geometry

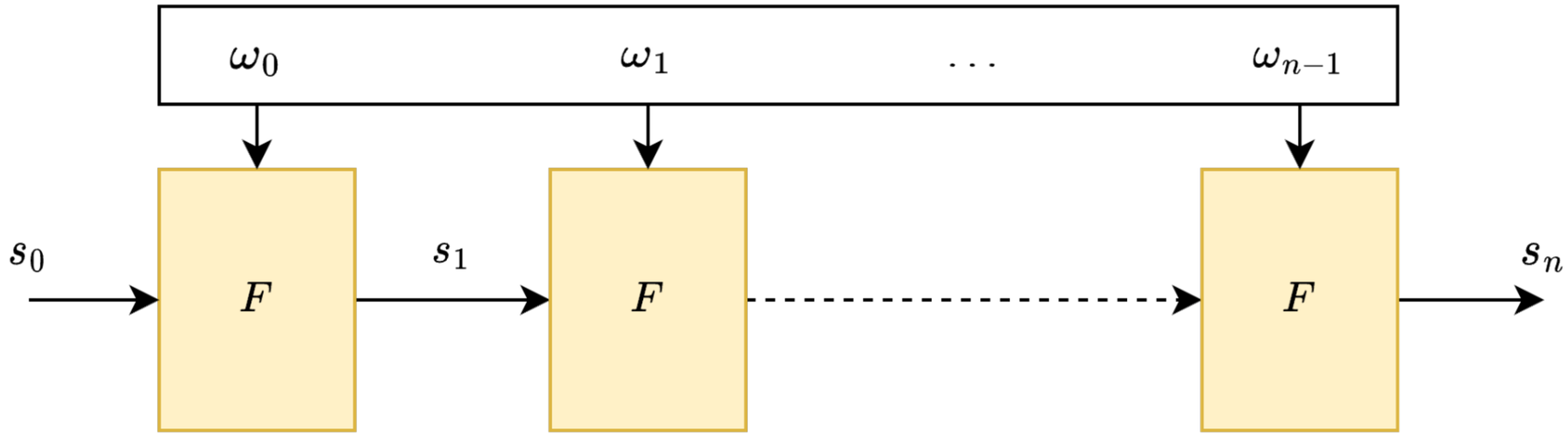
A Unifying Framework for Folding and Accumulation Schemes

Nicolas Mohnblatt

JUL | 23

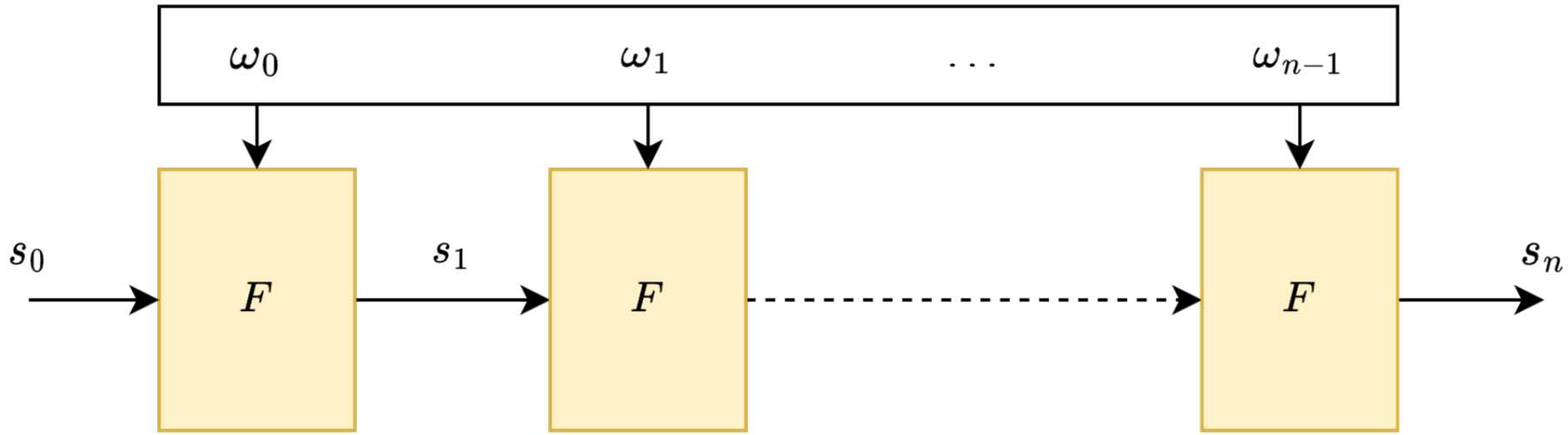
zkVM from Incrementally Verifiable Computation

Imagine a long computation that results from *n* applications of a function *F*:



zkVM from Incrementally Verifiable Computation

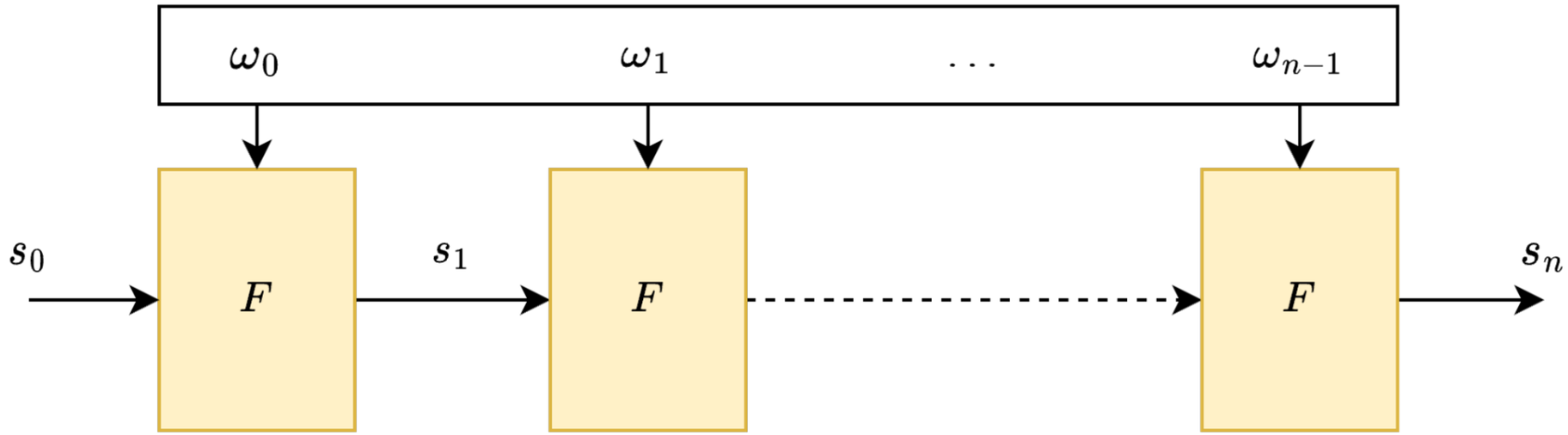
Imagine a long computation that results from *n* applications of a function *F*:



We want to produce a succinct proof that the prover knows $\omega_0, \omega_1, \ldots, \omega_{n-1}$ such that s_n is the correct final state

zkVM from Incrementally Verifiable Computation

Imagine a long computation that results from *n* applications of a function *F*:



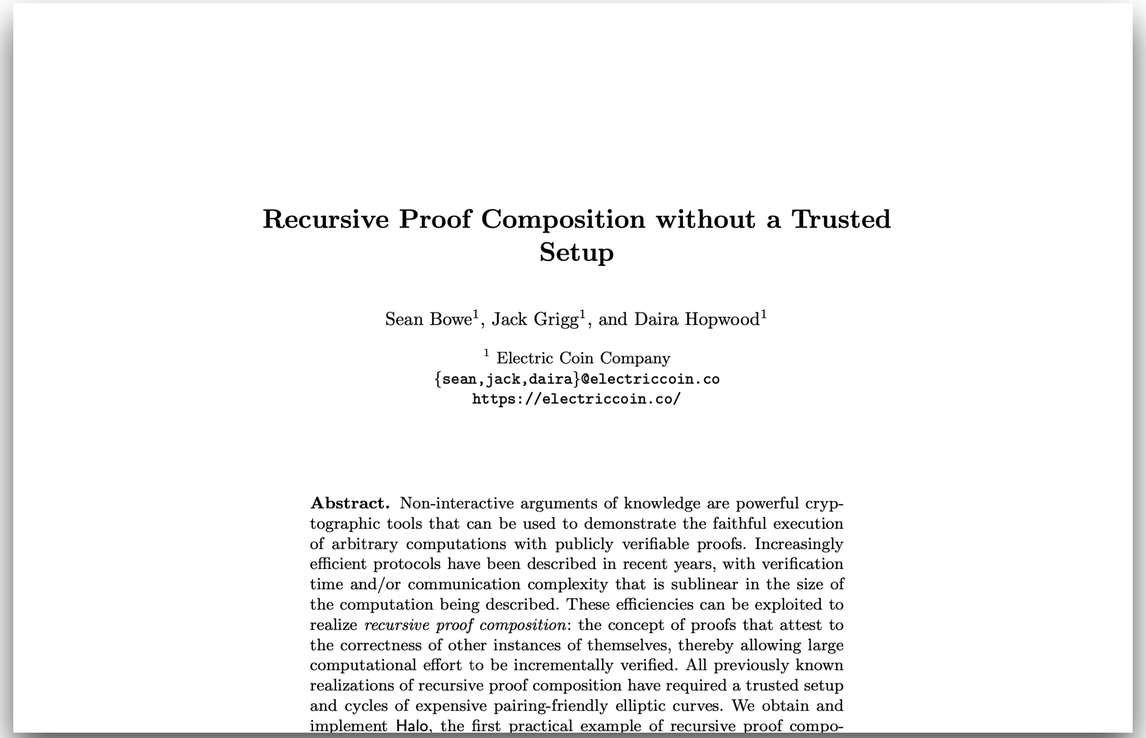
We want to produce a succinct proof that the prover knows $\omega_0, \omega_1, \ldots, \omega_{n-1}$ such that s_n is the correct final state

Setting *F* to be the transition function of a VM allows to produce proofs “as the VM advances”

How to build IVC?

How to build IVC?

Nested Amortization!



Recursive Proof Composition without a Trusted Setup

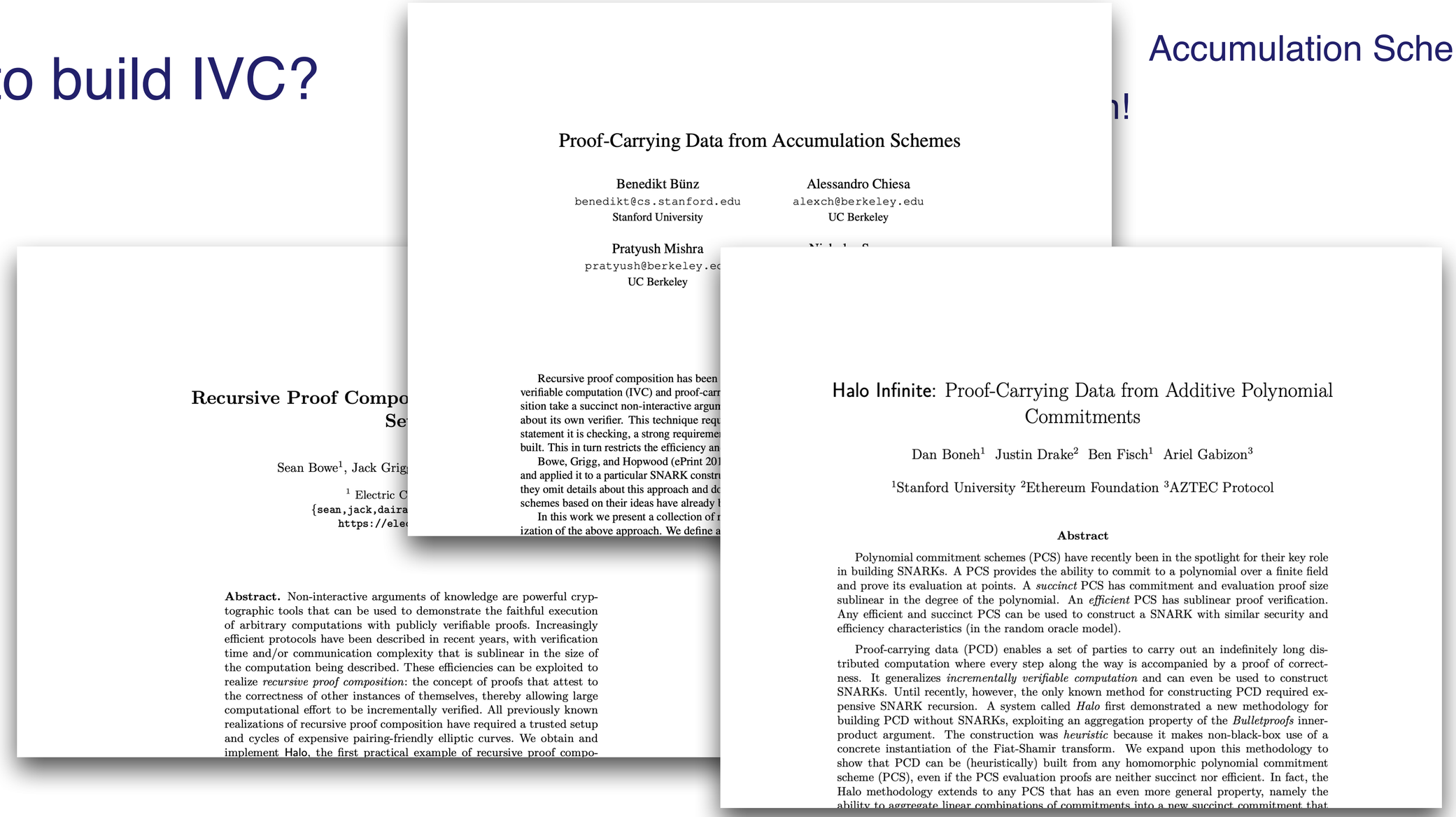
Sean Bowe[1], Jack Grigg[1], and Daira Hopwood[1]

[1] Electric Coin Company
{sean,jack,daira}@electriccoin.co
https://electriccoin.co/

Abstract. Non-interactive arguments of knowledge are powerful cryptographic tools that can be used to demonstrate the faithful execution of arbitrary computations with publicly verifiable proofs. Increasingly efficient protocols have been described in recent years, with verification time and/or communication complexity that is sublinear in the size of the computation being described. These efficiencies can be exploited to realize *recursive proof composition*: the concept of proofs that attest to the correctness of other instances of themselves, thereby allowing large computational effort to be incrementally verified. All previously known realizations of recursive proof composition have required a trusted setup and cycles of expensive pairing-friendly elliptic curves. We obtain and implement Halo, the first practical example of recursive proof compo-

How to build IVC?

Accumulation Schemes!



Recursive Proof Compo
Se

Sean Bowe[1], Jack Grig

[1] Electric C
{sean,jack,daira
https://ele

Abstract. Non-interactive arguments of knowledge are powerful cryptographic tools that can be used to demonstrate the faithful execution of arbitrary computations with publicly verifiable proofs. Increasingly efficient protocols have been described in recent years, with verification time and/or communication complexity that is sublinear in the size of the computation being described. These efficiencies can be exploited to realize *recursive proof composition*: the concept of proofs that attest to the correctness of other instances of themselves, thereby allowing large computational effort to be incrementally verified. All previously known realizations of recursive proof composition have required a trusted setup and cycles of expensive pairing-friendly elliptic curves. We obtain and implement Halo, the first practical example of recursive proof compo-

Proof-Carrying Data from Accumulation Schemes

Benedikt Bünz
benedikt@cs.stanford.edu
Stanford University

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Pratyush Mishra
pratyush@berkeley.ed
UC Berkeley

Recursive proof composition has been
verifiable computation (IVC) and proof-carr
sition take a succinct non-interactive argun
about its own verifier. This technique requ
statement it is checking, a strong requireme
built. This in turn restricts the efficiency an
Bowe, Grigg, and Hopwood (ePrint 20
and applied it to a particular SNARK constr
they omit details about this approach and d
schemes based on their ideas have already
In this work we present a collection of
ization of the above approach. We define a

Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments

Dan Boneh[1] Justin Drake[2] Ben Fisch[1] Ariel Gabizon[3]

[1]Stanford University [2]Ethereum Foundation [3]AZTEC Protocol

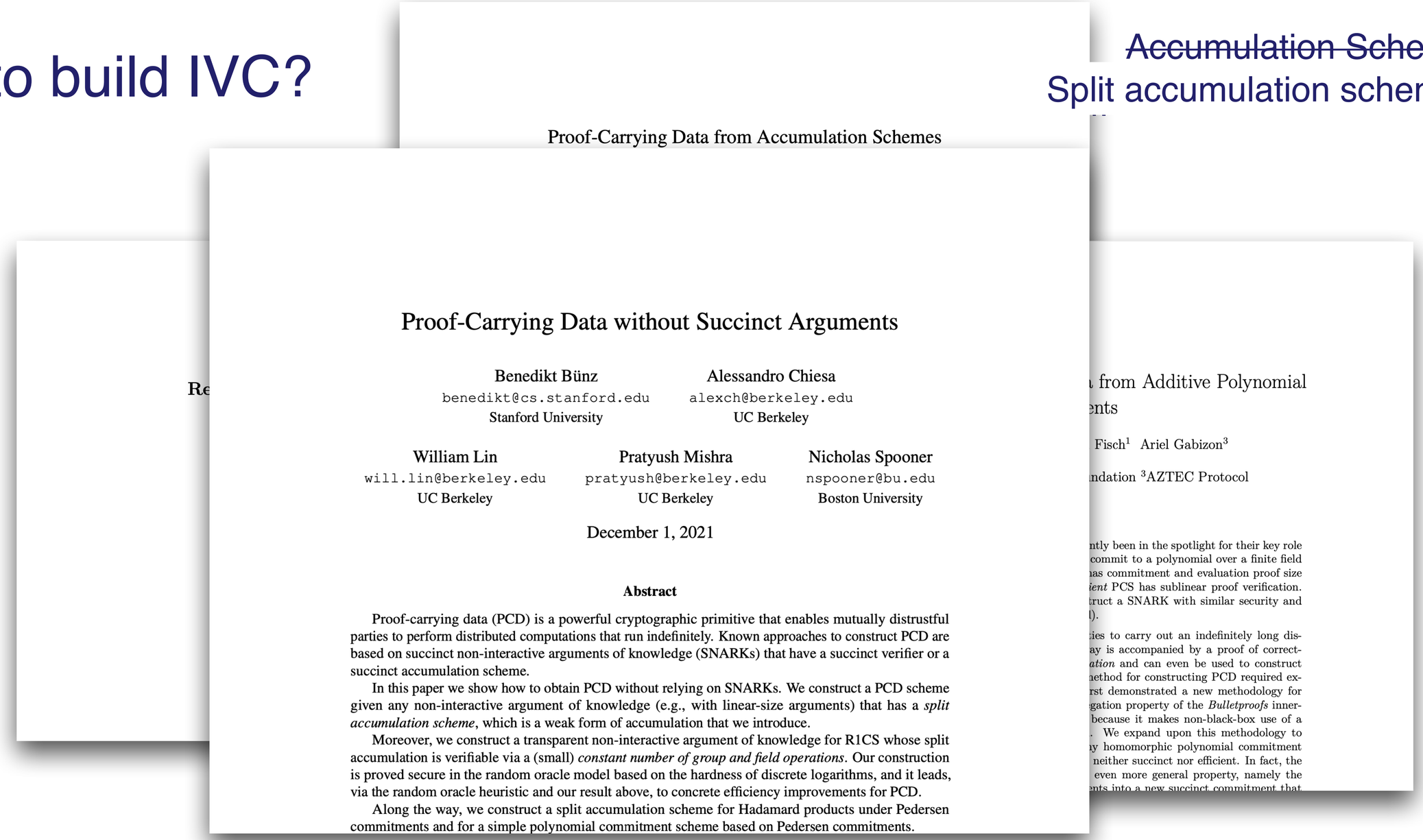
Abstract

Polynomial commitment schemes (PCS) have recently been in the spotlight for their key role in building SNARKs. A PCS provides the ability to commit to a polynomial over a finite field and prove its evaluation at points. A *succinct* PCS has commitment and evaluation proof size sublinear in the degree of the polynomial. An *efficient* PCS has sublinear proof verification. Any efficient and succinct PCS can be used to construct a SNARK with similar security and efficiency characteristics (in the random oracle model).

Proof-carrying data (PCD) enables a set of parties to carry out an indefinitely long distributed computation where every step along the way is accompanied by a proof of correctness. It generalizes *incrementally verifiable computation* and can even be used to construct SNARKs. Until recently, however, the only known method for constructing PCD required expensive SNARK recursion. A system called *Halo* first demonstrated a new methodology for building PCD without SNARKs, exploiting an aggregation property of the *Bulletproofs* inner-product argument. The construction was *heuristic* because it makes non-black-box use of a concrete instantiation of the Fiat-Shamir transform. We expand upon this methodology to show that PCD can be (heuristically) built from any homomorphic polynomial commitment scheme (PCS), even if the PCS evaluation proofs are neither succinct nor efficient. In fact, the Halo methodology extends to any PCS that has an even more general property, namely the ability to aggregate linear combinations of commitments into a new succinct commitment that

How to build IVC?

~~Accumulation Schemes!~~
Split accumulation schemes!



Proof-Carrying Data from Accumulation Schemes

Proof-Carrying Data without Succinct Arguments

Benedikt Bünz
benedikt@cs.stanford.edu
Stanford University

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

William Lin
will.lin@berkeley.edu
UC Berkeley

Pratyush Mishra
pratyush@berkeley.edu
UC Berkeley

Nicholas Spooner
nspooner@bu.edu
Boston University

December 1, 2021

Abstract

Proof-carrying data (PCD) is a powerful cryptographic primitive that enables mutually distrustful parties to perform distributed computations that run indefinitely. Known approaches to construct PCD are based on succinct non-interactive arguments of knowledge (SNARKs) that have a succinct verifier or a succinct accumulation scheme.

In this paper we show how to obtain PCD without relying on SNARKs. We construct a PCD scheme given any non-interactive argument of knowledge (e.g., with linear-size arguments) that has a *split accumulation scheme*, which is a weak form of accumulation that we introduce.

Moreover, we construct a transparent non-interactive argument of knowledge for R1CS whose split accumulation is verifiable via a (small) *constant number of group and field operations*. Our construction is proved secure in the random oracle model based on the hardness of discrete logarithms, and it leads, via the random oracle heuristic and our result above, to concrete efficiency improvements for PCD.

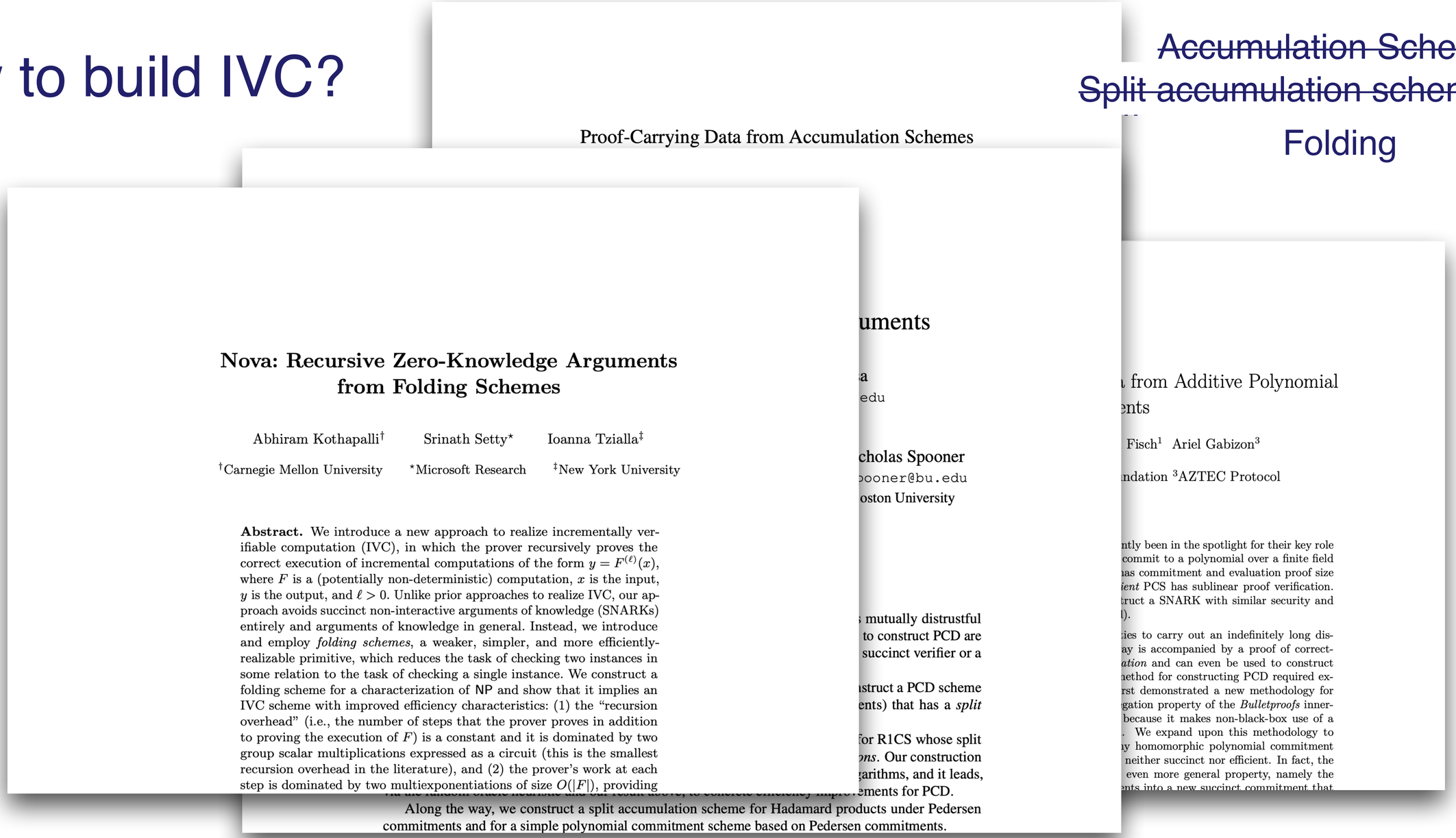
Along the way, we construct a split accumulation scheme for Hadamard products under Pedersen commitments and for a simple polynomial commitment scheme based on Pedersen commitments.

How to build IVC?

~~Accumulation Schemes!~~

~~Split accumulation schemes!~~

Folding



Proof-Carrying Data from Accumulation Schemes

uments

a

edu

cholas Spooner

ooner@bu.edu

oston University

mutually distrustful

to construct PCD are

succinct verifier or a

struct a PCD scheme

ents) that has a *split*

for R1CS whose split

ons. Our construction

garithms, and it leads,

ements for PCD.

Along the way, we construct a split accumulation scheme for Hadamard products under Pedersen commitments and for a simple polynomial commitment scheme based on Pedersen commitments.

from Additive Polynomial

ents

Fisch[1] Ariel Gabizon[3]

ndation [3]AZTEC Protocol

ntly been in the spotlight for their key role

commit to a polynomial over a finite field

has commitment and evaluation proof size

ient PCS has sublinear proof verification.

truct a SNARK with similar security and

).

ies to carry out an indefinitely long dis-

ay is accompanied by a proof of correct-

ation and can even be used to construct

ethod for constructing PCD required ex-

rst demonstrated a new methodology for

gation property of the *Bulletproofs* inner-

because it makes non-black-box use of a

. We expand upon this methodology to

y homomorphic polynomial commitment

neither succinct nor efficient. In fact, the

even more general property, namely the

nts into a new succinct commitment that

Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[†] Srinath Setty[⋆] Ioanna Tzialla[‡]

[†]Carnegie Mellon University [⋆]Microsoft Research [‡]New York University

Abstract. We introduce a new approach to realize incrementally verifiable computation (IVC), in which the prover recursively proves the correct execution of incremental computations of the form $y = F^{(\ell)}(x)$, where F is a (potentially non-deterministic) computation, x is the input, y is the output, and $\ell > 0$. Unlike prior approaches to realize IVC, our approach avoids succinct non-interactive arguments of knowledge (SNARKs) entirely and arguments of knowledge in general. Instead, we introduce and employ *folding schemes*, a weaker, simpler, and more efficiently-realizable primitive, which reduces the task of checking two instances in some relation to the task of checking a single instance. We construct a folding scheme for a characterization of NP and show that it implies an IVC scheme with improved efficiency characteristics: (1) the "recursion overhead" (i.e., the number of steps that the prover proves in addition to proving the execution of F) is a constant and it is dominated by two group scalar multiplications expressed as a circuit (this is the smallest recursion overhead in the literature), and (2) the prover's work at each step is dominated by two multiexponentiations of size $O(|F|)$, providing

How to build IVC?

~~Accumulation Schemes!~~
~~Split accumulation schemes!~~
Folding
Fooooolding !!!

Proof-Carrying Data from Accumulation Schemes

Nova: Recursive Zero-Knowledge Argu
from Folding Schemes

HyperNova: Recursive arguments for customizable constraint systems

Abhiram Kothapalli[†] Srinath Setty[⋆]

[†]Carnegie Mellon University [⋆]Microsoft Research

Abstract.

This paper introduces *HyperNova*, a recursive argument for provi
mental computations whose steps are expressed with CCS (Setty et a
2023), a customizable constraint system that simultaneously generalize
ish, R1CS, and AIR without overheads. A distinguishing aspect of Hy
is that the prover's cost at each step is dominated by a *single* mul
multiplication (MSM) of size equal to the number of variables in the c
system, which is optimal when using an MSM-based commitment sch

To construct HyperNova, we generalize folding schemes (CRYPTO
allow instances from two (potentially) different NP relations, that share t
structure, to be folded; we refer to this generalization as *multi-folding*

Sangria: A Folding Scheme for PLONK

Nicolas Mohnblatt
Geometry
nico@geometry.xyz

April 17, 2023

PROTOSTAR: Generic Efficient Accumulation/Folding for Special-sound Protocols

Benedikt Bünz Binyi Chen
Stanford University Espresso Systems

How to build IVC?

~~Accumulation Schemes!~~

~~Split accumulation schemes!~~

Folding

Fooooolding !!!

Schemes

Nova: Recursive Z

HyperNova: Recursive argumen
customizable constraint syste

Abhiram Kothapalli† Srinath Se

†Carnegie Mellon University *Microsoft Re

Abstract.

This paper introduces *HyperNova*, a recursive argument f
mental computations whose steps are expressed with CCS (Se
2023), a customizable constraint system that simultaneously g
ish, R1CS, and AIR without overheads. A distinguishing aspe
is that the prover's cost at each step is dominated by a *sin*
multiplication (MSM) of size equal to the number of variables
system, which is optimal when using an MSM-based commitr

To construct HyperNova, we generalize folding schemes (C
allow instances from two (potentially) different NP relations, that share t
structure, to be folded; we refer to this generalization as *multi-folding*

Sangria: A Folding Scheme for PLONK

Nicolas Mohnblatt
Geometry
nico@geometry.xyz

April 17, 2023

fficient Accumulation/Folding for
sound Protocols

Benedikt Bünz Binyi Chen

Stanford University Espresso Systems

How to build IVC? (continued)

Research is moving fast, how do we choose which scheme to implement?

How should we understand these schemes? Mental model?

How can we compare all these schemes? In theory and in practice

How to build IVC? (continued)

Research is moving fast, how do we choose which scheme to implement?

How should we understand these schemes? Mental model?

How can we compare all these schemes? In theory and in practice

Claim: all these schemes fall under the same framework/pattern, **reduce-and-combine**

Part 1 Finding the Pattern

Halo2 - Accumulation [BGH19, ZcashDoc]

Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that

$C_{PLONKish}(x, w) = 1$

Halo2 - Accumulation [BGH19, ZcashDoc]

I know *w* such that

$C_{PLONKish}(x, w) = 1$

q_L	q_R	q_O	q_M	q_C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

rule 1: copy constrains

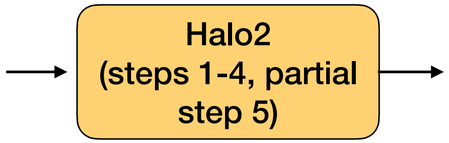
rule 2: row equation

rule 3: lookup

Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that

$C_{PLONKish}(x, w) = 1$



polynomial commitment

opening proof

q_L	q_R	q_O	q_M	q_C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

rule 1: copy constrains

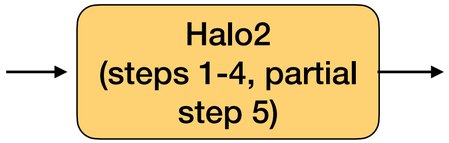
rule 2: row equation

rule 3: lookup

Halo2 - Accumulation [BGH19, ZcashDoc]

I know *w* such that

$C_{PLONKish}(x, w) = 1$



polynomial commitment

opening proof

q_L	q_R	q_O	q_M	q_C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

rule 1: copy constrains

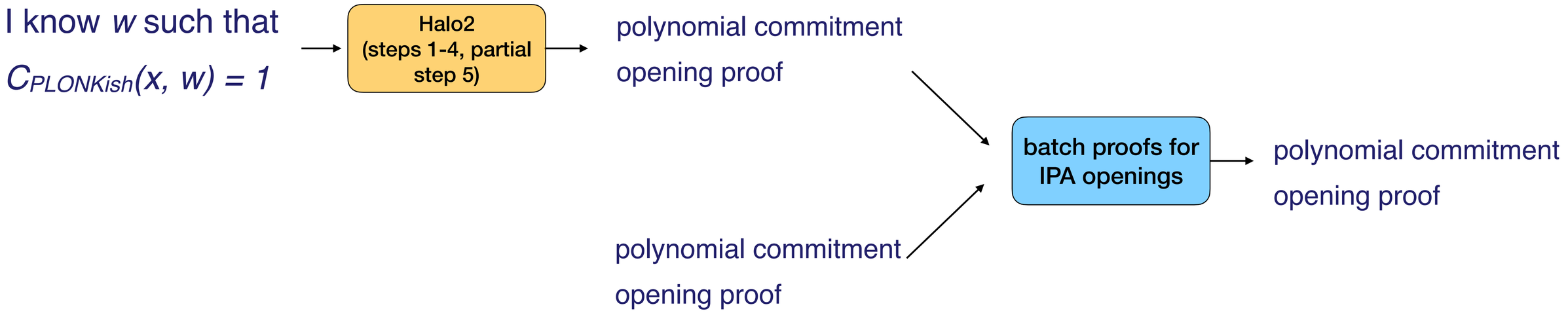
rule 2: row equation

rule 3: lookup

The prover will compute a grand product argument, the “quotient” polynomial, etc. Requires some FFTs and polynomial commitments.

The verifier will produce random challenges, linear combination of commitments, check some evaluations and part of the IPA opening

Halo2 - Accumulation [BGH19, ZcashDoc]



[BCLMS21] - Split Accumulation for R1CS

[BCLMS21] - Split Accumulation for R1CS

I know w such that

$C_{R1CS}(x, w) = 1$

[BCLMS21] - Split Accumulation for R1CS

I know w such that

$C_{R1CS}(x, w) = 1$

Recall R1CS:
$z = x \| w$
$Az \circ Bz = Cz$

[BCLMS21] - Split Accumulation for R1CS

I know *w* such that

$C_{R1CS}(x, w) = 1$

Recall R1CS:
z = x‖w
Az ∘ Bz = Cz

The almost-trivial NARK*

$\mathcal{P}(\mathsf{ck}, (A, B, C), x, w)$

$z := (x, w) \in \mathbb{F}^{\mathsf{N}}$
$z_A := Az \in \mathbb{F}^{\mathsf{M}} \quad C_A := \mathsf{Commit}(\mathsf{ck}, z_A) \in \mathbb{G}$
$z_B := Bz \in \mathbb{F}^{\mathsf{M}} \quad C_B := \mathsf{Commit}(\mathsf{ck}, z_B) \in \mathbb{G}$
$z_C := Cz \in \mathbb{F}^{\mathsf{M}} \quad C_C := \mathsf{Commit}(\mathsf{ck}, z_C) \in \mathbb{G}$

$\xrightarrow{\quad C_A, C_B, C_C, w \quad}$

$\mathcal{V}(\mathsf{ck}, (A, B, C), x)$

$z := (x, w)$
$z_A := Az \quad C_A \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, z_A)$
$z_B := Bz \quad C_B \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, z_B)$
$z_C := Cz \quad C_C \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, z_C)$
$C_C \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, z_A \circ z_B)$

*the paper extends this basic idea to also provide zero-knowledge

[BCLMS21] - Split Accumulation for R1CS

I know *w* such that

$C_{R1CS}(x, w) = 1$

I know openings to the commitments C_A C_B C_C

and

C_C is a commitment to the Hadamard product of the vectors committed in C_A and C_B

[BCLMS21] - Split Accumulation for R1CS

I know w such that

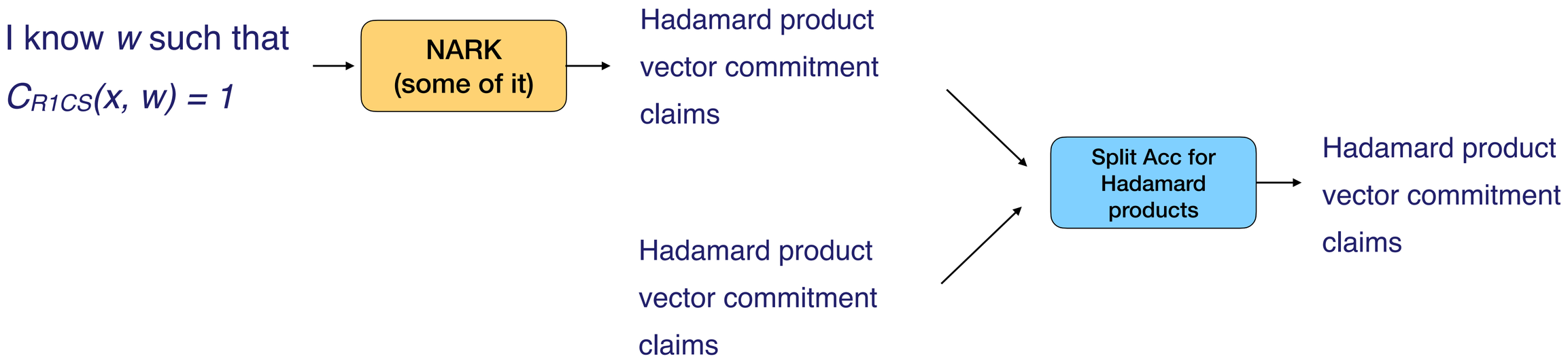
$C_{R1CS}(x, w) = 1$

Hadamard product

vector commitment

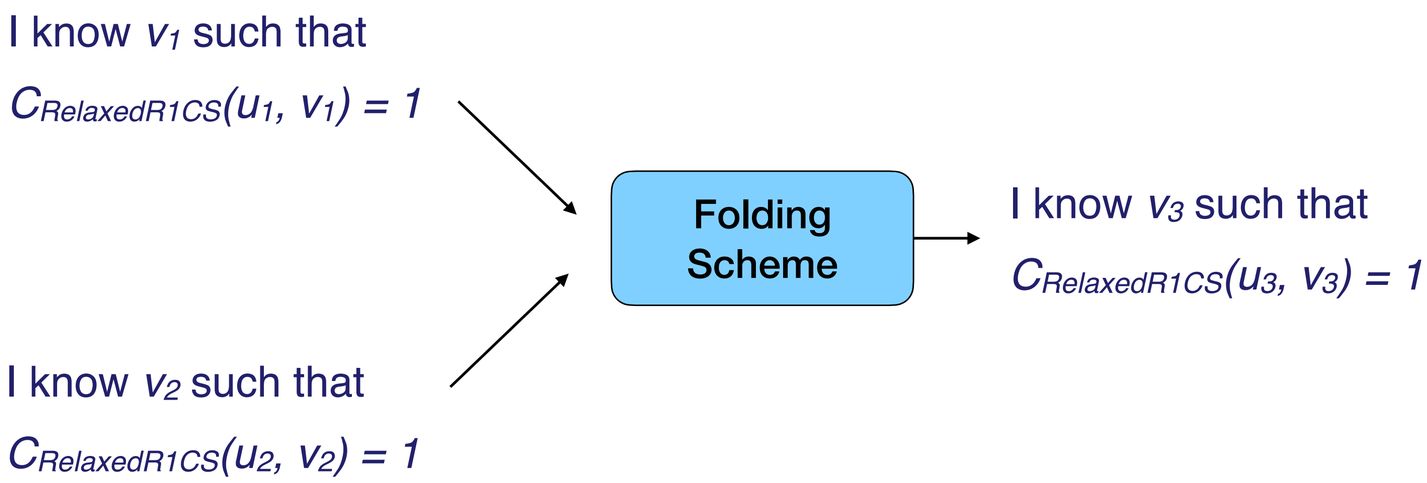
claims

[BCLMS21] - Split Accumulation for R1CS

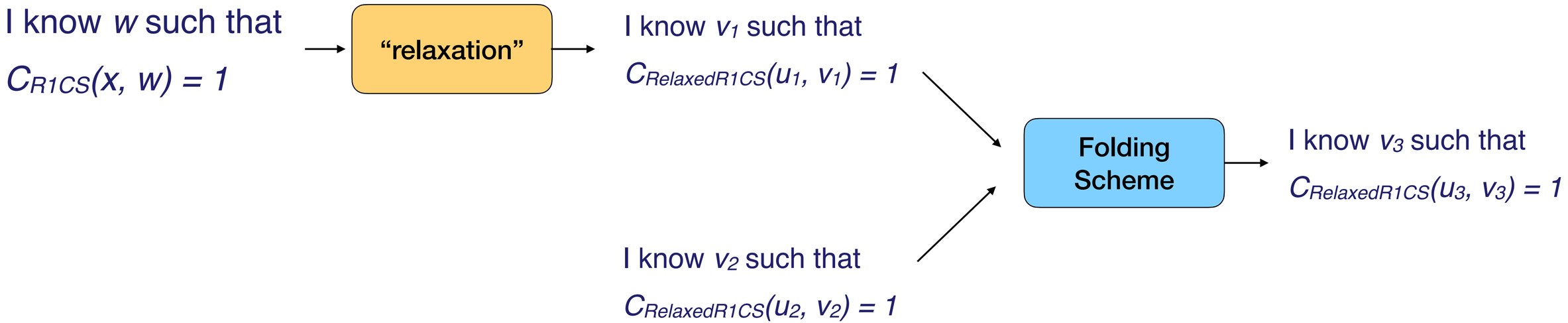


Nova [KST22]

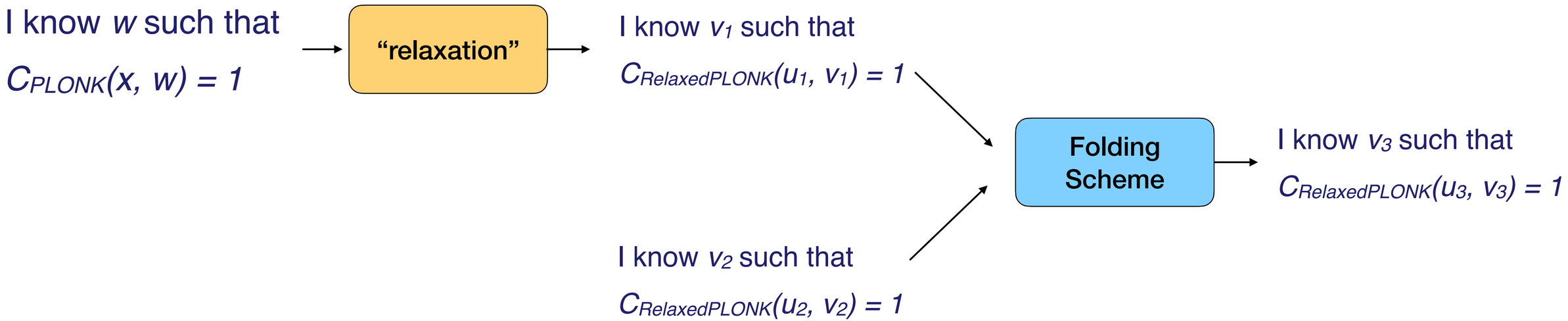
Nova [KST22]


I know v_1 such that
$C_{RelaxedR1CS}(u_1, v_1) = 1$
I know v_2 such that
$C_{RelaxedR1CS}(u_2, v_2) = 1$
Folding Scheme
I know v_3 such that
$C_{RelaxedR1CS}(u_3, v_3) = 1$

Nova [KST22]


I know w such that
$C_{R1CS}(x, w) = 1$
"relaxation"
I know v_1 such that
$C_{RelaxedR1CS}(u_1, v_1) = 1$
I know v_2 such that
$C_{RelaxedR1CS}(u_2, v_2) = 1$
Folding Scheme
I know v_3 such that
$C_{RelaxedR1CS}(u_3, v_3) = 1$

Sangria [Moh23]



The Reduce-and-Combine Pattern

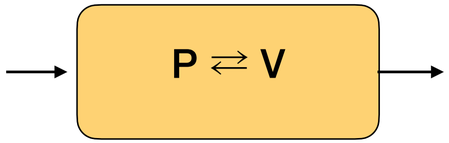
I know w such that

$(x, w) \in \mathcal{R}$

The Reduce-and-Combine Pattern

I know w such that

$(x, w) \in \mathcal{R}$

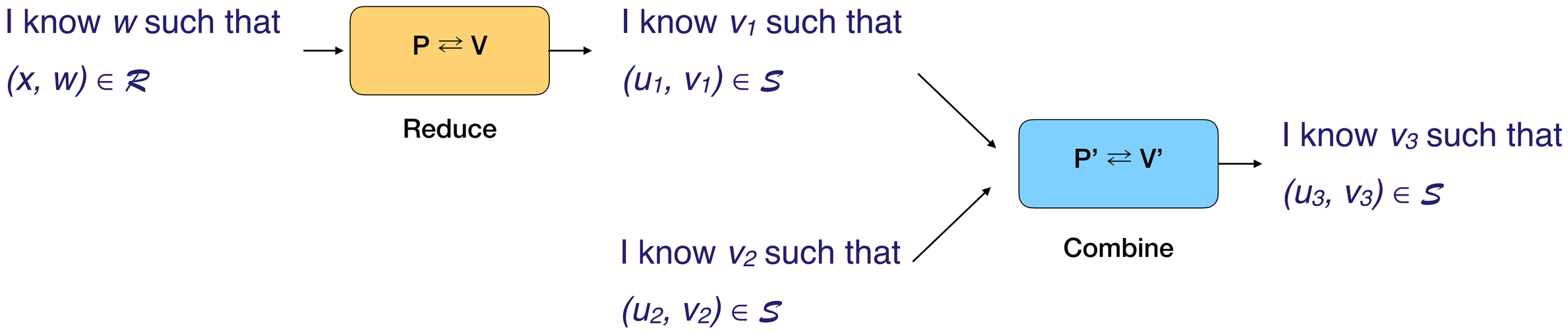


Reduce

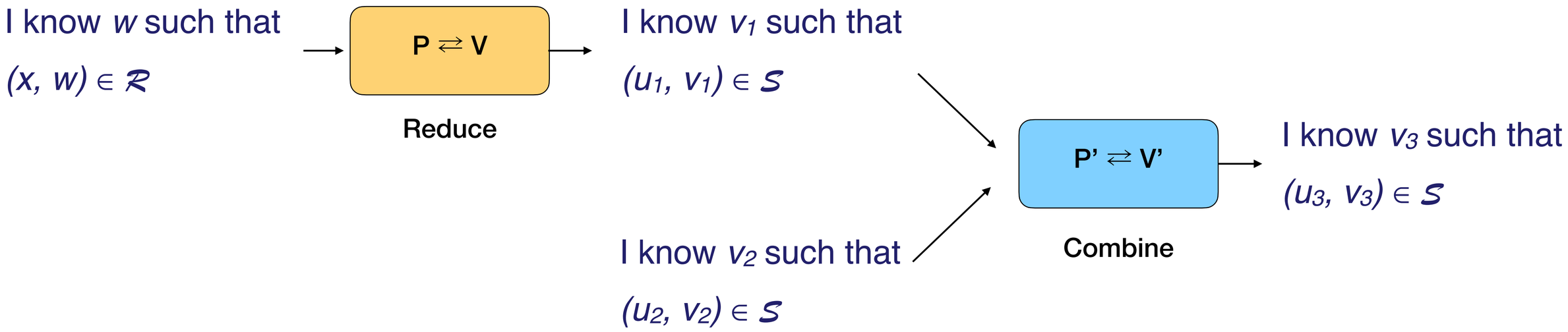
I know v_1 such that

$(u_1, v_1) \in \mathcal{S}$

The Reduce-and-Combine Pattern



The Reduce-and-Combine Pattern

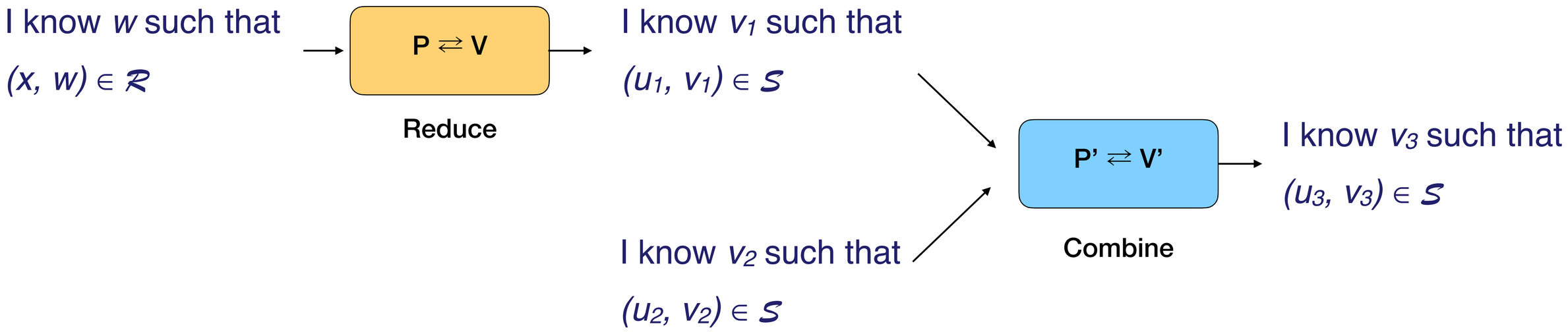


- Choosing $\mathcal{R}$ and $\mathcal{S}$ determines how much work each party does in the “Reduce” and the “Combine” stage
- At each IVC step, the IVC prover will perform the work of (P, P’) and evaluate a circuit performing (V, V’)

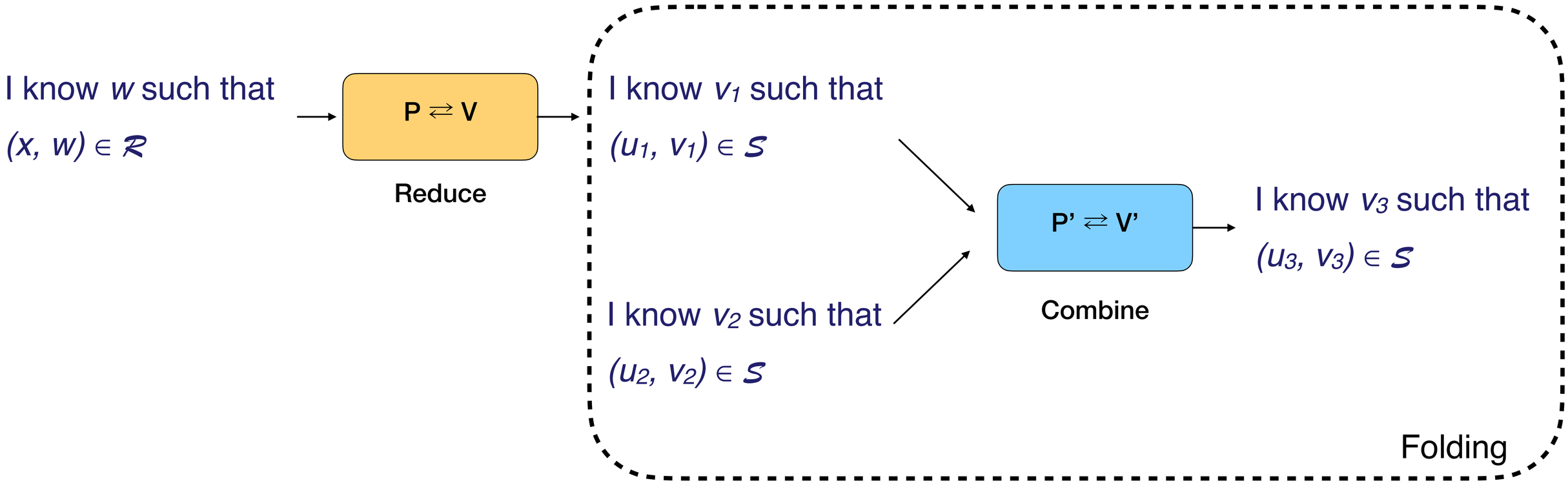
Comparing the examples

Protocol	$\mathcal{R}$	$\mathcal{S}$	“Reduce” work	“Combine” work
Halo2-IPA	PLONKish / UltraPLONK	IPA polynomial opening proofs	P: grand product, quotient poly (FFTs and commitments) V: produce challenges, linear comb. of comm., check evaluations and partial IPA opening	P: random linear combination and opening proof V: random linear combination
BCLSM21	R1CS	Hadamard product vector commitment claims	P: commit to the matrix-vector products V: none	P: commit to the error term V: add commitments with 1 error term
Nova	R1CS	committed relaxed R1CS	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term
Sangria	PLONK	committed relaxed PLONK	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term

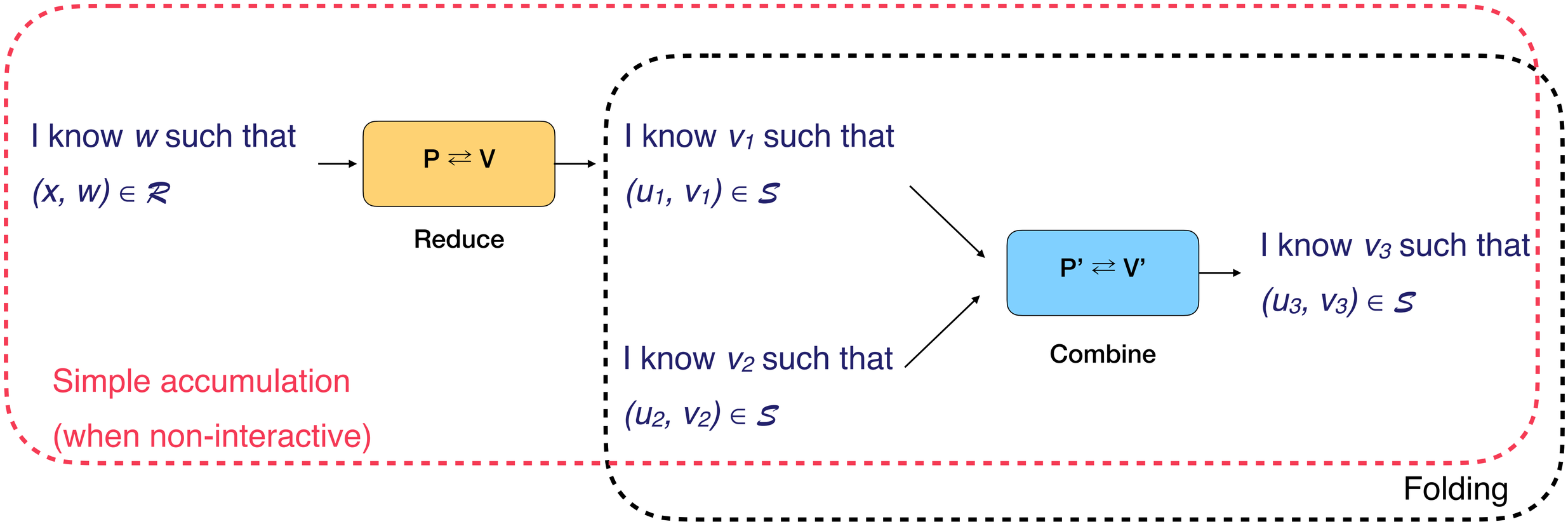
Mapping existing definitions



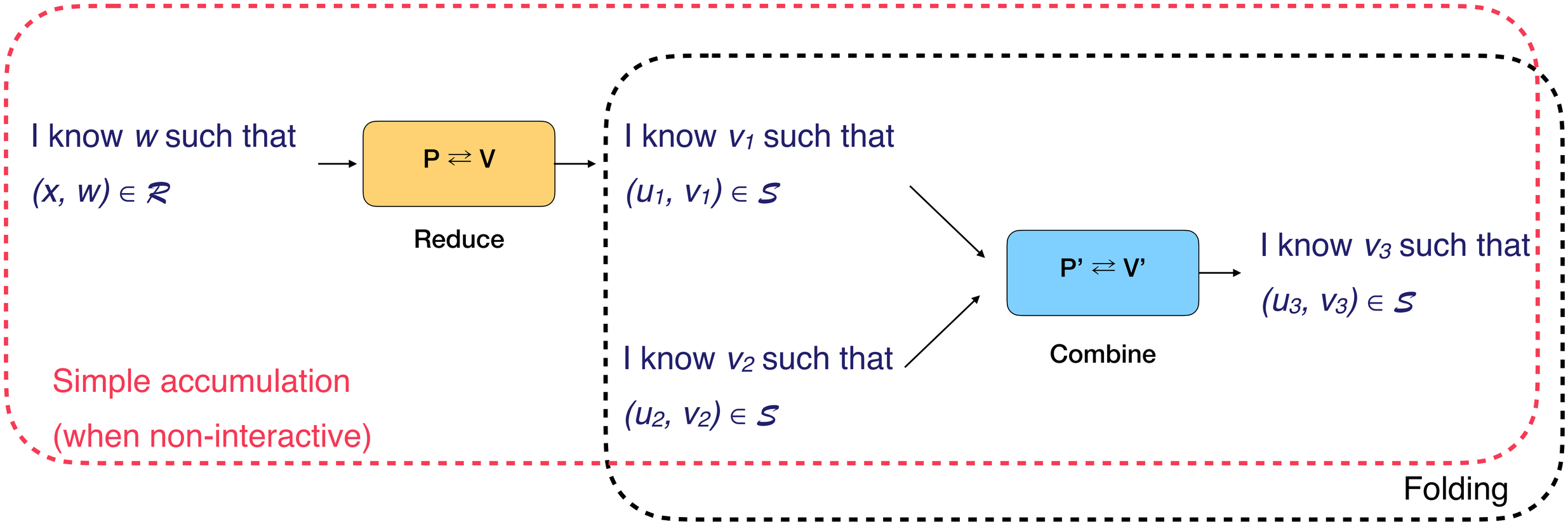
Mapping existing definitions



Mapping existing definitions



Mapping existing definitions



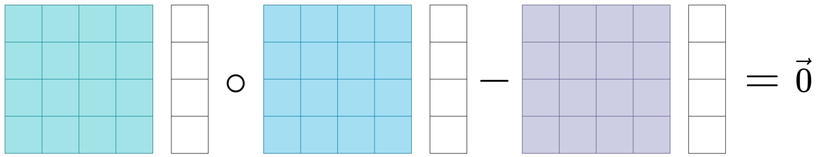
Split accumulation and multi-folding consider the case of a many-to-1 combination

Part 2 Understanding the Latest Constructions

HyperNova [KS23]

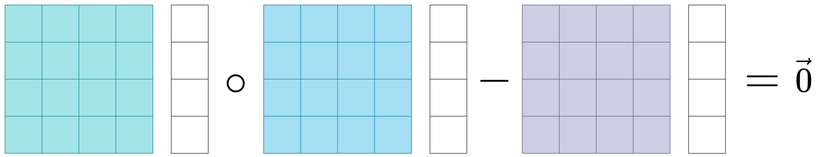
HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$Az \circ Bz - Cz = 0$$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

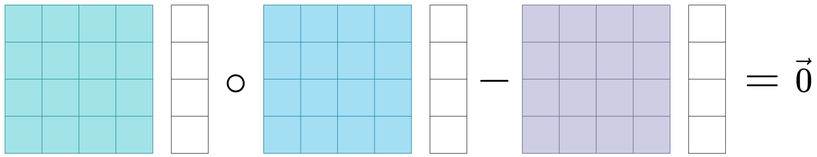
$$Az \circ Bz - Cz = 0$$



Spartan [Set20] is a two-layered sumcheck based proof for R1CS

HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

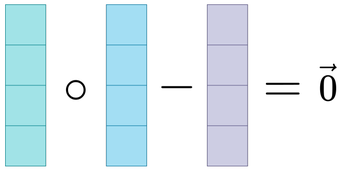
$$Az \circ Bz - Cz = 0$$



Spartan [Set20] is a two-layered sumcheck based proof for R1CS

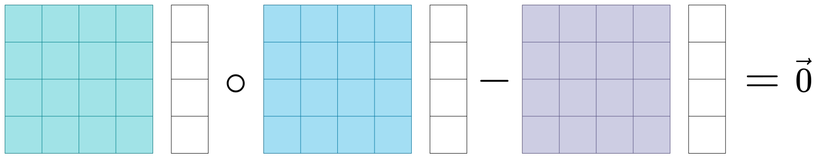
Sumcheck 1 (zero check):

$$a \circ b - c = 0$$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

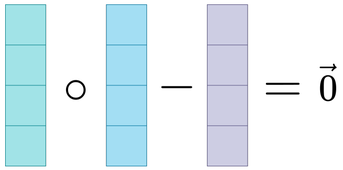
$$Az \circ Bz - Cz = 0$$



Spartan [Set20] is a two-layered sumcheck based proof for R1CS

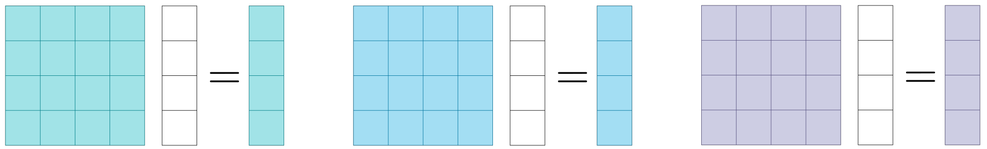
Sumcheck 1 (zero check):

$$a \circ b - c = 0$$



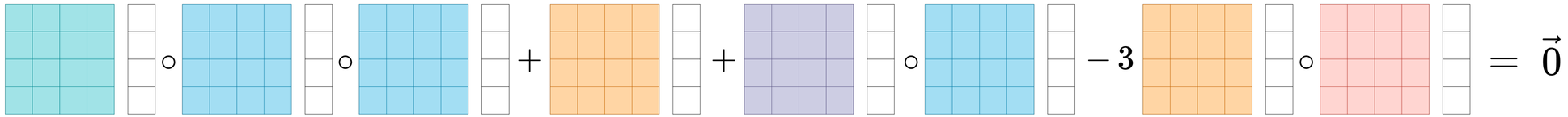
Sumcheck(s) 2 (“linchecks”, i.e. matrix-vector products):

$$Az = a,\ Bz = b,\ Cz = c$$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

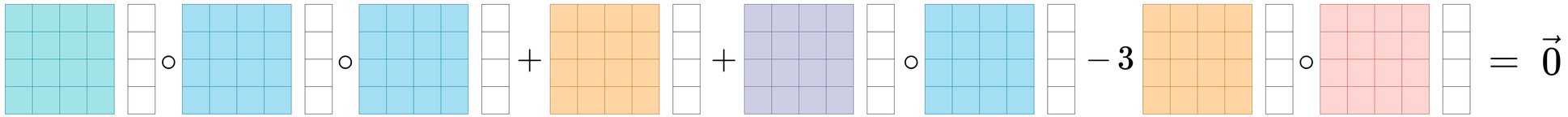
$M_1z \circ M_2z \circ M_2z + M_3z + M_4z \circ M_2z - 3\, M_3z \circ M_5z = 0$



$$\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} M_j \cdot z = \mathbf{0},$$

HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

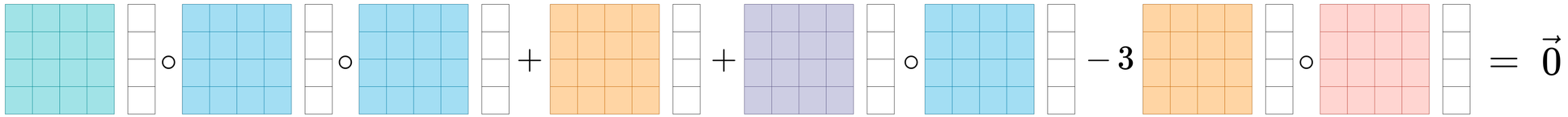
$M_1z \circ M_2z \circ M_2z + M_3z + M_4z \circ M_2z - 3\, M_3z \circ M_5z = 0$



SuperSpartan is a two-layered sumcheck based proof for CCS

HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

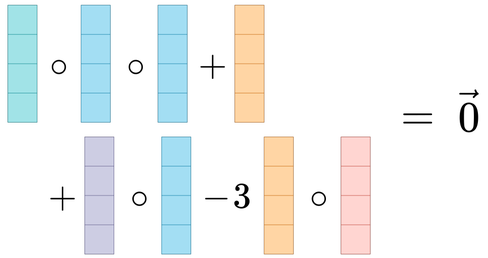
$M_1z \circ M_2z \circ M_2z + M_3z + M_4z \circ M_2z - 3\, M_3z \circ M_5z = 0$



SuperSpartan is a two-layered sumcheck based proof for CCS

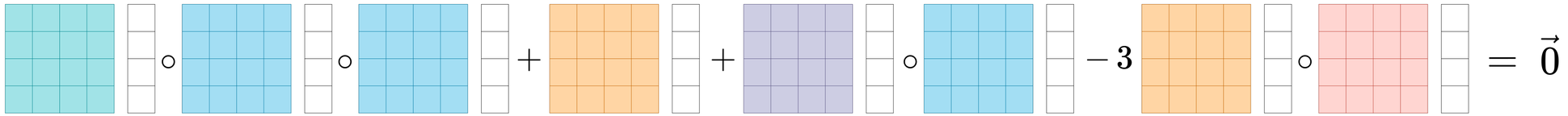
Sumcheck 1 (zero check):

$m_1 \circ m_2 \circ m_2 + m_3 + m_4 \circ m_2 - 3\, m_3 \circ m_5 = 0$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

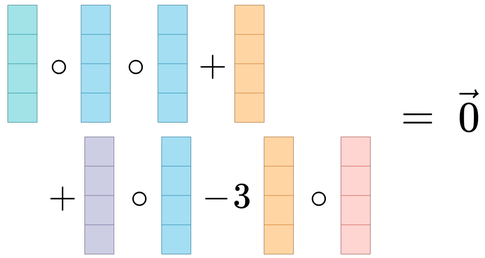
$M_1z \circ M_2z \circ M_2z + M_3z + M_4z \circ M_2z - 3\, M_3z \circ M_5z = 0$



SuperSpartan is a two-layered sumcheck based proof for CCS

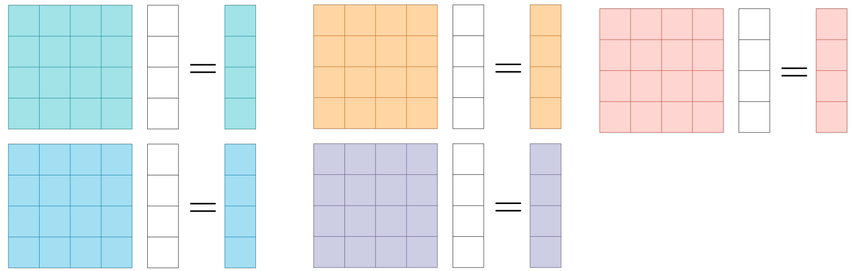
Sumcheck 1 (zero check):

$m_1 \circ m_2 \circ m_2 + m_3 + m_4 \circ m_2 - 3\, m_3 \circ m_5 = 0$



Sumcheck(s) 2 (“linchecks”, i.e. matrix-vector products):

$M_iz = m_i$



HyperNova [KS23]

I know w such that

$C_{CCS}(x, w) = 1$

I know w such that

$z = x \,||\, w$ and $M_i z = m_i$

HyperNova [KS23]

I know w such that

$C_{CCS}(x, w) = 1$

I know w such that

$z = x \,||\, w$ and $M_i z = m_i$

This is linearised CCS! (almost)

HyperNova [KS23]

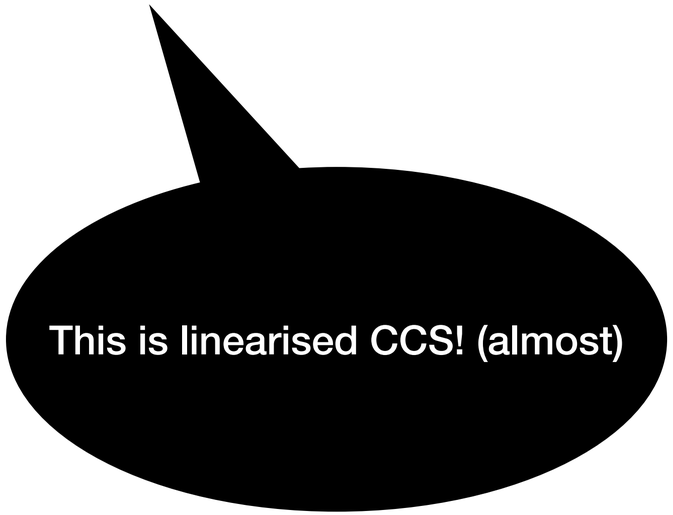
I know w such that

$C_{CCS}(x, w) = 1$

→ **Sumcheck 1 (sort of)** →

I know w such that

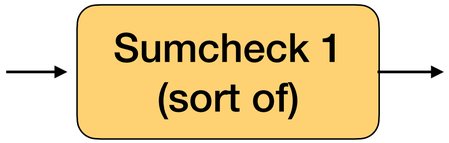
$z = x \| w$ and $M_i z = m_i$



HyperNova [KS23]

I know w such that

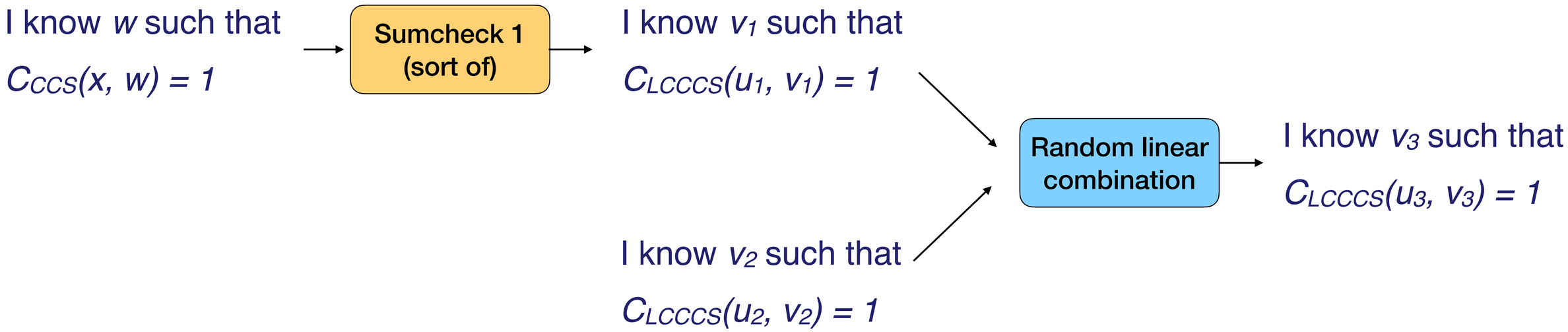
$C_{CCS}(x, w) = 1$



I know v_1 such that

$C_{LCCCS}(u_1, v_1) = 1$

HyperNova [KS23]



ProtoStar [BC23]

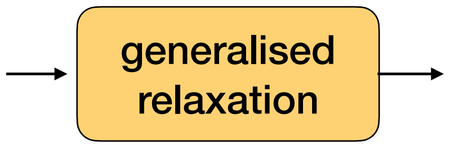
I know w such that

$f(x, w) = 0$

ProtoStar [BC23]

I know w such that

$f(x, w) = 0$



I know w, all the messages m_i

and challenges r_i such that:

$C_i = Comm(m_i)$

$relaxed\_verif(x, m_i, r_i, \mu) = e$

ProtoStar [BC23]

I know w such that

$f(x, w) = 0$

→ **generalised relaxation** →

I know w, all the messages m_i and challenges r_i such that:

$C_i = Comm(m_i)$

$relaxed\_verif(x, m_i, r_i, \mu) = e$

I know w, all the messages m_i and challenges r_i such that:

$C_i = Comm(m_i)$

$relaxed\_verif(x, m_i, r_i, \mu) = e$

→ **RLC with compressed error terms** →

I know w, all the messages m_i and challenges r_i such that:

$C_i = Comm(m_i)$

$relaxed\_verif(x, m_i, r_i, \mu) = e$

Comparing the latest schemes

Protocol	$\mathcal{R}$	$\mathcal{S}$	“Reduce” work	“Combine” work
Nova	R1CS	committed relaxed R1CS	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term
HyperNova	CCS	linearised committed CCS	P: commits to the witness, then P and V run the sumcheck protocol	P: random linear combination V: random linear combination
ProtoStar	any relation with algebraic verifier	commitments to all the messages and compressed verifier check	P: commit to each message V: produce random challenges	P: compute the compressed cross terms V: add commitments and compressed cross terms

Conclusion

- My mental model for folding schemes: **reduce-and-combine**
- Seems there is a trade-off between how much work we do in reduction and how easy it is to fold the resulting relation
- Language suggestion: let’s say folding for all protocol in this class, easier on the ear than split accumulation or multi-folding

References

[BC23] Bünz, Benedikt, and Binyi Chen. "Protostar: Generic efficient accumulation/folding for special sound protocols." *Cryptology ePrint Archive* (2023).

[BCLMS21] Bünz, B., Chiesa, A., Lin, W., Mishra, P., & Spooner, N. (2021). Proof-carrying data without succinct arguments. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41* (pp. 681-710). Springer International Publishing.

[BCMS20] Bünz, B., Chiesa, A., Mishra, P., & Spooner, N. (2020). Proof-carrying data from accumulation schemes. *Cryptology ePrint Archive*.

[BGH19] Bowe, S., Grigg, J., & Hopwood, D. (2019). Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*.

[KS23] Kothapalli, Abhiram, and Srinath Setty. "HyperNova: Recursive arguments for customizable constraint systems." *Cryptology ePrint Archive* (2023).

[KST22] Kothapalli, Abhiram, Srinath Setty, and Ioanna Tzialla. "Nova: Recursive zero-knowledge arguments from folding schemes." *Annual International Cryptology Conference*. Cham: Springer Nature Switzerland, 2022.

[Moh23] Mohnblatt, Nicolas. “Sangria: a folding scheme for PLONK”. *Github*. https://github.com/geometryresearch/technical_notes/blob/main/sangria_folding_plonk.pdf

[Set20] Setty, Srinath. "Spartan: Efficient and general-purpose zkSNARKs without trusted setup." *Annual International Cryptology Conference*. Cham: Springer International Publishing, 2020.

[STW23] Setty, Srinath, Justin Thaler, and Riad Wahby. "Customizable constraint systems for succinct arguments." *Cryptology ePrint Archive* (2023).

[ZcashDoc] Zcash. “halo2 Documentation”. https://zcash.github.io/halo2/index.html