KAIST

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

SpaceMint: A Cryptocurrency Based on Proofs of Space

2019.04.24.

20184327 Seunggeun Baek

Cynthia Dwork & Moni Naor

The Birth of PoW

1992

- **Cynthia Dwork** and **Moni Naor**. "Pricing via processing or combatting junk mail." *Annual International Cryptology Conference*. 1992.

2002

- Adam Back. "Hashcash-a denial of service counter-measure." 2002.

2008

- Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." 2008.

The Birth of Proofs of Space

2003

- Martin Abadi et al. "Moderately hard, memory-bound functions." Proceedings of the 10th Annual Network and Distributed System Security Symposium, 2003.

was in concurrent work with,

2003

- **Cynthia Dwork**, Andrew Goldberg, and **Moni Naor**. "On memory-bound functions for fighting spam." *Annual International Cryptology Conference*. 2003.

2005

- **Cynthia Dwork**, **Moni Naor**, and Hoeteck Wee. "Pebbling and proofs of work. " *Annual International Cryptology Conference*. 2005.

The Birth of Proofs of Space (cont.)

2010

- Daniele Perito and Gene Tsudik. "Secure code update for embedded devices via proofs of secure erasure." *European Symposium on Research in Computer Security*. 2010.

2014

- Giuseppe Ateniese et al. "Proofs of space: When space is of the essence." *International Conference on Security and Cryptography for Networks*. 2014.

2015

- Stefan Dziembowski et al. "Proofs of space." *Annual Cryptology Conference*. 2015.
- Spacecoin (First draft of this work, later changed to SpaceMint)

Contents

A Proofs of Space

1. Graph Pebbling
2. Proofs of Space (PoSpace)
3. Related Schemes

B SpaceMint

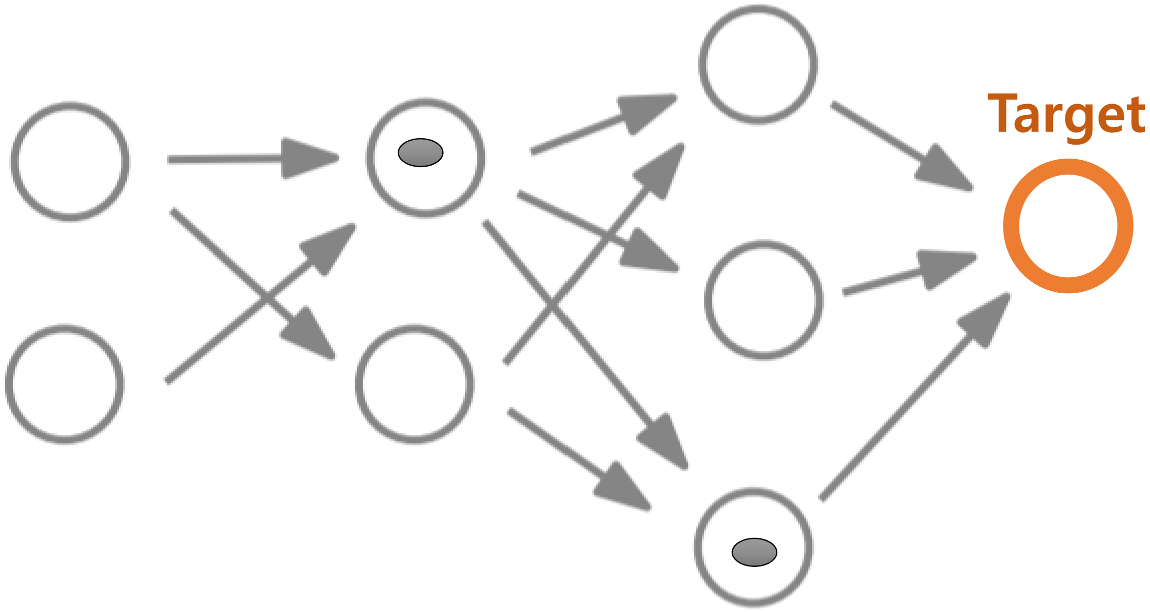
4. Protocol
5. Design Challenges
6. Experiments
7. Analysis based on Game Theory

Some diagrams were brought from Georg Fuchsbauer's presentation slides.

A Proofs of Space

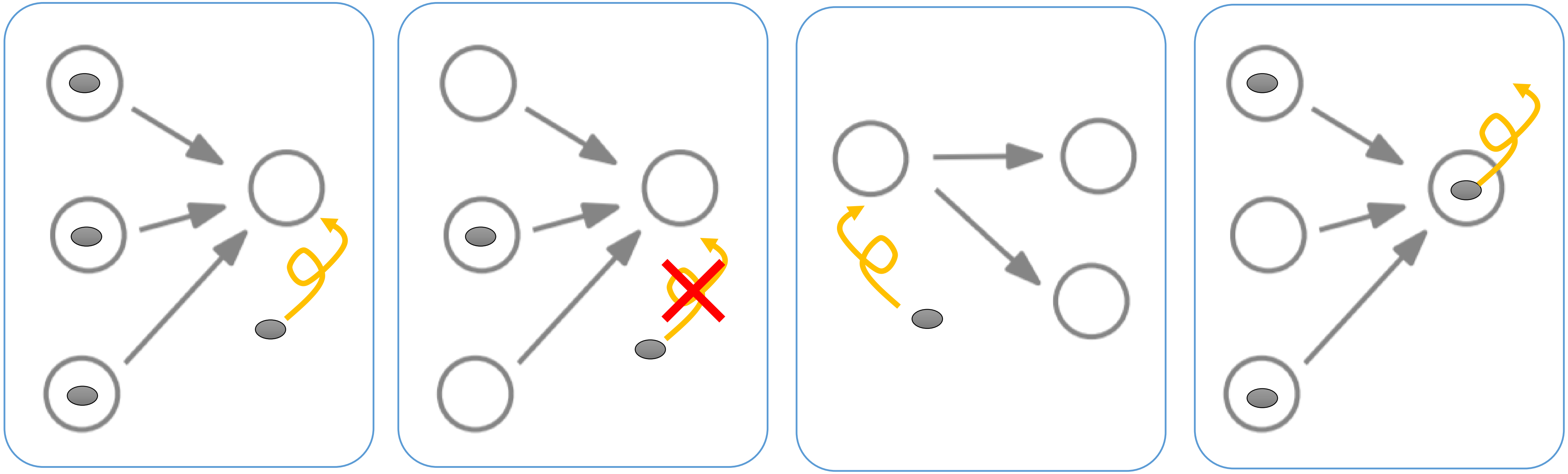
Graph Pebbling Game

- Consider a DAG that each node has a slot for pebble placement.
  - Some slots may have pebbles initially.
- Objective: Pebble the target node, according to some rules.



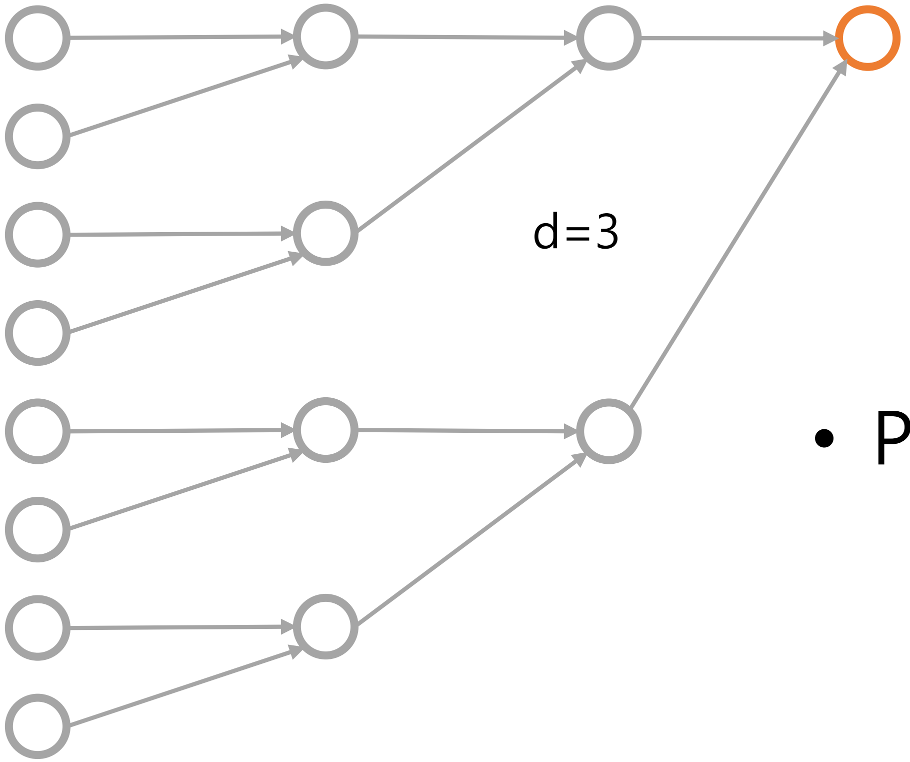
Pebbling Rules

- Placement: A node can be pebbled if it is either a source, or all its direct predecessors are pebbled.
- Removal: A pebble can be removed from a node, unconditionally.



Example: Binary Tree

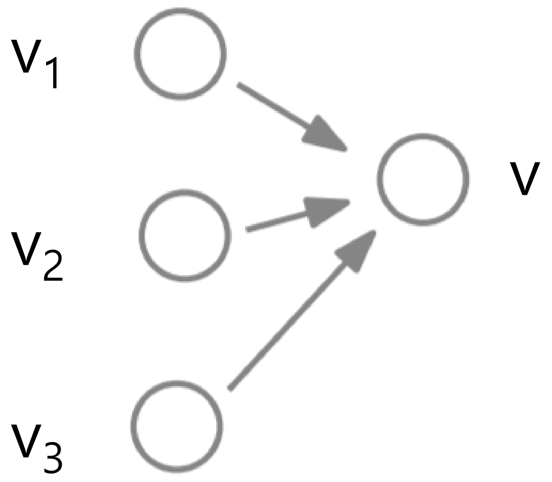
- A perfect binary tree with depth d (edge reversed)
- $2^{d+1}-1$ total nodes, 2^{d+1} total edges



- Pebbling Complexity
  - Required number of pebbles: d+2
  - Number of pebble placement: $2^{d+1}-1$

Link to Memory Usage

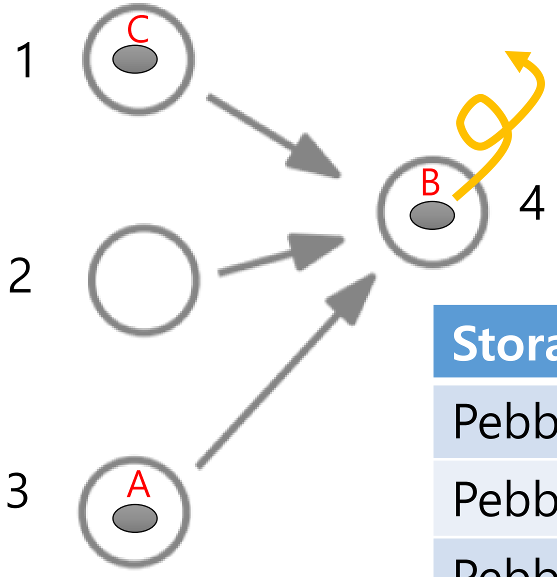
- Let a value of each non-source node is calculated by hash of its predecessor nodes.
  - Example: Merkle Tree
- It is computationally infeasible to calculate a node value, without storing values of predecessor nodes.



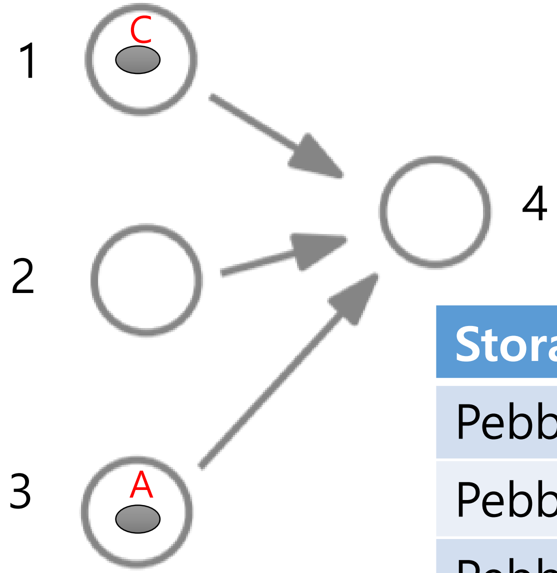
$$w(v) = H(v \,||\, w(v_1) \,||\, w(v_2) \,||\, w(v_3))$$

Link to Memory Usage (cont.)

- Pebbled Nodes: Nodes with their values currently stored
- Placement: To calculate and store the value of the corresponding node by hashing its predecessors
- Removal: To erase the node value from the memory.



Storage	Value
Pebble A	3
Pebble B	4
Pebble C	1



Storage	Value
Pebble A	3
Pebble B	-
Pebble C	1

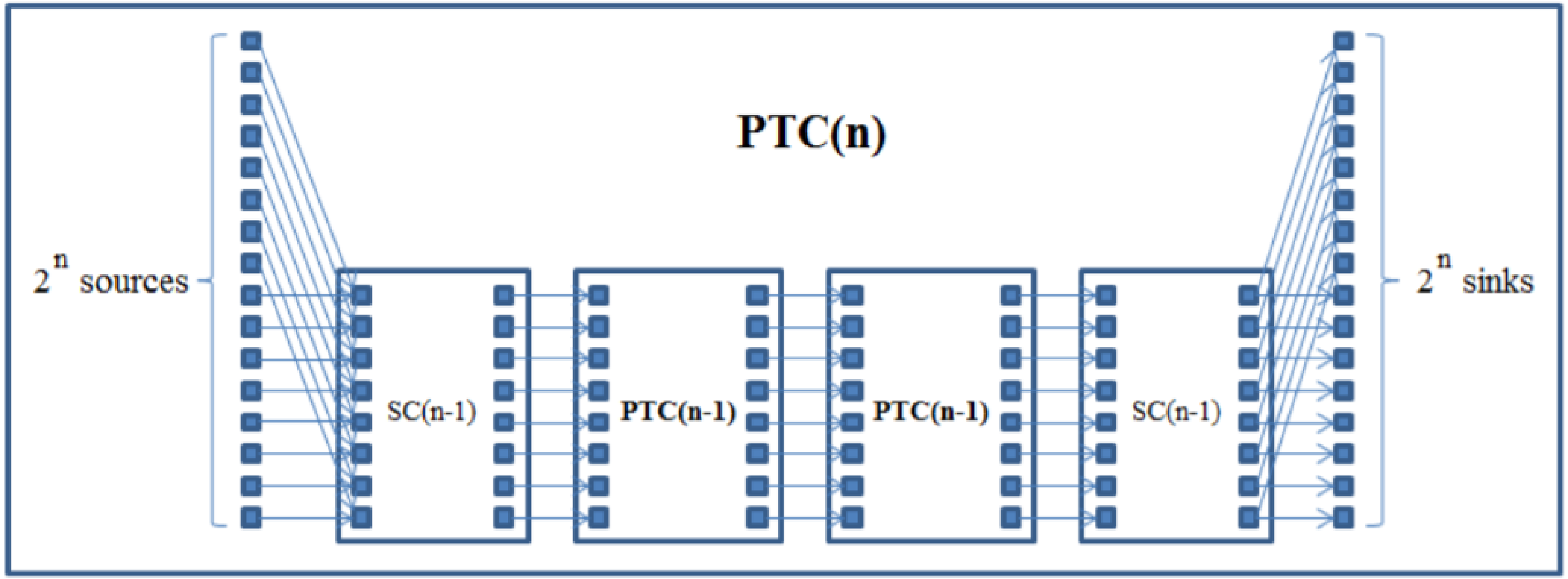
Link to Memory Usage (cont.)

- Pebbled Nodes: Nodes with their values currently stored
- Placement: To calculate and store the value of the corresponding node by hashing its predecessors
- Removal: To erase the node value from the memory.

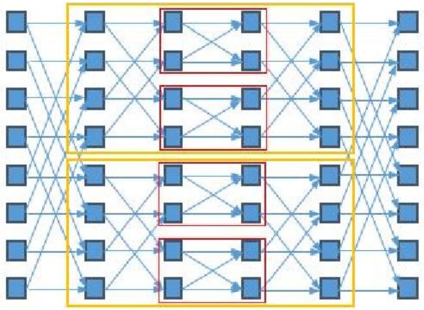
- **Required number of pebbles = Minimum storage required**

Hard-to-pebble Graphs

- There exist some families of graphs that require Ω(|V|/log|V|), or even Θ(|V|) pebbles.



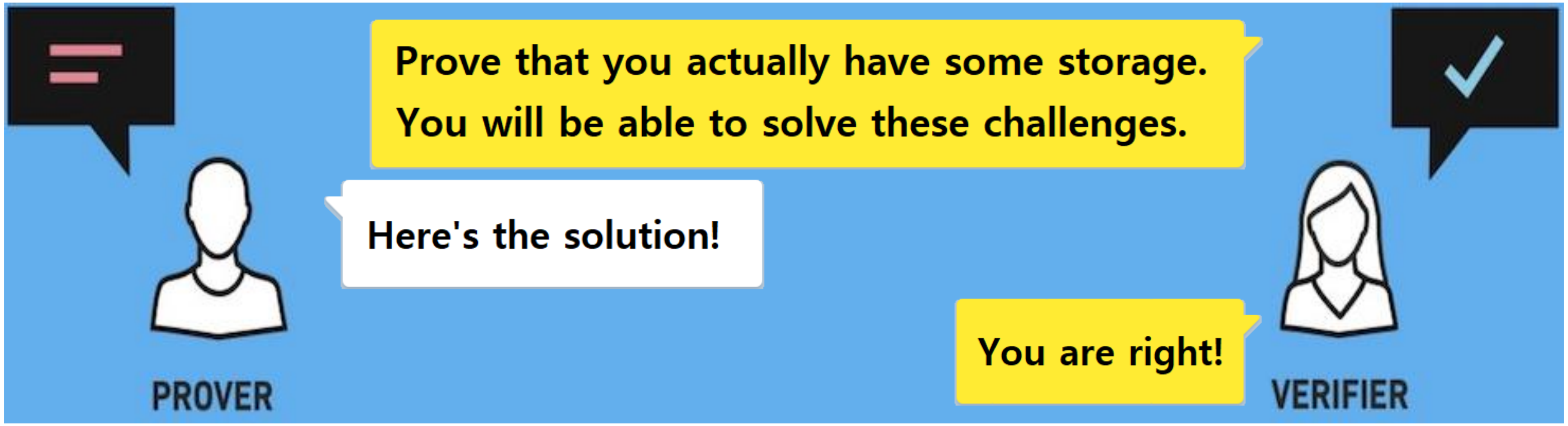
SC: Superconcentrators like Butterfly Graph



Images from Bhupatiraju et al. "On the Viability of Distributed Consensus by Proof of Space." 2017.

Proofs of Space (PoSpace)

- PoSpace
  - An interactive protocol between V (Verifier) and P (Prover)



- P opens a ‘proof’ to claim that P did memory-required work.
- From the proof, V should accept that P has utilized the corresponding amount of space.

Proofs of Space (PoSpace)

- Parameters $\mathsf{prm} = (\mathsf{id}, N, \ldots)$ N: Storage Bound

- Initialization

$$(\Phi, S) \leftarrow \langle \mathsf{V}, \mathsf{P} \rangle (\mathsf{prm})$$

Φ : Verifier's value, short
S : Prover's data with size N

- Execution $(\{\mathsf{accept}, \mathsf{reject}\}, \emptyset) \leftarrow \langle \mathsf{V}(\Phi), \mathsf{P}(S) \rangle (\mathsf{prm})$

Soundness and Completeness

Completeness: We will require that for any honest prover P:

$$\Pr[\mathsf{out} = \mathsf{accept} \;:\; (\Phi, S) \leftarrow \langle \mathsf{V}, \mathsf{P} \rangle(\mathsf{prm})\;,\; (\mathsf{out}, \emptyset) \leftarrow \langle \mathsf{V}(\Phi), \mathsf{P}(S) \rangle(\mathsf{prm})] = 1.$$

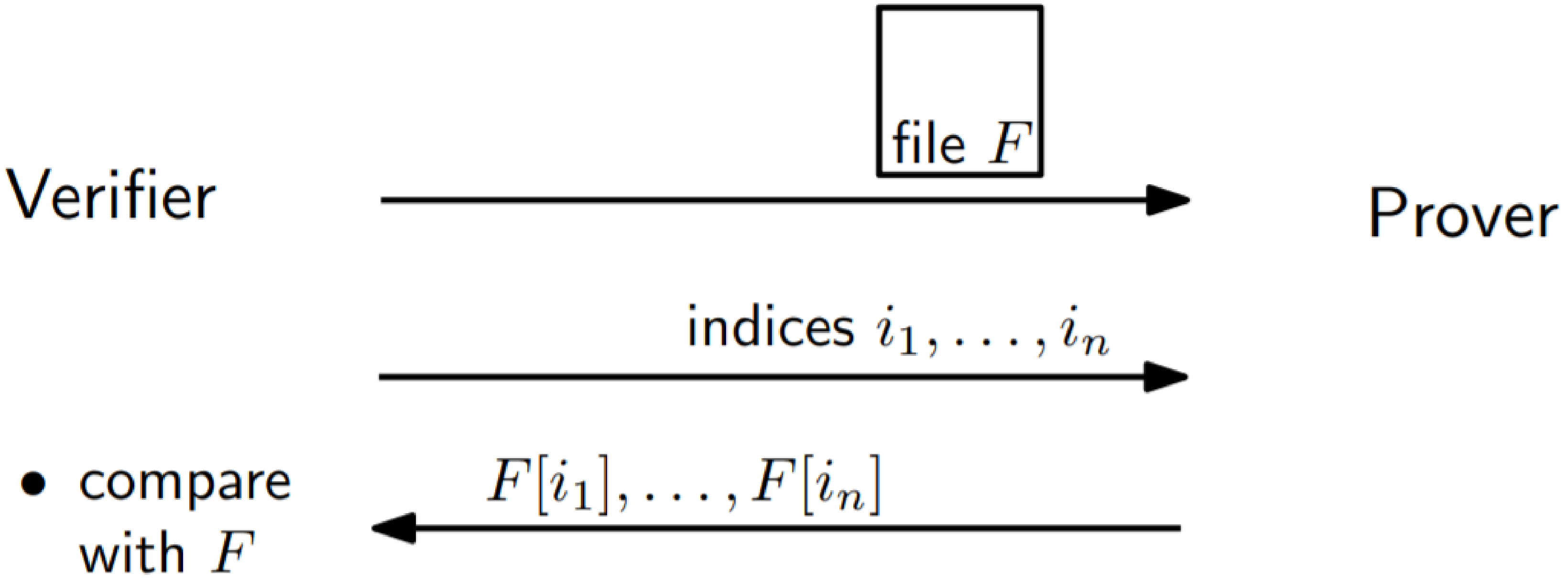
Note that the probability above is exactly 1, and hence the completeness is perfect.

Soundness: For any (N_0, N_1, T)-adversarial prover $\tilde{\mathsf{P}}$ the probability that V accepts is negligible in some statistical security parameter γ. More precisely, we have

$$\Pr[\mathsf{out} = \mathsf{accept} \;:\; (\Phi, S) \leftarrow \langle \mathsf{V}, \tilde{\mathsf{P}} \rangle(\mathsf{prm}), (\mathsf{out}, \emptyset) \leftarrow \langle \mathsf{V}(\Phi), \tilde{\mathsf{P}}(S) \rangle(\mathsf{prm})] \leq 2^{-\Theta(\gamma)} \quad (1)$$

Efficiency

Efficiency: We require the verifier V to be efficient, by which (here and below) we mean at most polylogarithmic in N and polynomial in some security parameter γ. Prover P must be efficient during execution, but can run in time poly(N) during initialization.[9]



A Basic, Inefficient Design

Parameters prm $= (\mathsf{id}, N, G = (V, E), \Lambda)$, where G is a graph on $|V| = N$ vertices and Λ is an efficiently samplable distribution over V^β (we postpone specifying β as well as the function of id to Sect. 6).

Initialization $(S, \emptyset) \leftarrow \langle \mathsf{P}_0, \mathsf{V}_0 \rangle(\mathsf{prm})$ where $S = w(V)$.

Execution $(\mathsf{accept}/\mathsf{reject}, \emptyset) \leftarrow \langle \mathsf{V}(\emptyset), \mathsf{P}(S) \rangle(\mathsf{prm})$

1. $\mathsf{V}_0(\emptyset)$ samples $C \leftarrow \Lambda$ and sends C to P_0.
2. $\mathsf{P}_0(S)$ answers with $A = w(C) \subset S$.
3. $\mathsf{V}_0(\emptyset)$ outputs accept if $A = w(C)$ and reject otherwise.

- The verifier is inefficient!

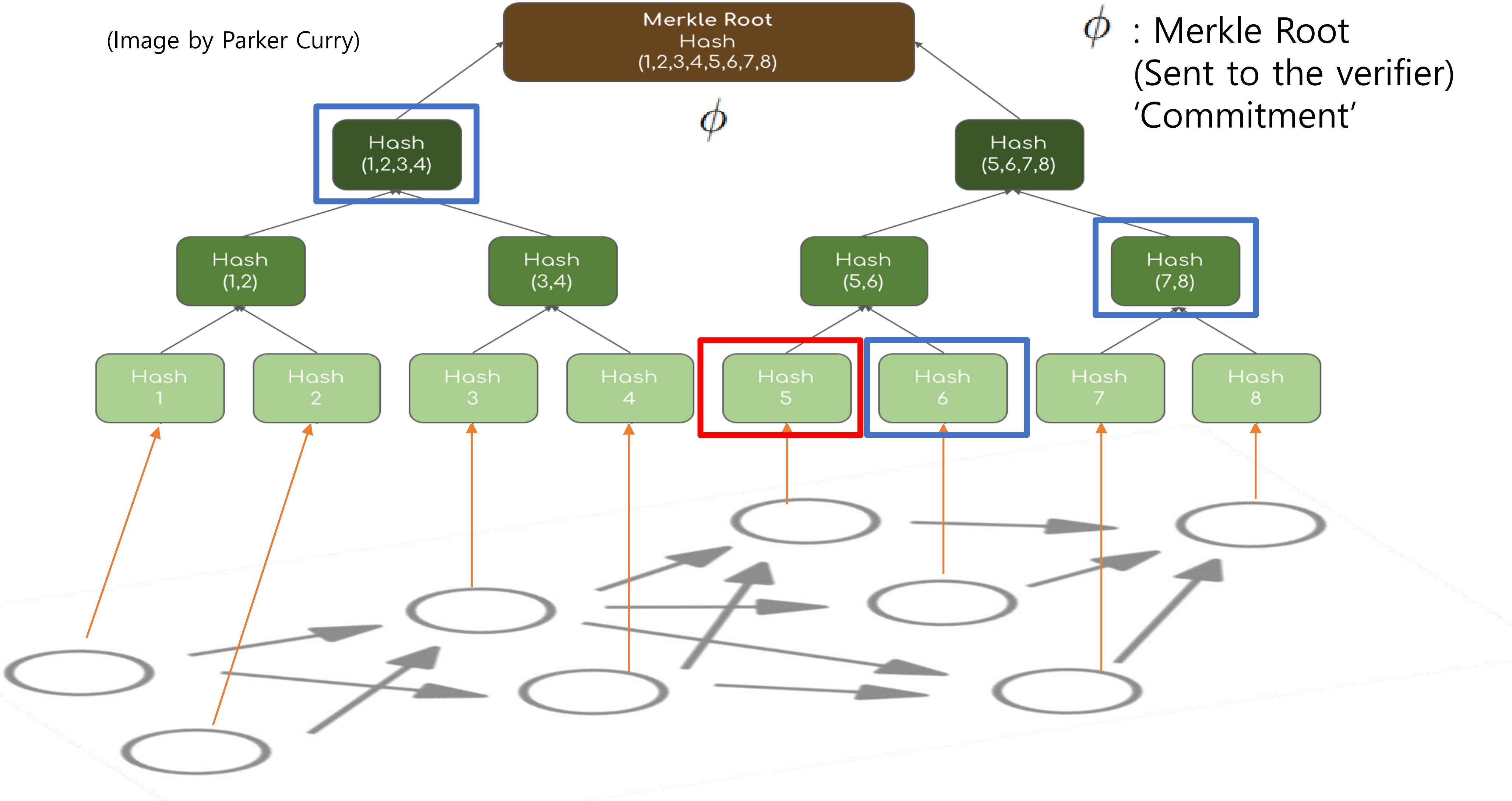
Efficient Verification with Merkle Tree

(Image by Parker Curry)

Merkle Tree

Total 2N-1 nodes

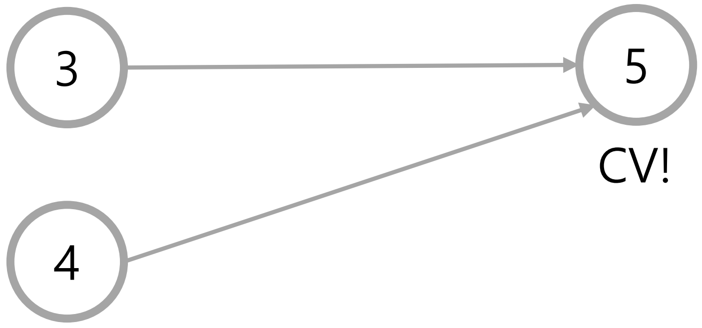
Underlying
Hard-to-pebble
Graph



ϕ : Merkle Root
(Sent to the verifier)
'Commitment'

Efficient Verification (cont.)

- Commitment Verification

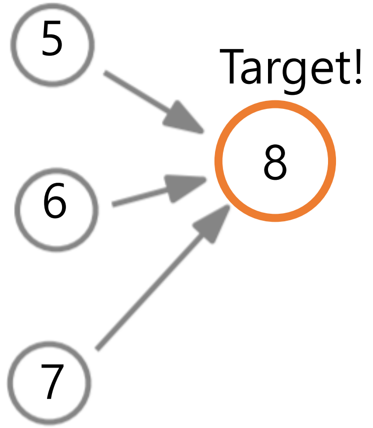


Prover gives:
w(3), open(3)
w(4), open(4)

open(5)

Verifier Calculates:
ϕ , from w(3) and open(3)
ϕ , from w(4) and open(4)
w(5), from w(3) and w(4)
ϕ , from w(5) and open(5)

- Proof Verification



Prover gives:
w(8), open(8)

Verifier Calculates:
ϕ , from w(8) and open(8)

Space-related Cryptocurrencies

	SpaceMint	Burstcoin	Permacoin
Proof of …	Space	Capacity	Retrievability
PoW-like?	X	Δ (Time-memory Tradeoff)	O
Meaningful Data?	X	Δ*	O
Verification	~100ms	8M hashes	~5ms

* Currently not, but development of PoC3 aims to use meaningful data as the plot file.

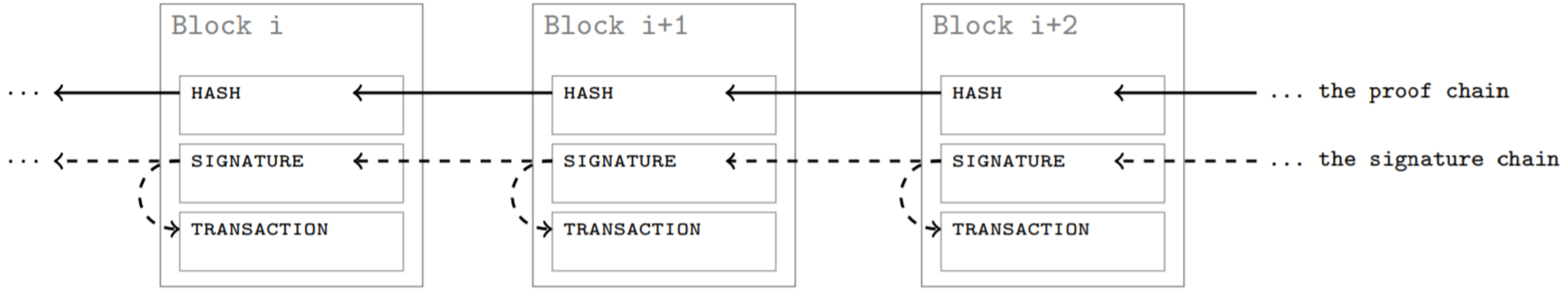
B SpaceMint

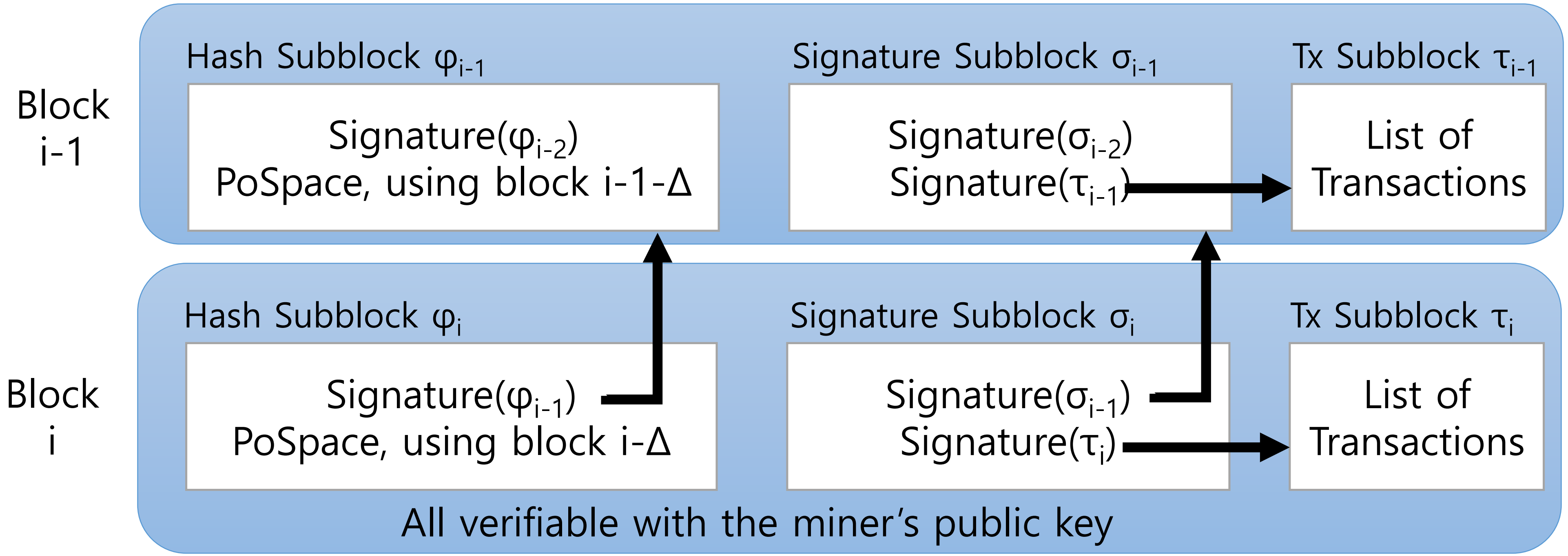
Designing SpaceMint

- Avoiding PoW-style consensus
  - Purely based on the storage
  - No memory-time tradeoff

- PoSpace-based
  - Guarantees that honest provers use corresponding amount of storage to extend a block
  - Proof size: logarithmic to the dedicated storage

Overall Block Structure





Each subblock contains the block number.

Initialization

- To dedicate some storage for PoSpace, a future prover should write a space commitment transaction.

$$(\gamma, S_\gamma) := \mathsf{Init}(pk, N)$$

Space size

Privately storing: (S_γ, sk)

Written transaction: $ctx = (\mathsf{commit}, txId, (pk, \gamma))$

Toward Non-interactive PoSpace

- Problem of interactive protocol
  - Prover should answer every verification request.
  - This means, miner should maintain connection and keep verify.
  - Impossible to implement in public blockchain

- Making non-interactive PoSpace
  - Derive randomness from some public information (previous blocks).
  - Replace verifiers' node selection with the randomness.

c is then expanded into sufficiently long random strings $\$_p, \$_{cv}$

Mining

2. samples $(c_1, \ldots, c_{k_p}) \leftarrow \mathsf{Chal}(n, k_p, \$_p)$ as in Algorithm 3;
3. computes the proof $a := \{a_1, \ldots, a_p\}$ as in Algorithm 3, i.e., $a_i = \mathsf{Ans}(pk, S_\gamma, c_i)$;

Block Quality

- Property of Quality Measure

$$\Pr_{\text{hash}}[\forall j \neq i : \text{Quality}(\pi_i) > \text{Quality}(\pi_j)] = \frac{N_{\gamma_i}}{\sum_{j=1}^{m} N_{\gamma_j}}$$

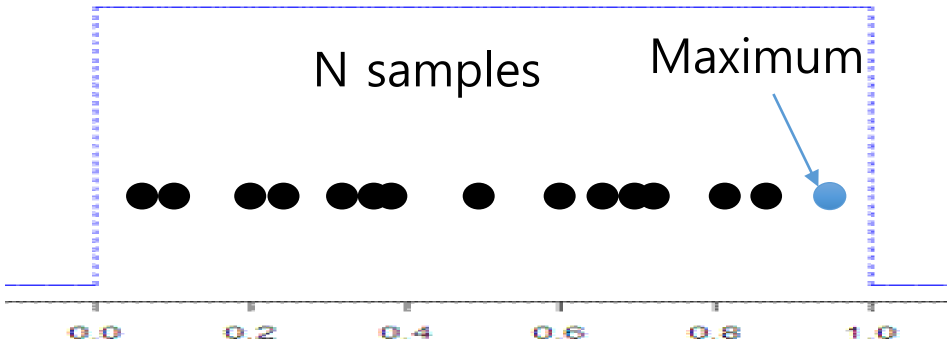
Probability that the block i becomes the best quality block = Portion of dedicated space to mine block i

$$\Pr_{\text{hash}}[\text{Quality}(\pi_i) > \text{Quality}(\pi_j)] = \frac{N_{\gamma_i}}{N_{\gamma_i} + N_{\gamma_j}}$$

Probability that the block i has better quality than j = Relative portion of dedicated space

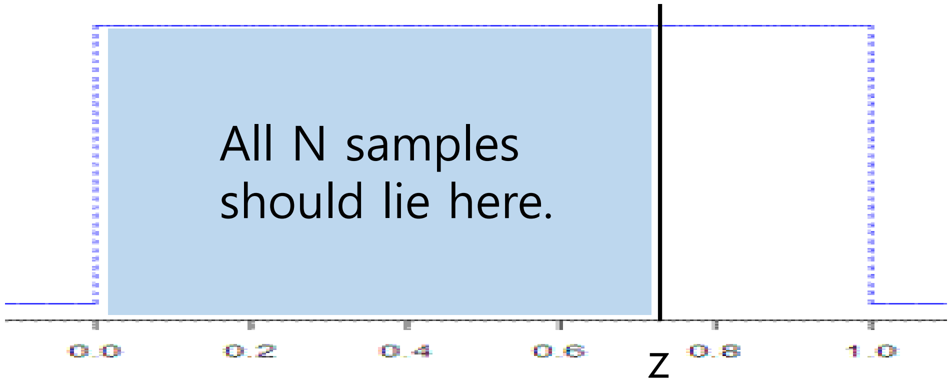
Block Quality (cont.)

- $D_N \sim \max\{r_1, \ldots, r_N : r_i \leftarrow [0,1], i \in [N]\}$
  - Satisfies properties of quality function



- CDF : $F_X(z) = z^N$

- For X $\leftarrow [0,1]$, $X^{1/N}$ follows D_N.



- $D_{N_{\gamma_i}}(\text{hash}(a_i)) := \left(\text{hash}(a_i)/2^L\right)^{1/N}$

X

Chain Quality

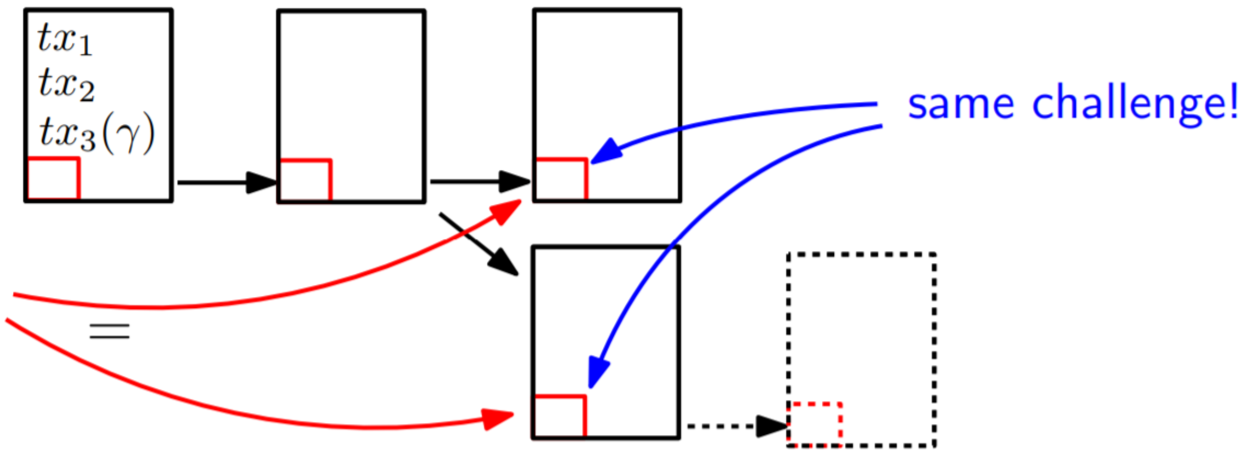
$$\mathcal{N}(v) = \min\{N \in \mathbb{N} : \Pr[v < w \mid w \leftarrow D_N] \geqslant 1/2\}$$

$$\mathsf{QualityPC}(\varphi_0, \ldots, \varphi_i) = \sum_{j=1}^{i} \log(\mathcal{N}(v_j)) \cdot \Lambda^{i-j}$$

- Miner may gossip the quality of the mined block and mined chain, and release the block with the full proof when the quality is competitive enough.

Selecting from Multiple Chains

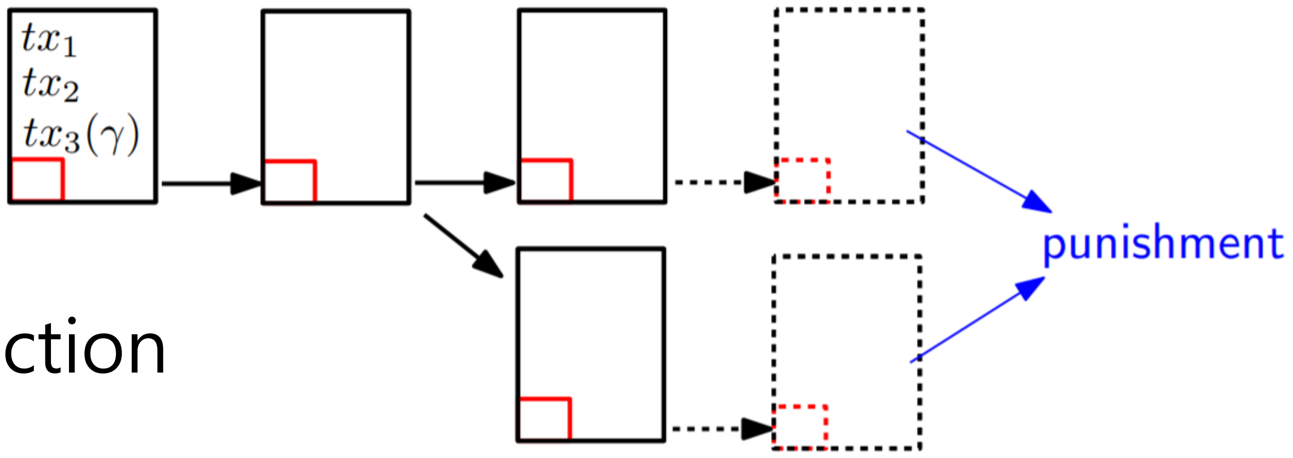
- Mining is easy! (Easy to generate proofs)
- Selecting best block from Multiple Chains
  - Leads to quality inversion
  - Slows down consensus
- Prevention: Derive challenge of block i from block i-Δ.



Multiple Chain Extending

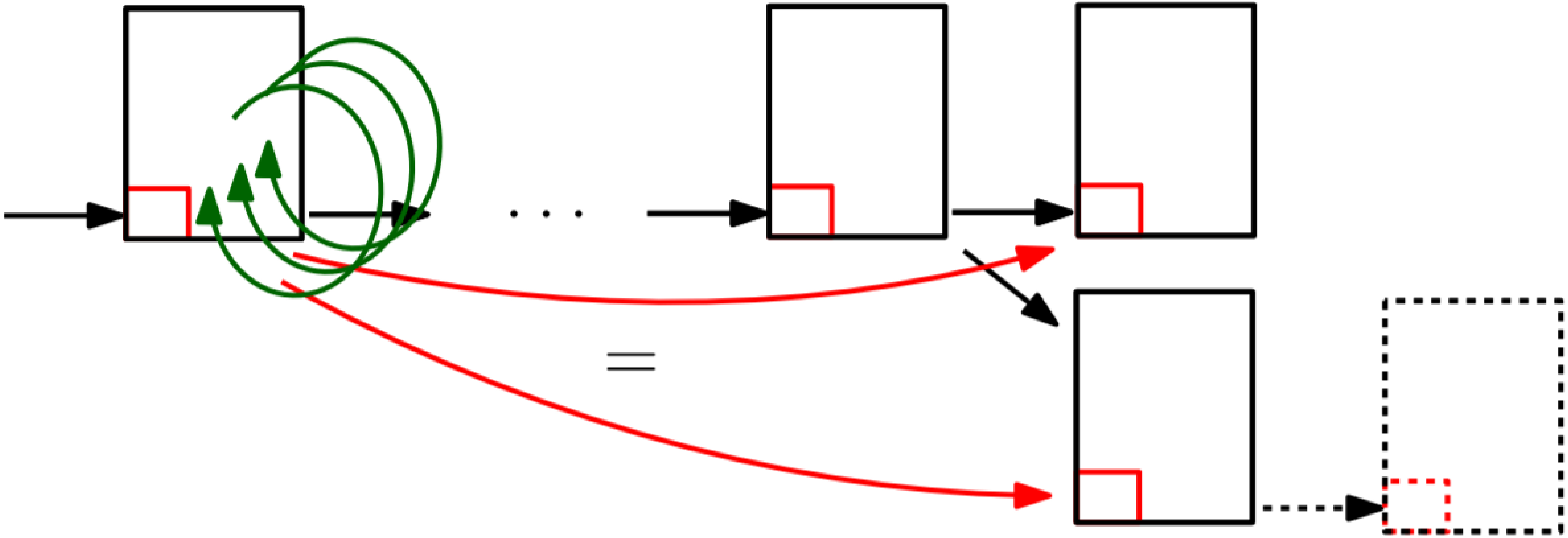
- Mining is easy! (Easy to generate proofs)

- Multiple Chain Extending
  - Best option for a miner against a fork
  - No consensus will be achieved.



- Prevention: 'Penalty' transaction

Block Grinding Attack

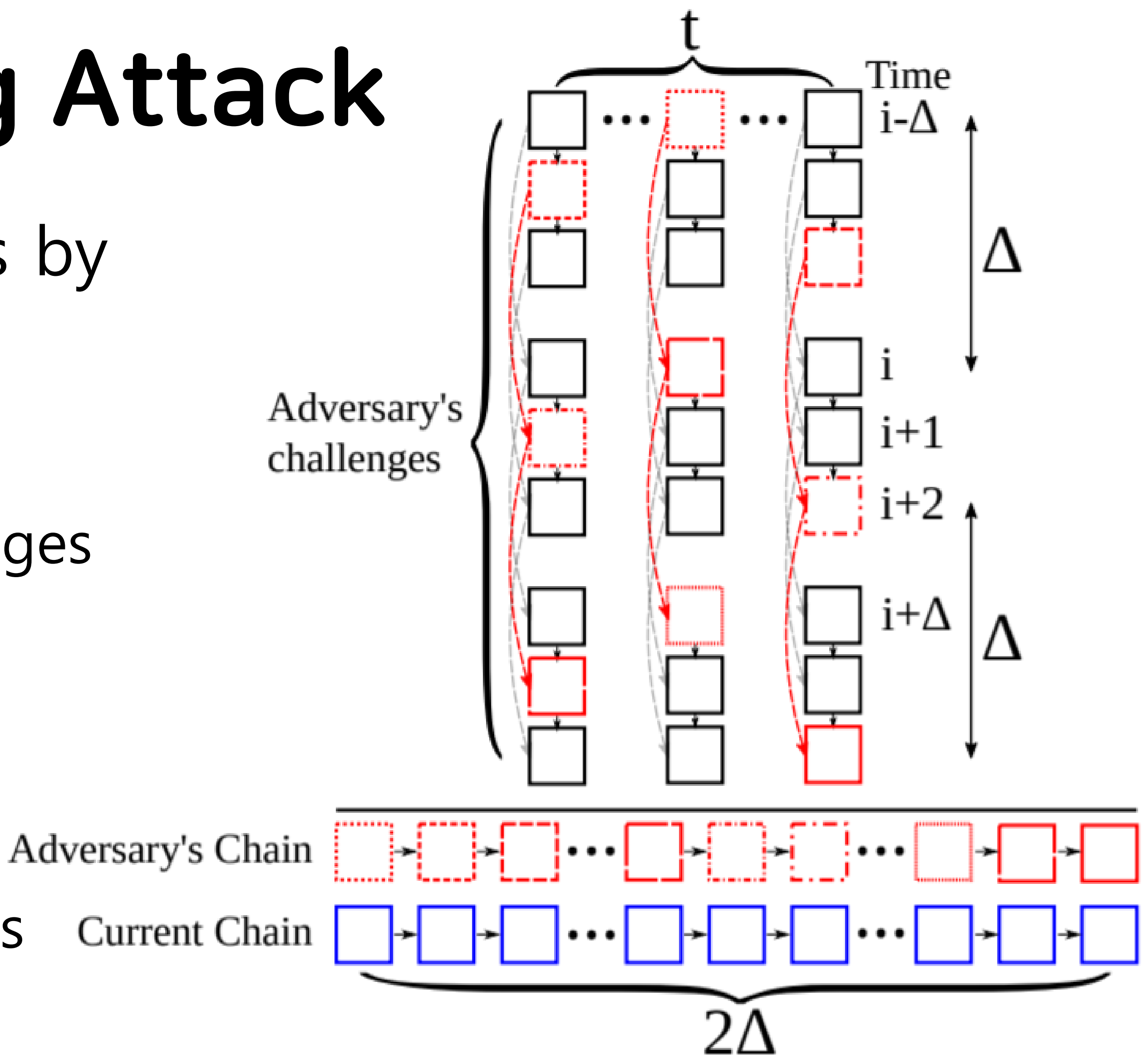


- Prevention: Separate proof chain from transactions

Challenge Grinding Attack

- Make better future challenges by mining multiple bad blocks!
  - Dividing the storage into t fragments to mine t chains
  - Select the best chain of challenges to mine even better blocks!

- Prevention
  - Log-quality function
  - Multiple use of same challenges



51% Attack

- Miner with >50% storage of active miners
- Controls everything
  - Decides which transaction to be included
  - (even prevent including penalty transaction!)
- The paper claims that the attack won't appear due to the drop of cryptocurrency value.

Denial-of-Service Attack

- Rush of fake commitments
  - Still valid transactions, though the commitments cannot be used for actual mining

- Countermeasures
  - Transaction fee for commitment transaction
  - Attaching commitment verification at the commitment transaction

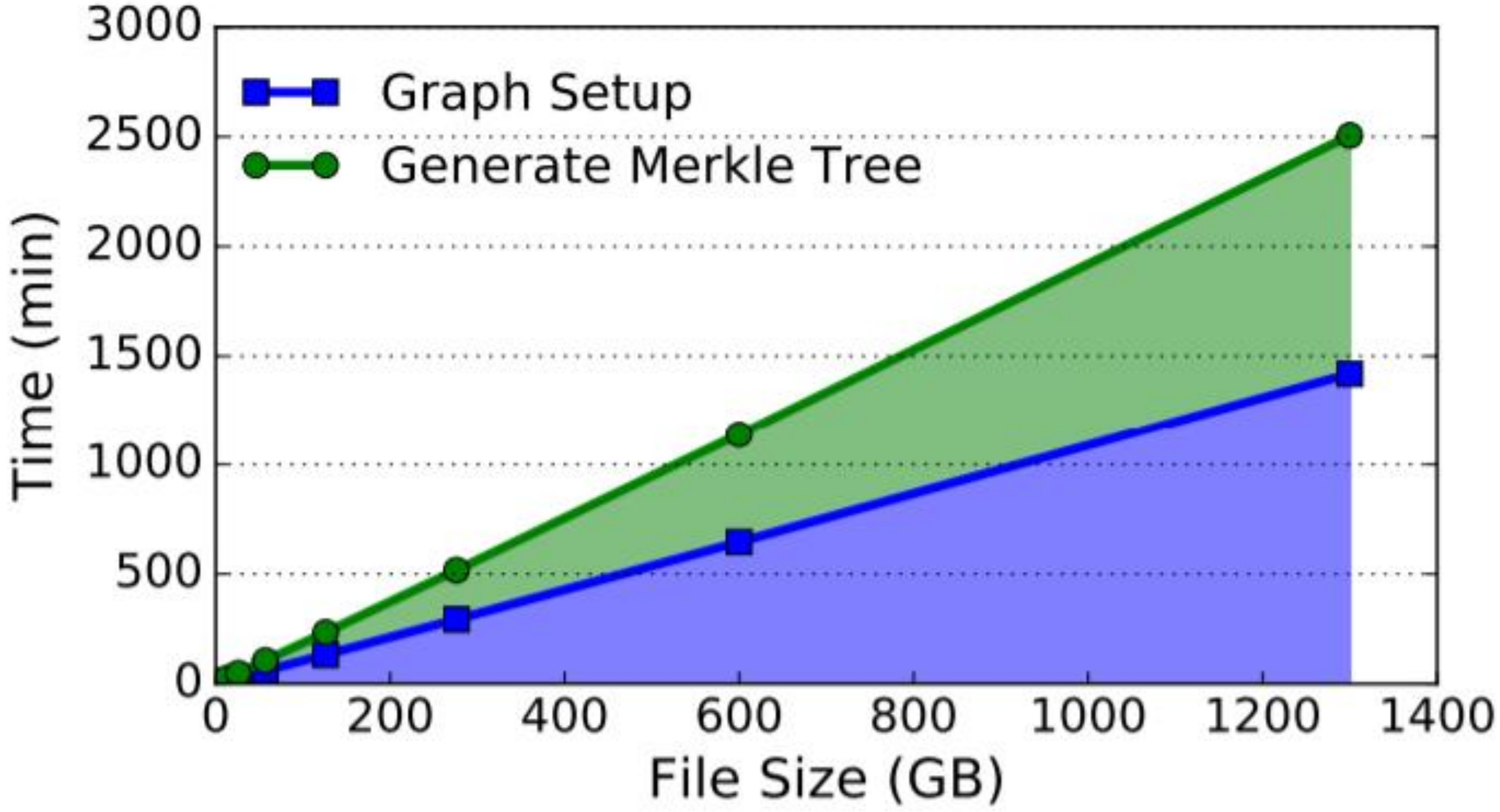
Cheap Storage?

- Mining requires random access.
- Tapes
  - Very cheap, but random access is impossible.
- HDD is the best option, currently.
- The authors expect that SpaceMint would mostly use the idle disk space on personal computers for mining.

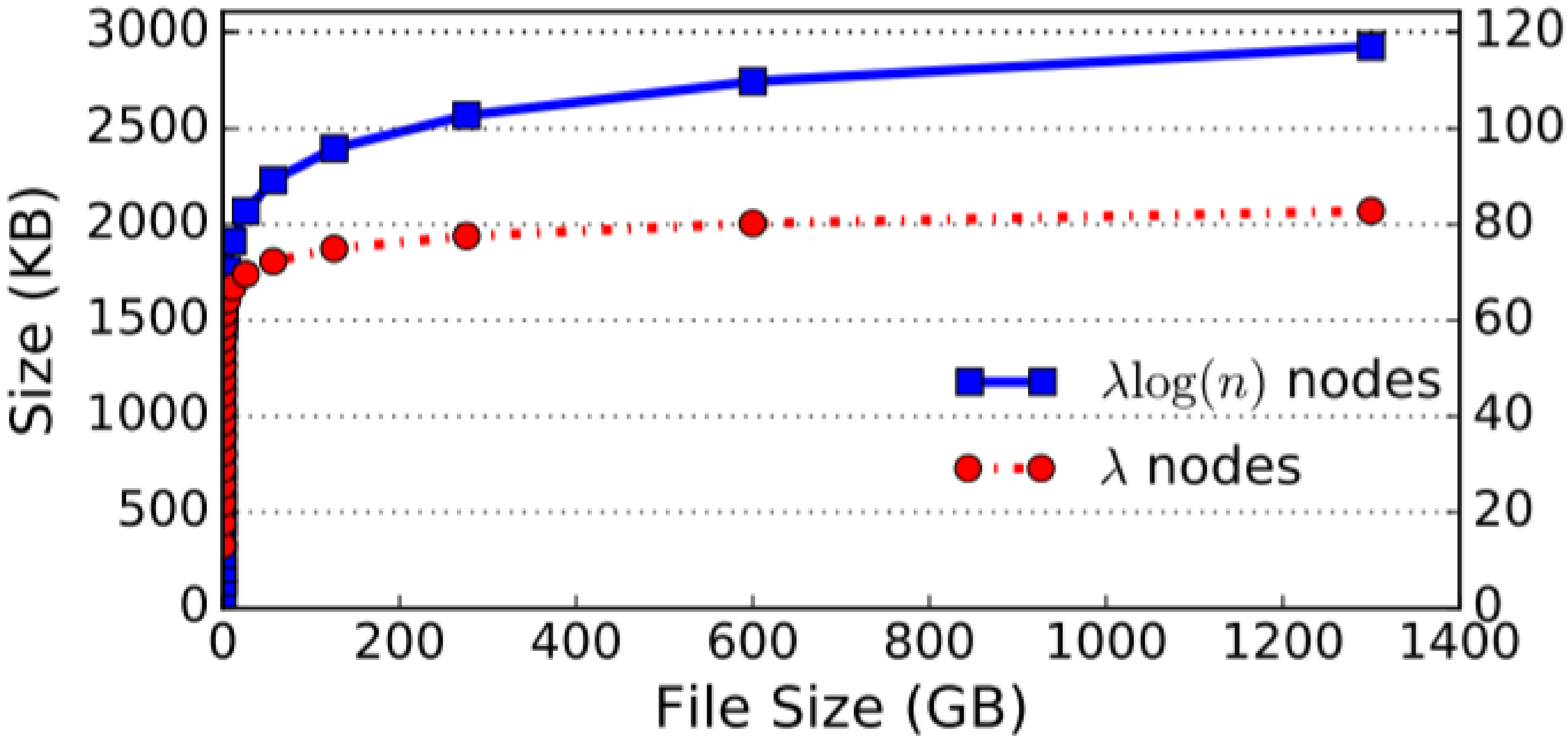
Evaluation Environment

- Software
  - Prototype implementation using Go
  - Graph with pebbling complexity $\Omega(N/\log(N))$
- Hardware
  - CPU: Intel i5-4690K Haswell
  - Memory: 8 GB
  - HDD: 2 TB (cache: 64 MB)

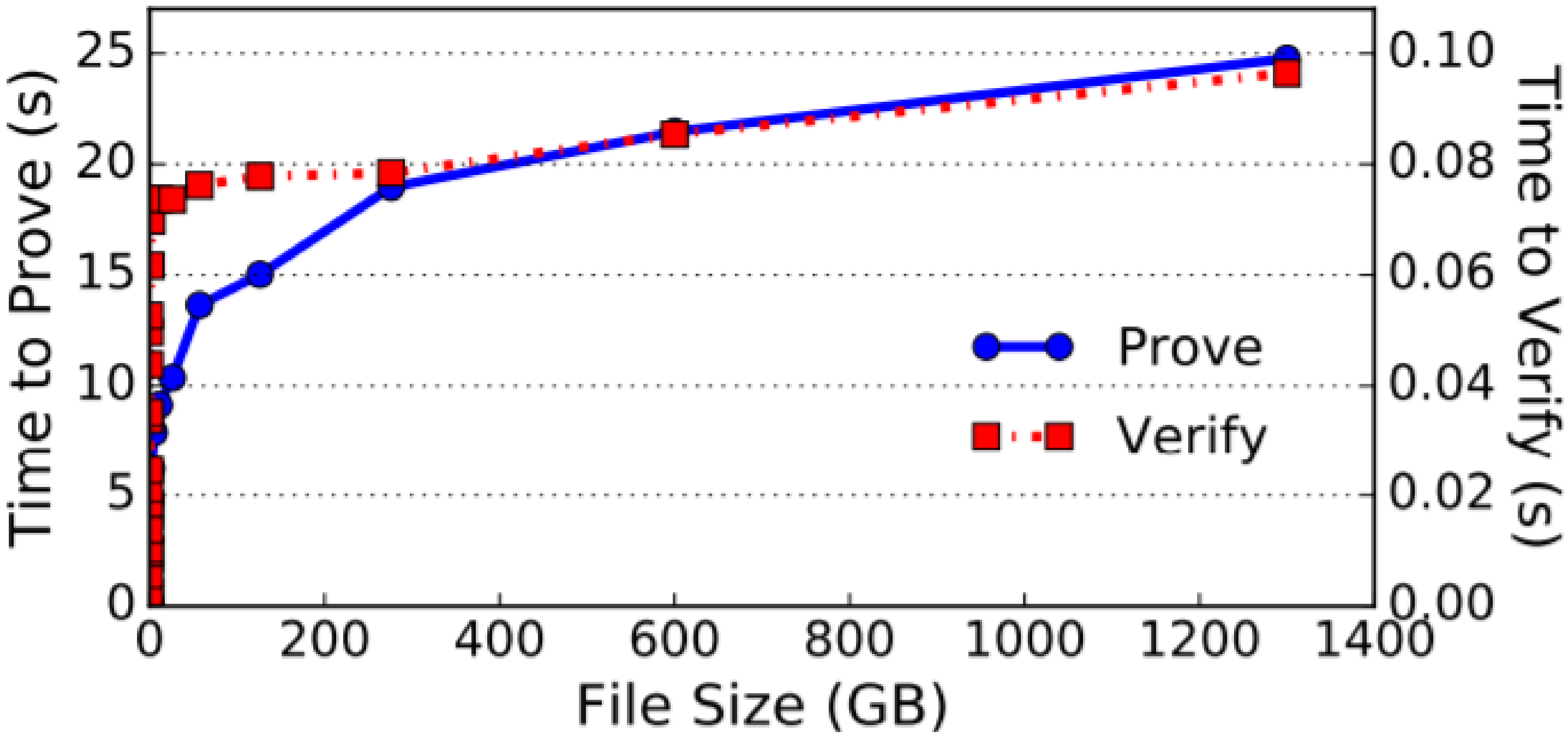
Initialization Performance



Proof Size


3000
2500
2000
1500
1000
500
0
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
Size (KB)
File Size (GB)
λlog(n) nodes
λ nodes

Proof / Verification Time



Energy Estimates

- 100K miners with 1TB each
- 0.01s for checking answer
- 1% of miners generate full answer (20s)
- 10W power consumption

$$10\text{W} \cdot 100\,000 \cdot 0.01s + 10\text{W} \cdot 1000 \cdot 20s = 210\,000\text{J}/\text{block}$$

< 1% of Bitcoin

Game Theoretical Analysis

- Required for analysis against various malicious mining strategies
  - cf) Selfish Mining

Theorem 1. *It is a* sequential equilibrium *of the SpaceMint game (defined in [27, §7]) for all computationally bounded players to adhere to the mining protocol, provided that no player holds more than* 50% *of all space.*

Equilibrium

let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ *be a pure strategy profile of* $\mathsf{SpaceMint}_{\Pi,K,\rho}$.
Then $\vec{\alpha}$ *is an* ε*-Nash equilibrium of* $\mathsf{SpaceMint}_{\Pi,K,\rho}$, *where*

$$\varepsilon = \exp\left(-\frac{1}{2K} \cdot \mathbb{E}\left[\mathsf{diff}_1\right]^2 \cdot \left(\sum_{j=0}^{K-1} \Lambda^{2j}\right)^2\right)$$

- Equilibrium strategy is robust on change of N.
  - If a miner buy more storage, making new commitment and behave like a new honest miner is the best option.

Deciding Confirmation Blocks

Table 2: **Bounding the probability of a successful overtake of the chain:** p is the probability of a successful overtake, ξ is the adversary's proportion of the network disk space, and the tabulated values are fork length (in blocks).

	$\Lambda = 0.99999$					$\Lambda = 0.99998$					$\Lambda = 0.99997$				
$\xi \setminus p$	2^{-8}	2^{-16}	2^{-32}	2^{-64}	2^{-128}	2^{-8}	2^{-16}	2^{-32}	2^{-64}	2^{-128}	2^{-8}	2^{-16}	2^{-32}	2^{-64}	2^{-128}
0.1	3	5	10	19	37	3	5	10	19	37	3	5	10	19	37
0.25	10	19	37	74	148	10	19	37	74	148	10	19	37	74	148
0.33	24	47	93	186	371	24	47	93	186	373	24	47	93	186	374
0.4	68	136	271	543	1092	68	136	272	546	1104	68	136	273	549	1116
0.45	277	554	1114	2254	4614	277	557	1127	2307	4852	278	561	1140	2365	5130

Summary

- This paper…
  - Made non-interactive version of PoSpace.
  - Used PoSpace for Blockchain Consensus.
  - Suggested a prototype, SpaceMint.

- For SpaceMint, the authors…
  - Solved design challenges.
    - Multiple chain extending, block grinding, challenge grinding
  - Evaluated the performance.
  - Had a game theory-based analysis of equilibrium.