UNIVERSITY OF WATERLOO

How practical zkSNARK enabled blockchain privacy?

Guang Gong

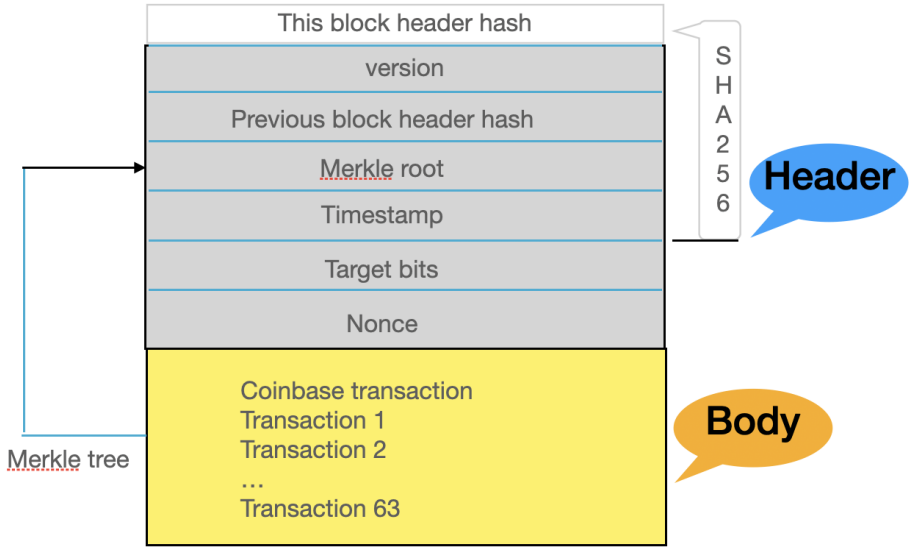
Department of Electrical and Computer Engineering
University of Waterloo
CANADA
https://uwaterloo.ca/scholar/ggong

CPI Annual Conference
University of Waterloo, October 6-7, 2022

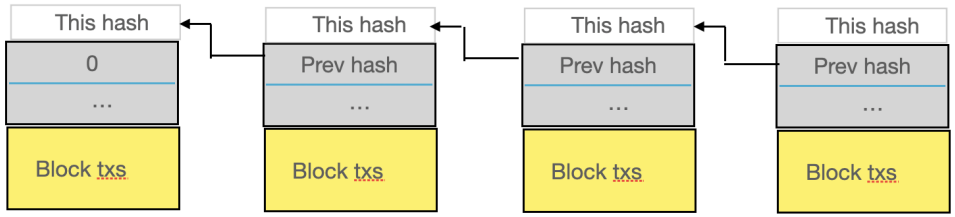
Target: Blockchain privacy

- **Blockchains**, a decentralized peer-to-peer (P2P) ledger system, is gaining interest as a possible solution to many applications:
  - decentralized finance (DeFi)
  - decentralized identity
  - supply chain management, healthcare, $\cdots$
- Blockchains can provide **trusted** consensus, computation, and immutable data between untrusted entities.

A close look at blocks and blockchain



Block structure



Blockchain structure: a blockchain is a singly linked list of blocks by a hash chain.

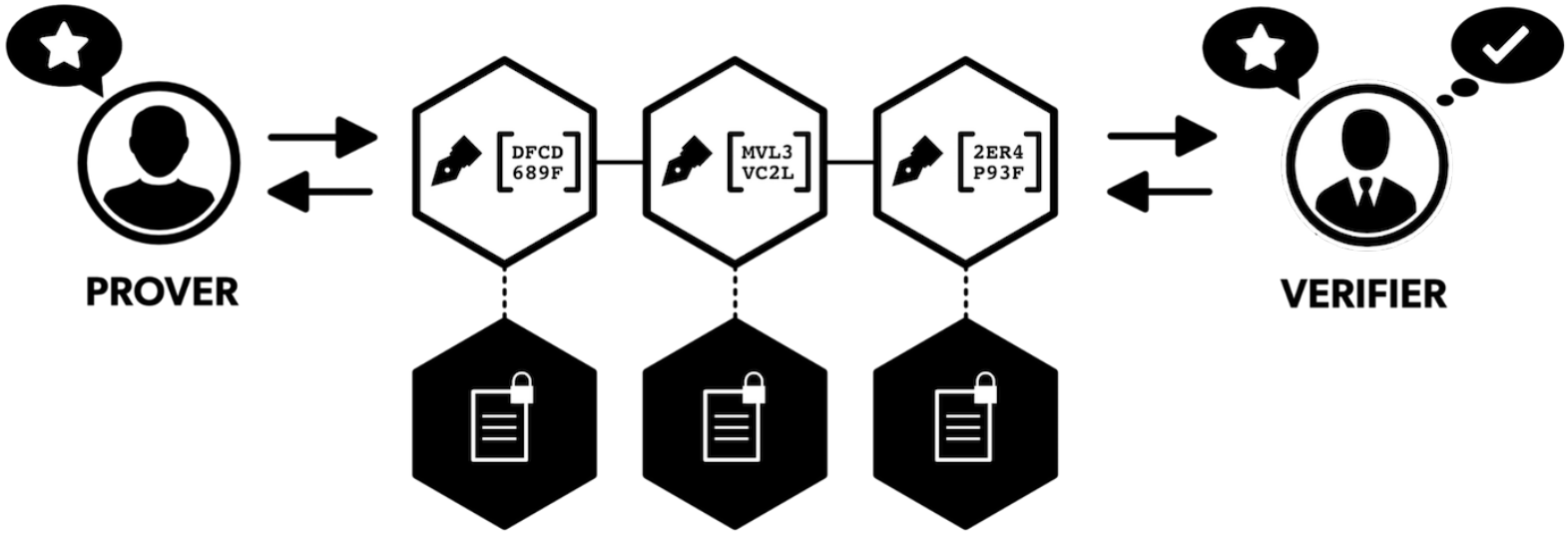
How to achieve blockchain privacy?

Or how to shade sender/receiver and transaction privacy?

Tool: zero-knowledge proof systems

Zero-Knowledge Proofs

Loosely speaking, zero-knowledge proofs are **proofs** that yields **nothing** beyond the validity of the assertion.



Zero-Knowledge Proofs (cont.)

$\mathcal{P}$rover
Alice

$\mathcal{V}$erifier
Bob

X = "I have one BTC
or I upload data for
product (organic salmon)"

I believe X is true.
But I do not know why!

- **Completeness**: $\mathcal{P}$ can convince $\mathcal{V}$ if X is true
- **Soundness**: No malicious $\mathcal{P}^*$ cannot convince $\mathcal{V}$ if X is not true
- **Zero Knowledge**: $\mathcal{V}^*$ learns nothing except for the validity of X

Zero-Knowledge Proofs (cont.)

$\mathcal{P}$rover
Alice

$\mathcal{V}$erifier
Bob

X = "I have one BTC
or I upload data for
product (organic salmon)"

I believe X is true.
But I do not know why!

- **Completeness**: $\mathcal{P}$ can convince $\mathcal{V}$ if X is true
- **Soundness**: No malicious $\mathcal{P}^*$ cannot convince $\mathcal{V}$ if X is not true
- **Zero Knowledge**: $\mathcal{V}^*$ learns nothing except for the validity of X

Zero-Knowledge Proofs (cont.)

$\mathcal{P}$rover
Alice

$\mathcal{V}$erifier
Bob

X = "I have one BTC
or I upload data for
product (organic salmon)"

I believe X is true.
But I do not know why!

- **Completeness**: $\mathcal{P}$ can convince $\mathcal{V}$ if X is true
- **Soundness**: No malicious $\mathcal{P}^*$ cannot convince $\mathcal{V}$ if X is not true
- **Zero Knowledge**: $\mathcal{V}^*$ learns nothing except for the validity of X

Zero-Knowledge Proofs (cont.)

$\mathcal{P}$rover
Alice

$\mathcal{V}$erifier
Bob

X = "I have one BTC
or I upload data for
product (organic salmon)"

I believe X is true.
But I do not know why!

- **Completeness**: $\mathcal{P}$ can convince $\mathcal{V}$ if X is true
- **Soundness**: No malicious $\mathcal{P}^*$ cannot convince $\mathcal{V}$ if X is not true
- **Zero Knowledge**: $\mathcal{V}^*$ learns nothing except for the validity of X

ZKP efficiency

- **Prover complexity**: Computational cost for the prover to run the protocol.
- **Round** complexity: Number of transmissions between prover and verifier.
- **Proof length (or communication**: Total size of communication between prover and verifier.
- **Verifier** complexity: Computational cost for the verifier.
- Setup cost: Size of setup parameters, e.g. a **common reference string (CRS)**, and computational cost of creating the setup.

How about integrity of computation?

$\mathcal{P}$rover
Alice

y

$\mathcal{V}$erifier
Bob

- How can a Alice to prove to Bob that a hash value $y = h(x)$ is correctly evaluated without sending Bob the pre-image x?

Verifiable computation

The integrity of computation is achieved by **verifiable computation**. It can be done through representing an algorithm/program as a circuit.

UNIVERSITY OF WATERLOO

zkSNARK

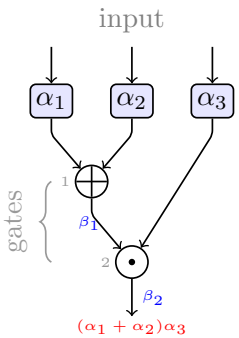
zkSNARK

zero-**k**nowledge **S**uccinct **N**on-interactive **AR**gument of **K**nowledge.

Properties of zkSNARK

- **Z**ero-**K**nowledge: does not leak any information about witness
- **S**uccinct: Proof size is independent of NP witness sizes, i.e., the computing complexity of the prover/verifier and communication (i.e., the proof length) are computationally bounded.
- **N**on-interactive: only one message is sent by prover.
- **AR**gument of **K**nowledge.

Some recent zkSNARKs



Properties of different zkSNARK schemes			
scheme	setup	security	implementation
QAP/QSP based (GGPR13, Groth16) (BCTV14a)	private	KOE	libsnark (BCTV14) Pinocchio, Zcach Hawk
Bullet proof (BCCGP16)	public	DLOG	experiments
Marlin (CHMMVW20)	private	Strong DH	experiments
Spartan$_{DL}$ (Setty20)	public	DLOG	experiments
Ligero (AHIV17)	public	CRH, PRG	Ligero cryptocurrency
Stark (BBHR18)	public	CRH, PRG	libstark
Aurora (BCRSVW19)	public	CRH, PRG	libiop
Polaris (HG2022)	public	CRH, PRG	partial tests

R1CS Relation

From now on, we assume that we have obtained R1CS relation from a circuit converted from a given algorithm/program.

R1CS instance

$\mathbf{x} = (\mathbb{F}, A, B, C, v, m, n)$ and corresponding **witness** w

- A, B, C are $m \times m$ matrices over a large finite field $\mathbb{F}$ representing the computation circuit
- v is the public input and output vector of the instance
- w is the private input vector of the instance
- there are at most n non-zero entries in each matrix

R1CS relation

There exists a witness $w \in \mathbb{F}^{m-|v|-1}$ such that

$$(A \cdot z) \circ (B \cdot z) - (C \cdot z) = \vec{\mathbf{0}},$$

where $z := (1, v, w) \in \mathbb{F}^m$, "$\cdot$" is the matrix-vector product, and "$\circ$" denotes the entry-wise product.

- The **goal** of a zkSNARK scheme is to prove the above relation.
- R1CS relation generalizes the problem of arithmetic circuit satisfiability.
- For the three matrices A, B, C, the vectors Az, Bz and Cz represent the **left input, right input and output** vectors of the **multiplicative gates** in the circuit respectively. The witness w consists of the circuit's private input and wire values.
- For example, if we would like to prove any transaction (i.e., UTXO) in Zcash, it suffices to show the miners that SHA256 circuit $y = SHA256(x)$ without giving the values in $x = UTXO$. Zcash implemented **Groth16** to achieve this goal.

Polaris: a new zkSNARK

- **Polaris** is a new zkSNARK for R1CS computational circuits with polylogarithmic for both proof size and verification time.
- By **instantiating** with different polynomial commitment schemes, it can obtain several zkSNARKs where the verifier's costs and the proof size range from polylogarithmic to sublinear depending on the underlying commitment scheme.
- $\text{Polaris}_{\text{RO}}$ is **public set-up**, with prover complexity $O(N \log N)$, proof size $O(\log^2 N)$, and verifier complexity $O(\log^2 N)$.
- Prover efficiency is improved using a **new efficient sparse encoding**, and verifier cost is reduced from linear to logarithmic by embedding the **GKR protocol into low degree test (LDT)** with **a new explicit** computation circuit as its input.

Encoding R1CS relation in Polaris

- There are two different methods to encode R1CS, one is to interpret the matrices into multi-variate polynomials (e.g., Spartan), and the other univariate polynomials (e.g., Groth16, Aurora). Polaris uses this method.
- Then to prove the relation is true only at a random point.

Product checking polynomial $F_w(X) := \bar{A}(X) \cdot \bar{B}(X) - \bar{C}(X)$ is converted to **Poly-SAT**

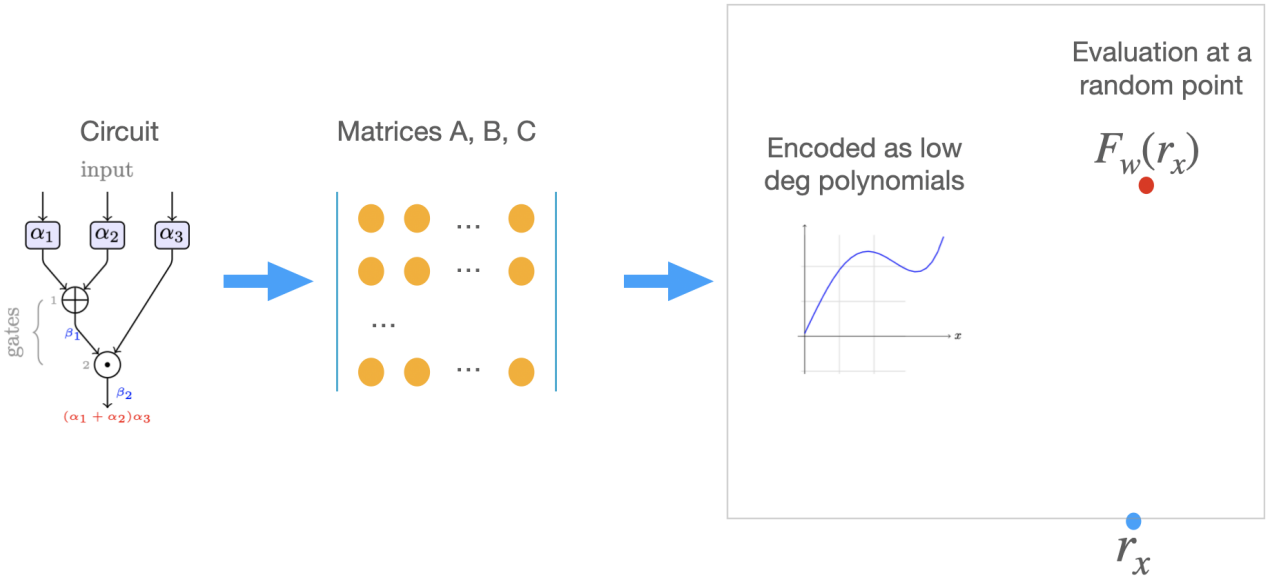
$$F_w(X) = \mathbb{Z}_H(X) \cdot G(X)$$

$$\Downarrow \Uparrow \text{soundness}$$

$$F_w(r_x) = \mathbb{Z}_H(r_x) \cdot G(r_x) \text{ for a random } r_x \in \mathbb{F} \setminus H$$

Univariate sum check This is to check whether the validity of three evaluations: $v_A = \bar{A}(r_x)$, $v_B = \bar{B}(r_x)$, $v_C = \bar{C}(r_x)$ through a random combination:

$$c = r_A v_A + r_B v_B + r_C v_C$$



Polaris in a nutshell

Polaris composes two protocols: the product check and univariate sum check protocols.

Polaris product check protocol

Prover:

- Run the **sparse encoding** to get three polynomials $\bar{A}(X)$, $\bar{B}(X)$, $\bar{C}(X)$ and product **check polynomial** $F_w(X)$. Then compute division to get $G(X)$.
- Run **polynomial commitment (PC)** to commit $G(X)$ (PC can be any, so the complexity depends on a specific PC).
- Generate a proof, called π_1, with 5 elements in the field $\mathbb{F}$.

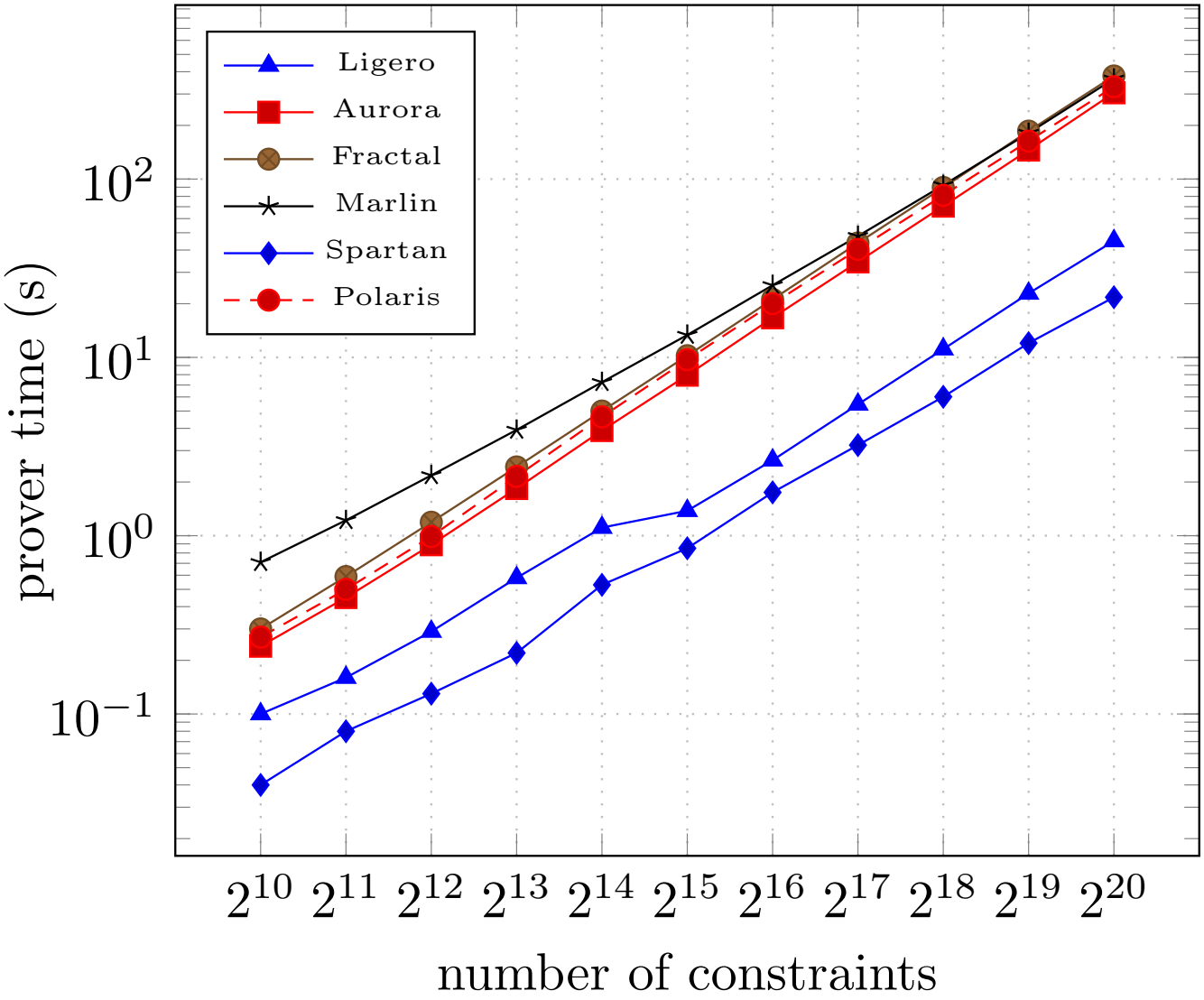
Verifier:

- Verify an identity with one commitment, and two multiplications.
- If it is true, then continue to run Polaris univariate sum-check protocol. Otherwise, abort.

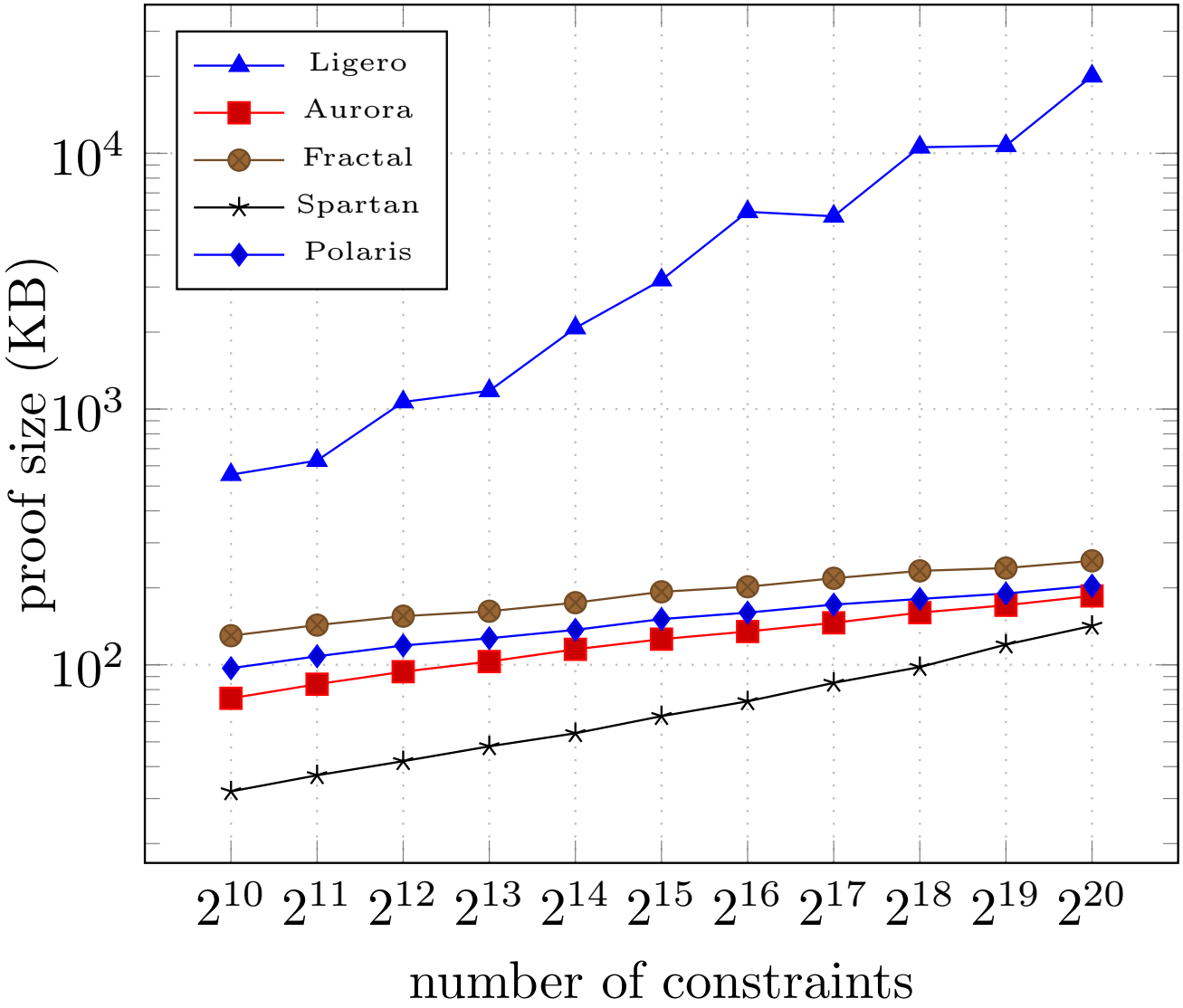
Polaris univariate sum check protocol

- Prover and Verifier together run LDT initial phase.
- Prover runs Merkle tree commitment to commit two Read-Solomon (RS) codewords.
- Prover and Verifier together run GKR protocol three times paralelly.
- Prover runs LDT final phase: generate the proof, called π_2, with logarithmic size.
- Verifier verifies π_2 with logarithmic complexity.

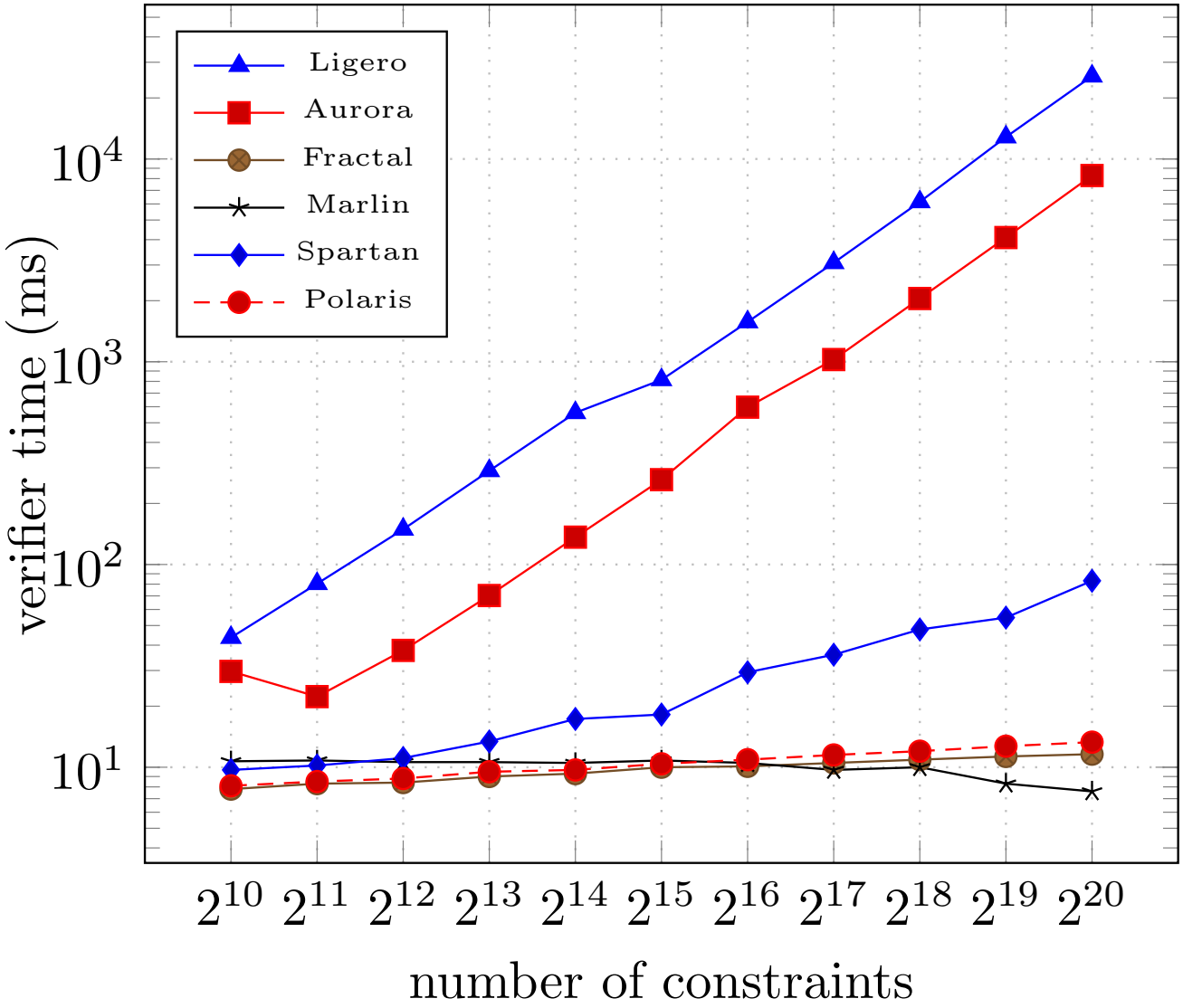
Prover Running Time



Argument Size



Verifier Running Time



Concluding remarks

- For SHA256 circuit, **Polaris** for verifying a SHA-256 preimage (about 23k AND gates) in zero-knowledge with 128 bits security, the proof size is less than 150kB and the verification time is less than 11ms, both **competitive** to existing systems with **better concrete verifiers' complexity**.
- **Polaris** has improved verifier's performance compared with **Ligero/Ligero++, Aurora** for R1CS circuits, and the underlying cryptographic schemes involved are only symmetric cryptography, i.e., collision-resistant hash functions and random number generators.

Concluding remarks (cont.)

- For **blockchain privacy**, a zkSNARK scheme deployed in the real-world systems is the QAP/QSP-based zkSNARK (GGPR13) in 2013. It has constant proof size and verifier complexity. However, it needs a **large CRS**.
- A **vulnerability** has been found in 2019 in their earlier implementation of GGPR13. Zcash advised not to use that implementation, and it is currently updated to implement **Groth16**.
- Contrastively, Polaris is one of the choices for zkSNARKs which do not need any trusted setup and perform heavy pairing cryptographic operations, and possess plausible **post-quantum security**, and can eliminate vulnerabilities in implementations of those heavy pairing operations as well as memory attacks on single point failure for accessing CRSs.

Concluding remarks (cont.)

- The dominant computations in any zkSNARKs are Lagrange interpolation in order to get uni/multi-variate polynomials (corresponding to **IFFT**), and polynomial evaluations or RS codeword generation (**FFT**).
- From coding theory, any practical codes are implemented by linear feedback shift register (LFSR)s through **FFT and IFFT**. So, it may be another way to speed up those computations for efficient zkSNARKs.
- Currently, we are investigating to implement **Polaris** for privacy of **supply chain management**. A bottleneck for the implementation lies in our second subprotocal which uses the GKR protocol.

UNIVERSITY OF WATERLOO

Reference

The content of Polaris is taken from

- Shihui Fu and Guang Gong, Polaris: Transparent Succinct Zero-Knowledge Arguments for R1CS with Efficient Verifier, the Proceedings on Privacy Enhancing Technologies, 2022 (1), pp. 544 - 564.