Curve Trees:
Practical and Transparent Zero-Knowledge Accumulators

Matteo Campanelli, Mathias Hall-Andersen, and Simon Holmgaard Kamp

USENIX Security 2023

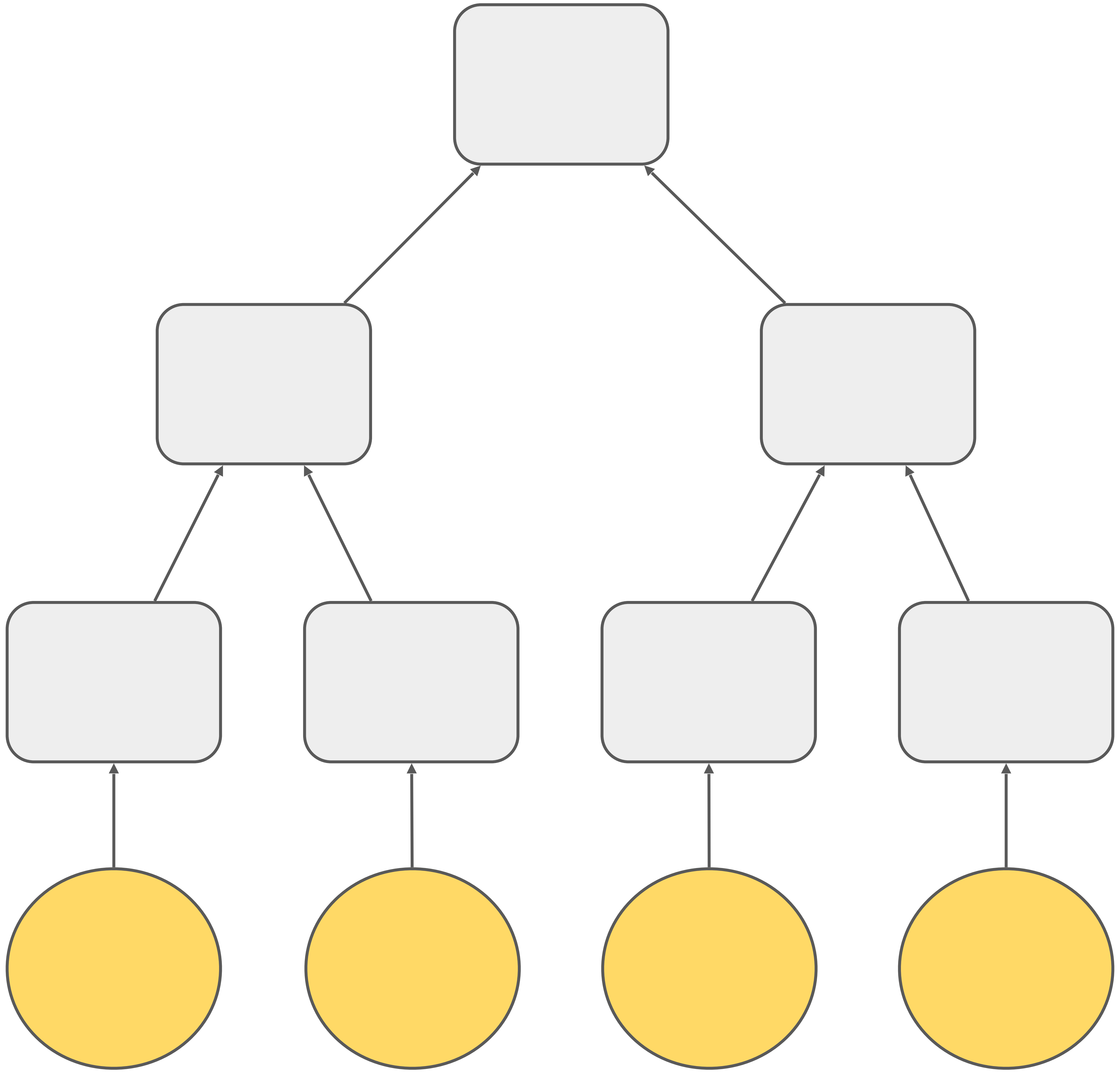
Zero-Knowledge Accumulators

- Short digest of a public set S.
  - Update S in public.
  - Prove statement about $x \in S$ in zk.

Anonymous Payments

- "I own a coin in S"
  - Coin: $value \cdot G_1 + nullifier \cdot G_2 + r \cdot H$
  - Reveal $nullifier$ when spending the coin.
- Spend a set of coins and add coins of equal value to S.
- Use rerandomized public keys as nullifiers and sign the transactions.

Merkle Trees



Which hash function should you use?

- Merkle Tree with SHA256: ≈ 800.000 R1CS constraints.
- Merkle Tree with Pedersen: ≈ 45.000 R1CS constraints.
- Curve Tree: 4668 R1CS constraints.

Merkle Trees with Pedersen Hashing

- Hashing a field element is “native" to the proof.
- The digest is a group element.
  - Not native to the proof system.
  - Proceed recursively using bit decomposition.

Commit and Prove!

- Replace Pedersen Hashing with Pedersen Commitments
  - $v_1 \cdot G_1 + \ldots + v_n \cdot G_n$ becomes $v_1 \cdot G_1 + \ldots + v_n \cdot G_n + r \cdot H$
- P gives the path of commitments to V.
  - Revealing the path to the leaf!?
  - Figure out zero knowledge later.
- But the digest is still not a native input to the hash function?

Cycles of Elliptic Curves

- What if the digest is native to another hash function?
- Pick elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0})$ and $\mathbb{E}_1(\mathbb{F}_{p_1})$, where $|\mathbb{E}_0| = p_1$ and $|\mathbb{E}_1| = p_0$.
  - The scalar field of one is the base field of the other.
- Points on $\mathbb{E}_i$ are native to the function hashing into $\mathbb{E}_{1-i}$.
- Commit to a point by committing to both coordinates.
  - A Curve Tree with arity ℓ needs 2ℓ generators.
  - Can we do better?

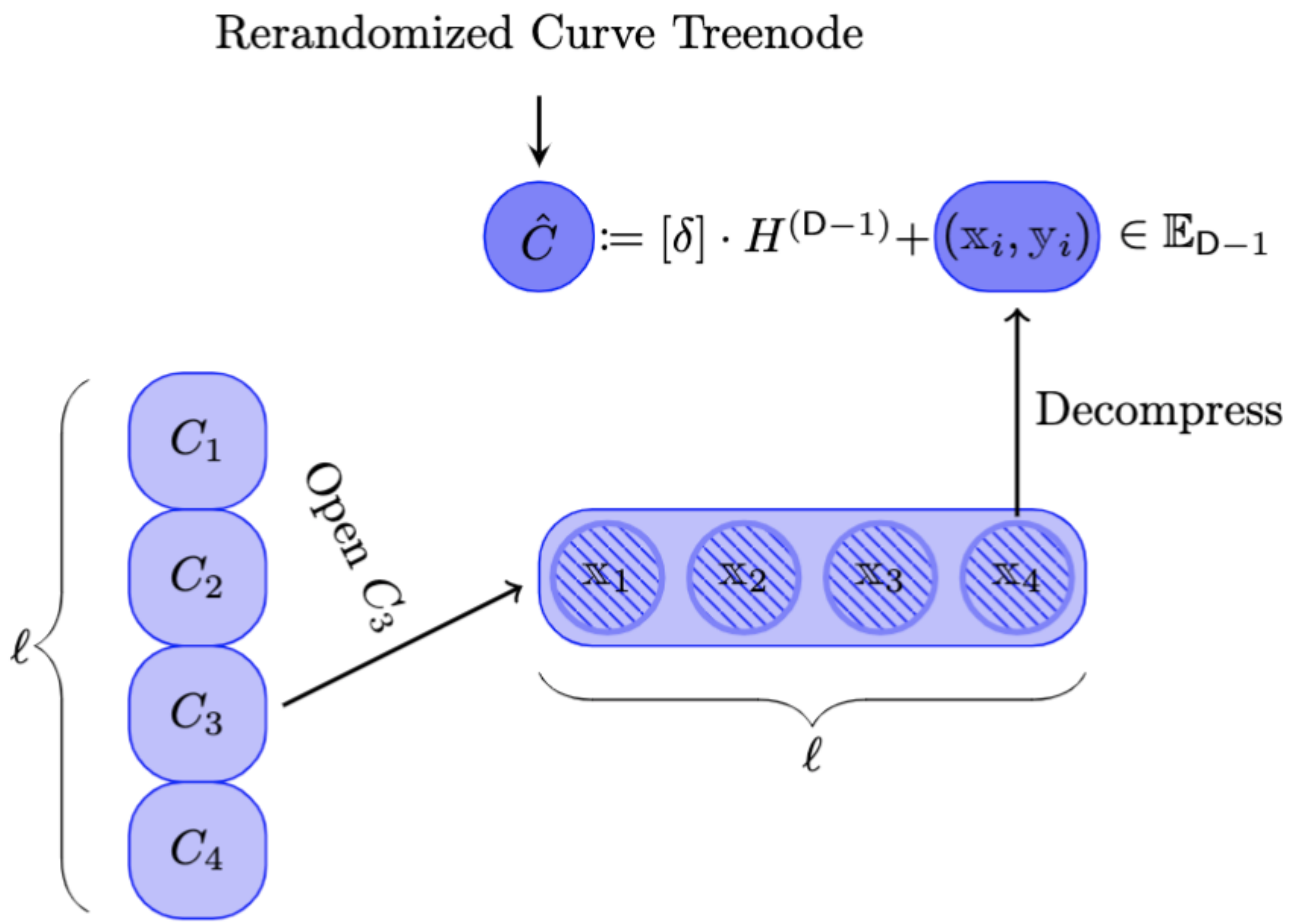
Removing the y-coordinates

- Standard trick: Compress a point to just the x-coordinate and a sign.
- Permissible points: Only points with positive sign are allowed in the tree.
- The sign function is often $y > p/2$ or $lsb(y)$
  - Computing the sign requires $O(\lambda)$ field operations.
- Instead: pick a universal hash function from $\mathscr{U}_{\alpha,\beta} : \mathbb{F} \to \{0,1\}$
  - $\mathscr{U}_{\alpha,\beta} \mapsto S(\alpha \cdot v + \beta)$ where $S(v) = 1 \iff v$ is a quadratic residue in $\mathbb{F}$.
  - Prove that $\mathscr{U}_{\alpha,\beta}(v) = 1$ with witness w where $w^2 = \alpha \cdot v + \beta$

Adding zero knowledge

- The path of commitments leaks the leaf.
- Rerandomize all the commitments!
- From the root onwards: "Select and Rerandomize"
  - Show that the next commitment on the path is a rerandomization of a child of the current commitment.

Select and Rerandomize



$$\mathcal{R}^{(\text{single-level}^\star,(\_))} := \left\{ \begin{pmatrix} i, r, \delta, \\ \vec{\mathbb{x}}, \mathbb{y} \end{pmatrix} : \begin{array}{c} C = \langle [\vec{\mathbb{x}}], \vec{G}^{\mathsf{x}}_{(\_)} \rangle \\ + [r] \cdot H_{(\_)} \\ \wedge (\mathbb{x}_i, \mathbb{y}) \in \mathcal{P}_{\text{other}(\_)} \\ \wedge \hat{C} = (\mathbb{x}_i, \mathbb{y}) + [\delta] \cdot H_{\text{other}(\_)} \end{array} \right\}$$

Circuit costs

$$\mathcal{R}^{(\textsf{single-level}^\star,(\_))} := \left\{ \begin{pmatrix} i, r, \delta, \\ \vec{\mathbb{x}}, \mathbb{y} \end{pmatrix} : \begin{array}{c} C = \langle [\vec{\mathbb{x}}], \vec{G}^{\mathsf{x}}_{(\_)} \rangle \\ + [r] \cdot H_{(\_)} \\ \wedge\ (\mathbb{x}_i, \mathbb{y}) \in \mathcal{P}_{\mathsf{other}(\_)} \\ \wedge\ \hat{C} = (\mathbb{x}_i, \mathbb{y}) + [\delta] \cdot H_{\mathsf{other}(\_)} \end{array} \right\}$$

- Select x-coordinate: $\ell - 1$ constraints.
- Decompress permissible point: 1 constraint.
- Point addition with native coordinates: ≈ 10 constraints.
- Fixed base scalar multiplication: ≈ 900 constraints.
  - Split algebraically incompatible elements into 3-bit windows.
  - Compute scalar multiplication with lookup tables an incomplete addition.

Select and Rerandomize

Curves	(D, ℓ)	$\lvert S\rvert$	# Constraints	Proof (kb)	Prove (s)	Verify (ms)	Verify batch (ms)
Pasta	$(2, 1024)$	2^{20}	3870	2.6	0.88	23.17	1.44
	$(4, 256)$	2^{32}	4668	2.9	1.71	39.63	2.35
	$(4, 1024)$	2^{40}	7740	2.9	1.74	40.41	2.73
Secp/Secq	$(2, 1024)$	2^{20}	3870	2.6	0.97	26.81	1.61
	$(4, 256)$	2^{32}	4668	2.9	1.89	47.39	2.64
	$(4, 1024)$	2^{40}	7740	2.9	1.92	48.40	3.02

Accumulator

Scheme	# Constraints	Prove (s)	Verify (ms)	Verify batch (ms)
Curve Trees (Pasta)	3565	1.5	31	1.8
Curve Trees (Secp/Secq)	3565	1.7	37	2
Poseidon 4:1	4515	8.8	651	-
Poseidon 8:1	4180	8.5	825	-

2-2 Pour

	Anonymity set size	Transparent setup	Tx size (kb)	Proving time (S)	Verification time (ms)	Amort. batch verification time (ms)
Zcash Sapling	2^{32}	✗	2.8	2.38	7	-
Zcash Orchard	2^{32}	✓	7.6	1.77	15	-
$\mathbb{V}$eksel	Any	✗⋆	5.3	0.44	61.88	-
Lelantus	2^{10}	✓	2.7	0.27†	-	6.8†
	2^{14}	✓	3.9	2.35†	-	10.2†
	2^{16}	✓	5.6	4.8†	-	52†
Omniring	2^{10}	✓	1	≈ 1.5‡	≈ 130‡	-
$\mathbb{V}$Cash (Pasta)	2^{20}	✓	3.4	1.76	41.40	2.87
	2^{32}	✓	4	3.43	78.40	4.98
	2^{40}	✓	4	3.48	80.52	5.77
$\mathbb{V}$Cash (Secp/Secq)	2^{20}	✓	3.4	1.95	48.27	3.15
	2^{32}	✓	4	3.80	90.40	5.60
	2^{40}	✓	4	3.86	91.97	6.32

Thank you!

Questions?