区块链技术与应用

Chapter 8 其他挖矿算法

苏 明

Puzzle requirements

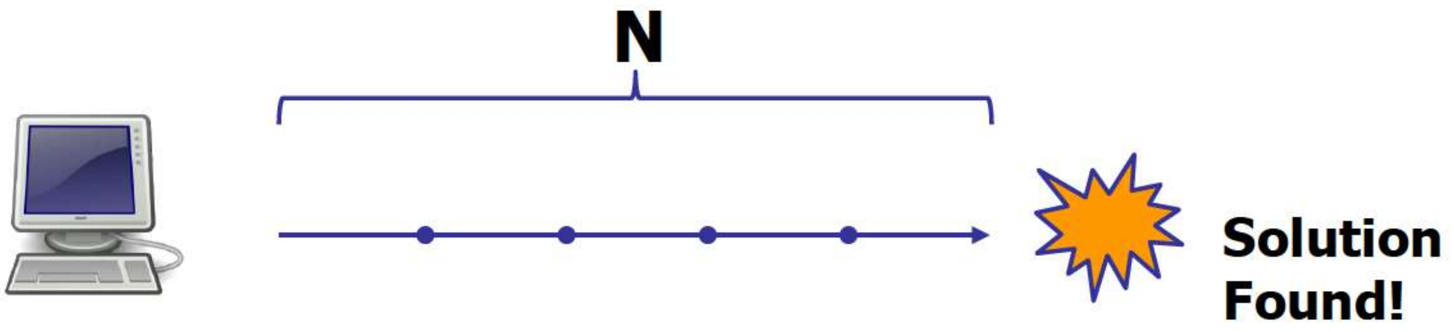
- Cheap to Verify
- Adjustable difficulty

...

- Chance of winning is proportional to hashpower
    - Large players get only proportional advantage
    - Even small players get proportional compensation

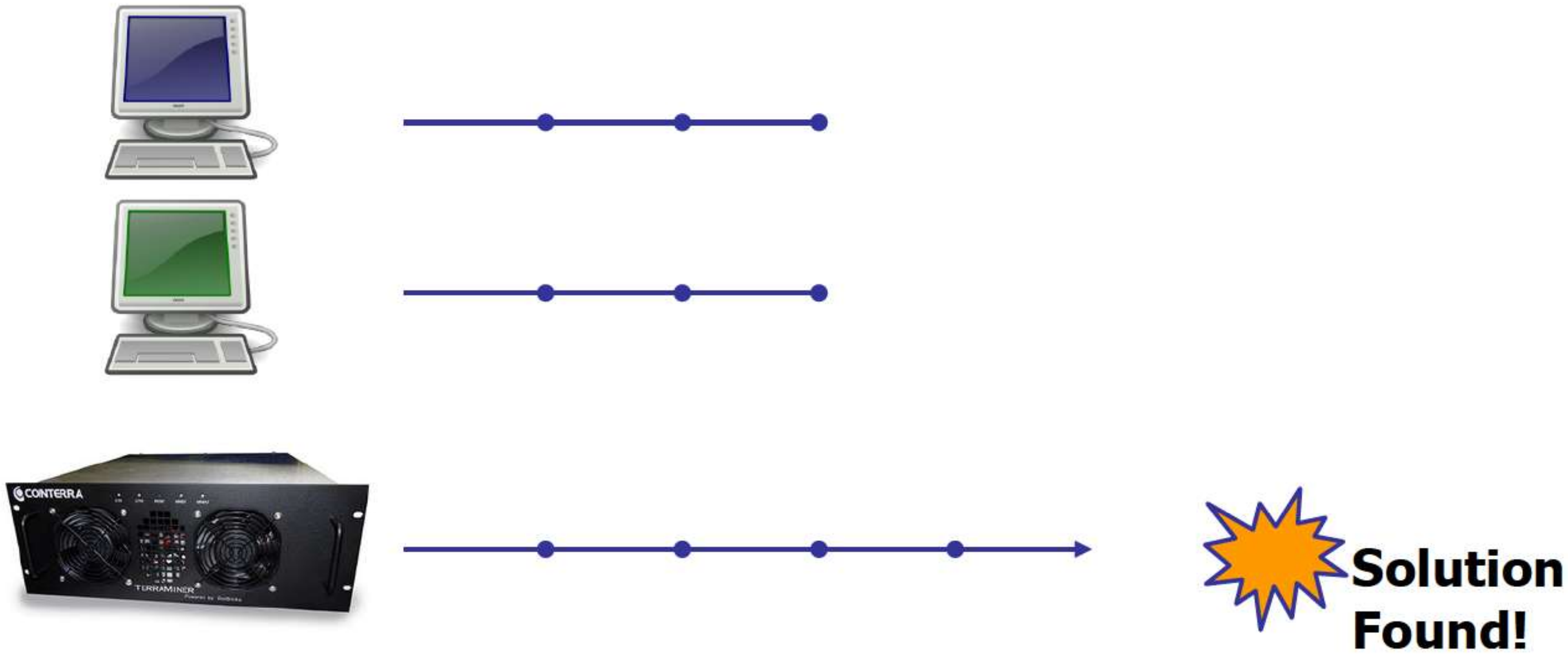
Bad puzzle: a sequential puzzle

Consider a puzzle that takes N steps to solve
a “Sequential” Proof of Work

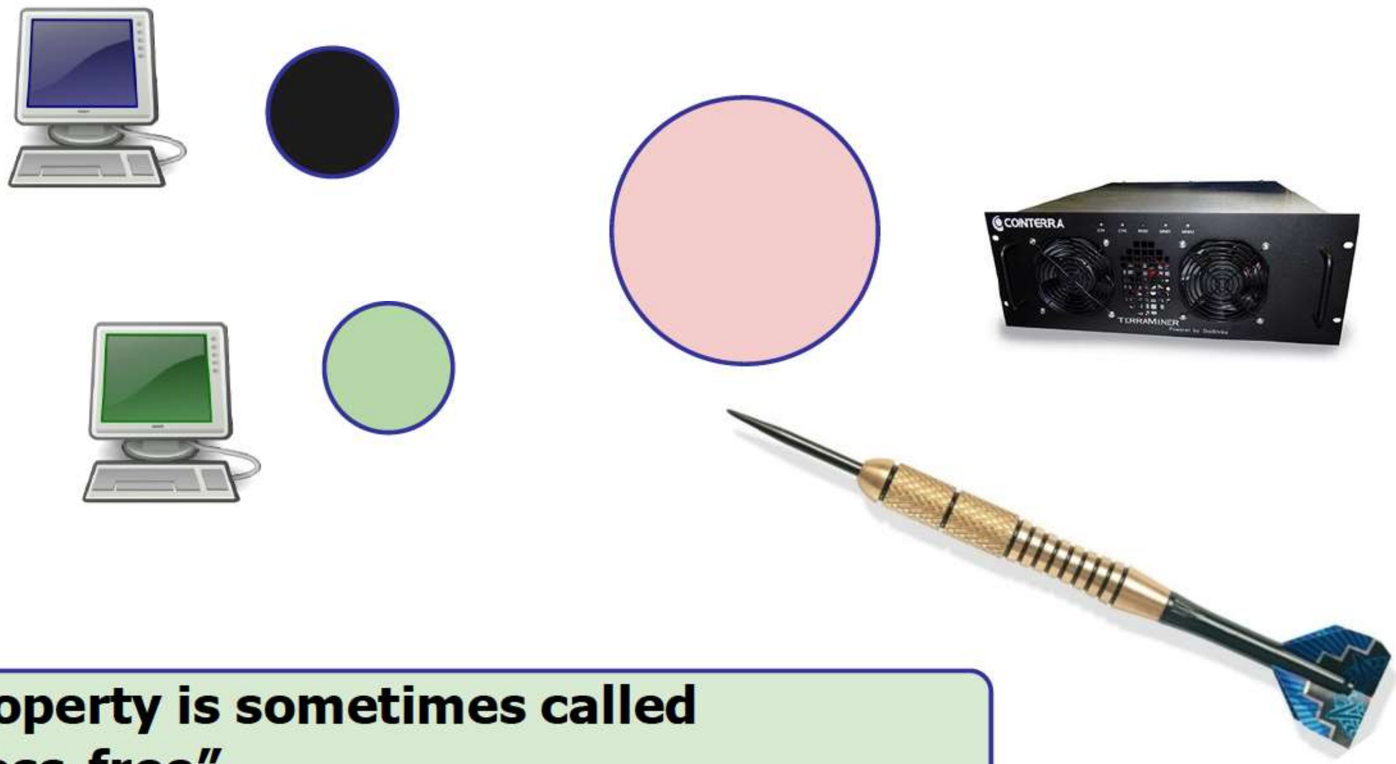


Bad puzzle: a sequential puzzle

Problem: fastest miner **always** wins the race!



Good puzzle → Weighted sample



This property is sometimes called "progress-free"

ASIC resistance - Why? (1 of 2)

Goal: Ordinary people with idle laptops, PCs, or even mobile phones can mine!

Lower barrier to entry

Approach: reduce the gap between custom hardware and general purpose equipment

ASIC resistance - Why? (2 of 2)

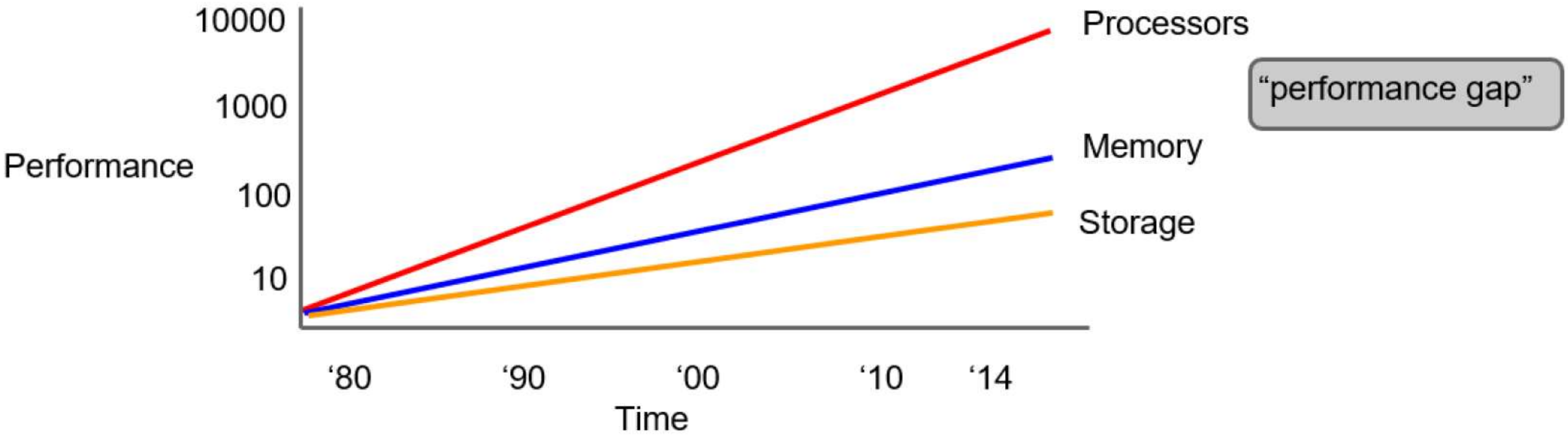
Goal: Prevent large manufacturers from dominating the game

- "Burn-in" advantage
- In-house designs

Approach: reduce the "gap" between future hardware and the custom ASICs we already have

Memory hard puzzles

Premise: the cost and performance of memory is more stable than for processors



scrypt

Colin Percival, 2009

- Memory hard hash function

  Constant time/memory tradeoff

- Most widely used alternative Bitcoin puzzle
- Also used elsewhere in security (PW-hashing)

1. Fill memory with random values
2. Read from the memory in random order

scrypt - step 1 of 2 (write)

Input: $\mathbf{x}$

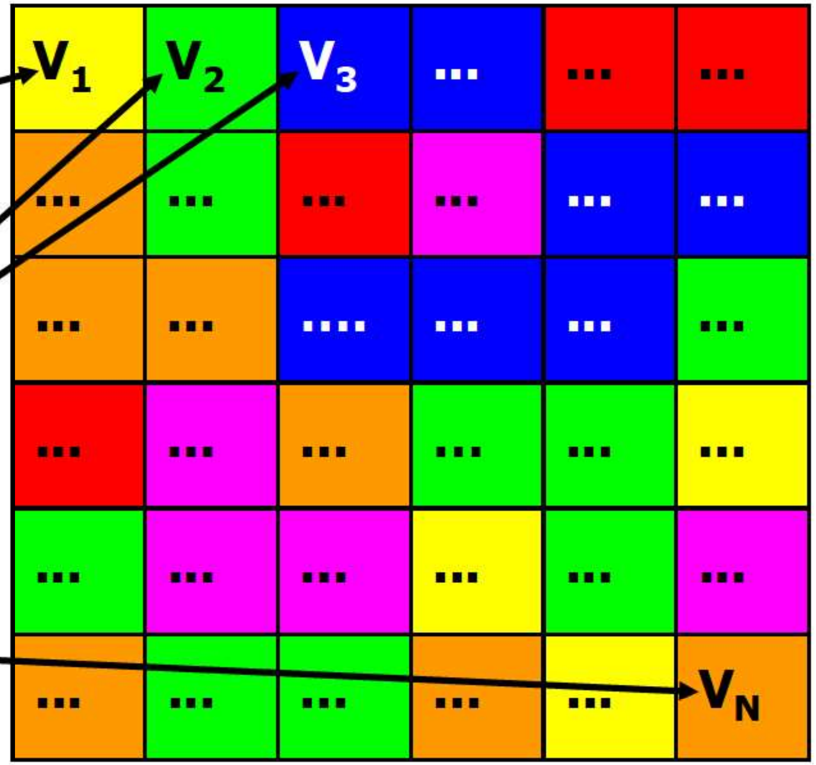
$V_1 = H(X)$

$V_2 = H(V_1) = H(H(X))$

$V_3 = H(V_2) = H^3(X)$

…

$V_N = H^N(x)$



scrypt - step 2 of 2 (read)

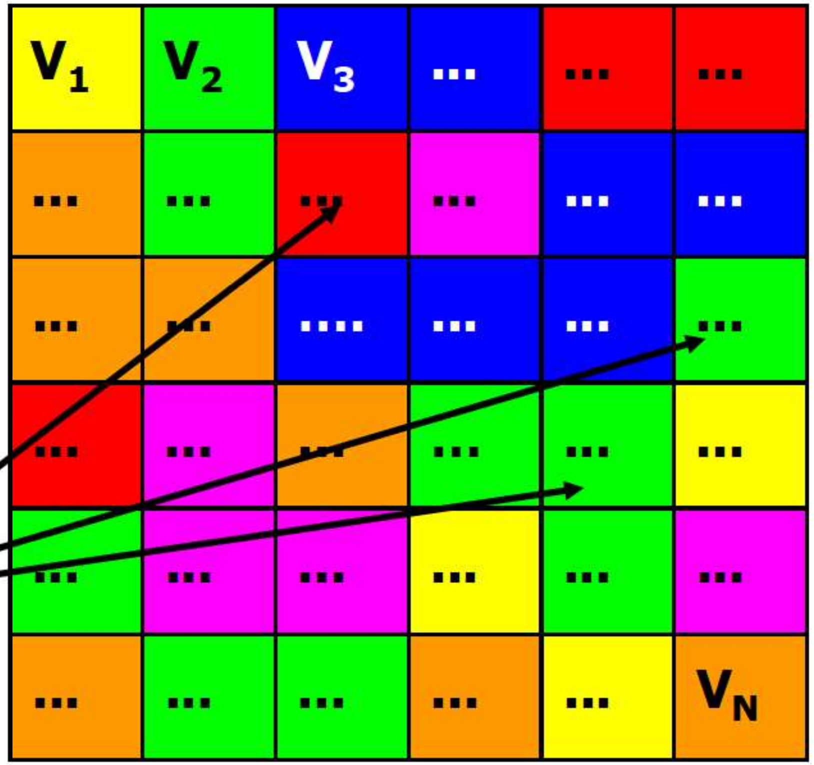
Input: X

`A := H^{N+1}(X)`

For N iterations:

```
    i := A    mod N
    A := H(A xor V_i)
```

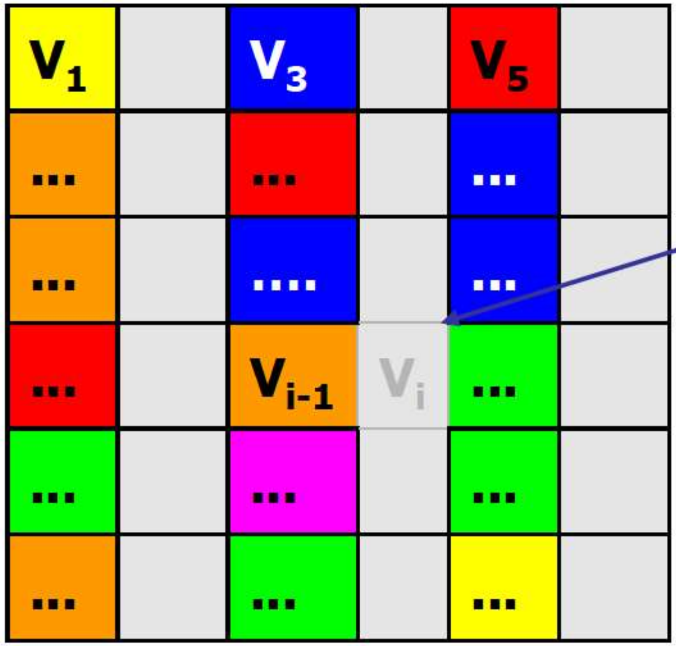
Output: A



scrypt - time/memory tradeoff

Why is this memory-hard?

Reduce memory by half, 1.5x the # steps



Need to access V_i where i is even?

Access V_{i-1}

Compute $V_i = H(V_{i-1})$

scrypt

Disadvantages:

Also requires N steps, N memory to check

Is it actually ASIC resistant?

scrypt ASICs are already available

http://zeusminer.com/

Future: PW-hashing research

Cuckoo hash cycles

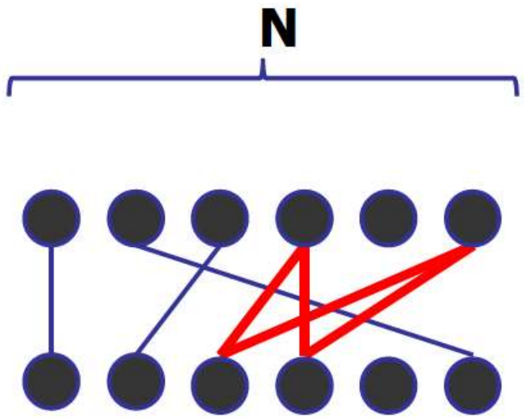
John Tromp, 2014

Memory hard puzzle that's cheap to verify

Input: X

For i = 1 to E:

```
    a := H0(X + i)
b := i + H1(X + i)
edge(a mod N, b mod N)
```



Is there a cycle of size K? If so, Output: X, K edges

Even more approaches

- More complicated hash functions

  X11: 11 different hash functions combined

- Moving target

  Change the puzzle periodically

Recovering wasted work

Recall:

between 150 MW - 900 MW power **(as of mid-2014)** consumed

Natural question:

Can we recycle this and do something useful?

Primecoin

Sunny King, 2013

Puzzle based on finding large prime numbers

Cunningham chain:

$p_1, p_2, \ldots p_n$ where $p_i = 2\,p_{i-1} + 1$

Each p_i is a large (probable) prime

For instance: p_1 is divisible by `H(prev || mrkl_root || nonce)`

Primecoin

- **Many of the largest known Cunningham chains have come from Primecoin miners**
- Hard problem? Studied by others (e.g., PrimeGrid)
- Usefulness? Maybe - at least one known use

Permacoin - Mining with storage

Bitcoin

Permacoin

Miller et al., 2014

Side effect:

Massively distributed, replicated storage system

Permacoin

Assume we have a large file **F** to store

For simplicity: **F** is chosen globally, at the beginning, by a trusted dealer

Each user stores a *random subset* of the file

Storage-based puzzle

1. Build a Merkle tree, where each leaf is a segment of the file

2. Generate a public signing key pk, which determines a random subset of file segments

F_1 F_2 F_4 F_5

3. Each mining attempt:

F_2 F_4

```
a) Select a random nonce
b) h1 := H(prev || mrkl_root || PK || nonce)
c) h1 selects k segments from subset
d) h2 := H(prev || mrkl_root || PK || nonce || F)
e) Winner if  h2 < TARGET
```