Equihash: Asymmetric Proof-of-Work based on the Generalized Birthday Problem

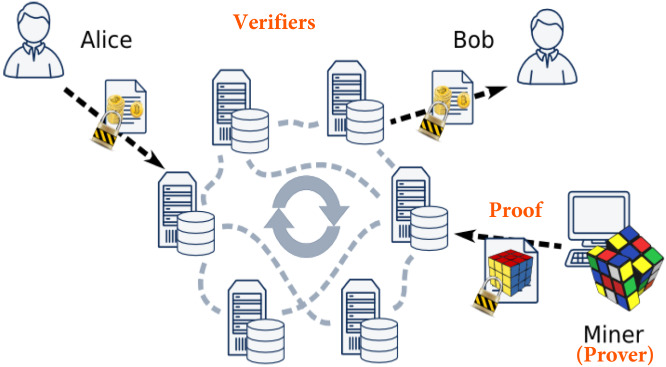
Alex Biryukov Dmitry Khovratovich

University of Luxembourg

February 22nd, 2016

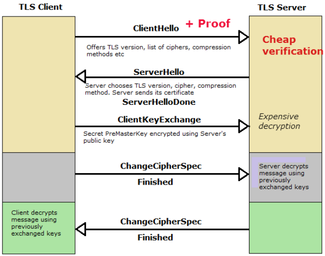
Proof of Work in cryptocurrencies

PoW – certificate of certain amount of work. In cryptocurrencies:



- Verifier – cryptocurrency users;
- Prover – cryptocurrency miner.

In TLS client puzzles:



- Verifier – server that establishes a secure connection;
- Prover – client that may want to DoS the server with signature computation.

Clearly, the proof search

Asymmetric verification

Clearly, the proof search

must be more expensive than verification

Asymmetric verification

HashCash/Bitcoin Proof-of-Work with hash function H:

$$S \text{ -- proof, if } H(S) = \underbrace{00\ldots0}_{q \text{ zeros}}.$$

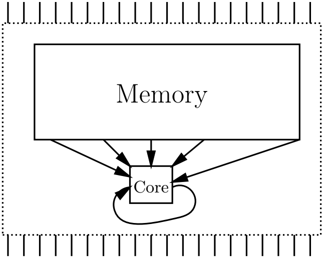
2^q calls to H for prover, 1 call for verifier.

But here come ASICs..

Regular cryptographich hash H is 30,000 less expensive on ASIC due to small custom chip.

Since 2003, *memory-intensive* computations have been proposed.

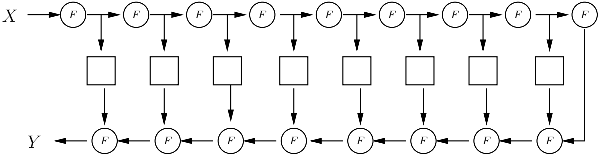
Computing with a lot of memory would require a very large and expensive chip.



With large memory on-chip, the ASIC advantage vanishes.

Hash function with two iterations over memory of size N.

- $V_i = F(V_{i-1})$;
- $V'_N = V_N$;
- $V'_i = F(V'_{i+1}||V_i)$.

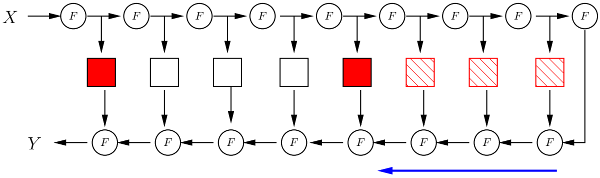


Trivial tradeoff

Compute the hash using $\frac{N}{m} + m$ memory units and $3N$ calls to F (instead of $2N$):

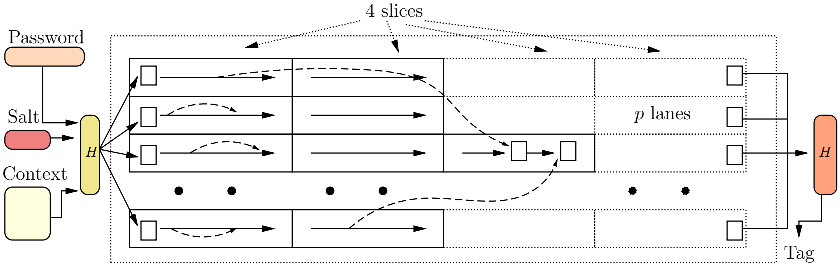
- Store every m-th block;
- When entering a new interval, precompute its m inputs.

Optimal point is $m = \sqrt{N}$.



Approach 2. Argon2

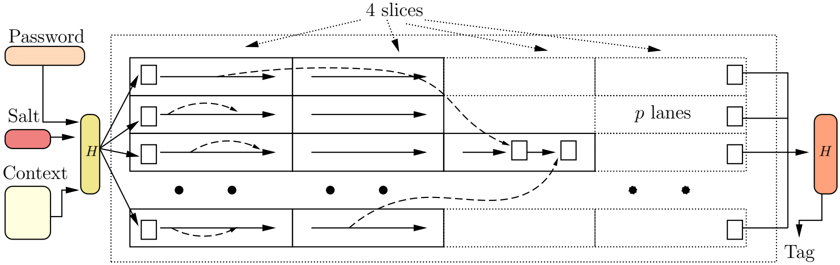
Memory-hard hashing function, that won Password Hashing Competition in 2015:



- Simple randomized-graph design with high-penalty tradeoffs.

Approach 2. Argon2

Memory-hard hashing function, that won Password Hashing Competition in 2015:



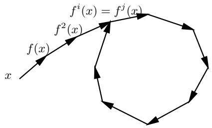
- Simple randomized-graph design with high-penalty tradeoffs.
- However, no easy verification.

Approach 3. Collision search

1. Verifier sends seed S;
2. Prover generates 2^k $2k$-bit hashes $H(S||1), H(S||2), \dots, H(S||2^k)$.
3. Prover shows a collision $H(S||i) = H(S||j)$. Short and efficient.

1. Verifier sends seed S;
2. Prover generates 2^k $2k$-bit hashes $H(S||1), H(S||2), \ldots, H(S||2^k)$.
3. Prover shows a collision $H(S||i) = H(S||j)$. Short and efficient.

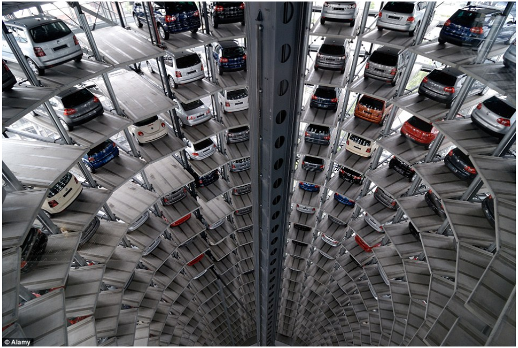
Problem: the ρ-based collision search finds collisions in the same 2^k time but no memory.



Original: given 2^k lists L_j of n-bit strings $\{X_i\}$, find distinct $\{X_{i_j} \in L_j\}$ such that

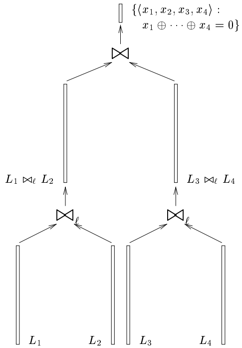
$$X_{i_1} \oplus X_{i_2} \oplus \cdots \oplus X_{i_{2^k}} = 0.$$

Solution is found by iterative sorting



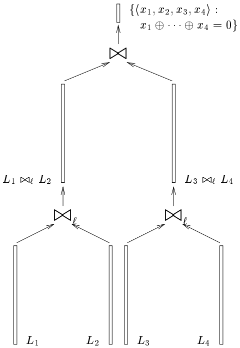
$O(2^{\frac{n}{k+1}})$ time and memory

- Sort by first $\frac{n}{k+1}$ bits;
- Store XOR of collisions;
- Repeat for next $\frac{n}{k+1}$ bits, etc.



$O(2^{\frac{n}{k+1}})$ time and memory

- Sort by first $\frac{n}{k+1}$ bits;
- Store XOR of collisions;
- Repeat for next $\frac{n}{k+1}$ bits, etc.



Problem: not amortization-free: it is easy to modify the algorithm to get many solutions quickly:

- Collide on other bits;
- Not collision but XOR to some constant.

After all, qM memory yields q^{k+1} solutions in time qT.

Interestingly, the solution reveals how it was found:

$$\underbrace{\underbrace{H(x_1) \oplus H(x_2)}_{\text{equal in } \frac{n}{k+1} \text{ bits}} \oplus \underbrace{H(x_3) \oplus H(x_4)}_{\text{equal in } \frac{n}{k+1} \text{ bits}}}_{\text{equal in } \frac{2n}{k+1} \text{ bits}} \cdots \oplus H(x_{2^k}) = 0.$$

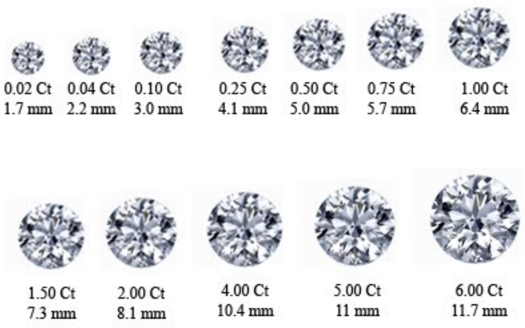
We then strongly require such pattern and disallow other solutions.

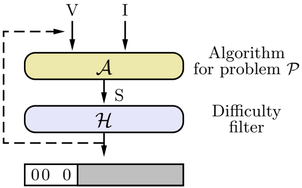
Amortization is impossible then.

To avoid centralization, there must be always a chance to find a solution (Poisson process).

$$\mathcal{P} : \mathcal{R} \times \mathcal{I} \times \mathcal{S} \to \{\text{true}, \text{false}\}.$$

How to increase expected solving time and make the probability non-zero at the beginning?





Difficulty filter: S is valid if $P(R, I, S) = true$ and $H(S)$ has q leading zeros.

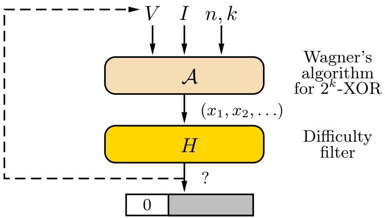
Problem composition takes the best properties from each component.

EQUIHASH: given seed I, find V and $\{x_j\}$ such that

$$H(I||V||x_1) \oplus H(I||V||x_2) \oplus \cdots \oplus H(I||V||x_{2^k}) = 0. \tag{1}$$

$$H(I||V||x_1||x_2||\cdots||x_{2^k}) = \underbrace{00\ldots0}_{q \text{ zeroes}} * * * * \,. \tag{2}$$

$$\underbrace{\underbrace{H(x_1) \oplus H(x_2)}_{\text{equal in } \frac{n}{k+1} \text{ bits}} \oplus \underbrace{H(x_3) \oplus H(x_4)}_{\text{equal in } \frac{n}{k+1} \text{ bits}}}_{\text{equal in } \frac{2n}{k+1} \text{ bits}} \cdots \oplus H(x_{2^k}) = 0. \tag{3}$$



Time penalty for reducing memory by the factor of q:

$$C_2(q) \approx 2^k q^{k/2} k^{k/2-1} = O(q^{k/2}).$$

Tunable steepness.

Memoryless computation: run recursive memoryless collision search for expanding functions (a bit worse):

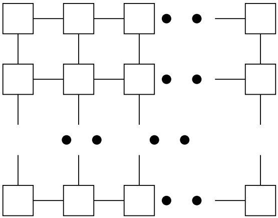
$$2^{\frac{n}{2}+2k+\frac{n}{k+1}}.$$

Using p processors, we can get p-factor speed-up in time.

On GPU and FPGA this leads to increased memory bandwidth (factor p), which becomes bottleneck.

Custom ASIC sorting

The only possible solution is mesh-based sorting with one core per memory block on custom ASIC, but this is expensive (10x larger chip).



		Complexity			
		Memory-full		Memoryless	
n	k	Peak memory	Time	Time	Solution size
96	5	2.5 MB	$2^{19.2}$	2^{74}	88 B
128	7	8.5 MB	2^{20}	2^{94}	292 B
160	9	32.5 MB	$2^{20.3}$	2^{114}	1.1 KB
176	10	64.5 MB	$2^{20.4}$	2^{124}	2.2 KB
192	11	128.5 MB	$2^{20.5}$	2^{134}	4.4 KB
96	3	320 MB	2^{27}	2^{78}	45 B
144	5	704 MB	$2^{27.5}$	2^{106}	120 B
192	7	2.2 GB	2^{28}	2^{134}	420 B
240	9	8.3 GB	$2^{28.2}$	2^{162}	1.6 KB
96	2	82 GB	$2^{34.5}$	2^{84}	37 B
288	8	11 TB	2^{36}	2^{192}	1.1 KB

Questions?