Client puzzles for TLS

Alex Biryukov, Dmitry Khovratovich

University of Luxembourg

Also Erik Nygren, Samuel Erb (Akamai)

July 19th, 2016

- TLS server performs expensive cryptographic operations at the first stage of handshake;
- A malicious client can initiate a connection and drop it after the initial message;
- Existing countermeasure: rate limit.

Problem of rate limiting: does not distinguish good from bad clients behind NAT or Proxy; well-behaved clients are more penalized than bad ones.

Complementary idea:

- Insert a puzzle into HelloRetryRequest;
- Client solves a puzzle and returns a solution;
- Verification should be fast.

Minor problem: clients with limited resources (e.g., IoT) will suffer.

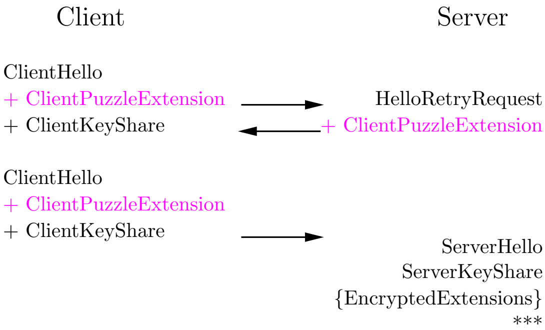
Clients not supporting puzzles can be rate limited.

Client		Server
ClientHello + ClientPuzzleExtension + ClientKeyShare	→	
	←	HelloRetryRequest + ClientPuzzleExtension
ClientHello + ClientPuzzleExtension + ClientKeyShare	→	
		ServerHello ServerKeyShare {EncryptedExtensions} ***

ClientPuzzleExtension

The only required extension is multi-purpose:

- Client indicates of supported puzzle types;
- Server sends puzzle type and input;
- Client sends a solution (shorter than 64 KB).



Types of puzzles

1. Cookie: client just echoes back the token;
2. SHA-256: server sends S, client finds N such that SHA-256(N, S) has certain number of initial zero bits;
3. SHA-512: similar.
4. Equihash: memory-hard puzzle with fast verification.

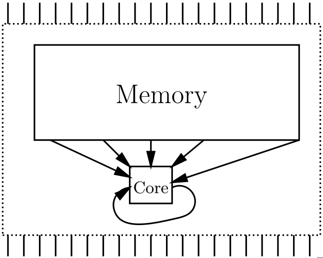
SHA puzzles are well known from the Bitcoin cryptocurrency. Equihash was developed by Biryukov and Khovratovich [NDSS'2016] and is employed as a Proof-of-Work in the anonymity-enhanced currency Zcash.

Brute-force attacks are most efficient on custom hardware: multiple computing cores on large ASICs:

- 2^{32} hashes/joule on ASIC;
- 2^{17} hashes/joule on laptop.

ASICs have high entry costs, but FPGA and GPU are employed too.

Memory-intensive computations have been as remedy, as computing with a lot of memory would require a very large and expensive chip, the ASIC advantage vanishes.



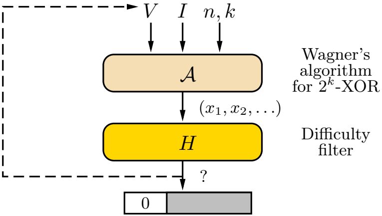
Equihash

Given seed I, find V and $\{x_j\}$ such that

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

where H is Blake2b, and

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



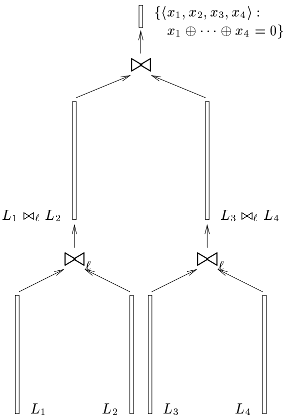
Solved by Wagner's algorithm. Verification: 2^k hashes.

Wagner's algorithm

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

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

Memory reduction by q gives $O(q^{k/2})$ time penalty.



Public implementation available:
`https://github.com/khovratovich/equihash`

Suggested parameters

		Complexity		
n	k	Peak memory	Time	Solution size
96	5	2.5 MB	0.25 sec	88 B
102	5	5 MB	$<$ 0.5 sec	92 B
114	5	20 MB	$<$ 2 sec	100 B
70	4	2 MB	0.1 sec	54 B
80	4	8 MB	0.2 sec	58 B
90	4	32 MB	1 sec	62 B
100	4	128 MB	4 sec	66 B
96	3	320 MB	10 sec	45 B
144	5	704 MB	15 sec	120 B

Time on single-thread 2.1 GHz CPU.