Jevil: A Catastrophic-Failure-by-Design Signature Scheme

Nadim Kobeissi

Symbolic Software

Abstract. Few-time signatures cap how many signatures a signer can safely issue. **Jevil** is, to our knowledge, the *first post-quantum and transparent (setup-free) few-time signature scheme with a sharp key-recovery cliff*: its cap is enforced by a single sharp threshold rather than a slow slope. Signatures one through $n^\star$ are existentially unforgeable at approximately 124-bit classical security; at the $(n^\star{+}1)$-th the entire secret polynomial becomes publicly recoverable, achieving catastrophic failure as a key design requirement. The cap is founded on a secret polynomial together with the degree-binding of a polynomial commitment, and is intrinsic to any accepted public key: even a malicious signer who chooses pk adversarially cannot construct one that lets them keep signing past the cliff without the same polynomial becoming publicly recoverable.

All prior post-quantum few-time schemes (HORS, FORS, PORS, HORSIC$^+$, eBiBa, Syrga$_2$) degrade softly as $(nK/T)^K$. Every prior cliff-style construction misses at least one of *post-quantum*, *transparent*, *sharp*, and *count-triggered*: one-time Schnorr/ECDSA nonce reuse is neither post-quantum nor a designed property; polynomial-witness key-extraction signatures (notably DSKE) over KZG or IPA commitments are sharp but neither post-quantum nor transparent, while DSKE's hash-based variant is post-quantum and transparent but degrades softly; and double-authentication-preventing signatures (DAPS), post-quantum and transparent in their lattice form, fire a sharp key-recovery cliff on a message *predicate* (signing conflicting messages) rather than on a signature *count*, so they are not few-time schemes.

Concretely, Jevil provides 68-byte public keys, 32-byte secret keys, and $\sim$ 40 KB to $\sim$ 500 KB signatures across the recommended signing budgets $n^\star \in \{1, 3, 7, 15, \ldots, 2^{14} - 1\}$, the range admitted by the working field's 2-adicity. All primitives are believed to be post-quantum.

Keywords: Few-time signatures · WHIR · Polynomial commitments · Post-quantum cryptography · Hash-based signatures

1 Introduction

A *few-time signature* (FTS) scheme caps how many signatures a signer can safely issue. The first FTS, BiBa[1] [1], and its successor HORS[2] [2] use t-subset cover constructions,[3]

E-mail: nadim@symbolic.software (Nadim Kobeissi)

[1] BiBa = "Bins and Balls". The signer holds many bins of pre-images and a signature reveals the contents of a few message-derived bins; security reduces to the difficulty of an adversary finding pre-images covering a fresh message's bins.

[2] HORS = "Hash to Obtain Random Subset". The secret key is a list of T pre-images of a one-way function; a signature on $\mathcal{M}$ publishes the K pre-images at the positions $H(\mathcal{M}) \to \{0, \ldots, T-1\}^K$.

[3] A function family is *t-subset-resilient* if, given a target subset $S^* \subseteq \{0, \ldots, T-1\}$ of size K, finding t messages whose hash-derived subsets jointly cover S^* is computationally hard.

and remain the workhorses inside the stateless hash-based signatures SPHINCS$^+$ [4] and SLH-DSA [5], where an FTS is composed with a hypertree[4] to yield a many-time scheme. In their bare form, FTSs are just a simple numbers game: interesting in their own right whenever an application has *intrinsic finitude* (a firmware vendor capping their own release count, an operator binding themselves to a per-tenure attestation budget, an ephemeral session signer, any audit-budgeted credential that issues its allowance and then would just as soon sleep for the other 100 years), and one wants a *cryptographic* guarantee on that cap, not a policy promise.

The natural way to enforce a count cap is statefully: a counter, a synced database, a hardware register. Stateful schemes are notoriously fragile in deployment: a single counter rollback (a restored backup, a cloned VM, a forked container) silently re-uses key material and breaks security. Stateless schemes route around this by making the cap a property of the cryptographic object itself rather than the signer's discipline. HORS is stateless in this sense, but its degradation is soft: at n signatures, forgery probability rises gradually as $(nK/T)^K$, so "how many signatures is safe" is a slider on a continuum.

This paper presents **Jevil**, a *self-enforcing* FTS whose degradation is the opposite of soft. The signer picks a budget $n^\star$ at KeyGen from the set $\{1, 3, 7, 15, \ldots, 2^{14} - 1\}$ (the values for which $(n^\star{+}1)K$ is exactly a power of two). Signatures $1, \ldots, n^\star$ are existentially unforgeable at approximately 124-bit classical security against adversaries making at most 2^{128} random-oracle queries (Theorem 4). At the $(n^\star{+}1)$-th signature the entire secret polynomial becomes publicly recoverable in $O(D^2)$ field operations. The cap is enforced not by a counter but by the algebraic structure of a single secret polynomial together with the degree-binding of a polynomial commitment. Crucially, the cliff is intrinsic to any accepted public key, not a property of honest KeyGen: a malicious signer who chooses pk adversarially in the hope of signing indefinitely cannot escape it (Corollary 2). To the best of our knowledge, Jevil is the *first* post-quantum, transparent (setup-free) few-time signature scheme whose key-recovery threshold is sharp rather than soft (§2).

Our contributions.

1. *A new design point in FTS.* Jevil is the first post-quantum, transparent (no trusted setup) few-time signature scheme whose key-recovery threshold is a single sharp cliff rather than a soft probabilistic slope. We map the prior landscape in §2, locating the design points carved out by HORS-family schemes, polynomial-witness FTSs, and one-time signatures, and identifying the cell Jevil occupies.
2. *The construction.* Jevil's design fits in a few lines: encode the secret as the coefficients of a degree-D polynomial f, commit to the coefficient vector under zk-WHIR [7], sign by opening f at K message-derived points via one batched zk-WHIR linear-form proof. zk-WHIR's honest-verifier zero-knowledge makes the openings reveal nothing beyond f's evaluations at the K chosen points; the lift's structure makes the cliff binding on the signer. Although we instantiate it with zk-WHIR, the construction and all its proofs use only an abstract linear-form commitment interface, namely knowledge soundness and a multi-opening hiding property (§3.5), so zk-WHIR is simply one post-quantum, transparent choice. (§4)
3. *The cliff theorem.* For any honestly-generated key and any $n \geq \lceil M/K \rceil$ signatures whose position sets are pairwise disjoint, anyone holding the transcript recovers f in $O(D^2)$ field operations. Jevil meets the honest-key cliff of Def. 3. (§5)
4. *Cap-binding.* Even an *adversarial* KeyGen can't delay the cliff. zk-WHIR's knowledge soundness forces every accepted opening to be the evaluation of a polynomial

[4] A *hypertree* is a tree of Merkle trees: each non-leaf node's authenticated value is itself the root of an inner Merkle tree of one-time-signature keys. A single hypertree root certifies an exponentially large set of FTS keys, lifting an FTS into a stateless many-time scheme without long-term state.

of degree at most D, and the cliff theorem does the rest, giving the adversarial-key cliff of Def. 4, which covers even maliciously generated public keys. (§6.1)

5. *Unforgeability below the cliff.* A concrete-security EUF-CMA proof reducing forgery to (i) the HORS coverage bound, (ii) zk-WHIR's knowledge soundness, and (iii) Schwartz–Zippel over the Fiat–Shamir batching coefficients. (§6.2)
6. *A practical parameter recipe.* A single input $n^\star \in \{1, 3, 7, 15, \ldots, 2^{14} - 1\}$ determines all derived sizes via two power-of-two derivations; the resulting deployments have 68-byte public keys, 32-byte secret keys, and $\sim$ 40 KB to $\sim$ 500 KB signatures across the deployable range. The signing-budget set is the one for which $(n^\star{+}1)K$ is exactly a power of two (so $n_{\text{cliff}} = n^\star{+}1$ exactly), and is upper-bounded by the working field's 2-adicity. (§7)

Roadmap. §2 surveys prior work and positions Jevil. §3 fixes notation and recalls the FTS security game, the Goldilocks/$\mathbb{F}$ tower, the three hash primitives, and the WHIR linear-form opening interface. §4 specifies KeyGen, Sign, Verify along with the parameter recipe and the lift. §5 states and proves the cliff. §6 gives the security analysis (cap-binding, EUF-CMA, hiding, post-quantum). §7 gives concrete parameters and rough timings. §8 discusses when Jevil is and isn't the right tool.

2 Related work

We organise prior work along two axes that together pick out the design point Jevil occupies: *degradation shape* (soft slope vs. sharp cliff) and *assumption profile* (post-quantum and transparent vs. classical or setup-dependent). Table 1 summarises the resulting landscape; the remainder of this section walks through each family.

Table 1: The few-time / one-time signature landscape. Representative members of each family are enumerated and cited in the paragraphs below. *Soft slope*: forgery probability rises gradually with n. *Hard counter*: cap enforced by signer state, not by the cryptographic object. *Cliff*: a designed threshold at which a target (identity, key) is publicly recovered. *Accidental cliff*: an unintended failure mode, not a designed property. The *trigger* column separates schemes whose failure is driven by a signature *count* from those driven by a message *predicate* (signing conflicting messages), which leaves the total count unbounded. DAPS/PAPS has classical [26, 27] and lattice [28] instances; the row shows the lattice (post-quantum, transparent) one. DSKE [25] ships both a KZG polynomial-witness variant (counted with the poly-witness family) and a hash-based variant. Jevil is the only entry in the post-quantum, transparent, *count*-triggered key-recovery-cliff cell.

Family	Degradation	Trigger	Assumption profile
HORS-family FTS	soft slope	count	post-quantum, transparent
Stateful HBS	hard counter	count	post-quantum, transparent
Schnorr / ECDSA reuse	accidental cliff	nonce reuse	classical (discrete log)
Offline e-cash	cliff on identity	double-spend	classical, setup-dependent
Poly-witness FTS, DSKE	cliff on key	count	classical, setup-dependent
DSKE (hash-based)	soft slope	count	post-quantum, transparent
DAPS / PAPS (lattice)	cliff on key	predicate	post-quantum, transparent
Jevil (this work)	**cliff on key**	**count**	**post-quantum, transparent**

HORS-family few-time signatures (soft degradation). The dominant FTS family descends from BiBa [1] and its successor HORS [2], both built on t-subset-resilient functions: a secret key is a list of T pre-images, and a signature publishes the K pre-images at message-derived positions. After n signatures, an adversary's forgery succeeds with probability at most $(nK/T)^K$ by the standard coverage bound; this is a *soft* curve that approaches but never reaches certainty. PORS [3] and HORST tighten the constants by using pseudo-random subsets and Merkle authentication respectively, but preserve the soft-degradation profile. FORS, the FTS at the leaves of SPHINCS$^+$ [4] and the standardised SLH-DSA [5], replaces HORS with a forest of authenticated trees; the leaf-level degradation is still the same coverage curve. More recent variants such as HORSIC$^+$ [13], eBiBa [14], and Syrga$_2$ [15] continue this lineage with size or speed improvements but unchanged degradation shape. In every case, the operationally meaningful cap is the largest n for which the coverage bound stays below a chosen probability; there is no $n^\star$ after which the key becomes recoverable in any meaningful sense.

Stateful hash-based signatures. XMSS, LMS, and the stateful variants of SLH-DSA cap signatures with a counter. Over-signing is *prevented*, not *punished*: the signer simply refuses to issue past the budget. The cap is hard in the sense that the signer can't go past it without falsifying state, but it is enforced by signer discipline plus durable storage, not by anything algebraic. The well-known failure mode: counter rollback (restored backup, cloned VM, forked container) re-uses state and silently breaks security. That fragility is what motivated the stateless SPHINCS$^+$/SLH-DSA line in the first place.

Cliff-style behaviour from nonce reuse. Standard Schnorr and ECDSA signatures exhibit a different kind of cliff: any two signatures produced with the same nonce k permit linear key recovery from $\sigma_1 - \sigma_2 = k^{-1}(h_1 - h_2)$. This is the closest *geometric* analogue to Jevil's cliff (one extra signature $\Rightarrow$ full key recovery). However: (i) the cliff is at $n = 2$ (the schemes are one-time under nonce reuse, not few-time); (ii) it is an *unintended* failure mode of the implementation, not a designed property of the scheme; (iii) Schnorr and ECDSA rest on the discrete-log assumption and are not post-quantum. We list nonce reuse here only because it is the most often-cited example of "$n + 1$ signatures reveal the key," to make clear that Jevil's cliff is qualitatively different: a designed, configurable threshold at $n^\star + 1$ within a post-quantum FTS.

Cliff-style behaviour from double-spending. Chaum-style offline electronic cash [16, 17] embeds a designed cliff: a user who spends the same coin twice has her *identity* (not her signing key) cryptographically extracted by the bank. Subsequent refinements [18] extend this to fair e-cash schemes with opening authorities. The geometry is similar (one extra use opens what one use kept hidden), but the setting is different: an interactive bank-merchant-user protocol, not a stand-alone signature scheme, with the recovered object being the user's identity inside a group-signature-like construction rather than the secret signing key. None of the well-known constructions in this line are believed to be post-quantum.

Self-enforcement: DAPS and PAPS. Double-authentication-preventing signatures (DAPS) [26, 27] embed a designed key-recovery cliff of a different flavour: a signer who ever signs two messages sharing a *subject* but differing in *payload* thereby exposes their own signing key. Boneh, Kim, and Nikolaenko [28] give the first *lattice-based* DAPS (hence the first post-quantum, transparent one) and generalise it to predicate-authentication-preventing signatures (PAPS), where signing any tuple of messages satisfying a fixed predicate ϕ leaks the key. On the assumption axis this is the closest prior work to Jevil: the cliff is sharp (one offending tuple recovers the key in full), and the lattice instance is

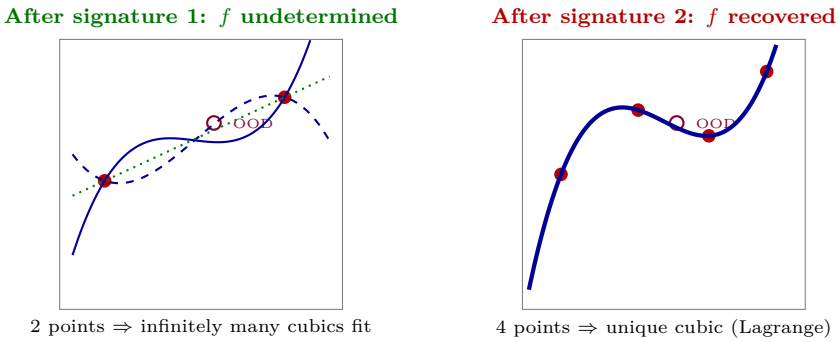


Figure 1: Toy example with $K = 2$ and $n^\star = 1$ (so $M = 4$, $D = 3$, $n_{\text{cliff}} = 2$). Red filled dots are evaluation pairs revealed by signatures; the purple hollow dot is the public out-of-domain (OOD) pair $(z, w) = (z, f(z))$ from pk (here just one more public evaluation of f). **Left**: after the first signature, two signature pairs plus the OOD pair (three total) are public, still under the four needed for a cubic, so f is information-theoretically undetermined. **Right**: after a second signature with two fresh positions, four signature pairs plus the OOD (five total) over-determine the cubic; Lagrange interpolation on any four reconstructs f in $O(D^2)$ field operations.

simultaneously post-quantum and transparent. The difference is the *trigger*. Jevil's cliff fires on a signature *count* (any $n^\star{+}1$ distinct messages suffice), whereas a DAPS/PAPS cliff fires on a message *predicate*: a DAPS signer may issue unboundedly many non-conflicting signatures and never leak, because the count is irrelevant until two messages collide on a subject. Even PAPS under the "$n^\star{+}1$ equal subjects, distinct payloads" predicate caps signatures *per subject*, not the total count of distinct-message signatures that a few-time scheme bounds. DAPS/PAPS is thus a sibling self-enforcement primitive rather than a few-time signature; it is why the "few-time" (count-triggered) qualifier in our claim does real work, and why Table 1 separates the two families on the trigger axis.

Polynomial-witness FTSs. A line of work commits to a degree-D polynomial and signs by opening K evaluations; the Lagrange-interpolation cliff at $\lceil (D+1)/K \rceil$ is then *intrinsic* to the scheme. Figure 1 illustrates the cliff geometry at toy parameters $K = 2$, $n^\star = 1$. Constructions in this family typically use KZG [19][5] or inner-product argument (IPA) [20, 21][6] polynomial commitments, which give succinct openings but inherit either a trusted setup (KZG) or a discrete-log assumption (KZG and IPA both). These schemes thus share Jevil's cliff geometry, but none are simultaneously post-quantum and transparent: KZG is broken by Shor and additionally requires a trusted setup; IPA is transparent but is broken by Shor. The most fully developed member of this family is **DSKE** (digital signatures with key extraction) [25]: it formalises the *extractable set* and states extraction games for both honest and untrusted key generation (close analogues of our honest- and adversarial-key cliffs, Defs. 3, 4), and its polynomial-commitment instantiation $\mathsf{DSKE}_{\mathsf{poly}}$ places the Lagrange cliff over KZG. The degree-binding it relies on, which [25] terms *interpolation binding*, is the commitment-level fact our cap-binding theorem (§6.1) re-

[5] *KZG* (Kate–Zaverucha–Goldberg): a degree-D polynomial is committed to in a single elliptic-curve group element by evaluating it "in the exponent" at a trapdoored point τ. Openings are constant-size proofs. KZG requires a trusted setup (the powers of τ must be generated and the secret τ destroyed) and is broken by Shor's algorithm.

[6] *IPA* (Inner Product Arguments) commits to a vector via a Pedersen-like commitment in a Schnorr-style group and proves inner-product relations in $O(\log n)$ communication. Bulletproofs [21] packages IPAs for range proofs and arithmetic circuits. Transparent (no trusted setup) but discrete-log-based and Shor-broken.

establishes over zk-WHIR. DSKE additionally offers a hash-based variant $\mathsf{DSKE}_{\mathsf{hash}}$ that, like Jevil, is post-quantum and transparent, but it is built from one-time-signature key leakage, so its extraction degrades *softly* (a HORS-style coverage curve) rather than at a sharp threshold. No single DSKE construction is at once post-quantum, transparent, and sharp: $\mathsf{DSKE}_{\mathsf{poly}}$ is sharp but classical and setup-dependent, $\mathsf{DSKE}_{\mathsf{hash}}$ is post-quantum and transparent but soft. Jevil's central technical move is to instantiate the *same* cliff geometry as $\mathsf{DSKE}_{\mathsf{poly}}$ over a hash-based polynomial commitment (zk-WHIR [7], descending from FRI [22], DEEP-FRI [9], and STIR [8]);[7] this brings the cliff into the post-quantum and transparent setting without losing the sharp threshold.

Threshold and secret-sharing-based signatures. Shamir [23][8] and Pedersen [24][9] secret sharing encode a secret as a degree-D polynomial recoverable from $D+1$ evaluations; threshold signatures (BLS threshold, threshold Schnorr/ECDSA)[10] exploit this. Jevil rests on the same Lagrange-interpolation fact but inverts its role: where threshold cryptography distributes evaluations across parties at KeyGen as the *intended* reconstruction path, Jevil accumulates them across a single signer's openings as the *penalty* for over-use, with evaluation positions tied to message hashes rather than party identities.

Hash-based polynomial commitments. zk-WHIR [7] is an honest-verifier zero-knowledge variant of WHIR, itself the most recent in a line of Reed–Solomon proximity tests (FRI [22], DEEP-FRI [9], STIR [8]) used as polynomial commitments in succinct arguments. We use zk-WHIR for its fast verifier, its linear-form opening interface, and its honest-verifier zero-knowledge guarantee; all three fit the lift cleanly.

3 Preliminaries

This section fixes notation (§3.1), the FTS security games (§3.2), the field tower (§3.3), the three hash functions (§3.4), and the zk-WHIR linear-form interface (§3.5). zk-WHIR is the only nontrivial primitive we assume as a black box.

3.1 Notation

For a finite field $\mathbb{F}$ and integer N, we write $\mathbb{F}^N$ for the N-dimensional vector space. For $a, b \in \mathbb{F}^N$, $\langle a, b\rangle := \sum_{i=0}^{N-1} a_i b_i$ is the standard inner product; its bilinearity is what makes the Fiat–Shamir batching of §4.3 work. Polynomials are written in capitalised variables: $f(X) = \sum c_k X^k$, with $\deg f$ its degree. We use $\xleftarrow{\$}$ for uniform sampling from the indicated set. $\mathsf{negl}(\lambda)$ denotes a function negligible in λ. H denotes a hash function modelled as a

[7] The FRI–DEEP-FRI–STIR–WHIR lineage is the family of Reed–Solomon proximity tests used as polynomial commitments. FRI [22] introduced the recursive-folding template; DEEP-FRI [9] added out-of-domain sampling to amplify soundness; STIR [8] reduced query complexity by a constant factor; WHIR built on STIR with a different folding structure for fast verifier and additional algebraic flexibility. zk-WHIR [7] layers honest-verifier zero-knowledge on top of WHIR.

[8] *Shamir secret sharing*: split a secret $s \in \mathbb{F}$ by sampling a random polynomial $p \in \mathbb{F}[X]$ of degree D with $p(0) = s$ and distributing $p(i)$ to party i. Any $D+1$ parties reconstruct p (and hence s) by Lagrange interpolation; any D or fewer learn nothing about s (their shares are uniform).

[9] *Pedersen secret sharing* augments Shamir with public commitments to the polynomial's coefficients (via Pedersen commitments), so that shares can be *verified*: a party can check that its share is consistent with the dealer's public commitment without revealing it. Used as the foundation for verifiable secret sharing and threshold signing.

[10] In a t-of-n *threshold signature* scheme, any t of n parties can jointly produce a signature, but fewer cannot. The schemes named here all use polynomial secret-sharing of the signing key plus an interactive signing protocol. The Lagrange-interpolation reconstruction is the *intended* use; Jevil inverts the same fact into a penalty.

random oracle. We use $\|$ for byte concatenation. $\mathbb{F}^\times$ denotes the multiplicative group $\mathbb{F} \setminus \{0\}$, of order $|\mathbb{F}| - 1$.

Table 5 (Appendix A) summarises the principal symbols used throughout the paper.

3.2 Few-time signatures: syntax and security

Definition 1 (FTS scheme)**.** A *few-time signature scheme* is a triple of PPT algorithms $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\mathsf{KeyGen}(1^\lambda, n^\star) \to (\mathsf{pk}, \mathsf{sk})$. On input the security parameter and a signing budget $n^\star \in \mathbb{N}$, output a public key and a secret key.
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, \mathcal{M}) \to \sigma$. Deterministically sign a message.
- $\mathsf{Verify}(\mathsf{pk}, \mathcal{M}, \sigma) \to \{0, 1\}$. Accept or reject.

Correctness: for all $n^\star$ and all $\mathcal{M}$, if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, n^\star)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, \mathcal{M})$, then $\mathsf{Verify}(\mathsf{pk}, \mathcal{M}, \sigma) = 1$.

The standard adaptive EUF-CMA game, restricted to at most $n^\star$ signing queries:

Definition 2 (EUF-CMA-$n^\star$)**.** The game runs as follows. Generate a fresh key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, n^\star)$ and hand pk to the adversary $\mathcal{A}$. Let them make up to $n^\star$ signing queries on messages of their choice (queries past $n^\star$ return $\perp$); record the queried messages in Q. They submit a forgery $(\mathcal{M}^*, \sigma^*)$ on a message $\mathcal{M}^* \notin Q$. They *win* if $\mathsf{Verify}(\mathsf{pk}, \mathcal{M}^*, \sigma^*) = 1$. The advantage is just the win probability:

$$\mathsf{Adv}^{\text{EUF-CMA-}n^\star}_{\Pi,\mathcal{A}}(\lambda) \;:=\; \Pr[\mathcal{A} \text{ wins}].$$

Π is $(n^\star, \epsilon)$-EUF-CMA-secure if this advantage is $\leq \epsilon$ for every PPT $\mathcal{A}$.

EUF-CMA captures unforgeability, but not the thing Jevil promises beyond it: that past a certain point the key is publicly recoverable. The FTS literature has a name for this: a *key-recovery* game where the adversary, given pk and n signatures, has to output an equivalent signing key. We pin it down for Jevil with two complementary definitions: the honest-key cliff (signers play by the rules), which recovers a key that can itself sign, and the adversarial-key cliff (signers might not), which recovers enough to predict every value the key admits.

Definition 3 (Honest-key n_{cliff}-cliff)**.** Throughout this definition we restrict to FTS schemes whose Sign algorithm, on input $\mathcal{M}$, internally derives K field elements $(x_1, \ldots, x_K) \in \mathbb{F}^K$ before producing σ: the *positions* of $\mathcal{M}$ under that key. HORS, FORS, and Jevil all fit this template. We write $x_t^{(i)}$ for the t-th position derived for $\mathcal{M}_i$.

Π has an *honest-key* n_{cliff}*-cliff* if there is a PPT extractor $\mathcal{E}$ such that for all $n^\star$, all honestly generated $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, n^\star)$, and any honestly-produced signatures $\{(\mathcal{M}_i, \sigma_i)\}_{i=1}^{n}$ on *distinct* messages whose accumulated distinct positions satisfy

$$\Big|\bigcup_{i=1}^{n} \{x_t^{(i)}\}_{t=1}^{K}\Big| \;\geq\; n_{\text{cliff}}(n^\star) \cdot K,$$

we have

$$\Pr\big[\,\mathcal{E}(\mathsf{pk}, \{(\mathcal{M}_i, \sigma_i)\}_i) = \mathsf{sk}^\dagger \\ \text{such that } \forall \mathcal{M}\colon \mathsf{Verify}(\mathsf{pk}, \mathcal{M}, \mathsf{Sign}(\mathsf{sk}^\dagger, \mathsf{pk}, \mathcal{M})) = 1\,\big] \;\geq\; 1 - \mathsf{negl}(\lambda).$$

The signature-count threshold $n_{\text{cliff}}(n^\star)$ is the smallest n for which the accumulation bound is achievable, attained when the per-signature position sets are pairwise disjoint; for general (non-disjoint) honest signatures the same extraction holds whenever the accumulation bound is met, possibly only at a larger n.

Definition 4 (Adversarial-key n_{cliff}-cliff)**.** Π has an *adversarial-key n_{cliff}-cliff* if there is a deterministic public map $\mathsf{predict}$ and a PPT extractor $\mathcal{E}$ such that for all $n^\star$ and all PPT adversaries $\mathcal{B}$ on input $(1^\lambda, n^\star)$ producing $(\mathsf{pk}, \{(\mathcal{M}_i, \sigma_i)\}_{i=1}^n)$ with all $\mathcal{M}_i$ distinct, $\mathsf{Verify}(\mathsf{pk}, \mathcal{M}_i, \sigma_i) = 1$ for all i, and accumulated distinct positions satisfying

$$\Big|\bigcup_{i=1}^{n} \{x_t^{(i)}\}_{t=1}^{K}\Big| \;\geq\; n_{\text{cliff}}(n^\star) \cdot K,$$

the extractor recovers a witness $\mathsf{w}^\dagger \leftarrow \mathcal{E}(\mathsf{pk}, \{(\mathcal{M}_i, \sigma_i)\}_i)$ that *predicts every revealed value the public key admits*: except with probability $\mathsf{negl}(\lambda)$, for *every* message $\mathcal{M}$ and *every* $\sigma = ((y_t)_{t=1}^K, \pi)$ with $\mathsf{Verify}(\mathsf{pk}, \mathcal{M}, \sigma) = 1$, the revealed values satisfy $y_t = \mathsf{predict}(\mathsf{w}^\dagger, x_t)$ at the positions x_t of $\mathcal{M}$.

Remark 1 (Honest vs. adversarial cliff)*.* The two definitions are complementary rather than nested. The honest-key cliff (Def. 3) draws the *stronger conclusion* (a $\mathsf{sk}^\dagger$ that can itself *sign*) from the *weaker hypothesis* that the key was generated honestly. The adversarial-key cliff draws the *weaker conclusion* (predictive recovery) from the *stronger hypothesis* that pk is anything a PPT $\mathcal{B}$ can make the verifier accept. Predictive (rather than sign-able) recovery is the most one can ask of an adversarial key: a malicious committer's accepted openings certify only that the committed vector is *close* to a codeword (§3.5), not equal to one, so a third party cannot in general re-encode it into accepting proofs against the adversarial root. It is nonetheless the security content that matters: every value the signer could ever reveal becomes publicly computable, so the secret is exposed regardless of who can wrap it in a fresh proof. Jevil meets the honest-key cliff in §5 and the adversarial-key cliff in §6.1.

For Jevil the recovered witness is in both cases the coefficient vector $c \in \mathbb{F}^M$, equivalently the polynomial f, obtained in $O(D^2)$ field operations by Lagrange interpolation from the accumulated openings. Under the *honest*-key cliff this is upgraded to a sign-able $\mathsf{sk}^\dagger$: zk-WHIR's knowledge extractor returns the committed pair (c, r_{zk}) *together* (§3.5), and since an honestly generated root is exactly $\mathsf{Enc}_C(c, r_{\mathsf{zk}})$, re-running the public encoding step reproduces the commitment state st, so a signer reading $\mathsf{sk}^\dagger := (c, r_{\mathsf{zk}})$ produces openings that verify against the original pk (Corollary 1). Under the *adversarial*-key cliff we keep only f itself, serving as the $\mathsf{predict}$ witness via $\mathsf{predict}(f, x) := f(x)$ (Remark 1).

3.3 The Goldilocks field tower

Jevil works over a degree-4 extension of an FFT-friendly 64-bit prime field:

- *Base field* $\mathbb{F}_{q_0}$: the *Goldilocks* prime $q_0 = 2^{64} - 2^{32} + 1$ [10].[11] $\mathbb{F}_{q_0}^\times$ has a subgroup of order 2^{32}, supporting radix-2 NTT[12] up to that length.
- *Working field* $\mathbb{F} := \mathbb{F}_{q_0^4}$, built as a tower: $\mathbb{F}_{q_0^2} := \mathbb{F}_{q_0}[X]/(X^2 - 7)$ with ξ the residue class of X (so $\xi^2 = 7$), then $\mathbb{F} := \mathbb{F}_{q_0^2}[Y]/(Y^2 - \xi)$. Both 7 and ξ are quadratic non-residues at their respective levels,[13] so both quotients are fields, and $|\mathbb{F}| = q_0^4 \approx 2^{256}$.

[11] The Goldilocks prime is engineered to make modular arithmetic cheap on 64-bit hardware: $2^{64} \equiv 2^{32} - 1 \pmod{q_0}$, so a 128-bit product reduces with a handful of shifts, subtractions, and conditional subtractions of q_0. Its multiplicative group has order $q_0 - 1 = 2^{32}(2^{32} - 1)$, divisible by a 32-bit power of two: the property NTTs need.

[12] The Number Theoretic Transform (NTT) is the finite-field analogue of the Discrete Fourier Transform: it converts a polynomial's coefficient representation into its evaluations at a multiplicative subgroup of size N (a power of two) in $O(N \log N)$ field operations. We use it inside zk-WHIR.Commit to encode the secret coefficient vector into its Reed–Solomon codeword.

[13] An element a of a finite field K is a *quadratic non-residue* if there is no $b \in K$ with $b^2 = a$. When a is a non-residue, $X^2 - a$ is an irreducible quadratic over K, so $K[X]/(X^2 - a)$ is a genuine field extension of degree 2 rather than a product of rings. Iterating this once over $\mathbb{F}_{q_0}$ and once over $\mathbb{F}_{q_0^2}$ gives the degree-4 tower.

We encode $\mathbb{F}$-elements as four little-endian Goldilocks limbs (32 bytes total); we write $\mathsf{enc}(\cdot)$ for this canonical encoding, used wherever $\mathbb{F}$-elements appear on the wire (e.g., the OOD value w, revealed evaluations y_t, FS prefix bytes). The bound $|\mathbb{F}| \approx 2^{256}$ is forced by quantum search over candidate y_t values (§6.3): $\sqrt{|\mathbb{F}|} = q_0^2 = 2^{128} - 2^{97} + O(2^{65})$, which is $2^{128}(1-2^{-31})$, i.e., short of 2^{128} by a multiplicative factor $1-2^{-31}$. The shortfall is negligible: it costs at most $\log_2(1/(1-2^{-31})) \approx 2^{-31}$ bits of security, well below the $\sim$ 4-bit margin already absorbed by the final $(R+3)\cdot 2^{-128} < 2^{-124}$ bound of Theorem 4.

3.4 Hash functions and domain separation

We model three hash primitives as independent random oracles:

- H_{VC}: a compression function over Poseidon2-Goldilocks [6][14] with state width $t = 12$, rate $r = 8$, and capacity $c = 4$.[15] Used *only* inside zk-WHIR as the binary Merkle tree's leaf-hash and internal-node-hash. Defined formally at the end of this section.
- H_{xof}: SHAKE256 [11].[16] Used for seed expansion, position derivation, Fiat–Shamir batching coefficients, the per-signature prover-randomness derivation, and the OOD binding-point derivation.
- H_{fs}: `SHAKE128` [11] wrapped as a duplex sponge.[17] Used *only* inside zk-WHIR's own Fiat–Shamir[18] compilation (turning the interactive IOPP[19] into a non-interactive proof). The Jevil layer never invokes H_{fs} directly; the only Jevil interaction with the zk-WHIR Fiat–Shamir transcript is via the `JV-OPEN` instance-bytes prefix injected at Sign/Verify time (§4.3), which the zk-WHIR layer absorbs into the duplex-sponge state.

At Poseidon2 capacity $c = 4$ Goldilocks elements (256-bit capacity portion of Poseidon2's permutation state), H_{VC} achieves $\sim 2^{256}$ classical / $\sim 2^{128}$ quantum second-preimage and $\sim 2^{128}$ classical / $\sim 2^{85}$ quantum collision [12].[20] H_{xof} (SHAKE256) matches the same classical floors and inherits SHAKE's standard quantum bounds. H_{fs} (`SHAKE128`) provides $\sim 2^{128}$-bit classical / $\sim 2^{64}$-bit quantum challenge unpredictability, sufficient for Fiat–Shamir compilation of zk-WHIR's interactive IOPP at the target 128-bit (classical)

[14] Poseidon2 is an arithmetic-circuit-friendly permutation: its round function is a low-degree polynomial map over $\mathbb{F}_{q_0}^{12}$, designed so that a SNARK or STARK circuit verifying a Poseidon2 hash has a small constraint count. We use it here purely as a hash, but its low arithmetic complexity makes it cheap to fold inside zk-WHIR's vector-commitment Merkle tree.

[15] In a *sponge* construction, the permutation acts on a state of $t = r + c$ field elements split into a *rate* portion (mixed with absorbed input or read out as output) and a *capacity* portion (never directly exposed). The capacity provides the security floor: collisions and pre-images each cost work proportional to $|\mathbb{F}|^{c/2}$ and $|\mathbb{F}|^{c}$ respectively. Here $c = 4$ Goldilocks limbs ≈ 256 bits of capacity.

[16] SHAKE256 is the NIST SHA-3 extendable-output function: it accepts an arbitrary-length input and emits an arbitrary-length output stream, derived from the Keccak sponge with 256-bit capacity. "XOF" (extendable output function) is just a hash whose output length is a parameter; the same construction emits 32 bytes for one use and 10^9 for another.

[17] A *duplex sponge* interleaves absorb and squeeze: the prover can absorb a message, squeeze a challenge derived from everything absorbed so far, absorb again, and so on. This makes it the natural primitive for Fiat–Shamir, where each verifier challenge depends on the full transcript up to that point.

[18] The *Fiat–Shamir transform* converts a public-coin interactive proof (the verifier's messages are uniform random challenges) into a non-interactive one by replacing each challenge with the hash of the transcript so far. Sound in the random-oracle model; the resulting non-interactive proof is publicly verifiable by re-deriving the same challenges from the same transcript.

[19] An *Interactive Oracle Proof of Proximity* (IOPP) is a multi-round interactive protocol in which the prover sends large strings as oracles (the verifier queries only a few positions per oracle) and the verifier replies with random challenges. The proximity flavour proves that a committed function is close to a Reed–Solomon codeword; FRI, STIR, and WHIR are all Reed–Solomon IOPPs.

[20] The Brassard–Høyer–Tapp (BHT) bound: collisions in a random function with N-bit output can be found in $O(2^{N/3})$ quantum queries, compared to the classical $O(2^{N/2})$ birthday bound. At $N = 256$ this is 2^{85} rather than 2^{128}. The quantum cube-root is why $c = 4$ Goldilocks elements ($\approx$ 256-bit capacity) only gives $\sim 2^{85}$ quantum collision security; raising c to 6 (in §6.4) restores 2^{128}.

soundness level, since FS challenges only require challenge-space unpredictability, not invertibility (§6.4).

All three are modelled as independent random oracles for the security analysis. Every H_{xof} invocation is prefixed by an 8-byte ASCII tag selecting one of seven disjoint H_{xof} domains; an eighth tag, `JV-WHIR`, is reserved for H_{VC}'s capacity-IV constituent and never addresses H_{xof}. Each label is right-padded with the ASCII space character (`0x20`) to exactly 8 bytes; the eight padded tags are pairwise distinct as 8-byte strings.[21] Table 6 (Appendix A) summarises the eight tags and their roles.

Why each tag exists. `JV-OPEN` pins the zk-WHIR transcript to the Jevil instance $(n^\star, \mathsf{root}, w, \mathcal{M}, y_1, \ldots, y_K)$, so a zk-WHIR proof produced under one Jevil tuple cannot be retro-fitted to another. `JV-OPRD` derandomises zk-WHIR.Open's sumcheck round-polynomial masks [7, Construction 6.3], code-switching mask oracles [7, Construction 9.7], and OOD answers [7, Lemma 9.3], so Sign is a function of $(\mathsf{sk}, \mathsf{pk}, \mathcal{M})$ rather than a randomised sampler. `JV-RZK` likewise derandomises zk-WHIR.Commit's Prop. 3.19 encoding randomness from σ_{zk}. Per-signature uniqueness of the derived ρ is by inclusion of $(\mathcal{M}, y_1, \ldots, y_K)$ in the `JV-OPRD` input; in the random-oracle model, the resulting stream is statistically indistinguishable from fresh uniform randomness assumed by Lemma 1, provided s remains secret.

Canonical input encoding. We write $H_{\mathsf{xof}}(\mathrm{tag}, x_1, \ldots, x_k; L)$ for the L-byte output applied to

$$\mathrm{tag} \,\|\, \mathsf{len}_8(x_1) \,\|\, x_1 \,\|\, \cdots \,\|\, \mathsf{len}_8(x_k) \,\|\, x_k,$$

where $\mathsf{len}_8(\cdot)$ is the byte length encoded as an 8-byte little-endian unsigned integer. Length prefixes make the encoding *injective*: distinct tuples $(x_1, \ldots, x_k)$ produce distinct byte strings, so no two H_{xof} calls can collide on construction.

The vector-commitment hash H_{VC}. Let $\pi : \mathbb{F}_{q_0}^{12} \to \mathbb{F}_{q_0}^{12}$ be the Poseidon2-Goldilocks-12 permutation. Let $g_{\texttt{WHIR}} := \mathsf{u64}{\to}\mathbb{F}_{q_0}(\texttt{4A 56 2D 57 48 49 52 20})$ be the canonical base-field image of the `JV-WHIR` tag bytes interpreted as a little-endian $u64$ and reduced mod q_0 (the bytes are below q_0, so the reduction is the identity). Define two capacity IVs in $\mathbb{F}_{q_0}^4$:

$$\begin{aligned} \mathsf{IV}_{\mathrm{leaf}} &:= (g_{\texttt{WHIR}}, 0, 0, 0), \\ \mathsf{IV}_{\mathrm{node}} &:= (g_{\texttt{WHIR}}, 1, 0, 0). \end{aligned}$$

The single 1 in the second slot of $\mathsf{IV}_{\mathrm{node}}$ versus the 0 in $\mathsf{IV}_{\mathrm{leaf}}$ is the leaf-versus-internal-node domain separator in the random-oracle model.

H_{VC} is invoked in two modes:

- *Leaf hash* $H_{\mathsf{VC}}^{\mathrm{leaf}}(m_1, \ldots, m_L)$, on $L \geq 1$ base-field input elements $m_i \in \mathbb{F}_{q_0}$ (in Jevil's deployment, the leaves are stripes of $\mathbb{F}$-elements decoded into base-field limbs in canonical little-endian order, so L is a multiple of 4 determined by the WHIR shape at the calling site and known to both parties). Let $L' := \lceil L/r \rceil \cdot r$ and let

[21] *Domain separation* prevents an output from one use of the hash function being mistaken for an output of another. Without separate tags, a single H_{xof} query could be re-interpreted in two contexts and would no longer be modellable as independent random oracles. Each tag effectively turns H_{xof} into a different oracle, so collisions across tags are impossible by construction.

$m'_1, \ldots, m'_{L'}$ extend $(m_i)_{i=1}^{L}$ with $L' - L$ zero Goldilocks elements. Run

$$\begin{aligned} s_0 &:= (\underbrace{0, \ldots, 0}_{r}, \mathsf{IV}_{\text{leaf}}) \in \mathbb{F}_{q_0}^{12}, \\ s_i &:= \pi\big(s_{i-1} + (m'_{(i-1)r+1}, \ldots, m'_{ir}, \underbrace{0, \ldots, 0}_{c})\big) \\ &\qquad \text{for } i = 1, \ldots, L'/r, \end{aligned}$$

and output the first 4 Goldilocks elements of $s_{L'/r}$ encoded as 32 bytes in canonical little-endian form (8 bytes per base-field limb).

- *Internal-node hash* $H_{\mathsf{VC}}^{\text{node}}(\mathsf{left}, \mathsf{right})$, on two 32-byte child digests $\mathsf{left}, \mathsf{right} \in \{0,1\}^{256}$. Decode each into 4 Goldilocks limbs in canonical little-endian order: $\mathsf{left} = (\ell_1, \ldots, \ell_4)$, $\mathsf{right} = (\mathsf{r}_1, \ldots, \mathsf{r}_4)$. Compute

$$s := \pi\big(\ell_1, \ldots, \ell_4, \mathsf{r}_1, \ldots, \mathsf{r}_4, \mathsf{IV}_{\text{node}}\big)$$

and output (s_1, s_2, s_3, s_4) as 32 bytes (canonical little-endian).

The leaf hash uses one permutation per rate-sized chunk of input (two permutations for Jevil's 4-$\mathbb{F}$-element interleaved stripe, $L = 16$; one permutation for a 1-$\mathbb{F}$-element width-1 leaf, $L = 4$); the internal-node hash uses one permutation. The absence of a 10^*-style padding marker is sound because the calling-site shape is fixed and known to both parties, so the input length is unambiguous in context; the leaf-versus-node IV separator rules out cross-mode collisions in the random-oracle model. Length-extension attacks are blocked by the $c = 4$ Goldilocks-element capacity portion of π's state being non-output-revealed across calls: an adversary holding $H_{\mathsf{VC}}^{\text{leaf}}(m_1, \ldots, m_L)$ sees only the first 4 state slots and cannot reconstruct the capacity needed to extend the absorb chain.

3.5 zk-WHIR as a linear-form polynomial commitment

Why zk-WHIR? We need a polynomial commitment that is post-quantum (no Shor), transparent (no trusted setup), supports linear-form openings,[22] verifies in polylog work, and gives honest-verifier zero-knowledge.[23] zk-WHIR [7], the honest-verifier-ZK variant of the Reed–Solomon-proximity-test[24] IOP WHIR, ticks every box.

We treat it as a black box, leaning on the following results from [7]: Definition 3.16 and Proposition 3.19 (Reed–Solomon ZK encoding), Definition 9.2 and Lemma 9.3 (private zero-evader for out-of-domain sampling), Theorem 4.5 (HVZK composition for IORs), and Theorem 4 / Corollary 3 (the polylog HVZK IOPP instantiated with Reed–Solomon). The rest of this subsection nails down the deployment parameters, restates the linear-form API in those terms, derives the per-opening codeword-position budget that the HVZK simulator consumes, and proves the one extension Jevil needs on top of Corollary 3: multi-opening HVZK.

[22] A *linear-form opening* proves that $\langle c, \alpha \rangle = v$ for committed c, public α, claimed v. Polynomial evaluations are a special case: by Lemma 3, $f(x) = \langle c, (1, x, \ldots, x^{M-1}) \rangle$, so the evaluation $f(x) = y$ is a linear form. Batching K evaluations into one linear form via a Fiat–Shamir random combination is then a single opening, which is how Jevil produces only *one* zk-WHIR proof per signature.

[23] *Honest-Verifier Zero-Knowledge* (HVZK): a proof system is HVZK if there is an efficient simulator that, on input the statement alone (no witness), produces a transcript whose distribution is indistinguishable from a transcript of an honest execution *when the verifier follows the protocol*. Stronger ZK notions (malicious-verifier ZK, simulation-sound ZK) tolerate cheating verifiers, but HVZK suffices once the protocol is Fiat–Shamir compiled, since the random-oracle-derived challenges are honest by construction.

[24] A Reed–Solomon *proximity test* is a probabilistic check that a function $f : D \to \mathbb{F}$ given as an oracle is close to some degree-d polynomial, in the sense that they agree on most of D. The verifier samples a few positions; soundness follows from the fact that two distinct degree-d polynomials disagree at most d places. FRI, DEEP-FRI, STIR, and WHIR are successive improvements in query complexity and verifier cost.

Definition 5 (zk-WHIR linear-form API)**.** A *linear-form polynomial commitment* over $\mathbb{F}$ is a triple, parameterised at setup by a message length M (a power of two) and an HVZK budget B:

- $\mathsf{zk\text{-}WHIR.Commit}(c, \sigma_{\mathsf{zk}}) \to (\mathsf{root}, \mathsf{st})$. On input $c \in \mathbb{F}^M$ and a 32-byte seed σ_{zk}, produce a digest $\mathsf{root} \in \{0,1\}^{256}$ and prover state st. The seed deterministically drives the Prop. 3.19 encoding randomness sampled inside $\mathsf{zk\text{-}WHIR.Commit}$;[25] no part of it is exposed to the caller.
- $\mathsf{zk\text{-}WHIR.Open}(\mathsf{st}, \alpha, v;\ \rho) \to \pi$. Given $\alpha \in \mathbb{F}^M$, $v \in \mathbb{F}$ with $\langle c, \alpha \rangle = v$, and a 32-byte randomness seed ρ, produce a proof π. The seed ρ drives the prover-internal randomness consumed by the linear-form opening (sumcheck round-polynomial masks,[26] code-switching mask oracles, OOD answers; see [7, Constructions 6.3, 9.7 and Lemma 9.3]); in the random-oracle model, ρ drawn from a uniform stream is indistinguishable from fresh uniform randomness.
- $\mathsf{zk\text{-}WHIR.Verify}(\mathsf{root}, \alpha, v, \pi) \to \{0,1\}$. Check the claim against the root.

The abstract interface we rely on. Although we instantiate this API with zk-WHIR, none of Jevil's security proofs (§5–§6.3) look inside it: they invoke only three properties of a linear-form commitment over $\mathbb{F}$ with parameters (M, B).

1. **Completeness:** honestly produced openings verify.
2. **Knowledge soundness with degree-binding:** from a prover that keeps convincing $\mathsf{zk\text{-}WHIR.Verify}$ of linear-form claims against a single root, an extractor recovers a witness $(c, r_{\mathsf{zk}}) \in \mathbb{F}^M \times \mathbb{F}^B$ with $\langle c, \alpha_i \rangle = v_i$ on every claim, so every accepted opening is a linear form in one length-M committed vector. This is what cap-binding consumes (§6.1).
3. **Multi-opening HVZK:** any n openings against one root are simulatable from the claims alone whenever $n\theta \le B$, for the per-opening leakage θ. This is what hiding consumes (§6.3).

Properties (1)–(2) are standard for the FRI/STIR/WHIR family; (3) is the one not immediate from [7], whose HVZK statement (Corollary 3) covers a single opening, so we establish it as Lemma 1 below. Stating the dependence this way keeps the analysis modular: any post-quantum, transparent linear-form commitment meeting (1)–(3) instantiates Jevil, and the proofs are insulated from revisions to a particular commitment's internals.

Reference instantiation. We instantiate the API by the Reed–Solomon zk-WHIR IOPP of [7, Corollary 3], Fiat–Shamir compiled in the random-oracle model with H_{fs} (§3.4), and with leaf and internal-node hashes provided by H_{VC} (§3.4). Every parameter is fixed deployment-wide: an implementation that changes any of them isn't the same primitive any more. Write $N := M+B$ rounded up to the next power of two for the WHIR-internal ZK-encoded message length (message slots plus HVZK budget slots). Table 7 (Appendix A) lists every deployment-wide value; the bullets that follow walk through the design choices.

- *Main code C*: Reed–Solomon[27] over $\mathbb{F}$ with rate $\rho = 1/4$ and evaluation domain a multiplicative subgroup of $\mathbb{F}^\times$ of size $4N$. Before encoding, the coefficient vector c

[25] The Prop. 3.19 encoding randomness $r_{\mathsf{zk}} \in \mathbb{F}^{N-M}$ pads the secret coefficient vector c before Reed–Solomon encoding. Its purpose is to make the resulting codeword's individual positions statistically independent of c: an adversary inspecting up to $N - M$ codeword positions learns nothing about c. This is what supports the HVZK guarantee: the simulator can re-sample these positions without ever seeing c.

[26] The *sumcheck protocol* reduces a claim $\sum_{b \in \{0,1\}^m} g(b) = S$ to a single evaluation $g(r_1, \ldots, r_m)$ at a verifier-chosen random point, via m rounds in which the prover sends a univariate polynomial and the verifier replies with a random challenge. ZK variants add a mask polynomial per round so the round messages reveal nothing about g beyond the final evaluation.

[27] The Reed–Solomon code $C = \mathrm{RS}[\mathbb{F}, L, k]$ with rate $\rho = k/|L|$ encodes a length-k message into a

is padded with $t := B = N - M$ entries of randomness r_{zk} (Prop. 3.19 ZK encoding); zk-WHIR.Commit derives r_{zk} from the caller-supplied σ_{zk} via $H_{\mathsf{xof}}(\texttt{JV-RZK}, \sigma_{\mathsf{zk}}; \cdot)$, and the committed codeword is $\mathsf{enc}(c) := \mathsf{Enc}_C(c, r_{\mathsf{zk}}) \in \mathbb{F}^{4N}$. The padding buys us *t-query perfect ZK* ($\zeta_C = 0$):[28] any set of $\leq t$ codeword positions looks like fresh uniform $\mathbb{F}$-elements.

- *Folding / interleaving.*[29] Folding parameter $k = 2$, interleaving factor $\kappa := 2^k = 4$ (distinct from the FTS positions-per-signature parameter $K = 16$ fixed in §4.1; we use κ throughout the WHIR description to avoid notational collision with that K). Each WHIR round reduces the inner-message length by a factor of $\kappa = 4$; folding halts once the inner message reaches $2^6 = 64$ symbols. The number of code-switching rounds is
$$R := \lceil \log_\kappa(N/2^6) \rceil \;=\; \lceil \log_4(N/2^6) \rceil.$$
- *Per-round queries.* $\theta = 64$ in-domain spotchecks per round and $\eta = 2$ out-of-domain (OOD) samples per round.[30] Where does $\theta = 64$ come from? Each spotcheck independently catches a cheating prover with probability $\geq \delta$, the proximity radius;[31] θ spotchecks fail to catch with probability $(1-\delta)^\theta$. At the conjectured $\delta = 1 - \rho = 3/4$, one spotcheck fails with prob $1/4$, and 64 of them fail jointly with prob $(1/4)^{64} = 2^{-128}$ per round. Over the largest deployable $R \leq 8$ rounds, the cumulative soundness error stays under $R \cdot 2^{-128} \leq 2^{-124}$, with OOD and combination-round contributions of [7, Lemma 9.9] absorbed into the same target.
- *Private OOD padding (mandatory).* OOD answers are produced via the *private zero-evader*[32] construction of [7, Lemma 9.3], with padding length $r_{\text{ood}} \geq t_{\text{ood}}$ per session: each OOD answer is then statistically independent of the committed witness and does not consume the ZK budget below. Any zk-WHIR implementation used to instantiate the API *must* apply Lemma 9.3's private padding; without it the HVZK property below does not hold.
- *Merkle multiproof encoding.*[33] For each set of in-domain spotcheck positions queried against a single Merkle root (whether on the main code C, the mask code C_{zk}, or

codeword of length $|L|$ by evaluating its associated polynomial at all $|L|$ points of the evaluation domain L. Two codewords disagree at $\geq |L| - k + 1$ positions, which gives the distance and the proximity-test soundness.

[28] *t-query perfect ZK* means any set of $\leq t$ codeword positions, viewed as random variables over uniform r_{zk}, has a distribution identical to t fresh uniform $\mathbb{F}$-elements. Perfect ($\zeta_C = 0$) is the strongest version: no statistical gap. The hypothesis $\theta \leq t$ is what the multi-opening lemma exploits, by treating zk-WHIR's θ first-round spotchecks as the budget consumed per signature.

[29] In each WHIR *code-switching* round the prover replaces the current Reed–Solomon code on a domain L with one on a smaller domain L' of $|L|/\kappa$ points (the prover "folds" the codeword by combining κ symbols per output symbol with a verifier-chosen random linear combination), commits to the new codeword, and proves consistency between the two. The verifier's per-round work is logarithmic in the original $|L|$.

[30] An *OOD sample* asks the prover to commit to the polynomial's value at a verifier-chosen point *outside* the codeword's evaluation domain. The verifier checks this value against the prover's earlier oracle. This amplifies soundness beyond what in-domain spotchecks alone can guarantee: a malicious prover would have to match the OOD value across rounds, which only the unique closest codeword can do.

[31] The *Reed–Solomon proximity-gap conjecture* [9], informally: if a set of polynomially-many functions is each δ-close to the code, then a uniform random linear combination of them is also δ-close, except with negligible probability. This is what lets us batch many closeness claims into one and underlies FRI/STIR/WHIR soundness at large proximity radii. It is conjectural; the unconditional bound is much weaker.

[32] The naive OOD answer is itself a polynomial evaluation, which would leak one degree of freedom in c per OOD query. The *private zero-evader* of [7, Lemma 9.3] adds a mask polynomial that zeroes out on a chosen public point but is uniform elsewhere; the OOD answer is then statistically independent of c and "evades" the ZK budget. Without this, every OOD sample would eat into the $t = N - M$ ZK budget.

[33] A *Merkle multiproof* authenticates θ leaves of a balanced binary tree of depth d using approximately $\theta(d - \log_2 \theta + 1)$ sibling hashes instead of θd (the naive cost of θ independent paths), because paths to nearby leaves share many ancestors and each ancestor's contribution is emitted only once. The verifier reconstructs the same union of siblings from the same sorted query set.

any per-round commitment), authentication paths are encoded as a single Merkle multiproof: the union of "outside" sibling hashes traversed bottom-up, with shared ancestors emitted only once. At θ queries against a balanced binary tree of depth d, this reduces the path-byte count from $\theta \cdot d$ to approximately $\theta \cdot (d - \log_2 \theta + 1)$ ($\sim$25–50% saving across Jevil's deployable range). The construction is the standard sorted-frontier walk: sort queried indices, walk level-by-level, and emit a node's sibling iff the sibling is not also a queried index at this level; the verifier reconstructs the same walk from the same sorted indices and consumes siblings from the proof in the same order. The pruning is purely a wire-format compaction and does not alter the random-oracle modelling of H_{VC} from §3.4.

- *Mask code* C_{zk}: Reed–Solomon over $\mathbb{F}$ at rate $\rho_{\mathsf{zk}} = 1/16$, with numerics in Table 7. Used for the sumcheck and randomness-padding masks of [7, Constructions 6.3 and 9.7]. The inner message has $\ell_{\mathsf{zk}} = 64$ slots, split in half: 32 reserved for mask data, 32 for Prop. 3.19 ZK randomness. That gives perfect 32-query ZK on the data half ($\zeta_{C_{\mathsf{zk}}} = 0$): any 32 codeword positions look like fresh uniform $\mathbb{F}$-elements, hiding whatever sits in the data slots. Of those 32 data slots, only the first 3 actually carry a sumcheck-round mask polynomial (degree-2 univariate, three coefficients) so the round-polynomial degree stays at 2, matching the non-ZK sumcheck wire format byte-for-byte;[34] the remaining 29 slots are zero-padded for the sumcheck use, and the full 32 are used by Construction 9.7's code-switching padding masks.

Why these mask-code parameters? Why not just use the main code's rate of 1/4? Because at $\rho_{\mathsf{zk}} = 1/16$, each spotcheck fails with probability 1/16 instead of 1/4, so half as many spotchecks (32 instead of 64) hit the same 2^{-128} soundness. The trade: 4× bigger codeword domain (one extra Merkle path level per mask spotcheck) for half the spotcheck count. Halving the count saves a full spotcheck's worth of leaf-and-path bytes per mask, which more than offsets the extra level. Same proximity-gap conjecture on both sides, so no extra conjectural surface.

Under this instantiation, [7, Theorem 4 / Corollary 3] gives the following guarantees, all conditional on the Reed–Solomon proximity-gap conjecture [9] and random-oracle modelling of H_{VC} and H_{fs}:

- **Completeness.** Honest openings always verify.
- **Multi-claim knowledge soundness** [7, Theorem 4].[35] If a malicious prover keeps convincing the verifier of multiple linear-form claims against the same root, an extractor pulls out the underlying witness $(c, r_{\mathsf{zk}}) \in \mathbb{F}^M \times \mathbb{F}^{N-M}$, with $\langle c, \alpha_i \rangle = v_i$ matched on every claim, except with probability $\epsilon_{\mathsf{zk\text{-}WHIR}} \leq R \cdot 2^{-128}$ (where R is the codeswitch round count of §3.5). The extraction returns $\mathsf{Enc}_C(c, r_{\mathsf{zk}})$ only δ-close in relative Hamming distance to a Reed–Solomon codeword,[36] so c is unique only modulo the proximity ball. Either way, it's the same c regardless of which opening you ran extraction on.

[34] Lemma 6.4 of [7] only needs the mask-poly slot count ≥ 2 for the sumcheck mask to do its hiding job. Picking 3 instead of 32 keeps the per-round wire format identical to plain WHIR's degree-2 sumcheck. The 32-query ZK budget applies to the encoding as a whole regardless of how many of those slots carry non-zero data.

[35] *Knowledge soundness* is strictly stronger than ordinary soundness: ordinary soundness says "if the prover convinces the verifier, then a valid witness exists"; knowledge soundness says "…and an extractor can *recover* that witness from the prover in polynomial time." In Jevil's cap-binding (Theorem 3) we extract the committed coefficient vector c from a malicious signer; that step would be vacuous under ordinary soundness alone.

[36] The *relative Hamming distance* between two strings of equal length is the fraction of positions where they differ. The *proximity ball* of radius δ around a codeword is the set of all strings within relative Hamming distance δ. zk-WHIR's proximity test certifies closeness, not equality: the extractor returns c up to this ball.

- **Single-opening HVZK** [7, Corollary 3]. One run of zk-WHIR.Open peeks at $\mathsf{enc}(c)$ at exactly $\theta = 64$ codeword positions (the first-round in-domain spotchecks) and never touches it again. Subsequent rounds work with freshly-committed codewords $g_2, \ldots, g_R$ that carry their own Prop. 3.19 randomness; OOD answers go through the private zero-evader of Lemma 9.3 and don't touch $\mathsf{enc}(c)$ either. So as long as $\theta \leq t = N - M$, one execution on a uniformly seeded zk-WHIR.Commit$(c, \sigma_{\mathsf{zk}})$ is honest-verifier zero-knowledge: $\mathcal{S}_{\text{single}}$ with θ oracle peeks at $\mathsf{enc}(c)$ produces a transcript that's statistically indistinguishable from[37] the real one (and identical for the Reed–Solomon instantiation). The derivation is by inspection of the per-round simulators of [7, Lemmas 6.4, 7.3, 9.8] composed via Theorem 4.5.

From single-opening to many-opening HVZK. Corollary 3 of [7] only covers *one* run of zk-WHIR.Open. Jevil runs it $n^\star$ times against the same root, so we need to lift the guarantee from one to many. Here's the extension.

Lemma 1 (Multi-opening HVZK)**.** *Let $c \in \mathbb{F}^M$ be fixed, σ_{zk} uniform, and* $(\mathsf{root}, \mathsf{st}) \leftarrow$ zk-WHIR.Commit$(c, \sigma_{\mathsf{zk}})$. *Let $(\alpha_1, v_1), \ldots, (\alpha_n, v_n) \in \mathbb{F}^M \times \mathbb{F}$ be any sequence of linear-form claims satisfying $\langle c, \alpha_i \rangle = v_i$, and let $\pi_i \leftarrow$* zk-WHIR.Open$(\mathsf{st}, \alpha_i, v_i; \rho_i)$ *with each ρ_i fresh uniform. If $n \cdot \theta \leq B = N - M$ (the WHIR commit's HVZK budget), then there is a PPT simulator $\mathcal{S}$ that, given only* $(\mathsf{root}, \{(\alpha_i, v_i)\}_{i=1}^n)$*, produces outputs $\{\tilde{\pi}_i\}_{i=1}^n$ such that*

$$\left\{(\alpha_i, v_i, \tilde{\pi}_i)\right\}_{i=1}^n \overset{s}{\approx} \left\{(\alpha_i, v_i, \pi_i)\right\}_{i=1}^n.$$

Here $\overset{s}{\approx}$ denotes statistical indistinguishability. For the Reed–Solomon instantiation the statistical distance is 0.

Proof. The whole game here: each opening's HVZK simulator only consults $\mathsf{enc}(c)$ at θ first-round positions; the rest of its randomness is fresh and independent of c. Over n openings, the union of consulted positions has size $\leq n\theta \leq B$, which is exactly the budget under which Prop. 3.19 says those positions are jointly distributed as fresh uniform $\mathbb{F}$-elements. So we sample the union *first*, then run each single-opening simulator as if it had read those values off a real $\mathsf{enc}(c)$. The joint distribution comes out perfect.

The single-opening simulator $\mathcal{S}_{\text{single}}$ from [7, Corollary 3], on input $(\mathsf{root}, \alpha, v)$, produces a transcript using fresh internal randomness (sumcheck masks, code-switching masks, OOD answers via the zero-evader, round-local codeword randomness) plus oracle access to $\mathsf{enc}(c)$ at exactly θ first-round positions, with $\zeta_{\text{zk-WHIR,HVZK}} = 0$ for Reed–Solomon. The fresh randomness is independent of c, so the only c-dependent input is those θ positions.[38]

Construct $\mathcal{S}$ as follows. For each $i \in [n]$:

1. Run the scaffold of $\mathcal{S}^{(i)}_{\text{single}}(\mathsf{root}, \alpha_i, v_i)$ up to (but not including) its first codeword-position query, recording the codeword-position query set $S_i \subseteq [4N]$ that it would make ($|S_i| = \theta$).

Let $S := \bigcup_{i=1}^n S_i$; by the hypothesis $|S| \leq n \cdot \theta \leq B = t$.

[37] *Statistical indistinguishability*: two distributions X, Y over the same domain are statistically indistinguishable if their total-variation distance $\frac{1}{2}\sum_z |\Pr[X = z] - \Pr[Y = z]|$ is negligible in the security parameter. This is stronger than *computational* indistinguishability, which only requires that no PPT algorithm tells them apart.

[38] In the ZK sumcheck of [7, Construction 6.3], each round's univariate polynomial must satisfy a linear consistency constraint with the previous round's challenge response. The valid round polynomials form an affine subspace; the simulator samples uniformly from that subspace, matching the honest prover's distribution because the honest mask is uniform within the same subspace.

2. Sample $\tilde{u} \leftarrow \mathsf{Sim}_C(S)$, where Sim_C is the simulator from [7, Definition 3.16, Proposition 3.19] applied to the joint query set S. Since $|S| \le t$, by Prop. 3.19 the distribution of $\tilde{u}$ is identical to that of $\mathsf{enc}(c)[S]$ over uniform σ_{zk} (statistical distance $\zeta_C = 0$ for Reed–Solomon).
3. For each i, resume $\mathcal{S}^{(i)}_{\text{single}}$, answering each codeword-position query $j \in S_i$ by the corresponding entry of $\tilde{u}|_{S_i}$. Output $\tilde{\pi}_i$.

Conditioned on a faithful oracle to $\mathsf{enc}(c)[S_i]$, each $\mathcal{S}^{(i)}_{\text{single}}$ produces a transcript distributed exactly as π_i (Corollary 3, perfect HVZK for RS). The values supplied on S_i are $\tilde{u}|_{S_i}$, jointly distributed as $\mathsf{enc}(c)[S]$ by step 2's Prop. 3.19 guarantee. The remaining randomness inside each $\mathcal{S}^{(i)}_{\text{single}}$ is independent across i and independent of c, so the joint distribution of $\{\tilde{\pi}_i\}_i$ matches $\{\pi_i\}_i$ exactly. □

The parameter recipe of §4.1 sets $B \ge n^\star \cdot \theta$, so Lemma 1's hypothesis holds for free across the entire signing budget.

3.6 HORS coverage bound

We will need the classical HORS coverage probability [2]. Operationally: the seen pool S is the union of all positions revealed across past signatures (size $\le nK$ after n signatures); the forger's fresh-message positions S^* are an independent uniform K-subset. The forger only wins on coverage if $S^* \subseteq S$, since otherwise they must supply at least one evaluation they have never seen.

Lemma 2 (HORS coverage)**.** *Let $T, K, n \in \mathbb{N}$ with $nK \le T$. Let $S \subseteq \{0, \dots, T-1\}$ with $|S| \le nK$, and let $S^* = \{i_1, \dots, i_K\}$ be a uniform random K-subset of $\{0, \dots, T-1\}$ independent of S. Then*

$$\Pr[S^* \subseteq S] \;=\; \binom{|S|}{K} \Big/ \binom{T}{K} \;\le\; \left(\frac{nK}{T}\right)^K.$$

The equality is just counting: among the $\binom{T}{K}$ equally likely K-subsets, exactly $\binom{|S|}{K}$ sit inside S. For the bound: each of the K draws lands in S with probability at most $|S|/T$, and sampling-without-replacement can only tighten this. Multiply K times, plug in $|S| \le nK$, done.

4 The Jevil construction

We first fix the parameter recipe (§4.1), then introduce the technical centrepiece (the monomial-basis lift, §4.2), and finally specify the three algorithms (§4.3). The whole, despite its moving parts, is just a simple chaos of standard ingredients.

4.1 Parameter recipe

The scheme takes one input from the deployer: the signing budget

$$n^\star \;\in\; \{1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, \dots, 2^{14} - 1\},$$

i.e., values of $n^\star$ for which $(n^\star{+}1)$ is a power of two. Why this restriction? It's the regime where $(n^\star{+}1)K$ is itself a power of two, which makes the cliff fire *exactly* at signature $n^\star{+}1$. Outside this set the cliff drifts up to $\sim 2(n^\star{+}1)$ while the HORS coverage parameter stays calibrated at $n = n^\star$: a worse user experience for no extra expressive power, so we don't allow it.

Why $K = 16$ deployment-wide? Each signature reveals f at K message-derived points, and K does two cooperating jobs at once. First, it sets the HORS-style coverage rate: a forger on a fresh message must hit a uniform random K-subset of $\{0, \ldots, T-1\}$, succeeding with probability $(nK/T)^K$ per attempt (Theorem 4, Case A); larger K exponentially tightens this bound. Second, it sets the cliff accumulation rate: distinct evaluation pairs pile up at K per signature, so the cliff fires at $\lceil M/K \rceil$ signatures (Theorem 2); larger K pulls the cliff in. The two roles trade off in opposite directions, and $K = 16$ is the sweet spot at the targeted ~ 124-bit security level with $|\mathbb{F}| \approx 2^{256}$; the only other plausible setting was $K = 32$. We fix K deployment-wide rather than leave it as a deployer knob: a mismatched K between signer and verifier produces silently incompatible artifacts, and the flexibility wasn't being used.

Three power-of-two sizes drop out of $n^\star$ and K:

- *Cliff dimension.* The polynomial f has M coefficients with $M := (n^\star{+}1)K$, set so the cliff fires exactly at $n_{\text{cliff}} := M/K = n^\star{+}1$. Because $n^\star{+}1$ is a power of two and so is K, M is automatically a power of two too. We write $M = 2^\nu$, $D := M - 1$.
- *WHIR HVZK budget.* We want zk-WHIR's hiding to cover all $n^\star$ signatures. Each opening leaks at most $\theta = 64$ codeword positions of $\mathsf{enc}(c)$ (the first-round spotchecks, §3.5). So we ask zk-WHIR.Commit for an HVZK budget $B \geq n^\star \cdot \theta$, large enough that Lemma 1's hypothesis is satisfied across the full budget. Concretely, $\nu' := \lceil \log_2(M + n^\star\theta) \rceil$, $N := 2^{\nu'}$, $B := N - M$.
- *Position space.* We need T large enough that even a grinding adversary running $q_H \leq 2^{\lambda_H}$ random-oracle queries can't get a K-subset of T to land in the seen pool with non-negligible probability. The single-attempt coverage probability is $(n^\star K/T)^K$ (Lemma 2); q_H tries multiply this by q_H. To keep the product under $2^{-\lambda}$ we set $T := \mathsf{nextpow2}\big(n^\star K \cdot 2^{(\lambda+\lambda_H)/K}\big)$. At $K = 16$, $\lambda = \lambda_H = 128$, this is $T = \mathsf{nextpow2}(2^{20} n^\star)$.

The position space T must be a power of two dividing $|\mathbb{F}^\times|$;[39] the 2-adicity of $\mathbb{F}^\times$ is 34,[40] so $T \leq 2^{34}$. Combined with the formula above (at $K = 16$, $\lambda = \lambda_H = 128$), this bounds the deployable signing budget by $n^\star \leq 2^{14} - 1$.

4.2 The monomial-basis lift

The lift is the metamorphosis between two worlds' edges: on one side we have f, a polynomial we want to evaluate. On the other side we have zk-WHIR.Open, which only knows how to prove linear-form claims $\langle c, \alpha \rangle = v$. The lift translates the first into the second.

Definition 6 (Monomial-basis lift)**.** For $x \in \mathbb{F}$, the *monomial-basis lift* at x is

$$\mathbf{u}(x) \;:=\; (1, x, x^2, \ldots, x^{M-1}) \in \mathbb{F}^M.$$

The defining identity:

Lemma 3 (Lift identity)**.** *For every $c \in \mathbb{F}^M$ and every $x \in \mathbb{F}$,*

$$\langle c, \mathbf{u}(x) \rangle \;=\; \sum_{k=0}^{M-1} c_k \cdot x^k \;=\; f(x),$$

[39] Because we want T to index a multiplicative subgroup of $\mathbb{F}^\times$, the size T must divide the group order. Powers of two divide the group order up to the 2-*adicity* (the largest k with 2^k dividing the order). Beyond 2-adicity, larger T stops being a subgroup size, so the position-to-field map ψ wouldn't be injective on its intended cycle.

[40] $|\mathbb{F}^\times| = q_0^4 - 1$. Since $q_0 = 2^{64} - 2^{32} + 1 = 2^{32}(2^{32} - 1) + 1$, $q_0 - 1 = 2^{32}(2^{32} - 1)$ contributes $v_2 = 32$ while $q_0 + 1$ and $q_0^2 + 1$ each contribute $v_2 = 1$, so the 2-part of $q_0^4 - 1 = (q_0 - 1)(q_0 + 1)(q_0^2 + 1)$ comes to 2^{34}, capping the largest NTT-friendly subgroup at length 2^{34}.

where $f(X) := \sum_{k=0}^{M-1} c_k X^k$. This is just polynomial evaluation written as a dot product against the powers-of-x vector: multiply it out and the equality is sitting right there.

Two consequences of the lift earn their keep further on:

1. **Cap-binding (§6.1).** Whatever witness gets extracted from root lives in $\mathbb{F}^M$, so $\langle c, \mathbf{u}(x)\rangle$ is always a polynomial of degree at most D in x. The degree cap is baked into the linear-form API itself: the signer can't sneak past it by sleight of hand, no zero-padding excuses.
2. **Hiding (§6.3).** The opening claim $\langle c, \mathbf{u}(x)\rangle = f(x)$ is purely a function of c. The signature's θ codeword peeks into $\mathsf{enc}(c)$ are absorbed by the encoding randomness r_{zk} that zk-WHIR.Commit samples (Prop. 3.19 of [7], extended to many signatures by Lemma 1). Nothing about f leaks beyond the explicit (x_t, y_t) pairs.

Multilinear structure (for the verifier). The verifier has a problem. zk-WHIR's sumcheck wants the value of $\mathbf{u}(x)$ at a random point $r^\star \in \mathbb{F}^\nu$, but $\mathbf{u}(x)$ is an M-vector and materialising it costs $O(M)$ work. We need to skip the materialisation and answer in $O(\log M)$. The way out: $\mathbf{u}(x)$ has a tensor structure. Index its M positions by bit-tuples $b \in \{0,1\}^\nu$ via the LSB-first expansion $k = b_1 + 2b_2 + \cdots + 2^{\nu-1}b_\nu$. Then

$$\mathbf{u}(x)_b \;=\; \prod_{j=1}^{\nu} \Big(1 - b_j + b_j \cdot x^{2^{j-1}}\Big)\,.$$

That right-hand side is multilinear in b, so it agrees with the unique multilinear polynomial extending $\mathbf{u}(x)$[41] on $\mathbb{F}^\nu$ everywhere. Plugging in $r^\star$ instead of b:

$$\mathbf{u}(x)(r^\star) \;=\; \prod_{j=1}^{\nu} \Big(1 - r_j^\star + r_j^\star \cdot x^{2^{j-1}}\Big)\,. \tag{1}$$

ν multiplications of two-term factors. $O(\log M)$ work, as promised.

4.3 The three algorithms

The position-to-field map. Shall we play the ring-around? Let $g \in \mathbb{F}^\times$ be a fixed deployment-wide generator of the unique subgroup of $\mathbb{F}^\times$ of order T, and set $\psi(i) := g^i$ for $i \in \{0, \ldots, T-1\}$, so that as i ranges the image cycles once around the subgroup. The map is deterministic, identical across signers, and injective.

Construction 1 (Jevil.KeyGen)**.** On input $n^\star$:

1. *Sample seed.* $s \xleftarrow{\$} \{0,1\}^{256}$.
2. *Derive coefficients.* Derive the M honest coefficients $c := (c_0, \ldots, c_{M-1}) \in \mathbb{F}^M$ from the extendable-output stream $H_{\mathsf{xof}}(\texttt{JV-SEED}, s; \infty)$, parsing 32-byte chunks as four little-endian Goldilocks limbs with per-limb rejection sampling[42] on raw values $\geq q_0$ (acceptance probability $\approx 1 - 2^{-32}$ per limb).
3. *Commit.* $(\mathsf{root}, \mathsf{st}) \leftarrow \mathsf{zk\text{-}WHIR.Commit}(c, s)$; the secret seed s plays the role of the API's σ_{zk} and feeds the JV-RZK derivation of zk-WHIR.Commit's internal encoding randomness (§3.5, Definition 5).

[41] Given a function $h : \{0,1\}^\nu \to \mathbb{F}$, its *multilinear extension* is the unique polynomial that has degree at most 1 in each variable and matches h on $\{0,1\}^\nu$. The point of these things in sumcheck-based arguments is that a sum over $\{0,1\}^\nu$ collapses to a single field evaluation.

[42] Rejection sampling: read a raw 64-bit value, accept if $< q_0 = 2^{64} - 2^{32} + 1$, otherwise discard and read again. This produces an unbiased uniform sample from $\mathbb{F}_{q_0}$ at the cost of $\approx 2^{-32}$ rejection probability per limb. Without rejection, raw 64-bit values would be biased: there are $2^{32} - 1$ values in $[q_0, 2^{64})$ that would all collapse to the same reduced value modulo q_0.

4. *Derive the OOD binding point.* Sample one uniform $\mathbb{F}$-element z from $H_{\mathsf{xof}}(\texttt{JV-OOD}, \mathsf{root}; \infty)$ via the same chunk-and-reject sampler as JV-SEED.
5. *Compute the OOD value.* $w := f(z) = \langle c, \mathbf{u}(z)\rangle$ via Horner[43] on c.
6. *Output.* $\mathsf{pk} := (\mathsf{root}, w, n^\star)$, $\mathsf{sk} := s$. Cache (c, st) for fast signing.

Everything in the construction (c, the internal encoding randomness, z, w) is a deterministic function of s (via root), so the 32-byte seed is the whole secret key. The OOD pair (z, w) rides along inside pk instead of being recomputed every signature: the verifier re-derives z from root itself (so we don't transmit it) and just reads w off the public key (one extra $\mathbb{F}$-element, 32 B).

Construction 2 (Jevil.Sign)**.** On input $\mathsf{sk}, \mathsf{pk}, \mathcal{M}$:

1. *Load state.* Recover (or load) c and st from s.
2. *Derive positions.* From $H_{\mathsf{xof}}(\texttt{JV-POSN}, \mathsf{root}, \mathcal{M}; \infty)$, sample K distinct indices $(i_1, \ldots, i_K) \in \{0, \ldots, T-1\}$ via partial Fisher–Yates with unbiased modular reduction;[44] sort ascending. Set $x_t := \psi(i_t)$ for $t \in \{1, \ldots, K\}$.
3. *Evaluate.* For each t, compute $y_t := f(x_t)$ by Horner on the coefficients $c = (c_0, \ldots, c_{M-1})$.
4. *Re-derive the OOD point.* Sample z from $H_{\mathsf{xof}}(\texttt{JV-OOD}, \mathsf{root}; \infty)$ identically to KeyGen, so the signer recomputes the same z used to fix w.
5. *Batch.* Initialise a Fiat–Shamir stream

$$\mathsf{tr} \leftarrow H_{\mathsf{xof}}(\texttt{JV-FSCH}, \mathsf{root}, \mathcal{M}, y_1, \ldots, y_K; \infty)$$

   and squeeze $K+1$ challenges $\beta_1, \ldots, \beta_K, \beta_{K+1} \in \mathbb{F}$.
6. *Form the linear form.* Symbolically, set $\alpha := \sum_{t=1}^{K} \beta_t \cdot \mathbf{u}(x_t) + \beta_{K+1} \cdot \mathbf{u}(z) \in \mathbb{F}^M$ and target $v := \sum_{t=1}^{K} \beta_t y_t + \beta_{K+1} w \in \mathbb{F}$. By Lemma 3 applied at each x_t and at z, $\langle c, \alpha\rangle = \sum_t \beta_t f(x_t) + \beta_{K+1} f(z) = v$.[45]
7. *Derive prover randomness.* Set $\rho \leftarrow H_{\mathsf{xof}}(\texttt{JV-OPRD}, s, \mathsf{root}, \mathcal{M}, y_1, \ldots, y_K; \infty)$ and use ρ as the deterministic source for all internal randomness consumed by zk-WHIR.Open (sumcheck round-polynomial masks, code-switching mask oracles, OOD answers); see §3.4.
8. *Open.* $\pi \leftarrow \mathsf{zk\text{-}WHIR.Open}(\mathsf{st}, \alpha, v;\ \rho)$.
9. *Output.* $\sigma_{\mathcal{M}} := ((y_1, \ldots, y_K), \pi)$.

The verifier never materialises $\mathbf{u}(x_t)$: it evaluates the multilinear extension (1) at the single sumcheck-derived point $r^\star$ chosen during WHIR.

Construction 3 (Jevil.Verify)**.** On input $\mathsf{pk} = (\mathsf{root}, w, n^\star)$, $\mathcal{M}$, $\sigma_{\mathcal{M}} = ((y_1, \ldots, y_K), \pi)$:

1. *Re-derive parameters.* Recompute the derived parameters (ν, M, ν', N, T) from $n^\star$ and K.
2. *Re-derive positions.* Recompute $(i_1, \ldots, i_K)$ and $x_t := \psi(i_t)$ exactly as the signer did.

[43] Horner's method evaluates $f(x) = c_0 + c_1 x + \cdots + c_{M-1} x^{M-1}$ in $M-1$ multiplications and $M-1$ additions, by rewriting as $f(x) = c_0 + x(c_1 + x(c_2 + x(\cdots)))$. Optimal for univariate evaluation when only the value (not the polynomial in factored form) is wanted.

[44] The *Fisher–Yates shuffle* samples a uniform random permutation of n elements in $O(n)$ time; the partial version stops after K steps to produce a uniform K-subset without repetition. "Unbiased modular reduction" refers to rejecting raw hash output in the high tail $[\lfloor 2^L/T\rfloor \cdot T, 2^L)$ to avoid biasing $i \bmod T$ toward small values, the same trick as rejection sampling on field limbs.

[45] This is the *Fiat–Shamir batching* trick: rather than open f at $K+1$ points with $K+1$ separate proofs, take a random linear combination of the openings into a single linear form whose value the verifier can also reconstruct. A cheating prover who tries to lie at even one position loses with probability $1 - 1/|\mathbb{F}|$ over the choice of β_t, by Schwartz–Zippel. Costs: one zk-WHIR.Open instead of $K+1$.

3. *Re-derive OOD point.* Recompute z from $H_{\mathsf{xof}}(\texttt{JV-OOD}, \mathsf{root}; \infty)$ identically to KeyGen.
4. *Re-derive challenges.* Recompute $(\beta_1, \ldots, \beta_K, \beta_{K+1})$ exactly as the signer did.
5. *Compute target.* $v := \sum_{t=1}^{K} \beta_t y_t + \beta_{K+1} w$.
6. *Verify.* Accept iff $\mathsf{zk\text{-}WHIR.Verify}(\mathsf{root}, \alpha, v, \pi) = 1$, with α passed symbolically as the tensor $\sum_t \beta_t \mathbf{u}(x_t) + \beta_{K+1}\mathbf{u}(z)$ and evaluated via (1) at zk-WHIR's sumcheck challenge point (the $\mathbf{u}(z)$ term uses the same closed-form $\prod_j (1 - r_j^\star + r_j^\star z^{2^{j-1}})$ as the message-derived points).

Binding the zk-WHIR transcript to $(n^\star, \mathsf{root}, w, \mathcal{M}, (y_t))$. zk-WHIR.Open runs its own Fiat–Shamir transcript internally, which means a π produced under one Jevil tuple is, in principle, a π that could be lifted and reused under a different $(n^\star)', \mathsf{root}', w', \mathcal{M}', (y_t)'$ tuple that happens to land on the same batched linear-form value. To slam that door, both Sign and Verify inject a deterministic prefix

$$\begin{aligned}\mathsf{prefix} \;:=\; & \texttt{JV-OPEN} \,\|\, \mathsf{LE}_4(n^\star) \,\|\, \mathsf{root} \,\|\, \mathsf{enc}(w) \\ & \|\, \mathsf{len}_8(\mathcal{M}) \,\|\, \mathcal{M} \,\|\, \mathsf{enc}(y_1) \,\|\, \cdots \,\|\, \mathsf{enc}(y_K)\end{aligned}$$

into the zk-WHIR transcript's initial state (the "instance bytes" of the underlying Fiat–Shamir construction), where $\mathsf{LE}_4(n^\star)$ is the 4-byte little-endian encoding of $n^\star \in [1, 2^{32})$ and enc is the canonical 32-byte $\mathbb{F}$-element encoding of §3.3. Signer and verifier must compute byte-identical prefix; otherwise zk-WHIR.Verify produces a different challenge sequence and rejects.[46] (The JV-OPEN tag is one of the seven H_{xof} domain tags of §3.4, and the only one that addresses the zk-WHIR FS instance bytes.)

Correctness is immediate from Lemma 3 and zk-WHIR's completeness.

5 The cliff

After signature n_{cliff} a marvellous fun is about to break free: anyone watching the transcript can sit down with a piece of paper and reconstruct f. We state the underlying mathematical fact, specialise to honest Jevil signatures, then deal with the gap between the worst-case bound and what actually happens to an honest signer.

5.1 The polynomial-determination slogan

Theorem 1 (Polynomial determination)**.** *Let $\mathbb{F}$ be a field and $D \in \mathbb{N}$. For any $D{+}1$ distinct points $x_0, \ldots, x_D \in \mathbb{F}$ and any values $y_0, \ldots, y_D \in \mathbb{F}$, there is a unique polynomial $f \in \mathbb{F}[X]$ of degree at most D with $f(x_i) = y_i$ for all i; moreover, f's coefficient vector is computable from $(x_i, y_i)_{i=0}^{D}$ in $O(D^2)$ field operations.*

Proof. The intuition is exact; we just make it precise. *Uniqueness.* If two polynomials of degree at most D agree at the $D{+}1$ distinct points x_i, their difference has degree at most D yet vanishes at $D{+}1$ points, and a nonzero polynomial of degree at most D has at most D roots, so the difference is identically zero and the two coincide. This is exactly why D points leave a degree-D polynomial undetermined (many fit) while the $(D{+}1)$-th pins it down. *Existence.* The Lagrange combination $f(X) = \sum_{i=0}^{D} y_i \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}$ has degree

[46] Without the prefix, a malicious signer could produce a zk-WHIR proof for one set of openings (y_t) and later present it as the proof for a different message $\mathcal{M}'$ with different declared openings (y'_t) that happened to give the same batched linear-form value. The prefix forecloses this: the FS challenges depend on the message and declared evaluations, so a different message produces a different challenge sequence and the original proof rejects.

at most D, and each basis term is 1 at x_i and 0 at every other node, so $f(x_i) = y_i$ for all i. *Cost.* Recovering the coefficient vector, whether by expanding this combination, by solving the (invertible, since the x_i are distinct) Vandermonde system, or by Newton's divided differences, takes $O(D^2)$ field operations. □

5.2 Worst-case cliff

Each Jevil signature reveals K evaluations of f at hash-derived positions. "Hash-derived" means random and unpredictable, but *not* "always fresh"; two signatures can collide on the same position. The worst case for the signer is an adversary who grinds messages until their position sets are pairwise disjoint, packing as many distinct evaluations into the public record as fast as possible.

Theorem 2 (Jevil cliff, honest signer)**.** *Let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Jevil.KeyGen}(1^\lambda, n^\star)$ *and let* $\sigma_1, \ldots, \sigma_n$ *be honest signatures on distinct messages* $\mathcal{M}_1, \ldots, \mathcal{M}_n$ *whose accumulated distinct positions satisfy* $|\bigcup_{i=1}^{n} \{x_t^{(i)}\}_{t=1}^{K}| \geq M$*. (In particular,* $n \geq n_{\text{cliff}} := \lceil M/K \rceil$ *honest signatures with pairwise disjoint position sets meet this bound.) Then there is a deterministic extractor that, given* pk *and* $\{(\mathcal{M}_i, \sigma_i)\}_{i=1}^{n}$*, recovers the honest coefficient vector* $(c_0, \ldots, c_{M-1})$ *in* $O(D^2)$ *field operations with probability* 1*.*

Proof. Every honest signature dumps K (point, value) pairs of f into the public record, and pk tosses in a freebie $(z, f(z))$ from the OOD binding. Once the public stash hits $D{+}1$ distinct pairs, Theorem 1 hands you back c in $O(D^2)$ work. The recipe forces $K \mid M$, so the OOD freebie doesn't actually pull the cliff in; it would have lived inside the next signature's K-pair bucket anyway. □

Corollary 1 (Forgery cliff)**.** *Jevil has an* n_{cliff}*-cliff in the sense of Def. 3 with* $n_{\text{cliff}} = \lceil M/K \rceil$*.*

Proof. You might think recovering c is enough: you know f, just sign with it. The catch is that Sign also needs the WHIR commit state st, which depends on an encoding randomness r_{zk} that Prop. 3.19 hides inside $\mathsf{zk\text{-}WHIR.Commit}$. We never saw r_{zk}. But zk-WHIR's knowledge extractor doesn't just recover c: it recovers (c, r_{zk}) together, except with probability $\leq \epsilon_{\mathsf{zk\text{-}WHIR}}$. Both halves in hand, st is just re-running the public encoding step. Set $\mathsf{sk}^\dagger := (c, r_{\mathsf{zk}})$; the modified signer reading $\mathsf{sk}^\dagger$ produces openings $\mathsf{zk\text{-}WHIR.Verify}$ accepts by completeness. □

5.3 Honest-case timing

$n_{\text{cliff}} = n^\star{+}1$ is the worst case. An honest signer's positions are uniform random K-subsets of a much larger pool, so collisions across signatures are common: the actual count of *distinct* positions after n signatures is strictly less than nK. Heuristically (coupon-collector[47]), for $nK \ll T$ the expected number is $nK(1 - O(nK/T))$, so an honest signer typically reaches the cliff a few signatures later than the bound says. The safe play is to treat n_{cliff} as an upper bound and never count on the slack.

Treat $n^\star$ as a hard budget. The next signature past $n^\star$ doesn't just degrade security, it removes it entirely: the polynomial becomes public, the HORS coverage bound is blown, and every further signature just feeds more Lagrange data to the world. Who keeps spinning the world around? The signer who didn't stop. The HORS coverage parameter

[47] The classical *coupon-collector* problem: drawing uniformly with replacement from T items, the expected number of draws to see m *distinct* coupons is $T \sum_{j=T-m+1}^{T} 1/j \approx T(\ln T - \ln(T-m))$. The complementary form (expected distinct items after d draws) gives $T(1 - (1-1/T)^d) \approx d - O(d^2/T)$ for $d \ll T$, which is what we use here at $d = nK$ draws (K positions per signature, n signatures).

is calibrated for $\sim$ 124-bit classical security *only* at $n = n^\star$. Discipline: never sign past the budget.

6 Security

Four things to prove, in order: cap-binding (§6.1), that even a malicious KeyGen can't dodge the cliff; EUF-CMA below the cliff (§6.2), that signatures up to $n^\star$ are unforgeable; hiding (§6.3), that each signature leaks nothing past its K openings; and post-quantum (§6.4), that all of the above is believed to survive a quantum adversary.

6.1 Cap-binding

Cap-binding: even a malicious KeyGen can't push the cliff past n_{cliff}. The cliff is baked into any public key the verifier ever accepts, not just the honestly-generated ones.

Theorem 3 (Cap-binding)**.** *Let* root *be any commitment produced by a PPT adversary $\mathcal{B}$ making at most q_H random-oracle queries, and let $\sigma_1, \ldots, \sigma_n$ be any signatures on distinct messages $\mathcal{M}_1, \ldots, \mathcal{M}_n$ that $\mathcal{B}$ then causes* $\mathsf{Verify}((\mathsf{root}, w, n^\star), \cdot, \cdot)$ *to accept. Then there exists a polynomial $g \in \mathbb{F}[X]$ of degree at most $D = M - 1$ such that every revealed y-value across the n signatures satisfies $y = g(x)$ for its associated x, except with probability at most $\epsilon_{\mathsf{zk\text{-}WHIR}} + q_H/|\mathbb{F}|$.*

Proof. zk-WHIR's knowledge extractor pulls some $c \in \mathbb{F}^M$ out of the transcript with $\langle c, \alpha_i \rangle = v_i$ for every accepted signature, except with probability $\epsilon_{\mathsf{zk\text{-}WHIR}}$. Set $g(X) := \sum_{k=0}^{M-1} c_k X^k$, a polynomial of degree $\le D$ by inspection. Lemma 3 applied to c gives $\langle c, \mathbf{u}(x) \rangle = g(x)$ for any x, so the WHIR check on the i-th signature rearranges to

$$\sum_{t=1}^{K} \beta_t^{(i)} \cdot \big(y_t^{(i)} - g(x_t^{(i)})\big) \;+\; \beta_{K+1}^{(i)} \cdot \big(w - g(z)\big) \;=\; 0. \tag{2}$$

Define the discrepancy vector $\delta^{(i)} := (y_1^{(i)} - g(x_1^{(i)}), \ldots, y_K^{(i)} - g(x_K^{(i)}), w - g(z)) \in \mathbb{F}^{K+1}$. The theorem says all the openings are evaluations of g, i.e. every $\delta^{(i)} = 0$; we bound the probability of a non-zero $\delta^{(i)}$ slipping past.

Suppose $\delta^{(i)} \neq 0$. Equation (2) reads $\langle \beta^{(i)}, \delta^{(i)} \rangle = 0$ with $\beta^{(i)}$ uniform over $\mathbb{F}^{K+1}$ (it's a fresh `JV-FSCH` oracle output). Schwartz–Zippel[48] on the degree-1 form $\langle \beta, \delta^{(i)} \rangle$ gives a $1/|\mathbb{F}|$ probability per query that this happens. The adversary may grind q_H candidate tuples; union bound, $q_H/|\mathbb{F}|$. Add the extractor's $\epsilon_{\mathsf{zk\text{-}WHIR}}$ failure and you've got the stated bound. □

Whatever an adversarial signer commits to, by Theorem 3, behaves like a polynomial of degree at most D. Compose with Theorem 1 and the answer to "can a malicious signer dodge the cliff?" is: no public key exists that lets them.

Corollary 2 (Adversarial-key cliff)**.** *Let* root *be any commitment produced by a PPT adversary $\mathcal{B}$ making at most q_H random-oracle queries, and let $\{(\mathcal{M}_i, \sigma_i)\}_{i=1}^n$ be any accepted signatures against* $(\mathsf{root}, w, n^\star)$ *whose accumulated distinct positions satisfy $|\bigcup_{i=1}^n \{x_t^{(i)}\}_{t=1}^K| \ge M$ (which holds, in particular, for $n \ge n_{\text{cliff}}$ signatures with pairwise disjoint position sets). Then Jevil meets the adversarial-key cliff of Def. 4 with the* predict *map instantiated as the recovered polynomial,* $\mathsf{predict}(g, x) := g(x)$*; in particular, except with probability at most $\epsilon_{\mathsf{zk\text{-}WHIR}} + q_H/|\mathbb{F}|$:*

[48] Schwartz–Zippel: a non-zero polynomial of total degree d over $\mathbb{F}$ vanishes on at most a $d/|\mathbb{F}|$ fraction of $\mathbb{F}^n$. We use it at $d = 1$ here, so a uniform $\beta^{(i)}$ misses by at least $1 - 1/|\mathbb{F}|$. Most random-challenge soundness arguments in algebraic proof systems collapse to one application of this lemma.

1. *there is a polynomial $g \in \mathbb{F}[X]$ of degree at most D such that every revealed y-value satisfies $y = g(x)$;*
2. *g is publicly recoverable from the transcript in $O(D^2)$ field operations by Lagrange interpolation.*

Proof. (1) is Theorem 3. (2) is Theorem 1 applied to the $\geq D{+}1$ accumulated pairs $(x, g(x))$, consistent with g by (1). □

As anticipated in Remark 1, we do not claim a third party can later produce *accepting* WHIR bundles against an adversarial root: that would need the Merkle-committed vector to match $\mathsf{Enc}_C(c, r_{\mathsf{zk}})$ bit-for-bit, but zk-WHIR's proximity test only certifies δ-closeness. The security content of the cliff is in (1) and (2): f is recovered, so every post-cliff y-value at any message-derived position is publicly computable, no matter who builds the surrounding proof.

6.2 EUF-CMA security below the cliff

Cap-binding tells us no public key lets a signer evade the cliff. Now we prove the complementary statement: as long as the signer stays inside the budget, no adversary can forge.

Theorem 4 (EUF-CMA-$n^\star$ for Jevil)**.** *For any PPT adversary $\mathcal{A}$ making at most $n^\star$ signing queries and q_H random-oracle queries against Jevil with parameters as in §4.1,*

$$\mathsf{Adv}^{\textit{EUF-CMA-}n^\star}_{\mathsf{Jevil},\mathcal{A}}(\lambda) \;\leq\; \epsilon_{\mathsf{zk\text{-}WHIR}} \;+\; \underbrace{q_H \cdot \left(\frac{n^\star K}{T}\right)^K}_{\textit{HORS coverage (Case A)}} \;+\; \underbrace{\frac{q_H}{|\mathbb{F}|}}_{\substack{\textit{Schwartz–Zippel}\\ \textit{on JV-FSCH}}} \;+\; \underbrace{\frac{q_H}{|\mathbb{F}|}}_{\substack{\textit{fresh-evaluation}\\ \textit{guessing (Case B)}}}. \tag{3}$$

Plug in the reference parameters ($K = 16$, $|\mathbb{F}| \approx 2^{256}$, T chosen with $\lambda = \lambda_H = 128$): the WHIR term gives $\epsilon_{\mathsf{zk\text{-}WHIR}} \leq R \cdot 2^{-128}$ with $R \leq 8$; the HORS term gives $\leq 2^{-128}$ at $q_H \leq 2^{128}$; the two field-size terms give $q_H/|\mathbb{F}| \leq 2^{-128}$ each. Summing,

$$\mathsf{Adv}^{\textit{EUF-CMA-}n^\star}_{\mathsf{Jevil},\mathcal{A}}(\lambda) \;\leq\; (R+3)\cdot 2^{-128} \;\leq\; 11 \cdot 2^{-128} \;<\; 2^{-124}.$$

This represents $\gtrsim 124$-bit classical EUF-CMA-$n^\star$ security against $q_H \leq 2^{128}$ adversaries at the worst-case $n^\star = 16{,}383$; at the small-$n^\star$ end ($R = 1$ at $n^\star = 1$) the bound tightens to $4 \cdot 2^{-128} = 2^{-126} < 2^{-125}$. Reducing the budgeted q_H tightens the field-size and HORS terms proportionally.

Proof. $\mathcal{A}$ makes $q \leq n^\star$ signing queries on $\mathcal{M}_1, \ldots, \mathcal{M}_q$ and at most q_H random-oracle queries, then submits a forgery $(\mathcal{M}^*, \sigma^* = ((y_t^*)_{t=1}^K, \pi^*))$ on a fresh $\mathcal{M}^* \notin \{\mathcal{M}_i\}$. There are only three ways this forgery can succeed:

- *They lied* (E_1): some y_t^* doesn't actually equal $f(x_t^*)$, but the WHIR proof was somehow fooled into accepting.
- *They got lucky on positions* (E_2): all K message-derived positions x_t^* already appear in the seen pool, so they could quote those evaluations honestly.
- *They guessed an unseen evaluation* (E_3): at least one position is fresh, so they had to guess f's value there.

These are pairwise disjoint. Bound each, sum.

Event E_1 (zk-WHIR / Schwartz–Zippel). The forgery is accepted, but $y_t^* \neq f(x_t^*)$ for some t. By Theorem 3 applied to the honestly generated root (whose extractor returns

the coefficient vector c exactly), the entire forgery acceptance event under root is governed by a Schwartz–Zippel test on $\delta^* := (y_1^* - f(x_1^*), \ldots, y_K^* - f(x_K^*))$. The theorem gives

$$\Pr[E_1] \;\le\; \epsilon_{\mathsf{zk\text{-}WHIR}} + q_H/|\mathbb{F}|.$$

Event E_2 (HORS coverage, Case A). $\neg E_1$: every $y_t^* = f(x_t^*)$, so the forgery is just quoting evaluations of f honestly. For that to work without ever signing $\mathcal{M}^*$, the forger needs every position x_t^* to already lie in the seen pool $S := \bigcup_i \{x_t^{(i)}\}_t$. But $\mathcal{M}^*$ is fresh, so its positions are a uniform random K-subset of $\{0, \ldots, T-1\}$ independent of S. Lemma 2 bounds the per-attempt coverage probability at $(n^\star K/T)^K$, and grinding q_H fresh JV-POSN queries multiplies this by q_H:

$$\Pr[E_2] \;\le\; q_H \cdot (n^\star K/T)^K.$$

The parameter recipe of §4.1 sets T so this is $\le 2^{-128}$ at $\lambda = \lambda_H = 128$, $q_H \le 2^{128}$.

Event E_3 (fresh-evaluation guessing, Case B). $\neg E_1 \wedge \neg E_2$: $y_t^* = f(x_t^*)$ for all t, and at least one $x_{t_0}^* \notin S \cup \{z\}$. The public evaluations of f (the qK past openings plus the OOD pair) total at most $n^\star K + 1 \le D$, since $M = (n^\star{+}1)K$ by construction. By Theorem 1, $f(x_{t_0}^*)$ is then uniform over $\mathbb{F}$ conditioned on those evaluations. The match probability is $1/|\mathbb{F}|$ per JV-POSN attempt, over q_H attempts:

$$\Pr[E_3] \;\le\; q_H/|\mathbb{F}|.$$

Summing the three disjoint events gives (3). □

Multi-target setting. With N_{keys} honest signers deployed in the wild, the adversary doesn't have to target a specific one: they can grind a single message and win if its positions happen to lie in *any* signer's seen pool. Position derivation absorbs root, so position sets across keys are independent in the random-oracle model; a union bound gives per-query advantage $N_{\text{keys}} \cdot (n^\star K/T)^K$, total $q_H N_{\text{keys}} (n^\star K/T)^K$. To keep 128-bit security at population scale, swap the exponent $\lambda + \lambda_H$ in T for $\lambda + \lambda_H + \log_2 N_{\text{keys}}$. This bumps T up, eating into the 2^{34} 2-adicity ceiling, something to watch.

6.3 Hiding

Each Jevil signature visibly hands out K evaluations of f. The claim of this subsection is that it hands out nothing else.

Theorem 5 (Hiding across the full signing budget)**.** *There is a PPT simulator $\mathcal{S}$ such that, for any sequence of messages $\mathcal{M}_1, \ldots, \mathcal{M}_{n^\star}$ and any honestly generated $(\mathsf{pk}, \mathsf{sk})$, the joint distribution of $\{\mathcal{S}(\mathsf{pk}, \mathcal{M}_i, (y_t^{(i)})_{t=1}^K)\}_{i=1}^{n^\star}$ is statistically indistinguishable from $\{\mathsf{Jevil.Sign}(\mathsf{sk}, \mathsf{pk}, \mathcal{M}_i)\}_{i=1}^{n^\star}$, with simulation error 0 for the Reed–Solomon instantiation.*

Proof. A signature is just $((y_t^{(i)})_t, \pi_i)$. The $y_t^{(i)}$ are handed to $\mathcal{S}$ as input. Everything else π_i depends on (the positions $x_t^{(i)}$, the OOD point z, the batching coefficients $\beta_t^{(i)}$) is a deterministic random-oracle function of public data $(\mathsf{root}, \mathcal{M}_i, (y_t^{(i)}))$ that anyone (including $\mathcal{S}$) can recompute. So $\mathcal{S}$ can build (α_i, v_i) on its own, and by Lemma 3 they satisfy $\langle c, \alpha_i \rangle = v_i$.

That leaves π_i, which $\mathcal{S}$ asks Lemma 1's simulator to produce. The parameter recipe sets $B \ge n^\star \cdot \theta$, so the lemma's hypothesis holds at $n = n^\star$. Define $\mathcal{S}(\mathsf{pk}, \mathcal{M}_i, (y_t^{(i)})_t) := ((y_t^{(i)})_t, \tilde{\pi}_i)$ where $\tilde{\pi}_i$ comes from the lemma; for Reed–Solomon the joint distribution matches the real one exactly. □

Two pieces are doing different jobs here. The lift identity (Lemma 3) keeps the opening claim $\langle c, \alpha \rangle = v$ a function of c alone, so the verifier is checking f's evaluations and nothing else. The encoding randomness r_{zk}, hidden inside zk-WHIR.Commit, absorbs the θ codeword positions of $\mathsf{enc}(c)$ that the transcript exposes per signature (Lemma 1). The same length-M constraint vector that hides f also *caps* its degree at D, which is what cap-binding needs.

What w leaks. $w = f(z) \in \mathsf{pk}$ is one public evaluation of f, no more and no less. To an adversary who hasn't yet queried Sign, w looks uniform in $\mathbb{F}$ (the high-degree term $c_{M-1} z^{M-1}$ is uniform whenever $z \neq 0$, which fails only with prob 2^{-256}), so pk is indistinguishable from $(\mathsf{root}, \text{uniform } \mathbb{F}\text{-element}, n^\star)$. The constraint matters only once K more evaluations per signature pile on top: that's where the $+1$ in the cliff theorem's $nK + 1 \geq M$ comes from. And the OOD point z does lie outside the codeword evaluation domain (size $4N \leq 2^{34}$) with probability $\geq 1 - 2^{-222}$, which is what makes "out-of-domain" a fair name for it.

6.4 Post-quantum analysis

All primitives are believed to be post-quantum (no Shor break applies).[49] At Poseidon2 capacity $c = 4$ Goldilocks elements, the relevant hash bounds are: second-preimage $\sim 2^{256}$ classical / $\sim 2^{128}$ quantum (Grover[50]), collision $\sim 2^{128}$ classical / $\sim 2^{85}$ quantum (BHT [12]); SHAKE256 with 256-bit output matches. The Reed–Solomon proximity-gap conjecture [9] underlies zk-WHIR's soundness; its quantum-query soundness is conjectured to lose at most a polynomial factor versus the classical bound.

Three quantum-attack vectors interact with the analysis:

Hash collision floor (commitment binding). zk-WHIR's vector commitment binds via collision resistance of Poseidon2. At $c = 4$ the BHT bound caps quantum binding at $\sim 2^{85}$ work. Raising the capacity to $c = 6$ Goldilocks elements (384-bit capacity, $\sim 2^{128}$ quantum collision) lifts the floor to 2^{128}; the rest of the construction is unchanged.

Quantum HORS coverage attack. The classical Case A bound reads $q_H \cdot (n^\star K/T)^K$: every random-oracle query is one fresh attempt, each succeeds with probability $(n^\star K/T)^K$. Under QROM,[51] a Grover-style search amplifies this: q_H queries effectively probe q_H^2 attempts, so the bound becomes $\sim q_H^2 \cdot (n^\star K/T)^K$ by Zhandry's compressed-oracle technique.[52] You've lost a factor of q_H in the margin.

Two ways to put it back. Either bump T up by replacing the $(\lambda + \lambda_H)/K$ exponent with $(\lambda + 2\lambda_H)/K$, which at $\lambda = \lambda_H = 128$, $K = 16$ caps the deployable budget at $n^\star \leq 63$ (the largest recommended-regime value that still fits under the 2^{34} 2-adicity ceiling). Or accept the NIST-style PQ convention that a quantum adversary makes at most $q_H \leq 2^{64}$

[49] *Shor's algorithm* solves integer factoring and the discrete logarithm problem in polynomial quantum time. Schemes whose security reduces to either problem (RSA, ECDSA, Schnorr, BLS, KZG, IPA) are catastrophically broken by a sufficiently large quantum computer. Hash-based and lattice-based schemes (Jevil, SLH-DSA, FALCON, ML-DSA) avoid Shor by not resting on either assumption.

[50] *Grover's algorithm* finds a marked element in an N-sized unstructured search space in $O(\sqrt{N})$ quantum queries, halving the bit-security of any search-based attack. A 256-bit pre-image, classically 2^{256} work, becomes 2^{128} quantumly. The square-root speedup is tight for unstructured search but does not apply to attacks with algebraic structure (e.g., Shor on factoring is exponentially faster than Grover-on-factoring).

[51] The *Quantum Random Oracle Model* (QROM) is the natural quantum-accessible version of ROM: the adversary queries the random oracle in superposition, $\sum_x \alpha_x |x\rangle \mapsto \sum_x \alpha_x |x, H(x)\rangle$ in one query. Many ROM proofs need new techniques to lift to QROM (compressed-oracle, one-way-to-hiding lemma); collision-finding and pre-image search get the expected Grover/BHT speedups.

[52] Zhandry's *compressed-oracle* technique gives tight quantum-query bounds for properties of random functions (collisions, multi-collisions, pre-images, "hard" relations) in the QROM. The classical bound $q_H \cdot p$ for any single-query success probability p becomes $\sim q_H^2 \cdot p$ in QROM, with the extra q_H absorbing Grover's quadratic speedup.

queries; the classical parameters of §4.1 then deliver $\sim$ 128-bit quantum HORS security across the full deployable range.

Fresh-evaluation guess. The $1/|\mathbb{F}|$ guessing bound of Case B does not benefit from Grover: a forger who knows the correct $y^*_{t_0}$ must still produce a matching WHIR opening, which by knowledge soundness requires the witness c. The relevant attack reduces to breaking knowledge soundness, bounded by $\epsilon_{\mathsf{zk\text{-}WHIR}}$. The $1/|\mathbb{F}|$ term therefore remains the right concrete bound in QROM, modulo the conjectural quantum soundness of zk-WHIR.

Summary. For full 128-bit quantum security: raise Poseidon2 capacity to $c = 6$, and either accept $q_H \leq 2^{64}$ quantum oracle queries (keeping the existing parameters, the typical PQ-target convention) or enforce $n^\star \leq 63$ with the rebalanced T exponent (for worst-case threat models).

7 Concrete parameters and performance

The construction is fast, fast where it can be (cheap signing on top of a cached commitment, log-sized openings, and the lift's tensor structure that keeps verification at $O(K\nu)$ field ops regardless of M), and clever, clever where it must be (batching K message-derived openings into a single zk-WHIR linear-form proof via Fiat–Shamir). The two tables in this section close the loop with §4.1: Table 2 gives the per-deployment derivation as a single sheet; Table 3 instantiates it at three representative budgets.

7.1 The recipe in one table

The deployer picks $n^\star$; the rest follows from §4.1 as summarised in Table 2.

Table 2: Per-deployment derivation, costs, and artifact sizes. Costs are expressed in field operations and Poseidon2 permutation invocations; sizes are in bytes.

Item	Formula
Cliff dim.	$M = 2^\nu$, $\nu = \lceil \log_2((n^\star{+}1)K) \rceil$, $D = M - 1$
Position space	$T = \mathsf{nextpow2}\big(n^\star K \cdot 2^{(\lambda+\lambda_H)/K}\big)$; for $K = 16$ and $\lambda = \lambda_H = 128$, $T = \mathsf{nextpow2}(2^{20}\, n^\star)$
KeyGen cost	one zk-WHIR.Commit on an M-vector: $\approx M$ Poseidon2 invocations + an $O(M \log M)$ NTT
Sign cost	$K \cdot D$ extension-field multiplications (Horner) + one zk-WHIR.Open ($\approx \frac{8}{3}M$ Poseidon2 invocations) with st cached
Verify cost	one zk-WHIR.Verify + $K\nu$ ops to evaluate α at $r^\star$ via (1)
Per-sig. size	$K \cdot 32$ B for (y_t) + one zk-WHIR proof ($\sim$ 40 KB at $n^\star = 1$, $\sim$ 337 KB at $n^\star = 1023$)
Public-key size	32-B zk-WHIR root + 32-B OOD value w + 4-B $n^\star$ = 68 B
Secret-key size	32 B (the seed s)

7.2 Sample sizes at $K = 16$

Table 3 reads off the recipe at three representative signing budgets.
The largest deployable signing budget, $n^\star = 2^{14} - 1$, is set by the requirement that the position-space size $T \leq 2^{34}$ saturate the working field's 2-adicity ceiling (taken together with the recommended-regime restriction $n^\star{+}1 \in \{2, 4, 8, \ldots\}$). Signature sizes grow only

Table 3: Sample sizes and wall-clock timings at $K = 16$, measured with the reference implementation (§8) on a single core of an Apple M4 (`RAYON_NUM_THREADS=1`); Sign uses a cached st. The $\sim$ symbols absorb run-to-run variance, including the signature-size jitter induced by Merkle-multiproof pruning. Enabling multi-threading parallelises KeyGen and Sign across cores for a further several-fold speedup.

$n^\star$	M	T	KeyGen	Sign	Verify	Sig
1	2^5	2^{20}	$\sim$0.001 s	$\sim$0.01 s	$\sim$0.001 s	$\sim$40 KB
127	2^{11}	2^{27}	$\sim$0.05 s	$\sim$0.06 s	$\sim$0.02 s	$\sim$224 KB
1023	2^{14}	2^{30}	$\sim$0.4 s	$\sim$0.2 s	$\sim$0.1 s	$\sim$337 KB

modestly with $n^\star$: a $\sim 8\times$ size increase across a $\sim 1000\times$ jump in $n^\star$ ($1 \to 1023$), because M grows as $n^\star$ does but the per-signature zk-WHIR proof grows only logarithmically in M. Public keys are exactly 68 B uniformly across all rows: the 32-B zk-WHIR root, the 32-B OOD value w from §4.3 (binding the signer's polynomial directly from pk), and the 4-B little-endian $n^\star$. Sign with a cached commitment state and Verify both run well under a second across the deployable range.

7.3 Reference configuration

If you want a concrete pick: $n^\star = 1023$ (a typical mid-range budget). Then $M = 2^{14}$, $T = 2^{30}$, $D = 16{,}383$. KeyGen takes about 0.4 s. Each signature is $\sim$ 337 KB; Sign runs in about 0.2 s and Verify in about 0.1 s with the commitment state cached. Sign 1023 times. The next one publishes f: finds the key, sets it free, free.

8 Discussion

Comparison with stateful HBS. Stateful schemes (XMSS, LMS)[53] cap signatures with a counter, and state-sync failures are silently catastrophic: a rolled-back counter re-uses key material and breaks unforgeability with no outward sign. Jevil also asks the signer to stop at $n^\star$ (that discipline does not disappear), but it relocates the *consequence* of failing to do so into the cryptographic object: a Jevil signer who oversigns publishes the polynomial. The difference is accountability, not the absence of a cap to respect. Where a stateful over-sign fails silently and in the signer's favour (they keep producing accepting signatures), a Jevil over-sign fails loudly and against the signer (the key becomes public), so over-use is detectable by any observer and cannot be undone by restoring a backup.

Comparison with HORS-family schemes. HORS gets a stateless cap from t-subset coverage; security degrades gradually as $(nK/T)^K$. Jevil keeps the HORS coverage analysis intact below the cap and bolts a polynomial-recovery cliff on top. The cliff makes over-signing visibly disastrous: if the signer oversigns, the world reconstructs the polynomial in $O(D^2)$, and there is no damage control left to do.

The size cost of the cliff. Table 4 makes the trade-off explicit. Jevil's signatures are one to three orders of magnitude larger than the soft-degrading hash-based schemes

[53] *XMSS* (eXtended Merkle Signature Scheme) and *LMS* (Leighton–Micali Signature) are standardised stateful hash-based signature schemes (RFC 8391, RFC 8554). Each signs by consuming a fresh leaf of a Merkle tree of one-time-signature keys; the signer must persistently track which leaf was last used. Standardised because they avoid the soft-degradation curve of FORS, but only at the cost of mandating durable counter state.

Table 4: Sizes and assumption profiles at the ∼128-bit (NIST level 1) classical level for representative few-time / hash-based schemes. SLH-DSA and FORS byte counts are exact (FIPS 205 [5]; SPHINCS$^+$ [4]). Entries marked † are *derived* from the published construction and standard element sizes (the cited works report timings or asymptotics rather than byte counts [25, 2]) and are order-of-magnitude only. The HORST row is the SPHINCS instance (n=256, t=2^{16}, k=32); HORS-family sizes are otherwise parameter-dependent. Jevil sizes are measured (Table 3). "Transp." = transparent (no trusted setup).

Scheme	Sig. size	PK	Degradation	Trigger	PQ	Transp.
HORST (SPHINCS FTS)	∼16 KB†	32 B	soft slope	count	✓	✓
FORS (SLH-DSA L1, s/f)	2.9/3.7 KB	16 B	soft slope	count	✓	✓
SLH-DSA-128s / 128f	7.9/17.1 KB	32 B	soft slope	count	✓	✓
$\mathsf{DSKE}_{\mathsf{poly}}$ (KZG)	∼80 B†	48 B†	**sharp cliff**	count	—	—
$\mathsf{DSKE}_{\mathsf{hash}}$	∼2 KB†	∼4 KB†	soft slope	count	✓	✓
Jevil (this work)	40–337 KB	68 B	**sharp cliff**	count	✓	✓

(FORS, SLH-DSA), and dramatically larger than $\mathsf{DSKE}_{\mathsf{poly}}$, whose KZG openings are constant-size at a few dozen bytes. That gap is precisely the price of the design point: $\mathsf{DSKE}_{\mathsf{poly}}$ buys its tiny signatures with a trusted setup and a discrete-log assumption (Shor-broken), whereas Jevil's hash-based commitment (zk-WHIR, in the FRI/STIR lineage) is simultaneously post-quantum and transparent but inherits the FRI-family proof size of tens-to-hundreds of kilobytes. For Jevil's target use (a small number of high-value, accountability-bearing signatures, §1), bandwidth is rarely the binding constraint, so we take that trade deliberately. Jevil is not competitive on size with SLH-DSA for general high-volume signing, and is not intended to be.

Reference implementation. An open-source reference implementation of Jevil in Rust is available at `https://github.com/symbolicsoft/jevil`.

Acknowledgements

Designing Jevil was a substantial personal and pedagogical undertaking, growing out of a year-long effort to deepen my familiarity with number-theoretic cryptography and, more recently, to understand the role of polynomial-based constructions in modern cryptographic design. To put it plainly: a great deal had to be learned from scratch, and many of the constructions and proof techniques resisted understanding for quite some time. Translating the design into a practical, implementable scheme with a reasonable performance profile required further weeks of iteration. I sincerely thank Giacomo Fenzi for his clarifications on zk-WHIR, most usefully his `tiny_whir` crate which helped me better understand how the protocol worked in practice, and for the suggestion of an OOD-based binding mechanism in KeyGen now incorporated into Constructions 1–3. I also thank Arantxa Zapico and Markulf Kohlweiss for fruitful conversations at Eurocrypt 2026 that meaningfully shaped Jevil's development, and the zk-WHIR authors and the broader Reed–Solomon-proximity community for the polynomial-commitment primitive on which Jevil rests. Most of all, I would like to thank my applied cryptography students at the American University of Beirut for motivating me so strongly to improve my knowledge across all areas of cryptography so as to better serve them in my pedagogical mission. Finally, I thank Toby Fox for *Deltarune* and for the character whose taste for sudden liberation gave this scheme its name.

A Tables

Table 5: Principal symbols. Forward references point to the section in which each symbol is defined or fixed; the reader can return here while reading later sections.

Symbol	Defined in	Meaning
Security parameters and the adversary		
λ	–	Classical security parameter (set to 128)
λ_H	§4.1	RO-query bound budgeted into HORS coverage (set to 128)
q_H	§6.2	Adversary's random-oracle query budget
$\mathsf{negl}(\lambda)$	§3.1	A function negligible in λ
$\mathsf{Adv}^{G}_{\Pi,\mathcal{A}}$	Def. 2	Adversary $\mathcal{A}$'s advantage in game G against scheme Π
Jevil scheme parameters		
$n^\star$	§4.1	Signing budget (input to KeyGen); in $\{1, 3, 7, \ldots, 2^{14}-1\}$
n_{cliff}	§5	First signature index past the cliff $= n^\star + 1$
K	§4.1	Positions per signature (fixed at 16)
T	§4.1	Position-space size; power of two dividing $\lvert\mathbb{F}^\times\rvert$
M	§4.1	Polynomial dimension $= (n^\star+1)K$; equals $\deg f + 1$
D	§4.1	Polynomial degree $= M - 1$
zk-WHIR primitive parameters		
N	§3.5	WHIR-internal ZK-encoded message length, smallest power of $2 \geq M + B$
B	§3.5	WHIR HVZK budget $= N - M$
θ	§3.5	In-domain spotchecks per round (fixed at 64)
η	§3.5	Out-of-domain samples per round (fixed at 2)
R	§3.5	Number of code-switching rounds
ρ	§3.5	Main-code Reed–Solomon rate (fixed at 1/4)
κ	§3.5	Folding/interleaving factor $= 2^k = 4$
Field and primitives		
$\mathbb{F}$	§3.3	Working field $\mathbb{F}_{q_0^4}$, $\lvert\mathbb{F}\rvert \approx 2^{256}$
q_0	§3.3	Goldilocks prime $= 2^{64} - 2^{32} + 1$
H_{xof}	§3.4	Domain-separated SHAKE256 XOF
H_{VC}	§3.4	Poseidon2-based vector-commitment hash
H_{fs}	§3.4	SHAKE128 duplex sponge (used inside zk-WHIR's Fiat–Shamir)
Cryptographic objects		
s	Constr. 1	32-byte secret seed (the entire sk)
f, c	§4.2	Secret polynomial; its coefficient vector $\in \mathbb{F}^M$
root	§3.5	zk-WHIR commitment digest (32 bytes)
(z, w)	§4.3	OOD binding point and value $w = f(z) \in$ pk
(x_t, y_t)	§4.3	The K revealed position–evaluation pairs per signature
(β_t)	§4.3	Fiat–Shamir batching coefficients $\in \mathbb{F}$
α, v, π	Def. 5	zk-WHIR linear form, target value, opening proof
$\mathbf{u}(x)$	§4.2	Monomial-basis lift $(1, x, \ldots, x^{M-1}) \in \mathbb{F}^M$

Table 6: Domain-separation tags. Tags addressing H_{xof} are read as: "the consumer feeds these inputs (in the canonical encoding below) and reads the resulting XOF stream for the stated role."

Tag	Consumer	Inputs	Role
`JV-SEED`	KeyGen	s	Expand secret seed into coefficient vector $c \in \mathbb{F}^M$
`JV-RZK`	zk-WHIR.Commit	σ_{zk}	Derive Prop. 3.19 encoding randomness r_{zk}
`JV-POSN`	Sign, Verify	$(\mathsf{root}, \mathcal{M})$	Derive K message-derived positions $(i_1, \ldots, i_K)$
`JV-FSCH`	Sign, Verify	$(\mathsf{root}, \mathcal{M}, y_1, \ldots, y_K)$	Squeeze Fiat–Shamir batching coefficients (β_t)
`JV-OPEN`	Sign, Verify	$(n^\star, \mathsf{root}, w, \mathcal{M}, y_1, \ldots, y_K)$	Prefix bytes binding the zk-WHIR Fiat–Shamir instance state
`JV-OPRD`	Sign	$(s, \mathsf{root}, \mathcal{M}, y_1, \ldots, y_K)$	Deterministic prover randomness ρ for zk-WHIR.Open
`JV-OOD`	KeyGen, Sign, Verify	root	Derive OOD binding point $z \in \mathbb{F}$
`JV-WHIR`	H_{VC}	—	Capacity-IV constituent; second IV slot separates leaf vs internal-node hashes

Table 7: zk-WHIR deployment parameters for Jevil's Reed–Solomon instantiation of [7, Corollary 3]. All values are fixed deployment-wide.

Parameter	Symbol	Value	Role
Main code C (commits the secret coefficient vector c)			
Reed–Solomon rate	ρ	$1/4$	Per-round in-domain soundness
Evaluation-domain size	–	$4N$	Multiplicative subgroup of $\mathbb{F}^{\times}$
ZK encoding randomness length	t	$N - M$	Caller-driven via σ_{zk}
Folding and interleaving			
Folding parameter	k	2	–
Interleaving factor	$\kappa = 2^k$	4	Per-round message-length reduction
Folding floor	–	$2^6 = 64$	Folding halts
Code-switching rounds	R	$\lceil \log_4(N/2^6) \rceil \leq 8$	–
Per-round queries			
In-domain spotchecks	θ	64	Per-round soundness $(1-\delta)^{\theta}$
Out-of-domain samples	η	2	Proximity-amplification (DEEP)
Mask code C_{zk} (sumcheck and code-switching masks)			
Reed–Solomon rate	ρ_{zk}	$1/16$	Tighter than C to halve spotchecks
Total inner-message length	ℓ_{zk}	$64 = 32{+}32$	Data + Prop. 3.19 randomness
Per-mask spotchecks	θ_{mask}	32	Per-mask soundness ρ_{zk}^{32}
ZK query threshold	t_{zk}	32	$\geq \theta_{\text{mask}}$ for perfect ZK
Evaluation-domain size	m_{zk}	1024	NTT-friendly subgroup

References

[1] A. Perrig. *The BiBa One-Time Signature and Broadcast Authentication Protocol.* ACM CCS 2001.

[2] L. Reyzin and N. Reyzin. *Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying.* ACISP 2002.

[3] J.-P. Aumasson and G. Endignoux. *Improving Stateless Hash-Based Signatures.* CT-RSA 2018.

[4] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe. *The SPHINCS$^+$ Signature Framework.* ACM CCS 2019.

[5] NIST. *Stateless Hash-Based Digital Signature Standard (SLH-DSA).* FIPS PUB 205, 2024.

[6] L. Grassi, D. Khovratovich, and M. Schofnegger. *Poseidon2: A Faster Version of the Poseidon Hash Function.* AFRICACRYPT 2023.

[7] A. Chiesa, G. Fenzi, and G. Weissenberg. *Zero-Knowledge IOPPs for Constrained Interleaved Codes.* Cryptology ePrint Archive, Paper 2026/391, 2026.

[8] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. *STIR: Reed–Solomon Proximity Testing with Fewer Queries.* CRYPTO 2024.

[9] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. *DEEP-FRI: Sampling Outside the Box Improves Soundness.* ITCS 2020.

[10] Polygon Zero Team. *Plonky2: Fast Recursive Arguments with PLONK and FRI.* Technical report, September 2022.

[11] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* FIPS PUB 202, 2015.

[12] G. Brassard, P. Høyer, and A. Tapp. *Quantum Cryptanalysis of Hash and Claw-Free Functions.* LATIN 1998.

[13] J. Lee and Y. Park. *HORSIC+: An Efficient Post-Quantum Few-Time Signature Scheme.* Applied Sciences 11(16):7350, 2021.

[14] L. Li, X. Lu, and K. Wang. *eBiBa: A Post-Quantum Hash-Based Signature With Small Signature Size in the Continuous Communication of Large-Scale Data.* The Computer Journal 67(4):1405–1424, 2024.

[15] K. Algazy, K. Sakan, S. Nyssanbayeva, and O. Lizunov. *Syrga2: Post-Quantum Hash-Based Signature Scheme.* Computation 12(6):125, 2024.

[16] D. Chaum. *Blind Signatures for Untraceable Payments.* CRYPTO 1982, pp. 199–203, Plenum Press, 1983.

[17] D. Chaum, A. Fiat, and M. Naor. *Untraceable Electronic Cash.* CRYPTO 1988, LNCS 403, pp. 319–327, Springer, 1990.

[18] S. Canard and A. Gouget. *Divisible E-Cash Systems Can Be Truly Anonymous.* EUROCRYPT 2007, LNCS 4515, pp. 482–497, Springer.

[19] A. Kate, G. M. Zaverucha, and I. Goldberg. *Constant-Size Commitments to Polynomials and Their Applications.* ASIACRYPT 2010, LNCS 6477, pp. 177–194, Springer.

[20] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. *Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting.* EUROCRYPT 2016, LNCS 9666, pp. 327–357, Springer.

[21] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. *Bulletproofs: Short Proofs for Confidential Transactions and More.* IEEE S&P 2018, pp. 315–334.

[22] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. *Fast Reed–Solomon Interactive Oracle Proofs of Proximity.* ICALP 2018, LIPIcs 107, pp. 14:1–14:17.

[23] A. Shamir. *How to Share a Secret.* Communications of the ACM 22(11):612–613, 1979.

[24] T. P. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing.* CRYPTO 1991, LNCS 576, pp. 129–140, Springer.

[25] Z. Wang, O. Alpos, A. Kavousi, H. W. H. Wong, S. Y. Chau, D. V. Le, and C. Cachin. *DSKE: Digital Signatures with Key Extraction.* CT-RSA 2025. Cryptology ePrint Archive, Paper 2022/1753.

[26] B. Poettering and D. Stebila. *Double-Authentication-Preventing Signatures.* ESORICS 2014, LNCS 8712, pp. 436–453, Springer.

[27] M. Bellare, B. Poettering, and D. Stebila. *Deterring Certificate Subversion: Efficient Double-Authentication-Preventing Signatures.* PKC 2017, LNCS 10175, pp. 121–151, Springer.

[28] D. Boneh, S. Kim, and V. Nikolaenko. *Lattice-Based DAPS and Generalizations: Self-Enforcement in Signature Schemes.* ACNS 2017, LNCS 10355, pp. 457–477, Springer.