Secret-Subspace Recovery in MAYO via Linearization of Errors from a Single Fault

Alberto Marcos
Arquimea Research Center
amarcos@arquimea.com

Keywords: MAYO, fault injection, post-quantum signatures, multivariate cryptography

Abstract

We present fault injection attacks against MAYO in which a single faulty execution reveals structural information about the secret. We consider two closely related single-block fault models. In the first one, a controlled perturbation affects one *oil* coordinate of a signature block and induces an error $e \in \mathcal{O}$. In the second one, the perturbation may affect several *oil* coordinates of the same block. In both cases, we show that the observable verification mismatch can be written as the image of the induced error under a publicly derivable linear operator. This yields two recovery strategies. In the linear route, when the relevant *oil* part of the error can be identified, recovery reduces to solving a linear system over $\mathbb{F}_q$. In the reduced-quadratic route, the same linear relation is combined with the constraint $P(e) = 0$, which leaves, in the generic full-rank case, a reduced system of m quadratic equations in $n - m$ variables. For the MAYO parameter sets, $n - m = o$ is small, and the resulting reduced quadratic systems can be solved over the base field. Recovering e yields a nonzero vector of the secret subspace and provides a starting point for key-recovery techniques against the *oil space*. We also discuss the practical cost of both approaches and explain why the attack applies to randomized MAYO.

1 Introduction

The National Institute of Standards and Technology (NIST) has recently concluded its post-quantum cryptography (PQC) standardization process, selecting schemes designed to withstand threats arising from quantum computing. Among the adopted standards are the digital signature schemes ML-DSA [Nat24a] and SLH-DSA [Nat24b]. With the aim of broadening the range of viable alternatives in PQC, NIST launched in September 2022 a second call focused on identifying new signature algorithms that offer different security assumptions and performance trade-offs from those of ML-DSA and SLH-DSA.

In this context, the MAYO signature scheme [Beu22, BCC$^+$25] has emerged as one of the most relevant candidates selected by NIST for consideration in the new PQC signature call. MAYO belongs to the family of multivariate schemes, whose security relies on the computational hardness of solving systems of nonlinear polynomial equations over finite fields. Beyond this algorithmic security model, implementations must also be assessed against adversaries with physical access. Fault injection attacks are a particularly relevant threat in this setting, since they can disturb the signing computation and expose information that is not available in the fault-free model.

In recent years, fault injection attacks against multivariate schemes have received increasing attention in the literature, with a particular focus on the MAYO scheme [AMSU24, ABCR25, Zho25, BSK25]. These works consider different attack models and fault scenarios, and analyze how perturbations during the signing process can lead to the exposure of secret information.

In this article, we study a fault injection scenario in MAYO in which perturbations on internal variables during the signing process give rise to additional algebraic relations. From these relations, we show that it is possible to recover information about the secret key, even when the randomized variant of the algorithm is used.

Contributions. To the best of our knowledge, this is the first work that presents a *single-fault* attack against *randomized* MAYO based on perturbing intermediate data in the oil-solution computation. The attack enables recovery of the secret oil subspace from one faulty signature, without relying on known or predictable vinegar values, repeated signing randomness, a non-faulty reference signature, or skipped high-level signing steps.

- We propose two closely related fault settings. For the first attack, we consider a simple *fault model* in which a controlled perturbation affects a single coordinate of the oil-solution vector α before the corresponding oil-space vector $(O\alpha\|\alpha)$ is formed. For the second attack, we relax this condition and allow the perturbation to affect several coordinates of the same oil-solution vector, provided that the resulting error remains confined to a single signature block.
- We derive an *explicit linearization* of the observable mismatch during verification as a function of the induced error, exploiting the bilinear structure of the scheme. This common derivation is the starting point for both recovery attacks considered in the paper.
- We present a first recovery attack, namely a *linear recovery attack*: under the single-coordinate fault model, when the relevant *oil* component of the induced error is known, the full error is recovered by solving a linear system over $\mathbb{F}_q$.
- We present a second recovery attack, namely a *reduced-quadratic recovery attack*: starting from the same public linear relation, but without assuming that the injected error is known in advance, we combine it with the constraint $P(e) = 0$, eliminate variables, and obtain a reduced quadratic system. In this way, the second route no longer requires prior knowledge of the injected error and applies to the relaxed fault setting in which several oil coordinates may be affected, as long as the perturbation remains confined to a single block.
- We analyze how both recovery attacks, despite differing only in the way the error is reconstructed, ultimately produce the same useful output: a nonzero vector of the secret subspace. We then discuss how this common output can be used for *key recovery* through existing post-processing techniques for UOV schemes, following the line of work initiated by Aulbach *et al.* [ACK+23] and further developed by Pébereau [Péb24].

2 Preliminaries and Scheme Description

Let $\mathbb{F}_q$ be a finite field of characteristic 2 (in the proposed parameter sets one uses $q = 16$). The scheme is parameterized by (n, m, o, k, q). The public key defines a homogeneous quadratic map

$$P : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m, \qquad P(x) = \big(p_0(x), \ldots, p_{m-1}(x)\big),$$

with the trapdoor property that there exists a secret linear subspace $\mathcal{O} \subset \mathbb{F}_q^n$ of dimension o on which P vanishes: $P(u) = 0$ for all $u \in \mathcal{O}$.

In Oil–Vinegar-type implementations it is natural to partition a vector $x \in \mathbb{F}_q^n$ into two blocks

$$x = (w\|x_O), \qquad w \in \mathbb{F}_q^{n-o}, \;\; x_O \in \mathbb{F}_q^o,$$

and to represent the secret subspace in systematic form by means of a matrix $O \in \mathbb{F}_q^{(n-o)\times o}$:

$$\mathcal{O} = \{(O\alpha \| \alpha) : \alpha \in \mathbb{F}_q^o\}. \tag{1}$$

The secret data is essentially O (or an equivalent encoding); the public key determines P.

The MAYO design introduces a *public amplification* that transforms P into a new map

$$P^\star : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m, \qquad P^\star(x_0, \ldots, x_{k-1}) = \sum_{i=0}^{k-1} E_{ii}\, P(x_i) + \sum_{0 \le i < j \le k-1} E_{ij}\, P'(x_i, x_j),$$

where $\{E_{ij}\} \subset \mathbb{F}_q^{m\times m}$ are fixed public matrices and

$$P'(x, y) := P(x + y) - P(x) - P(y) \tag{2}$$

is the differential of P. The map P' is a symmetric bilinear map.

2.1 Key Generation Algorithm

Key generation produces a public instance P with Oil–Vinegar structure and a secret subspace $\mathcal{O}$ such that $P(u) = 0$ for all $u \in \mathcal{O}$. The secret key encodes a basis of $\mathcal{O}$ (equivalently, the matrix O of (1) in systematic form), while the public key encodes the coefficients of P and the matrices E_{ij} (or seeds from which they are expanded).

2.2 Signing Algorithm

Given a message M, the signer derives a target $t \in \mathbb{F}_q^m$ via a hash with salt. It then randomly samples k base vectors $v_0, \ldots, v_{k-1} \in \mathbb{F}_q^n$ and searches for corrections $o_0, \ldots, o_{k-1} \in \mathcal{O}$ such that

$$P^\star(v_0 + o_0, \ldots, v_{k-1} + o_{k-1}) = t. \tag{3}$$

The key point is that (3) reduces to a linear system in the ko free coordinates that parameterize $(o_0, \ldots, o_{k-1})$. To see this, one expands using (2):

$$P(v_i + o_i) = P(v_i) + P'(v_i, o_i) + P(o_i) = P(v_i) + P'(v_i, o_i),$$

since $P(o_i) = 0$ by definition of $\mathcal{O}$. Similarly, when $P'(v_i + o_i,\, v_j + o_j)$ appears in the cross part of $P^\star$, all terms involving o_i, o_j enter through P', which is bilinear, and moreover $P'(o_i, o_j) = P(o_i + o_j) - P(o_i) - P(o_j) = 0$ because $o_i, o_j, o_i + o_j \in \mathcal{O}$. Therefore, (3) becomes a linear system in the unknowns describing $o_0, \ldots, o_{k-1}$. A solution yields a signature

$$s = (s_0, \ldots, s_{k-1}), \qquad s_i = v_i + o_i \in \mathbb{F}_q^n,$$

together with the salt.

2.3 Verification Algorithm

The verifier recomputes the target $t \in \mathbb{F}_q^m$ from (M, salt) and the public key, and accepts if

$$P^\star(s_0, \ldots, s_{k-1}) = t. \tag{4}$$

In efficient implementations, $P^\star$ is evaluated as a linear combination of bilinear products induced by P' and of quadratic evaluations of P weighted by the matrices E_{ij}.

3 Related Work

3.1 Fault injection attacks relevant to MAYO

MAYO belongs to the broader family of UOV-based signature schemes, and this family has attracted increasing interest from the perspective of physical security, as practical implementations introduce attack surfaces not captured by the classical algorithmic model. In particular, *fault injection* (FI) attacks can perturb the signing process and lead to the exposure of internal information. Aulbach, Campos, and Krämer [ACK25] provide a systematic overview of physical attacks on UOV-based schemes, including UOV, MAYO, QR-UOV, and SNOVA. For the purposes of this work, the most relevant part of that survey is the classification of FI attacks according to the *algorithmic effect* of the fault: fixing or reusing vinegar values, preventing the correct combination of vinegar and oil terms, disturbing the setup or solution of the signing linear system, or perturbing secret-dependent data used to form the oil space.

FI that fixes or reveals vinegar values. A first family of attacks targets the generation of the vinegar vectors. The SoK of Aulbach *et al.* [ACK25] explains this attack principle in the UOV setting and then discusses its transfer to MAYO. In MAYO, the relevant observation is blockwise: each signature component has the form $s_i = v_i + o_i$, with $o_i \in \mathcal{O}$. Thus, if a fault makes v_i known, fixed, or reused in a controlled way, the corresponding oil-space contribution can be isolated.

Aulbach *et al.* [AMSU24] instantiate this idea for MAYO by inducing a fault during vinegar sampling. If one or more vinegar blocks are not freshly sampled and instead remain equal to a known value, for instance zero, then the corresponding signature block directly exposes an oil vector. They also discuss a differential setting in which a faulty execution is compared with a related correct execution; if the same vinegar block is reused, subtracting the two signatures cancels the vinegar contribution and yields an oil-space vector. The attack therefore exploits a fault before the oil correction is combined with the vinegar block. Its output is a nonzero vector of the secret oil space, which can then be used for full oil-space recovery.

Jendral and Dubrova [JD25] target the same high-level object, namely the vinegar values, but at an earlier point in the computation. Instead of faulting the sampled vinegar vector directly, they fault the derivation of the seed from which the vinegar values are expanded. The effect is that the vinegar seed becomes constant or predictable from public information. Given a faulty signature and the predictable vinegar seed, the attacker reconstructs the vinegar blocks and obtains enough information to recover the secret oil space. This line of attack applies to both deterministic and randomized MAYO, because the fault removes or bypasses the intended unpredictability of the vinegar-generation process.

FI that prevents or partially omits the oil contribution. Another family of attacks targets the final combination of vinegar and oil terms in the signature. The SoK [ACK25] discusses this as an attack on the addition of the vinegar and oil parts. If the implementation outputs only the vinegar part, or omits part of the oil contribution, the faulty signature can reveal information about the missing oil term. In deterministic variants, a correct and a faulty signature for the same message may be compared so that the repeated vinegar contribution cancels. For MAYO, the SoK points out that the signature is formed from k pairs of vinegar and oil vectors, and therefore a fault in this final combination may expose information from one block even if the other blocks are unaffected. The exact effect depends on how the implementation copies or adds the two parts, but the relevant algebraic situation is always that one contribution to $s_i = v_i + o_i$ is suppressed, left constant, or only partially incorporated.

Abdelmonem *et al.* [ABCR25] extend this direction to randomized MAYO using a correction-based strategy. Their attack faults the operation that adds oil coefficients to the corresponding

vinegar coefficients in the signature. Rather than requiring the same vinegar values to appear in two executions, the attacker uses public verification as a correction oracle: for a faulty signature in which the addition of one or several oil coefficients to the corresponding vinegar coefficients has been skipped, candidate corrections are tested until verification succeeds. Repeating this process accumulates partial information about oil vectors, and linear algebra is then used to reconstruct the oil space. The correction-based strategy is designed to overcome the randomized setting, where a direct comparison with a related non-faulty signature is not available. The underlying fault effect is not inherently restricted to randomized MAYO, but the attack relies on many targeted faulty signatures and on signature-correction queries.

FI on expanded secret-dependent data. The SoK [ACK25] also discusses attacks that perturb secret-dependent data used during signing, such as the representation of the oil space or secret matrices in an expanded secret key. For MAYO, this class of attacks is more delicate than for schemes where the full secret transformation is stored permanently, because MAYO secret keys are compressed and the oil-space representation is expanded only during signing. Consequently, attacks that rely on persistent memory faults against the secret representation do not transfer directly to standard compressed-key MAYO implementations. However, if an adversary can fault the expanded secret-dependent data while it is present during signing, the resulting signature may contain a structured deviation related to the oil space. In deterministic MAYO, such deviations can be especially damaging because comparison with a correct signature for the same message may cancel the vinegar part and reveal a nonzero vector in $\mathcal{O}$.

FI that disturbs the signing linear system. Another attack class identified in the SoK [ACK25] targets the setup of the linear system solved during signing. In MAYO, this means perturbing the matrix A or the right-hand side y used to compute the oil coefficients. If this perturbation changes the solution vector but the signing procedure still outputs a signature, then the oil contribution in one or more blocks may differ from the one produced in an unfaulted execution. As with other differential attacks, this is most directly exploitable in deterministic settings, where two signatures of the same message share the same vinegar contribution. Subtracting the correct and faulty signatures can then isolate an oil-space vector.

FI that exploits non-unique valid solutions. Zhong [Zho25] proposes a theoretical FI framework based on the fact that MAYO signing solves an underdetermined linear system. Since the vector of oil coefficients has ko entries while the target has only m equations, the signing procedure contains free variables whenever the relevant matrix has full rank. In the deterministic setting, the same message fixes the signing instance, while faulting the pseudorandom vector r, or its use inside the solution-sampling routine, can lead to a different valid solution of the same linear system. Comparing the original valid signature with this fault-induced valid signature cancels the common part and reveals oil-space information. This attack differs from attacks that produce invalid faulty signatures: the faulty output is still accepted by verification. However, its applicability is closely tied to the ability to obtain such related valid signatures for the same signing instance. In randomized MAYO, the same subtraction argument no longer applies directly, because fresh randomness changes the underlying instance and the difference of two signatures need not lie in the oil space.

Deterministic FI via re-signing. Bauer, De Santis, and Koleci [BSK25] study fault attacks against deterministic UOV-based signatures, including deterministic MAYO. Their attacks perturb either the target value used in the signing equation or the sampling of the solution variables. In deterministic MAYO, signing the same message reuses the same vinegar values. Therefore, from one correct and one faulty signature for the same message, the vinegar contribution cancels and the difference reveals an oil-space vector. This gives full key recovery once

the recovered vector is fed to known oil-space reconstruction methods. The attack explicitly exploits determinism and does not directly apply to randomized MAYO.

This work. Our attack follows a different route. We do not force the vinegar values to be known, fixed, or reused; we do not require a non-faulty reference signature; and we do not rely on an adaptive correction phase over many faulty signatures. Instead, we consider a single faulty execution in which the induced block error belongs to the secret oil subspace. The attacker observes only the faulty signature, the message, the salt, and the public key, and computes the public verification mismatch $\Delta t = P^\star(s') - t$. The main algebraic point is that, under the single-block fault model, this mismatch can be written as a public linear relation

$$\Delta t = \mathcal{L}(e),$$

where $e \in \mathcal{O}$ is the induced error. When the oil part of e is known or can be enumerated, this gives the linear recovery route. When it is not known, the same relation can be combined with $P(e) = 0$, yielding the reduced-quadratic recovery route. In both cases, one faulty randomized MAYO execution can reveal a nonzero vector of the secret oil subspace.

4 Threat Model and Fault Model

4.1 Adversary Capabilities

We assume an adversary with access to the signing device who can:

- request signatures on messages that are known to the adversary, with adaptive access to a legitimate signing oracle;
- induce, in a controlled manner, a transient and localized fault during a single execution of the signing algorithm;
- observe the signature produced by the faulty execution and know the signed message and the associated public key, so that they can locally run the public verification algorithm and compute the observable mismatch introduced by the fault.

4.2 Fault Model

For both attacks, we assume that the perturbation remains confined to a *single signature block*. For notational simplicity, we write the derivation for block 0, so only the first signed component is modified; the same construction applies to any block by using the corresponding public matrices E_{ij}. What differs between the two attacks is the structure assumed for the perturbation inside that block.

Fault model for the linear attack. For the first attack, we assume that the fault affects a *single oil coordinate*. Without loss of generality, we take this coordinate to be coordinate 0 of the oil vector, so that the injected oil correction is

$$\alpha = \delta\, e_0 \in \mathbb{F}_q^o, \qquad \delta \in \mathbb{F}_q^\times, \qquad e_0 = (1, 0, \ldots, 0). \tag{5}$$

This is the setting in which the attacker can later exhaustively guess the affected coordinate and the nonzero value, and therefore recover the full error through the linear route.

Fault model for the reduced-quadratic attack. For the second attack, we relax the previous condition and allow the perturbation to affect several oil coordinates of the same block. We therefore only assume that the injected oil correction is some nonzero vector

$$\alpha \in \mathbb{F}_q^o \setminus \{0\}, \tag{6}$$

without requiring that the attacker knows α or even which coordinates inside the block were affected. The only structural restriction that remains is that the perturbation must stay confined to a single block.

In the systematic representation (1), both fault models induce an error vector in the full space

$$e \;:=\; (O\alpha \| \alpha) \in \mathbb{F}_q^n, \tag{7}$$

which belongs to the secret subspace $\mathcal{O}$. The faulty signature then has the form

$$s' = (s'_0, s_1, \ldots, s_{k-1}), \qquad s'_0 = s_0 + e, \tag{8}$$

while all other blocks remain unchanged.

4.3 Fault Observation

The immediate consequence is that the verification equation (4) generally no longer holds:

$$P^\star(s') \neq t.$$

Since the attacker has access to (M, salt) and to the faulty signature s', they can also compute the *observable mismatch*

$$\Delta t \;:=\; P^\star(s') - t \in \mathbb{F}_q^m. \tag{9}$$

If the original signature s was valid, then $t = P^\star(s)$ and

$$\Delta t \;=\; P^\star(s') - P^\star(s), \tag{10}$$

that is, Δt coincides exactly with the change induced by the error e.

5 Attack Description

We describe two attack variants:

1. **Linear attack.** Recovery of the induced error: from Δt, the attacker constructs a linear system to recover e (in particular, its vinegar part), exploiting the more restrictive fault model in which the oil part of the error can be identified, for instance by exhaustive search over the faulty coordinate and the injected value using public verification.

2. **Reduced-quadratic attack.** Recovery of the induced error: starting from the same observable mismatch Δt, the attacker constructs a public linear relation in e and combines it with the constraint $P(e) = 0$, which leads to a reduced quadratic system whose solutions yield candidate errors, without requiring prior knowledge of the oil part of the error.

3. **Common post-processing for secret key recovery.** In both attacks, once a nonzero vector $e \in \mathcal{O}$ has been recovered, this vector of the secret subspace serves as a witness to attempt reconstruction of an equivalent representation of the secret key, for example a matrix O. Thus, the two attack variants differ only in the way the error is recovered from Δt, while the final step that scales from one secret-subspace vector to full key recovery is the same in both cases.

5.1 Error Recovery I: Linearization via the Public Map $P^\star$

The goal is to reconstruct e (or, more precisely, its vinegar part) from Δt, leveraging the fact that the attacker knows *which* oil coordinate was perturbed, that is, they know (5).

5.1.1 Step 1: Expanding $P^\star(s') - P^\star(s)$ and isolating the changing terms

By (8), the only block that changes is s_0. Substituting into the definition of $P^\star$, all summands that do not involve s_0 cancel out in (10). We obtain

$$\Delta t = E_{00}\big(P(s_0+e) - P(s_0)\big) \; + \; \sum_{j=1}^{k-1} E_{0j}\big(P'(s_0+e, s_j) - P'(s_0, s_j)\big). \tag{11}$$

In the second term, the bilinearity of P' implies

$$P'(s_0+e, s_j) - P'(s_0, s_j) = P'(e, s_j),$$

so the cross terms contribute as a sum of linear terms in e.

In the first term, we use the general identity for homogeneous quadratic maps:

$$P(x+e) - P(x) = P'(x, e) + P(e). \tag{12}$$

Here the decisive structural condition appears: since $e \in \mathcal{O}$, we have $P(e) = 0$. Therefore,

$$P(s_0+e) - P(s_0) = P'(s_0, e).$$

Moreover, in characteristic 2 one has $P'(e, e) = 0$ for all e, and hence

$$P'(s_0, e) = P'(s_0+e, e).$$

This allows the diagonal term to be rewritten *as a function of* $s_0' = s_0 + e$, which is known to the attacker. Substituting, we obtain a fully observable expression:

$$\Delta t = E_{00}\, P'(s_0', e) \; + \; \sum_{j=1}^{k-1} E_{0j}\, P'(e, s_j). \tag{13}$$

Equation (13) is the starting point of the recovery.

5.1.2 Step 2: Why (13) is a linear relation in e

For each fixed $y \in \mathbb{F}_q^n$, the map

$$\Phi_y : \mathbb{F}_q^n \to \mathbb{F}_q^m, \qquad \Phi_y(x) := P'(x, y),$$

is linear in x because P' is bilinear. The same holds for $x \mapsto P'(y, x)$. Moreover, left multiplication by $E_{ij} \in \mathbb{F}_q^{m\times m}$ is linear on $\mathbb{F}_q^m$. Therefore, the right-hand side of (13) is a linear combination of linear maps applied to e. Consequently, there exists a unique linear transformation

$$\mathcal{L} : \mathbb{F}_q^n \to \mathbb{F}_q^m \quad \text{such that} \quad \Delta t = \mathcal{L}(e), \tag{14}$$

with

$$\mathcal{L}(e) := E_{00}\, P'(s_0', e) + \sum_{j=1}^{k-1} E_{0j}\, P'(e, s_j). \tag{15}$$

Observe that, since P' is bilinear and symmetric, the terms defining $\mathcal{L}$ can equivalently be rewritten by always placing the error e as the first argument. In particular, symmetry implies that $P'(s_0', e) = P'(e, s_0')$, and bilinearity guarantees that, for each fixed vector y, the map $e \mapsto P'(e, y)$ is linear. In this way, all summands composing $\mathcal{L}(e)$ can be uniformly interpreted as evaluations of linear maps in the variable e, which sets the stage for the explicit analysis of the linear dependence on the coordinates of e in the next step.

5.1.3 Step 3: Separating vinegar/oil coordinates and exploiting the fact that the oil part of the error is known

We decompose the error as

$$e = (e_V \| e_O), \qquad e_V \in \mathbb{F}_q^{n-o}, \quad e_O \in \mathbb{F}_q^o. \tag{16}$$

Under the fault model (5), the attacker knows

$$e_O = \delta e_0. \tag{17}$$

The vinegar part e_V is unknown (and, in the systematic model, coincides with $O(\delta e_0)$).

Since $\mathcal{L}$ is linear, it also decomposes uniquely into two linear contributions, one associated with the first $n-o$ coordinates and the other with the last o coordinates:

$$\mathcal{L}(e_V \| e_O) \;=\; \mathcal{L}_V(e_V) + \mathcal{L}_O(e_O), \tag{18}$$

where $\mathcal{L}_V : \mathbb{F}_q^{n-o} \to \mathbb{F}_q^m$ and $\mathcal{L}_O : \mathbb{F}_q^o \to \mathbb{F}_q^m$ are linear. Substituting (18) into (14) and using (17) yields:

$$\mathcal{L}_V(e_V) \;=\; \Delta t - \mathcal{L}_O(\delta e_0). \tag{19}$$

Equation (19) is a linear system that determines e_V from observable quantities: Δt is computed via verification, and $\mathcal{L}_O(\delta e_0)$ can be computed because both δe_0 and $\mathcal{L}$ are publicly evaluable (they depend on $P^\star$, on s', and on the matrices E_{ij}).

The behaviour of (19) depends on the rank of $\mathcal{L}_V$ and on the consistency of the corresponding linear system.

If the system is inconsistent, then the current hypothesis on the fault position or value cannot explain the observed mismatch Δt, and this hypothesis is discarded. If the system is consistent and $\mathcal{L}_V$ has full column rank $n-o$, then the solution for e_V is unique and recovery proceeds exactly as in the simplest case.

If, on the other hand, the system is consistent but $\mathcal{L}_V$ is rank-deficient, then (19) does not determine a unique vinegar part. In that case, the set of solutions is an affine subspace of dimension

$$(n-o) - \operatorname{rank}(\mathcal{L}_V).$$

Equivalently, once one particular solution has been found, all other solutions are obtained by adding elements of $\ker(\mathcal{L}_V)$. Thus the attacker does not obtain a single candidate e_V, but a family of candidates compatible with the same public mismatch Δt.

This does not invalidate the attack. Rather, it means that an additional search must be performed inside this affine solution space. Concretely, the attacker enumerates the candidate vinegar parts compatible with (19), reconstructs the corresponding full error using (16) and (17), corrects the faulty signature by replacing s'_0 with $s'_0 - e$, and checks the public verification equation. A candidate is accepted as a recovered error only if it restores a valid signature. If none of them does, the current hypothesis on the fault is discarded.

Therefore, in the linear route there are really three cases: inconsistent hypotheses are rejected immediately; full-rank consistent hypotheses yield a unique recovery; and consistent rank-deficient hypotheses lead to a residual brute-force search over the affine family of solutions. The practical relevance of this third case depends on the dimension of the kernel of $\mathcal{L}_V$: when it is small, exhaustive enumeration remains feasible; when it is large, the search space grows quickly. We quantify this behaviour experimentally later on.

If the affected coordinate or the value of the error δe_0 are not known exactly, the same logic is applied to each hypothesis in turn. The correct hypothesis is identified when the corresponding recovered error yields a corrected signature that satisfies the public verification equation.

5.2 Error Recovery II: Reduced Quadratic Recovery without Knowing α

The derivation up to (14) does *not* require the attacker to know the oil part of the error. What changes is only the recovery phase after the linear relation has been obtained. In the linear route, the decomposition $e = (e_V \| e_O)$ is exploited together with the fact that e_O is known. In the quadratic route, by contrast, the attacker keeps the same unknown $e \in \mathbb{F}_q^n$ as in (14) and does not separate it into known and unknown parts at the outset.

5.2.1 Step 1: Reusing the same linearization under the single-block fault model

The starting point is exactly the same linearization as in Error Recovery I. As in the linear attack, we assume that the perturbation remains confined to a single signature block, because this is what allows the observable mismatch to be explained by one block error e. What changes with respect to the linear route is not the number of affected blocks, but the structure allowed inside the faulty block: in the linear route the fault model is restricted to a single affected oil coordinate, whereas in the reduced-quadratic route several oil coordinates may be affected, provided that they all belong to the same block. Under this single-block model, the attacker again obtains the public linear relation

$$\mathcal{L}(e) = \Delta t, \tag{20}$$

where now $e \in \mathbb{F}_q^n$ is treated as a completely unknown block error.

This single-block assumption is essential for both attacks. If the fault affected several blocks, say with block errors $e_{i_1}, \ldots, e_{i_r}$, then the observable mismatch would no longer be described by one equation of the form $\mathcal{L}(e) = \Delta t$ in a single unknown vector. Instead, Δt would contain the sum of several block contributions,

$$\Delta t = \mathcal{L}_{i_1}(e_{i_1}) + \cdots + \mathcal{L}_{i_r}(e_{i_r}),$$

so errors coming from different blocks would be mixed together in the same observable quantity. The distinction between the two fault models is therefore not "single block" versus "multiple blocks", but rather "single oil coordinate inside one block" for the linear route versus "arbitrary nonzero oil vector inside one block" for the reduced-quadratic route. The effect of allowing several affected coordinates in the linear route is discussed later in the practicality analysis through the corresponding brute-force cost.

5.2.2 Step 2: Describing all linear candidates through a particular solution and the kernel

Assume that $\mathcal{L}$ has full rank m. Then (20) is a linear system with n unknown coordinates and m independent equations. Therefore, by the rank–nullity theorem, the solution set has dimension

$$n - m.$$

Equivalently, once one particular solution has been found, all other solutions differ from it by an element of the kernel of $\mathcal{L}$.

More precisely, let $e^{(0)} \in \mathbb{F}_q^n$ be any particular solution of

$$\mathcal{L}(e) = \Delta t,$$

and let

$$b_1, \ldots, b_{n-m}$$

be a basis of $\ker(\mathcal{L})$. Then every solution of (20) is uniquely of the form

$$e \;=\; e^{(0)} + \sum_{j=1}^{n-m} u_j b_j, \qquad u_1, \ldots, u_{n-m} \in \mathbb{F}_q. \tag{21}$$

If we arrange the kernel basis vectors as the columns of a matrix

$$K = \begin{bmatrix} b_1 & \cdots & b_{n-m} \end{bmatrix} \in \mathbb{F}_q^{n \times (n-m)},$$

and write

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_{n-m} \end{pmatrix} \in \mathbb{F}_q^{n-m},$$

then (21) becomes simply

$$e = e^{(0)} + Ku. \tag{22}$$

If we denote $u = (u_1, \ldots, u_{n-m})$, we may write this more compactly as

$$e = e(u).$$

More generally, if the system $\mathcal{L}(e) = \Delta t$ is consistent but $\mathcal{L}$ has rank

$$r < m,$$

then exactly the same description remains valid after replacing m by r. In that case, the solution set is an affine subspace of dimension $n - r$: one chooses a particular solution $e^{(0)}$, a basis $b_1, \ldots, b_{n-r}$ of $\ker(\mathcal{L})$, and writes

$$e = e^{(0)} + \sum_{j=1}^{n-r} u_j b_j$$

or equivalently $e = e^{(0)} + Ku$ with $u \in \mathbb{F}_q^{n-r}$. Thus the reduced-quadratic route still applies in the rank-deficient case, but the number of free variables increases from $n - m$ to $n - r$, which enlarges the nonlinear search space.

The linear relation $\mathcal{L}(e) = \Delta t$ does *not* determine the correct error uniquely, but it reduces the search space from all q^n vectors of $\mathbb{F}_q^n$ to an affine family. In the generic full-rank case this family has only q^{n-m} candidates; in the rank-deficient case it has q^{n-r} candidates. Thus, the public information Δt eliminates r degrees of freedom before the quadratic constraint is used, and eliminates m degrees of freedom in the full-rank case.

5.2.3 Step 3: Imposing the oil-space condition and obtaining the reduced quadratic system

The true induced error lies in the oil space $\mathcal{O}$, and therefore it also satisfies

$$P(e) = 0. \tag{23}$$

Substituting (21) into (23) gives the reduced system

$$g_a(u) \;:=\; p_a\big(e(u)\big) \;=\; 0, \qquad a = 0, \ldots, m-1, \tag{24}$$

that is, m quadratic equations in the free variables parametrizing the affine solution space of $\mathcal{L}(e) = \Delta t$. In the full-rank case above, these are the $n - m$ variables $u_1, \ldots, u_{n-m}$; more generally, if $\operatorname{rank}(\mathcal{L}) = r < m$, then the reduced system is written in $n - r$ variables. Since the substitution $e = e(u)$ is affine linear and each p_a is quadratic, each g_a has degree at most 2. For the official MAYO parameter sets one has $n - m = o$, so in the generic full-rank case the reduced system has exactly o variables: 8 for MAYO1, 17 for MAYO2, 10 for MAYO3, and 12 for MAYO5.

The recovery therefore proceeds as follows. Step 1 uses the faulty signature and public verification data to build the linear relation $\mathcal{L}(e) = \Delta t$. Step 2 parametrizes the affine family of candidates $e = e(u)$ that satisfy this relation. Step 3 keeps only those candidates that are compatible with the oil-space structure, using the reduced quadratic system (24).

5.2.4 Why this reduced quadratic system is practical

The public relation $\Delta t = \mathcal{L}(e)$ first restricts the unknown error to an affine space of dimension $n - \mathrm{rank}(\mathcal{L})$. After the parametrization $e = e^{(0)} + Ku$, the nonlinear part of the recovery is therefore the system (24) in only this number of variables. In the generic full-rank case, $\mathrm{rank}(\mathcal{L}) = m$, so the number of variables is $n - m = o$.

For the official MAYO parameter sets, the reduced systems over $\mathbb{F}_{16}$ have the following sizes: 78 equations in 8 variables for MAYO1, 64 equations in 17 variables for MAYO2, 108 equations in 10 variables for MAYO3, and 142 equations in 12 variables for MAYO5. For the official MAYO parameter sets, this yields small overdetermined systems; in our proof of concept, these systems were solved exactly over $\mathbb{F}_{16}$.

If $\mathcal{L}$ is rank-deficient, the same reduction is still used, but the reduced system has $n - \mathrm{rank}(\mathcal{L})$ variables instead of $n - m$. Such instances are not excluded by the method, although they increase the nonlinear solving cost.

5.2.5 Solver used in the proof of concept

Our proof of concept solves (24) exactly over $\mathbb{F}_q$. If $d = n - \mathrm{rank}(\mathcal{L})$ denotes the number of free variables left by the linear relation, we work in the polynomial ring

$$R = \mathbb{F}_q[u_1, \ldots, u_d],$$

form the ideal

$$I = \langle g_0, \ldots, g_{m-1} \rangle \subseteq R,$$

and enumerate the affine variety $V(I)(\mathbb{F}_q)$, that is, all tuples $u \in \mathbb{F}_q^d$ that satisfy all equations simultaneously. In the generic full-rank case, $d = n - m$.

In the implementation, this is done with SageMath via `ideal(...).variety(F16)`. Sage takes the system

$$g_0(u) = \cdots = g_{m-1}(u) = 0$$

and returns the tuples $u \in \mathbb{F}_{16}^d$ that satisfy all reduced equations. The call is exact: it works over the finite field, constructs the ideal generated by the equations g_a, and enumerates its $\mathbb{F}_{16}$-rational solutions.

The routine solves the reduced equations in the free variables $u_1, \ldots, u_d$, substitutes each solution back into the affine parametrization $e = e(u)$, and obtains candidate block errors. The solver works only with the reduced variables, not with the original n coordinates of e, which is what makes the proof of concept practical.

Each solution $u \in V(I)(\mathbb{F}_q)$ yields a candidate block error $e(u)$. The attacker then corrects the corresponding block of the faulty signature by subtracting $e(u)$ and runs the public verification algorithm. A candidate is accepted only if the corrected signature verifies.

6 Post-Processing: From Error Recovery to Key Recovery

The previous analysis shows that, from a single faulty signature, both recovery routes aim to recover the induced block error e. The central point is to exploit the structural information revealed by the fault: the induced error is not an arbitrary vector of $\mathbb{F}_q^n$, but an element of the secret subspace $\mathcal{O}$.

Indeed, in both fault settings considered above, the useful error is an oil-space error in one signature block and has the form

$$e = (O\alpha \| \alpha), \qquad \alpha \in \mathbb{F}_q^o \setminus \{0\},$$

so that $e \in \mathcal{O}$ and, moreover, $e \neq 0$. In the linear route, α is a single-coordinate oil perturbation, possibly identified by enumeration. In the reduced-quadratic route, α may have several nonzero coordinates and is recovered algebraically together with the full error. In both cases, after the linearization phase via $P^\star$ and the recovery of e, the attacker obtains *a nonzero vector of the secret subspace*.

From this point, the natural objective becomes the recovery of an equivalent systematic representation of the subspace $\mathcal{O}$, for instance through a matrix O. In particular, existing results for Oil–Vinegar-type schemes show how a single nonzero vector of the secret subspace, together with the public key, can be used to reconstruct the entire oil structure and obtain an equivalent secret key.

Recovery of the full oil structure from a single nonzero vector of the secret subspace was already achieved by Aulbach *et al.* [ACK+23], who combine an adapted Kipnis–Shamir step with the reconciliation attack. This line was later extended and made more direct by Pébereau [Péb24], who shows that one nonzero vector of the secret oil subspace is sufficient to recover an equivalent secret-key representation, and discusses extensions to UOV-related schemes such as MAYO. A direct reference for the concrete post-processing procedure used in our discussion and implementation is therefore the work *One Vector to Rule Them All: Key Recovery from One Vector in UOV Schemes* [Péb24].

7 Point of attack on a real implementation

For both attack variants, a relevant point of attack in a software implementation is the stage at which the solution of the signing linear system is written into the temporary vector that is later transformed into the signature block. In the public C implementation of MAYO provided in the repository `PQCMayo/MAYO-C` [BCC+26a], this occurs in the file `src/arithmetic.c`, inside the function `sample_solution`. The relevant code path is reproduced below.

Listing 1: Excerpt of `sample_solution` in `src/arithmetic.c` of `PQCMayo/MAYO-C`.

```
int sample_solution(const mayo_params_t *p, unsigned char *A,
                    const unsigned char *y, const unsigned char *r,
                    unsigned char *x, int k, int o, int m, int A_cols) {
    ...
    // x <- r
    for (int i = 0; i < k * o; i++) {
        x[i] = r[i];
    }

    // compute Ar
    unsigned char Ar[M_MAX];
    ...

    // move y - Ar to last column of matrix A
    for (int i = 0; i < m; i++) {
        A[k * o + i * (k * o + 1)] = sub_f(y[i], Ar[i]);
    }

    EF(A, m, k * o + 1);
    ...

    for (int row = m - 1; row >= 0; row--) {
        finished = 0;
        col_upper_bound = MAYO_MIN(row + (32/(m-row)), k*o);

        for (int col = row; col <= col_upper_bound; col++) {
```

```
27              correct_column =
28                  ct_compare_8((A[row * A_cols + col]), 0) & ~finished;
29
30              unsigned char u =
31                  correct_column & A[row * A_cols + A_cols - 1];
32              x[col] ^= u;  // linear recovery target
33
34              for (int i = 0; i < row; i += 8) {
35                  uint64_t tmp = ...;
36                  tmp = mul_fx8(u, tmp);  // reduced-quadratic target begins
37                  A[i * A_cols + A_cols - 1] ^= (tmp) & 0xf;
38                  A[(i+1) * A_cols + A_cols - 1] ^= (tmp >> 8) & 0xf;
39                  A[(i+2) * A_cols + A_cols - 1] ^= (tmp >> 16) & 0xf;
40                  ...                         // residual RHS update
41              }
42
43              finished = finished | correct_column;
44          }
45      }
46      return 1;
47  }
```

Linear recovery attack. For the linear recovery attack, the most direct implementation-level target is the update at line 32 of Listing 1,

```
x[col] ^= u;
```

because this is the point at which one coordinate of the solution vector x is written during back-substitution. A fault affecting this instruction may corrupt the value eventually stored in that coordinate and thereby induce an error in a single oil coordinate of one signature block. This matches the fault model used by the linear recovery attack, where the attacker later identifies the faulty block, coordinate, and value by exhaustive search and public verification. At this level, the attacker targets a concrete write or update operation on one coordinate of the temporary solution vector. The exact machine instruction and the practical injection window depend on the compiler, optimization level, register allocation, and target architecture, but the algebraic effect sought by the attacker is to perturb the value eventually stored in one coordinate of the block being signed.

Reduced-quadratic recovery attack. The same instruction `x[col] ^= u;` is also relevant to the reduced-quadratic recovery attack, since a single-coordinate error is a special case of the single-block model. However, for the reduced-quadratic setting there is a second implementation-level target in Listing 1: the subsequent block of assignments at lines 36–40 that updates the last column `A[*][A_cols-1]`. Concretely, this is the code path beginning with

```
tmp = mul_fx8(u, tmp);
```

and continuing with the assignments

```
A[(i+t) * A_cols + A_cols - 1] ^= ...
```

for the packed entries $t = 0, \ldots, 7$. Algebraically, this last column is the residual right-hand side of the partially solved system. Therefore, faulting this update does not directly overwrite a single coordinate of x; instead, it perturbs the residual term that will be used to reconstruct the remaining coordinates. Such a disturbance can produce, within one block, a multi-coordinate

error pattern compatible with the reduced-quadratic attack model. The C code stores the evolving residual right-hand side in place as the last column of the augmented matrix and updates only the entries that are still relevant for future iterations, namely those with index $i < \texttt{row}$.

If the goal is to keep the induced error confined to the block indexed by 0, this second type of fault should be injected only once the selected pivot column already belongs to that block, that is, when `col < o`. Since back-substitution processes pivot coordinates from larger indices towards smaller indices, block 0 is reached in the late part of the loop. For the official MAYO parameter sets, the corresponding block sizes are $o = 8, 17, 10, 12$ for MAYO1, MAYO2, MAYO3, and MAYO5, respectively.

More generally, the precise low-level location of the fault is implementation-dependent. A similar effect could arise not only during the in-place update of the solution vector, but also during later data movement, for instance if the computed vector were copied to another buffer before signature assembly. In a software setting, a corrupted `memcpy`-like transfer, a faulty load/store, or an erroneous register-to-memory write could therefore serve as an alternative injection point. Depending on the fault model, such a disturbance may affect one coordinate or several coordinates of the same block, which matches the distinction between the linear and reduced-quadratic variants discussed above.

8 Experimental simulation

Our experimental evaluation is a software proof of concept carried out in SageMath, consistently with the fact that the public MAYO reference algebraic code used as a basis for the simulations is itself provided in Sage in the repository `PQCMayo/MAYO-sage` [BCC+26b]. All simulations were executed on an HP ProBook 450 G9 equipped with a 12th Gen Intel(R) Core(TM) i7-1255U CPU at 1700 MHz, with 10 physical cores and 12 logical processors.

The goal of this section is not to claim a hardware-level reproduction of the fault, but to emulate in the reference algebraic implementation the effect of perturbing the computation of the solution vector during signing. The relevant code is the function `_sample_solution` in `mayo.sage` from `PQCMayo/MAYO-sage` [BCC+26b], reproduced below.

Listing 2: Function `_sample_solution` from `mayo.sage` in `PQCMayo/MAYO-sage`.

```
def _sample_solution(self, A, y, r):
    """
    takes as input a matrix A in F_q^{m x n} of rank m with n >= m,
    a vector y in F_q^m, and a vector r in F_q^n
    and outputs a solution x such that Ax = y
    """

    use_sage_linear_albegra = False

    if use_sage_linear_albegra:
        if A.rank() != self.m:
            return None
        x = A.solve_right(y - A*r)

        assert A*x == y - A*r
        return x + r

    # Above is the easy 'SAGE' way. To test if the spec is correct,
    # we implement it below without using A.solve_right
    x = r
    y -= A*r
```

```
22
23      Augmented_matrix = A.augment(matrix(self.m,1,y))
24      Augmented_matrix = self._ef(Augmented_matrix)
25
26      A = Augmented_matrix[:,0:self.k*self.o]
27      y = Augmented_matrix.column(self.k*self.o)
28
29      last_row_zero = True
30      for i in range(self.k*self.o):
31          if A[self.m-1,i] != 0:
32              last_row_zero = False
33              break
34
35      if last_row_zero:
36          return None
37
38      for r in range(self.m-1,-1,-1):
39          c = 0
40          while A[r,c] == 0:
41              c += 1
42          x[c] += y[r]                    # linear recovery target
43          y -= vector(y[r]*A[:,c])        # reduced-quadratic target
44
45      return x
```

For the experiments reported in this work, we do *not* use the direct Sage linear-algebra call `A.solve_right(y - A*r)`. Instead, we work with the explicit manual code path shown in Listing 2, that is, the row-elimination and back-substitution procedure appearing below the flag `use_sage_linear_albegra`. This is important because it provides a concrete algebraic location where a fault can be injected during the reconstruction of the solution vector.

Concretely, for the linear attack the simulated fault is introduced at the update appearing at line 42 of Listing 2,

```
x[c] += y[r]
```

in the back-substitution loop of Listing 2. This is the Sage-level analogue of the C-level update discussed in Section 7: rather than faulting an abstract variable after the solve has finished, the simulation perturbs the step at which one coordinate of the solution vector is computed.

For the reduced-quadratic variant, the direct update `x[c] += y[r]` is again a valid single-coordinate target. The second Sage-level analogue, which may affect several coordinates inside one block, is the immediately following residual update

```
y -= vector(y[r] * A[:,c]).
```

This line plays the same algebraic role as the updates to `A[*][A_cols-1]` in Listing 1: in both cases, the code modifies the residual right-hand side that drives the remaining back-substitution steps. The correspondence is not purely syntactic, because the C implementation stores this residual term in place as the last column of the augmented matrix, whereas the Sage code keeps it as an explicit vector y. Moreover, the C code updates only the entries of that last column that still matter for future iterations, namely the rows above the current pivot, while the Sage code writes the same effect in the compact algebraic form `y -= vector(y[r] * A[:,c])`. For our purposes these two views are equivalent: in both cases, the fault perturbs the residual right-hand side seen by the remaining pivot computations. In the Sage loop, the relevant condition for targeting block 0 is $c < o$. Since back-substitution goes from larger pivot indices towards smaller ones, this condition corresponds to the late part of the loop and keeps the remaining coordinate updates inside block 0. For the official MAYO parameter sets, the corresponding block sizes are

$o = 8, 17, 10, 12$ for MAYO1, MAYO2, MAYO3, and MAYO5, respectively. The precise row at which this late regime begins depends on the pivot pattern of the echelon form. As discussed below, faulting this update in that window yields an error that remains confined to a single block while possibly affecting several of its oil coordinates.

The simulation framework then evaluates, for each attack variant, whether the induced faulty signature yields an observable mismatch Δt from which the error can be recovered and, subsequently, whether the recovered vector of the oil space is sufficient to trigger the post-processing phase towards full key recovery.

8.1 Linear recovery attack

For the linear recovery attack, the experimental goal is to validate the complete end-to-end workflow under the single-coordinate fault model described in Section 5.1. The proof of concept is built on top of the official Sage implementation of MAYO provided in the repository `PQCMayo/MAYO-sage` [BCC+26b]. The simulation is run independently for each official MAYO parameter set. Concretely, we instantiate the corresponding parameter object from the public specification, namely MAYO1, MAYO2, MAYO3, or MAYO5, and then perform 100 independent executions for that parameter set.

Each execution starts from fresh randomness. A new message is sampled at random, a new compact key pair is generated, and the secret and public keys are then expanded to obtain the expanded secret key and expanded public key used by the signing and verification procedures. The faulted signing experiment is then run once on that fresh instance. In this setting, the simulated fault modifies a single pivot coordinate during the manual back-substitution stage of the signing solver, so that the induced error matches the linear single-coordinate fault model.

For validation purposes only, the experiment also generates a non-faulty signature using the same signing randomness as the faulty one. This non-faulty signature is not given to the recovery procedure. It is used only to classify the real observable error and to check whether the recovered error is exactly the same as the difference observed between the non-faulty and faulty signatures.

After obtaining the faulty signature, the experiment applies the public recovery procedure of the linear attack. This procedure exhaustively tests the candidate fault hypotheses consisting of a block index, an oil coordinate inside that block, and a nonzero fault value in $\mathbb{F}_{16}$. For each hypothesis, the corresponding linear system is built from the public verification mismatch Δt, the candidate error is reconstructed, the faulty signature is corrected, and the corrected signature is checked by public verification. When a valid correction is found, the recovered vector is then passed to the post-processing phase that attempts to recover the secret oil matrix from that vector.

The output recorded for each execution is therefore not limited to whether the linear system was solvable. We also record whether the correct fault block was identified, whether the corrected signature verifies, and whether the subsequent one-vector post-processing step recovers the same oil matrix O as the one used to generate the fresh secret key of that execution. This last condition should be interpreted as exact recovery of the secret oil matrix in the representation used by the experiment, not merely as the recovery of some unspecified equivalent key. Repeating this process 100 times for each MAYO parameter set provides an experimental estimate of how often the complete linear recovery route succeeds on independently generated instances.

Table 1 summarizes these experiments for the four official MAYO parameter sets. The column "Exact error recovered" counts the executions in which the recovered error is exactly equal to the real observable error obtained by comparing a non-faulty and a faulty signature generated with the same signing randomness. The column "Rank deficient" counts the executions in which the linear system associated with the real fault guess has a vinegar matrix M_V of rank strictly

smaller than $n - o$. The column "Def. kernel dim." reports the nullity $d = (n - o) - \mathrm{rank}(M_V)$ only for those rank-deficient executions; in all our rank-deficient linear experiments this value was 1. The column "Post-proc. fails" counts the executions in which a candidate error is recovered but the subsequent one-vector post-processing step does not return the same oil matrix O used in that execution. The final column, "Secret subspace recovered", counts the executions in which the whole linear recovery route succeeds end-to-end and the recovered oil subspace is equivalent to the original secret oil subspace.

Table 1: Experimental summary of the linear recovery attack. Each row corresponds to 100 independent executions with fresh keys and messages for the indicated MAYO parameter set.

Parameter set	**Exec.**	**Exact error recovered**	**Rank deficient**	**Def. kernel dim.**	**Post-proc. fails**	**Secret subspace recovered**
MAYO1	100	100	6/100	1	0	100
MAYO2	100	100	5/100	1	0	100
MAYO3	100	100	8/100	1	0	100
MAYO5	100	100	9/100	1	0	100

8.2 Reduced quadratic recovery attack

For the reduced-quadratic recovery attack, the experimental goal is to validate the complete end-to-end workflow under the relaxed single-block fault model described in Section 5.2. As in the linear case, the proof of concept is built on top of the official Sage implementation of MAYO provided in the repository `PQCMayo/MAYO-sage` [BCC+26b]. The simulation is run independently for each official MAYO parameter set. Concretely, we instantiate MAYO1, MAYO2, MAYO3, or MAYO5, and then perform 100 independent executions for that parameter set.

Each execution starts from fresh randomness. A new message is sampled at random, a new compact key pair is generated, and the secret and public keys are expanded. The faulted signing experiment is then run once on that fresh instance. In this setting, the simulated fault perturbs the residual right-hand side during the manual back-substitution stage of the signing solver, in the late window where the pivot coordinate already belongs to one signature block. This yields an induced error that remains confined to a single block while possibly affecting several oil coordinates inside that block, which matches the relaxed fault model of the reduced-quadratic route.

For validation purposes only, the experiment also generates a non-faulty signature using the same signing randomness as the faulty one. This non-faulty signature is not given to the recovery procedure. It is used only to classify the real observable error and to check whether the recovered error is exactly the same as the difference observed between the non-faulty and faulty signatures.

After obtaining the faulty signature, the experiment applies the public recovery procedure of the reduced-quadratic attack. In contrast with the linear route, this procedure does not guess the affected oil coordinate or the nonzero fault value. Instead, for each candidate block, it builds the public linear relation $\mathcal{L}(e) = \Delta t$, parametrizes the affine space of solutions, and imposes the oil-space condition $P(e) = 0$. This gives the reduced quadratic system described in Section 5.2. Each solution gives a candidate error; the faulty signature is corrected with that candidate and the result is checked by public verification. When a valid correction is found, the recovered vector is passed to the same post-processing phase used in the linear experiments. In our experiments, every successful execution produced a single verifying candidate, which was then confirmed, using the non-faulty reference signature available only for validation, to coincide with the induced error.

The output recorded for each execution includes whether the recovered error is exactly equal to the real observable error, whether the linear relation for the real block is rank-deficient, the number of free variables left in the reduced quadratic system, whether the one-vector post-processing step fails after a candidate error is found, and whether the complete recovery route recovers the secret oil structure. Repeating this process 100 times for each MAYO parameter set provides an experimental estimate of how often the complete reduced-quadratic recovery route succeeds on independently generated instances.

Table 2 summarizes these experiments for the four official MAYO parameter sets. The column "Exact error recovered" counts the executions in which the recovered error is exactly equal to the real observable error obtained by comparing a non-faulty and a faulty signature generated with the same signing randomness. The column "Rank deficient" counts the executions in which the linear relation for the real block has rank strictly smaller than m. The column "Free vars" reports the number of variables left in the reduced quadratic system for the real block, namely $n - \mathrm{rank}(\mathcal{L})$; in the generic full-rank case this value is $n - m = o$. The column "Post-proc. fails" counts the executions in which a candidate error is recovered but the subsequent one-vector post-processing step does not return the expected oil structure. The final column, "Secret subspace recovered", counts the executions in which the whole reduced-quadratic recovery route succeeds end-to-end.

Table 2: Experimental summary of the reduced-quadratic recovery attack. Each row corresponds to 100 independent executions with fresh keys and messages for the indicated MAYO parameter set.

Parameter set	**Exec.**	**Exact error recovered**	**Rank deficient**	**Free vars**	**Post-proc. fails**	**Secret subspace recovered**
MAYO1	100	100	0/100	8	0	100
MAYO2	100	100	0/100	17	0	100
MAYO3	100	100	0/100	10	0	100
MAYO5	100	100	0/100	12	0	100

9 Comments on the Practicality of the Attack

The practical question is not only whether the algebraic recovery works once the right fault has been obtained, but also whether an attacker can recognize and exploit useful faulty outputs. In the setting considered here, a useful faulty signature must satisfy two structural requirements: the induced error must belong to the secret oil space, and the effect must remain confined to one signature block. If the device crashes, outputs a malformed signature, or produces a valid signature with $\Delta t = 0$, the execution does not provide the mismatch needed by the attack. If the output is well formed but invalid, the attacker can compute

$$\Delta t = P^{\star}(s') - t$$

from public information and then test whether this mismatch is compatible with one of the fault models below.

This public filtering means that the attacker does not need to know in advance that a given physical perturbation was exploitable. Instead, faulty executions can be collected and processed offline: inconsistent hypotheses are discarded by the linear systems or by public verification after correction, while successful hypotheses return a candidate error e whose correction makes the signature verify. The cost of this filtering is different for the two recovery routes.

9.1 Practicality of the linear recovery route

The linear route is the cheaper algebraic recovery method, but it uses the more restrictive fault model. It assumes that the induced perturbation affects a single oil coordinate of one signature block, or at least that the attacker can enumerate the affected coordinate and the nonzero value. Once the oil part e_O is fixed by such a hypothesis, the unknown part of the error is only the vinegar part e_V, and recovery reduces to the linear system (19).

If the affected block, coordinate, and value are not known, the attacker enumerates them. For a single-coordinate fault $x_{i^\star}[\ell]$, with $i^\star \in \{0, \ldots, k-1\}$ and $\ell \in \{0, \ldots, o-1\}$, there are $k \cdot o$ possible positions and $q-1$ possible nonzero values. Since MAYO uses $q = 16$, the number of hypotheses is

$$N_{\text{lin}} \;=\; k \cdot o \cdot (q-1) \;=\; 15ko. \tag{25}$$

For the official parameter sets this gives 1200 hypotheses for MAYO1, 1020 for MAYO2, 1650 for MAYO3, and 2160 for MAYO5. Each hypothesis requires building the corresponding public linear system, solving for e_V, reconstructing the full candidate error, correcting the faulty block, and checking public verification.

This cost is small because the exhaustive search is only over one field element and one oil-coordinate position. The limitation is that this route does not naturally absorb multi-coordinate oil faults. If t oil coordinates may be affected but the perturbation is still assumed to remain inside a single block, then the attacker must guess one block, t coordinates inside that block, and t nonzero values. The number of hypotheses becomes

$$N_t \;=\; k\binom{o}{t}(q-1)^t, \tag{26}$$

or $\binom{o}{t}(q-1)^t$ if the faulty block is known. This count deliberately does not choose coordinates among all ko oil positions, because such a count would also include errors spread across several blocks, which is outside the single-block model used by both attacks. Even under this single-block restriction, the search quickly becomes impractical as t increases. Thus, the linear route is best interpreted as a fast recovery method for simple, localized, single-coordinate oil faults.

9.2 Practicality of the reduced-quadratic recovery route

The reduced-quadratic route is designed for the case where the oil part of the error is not known. It keeps $e \in \mathbb{F}_q^n$ as an unknown vector, uses the public relation $\mathcal{L}(e) = \Delta t$ to parametrize an affine space of candidates, and then imposes the oil-space condition $P(e) = 0$. Consequently, the attacker does not enumerate the affected oil coordinate or the induced value. If the faulty block is unknown, the outer search is only over the k possible block hypotheses:

$$N_{\text{quad}} = k. \tag{27}$$

For MAYO1, MAYO2, MAYO3, and MAYO5 this gives 10, 4, 11, and 12 block hypotheses, respectively.

For each block hypothesis, the attacker builds the public relation $\mathcal{L}(e) = \Delta t$. It then computes an affine parametrization of its solution space, substitutes this parametrization into $P(e) = 0$, and solves the reduced quadratic system (24). In the generic full-rank case, the number of remaining variables is $n - m = o$, namely 8, 17, 10, and 12 variables for MAYO1, MAYO2, MAYO3, and MAYO5. The dominant cost is therefore not the block search, but the exact solution of this reduced MQ system over $\mathbb{F}_{16}$.

Unlike the linear route, this second route can accommodate an error affecting several oil coordinates inside the same signature block, because the entire oil vector α is absorbed into the unknown error $e = (O\alpha \| \alpha)$. Equivalently, it avoids the brute-force factor $k\binom{o}{t}(q-1)^t$ by solving for the whole block error algebraically. The crucial restriction is instead that the error

must remain confined to one block and must still correspond to an element of the oil space $\mathcal{O}$. If the fault mixes several signature blocks, or produces an error that is no longer modeled as a single vector $e \in \mathcal{O}$, then the mismatch is no longer described by one single-block equation $\mathcal{L}(e) = \Delta t$, and the reduced-quadratic construction no longer applies in the same form.

9.3 Rank-deficient cases

Both recovery variants are simplest in the full-rank case, but rank deficiency affects them differently. In the linear route, the map $\mathcal{L}_V$ in (19) should have full column rank to determine e_V uniquely. If it is rank-deficient but the system is consistent, the attacker obtains an affine family of possible vinegar parts and must search inside this family, checking candidates by public verification. The additional cost is governed by the kernel dimension $(n-o) - \mathrm{rank}(\mathcal{L}_V)$.

In the reduced-quadratic route, a rank-deficient $\mathcal{L}$ does not invalidate the method, but it increases the number of free variables in the reduced system. If $r = \mathrm{rank}(\mathcal{L}) < m$, then the affine solution space has dimension $n-r$ rather than $n-m$. The reduced system is still obtained by substituting $e = e(u)$ into $P(e) = 0$, but it is larger and therefore harder to solve.

For a given concrete faulty signature, one of these rank conditions may fail for the correct block hypothesis. In that case, the information contained in Δt remains valid, but the corresponding recovery procedure becomes more expensive than in the generic full-rank description. Our proof of concept records the rank behaviour in both experimental settings. In the reduced-quadratic experiments, rank-deficient instances are not discarded: the same parametrization is used with $n - \mathrm{rank}(\mathcal{L})$ free variables, and the larger reduced system is then passed to the MQ solver.

10 Countermeasures

The attacks described in this work rely on two structural conditions. First, the induced error must be compatible with the secret oil space, namely it must have the form

$$e = (O\alpha \| \alpha) \in \mathcal{O}$$

for some nonzero $\alpha \in \mathbb{F}_q^o$. Second, the effect of the fault must remain confined to a single signature block, so that the observable mismatch can be written as one public linear relation

$$\Delta t = \mathcal{L}_i(e).$$

The linear and reduced-quadratic routes exploit these conditions differently. The linear route additionally benefits from knowing, or cheaply enumerating, the affected oil coordinate and value. The reduced-quadratic route does not need this knowledge, but it still requires an isolated single-block error belonging to $\mathcal{O}$. Countermeasures should therefore be evaluated against both routes separately.

Natural countermeasures can be organized around three main principles.

- **Internal verification before output.** The most direct countermeasure is to verify the signature internally before releasing it. Given a candidate signature s', the signer recomputes the target t and outputs the signature only if

  $$P^\star(s') = t.$$

  This measure blocks both recovery routes considered in this paper, since the attacker never obtains an invalid faulty signature from which to compute

  $$\Delta t = P^\star(s') - t.$$

  This is especially relevant here because the attacks exploit invalid but algebraically structured faulty signatures, rather than producing another valid signature.

- **Protecting the signing solver and oil-space encoding.** The implementation can protect the computation that produces the oil coefficients before these coefficients are used to construct the signature. This includes both direct faults on the final solution vector x and faults on intermediate solver state. For example, after solving the signing linear system, the implementation may recompute the residual and check that the obtained vector still satisfies the expected relation

$$Ax = y$$

or the equivalent MAYO signing equation used internally. Alternatively, the back-substitution can be recomputed, preferably with a different instruction schedule or representation, and the two solver outputs can be compared.

This point is important for both attack routes. The linear route can be instantiated by faulting the update that writes one coordinate of x. The reduced-quadratic route can instead be instantiated by perturbing the residual right-hand side during back-substitution, before the remaining coordinates of x are reconstructed. Therefore, checking only the value of x after it has already been transformed into a signature block is too late unless the check detects the corrupted solver output before release.

The subsequent transformation from oil coefficients to an oil-space vector should also be included in this protected computation. In particular, the multiplication $\alpha_i \mapsto O\alpha_i$ can be recomputed or checked before the block $s_i = v_i + o_i$ is formed. The goal is to prevent a single local fault from coherently producing both parts of

$$o_i = (O\alpha_i \| \alpha_i) \in \mathcal{O}.$$

A fault that only corrupts one side of this encoding is less useful for the attacks, because the induced error is then not guaranteed to have the required form $e = (O\alpha \| \alpha)$. However, when the fault is injected earlier in the computation of α_i, both α_i and $O\alpha_i$ may be generated consistently from the corrupted value. For this reason, the solver stage and the oil-space encoding should be protected as one computation path, not as unrelated checks.

- **Breaking the structural fault conditions.** The remaining defenses should aim at invalidating at least one of the two structural conditions used by the attacks. The first condition is that the induced error belongs to the secret oil space. Against the reduced-quadratic route in particular, hiding the affected oil coordinate is not sufficient, because that route is designed to work without knowing α. The attack must be prevented from producing an isolated single-block oil-space error of the form

$$e = (O\alpha \| \alpha) \in \mathcal{O}.$$

Countermeasures should therefore make local corruptions either detectable or incompatible with the algebraic relation between the first $n - o$ coordinates and the last o coordinates that characterizes $\mathcal{O}$.

The second condition is that the useful error is confined to one signature block. If a fault affects several blocks, then the mismatch no longer has the form

$$\Delta t = \mathcal{L}_i(e).$$

Instead, it contains several contributions,

$$\Delta t = \mathcal{L}_{i_1}(e_{i_1}) + \mathcal{L}_{i_2}(e_{i_2}) + \cdots,$$

and the recovery procedures described in this paper no longer apply directly. Interleaving or jointly checking the computation of several signature blocks, randomizing the order

in which block coordinates are processed, or adding cross-block consistency checks can make isolated single-block faults harder to obtain. These mechanisms should be treated as structural hardening: they are useful only if they prevent an isolated single-block oil-space error, not merely if they obscure which coordinate was affected.

In summary, effective countermeasures should ensure at least one of the following properties: the faulty signature is not released; faults in the solver or oil-space encoding are detected before signature construction; the induced error is not an element of the oil space; or the induced error is not confined to a single signature block. A countermeasure that only prevents the attacker from knowing the affected oil coordinate may block the linear route, but it does not necessarily block the reduced-quadratic route.

11 Conclusions

We presented two fault-injection attacks against MAYO in which a single faulty execution can reveal a nonzero vector of the secret oil subspace. The attacks are based on the observation that, under a single-block oil-space fault, the public verification mismatch can be written as a linear function of the induced error. This gives a linear recovery route when the oil part of the error is known or enumerable, and a reduced-quadratic route when it is not.

The recovered vector can be used as input to existing post-processing techniques for UOV-type schemes, leading to recovery of an equivalent secret representation. In contrast to several previous attacks, our approach does not rely on deterministic signing, repeated vinegars, predictable vinegar seeds, a non-faulty reference signature, or an adaptive correction phase over many faulty signatures. The recovery uses only the faulty signature, the corresponding message and salt, and the public key, which is why the attack applies to randomized MAYO under the stated single-block fault model.

Our software simulations validate the proposed recovery procedures and illustrate the different practical profiles of the two routes. The linear route is cheaper but assumes a simple localized fault, while the reduced-quadratic route is more general inside one block at the cost of solving a small reduced quadratic system. The analysis also shows that countermeasures should be assessed against both variants: preventing the linear shortcut does not necessarily prevent the reduced-quadratic attack.

References

[ABCR25] Mohamed Abdelmonem, Lejla Batina, Durba Chatterjee, and Håvard Raddum. Correction-based fault attack against randomized mayo. IACR Cryptology ePrint Archive, Report 2025/2163, 2025.

[ACK+23] Thomas Aulbach, Fabio Campos, Juliane Krämer, Svetla Nikova Samardjiska, and Markus Stöttinger. Separating oil and vinegar with a single trace: Side-channel assisted kipnis-shamir attack on uov. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):221–245, 2023.

[ACK25] Thomas Aulbach, Fabio Campos, and Juliane Krämer. Sok: On the physical security of uov-based signature schemes. In *Post-Quantum Cryptography (PQCrypto 2025)*, Lecture Notes in Computer Science, pages 199–231. Springer, 2025.

[AMSU24] Thomas Aulbach, Soundes Marzougui, Jean-Pierre Seifert, and Vincent Quentin Ulitzsch. Mayo or may-not: Exploring implementation security of the post-quantum signature scheme mayo against physical attacks. In *Workshop on Fault Detection and Tolerance in Cryptography (FDTC 2024)*, pages 28–33. IEEE, 2024.

[BCC+25] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. Mayo specification (round-2 submission). Submitted to NIST PQC (Additional Digital Signature Schemes), Round-2, 2025. Latest update: February 5, 2025. Resources: `https://pqmayo.org/resources/`.

[BCC+26a] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. *MAYO C implementation*, June 2026. Available at `https://github.com/PQCMayo/MAYO-C`. Accessed June 11, 2026.

[BCC+26b] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. *MAYO Sage implementation*, June 2026. Available at `https://github.com/PQCMayo/MAYO-sage`. Accessed June 11, 2026.

[Beu22] Ward Beullens. Mayo: Practical post-quantum signatures from oil-and-vinegar maps. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 355–376, Cham, 2022. Springer International Publishing.

[BSK25] Sven Bauer, Fabrizio De Santis, and Kristjane Koleci. Fault attacks against uov-based signatures. Cryptology ePrint Archive, Paper 2025/2101, 2025.

[JD25] Sönke Jendral and Elena Dubrova. Mayo key recovery by fixing vinegar seeds. *IACR Communications in Cryptology*, 1(4), 2025.

[Nat24a] National Institute of Standards and Technology. Module-lattice-based digital signature standard. NIST FIPS 204, National Institute of Standards and Technology, Gaithersburg, MD, 2024.

[Nat24b] National Institute of Standards and Technology. Stateless hash-based digital signature standard. NIST FIPS 205, National Institute of Standards and Technology, Gaithersburg, MD, 2024.

[Péb24] Pierre Pébereau. One vector to rule them all: Key recovery from one vector in uov schemes. In *Post-Quantum Cryptography (PQCrypto 2024), Part II*, pages 92–108. Springer, 2024.

[Zho25] Yadi Zhong. Variables for free: Fault injection attack on mayo via valid solutions. IACR Cryptology ePrint Archive, Report 2025/1808, 2025.