leftarrow 2^T \bmod \varphi(N)$, $x^{2^T} = x^e \bmod N$

conjectured to require T sequential squarings given only N

$x \rightarrow x^2 \rightarrow x^{2^2} \rightarrow \ldots x^{2^T} \bmod N$

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

puzzle: $(N = p \cdot q, x, T)$, solution: $x^{2^T} \bmod N$

solution computed with two exponentiation given p, q:

$e \leftarrow 2^T \bmod \varphi(N)$, $x^{2^T} = x^e \bmod N$

conjectured to require T sequential squarings given only N

$x \rightarrow x^2 \rightarrow x^{2^2} \rightarrow \ldots x^{2^T} \bmod N$

sequential computation ~
computation time ⇒
"send message to the future"

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody* Tal Morant Salil Vadhan+

February 18, 2013

PoSW vs. Time-Lock Puzzles

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody* Tal Moran† Salil Vadhan‡

February 18, 2013

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

Functionality

- Prove that time has passed
  ⇒ Non-interactive time-stamps
- Send message to the future

PoSW vs. Time-Lock Puzzles

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody* Tal Moran† Salil Vadhan‡

February 18, 2013

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

Functionality

- Prove that time has passed ⇒ Non-interactive time-stamps
- Send message to the future

Assumption

- Random oracle model or "sequential" hash-function
- Non-standard algebraic assumption

PoSW vs. Time-Lock Puzzles

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody* Tal Moran† Salil Vadhan‡

February 18, 2013

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

Functionality

- Prove that time has passed ⇒ Non-interactive time-stamps
- Send message to the future

Assumption

- Random oracle model or "sequential" hash-function
- Non-standard algebraic assumption

Public vs. Private

- Public-coin ⇒ Publicly verfiable
- Private-coin ⇒ Designated verifier

Proofs of Sequential Work

aka. Verifiable Delay Algorithm

Prover P

statement x

Time $T \in \mathrm{N}$

Verifier V

$x \leftarrow$ 

Proofs of Sequential Work

aka. Verifiable Delay Algorithm

Prover P

statement χ

Time $T \in \mathbb{N}$

$\pi = \pi(\chi, T)$

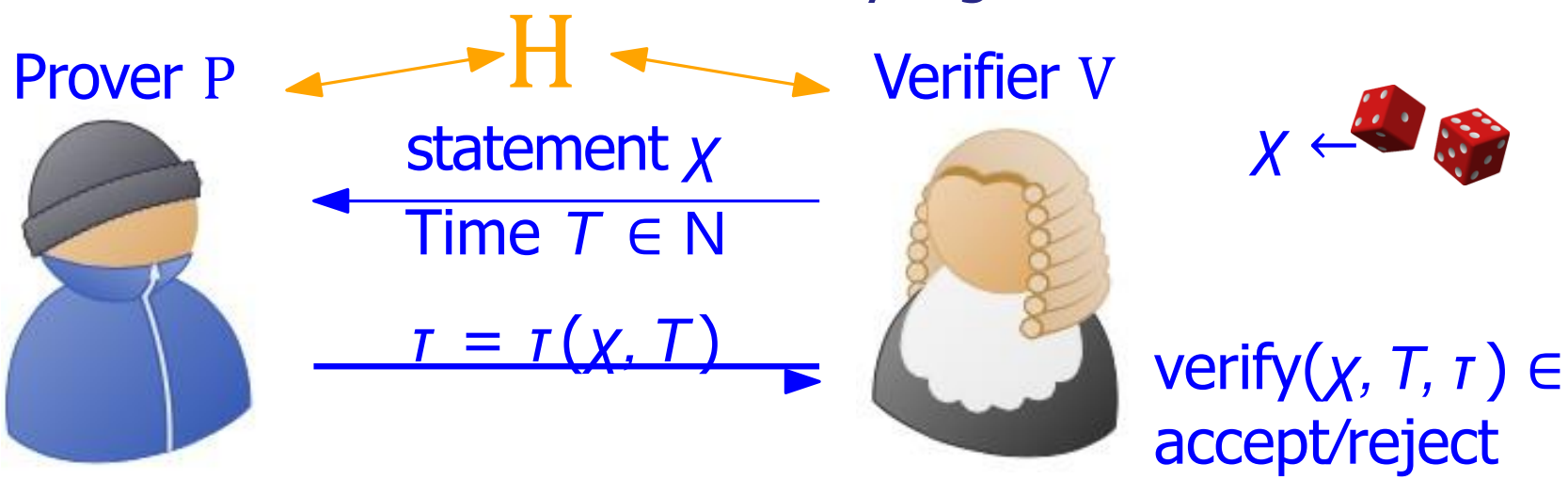
Verifier V

$\chi \leftarrow$ 

verify(χ, T, π) $\in$
accept/reject

Proofs of Sequential Work

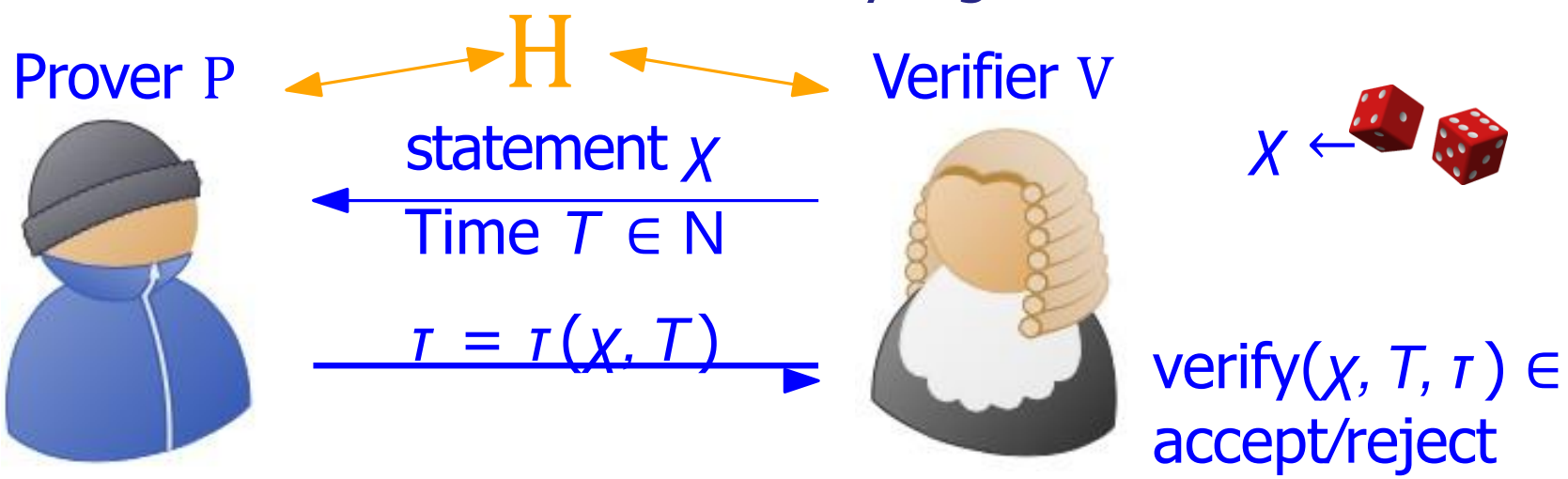
aka. Verifiable Delay Algorithm



Completeness and Soundness *in the random oracle model:*

Proofs of Sequential Work

aka. Verifiable Delay Algorithm



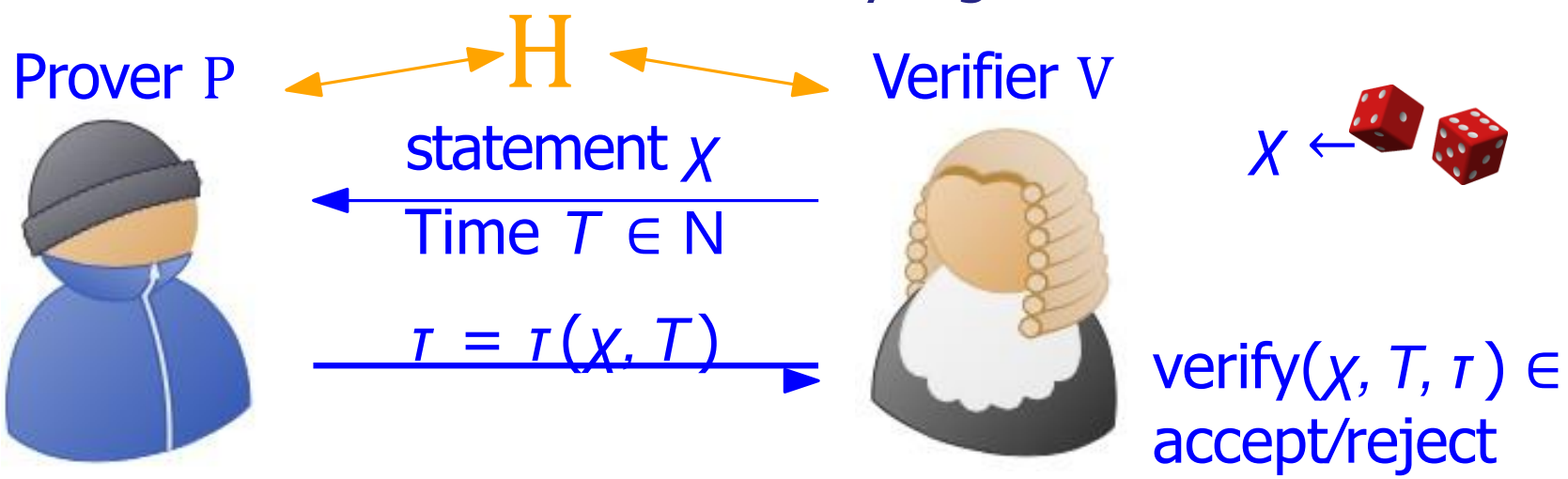
Completeness and Soundness *in the random oracle model:*

Completeness: $\tau(c, T)$ can be computed making T queries to H

Soundness: Computing any τ' s.t. verify(χ, T, τ') =accept for random χ requires almost T *sequential* queries to H

Proofs of Sequential Work

aka. Verifiable Delay Algorithm



Completeness and Soundness *in the random oracle model:*

Completeness: $\tau(c, T)$ can be computed making T queries to H

Soundness: Computing any τ' s.t. verify(χ, T, τ') =accept for random χ requires almost T *sequential* queries to H

massive parallelism useless to generate valid proof faster ⇒
prover must make almost T sequential queries ~ T time

Three Problems of the [MMV'13] PoSW

1) **Space Complexity :** Prover needs massive (linear in T) space to compute proof.
2) **Poor/Unclear Parameters** due to usage of sophisticated combinatorial objects.
3) **Uniqueness :** Once an accepting proof is computed, many other valid proofs can be generated (not a problem for time-stamping, but for blockchains).

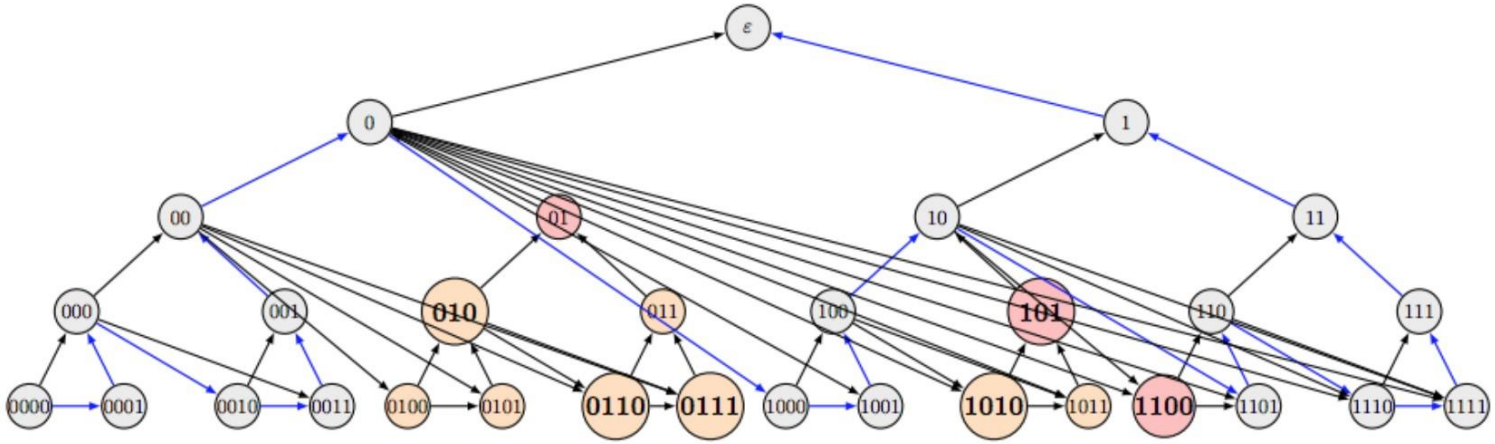
Three Problems of the [MMV'13] PoSW

1) **Space Complexity :** Prover needs massive (linear in T) space to compute proof.
2) **Poor/Unclear Parameters** due to usage of sophisticated combinatorial objects.
3) **Uniqueness :** Once an accepting proof is computed, many other valid proofs can be generated (not a problem for time-stamping, but for blockchains).

New Construction

1) Prover needs only $O(log(T))$ (not $O(T)$) space, e.g. for $T = 2^{42}$ ($\approx$ a day) that's $\approx$ $10KB$ vs. $\approx$ $1PB$.
2) Simple construction and proof with good concrete parameters.
3) Awesome open problem!

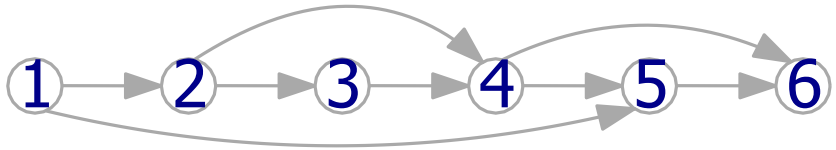
Construction and Proof Sketch



Three Basic Concepts

Three Basic Concepts

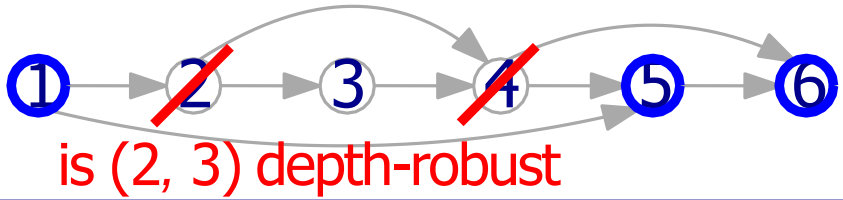
Depth-Robust Graphs (only [MMV'13])



DAG $G = (V, E)$ is (e, d) **depth-robust** if after removing any e nodes a path of length d exists.

Three Basic Concepts

Depth-Robust Graphs (only [MMV'13])

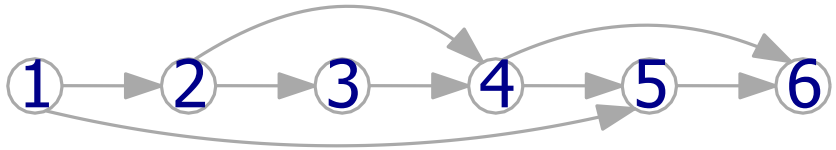


is (2, 3) depth-robust

DAG $G = (V, E)$ is (e, d) **depth-robust** if after removing any e nodes a path of length d exists.

Three Basic Concepts

Depth-Robust Graphs (only [MMV'13])



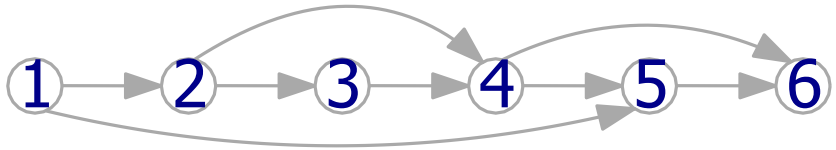
DAG $G = (V, E)$ is (e, d) **depth-robust** if after removing any e nodes a path of length d exists.

Graph **Labelling**

label $£_i = \mathrm{H}(£_{parents(i)})$, e.g. $£_4 = \mathrm{H}(£_3, £_4)$

Three Basic Concepts

Depth-Robust Graphs (only [MMV'13])

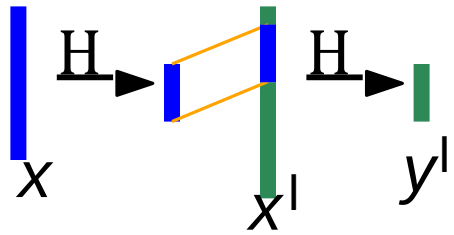


DAG $G = (V, E)$ is (e, d) **depth-robust** if after removing any e nodes a path of length d exists.

Graph **Labelling**

label $\pounds_i = \mathrm{H}(\pounds_{parents(i)})$, e.g. $\pounds_4 = \mathrm{H}(\pounds_3, \pounds_4)$

Random Oracles are **Sequential**



queries $y = \mathrm{H}(x)$, $y' = \mathrm{H}(x')$ where
$y \subseteq x' \Rightarrow$ query x' was made after x

y

The MMV'13 Construction

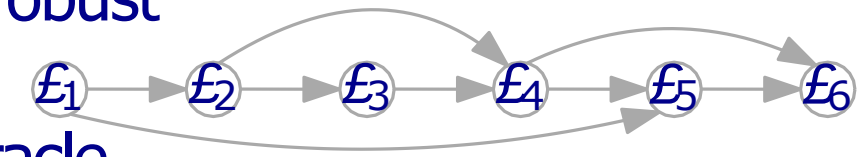
Prover P $\leftrightarrow$ H $\leftrightarrow$ Verifier V

statement χ

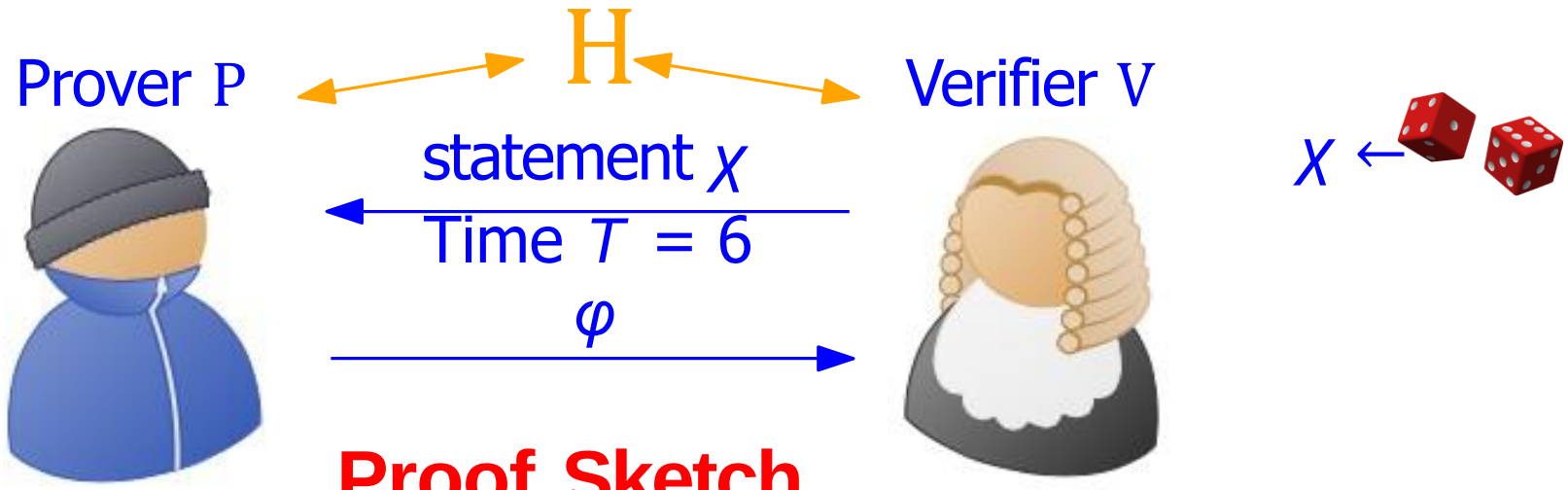
Time $T = 6$

$\chi \leftarrow$ 

- Protocol specifies depth-robust DAG G on T nodes
- Define "fresh" random oracle $\mathrm{H}_\chi(\cdot) \equiv \mathrm{H}(\chi\,\cdot)$
- Compute labels of G using H_χ

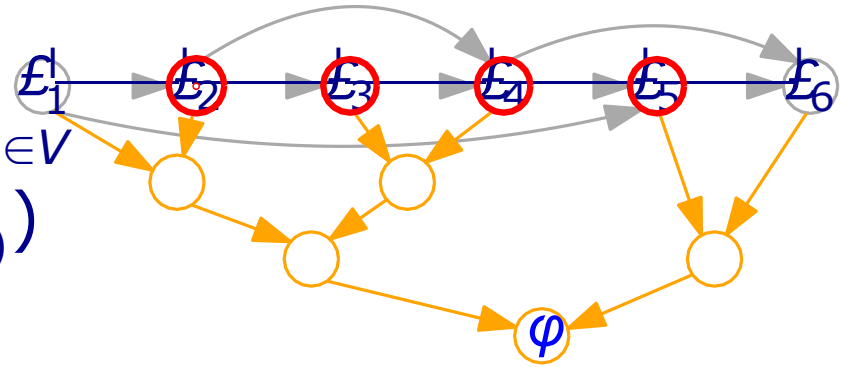


The MMV'13 Construction



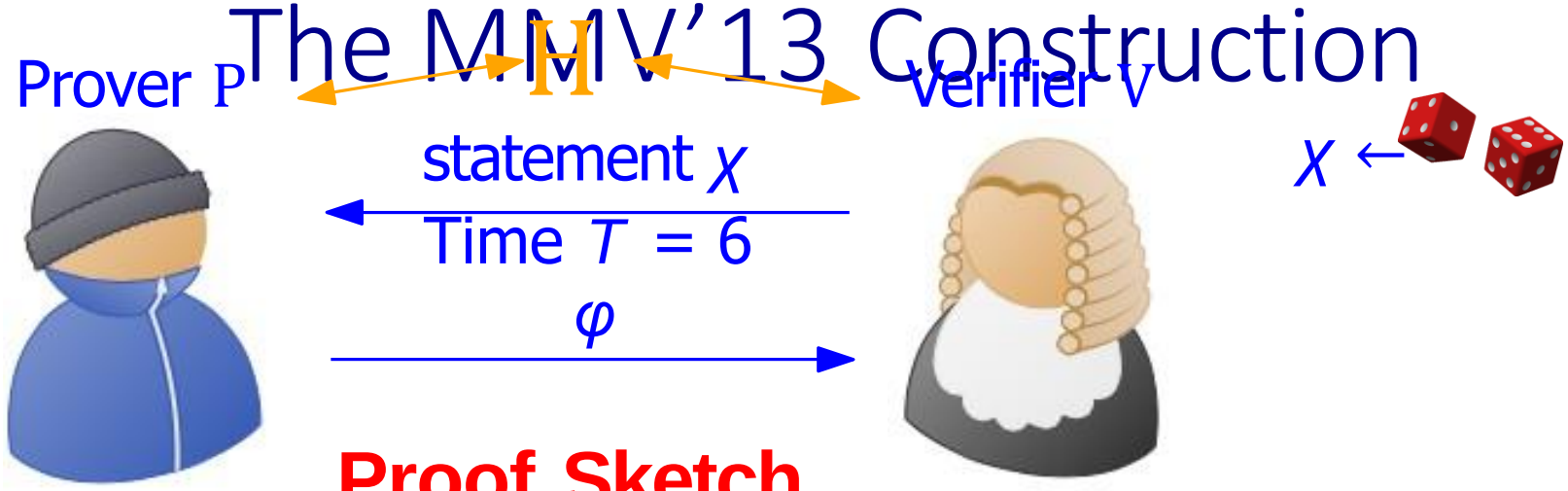
Proof Sketch

- G is (e, d) depth-robust
- φ commits $\tilde{P}$ to labels $\{\pounds'_i\}_{i \in V}$
- i is **bad** if $\pounds'_i \neq H(\pounds'_{parents(i)})$



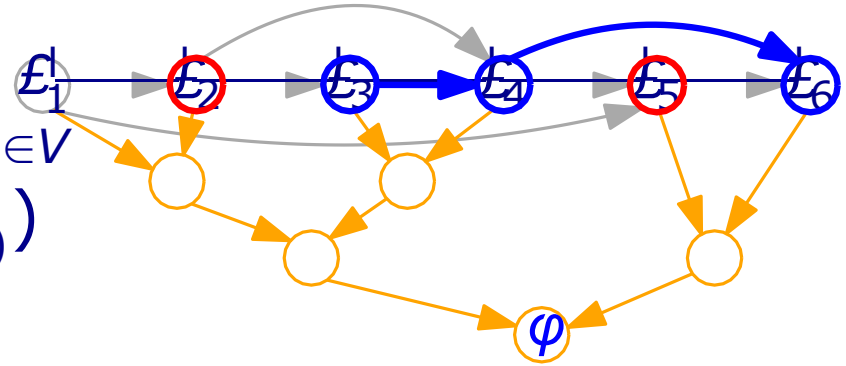
- Case 1: ≥ e bad nodes ⇒ will fail opening phase whp.

The MMV'13 Construction



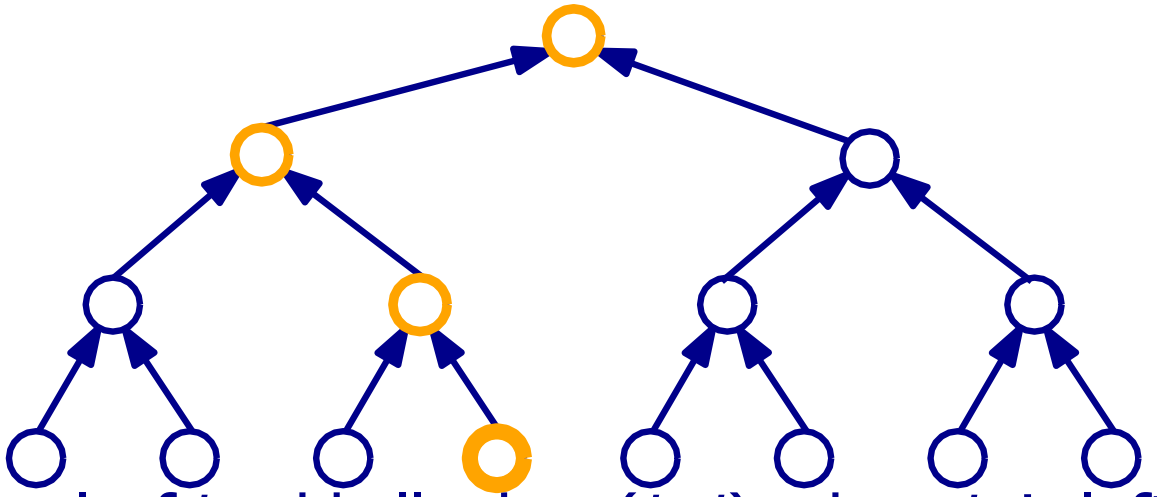
Proof Sketch

- G is (e, d) depth-robust
- φ commits $\tilde{P}$ to labels $\{\mathcal{L}_i\}_{i \in V}$
- i is **bad** if $\mathcal{L}_i \neq H(\mathcal{L}_{parents(i)})$



- Case 1: $\geq e$ bad nodes $\Rightarrow$ will fail opening phase whp.
- Case 2: Less than e bad labels $\Rightarrow$ $\exists$ path of good nodes (by (e, d) depth-robustness) $\Rightarrow$ $\tilde{P}$ made d sequential queries (by sequantality of RO)

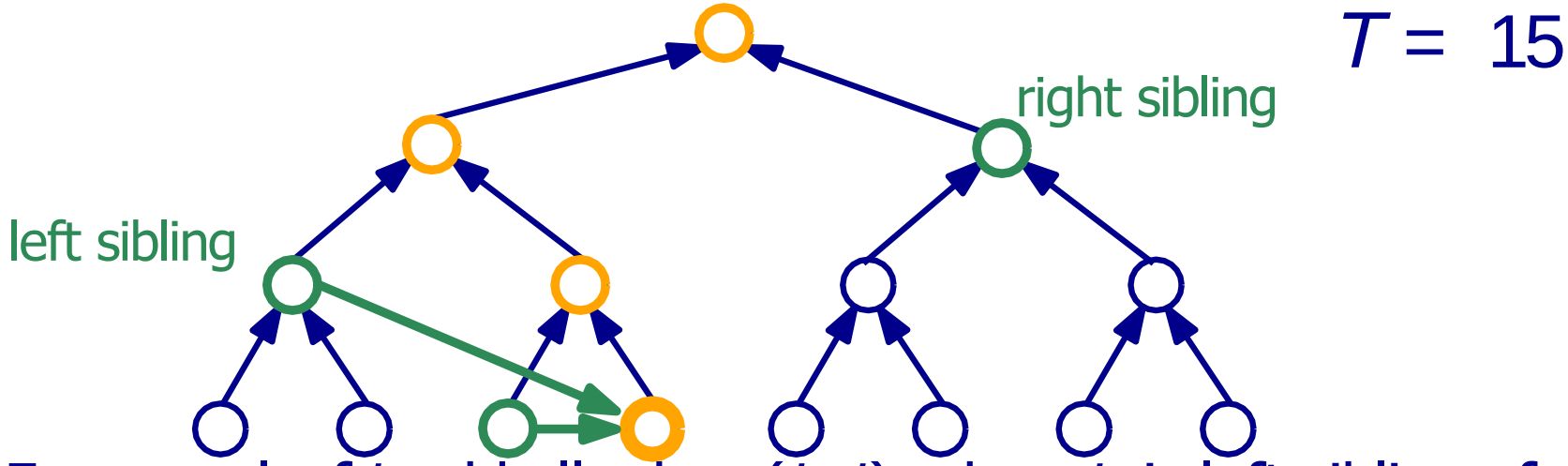
The New Construction



$T = 15$

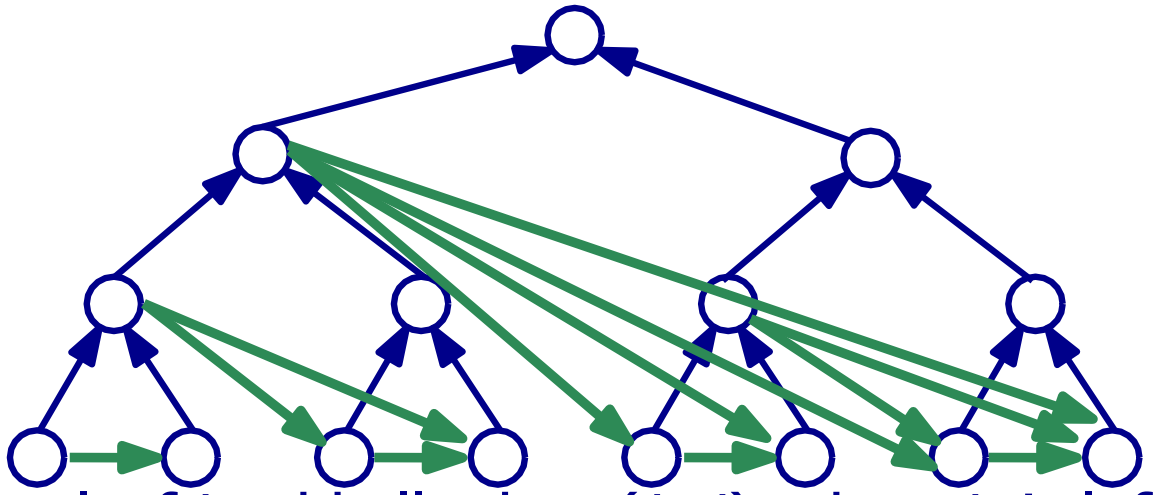
For every leaf i add all edges (j, i) where j is left sibling of node on path $i \rightarrow root$

The New Construction



For every leaf i add all edges (j, i) where j is left sibling of node on path $i \rightarrow root$

The New Construction

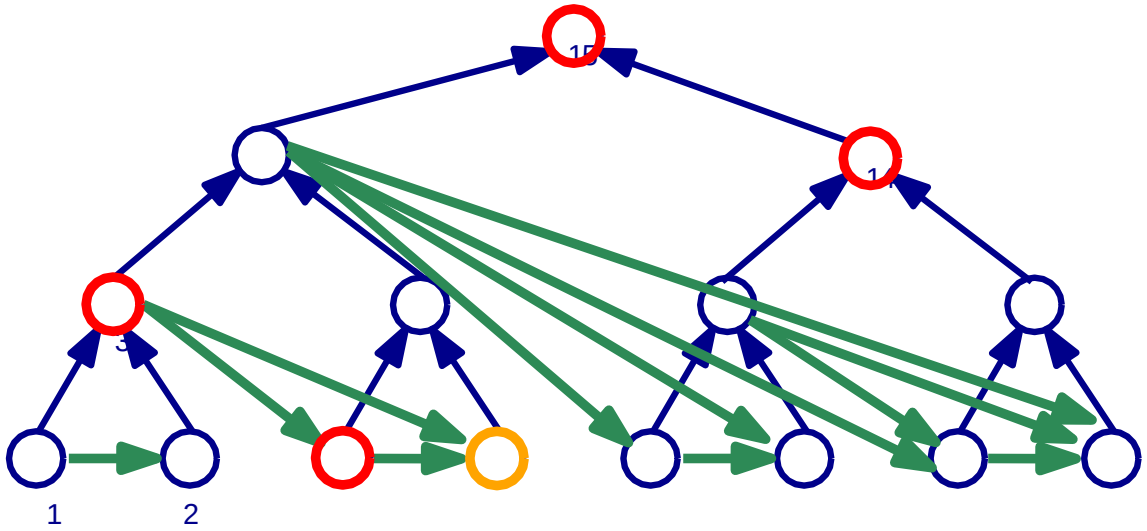


$T = 15$

For every leaf i add all edges (j, i) where j is left sibling of node on path $i \rightarrow root$

The New Construction

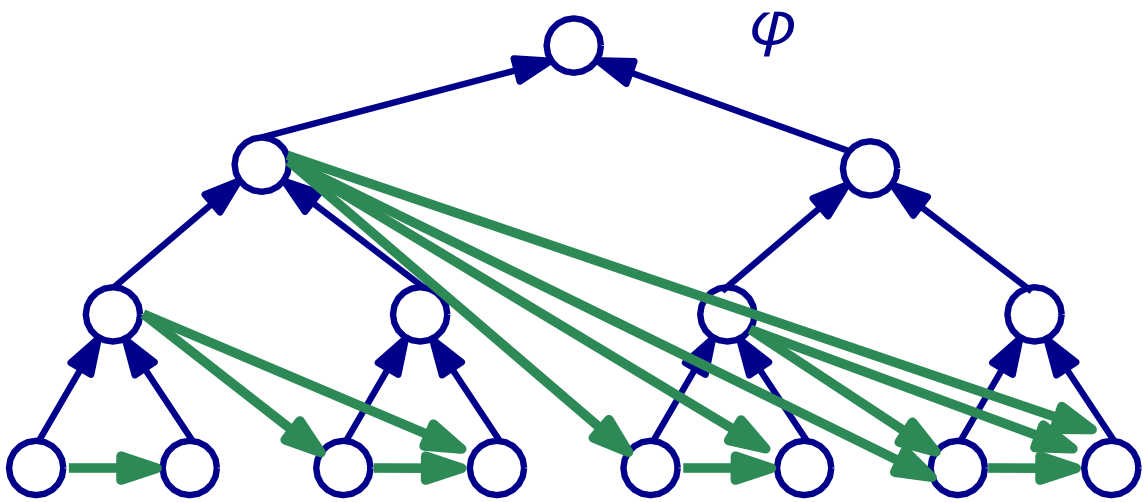
$T = 15$



For every leaf i add all edges (j, i) where j is left sibling of node on path $i \rightarrow root$

- P computes labelling $\pounds_i = \mathrm{H}(\pounds_{parents(i)})$ and sends root label $\varphi = \pounds_T$ to V. Can be done storing only $\log(T)$ labels.
- V challenges P to open a subset of leaves and checks consistency (blue and green edges!)

The New Construction

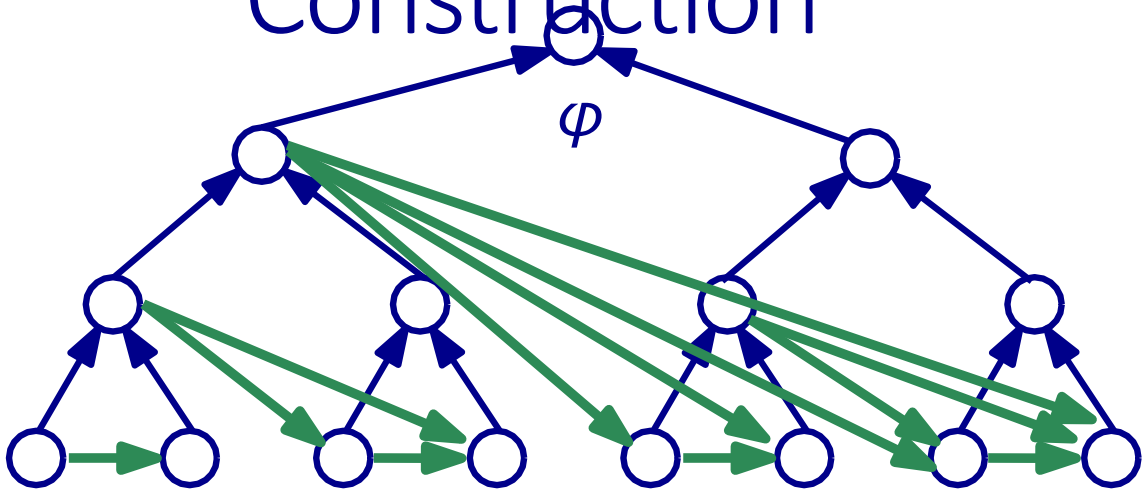


$T = 15$

Proof Sketch

The New Construction

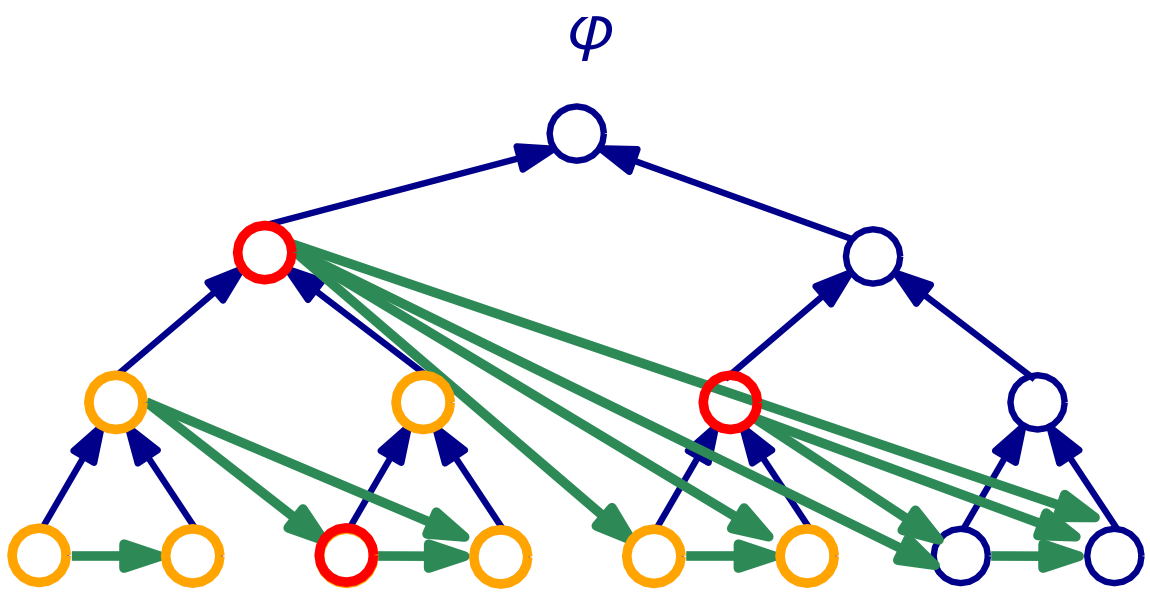
$T = 15$



Proof Sketch

- $\tilde{P}$ committed to labels $\pounds_i^{\mathrm{I}}$ after sending $\varphi = \pounds_{15}$.
- i is bad if $\pounds_i^{\mathrm{I}} \neq \mathrm{H}(\pounds^{\mathrm{I}}_{parents(i)})$.

The New Construction

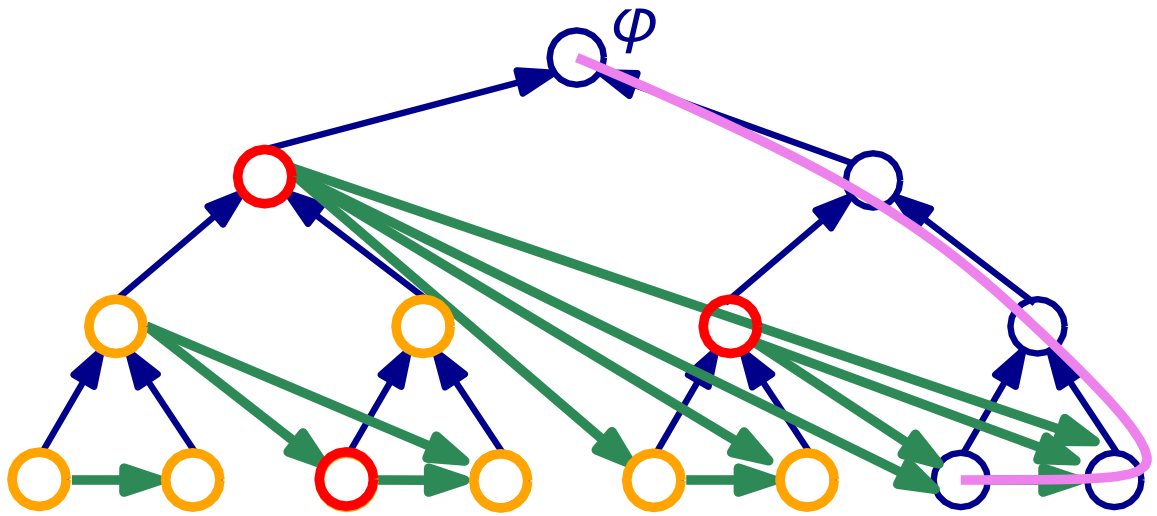


$T = 15$

Proof Sketch

- $\tilde{P}$ committed to labels ℓ_i after sending $\varphi = \ell_{15}$.
- i is bad if $\ell_i \neq \mathrm{H}(\ell_{parents(i)})$.
- Let $S \subset V$ denote the bad nodes and all nodes below.

The New Construction



$T = 15$

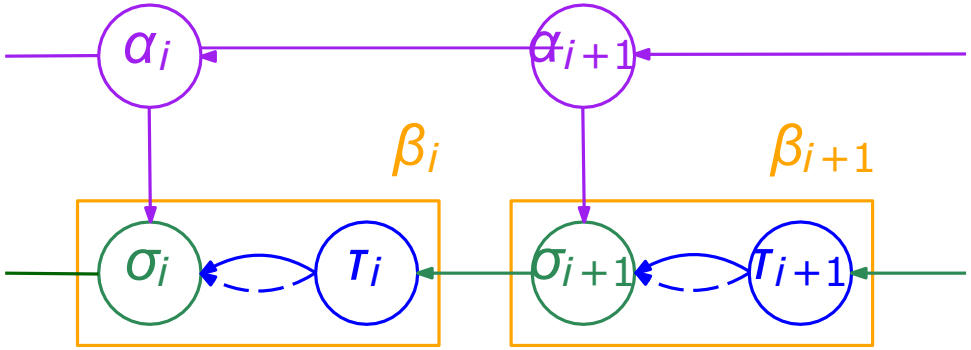
Proof Sketch

- $\tilde{P}$ committed to labels ℓ_i after sending $\varphi = \ell_{15}$.
- i is bad if $\ell_i \not\models \mathrm{H}(\ell_{parents(i)})$.
- Let $S \subset V$ denote the bad nodes and all nodes below.
- Claim 1: $\exists$ path going through $V - S$ (of length $T - |S|$).
- Claim 2: $\tilde{P}$ can't open $|S|/T$ fraction of leafs.

Theorem: $\tilde{P}$ made only $T(1 - c)$ sequential queries
$\Rightarrow$ will pass opening phase with prob. $\leq (1 - c)^{\#\text{of challenges}}$

why we care

Sustainable Blockchains



Mining Bitcoin (Proofs of Work)

Mining Bitcoin (Proofs of Work)

Can we have a more "sustainable" Blockchain?

Thanks for Listening