Light Node Catchup Using Incrementally Verifiable Chain of Signatures

Rasmus Kirk Jakobsen

Computer Science Aarhus

2025-11-07 - 13:24:36 UTC

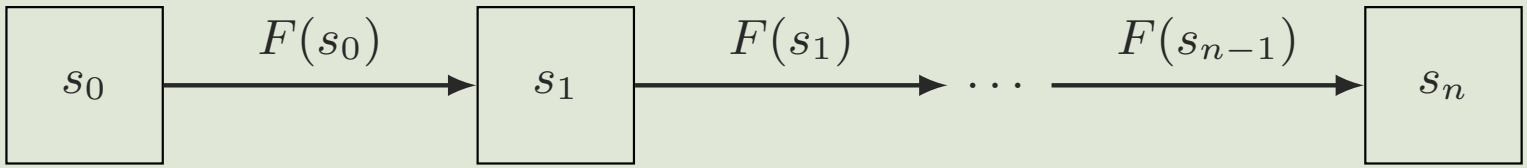
Introduction

Motivation

IVC is designed to solve the following problem:

If a computation runs for hundreds of years and ultimately outputs 42, how can we check its correctness without re-executing the entire process?

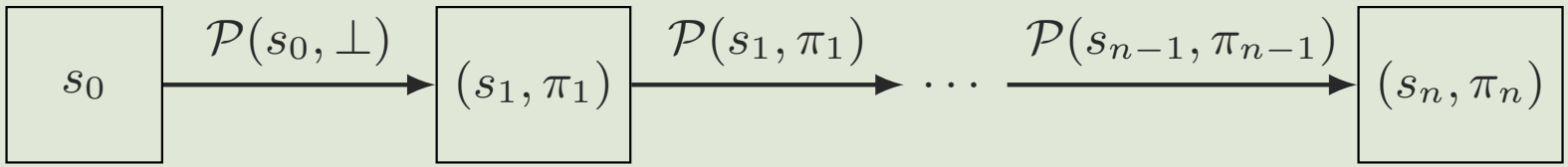
We define the transition function F run on an initial state s_0:



- *How can we verify $s_n = F^n(s_0)$ without re-executing the computation?*

IVC chain

We can use a SNARK to prove each computation step:



$\mathcal{P}(s_{i-1}, \pi_{i-1})$ represents:

- $s_i = F(s_{i-1})$
- $\pi_i = \mathsf{SNARK.PROVER}(R, x = \{s_0, s_i\}, w = \{s_{i-1}, \pi_{i-1}\})$
- $R :=$ I.K. $w = \{\pi_{i-1}, s_{i-1}\}$ s.t.
  - $s_i \stackrel{?}{=} F(s_{i-1}) \ \wedge \ (s_{i-1} \stackrel{?}{=} s_0 \vee \mathcal{V}(R, x = \{s_0, s_i\}, \pi_{i-1}))$

Proof

- R gives us a series of proofs of the claims:

$$\begin{aligned}
&\text{I.K. } w \text{ s.t. } && s_n && = F(s_{n-1}) && \wedge && (s_{n-1} = s_0 \vee \mathcal{V}(R, x, \pi_{n-1}) = \top),\\
&\text{I.K. } w \text{ s.t. } && s_{n-1} && = F(s_{n-2}) && \wedge && (s_{n-2} = s_0 \vee \mathcal{V}(R, x, \pi_{n-2}) = \top), \ \ldots\\
&\text{I.K. } w \text{ s.t. } && s_1 && = F(s_0) && \wedge && (s_0 = s_0 \vee \mathcal{V}(R, x, \pi_0) = \top)
\end{aligned}$$

- Which, if all verify means that:

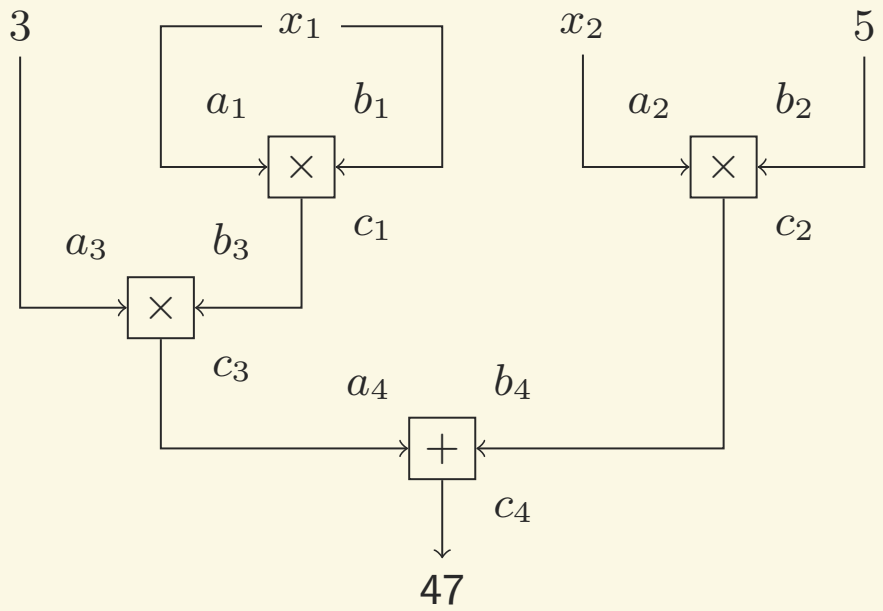
$$\begin{aligned}
&\textsf{SNARK.Verifier}(R, x, \pi_n) = \top \implies\\
&s_n = F(s_{n-1}) \wedge\\
&\textsf{SNARK.Verifier}(R, x, \pi_{n-1}) = \top \implies\\
&s_{n-1} = F(s_{n-2}) \wedge\\
&\textsf{SNARK.Verifier}(R, x, \pi_{n-2}) = \top \implies \ldots\\
&s_1 = F(s_0)
\end{aligned}$$

Plonk

Halo Overview

- $\mathbf{PC_{DL}}$: A Polynomial Commitment Scheme.
- $\mathbf{AS_{DL}}$: An Accumulation Scheme for Evaluation Proof instances.
- **Plonk**: A general-purpose, potentially zero-knowledge, SNARK.
- **Pasta**: A cycle of elliptic curves, Pallas and Vesta.

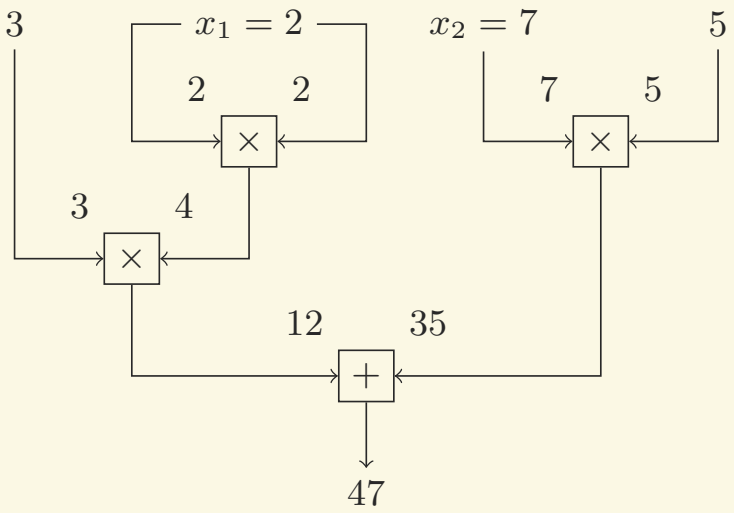
The Gate Constraints $3x_1^2 + 5x_2 = 47$



$$a(X) = \mathsf{ifft}(\boldsymbol{a})$$
$$b(X) = \mathsf{ifft}(\boldsymbol{b})$$
$$c(X) = \mathsf{ifft}(\boldsymbol{c})$$
$$a(\omega^1) = a_1,\ a(\omega^2) = a_2,\ \ldots$$
$$b(\omega^1) = b_1,\ b(\omega^2) = b_2,\ \ldots$$
$$c(\omega^1) = c_1,\ c(\omega^2) = c_2,\ \ldots$$

$$f_{GC}(X) = a(X)q_l(X) + b(X)q_r(X) + c(X)q_o(X) + a(X)b(X)q_m(X) + q_c(X),$$
$$f_{GC}(X) \stackrel{?}{=} 0$$

The Gate Constraints



$$
\begin{aligned}
f_{GC}(X) = a(X)q_l(X) \\
&+ b(X)q_r(X) \\
&+ c(X)q_o(X) \\
&+ a(X)b(X)q_m(X) \\
&+ q_c(X)
\end{aligned}
$$

$$\forall h \in H = \{\omega^1, \omega^2, \ldots, \omega^n\} :$$

$$f_{GC}(h) \stackrel{?}{=} 0$$

ω^i	$a(\omega^i)$	$b(\omega^i)$	$c(\omega^i)$	$q_l(\omega^i)$	$q_r(\omega^i)$	$q_o(\omega^i)$	$q_m(\omega^i)$	$q_c(\omega^i)$
ω^1	3	0	0	1	0	0	0	-3
ω^2	5	0	0	1	0	0	0	-5
ω^3	47	0	0	1	0	0	0	-47
ω^4	2	2	4	0	0	-1	1	0
ω^5	5	7	35	0	0	-1	1	0
ω^6	4	3	12	0	0	-1	1	0
ω^7	35	12	47	1	1	-1	0	0
ω^8	0	0	0	0	0	0	0	0

Vanishing Argument: $\forall h \in H : f(h) \stackrel{?}{=} 0$

Prover$(f \in \mathbb{F}_{\leq d}[X])$		**Verifier**
$C_f = \mathsf{PC.Commit}(f(X), d, \bot)$		
$z_H(X) = \prod_{h \in H} (X - h)$		
$t(X) = \frac{f(X)}{z_H(X)}$		
$C_t = \mathsf{PC.Commit}(t(X), d, \bot)$	$\xrightarrow{C_f, C_t}$	$\xi \in_R \mathbb{F}$
	$\xleftarrow{\xi}$	
$v_f = f(\xi)$		
$\pi_f = \mathsf{PC.Open}(v_f, C_f, d, \xi, \bot)$		
$v_t = t(\xi)$		
$\pi_t = \mathsf{PC.Open}(v_t, C_t, d, \xi, \bot)$	$\xrightarrow{v_f, \pi_f, v_t, \pi_t}$	$v_f \stackrel{?}{=} v_t \cdot z_H(\xi)$
		$\mathsf{PC.Check}(C_f, d, \xi, v_f, \pi_f)$
		$\mathsf{PC.Check}(C_t, d, \xi, v_t, \pi_t)$

Vanishing Argument: $\forall h \in H : f(h) \stackrel{?}{=} 0$

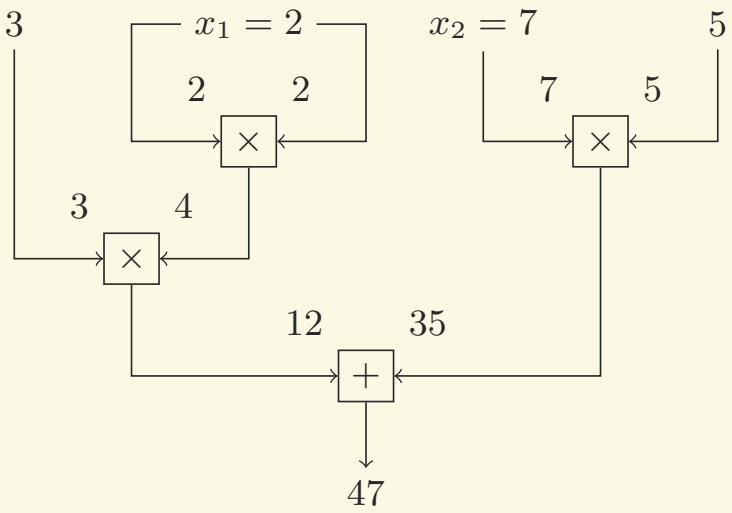
Correctness

$$\begin{aligned} p(\xi) &= f(\xi) - t(\xi) z_H(\xi) \\ &= f(\xi) - \left(\frac{f(\xi)}{z_H(\xi)} \right) z_H(\xi) \\ &= 0 \end{aligned}$$

Soundness

- $z_H \mid f$ only if all of $h \in H : f(h) = 0$ (Factor Theorem)
- Schwartz-Zippel Lemma: $\xi \in_R \mathbb{F} : Pr[p(\xi) = 0 \mid p \neq 0] \leq \frac{\deg(p)}{|\mathbb{F}|}$
- $|\mathbb{F}| \gg \deg(p) \implies Pr[p(\xi) = 0 \mid p \neq 0] \leq \epsilon$
- $\deg(p) \leq d \leq n$

Copy Constraints



$$a(\omega^4) = b(\omega^4)$$
$$b(\omega^5) = a(\omega^2)$$
$$a(\omega^6) = c(\omega^4) \quad b(\omega^6) = a(\omega^1)$$
$$a(\omega^7) = c(\omega^5) \quad b(\omega^7) = c(\omega^6)$$
$$c(\omega^7) = a(\omega^3)$$

ω^i	$a(\omega^i)$	$b(\omega^i)$	$c(\omega^i)$	$q_l(\omega^i)$	$q_r(\omega^i)$	$q_o(\omega^i)$	$q_m(\omega^i)$	$q_c(\omega^i)$
ω^1	3	0	0	1	0	0	0	-3
ω^2	5	0	0	1	0	0	0	-5
ω^3	47	0	0	1	0	0	0	-47
ω^4	2	2	4	0	0	-1	1	0
ω^5	5	7	35	0	0	-1	1	0
ω^6	4	3	12	0	0	-1	1	0
ω^7	35	12	47	1	1	-1	0	0
ω^8	0	0	0	0	0	0	0	0

Copy Constraints

$$
\begin{aligned}
a(\omega^4) &= b(\omega^4) \\
b(\omega^5) &= a(\omega^2) \\
a(\omega^6) &= c(\omega^4) \quad b(\omega^6) = a(\omega^1) \\
a(\omega^7) &= c(\omega^5) \quad b(\omega^7) = c(\omega^6) \\
c(\omega^7) &= a(\omega^3)
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{f} &= \boldsymbol{a} \mathbin{+\!\!+} \boldsymbol{b} \mathbin{+\!\!+} \boldsymbol{c} \\
&= [a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8], \\
\sigma(\boldsymbol{f}) &= [b_6, b_5, a_3, b_4, a_5, c_4, c_5, a_8, b_1, b_2, b_3, a_4, a_2, a_1, c_6, b_8, c_1, c_2, c_3, a_6, a_7, b_7, a_3, c_8]
\end{aligned}
$$

$$
\forall i \in [n] : f(\omega^i) \stackrel{?}{=} f(\omega^{\sigma(i)})
$$

Copy Constraints - The Grand Product Argument

$$\forall \omega \in H : f(\omega) = g(\omega),$$
$$f'(X) := f(X) + \gamma,$$
$$g'(X) := g(X) + \gamma,$$
$$\prod_{i \in [n]} f'(\omega^i) \stackrel{?}{=} \prod_{i \in [n]} g'(\omega^i)$$

$$p(X) = \prod_{i \in [n]} f(\omega^i) + X$$
$$q(X) = \prod_{i \in [n]} g(\omega^i) + X$$

Soundness $p(X) = q(X) \implies \{a_1, \ldots, a_n\} = \{b_1, \ldots, b_n\}$

$$\begin{aligned} \text{roots}(p(X)) &= \{-f(\omega^1), \ldots, -f(\omega^n)\} \\ &= \{-a_1, \ldots, -a_n\} \end{aligned} \qquad \begin{aligned} \text{roots}(q(X)) &= \{-g(\omega^1), \ldots, -g(\omega^n)\} \\ &= \{-b_1, \ldots, -b_n\} \end{aligned}$$

- Since the two polynomials are equal, they must have the same roots

$$\begin{gathered} \text{roots}(p(X)) = \text{roots}(q(X)) \implies \\ \{-a_1, \ldots, -a_n\} = \{-b_1, \ldots, -b_n\} \implies \\ \{a_1, \ldots, a_n\} = \{b_1, \ldots, b_n\} \end{gathered}$$

Copy Constraints - The Grand Product Argument

$$z(\omega^1) = 1$$
$$z(\omega^i) = \prod_{1 \leq j < i} \frac{f'(\omega^j)}{g'(\omega^j)}$$

$$z(\omega^i) = \prod_{1 \leq j < i} \frac{f'(\omega^j)}{g'(\omega^j)}$$
$$z(\omega^i) = z(\omega^{i-1}) \frac{f'(\omega^{i-1})}{g'(\omega^{i-1})}$$
$$z(\omega^{i+1}) = z(\omega^i) \frac{f'(\omega^i)}{g'(\omega^i)}$$
$$z(\omega^{i+1}) g'(\omega^i) = z(\omega^i) f'(\omega^i)$$
$$z(\omega^i) f'(\omega^i) = z(\omega \cdot \omega^i) g'(\omega^i)$$

Verifier Checks

$$f_{CC_1}(X) = l_1(X)(z(X) - 1)$$
$$f_{CC_2}(X) = z(X) f'(X) - z(\omega X) g'(X)$$

Copy Constraints - The Grand Product Argument

Checking implies $\prod_{\omega \in H} f'(\omega) = \prod_{\omega \in H} g'(\omega)$

- For the $i = n$ case:

$$z(\omega^n) f'(\omega^n) = z(\omega^{n+1}) g'(\omega)$$

$$\prod_{1 \leq j < i} \frac{f'(\omega^j)}{g'(\omega^j)} f'(\omega^n) = g'(\omega)$$

$$\prod_{1 \leq j < i} \frac{f'(\omega^j) f'(\omega^n)}{g'(\omega^j) g'(\omega^n)} = 1$$

$$\frac{\prod_{i \in [n]} f'(\omega^i)}{\prod_{i \in [n]} g'(\omega^i)} = 1$$

$$\prod_{i \in [n]} f'(\omega^i) = \prod_{i \in [n]} g'(\omega^i)$$

Copy Constraints - The Grand Product Argument Protocol

Prover$(f, g \in \mathbb{F}_{\leq d}[X])$		**Verifier**
$C_f = \text{PC.COMMIT}(f(X), d, \perp)$		
$C_g = \text{PC.COMMIT}(g(X), d, \perp)$	$\xrightarrow{C_f, C_g}$	$\gamma \in_R \mathbb{F}$
$z(\omega^1) = 1$		
$z(\omega^i) = \prod_{1 \leq j < i} \frac{f(\omega^j) + \gamma}{g(\omega^j) + \gamma}$	$\longleftrightarrow$	$\forall h \in H:$
	$\longleftrightarrow$	$f_{CC_1}(h) \stackrel{?}{=} l_1(h)(z(h) - 1)$
	$\longleftrightarrow$	$f_{CC_2}(h) \stackrel{?}{=} z(h)f'(h) - z(\omega h)g'(h)$

Copy Constraints - Applying the Grand Product Argument

- $\mathcal{P}$ wants to prove that for $i \in [n] : f_i = f_{\sigma(i)}$:

$$\{(f_i, i) \mid i \in [1, n]\} = \{(f_i, \sigma(i)) \mid i \in [1, n]\} \implies$$
$$f_i = f_{\sigma_i} \implies$$
$$\boldsymbol{f} = \sigma(\boldsymbol{f})$$

- Meaning for polynomials:

$$\{(f(\omega^i), \mathrm{id}(\omega^i)) \mid i \in [1, n]\} = \{(f(\omega^i), \sigma(\omega^i)) \mid i \in [1, n]\} \implies$$
$$f(\omega^i) = f(\omega^{\sigma(i)})$$

- Which is exactly suited for the **Grand Product Argument**, with:

$$f'(X) = f(X) + \beta \mathrm{id}(X), \quad g'(X) = f(X) + \beta\sigma(X)$$

Public Inputs

- Public inputs, $\boldsymbol{x} : |\boldsymbol{x}| = \ell_2$
- Leading to a public input polynomial, $x(X) = \text{ifft}(-\boldsymbol{x})$

$$f_{GC}(X) = a(X)q_l(X) + b(X)q_r(X) + c(X)q_o(X) + a(X)b(X)q_m(X) + q_c(X) + x(X)$$

Custom Gates

1. Add a new selector polynomial
2. Create a constraint table
3. Convert constraint table to f_{GC} terms

Custom Gates - Double-and-Add

Double-and-Add Scalar Multiplication $A = x \cdot P$

Let $\boldsymbol{b}$ be the bits of x, from LSB to MSB.
Let $A = \mathcal{O}$.
Let $acc = 0$.
for $i \in (255, 0]$ **do**
    $acc \mathrel{+}= b_i \cdot 2^i$
    $Q := 2A$
    $R := P + Q$
    $S :=$ **if** $b_i = 1$ **then** R **else** Q
    $A := S$
end for
assert $acc \stackrel{?}{=} x$
return A

Custom Gates - Double-and-Add - Double

- $A = \mathcal{O} \implies Q = \mathcal{O}$:
- $A \neq \mathcal{O} \implies Q = 2A$:

$$\lambda = \frac{3x_a^2}{2y_a}$$
$$x_q = \lambda_q^2 - 2 \cdot x_a$$
$$y_q = \lambda_q \cdot (x_a - x_q) - y_a$$

- Witness:

$$\gamma_q = \mathsf{inv0}(x_a)$$
$$\lambda_q = \begin{cases} \frac{3x_a^2}{2y_a}, & \text{if } A \neq \mathcal{O}, \\ 0, & \text{otherwise.} \end{cases}$$

Custom Gates - Double-and-Add - Double Constraints

Degree	Constraint	Meaning
4	$(1 - x_a \cdot \gamma_q) \cdot x_q$	$x_a = 0 \implies x_q = 0$
4	$(1 - x_a \cdot \gamma_q) \cdot y_q$	$x_a = 0 \implies y_q = 0$
3	$(2 \cdot y_a \cdot \lambda_q - 3 \cdot x_a^2)$	$\lambda_q = \frac{3x_a^2}{2y_a}$
3	$(\lambda_q^2 - 2 \cdot x_a - x_q)$	$x_q = \lambda_q^2 - 2 \cdot x_a$
3	$(\lambda_q \cdot (x_a - x_q) - y_a - y_q)$	$y_q = \lambda_q \cdot (x_a - x_q) - y_a$

Custom Gates - Double-and-Add - Ternary Constraints

Degree	Constraint	Meaning
3	$b_i \cdot (b_i - 1)$	$b_i \in \mathbb{B}$
3	$x_s - (b_i \cdot x_r + (1 - b_i) \cdot x_q)$	$x_s =$ **if** $b_i = 1$ **then** x_r **else** x_q
3	$y_s - (b_i \cdot y_r + (1 - b_i) \cdot y_q)$	$y_s =$ **if** $b_i = 1$ **then** y_r **else** y_q
3	$acc_{i+1} - (acc_i + b_i \cdot 2^i)$	$acc_{i+1} = (acc_i + b_i \cdot 2^i)$

Custom Gates - Double-and-Add

Degree	Constraint	Meaning
4	$(1 - x_a \cdot \gamma_q) \cdot x_q$	$x_a = 0 \implies x_q = 0$
4	$(1 - x_a \cdot \gamma_q) \cdot y_q$	$x_a = 0 \implies y_q = 0$
3	$2 \cdot y_a \cdot \lambda_q - 3 \cdot x_a^2$	$\lambda_q = \frac{3x_a^2}{2y_a}$
3	$\lambda_q^2 - 2 \cdot x_a - x_q$	$x_q = \lambda_q^2 - 2 \cdot x_a$
3	$(\lambda_q \cdot (x_a - x_q) - y_a - y_q)$	$y_q = \lambda_q \cdot (x_a - x_q) - y_a$
3	$b_i \cdot (b_i - 1)$	$b_i \in \mathbb{B}$
3	$x_s - (b_i \cdot x_r + (1 - b_i) \cdot x_q)$	$x_s =$ **if** $b_i = 1$ **then** x_r **else** x_q
3	$y_s - (b_i \cdot y_r + (1 - b_i) \cdot y_q)$	$y_s =$ **if** $b_i = 1$ **then** y_r **else** y_q
3	$acc_{i+1} - (acc_i + b_i \cdot 2^i)$	$acc_{i+1} = (acc_i + b_i \cdot 2^i)$

$$
\begin{aligned}
f_{GC}(X) = \cdots + q_{(\cdot)} \cdot (& \\
& \zeta^0 \cdot ((1 - x_a \cdot \gamma_q) \cdot x_q) + \\
& \zeta^1 \cdot ((1 - x_a \cdot \gamma_q) \cdot y_q) + \\
& \zeta^2 \cdot (2 \cdot y_a \cdot \lambda_q - 3 \cdot x_a^2) + \\
& \cdots \\
) &
\end{aligned}
$$

Custom Gates - Double-and-Add

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
x_a	y_a	acc	x_p	y_p	x_q	y_q	x_r
w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
y_r	b_i	γ_q	λ_q	α_r	β_r	δ_r	λ_r

$$
\begin{aligned}
f_{GC}(X) = \cdots + q_{(\cdot)} \cdot (& \\
& \zeta^0 \cdot ((1 - x_a \cdot \gamma_q) \cdot x_q) + \\
& \zeta^1 \cdot ((1 - x_a \cdot \gamma_q) \cdot y_q) + \\
& \zeta^2 \cdot (2 \cdot y_a \cdot \lambda_q - 3 \cdot x_a^2) + \\
& \cdots \\
) &
\end{aligned}
$$

$$
\begin{aligned}
f_{GC}(X) = \cdots + q_{(\cdot)}(X) \cdot (& \\
& \zeta^0 \cdot ((1 - w_1(X) \cdot w_{11}(X)) \cdot w_6(X)) + \\
& \zeta^1 \cdot ((1 - w_1(X) \cdot w_{11}(X)) \cdot w_6(X)) + \\
& \zeta^2 \cdot (2 \cdot w_2(X) \cdot w_{12}(X) - 3 \cdot w_1(X)^2) + \\
& \cdots \\
) &
\end{aligned}
$$

IVC

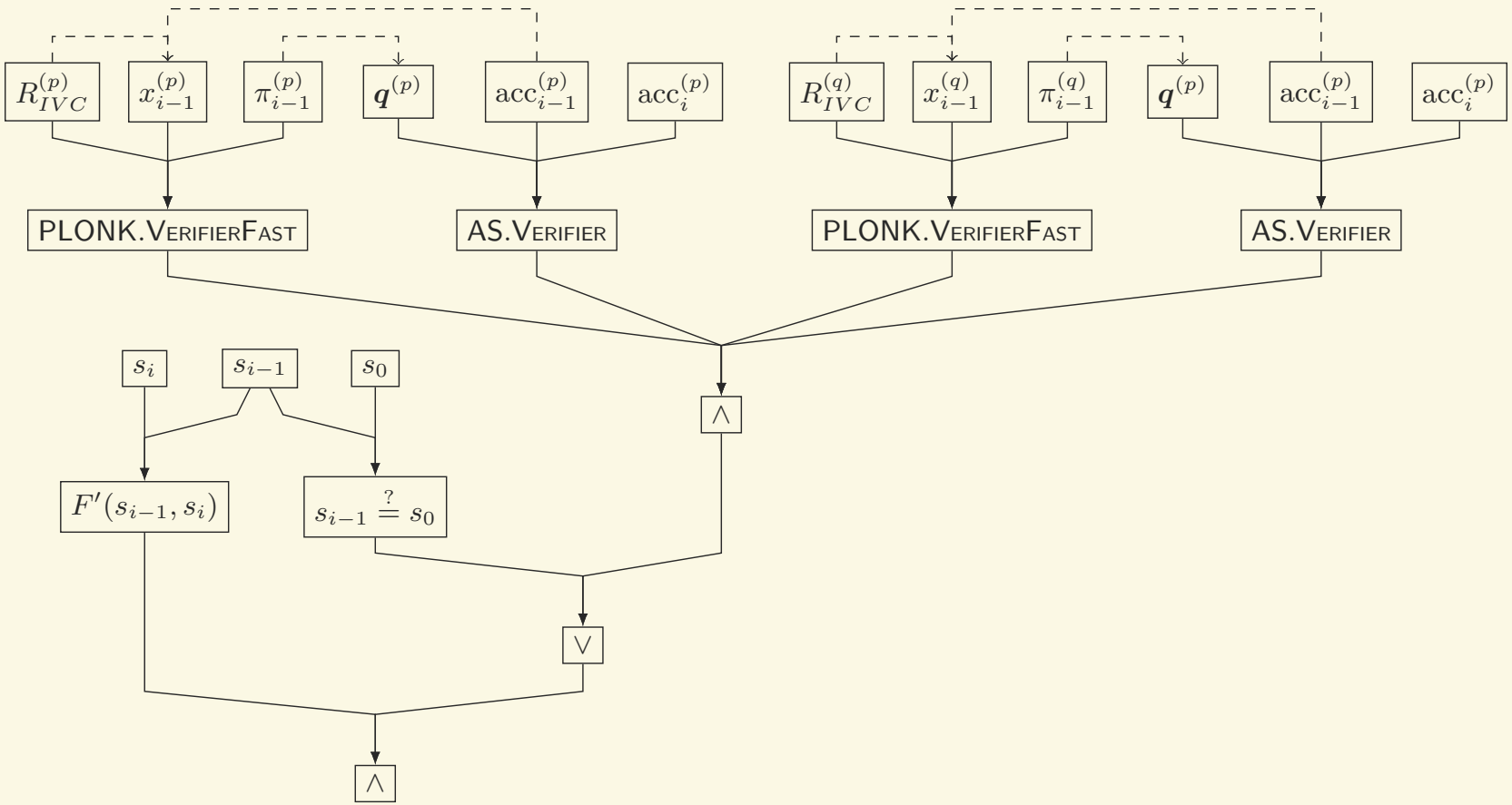
Chain of Signatures

$$B_i = \{\sigma_i^{(pk)}, j_i = i, pk_i, ptr_i \in \mathbb{B}^{256}, \sigma_i^{(ptr)}\}$$

- $\sigma_i^{(pk)}$: A signature on the public key of the current committee (pk_i), signed by the previous committee identified by the public key pk_{i-1}.
- j_i: A sequential block-id. This must be present for the soundness of the IVC circuit.
- pk_i: The public key of the current committee.
- ptr_i: A hash of the most recent block on the main blockchain.
- $\sigma_i^{(ptr)}$: A signature on ptr_i, signed by the current committee.

$$\text{Verify}_{pk_{i-1}}(\sigma_i^{(pk)}, pk_i) \wedge \text{Verify}_{pk_i}(\sigma_i^{(ptr)}, ptr_i) \wedge j_i \stackrel{?}{=} j_{i-1} + 1$$

IVC


$R_{IVC}^{(p)}$
$x_{i-1}^{(p)}$
$\pi_{i-1}^{(p)}$
$\boldsymbol{q}^{(p)}$
$\text{acc}_{i-1}^{(p)}$
$\text{acc}_{i}^{(p)}$
$R_{IVC}^{(q)}$
$x_{i-1}^{(q)}$
$\pi_{i-1}^{(q)}$
$\boldsymbol{q}^{(p)}$
$\text{acc}_{i-1}^{(p)}$
$\text{acc}_{i}^{(p)}$
PLONK.VerifierFast
AS.Verifier
PLONK.VerifierFast
AS.Verifier
s_i
s_{i-1}
s_0
$\wedge$
$F'(s_{i-1}, s_i)$
$s_{i-1} \stackrel{?}{=} s_0$
$\vee$
$\wedge$

Benchmarks and Conclusion

Benchmarks

- **IVC-Prover:** Parallel: ~300 s. Single Threaded: ~900 s.
- **IVC-Verifier:** Parallel: ~3 s. Single Threaded: ~9 s.
- **Naive Signature Verification:** Parallel: ~1300 signatures per second. Single Threaded: ~310 signatures per second.

When is it better?

- $1300 \cdot 3 \ / \ 2 = 1950$ days before the IVC verifier would be faster
- The ~10 kB Proof would be smaller after only 87 days

Conclusion

The goal:

- To fully implement and analyze a complex IVC-scheme
- Showed IVC may be useful in the context of blockchain catch-up
- Optimizations, lookups, quantum security...