WARP

Linear-Time Accumulation Schemes

Benedikt Bünz

NYU

Giacomo Fenzi

EPFL

Alessandro Chiesa

EPFL

William Wang

NYU

ePrint 2025/753

Motivation

aka
why you should care about accumulation schemes

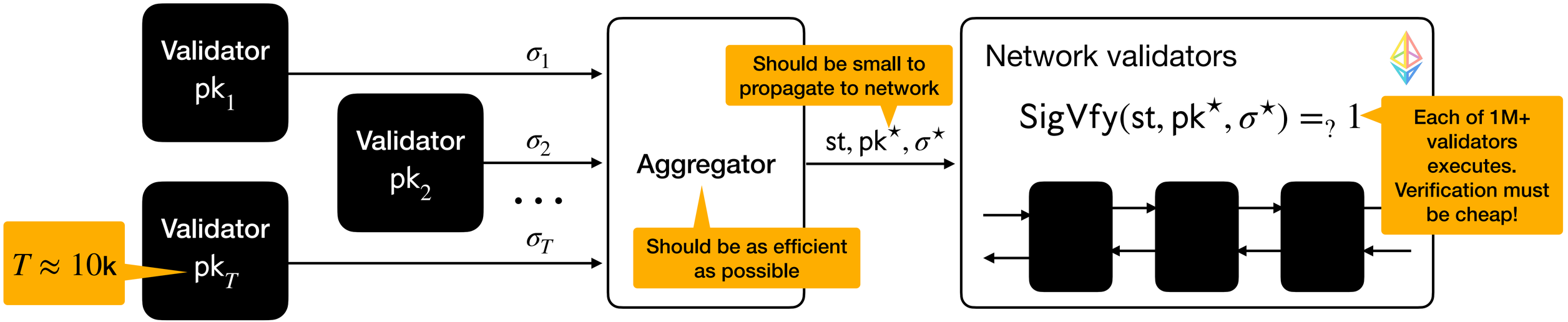
Application: PQ-signature aggregation

Ethereum's consensus

(1) Randomly chosen subcommittee of validators agrees on a state st
(2) Each validator in the committee generates a signature

(3) Aggregator batches signatures into single one
(4) & propagates to the network

(5) Each validator checks the aggregated signature



Today: BLS signatures. **Ethereum is looking for a post-quantum alternative.**

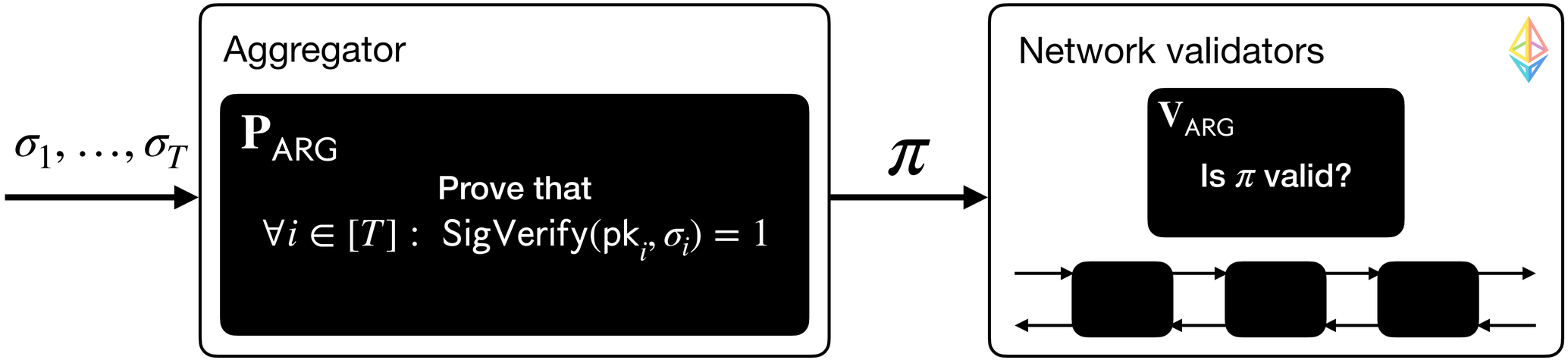
Idea: a pq-signature such as *hash-based XMSS*? **Problem:** how to efficiently aggregate? (no homomorphisms...)

Application: PQ-signature aggregation

A first idea: use a pqSNARK

Let $(\mathbf{P}_{\text{ARG}}, \mathbf{V}_{\text{ARG}})$ be a general purpose pqSNARK (e.g. Spartan+WHIR).

Wednesday at 9:00
Proof systems track



PQ secure ✅

Cheap verification ✅

Compressing $|\pi| \ll T \cdot |\sigma|$ ✅

$|\pi|$ depends on $\log T$

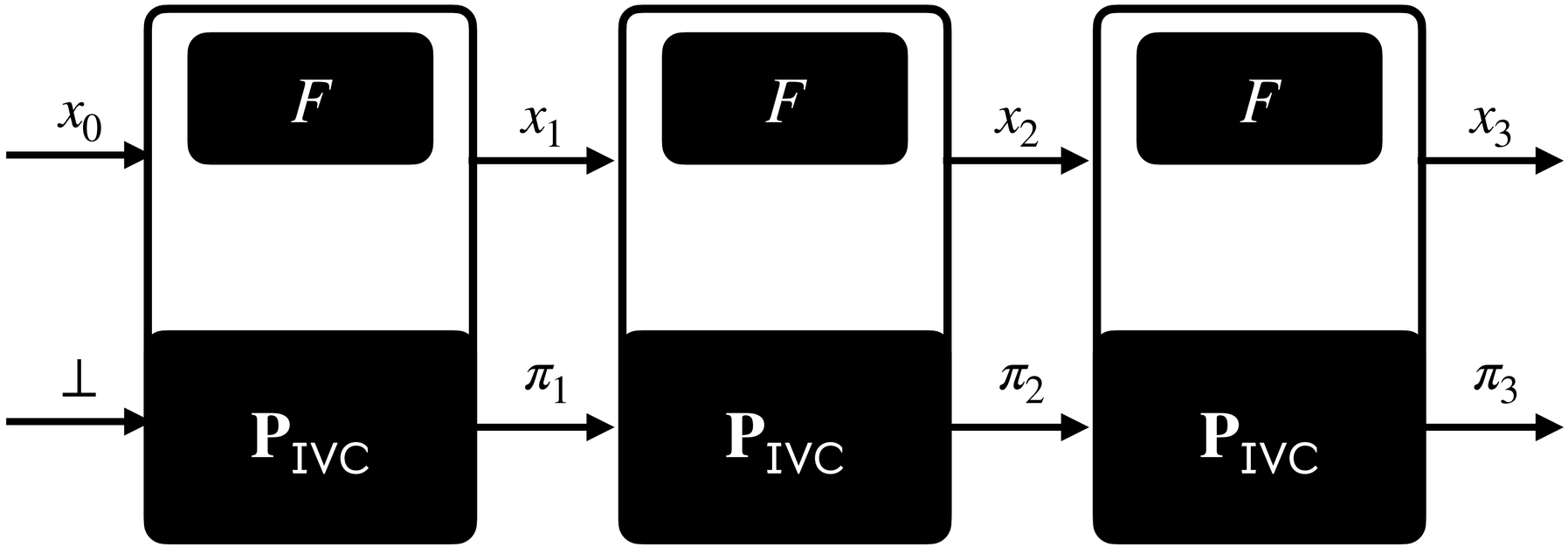
Aggregator needs memory $\Omega(T)$

Can we do better?

Incrementally Verifiable Computation (IVC)

To prove $x_T = F^T(x_0)$, prove $\exists x_1, \ldots, x_{T-1}$ such that $\forall i \in [T], x_i = F(x_{i-1})$.

In signature aggregation:
$F((\sigma_i, pk_i), b_i) := b_i \wedge \mathsf{SigVfy}(\mathsf{st}, pk_i, \sigma_i)$



$\mathbf{V}_{\mathrm{IVC}}(x_{i-1}, x_i, \pi_i)$ checks that π_i attests the **whole computation!**

$\mathbf{P}_{\mathrm{IVC}}$ costs independent from T ✅

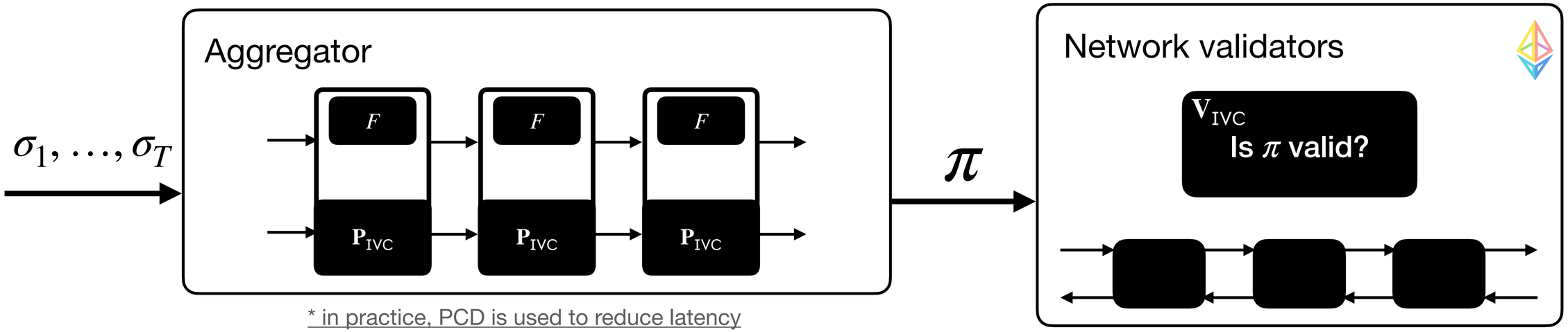
IVC can be generalized to **Proof-Carrying-Data** (PCD).
PCD considers a directed acyclic graph instead of a line.
PCD in practice is preferable to IVC, as it enables reducing the prover's latency.

Let's apply IVC to the initial idea.

Application: PQ-signature aggregation

Final blueprint:

Let $(\mathbf{P}_{\text{IVC}}, \mathbf{V}_{\text{IVC}})$ be a post-quantum secure IVC scheme.



* in practice, PCD is used to reduce latency

PQ secure ✅

$|\pi|$ independent from T ✅

Cheap aggregator ✅

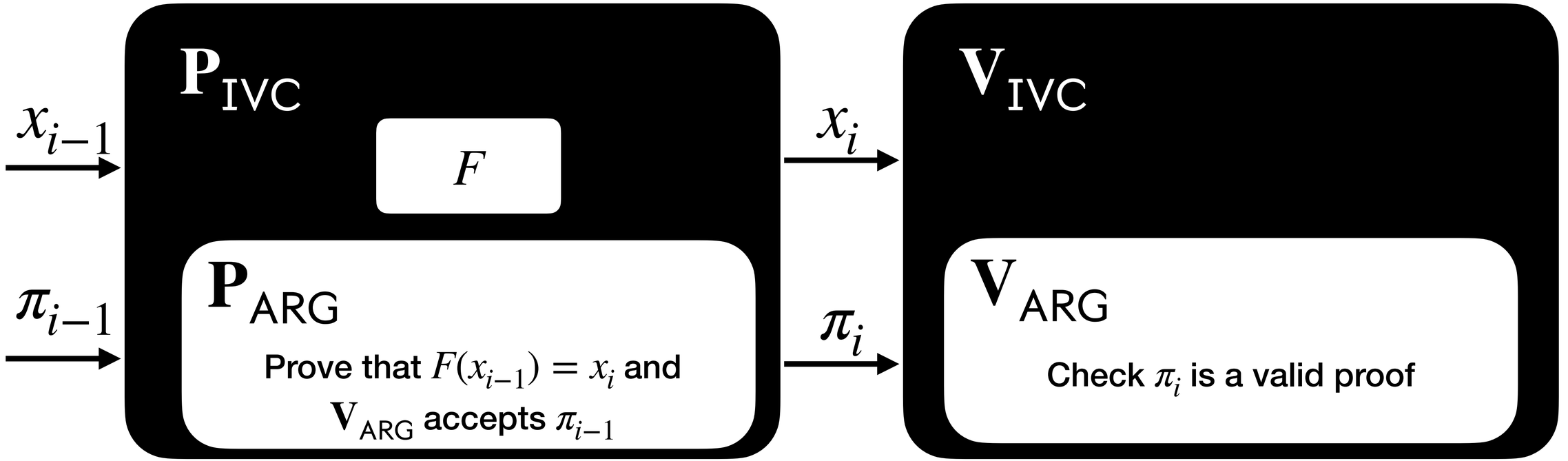
Cheap verification ✅

Wonderful. Where can I get IVC?

IVC from SNARKs

Recursive proof composition

(*) more complex than this, needs preprocessing



PQ SNARK $\Longrightarrow$ PQ IVC ✅

$|\pi|$ independent from T ✅

Cheap verification ✅

Memory costs independent from T ✅

Cost of $\mathbf{P}_{\text{IVC}} \approx |F| + |\mathbf{V}_{\text{ARG}}|$

Concretely: $|\mathbf{V}_{\text{ARG}}| \approx 2^{20}$ constraints

i.e. recursive overhead is quite large

Good starting point, but can be improved!

Accumulation Schemes

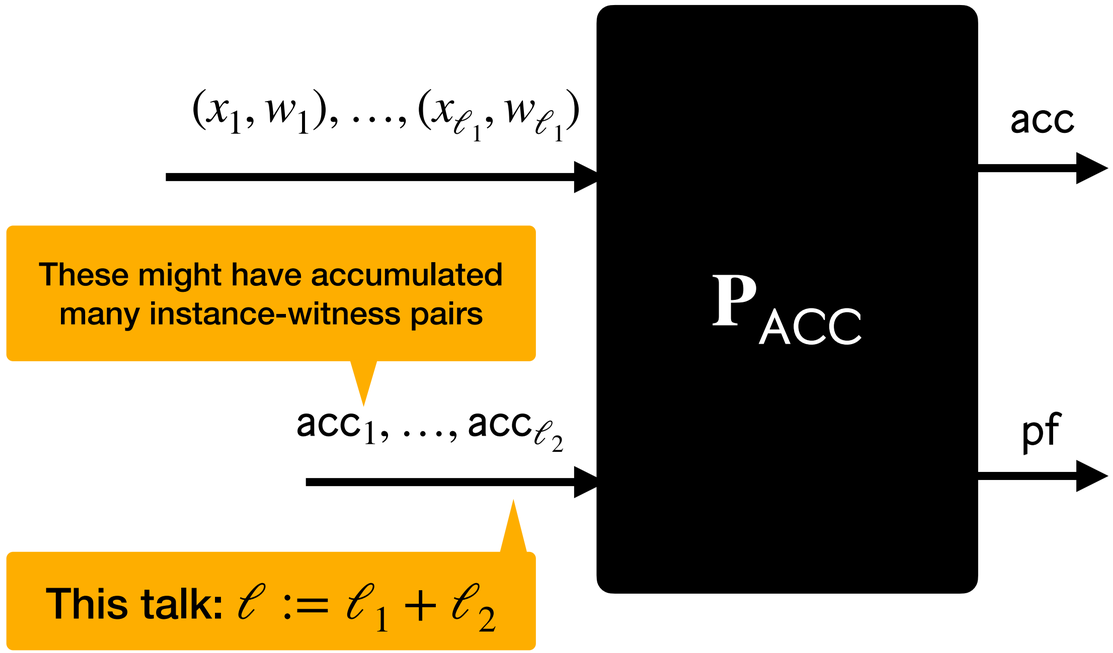
A lightweight tool for batching

Any ARG yields ACC with $|\mathbf{V}_{\mathsf{ACC}}| \approx \ell_1 \cdot |\mathbf{V}_{\mathsf{ARG}}|$.
We can do (significantly) better!

Enables batching many checks $(x_i, w_i) \in_? \mathscr{R}$ into an accumulator acc.

$\mathbf{V}_{\mathsf{ACC}}$ verifies that adding the inputs into acc was done correctly

$\mathbf{D}_{\mathsf{ACC}}$ decides whether acc is valid.



If:

a) $\mathbf{V}_{\mathsf{ACC}}((x_i)_i, (\mathsf{acc}_j)_j, \mathsf{acc}, \mathsf{pf}) = 1$

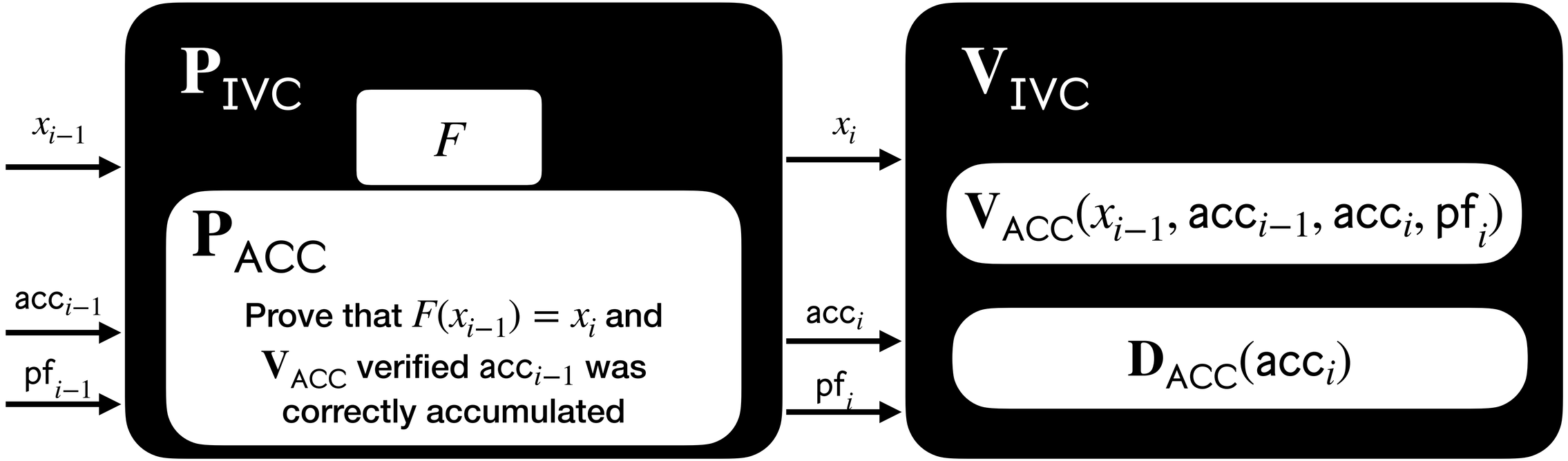
b) $\mathbf{D}_{\mathsf{ACC}}(\mathsf{acc}) = 1$

Then w.h.p:

$\forall i \in [\ell_1] : \ (x_i, w_i) \in \mathscr{R}$
$\forall j \in [\ell_2] : \ \mathbf{D}_{\mathsf{ACC}}(\mathsf{acc}_j) = 1$

IVC from accumulation

(*) actually we need a more refined notion: "split" accumulation schemes



PQ Accumulation $\Longrightarrow$ PQ IVC ✅

Memory costs independent from T ✅

$|\pi|$ independent from T ✅

Cost of $\mathbf{P}_{\text{IVC}} \approx |F| + |\mathbf{V}_{\text{ACC}}|$ ✅ $\ll |\mathbf{V}_{\text{ARG}}|$

Not succinct

Cost of $\mathbf{V}_{\text{IVC}} \approx |\mathbf{V}_{\text{ACC}}| + |\mathbf{D}_{\text{ACC}}|$

Wrap with a final SNARK $\Longrightarrow$ succinct verification ✅

One more thing…

ACC is not limited to signature aggregation

Accumulation schemes are broadly useful for integrity in distributed systems with repeated computations.

Verifiable Virtual Machines (VVMs)

SP 1

RISC ZERO

NEXUS

OpenVM

+ At least 20 more…

Digital provenance

VIMz: Private Proofs of Image Manipulation using Folding-based zkSNARKs*

Stefan Dziembowski Shahriar Ebrahimi Parisa Hassanizadeh

Eva: Efficient Privacy-Preserving Proof of Authenticity for Lossily Encoded Videos

Chengru Zhang[1], Xiao Yang[2], David Oswald[2], Mark Ryan[2], and Philipp Jovanovic[3]

Consensus

Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party

Elette Boyle* Ran Cohen† Aarushi Goel‡

And more…

Reef: Fast Succinct Non-Interactive Zero-Knowledge Regex Proofs

Sebastian Angel* Eleftherios Ioannidis* Elizabeth Margolin* Srinath Setty† Jess Woods*

*University of Pennsylvania †Microsoft Research

ALPACA: Anonymous Blocklisting with Constant-Sized Updatable Proofs

Jiwon Kim *University of Michigan* Abhiram Kothapalli *University of California, Berkeley* Orestis Chardouvelis *Carnegie Mellon University*

Riad S. Wahby *Carnegie Mellon University* Paul Grubbs *University of Michigan*

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

Accumulation schemes:

Group-based

Nova, Supernova, Hypernova, Protostar, Protogalaxy, NeutronNova, KZHFold, …

Must use 256-bit fields, accumulation time super-linear, cycles of curves required for recursion, not pq

Lattice-based

Latticefold, Lova, Latticefold+, Neo

Very promising, accumulation costs super-linear, plausibly pq some field flexibility

Hash-based

Awh, ARC, [TODAY]

Accumulation costs can be linear, plausibly pq, full field flexibility

Our results

Polynomial Equation Satisfiability

$$\mathscr{R}_{\mathsf{PESAT}}(\mathbb{F}) = \left\{ (i, x, w) : \begin{array}{c} i = (\hat{\mathbf{p}}, M, N, k) \\ x \in \mathbb{F}^{N-k} \\ w \in \mathbb{F}^{k} \\ \forall i \in [M] : \ \hat{\mathbf{p}}_i(x, w) = 0 \end{array} \right\}$$

Polynomial over $\mathbb{F}$ in N variables.

PESAT generalizes:
R1CS, CCS, GR1CS...

e.g. R1CS: for $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{M \times N}$ and $x \in \mathbb{F}^{N-k}$: $\exists w \in \mathbb{F}^{N-k}$ such that $\mathbf{A} \begin{bmatrix} x \\ w \end{bmatrix} \circ \mathbf{B} \begin{bmatrix} x \\ w \end{bmatrix} = \mathbf{C} \begin{bmatrix} x \\ w \end{bmatrix}$

Define $\hat{\mathbf{p}}_i(\mathbf{Z}) = \langle \mathbf{a}_i, \mathbf{Z} \rangle \cdot \langle \mathbf{b}_i, \mathbf{z} \rangle - \langle \mathbf{c}_i, \mathbf{z} \rangle$. The equivalent PESAT condition becomes:

"$\exists w \in \mathbb{F}^{N-k}$ such that $\forall i \in [M] : \ \hat{\mathbf{p}}_i(x, w) = 0$"

WARP 🌀

An essentially optimal hash-based accumulation scheme

To accumulate ℓ instances of $\mathscr{R}_{\mathsf{PESAT}}(\mathbb{F})$ and accumulators

Same complexity as deciding the instances and accumulators!

Prover cost: $O(\ell \cdot |\hat{\mathbf{p}}|)$ $\mathbb{F}$-ops and $O(k)$ random oracle queries

Verifier cost: $O(\ell \cdot (\log N + \log M + \lambda))$ $\mathbb{F}$-ops and
$O(\ell \cdot \lambda \cdot \log k)$ random oracle queries

Optimal for hash-based

Decider cost: $O(\hat{\mathbf{p}})$ $\mathbb{F}$-ops and $O(k)$ random oracle queries

Secure in the pure random oracle model (no other cryptography needed).

Can be instantiated over every $\mathbb{F}$ that is sufficiently large for soundness.

In fact, can be instantiated over **every** $\mathbb{F}$ using field extensions. Asymptotics vary.

Comparison

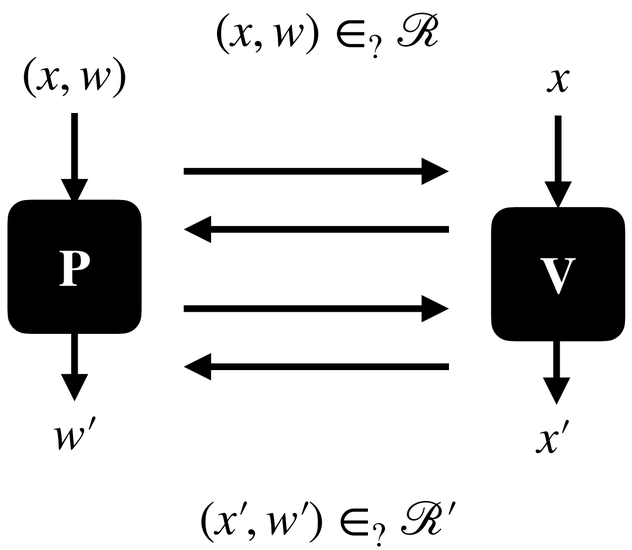
In this slide
$\ell = O(1)$

	hash-based?	linear prover?	verifier size (RO queries)
Brakedown	✅	✅	$O(\lambda \cdot \sqrt{k})$
Blaze	✅	✅	$O(\lambda \cdot \log^2 k)$
Group or lattice-based accumulation (Nova, etc.)	❌	❌	$O(1)$
Arc	✅	❌	$O(\lambda \cdot \log k)$
This work	✅	✅	$O(\lambda \cdot \log k)$
FACS (concurrent)	✅	✅	$O(\lambda \cdot \log k)$

On Hash-Based Accumulation

Hash-Based Reductions

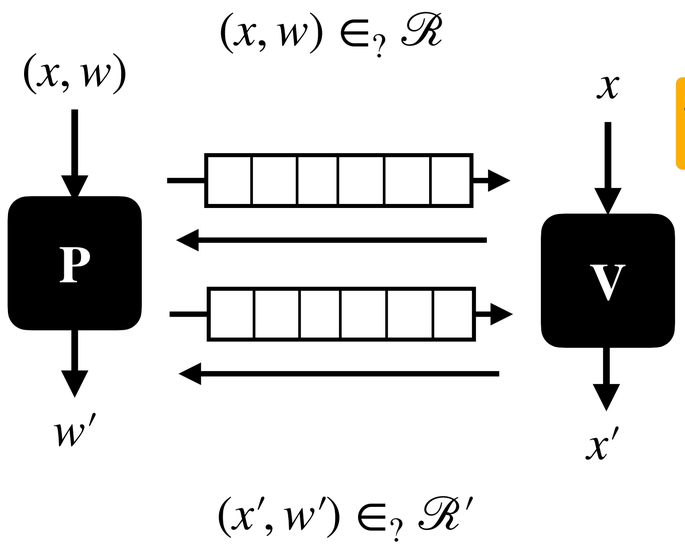
Interactive reduction
$\mathscr{R} \to \mathscr{R}'$



e.g. sumcheck protocol

Typically, want to reduce
$\mathscr{R}^{\ell} \to \mathscr{R}$

Interactive oracle reduction

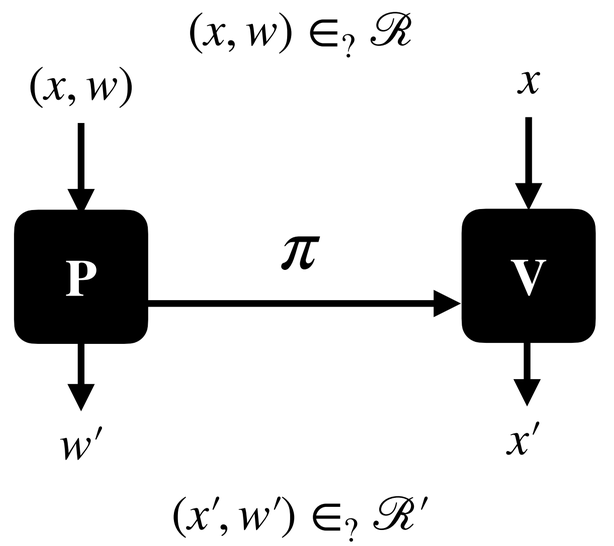


Oracles allow for succinct verification

Our focus!

Standard techniques: Merkle Trees + FS

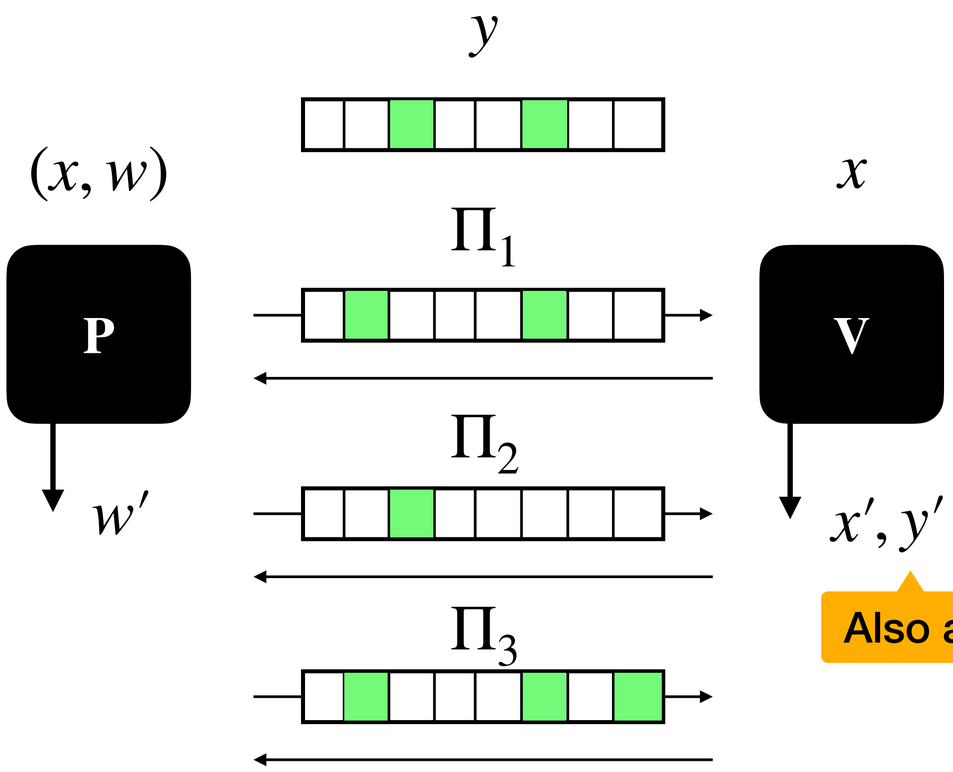
Hash-Based (Non-Interactive) Reduction



Core of hash-based accumulation schemes

IORs of Proximity

IOPP : ARG = IORP : ACC



Also an oracle

y' can depend on $(y, \Pi_1, \Pi_2, \ldots,)$

Completeness

If $(x, y, w) \in R$ then $(x', y', w') \in R$

Soundness

If $\Delta(y, R[x]) > \delta$ then w.h.p. $\Delta(y', R[x']) > \delta'$

Not enough must be knowledge-sound too

Not enough, must be state-restoration sound for FS security

Large, think 2^{20}

Proof length l $\approx O(k)$

Queries q $\approx O(\lambda)$

Small, think ~100

+ RO

Prover RO queries $O(\mathrm{l})$

Verifier RO queries $O(\mathrm{q} \cdot \log \mathrm{l})$

Accumulation from IORs

PESAT IOR_1

Reduce PESAT to proximity of an (encoded) witness to a relation

$$\mathscr{R}_{\mathsf{PESAT}}(\mathbb{F}) \to \mathscr{R}_{\mathsf{acc}}$$

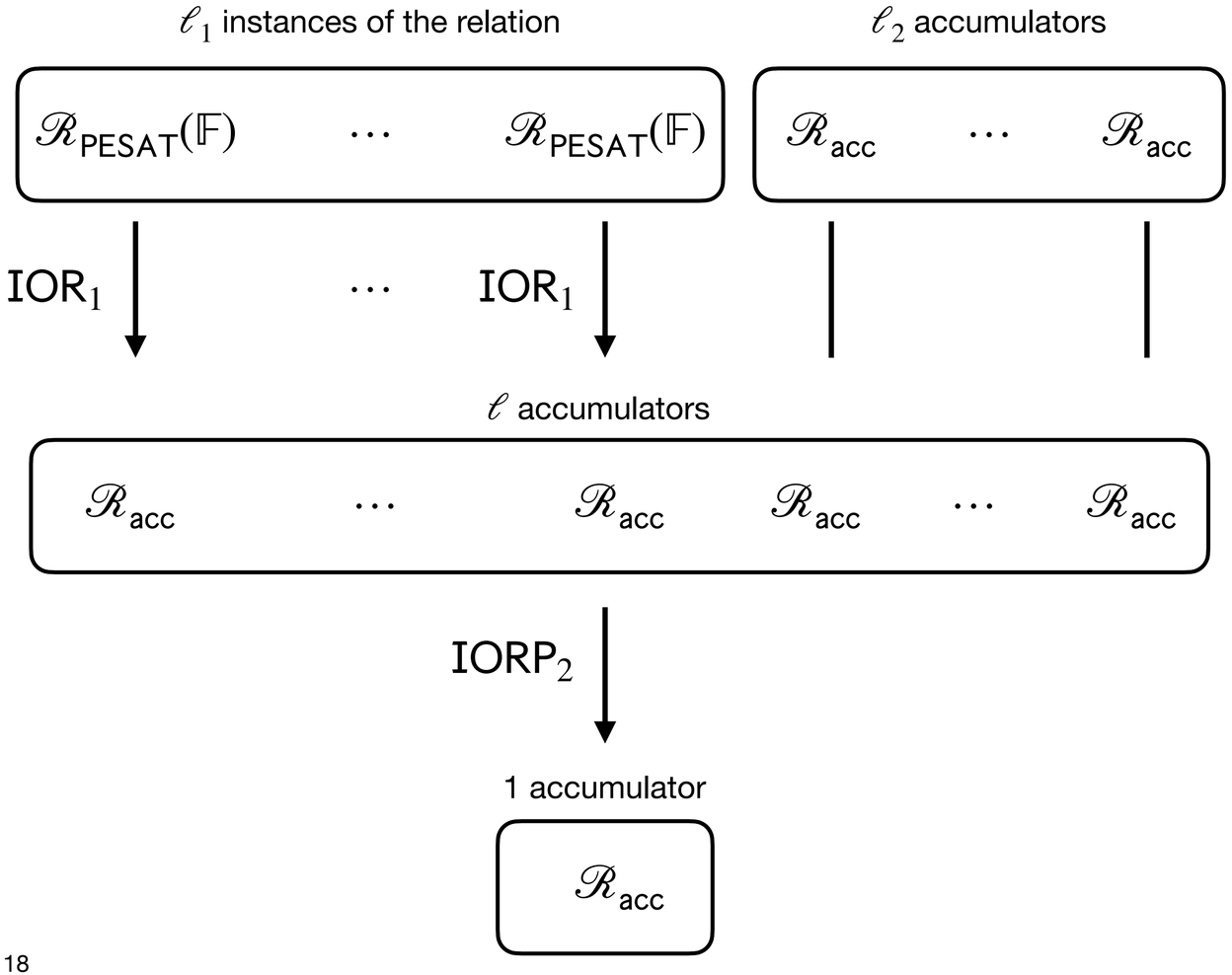
Batching IORP_2

Batches many instances of accumulation relation into a single one

$$\mathscr{R}_{\mathsf{acc}}^{\ell} \to \mathscr{R}_{\mathsf{acc}}$$

Hash-based accumulation constructed by compiling with Merkle Trees and Fiat-Shamir

Final IOR $\mathscr{R}_{\mathsf{PESAT}}(\mathbb{F})^{\ell_1} \times \mathscr{R}_{\mathsf{acc}}^{\ell_2} \to \mathscr{R}_{\mathsf{acc}}$



Conclusion

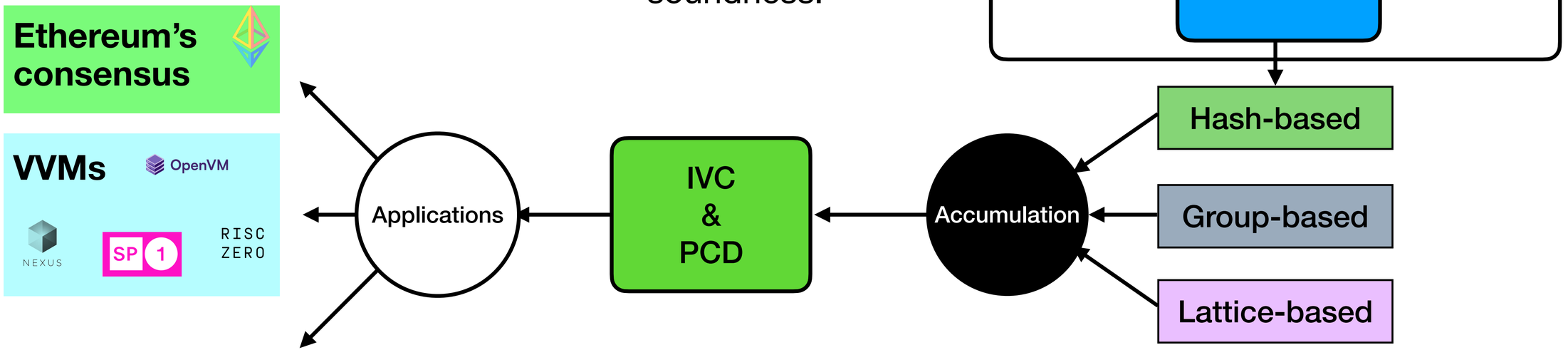
Recap

Lots I could not cover today!

Out of domain samples for general linear codes

Twin-constraint pseudobatching

New notions of round-by-round knowledge soundness!



Want to hear more?

William will present WARP 🌀!
May 12th in Toronto.
More details @ **zksummit.com**

Extra slides

Application: PQ-signature aggregation

Ethereum's consensus

- Ethereum's consensus requires validator to sign a message, which is aggregated to a single signature and distributed to the network. Currently using BLS signatures (**vulnerable** to quantum attacks).

- Replace the signature with hash-based XMSS. **Problem:** how to efficiently aggregate? No homomorphic structure to exploit.

Approach a): use pqSNARK to show:
$\forall i \in [T] : \ \mathsf{SigVfy}(\mathsf{pk}_i, m, \sigma_i)$

Pros:

- $|\pi| \ll T \cdot |\sigma_i|$
- PQ security

Cons:

- $|\pi| = O(T)$
- Memory usage is also $O(T)$

Approach b): use IVC with:
$F(i, \sigma_i) = \mathsf{SigVfy}(\mathsf{pk}_i, m, \sigma_i)$

- $|\pi|$ **independent** of T
- Memory usage also **independent of** T