

# S-ZKPoK: Sedenionic Zero-Knowledge Proof of Knowledge

## Leveraging the Isomorphism of Polynomials

Aaron M. Schutza

February 24, 2026

### Abstract

Traditional Zero-Knowledge Proofs (ZKPs), such as Schnorr protocols and modern SNARKs, rely heavily on associative homomorphisms to verify cryptographic statements without revealing underlying secrets. The non-associative nature of Sedenionic algebra inherently breaks these associative verification loops. This paper introduces S-ZKPoK, a non-interactive Zero-Knowledge Proof of Knowledge that completely bypasses Sedenionic non-associativity by executing the proof upon the external affine matrices that mask the algebra. By utilizing the Isomorphism of Polynomials (IP) problem as a  $\Sigma$ -protocol, a prover can perfectly demonstrate knowledge of a Sedenionic trapdoor without exposing the private matrices.

## 1 Introduction

In Multivariate Quadratic (MQ) systems, such as the SQE Key Encapsulation Mechanism, a public key is a quadratic map  $\mathcal{P}(X)$  masked by secret affine transformations  $L_1$  and  $L_2$ . Proving ownership of this identity without revealing  $L_1$  and  $L_2$  requires a paradigm shift from traditional discrete logarithm ZKPs.

S-ZKPoK resolves this by treating the Sedenionic core as an immutable, non-associative black box. The proof operates entirely on the external linear algebra, challenging the prover to demonstrate an exact isomorphism between a randomized commitment map and the established public key.

## 2 The 3-Pass $\Sigma$ -Protocol

Let Alice possess a public Sedenionic map  $\mathcal{P}(X) \equiv L_1 \cdot ((L_2 \cdot X)^2) \pmod{p}$ . She wishes to prove knowledge of  $(L_1, L_2)$  to Bob.

### 2.1 Step 1: The Commitment

Alice generates two random, invertible  $16 \times 16$  blinding matrices  $R_1, R_2 \in GL_{16}(\mathbb{GF}(p))$ . She constructs a blinded quadratic map:

$$\mathcal{C}(X) \equiv R_1 \cdot ((R_2 \cdot X)^2) \pmod{p}$$

Alice hashes the evaluations of this map on a set of deterministic test vectors and sends the resulting *Commit* hash to Bob.

### 2.2 Step 2: The Challenge

Bob replies with a random challenge bit  $c \in \{0, 1\}$ .

### 2.3 Step 3: The Response

Alice provides a response strictly determined by  $c$ :

- **If  $c = 0$  (Prove Commitment):** Alice reveals the random matrices  $R_1$  and  $R_2$ . Bob recalculates  $\mathcal{C}(X)$  and verifies the *Commit* hash. This proves Alice generated a valid Sedenionic structure.
- **If  $c = 1$  (Prove Isomorphism):** Alice reveals two bridge matrices that link her random commitment to her secret key:

$$Q_1 \equiv R_1 \cdot L_1^{-1} \pmod{p}$$
$$Q_2 \equiv L_2^{-1} \cdot R_2 \pmod{p}$$

Bob verifies the isomorphism by evaluating the public key through the bridge matrices:  $Q_1 \cdot \mathcal{P}(Q_2 \cdot X)$ . Algebraically, this evaluates as:

$$Q_1 \cdot L_1 \cdot ((L_2 \cdot Q_2 \cdot X)^2) \equiv R_1 \cdot ((R_2 \cdot X)^2) \equiv \mathcal{C}(X)$$

Bob verifies the resulting map matches the *Commit* hash.

## 3 Non-Interactive Zero-Knowledge (NIZK)

To adapt S-ZKPoK into a non-interactive proof suitable for decentralized networks, the Fiat-Shamir Heuristic is applied.

The prover executes 128 parallel commitment rounds ( $\mathcal{C}_1 \dots \mathcal{C}_{128}$ ). Instead of receiving a challenge from a verifier, the prover concatenates and hashes all 128 commitments to deterministically generate the 128-bit challenge vector. The resulting proof bundle contains the commitments and the mathematically forced responses, achieving 128-bit quantum security in a standalone, verifiable broadcast.

## 4 Security Analysis

**Zero-Knowledge:** If  $c = 0$ , the verifier learns purely random matrices  $R_1, R_2$ . If  $c = 1$ , the verifier learns  $Q_1, Q_2$ , which act as perfect one-time pads blinding the private keys  $L_1, L_2$ . No information regarding the private key is leaked.

**Soundness:** An adversary without  $L_1, L_2$  cannot simultaneously answer both  $c = 0$  and  $c = 1$ . Generating a fraudulent proof for 128 rounds without the private key requires guessing the correct 128-bit hash output, rendering forgery computationally intractable.