Lelantus Spark with Curve Trees

Aram Jivanyan

Abstract

A recent construction referred to as Curve Trees is a novel and efficient design for membership proofs which significantly optimizes the communication and computational complexity of the argument including the proof sizes, proving time, and verification time. This enables efficient scaling of the set size to billions of elements and very importantly also provides efficient batch verification techniques which further can decrease the marginal cost of proof verification. We discuss how Lelantus Spark can be implemented with Curve Trees to support full membership proofs.

1 Curve Tree Construction: Commitment Phase

Curve Trees operate over two amicable elliptic curve groups denoted as E_{F_p} and E_{F_q}. The scalar field of E_{F_p} is F_p and its element's affine coordinates belong to the field F_q. Similarly, the scalar field of the elliptic curve group E_{F_q} is F_q and its elements affine coordinates belong to the field F_p. For an elliptic group element C, we denote the x and y coordinates by $x(C)$ and $y(C)$. In Lelantus Spark, the elliptic curve group used is $secp256k1$, which we will refer to as E_{F_p}. The corresponding amicable curve is $secq256k1$ which will be referred to as E_{F_q}.
In Lelantus Spark both the serial commitments and value commitments belong to E_{F_p}. To commit to these commitments, we fix the leaf length parameter d and generate $2d + 1$ generator points

$$\{G_1, \ldots, G_d, H_1, \ldots, H_d, H_{F_q}\} \in E_{F_q}^{2d+1}$$

and $d + 1$ generator points

$$\{F_1, \ldots, F_d, H_{F_p}\} \in E_{F_p}^{d+1}$$

.

First layer commitment: First layer commitment is done through the following steps

- Take all coins and extract the list of serial and value commitment pairs $\{(C_1, V_1), (C_2, V_2) \ldots, (C_N, V_N)\}$. Pad the list by repeating the last pair until the array size becomes equal to $N = d^p$
- Divide the array of all commitment pairs into successive sub-arrays of size d. For brevity, we will skip adding extra sub-indexes and will denote the current sub-array as $\{(C_1, V_1), \ldots, (C_d, V_d)\}$
- Note that all commitment values should be permissible points. We commit each sub-array through a generalized Pedersen commitment scheme to get the first layer commitment element

  $$C_1^1 = x(C_1)G_1 + x(C_2)G_2 + \ldots x(C_d)G_d + x(V_1)H_1 + x(V_2)H_2 + \ldots + x(V_d)H_d + \rho H_{F_q}$$

  Note that the blinding factor ρ should be selected to make the group element C_1^1 a permissible point.
- After committing to all sub-arrays the result will be a set of N/d points

  $$C_1^1, C_2^1, \ldots, C_{N/d}^1$$

  where $\forall k \in 1, \ldots, N/d \quad C_k^1 \in E_{F_q}$

Other layer commitments: Except for the first layer where $2d$ elements are committed through a single commitment, all successive layer commitments will commit to exactly d elements. The fixed generator points for those commitments will be denoted respectively $G_1, \ldots G_d, H_{F_q}$ and $F_1, \ldots F_d, H_{F_p}$ for two consecutive layers. The other layer commitment process for others but the first layer will work as follows

- Take all commitments from the previous layer and divide them into sub-arrays of size d.
- Lets assume $\{C_1^l, C_2^l, \ldots, C_d^l\}$is the current sub-array we need to commit.
  - If $C_k^l \in E_{F_p}^d$ then the next commitment is computed as

$$C^{l+1} = x(C_1^l)G_1 + x(C_2^l)G_2 + \ldots + x(C_d^l)G_d + \rho^l H_{F_p}$$

    . In this case the final commitment point $C^{l+1} \in E_{F_q}$
  - If $C_k^l \in E_{F_q}^d$ then next commitment is computed as

$$C^{l+1} = x(C_1^l)F_1 + x(C_2^l)F_2 + \ldots + x(C_d^l)F_d + \rho^l H_{F_q}$$

    In this case the final commitment point $C^{l+1} \in E_{F_p}$

Note that the blinding factor ρ^l should be selected to make C^{l+1} a permissible point.

2 Implementing Parallel Membership Proofs

In original Lelantus Spark, we have to generate parallel membership proofs over two sets

$$\{C_0, C_1, \ldots, C_{N-1}\}$$

and

$$\{V_0, V_1, \ldots, V_{N-1}\}$$

of commitments where the two values C_i and V_i are logically paired at each position i. In the parallel one out of many proofs, the prover reveals a pair of group elements $\hat{C}$ and $\hat{V}$ and proves that C commits to the same value as some secret element C_i from the set $C_0, C_1, \ldots, C_{N-1}$. He also proves that the revealed V commits to the same value as the secret commitment V_i. Note that C_i and V_i share the same index i here. In existing Spark Implementation the commitments are of the following form

$$C = Comm(s, k, 0) = sF + kG$$

$$V = Com(v, t) = vG + tH$$

These two commitment functions $Comm$ and Com share the same generator points G and H. During the Spend operation, the prover reveals two new commitments referred to as offset commitments of the form

$$\hat{C} = Com(s, k, r)$$

$$\hat{V} = Com(v, t')$$

and proves that the commitments $C_i - \hat{C}$ and $V_i - \hat{V}$ are both a commitment to zero for a secret index i.

$$C_i - \hat{C} = Comm(0, 0, -r)$$

$$V_i - \hat{V} = Com(0, t - t')$$

In Curve Trees, the membership proof works through the following steps

- Given the two sets of commitments $C_1, C_2, \ldots, C_N$ and $V_1, V_2, \ldots, V_N$, the Curve Tree commitment over the set is built resulting in the curve tree root R as is described in the section above.

- To prove the knowledge of a serial and value commitment pair (C_i, V_i) belonging to the set $C_1, C_2, \ldots, C_N$ $V_1, V_2, \ldots, V_N$, two offset commitments $\hat{C}$ and $\hat{V}$ are revealed. Next, a proof is generated that $\hat{C} = C_i + r_c H_{F_P}$ and $\hat{V} = V_i + r_v H_{F_P}$ where H_{F_P} is an independent generator for some secret index i.

It is not trivial to implement parallel membership proofs with Curve Trees hence a question arises as to how to efficiently bind two commitments together during the membership proofs. The suggested steps are the following:

$$\mathbf{R}_{SR}^{\text{first\_level}} := \left\{ \begin{pmatrix} ck = (G_1, \ldots, G_d, H_1, \ldots, H_d, H_p) \in E_{F_p}, \\ H_q \in E_{F_q}, C \in E_{F_p}, \widehat{C} \in E_{F_q}; \\ i, \rho, \widehat{\rho}_C, \widehat{\rho}_V, y(C_i), y(V_i) \\ \vec{x} = (x(C_1), \ldots, x(C_d), x(V_1), \ldots, x_(V_d)) \end{pmatrix} \middle| \begin{array}{r} C = x(C_1)G_1 + \ldots + x(C_d)G_d + \\ x(V_1)H_1 + \ldots + x(V_d)H_d + \rho H_{F_q} \\ \widehat{C} = (x(C_i), y(C_i)) + \widehat{\rho}_C H_q, \\ \widehat{V} = (x(V_i), y(V_i)) + \widehat{\rho}_V H_q, \\ (x(C_i), y(C_i) \in E_{F_q}, (x(V_i), y(V_i)) \in E_{F_q} \\ \text{are permissible points} \end{array} \right\}$$

This relation partially incorporates the logic of the non-optimized version of single-layer proof in the original Curve Tree construction, which has been defined for curve points with two affine coordinates committed in a specified order through independent generators and also requires index correspondence to be proven for two committed coordinate values. Its implementation has never been coded or described in detail, hence we discuss the core implementation difference that differs this layer proof from the other layer proofs.

The challenge here is proving that given two commitments $\widehat{C}$ and $\widehat{V}$, their first affine coordinates $x(C_i)$ and $x(V_i)$ are respectively the i-th and $d + i$-th elements in the vector committed by C. This can be probably proven by different arithmetic relation tricks in the Bulletproof circuit and the described technique below is a suggestion.

- Find the position i of the $\widehat{C}$-s first affine argument $x(C)$ in the committed vector.
- Find the position j of the $\widehat{V}$-s first affine argument $x(V)$ in the committed vector.
- Assert $j = i + d$

For all other layers, the relation will be the same as in the original Curve Tree construction. Its formal description is the following

$$\mathbf{R}_{SR}^{\text{other\_levels}} := \left\{ \begin{pmatrix} ck = (G_1, \ldots, G_d, H_p) \in E_{F_p}, H_q \in E_{F_q}, \\ C \in E_{F_p}, \widehat{C} \in E_{F_q}; \\ i, r, \rho, y \\ \vec{x} = (x(C_1), \ldots, x(C_d)) \end{pmatrix} \middle| \begin{array}{l} C = x_1 G_1 + \ldots + x_d G_d + r H_P \\ \widehat{C} = (x_i, y) + \rho H_q \\ (x_i, y) \in E_{F_q} \text{ is a permissible point} \end{array} \right\}$$

The proof of this relation is described in the original paper in detail and the implementation is the same as implemented in the original paper.

2.1 Batching

The batching of Membership proofs basically will boil down to the batching of Bulletproof proofs which is described in the original bulletproof paper [3]

3 Required Modifications in Lelantus Spark

Making Coins Permissible Points: To support Curve Trees, we need each generated commitment to be a permissible point as defined in [1]. This requires the transaction generator to choose the random blinding parameters so the resulting commitment will become a permissible point. Checking if all commitments are permissible group elements should become part of the transaction verification process.

Lelantus coins are comprised of serial and value commitments denoted by C and V, which are Pedersen commitments computed through a **CreateCoin** algorithm described in the original paper. Below we replicate the algorithm steps with all required modifications to produce commitments represented through permissible points.

1. Parse the recipient address $addr_{\text{pk}} = (Q_0, Q_1, Q_2)$.
2. Do
   (a) Sample $k \in F_p$.
   (b) Compute the recovery key $K = k \cdot Q_0$ and derived recovery key $K_{\text{der}} = k \cdot Q_1$.
   (c) Compute the serial number commitment
   $$C = \text{Comm}(\mathsf{H}_{\text{ser}}(K_{\text{der}}), 0, 0) + Q_2$$
   (d) Compute the value commitment
   $$V = Com(v, \mathsf{H}_{\text{val}}(K_{\text{der}}))$$
   .

   While both C and V are not produced as permissible points

As approximately 1 out of 4 points on the curve are permissible, this will require 16 iterations over k in average before finding an appropriate value that makes both C and V permissible.
Transition from SPARK anonymity sets to Curve Tree-based anonymity sets: There can be different ways to ensure transformation of the anonymity sets:

1. **Naive Approach:** The naive transition process could be spending all Spark coins and minting them as new fresh CT coins. This may be more risky from the privacy violation perspective.
2. **Transition:** A simple Spark spend transaction can be executed to spend all existing Spark coins and Mint new CT coins which are permissible. This can help to start a fresh new anonymity set whose size can grow to billions over time.

3.1 Acknowledgment

The author thanks Luke Parker and Aaron Feickert for their invaluable support along the whole process of understanding Curve Trees and how Lelantus Spark can leverage them.

References

[1] Matteo Campanelli, Mathias Hall-Andersen, Simon Holmgaard Kamp. Curve Trees: Practical and Transparent Zero-Knowledge Accumulators https://eprint.iacr.org/2022/756

[2] Liam Eagen. Zero Knowledge Proofs of Elliptic Curve Inner Products from Principal Divisors and Weil Reciprocity. https://eprint.iacr.org/2022/596

[3] Benedict Bunz et al. Bulletproofs. Short proofs for confidential transactions and more https://eprint.iacr.org/2017/1066