Groth16 is not dead

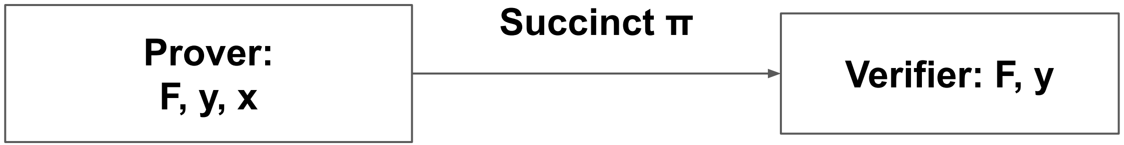
Exploring the tradeoff space of zero knowledge proof systems

Georgios Konstantopoulos
Independent Consultant & ~~Researcher~~ Engineer
Twitter: @gakonst / me@gakonst.com
Slides available: gakonst.com/zksummit2019.pdf

note to cryptographers:
i’m not a cryptographer,
please excuse loose notation

SNARK

I know x: f(x)= y

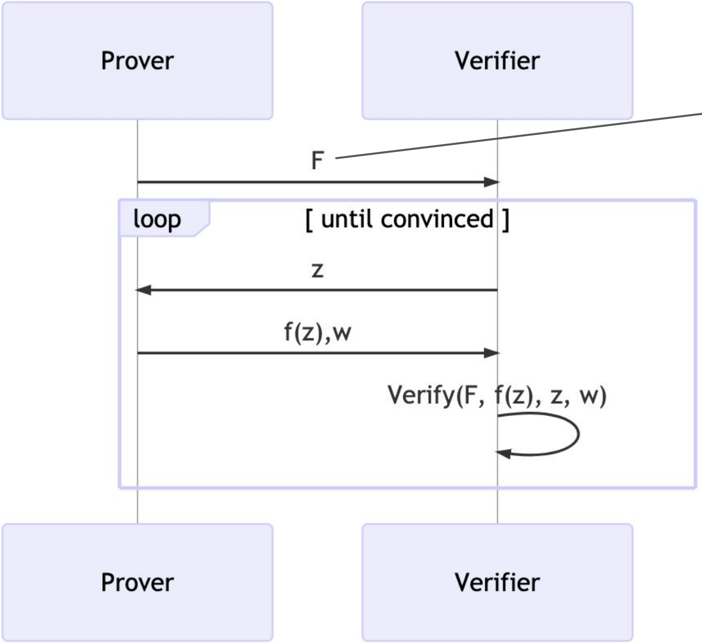


Toolbox

- Pairing-based cryptography
- Accumulators (for committing to data)
- Knowledge of Exponent (fundamentally required)
- Standard assumptions / Generic Group Model / Algebraic Group Model
- Recursion? BLS12-377, MNT, …?
- 2-adic pairing-friendly curve for efficient verifiers (eg BN254, BLS12-381)
- Instantiation: public parameters (random)
    - Public coin
    - MPC
    - ~~Trusted~~
- ...

Statement →
Information theoretic protocol →
SNARK

CS Proofs / PCPs / IOPs / Linear IOPs, oh my



Commitment to f(X):

- Polynomial commitment
- Merkle tree
- …?

Non-interactive via Fiat-Shamir in ROM

Loop iters = 1: PCP, else IOP

SNARKs

$$\pi \leftarrow P(R, x)$$

$$\{1, 0\} \leftarrow V(R, \pi)$$

Preprocessing snarks

$\pi \leftarrow P(\mathbf{PK}, R, x)$

$\{1, 0\} \leftarrow V(\mathbf{VK}, w)$

Evaluate circuit at random points
→ (PK, VK) → encode in CRS

Preprocessing snarks

$\pi \leftarrow P(\mathbf{PK}, R, x)$

$\{1, 0\} \leftarrow V(\mathbf{VK}, w)$

Evaluate circuit at random points

→ (PK, VK) → encode in CRS

trusted or expensive MPC :(

What’s deployed today?

State of the art: Groth16

arith circuits	CRS Size	Proof Size	Prover Time (exp)	Verification Time
PHGR13/ BCTV14	linear circuit size	7 G1, 1 G2	7G1, 1 G2	12 pairings
Gro16		2 G1, 1 G2	2 G1, 1 G2	3 pairings

Universal setup

1. Emulate CPU inside SNARK (TinyRAM / vnTinyRAM):

Universal setup

1. Emulate CPU inside SNARK (TinyRAM / vnTinyRAM):
2. Unstructured CRS → Circuit-specific SRS
    a. BGM17: O(N) CRS, *not updateable*
    designed for Groth16, was used in Sapling
    b. GKMMM18: O(N^2) CRS - impractical

universal & updateable & practical?

Polynomial Commitments & IOPs

$$f(X) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_0$$

- $F \leftarrow \text{Commit}(f(X), \ldots)$
- $(f(z), w) \leftarrow \text{Open}(f(X), z, \ldots)$
- $\{0, 1\} \leftarrow \text{Verify}(F, w, z)$

Constant size PC schemes require a setup

$$f(X) = a_n x^n + a_{n-1} x^{n-1} + \dots a_0$$

- $F \leftarrow \text{Commit}(\mathbf{pp}, f(X), \dots)$
- $(f(z), w) \leftarrow \text{Open}(\mathbf{pp}, f(X), z, \dots)$
- $\{0, 1\} \leftarrow \text{Verify}(\mathbf{pp}, F, w, z)$

Which polynomial commitment scheme?

- KZG10 + variants → setup, but O(1) openings
- DARK: Groups of unknown order
  - RSA: Fast → setup
  - UFO → too big / no setup
  - Class Groups → slow (?) / no setup
- FRI- w/ merkle trees: no setup

Which polynomial commitment scheme?

- KZG10 + variants → setup, but O(1) openings
- DARK: Groups of unknown order
  - RSA: Fast → setup
  - UFO → too big / no setup
  - Class Groups → slow (?) / no setup
- FRI- w/ merkle trees: no setup (STARK-like efficiency)

Sonic / PLONK / Marlin $\xrightarrow{\text{PC scheme}}$ **SNARK**

A DARK caveat

Scheme	Transp.	$\lvert \mathsf{pp} \rvert$	Prover	Verifier	$\lvert \pi \rvert$
DARK *(this work)*	yes	$O(1)$	$O(d^\mu \mu \log(d))$ EXP	$2\mu \log(d)$ EXP	$2\mu \log(d)\ \mathbb{G}_U$
Based on Pairings	no	$d^\mu\ \mathbb{G}_B$	$O(d^\mu)$ EXP	μ Pairing	$\mu\ \mathbb{G}_B$
[BCC+16b, $\sqrt{\cdot}$]	yes	$\sqrt{d^\mu} \mathbb{G}_P$	$O(d^\mu)$ EXP	$O(\sqrt{d^\mu})$EXP	$O(\sqrt{d^\mu})\ \mathbb{G}_P$
Bulletproofs	yes	$2d^\mu \mathbb{G}_P$	$O(d^\mu)$ EXP	$O(d^\mu)$EXP	$2\mu \log(d)\ \mathbb{G}_P$
FRI ($\mu = 1$)	yes	$O(1)$	$O(\lambda d)$ H	$O(\lambda \log^2(d))$ H	$O(\lambda \log^2(d))$ H

Table 2: Comparison table between different polynomial commitment schemes for an μ-variate polynomial of degree d.

A DARK caveat

Class group benchmarks: https://github.com/cambrian/accumulator/pull/35)

If using RSA group then it's fast but it's not transparent :/

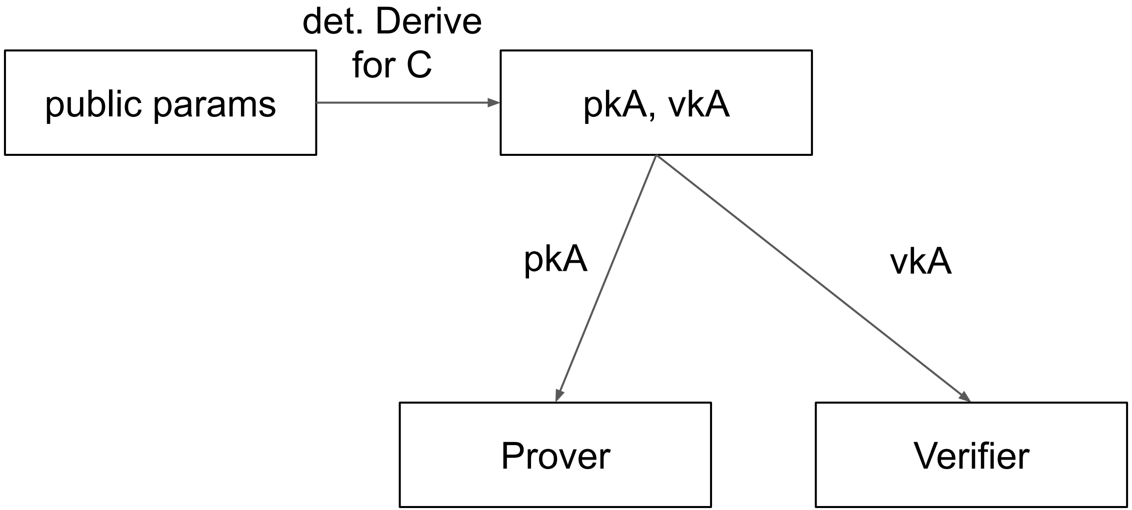
Scheme	Transp.	$\|pp\|$	Prover	Verifier	$\|\pi\|$
DARK *(this work)*	yes	$O(1)$	$O(d^\mu \mu \log(d))$ EXP	$2\mu \log(d)$ EXP	$2\mu \boxed{\log(d)\ \mathbb{G}_U}$
Based on Pairings	no	$d^\mu\ \mathbb{G}_B$	$O(d^\mu)$ EXP	μ Pairing	$\mu\ \mathbb{G}_B$
[BCC+16b, $\sqrt{\cdot}$]	yes	$\sqrt{d^\mu}\mathbb{G}_P$	$O(d^\mu)$ EXP	$O(\sqrt{d^\mu})$EXP	$O(\sqrt{d^\mu})\ \mathbb{G}_P$
Bulletproofs	yes	$2d^\mu\mathbb{G}_P$	$O(d^\mu)$ EXP	$O(d^\mu)$EXP	$2\mu \log(d)\ \mathbb{G}_P$
FRI ($\mu = 1$)	yes	$O(1)$	$O(\lambda d)$ H	$O(\lambda \log^2(d))$ H	$O(\lambda \boxed{\log^2(d))\ \mathrm{H}}$

Table 2: Comparison table between different polynomial commitment schemes for an μ-variate polynomial of degree d.

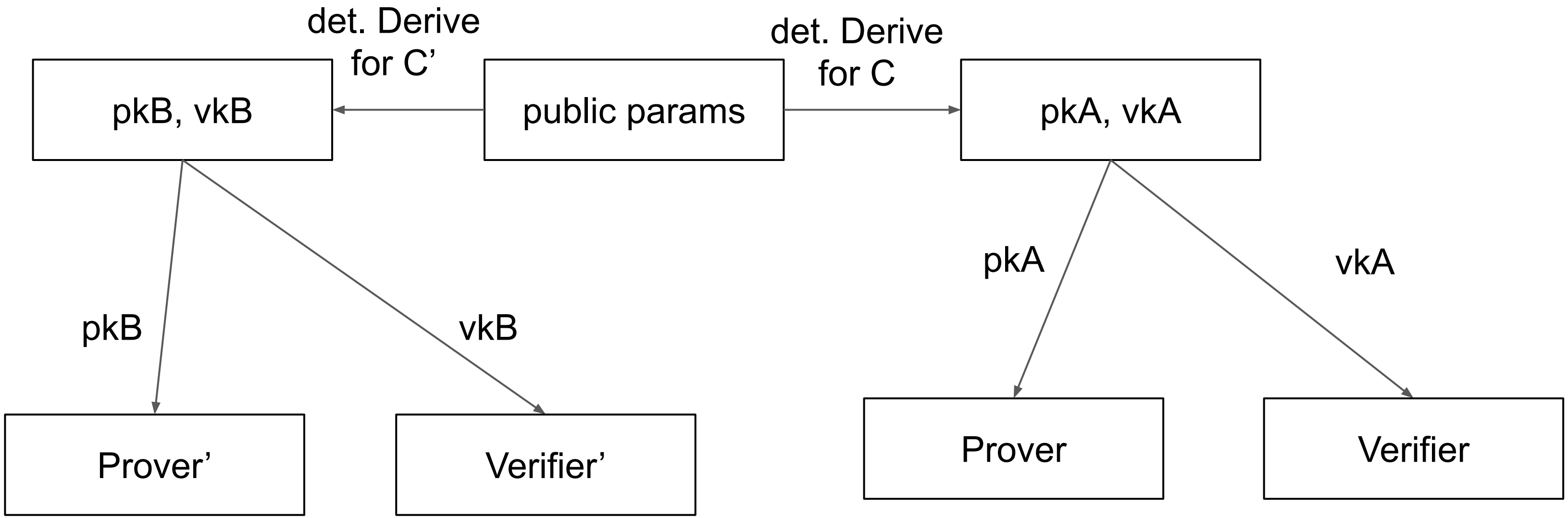
A universal setup example

public params

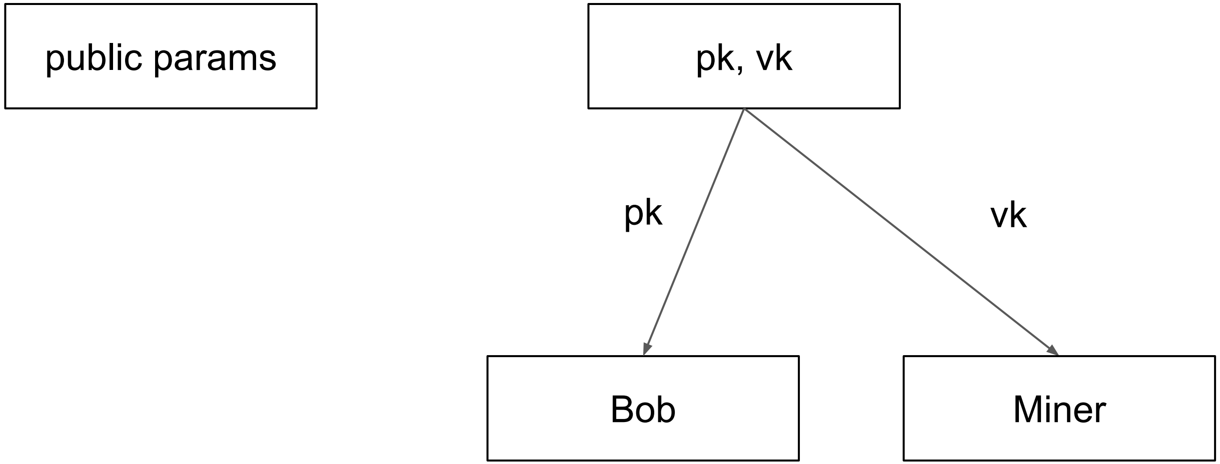
A universal setup example



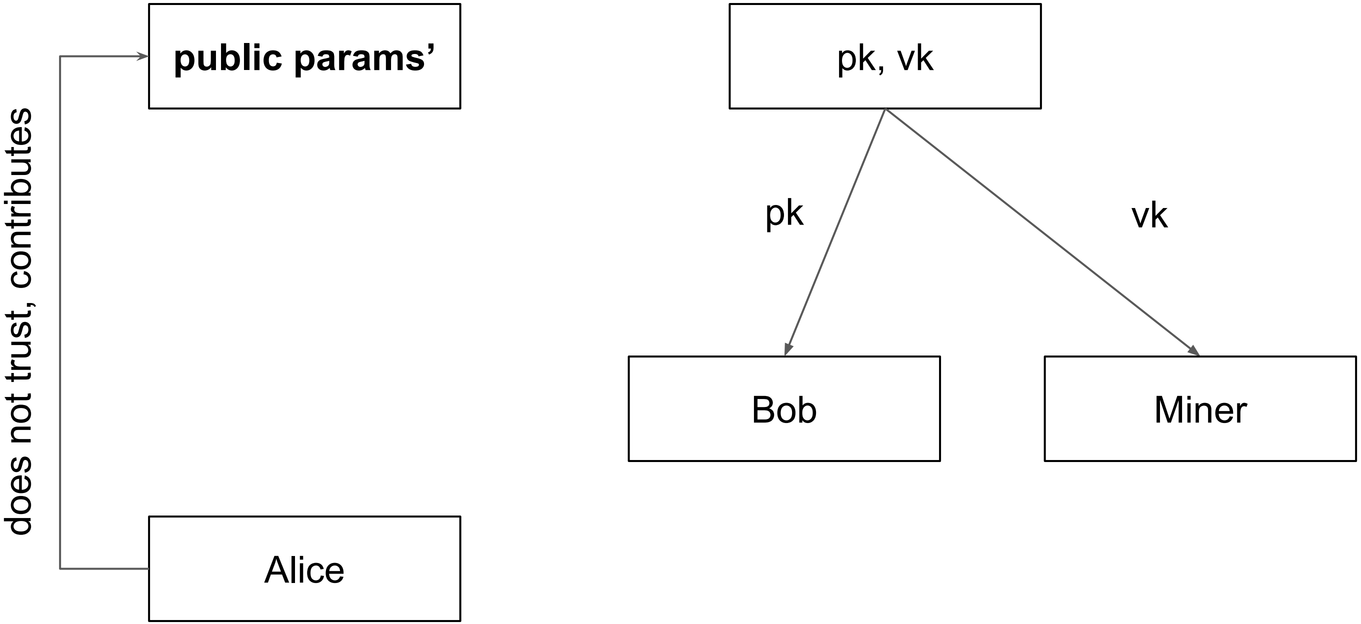
A universal setup example



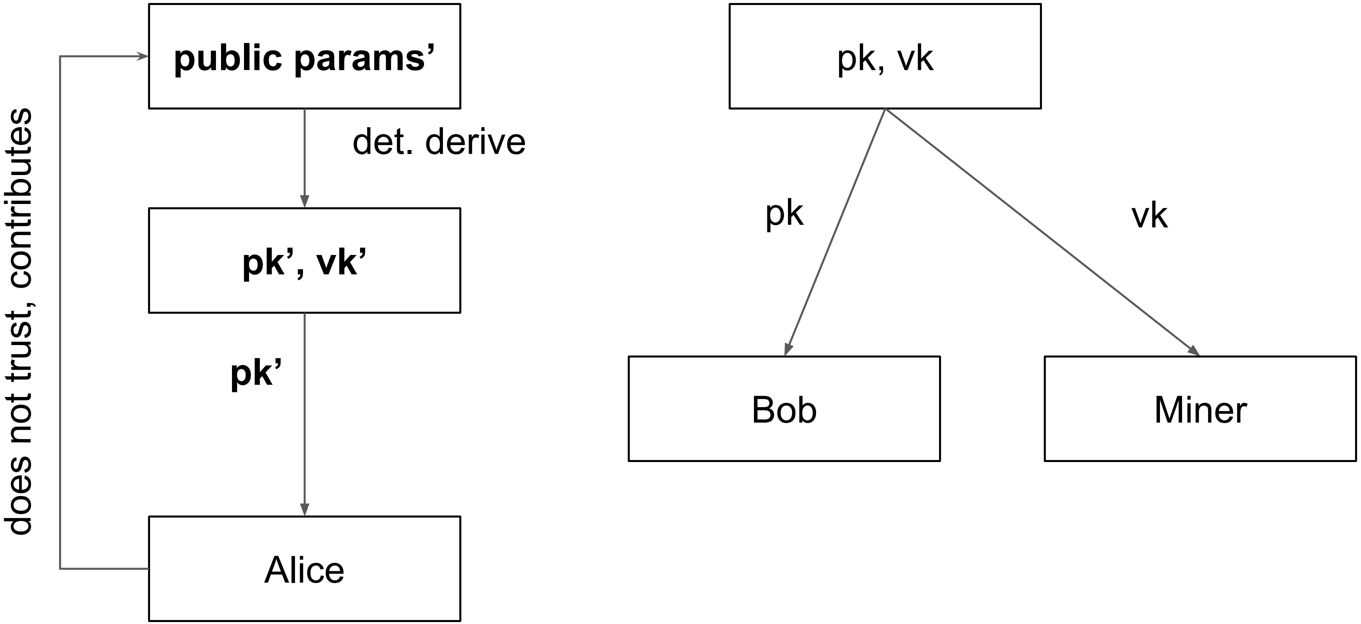
“Implementation detail” of continuous setup



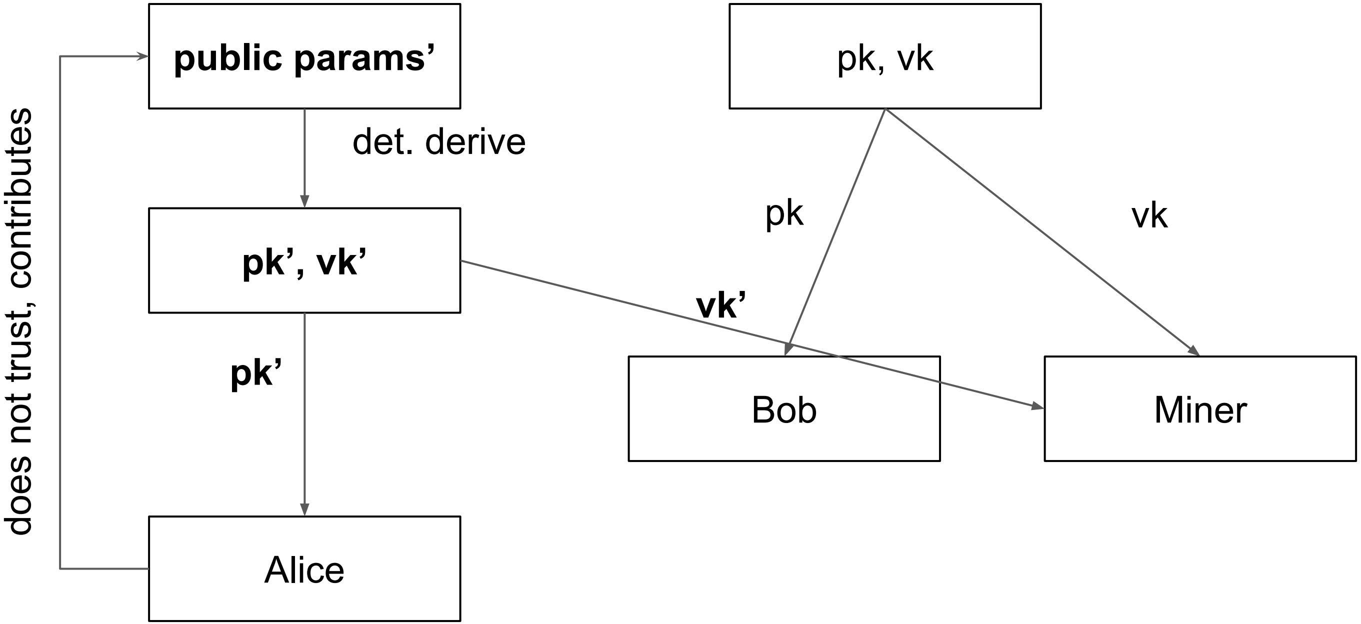
“Implementation detail” of continuous setup



“Implementation detail” of continuous setup



Updated CRS must be distributed



Bob needs pk' to make proofs / Miner has to keep vk (bad?)

groth16 dead?

Sonic

R1CS → S(X, Y) → S(XY, 1) → univariate polycommit → SNARK

Scheme	Universal SRS	Circuit SRS	Size	Prover computation	Verifier computation
Groth'16 [45]	—	$3n + m\ \mathbb{G}$	$3\ \mathbb{G}$	$4n + m - \ell$ Ex	$3P + \ell$ Ex
Bulletproofs	$\frac{n}{2}\mathbb{G}$	—	$2\log_2(n) + 6\ \mathbb{G}$	$8n$ Ex	$4n$ Ex
This work (helped)	$4d\mathbb{G}$	$12n\ \mathbb{G}$	$7\ \mathbb{G}$, $5\ \mathbb{F}$	$18n$ Ex	$10P$
This work (unhelped)	$4d\mathbb{G}$	$36n\ \mathbb{G}$	$20\ \mathbb{G}$, $16\ \mathbb{F}$	$273n$ Ex	$13P$

Evaluate S(X, Y) at (z,y) chosen by verifier → can we abuse that?

1. Parallel proof gen (eval of s does not depend on statement)
2. Helped mode: batch $s(z_i, y_i)$ → bigger asymptotic, but more practical
3. Fully succinct: write polynomial as sum for permutations → more constraints → big constants

PLONK

$$(Q_{L_i})a_i + (Q_{R_i})b_i + (Q_{O_i})c_i + (Q_{M_i})a_ib_i + Q_{C_i} = 0$$

1. Addition & multiplication gates: set Q coeffs depending on wires
2. Output of gate as input to other: “copy” wires → permutation argument

Table 1: Prover comparison. m = number of wires, n = number of multiplication gates, a = number of addition gates

	size $\leq d$ **SRS**	**size** $= n$ **CRS/SRS**	**prover work**	**proof length**	**succinct**	**universal**
Groth'16	-	$3n + m\ \mathbb{G}_1$	$3n + m - \ell\ \mathbb{G}_1$ exp, $n\ \mathbb{G}_2$ exp	$2\ \mathbb{G}_1$, $1\ \mathbb{G}_2$	✓	✗
Sonic (helped)	$12d\ \mathbb{G}_1$, $12d\ \mathbb{G}_2$	$12n\ \mathbb{G}_1$	$18n\ \mathbb{G}_1$ exp	$4\ \mathbb{G}_1$, $2\ \mathbb{F}$	✗	✓
Sonic (succinct)	$4d\ \mathbb{G}_1$, $4d\ \mathbb{G}_2$	$36n\ \mathbb{G}_1$	$273n\ \mathbb{G}_1$ exp	$20\ \mathbb{G}_1$, $16\ \mathbb{F}$	✓	✓
Auroralight	$2d\ \mathbb{G}_1$, $2d\ \mathbb{G}_2$	$2n\ \mathbb{G}_1$	$8n\ \mathbb{G}_1$ exp	$6\ \mathbb{G}_1$, $4\ \mathbb{F}$	✗	✓
This work (small)	$3d\ \mathbb{G}_1$, $1\ \mathbb{G}_2$	$3n + 3a\ \mathbb{G}_1$, $1\ \mathbb{G}_2$	$11n + 11a\ \mathbb{G}_1$ exp, $\approx 54(n+a)\log(n+a)\ \mathbb{F}$ mul	$7\ \mathbb{G}_1$, $7\ \mathbb{F}$	✓	✓
This work (fast prover)	$d\ \mathbb{G}_1$, $1\ \mathbb{G}_2$	$n + a\ \mathbb{G}_1$, $1\ \mathbb{G}_2$	$9n + 9a\ \mathbb{G}_1$ exp, $\approx 54(n+a)\log(n+a)\ \mathbb{F}$ mul	$9\ \mathbb{G}_1$, $7\ \mathbb{F}$	✓	✓

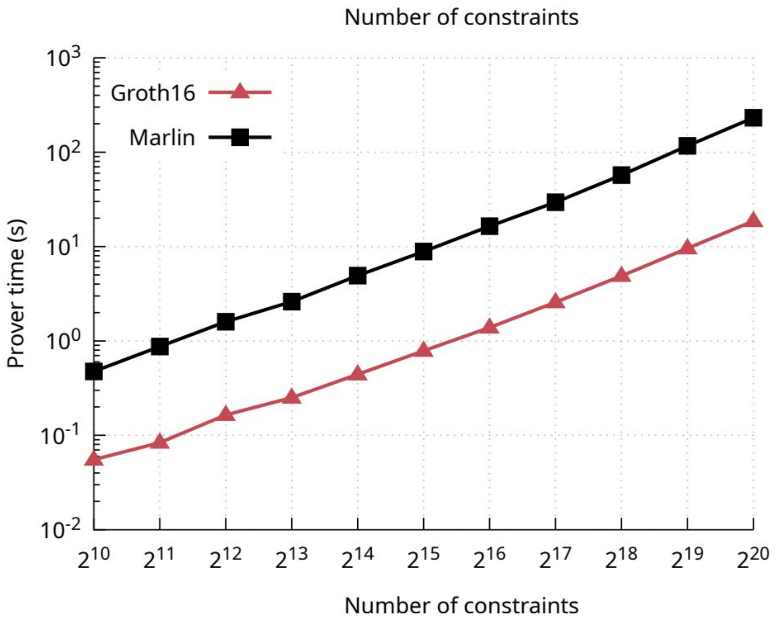
https://vitalik.ca/general/2019/09/22/plonk.html

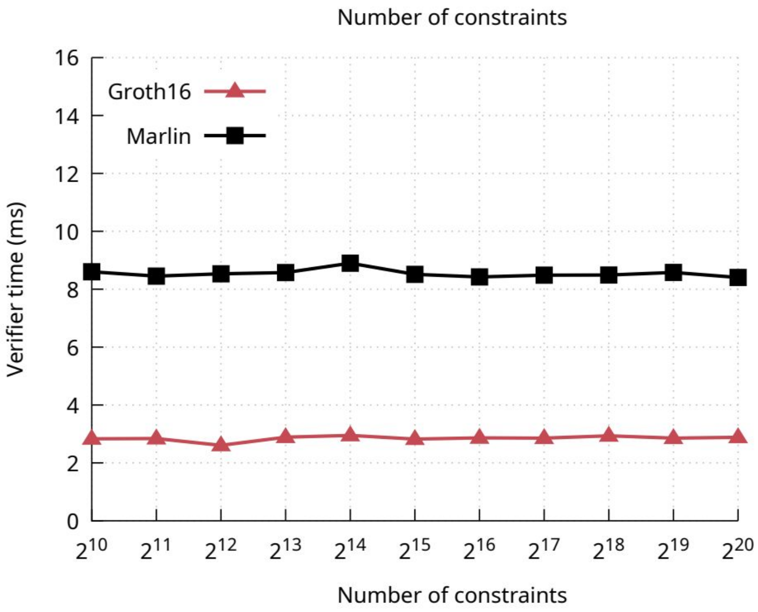
PLONK verifying performance

	verifier work	elem. from helper	extra verifier work in helper mode
Groth'16	$3P$, ℓ $\mathbb{G}_1$ exp	-	-
Sonic (helped)	$10P$	3 $\mathbb{G}_1$, 2 $\mathbb{F}$	$4P$
Sonic (succinct)	$13P$	-	-
Auroralight	$5P$, 6 $\mathbb{G}_1$ exp	8 $\mathbb{G}_1$, 10 $\mathbb{F}$	$12P$
This work (small)	$2P$, 16 $\mathbb{G}_1$ exp	-	-
This work (fast prover)	$2P$, 18 $\mathbb{G}_1$ exp	-	-

Marlin performance

(concurrent work with PLONK & DARKs, targetted to R1CS instead of CSAT)



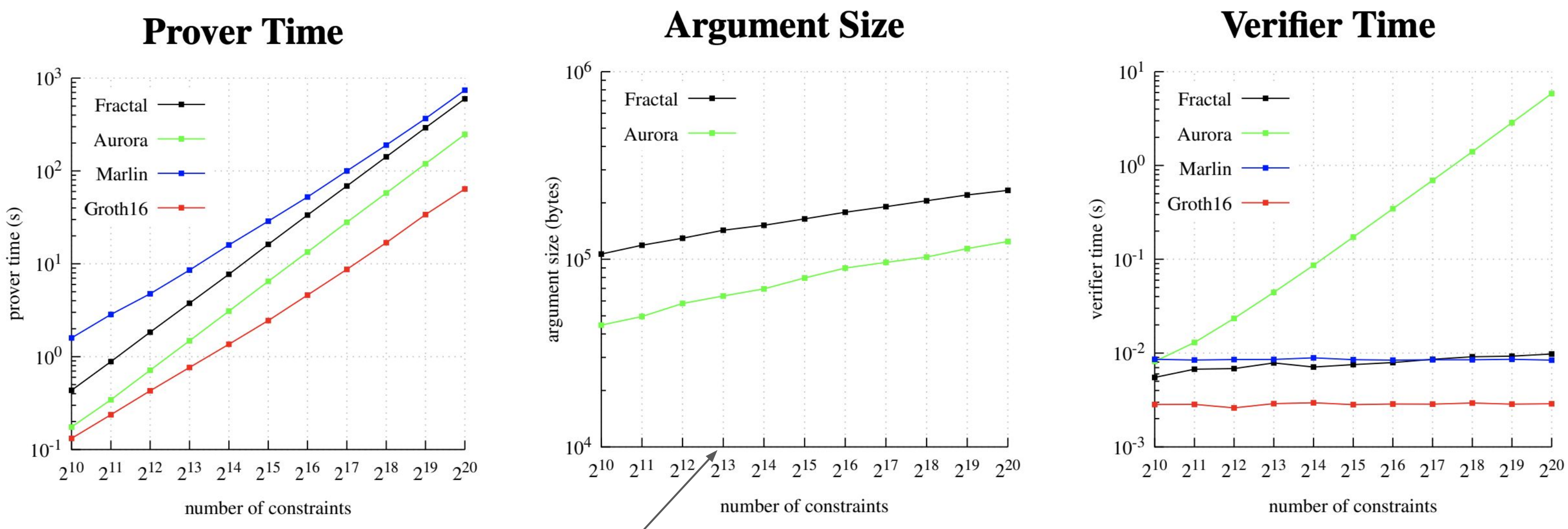


post quantum?

FRI-based SNARKs (or STARKs?)

1. Quantum-secure polycommit → quantum-secure SNARK
2. FRI + low-degree testing
3. Hashes are quantum-secure (to some extent)
4. COS19 (Fractal): “marlin extension”
5. VP19 (proof size $O(\log^2 n)$)

Verifier & Prover are *practically* the same



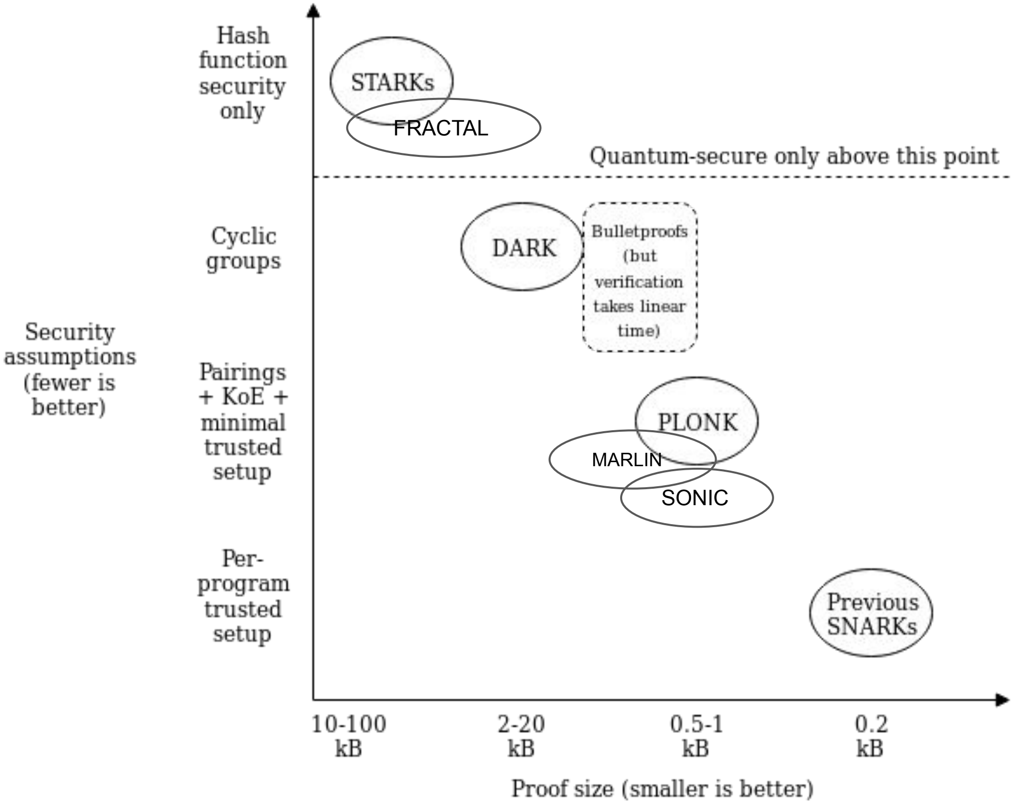
Gro16 + Marlin too small to be shown

https://eprint.iacr.org/2019/1076.pdf

Practical arguments for Groth16

- R1CS!!!
- Groth16 implemented w/ multiple backends
  - Circom
  - Bellman
  - libsnark
- Batch verifiable
- GPU Provers (Coda), Distributed Provers (DIZK)
- Real world bounty to break it (Zcash)
- Proofs can be aggregated using MV19 inner-product argument [logN * (2G1 + G2) size for N proofs], *I think*

Tradeoff Space



https://vitalik.ca/general/2019/09/22/plonk.html

Thank you for your attention

Is the R1CS abstraction enough? Does it need to be lifted to be more expressive?

[@gakonst](https://twitter.com/gakonst) / me@gakonst.com

gakonst.com/zksummit2019.pdf