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: \( \approx 800,000 \) R1CS constraints.
- Merkle Tree with Pedersen: \( \approx 45,000 \) 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 $\ell$ needs $2\ell$ 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 \( \mathcal{U}_{\alpha,\beta} : \mathbb{F} \to \{0,1\} \)
  - \( \mathcal{U}_{\alpha,\beta} \mapsto S(\alpha \cdot v + \beta) \) where \( S(v) = 1 \iff v \) is a quadratic residue in \( \mathbb{F} \).
  - Prove that \( \mathcal{U}_{\alpha,\beta}(v) = 1 \) 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

Rerandomized Curve Treenode

\[ \hat{C} := [\delta] \cdot H^{(D-1)} + (x_i, y_i) \in \mathbb{E}_{D-1} \]

Decompress

\[ R_{\text{single-level}^{*}, (\_)} := \left\{ \left( i, r, \delta, \begin{pmatrix} \vec{x} \\ y \end{pmatrix} \right) : 
\begin{align*}
C &= \langle [\vec{x}], \hat{G}^x \rangle \\
+ [r] \cdot H_{(\_)} \\
\wedge (x_i, y) &\in \mathcal{P}_{\text{other}(\_)} \\
\wedge \hat{C} &= (x_i, y) + [\delta] \cdot H_{\text{other}(\_)}
\end{align*} \right\} \]
Circuit costs

- Select \( x \)-coordinate: \( \ell - 1 \) constraints.

- Decompress permissible point: 1 constraint.

- Point addition with native coordinates: \( \approx 10 \) constraints.

- Fixed base scalar multiplication: \( \approx 900 \) constraints.
  - Split algebraically incompatible elements into 3-bit windows.
  - Compute scalar multiplication with lookup tables an incomplete addition.
Select and Rerandomize

| Curves   | (D, ℓ) | |S| | # 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

<table>
<thead>
<tr>
<th>Scheme</th>
<th># Constraints</th>
<th>Prove (s)</th>
<th>Verify (ms)</th>
<th>Verify batch (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curve Trees (Pasta)</td>
<td>3565</td>
<td>1.5</td>
<td>31</td>
<td>1.8</td>
</tr>
<tr>
<td>Curve Trees (Secp/Secq)</td>
<td>3565</td>
<td>1.7</td>
<td>37</td>
<td>2</td>
</tr>
<tr>
<td>Poseidon 4:1</td>
<td>4515</td>
<td>8.8</td>
<td>651</td>
<td>-</td>
</tr>
<tr>
<td>Poseidon 8:1</td>
<td>4180</td>
<td>8.5</td>
<td>825</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Anonymity</td>
<td>Transparent</td>
<td>Tx size (kb)</td>
<td>Proving time (S)</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>Zcash Sapling</strong></td>
<td>$2^{32}$</td>
<td>X</td>
<td>2.8</td>
<td>2.38</td>
</tr>
<tr>
<td><strong>Zcash Orchard</strong></td>
<td>$2^{32}$</td>
<td>✓</td>
<td>7.6</td>
<td>1.77</td>
</tr>
<tr>
<td><strong>Veksel</strong></td>
<td>Any</td>
<td>X</td>
<td>5.3</td>
<td>0.44</td>
</tr>
<tr>
<td><strong>Lelantus</strong></td>
<td>$2^{10}$</td>
<td>✓</td>
<td>2.7</td>
<td>0.27†</td>
</tr>
<tr>
<td></td>
<td>$2^{14}$</td>
<td>✓</td>
<td>3.9</td>
<td>2.35†</td>
</tr>
<tr>
<td></td>
<td>$2^{16}$</td>
<td>✓</td>
<td>5.6</td>
<td>4.8†</td>
</tr>
<tr>
<td><strong>Omniring</strong></td>
<td>$2^{10}$</td>
<td>✓</td>
<td>1</td>
<td>$\approx 1.5^\dagger$</td>
</tr>
<tr>
<td><strong>VCash (Pasta)</strong></td>
<td>$2^{20}$</td>
<td>✓</td>
<td>3.4</td>
<td>1.76</td>
</tr>
<tr>
<td></td>
<td>$2^{32}$</td>
<td>✓</td>
<td>4</td>
<td>3.43</td>
</tr>
<tr>
<td></td>
<td>$2^{40}$</td>
<td>✓</td>
<td>4</td>
<td>3.48</td>
</tr>
<tr>
<td><strong>VCash (Secp/Secq)</strong></td>
<td>$2^{20}$</td>
<td>✓</td>
<td>3.4</td>
<td>1.95</td>
</tr>
<tr>
<td></td>
<td>$2^{32}$</td>
<td>✓</td>
<td>4</td>
<td>3.80</td>
</tr>
<tr>
<td></td>
<td>$2^{40}$</td>
<td>✓</td>
<td>4</td>
<td>3.86</td>
</tr>
</tbody>
</table>
Thank you!

Questions?