State of the Art of “ZK” in 2025

(from a researcher’s perspective)

Nicolas Mohnblatt - 17th June, 2025

Confusing acronyms

Acronym	Common use	Academic use
ZK	cryptographic proofs	proofs that are zero-knowledge
SNARK	most proof systems, usually using elliptic curves	systems that are succinct and non-interactive (incl. hash-based)
STARK	proof systems that use hash functions	systems that are scalable* and transparent

* scalable = succinct + decent prover

Confusing acronyms

Acronym	Common use	Academic use
ZK	cryptographic proofs	proofs that are zero-knowledge
SNARK	most proof systems, usually using elliptic curves	systems that are succinct and non-interactive (incl. hash-based)
STARK	proof systems that use hash functions	systems that are scalable* and transparent

* scalable = succinct + decent prover

The state of the art SNARK in 2025 is …

… it depends.

Outline

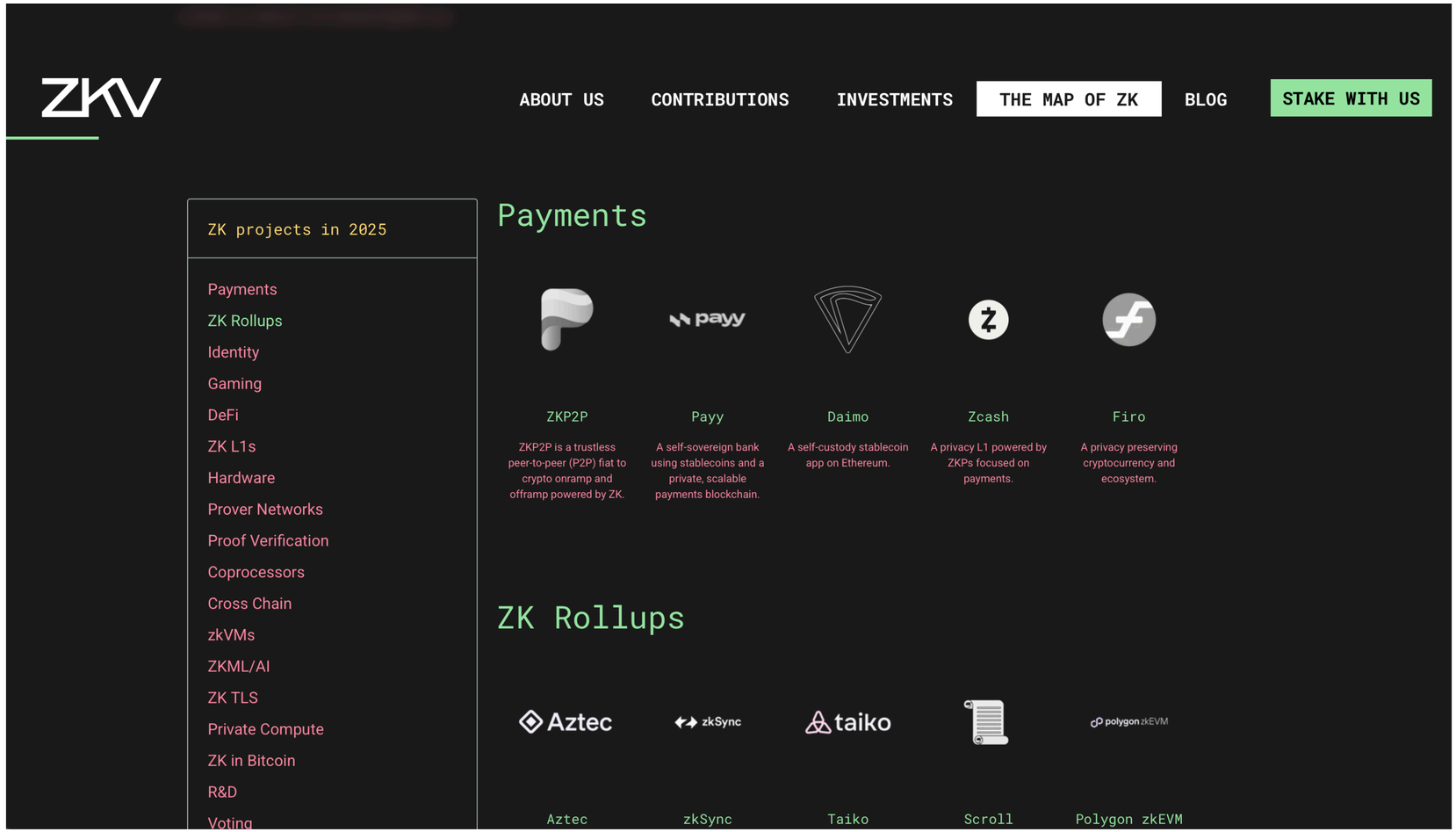
Layered approach:

1. **Where are SNARKs deployed?** Real-world applications and use-cases.
2. **How are SNARKs deployed?** Developer experience for SNARKs.
3. **What SNARKs are deployed?** Proof system design and state of the art.

Where are SNARKs deployed?

Where are SNARKs deployed?

The map of ZK zkv.xyz/the-map-of-zk/



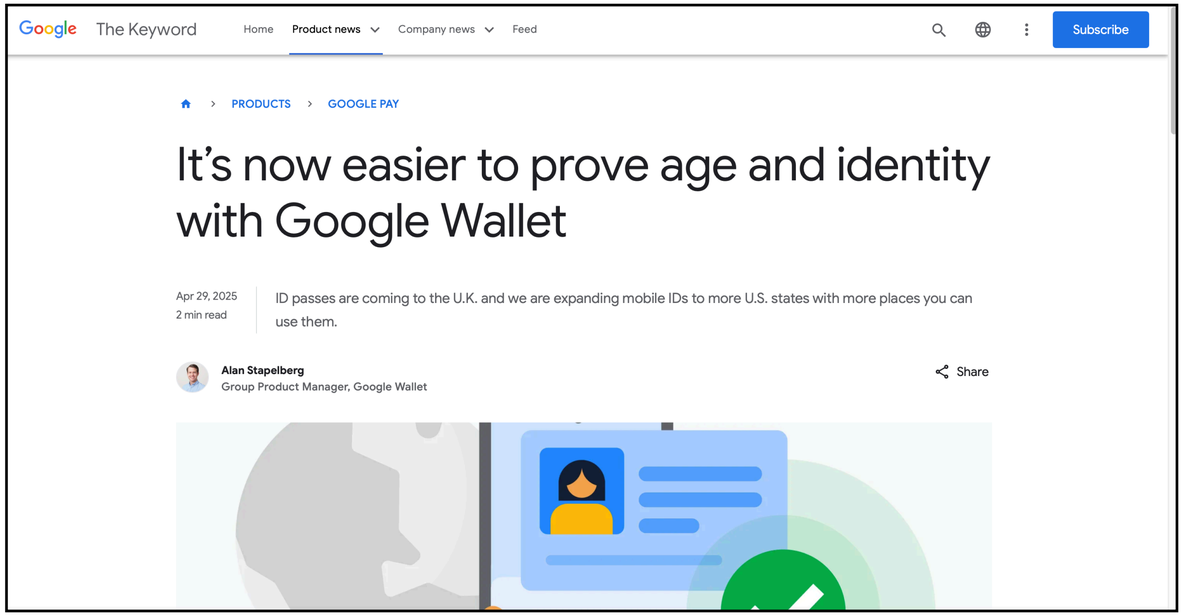
Where are SNARKs deployed?

Some highlights

zkvm	ISA	team	open source	proves mainnet blocks	Ethproofs
Airbender	RISC-V	MatterLabs	✓✓ dual	✓	ETA: Q2
Ceno	RISC-V	Scroll	✓✓ dual	ETA: Q4 (no recursion)	ETA: H2
Euclid (OpenVM)	RISC-V	Scroll	✓✓ dual	✓	ETA: Q2
Ix	Lean 4	Argument	✓✓ dual	ETA: Q4 (no recursion)	ETA: H2
Jolt	RISC-V	a16z	✓ MIT	ETA: Q3 (no streaming)	ETA: H2
Keth (Cairo)	Cairo ISA	Kakarot	✓✓ dual	ETA: Q2 (no continuations)	ETA: Q2
Linea EVM	EVM	Linea	✓✓ dual	ETA: Q4 (no MPT)	ETA: H2
Miden VM	Miden ISA	Miden	✓ MIT	ETA: Q4 (no continuations)	ETA: H2
Nexus zkVM 3.0	RISC-V	Nexus	BUSL 1.1	ETA: Q3 (no recursion)	ETA: H2
Nock VM	Nock ISA	Zorp	✓ MIT	ETA: 2026	ETA: 2026
o1VM	RISC-V	O(1) Labs	✓ Apache 2.0	ETA: Q3 (no recursion)	ETA: H2
OpenVM	RISC-V	Axiom	✓✓ dual	✓	ETA: Q2
Petra	Petra ISA	Irreducible	✓ Apache 2.0	ETA: Q3 (no recursion)	ETA: H2
Pico	RISC-V	Brevis	✓✓ dual	✓	ETA: Q2
powdrVM	RISC-V	powdr	✓✓ dual	ETA: Q3 (no recursion)	ETA: H2
R0VM	RISC-V	RISC Zero	✓ Apache 2.0	✓	ETA: Q2
SP1	RISC-V	Succinct	✓✓ dual	✓	✓ live
SP1 Hypercube	RISC-V	Succinct	unlicensed	✓	ETA: Q2
zkEngine	WASM	ICME	✓✓ dual	ETA: Q2	ETA: H2
ZisK	RISC-V	Polygon	✓✓ dual	✓	ETA: Q2
zkMIPS	MIPS	ZKM	✓✓ dual	✓	✓ live
zkWASM	WASM	Delphinus	✓✓ dual	✓	ETA: Q2
Ligetron	WASM	Ligero	ETA: Q2	ETA: Q3 (no recursion)	ETA: H2
[redacted]	RISC-V	[redacted]	ETA: Q3	ETA: Q4	ETA: H2
StarkV	RISC-V	StarkWare	ETA: Q4	ETA: Q4 (not started)	ETA: H2
Valida	Valida ISA	Lita	ETA: Q3	ETA: Q3 (no recursion)	ETA: H2

Message *@ethproofs_community* on Telegram for additions and corrections. ETAs are best guesses.

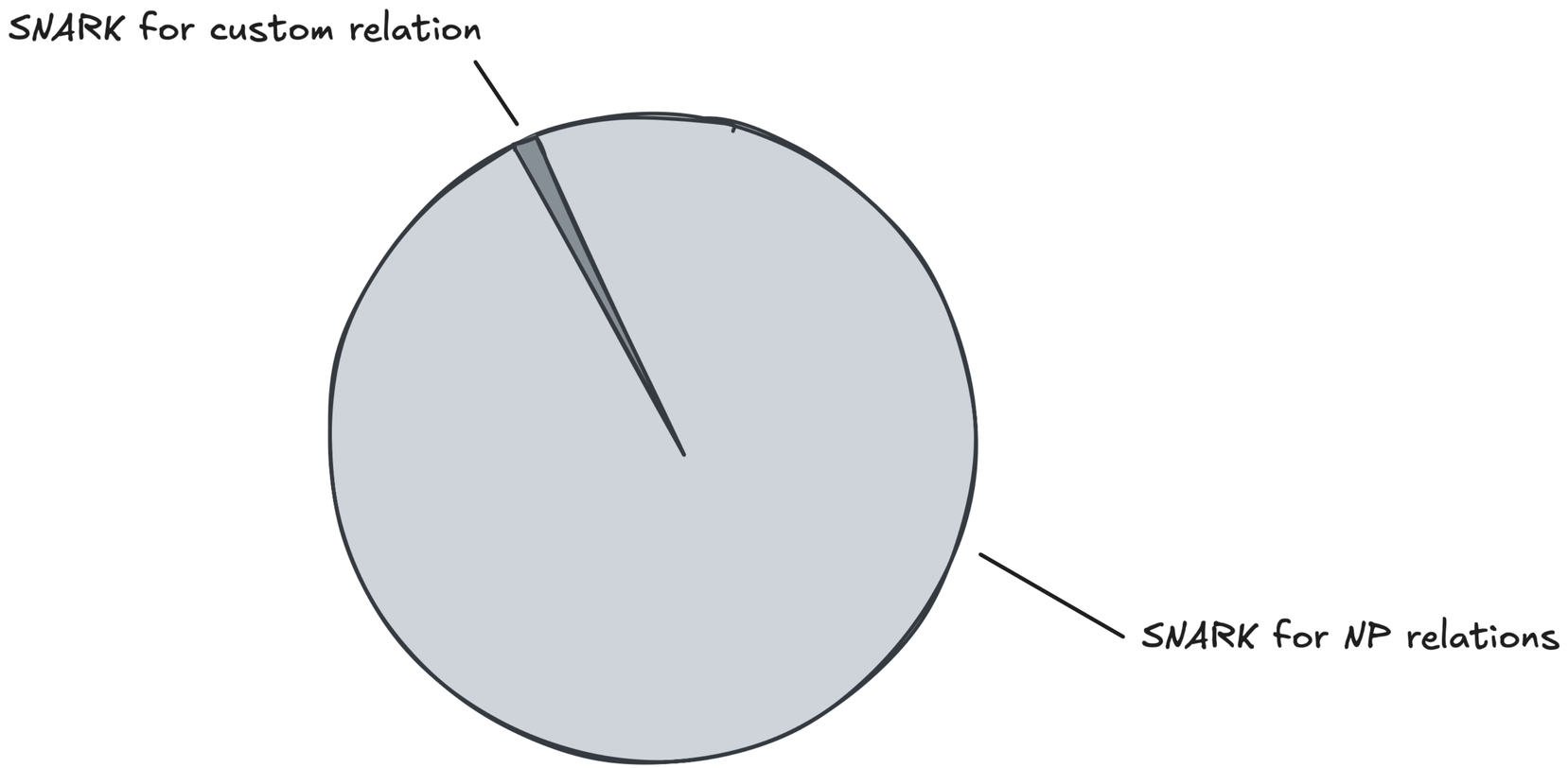
ethproofs: proving Ethereum blocks



libZK: selective disclosure for government credentials

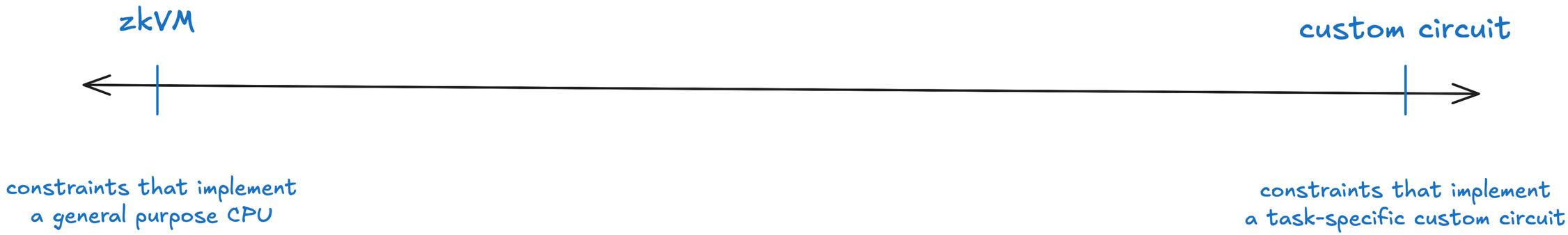
How are SNARKs deployed?

General purpose SNARKs

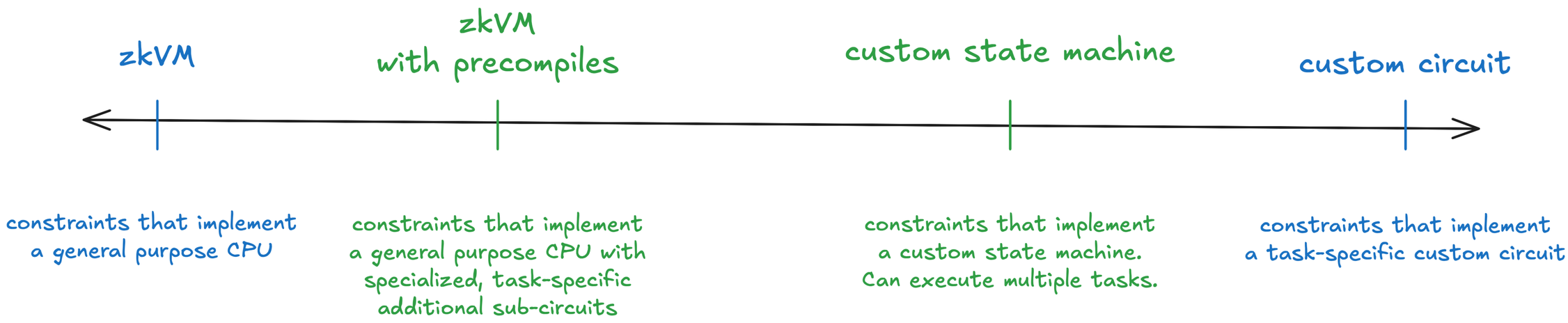


* not based on real data, but the point still stands

What kind of circuit?



What kind of circuit?



ethproofs

libZK

Focus: zkVMs

What ISA?

zkvm	ISA	team	open source	proves mainnet blocks	Ethproofs
Airbender	RISC-V	MatterLabs	✓✓ dual	✓	ETA: Q2
Ceno	RISC-V	Scroll	✓✓ dual	ETA: Q4 (no recursion)	ETA: H2
Euclid (OpenVM)	RISC-V	Scroll	✓✓ dual	✓	ETA: Q2
Ix	Lean 4	Argument	✓✓ dual	ETA: Q4 (no recursion)	ETA: H2
Jolt	RISC-V	a16z	✓ MIT	ETA: Q3 (no streaming)	ETA: H2
Keth (Cairo)	Cairo ISA	Kakarot	✓✓ dual	ETA: Q2 (no continuations)	ETA: Q2
Linea EVM	EVM	Linea	✓✓ dual	ETA: Q4 (no MPT)	ETA: H2
Miden VM	Miden ISA	Miden	✓ MIT	ETA: Q4 (no continuations)	ETA: H2
Nexus zkVM 3.0	RISC-V	Nexus	BUSL 1.1	ETA: Q3 (no recursion)	ETA: H2
Nock VM	Nock ISA	Zorp	✓ MIT	ETA: 2026	ETA: 2026
o1VM	RISC-V	O(1) Labs	✓ Apache 2.0	ETA: Q3 (no recursion)	ETA: H2
OpenVM	RISC-V	Axiom	✓✓ dual	✓	ETA: Q2
Petra	Petra ISA	Irreducible	✓ Apache 2.0	ETA: Q3 (no recursion)	ETA: H2
Pico	RISC-V	Brevis	✓✓ dual	✓	ETA: Q2
powdrVM	RISC-V	powdr	✓✓ dual	ETA: Q3 (no recursion)	ETA: H2
R0VM	RISC-V	RISC Zero	✓ Apache 2.0	✓	ETA: Q2
SP1	RISC-V	Succinct	✓✓ dual	✓	✓ live
SP1 Hypercube	RISC-V	Succinct	unlicensed	✓	ETA: Q2
zkEngine	WASM	ICME	✓✓ dual	ETA: Q2	ETA: H2
ZisK	RISC-V	Polygon	✓✓ dual	✓	ETA: Q2
zkMIPS	MIPS	ZKM	✓✓ dual	✓	✓ live
zkWASM	WASM	Delphinus	✓✓ dual	✓	ETA: Q2
Ligetron	WASM	Ligero	ETA: Q2	ETA: Q3 (no recursion)	ETA: H2
[redacted]	RISC-V	[redacted]	ETA: Q3	ETA: Q4	ETA: H2
StarkV	RISC-V	StarkWare	ETA: Q4	ETA: Q4 (not started)	ETA: H2
Valida	Valida ISA	Lita	ETA: Q3	ETA: Q3 (no recursion)	ETA: H2

Message *@ethproofs_community* on Telegram for additions and corrections. ETAs are best guesses.

Focus: custom circuits

Which DSLs are we using?

DSL	# Repos	# Repos with pushes since 1 Jan, 2025
Circom	1197 (incl. 632 forks)	277 (incl. 153 forks)
Noir	489 (incl. 186 forks)	342 (incl. 144 forks)

- Other DSLs but harder to search for: Halo2, gnark, arkworks, SnarkyJS, Cairo
- For more data, see github.com/ArmanKolozyan/ZKP-Languages

Data collected on 11 June, 2025 from the Github search API.

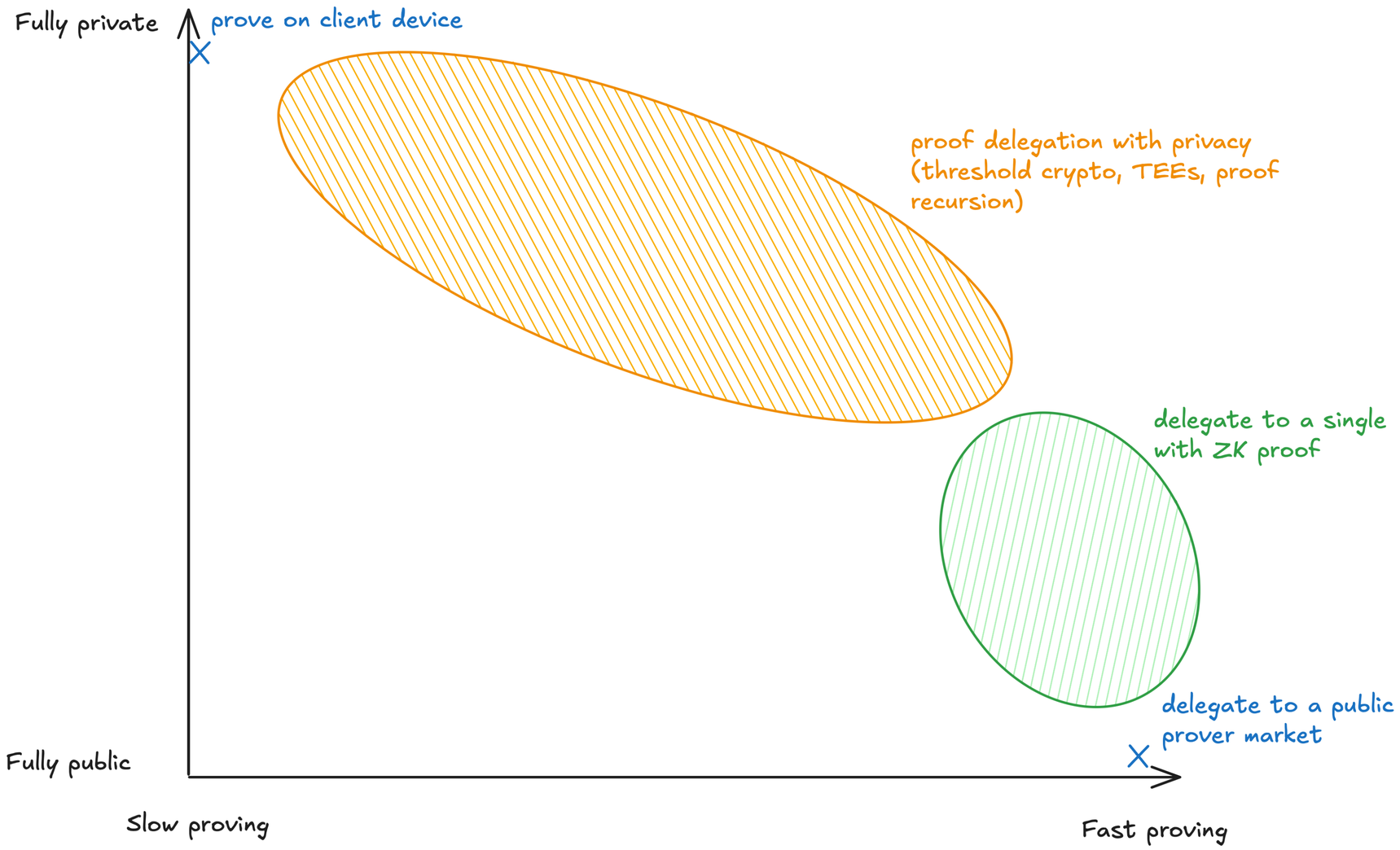
Who is proving?

- Client-side (mobile phones, laptops, browsers)
- Private delegation (TEE, MPC networks, proof recursion)
- Delegation to a single server
- Delegation to a prover market

Privacy vs speed for ZK proofs

Privacy vs speed for ZK proofs

Privacy vs speed for ZK proofs



What SNARKs are deployed?

Performance criteria and requirements

- Prover resources
- Verifier resources
- Proof size
- Security model

A large design space

Hash functions, curves or lattices

Monolithic or piecemeal SNARK (recursion, folding, IVC)

Univariate or multilinear PIOP

Arithmetization and commitments costs

SNARK-friendly or standard hash functions

Zero-knowledge

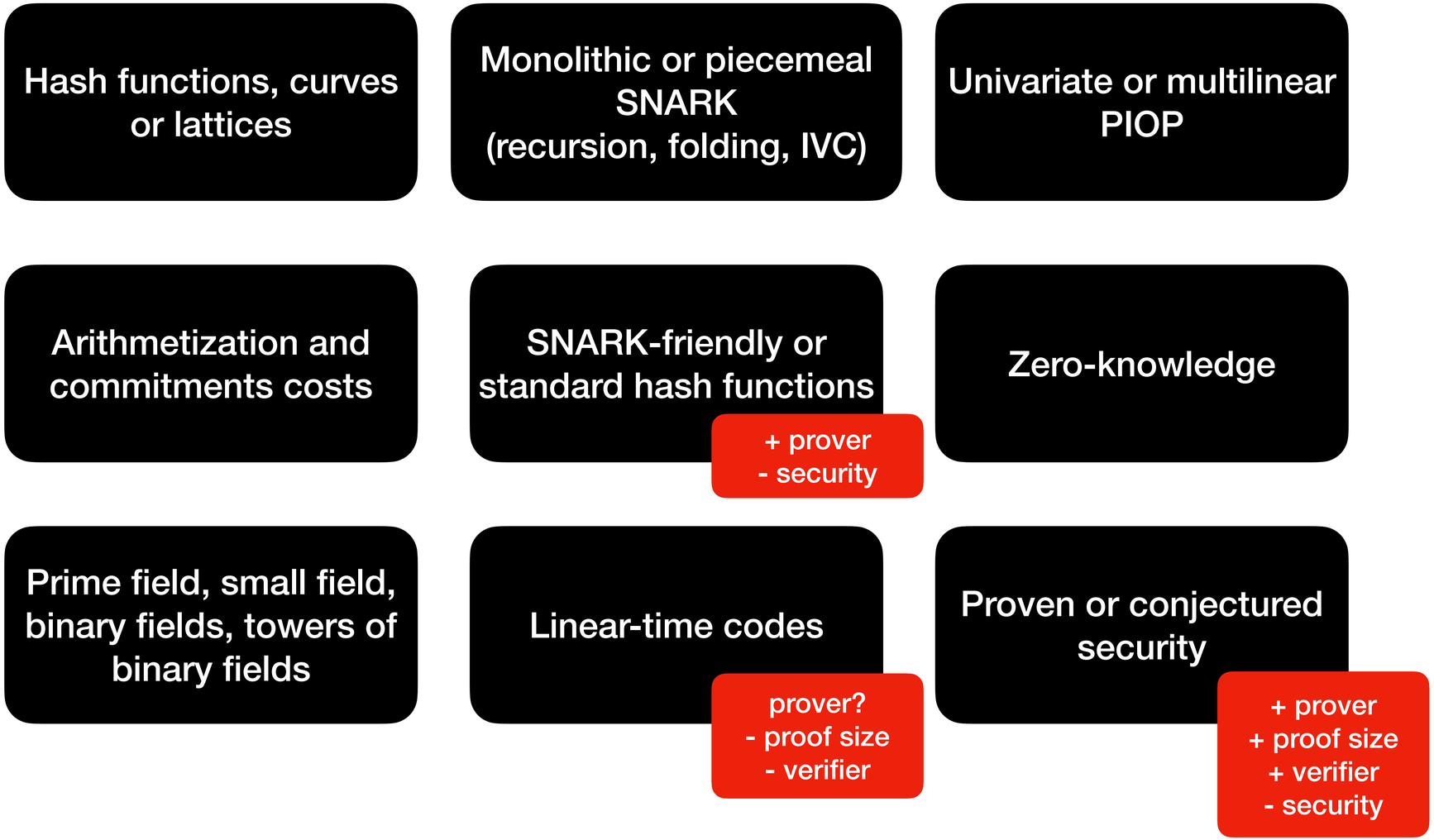
Prime field, small field, binary fields, towers of binary fields

Linear-time codes

Proven or conjectured security

… and many more

Some quick evaluations



Hash functions, curves or lattices

Overview

Cryptographic tool	Prover resources	Verifier resources	Proof size	Security model
Hash functions	very fast	fast	ok	post-quantum secure, proven security only in (Q)ROM
Elliptic curves	ok	fast	tiny	pre-quantum, proven security in ROM+DLOG+assumptions or knowledge assumptions
Lattices	fast	TBD	small	post-quantum secure, (Q)ROM + lattice assumptions

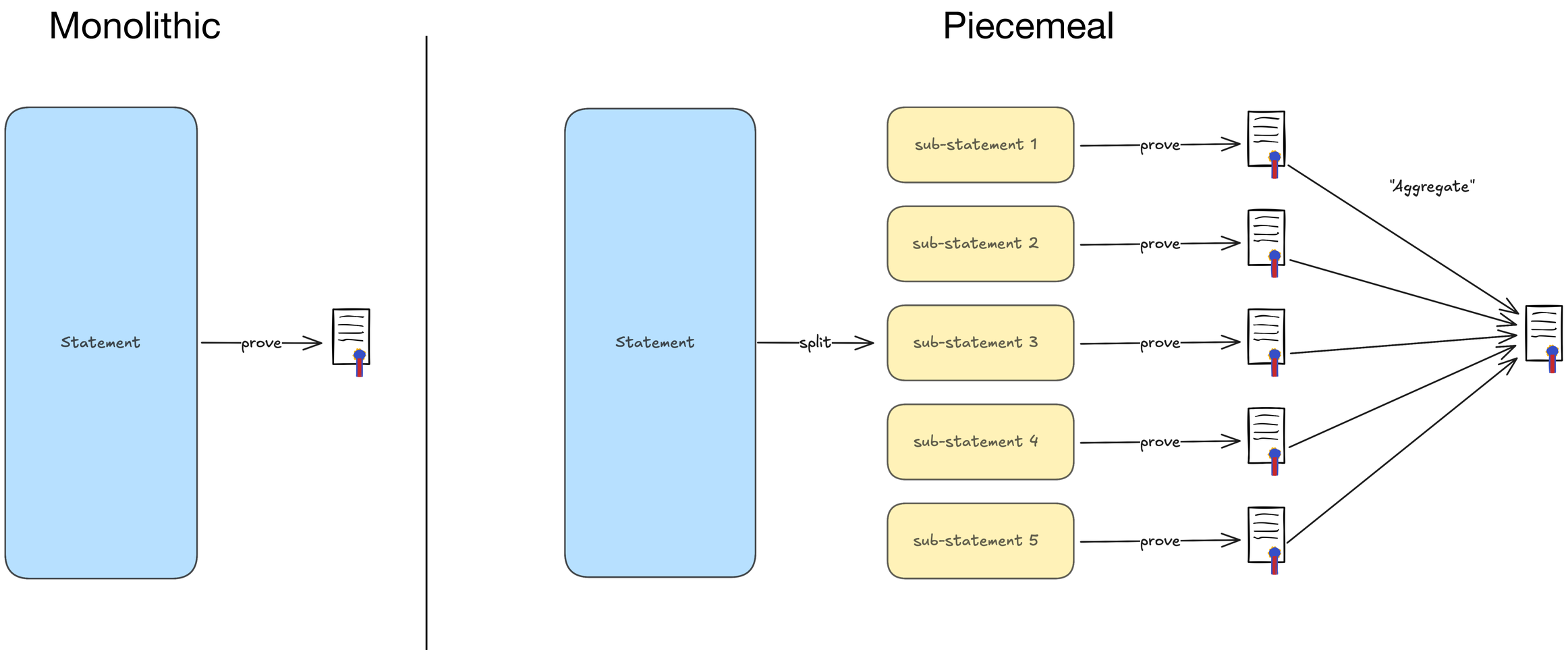
Hash functions, curves or lattices

Quantum threat or quantum hype?

- Known unknowns: when will we have quantum computers?
- Known knowns:
  - quantum only breaks soundness, zero-knowledge is preserved. In other words, **no harvest-now-decrypt-later attacks**.
  - most blockchains still rely on elliptic curve signatures.
  - most government issued ID’s still rely on RSA.

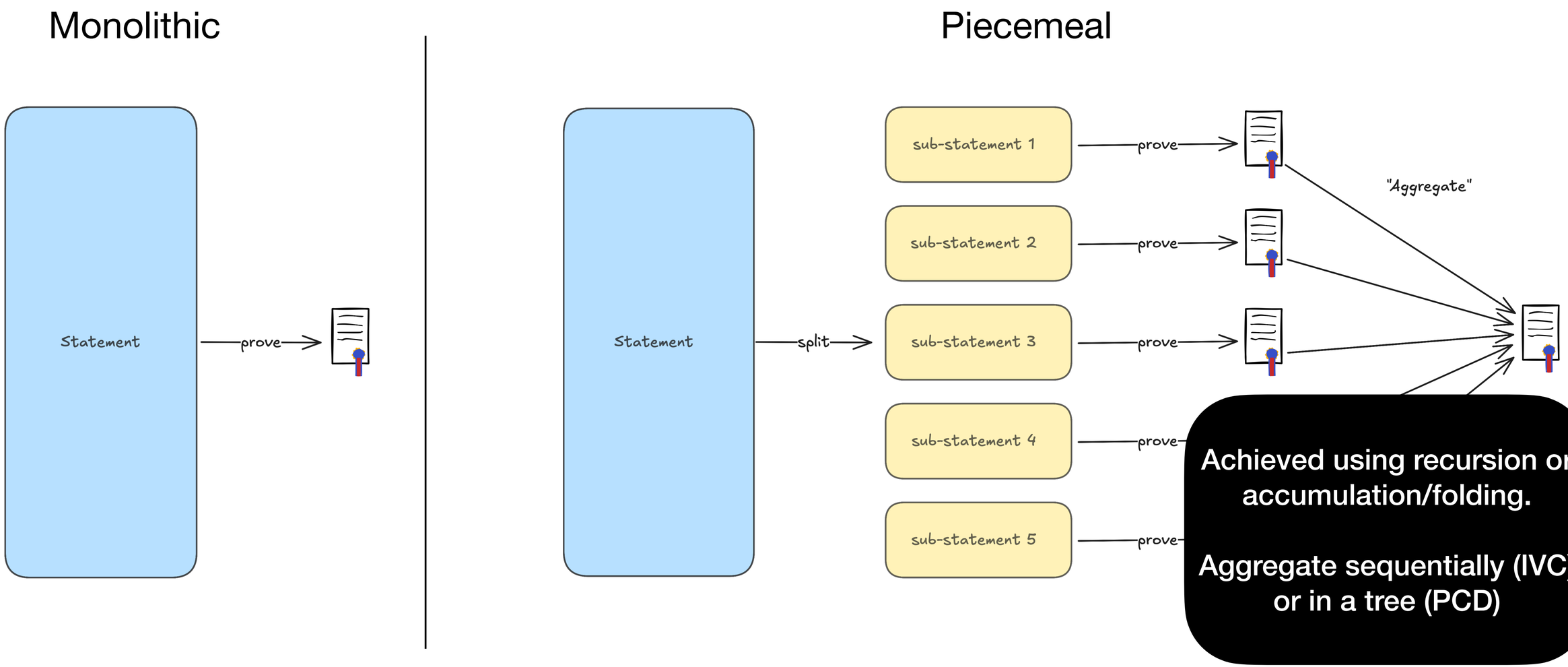
Monolithic vs Piecemeal SNARKs

High-level description



Monolithic vs Piecemeal SNARKs

High-level description



Monolithic vs Piecemeal SNARKs

Performance comparison

Monolithic SNARK	Piecemeal SNARK
	Treat sub-statements in parallel or streaming
	Sub-statements require fewer resources to prove
Simple security analysis	Limited security analysis, heuristic instantiation of the random oracle

+ prover
- security

Univariate or multilinear PIOP

Why multilinear polynomials are in vogue

- Efficient evaluation at random point:

$$\tilde{f}(X) = \sum_{x_1,\ldots,x_m \in \{0,1\}} f(x_1, \ldots, x_m) \cdot eq(X, x_1, \ldots, x_m)$$

extension of f
at any point.

sum of the evaluations of f times something
easy to compute.

sumcheck statement!

- Commitment costs are comparable to those for univariate polynomials.

+ prover
≈ proof

Choice of field

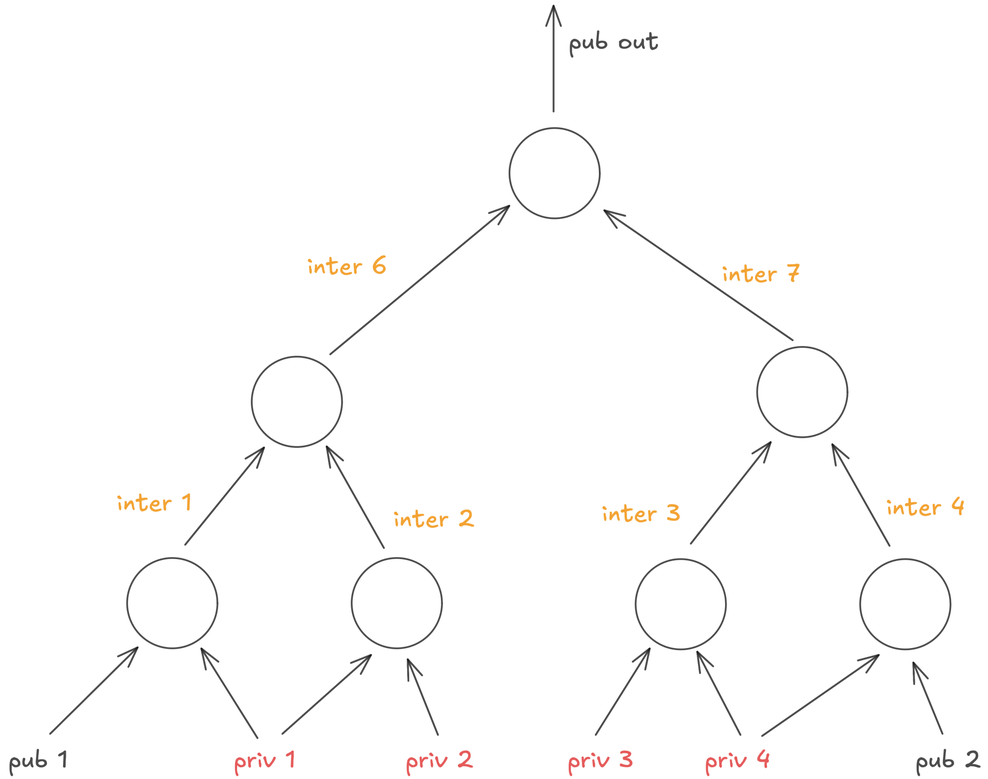
Prime field, small fields, binary fields or towers of binary fields

- Large fields are inherent to our PIOP. Question: is it prime, or an extension field?
- Large prime fields:
  - inherent to elliptic curve cryptography.
  - good for signature verification (and any protocol that uses the large prime field)
- Small primes + extension: efficient hardware implementation.
- Binary fields + extension: great at expressing bit- or byte-wise operations.
- Towers of binary fields: same as binary field with additional flexibility.

+ prover

Arithmetization and commitment costs

- One of prover's main task is committing to witness data.
- Different ways of expressing statements yield smaller or larger witness.
- In general, we commit to secret inputs and intermediate values (Plonk, AIR, R1CS, CCS).
- GKR allows commitment to secret inputs only.
- However, recent attack on FS-GKR (!): large circuits might contain backdoors.



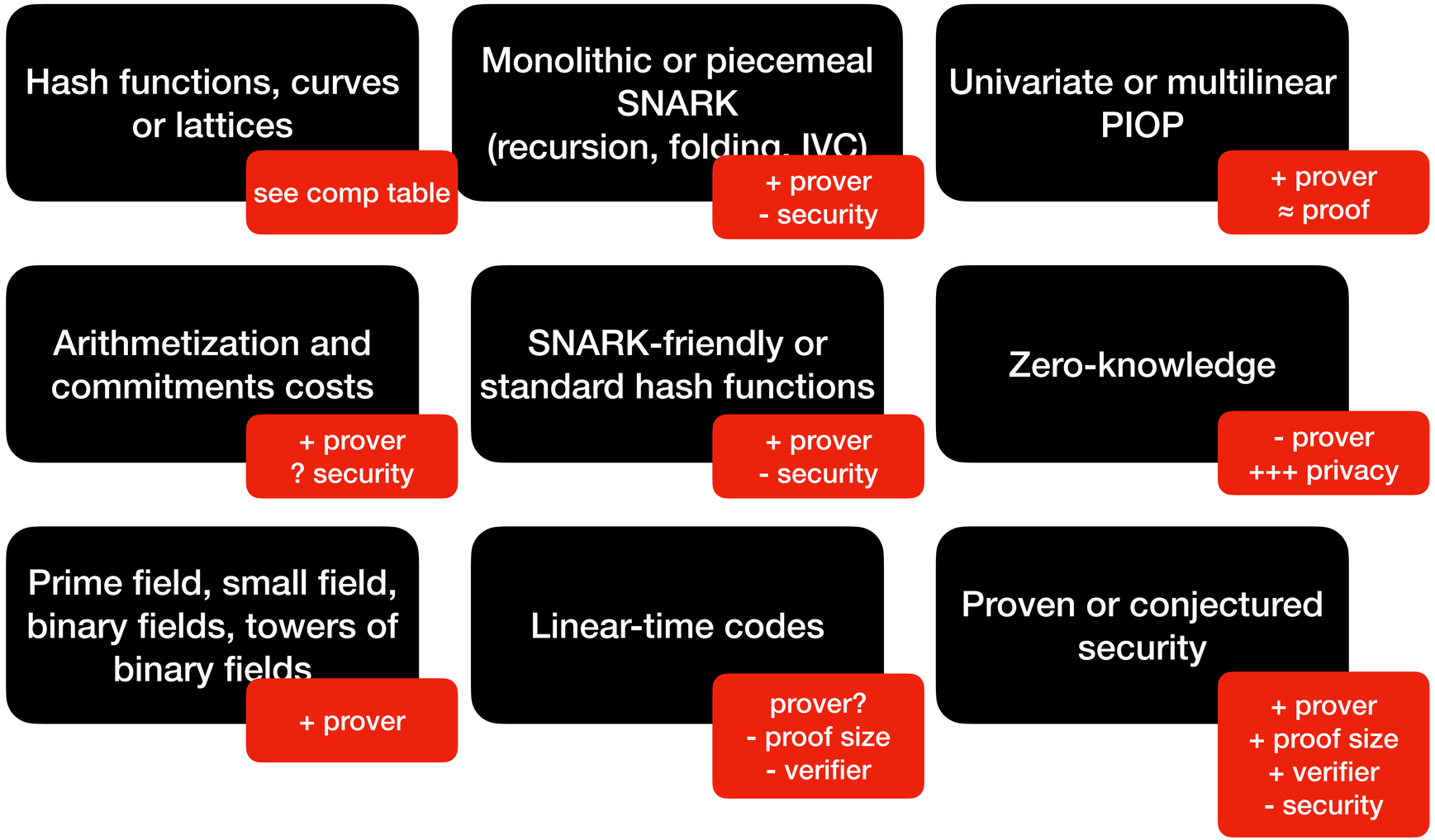
+ prover
? security

Zero-knowledge

- ZK is absent from most libraries, present in all marketing.
- Mandatory for privacy (see ZK Hack V puzzle 1).
- Two general methods:
  - make a zk-variant of your proof system.
  - wrap your proof system in a ZK proof.

- prover
+++ privacy

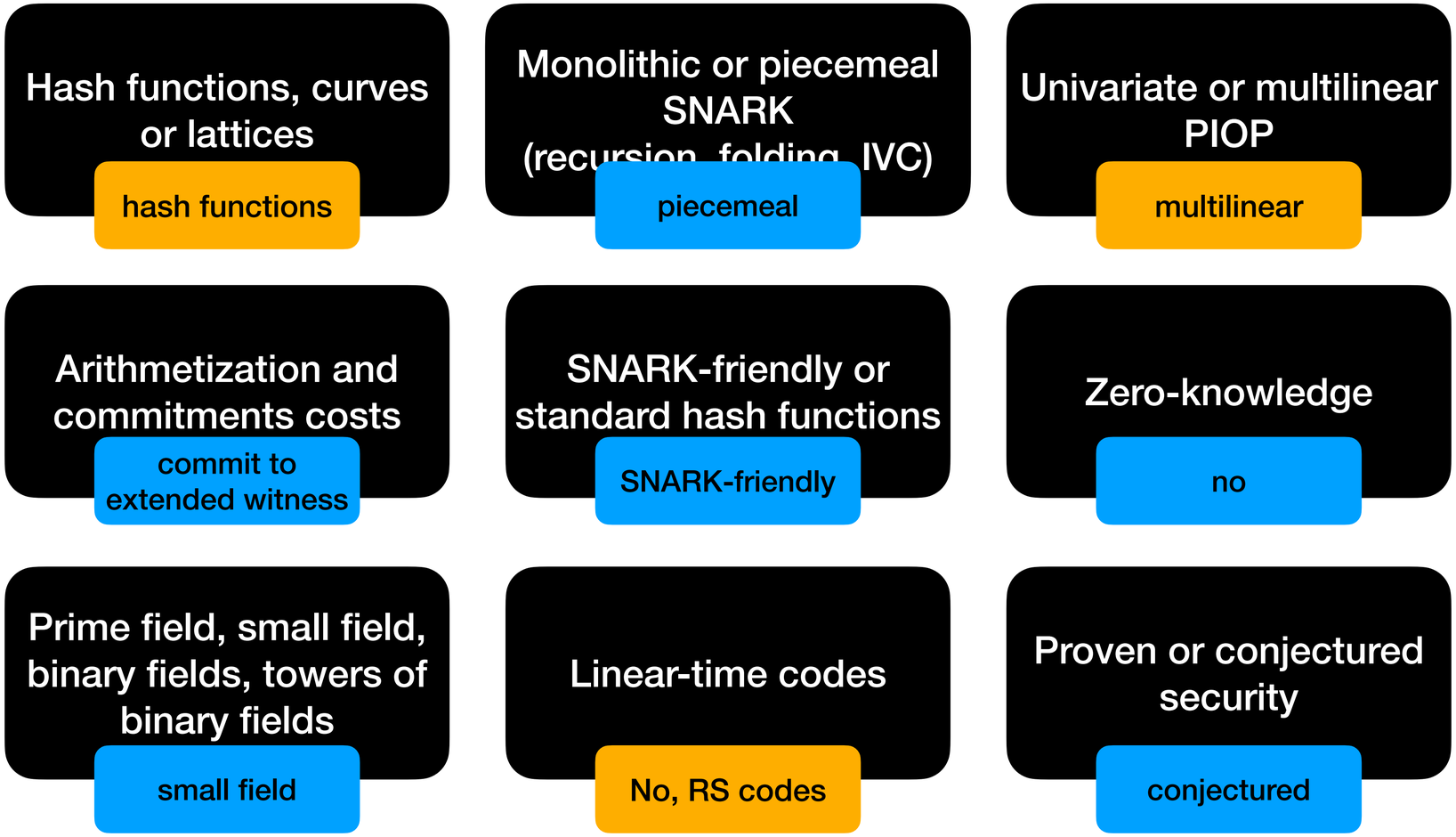
Completing our evaluations



Choices in libZK

Hash functions, curves or lattices hash functions	Monolithic or piecemeal SNARK (recursion, folding, IVC) monolithic	Univariate or multilinear PIOP multilinear
Arithmetization and commitments costs GKR	SNARK-friendly or standard hash functions standard	Zero-knowledge yes
Prime field, small field, binary fields, towers of binary fields Dual: prime & binary	Linear-time codes No, RS codes	Proven or conjectured security proven

Choices in SP1 Hypercube



State of the art in 2026?

My wish list

- More exploration of lattice-based SNARKs.
- More scrutiny on security of piecemeal SNARK.
- More scrutiny on security of SNARK-friendly hash functions.
- More scrutiny on error-correcting codes and related conjectures.
- Zero-knowledge as an option in all libraries.