High-speed zkSNARKs without trusted setup

Srinath Setty
Microsoft Research

https://github.com/Microsoft/Spartan

Based on work with Jonathan Lee, Justin Thaler, and Riad Wahby
(ePrint 2019/550, 2020/1274, 2020/1275, and 2021/030)

Spartan [Set19, SL20, LSTW21] in a nutshell

A family of zkSNARKs without trusted setup for R1CS

Achieves sub-linear verification costs for arbitrary, non-uniform computations

- First work to achieve this without requiring a trusted setup

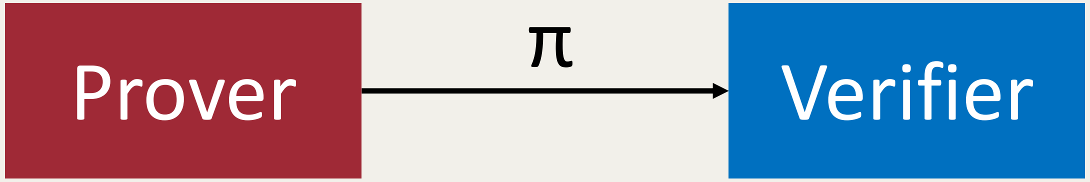
State-of-the-art performance

- Fastest prover among general-purpose proof systems
- Shortest proofs and verification among zkSNARKs without trusted setup

Ingredients: The sum-check protocol + polynomial commitments

- Information-theoretic component: linear-time prover, logarithmic verifier and proof sizes
- Cryptographic assumptions and costs are inherited from the PC scheme
- Several candidate PC schemes with different assumptions (e.g., DLOG, SXDH, CRHF) and properties (e.g., a linear-time prover, post-quantum security)

What is a zkSNARK?



1. Argument of Knowledge

- Prove the knowledge of w :: C(w, x) = 1

2. Zero-knowledge

- π hides w

3. Non-interactive

4. Succinct

- $|\pi|$ is sub-linear in $|C|$
- Verifier runs in time sub-linear in $|C|$

Many approaches to build zkSNARKs

Early theory: Short PCPs + Merkle trees [BFLS91, Kilian92, Micali94], …

- Extreme expense → Impractical

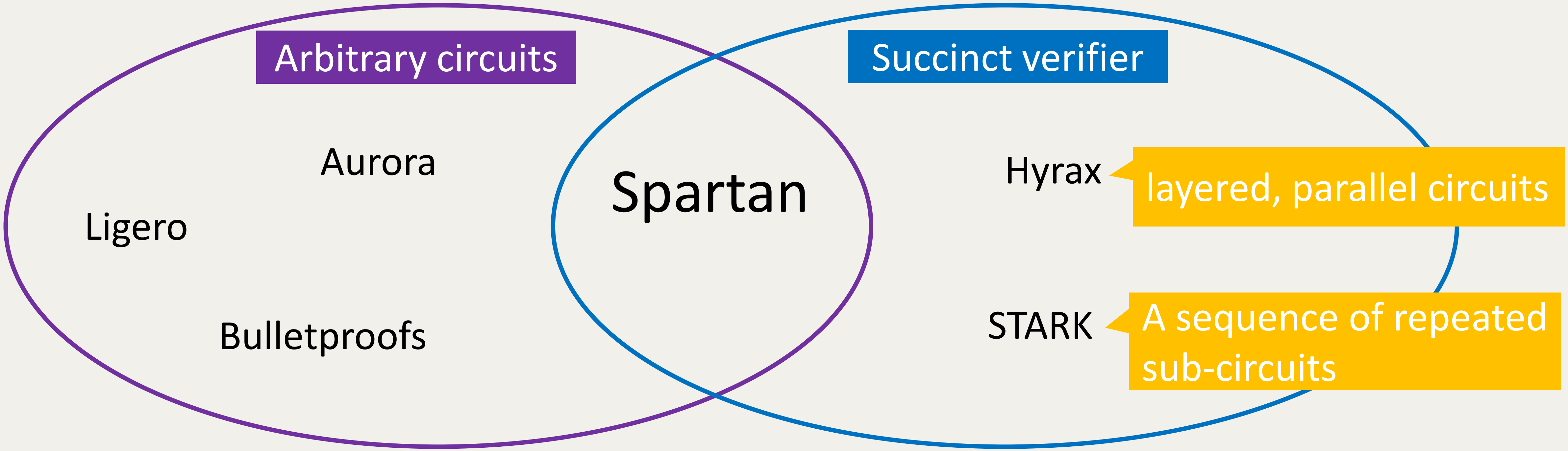
A seminal result: [GGPR13] building on [IKO07, Groth10, Lipmaa12]

- Supports arbitrary, non-uniform circuits
- Achieves near-optimal asymptotics with good constants
  - Prover: $O(|C| \log |C|)$
  - Proof size: $O(1)$
  - Verifier: $O(|x|)$
- Requires a per-circuit trusted setup (trapdoor used must be kept secret)

Recent focus: zkSNARKs without trusted setup

Several schemes: Ligero, Hyrax, STARK, Bulletproofs, and Aurora

These works can support arbitrary circuits or a succinct verifier, not both



Challenges with achieving succinct verification

1. Arbitrary circuits have no structure
2. The verifier must know what statement is being proven!

(1) + (2) ➔ Verification must be at least $O(|C|)$

Spartan's verifier preprocesses circuits—*but without secret trapdoors*

- The verifier retains a commitment to circuit—computation commitment
- Preprocessing incurs $O(|C|)$ costs, but is reusable (like that of [GGPR13])
- Follow-up works refer to this as leveraging "holography"

Rest of this talk

- Background
- An overview of Spartan
- Performance results

Background: The sum-check protocol [LFKN90]

An interactive proof system for *sum-check instances*:

$$T = \sum_{x \in \{0,1\}^l} G(x),$$

where G is a multivariate polynomial in l variables

The protocol proceeds in l rounds

- In each round:
  - V → P: one random field element, and
  - P → V: $\mathrm{O}(d)$ where, d is the degree of G in any variable
- In each round, V performs $\mathrm{O}(d)$ finite field operations
- At the end, the verifier must evaluate $G(r)$, where r is chosen by the verifier over the course of the protocol

Advantages of sum-check:

- No trusted setup
- Public coin → NI in the ROM
- Prover can be implemented in linear-time for certain G [Tha13, XZXPS19]

Gaps for constructing SNARKs:

- Encode R1CS as sum-check instances
- The protocol is not succinct: proof sizes and verification

Encoding R1CS as sum-check instances

R1CS: Given three m x m matrices A, B, C, does there exist z such that:

$$Az \circ Bz = Cz, \text{ where } \circ \text{ is entry-wise multiplication of vectors}$$

1. **View $\boldsymbol{A, B, C, z}$ as functions:**
   - $z: \{0,1\}^s \to \mathbb{F}$, where $s = \log(m)$ i.e., z maps s bits to field elements
   - A, B, C: $\{0,1\}^s$ x $\{0,1\}^s \to \mathbb{F}$
2. **Let $\boldsymbol{\widetilde{A}, \widetilde{B}, \widetilde{C}, \tilde{z}}$ denote multilinear polynomials that extend $\boldsymbol{A, B, C, z}$**
   - $\forall y \in \{0,1\}^s \quad z(y) = \tilde{z}(y)$
   - $\forall x, y \in \{0,1\}^s \quad A(x,y) = \tilde{A}(x,y), B(x,y) = \tilde{B}(x,y), C(x,y) = \tilde{C}(x,y)$
3. **Consider:**

$$F(x) = \sum_y \tilde{A}(x,y) \cdot \tilde{z}(y) \times \sum_y \tilde{B}(x,y) \cdot \tilde{z}(y) - \sum_y \tilde{C}(x,y) \cdot \tilde{z}(y)$$

Lemma: $\forall x \in \{0,1\}^s \quad F(x) = 0$ if and only if $Az \circ Bz = Cz$

Encoding R1CS as sum-check instances

R1CS: Given three m x m matrices A, B, C, does there exist z such that:

$$Az \circ Bz = Cz, \text{ where } \circ \text{ is entry-wise multiplication of vectors}$$

3. **Consider:**

$$F(x) = \sum_y \tilde{A}(x,y) \cdot \tilde{z}(y) \times \sum_y \tilde{B}(x,y) \cdot \tilde{z}(y) - \sum_y \tilde{C}(x,y) \cdot \tilde{z}(y)$$

Lemma: $\forall x \in \{0,1\}^s \;\; F(x) = 0$ if and only if $Az \circ Bz = Cz$

4. **Theorem: Proving that $\forall \boldsymbol{x} \in \{\mathbf{0}, \mathbf{1}\}^s \;\; \boldsymbol{F}(\boldsymbol{x}) = \mathbf{0}$ is equivalent to:**

$$0 = \sum_{x \in \{0,1\}^s} F(x) \cdot \tilde{E}(x, \tau),$$

- Except for a soundness error of $\log(m)\,/|\mathbb{F}|$ over the choice of τ
- $\tilde{E}$ is the unique multilinear polynomial that evaluates to 1 if the arguments are equal and 0 otherwise (on the Boolean hypercube).

R1CS instance

$$\exists z: Az \circ Bz = Cz$$

$$\forall x \in \{0,1\}^s \quad F(x) = 0, \textit{ where}$$

$$F(x) = \sum_y \tilde{A}(x,y) \cdot \tilde{z}(y) \times \sum_y \tilde{B}(x,y) \cdot \tilde{z}(y) - \sum_y \tilde{C}(x,y) \cdot \tilde{z}(y)$$

Except for a soundness error of $\log(m) / |\mathbb{F}|$

Sum-check instance

$$0 = \sum_{x \in \{0,1\}^s} F(x) \cdot \tilde{E}(x, \tau)$$

- If we apply the sum-check protocol, the verifier must evaluate five polynomials:
  - $\tilde{E}(r_x, \tau)$
  - $\tilde{z}(r_y)$
  - $\tilde{A}(r_x, r_y)$, $\tilde{B}(r_x, r_y)$, $\tilde{C}(r_x, r_y)$
- $\tilde{E}(r_x, \tau)$ can be evaluated in $O(\log(m))$ time
- $\tilde{z}(r_y)$ depends on witness
- $\tilde{A}(r_x, r_y)$, $\tilde{B}(r_x, r_y)$, $\tilde{C}(r_x, r_y)$ depends on the statement

Evaluating $\tilde{z}(r_y)$ succinctly (for the verifier)

- Idea: Employ an extractable polynomial commitment scheme

- Prover → Verifier: Commit(pp, $\tilde{z}$) (*before* the sum-check protocol)
- [Run the sum-check protocol]
- Prover → Verifier: $(\tilde{z}(r_y), \pi)$, where π is a proof of correct evaluation
- Verifier: Verify π using the commitment; use the supplied $\tilde{z}(r_y)$ to evaluate $F(x)$

- Many polynomial commitment schemes: Hyrax-PC, Dory, Ligero-PC, BCG-PC etc.
- |π| and the cost to verify π are succinct (i.e., sub-linear in |z|)

Evaluating $\tilde{A}(r_x, r_y), \tilde{B}(r_x, r_y), \tilde{C}(r_x, r_y)$ succinctly

Computation commitment

- Idea: Employ a polynomial commitment scheme
- In a preprocessing phase, the verifier computes:

$$\Gamma \leftarrow \left(Commit\left(pp, \tilde{A}\right), Commit\left(pp, \tilde{B}\right), Commit(pp, \tilde{C})\right)$$

- Prover → Verifier: Commit(pp, $\tilde{z}$) (*before* the sum-check protocol)
- [Run the sum-check protocol]
- Prover → Verifier:
  - $\tilde{z}(r_y), \pi_z$
  - $\left(\tilde{A}(r_x, r_y), \tilde{B}(r_x, r_y), \tilde{C}(r_x, r_y)\right), (\pi_A, \pi_B, \pi_C)$
- Verifier: Verify proofs against commitments and evaluate $F(x)$ using the supplied evaluations

This almost works, but …

$\tilde{A}, \tilde{B}, \tilde{C}$ are sparse polynomials. That is, A, B, C are sparse matrices

- In practice, the number of non-zero entries is n = O(m)
- But the total number of entries is $O(m^2)$

$$\begin{bmatrix} 5 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Existing polynomial commitment schemes incur $O(m^2)$ costs (for creating commitments and producing evaluation proofs)

- Prover is quadratic in the statement size

Our solution: SPARK

PolyCommit for multilinear polynomials → **SPARK compiler** → PolyCommit for sparse multilinear polynomials

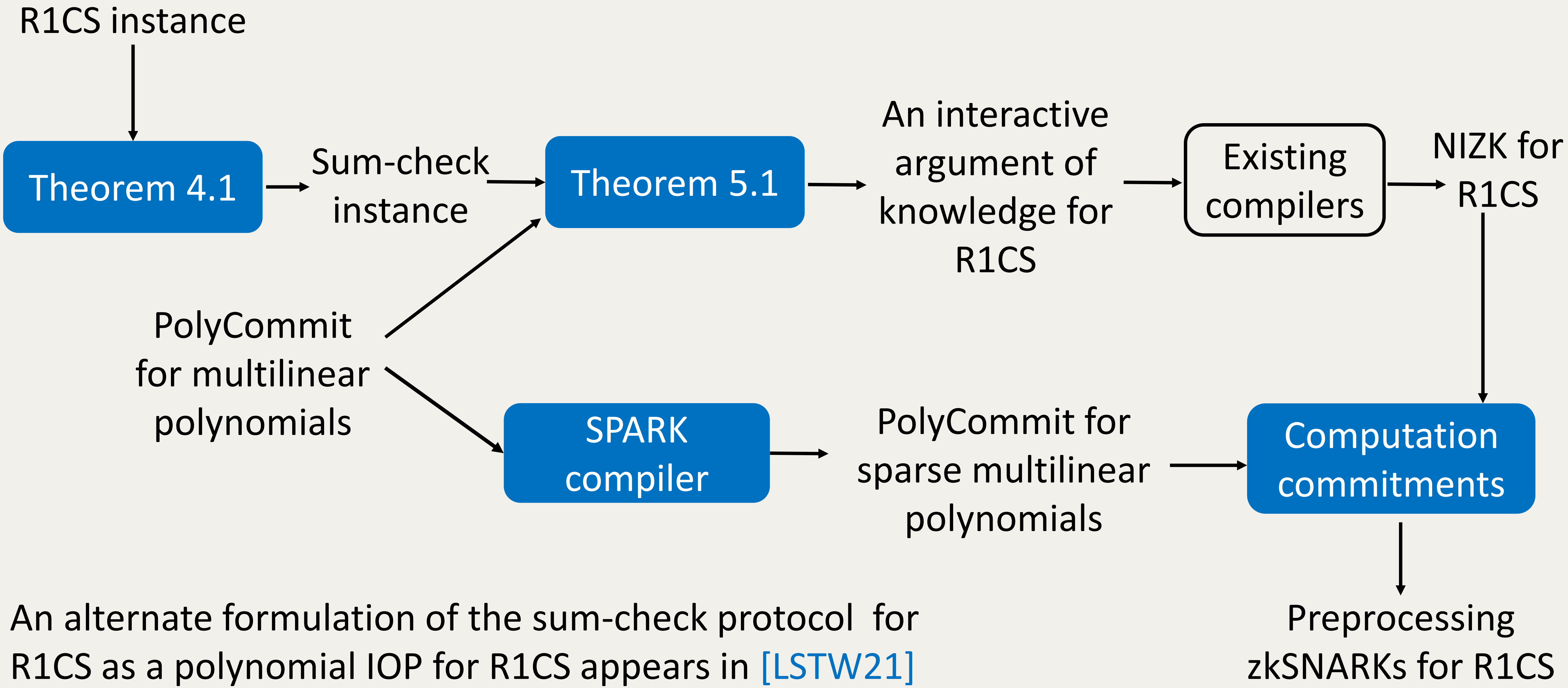
- Commitments: commit to dense representations

$$\begin{bmatrix}0&5&0\\8&0&9\\0&0&0\\0&0&0\end{bmatrix} \rightarrow \left.\begin{array}{ccc} \text{row} & \text{col} & \text{val}\\ \begin{bmatrix}0\\1\\1\end{bmatrix} & \begin{bmatrix}1\\0\\2\end{bmatrix} & \begin{bmatrix}5\\8\\9\end{bmatrix}\end{array}\right\}$$

Each polynomial size = n, #non-zero entries

- Sparse polynomial evaluation (see [Set19, SL20, LSTW21] for details):
  - Devise an efficient sum-check instance for proving sparse polynomial evaluations (employs offline memory-checking primitives)
  - Apply the sum-check protocol with polynomial commitments

Spartan's techniques in a nutshell



An alternate formulation of the sum-check protocol for R1CS as a polynomial IOP for R1CS appears in [LSTW21]

We have implemented Spartan with multiple polynomial commitments

- Spartan [Set19] uses Hyrax-PC [WTsTW18] (DLOG)
- Kopis [SL20] uses a multilinear adaptation of [BMMTV19] (SXDH)
- Xiphos [SL20] uses Dory [Lee20] (SXDH)
- Cerberus [LSTW21] uses a scheme implicit in Ligero [AHIV17] (CRHFs)

Other potential polynomial commitment schemes:

- [LSTW21] describes a scheme we distill from [BCG20]: Provides better asymptotics for the prover
- Virgo: Provides better asymptotics for the verifier compared to Ligero-PC
- DARK: Dory dominates this scheme on *nearly all* aspects
- [KZG10, PST13]: Requires a trusted setup

Performance of zkSNARKs for 2^{20} R1CS constraints

	Prover time (s)	Proof size (KB)	Verifier time (ms)	
Groth16 [EUROCRYPT16]	76	0.2	3	sub-linear verifier with trusted setup
Spartan [CRYPTO20]	45	131	97	sub-linear verifier without trusted setup
SuperSonic* [EUROCRYPT 20]	63,800	48	2,570	
Fractal [EUROCRYPT20]	864	2,500	220	
Xiphos	169	61	65	
Kopis	168	39	390	
Cerberus	50	20,828	280	

*Estimates based on microbenchmarks (see [SL20] for details)

- Spartan offers the fastest prover
- Among schemes without trusted setup, Kopis offers the shortest proofs and Xiphos provides the fastest verifier
- Among post-quantum secure schemes, Cerberus offers the fastest prover
- See [Set19, SL20, LSTW21] for a more detailed performance comparison (with other schemes, instance sizes, etc.)

Summary: Spartan [Set19, SL20, LSTW21]

A family of zkSNARKs without trusted setup for R1CS

Achieves sub-linear verification costs for arbitrary computations

- First work to achieve this without requiring a trusted setup

State-of-the-art performance

- Fastest prover among any general-purpose proof system
- Shortest proofs and verification among zkSNARKs without trusted setup

Ingredients: The sum-check protocol + polynomial commitments

- Assumptions and principal costs are inherited from the PC scheme
- Several candidate PC schemes with different assumptions (e.g., DLOG, SXDH, CRHF) and properties (e.g., a linear-time prover, post-quantum security)