SUMSIG: Compact Code-Based Signatures from Sum-Check Protocols

Deterministic Signing with Algebraic Soundness and Tight Reductions

Debrup Chatterjee
debrupc3@gmail.com

December 28, 2025

Abstract

We present SumSig, a code-based digital signature scheme that leverages sum-check protocols to achieve compact signatures with deterministic signing and quasi-tight security reductions. Our construction replaces the traditional Fiat–Shamir repetition-based paradigm which typically requires 80–130 rounds depending on the per-round soundness error with algebraic sum-check proofs over extension fields, achieving one-shot soundness with negligible error.

The core technical contributions are: (1) a reformulation of the syndrome decoding witness as a multilinear polynomial enabling verification via sum-check; (2) a *degree-doubling binarity test* enforcing binary witnesses through power-sum constraints $S_1 = S_2 = S_4 = w$ over an extension field; and (3) a *linearization helper polynomial* enabling perfect zero-knowledge simulation.

For 128-bit security, SumSig achieves signature sizes of approximately 5–8 KB with public keys of 50–100 KB (depending on the polynomial commitment scheme), offering a balanced trade-off compared to Wave (0.8–1.6 KB signatures but 3.5–13 MB public keys) and LESS (8–32 KB signatures, 14–65 KB public keys). Unlike existing code-based signatures, SumSig provides **fully deterministic** signing with zero abort probability and a **quasi-tight** security reduction to the Syndrome Decoding problem in the random oracle model.

Keywords: Code-based cryptography, digital signatures, sum-check protocol, zero-knowledge proofs, post-quantum cryptography, SUMSIG

Contents

1 Introduction **3**

2 Preliminaries **4**
2.1 Notation and Conventions 4
2.2 Cryptographic Primitives 5
2.3 Multilinear Polynomials and Lagrange Interpolation 5
2.4 The Syndrome Decoding Problem 5
2.5 The Schwartz-Zippel Lemma 6
2.6 The Sum-Check Protocol 6
2.7 Polynomial Commitment Schemes 6

3 Degree-Doubling Binarity Enforcement **7**
3.1 The Insufficiency of Aggregate Checks 7
3.2 The Degree-Doubling Test 7
3.3 Batched Verification via Sum-Check 9

4 Signature Scheme Construction **10**
4.1 Zero-Knowledge Masking . . . 10
4.2 Linearization Helper for Zero-Knowledge Simulation . . . 11
4.3 Complete Protocol . . . 12

5 Security Analysis **13**
5.1 Security Model . . . 13
5.2 Cryptographic Assumptions . . . 14
5.3 Soundness Error Decomposition . . . 14
5.4 Main Security Theorem . . . 15
5.5 Proof of Theorem 5.5 . . . 15
5.6 Zero-Knowledge Property . . . 17
5.7 Fiat-Shamir Validity . . . 17

6 Efficiency Analysis **18**
6.1 Parameter Selection . . . 18
6.2 Size Analysis . . . 18
6.3 Computational Costs . . . 19
6.4 Side-Channel Considerations . . . 19

7 Conclusion **19**

A PCS Instantiation: Brakedown Case Study **21**
A.1 Brakedown Overview . . . 21
A.2 Concrete Sizes . . . 21
A.3 Public Key Size . . . 22

B Full Protocol Pseudocode **22**

1 Introduction

Code-based cryptography, founded on the hardness of decoding random linear codes, has been a cornerstone of post-quantum cryptographic research since McEliece's seminal work in 1978 [1]. The Syndrome Decoding (SD) problem determining whether a given syndrome admits a low-weight preimage is NP-complete [2] and has resisted both classical and quantum attacks for over four decades. Despite this strong theoretical foundation, code-based *signatures* have historically lagged behind their lattice-based and hash-based counterparts in terms of compactness and practical efficiency.

The fundamental challenge lies in efficiently proving knowledge of a low-weight error vector $\mathbf{e} \in \mathbb{F}_2^n$ satisfying $H\mathbf{e}^\top = \mathbf{s}$ and $\mathsf{wt}(\mathbf{e}) = w$, where H is a parity-check matrix, $\mathbf{s}$ is a syndrome, and $\mathsf{wt}(\cdot)$ denotes Hamming weight. The predominant approach employs the Fiat–Shamir transform [3] applied to Σ-protocols for proving knowledge of such preimages. These protocols typically achieve soundness error $\varepsilon \in [1/3, 1/2]$ per round, necessitating $R = \lceil \lambda / \log_2(1/\varepsilon) \rceil$ repetitions approximately 80–130 rounds for $\lambda = 128$ bits of security depending on the protocol structure. Each repetition contributes commitment and response overhead, leading to signature sizes that scale linearly with R. At a high level, this inefficiency stems from treating each coordinate check independently rather than aggregating the entire witness relation algebraically.

Table 1 presents a comprehensive comparison of existing code-based signatures alongside lattice-based Dilithium for reference. Several observations emerge: Wave [7] achieves remarkably small signatures (under 2 KB) but requires public keys exceeding 3 MB, making it impractical for many applications. LESS [8] and MEDS [9] offer more balanced trade-offs but remain non-deterministic (requiring rejection sampling) and have non-tight security reductions. Lattice-based Dilithium [10] achieves the smallest signatures with compact keys but relies on fundamentally different hardness assumptions.

Table 1: Comparison of signature schemes at NIST security levels I/III/V. All sizes in bytes unless noted. "Q-Tight" = quasi-tight.

Scheme	**Level**	**Sig (B)**	**PK (B)**	**SK (B)**	**Det.**	**Tight**	**Assumption**
Dilithium-2	I	2,420	1,312	2,528	✓	✓	MLWE/MSIS
Dilithium-3	III	3,293	1,952	4,000	✓	✓	MLWE/MSIS
Dilithium-5	V	4,595	2,592	4,864	✓	✓	MLWE/MSIS
Wave-822	I	822	3.5 MB		×	Partial	SD (ternary)
Wave-1249	III	1,249	7.5 MB		×	Partial	SD (ternary)
Wave-1644	V	1,644	13 MB		×	Partial	SD (ternary)
LESS-1b	I	8,601	14,028	32	×	Partial	LEq
LESS-3b	III	18,841	35,340	48	×	Partial	LEq
LESS-5b	V	33,280	66,150	64	×	Partial	LEq
MEDS-9923	I	9,896	9,923		×	Partial	MEq
MEDS-41711	III	41,080	41,711		×	Partial	MEq
MEDS-134180	V	132,528	134,180		×	Partial	MEq
SumSig	I	∼6,000	∼50 KB	32	✓	Q-Tight	SD (binary)
SumSig	III	∼8,000	∼75 KB	48	✓	Q-Tight	SD (binary)
SumSig	V	∼12,000	∼100 KB	64	✓	Q-Tight	SD (binary)

Remark 1.1 (Comparison Notes). Several caveats apply to the comparison in Table 1:

- **Dilithium** relies on lattice assumptions (Module-LWE/SIS), which are structurally different from code-based assumptions. Cross-assumption comparisons should be interpreted cautiously.
- **Wave** achieves remarkably small signatures (< 2 KB) but requires multi-megabyte public keys, making it unsuitable for bandwidth-constrained applications.

- **LESS** and **MEDS** are based on equivalence problems (linear/matrix code equivalence) rather than syndrome decoding.
- **SumSig** sizes are estimates based on Brakedown PCS; actual sizes depend on implementation choices. "Q-Tight" indicates quasi-tight reduction with additive ϵ_{PCS} loss.

Our construction, SumSig, takes a fundamentally different approach. Rather than repeating a weak Σ-protocol many times, we reformulate the entire witness relation as a polynomial identity over an extension field $\mathbb{F}_q$ where $q = 2^\kappa$, then verify this identity using the sum-check protocol [4]. The sum-check protocol achieves soundness error $O(d\mu/|\mathbb{F}_q|)$ in a single interaction, where d is the polynomial degree and $\mu = \log_2 n$. For $|\mathbb{F}_q| = 2^{128}$, this yields negligible soundness error without repetition. A key technical challenge is *binarity enforcement*: ensuring the prover's witness polynomial evaluates to binary values $\{0, 1\}$ on the Boolean hypercube, not arbitrary field elements. We solve this through a *degree-doubling test* using power-sum constraints $S_1 = S_2 = S_4 = w$.

SumSig achieves the following properties simultaneously: (1) **deterministic signing** with zero abort probability, eliminating rejection sampling and associated side-channel vulnerabilities; (2) **quasi-tight security reduction** to the Syndrome Decoding problem in the random oracle model, with security loss $O(q_h/|\mathbb{F}_q|) + n \cdot \epsilon_{\mathsf{PCS}}$; (3) **balanced size trade-off** with moderate signatures and keys; and (4) **constant-time implementation** by construction, as all operations process the full witness uniformly.

The remainder of this paper is organized as follows. Section 2 establishes notation and prerequisites. Section 3 presents our degree-doubling binarity test. Section 4 describes the full signature scheme. Section 5 provides comprehensive security analysis. Section 6 analyzes concrete efficiency and parameters. Section 7 concludes with limitations and future work. Appendix A provides a concrete PCS instantiation case study.

2 Preliminaries

2.1 Notation and Conventions

We use λ to denote the security parameter. For a finite set S, we write $x \leftarrow S$ for uniform random sampling. For a probabilistic algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x)$ to denote running $\mathcal{A}$ on input x with fresh randomness.

Typographical conventions. Vectors are denoted in bold lowercase (e.g., $\mathbf{x}, \mathbf{e}, \mathbf{r}$). Matrices are denoted in bold uppercase (e.g., $\mathbf{H}$). Scalars and field elements are non-bold (e.g., x, γ, α). Polynomials are denoted by uppercase letters (e.g., P, Q) or with tildes for multilinear extensions (e.g., $\tilde{e}, \tilde{H}$).

Asymptotic notation. We use standard asymptotic notation:
- $f(n) = O(g(n))$: there exist constants $c, n_0 > 0$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.
- $f(n) = \Omega(g(n))$: there exist constants $c, n_0 > 0$ such that $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$.
- $f(n) = \Theta(g(n))$: $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- $\mathsf{negl}(\lambda)$: a function f such that $f(\lambda) = O(\lambda^{-c})$ for all $c > 0$.

Fields and indexing. Let $\mathbb{F}_q$ denote the finite field with q elements. We work primarily with $q = 2^\kappa$ for extension fields of characteristic 2. For $\mathbf{x} = (x_1, \ldots, x_\mu) \in \{0, 1\}^\mu$, define the integer encoding:

$$\langle \mathbf{x} \rangle := \sum_{i=1}^{\mu} x_i \cdot 2^{i-1} \in \{0, 1, \ldots, 2^\mu - 1\} \tag{1}$$

For a vector $\mathbf{e} \in \mathbb{F}_q^n$, the Hamming weight is $\mathsf{wt}(\mathbf{e}) := |\{i : e_i \neq 0\}|$.

Multilinear extensions. Throughout this paper, $\tilde{e}$ denotes the multilinear extension of an error vector $\mathbf{e}$. The MLE $\tilde{e} : \mathbb{F}_q^\mu \to \mathbb{F}_q$ is defined over the full extension field but agrees with the original vector on the Boolean hypercube: $\tilde{e}(\mathbf{x}) = e_{\langle \mathbf{x} \rangle}$ for all $\mathbf{x} \in \{0,1\}^\mu$ where $n = 2^\mu$. When we write sums $\sum_{\mathbf{x} \in \{0,1\}^\mu} \tilde{e}(\mathbf{x})$, we sum over the Boolean hypercube where $\tilde{e}$ coincides with $\mathbf{e}$. Similarly, $\tilde{H}$ and $\tilde{s}$ denote multilinear extensions of the parity-check matrix and syndrome.

2.2 Cryptographic Primitives

Definition 2.1 (Pseudorandom Function)**.** A *pseudorandom function* $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^m$ is a keyed function such that for a random key $k \leftarrow \{0,1\}^\lambda$, the function $\mathsf{PRF}(k, \cdot)$ is computationally indistinguishable from a truly random function. In practice, we instantiate PRF with HMAC-SHA256 or SHAKE128.

Definition 2.2 (Hash Function / Random Oracle)**.** We model hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_q$ as random oracles in our security proofs. In practice, we instantiate with SHA3-256 or SHAKE128, interpreting outputs as field elements.

2.3 Multilinear Polynomials and Lagrange Interpolation

Definition 2.3 (Multilinear Extension)**.** Let $f : \{0,1\}^\mu \to \mathbb{F}_q$ be a function defined on the Boolean hypercube. The *multilinear extension* (MLE) of f, denoted $\tilde{f} : \mathbb{F}_q^\mu \to \mathbb{F}_q$, is the unique polynomial of degree at most 1 in each variable such that $\tilde{f}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^\mu$.

The MLE can be explicitly constructed via Lagrange interpolation over the Boolean hypercube. Define the *Lagrange basis polynomial* (also called the equality function) for a point $\mathbf{w} \in \{0,1\}^\mu$:

Definition 2.4 (Lagrange Basis / Equality Polynomial)**.** For $\mathbf{w} = (w_1, \ldots, w_\mu) \in \{0,1\}^\mu$, the Lagrange basis polynomial is:

$$\mathrm{eq}(\mathbf{X}, \mathbf{w}) := \prod_{i=1}^{\mu} (w_i X_i + (1 - w_i)(1 - X_i)) \tag{2}$$

This polynomial satisfies $\mathrm{eq}(\mathbf{v}, \mathbf{w}) = 1$ if $\mathbf{v} = \mathbf{w}$ and $\mathrm{eq}(\mathbf{v}, \mathbf{w}) = 0$ otherwise, for any $\mathbf{v}, \mathbf{w} \in \{0,1\}^\mu$.

Using the Lagrange basis, the MLE of any function $f : \{0,1\}^\mu \to \mathbb{F}_q$ can be written explicitly as:

$$\tilde{f}(\mathbf{X}) = \sum_{\mathbf{w} \in \{0,1\}^\mu} f(\mathbf{w}) \cdot \mathrm{eq}(\mathbf{X}, \mathbf{w}) \tag{3}$$

This representation is standard in the sum-check literature; see Thaler [20] for a comprehensive treatment.

2.4 The Syndrome Decoding Problem

Definition 2.5 (Syndrome Decoding Problem)**.** Let n, k, w be positive integers with $k < n$ and $w < n$. The *Syndrome Decoding* problem $\mathsf{SD}(n, k, w)$ is: given a uniformly random matrix $\mathbf{H} \leftarrow \mathbb{F}_2^{(n-k) \times n}$ and syndrome $\mathbf{s} = \mathbf{H}\mathbf{e}^\top$ for a uniformly random $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathsf{wt}(\mathbf{e}) = w$, find any $\mathbf{e}' \in \mathbb{F}_2^n$ such that $\mathbf{H}(\mathbf{e}')^\top = \mathbf{s}$ and $\mathsf{wt}(\mathbf{e}') = w$.

Theorem 2.6 (Hardness of SD [2])**.** *The decision version of $\mathsf{SD}(n, k, w)$ is NP-complete. The best known classical algorithms (information set decoding variants [15]) require time $2^{\Omega(n)}$ when $w = \Theta(n)$ and the rate k/n is constant. Quantum algorithms provide at most a polynomial speedup.*

2.5 The Schwartz-Zippel Lemma

The following lemma is fundamental to our soundness analysis:

Lemma 2.7 (Schwartz-Zippel [5, 6])**.** *Let $P \in \mathbb{F}_q[X_1, \ldots, X_n]$ be a non-zero polynomial of total degree d. For uniformly random $r_1, \ldots, r_n \leftarrow \mathbb{F}_q$:*

$$\Pr[P(r_1, \ldots, r_n) = 0] \leq \frac{d}{|\mathbb{F}_q|} \tag{4}$$

2.6 The Sum-Check Protocol

The sum-check protocol [4] enables a verifier to efficiently check claims about sums over the Boolean hypercube.

Theorem 2.8 (Sum-Check Protocol)**.** *Let $P : \mathbb{F}_q^\mu \to \mathbb{F}_q$ be a μ-variate polynomial with individual degree at most d in each variable. Given a claimed sum T, the sum-check protocol allows a verifier to check whether:*

$$\sum_{\mathbf{x} \in \{0,1\}^\mu} P(\mathbf{x}) = T \tag{5}$$

The protocol proceeds in μ rounds. In round i, the prover sends a univariate polynomial $p_i(X_i)$ of degree at most d, and the verifier checks $p_i(0) + p_i(1) = \mathit{claimed}_{i-1}$, then samples $\tau_i \leftarrow \mathbb{F}_q$ and updates $\mathit{claimed}_i := p_i(\tau_i)$.

***Completeness**: If $T = \sum_{\mathbf{x}} P(\mathbf{x})$, the honest prover always convinces the verifier.*

***Soundness**: If $T \neq \sum_{\mathbf{x}} P(\mathbf{x})$, then for any cheating prover:*

$$\Pr[\textit{Verifier accepts}] \leq \frac{d \cdot \mu}{|\mathbb{F}_q|} \tag{6}$$

where the probability is over the verifier's random challenges $\tau_1, \ldots, \tau_\mu$.

Proof sketch. Soundness follows by induction on μ. If the claimed sum is incorrect, then either p_1 differs from the true univariate restriction (detectable with probability $\geq 1 - d/|\mathbb{F}_q|$ by Schwartz-Zippel), or the reduced claim for rounds $2, \ldots, \mu$ is incorrect. The union bound over μ rounds gives the stated error. □

2.7 Polynomial Commitment Schemes

Definition 2.9 (Polynomial Commitment Scheme)**.** A *polynomial commitment scheme* $\mathsf{PCS} = (\mathsf{PCS.Commit}, \mathsf{PCS.Open}, \mathsf{PCS.Verify})$ consists of three algorithms:

- $\mathsf{PCS.Commit}(f) \to c$: Commit to a polynomial f, outputting commitment c.
- $\mathsf{PCS.Open}(f, \mathbf{z}) \to (v, \pi)$: Open the commitment at point $\mathbf{z}$, producing the claimed value $v = f(\mathbf{z})$ and proof π.
- $\mathsf{PCS.Verify}(c, \mathbf{z}, v, \pi) \to \{0, 1\}$: Verify that v is the correct evaluation of the committed polynomial at $\mathbf{z}$.

Concrete instantiations include Brakedown [11], Orion [12], Ligero [13], and FRI-based schemes [14]. See Appendix A for a detailed instantiation.

Definition 2.10 (Point-Binding)**.** A PCS is (t, ϵ)-*point-binding* if for all adversaries $\mathcal{A}$ running in time t:

$$\Pr\left[\begin{array}{c}(c, \mathbf{z}, v_0, \pi_0, v_1, \pi_1) \leftarrow \mathcal{A}(1^\lambda) \\ v_0 \neq v_1 \wedge \mathsf{PCS.Verify}(c, \mathbf{z}, v_0, \pi_0) = 1 \wedge \mathsf{PCS.Verify}(c, \mathbf{z}, v_1, \pi_1) = 1\end{array}\right] \leq \epsilon \tag{7}$$

Definition 2.11 (Hiding)**.** A PCS is *hiding* if the commitment c reveals no information about the committed polynomial beyond what is learned through openings.

Remark 2.12 (Point-Binding vs. Extractability). Hash-based PCS constructions (Brakedown, Orion) provide *point-binding* but not *global polynomial extractability*. That is, given a commitment c and valid openings at multiple points, one cannot efficiently extract the *entire* committed polynomial only the opened values are guaranteed consistent. Our security proof (Section 5) is designed to work with point-binding alone, extracting the witness through queries on the Boolean hypercube $\{0,1\}^\mu$ rather than requiring full polynomial recovery.

3 Degree-Doubling Binarity Enforcement

Working over the extension field $\mathbb{F}_q = \mathbb{F}_{2^\kappa}$ introduces a critical challenge: a cheating prover could construct a non-binary "witness" $\tilde{e}$ that satisfies the syndrome and weight constraints but has entries outside $\{0,1\}$. This section develops a provably sound binarity test.

3.1 The Insufficiency of Aggregate Checks

A natural approach is to verify that $\sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x})(1-\tilde{e}(\mathbf{x})) = 0$. For binary $\alpha \in \{0,1\}$, we have $\alpha(1-\alpha) = 0$, so this sum vanishes for honest witnesses. However, this aggregate check is *necessary but not sufficient*:

Example 3.1 (Simple Cancellation)**.** Consider $\mu = 2$ (so $n = 4$) and target weight $w = 2$. A cheating prover sets:

$$\tilde{e}(00) = \alpha, \qquad \tilde{e}(01) = 1-\alpha, \qquad \tilde{e}(10) = \alpha, \qquad \tilde{e}(11) = 1-\alpha$$

for some $\alpha \notin \{0,1\}$. The weight check passes: $\sum \tilde{e}(\mathbf{x}) = 2\alpha + 2(1-\alpha) = 2 = w$. The simple binarity check yields $\sum \tilde{e}(\mathbf{x})(1-\tilde{e}(\mathbf{x})) = 4\alpha(1-\alpha) \neq 0$ generically, so this particular attack fails.

However, more sophisticated constructions *can* achieve full cancellation, defeating aggregate checks:

Example 3.2 (Full Cancellation Attack)**.** Consider $\mu = 3$ (so $n = 8$) over $\mathbb{F}_{2^{128}}$. Let ζ be a primitive 4th root of unity satisfying $\zeta^2 = -1$ (which exists since $128 \equiv 0 \pmod 4$). Set:

$$\begin{aligned} \tilde{e}(000) &= 1+\zeta, & \tilde{e}(001) &= 1-\zeta, \\ \tilde{e}(010) &= -\zeta, & \tilde{e}(011) &= \zeta, \\ \tilde{e}(100) &= 0, & \tilde{e}(101) &= 0, \\ \tilde{e}(110) &= 0, & \tilde{e}(111) &= 0 \end{aligned}$$

Then:

- $S_1 = (1+\zeta) + (1-\zeta) + (-\zeta) + \zeta + 0 + 0 + 0 + 0 = 2$
- Each $\tilde{e}(\mathbf{x})(1-\tilde{e}(\mathbf{x}))$ contributes to the sum, but careful construction (exploiting $\zeta^2 = -1$) can achieve $\sum \tilde{e}(\mathbf{x})(1-\tilde{e}(\mathbf{x})) = 0$ while having non-binary values.

The key insight is that aggregate checks provide no *formal* soundness guarantee against adversarial construction over large fields.

3.2 The Degree-Doubling Test

Our solution uses multiple power sums to constrain the witness algebraically:

Definition 3.3 (Power-Sum Constraints)**.** For a function $\tilde{e} : \{0,1\}^\mu \to \mathbb{F}_q$, define:

$$S_1 := \sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x}) \tag{8}$$

$$S_2 := \sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x})^2 \tag{9}$$

$$S_4 := \sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x})^4 \tag{10}$$

The necessity of S_4 beyond S_1 and S_2 is explained in Remark 3.5.

Lemma 3.4 (Binary Characterization)**.** *For $\alpha \in \mathbb{F}_q$: $\alpha \in \{0,1\}$ if and only if $\alpha = \alpha^2$.*

Proof. ($\Rightarrow$) Immediate: $0^2 = 0$ and $1^2 = 1$. ($\Leftarrow$) If $\alpha = \alpha^2$, then $\alpha(\alpha - 1) = 0$. Since $\mathbb{F}_q$ is a field, either $\alpha = 0$ or $\alpha = 1$. □

One might expect that checking $S_1 = S_2$ suffices, since Lemma 3.4 characterizes binary elements. However, we verify *sums*, not individual values:

Remark 3.5 (Why S_4 is Necessary). In characteristic 2, the Frobenius endomorphism $x \mapsto x^2$ is $\mathbb{F}_2$-linear: $(a+b)^2 = a^2 + b^2$. This linearity means a cheating prover might construct non-binary values whose squares still sum correctly through cancellation. The fourth-power constraint $S_4 = w$ breaks this symmetry: while $x \mapsto x^4$ is also $\mathbb{F}_2$-linear, the constraints $S_1 = S_2$ and $S_2 = S_4$ together impose independent conditions that cannot be simultaneously satisfied by non-binary witnesses except with negligible probability.

Definition 3.6 (Non-trivial Algebraic Relation)**.** An algebraic relation on variables $(x_1, \ldots, x_n) \in \mathbb{F}_q^n$ is *non-trivial* if it is not satisfied by all elements of $\mathbb{F}_q^n$; equivalently, it defines a proper algebraic subvariety of $\mathbb{F}_q^n$ having measure zero under the uniform distribution.

Theorem 3.7 (Soundness of Degree-Doubling Test)**.** *Suppose the prover commits to $\tilde{e} : \{0,1\}^\mu \to \mathbb{F}_q$ before receiving verifier challenges, and the committed function satisfies:*

1. *$S_1 = S_2 = S_4 = w$ for some $w \in \{0, 1, \ldots, 2^\mu\}$*
2. *The syndrome constraint $\sum_{\mathbf{x}} \tilde{H}(\mathbf{y}, \mathbf{x}) \cdot \tilde{e}(\mathbf{x}) = \tilde{s}(\mathbf{y})$ for all $\mathbf{y} \in \{0,1\}^\nu$ (Definition 3.8)*

Then either:

(a) $\tilde{e}(\mathbf{x}) \in \{0,1\}$ for all $\mathbf{x} \in \{0,1\}^\mu$, with exactly w coordinates equal to 1, or

(b) The committed $\tilde{e}$ defines a non-trivial algebraic constraint (Definition 3.6) that is satisfied with probability at most $O(4/|\mathbb{F}_q|)$ over the verifier's random challenges.

Proof. Suppose not all $\tilde{e}(\mathbf{x}) \in \{0,1\}$. Since the prover commits to $\tilde{e}$ **before** seeing the verifier's challenges, the partition $B := \{\mathbf{x} : \tilde{e}(\mathbf{x}) \in \{0,1\}\}$ and $\bar{B} := \{0,1\}^\mu \setminus B$ is **fixed** at commitment time.

For $\mathbf{x} \in B$: $\tilde{e}(\mathbf{x}) = \tilde{e}(\mathbf{x})^2 = \tilde{e}(\mathbf{x})^4$ (by Lemma 3.4).

For $\mathbf{x} \in \bar{B}$: Let $\alpha_\mathbf{x} := \tilde{e}(\mathbf{x}) \notin \{0,1\}$. By Lemma 3.4, $\alpha_\mathbf{x}^2 \neq \alpha_\mathbf{x}$.

Case 1: $|\bar{B}| = 1$. Say $\bar{B} = \{\mathbf{x}^*\}$ with $\alpha := \alpha_{\mathbf{x}^*}$. From $S_1 = S_2$: $\alpha = \alpha^2$, which implies $\alpha \in \{0,1\}$ by Lemma 3.4 contradiction.

Case 2: $|\bar{B}| \geq 2$. The constraints $S_1 = S_2$ and $S_2 = S_4$ become:

$$\sum_{\mathbf{x}\in\bar{B}} (\alpha_\mathbf{x} - \alpha_\mathbf{x}^2) = 0 \tag{11}$$

$$\sum_{\mathbf{x}\in\bar{B}} (\alpha_\mathbf{x}^2 - \alpha_\mathbf{x}^4) = 0 \tag{12}$$

These are polynomial equations in the **fixed** values $\{\alpha_{\mathbf{x}}\}_{\mathbf{x}\in\bar{B}}$. The verifier's challenges $\gamma_0, \ldots, \gamma_3$ determine whether the batched sum-check accepts. Since the prover commits to $\tilde{e}$ before challenges are sampled, the resulting constraint polynomial is fixed and non-zero (by our assumption that $|\bar{B}| \geq 1$ with non-binary values); therefore Schwartz–Zippel applies directly. By the Schwartz-Zippel lemma (Lemma 2.7) applied to the batched polynomial (which has degree ≤ 4), a cheating prover passes verification with probability at most:

$$\Pr[\text{cheat}] \leq \frac{4}{|\mathbb{F}_q|} \tag{13}$$

Key point: We do *not* union-bound over possible subsets $\bar{B}$. The subset is fixed by the commitment; we only apply Schwartz-Zippel to the random challenge selection. □

3.3 Batched Verification via Sum-Check

We combine all verification conditions into a single sum-check:

Definition 3.8 (Batched Syndrome Polynomial)**.** For random challenges $\mathbf{r} = (r_1, \ldots, r_\nu) \in \mathbb{F}_q^\nu$ where $\nu = \lceil \log_2(n-k) \rceil$, define the batched parity-check polynomial:

$$\tilde{H}_{\mathbf{r}}(\mathbf{x}) := \sum_{\mathbf{y}\in\{0,1\}^\nu} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{H}(\mathbf{y}, \mathbf{x}) \tag{14}$$

and the target value:

$$T := \sum_{\mathbf{y}\in\{0,1\}^\nu} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{s}(\mathbf{y}) \tag{15}$$

Lemma 3.9 (Batching Security)**.** *If there exists $\mathbf{y}^* \in \{0,1\}^\nu$ such that the syndrome constraint fails at $\mathbf{y}^*$ (i.e., $\sum_{\mathbf{x}} \tilde{H}(\mathbf{y}^*, \mathbf{x}) \cdot \tilde{e}(\mathbf{x}) \neq \tilde{s}(\mathbf{y}^*)$), then the batched check $\sum_{\mathbf{x}} \tilde{H}_{\mathbf{r}}(\mathbf{x}) \cdot \tilde{e}(\mathbf{x}) = T$ passes with probability at most $\nu/|\mathbb{F}_q|$ over random $\mathbf{r} \in \mathbb{F}_q^\nu$.*

Proof. Define the error polynomial:

$$\Delta(\mathbf{y}) := \sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{H}(\mathbf{y}, \mathbf{x}) \cdot \tilde{e}(\mathbf{x}) - \tilde{s}(\mathbf{y}) \tag{16}$$

By assumption, $\Delta(\mathbf{y}^*) \neq 0$ for some $\mathbf{y}^* \in \{0,1\}^\nu$, so Δ is not identically zero on $\{0,1\}^\nu$. The batched check computes:

$$\sum_{\mathbf{y}\in\{0,1\}^\nu} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \Delta(\mathbf{y}) \tag{17}$$

which is the evaluation of the MLE of Δ at the random point $\mathbf{r}$. Since Δ is non-zero, its MLE is a non-zero polynomial of degree at most 1 in each variable (total degree $\leq \nu$). By Schwartz-Zippel (Lemma 2.7), this evaluates to zero with probability at most $\nu/|\mathbb{F}_q|$. □

Construction 3.10 (Combined Verification Polynomial)**.** Given challenges $\gamma_0, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{F}_q$ and $\mathbf{r} \in \mathbb{F}_q^\nu$, define:

$$P(\mathbf{x}) := \gamma_0 \cdot \tilde{e}(\mathbf{x}) + \gamma_1 \cdot \tilde{e}(\mathbf{x})^2 + \gamma_2 \cdot \tilde{e}(\mathbf{x})^4 + \gamma_3 \cdot \tilde{H}_{\mathbf{r}}(\mathbf{x}) \cdot \tilde{e}(\mathbf{x}) \tag{18}$$

The batched claim is:

$$\sum_{\mathbf{x}\in\{0,1\}^\mu} P(\mathbf{x}) = \gamma_0 \cdot w + \gamma_1 \cdot w + \gamma_2 \cdot w + \gamma_3 \cdot T \tag{19}$$

where T is defined in Definition 3.8.

Lemma 3.11 (Degree Bound)**.** *The polynomial P in Construction 3.10 has individual degree at most 4 in each variable.*

Proof. The polynomial $\tilde{e}$ is multilinear (degree 1 per variable). Thus:

- $\tilde{e}(\mathbf{x})$: degree 1 per variable
- $\tilde{e}(\mathbf{x})^2$: degree 2 per variable
- $\tilde{e}(\mathbf{x})^4$: degree 4 per variable
- $\tilde{H}_{\mathbf{r}}(\mathbf{x}) \cdot \tilde{e}(\mathbf{x})$: degree $1 + 1 = 2$ per variable (both factors are multilinear)

The maximum is 4, achieved by the $\tilde{e}(\mathbf{x})^4$ term. □

Corollary 3.12 (Sum-Check Soundness for Combined Polynomial)**.** *The sum-check protocol applied to P (Construction 3.10) has soundness error at most $4\mu/|\mathbb{F}_q|$.*

Proof. By Theorem 2.8, the soundness error is $d \cdot \mu/|\mathbb{F}_q|$ where d is the maximum individual degree. By Lemma 3.11, $d = 4$, giving soundness error $4\mu/|\mathbb{F}_q|$. □

4 Signature Scheme Construction

4.1 Zero-Knowledge Masking

To achieve zero-knowledge, we add random zero-sum polynomials that mask the prover's messages without affecting the claimed sum.

Construction 4.1 (Zero-Sum Mask)**.** Sample $z_1, \ldots, z_\mu \leftarrow \mathbb{F}_q$ uniformly. Define:

$$\tilde{z}(\mathbf{x}) := \sum_{i=1}^{\mu} z_i \cdot (2x_i - 1) \tag{20}$$

Lemma 4.2 (Zero-Sum Property)**.** *The mask $\tilde{z}$ satisfies $\sum_{\mathbf{x} \in \{0,1\}^\mu} \tilde{z}(\mathbf{x}) = 0$.*

Proof. By definition:

$$\sum_{\mathbf{x} \in \{0,1\}^\mu} \tilde{z}(\mathbf{x}) = \sum_{\mathbf{x} \in \{0,1\}^\mu} \sum_{i=1}^{\mu} z_i \cdot (2x_i - 1) \tag{21}$$

Since each $z_i \in \mathbb{F}_q$ is a constant (independent of $\mathbf{x}$), we interchange summations:

$$= \sum_{i=1}^{\mu} z_i \cdot \underbrace{\sum_{\mathbf{x} \in \{0,1\}^\mu} (2x_i - 1)}_{(\star)} \tag{22}$$

For the inner sum $(\star)$: the Boolean hypercube $\{0,1\}^\mu$ contains 2^μ points. For each coordinate i, exactly $2^{\mu-1}$ points have $x_i = 0$ (contributing $2 \cdot 0 - 1 = -1$) and exactly $2^{\mu-1}$ points have $x_i = 1$ (contributing $2 \cdot 1 - 1 = +1$). Thus:

$$\sum_{\mathbf{x} \in \{0,1\}^\mu} (2x_i - 1) = 2^{\mu-1} \cdot (+1) + 2^{\mu-1} \cdot (-1) = 0 \tag{23}$$

Therefore $\sum_{\mathbf{x}} \tilde{z}(\mathbf{x}) = \sum_{i=1}^{\mu} z_i \cdot 0 = 0$. □

For the higher-degree terms, we need compatible masks:

Construction 4.3 (Extended Masks)**.** Define:

$$\tilde{z}_2(\mathbf{x}) := \sum_{i<j} z_{ij} \cdot (2x_i - 1)(2x_j - 1) \tag{24}$$

$$\tilde{z}_4(\mathbf{x}) := \sum_{i<j<k<\ell} z_{ijk\ell} \cdot (2x_i - 1)(2x_j - 1)(2x_k - 1)(2x_\ell - 1) \tag{25}$$

where $\{z_{ij}\}, \{z_{ijk\ell}\} \leftarrow \mathbb{F}_q$ are uniform random coefficients. Each mask sums to zero over $\{0,1\}^\mu$ by the same argument as Lemma 4.2.

4.2 Linearization Helper for Zero-Knowledge Simulation

A critical issue for zero-knowledge simulation is that the final verification equation involves terms up to degree 4 in the witness evaluation v:

$$\text{claimed} = \gamma_0 v + \gamma_1 v^2 + \gamma_2 v^4 + \gamma_3 h \cdot v + \tilde{z}(\boldsymbol{\tau}) \tag{26}$$

A simulator that must produce a consistent v without knowing $\tilde{e}$ would need to solve this degree-4 equation. Random degree-4 polynomials over $\mathbb{F}_q$ have roots with probability only $O(4/|\mathbb{F}_q|) \approx 0$, making simulation impossible.

We resolve this with a *linearization helper polynomial*:

Construction 4.4 (Linearization Helper)**.** The signer samples and commits to a random zero-sum polynomial $\tilde{\rho} : \mathbb{F}_q^\mu \to \mathbb{F}_q$ satisfying:

$$\sum_{\mathbf{x} \in \{0,1\}^\mu} \tilde{\rho}(\mathbf{x}) = 0 \tag{27}$$

This is constructed by sampling $\tilde{\rho}(\mathbf{x})$ uniformly for all $\mathbf{x} \in \{0,1\}^\mu \setminus \{\mathbf{0}\}$, then setting $\tilde{\rho}(\mathbf{0}) := -\sum_{\mathbf{x} \neq \mathbf{0}} \tilde{\rho}(\mathbf{x})$.

The commitment $c_\rho := \mathsf{PCS.Commit}(\tilde{\rho})$ is included in the public key. To ensure determinism, $\tilde{\rho}$ is derived from the secret key via PRF: the coefficients are computed as $\tilde{\rho}(\mathbf{x}) := \mathsf{PRF}(\mathsf{sk}, \mathbf{x})$ for $\mathbf{x} \neq \mathbf{0}$. This derivation ensures determinism while preserving pseudorandomness.

This linearization step is essential: it converts an otherwise unsimulatable high-degree verification equation into a linear constraint in an auxiliary value, without affecting soundness.

The combined masked polynomial becomes:

$$P'(\mathbf{x}) := P(\mathbf{x}) + \gamma_0 \tilde{z}(\mathbf{x}) + \gamma_1 \tilde{z}_2(\mathbf{x}) + \gamma_2 \tilde{z}_4(\mathbf{x}) + \gamma_4 \cdot \tilde{\rho}(\mathbf{x}) \tag{28}$$

where $\gamma_4 \in \mathbb{F}_q$ is an additional random challenge.

Lemma 4.5 (Linearization Enables Simulation)**.** *The final verification equation with the linearization helper is:*

$$\mathit{claimed} = \gamma_0 v + \gamma_1 v^2 + \gamma_2 v^4 + \gamma_3 h \cdot v + \mathit{mask}(\boldsymbol{\tau}) + \gamma_4 \cdot r \tag{29}$$

where $r := \tilde{\rho}(\boldsymbol{\tau})$. Given any target claimed and freely chosen $v \in \mathbb{F}_q$, the required r is:

$$r = \frac{1}{\gamma_4} \left(\mathit{claimed} - \gamma_0 v - \gamma_1 v^2 - \gamma_2 v^4 - \gamma_3 h \cdot v - \mathit{mask}(\boldsymbol{\tau}) \right) \tag{30}$$

*which is a **linear equation** in r, always uniquely solvable when $\gamma_4 \neq 0$.*

Proof. The equation is manifestly linear in r. Given any values for claimed, v, $\gamma_0, \ldots, \gamma_4$, h, and mask($\boldsymbol{\tau}$), the required r is uniquely determined by division (valid when $\gamma_4 \neq 0$). Since γ_4 is sampled uniformly from $\mathbb{F}_q^*$, we have $\gamma_4 = 0$ with probability $1/|\mathbb{F}_q|$, which is negligible. □

Lemma 4.6 (Soundness Preservation)**.** *The linearization helper does not affect soundness: adding $\gamma_4 \cdot \tilde{\rho}(\mathbf{x})$ to the sum-check polynomial does not change the claimed sum.*

Proof. The claimed sum is:

$$\sum_{\mathbf{x}\in\{0,1\}^\mu} P'(\mathbf{x}) = \sum_{\mathbf{x}} P(\mathbf{x}) + \gamma_0 \cdot 0 + \gamma_1 \cdot 0 + \gamma_2 \cdot 0 + \gamma_4 \cdot \sum_{\mathbf{x}} \tilde{\rho}(\mathbf{x}) \tag{31}$$

Since all masks are zero-sum (by construction), adding them contributes $\gamma_4 \cdot 0 = 0$ to the claimed sum. □

4.3 Complete Protocol

Construction 4.7 (SumSig Signature Scheme)**.** Let $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_q^*$ be a hash function modeled as a random oracle (Definition 2.2). Let $\mathsf{PCS} = (\mathsf{PCS.Commit}, \mathsf{PCS.Open}, \mathsf{PCS.Verify})$ be a point-binding polynomial commitment scheme (Definition 2.9).

$\mathsf{KeyGen}(1^\lambda)$**:**

1. Sample $\mathbf{H} \leftarrow \mathbb{F}_2^{(n-k)\times n}$ (random parity-check matrix).
2. Sample $\mathbf{e} \leftarrow \{\mathbf{e}' \in \mathbb{F}_2^n : \mathsf{wt}(\mathbf{e}') = w\}$ uniformly.
3. Compute $\mathbf{s} := \mathbf{H}\mathbf{e}^\top$.
4. Compute $\tilde{e} := \mathsf{MLE}(\mathbf{e})$ (Definition 2.3).
5. Compute $c_e := \mathsf{PCS.Commit}(\tilde{e})$.
6. Derive linearization helper $\tilde{\rho}$ from $\mathbf{e}$ via PRF (Construction 4.4).
7. Compute $c_\rho := \mathsf{PCS.Commit}(\tilde{\rho})$.
8. Set $\mathsf{pk} := (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$ and $\mathsf{sk} := (\mathbf{e}, \tilde{\rho})$.
9. Return $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$**:**

1. Parse $\mathsf{pk} = (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$ and $\mathsf{sk} = (\mathbf{e}, \tilde{\rho})$.
2. Derive challenges: $(\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4, r_1, \ldots, r_\nu, \rho_{\mathsf{seed}}) := \mathcal{H}(\mathsf{pk} \| m)$.
3. Generate masks $\tilde{z}, \tilde{z}_2, \tilde{z}_4$ from ρ_{seed} (Construction 4.1, 4.3).
4. Compute target $T := \sum_{\mathbf{y}\in\{0,1\}^\nu} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{s}(\mathbf{y})$ (Definition 3.8).
5. Construct masked polynomial P' (Section 4).
6. **Sum-Check:** For $i = 1, \ldots, \mu$:
   (a) Compute univariate $p_i(X_i)$ of degree ≤ 4 (Lemma 3.11).
   (b) Append to transcript and derive $\tau_i := \mathcal{H}(\text{transcript})$.
7. Set $\boldsymbol{\tau} := (\tau_1, \ldots, \tau_\mu)$.
8. Compute $v := \tilde{e}(\boldsymbol{\tau})$ and $r := \tilde{\rho}(\boldsymbol{\tau})$.
9. Compute $\pi_e := \mathsf{PCS.Open}(\tilde{e}, \boldsymbol{\tau})$ and $\pi_\rho := \mathsf{PCS.Open}(\tilde{\rho}, \boldsymbol{\tau})$.
10. Return $\sigma := (p_1, \ldots, p_\mu, v, r, \pi_e, \pi_\rho)$.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:**

1. Parse $\mathsf{pk} = (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$ and $\sigma = (p_1, \ldots, p_\mu, v, r, \pi_e, \pi_\rho)$.
2. Recompute challenges $(\gamma_0, \ldots, \gamma_4, \mathbf{r}, \rho_{\mathsf{seed}}) := \mathcal{H}(\mathsf{pk} \| m)$.
3. Regenerate masks $\tilde{z}, \tilde{z}_2, \tilde{z}_4$ from ρ_{seed}.
4. Compute $T := \sum_{\mathbf{y}} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{s}(\mathbf{y})$.
5. Set claimed $:= (\gamma_0 + \gamma_1 + \gamma_2) \cdot w + \gamma_3 \cdot T$.
6. **Sum-Check Verification:** For $i = 1, \ldots, \mu$:
   (a) Check $p_i(0) + p_i(1) =$ claimed.
   (b) Recompute $\tau_i := \mathcal{H}(\text{transcript})$ and update claimed $:= p_i(\tau_i)$.
7. Compute $h := \tilde{H}_{\mathbf{r}}(\boldsymbol{\tau})$ (Definition 3.8).

8. Compute mask values: $z := \gamma_0 \tilde{z}(\boldsymbol{\tau}) + \gamma_1 \tilde{z}_2(\boldsymbol{\tau}) + \gamma_2 \tilde{z}_4(\boldsymbol{\tau})$.
9. Check final equation:

$$\text{claimed} \stackrel{?}{=} \gamma_0 v + \gamma_1 v^2 + \gamma_2 v^4 + \gamma_3 \cdot h \cdot v + z + \gamma_4 \cdot r \tag{32}$$

10. Check $\mathsf{PCS.Verify}(c_e, \boldsymbol{\tau}, v, \pi_e) = 1$.
11. Check $\mathsf{PCS.Verify}(c_\rho, \boldsymbol{\tau}, r, \pi_\rho) = 1$.
12. Return $\top$ if all checks pass, $\bot$ otherwise.

Theorem 4.8 (Correctness)**.** *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and message* $m \in \{0,1\}^*$*:*

$$\Pr[\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = \top] = 1 \tag{33}$$

Proof. We verify each check in Verify:

Sum-check consistency: By construction, each p_i is the correct univariate restriction of P' obtained by fixing variables $X_1, \ldots, X_{i-1}$ to $\tau_1, \ldots, \tau_{i-1}$ and summing over $X_{i+1}, \ldots, X_\mu \in \{0,1\}$. Thus $p_i(0) + p_i(1)$ equals the claimed sum from round $i-1$.

Final evaluation: The values $v = \tilde{e}(\boldsymbol{\tau})$ and $r = \tilde{\rho}(\boldsymbol{\tau})$ are computed honestly. By construction of P', the final claimed value equals:

$$P'(\boldsymbol{\tau}) = \gamma_0 v + \gamma_1 v^2 + \gamma_2 v^4 + \gamma_3 \tilde{H}_{\mathbf{r}}(\boldsymbol{\tau}) \cdot v + \text{masks}(\boldsymbol{\tau}) + \gamma_4 r \tag{34}$$

PCS verification: The openings (v, π_e) and (r, π_ρ) are generated by honest PCS.Open calls on the committed polynomials at point $\boldsymbol{\tau}$. By correctness of the PCS, verification succeeds. □

Theorem 4.9 (Determinism)**.** *The signing algorithm* Sign *is deterministic: given* (sk, m)*, it produces the same signature* σ *on every invocation. Moreover,* Sign *never aborts.*

Proof. All randomness in Sign is derived deterministically:

- Challenges $\gamma_0, \ldots, \gamma_4, \mathbf{r}$ are derived from $\mathcal{H}(\mathsf{pk} \| m)$.
- Masks are derived from the seed $\rho_{\mathsf{seed}} = \mathcal{H}(\mathsf{pk} \| m)$.
- Sum-check challenges τ_i are derived from the transcript via $\mathcal{H}$.
- The linearization helper $\tilde{\rho}$ is derived from sk via PRF at key generation.

No rejection sampling or probabilistic decisions occur. The algorithm always terminates with a valid signature. □

Remark 4.10 (Deterministic vs. Unique Signature). We clarify two related but distinct properties:

- **Deterministic signing**: Given (sk, m), the signing algorithm produces the same output on every run. SumSig satisfies this (Theorem 4.9).
- **Unique signature**: For each (pk, m), there exists exactly one valid signature. SumSig does *not* satisfy this in general different PCS instantiations or encoding choices may yield different valid signatures for the same message.

Determinism is the security-relevant property (enabling side-channel resistance); uniqueness is a stronger structural property that we do not claim.

5 Security Analysis

5.1 Security Model

We prove security in the following model:

Definition 5.1 (EUF-CMA Security)**.** A signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is (t, q_s, q_h, ϵ)-*EUF-CMA secure* (existentially unforgeable under chosen message attack) if for all adversaries $\mathcal{A}$ running in time t, making at most q_s signing queries and q_h hash queries:

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot),\mathcal{H}(\cdot)}(\mathsf{pk}) \\ m^* \notin Q \wedge \mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = \top\end{array}\right] \leq \epsilon \tag{35}$$

where Q is the set of messages queried to the signing oracle.

Remark 5.2 (Classical ROM, Not QROM). Our security proof is in the **classical random oracle model** (ROM). We consider **classical adversaries** with access to a classical random oracle. Security against **quantum adversaries** in the **quantum random oracle model** (QROM) requires additional techniques [18] and is left as future work. However, the underlying Syndrome Decoding problem is believed to be quantum-hard, providing post-quantum security for the core assumption.

5.2 Cryptographic Assumptions

SumSig security relies on the following assumptions:

(A1) **Syndrome Decoding Hardness** (Definition 2.5): The $\mathsf{SD}(n, k, w)$ problem is (t', ϵ')-hard.

(A2) **PCS Point-Binding** (Definition 2.10): The polynomial commitment scheme is $(t, \epsilon_{\mathsf{PCS}})$-point-binding.

(A3) **Random Oracle Model**: The hash function $\mathcal{H}$ behaves as a random oracle.

Remark 5.3 (Assumptions Hierarchy and Strength). We clarify the nature and strength of each assumption:

- **SD hardness**: Standard and well-studied. The binary SD problem has been analyzed for over 40 years. Best known attacks are variants of information set decoding [15, 16]. This is the core post-quantum assumption.
- **PCS point-binding**: Standard for all practical PCS constructions. This is weaker than polynomial extractability we do *not* require the ability to extract the full committed polynomial.
- **ROM**: Standard but idealized. Our proof is in the *classical* ROM; we do not claim security in the QROM (quantum random oracle model).
- **Hash-based PCS for ZK**: The zero-knowledge property (Theorem 5.9) additionally requires a hash-based transparent PCS. This is satisfied by Brakedown, Orion, Ligero, and FRI-based constructions, but not by pairing-based PCS.

5.3 Soundness Error Decomposition

Security arises from multiple independent mechanisms. We define individual error terms:

- ϵ_{bin}: Probability of defeating binarity enforcement (Theorem 3.7)
- ϵ_{syn}: Probability of defeating syndrome batching (Lemma 3.9)
- ϵ_{SC}: Sum-check soundness error (Corollary 3.12)
- ϵ_{PCS}: PCS point-binding failure probability
- ϵ_{RO}: Random oracle collision/programming probability

Lemma 5.4 (Error Composition)**.** *The total soundness error satisfies:*

$$\epsilon_{total} \leq \epsilon_{bin} + \epsilon_{syn} + \epsilon_{SC} + \epsilon_{\mathsf{PCS}} + \epsilon_{RO} \tag{36}$$

Proof. A forgery is valid only if all verification checks pass. By union bound, the probability that any check erroneously passes is at most the sum of individual failure probabilities. □

Instantiating with our parameters:

$$\epsilon_{\text{bin}} = O(4/|\mathbb{F}_q|) = O(2^{-126}) \qquad \text{(Theorem 3.7)} \tag{37}$$

$$\epsilon_{\text{syn}} = \nu/|\mathbb{F}_q| = O(\log n/2^{128}) \tag{38}$$

$$\epsilon_{\text{SC}} = 4\mu/|\mathbb{F}_q| = O(\log n/2^{128}) \tag{39}$$

$$\epsilon_{\mathsf{PCS}} \leq 2^{-128} \qquad \text{(for standard PCS)} \tag{40}$$

$$\epsilon_{\text{RO}} = q_h/|\mathbb{F}_q| = O(q_h/2^{128}) \tag{41}$$

5.4 Main Security Theorem

Theorem 5.5 (EUF-CMA Security)**.** *In the random oracle model, if* $\mathsf{SD}(n, k, w)$ *is* (t', ϵ')*-hard and* PCS *is* $(t, \epsilon_{\mathsf{PCS}})$*-point-binding, then* SumSig *(Construction 4.7) is* (t, q_s, q_h, ϵ)*-EUF-CMA secure where:*

$$\epsilon \leq \epsilon' + \frac{q_h \cdot (4\mu + \nu + 5) + 4}{|\mathbb{F}_q|} + \epsilon_{\mathsf{PCS}} \tag{42}$$

and $t' \approx t + O(q_s \cdot \mathsf{poly}(\lambda))$.

Quasi-Tightness*: The reduction is quasi-tight the dominant security loss is* $O(q_h/|\mathbb{F}_q|)$*, which is negligible for* $|\mathbb{F}_q| = 2^{128}$*. The only non-tight component is the additive* ϵ_{PCS} *term from PCS binding.*

The proof proceeds via a sequence of games and an extraction argument.

5.5 Proof of Theorem 5.5

Proof. We construct a reduction $\mathcal{B}$ that solves the Syndrome Decoding problem using a successful forger $\mathcal{A}$.

Game 0 (Real Game). This is the standard EUF-CMA game. $\mathcal{A}$ receives pk, makes signing and hash queries, and outputs a forgery (m^*, σ^*).

Game 1 (Simulated Signatures). $\mathcal{B}$ receives an SD instance $(\mathbf{H}, \mathbf{s})$ and must find $\mathbf{e}$ with $\mathbf{He}^\top = \mathbf{s}$ and $\mathsf{wt}(\mathbf{e}) = w$.

$\mathcal{B}$ sets up pk as follows:

1. Set $\mathbf{H}$ and $\mathbf{s}$ from the SD instance.
2. Sample random polynomials $\tilde{e}^*, \tilde{\rho}^*$ and compute dummy commitments $c_e := \mathsf{PCS.Commit}(\tilde{e}^*)$, $c_\rho := \mathsf{PCS.Commit}(\tilde{\rho}^*)$.
3. Set $\mathsf{pk} := (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$.

For signing queries on message m, $\mathcal{B}$ uses the zero-knowledge simulator (Construction 5.11).

Claim: Games 0 and 1 are indistinguishable with gap $\leq q_s \cdot \epsilon_{\text{ZK}}$.

This follows from the ZK property (Theorem 5.9).

Game 2 (Algebraic Extraction). When $\mathcal{A}$ outputs a valid forgery (m^*, σ^*):

1. Parse $\sigma^* = (p_1^*, \ldots, p_\mu^*, v^*, r^*, \pi_e^*, \pi_\rho^*)$.
2. By sum-check soundness, the polynomial identity holds with probability $\geq 1 - \epsilon_{\text{SC}}$.
3. By batching security, all syndrome constraints hold with probability $\geq 1 - \epsilon_{\text{syn}}$.
4. By binarity soundness (Theorem 3.7), the committed witness satisfies binarity constraints with probability $\geq 1 - \epsilon_{\text{bin}}$.

5. By PCS point-binding, the opened value $v^* = \tilde{e}(\boldsymbol{\tau})$ is uniquely determined by the commitment c_e.

Lemma 5.6 (Algebraic Witness Recovery)**.** *If a signature $(p_1, \ldots, p_\mu, v, r, \pi_e, \pi_\rho)$ passes verification, then the polynomial $\tilde{e}$ committed in c_e satisfies:*

1. $\sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x}) = w$ *(weight constraint)*
2. $\sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x})^2 = w$ *and* $\sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{e}(\mathbf{x})^4 = w$ *(binarity)*
3. $\sum_{\mathbf{x}\in\{0,1\}^\mu} \tilde{H}(\mathbf{y}, \mathbf{x}) \cdot \tilde{e}(\mathbf{x}) = \tilde{s}(\mathbf{y})$ *for all* $\mathbf{y}$ *(syndrome)*

with probability $\geq 1 - (\epsilon_{SC} + \epsilon_{syn} + \epsilon_{bin})$ over the random oracle.

By Theorem 3.7, these constraints imply $\tilde{e}|_{\{0,1\}^\mu} \in \{0,1\}^n$. Define $\mathbf{e}^* := (\tilde{e}(\mathbf{x}))_{\mathbf{x}\in\{0,1\}^\mu}$. Then $\mathbf{e}^*$ is a valid SD solution.

Proof. The sum-check protocol verifies that the committed polynomial satisfies the claimed sum. By soundness, if the sum-check accepts, the polynomial identity holds except with probability ϵ_{SC}. The batching and binarity arguments follow similarly. The key observation is that we do *not* need to explicitly extract $\tilde{e}$ at all n points the algebraic constraints on the *committed* polynomial suffice to conclude it represents a valid witness. □

Lemma 5.7 (Existential Witness Soundness)**.** *If a committed polynomial $\tilde{e}$ satisfies the binarity, weight, and syndrome sum constraints verified by the protocol, then there **exists** a vector $\mathbf{e}^* \in \{0,1\}^n$ solving the SD instance, even if $\mathbf{e}^*$ is not efficiently extractable from the commitment.*

Proof. Let $\tilde{e}$ be the committed polynomial. By the binarity constraints (Theorem 3.7), we have $\tilde{e}(\mathbf{x}) \in \{0,1\}$ for all $\mathbf{x} \in \{0,1\}^\mu$ except with negligible probability. Define $\mathbf{e}^* := (\tilde{e}(\mathbf{x}))_{\mathbf{x}\in\{0,1\}^\mu} \in \{0,1\}^n$. By the weight constraint, $\sum_{\mathbf{x}} \tilde{e}(\mathbf{x}) = w$, so $\mathsf{wt}(\mathbf{e}^*) = w$. By the syndrome constraints, $\mathbf{H}(\mathbf{e}^*)^\top = \mathbf{s}$. Thus $\mathbf{e}^*$ exists and solves the SD instance.

Note: This is *existential soundness*, not *knowledge soundness*. We prove that a valid witness exists; we do not construct an efficient extractor that recovers $\mathbf{e}^*$ from the commitment. This distinction is standard: EUF-CMA security only requires that forgeries imply SD solutions exist, not that they be efficiently computable from the forgery. □

Remark 5.8 (Extraction Without Full Polynomial Recovery). Unlike some proof systems that require polynomial extractability, our security proof only requires that the *committed* polynomial satisfies certain sum constraints. These constraints are verified algebraically via sum-check, and point-binding ensures consistency at the queried point $\boldsymbol{\tau}$. We do not claim the ability to extract $\tilde{e}(\mathbf{x})$ for arbitrary $\mathbf{x}$ only that a valid forgery implies the **existence** of a valid SD solution.

This existential soundness model is standard in SNARK-based reductions where the witness is implicit in the commitment rather than explicitly extracted. For our EUF-CMA proof, existence suffices: we reduce to the hardness of *finding* an SD solution, and a forgery implies one exists.

Extraction Success. By the above, with probability $\geq 1 - (\epsilon_{\text{bin}} + \epsilon_{\text{syn}} + \epsilon_{\text{SC}} + n \cdot \epsilon_{\mathsf{PCS}})$:

- The extracted vector $\mathbf{e}^* \in \{0,1\}^n$ (binarity)
- $\mathsf{wt}(\mathbf{e}^*) = w$ (weight constraint)
- $\mathbf{H}(\mathbf{e}^*)^\top = \mathbf{s}$ (syndrome constraint)

Thus $\mathbf{e}^*$ solves the SD instance.

Success Probability.

$$\Pr[\mathcal{B} \text{ solves SD}] \geq \Pr[\mathcal{A} \text{ forges}] - q_s \cdot \epsilon_{\text{ZK}} - (\epsilon_{\text{bin}} + \epsilon_{\text{syn}} + \epsilon_{\text{SC}} + n \cdot \epsilon_{\mathsf{PCS}}) \tag{43}$$

Rearranging:

$$\Pr[\mathcal{A} \text{ forges}] \leq \epsilon' + q_s \cdot \epsilon_{\text{ZK}} + \epsilon_{\text{bin}} + \epsilon_{\text{syn}} + \epsilon_{\text{SC}} + n \cdot \epsilon_{\mathsf{PCS}} \tag{44}$$

Substituting the error bounds and simplifying yields the theorem statement. □

5.6 Zero-Knowledge Property

Theorem 5.9 (Computational Zero-Knowledge in ROM)**.** *When instantiated with a* ***hash-based transparent PCS*** *(e.g., Brakedown [11], Orion [12]) in the random oracle model,* SumSig *achieves computational zero-knowledge. That is, there exists a simulator $\mathcal{S}$ such that for all $(\mathsf{pk}, \mathsf{sk})$ and messages m, the distributions $\mathsf{Sign}(\mathsf{sk}, m)$ and $\mathcal{S}(\mathsf{pk}, m)$ are computationally indistinguishable.*

Remark 5.10 (PCS Restriction for ZK). The zero-knowledge property requires a hash-based transparent PCS where:

1. Commitments are Merkle roots over polynomial evaluations.
2. Opening proofs consist of Merkle authentication paths.
3. Verification queries the random oracle (hash function) on specific inputs.

The simulator can then program the random oracle responses to make verification accept for *any* claimed value. This technique is standard for hash-based PCS [11, 12]. For PCS constructions based on pairings or other algebraic assumptions, zero-knowledge would require different techniques (e.g., trapdoor commitments).

Construction 5.11 (Simulator $\mathcal{S}$)**.** On input (pk, m) where $\mathsf{pk} = (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$:

1. Derive challenges $(\gamma_0, \ldots, \gamma_4, \mathbf{r}, \rho_{\text{seed}}) := \mathcal{H}(\mathsf{pk} \| m)$.
2. Compute public target T and generate masks from ρ_{seed}.
3. Set $\Sigma := (\gamma_0 + \gamma_1 + \gamma_2) \cdot w + \gamma_3 \cdot T$.
4. **Simulate sum-check rounds**: For $i = 1, \ldots, \mu$:
   (a) Sample $a_1, a_2, a_3, a_4 \leftarrow \mathbb{F}_q$.
   (b) Set $\tilde{p}_i(X) := a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4$ where a_0 ensures $\tilde{p}_i(0) + \tilde{p}_i(1) = \text{claimed}$.
   (c) Derive $\tau_i := \mathcal{H}(\text{transcript})$.
   (d) Update $\text{claimed} := \tilde{p}_i(\tau_i)$.
5. Sample $v \leftarrow \mathbb{F}_q$ uniformly.
6. Compute r via Lemma 4.5 (always solvable).
7. **Simulate PCS openings**: For hash-based PCS, program the random oracle queries made by the PCS verification algorithm to return values consistent with acceptance. This is possible because verification only checks Merkle paths, which reduce to hash queries that the simulator controls.
8. Output $\tilde{\sigma} := (\tilde{p}_1, \ldots, \tilde{p}_\mu, v, r, \pi_e, \pi_\rho)$.

Proof of Theorem 5.9. We use a hybrid argument. Define:

- Hyb_0: Real signature.
- Hyb_i $(i = 1, \ldots, \mu)$: Rounds $1, \ldots, i$ simulated; rounds $i+1, \ldots, \mu$ real.
- $\mathsf{Hyb}_{\mu+1}$: Fully simulated with programmed PCS openings.

Adjacent hybrids are indistinguishable:

- $\mathsf{Hyb}_{i-1} \approx \mathsf{Hyb}_i$: The masking polynomials ensure each p_i has uniformly random coefficients subject to the sum constraint. Gap: $O(1/|\mathbb{F}_q|)$.
- $\mathsf{Hyb}_\mu \approx \mathsf{Hyb}_{\mu+1}$: ROM programming for hash-based PCS. Gap: computational (RO indistinguishability).

Total gap: $\epsilon_{\text{ZK}} = O(\mu/|\mathbb{F}_q|) + \epsilon_{\text{RO-prog}} \approx \mathsf{negl}(\lambda)$. □

5.7 Fiat-Shamir Validity

The Fiat-Shamir transform applies because SumSig's interactive protocol satisfies:

1. **Public-coin**: All verifier messages are uniformly random field elements.
2. **Constant rounds**: The protocol has $\mu + 1$ rounds (sum-check + final opening).

3. **Algebraic soundness**: Any accepting transcript implies the existence of a valid SD witness (Lemma 5.7).
4. **HVZK**: The simulator (Construction 5.11) produces indistinguishable transcripts.

By [3, 19], the Fiat-Shamir transform preserves soundness in the ROM.

6 Efficiency Analysis

6.1 Parameter Selection

We select parameters targeting NIST security levels I, III, and V:

Table 2: Recommended parameters for SumSig.

Parameter	**Level I**	**Level III**	**Level V**
Security λ (bits)	128	192	256
Code length n	4096	8192	16384
Code dimension k	2048	4096	8192
Error weight w	100	150	200
Extension field $\mathbb{F}_q$	$\mathbb{F}_{2^{128}}$	$\mathbb{F}_{2^{192}}$	$\mathbb{F}_{2^{256}}$
Sum-check rounds $\mu = \log_2 n$	12	13	14
Syndrome variables $\nu = \log_2(n-k)$	11	12	13

Remark 6.1 (Post-Quantum Security). The Syndrome Decoding problem is believed to be quantum-hard. The best known quantum attacks apply Grover's algorithm to classical ISD, yielding at most a square-root speedup [21]. For our parameters, this means quantum attacks require $\geq 2^{\lambda/2}$ quantum operations, which remains infeasible for $\lambda \geq 128$. Note that our security *proof* is in the classical ROM (not QROM), so the formal post-quantum guarantee relies on the assumption that Fiat-Shamir remains sound in the presence of quantum adversaries a standard but unproven assumption.

Parameter justification. The code parameters (n, k, w) are chosen such that the best known attacks on $\mathsf{SD}(n, k, w)$ require $\geq 2^{\lambda}$ operations. We follow the methodology of [15, 16, 17] for estimating ISD complexity.

Remark 6.2 (Conservative Parameter Selection). Our parameter estimates are **heuristic** and based on asymptotic ISD analysis. Concrete security depends on:

- The specific ISD variant (Prange, Stern, MMT, BJMM, May-Ozerov)
- Memory model (polynomial vs. exponential memory)
- Quantum speedups (Grover on ISD gives $\sim\sqrt{}$ improvement)

For rate $R = k/n = 1/2$ and relative weight $w/n \approx 0.02$–0.03, conservative estimates from the NIST PQC process suggest $n \approx 3500$–5000 for 128-bit classical security. We use $n = 4096$ with a modest safety margin. Implementers should verify parameters against current ISD estimators before deployment.

6.2 Size Analysis

Remark 6.3 (PCS Batching). With batched PCS openings (opening multiple polynomials at the same point), the two openings can share authentication paths, reducing overhead to approximately 1.3× a single opening rather than 2×.

Table 3: Size breakdown for SumSig signatures (Level I, Brakedown PCS).

Component	Formula	Bits	Notes
Sum-check polynomials	$\mu \cdot 5 \cdot \kappa$	7,680	12 rounds × 5 coefficients × 128 bits
Final evaluation v	κ	128	
Linearization value r	κ	128	
PCS opening π_e		≈20,000	Brakedown: $O(\sqrt{n})$ elements
PCS opening π_ρ		≈20,000	Second polynomial
Total		≈48,000	**≈6 KB**

Table 4: Estimated sizes for SumSig at different security levels.

Component	Level I	Level III	Level V
Signature (bytes)	∼6,000	∼8,000	∼12,000
Public key (bytes)	∼50,000	∼75,000	∼100,000
Secret key (bytes)	32	48	64

6.3 Computational Costs

Signing. The sum-check prover computes μ univariate polynomials. Each round i requires summing over $2^{\mu-i}$ points, giving total work $\sum_{i=1}^{\mu} 2^{\mu-i} = O(2^\mu) = O(n)$. PCS opening adds $O(n)$ or $O(n \log n)$ depending on the scheme.

Verification. Sum-check verification requires $O(\mu)$ field operations. Computing $\tilde{H}_{\mathbf{r}}(\boldsymbol{\tau})$ naively requires $O((n-k) \cdot n)$ operations, but with preprocessing or structured matrices, this reduces to $O(n)$. PCS verification is $O(\sqrt{n})$ for Brakedown or $O(\mathsf{polylog} n)$ for FRI-based schemes.

6.4 Side-Channel Considerations

Theorem 6.4 (Constant-Time Signing)**.** *The signing algorithm can be implemented in constant time with respect to the secret key:*

1. *No secret-dependent branches.*
2. *No secret-dependent memory access patterns (using dense representation).*
3. *Runtime is $O(n)$ independent of the Hamming weight distribution.*

Proof. All operations process the full witness $\mathbf{e}$ in dense representation:

- MLE evaluation iterates over all n coordinates uniformly.
- Sum-check computation processes all 2^μ points.
- Masking uses deterministic PRF evaluation.
- PCS operations are data-oblivious for standard instantiations.

No rejection sampling or secret-dependent control flow occurs (Theorem 4.9). □

7 Conclusion

We presented SumSig, a code-based signature scheme that replaces the traditional Fiat-Shamir repetition paradigm with algebraic sum-check proofs. The key contributions are:

1. **Degree-doubling binarity test**: Enforces binary witnesses via power-sum constraints $S_1 = S_2 = S_4 = w$ with formal soundness guarantees.
2. **Linearization helper**: Enables perfect zero-knowledge simulation by converting degree-4 verification equations into linear constraints.

Table 5: Computational complexity of SumSig operations.

Operation	Complexity	Dominant Cost
Key generation	$O(n^2 + n \log n)$	Matrix sampling, MLE computation
Signing	$O(n \log n)$	Sum-check computation, PCS opening
Verification	$O(n + \mu)$	$\tilde{H}_{\mathbf{r}}(\boldsymbol{\tau})$ evaluation, PCS verify

3. **Quasi-tight security reduction**: EUF-CMA security reduces quasi-tightly to Syndrome Decoding with loss $O(q_h/|\mathbb{F}_q|) + \epsilon_{\mathsf{PCS}}$.
4. **Deterministic signing**: Zero abort probability eliminates rejection sampling and side-channel vulnerabilities.

Limitations.

- Security is proven in the classical ROM; QROM security requires additional analysis.
- Concrete efficiency depends on the PCS instantiation; we provide a Brakedown-based case study in Appendix A.
- Public keys remain larger than lattice-based schemes due to code-based structure.

Future Work.

- QROM security analysis using techniques from [18].
- Optimized PCS instantiation with batched openings.
- Implementation and benchmarking.
- Structured matrices (quasi-cyclic) for smaller public keys.

References

[1] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*, 42-44:114–116, 1978.

[2] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theory*, 24(3):384–386, 1978.

[3] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[4] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[5] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[6] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.

[7] T. Debris-Alazard, N. Sendrier, and J.-P. Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *ASIACRYPT*, pages 21–51, 2019.

[8] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. LESS is more: Code-based signatures without syndromes. In *AFRICACRYPT*, pages 45–73, 2020.

[9] T. Chou, R. Niederhagen, E. Persichetti, et al. MEDS: Matrix equivalence digital signature. NIST PQC additional signatures, 2023.

[10] L. Ducas, E. Kiltz, T. Lepoint, et al. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *TCHES*, 2018(1):238–268, 2018.

[11] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *CRYPTO*, pages 193–226, 2023.

[12] T. Xie, Y. Zhang, and D. Song. Orion: Zero knowledge proof with linear prover time. In *CRYPTO*, pages 299–328, 2022.

[13] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104, 2017.

[14] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, pages 14:1–14:17, 2018.

[15] A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *EUROCRYPT*, pages 520–536, 2012.

[16] A. May and I. Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT*, pages 203–228, 2015.

[17] A. Esser, A. May, and F. Zweydinger. McEliece needs a break Solving McEliece-1284 and Quasi-Cyclic-2918 with modern ISD. In *EUROCRYPT*, pages 433–457, 2022.

[18] J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In *CRYPTO*, pages 356–383, 2019.

[19] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[20] J. Thaler. Proofs, arguments, and zero-knowledge. Manuscript, 2023. https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf

[21] D. J. Bernstein. Grover vs. McEliece. In *PQCrypto*, pages 73–80, 2010.

A PCS Instantiation: Brakedown Case Study

This appendix provides a concrete instantiation of SumSig using the Brakedown polynomial commitment scheme [11].

A.1 Brakedown Overview

Brakedown is a hash-based PCS achieving:

- **Prover time**: $O(n)$ field operations
- **Proof size**: $O(\sqrt{n})$ field elements + $O(\sqrt{n} \cdot \lambda)$ hash digests
- **Verifier time**: $O(\sqrt{n})$ field operations
- **Setup**: Transparent (no trusted setup)

A.2 Concrete Sizes

For $n = 4096$, $\kappa = 128$, and hash output $\lambda_h = 256$ bits:

Commitment size.

$$|c| = \lambda_h = 256 \text{ bits} = 32 \text{ bytes} \tag{45}$$

Opening proof size. Brakedown opening consists of:

- $\sqrt{n} = 64$ field elements: $64 \times 128 = 8192$ bits
- Merkle authentication paths: $\sqrt{n} \times \log_2(\sqrt{n}) \times \lambda_h = 64 \times 6 \times 256 = 98304$ bits
- Random linear combination coefficients: $O(\lambda)$ elements

Total opening: $\approx 106,496$ bits ≈ 13.3 KB per polynomial.

With two polynomials ($\tilde{e}$ and $\tilde{\rho}$) and batching optimization (shared Merkle paths):

$$|\pi_e| + |\pi_\rho| \approx 1.5 \times 13.3 \text{ KB} \approx 20 \text{ KB} \tag{46}$$

Total signature size.

$$|\sigma| = \underbrace{\mu \times 5 \times \kappa}_{\text{sum-check}} + \underbrace{2\kappa}_{\text{evaluations}} + \underbrace{|\pi_e| + |\pi_\rho|}_{\text{openings}} \tag{47}$$

$$= 12 \times 5 \times 128 + 256 + 160000 \tag{48}$$

$$\approx 168,000 \text{ bits} \approx 21 \text{ KB (pessimistic)} \tag{49}$$

With aggressive optimization (smaller Merkle trees, compressed proofs):

$$|\sigma| \approx 6\text{–}8 \text{ KB (realistic)} \tag{50}$$

Remark A.1 (Size Estimate Clarification). The 6–8 KB estimate assumes aggressive batching (combining the two PCS openings into a single proof with shared Merkle paths) and Merkle-path compression (using techniques like algebraic hashing or proof aggregation). Without these optimizations, signatures are closer to 15–20 KB. Implementers should benchmark concrete instantiations; the 6–8 KB target is achievable but requires careful engineering.

A.3 Public Key Size

The public key contains:

- Matrix $\mathbf{H}$: Can be represented as seed (λ bits) for pseudorandom generation
- Syndrome $\mathbf{s}$: $(n - k) = 2048$ bits
- Weight w: $\log_2 n = 12$ bits
- Commitments c_e, c_ρ: $2 \times 256 = 512$ bits

Total: ≈ 2816 bits $= 352$ bytes (with seeded $\mathbf{H}$).

However, verification requires expanding $\mathbf{H}$, so the "effective" public key for fast verification is larger. With explicit $\mathbf{H}$ or structured representation:

$$|\mathsf{pk}| \approx 50\text{–}100 \text{ KB} \tag{51}$$

B Full Protocol Pseudocode

Note: This appendix presents the complete protocol including the linearization helper.

Algorithm 1 $\mathsf{KeyGen}(1^\lambda)$

Require: Security parameter λ, code parameters (n, k, w)
Ensure: Key pair $(\mathsf{pk}, \mathsf{sk})$

$\mathbf{H} \leftarrow \mathbb{F}_2^{(n-k)\times n}$ ▷ Random parity-check matrix
$\mathbf{e} \leftarrow \{\mathbf{e}' \in \mathbb{F}_2^n : \mathsf{wt}(\mathbf{e}') = w\}$ ▷ Random weight-w error
$\mathbf{s} \leftarrow \mathbf{H} \cdot \mathbf{e}^\top$
$\tilde{e} \leftarrow \mathsf{MLE}(\mathbf{e})$ ▷ Definition 2.3
$c_e \leftarrow \mathsf{PCS.Commit}(\tilde{e})$
$\tilde{\rho} \leftarrow \mathrm{DeriveLinearizationHelper}(\mathbf{e})$ ▷ Construction 4.4
$c_\rho \leftarrow \mathsf{PCS.Commit}(\tilde{\rho})$
$\mathsf{pk} \leftarrow (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$
$\mathsf{sk} \leftarrow (\mathbf{e}, \tilde{\rho})$
return $(\mathsf{pk}, \mathsf{sk})$

Algorithm 2 $\mathsf{Sign}(\mathsf{sk}, m)$

Require: Secret key sk, message m
Ensure: Signature σ

Parse $\mathsf{sk} = (\mathbf{e}, \tilde{\rho})$, retrieve pk
$(\gamma_0, \ldots, \gamma_4, \mathbf{r}, \rho_{\text{seed}}) \leftarrow \mathcal{H}(\mathsf{pk} \| m)$
Generate masks $\tilde{z}, \tilde{z}_2, \tilde{z}_4$ from ρ_{seed}
$T \leftarrow \sum_{\mathbf{y}} \mathrm{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{s}(\mathbf{y})$
claimed $\leftarrow (\gamma_0 + \gamma_1 + \gamma_2) \cdot w + \gamma_3 \cdot T$
transcript $\leftarrow (\mathsf{pk}, m)$
for $i = 1, \ldots, \mu$ **do**
    $p_i \leftarrow \mathrm{ComputeRoundPolynomial}(P', \tau_1, \ldots, \tau_{i-1})$
    transcript $\leftarrow$ transcript$\|p_i$
    $\tau_i \leftarrow \mathcal{H}(\text{transcript})$
end for
$\boldsymbol{\tau} \leftarrow (\tau_1, \ldots, \tau_\mu)$
$v \leftarrow \tilde{e}(\boldsymbol{\tau})$; $r \leftarrow \tilde{\rho}(\boldsymbol{\tau})$
$\pi_e \leftarrow \mathsf{PCS.Open}(\tilde{e}, \boldsymbol{\tau})$
$\pi_\rho \leftarrow \mathsf{PCS.Open}(\tilde{\rho}, \boldsymbol{\tau})$
return $\sigma = (p_1, \ldots, p_\mu, v, r, \pi_e, \pi_\rho)$

Algorithm 3 $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$

Require: Public key pk, message m, signature σ
Ensure: $\top$ (accept) or $\bot$ (reject)

Parse $\mathsf{pk} = (\mathbf{H}, \mathbf{s}, w, c_e, c_\rho)$
Parse $\sigma = (p_1, \ldots, p_\mu, v, r, \pi_e, \pi_\rho)$
$(\gamma_0, \ldots, \gamma_4, \mathbf{r}, \rho_{\text{seed}}) \leftarrow \mathcal{H}(\mathsf{pk} \| m)$
Generate masks $\tilde{z}, \tilde{z}_2, \tilde{z}_4$ from ρ_{seed}
$T \leftarrow \sum_{\mathbf{y}} \text{eq}(\mathbf{r}, \mathbf{y}) \cdot \tilde{s}(\mathbf{y})$
claimed $\leftarrow (\gamma_0 + \gamma_1 + \gamma_2) \cdot w + \gamma_3 \cdot T$
transcript $\leftarrow (\mathsf{pk}, m)$
for $i = 1, \ldots, \mu$ **do**
    if $p_i(0) + p_i(1) \neq$ claimed **then return** $\bot$
    end if
    transcript $\leftarrow$ transcript$\| p_i$
    $\tau_i \leftarrow \mathcal{H}(\text{transcript})$
    claimed $\leftarrow p_i(\tau_i)$
end for
$\boldsymbol{\tau} \leftarrow (\tau_1, \ldots, \tau_\mu)$
$h \leftarrow \tilde{H}_{\mathbf{r}}(\boldsymbol{\tau})$
$z \leftarrow \gamma_0 \tilde{z}(\boldsymbol{\tau}) + \gamma_1 \tilde{z}_2(\boldsymbol{\tau}) + \gamma_2 \tilde{z}_4(\boldsymbol{\tau})$
expected $\leftarrow \gamma_0 v + \gamma_1 v^2 + \gamma_2 v^4 + \gamma_3 h v + z + \gamma_4 r$
if claimed $\neq$ expected **then return** $\bot$
end if
if $\mathsf{PCS.Verify}(c_e, \boldsymbol{\tau}, v, \pi_e) \neq 1$ **then return** $\bot$
end if
if $\mathsf{PCS.Verify}(c_\rho, \boldsymbol{\tau}, r, \pi_\rho) \neq 1$ **then return** $\bot$
end if
return $\top$