Blockchains from Proofs of Space and Time: from Spacemint to Chia

Krzysztof Pietrzak

Guest Lecture, Blockchains and Cryptocurrencies (Spring 2018)

Outline

- Bitcoin and Proofs of Work
- Proofs of Stake
- Proofs of Space
- Proofs of Sequential Work
- Putting it all together (Chia)

Mining Bitcoin (Proofs of Work)

Mining Bitcoin (Proofs of Work)

Ecological: Massive energy & hardware waste.

Economical: Requires high rewards $\Rightarrow$ inflation and/or high transaction fees.

Can we have a more “sustainable” Blockchain?

Alternative Proof Systems: Proof of Stake

PoW based blockchain (Bitcoin): Probability a miner can add a block proportional to its hashing power.

Proof of Stake: Probability proportional to the fraction of coins the miner owns.

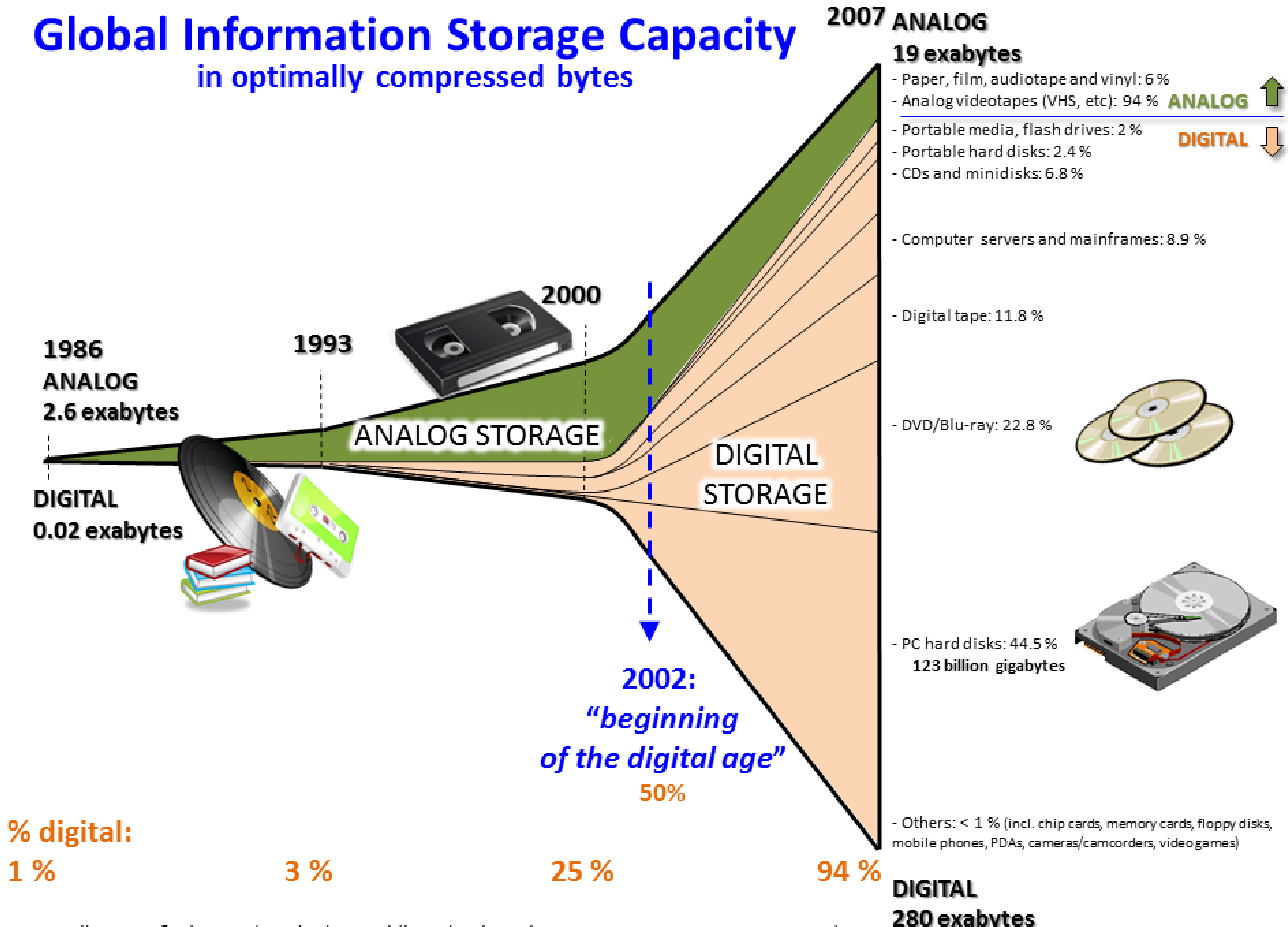
Alternative Proof Systems: Proof of Stake

PoW based blockchain (Bitcoin): Probability a miner can add a block proportional to its hashing power.

Proof of Stake: Probability proportional to the fraction of coins the miner owns.

Nxt, Algorand, Snow White, Ouroboros,...

Global Information Storage Capacity
in optimally compressed bytes



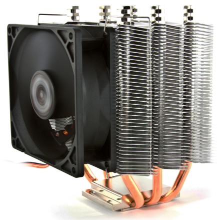
Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

First Ingredient

Proofs of Space

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015



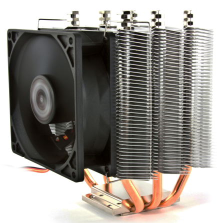
Parameter N

$\mathcal{P}$

$\mathcal{V}$

Proofs of Space
Dziembowski-Faust-Kolmogorov-Pietrzak 2015

N



Parameter N

init

$\mathcal{P}$

$\mathcal{V}$

communication $\tilde{O}(1)$

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015

Parameter N

N

$\mathcal{P}$

$\mathcal{V}$

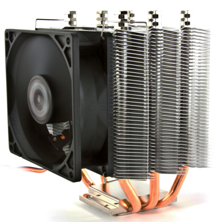
communication $\tilde{O}(1)$

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015

Parameter N

N



$\tilde{O}(1)$

$\mathcal{P}$

challenge $\longleftarrow$

response $\longrightarrow$

communication $\tilde{O}(1)$

$\mathcal{V}$

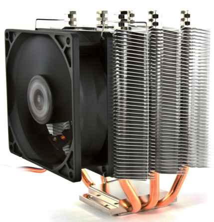
$\tilde{O}(1)$

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015

Parameter N

N



$\tilde{O}(1)$

$\mathcal{P}$

challenge

response

$\mathcal{V}$

communication $\tilde{O}(1)$

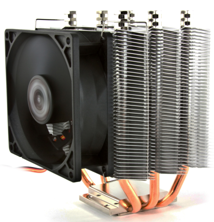
$\tilde{O}(1)$

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015

Parameter N

S



T

$\mathcal{P}$

challenge

response

$\mathcal{V}$

communication $\tilde{O}(1)$

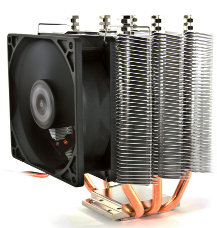
$\tilde{O}(1)$

Proofs of Space

Dziembowski-Faust-Kolmogorov-Pietrzak 2015

Parameter N

S



T

$\mathcal{P}$

challenge

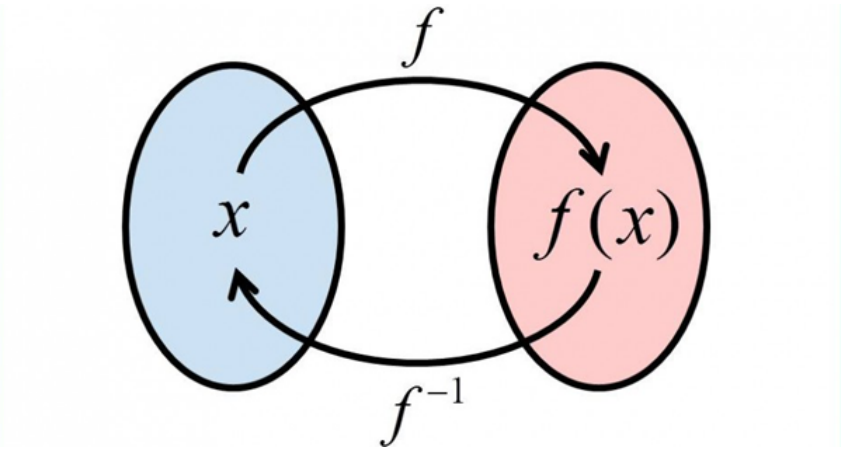
response

$\mathcal{V}$

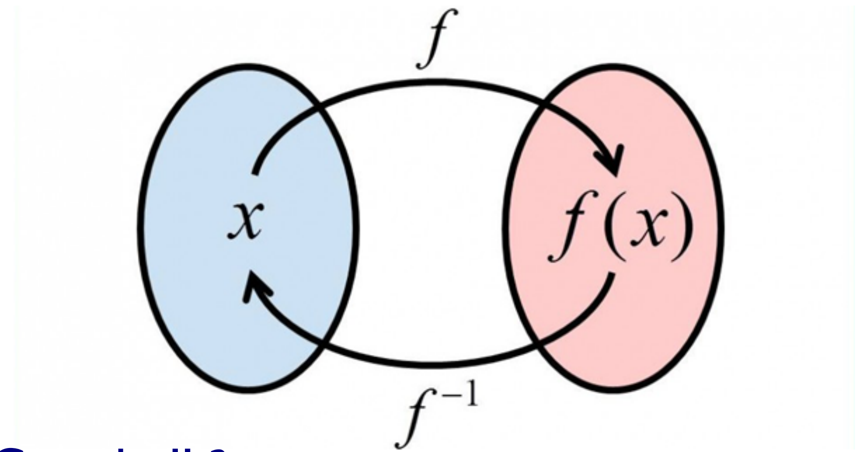
communication $\tilde{O}(1)$ $\tilde{O}(1)$

Security: either $S \approx N$ space before exec or $T \approx N$ time in exec

Two Types of Proofs of Space



Two Types of Proofs of Space

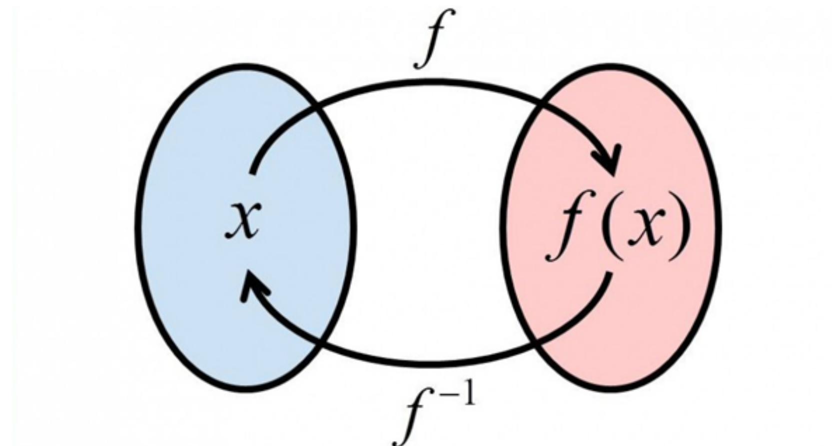


Constructions from "Hard to Pebble Graphs"[a]

- Optimal bounds: either $\Theta(N)$ space or $\Theta(N)$ time
- Non-Interactive Initialization Phase, Complicated

[a] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak: Proofs of Space. CRYPTO 2015

Two Types of Proofs of Space



Constructions from "Hard to Pebble Graphs"[a]

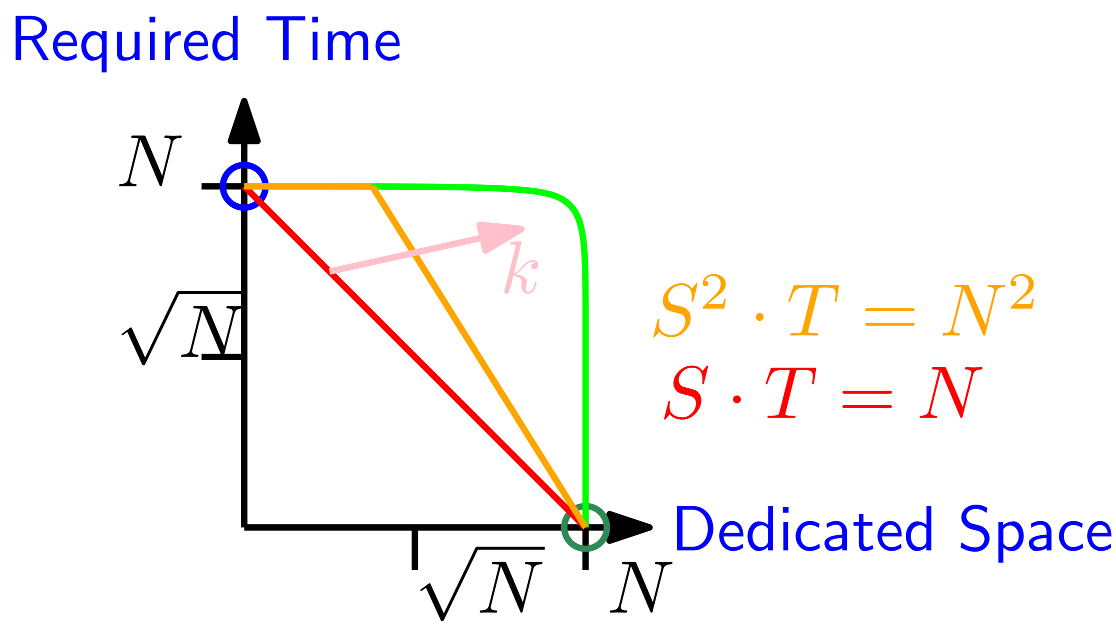
- Optimal bounds: either $\Theta(N)$ space or $\Theta(N)$ time
- Non-Interactive Initialization Phase, Complicated

[a] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak: Proofs of Space. CRYPTO 2015

Inverting Random Functions[a]

- Bounds (only) asymptotically optimal: $T \cdot S^k \geq N^k$ for "small" k, e.g. $S = T = N^{k/(1+k)}$ (proof size exponential in k)
- Non-Interactive Initialization Phase, Simple!

[a] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, L. Reyzin: Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space. ASIACRYPT 2017



Constructions from "Hard to Pebble Graphs"[a]

- Optimal bounds: either $\Theta(N)$ space or $\Theta(N)$ time
- Non-Interactive Initialization Phase, Complicated

[a] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak: Proofs of Space. CRYPTO 2015

Inverting Random Functions[a]

- Bounds (only) asymptotically optimal: $T \cdot S^k \geq N^k$ for "small" k, e.g. $S = T = N^{k/(1+k)}$ (proof size exponential in k)
- Non-Interactive Initialization Phase, Simple!

[a] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, L. Reyzin: Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space. ASIACRYPT 2017

Spacemint***:**
A Cryptocurrency Based on Proofs of Space

Sunoo Park*, Krzysztof Pietrzak†, Albert Kwon*, Joël Alwen†, Georg Fuchsbauer†, and Peter Gaži†

* MIT
† IST Austria

Constructions from "Hard to Pebble Graphs" [a]

- Optimal bounds: either $\Theta(N)$ space or $\Theta(N)$ time
- Non-Interactive Initialization Phase, Complicated

[a] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak: Proofs of Space. CRYPTO 2015

Inverting Random Functions[a]

- Bounds (only) asymptotically optimal: $T \cdot S^k \geq N^k$ for "small" k, e.g. $S = T = N^{k/(1+k)}$ (proof size exponential in k)
- Non-Interactive Initialization Phase, Simple!

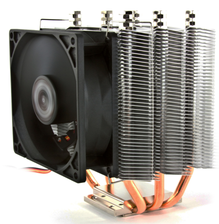
[a] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, L. Reyzin: Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space. ASIACRYPT 2017

Proofs of Space From Hard to Invert Functions

Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space

Hamza Abusalah[1], Joël Alwen[1], Bram Cohen[2], Danylo Khilko[3], Krzysztof Pietrzak[1], and Leonid Reyzin[4]

Towards a Simple Construction



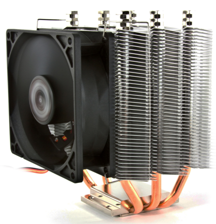
$\mathcal{P}$ $\xleftarrow{\;L\;}$ $\mathcal{V}$

Random Table L

$1, \quad y_1$
$2, \quad y_2$
$\dots$
$\dots$
$N, \quad y_N$

Towards a Simple Construction

N



$$\mathcal{P} \xleftarrow{\;L\;} \mathcal{V}$$

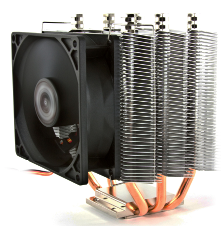
Random Table L

$1,$	y_1
$2,$	y_2
$\dots$	
$\dots$	
$N,$	y_N

$1,$	y_1
$2,$	y_2
$\dots$	
$\dots$	
$N,$	y_N

Towards a Simple Construction

N



$\mathcal{P}$

$\mathcal{V}$

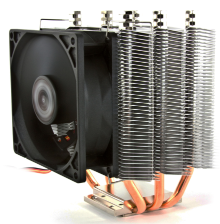
$1, \quad y_1$
$2, \quad y_2$
$\cdots$ $\cdots$
$N, \quad y_N$

Random Table L

$1, \quad y_1$
$2, \quad y_2$
$\cdots$ $\cdots$
$N, \quad y_N$

Towards a Simple Construction

N



$\mathcal{P}$

$i \leftarrow [N]$

$\mathcal{V}$

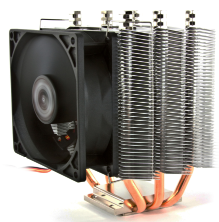
Random Table L

$1, \quad y_1$
$2, \quad y_2$
$\dots$
$\dots$
$N, \quad y_N$

$1, \quad y_1$
$2, \quad y_2$
$\dots$
$\dots$
$N, \quad y_N$

Towards a Simple Construction

N



lookup

$1, \quad y_1$
$2, \quad y_2$
$\dots$
$\dots$
$N, \quad y_N$

$\mathcal{P}$

$i \leftarrow [N]$

y_i

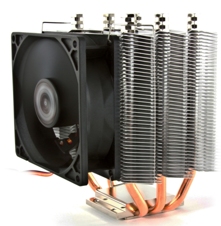
$\mathcal{V}$

Random Table L

$1, \quad y_1$
$2, \quad y_2$
$\dots$
$\dots$
$N, \quad y_N$

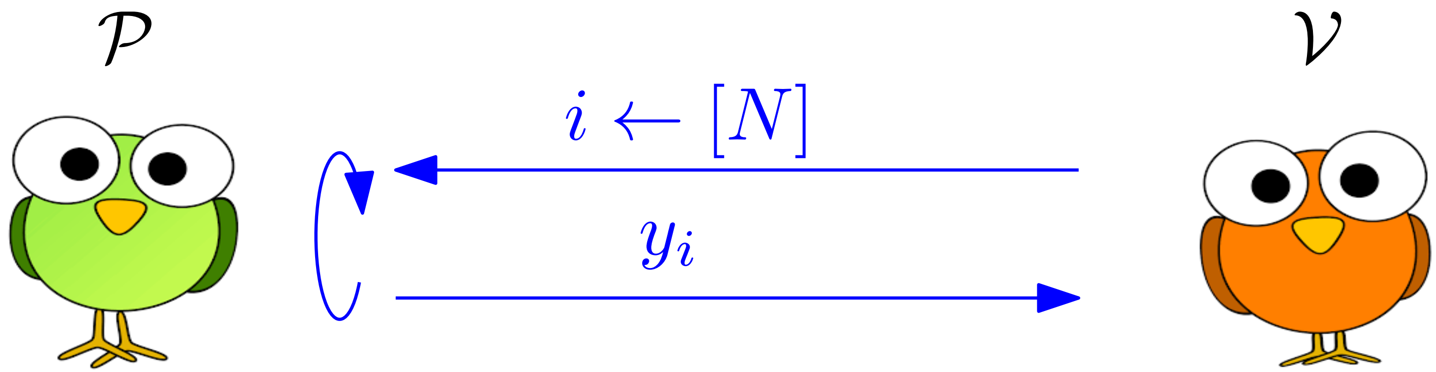
Towards a Simple Construction

N



lookup

$1,$	y_1
$2,$	y_2
$\ldots$	
$\ldots$	
$N,$	y_N

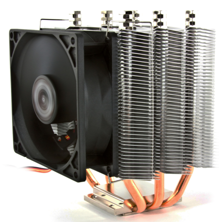


Random Table L

$1,$	y_1
$2,$	y_2
$\ldots$	
$\ldots$	
$N,$	y_N

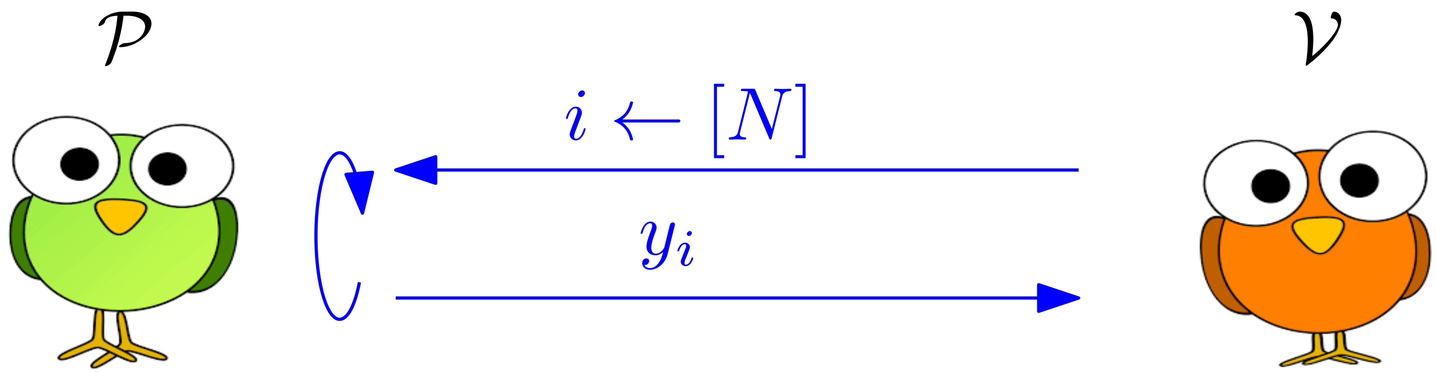
Towards a Simple Construction

N



lookup

$1, \quad y_1$
$2, \quad y_2$
$\ldots$
$\ldots$
$N, \quad y_N$



Random Table L

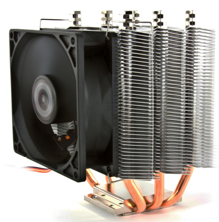
$1, \quad y_1$
$2, \quad y_2$
$\ldots$
$\ldots$
$N, \quad y_N$

Problems:

1. large communication
2. space-inefficient $\mathcal{V}$

Towards a Simple Construction

N



$\mathcal{P}$

$\xleftarrow{\pi}$

$\mathcal{V}$

randomly pick

$\pi : [N] \to [N]$

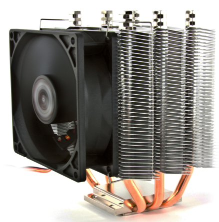
$1, \pi(1)$
$2, \pi(2)$
$\dots$
$\dots$
$N, \pi(N)$

Towards a Simple Construction

N

$\mathcal{P}$

$\mathcal{V}$



randomly pick

$\pi : [N] \to [N]$

$1, \pi(1)$
$2, \pi(2)$
$\cdots$
$\cdots$
$N, \pi(N)$

Towards a Simple Construction

N

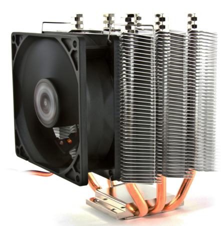
$\mathcal{P}$

$y \leftarrow [N]$

$\mathcal{V}$

randomly pick

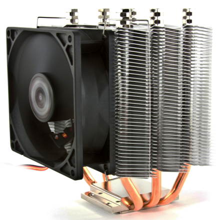
$\pi : [N] \to [N]$



$1,\pi(1)$
$2,\pi(2)$
$\cdots$
$\cdots$
$N,\pi(N)$

Towards a Simple Construction

N



lookup

$1,\pi(1)$
$2,\pi(2)$
$\dots$ $\dots$
$N,\pi(N)$

$\mathcal{P}$

$y \leftarrow [N]$

$\pi^{-1}(y)$

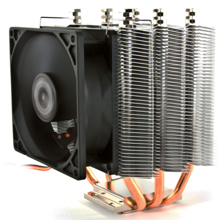
$\mathcal{V}$

randomly pick

$\pi : [N] \to [N]$

Towards a Simple Construction

N



lookup

$1, \pi(1)$
$2, \pi(2)$
$\dots$ $\dots$
$N, \pi(N)$

$\mathcal{P}$

$y \leftarrow [N]$

$\pi^{-1}(y)$

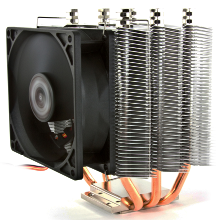
$\mathcal{V}$

randomly pick

$\pi : [N] \to [N]$

Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

$\pi^{-1}(y)$

$\mathcal{V}$

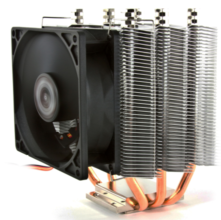
randomly pick

Problem: Hellman 1980

$\pi : [N] \rightarrow [N]$

Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

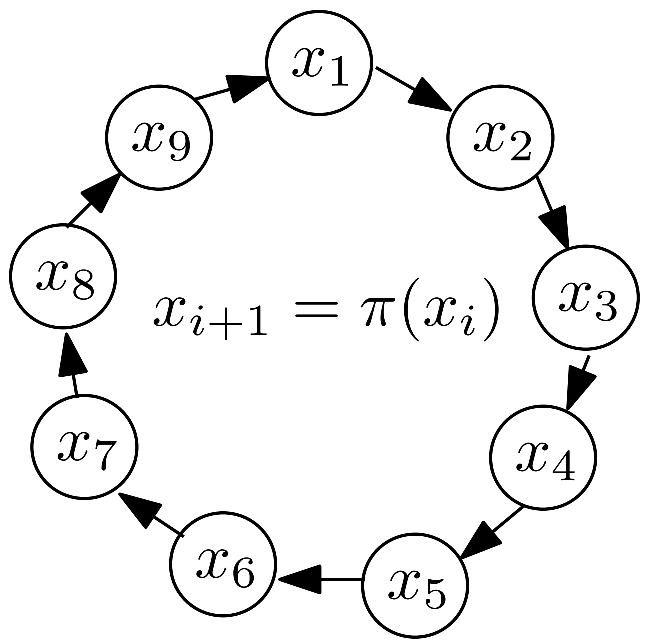
$\pi^{-1}(y)$

$\mathcal{V}$

randomly pick

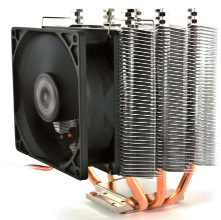
$\pi : [N] \to [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

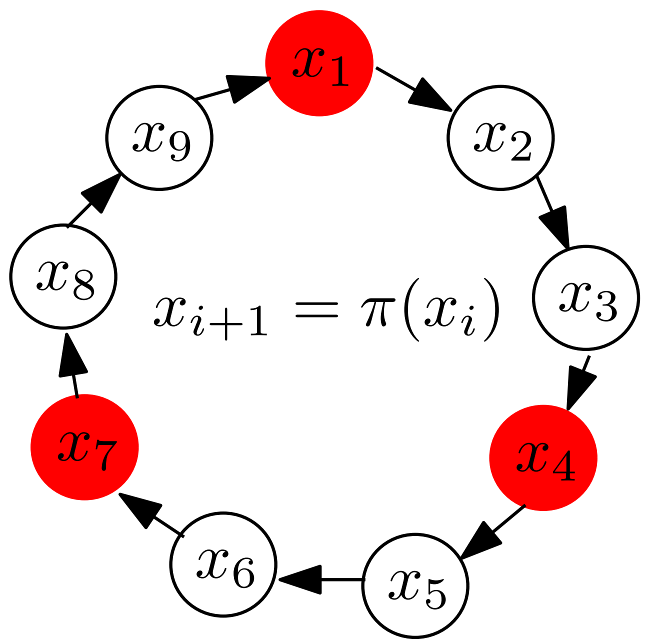
$\pi^{-1}(y)$

$\mathcal{V}$

randomly pick

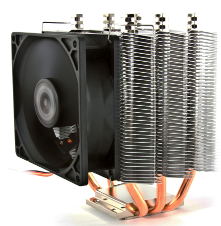
$\pi : [N] \to [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\color{red}{\sqrt{N}}$



$\color{blue}{\sqrt{N}}$

$\mathcal{P}$

$\mathcal{V}$

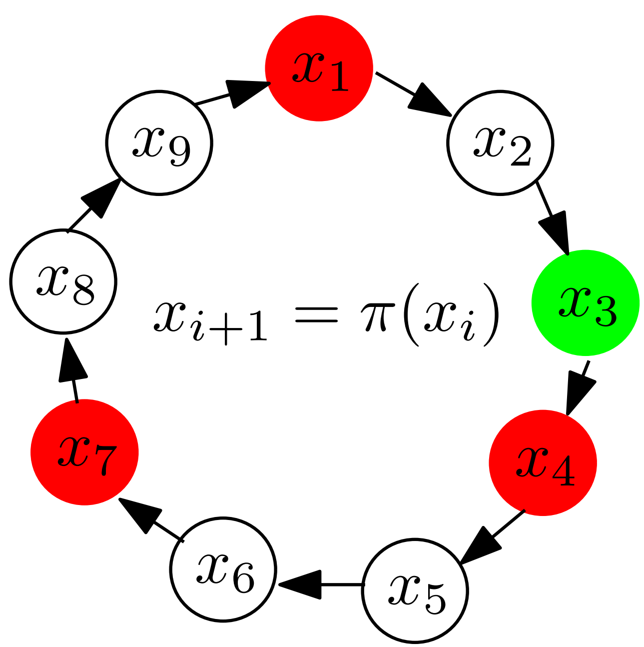
$y \leftarrow [N]$

$\pi^{-1}(y)$

randomly pick

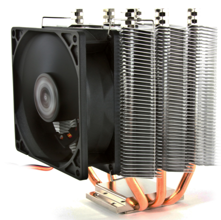
$\pi : [N] \to [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

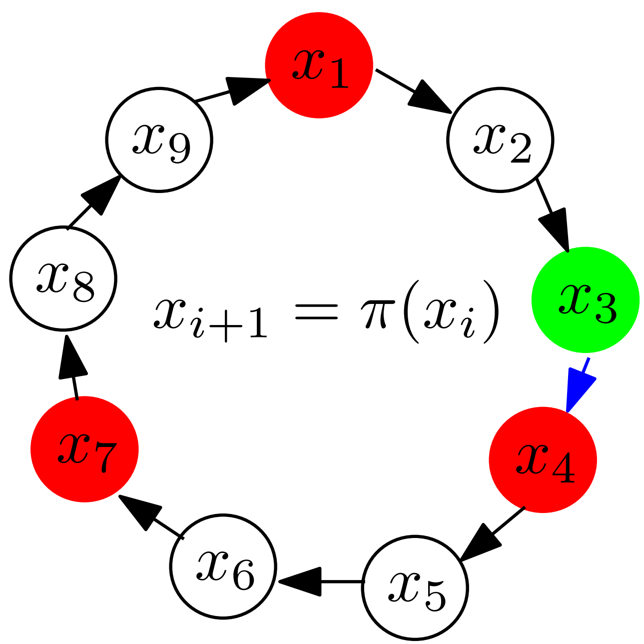
$\pi^{-1}(y)$

$\mathcal{V}$

randomly pick

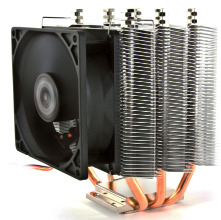
$\pi : [N] \to [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

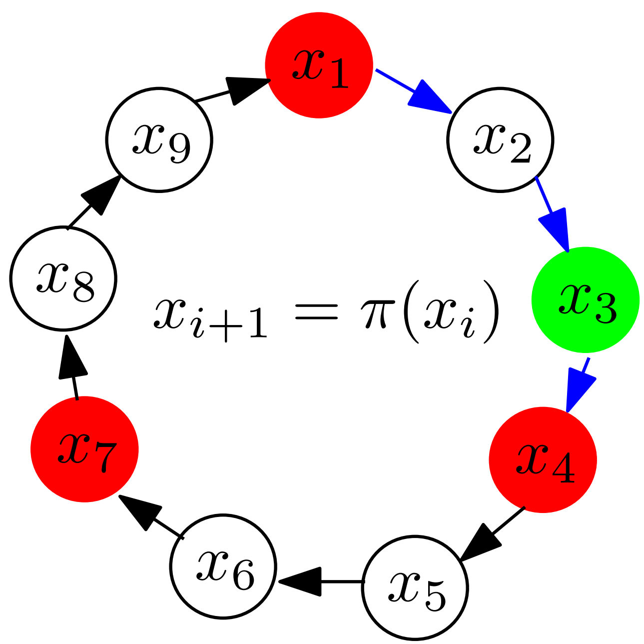
$\pi^{-1}(y)$

$\mathcal{V}$

randomly pick

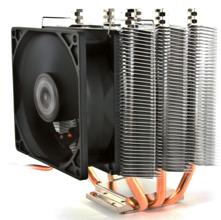
$\pi : [N] \to [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$\mathcal{V}$

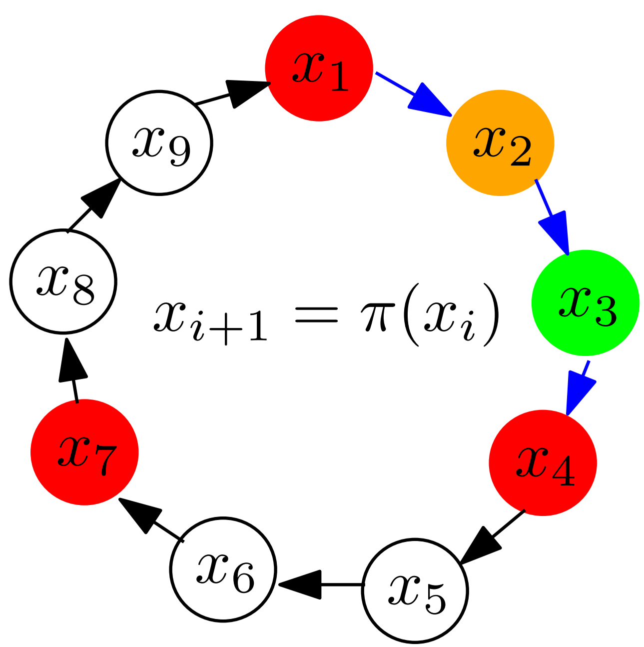
$y \leftarrow [N]$

$\pi^{-1}(y)$

randomly pick

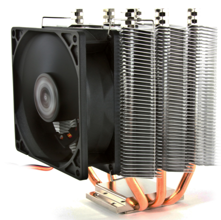
$\pi : [N] \rightarrow [N]$

Problem: Hellman 1980



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

$\pi^{-1}(y)$

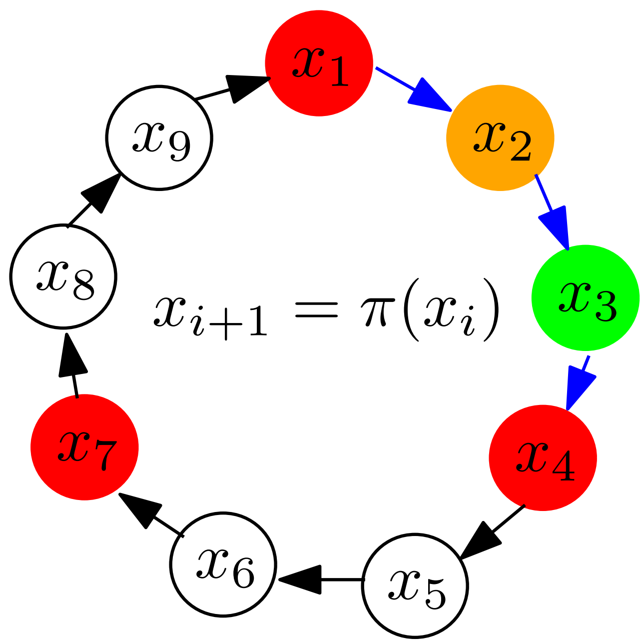
$\mathcal{V}$

randomly pick

$\pi : [N] \to [N]$

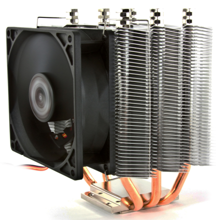
Problem: Hellman 1980

π on $[N] : ST \leq N$



Towards a Simple Construction

$\sqrt{N}$



$\sqrt{N}$

$\mathcal{P}$

$y \leftarrow [N]$

$\pi^{-1}(y)$

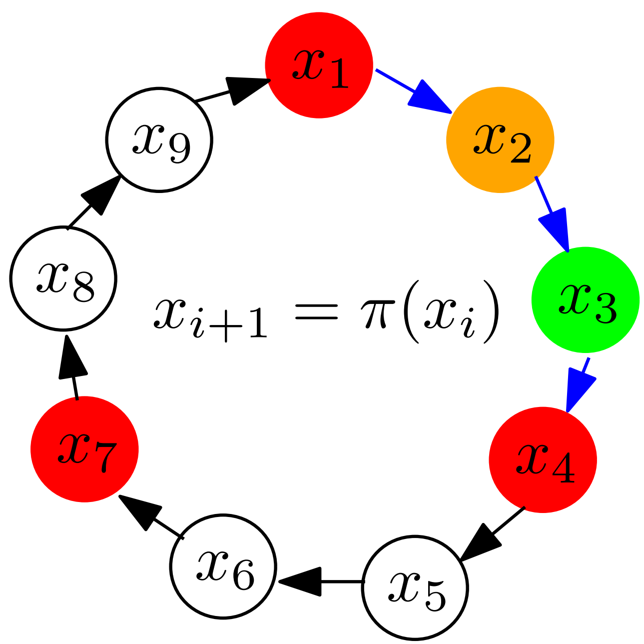
$\mathcal{V}$

randomly pick

$\pi : [N] \to [N]$

Problem: Hellman 1980

π on $[N]$: $ST \leq N$



Yao 1990: π random: $ST \geq N$

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

	lower bound	upper bound	
permutation	$ST \in \tilde{\Omega}(\epsilon N)^*$	$ST \in \tilde{O}(\epsilon N)^*$	$S = T \approx N^{1/2}$
random functions	$ST \in \tilde{\Omega}(\epsilon N)^*$		
general functions			
our functions			

*: Yao-1990, Gennaro-Trevisan'00, Wee05, De-Trevisan-Tulsiani-2010

*: Hellman1980

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

	lower bound	upper bound	
permutation	$ST \in \tilde{\Omega}(\epsilon N)^*$	$ST \in \tilde{O}(\epsilon N)^*$	$S = T \approx N^{1/2}$
random functions	$ST \in \tilde{\Omega}(\epsilon N)^*$	$S^2T \in \tilde{O}(\epsilon^2 N^2)^*$	$S = T \approx N^{2/3}$
general functions			
our functions			

*: Yao-1990, Gennaro-Trevisan'00, Wee05, De-Trevisan-Tulsiani-2010

*: Hellman1980

*: Fiat-Naor1991

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

	lower bound	upper bound	
permutation	$ST \in \tilde{\Omega}(\epsilon N)^{*}$	$ST \in \tilde{O}(\epsilon N)^{*}$	$S = T \approx N^{1/2}$
random functions	$ST \in \tilde{\Omega}(\epsilon N)^{*}$	$S^2 T \in \tilde{O}(\epsilon^2 N^2)^{*}$	$S = T \approx N^{2/3}$
general functions		$S^3 T \in \tilde{O}(\epsilon^3 N^3)^{*}$	$S = T \approx N^{3/4}$
our functions			

*: Yao-1990, Gennaro-Trevisan'00, Wee05, De-Trevisan-Tulsiani-2010

*: Hellman1980

*: Fiat-Naor1991

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

	lower bound	upper bound	
permutation	$ST \in \tilde{\Omega}(\epsilon N)^*$	$ST \in \tilde{O}(\epsilon N)^*$	$S = T \approx N^{1/2}$
random functions	$ST \in \tilde{\Omega}(\epsilon N)^*$	$S^2T \in \tilde{O}(\epsilon^2 N^2)^*$	$S = T \approx N^{2/3}$
general functions		$S^3T \in \tilde{O}(\epsilon^3 N^3)^*$	$S = T \approx N^{3/4}$
our functions	$S^kT \in \tilde{\Omega}(\epsilon^k N^k)$	$S = T \approx N^{k/(k+1)}$	

*: Yao-1990, Gennaro-Trevisan'00, Wee05, De-Trevisan-Tulsiani-2010

*: Hellman1980

*: Fiat-Naor1991

Inverting Functions

$f : [N] \to [N]$

$\mathcal{A}$ inverts f on ϵ fraction with
S bits advice
T oracle queries

	lower bound	upper bound	
permutation	$ST \in \tilde{\Omega}(\epsilon N)^*$	$ST \in \tilde{O}(\epsilon N)^*$	$S = T \approx N^{1/2}$
random functions	$ST \in \tilde{\Omega}(\epsilon N)^*$	$S^2T \in \tilde{O}(\epsilon^2 N^2)^*$	$S = T \approx N^{2/3}$
general functions		$\boxed{S^3T \in \tilde{O}(\epsilon^3 N^3)^*}$	$S = T \approx N^{3/4}$
our functions	$\boxed{S^kT \in \tilde{\Omega}(\epsilon^k N^k)}$	$S = T \approx N^{k/(k+1)}$	

*: Yao-1990, Gennaro-Trevisan'00, Wee05, De-Trevisan-Tulsiani-2010

*: Hellman1980

*: Fiat-Naor1991

Two Observations

1) For Hellman's attack to work, the function should be easy to evaluate in forward direction

Two Observations

1) For Hellman's attack to work, the function should be easy to evaluate in forward direction

2) Usefulness for PoS: sufficient that the function table is computable in linear time

Our Function

$f : [N] \to [N]$ permutation

$g : [N] \times [N] \to [N]$

$\sigma : [N] \to [N]$ involution without fixed points, e.g., flip all bits

Our Function

$f : [N] \to [N]$ permutation

$g : [N] \times [N] \to [N]$

$\sigma : [N] \to [N]$ involution without fixed points, e.g., flip all bits

$$g_f : [N] \to [N]$$
$$g_f(x) = g(x, x') \quad \text{s.t.} \quad \sigma(f(x)) = f(x')$$

Our Function

$f : [N] \to [N]$ permutation

$g : [N] \times [N] \to [N]$

$\sigma : [N] \to [N]$ involution without fixed points, e.g., flip all bits

$$g_f : [N] \to [N]$$
$$g_f(x) = g(x, x') \quad \text{s.t.} \quad \sigma(f(x)) = f(x')$$

equivalently

$$g_f(x) = g(x, f^{-1}(\sigma(f(x))))$$

Our Function

$f : [N] \to [N]$ permutation

$g : [N] \times [N] \to [N]$

$\sigma : [N] \to [N]$ involution without fixed points, e.g., flip all bits

$$g_f : [N] \to [N]$$
$$g_f(x) = g(x, x') \quad \text{s.t.} \quad \sigma(f(x)) = f(x')$$

equivalently

$$g_f(x) = g(x, f^{-1}(\sigma(f(x))))$$

Theorem: If $\mathcal{A}$ has S bits of advice and makes up to T queries to f and g and succeeds in inverting g_f with ϵ probability, then

$$S^2 T \in \Omega(\epsilon^2 N^2)$$

Our Function

$f : [N] \to [N]$ permutation

$g : [N] \times [N] \to [N]$

$\sigma : [N] \to [N]$ involution without fixed points, e.g., flip all bits

$$g_f : [N] \to [N]$$
$$g_f(x) = g(x, x') \quad \text{s.t.} \quad \sigma(f(x)) = f(x')$$

equivalently

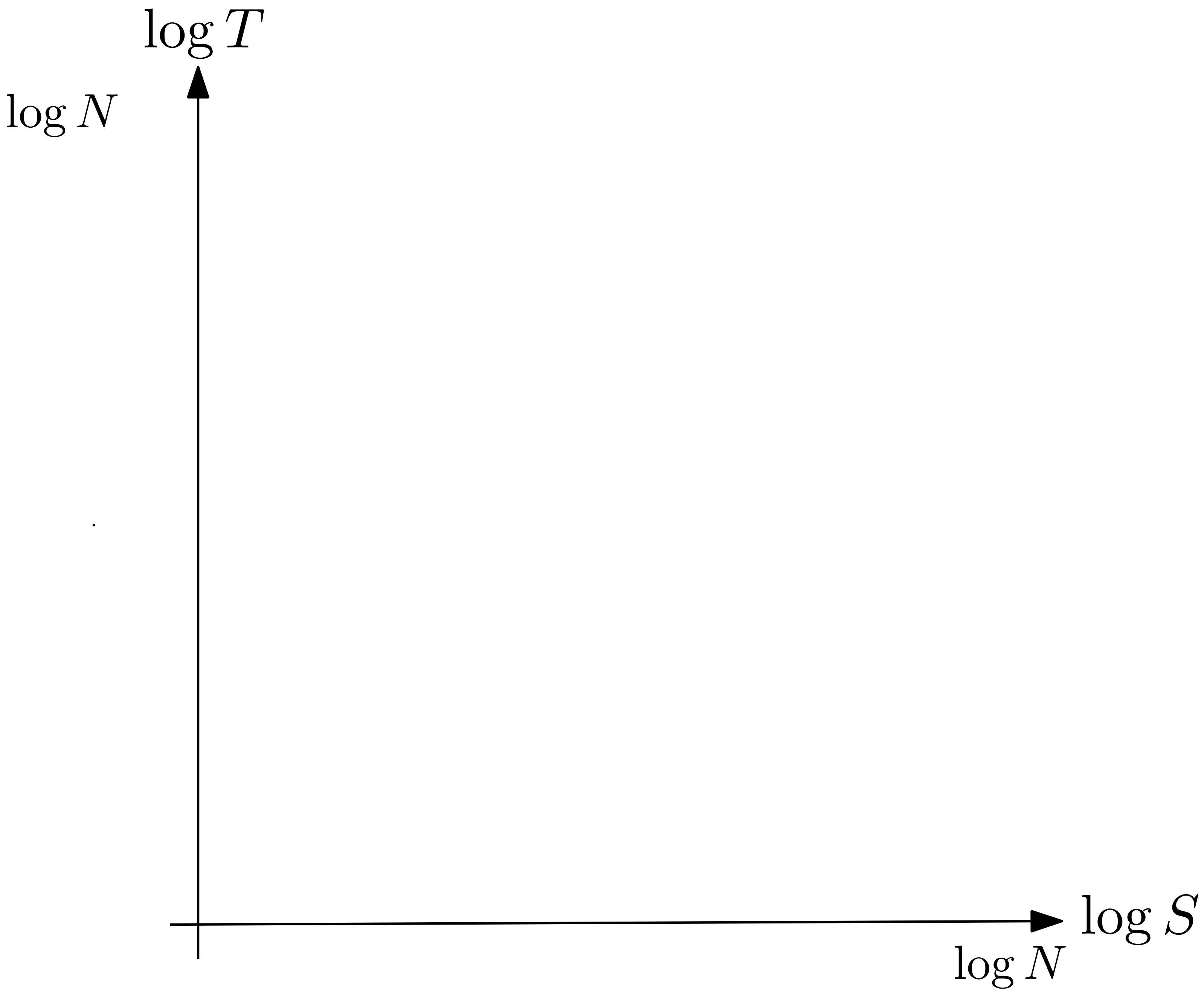
$$g_f(x) = g(x, f^{-1}(\sigma(f(x))))$$

Theorem: If $\mathcal{A}$ has S bits of advice and makes up to T queries to f and g and succeeds in inverting g_f with ϵ probability, then

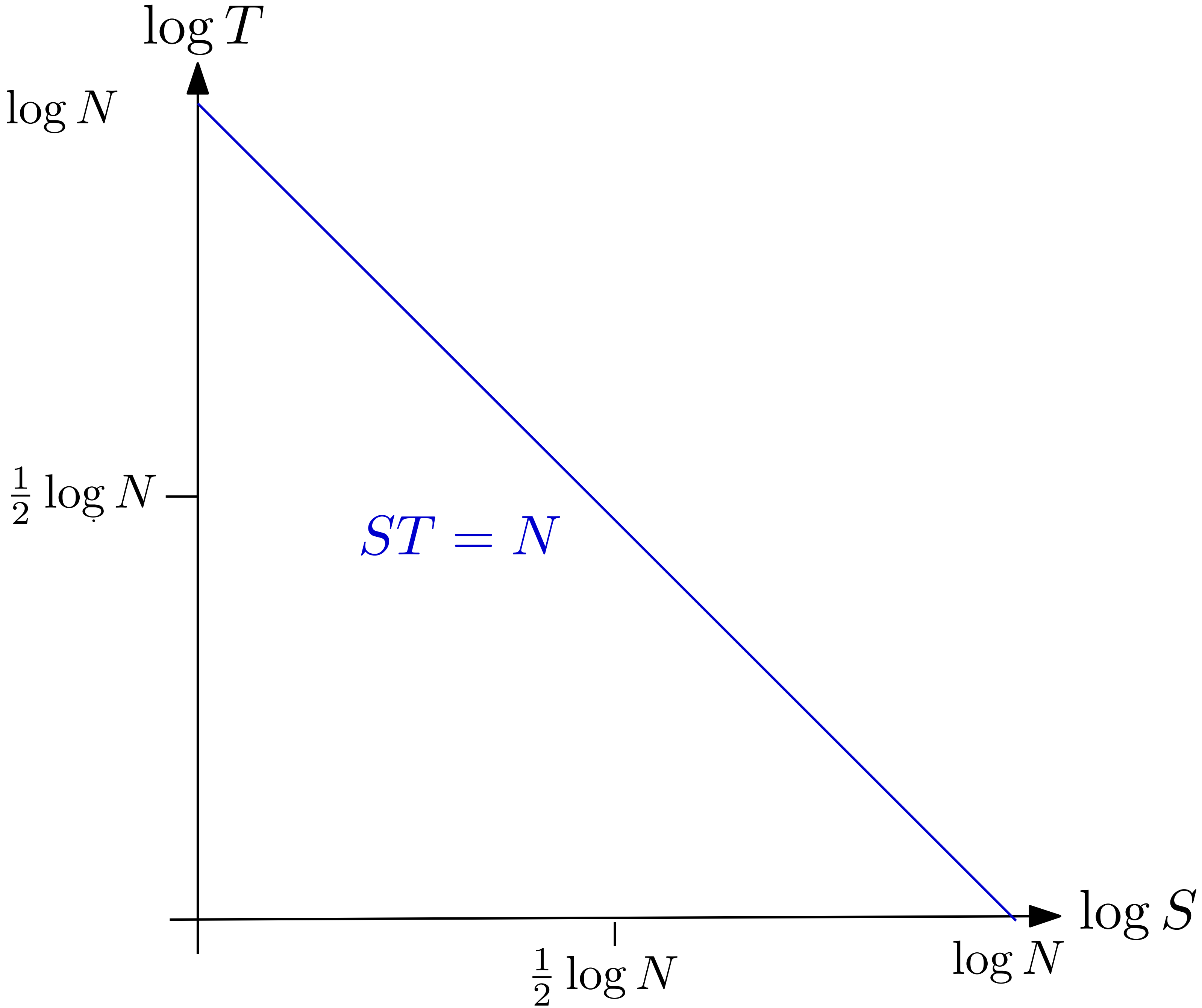
$$S^2 T \in \Omega(\epsilon^2 N^2)$$

Caveat: holds for $T \leq N^{2/3}$

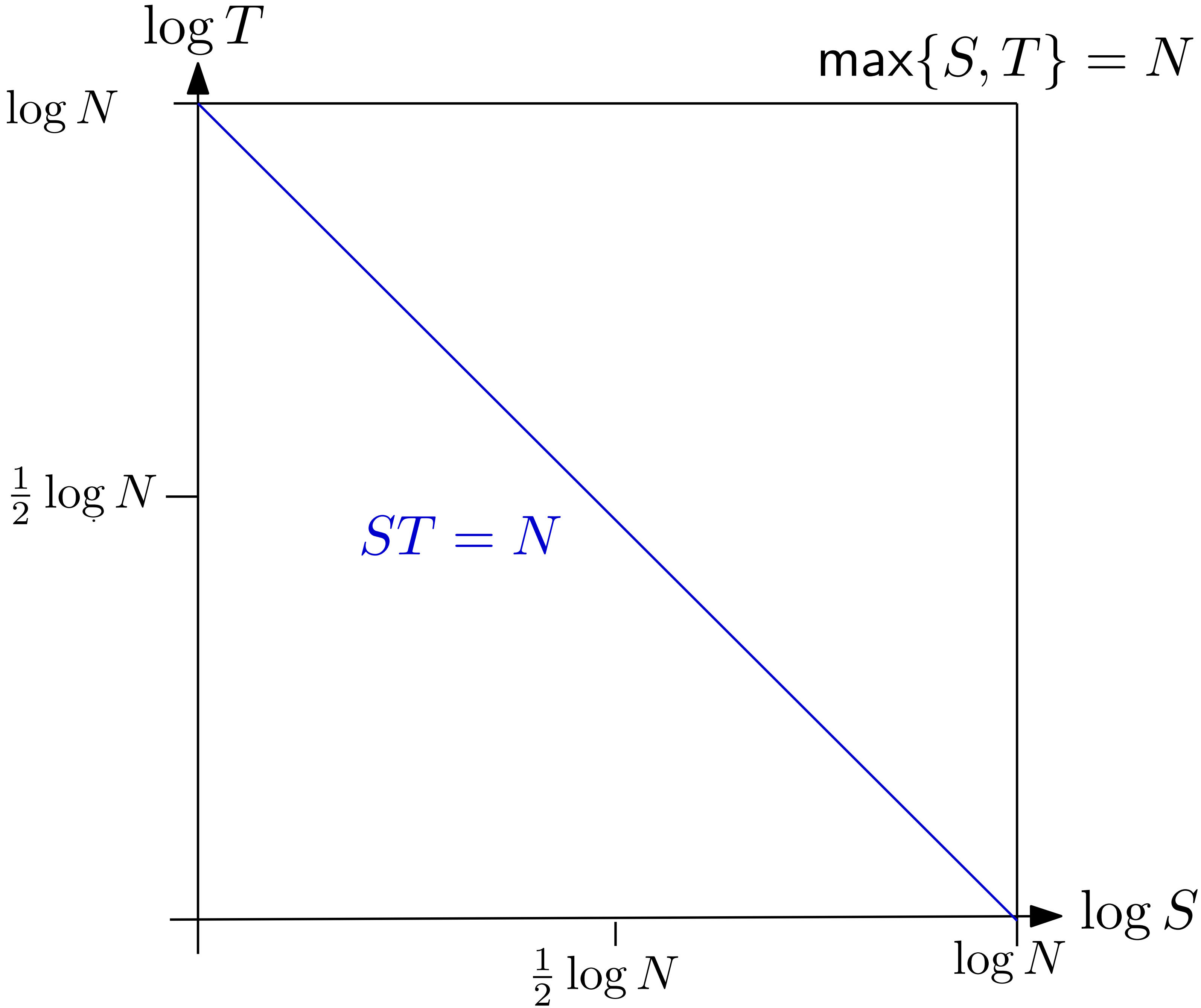
Function Inversion for PoS



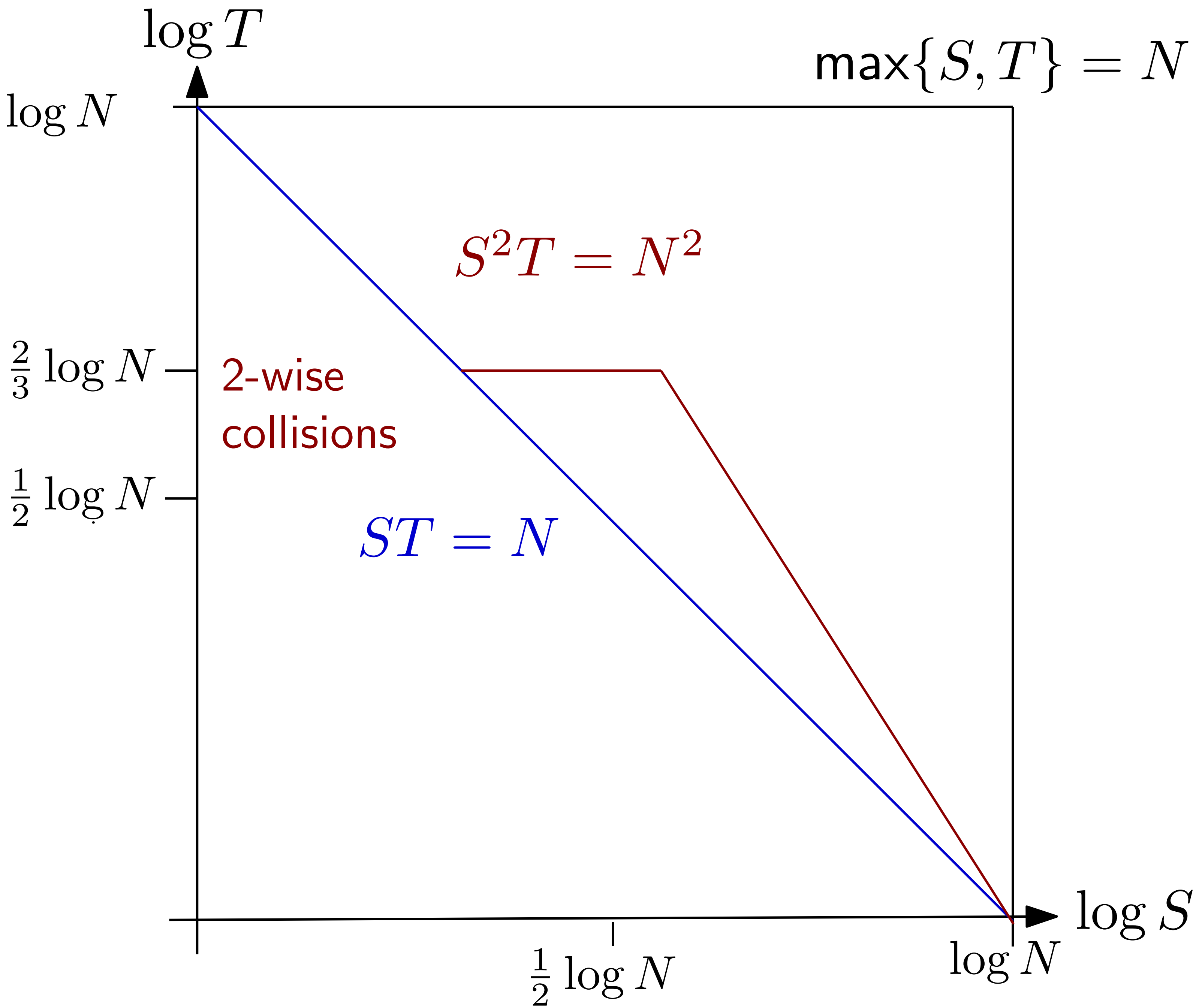
Function Inversion for PoS



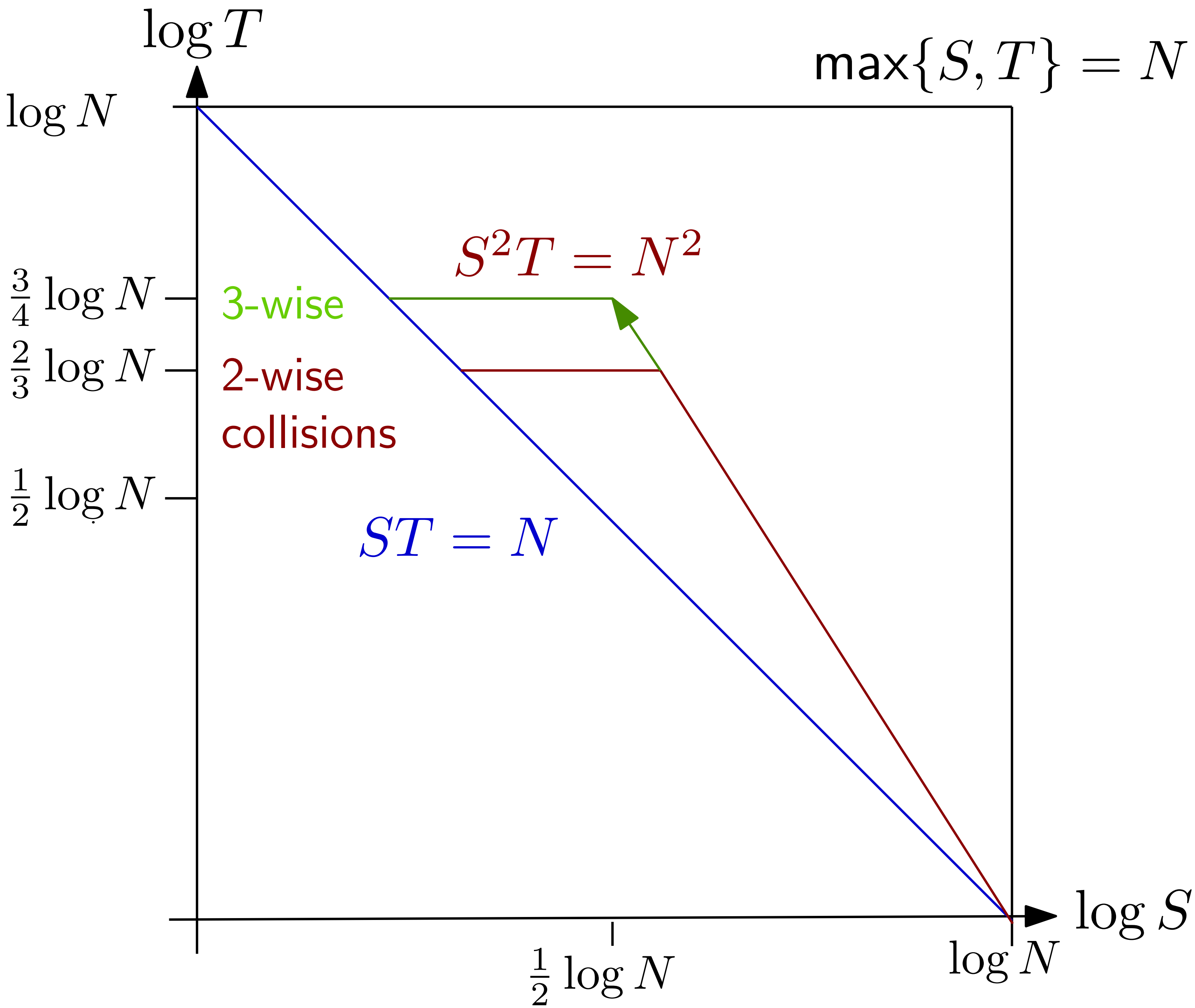
Function Inversion for PoS



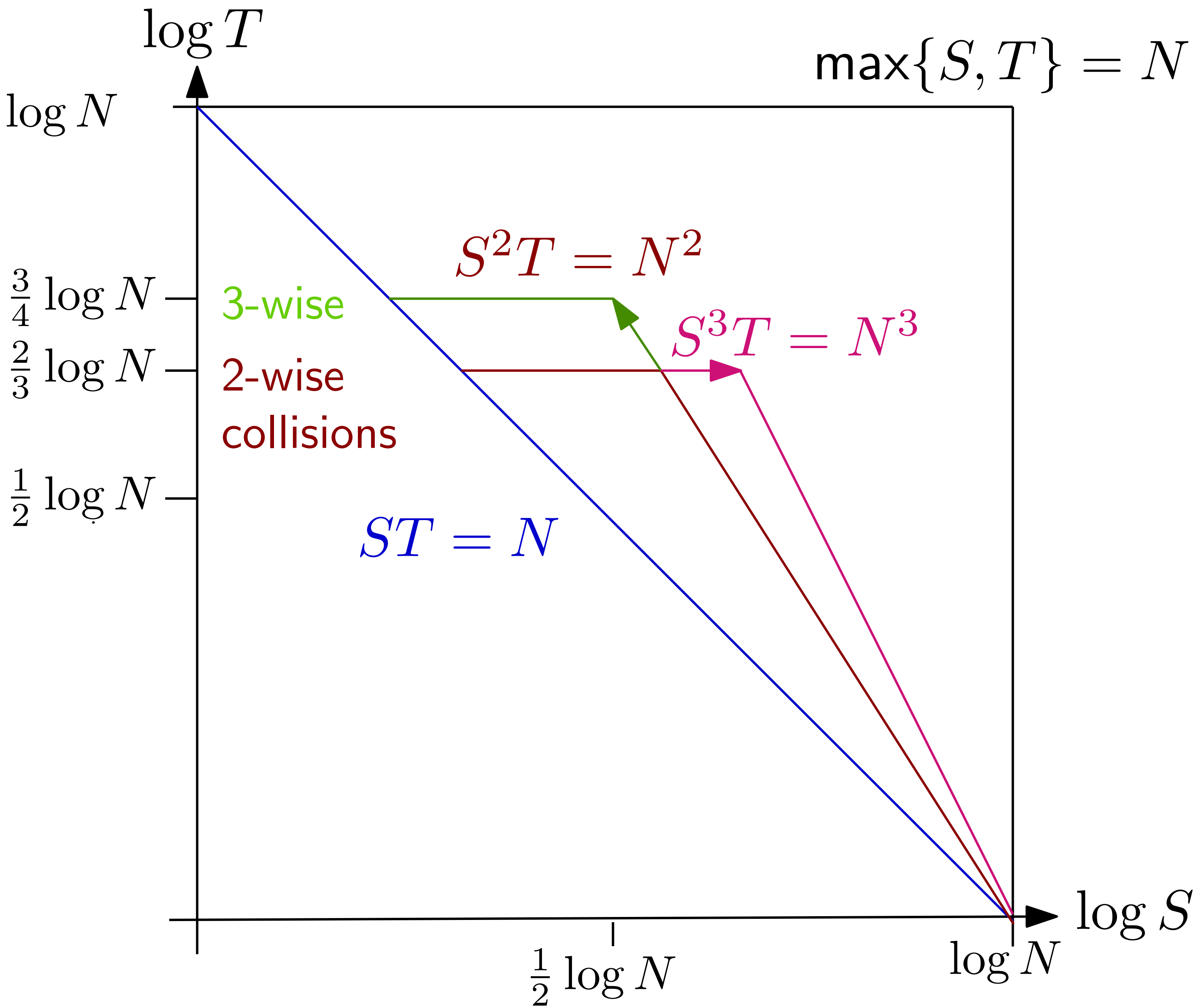
Function Inversion for PoS



Function Inversion for PoS



Function Inversion for PoS



Proof Sketch

Theorem: If $\mathcal{A}$ has S bits of advice and makes up to T queries to f and g and succeeds in inverting g_f with ϵ probability, then

$$S^2T \in \Omega(\epsilon^2 N^2)$$

Proof Sketch

Theorem: If $\mathcal{A}$ has S bits of advice and makes up to T queries to f and g and succeeds in inverting g_f with ϵ probability, then

$$S^2 T \in \Omega(\epsilon^2 N^2)$$

- Compression argument:
  - Use $\mathcal{A}$ (given S bits advice and T queries per challenge) to "compress" f, g by X bits.
  - As random f, g are incompressible $\Rightarrow$ advice S was at least X.

Proof Sketch

Theorem: If $\mathcal{A}$ has S bits of advice and makes up to T queries to f and g and succeeds in inverting g_f with ϵ probability, then

$$S^2 T \in \Omega(\epsilon^2 N^2)$$

- $\mathcal{A}(y) \to g_f^{-1}(y)$ makes $\leq T$ queries total, of which T_g to g_f.
- Case 1, $T_g \leq \sqrt{T}$: compress g using $S \cdot T_g \geq N$ [Yao90] $\Rightarrow S^2 \cdot T \geq N^2$.
- Case 2 $T_g > \sqrt{T}$:
  - can use every g_f query (x, x') to compress an f value (as $f(x) = f(x') + 1$) if $f(x)$ is "fresh".
  - During $\approx N/T$ invocations of $\mathcal{A}$ most f "fresh", as every invocation "spoils" $\leq T$ f values.
  - Can compress total of $\frac{N}{T} \cdot T_g \geq \frac{N}{\sqrt{T}}$ values

    $\Rightarrow S \geq N/\sqrt{T} \Rightarrow S^2 \cdot T \geq N^2$.

Second Ingredient

Proofs of Sequential Work

aka Verifiable Delay Algorithm

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: given $(N = p \cdot q, x \in Z_N^*, T \in \mathbb{N})$ find $x^{2^T} \mod N$

requires T sequential squarings if p, q unknown:

$$x \to x^2 \to x^{2^2} \to \ldots x^{2^T} \mod N$$

Given p, q, compute $\phi(N) := (p-1)(q-1)$

$$m := 2^T \mod \phi(N) \mod N$$

$$x^m := x^{2^T} \mod N.$$

Time-lock puzzles and timed-release Crypto

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

Revised March 10, 1996

Puzzle: given $(N = p \cdot q, x \in Z_N^*, T \in \mathbb{N})$ find $x^{2^T} \mod N$

requires T sequential squarings if p, q unknown:

$$x \to x^2 \to x^{2^2} \to \dots x^{2^T} \mod N$$

Given p, q, compute $\phi(N) := (p-1)(q-1)$

$$m := 2^T \mod \phi(N) \mod N$$

$$x^m := x^{2^T} \mod N.$$

"Dedicated verifier": if one can efficiently verify

$y \stackrel{?}{=} 2^{2^T} \mod N$ one can also efficiently compute it.

Proofs of **Sequential** Work

- Function τ : challenge $\times$ time parameter $\rightarrow$ proof
- $\tau(c, t)$ can be computed making t sequential queries to some hash function H
- There's an efficient verification algorithm that outputs 1 on input $c, t, \tau(c, t)$.
- For random c, no algorithm making (slightly less than) t parallel queries to H can produce τ that passes verification

compute $\tau(c, t)$
making t
sequential
queries to H

$\mathcal{P}$

random challenge c
time parameter t

$\tau(c, t)$

$\mathcal{V}$

(efficiently)
verify $\tau(c, t)$

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody* Tal Moran† Salil Vadhan‡

February 18, 2013

- Prover needs not just T sequential time, but also T <u>space</u> to compute proof.
- Proof not unique: given a valid proof ϕ, can generate different accepting proofs $\phi' \neq \phi \Rightarrow$ GRINDING

Publicly Verifiable Proofs of Sequential Work

Mohammad Mahmoody[*] Tal Moran[†] Salil Vadhan[‡]

February 18, 2013

- Prover needs not just T sequential time, but also T space to compute proof.
- Proof not unique: given a valid proof ϕ, can generate different accepting proofs $\phi' \neq \phi \Rightarrow$ GRINDING

Simple Proofs of Sequential Work

Bram Cohen[1] and Krzysztof Pietrzak[2*]

[1] Chia Network, bram@chia.network
[2] IST Austria, pietrzak@ist.ac.at

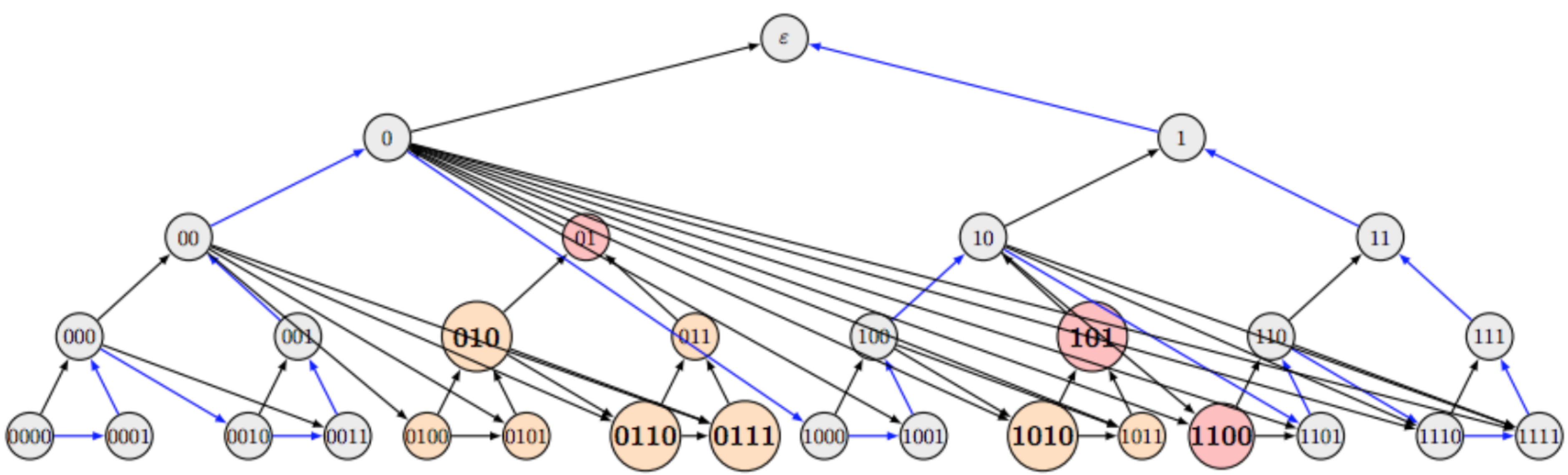
Super simple and efficient, prover just needs $\log(T)$ space, but still not unique!

Construction:

- $\mathcal{V}$ sends χ to $\mathcal{P}$, this defines hash (modelled as RO in proof) $H(.)$.
- $\mathcal{P}$ computes "labels" $\ell_{0000}, \ldots, \ell_{\varepsilon}$ of nodes where

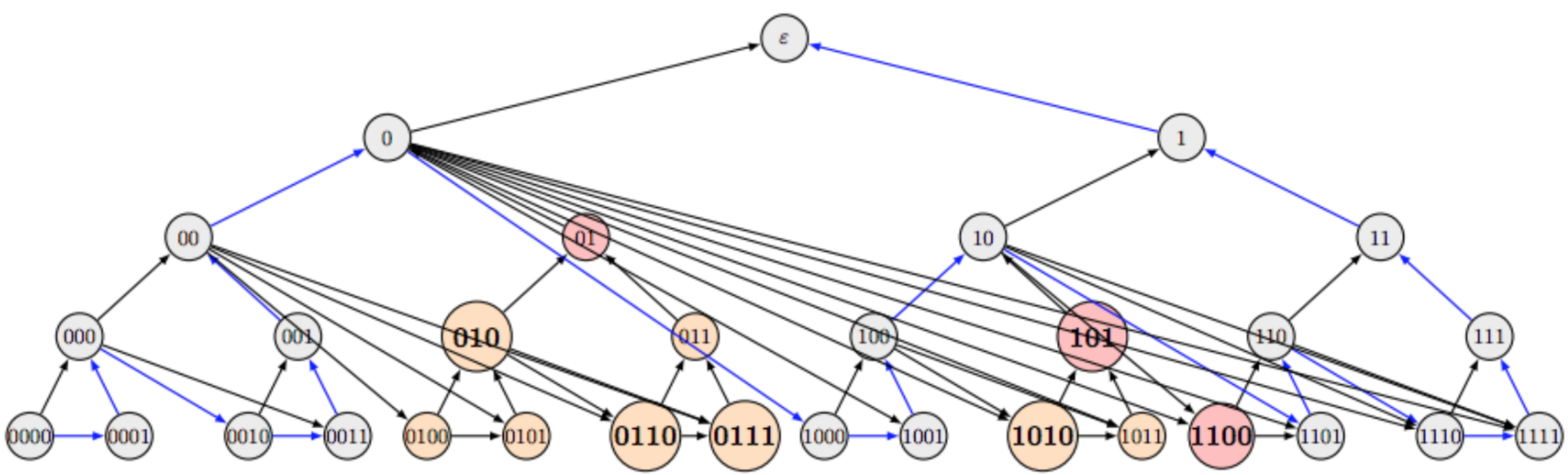
$$\ell_i = H(\ell_{p_1}, \ldots, \ell_{p_d}) \quad , \quad (p_1, \ldots, p_d) = parents(i)$$

  Sends label of root ℓ_{ε} to $\mathcal{V}$ (kinda Merkle-tree commitment to the ℓ_i's).
- $\mathcal{V}$ challenges $\mathcal{P}$ to open some random leaves with its parents, checks consistency.



Security

- After sending ℓ_ε $\mathcal{P}$ is "committed" to all labels.
- Assume $\mathcal{P}$ "cheated" on some labels $\ell_i \neq H(\ell_{p_1}, \ldots, \ell_{p_d})$. call those and all labels below them "bad"., let α be fraction of bad labels.
- $\mathcal{P}$ must have made $(1-\alpha)T$ sequential queries (in RO).
- $\mathcal{P}$ will fail if challenged to open a bad label $\Rightarrow$ will succeed on t random labels with prob. $\leq (1-\alpha)^t$.

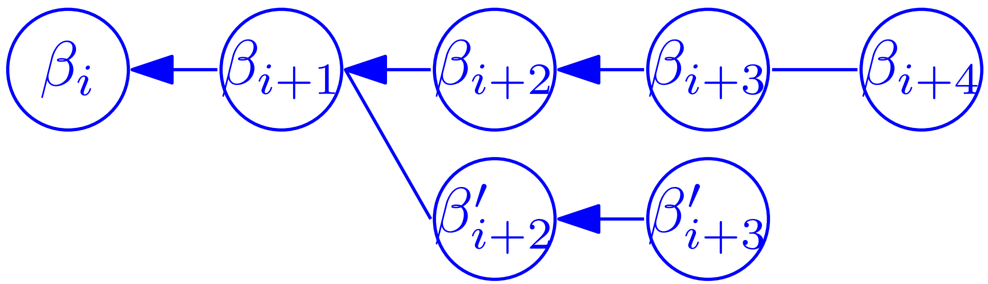


https://chia.net/

Bitcoin Mining Recap

Bticoin block $\beta_i = (\tilde{\beta}_i, \phi)$, $\tilde{\beta}_i = (i, H(\beta_{i-1}), pk, \tau_i)$ contains

- Transactions τ_i (consistent with chain so far).
- pk (block-reward and transactions fees go to pk)
- hash of previous block $H(\beta_{i-1})$
- PoW ϕ where $H(\phi, \tilde{\beta}_i) \leq treshhold$



Bitcoin mining:

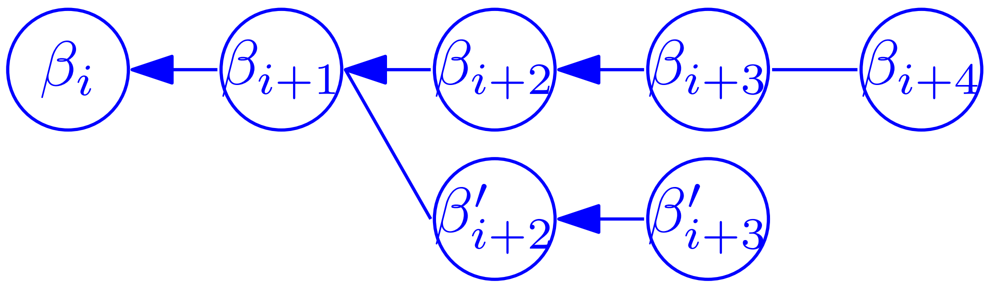
Init: sample signature key-pair (pk, sk)

1. Find head of longest chain β_{i-1} & compile block of transactions τ_i.
2. (PoW) hash distinct ϕ until $H(\phi, \ldots) \leq treshhold$
   - announce new block $\beta_i = (\phi, \ldots)$ and goto step 1.
   - if new longer chain observed immediately go to step 1.

Bitcoin Mining Recap

Bticoin block $\beta_i = (\tilde{\beta}_i, \phi)$, $\tilde{\beta}_i = (i, H(\beta_{i-1}), pk, \tau_i)$ contains

- Transactions τ_i (consistent with chain so far).
- pk (block-reward and transactions fees go to pk)
- hash of previous block $H(\beta_{i-1})$
- PoW ϕ where $H(\phi, \tilde{\beta}_i) \leq treshhold$



Bitcoin mining:

Init: sample signature key-pair (pk, sk)

1. Find head of longest chain β_{i-1} & compile block of transactions τ_i.
2. (PoW) hash distinct ϕ until $H(\phi, \ldots) \leq treshhold$
   - announce new block $\beta_i = (\phi, \ldots)$ and goto step 1.
   - if new longer chain observed immediately go to step 1.

Chia Building Blocks

Unique Digital Signatures

$$\forall m \ : \ \Pr[\mathsf{Sig.verify}(pk, m, \phi) = \mathsf{accept}] = 1$$

where $(pk, sk) \leftarrow \mathsf{Sig.keygen}$; $\phi \leftarrow \mathsf{Sig.sign}(sk, m)$

Unique : $(\mathsf{Sig.verify}(pk, m, \phi) = \mathsf{Sig.verify}(pk, m, \phi') = \mathsf{accept}) \Rightarrow (\phi = \phi')$

Unique & Signed Proofs of Space

$$S \leftarrow \mathsf{PoSpace.init}(pk, N)$$

$$\forall pk, N \ : \ \mathsf{PoSpace.verify}(c, \mathsf{PoSpace.prove}(S, pk, c)) = \mathsf{accept}$$

Weakly Unique : $\mathbb{E}_c \left[\{ \sigma \ : \ \mathsf{PoSpace.verify}(pk, c, \sigma) = \mathsf{accept} \} \right] = 1$

Signed : $\sigma = (\sigma', \mathsf{Sig.sign}(sk, \sigma'))$

Unique & Publicly Verifiable Proofs of Sequential Work

$$\forall t, c \ : \ \mathsf{PoSW.verify}(c, t, \mathsf{PoSW.prove}(c, t)) = \mathsf{accept}$$

$\mathsf{PoSW.prove}(c, t)$ should take almost sequential time t to compute

Unique : $(\mathsf{PoSW.verify}(c, t, \tau) = \mathsf{PoSW.verify}(c, t, \tau') = \mathsf{accept}) \Rightarrow \tau = \tau'$

Blockchain from Proofs of Space and Time

Space Miners (Farmers)

Initialization: $(pk, sk) \leftarrow$ Sig.KeyGen, $\Sigma \leftarrow$ PoSpace.Init(pk, N)

Mining: When new longest chain with head β_i observed:

compute $\phi \leftarrow$ PoSpace(Σ, c) for challenge $c := H(i, \beta_i, \tau_i, pk)$

gossip ϕ and define "quality" of ϕ as $q(\phi) := H(\phi)$.

Blockchain from Proofs of Space and Time

Space Miners (Farmers)

Initialization:$(pk, sk) \leftarrow$ Sig.KeyGen$, \Sigma \leftarrow$ PoSpace.Init(pk, N)
Mining: When new longest chain with head β_i observed:
compute $\phi \leftarrow$ PoSpace(Σ, c) for challenge $c := H(i, \beta_i, \tau_i, pk)$

gossip ϕ and define "quality" of ϕ as $q(\phi) := H(\phi)$.

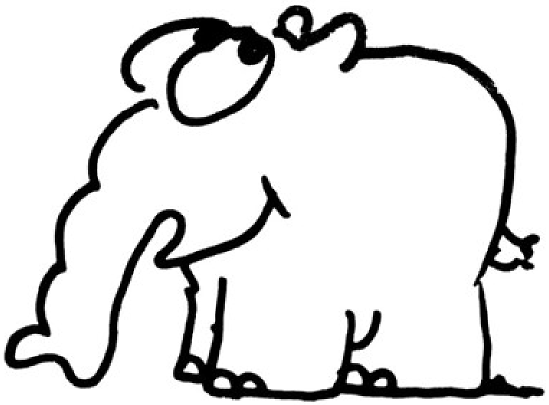
Time "Miners"

If PoSpace PoSpace ϕ observed, start computing

$\tau \leftarrow$ PoSW(challenge $= \phi$, time $= q(\phi) \cdot$ hardness parameter)

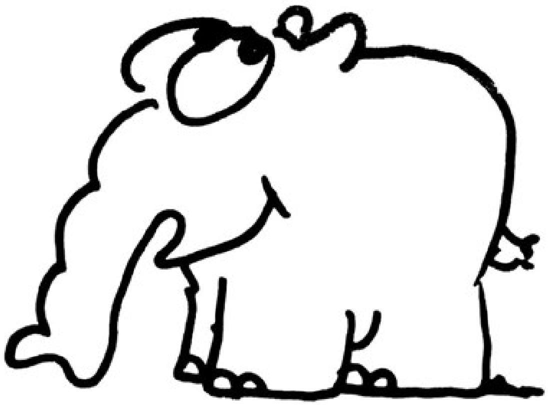
ONLY IF (given local view) this will be the first PoSW to finalize a block at this level.

Gossip τ once finished.



Attacks

Mining: When new longest chain with head β_i observed: compute $\phi \leftarrow \mathrm{PoSpace}(\Sigma, c)$ for challenge $c := H(i, \beta_i, \tau_i, pk)$



Attacks

Mining: When new longest chain with head β_i observed: compute $\phi \leftarrow \text{PoSpace}(\Sigma, c)$ for challenge $c := H(i, \beta_i, \tau_i, pk)$

Extending Multiple Chains: As (unlike PoW) computing a PoSpace is cheap, the miner can try to extend <u>all</u> blocks it learns about.

Attacks

Mining: When new longest chain with head β_i observed: compute $\phi \leftarrow \text{PoSpace}(\Sigma, c)$ for challenge $c := H(i, \beta_i, \tau_i, pk)$

Extending Multiple Chains: As (unlike PoW) computing a PoSpace is cheap, the miner can try to extend all blocks it learns about.
provably not such a big problem (wait two slides)

Attacks

Mining: When new longest chain with head β_i observed: compute $\phi \leftarrow \text{PoSpace}(\Sigma, c)$ for challenge $c := H(i, \beta_i, \tau_i, pk)$

Extending Multiple Chains: As (unlike PoW) computing a PoSpace is cheap, the miner can try to extend all blocks it learns about.
provably not such a big problem (wait two slides)

Grinding: try out many τ_i's to get different c's until one found which gives me a super high quality PoSpace for next round $\Rightarrow$ can hijack chain forever!

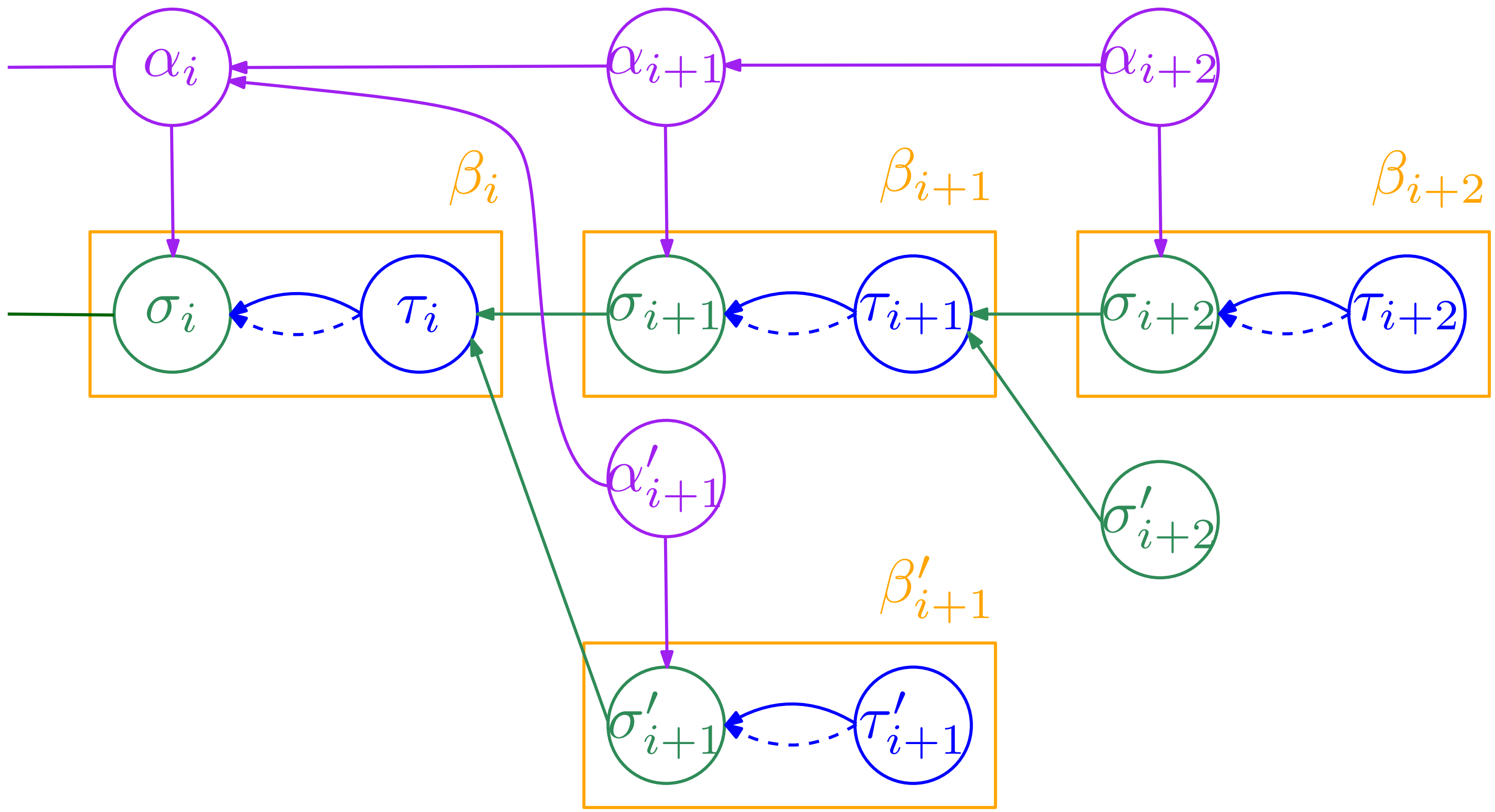
Attacks

Mining: When new longest chain with head β_i observed: compute $\phi \leftarrow \text{PoSpace}(\Sigma, c)$ for challenge $c := H(i, \beta_i, \tau_i, pk)$

Extending Multiple Chains: As (unlike PoW) computing a PoSpace is cheap, the miner can try to extend all blocks it learns about.
provably not such a big problem (wait two slides)

Grinding: try out many τ_i's to get different c's until one found which gives me a super high quality PoSpace for next round $\Rightarrow$ can hijack chain forever!
separate proofs from everything "graindable", Chia block format (next slide) kills the problem!

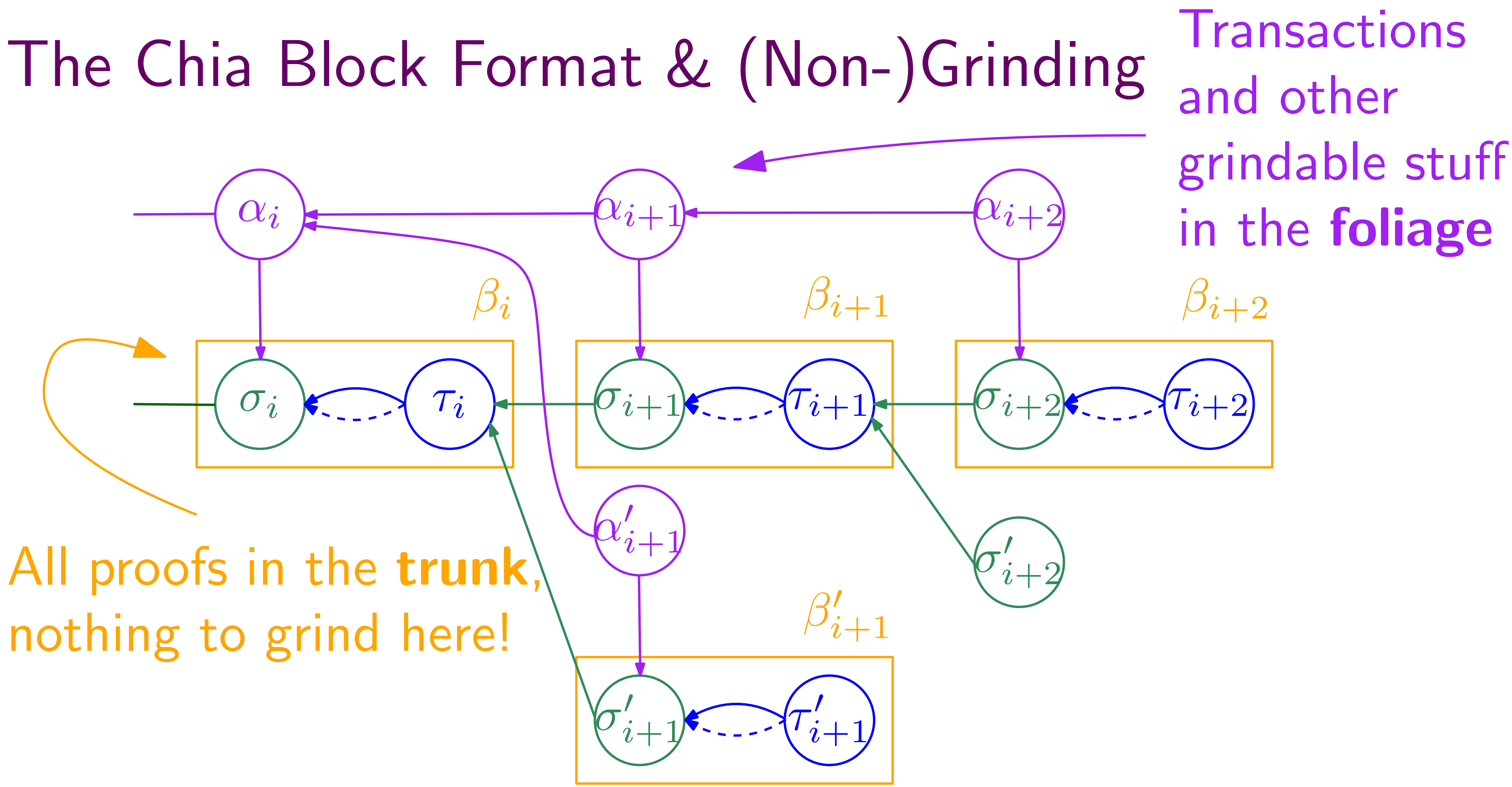
The Chia Block Format & (Non-)Grinding



A full block $\gamma_i = (\beta_i, \alpha_i)$ contains
$\beta_i = (i, (pk_i, \sigma_i), \tau_i)$ and $\alpha_i = (\phi_i, data_i)$

1. PoSpace.verify$(pk_i, H(\tau_{i-1}), \sigma_i, N) = 1$
2. PoSW.verify$(c, t, \tau_i) = 1$ where $c = H(\sigma_i)$, $t = 0.H(\sigma_i) \cdot T$
3. Sig.verify$(pk_i, H(\alpha_{i-1}, \sigma_i, data_i), \phi_i) = 1$

The Chia Block Format & (Non-)Grinding



A full block $\gamma_i = (\beta_i, \alpha_i)$ contains
$\beta_i = (i, (pk_i, \sigma_i), \tau_i)$ and $\alpha_i = (\phi_i, data_i)$

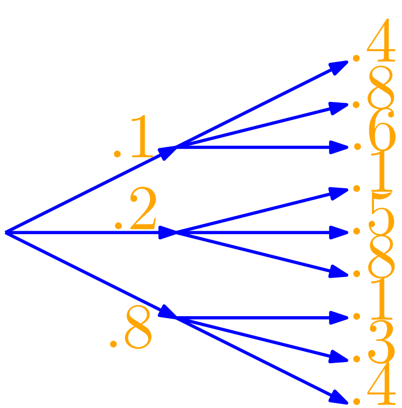
1. PoSpace.verify$(pk_i, H(\tau_{i-1}), \sigma_i, N) = 1$
2. PoSW.verify$(c, t, \tau_i) = 1$ where $c = H(\sigma_i)$, $t = 0.H(\sigma_i) \cdot T$
3. Sig.verify$(pk_i, H(\alpha_{i-1}, \sigma_i, data_i), \phi_i) = 1$

Analysing Chain Growth

- h honest miners, each has one unit of space.
- adversarial miner with m units of space.
- every unit of space for every challenge gives a proof of quality uniform in $[0, 1]$.
- to finalize a proof of quality α takes time α (all PoSW equally fast).
- adversary can run infinite number of PoSW.
- no network delays.

Analysing Chain Growth

- h honest miners, each has one unit of space.
- adversarial miner with m units of space.
- every unit of space for every challenge gives a proof of quality uniform in $[0,1]$.
- to finalize a proof of quality α takes time α (all PoSW equally fast).
- adversary can run infinite number of PoSW.
- no network delays.
- Consider h-ary tree of depth ℓ.
- Label every edge with random value from $[0,1]$.
- Random Variable $C^{\ell}_{\kappa,h}$ is length of shortest path we find when always following the κ best edges from root to a leave.

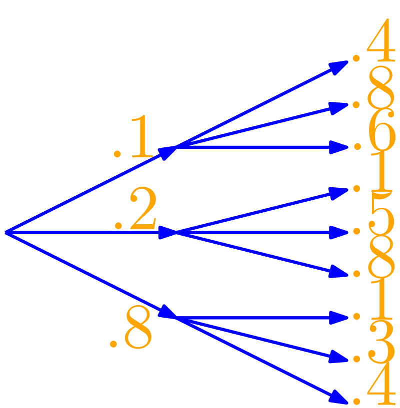


$C^{2}_{1,3} = .5$

$C^{2}_{\infty,3} = .3$

Analysing Chain Growth

- h honest miners, each has one unit of space.
- adversarial miner with m units of space.
- every unit of space for every challenge gives a proof of quality uniform in $[0,1]$.
- to finalize a proof of quality α takes time α (all PoSW equally fast).
- adversary can run infinite number of PoSW.
- no network delays.
- Consider h-ary tree of depth ℓ.
- Label every edge with random value from $[0,1]$.
- Random Variable $C^{\ell}_{\kappa,h}$ is length of shortest path we find when always following the κ best edges from root to a leave.



$C^2_{1,3} = .5$

$C^2_{\infty,3} = .3$

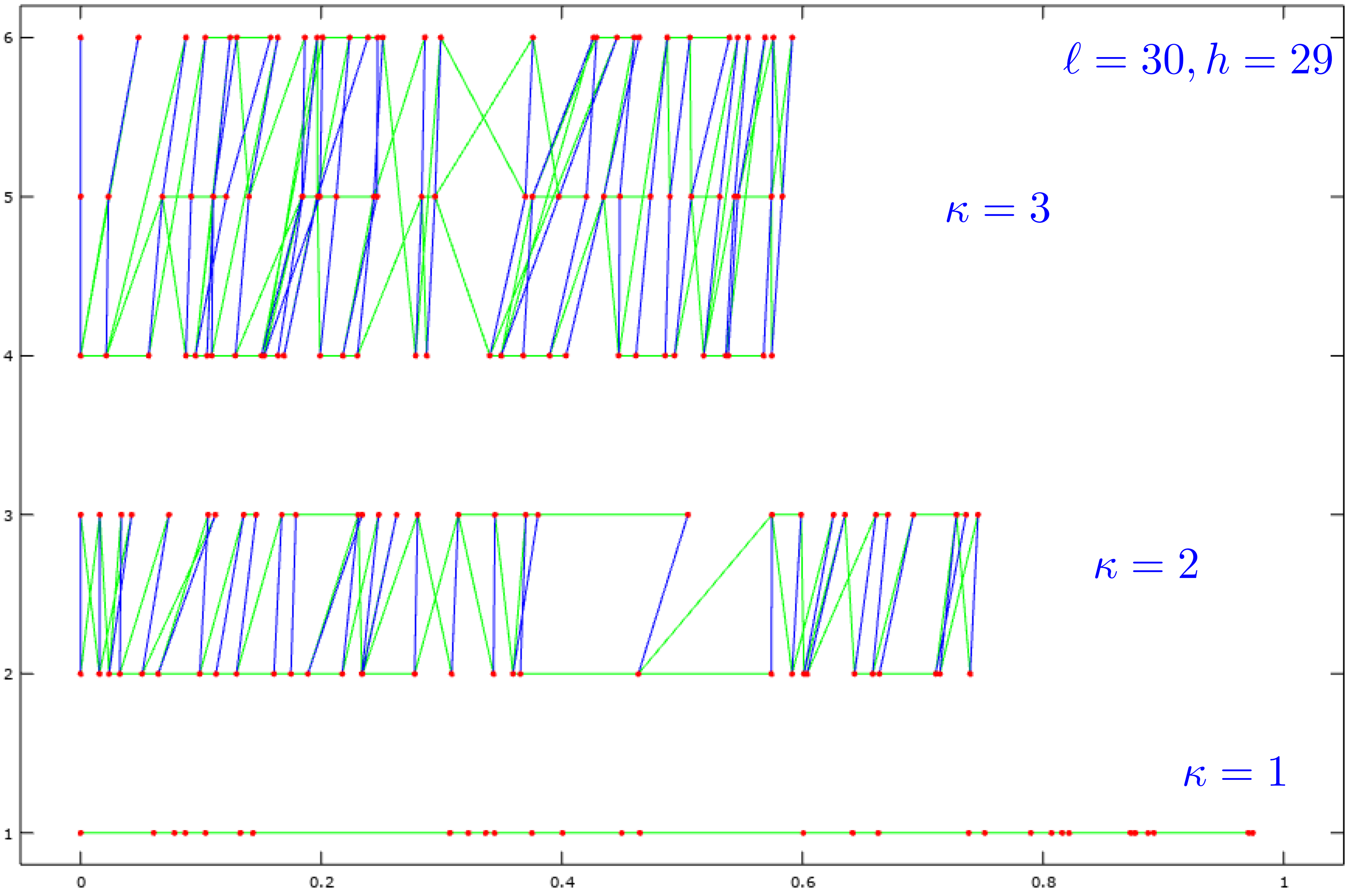
- $C^{\ell}_{\kappa,h}$ is expected time h honest miners need to grow chain of length ℓ.
- $C^{\ell}_{\infty,m}$ is expected time adversary controlling m space needs to grow chain of length ℓ.

Pseudocode For Sampling $C^{\ell}_{\kappa,h}$

Algorithm 1 sample $C^{\ell}_{\kappa,h}$

Input: κ, ℓ, h
$s[1,\ldots,\kappa] = 0$ ▷ initially we have κ paths of length 0
for $i = 1$ to ℓ **do** ▷ sample ℓ steps
    for $j = 1$ to κ **do** ▷ extend each of the κ states...
        for $k = 1$ to h **do** ▷ by h values...
            $p[j,k] = s[j] + \text{rand}([0,1])$ ▷ chosen uniform from $[0,1]$
        end for
    end for
    $z = \text{sort}(p[1,1],\ldots,p[\kappa,h])$ ▷ sort the $\kappa \cdot h$ values
    $s = z[1,\ldots,\kappa]$ ▷ new state are the κ shortest paths
end for
Return min(s)

Simulation of $C^{\ell}_{\kappa,h}$



What we know about $C^{\ell}_{\kappa,h}$

1. $C^{\ell}_{\kappa,h}$ is expected time h honest miners need to grow chain of length ℓ <u>without adversarial interference</u>
2. **No Slowdown Lemma**: an adversary with unbounded space and parallelism (but which cannot break the underlying signature scheme) cannot slow down the rate at which this chain grows.
3. We know exact expectation for $\kappa = 1$
$$E[C^{\ell}_{1,h}] = \frac{\ell}{h+1}$$
4. We can lower bound for $\kappa = \infty$
$$E[C^{\ell}_{\infty,h}] \geq \frac{\ell}{h+1} \cdot \frac{1}{e}$$

What we know about $C^{\ell}_{\kappa,h}$

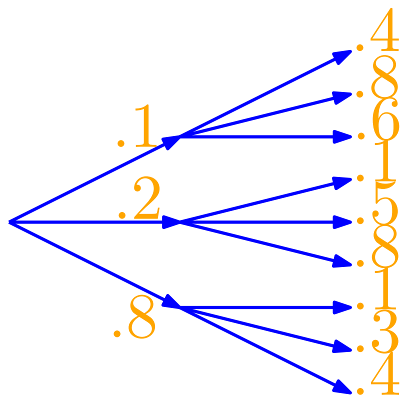
1. $C^{\ell}_{\kappa,h}$ is expected time h honest miners need to grow chain of length ℓ without adversarial interference
2. **No Slowdown Lemma**: an adversary with unbounded space and parallelism (but which cannot break the underlying signature scheme) cannot slow down the rate at which this chain grows.
3. We know exact expectation for $\kappa = 1$
$$E[C^{\ell}_{1,h}] = \frac{\ell}{h+1}$$
4. We can lower bound for $\kappa = \infty$
$$E[C^{\ell}_{\infty,h}] \geq \frac{\ell}{h+1} \cdot \frac{1}{e}$$

(Weak) Chain Quality Lemma: If $m < h/e$ (m space controlled by adversary, h honest space) then the fraction of honestly mined blocks is > 0.

Proof Sketch

- We can lower bound for $\kappa = \infty$

$$E[C^{\ell}_{\infty,h}] \geq \frac{\ell}{h+1} \cdot \frac{1}{e}$$



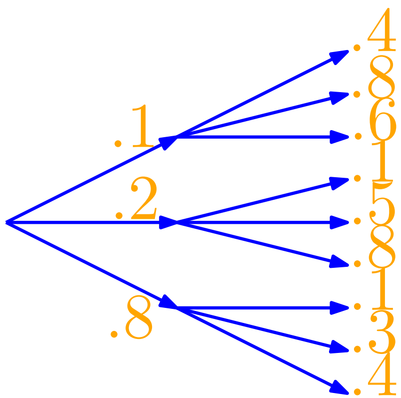
$C^2_{1,3} = .5$

$C^2_{\infty,3} = .3$

Proof Sketch

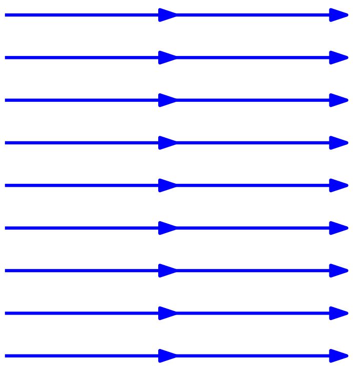
- We can lower bound for $\kappa = \infty$

$$E[C^{\ell}_{\infty,h}] \geq \frac{\ell}{h+1} \cdot \frac{1}{e}$$



$C^2_{1,3} = .5$

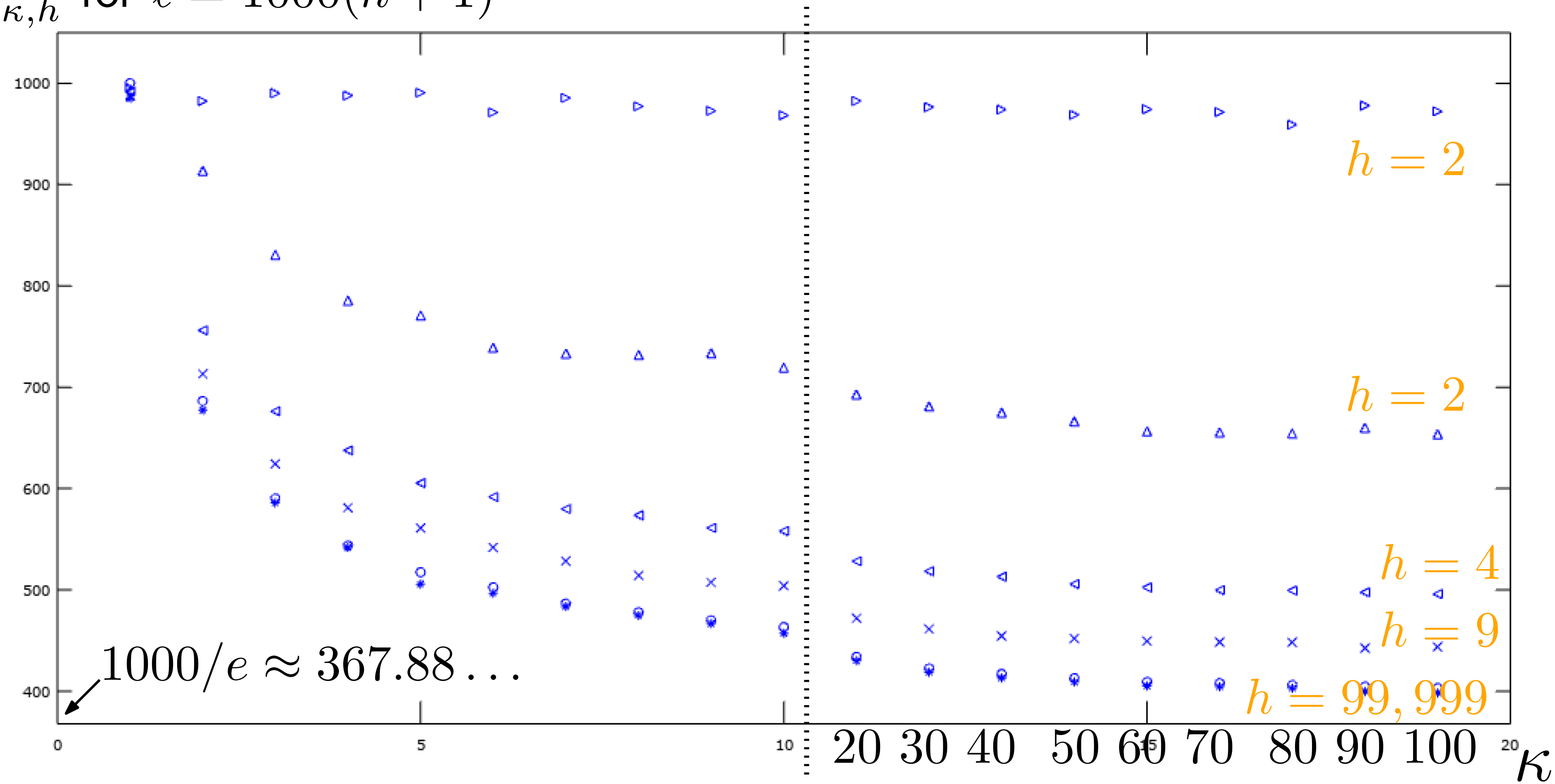
$C^2_{\infty,3} = .3$



- Instead of analyzing shortest path in h-ary tree of depth ℓ, consider h^ℓ independent paths, prove that this tilts the bound in right direction.
- (Chernoff) Show that probability that any of those is shorter than x is $\ll \frac{1}{h^\ell}$.
- (Union Bound) Whp. all h^ℓ of them are shorter than x.

Choosing κ

$C^{\ell}_{\kappa,h}$ for $\ell = 1000(h+1)$



κ :	1	2	3	4	5	6	7	8	9
$C^{\ell}_{1,h}/C^{\ell}_{\kappa,h}$:	1	1.45	1.69	1.83	1.93	1.99	2.05	2.09	2
$e_\kappa \approx \frac{e}{C^{\ell}_{1,h}/C^{\ell}_{\kappa,h}}$:	2.71	1.86	1.60	1.47	1.40	1.36	1.32	1.29	1

Pseudocode

Space Miner Pseudocode (1/2)

Algorithm 2 SpaceMiner.init

Global Parameters: N

$\mathcal{C} \leftarrow$ Chain.init ▷ extract view from network

$(pk, sk) \leftarrow$ Sig.keygen ▷ generate a signature key pair

$S \leftarrow$ PoSpace.init(N, pk). ▷ run PoSpace initialisation with space N and identity pk to get a file S of size $|S| = N$.

Initalize a vector pos_count to all 0 ▷ see Remark **??**

Output:

$(pk, sk), S,$ pos_count ▷ State for SpaceMiner.mine

$\mathcal{C}$ ▷ State for Chain.update

Algorithm 3 SpaceMiner.loop

loop

  Wait for block(s) Γ to be received from the network

  $(\Gamma_f, \Gamma_n) \leftarrow$ Chain.update(Γ)

  $\forall \gamma \in \Gamma_f \ :$ SpaceMiner.mine(γ) ▷ Algorithm 4

end loop

Space Miner Pseudocode (2/2)

Algorithm 4 SpaceMiner.mine

Global Parameters: κ
Input: $\gamma_i = (\beta_i = (i, \sigma_i, \tau_i), \alpha_i)$. $\triangleright$ finalized, fresh & valid block for slot i
State: (pk, sk), S, pos_count
if pos_count$(i) = \kappa$ **then** $\triangleright$ already generated κ PoS for slot i
    return without output
end if
pos_count$(i) \leftarrow$ pos_count$(i) + 1$
$\sigma_{i+1} \leftarrow$ PoSpace.prove$(S, pk, \mathsf{H}(\tau_i))$ $\triangleright$ produce PoSpace
Generate data$_{i+1}$ $\triangleright$ application specific
$\phi_{i+1} \leftarrow$ Sig.sign$(sk, (\alpha_i, \sigma_{i+1}, \text{data}_{i+1})$ $\triangleright$ signature for signature chain

Chain.update$((i+1, \sigma_{i+1}), \alpha_{i+1} = (\phi_{i+1}, \text{data}_{i+1}))$ $\triangleright$ Cf. §**??**

Time Miner Pseudocode (1/3)

Algorithm 5 TimeMiner.init

$\mathcal{C} \leftarrow$ Chain.init ▷ extract view from network
Initalize a vectors finalized and running to all 0
Output:
finalized, running ▷ State for TimeMiner.mine/finalized/runPoSW
$\mathcal{C}$ ▷ State for Chain.update

Algorithm 6 TimeMiner.loop

loop
    Wait for block(s) Γ to be received from the network
    $(\Gamma_f, \Gamma_n) \leftarrow$ Chain.update(Γ)
    $\forall((i,\sigma),\alpha) \in \Gamma_n$: TimeMiner.mine(i,σ) ▷ Algorithm 7
    $\forall((i,\sigma,\tau),\alpha) \in \Gamma_f$: TimeMiner.finalized(i) ▷ Algorithm 9
end loop

Algorithm 7 TimeMiner.mine

Global Parameters: T, κ
Input: $\beta_i = (i, \sigma_i)$ ▷ non-finalized, fresh & valid block for slot i received
State: finalized, running
if finalize$[i] = \kappa$ **then** ▷ already finalized κ blocks for this slot
    return with no output
end if
$t := 0.\mathsf{H}(\sigma_i) \cdot T$ ▷ time required to finalize this block
if finalize$[i]$ + running$[i] < \kappa$ **then** ▷ $< \kappa$ proofs finalized or running
    start thread TimeMiner.runPoSW$(i, \mathsf{H}(\sigma_i), t)$ ▷ to finish at time now $+ t$
    running$[i]$ = running$[i] + 1$
end if
if finalize$[i]$ + running$[i] = \kappa$ **then** ▷ exactly κ proofs finalized or running
    if the slowest PoSW for slot i will finish at time $> t +$ now **then**
        abort the thread of this PoSW
        start thread TimeMiner.runPoSW$(i, \mathsf{H}(\sigma_i), t)$
    end if
end if

Time Miner Pseudocode (3/3)

Algorithm 8 TimeMiner.runPoSW

State: finalized, running
Input: $i, (c, t)$
$\tau_i \leftarrow$ PoSW(c, t) ▷ start PoSW, if not aborted will output proof τ_i in time t
finalized$[i] =$ finalized$[i] + 1$
running$[i] =$ running$[i] - 1$
Chain.update(τ_i)

Algorithm 9 TimeMiner.finalized

State: finalized, running
Input: i ▷ fresh, valid & finalized block for slot i was received
if running$[i] > 0$ and running$[i]$+finalized$[i] = \kappa$ **then**
    abort the thread TimeMiner.runPoSW for slot i scheduled to finish last
    running$[i] =$ running$[i] - 1$
end if
finalized$[i] = \min\{$finalized$[i] + 1, \kappa\}$

Some Open Problems

- (PoSW) Construct unique proofs of sequential work without heavy crypto machinery (SNARKs).
- (PoS) Is there a proof of space with non-interactive initialization and (at least asymptotically) optimal bounds?
- (Analysis) Better chain quality, persistience etc. analysis? Can we say something about rational (not just honest) miners?