

KZH-Fold: Sublinear accumulation

George Kadianakis - Aranxta Zapico - Hossein Hafezi - Benedikt Bünz



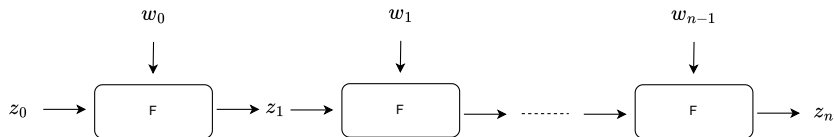
Table of Contents

- 1 Background
- 2 Motivation
- 3 KZH and Its Accumulation Scheme: KZH-Fold
- 4 IVC for R1CS from PCS accumulation
- 5 Non-Uniform PCD for R1CS from PCS Accumulation

Background

What is Incremental Verifiable Computation (IVC)? [Val08]

- **IVC:**

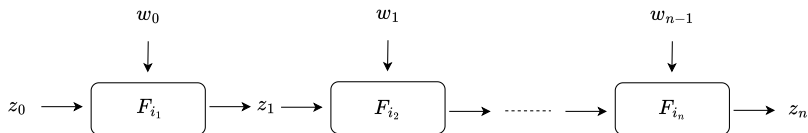


- **PCD:** Generalizes IVC to DAGs

Non-Uniform IVC/PCD?

Step function F can be one of multiple predefined instructions F_1, \dots, F_k

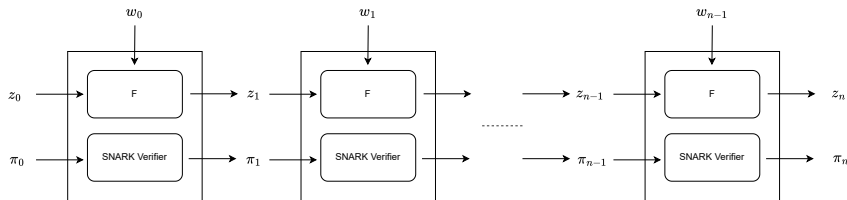
- Motivation: Verifiable CPU



Traditional IVC construction? [BCC+13]

IVC \implies Augmented circuit:

- 1 Subcircuit F
- 2 SNARK verifier subcircuit (**recursive overhead**)



IVC Construction from Accumulation Schemes [Halo]

✗ SNARK verifier in circuit is expensive!

Observation. Defer checking all proof \implies accumulation schemes

Accumulation Scheme

Accumulation Scheme [BCMS21;KST21;KP22; BC23]

- Let \mathcal{L}_π and \mathcal{L}_{acc} be two NP languages.
- An accumulation:

$$acc \in \mathcal{L}_{acc}, \pi \in \mathcal{L}_\pi \xRightarrow{\text{accumulate}} acc' \in \mathcal{L}_{acc}$$

satisfies *completeness* and *knowledge soundness*.

Accumulation Scheme [BCMS21;KST21;KP22; BC23]

- Let \mathcal{L}_π and \mathcal{L}_{acc} be two NP languages.
- An accumulation:

$$acc \in \mathcal{L}_{acc}, \pi \in \mathcal{L}_\pi \xRightarrow{\text{accumulate}} acc' \in \mathcal{L}_{acc}$$

Completeness: acc and π are satisfied $\iff acc'$ is satisfied.

- Decider = Satisfiability function of \mathcal{L}_{acc}

Accumulation Scheme [BCMS21;KST21;KP22; BC23]

- Let \mathcal{L}_π and \mathcal{L}_{acc} be two NP languages.
- An accumulation:

$$acc \in \mathcal{L}_{acc}, \pi \in \mathcal{L}_\pi \xRightarrow{\text{accumulate}} acc' \in \mathcal{L}_{acc}$$

Knowledge soundness: witness for acc' \implies witness for acc and π .

Sublinear Accumulation

Sublinearity:

- $\text{acc} \in \mathcal{L}_{\text{acc}}, \pi \in \mathcal{L}_{\pi} \implies |\text{acc}| \in o(|\pi|)$
- Decider time < Verification of π

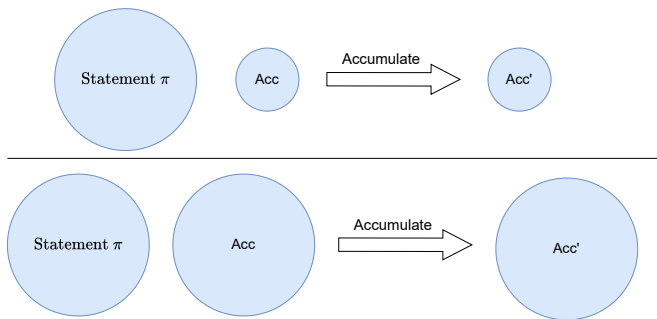
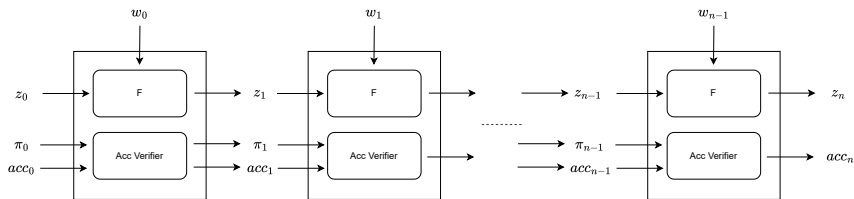


Figure: Sublinear vs Linear-Sized Accumulation Scheme

Build IVC from Accumulation Scheme

How to Use Accumulation to Build IVC?



$\pi_1, \pi_2, \dots, \pi_n \implies$ Accumulate them into acc_n and check acc_n .

BCLMS Compiler: Accumulation for NARK \implies IVC

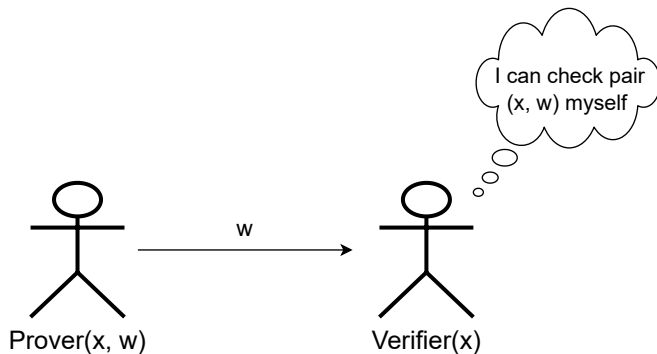
BCLMS Compiler: NARK + NARK Verifier Accumulation \implies IVC

- IVC proof = $|\text{acc}|$
- IVC verifier = $\text{Decider}_{\text{acc}}$
- IVC prover = $\text{Prover}_{\text{NARK}}$ (for augmented circuit) + $\text{Prover}_{\text{acc}}$

Sublinear accumulator \implies IVC/PCD with sublinear proofs/decider

Example of Nova

Trivial Nark



Nova (Accumulation for Trivial R1CS NARK)

Nova accumulator for (relaxed) R1CS: (n = instance + witness size of R1CS)

- Accumulator size: $O(n)$
- Accumulator prover time: $O(n)$
- Accumulator decider time: $O(n)$
- Verifier circuit: $O(1)$, 2 or 3 scalar multiplications

Nova (Accumulation for Trivial R1CS NARK)

Nova accumulator for (relaxed) R1CS: (n = instance + witness size of R1CS)

- Accumulator size: $O(n)$
- Accumulator prover time: $O(n)$
- Accumulator decider time: $O(n)$
- Verifier circuit: $O(1)$, 2 or 3 scalar multiplications

Nova IVC via BCLMS compiler:

- IVC proof size: $O(n)$
- IVC prover time: $O(n)$
- IVC decider time: $O(n)$

Nova Compression Phase

Consider a function (circuit) F to be $1M$ constraints \implies

- accumulator size: $80MB$
- decider time: $4s$

\implies Impractical distributed proving.

Nova Compression Phase

Consider a function (circuit) F to be $1M$ constraints \implies

- accumulator size: $80MB$
- decider time: $4s$

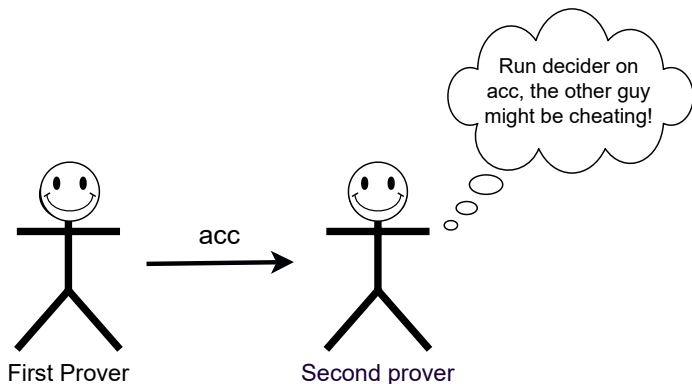
\implies Impractical distributed proving.

MicroNova/Nova solution: Run a zkSNARK at the last step of IVC \implies
reduces proof size ($\approx 1KB$)

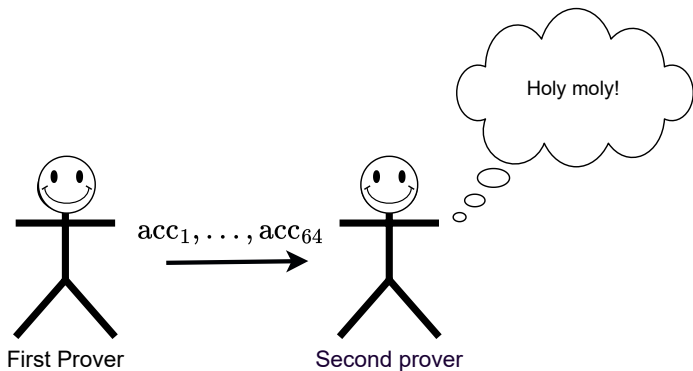
- ✗ $24\times$ prover overhead
- ✗ No longer incremental.

Motivations of Paper

Motivation: Distributed Proving



Motivation: Distributed Proving of zkVM with SuperNova



- N-IVC/N-PCD with SuperNova \implies One accumulator per instruction

Motivation For Sublinear Decider

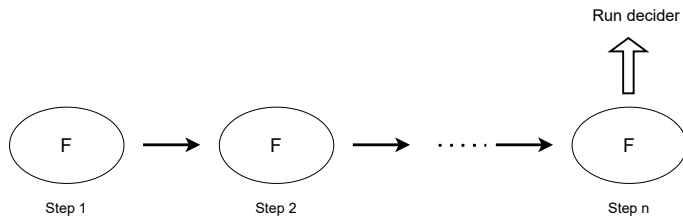


Figure: IVC when proof of the last step is needed.

Motivation: IVC When All Steps Matter!

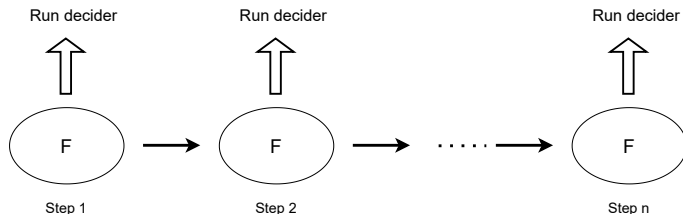


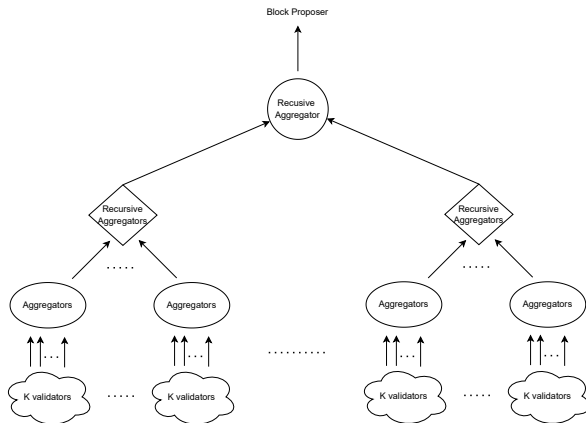
Figure: IVC when proof of all steps are needed.

Applications:

- Light clients
- Verifiable key directory
- Succinct blockchain

Motivation: Distributed signature aggregation (1)

- Ethereum finality time: 15min \implies dominated by signature aggregation computation



Motivation: Distributed signature aggregation (2)

Computation of aggregators:

- BLS aggregation
- Union of bit vectors (mini PIOP)

Signature aggregation is a layered tree \implies model as PCD

- Communication is P2P \implies we require low communication

Motivation: Distributed signature aggregation (3)

Signature Aggregation Based on PCD with sublinear:

- Improve verifier time and communication by 5x
- 5x improvement in finality time

Contribution

- **KZH, PCS with Sublinear Opening**
- **Sublinear Accumulator Based on KZH**
 - IVC/PCD with sublinear proof and verifier.
 - Signature aggregation for Ethereum via PCD.
- **New Approach to N-IVC / N-PCD, First efficient N-PCD scheme based on any polynomial accumulation**

See numbers for KZH-fold

Scheme	Prover	Verifier	Acc	# Constraints
Spartan+KZH-fold	16.5 s	135 ms	37 KB	$2^{21} \approx 2097k$
Nova	4.8 s	5.6 s	80.8 MB	1185k

Table: Spartan+KZH-Fold vs Nova for Circuit With 2000 Poseidon Hashes

Starting Point

- IVC with sublinear proof/decider \iff Sublinear accumulation.
- PCS with sublinear predicate \implies Sublinear accumulation.

Goal: PCS with following properties:

- Homomorphic
- Constant size commitment
- Sublinear opening size
- Algebraic and low degree verifier checks
- No degree 2 pairing

$$\sum_i e(P_i, g_i) = \sum_j e(P'_j, g'_j) \implies g_i, g'_j \text{ are fixed.}$$

KZH and Its Accumulation Scheme: KZH-Fold

Starting point: Hyrax

Hyrax:

- + PCS with square root opening size \implies square root accumulator size
- X Commitment = $O(\sqrt{n})$ group elements \implies High recursive overhead

Starting point: Hyrax

Given a Pedersen setup (g_1, g_2, \dots, g_m) , commit to rows of the coefficient matrix:

$$\begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{k1} & t_{k2} & \cdots & t_{km} \end{bmatrix} \begin{matrix} \implies c_1 \\ \implies c_2 \\ \vdots \\ \implies c_k \end{matrix}$$

Commitment size is $O(\sqrt{n}) \Rightarrow$ homomorphism/acc verifier is $O(\sqrt{n})$

KZH: Commitment Phase

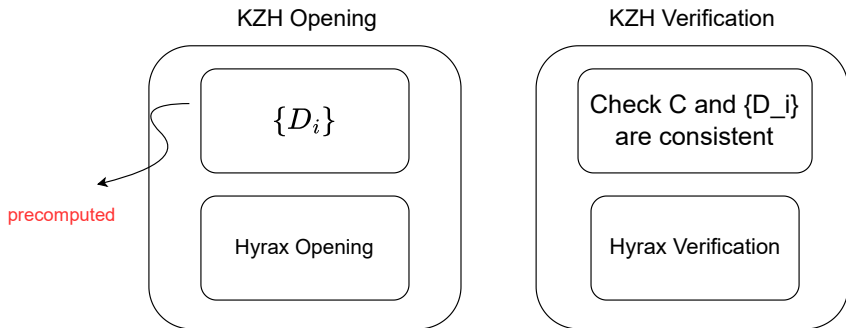
Commitment phase of KZH: (with SRS)

- 1 C : commit to the whole matrix in $1\mathbb{G}$ element, using a universal-SRS.
- 2 $\{D_i\}$: Commit to each row using g_1, \dots, g_m .

$$\underbrace{\begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{k1} & t_{k2} & \cdots & t_{km} \end{bmatrix}}_C \begin{matrix} \implies D_1 \\ \implies D_2 \\ \vdots \\ \implies D_k \end{matrix}$$

Prove via pairing C and $\{D_i\}$ correspond to the same matrix.

KZH: Opening and Verification



For a multilinear polynomial $f(\vec{X})$ with on boolean hypercube of size ℓ :

- commitment time = $O(\ell)$ group operations.
- Proof size = $\sqrt{\ell} \mathbb{G}_1 + \sqrt{\ell} \mathbb{F}$
- Opening time = $\ell \mathbb{F}$ ¹
- Verifier time = $\sqrt{\ell}$ pairing + MSM($2\sqrt{\ell}$) + $\sqrt{\ell} \mathbb{F}$.

¹Dominated by polynomial evaluation.

We extend KZH matrix construction to tensors of higher degree \implies
Lower verifier cost at the cost of higher opening.

- Commitment cost: $O(k \cdot n)$
- Opening cost: $O(n^{1/2})$ via pre-processing
- Verifier cost: $O(k \cdot n^{1/k})$

The verifier function for KZH = degree 2 scalar multiplications + degree 1 pairing check $\xRightarrow{\text{Protostar compiler}}$

- Accumulator size. $O(\ell^{\frac{1}{2}})$
- Decider complexity. $O(\ell^{\frac{1}{2}})$
- Prover complexity. $O(\ell)$
- Verifier complexity. $3 - 4 \mathbb{G}_1$ scalar multiplications + $O(1) \mathbb{F}$

KZH-k fold:

- $O(k \cdot n^{1/k})$ decider and accumulator size
- $k + 1$ scalar multiplication in recursive circuit.

Compariosn of kzh-fold and other schemes

Scheme	Recursive Overhead	Decider	—acc—
Nova	2 group ops	$\text{MSM}(n)$	$O(n)$
KZH2-fold	3 group ops	$n^{\frac{1}{2}} P$	$O(n^{\frac{1}{2}})$
KZH-k fold	$k + 1$ group ops	$k \cdot n^{\frac{1}{k}} P$	$O(k \cdot n^{\frac{1}{k}})$
Halo	$O(\log n)$ group ops	$\text{MSM}(n)$	$O(\log n)$

IVC/PCD for R1CS From PCS Accumulation



Accumulation for NP from PCS Accumulation

$\text{NP} \xRightarrow{\text{PIOP}} \text{Polynomial checks} \xRightarrow{\text{PCS Acc}} \text{Accumulate polynomial checks}$

$\text{R1CS} \xRightarrow{\text{Spartan}} \text{Polynomial checks} \xRightarrow{\text{PCS Acc}} \text{Accumulate polynomial checks}$

Accumulation for R1CS from PCS Accumulation

R1CS $\xRightarrow{\text{Spartan}}$ witness polynomial $w(\cdot)$ + matrices A, B, C evaluations

- $w(\cdot) \Rightarrow$ interpolate $w(\cdot)$ through a PCS and accumulate.
- Matrices A, B, C can be evaluated as KZH, i.e. KZH works for sparse matrices too.

Better way \Rightarrow accumulate following relation directly:

$$\mathcal{R}_{\tilde{A}} = \{(r_x \in \mathbb{F}^{\mu_n}, r_y \in \mathbb{F}^{\mu_m}, z \in \mathbb{F}) : \tilde{A}(r_x, r_y) = z\}$$

Accumulate A , B , C matrix evaluations

- Prover cost: $\log n$ evaluation of \tilde{A} (out of circuit)
- Verifier cost (circuit size): $O(\log n)\mathbb{F}$
- Proof size: $O(\log n)\mathbb{F}$

Accumulation for R1CS

R1CS $\xRightarrow{\text{Spartan}}$ witness polynomial $w(\cdot)$ + matrices A, B, C evaluations

- $w(\cdot) \Rightarrow$ interpolate $w(\cdot)$ as PCS (KZH) and accumulate with PCS accumulator (KZH-fold).
- Accumulate matrix evaluation of A, B, C directly.

IVC from PCS accumulation

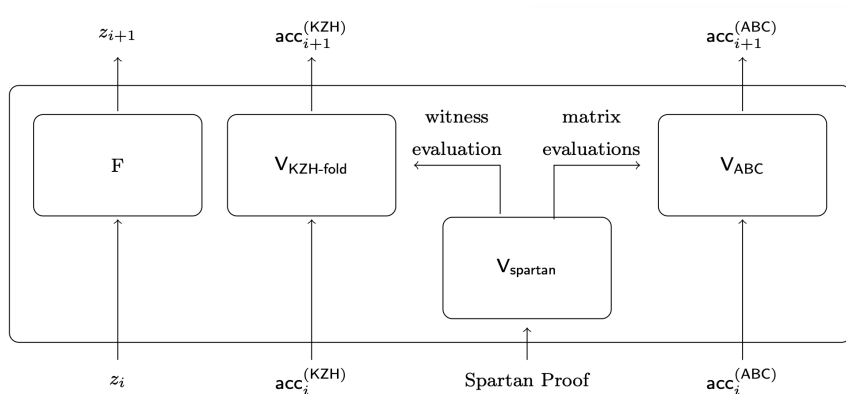


Figure: Augmented circuit initiated with KZH-fold

Non-Uniform PCD from PCS Accumulation



Comparison of Approaches to N-PCD

Scheme	Prover Time	Verifier Time	Witness Size
SuperNova	$O(\sum F_i)\mathbb{G}$	$O(\sum F_i)\mathbb{G}$	$O(\sum_i F_i)$
Protostar	$O(\sum F_i)\mathbb{G}$	$O(\sum F_i)\mathbb{G}$	$O(\sum_i F_i)$
KiloNova	$O(\sum F_i)\mathbb{G}$	$O(\sum F_i)\mathbb{G}$	$O(\sum_i F_i)$
Spartan+PA	$\mathcal{P}_{\text{acc}}(\max_i F_i) + \sum_i \log F_i $	$\mathcal{D}_{\text{acc}}(\max_i F_i) + O(\sum_i F_i)\mathbb{F}$	$O(\text{acc} + \sum \log_i F_i)$

PA=KZH-Fold:

- Prover time: $O(\max_i |F_i|) + \sum_i \log |F_i|$
- Verifier time: $O(\sqrt{\max_i |F_i|}) + O(\sum_i |F_i|)\mathbb{F}$
- Witness size: $\sqrt{\max_i |F_i|} + \sum \log_i |F_i|$

Non-Uniform PCD from PCS Accumulation

High-level idea:

- PCS accumulation \implies more flexible than circuit accumulation
- Polynomials of different degrees can be accumulated.

Comparison to Supernova:

- directly accumulates circuit \implies one running accumulator for each instruction.

Non-Uniform IVC from PCS Accumulation (1)

$F_i \xRightarrow{\text{Spartan}} \omega_i(\cdot)$ and matrix evaluation of A_i , B_i and C_i .

- $\omega_i(\cdot) \implies$
 - 1 Consider a running polynomial of degree $\deg(\omega_i) < D$.
 - 2 Accumulate $\omega_i(\cdot)$ with this running PCS accumulator.
- matrix evaluations of A_i , B_i and C_i
 - Similar strategy to SuperNova

Key to efficiency: Matrix evaluations scale logarithmically with the size of the original circuit.

THNAKS!

IronDict: Transparent Dictionary from Polynomial Commitments

Benedikt Bünz
NYU

Joseph Bonneau
NYU / A16Z Crypto

Hossein Hafezi
NYU / IMDEA Software

Alireza Shirzad
Upenn / Lagrange