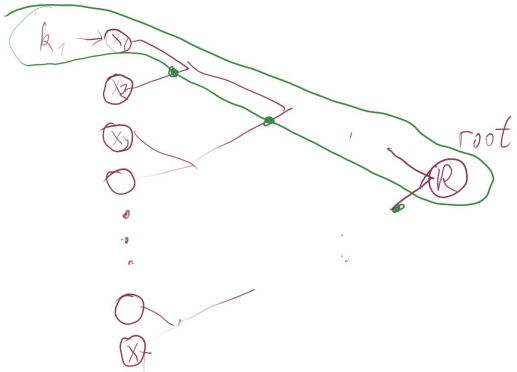
Security of ZK Friendly Hash Functions

Dmitry Khovratovich
Ethereum Foundation
Cryptographic Frontier 2021

ZK-Hash Functions in Ethereum ecosystem

Merkle Trees as accumulators

1. Parties add entries (e.g. public keys) $X_1, X_2, ..., X_T$ to the tree
2. To prove eligibility, P_i proves that he knows a secret key k_j to some X_j in the tree.
3. To ensure one-timeness, present and prove *nullifier* -- $H_2(k_j)$

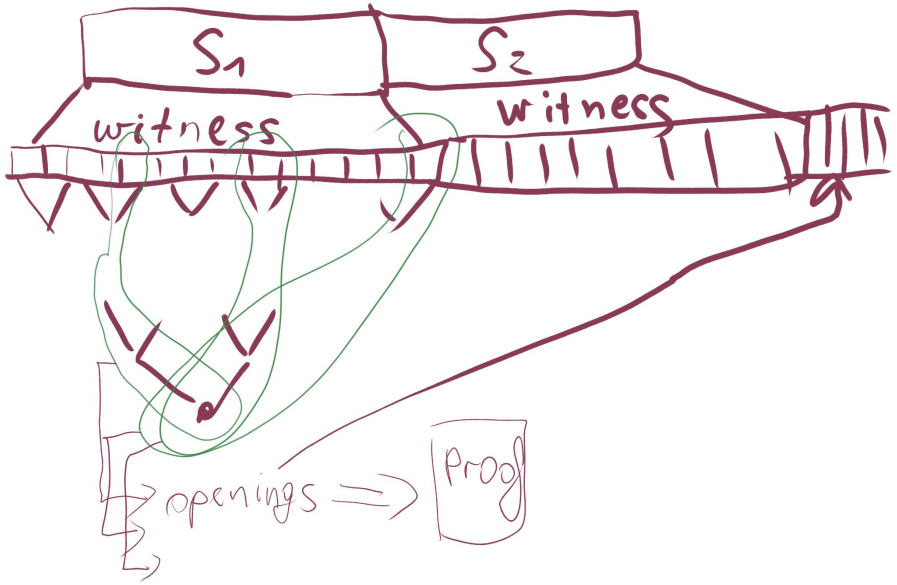


ZK-Hash Functions in Ethereum ecosystem

ZK proof systems with recursive composition (Fractal, Halo, etc.)

How a composite statement (S1-S2-...St) is proven

1. P1 constructs witness W1, encodes it as a vector, and put to a Merkle tree T.
2. As part of the proof, tree T is opened at several positions X1,...,Xk.
3. P2 proves that he saw a proof from P1, i.e. knows k openings



What makes a good ZK hash function

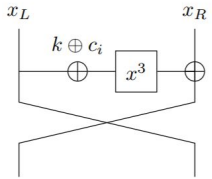
1. Performance in native computation
2. Performance in circuits for zero knowledge
3. Security

Circuit complexity

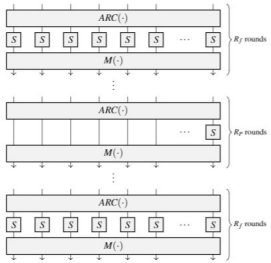
1. Circuits in popular ZK proof systems are arithmetic (ADD and MUL) over F_p. Recent addition -- lookups.
2. Circuit size X is ~ total number of [additions and] multiplications and lookups.
3. Prover time is O (X log X)
4. Verifier is constant or $O(\log^k X)$

Existing designs

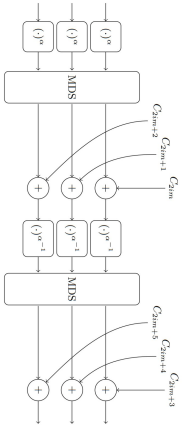
MIMC



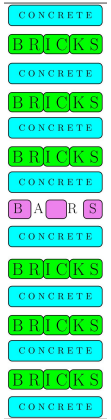
Poseidon



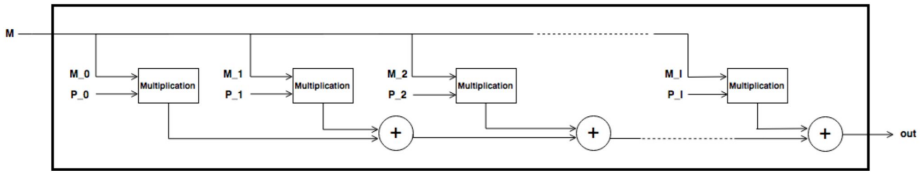
Rescue Prime



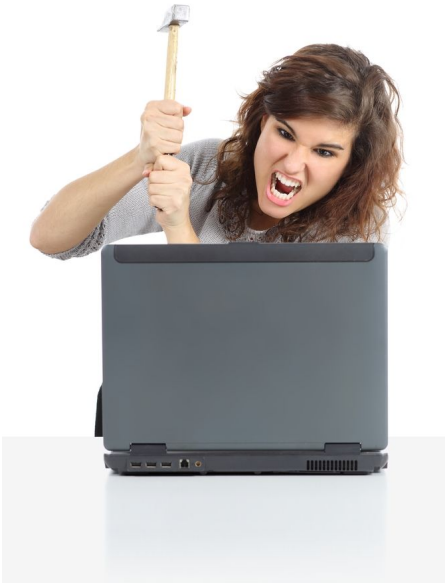
Reinforced Concrete



Pedersen hash



Analysis



1. To make circuit small, almost all designs are described by low-degree polynomials.
2. Number of rounds chosen high (R=dozens) to make the overall degree $>2^{128}$ and Groebner basis attacks invalid.
3. Value R is far beyond what needed to protect from traditional statistical attacks. It also make functions much slower.
4. Estimates for algebraic attacks complexity are very imprecise.

Performance

Function	Circuit size (512 to 256 bits compression)	Merkle tree proof time (16 layers)	Tree construction time (2^{16})
Blake2	21000 (2000 with LU)	30 sec	10 sec
SHA-256	27500 (3000 with LU)	45 sec	20 sec
MIMC	1326	5 sec	40 min
ReinforcedConcrete	267 with LU	1 sec	1 min
RescuePrime	252	0.6 sec	6 hrs
Poseidon	243	0.6 sec	20 min
Pedersen hash	869	2 sec	2 hrs

Open Problems

1. Create a hash function:
    a. Fast as Blake2
    b. Small in circuit.
2. What is the real complexity of Groebner attacks?
3. Can we reduce number of rounds for existing designs?
4. Algebraic attacks we are not aware of.

Bounties

1. Attacks on reduced-round versions of MIMC, Poseidon, Rescue, Sinsemilla, RC hash
2. Groebner basis attack complexity.
3. Best paper awards.
4. Details soon.