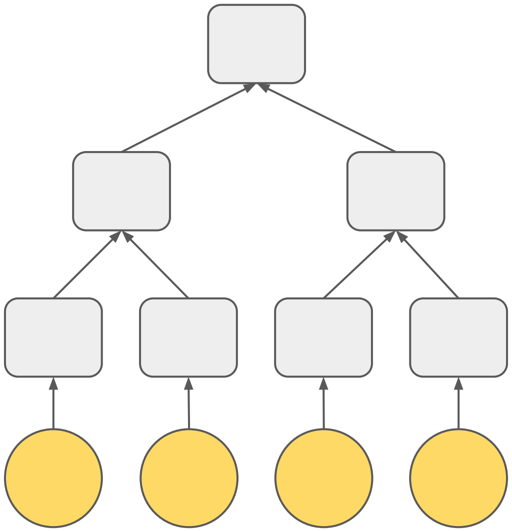
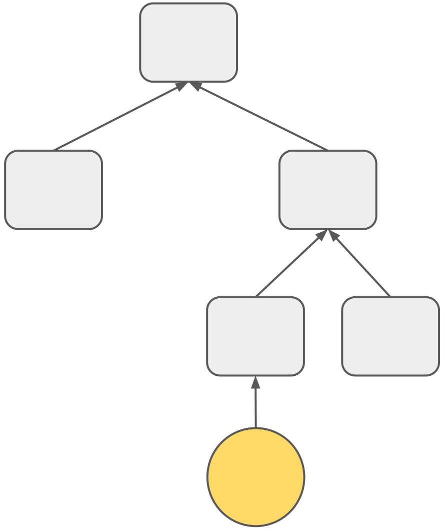
Curve Trees:
Practical and Transparent Zero-Knowledge Accumulators

Matteo Campanelli[1], Mathias Hall-Andersen[2],
and *Simon Holmgaard Kamp*[2]
[1] Protocol Labs, [2] Aarhus University



- Prove membership
- Prove ownership
- Reveal nullifier



Trusted Setup

https://z.cash/technology/paramgen/

A simple transparent ZCash using Bulletproofs?

- Use a “native” hash function: Pedersen hashing.
- Constrain hashes recursively using bit decomposition.
- A single membership proof for set size 2^32: about 45,000 constraints
- A single membership proof using a Curve Tree: <5000 constraints

Commit and Prove

- Given a commitment, prove properties of the committed values.
- Replace Pedersen hashing with Pedersen commitments.
- P provides the path of commitments to V.
  - Show parent child relations.
  - Reveals the path to the leaf!
- The digest is not a native input of the hash function :(

Towers of elliptic curves

- What if the digest is native to another hash function?

Towers of elliptic curves

- What if the digest is native to another hash function?
- Pick a sequence of elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0}), \ldots, \mathbb{E}_{\mathsf{D}}(\mathbb{F}_{p_{\mathsf{D}}})$ where each $\mathbb{E}_i(\mathbb{F}_{p_i})$ has p_{i+1} points.

Towers of elliptic curves

- What if the digest is native to another hash function?
- Pick a sequence of elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0}), \ldots, \mathbb{E}_\mathsf{D}(\mathbb{F}_{p_\mathsf{D}})$ where each $\mathbb{E}_i(\mathbb{F}_{p_i})$ has p_{i+1} points.
- Now points on the i+1’th curve are native inputs to the Pedersen hash function over the i’th curve.

Towers of elliptic curves

- What if the digest is native to another hash function?
- Pick a sequence of elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0}), \ldots, \mathbb{E}_{\mathsf{D}}(\mathbb{F}_{p_{\mathsf{D}}})$ where each $\mathbb{E}_i(\mathbb{F}_{p_i})$ has p_{i+1} points.
- Now points on the i+1'th curve are native inputs to the Pedersen hash function over the i'th curve.
- Commit to a point by committing to both coordinates.

Towers of elliptic curves

- What if the digest is native to another hash function?
- Pick a sequence of elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0}), \ldots, \mathbb{E}_{\mathsf{D}}(\mathbb{F}_{p_{\mathsf{D}}})$ where each $\mathbb{E}_i(\mathbb{F}_{p_i})$ has p_{i+1} points.
- Now points on the i+1'th curve are native inputs to the Pedersen hash function over the i'th curve.
- Commit to a point by committing to both coordinates.
  - A Curve Tree with branching factor l has 2l generators.

Towers of elliptic curves

- What if the digest is native to another hash function?
- Pick a sequence of elliptic curves $\mathbb{E}_0(\mathbb{F}_{p_0}), \ldots, \mathbb{E}_{\mathsf{D}}(\mathbb{F}_{p_{\mathsf{D}}})$ where each $\mathbb{E}_i(\mathbb{F}_{p_i})$ has p_{i+1} points.
- Now points on the i+1'th curve are native inputs to the Pedersen hash function over the i'th curve.
- Commit to a point by committing to both coordinates.
  - A Curve Tree with branching factor l has 2l generators.
  - Can we do better?

Committing only to x

- Standard trick: compress a point into the x-coordinate and a sign.
- Permissible points: points with a positive sign.
- Only permissible points are added to the tree.
- A sign is often $y>p/2$ or $lsb(y)$.
    - Proving this inside the circuit adds roughly λ constraints.

Committing only to x

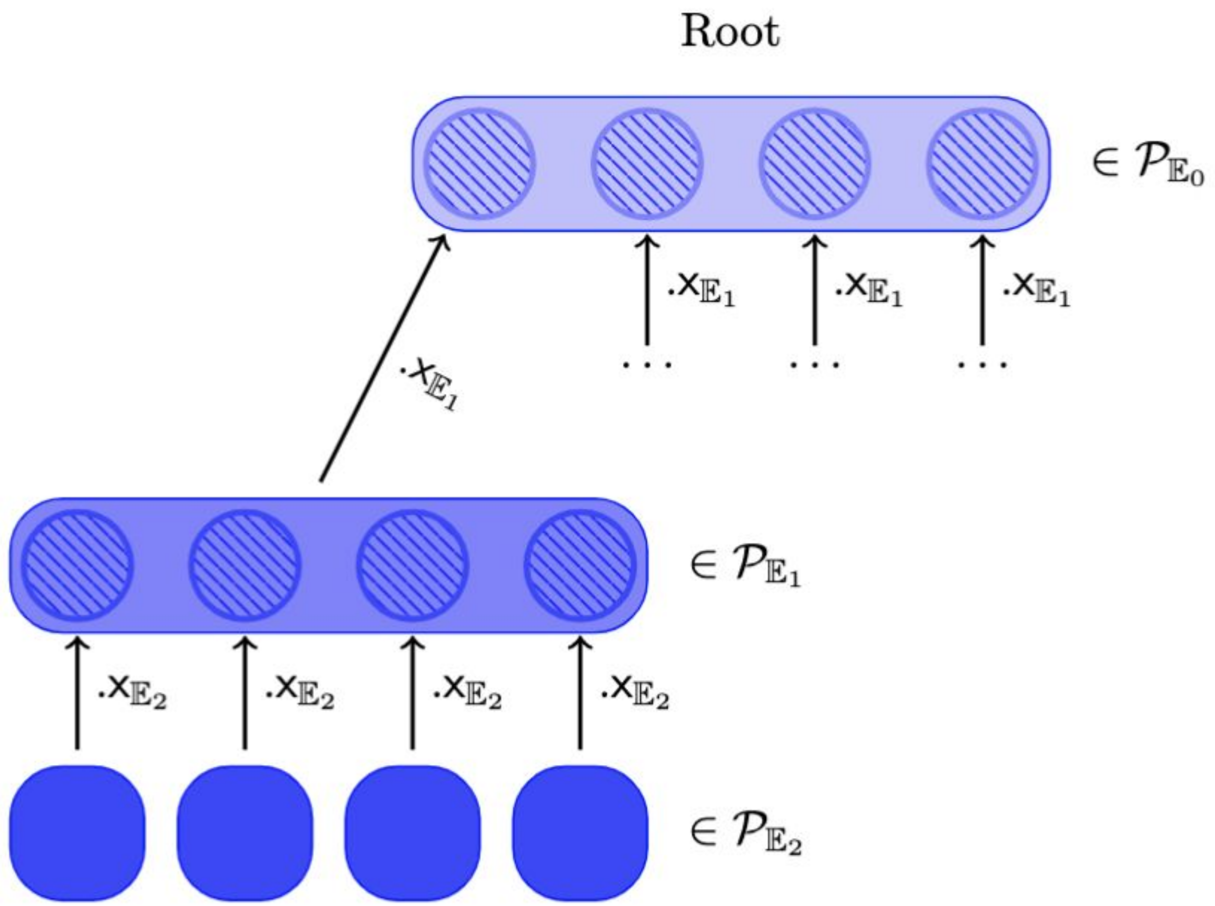
- Let $\mathcal{U}_{\alpha,\beta}(v) \mapsto S(\alpha \cdot v + \beta)$ where S(v) is 1 if v is a quadratic residue, and otherwise 0.

Committing only to x

- Let $\mathcal{U}_{\alpha,\beta}(v) \mapsto S(\alpha \cdot v + \beta)$ where S(v) is 1 if v is a quadratic residue, and otherwise 0.
- $\mathcal{P}_{\mathbb{E}} = \{(\mathrm{x}, \mathrm{y}) \mid (\mathrm{x}, \mathrm{y}) \in \mathbb{E}(\mathbb{F}_p) \wedge \mathcal{U}_{\alpha,\beta}(\mathrm{y}) = 1 \wedge \mathcal{U}_{\alpha,\beta}(-\mathrm{y}) = 0\}$

Committing only to x

- Let $\mathcal{U}_{\alpha,\beta}(v) \mapsto S(\alpha \cdot v + \beta)$ where S(v) is 1 if v is a quadratic residue, and otherwise 0.
- $\mathcal{P}_{\mathbb{E}} = \{(\mathrm{x}, \mathrm{y}) \mid (\mathrm{x}, \mathrm{y}) \in \mathbb{E}(\mathbb{F}_p) \wedge \mathcal{U}_{\alpha,\beta}(\mathrm{y}) = 1 \wedge \mathcal{U}_{\alpha,\beta}(-\mathrm{y}) = 0\}$
- Proving permissibility inside the circuit: $w^2 = (\alpha \cdot v + \beta)$



Root
$\in \mathcal{P}_{\mathbb{E}_0}$
$.x_{\mathbb{E}_1}$
$.x_{\mathbb{E}_1}$
$.x_{\mathbb{E}_1}$
$.x_{\mathbb{E}_1}$
$\cdots$
$\cdots$
$\cdots$
$\in \mathcal{P}_{\mathbb{E}_1}$
$.x_{\mathbb{E}_2}$
$.x_{\mathbb{E}_2}$
$.x_{\mathbb{E}_2}$
$.x_{\mathbb{E}_2}$
$\in \mathcal{P}_{\mathbb{E}_2}$

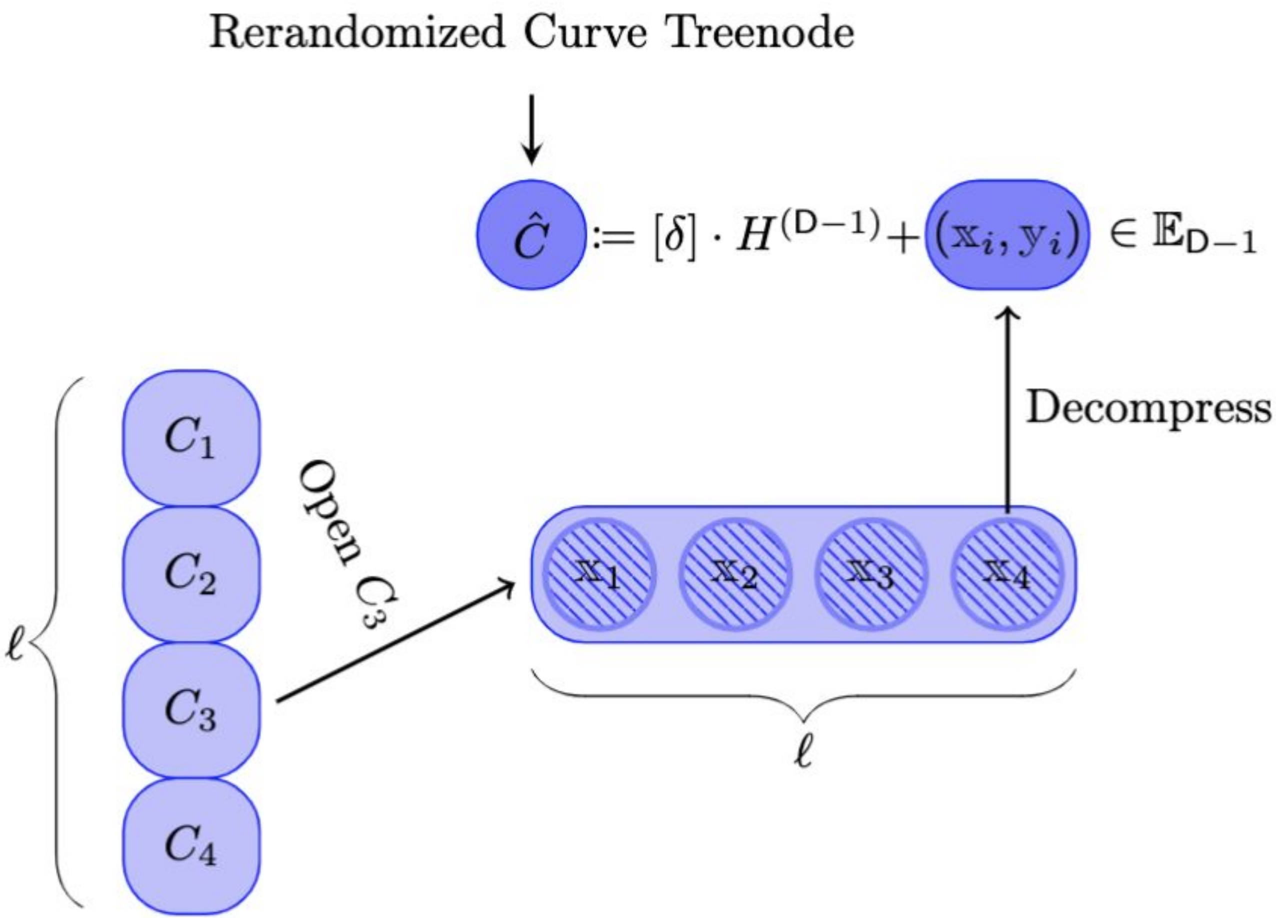
Reducing the number of proofs

- Use a cycle of curves.
- Combine all constraints over the same curve in one proof.

Adding zero knowledge

- Rerandomize the path!
  - **Root, C1, C2, …, Leaf** becomes **Root, C1*, C2*, …, Leaf***
- Show:
  - The first commitment, **C1***, on the path is a rerandomization of a child of the root.
  - The second commitment on the path is rerandomization of a child of **C1***.
  - Etc.

$$\mathcal{R}^{(\text{single-level}^\star, d)} := \left\{ \begin{pmatrix} i, r, \delta, \\ \vec{\mathbb{x}}, \mathbb{y} \end{pmatrix} : \begin{array}{c} C = \langle [\vec{\mathbb{x}}], \vec{G}_{\mathsf{x}}^{(d-1)} \rangle \\ + [r] \cdot H^{(d-1)} \\ \wedge (\mathbb{x}_i, \mathbb{y}) \in \mathcal{P}_{\mathbb{E}^{(d)}} \\ \wedge \hat{C} = (\mathbb{x}_i, \mathbb{y}) + [\delta] \cdot H^{(d)} \end{array} \right\}$$


Rerandomized Curve Treenode
$\hat{C} \coloneqq [\delta] \cdot H^{(\mathsf{D}-1)} + (\mathbb{x}_i, \mathbb{y}_i) \in \mathbb{E}_{\mathsf{D}-1}$
Decompress
Open C_3
C_1
C_2
C_3
C_4
ℓ
$\mathbb{x}_1$
$\mathbb{x}_2$
$\mathbb{x}_3$
$\mathbb{x}_4$
ℓ

Benchmarks of set membership

(D, ℓ)	Set size	# Constraints	Proof size (kb)	Proving time (s)	Verification time (ms)	Amort. batch verification time (ms)
$(2, 1024)$	2^{20}	3870	2.6	1	24.03	1.43
$(4, 256)$	2^{32}	4668	3	1.94	41.78	2.36
$(4, 1024)$	2^{40}	7740	3	1.96	42.88	2.69

- Implemented using the Pasta curves and bulletproofs.
- Bulletproofs R1CS implementation supporting arkworks curves, batch verification, and commitments of arbitrary dimension.
- Code available at github.com/simonkamp/curve-trees

VCash

- Store commitments to coins in a Curve Tree
    a. The value of the coin
    b. The hash of a rerandomizable public key
        - Used for opening *and* nullifying.
- Sending a transaction
    a. Commit to each receivers output value and a rerandomization of their public key.
    b. Open the rerandomized public key of each spent coin.
    c. Show positive value of and balance between minted and spent coins.
    d. Sign the transaction with each spent public key.

Benchmarks of 2-2-pour

	Anonymity set size	Transparent setup	Tx size (kb)	Proving time (S)	Verification time (ms)	Amort. batch verification time (ms)
Zcash	2^{32}	✗	$\boxed{1}$	$\boxed{2.38}$	$\boxed{7}$	-
Veksel	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‡	-
VCash	2^{20}	✓	3.6	1.98	42.75	2.82
	2^{32}	✓	$\boxed{4.1}$	$\boxed{3.85}$	$\boxed{81.27}$	$\boxed{4.94}$
	2^{40}	✓	4.1	3.91	82.83	5.66

Future work

- Batch membership proofs in Curve Trees.
- Stacking the odd and even layers of the Curve Tree.
- The Curve Tree technique applies in any Commit-and-Prove system.

Thank you!

ia.cr/2022/756