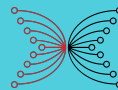


ProtogaLattice

Lattice-based folding

Sept 2026

Anca Nitulescu



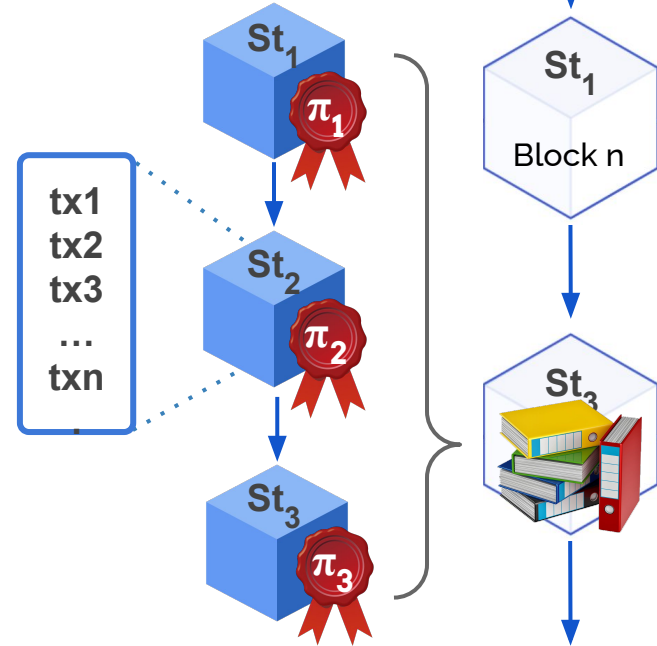
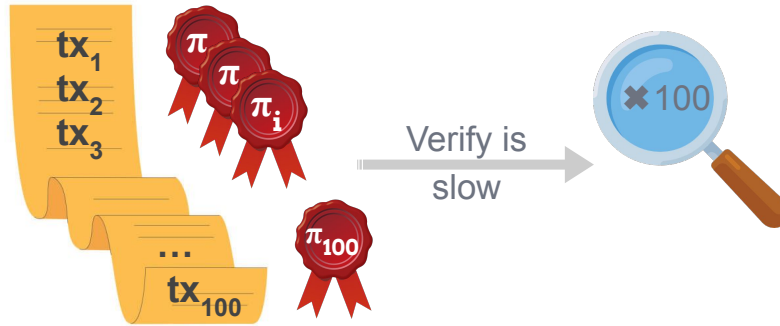
INPUT | OUTPUT

Blockchain scaling

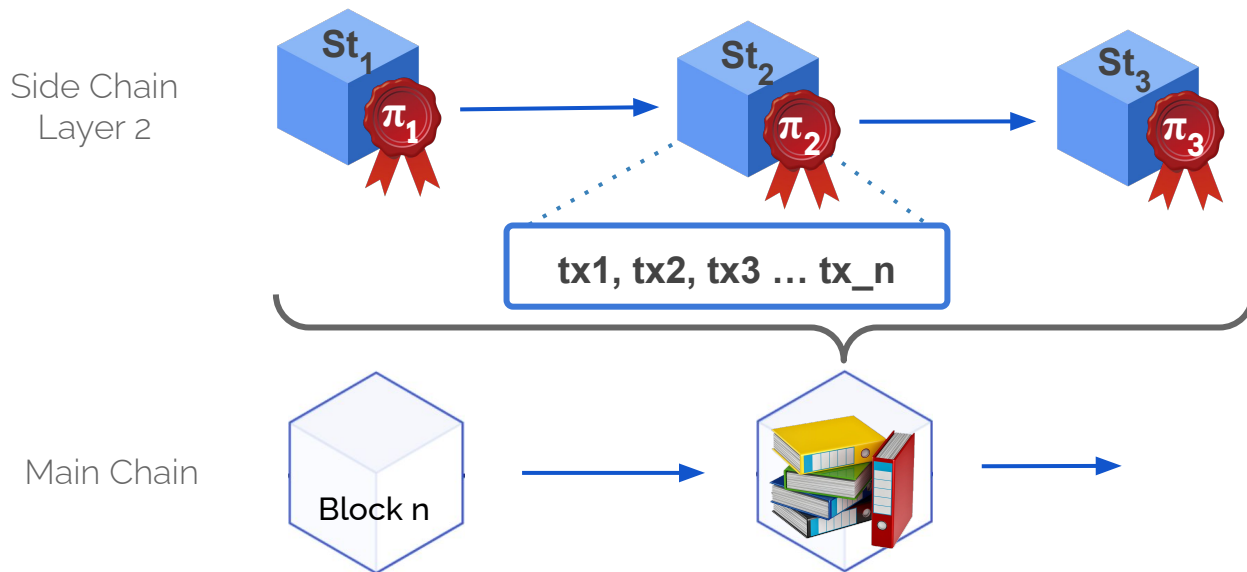


ZK Roll-ups

Periodically provide proofs for valid transaction batches



Prove & scale ?



Motivation: scaling

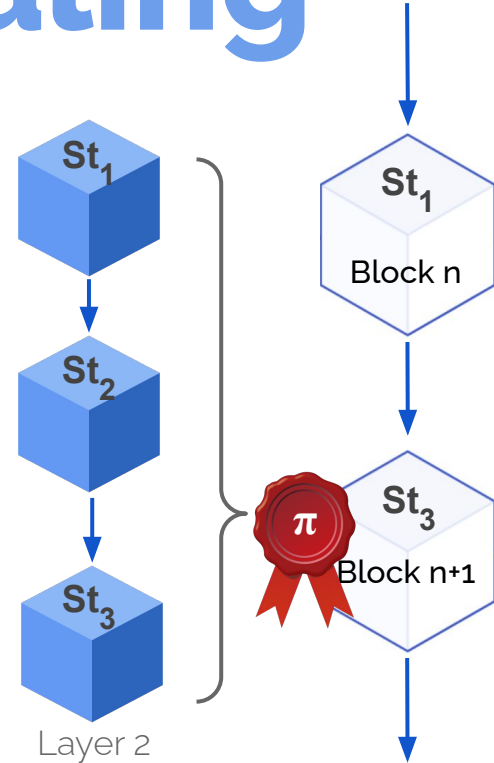


The Problem - Scaling zk-Rollups

Rollups need to aggregate thousands of proofs efficiently

Current state:

- **Monolithic proving:** $O(N)$ prover time for N transactions
- **Proof aggregation** (SnarkPack, FLIP): reduces verification, but prover still heavy
- **Recursive SNARKs:** expensive, recursion step encodes verifier, can require cycles of curves
- **Folding?**

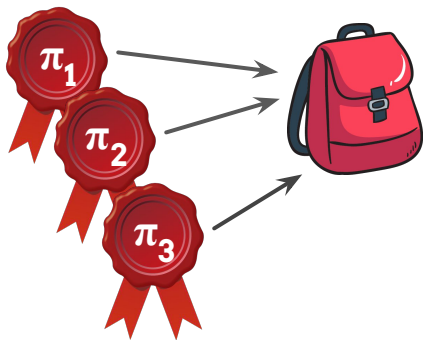




Prove many instances



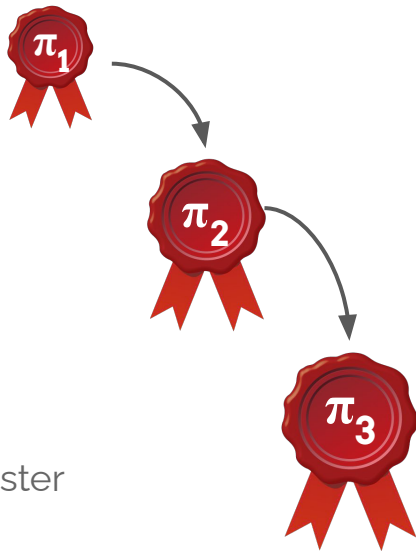
Aggregation



Fast



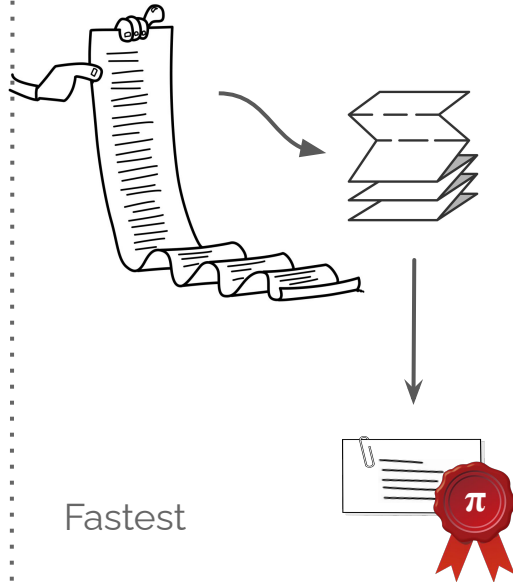
Recursion



Faster



Folding



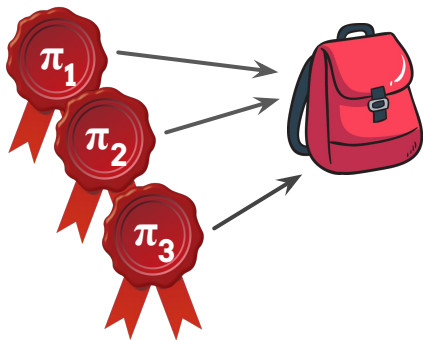
Fastest



Prove many instances



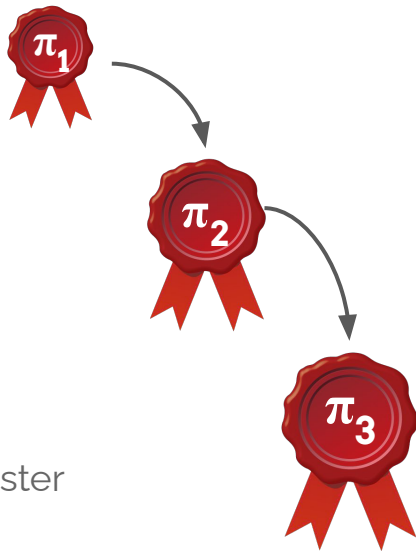
Aggregation



Fast



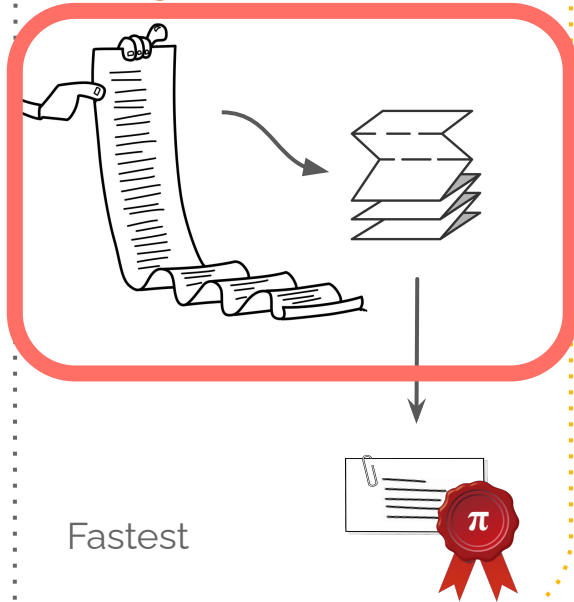
Recursion



Faster

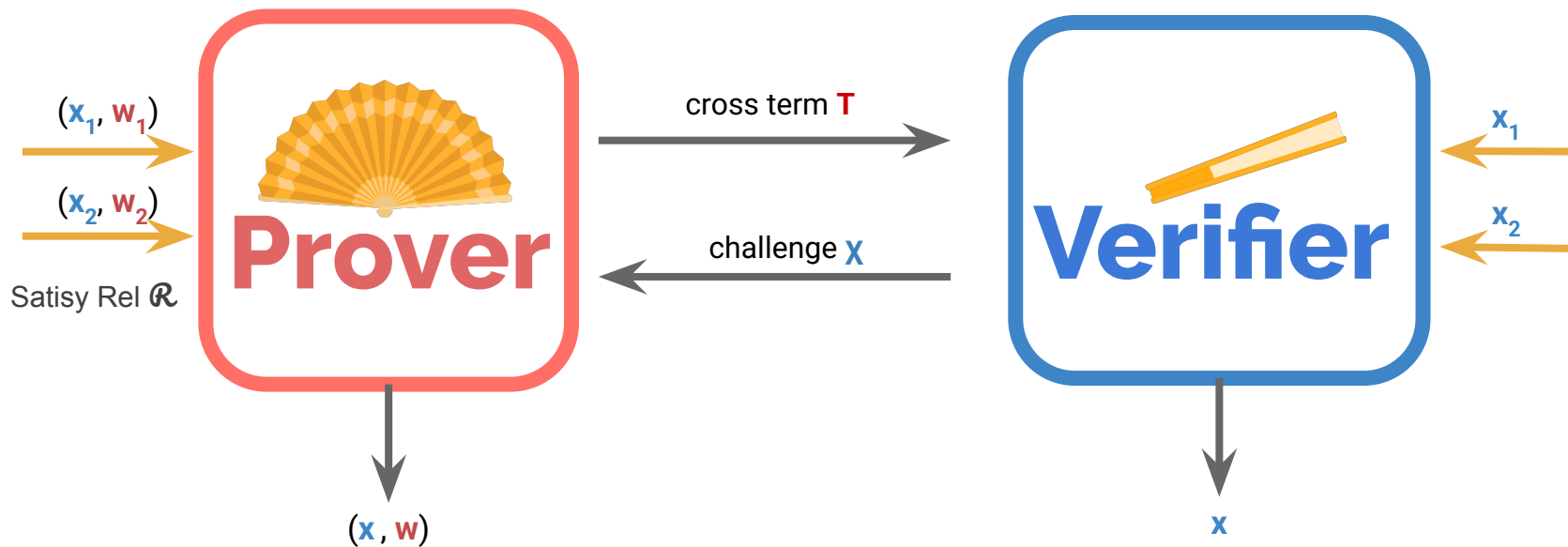


Folding



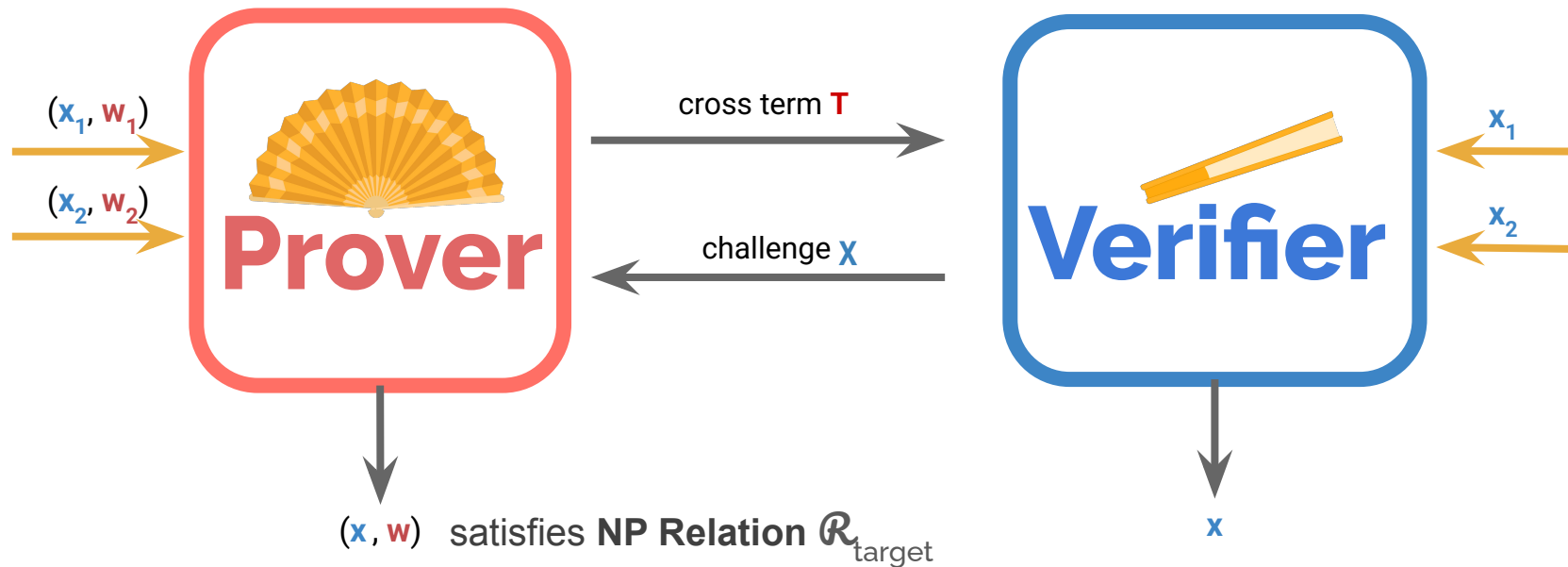
Fastest

Folding Scheme



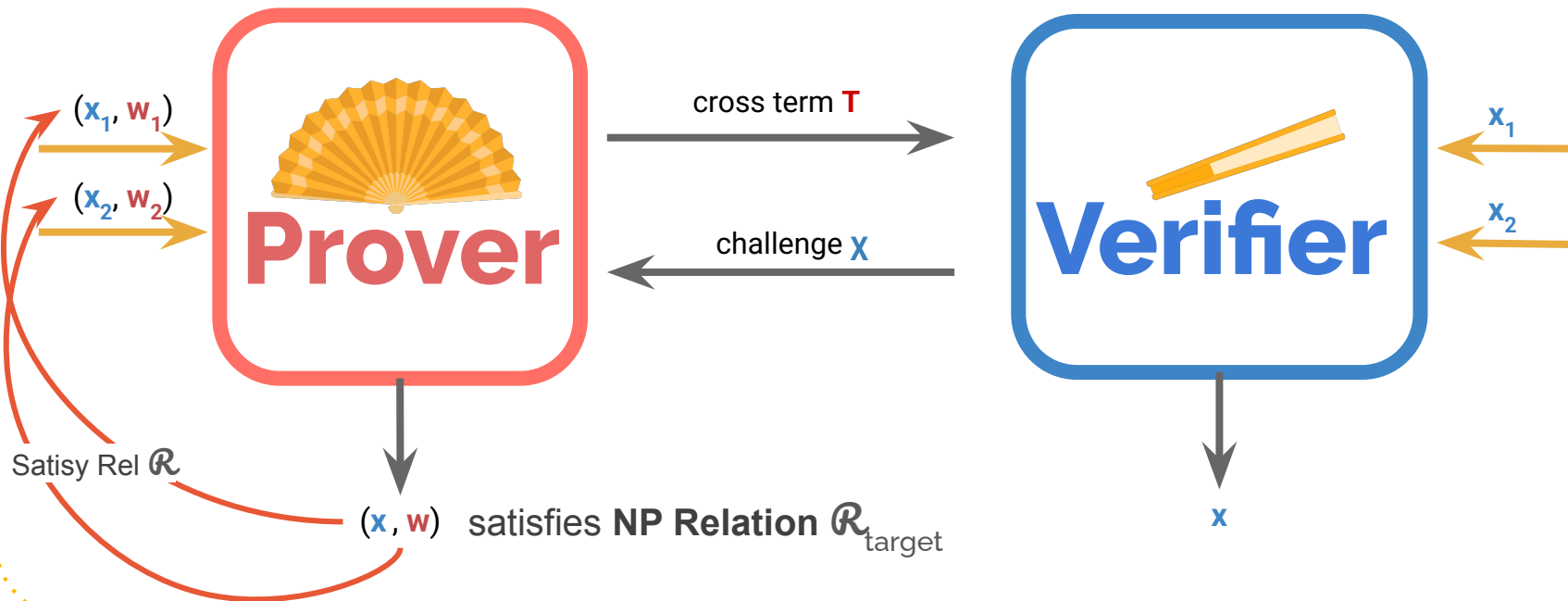


Folding Soundness

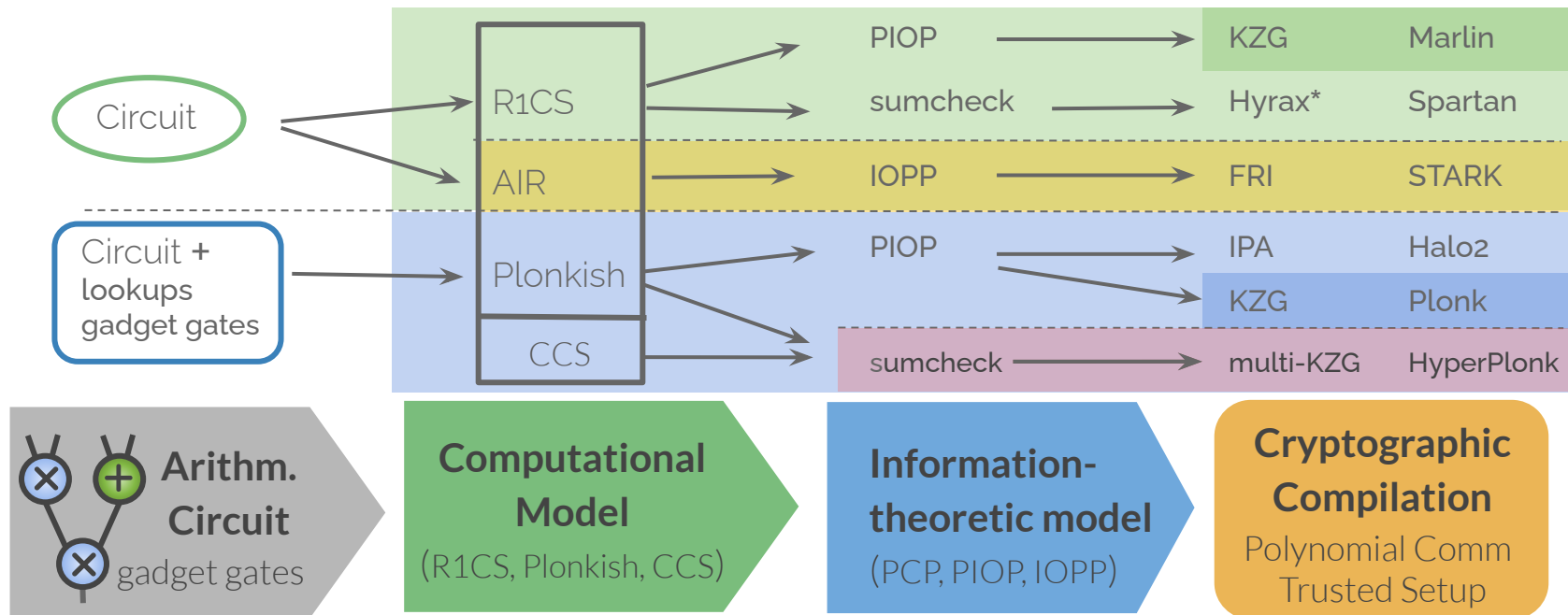




Folding Soundness

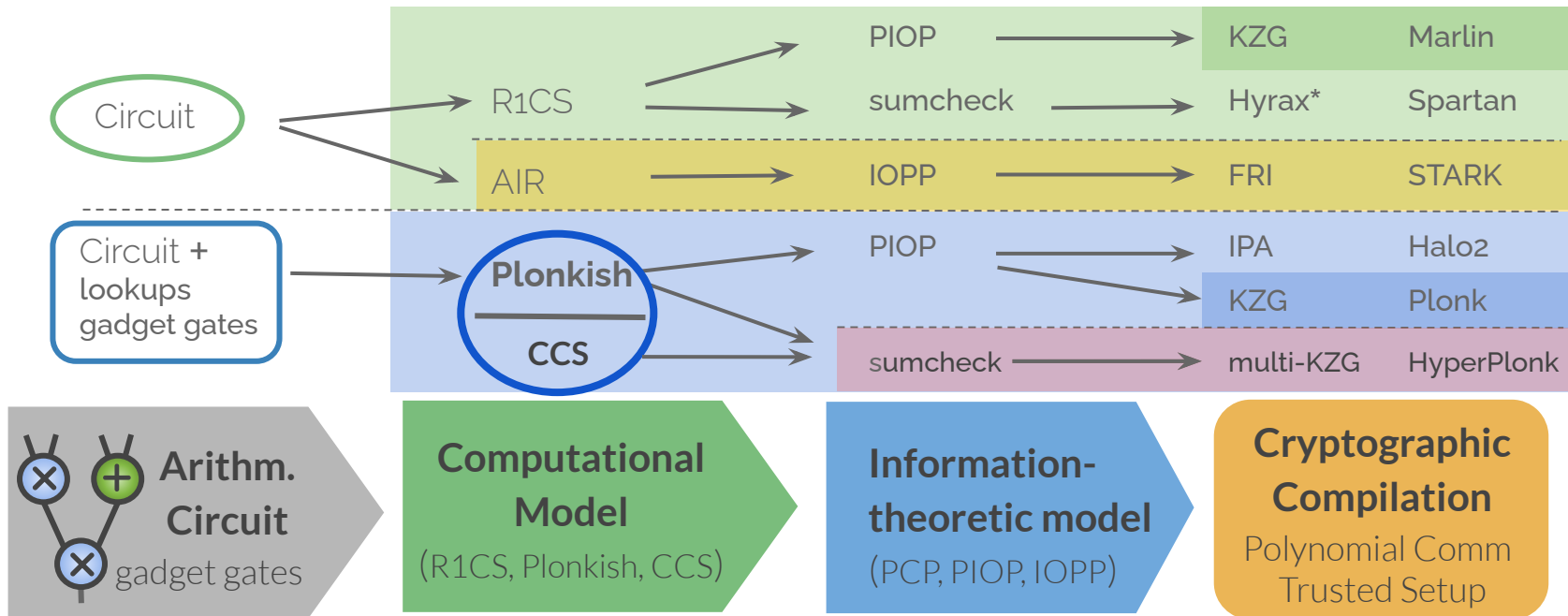
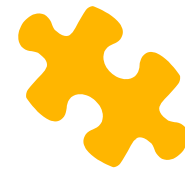


SNARKs Relations



*SPARK PolyCommit for sparse multilinear polynomials

SNARKs Relations



*SPARK PolyCommit for sparse multilinear polynomials

Folding Schemes



Nova - for R1CS

- Folding instances & witness
- **relaxed-R1CS** arithmetization
- Transparent/Universal setup
- Any plain cycle of elliptic curves
- The prover sends cross-terms
- One single final proof
- Needs a Commit-and-Prove SNARK for cr-R1CS



Folding Schemes



Sangria – Nova for Plonk arithmetization

- verifier's work is constant in the depth of the circuit
- constants are worse than in Nova

SuperNova – generalized Nova

- different functions at every step
- the cost of proving a step is proportional only to the circuit size for that step

HyperNova – folding with sum-checks

- customizable constraint systems (CCS)
- supports lookups
- sum-check protocol for high-degree constraints polynomial



ProtoStar – Improves HyperNova

- supports lookups with smaller overhead
- sum-check protocol is not required.

More Expressive $\mathcal{R}$



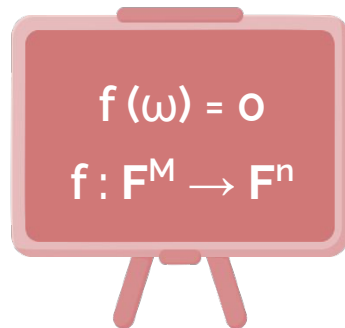
ProtoGalaxy - folding polynomial relations

- folding scheme for the relations $R^{\text{rand}} \times R^k \rightarrow R^{\text{rand}}$, where

$$R = R_f = \{(\varphi, \omega) : \varphi = \text{cm}(\omega) \wedge f(\omega) = 0\}$$

for some degree-d polynomial-map $f : F^M \rightarrow F^n$.

- cm = an additively homomorphic commitment scheme
- The paper does not show a **Decider** for the final relation R^{rand}



More Expressive $\mathcal{R}$



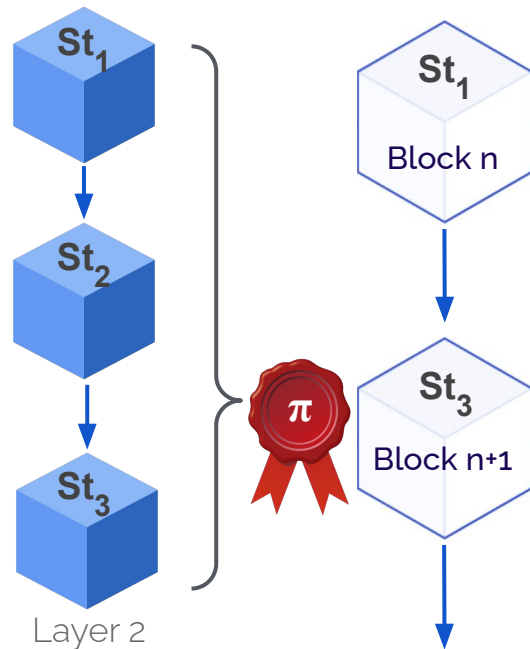
ProtoGalaxy - folding polynomial relations

- folding scheme for the relations $R^{\text{rand}} \times R^k \rightarrow R^{\text{rand}}$, where

$$R = R_f = \{(\varphi, \omega) : \varphi = \text{cm}(\omega) \wedge f(\omega) = o\}$$

for some degree-d polynomial-map $f : F^M \rightarrow F^n$.

- cm = an additively homomorphic commitment scheme
- The paper does not show a **Decider** for the final relation R^{rand}





Folding Paradigms

Algebraic vs. Sumcheck-based folding

- Algebraic:**
- closed algebraic transformation on the relation itself,
 - involves evaluating and linearly combining polynomial constraints on the relation witnesses
- Sumcheck:**
- phrase the folding as a claim about a big polynomial (or products of multilinear polynomials)
 - use the sumcheck protocol to reduce the claim to a small number of scalar checks

Algebraic Folding

Fold via random linear combination

Pros: Constant rounds, small proofs, expressive relations: arbitrary polynomial constraints

Cons: Trusted setup (KZG), prover costs (polynomial evaluations, divisions)

Sumcheck Folding

Fold via sumcheck protocol

Pros: linear proving time, transparent setup

Cons: $O(\log N)$ rounds, larger proofs, more involved verifier, limited to linear/linearized relations

Incomplete Solution



The Problem - Scaling zk-Rollups

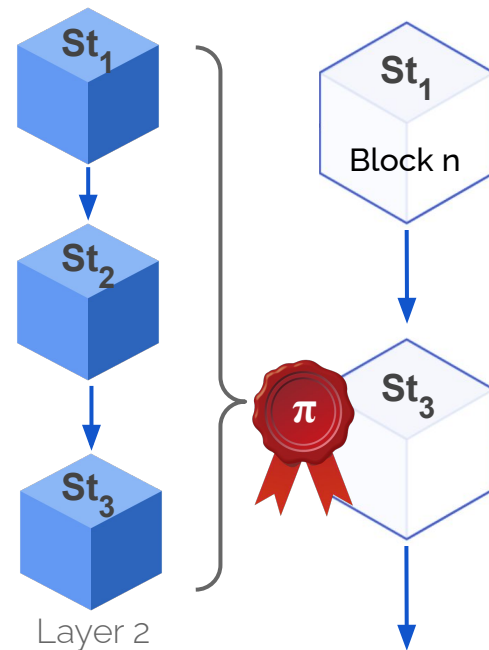
zkRoll-ups need to aggregate thousands of proofs efficiently

New solution: **Folding Schemes**

Key bottlenecks:

Existing folding schemes:

- no **Decider** (proof system) to verify the final instance
- Some early works (Nova, FLIP) construct a decider, but are R1CS-only
- State-of-the-art constructions: folding more expressive relations w/o a concrete decider (HyperNova, Protostar)
- No ZK by itself (need to have a final proof + hiding notion)

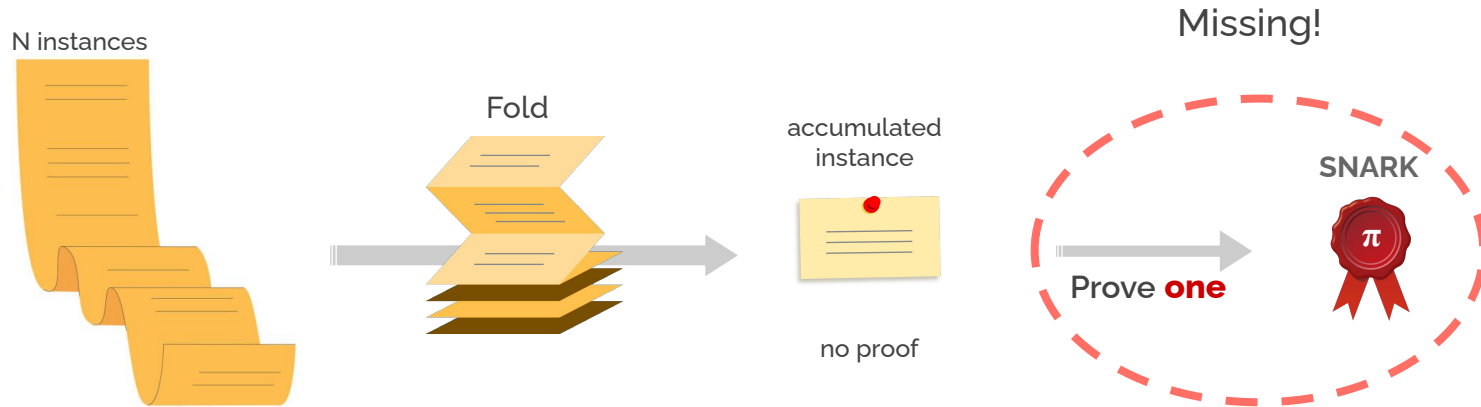


Missing Decider



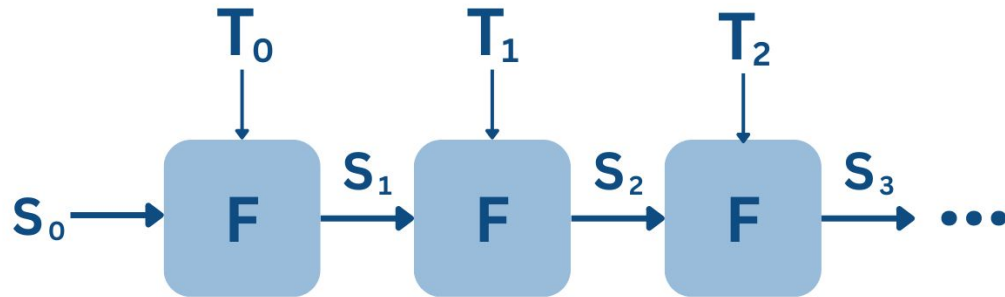
The Problem - Scaling zk-Rollups

New solution: **Folding Schemes**



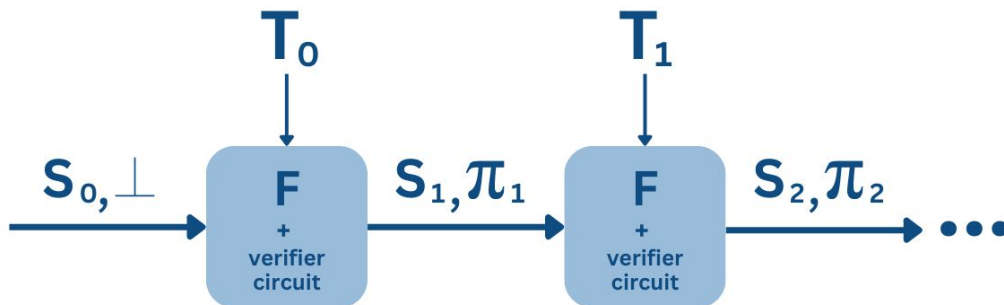
IVC

Incremental Verifiable Computation

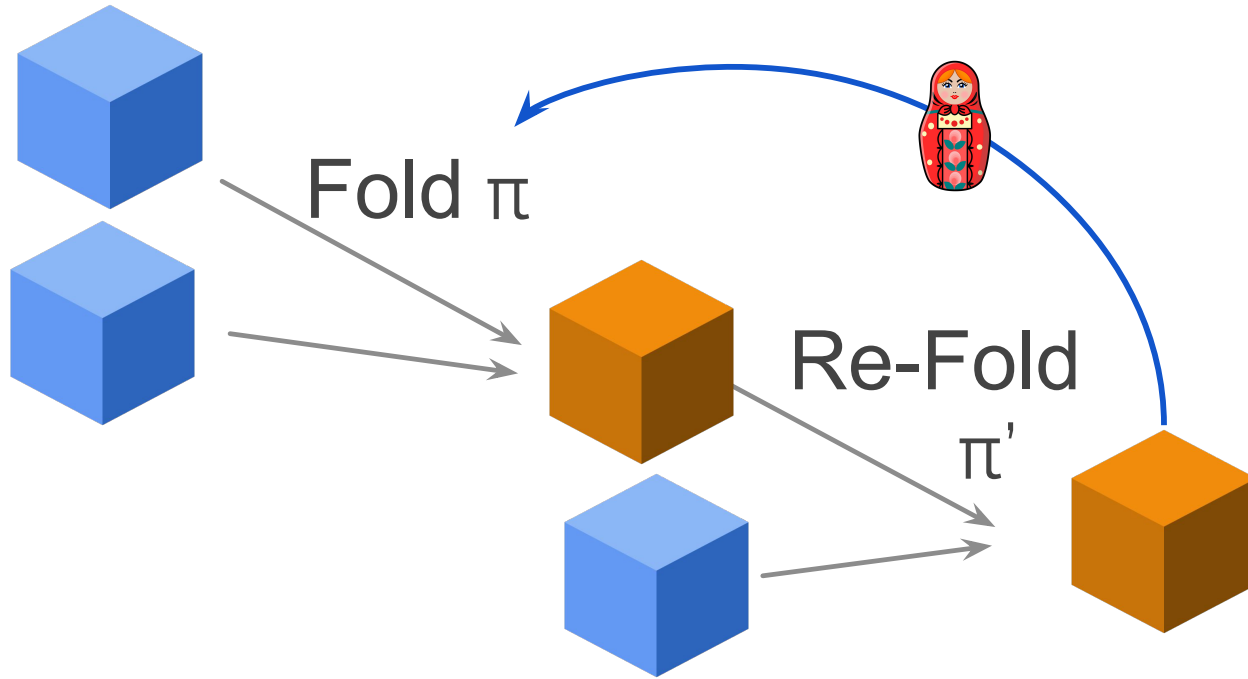


IVC

Incremental Verifiable Computation



Recursion via Folding



Recursion & Decider



Folding Schemes compatible with practical SNARKs relations:

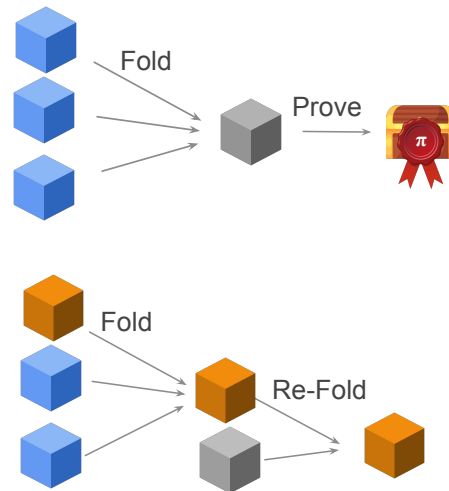
- **One-shot Folding**

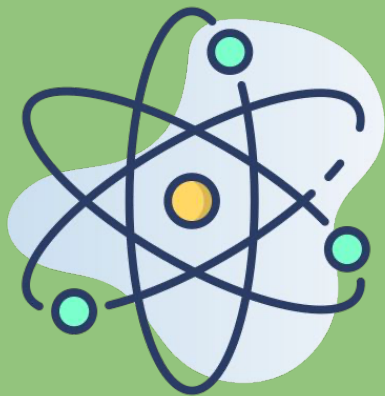
- scheme for expressive relations $R_o^k \rightarrow R_{rand}$
- R_{rand} has an efficient Decider

- **Self-Folding**

- scheme for flexible relations $R_{acc}^k \rightarrow R_{acc}$
- the Folding verification is compatible with R_{acc} (min rounds)
- allows for **Incremental Verifiable Computation** (IVC), PCD
- hard to construct a Decider

We would like both IVC & Decider





Lattice-based Folding

Lattice Folding



Lattice-fold

- inspired by HyperNova - **sumcheck Log round**
- assumption: Module Short Integer Solution (MSIS)
- **Atjai commitments** for the witness (**small norm** only)
- bit decomposition and range proofs

Lattice-fold+

- new purely algebraic range proof
- double commitments (commitments of commitments)

Neo

- avoids running Number Theoretic Transforms (NTTs)
- different encoding
- "pay-per-bit" commitment scheme (efficient for small)



Main challenges:

- managing norm growth after folding – **range proofs**
- embedding fields from CCS into polynomial rings in MSIS

Lattice Folding



ProtogaLattice

- A lattice-based version of Protogalaxy
- The first lattice-based **algebraic** folding scheme supporting Plonkish
- Supports full PCD/ IVC
- **Constant-round** for folding via multivariate polynomials
- Bootstrapping to reduce accumulator norm
- Black-box Range Proofs for witnesses norm bound





Lattice Folding

	SuperNeo [KS25]	Latticefold+ [BC25]	Cyclo [GLLO26]	ProtogaLattice [BNP26]
Folded relation*	CCS/LCCS	CCS/LCCS	CCS/LCCS	Arbitrary polynomials
Techniques	HyperNova style sumcheck	HyperNova style sumcheck	HyperNova style sumcheck	Multivariate polynomial interpolation
Range proof	Binary sumcheck proof	Wider proof, uses ring monomials	Ternary sumcheck proof	Ternary sumcheck proof
Proof size (2^{27} witness size)	~100 KB	~100 KB	32 KB	6.52 KB
# Rounds	>60	>100	~60	~25

*Folding degree-2 Relations: R1CS

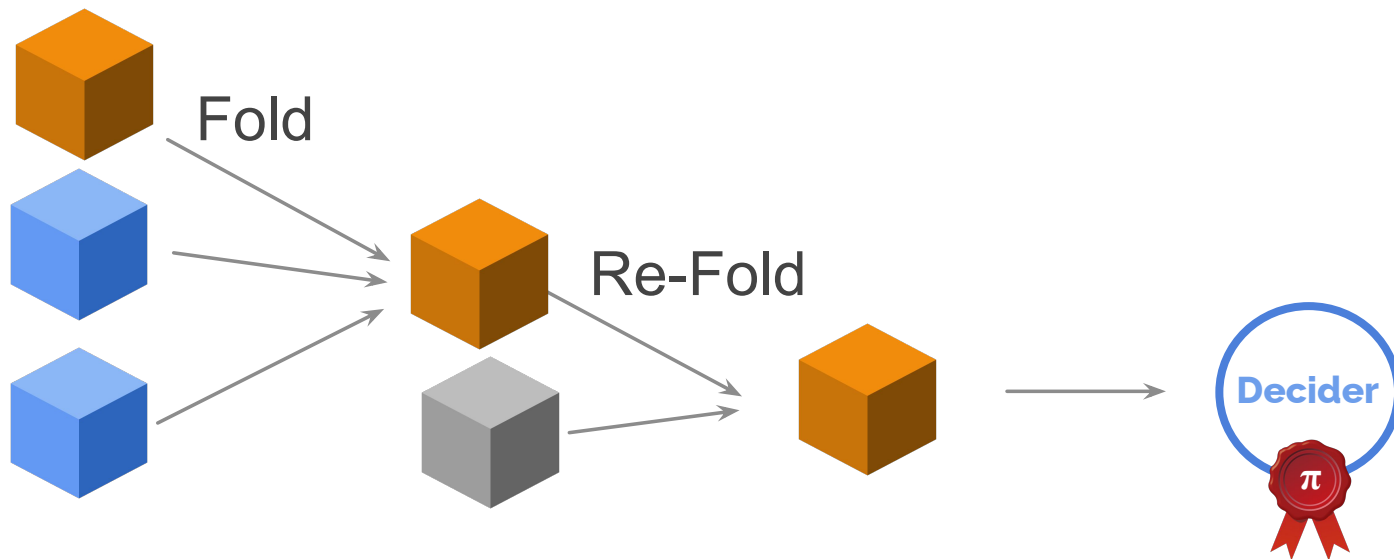


Lattice Folding

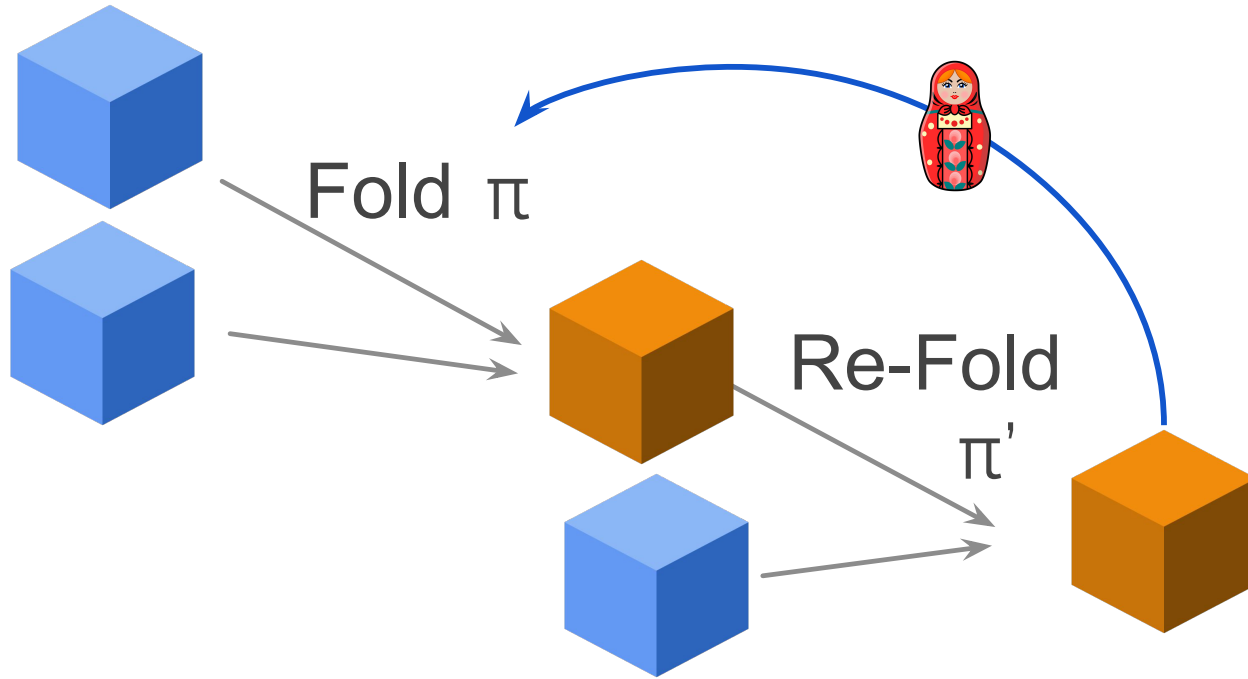
	SuperNeo [KS25]	Latticefold+ [BC25]	Cyclo [GLLO26]	Protoga Lattice[BNP26]	PikkuFold [Osadnik26]
Folded relation	CCS/LCCS	CCS/LCCS	CCS/LCCS	Arbitrary polynomials	CCS/LCCS
Techniques	HyperNova style sumcheck	HyperNova style sumcheck	HyperNova style sumcheck	Multivariate polynomial interpolation	HyperNova style sumcheck
Range proof	Binary sumcheck proof	Wider proof, uses ring monomials	Ternary sumcheck proof	Ternary sumcheck proof	Approximate proof*
Proof size (2^{27} witness size)	~100KB	~100 KB	32KB for 64 folds	6.52 KB for 64 folds	~5-11 KB for 32 folds
# Rounds	>60	>100	~60	~25	>46

* Johnson–Lindenstrauss Projection

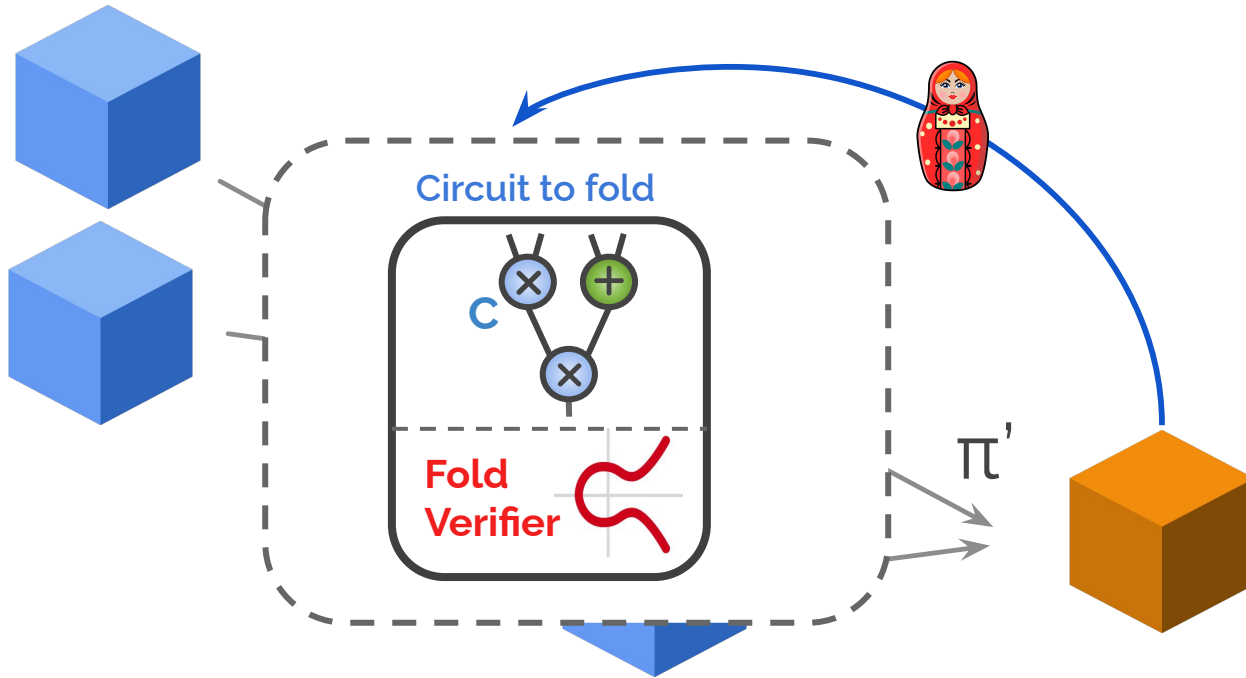
Recursion & Decider



Recursion via Folding

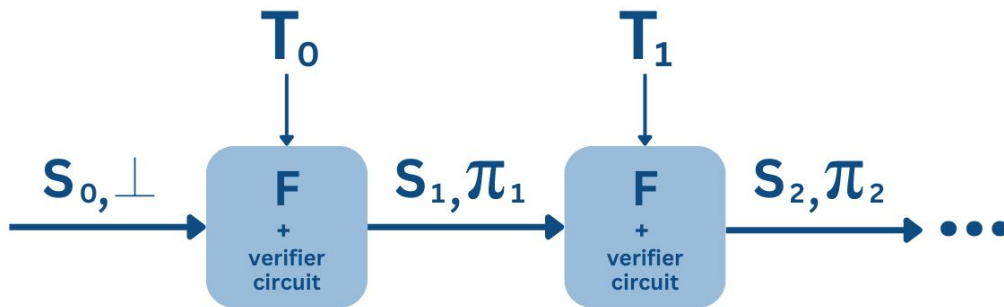


Recursion via Folding



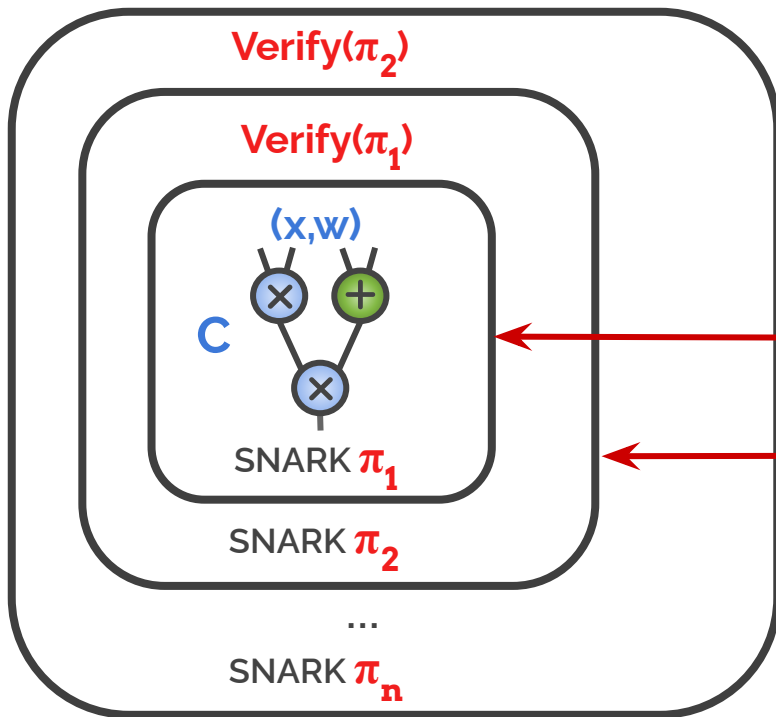
IVC

Incremental Verifiable Computation





Proof Recursion



Verification Runtime

$$|\text{Verify}| = 2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify}_1}| = 2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify}_2}| = 2|C_{\text{Verify}_1}| = 2^2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify}_3}| = 2|C_{\text{Verify}_2}| = 2^3|C_{\text{SNARK}}|$$

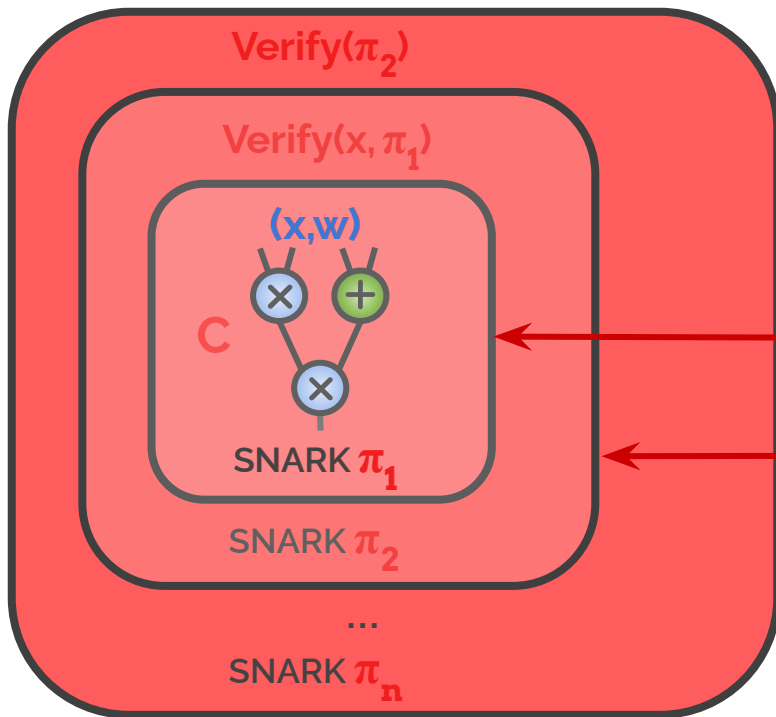


Linear Verification

*in the size of the computation to be

verified

Exponential Growth!!!



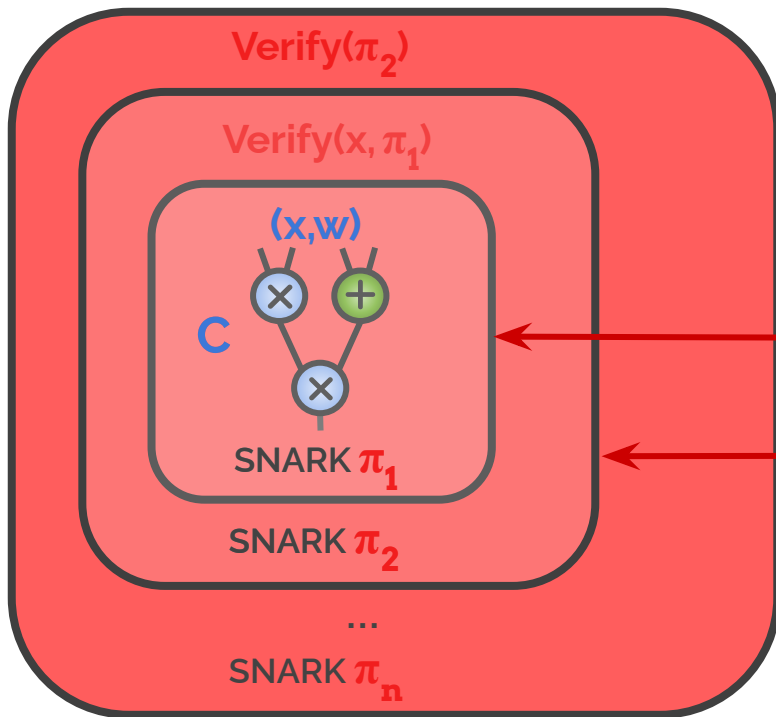
$$|C_{\text{Verify1}}| = 2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify2}}| = 2|C_{\text{Verify1}}| = 2^2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify3}}| = 2|C_{\text{Verify2}}| = 2^3|C_{\text{SNARK}}|$$



Proof Recursion



Needs Sublinear Verification!

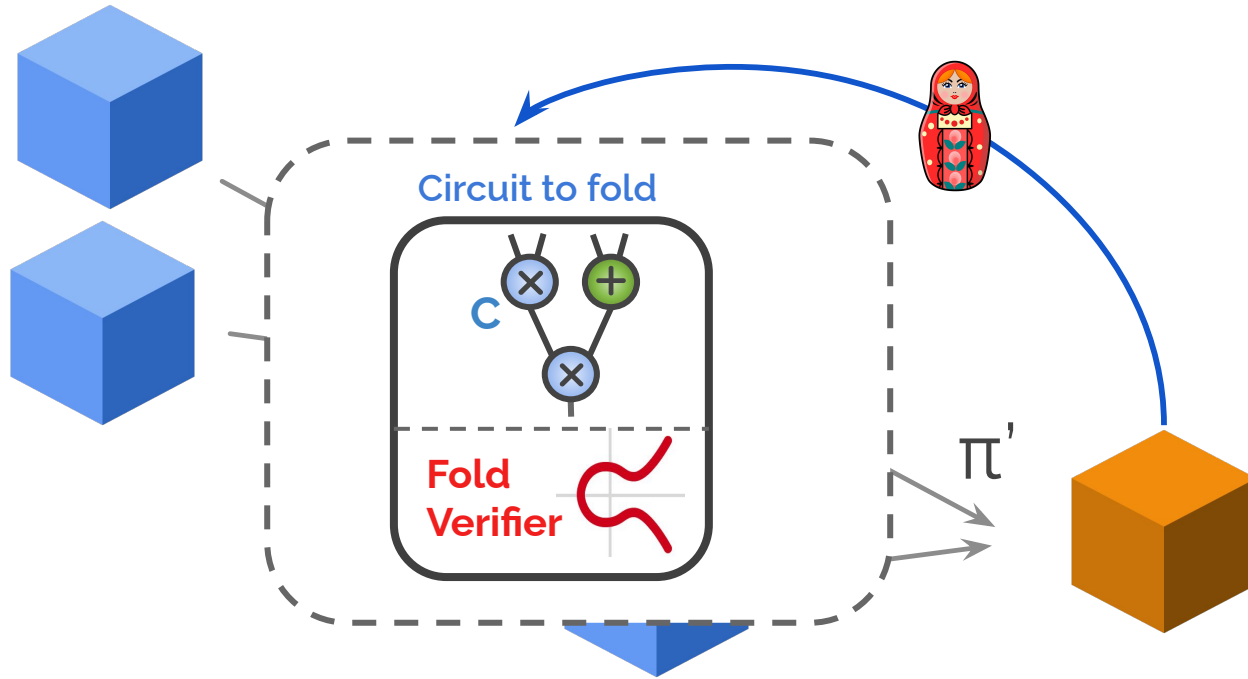
*in the size of the computation to be verified

$$|C_{\text{Verify}_1}| = 2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify}_2}| = 2|C_{\text{Verify}_1}| = 2^2|C_{\text{SNARK}}|$$

$$|C_{\text{Verify}_3}| = 2|C_{\text{Verify}_2}| = 2^3|C_{\text{SNARK}}|$$

Recursion via Folding





Folding Paradigms

Algebraic vs. Sumcheck-based folding

- Algebraic:**
- closed algebraic transformation on the relation itself,
 - involves evaluating and linearly combining polynomial constraints on the relation witnesses
- Sumcheck:**
- phrase the folding as a claim about a big polynomial (or products of multilinear polynomials)
 - use the sumcheck protocol to reduce the claim to a small number of scalar checks

Algebraic Folding

Fold via random linear combination

Pros: Constant rounds, small proofs, expressive relations: arbitrary polynomial constraints

Cons: Trusted setup (KZG), prover costs (polynomial evaluations, divisions)

Sumcheck Folding

Fold via sumcheck protocol

Pros: linear proving time, transparent setup

Cons: $O(\log N)$ rounds, larger proofs, more involved verifier, limited to linear/linearized relations



Why Algebraic Folding ?

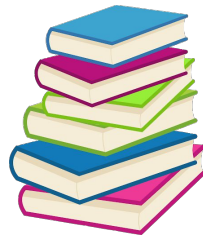
Algebraic folding is the right fit for rollups

- **Expressivity:** Supports arbitrary-degree polynomial constraints
 - Plonkish arithmetization
 - Custom gates (SHA256)
 - Lookup arguments
- **Efficiency:** Constant-round folding
 - Small folding proof: $O(1)$ elements
 - **Bigger** prover costs
- **Gap:** No Lattice-based scheme with **Decider**

Open Questions



- **Extensions:**
 - adding a Decider for end-to-end proving
 - explore ZK for folding
 - better NP representations: other arithmetization to capture lookups, custom gates
 - fold instances for different relations, e.g. [SuperNova](#)
- **Collaborative SNARKs / Folding schemes**
- **Lattice-based** folding schemes:
 - Build a Decider
 - Better range proof that maintains round complexity for IVC
- **Understand the IVC security implications for soundness** – exponential loss in the number of iterations
- **Machine learning ZK Proofs** - can we use folding schemes?



References

- [GGPRR13] **Quadratic span programs and succinct NIZKs without PCPs** – R. Gennaro, C. Gentry, Bryan Parno, Mariana Raykova.
- [Groth16] **On the Size of Pairing-based Non-interactive Arguments** – Jens Groth, EUROCRYPT 2016
- [GWC19] **PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge** – A Gabizon, Z. Williamson, O. Ciobotaru
- [GMN20] **SnarkPack: Practical SNARK Aggregation** – Nicolas Gailly, Mary Maller, Anca Nitulescu, FC 2022
- [KST22] **Nova: Recursive Zero-Knowledge Arguments from Folding Schemes** – Abhiram Kothapalli, Srinath Setty, Ioanna Tzialla, CRYPTO 2022
- [NPR24] **FLIP-and-Prove R1CS** – Anca Nitulescu, Nikitas Paspis, Carla Ràfols, eprint.iacr.org/2024/1364
- [BC24] **LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems** – D. Boneh, B. Chen, eprint.iacr.org/2024/257
- [BC25] **LatticeFold+: Faster, Simpler, Shorter Lattice-Based Folding for Succinct Proof Systems** – D. Boneh, B. Chen, eprint.iacr.org/2025/247
- [NS26] **Neo and SuperNeo: Post-quantum Folding with Pay-per-Bit Costs over Small Fields** – W. Nguyen, S. Setty, eprint.iacr.org/2026/242
- [BNP26] **ProtogaLattice: Constant-Round Lattice-based Folding for General Polynomial Relations** – D. Balbás, A. Nitulescu, M. Plançon, eprint.iacr.org/2026/1317
- [Osa26] **PikkuFold: Efficient Folding in a Few Kilobytes** – M. Osadnik, eprint.iacr.org/2026/1809

Thanks





Questions?



Credits

Special thanks to all those who made and released these resources for free:

- Presentation template by [SlidesCarnival](#)
- Illustrations by [Iconfinder](#)