

A Universal Forgery Attack on the Origami Signature Scheme from the Public Key Alone

Hugo Louiso¹ Hao Guo² Peigen Li³ Pierre Pébereau⁴ Siyong Tao² Jintai Ding¹

¹Xi'an Jiaotong-Liverpool University, Suzhou, China

²Tsinghua University, Beijing, China

³Beijing Institute of Mathematical Sciences and Applications, Beijing, China

⁴COSIC, KU Leuven, Belgium

Hugo.Louiso26@student.xjtlu.edu.cn guoh22@mails.tsinghua.edu.cn lpg22@bimsa.cn

pierre.pebereau@esat.kuleuven.be taosy22@mails.tsinghua.edu.cn jintai.ding@gmail.com

October 2026

Abstract

Origami is a multivariate signature scheme submitted to the NGCC round-1 public-key call. We show that the submitted bilinear construction is a Rainbow-type scheme whose central map is exposed. The verification map is the layered signing map in a public coordinate order. The public key therefore gives an equivalent secret key, allowing signature forgery at about the cost of one verification (about 0.01 s at Origami-128). Our attack exploits this public layered structure and succeeds against the unmodified reference implementation on all four parameter sets and the official test vectors. We also refute the security proof's inversion assumption in a bilinear independent-coefficient model. The hidden local algebras introduced by Origami do not prevent the attack. They raise a separate problem: the specification treats constrained matrix coordinates as independent, and honest signatures lie in a proper subspace.

1 Introduction

Post-quantum standardization programmes evaluate signature schemes through successive rounds of public review. The NIST process [8] began in 2016 with 69 complete round-1 submissions, retained 26 for the second round and 15 for the third, and led to the standards FIPS 203–205 in 2024 [9].

China's Institute of Commercial Cryptography Standards (ICCS) launched a comparable programme, the Next-generation Commercial Cryptographic Algorithms Program (NGCC), in February 2025 [21] and issued its public-key call in October 2025 [22]. The round-1 candidates were published on 20 September 2026: 34 digital signature schemes, 41 key-encapsulation mechanisms and 9 key-exchange protocols, with 35 hash functions in a separate list [23]. The programme evaluates candidates in successive public rounds, with the finalists to be considered for standardization.

Each candidate has an ICCS forum thread for public comments. An independent community tracker records the findings and provides reproductions [24, 28]. Origami [1] is candidate **sign-18** of that round.

Oil-and-vinegar signatures [2, 4] use a central map that becomes linear in the hidden “oil” variables once the “vinegar” variables are fixed. The signer can then invert the map by solving a linear system. Structural attacks on these schemes [3, 4, 15, 16] recover the oil subspace, or an equivalent object, from the public key. Security therefore requires this linear structure to remain secret. It is concealed by *secret invertible linear maps*: UOV publishes $F \circ T_{\text{in}}$ with a secret input map, and Rainbow [5] publishes $T_{\text{out}} \circ F \circ T_{\text{in}}$ with both maps secret. Here F is the structured central map; see [13] for a survey.

Origami retains the central map F and solves its zones in sequence. In place of secret linear maps, it uses a compressed public key and a coordinate permutation. The key consists of a seed and a residual coefficient vector, from which the verifier reconstructs the map; the permutation Π_{pub} is derived from that public seed. The designers intend the flattening, cross-zone coupling and coefficient mixing to prevent the public map from exposing a coherent global oil space.

Our result. Origami’s zones are Rainbow’s layers in matrix notation (Section 8), with the central map exposed. Our forgery algorithm uses only the public key and performs polynomial work per pass. Each successful pass solves $d + 1$ small linear systems over $\mathbb{F}_{16}$. It needs neither the secret seed, the hidden local algebras, the private oil-coordinate partition, nor any signature. The unmodified reference verifier accepts the forgeries on all four parameter sets and the official known-answer tests (KATs).

Section 3 proves that the shipped map is bilinear. The specification explicitly makes the coordinate permutation public and omits T_{in} and T_{out} , so changing the hash or encodings would leave these linear systems exposed. Section 4 gives the success condition on the prefixes produced by the attack. Section 6 refutes the security proof’s inversion assumption in a bilinear independent-coefficient model.

We also examine two defects that are independent of the forgery. The specification treats algebra-constrained coordinates as independent, while the implemented vinegar sampling confines honest signatures to a proper subspace (Remark 1). In addition, the submitted tail zone has no variables to resample. A singular tail matrix makes the specified signer loop indefinitely and the shipped signer abort (Section 7).

Disclosure and related work. We disclosed the attack to the designers on 22 September 2026 and announced it publicly on 24 September on the NIST pqc-forum [25] and the NGCC public-comment forum, together with an Origami-128 implementation in SageMath [27].

The public tracker records the forgery as **sign-18-5** [24]. Its replay uses the SageMath implementation for Origami-128; its coverage of the other three levels relies on our experiments (Section 5) and the identical public-permutation construction in all four submitted sources. The designers acknowledged the forgery on the ICCS forum on 24 September and promised an updated algorithm [30].

The tracker’s other entries [24], with independent reproducer builds in [28], concern separate weaknesses:

1. Entry **sign-18-1** describes a generic 2^{256} collision attack on the fixed 64-byte message digest $h_\mu = H_{\text{msg}}(\mu)$. A collision allows a signature to transfer between two messages at every salt, capping Origami-384 and Origami-512 at 256 bits.

2. Entries **sign-18-3** and **sign-18-4**, filed as *probable* and confirmed on 2 October, report side-channel weaknesses in the signer: data-dependent pivoting and table lookups indexed by secrets.
3. Entry **sign-18-2** records our observation on the hidden local algebras [29]; we discuss it in Remark 1.

Our attack treats every signature coordinate as a free field element, as allowed by the verifier.

Outline. Section 2 recalls Origami in the specification’s notation. Section 3 establishes the shipped map’s bilinear form and examines independent coefficient generation. Sections 4–6 give the forgery and its cost, experiments on the reference code and KATs, and an analysis of the security proof. Section 7 compares the specification with the code and explains the signing-abort defect. Section 8 discusses the relations to Rainbow, SNOVA and QR-UOV, and possible repairs; Appendix A gives reproduction details.

2 The Origami scheme

We use the submission’s notation [1] and the implementation’s coordinate order, specified below. Equation, algorithm and section numbers in this section refer to the submission. All submitted instances use $q = 16$; we retain $\mathbb{F}_q$ in formulas and use $\mathbb{F}_{16}$ in numerical estimates.

Local zones and coordinates. The internal assignment $Y = (Y^{(1)}, \dots, Y^{(d+1)})$ has one block per *local zone*, with k_j local copies in zone j . Copy i contains vinegar and oil parts:

$$Y_i^{(j)} = (V_i^{(j)}, O_i^{(j)}) \in K_{j,i}^{v_j} \times K_{j,i}^{o_j}, \quad K_{j,i} = \mathbb{F}_q[T_{j,i}] \subseteq \text{Mat}_{l_j}(\mathbb{F}_q).$$

The hidden algebra $K_{j,i}$ has dimension l_j . Its generator $T_{j,i}$ has irreducible characteristic polynomial, so $K_{j,i} \cong \mathbb{F}_{q^{l_j}}$ (Alg. 1; eq. 1, 9, 17–18). Each element is expanded into l_j^2 matrix coordinates over $\mathbb{F}_q$ in row-major order (§2.5.6). The prefix $Y^{(<j)} = (Y^{(1)}, \dots, Y^{(j-1)})$ comprises the zones fixed before solving zone j (eq. 10).

The verifier concatenates zones in increasing order, with blocks ordered as

$$(V_1^{(j)}, \dots, V_{k_j}^{(j)}, O_1^{(j)}, \dots, O_{k_j}^{(j)}).$$

We use this order throughout and denote its flattening by Flat , with $y = \text{Flat}(Y) \in \mathbb{F}_q^N$. The specification interleaves vinegar and oil within each copy (eq. 39); the two orders differ by a fixed public permutation. Zone j has $k_j v_j l_j^2$ vinegar coordinates and $k_j o_j l_j^2$ oil coordinates, collected in $Z_j \in \mathbb{F}_q^{k_j o_j l_j^2}$ (eq. 3). The map FlatOut concatenates the output blocks $F^{(j)}(Y^{(\leq j)})$ in zone order (eq. 42). The ambient dimensions are (eq. 4, 5, 32)

$$N = \sum_{j=1}^{d+1} k_j (v_j + o_j) l_j^2, \quad M_j = m_j l_j^2, \quad M = \sum_{j=1}^{d+1} M_j.$$

We use the following index sets in $[N]$ for the coordinates of y :

$$\begin{aligned} \text{Pre}_j &= \text{positions of } Y^{(<j)}, \\ \text{Vin}_j &= \text{positions of } V_1^{(j)}, \dots, V_{k_j}^{(j)}, \\ \text{Oil}_j &= \text{positions of } Z_j, \end{aligned}$$

and $\text{Known}_j := \text{Pre}_j \cup \text{Vin}_j$. These are the “prefix or current known-coordinate positions” and “current-zone oil-coordinate positions” of §2.3, with $|\text{Vin}_j| = k_j v_j l_j^2$ and $|\text{Oil}_j| = k_j o_j l_j^2$. The public parameters and flattening convention determine all four sets. The specification’s Table 2 gives (d, k) and (N, M) but omits the per-zone values $(v_j, o_j, m_j, l_j, a_j, b_j, \delta_j)$. We take these from `origami_params.h`, identical in the reference and optimized trees; see Table 1 and the accompanying dimensions.

Central map. The internal central map $F = (F^{(1)}, \dots, F^{(d+1)})$ (eq. 11) is *staged*: zone j is affine in the oil coordinates to be solved once the earlier zones, current vinegar and sampled oil coordinates are fixed (eq. 14–16),

$$F^{(j)}(Y^{(<j)}) = D_j(X_j) + A_j(X_j) W_j, \quad X_j = (Y^{(<j)}, V_1^{(j)}, \dots, V_{k_j}^{(j)}, U_j), \quad (1)$$

with $D_j(X_j) \in \mathbb{F}_q^{M_j}$ quadratic and $A_j(X_j) \in \mathbb{F}_q^{M_j \times b_j}$ affine in the known variables X_j . The specification requires: “No quadratic term in W_j is allowed, and no term may depend on future-zone variables” (§4; §2.3 states the same requirement in two sentences). The flattened oil-coordinate vector of zone j is privately partitioned as

$$\rho_j: Z_j \mapsto (U_j, W_j), \quad U_j \in \mathbb{F}_q^{a_j}, \quad W_j \in \mathbb{F}_q^{b_j}, \quad a_j + b_j = k_j o_j l_j^2, \quad b_j \geq M_j + \delta_j$$

(eq. 6, 12). This partition is induced by a secret permutation π_j^{oil} of the $k_j o_j l_j^2$ oil positions, derived from seed_{sk} (§2.2). The signer *samples* U_j and *solves* for W_j ; the division between these two parts is private.

Signing (§2.6.2). To sign, the signer computes the message digest $h_\mu = H_{\text{msg}}(\mu)$ (eq. 28), samples a salt and derives the target (eq. 29, 31)

$$t = \text{HashToFieldVector}(\text{target} \| h_\mu \| \text{salt}, M, \mathbb{F}_q) = (u^{(1)}, \dots, u^{(d+1)}) \in \mathbb{F}_q^M, \quad u^{(j)} \in \mathbb{F}_q^{M_j},$$

then processes the zones in order. In each zone it samples the vinegar $V_i^{(j)} \in K_{j,i}^{v_j}$ and the part $U_j \in \mathbb{F}_q^{a_j}$, then solves $A_j(X_j) W_j = u^{(j)} - D_j(X_j)$ for W_j (Alg. 2, Alg. 4). It resamples whenever $\text{rank}(A_j(X_j)) \neq M_j$. After solving all zones, it flattens the internal assignment $y = \text{Flat}(Y)$ and outputs $\sigma = \Pi_{\text{pub}}(y)$ together with the salt.

Remark 1 (The hidden local algebras). We reported this structural observation on 22 September 2026 [29], before the public forgery announcements. An element of $K_{j,i} \cong \mathbb{F}_{q^{l_j}}$ has l_j degrees of freedom. Its encoding uses l_j^2 matrix coordinates, which cannot be treated as independent while requiring membership in $K_{j,i}$.

At Origami-128 the ambient encoding has 200 coordinates, whereas the literal domain, with both vinegar and oil elements in their degree-2 algebras, has 100 degrees of freedom. The shipped signer instead constrains only the 18 vinegar elements to those algebras and samples or solves oil coordinates over $\mathbb{F}_{16}$. Its honest signature vectors therefore lie in a subspace of dimension at most $200 - 18(4 - 2) = 164$, a bound also derived in the tracker’s extension of our observation [24].

The linear span of observed signatures can be computed directly. It recovers the whole constrained subspace only if the samples span that subspace. Under independent uniform sampling, $O(N)$ signatures would suffice with high probability. The dimension bound alone does not establish this sampling property.

In the public internal coordinate order, projections onto a copy’s vinegar blocks recover its algebra if the projected samples span it. The tail has no vinegar, so this argument does not recover its algebra. Our observation was acknowledged by the designers [30] and is recorded as confirmed in entry **sign-18-2** [24]. The forgery below does not use this leakage, and repairing that forgery alone would leave the constraint in place (Section 8).

Public key and verification (§2.5.8, §2.6.3). The public key is $pk = (\text{params}, \text{seed}_{\text{pk}}, R_{\text{pk}})$ (eq. 22). The verifier regenerates an ordinary MQ map $P_{\text{pub}} = P_{\text{seed}} + \text{Embed}_{\mathcal{I}_{\text{res}}}(R_{\text{pk}}): \mathbb{F}_q^N \rightarrow \mathbb{F}_q^M$ from pk and accepts iff $P_{\text{pub}}(\sigma) = t$ (Alg. 5). With Flat as fixed above and Permute denoting the verifier’s implemented permutation routine, the public map satisfies the same identity as eq. 43, 46:

$$P_{\text{pub}}(\sigma) = G(\Pi_{\text{pub}}^{-1}(\sigma)), \quad \Pi_{\text{pub}} = \text{Permute}(\text{seed}_{\text{pk}}, \text{secret-to-public}, N), \quad (2)$$

where $G: \mathbb{F}_q^N \rightarrow \mathbb{F}_q^M$ is the flattening of the central map, $G(\text{Flat}(Y)) = \text{FlatOut}(F(Y))$, and Π_{pub} is described as “public because it is derived from the public expansion seed” (§2.5.6). The security analysis also states (§4, eq. 77–78): “*The current construction does not instantiate external input/output linear hiding maps T_{in} and T_{out} .*”

The claimed hardness. Theorem 14 of the submission reduces EUF-CMA security in the random-oracle model to two computational assumptions: hidden-structure indistinguishability (HP-OV, Assumption 5) and multi-target MQ inversion (MT-MQ, Assumption 7). The reduction also uses two distributional assumptions (Assumptions 8 and 9), which relate the real and simulated signing distributions. MT-MQ (Def. 6) posits that inverting a map drawn from the *eligible-affine reference distribution* $D_{\text{elig}}^{\text{rand}}$ (uniform coefficients on the public eligible positions $\mathcal{I}_{\text{elig}}$, same permutation) is hard. Section 6 refutes this assumption in a bilinear independent-coefficient model and specifies its relation to the reference distribution.

3 The structure of the public map

The attack uses the bilinear form of P_{pub} , established below from the shipped code.

Proposition 1 (The shipped map and its staged form). *Read the regenerated verification map in internal coordinates $y = \Pi_{\text{pub}}^{-1}(\sigma)$, and write*

$$\hat{X}_j := y_{\text{Known}_j} = \text{Flat}(Y^{(<j)}, V_1^{(j)}, \dots, V_{k_j}^{(j)})$$

for the coordinates fixed before solving zone j ’s oil block. This is the known tuple X_j of (1) without the sampled part U_j . Let e range over the M_j output coordinates of zone j , which occupy the positions of $u^{(j)}$ in t .

(i) *For the shipped implementation, every such coordinate is the pure bilinear form*

$$G_e(y) = \sum_{a \in \text{Known}_j} \sum_{b \in \text{Oil}_j} c_e^{(j)}(a, b) y_a y_b,$$

where each coefficient $c_e^{(j)}(a, b)$ can be reconstructed from pk by the verifier’s public procedure.

(ii) More generally, suppose a public quadratic map has no dependence on future zones and no monomial pairing two coordinates of Oil_j in a zone- j equation. Then every such coordinate has the form

$$(P_{\text{pub}} \circ \Pi_{\text{pub}})_e(y) = G_e(y) = \widehat{D}_{j,e}(\widehat{X}_j) + \sum_{b \in \text{Oil}_j} \widehat{A}_{j,e,b}(\widehat{X}_j) y_b, \quad (3)$$

where $\widehat{D}_{j,e}$ is quadratic in $\widehat{X}_j$, each $\widehat{A}_{j,e,b}$ is affine in $\widehat{X}_j$, every coefficient is reconstructible from pk , and no monomial pairs two coordinates of Oil_j . In matrix form,

$$F^{(j)}(Y^{(\leq j)}) = \widehat{D}_j(\widehat{X}_j) + \widehat{A}_j(\widehat{X}_j) Z_j, \quad \widehat{D}_j(\widehat{X}_j) \in \mathbb{F}_q^{M_j}, \quad \widehat{A}_j(\widehat{X}_j) \in \mathbb{F}_q^{M_j \times k_j o_j l_j^2}, \quad (4)$$

which is (1) with the whole oil block Z_j as unknown. The specification's $A_j(X_j)$ is the submatrix of $\widehat{A}_j(\widehat{X}_j)$ on the columns of W_j , and $D_j(X_j) = \widehat{D}_j(\widehat{X}_j) + \widehat{A}_j(\widehat{X}_j)_{U_j} U_j$, where $\widehat{A}_j(\widehat{X}_j)_{U_j}$ denotes the columns of $\widehat{A}_j(\widehat{X}_j)$. Case (i) is the special case $\widehat{D}_{j,e} \equiv 0$ with each $\widehat{A}_{j,e,b}$ linear homogeneous.

Proof. (i) The shipped map. The verifier's map evaluation (`evaluate_public_map`, `origami_ref.c` lines 469–514) and the signer's system builder (`build_zone_system`, lines 559–609) assemble each zone- j equation e as $\sum_{a \in \text{Known}_j} \sum_{b \in \text{Oil}_j} c_e^{(j)}(a, b) y_a y_b$, iterating a over the non-oil range Known_j (prefix and zone- j vinegar) and b over the zone- j oil range Oil_j .

The coefficient $c_e^{(j)}(a, b)$, written $c(a, b)$ for fixed j and e , is obtained from the domain-separated stream `P_affine` keyed by seed_{pk} . The specification uses SHAKE256 [20], whereas the shipped code routes every SHAKE call to the ICCS placeholder XOF (`pseudoXOF`, KDF-SM3). This placeholder is provided in `auxfunc.c` in every submission package and is required by the call; the attack works with either choice. On residual positions, the stream coefficient is replaced by an entry of R_{pk} , selected by the public stride-and-offset schedule (§2.4.2; `residual_rank_from_index`). The code generates no other monomials, so the map is homogeneous bilinear. This proves (i), namely (3) with $\widehat{D}_j \equiv 0$ and $\widehat{A}_j$ homogeneous.

The signer assigns each oil coordinate of Z_j to the sampled part U_j or the solved part W_j according to the secret split ρ_j (`rho_inv`). Its coefficient remains $c(a, b) y_a$ in either case: the split simply reorganizes the columns of the same linear system. The source file `origami_ref.c` is byte-identical across the four parameter sets, so these line numbers hold for all of them.

(ii) *The conditional general form.* Collect the terms with no current-zone oil coordinate into $\widehat{D}_{j,e}$. Every remaining term contains exactly one such coordinate y_b , whose coefficient is affine in $\widehat{X}_j$ because the map is quadratic. This gives (3) and (4). Splitting the columns into U_j and W_j gives the stated relation to (1). $\square$

Independent coefficient generation. The specification requires affineness in W_j (eq. 14; Alg. 3 line 12), which alone permits $U_j \times U_j$ and $U_j \times W_j$ terms. It also prescribes an XOF-generated residual (eq. 59; Alg. 3 line 18):

$$R_{\text{pk}} = \text{ExpandToField}(\text{seed}_{\text{coeff}}, \text{Rpk-coeff} \parallel \text{param_id} \parallel \lambda_{\text{sec}} \parallel |\mathcal{I}_{\text{res}}|, |\mathcal{I}_{\text{res}}|).$$

The residual is sampled directly, without solving for its values to enforce a trapdoor constraint. We examine compatibility with the affineness requirement in an idealized model with fixed, key-independent monomial support. Each included, collected coefficient is uniform over $\mathbb{F}_q$ and independent of a uniformly chosen solved-coordinate subset W_j of size b_j in Z_j . There is no rejection or coefficient adjustment. These independence assumptions belong to the model; they do not follow exactly from domain-separated deterministic XOF calls over all keys.

Write $n_j = k_j o_j l_j^2$. For one included oil–oil monomial $y_b y_{b'}$, its coefficient is nonzero and both coordinates belong to W_j with probability

$$(1 - q^{-1}) \frac{b_j(b_j - 1)}{n_j(n_j - 1)} \quad (b \neq b'), \quad (1 - q^{-1}) \frac{b_j}{n_j} \quad (b = b').$$

Both probabilities are positive at the submitted dimensions. The model therefore cannot include such a monomial and satisfy the no- $W_j \times W_j$ requirement with probability one. This argument does not exclude oil–oil terms for every key of a deterministic implementation. If the coefficients depend on the private partition, public access to them alone need not make the partition efficiently recoverable. Our forgery uses the shipped bilinear family proved in part (i).

Remark 2 (Public access to the signing structure). Proposition 1(i) shows that the shipped map exposes its signing structure. In UOV, the analogous identity holds in secret coordinates: the public key is G composed with secret invertible maps, which a structural attack seeks to recover [3, 4]. Origami composes G only with the public permutation Π_{pub} (§4, eq. 77–78: no $T_{\text{in}}, T_{\text{out}}$). An attacker can invert this permutation directly and work in the coordinates of (3). A holder of pk thus has the signing capability of the key owner. The secret seed contributes to P_{pub} only through the two pseudorandom strings ($\text{seed}_{\text{pk}}, R_{\text{pk}}$) already published in pk . The forgery below can therefore also be viewed as recovery of an equivalent key.

4 The forgery

Lemma 1 (Inversion from the public key). *Let P_{pub} be any map of the form (3) whose zone- j oil block satisfies $k_j o_j l_j^2 \geq M_j$ (guaranteed by eq. 6: $a_j + b_j = k_j o_j l_j^2$ and $b_j \geq M_j + \delta_j$). Fix a public key pk , a target $t \in \mathbb{F}_q^M$, and polynomial-time rules for sampling vinegar and choosing solutions. A sequential pass processes each zone once and stops at the first rank failure. Every successful pass returns a preimage σ with $P_{\text{pub}}(\sigma) = t$, using $d + 1$ linear solves over $\mathbb{F}_q$ of sizes at most $M_j \times k_j o_j l_j^2$. If a pass succeeds with probability $p(pk, t) > 0$, independent restarts from zone 1 require $1/p(pk, t)$ passes in expectation. A constant lower bound on $p(pk, t)$ gives an expected $O(1)$ number of passes. An inverse-polynomial lower bound suffices for expected polynomial time, including public-map reconstruction.*

Proof. Work in internal coordinates y and process zones in order. When zone j is reached, all coordinates in Pre_j are already fixed. Assign the zone- j vinegar coordinates Vin_j arbitrarily, for example uniformly at random, to fix $\hat{X}_j$. By (3) and (4), the M_j equations of zone j become

$$\sum_{b \in \text{Oil}_j} \hat{A}_{j,e,b}(\hat{X}_j) y_b = u_e^{(j)} - \hat{D}_{j,e}(\hat{X}_j), \quad \text{i.e.} \quad \hat{A}_j(\hat{X}_j) Z_j = u^{(j)} - \hat{D}_j(\hat{X}_j),$$

a linear system in the whole zone- j oil block Z_j . The split ρ_j into (U_j, W_j) is irrelevant to this system. Both the matrix $\hat{A}_j(\hat{X}_j) \in \mathbb{F}_q^{M_j \times k_j o_j l_j^2}$ and the right-hand side are publicly computable. If $\text{rank}(\hat{A}_j(\hat{X}_j)) = M_j$, choose a solution according to the fixed rule and proceed to the next zone; otherwise stop the pass. Later zones do not alter earlier equations, so a successful pass gives $G(y) = t$. Output $\sigma = \Pi_{\text{pub}}(y)$. For fixed pk and t , independent passes have the same success probability $p(pk, t)$, giving the stated geometric expectation when $p(pk, t) > 0$. $\square$

The success probability depends on the prefixes produced by the preceding solves. These lie in fibres fixed by the earlier targets and need not be uniformly distributed. If the conditional probability of full row rank at zone j , given that the pass has reached it, is at least α_j , then

$p(pk, t) \geq \prod_{j=1}^{d+1} \alpha_j$. For a variable number of zones, constant individual bounds do not imply a constant product. Full-rank probabilities for uniformly sampled known-coordinate tuples alone do not establish these conditional bounds, and restarting an earlier zone does not by itself guarantee success.

Algorithm 1 (universal forgery). *Input:* a public key pk and a message μ . *Output on success:* a signature (σ, salt) with $\text{Verify}(pk, \mu, (\sigma, \text{salt})) = 1$.

1. Reconstruct from pk , exactly as the verifier does, the coefficients of (3) and the permutation $\Pi_{\text{pub}} = \text{Permute}(\text{seed}_{\text{pk}}, \text{secret-to-public}, N)$.
2. Pick a salt and set $h_\mu = H_{\text{msg}}(\mu)$ and $t = \text{HashToFieldVector}(\text{target} \| h_\mu \| \text{salt}, M, \mathbb{F}_q)$, and split $t = (u^{(1)}, \dots, u^{(d+1)})$ into zone blocks.
3. Run sequential passes as in Lemma 1, up to a chosen polynomial attempt bound. Report failure if none succeeds. On success, obtain y with $G(y) = t$ and set $\sigma = \Pi_{\text{pub}}(y)$.
4. Output (σ, salt) .

By (2), $P_{\text{pub}}(\sigma) = G(\Pi_{\text{pub}}^{-1}(\sigma)) = G(y) = t$, so the unmodified verifier accepts. The forger uses pk only, with no secret seed, no hidden algebra and no signing oracle.

Cost. A pass uses at most $d + 1$ Gaussian eliminations over $\mathbb{F}_{16}$ of dimension at most $\max_j M_j \times k_j o_j l_j^2$. For Origami-128 each main zone is a 32×40 solve and the tail an 8×8 solve (Table 1), with the corresponding dimensions at higher levels. Public coefficient reconstruction accounts for most of the measured cost. On all four submitted sets, the total forgery cost is within a small constant factor of one verification (Section 5).

We estimate rank failures by treating the matrices encountered along the generated prefixes as uniform random matrices. This heuristic gives no bound for every fixed key and target. A forger’s main-zone system has size $M_j \times k_j o_j l_j^2$ with $k_j o_j l_j^2 = 20k > 16k = M_j$, giving an estimated rank-deficiency probability $\approx q^{-(k_j o_j l_j^2 - M_j + 1)} = q^{-(4k+1)}$, or 16^{-9} at Origami-128. Failures are therefore expected to occur almost exclusively in the square 8×8 tail. A random 8×8 matrix over $\mathbb{F}_{16}$ is singular with probability $1 - \prod_{i=1}^8 (1 - 16^{-i}) \approx 6.6\%$. The tail zone has neither vinegar nor a sampled part ($v_{d+1} = a_{d+1} = 0$ in the reference source; Table 1). It cannot be resampled locally, so the forger restarts from zone 1, where fresh vinegar can change the eventual tail matrix. Under the same heuristic, a pass reaches and solves the tail with probability about 0.934, giving about 1.07 passes on average if independent restarts have this success rate. The observed restarts in Section 5 are consistent with the singular probability of a random 8×8 matrix over $\mathbb{F}_{16}$.

The 2–6% rates reported in the submission’s Table 10 concern the signer. It solves square $M_j \times b_j$ systems: at every level the source sets $b_j = M_j = 16k$ solved and $a_j = 4k$ sampled oil coordinates per main zone, with $\delta_j = 0$ (ORIGAMI_MAIN_A, ORIGAMI_MAIN_B). The same heuristic predicts a local rank-failure rate of about 6.6% for these systems.

Parameter dimensions. All hidden local algebras have $l_j = 2$, with $l_j^2 = 4$ coordinates per element. In a main zone $j \leq d$, $k_j = k$ and

$$(v_j, o_j, m_j, l_j) = (3, 5, 4k, 2), \quad (a_j, b_j, \delta_j) = (4k, 16k, 0).$$

Thus $|\text{Vin}_j| = k_j v_j l_j^2 = 12k$, $|\text{Oil}_j| = k_j o_j l_j^2 = 20k$ and $M_j = m_j l_j^2 = 16k$. The signer samples a_j oil coordinates and solves for $b_j = M_j$; the forger solves the $16k \times 20k$ system for the whole oil block.

The tail is a single copy ($k_{d+1} = 1$) with

$$(v_{d+1}, o_{d+1}, m_{d+1}, l_{d+1}, a_{d+1}, b_{d+1}, \delta_{d+1}) = (0, 2, 2, 2, 0, 8, 0).$$

It gives an 8×8 system with no variables to resample. These dimensions yield $N = 32dk + 8$ and $M = 16dk + 8$, as in the submission. Table 1 lists the four sets; their $d+1$ systems grow polynomially with the level.

Set	(d, k)	(N, M)	$ pk $	$ \text{sig} $	main zone $j \leq d$ $ \text{Vin}_j / \text{Oil}_j /M_j$	tail $j = d + 1$ $ \text{Vin}_j / \text{Oil}_j /M_j$
Origami-128	(3, 2)	(200, 104)	2 996 B	116 B	24/40/32	0/8/8
Origami-256	(6, 5)	(968, 488)	14 968 B	516 B	60/100/80	0/8/8
Origami-384	(8, 7)	(1800, 904)	27 940 B	948 B	84/140/112	0/8/8
Origami-512	(9, 8)	(2312, 1160)	35 924 B	1 220 B	96/160/128	0/8/8

Table 1: Submitted parameter sets (§3.2, Tables 2 and 4 of [1]) and the per-zone dimensions fixed by `origami_params.h`. Algorithm 1 solves $M_j \times k_j o_j l_j^2$ systems.

5 Experimental verification

We provide independent C [26] and SageMath [27] implementations of Algorithm 1, covering all four sets and Origami-128, respectively. Both reconstruct the public map and Π_{pub} from pk and invert the zones in order. We audited the C forgery routine to check that it references no secret-key field; the tracker runs the SageMath code after erasing the secret key.

We check each forgery with the unmodified reference routine $\text{Verify}(pk, \mu, (\sigma, \text{salt}))$ of Alg. 5. This routine recomputes the digest $h_\mu = H_{\text{msg}}(\mu)$ and reads the salt from the encoded signature $\text{sig} = \text{EncodeVec}_q(\sigma) \parallel \text{salt}$. At all four levels, the optimized tree’s `origami_opt.c` is byte-identical to `origami_ref.c`; only the compile flags differ, so the results cover both shipped code paths.

The C forger derives vinegar coordinates deterministically from seed_{pk} under `forge_vin`, indexed by zone, local attempt and outer pass, and sets free variables to zero. These pseudorandom vinegar candidates and deterministic solutions differ from the independent uniform sampling of Proposition 2. Each main zone allows up to 256 local attempts with new vinegar; a zone without vinegar allows only one. A singular tail or exhausted main-zone attempts trigger a restart from zone 1, with at most 4096 outer passes. The SageMath forger restarts from zone 1 on any rank failure, with at most 100 passes.

The C measurements were rerun after correcting retries in zones without vinegar. On a fixed Origami-128 input with two tail failures, the corrected code restarted from zone 1 twice and produced the same accepted signature as the earlier code.

Table 2 reports acceptance and three negative controls: (A) flip a signature byte, (B) flip a salt byte, and (C) verify against a different message. All three must reject.

Set	keys $\times$ msgs	forged accepted	controls A/B/C reject	false accepts
Origami-128	$100 \times 3 = 300$	300/300	300/300/300	0/0/0
Origami-256	$15 \times 3 = 45$	45/45	45/45/45	0/0/0
Origami-384	$8 \times 3 = 24$	24/24	24/24/24	0/0/0
Origami-512	$5 \times 3 = 15$	15/15	15/15/15	0/0/0

Table 2: Forgery against the unmodified reference verifier, random keys and messages.

The batches required 24, 3, 1, and 1 tail restarts, respectively, and no main-zone restarts. On an Intel Core i5-9300H, the mean wall-clock times for forgery and verification of an accepted signature were 0.0107/0.00986 s at Origami-128, 0.577/0.546 s at Origami-256, 2.73/2.60 s at Origami-384, and 5.16/4.80 s at Origami-512. Forgery time includes public-key expansion and target derivation, but excludes computing $H_{\text{msg}}(\mu)$ and preparing the salt, which precede the timed call. Verification time includes computing $H_{\text{msg}}(\mu)$; control verifications are excluded from the verification mean.

Official test vectors. For all four sets, freshly built reference code reproduces the official KAT files byte for byte. We regenerate each KAT key pair from its seed, verify its signature and forge on the KAT message. The shipped verifier accepts the forgery, which differs from the KAT signature and is therefore not a replay. At Origami-512, regenerating the key pair, forging and running the verifier twice took 14.5 s on the same machine. At every level, forgery cost was within a small constant factor of verification: both spend most of their time regenerating the coefficient stream equation by equation, with a smaller part spent on linear algebra. In a separate check at each level, `forge_from_pk` takes only the KAT public-key bytes and message, without regenerating the key pair. The program `verify_pk` accepts its output and rejects all three controls.

Use of the reference code. The C forger (`forge_demo.c`, about 300 lines) links the submission’s field tables (`origami_gf.h`), XOF wrapper (`symmetric_iccs.c`) and public API. It follows the verifier’s coefficient-stream and residual-schedule reconstruction, then inverts the map. The SageMath forger reimplements public-key expansion in Python, calling the reference library only for key generation and verification. Both implement their own vinegar selection, linear solves and restarts; neither calls the signer. The unmodified verifier checks $P_{\text{pub}}(\sigma) = t$, as required by Algorithm 5.

6 Consequence for the security proof

Theorem 14 of [1] reduces EUF-CMA security to the computational assumptions HP-OV (Assumption 5) and MT-MQ (Assumption 7), together with two distributional assumptions (8 and 9). The forgery breaks EUF-KOA security using only the public key and no signing queries. The distributional assumptions concern the signing oracle and are therefore irrelevant to this attack. We first analyze a bilinear independent-coefficient experiment, then relate it to the specification’s reference distribution.

Fix the submitted parameters and a public seed seed_{pk} , retaining its residual-position set $\mathcal{I}_{\text{res}}$ and public permutation Π_{pub} . Keep exactly the bilinear coefficient-position universe of Proposition 1(i), indexed by (j, e, a, b) with $a \in \text{Known}_j$ and $b \in \text{Oil}_j$. Thus $\mathcal{I}_{\text{all}} = \mathcal{I}_{\text{elig}}$, with no other monomials present. Replace every coefficient on this universe, including the normally seed-generated positions, by an independent uniform element of $\mathbb{F}_q$. The adversary receives the expanded map and the public metadata $(\text{seed}_{\text{pk}}, \mathcal{I}_{\text{res}}, \Pi_{\text{pub}})$. In this experiment on expanded maps, the seed specifies the coordinate convention and residual schedule. It need not regenerate the replaced coefficients.

Proposition 2 (MT-MQ with independent coefficients). *In this experiment, let the target be independent of the map. Sample each main-zone vinegar vector independently and uniformly among nonzero vectors, and choose solutions using only the current and earlier zones. One sequential pass succeeds with probability*

$$p_{\text{ind}} = \prod_{j=1}^{d+1} \prod_{r=0}^{M_j-1} (1 - q^{-k_j o_j l_j^2}),$$

over the map and the pass randomness. This probability exceeds 0.93 for each submitted parameter set, with polynomial work per pass.

Proof. Condition on any history reaching zone j , including the current vinegar choice. The known-coordinate tuple $\hat{X}_j$ is nonzero. In a main zone it contains the chosen nonzero vinegar vector; in the tail it contains the first zone’s vinegar as part of the prefix. The history depends only on the target, vinegar choices and coefficients of earlier zones, so the coefficients of zone j remain independent and uniform under this conditioning. For each equation e and oil coordinate b , the entry $\hat{A}_{j,e,b} = \sum_{a \in \text{Known}_j} c_e^{(j)}(a, b) y_a$ is uniform over $\mathbb{F}_q$. Distinct entries use disjoint coefficient families and are independent. The matrix evaluated at the generated prefix is therefore a uniform $M_j \times k_j o_j l_j^2$ matrix, with full-row-rank probability $\prod_{r=0}^{M_j-1} (1 - q^{r-k_j o_j l_j^2})$. Multiplying these conditional probabilities gives the formula, and Lemma 1 proves correctness on success. Substituting the dimensions from Table 1 gives the numerical bound, dominated by the 8×8 tail factor $\prod_{i=1}^8 (1 - 16^{-i}) \approx 0.934$. $\square$

An MT-MQ adversary can take the first independent uniform target and run one pass, succeeding with probability p_{ind} . This refutes Assumption 7 in the stated model; inversion of every target for every key is not required. The probability is averaged over maps and gives no expected restart bound for a fixed map. An EUF-KOA break similarly requires non-negligible success over the key and target experiment, without a guarantee for every key and target.

Relation to Definition 4. The specification (pp. 40–41) retains the public parameters, eligible-position universe, residual-selection rule, encoding format and public coordinate convention, while replacing the eligible-affine public coefficients by independent uniform values. Definition 6 gives the expanded map as input. Our experiment also supplies the coordinate data described as public in §4.4 and uses the shipped bilinear support. It interprets the replacement as applying to all eligible coefficients, including those normally regenerated from the seed. An independently sampled expanded map need not admit a compact encoding whose seed regenerates all non-residual coefficients. If these coefficients must still match the public seed and only R_{pk} is replaced, the resulting experiment is different and the calculation in Proposition 2 does not apply. The proposition also does not cover an enlarged support with additional monomials. Neither the direct forgery nor the one-pass calculation requires HP-OV, and we make no claim that this indistinguishability assumption holds.

7 Specification versus reference implementation

The shipped scheme is case (i) of Proposition 1, $\hat{D}_j \equiv 0$ and $\hat{A}_j$ homogeneous. The map is linear in the entire zone- j oil block Z_j once $\hat{X}_j$ is fixed. Algorithm 1 therefore needs neither the secret split ρ_j nor a structure-recovery step. This is the case tested in Section 5.

Equation 14 of the specification allows a broader class of maps. It permits D_j to be quadratic in X_j , which contains U_j , and A_j to be affine in X_j , hence in U_j . Terms of the form $U_j \times U_j$ and $U_j \times W_j$ are therefore allowed, provided no $W_j \times W_j$ term appears. This condition does not imply affineness in the whole block Z_j . The independent-generation argument of Section 3 identifies an incompatibility under explicit probabilistic assumptions. It does not extend Proposition 1(i) to every conforming implementation. Part (ii) applies only when the stated absence of oil–oil terms holds. Efficient inversion also requires the success condition of Lemma 1.

Other differences between the code and the text. The source audit found several differences in hashing and encoding. Every SHAKE256 call uses the ICCS placeholder XOF, and H_{msg} uses the placeholder hash without the `msg` label and length prefix of eq. 28. The routine `Permute` samples indices by reduction modulo $i + 1$ rather than by rejection. Some domain labels also differ: the oil

partition uses `rho` without `param_id` instead of `oil-partition` for π_j^{oil} . The within-zone coordinate order differs as described in Section 2. An implementation written from the text alone would consequently not reproduce the KATs. Correcting these differences while retaining the shipped bilinear support would leave the sequential linear systems publicly computable. The attack does not rely on the placeholder hash, these encoding choices or an error in verifier acceptance.

A functional defect, independent of the forgery. The tail zone has eight coordinates and eight equations (§3.2: $N = 32dk + 8$, $M = 16dk + 8$), so that $k_{d+1}(v_{d+1} + o_{d+1})l_{d+1}^2 = 8$ and $M_{d+1} = m_{d+1}l_{d+1}^2 = 8$. With eq. 6, $a_{d+1} + b_{d+1} = k_{d+1}o_{d+1}l_{d+1}^2 \leq 8$ and $b_{d+1} \geq M_{d+1} + \delta_{d+1} = 8 + \delta_{d+1}$, this forces $v_{d+1} = a_{d+1} = \delta_{d+1} = 0$ and $b_{d+1} = 8$ for every admissible l_{d+1} . The tail thus has no vinegar and no sampled part. The repeat loop of Alg. 4 cannot change X_{d+1} . If the square 8×8 matrix $A_{d+1}(X_{d+1})$ is singular, Alg. 4 never terminates. The shipped signer instead retries the identical system $R_{\text{max}} = \text{MAX_SIGN_ATTEMPTS} = 8192$ times before returning an error (`sign`, lines 753–830). Its `sample_zone_known` routine (lines 530–557) samples nothing when $v_{d+1} = a_{d+1} = 0$.

Using the pristine code with a fresh salt per call, we observed `sig_sign` abort on 20 of 300 calls at Origami-128 (6.67%) and 9 of 80 calls at Origami-256 (11.25%). Every abort occurred in the tail zone. At Origami-128, an abort took 8.3s, compared with 0.02s for a successful call on the same machine, making the failure visible in timing. The uniform-random-matrix heuristic predicts a tail failure rate of about 6.6% at every level, since all four tails are 8×8 . Applying this estimate to the signer assumes that its generated prefixes give approximately uniform tail matrices; we do not prove this assumption. An audit posted on the NGCC forum on 6 October 2026 independently reported the same failure [31]. It measured 213 aborts in 3300 calls at Origami-128 (6.45%) and observed aborts at all four levels of the optimized implementation. Every abort exhausted all R_{max} retries in the tail zone, and the audit left the cause open. The mechanism above explains this exhaustion, and the reported Origami-128 rate is consistent with the heuristic prediction of about 6.6%.

The failure mechanism invalidates the application of Theorem 18 in Table 13 of [1]. Under the worst-prefix definition of β_j (Definition 11), a singular tail prefix gives $\beta_{d+1} = 1$, so the bound $P_{\text{abort}} \leq \sum_j \beta_j^{R_{\text{max}}}$ is trivial. Table 13 instead uses measured average failure rates to obtain, for example, $P_{\text{abort}} < 2^{-33482}$ at Origami-128. The average tail failure rate cannot be raised to the power R_{max} , because all local retries use the same matrix. A singular prefix makes every attempt fail. The submission’s benchmark driver (`Common/bench_sig.c`, `sign_with_retry`) calls `sig_sign` again with a fresh salt until it succeeds. Its successful outputs therefore do not rule out aborted calls. The submitted tail dimensions cause this defect in any signer following Alg. 4: local retries cannot recover from a singular tail. Recovery requires restarting from an earlier zone, as a fresh signing call does.

8 Discussion

Relation to Rainbow. Rainbow [5] generalizes UOV to several layers, with variables split as $v_1 < v_2 < \dots < v_{u+1} = n$. Layer i has oil variables $x_{v_i+1}, \dots, x_{v_{i+1}}$ and treats every earlier variable as vinegar. Its $o_i = v_{i+1} - v_i$ central polynomials have no oil–oil term within the layer, allowing the signer to solve the layers in order. The public key is $T_{\text{out}} \circ F \circ T_{\text{in}}$ with secret affine maps T_{in} and T_{out} . The output map hides the assignment of equations to layers. Origami’s local zones have the same structure under the signer’s partition: reclassify the sampled coordinates U_j as vinegar and retain the solved coordinates W_j as oil. At Origami-128 the 72 vinegar coordinates and the 24 sampled oil

coordinates then form an initial vinegar set of size 96, giving

$$(v_1; o_1, o_2, o_3, o_4) = (96; 32, 32, 32, 8), \quad n = N = 200, \quad m = M = 32 + 32 + 32 + 8 = 104,$$

over $\mathbb{F}_{16}$. All sampled coordinates can be placed first in this conceptual order, since each zone is affine in its W_j and independent of future W blocks. A zone may use only part of the initial vinegar set. This correspondence describes the signer’s central map; it does not recover the private partition. The forger solves the whole blocks Z_j , of sizes $(40, 40, 40, 8)$, using the rectangular systems of Table 1.

Origami differs from textbook Rainbow in three respects. It omits the secret maps $T_{\text{in}}, T_{\text{out}}$, which allows the attack in this paper. Its central map is sparse, with the bilinear support of Proposition 1(i). It also groups variables into matrix blocks with $l_j^2 = 4$ coordinates, while prescribing hidden local subalgebras for those blocks. The submission’s “local zones” are thus Rainbow’s layers expressed in matrix notation.

Relation to SNOVA and QR-UOV. Origami appears to combine SNOVA’s matrix-variable formulation [10, 11] with the block-matrix structure of QR-UOV [18]. In SNOVA, the variables range over the full matrix ring $R = \text{Mat}_\ell(\mathbb{F}_q)$; a commutative subalgebra $\mathbb{F}_q[S]$ is used in the secret transformation and in coefficient mixing. QR-UOV represents elements of $\mathbb{F}_q[x]/(f)$, with $\deg f = \ell$, by $\ell \times \ell$ matrices and uses these representations to construct the blocks of its key matrices. Such a block has ℓ degrees of freedom, despite having ℓ^2 entries. For the chosen irreducible f , the quotient is an extension field. Origami similarly introduces the degree- l_j algebras $K_{j,i}$, but its signer does not preserve their constraints when solving for the oil coordinates (Remark 1).

The signing procedure does not exploit the irreducibility of $K_{j,i}$ to invert the central map. It solves a linear system over $\mathbb{F}_q$ in the individual matrix entries, without imposing membership of the resulting oil blocks in $K_{j,i}$. A formulation that treats all l_j^2 entries as independent should instead take the variable blocks in $R_j = \text{Mat}_{l_j}(\mathbb{F}_q)$ and sample vinegar in that full ring as well. The designers acknowledged the algebra-domain issue in their NGCC response [30]. This change would remove the particular sampling constraint of Remark 1, but would leave the public layered inversion of the shipped bilinear map intact.

Whether one uses full matrix rings or retains the original $K_{j,i}$, the resulting Origami construction remains susceptible to attacks analogous to those against SNOVA. Changing the variable domain alone leaves the submitted public map and its layered oil-and-vinegar structure exposed. Li and Ding [12] analyze SNOVA through a core oil subspace of dimension ℓo in $\mathbb{F}_q^{\ell(v+o)}$, whose tensor extension gives the oil space for the matrix variables. This permits Kipnis–Shamir, reconciliation and intersection attacks to exploit a smaller structure than an unstructured UOV model with $\ell^2(v+o)$ scalar variables would suggest. Replacing the local algebras by full matrix rings therefore does not repair Origami’s structural weaknesses.

Effect of larger parameters. Each pass of Algorithm 1 is polynomial in N . Increasing (d, k) increases the size of the forger’s linear systems along with the cost of honest verification. The total expected cost also depends on the success probability in Lemma 1. None of the four submitted parameter sets prevents the practical forgery of Section 5.

Requirements for a repair. Public-key compression is compatible with a secret trapdoor. The approach of choosing part of the public key first and deriving the rest from the trapdoor goes back to Petzoldt et al. [14]. The UOV submission to NIST [19], MAYO [17] and QR-UOV [18] all publish a seed-plus-residual key of the same form as Origami’s. A public seed expands most of the map,

and a small residual is stored explicitly. These schemes keep the oil space secret by computing the residual from the secret oil space O . In particular, they compute the oil–oil block of the public map, rather than sampling it, so that the map vanishes on O .

Origami imposes no such secret constraint on its residual. The values of R_{pk} are XOF output, and its positions are fixed by the public parameters and seed_{pk} , independently of any secret. The residual coefficients are never solved for to satisfy a trapdoor constraint. As a result, pk does not tie the public map to a hidden oil space that an attacker would need to recover. A repair would have to introduce this secret dependence, through a computed residual or secret input/output maps T_{in}, T_{out} , while retaining compatibility with compression.

The leakage of Remark 1 would also need to be addressed. An invertible linear map preserves dimension, so honest signatures would still lie in a proper subspace: the image under T_{in}^{-1} of the subalgebra-constrained space. Samples spanning this image would reveal a secret-dependent subspace, subject to the sampling qualification in Remark 1. A repair must therefore also change the sampling of hidden-algebra elements.

Security after restoring the secret maps. Adding secret input/output maps to the submitted central map would give a sparse Rainbow-type construction, with four layers at Origami-128. Its security would need to be assessed against MinRank and Billet–Gilbert [6], Rainbow band separation [7], the rectangular MinRank and intersection attacks [15], and the simple attack [16] that broke the NIST round-3 parameters [8]. The submission’s security analysis names several of these families (§4.4: Kipnis–Shamir, reconciliation, wedge, intersection, MinRank and rectangular MinRank), but assigns only symbolic costs and asserts that none falls below the target without computing parameter-level estimates. Its Tables 11 and 12 estimate only exhaustive key search, claw-finding and preimage search. A reduction to HP-OV and MT-MQ cannot establish security unless those assumptions withstand the known attacks at the chosen parameters. Retaining matrix structure also calls for analysis of the SNOVA attack methods discussed above. We do not claim an attack on a hypothetical version with both secret maps.

9 Conclusion

Origami exposes a Rainbow-type layered signing map in public coordinates. Its shipped bilinear form, public permutation and absence of secret input/output maps let a holder of the public key forge through sequential linear solves. This classical attack breaks all four claimed security levels, as demonstrated against the unmodified reference implementation and official test vectors. The hashing and encoding differences between the specification and the code do not affect it. The security proof’s multi-target inversion assumption also fails in the independent-coefficient model of Proposition 2.

Two further defects would remain after repairing the forgery alone. With the submitted tail-zone dimensions, local retries cannot recover from a singular system. Honest signatures also lie in a proper constraint subspace whose dimension would remain unchanged by a secret input map. Using full matrix rings for both vinegar and oil would remove this algebra-membership constraint, but would leave the submitted public signing structure exposed. Restoring secret maps around the submitted central map would produce a sparse Rainbow-type construction. Its security would require an analysis of the known Rainbow attacks and of attacks exploiting the matrix structure, as in SNOVA, at the proposed parameters.

Use of AI tools

We identified the inconsistency of the hidden local algebras and the exposure of the central map without AI assistance. Anthropic’s Claude helped with our C implementation and the experiments in Section 5. OpenAI’s ChatGPT 5.6 (Daybreak Blue) helped with the SageMath implementation [27]. We also used Claude to draft and revise the text. We take full responsibility for the content and conclusions of the paper.

References

- [1] Y. Shen, K. Cheng, A. Song, X. Liu, X. Shi, J. Wu, Y. Wu, X. Mu, L. Zheng, Z. Niu. *Origami*. Submission to the NGCC round-1 public-key call (candidate `sign-18`), Xidian University, 26 June 2026. Package <https://www.niccs.org.cn/niccs/Proposal/Public-Key%20Cryptographic%20Algorithms/Round%201%20candidates/Origami.zip>, SHA-256 `e34f18832e968681dd0c51ce0d4b29d80e01ad29daa83dfb805718b76fdffa80`.
- [2] J. Patarin. The Oil and Vinegar signature scheme. Presented at the Dagstuhl Workshop on Cryptography, September 1997.
- [3] A. Kipnis and A. Shamir. Cryptanalysis of the Oil and Vinegar signature scheme. In *CRYPTO 1998*, LNCS 1462, pp. 257–266. Springer, 1998.
- [4] A. Kipnis, J. Patarin, and L. Goubin. Unbalanced Oil and Vinegar signature schemes. In *EUROCRYPT 1999*, LNCS 1592, pp. 206–222. Springer, 1999.
- [5] J. Ding and D. Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *ACNS 2005*, LNCS 3531, pp. 164–175. Springer, 2005.
- [6] O. Billet and H. Gilbert. Cryptanalysis of Rainbow. In *SCN 2006*, LNCS 4116, pp. 336–347. Springer, 2006.
- [7] J. Ding, B.-Y. Yang, C.-H. O. Chen, M.-S. Chen, and C.-M. Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In *ACNS 2008*, LNCS 5037, pp. 242–257. Springer, 2008.
- [8] G. Alagic et al. Status report on the third round of the NIST post-quantum cryptography standardization process. NIST IR 8413 (updated), National Institute of Standards and Technology, September 2022.
- [9] National Institute of Standards and Technology. FIPS 203 (ML-KEM), FIPS 204 (ML-DSA) and FIPS 205 (SLH-DSA), August 2024.
- [10] L.-C. Wang, P.-E. Tseng, Y.-L. Kuan, and C.-Y. Chou. A simple noncommutative UOV scheme. IACR Cryptology ePrint Archive, Report 2022/1742, 2022. <https://eprint.iacr.org/2022/1742>.
- [11] The SNOVA team. *SNOVA: Proposal for NISTPQC: Digital Signature Schemes project*. Specification document, 25 May 2023, §1.3–1.5. <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/SNOVA-spec-web.pdf>.
- [12] P. Li and J. Ding. Cryptanalysis of the SNOVA signature scheme. In *PQCrypto 2024*, LNCS 14772, pp. 79–91. Springer, 2024. <https://eprint.iacr.org/2024/110>.

- [13] J. Ding, A. Petzoldt, and D. S. Schmidt. *Multivariate Public Key Cryptosystems*, 2nd edition. Advances in Information Security, vol. 80. Springer, 2020.
- [14] A. Petzoldt, E. Thomae, S. Bulygin, and C. Wolf. Small public keys and fast verification for multivariate quadratic public key systems. In *CHES 2011*, LNCS 6917, pp. 475–490. Springer, 2011.
- [15] W. Beullens. Improved cryptanalysis of UOV and Rainbow. In *EUROCRYPT 2021*, LNCS 12696, pp. 348–373. Springer, 2021.
- [16] W. Beullens. Breaking Rainbow takes a weekend on a laptop. In *CRYPTO 2022*, LNCS 13508, pp. 464–479. Springer, 2022.
- [17] W. Beullens. MAYO: Practical post-quantum signatures from Oil-and-Vinegar maps. In *SAC 2021*, LNCS 13203, pp. 355–376. Springer, 2022.
- [18] H. Furue, Y. Ikematsu, Y. Kiyomura, and T. Takagi. A new variant of Unbalanced Oil and Vinegar using quotient ring: QR-UOV. In *ASIACRYPT 2021*, LNCS 13093, pp. 187–217. Springer, 2021. <https://eprint.iacr.org/2020/1243>.
- [19] W. Beullens, M.-S. Chen, J. Ding, B. Gong, M. J. Kannwischer, J. Patarin, B.-Y. Peng, D. Schmidt, C.-J. Shih, C. Tao, and B.-Y. Yang. UOV: Unbalanced Oil and Vinegar. Algorithm specifications and supporting documentation. Submission to the NIST call for additional post-quantum digital signature schemes, 2023. <https://www.uovsig.org/>.
- [20] National Institute of Standards and Technology. SHA-3 standard: permutation-based hash and extendable-output functions. FIPS PUB 202, August 2015.
- [21] Institute of Commercial Cryptography Standards (ICCS). Announcement on launching the Next-generation Commercial Cryptographic Algorithms Program (NGCC), 5 February 2025. https://www.niccs.org.cn/niccs/Notice/pc/content/content_1937428197396713472.html.
- [22] Institute of Commercial Cryptography Standards (ICCS). Call for proposals for the next-generation public-key cryptographic algorithms, 9 October 2025. https://www.niccs.org.cn/niccs/Notice/pc/content/content_1975896137741635584.html.
- [23] Institute of Commercial Cryptography Standards (ICCS). NGCC round-1 public-key candidates, published 20 September 2026. <https://www.niccs.org.cn/niccs/Round1Additional/pc/list.html>.
- [24] M.-J. O. Saarinen (maintainer). NGCC candidate reports: Origami (sign-18), entries sign-18-1 (21 September 2026), sign-18-2 (22 September 2026), sign-18-3 and sign-18-4 (23 September 2026) and sign-18-5 (24 September 2026). <https://ngcc.dev/reports/sign-18.html>, accessed 2 October 2026.
- [25] H. Louiso and J. Ding. The total break of Origami, a NGCC PQC candidate. Message to the NIST pqc-forum mailing list, 24 September 2026. <https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/N1Wfpgv1I2s>.
- [26] H. Louiso. *origami-forgery*: C implementation of the forgery and reproduction scripts. <https://github.com/hlouiso/origami-forgery>.
- [27] P. Pébereau. Re: Round1: Public Comment: Origami. NGCC PKC Forum, 24 September 2026. <https://list.niccs.org.cn/archives/list/pkcforum@list.niccs.org.cn/message/HTZTMJ42AUXCWEDA3AURP4ZOEGSGSV4Z/>. Code: *UnfoldOrigami*, <https://github.com/pi-r2/UnfoldOrigami>, commit defa3405 (the revision pinned by [24]).

- [28] M.-J. O. Saarinen (maintainer). NGCC round-1 reproduction harness: build tooling and runnable reproducers for the findings published at [ngcc.dev](https://github.com/ngcc-dev/ngcc-harness). <https://github.com/ngcc-dev/ngcc-harness>, accessed 1 October 2026.
- [29] P. Li. Re: Round1: Public Comment: Origami. NGCC PKC Forum, 22 September 2026. <https://list.niccs.org.cn/archives/list/pkcforum@list.niccs.org.cn/message/7LD2222NOYVSL3KXW4UQCLPHMU7JFKQW/>.
- [30] A. Song (for the Origami designers). Re: Round1: Public Comment: Origami. NGCC PKC Forum, 24 September 2026. <https://list.niccs.org.cn/archives/list/pkcforum@list.niccs.org.cn/message/UDPVTMF6PCXSDEAGVE2IWZSBUKVMNVNW/>.
- [31] LK-PQC-Hunter. Re: Round1: Public Comment: Origami. NGCC PKC Forum, 6 October 2026. <https://list.niccs.org.cn/archives/list/pkcforum@list.niccs.org.cn/message/MP3WUOWQBA4SJ6TFEZY562D60E56K3L7/>.

A Reproduction

The submission package is the archive listed in [1]. For each parameter set, we compile the unmodified reference sources and the C forger with `gcc -std=c11 -O2 -DORIGAMI_OPT=REF`, using that level’s headers. The forger additionally uses `-DORIGAMI_MEASURE` for instrumentation, and the KAT driver uses `-DKAT_NUM_TESTS=1`. The submission’s Makefiles use `-O3`. The reference and optimized trees contain identical algorithm sources, with different compile flags. The file `origami_ref.c` has 867 lines and is identical at all four levels. The routines relevant to the attack are `genkeys` (lines 699–715), `derive_residual_coefficients` (219–233), `derive_public_permutation` (438–444), `evaluate_public_map` (469–514), `build_zone_system` (559–609), `solve_rect_random` (611–697), `sign` (753–830) and `verify` (850–867).

The C code and reproduction instructions are available at <https://github.com/hlouiso/origami-forgery>. The batch driver `forge_demo.c` implements Algorithm 1 and the three negative controls of Section 5; `scripts/run.sh` runs the random-key batches. The script `scripts/measure_paper.py` (commit 8800310) rebuilds the reference sources, reproduces the KAT files, checks forgeries through `forge_demo --one`, and records the batch timings and restart counts. Its optional `--old-forger` argument enables the fixed-input comparison with the earlier retry implementation. Direct forgery from KAT public-key bytes uses `forge_from_pk` and `verify_pk`, as documented in `docs/REPRODUCE.md`. The SageMath implementation for Origami-128 is available at <https://github.com/pi-r2/UnfoldOrigami> (commit defa3405, the revision the tracker replays).