Ring Proof Specification

21-08-2024-draft-4

Abstract

This document describes a cryptographic scheme based on SNARKs (Succinct Non-Interactive Arguments of Knowledge) that enables a prover to demonstrate knowledge of a secret scalar t and a secret index k within a group of public keys, where each public key is a point on an elliptic curve. The scheme ensures that, when combined with a public elliptic curve point H, the relation $R = PK_k + t\text{ů}H$ is satisfied. It leverages elliptic curve operations, a polynomial commitment scheme, and the Fiat-Shamir heuristic to achieve non-interactivity and zero-knowledge properties.

1. Notation

1.1. Basics

Basic Sets

- $\mathbb{N}_k = \{0, \ldots, k-1\}$
- $\mathbb{B} = \mathbb{N}_2$

Vectors Operations

- $\overline{x} = (x_0, \ldots, x_{n-1}),\ \overline{x}_i = x_i,\ 0 \leq i < n$
- $\overline{a} \| \overline{b} = (a_0, \ldots, a_{n-1}, b_0, \ldots, b_{m-1})$
- $x^{\|n} = (x, \ldots, x) \in X^n$

Kronecker Delta

- $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

Lagrange Basis Polynomials

- $L_i = L_{\mathbb{D},i} \in \mathbb{F}[X]^{<N}, \quad i \in \mathbb{N}_N, \quad L_i(\omega^j) = \delta_{ij}$

1.2. Curves and Fields

- $\langle \omega \rangle = \mathbb{D} \subseteq \mathbb{F}^*, \quad |\mathbb{D}| = N \in \mathbb{N}$
  - Cyclic subgroup generated by ω in the multiplicative group $\mathbb{F}^*$.

Elliptic Curves

- $J = J/\mathbb{F}$ – Elliptic curve J defined over the field $\mathbb{F}$.
- $\tilde{\mathbb{J}} = J(\mathbb{F})$ – Group of $\mathbb{F}$-rational points on J.
- $\mathbb{J} \subset \tilde{\mathbb{J}}$ – Prime order subgroup of $\tilde{\mathbb{J}}$.

Scalar Field

- $\mathbb{F}_{\mathbb{J}}$ – Field associated with the elliptic curve $\mathbb{J}$, with $|\mathbb{F}_{\mathbb{J}}| = |\mathbb{J}|$.
- $N_J = \lceil \log_2 |\mathbb{F}_{\mathbb{J}}| \rceil$ – Number of bits to represent an element of $\mathbb{F}_{\mathbb{J}}$.
- $N_K = N - N_J - 4$ – Maximum size of the ring handled with a domain of size N.

1.3. Support Functions

Unzip

- $\text{unzip} : \mathbb{J}^k \to (\mathbb{F}^k, \mathbb{F}^k); \quad \overline{p} \to (\overline{p}_x, \overline{p}_y)$
  - Given a vector $\overline{p}$ of k elliptic curve points, unzip separates $\overline{p}$ into two vectors: $\overline{p}_x$ and $\overline{p}_y$, containing the x and y coordinates of each point, respectively.

Polynomial Interpolation

- $\text{Interpolate} : \mathbb{F}^k \to \mathbb{F}[x]^{<k}; \quad \overline{x} \to f$

Polynomial Commitment Scheme

- $\text{PCS.Commit} : \mathbb{F}[x] \to \mathbb{G}; \quad f \to C_f$
  - Commits to a polynomial f over $\mathbb{F}$, with commitment in group $\mathbb{G}$. When applied to a vector $\bar{x}$, the components are interpolated over the domain $\mathbb{D}$ to form f.
- $\text{PCS.Open} : (\mathbb{G}, \mathbb{F}) \to (\mathbb{F}, \Pi); \quad (C_f, x) \to (y, \pi)$
  - Evaluates the committed polynomial f at point x, returning evaluation y and proof π. The proof domain Π depends on the PCS.
- $\text{PCS.Verify} : (\mathbb{G}, \mathbb{F}, \mathbb{F}, \Pi) \to \mathbb{B}; \quad (C_f, x, y, \pi) \to (0|1)$
  - Verifies whether $y = f(x)$ given the commitment C_f and proof π.

Fiat-Shamir Transform

- $\text{FS} : \mathbb{S} \to \mathbb{F}; \ \mathbf{s} \to x$
  - Maps a serializable object $\mathbf{s} \in \mathbb{S}$ to $\mathbb{F}$, typically via some cryptographically secure hash function.

2. Parameters

2.1. Scheme Specific

- $\square \in \mathbb{J}$ – Padding element, a point on $\mathbb{J}$ with unknown discrete logarithm.
- $H \in \mathbb{J}$ – Pedersen blinding base point.
- $\overline{H} = (H, 2H, 4H, \dots, 2^{N_J - 1}H) \in \mathbb{J}^{N_J}$ – Vector of scaled multiples of H.
- $S \in \tilde{\mathbb{J}} \setminus \mathbb{J}$ – Point in $\tilde{\mathbb{J}}$ used as seed for accumulation, ensuring the result is never the identity.

2.2. Public Data

- $\overline{PK} \in \mathbb{J}^{N_K}$ – Vector of public keys in the ring, padded with $\square$ to length N_K if needed.

2.3. Witness Data

- $t \in \mathbb{F}_{\mathbb{J}}$ – Prover's one-time secret.
- $k \in \mathbb{N}_{N_K}$ – Prover's index within the ring, identifying which public key in $\overline{PK}$ belongs to the prover.

2.4. Preprocessing

2.4.1. Public Input Preprocessing Concatenate ring points with scaled multiples of H:

$$\overline{P} = \overline{PK} \| \overline{H} = (P_0, \dots, P_{N-5}) \in \mathbb{J}^{N-4}$$

$$\overline{p}_x = (P_{x,0}, \dots, P_{x,N-5}, 0, 0, 0, 0) \in \mathbb{F}^N$$

$$\overline{p}_y = (P_{y,0}, \dots, P_{y,N-5}, 0, 0, 0, 0) \in \mathbb{F}^N$$

Ring items selector:

$$\overline{s} = 1^{\|N_K} \ \| \ 0^{\|N - N_K} \in \mathbb{F}^N$$

2.4.1 Interpolation The resulting vectors are interpolated over $\mathbb{D}$:

- $p_x = \text{Interpolate}(\overline{p}_x)$.
- $p_y = \text{Interpolate}(\overline{p}_y)$.
- $s = \text{Interpolate}(\overline{s})$.

2.4.2. Commit to the constructed vectors:

$$C_{p_x} = \text{PCS.Commit}(\overline{p}_x)$$
$$C_{p_y} = \text{PCS.Commit}(\overline{p}_y)$$
$$C_s = \text{PCS.Commit}(\overline{s})$$

2.5. Relation to Prove

Knowledge of k and t such that $R = PK_k + tH$.

$$\mathfrak{R}_H = \{(R, \overline{PK}; k, t) \mid R = PK_k + tH \,;\, R \in \mathbb{J},\, \overline{PK} \in \mathbb{J}^{N_K},\, k \in \mathbb{N}_{N_K},\, t \in \mathbb{F}_{\mathbb{J}}\}$$

---

3. Prover

3.1. Witness Polynomials

3.1.1. Bits Vector

- $\overline{k} \in \mathbb{B}^{N_K}$ – Binary vector representing the index k in the ring. $\overline{k}$ has N_K elements where $k_i = \delta_{ik}$
- $\overline{t} \in \mathbb{B}^{N_J}$ – Binary representation of the secret scalar t, with t_i representing the i-th bit of t in little-endian order, i.e., $t = \sum t_i 2^i$, for $i \in \mathbb{N}_{N_J}$

The bits vector $\overline{b}$ is constructed by concatenating $\overline{k}$ and $\overline{t}$, followed by a single 0.

$$\overline{b} = \overline{k} \| \overline{t} \| (0)$$

3.1.2. Conditional Sum Accumulator Vectors

$$ACC_0 = S, \quad ACC_i = ACC_{i-1} + b_{i-1} P_{i-1}, \quad i = 1, \ldots, N-4$$

- The accumulator is initialized with the seed point S.
- The accumulator is updated at each index i based on the previous value and the product of b_{i-1} and P_{i-1}.

The resulting accumulator points are finally separated into x and y coordinates:

$$(\overline{acc}_x, \overline{acc}_y) = \text{unzip}(\overline{ACC})$$

3.1.3. Inner Product Accumulator Vector

$$acc_{ip_0} = 0, \quad acc_{ip_i} = acc_{ip_{i-1}} + b_{i-1} s_{i-1}, \quad i = 1, \ldots, N-4$$

- The accumulator is initialized with 0.
- The accumulator is updated at each index i based on the previous value and the product of b_{i-1} and s_{i-1}

3.1.4. Interpolation

The resulting vectors are interpolated over $\mathbb{D}$ with random values $\{r_i\}$ appended as padding for the final entries. This padding helps obscure the resulting polynomial, even when committing to identical witness values.

- $b = \text{Interpolate}(\bar{b} \| (r_1, r_2, r_3))$.
- $acc_x = \text{Interpolate}(\overline{acc}_x \| (r_4, r_5, r_6))$.
- $acc_y = \text{Interpolate}(\overline{acc}_y \| (r_7, r_8, r_9))$.
- $acc_{ip} = \text{Interpolate}(\overline{acc}_{ip} \| (r_{10}, r_{11}, r_{12}))$.

3.2. Constraints

Constraints are polynomials constructed to evaluate to zero when satisfied; a non-zero evaluation indicates a violation.

Note. When evaluating a polynomial f at $x = \omega^k \in \mathbb{D}$ for some $k \in \mathbb{N}$, $f(\omega x)$ gives the value of the polynomial at the next position in the evaluation domain ($\omega x = \omega^{k+1}$).

3.2.1. Inner Product

$$c_1(x) = \big(acc_{ip}(\omega x) - acc_{ip}(x) - b(x)s(x)\big)(x - \omega^{N-4})$$

This constraint ensures the inner product accumulator $acc_{ip}(x)$ is correctly updated, satisfying $acc_{ip}(\omega x) = acc_{ip}(x) + b(x)s(x)$.

The factor $(x - \omega^{N-4})$ ensures the constraint holds at all points including $x = \omega^{N-4}$, where $c_1(x)$ automatically vanishes.