Proof of Space from Stacked Expanders

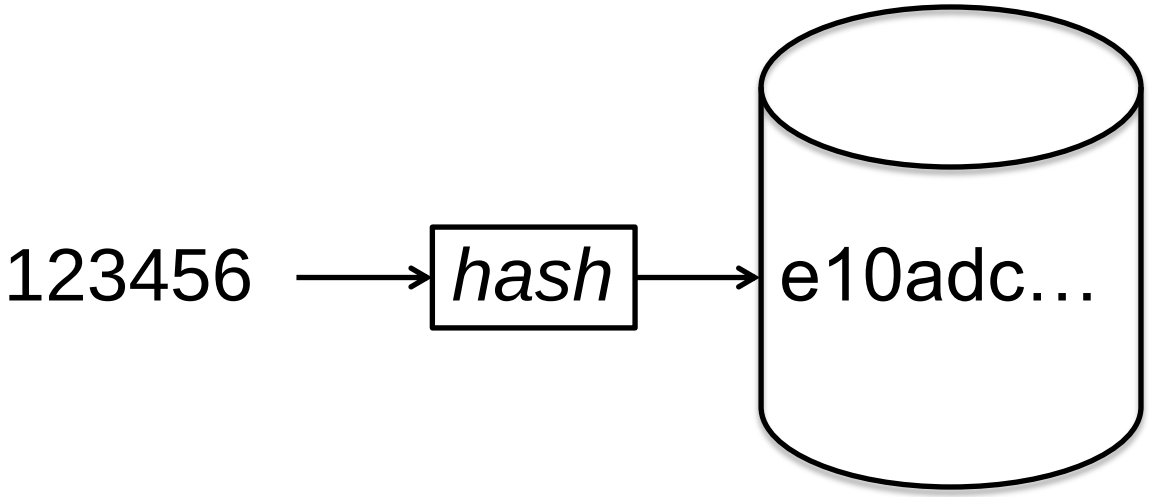
Ling Ren and Srinivas Devadas

TCC - Nov. 2016

Outline

- Why memory hardness
- How to define memory hardness
- Improved analysis & constructions

Password hashing



ASIC Hash Unit

Advertised Capacity:
4.73 Th/s

10 Mh/s for CPU, 400,000x

Power Efficiency:
0.25 W/Gh

10 KW/Gh for CPU, 40,000x

- Want ASIC-resistance hash function
- Memory hard function → ASIC resistance
  - [Abadi et al.'05], [Dwork et al.'03'05], [Percival'09 (scrypt)], [Tromp'14], [Corrigan-Gibbs et al.'16 (Balloon)], [Alwen et al.'15,'16a,'16b'16c]
  - 4 out of top 5 in Password Hashing Competition [Argon2, Catena, Lyra2, yescrypt]

How to Define Memory Hardness?

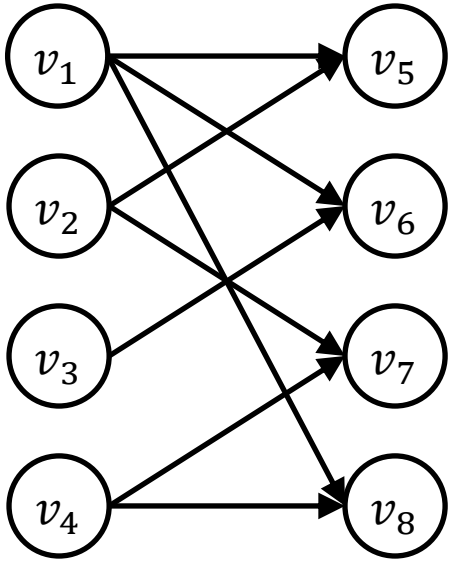
A Natural Definition

- A MHF f(x) is computable with N space
  - Not computable with $S < N$ space [DKW'11]

- Remark: Assume random oracle H

Graph Pebbling

- Evaluate *H* in a DAG
  - "not computable" = "have to guess output of *H*"
  - *T* = # calls to *H*



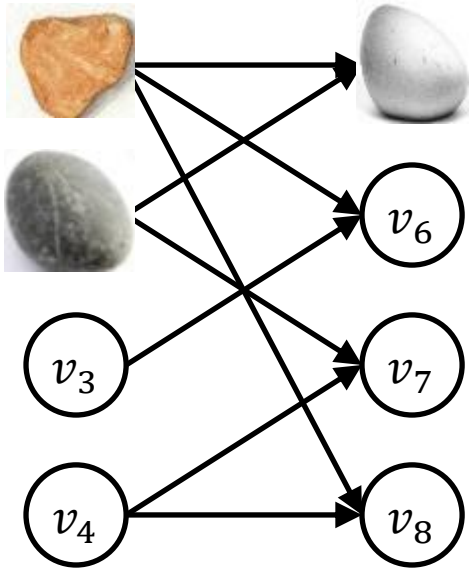
$$v_5 = H(5 \mid v_1 \mid v_2)$$

$$v_6 = H(6 \mid v_1 \mid v_3)$$

……

Graph Pebbling

- Rule: all predecessors must be pebbled
- Model *S*, *T* with pebbles [Dwork et al.'03'05], [DKW'11]



$$v_5 = H(5 \mid v_1 \mid v_2)$$

$$v_6 = H(6 \mid v_1 \mid v_3)$$

……

Relax the Strict Definition

- Computable with N space and T time
  - Not computable with $S < N$ space [DKW'11]
  - Infeasible to compute with $S < cN$ space

- Problem of [DKW'11]: $T = O(N^2)$
- Infeasible = 2^k calls to H
- c: memory-hardness; T: efficiency

Previous and Our Results

Scheme	*T*	*c*
DKW	$O(N^2)$	1
PoSE	$O(N \log^2 N)$	1/32
Catena2	$O(kN \log N)$	1/20
Balloon	$O(kN)$	1/8 → 1

stacked expanders

Other Definitions & Properties

Memory hard at most steps, not just at some step [AS'15]

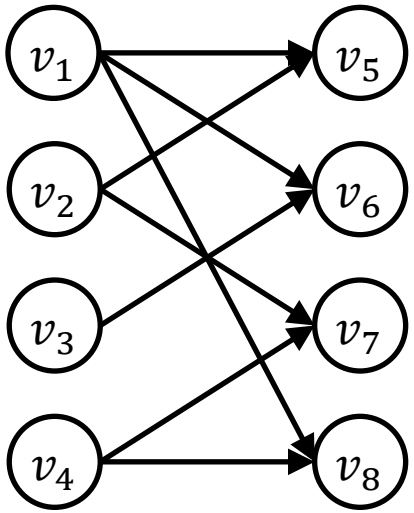
Lower bounding $ST = \Omega(N^2)$

Security against infinite parallelism [AS'15]

Construction from Stacked Expanders

Expander

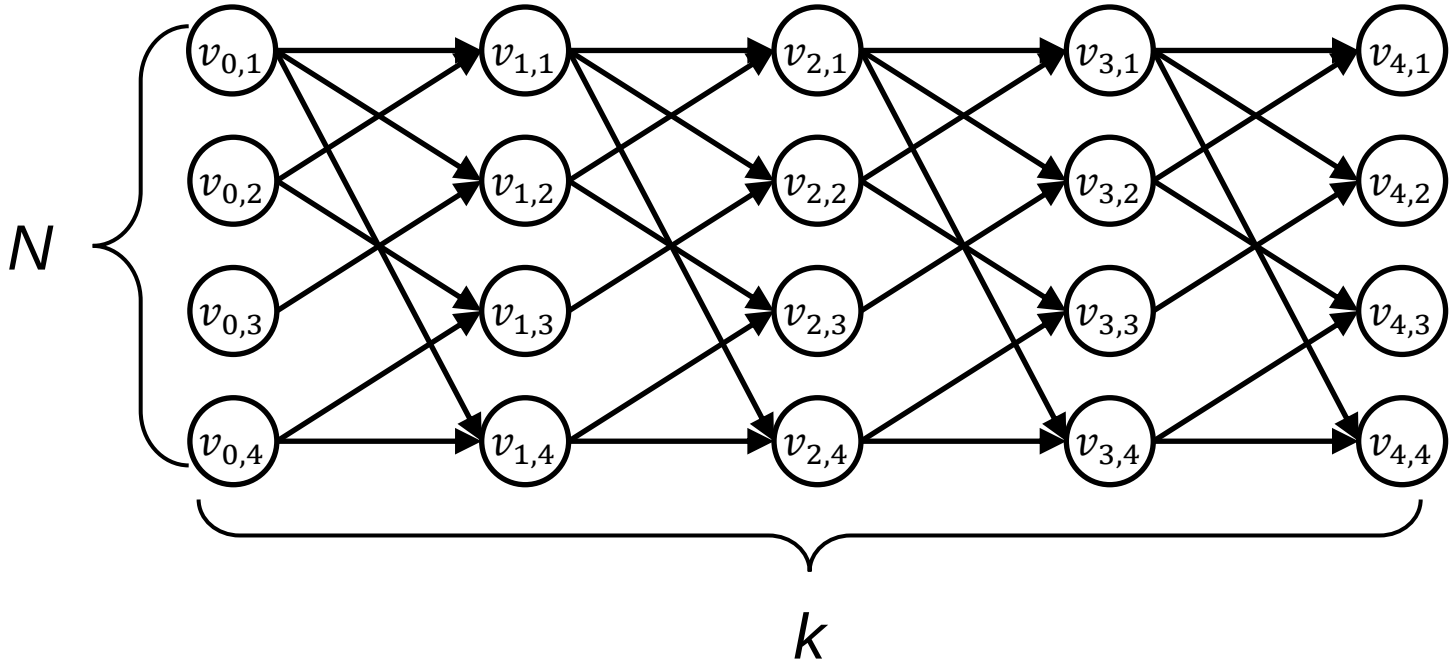
- *(N, α, β)* bipartite expander (*0* < α < β < *1*)
  - *N* sources and *N* sinks
  - A few sinks (αN) ← many sources (βN)



(4, 1/4, 1/2)-expander

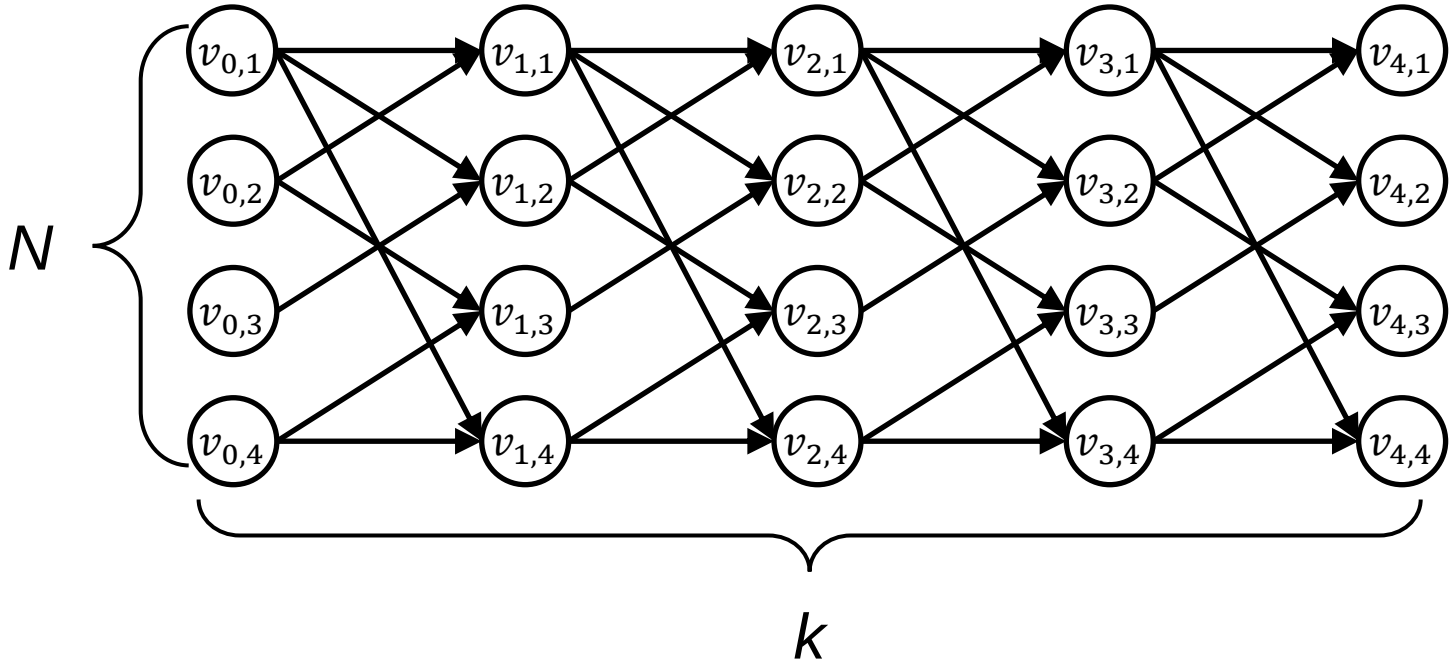
Stacked Expanders

- *k* back-to-back *(N, α, β)*-expanders
- Can be pebbled in *S = 2N, T = kN*
  - [balloon] With modifications *S = N, T = kN*



Stacked Expanders - Lower Bound

- If $S < (\beta - 2\alpha)N$, $T > 2^k$ $c = (\beta - 2\alpha)$ is tight
  - Proof by induction, similar to [Paul & Tarjan'78]
- Hard at most steps




Memory hard function
Memory hard PoW
Proof of space

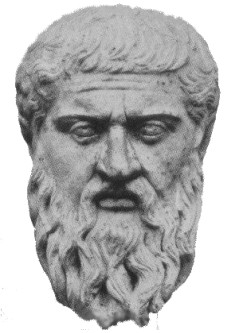
Memory Hard PoW

- ASIC-resistant Proof of work
- MHF + cheap verification: poly(k, log$N$)

Verifier

"I used N space.
Here is a proof"

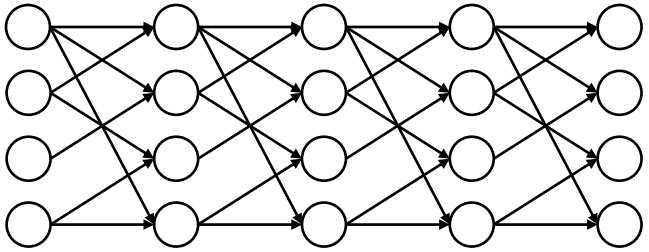
Prover



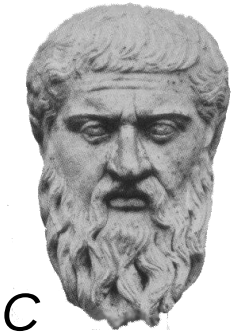
Memory Hard PoW [ABFG'14]

Verifier

input x



Prover



commit all vertices to a Merkle root C

C

Build the Merkle tree incrementally

Check $O(k^2)$ vertices

"Let me check $\{v_i\}$"

Merkle paths for each v_i and its predecessors

Rebuild the Merkle

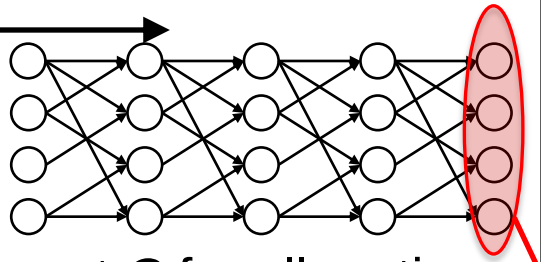
Check each v_i is correctly pebbled


Memory hard function
Memory hard PoW
Proof of space

Proof of Space

Verifier

input x

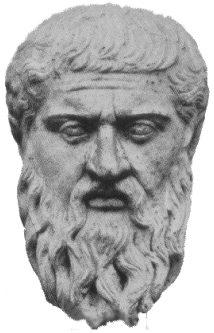


Merkle root *C* for all vertices

Verify(*x*, *C*)

Memory-hard PoW

Prover



Y

sill storing *Y*?

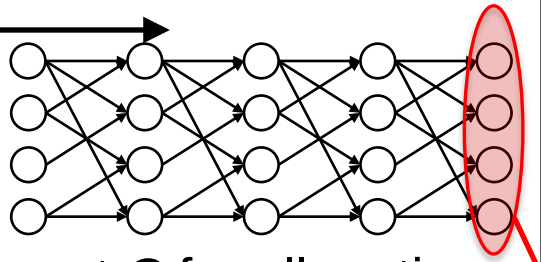
prove yes

If storing *Y*, trivial to answer
Otherwise, hard to answer
Motivation: save energy

Proof of Space

Verifier

input x

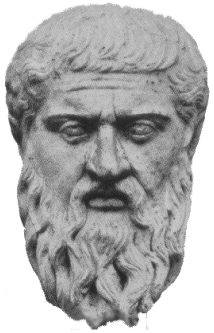


Merkle root *C* for all vertices

Verify(*x*, *C*)

Memory-hard PoW

Prover



Y

Check *O(k)* sinks

"Let me check some sinks $\{v_i\}$"

Merkle paths for $\{v_i\}$

Check $\{v_i\}$ against *C*

Memory-hard PoW

Scheme	Prover time	V space/time	*c*
ABFG	$O(kN \log N)$	$O(k^2 \log^2 N)$	→1/6
This work	***O(kN)***	***O(k² logN)***	**→ 1**

Proof of space

Scheme	P time	V space/time	*c*
DFKP	0	$O(k \log N)$	$\rightarrow \frac{1}{2\times256\times28\times\log N}$
This work	**0**	***O(k logN)***	**→ 1/2**

Summary

- More efficient, and/or
- Stronger memory hardness
  - No big gap, hard at most steps

Thank you!