MAYo or MAY-not: Exploring Implementation Security of the Post-Quantum Signature Scheme MAYO Against Physical Attacks

Thomas Aulbach (University of Regensburg)
Soundes Marzougui (STMicroelectronics)
Jean-Pierre Seifert (TU Berlin -- SECT)
Vincent Ulitzsch vincent@sect.tu-berlin.de(TU Berlin -- SECT)

SECT Security IN TELECOMMUNICATIONS

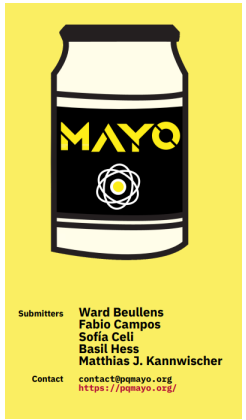
Attacking MAYO with loop-abort/Zeroing faults

1	MAYO is post-quantum secure multivariate signature scheme notable for its efficient and compact key size
2	This paper: Studying the resilience of MAYO against fault injection attacks
3	We present loop-abort fault attacks on MAYO, which allow for full key recovery with only one fault

Motivation: MAYO is promising multi-variate digital signature candidate, but only limited research on Mayo's resilience against physical attacks

Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process

Updated October 2022 to reflect that IP statements can be accepted digitally.



- MAYO, a multi-variate signature scheme was submitted, to the NIST call for additional post-quantum secure digital signature schemes
- MAYO features compact key and signature size and is notable for its efficiency
- MAYO has so far withstood cryptanalysis
- But: Only limited research on MAYO's implementation security against physical attack exists

Research Question: What is MAYO's attack surface against fault injection attacks?

Recap: Multivariate Signature Schemes -- Signing

- Multivariate cryptographic schemes base their security on the hardness of solving a set of multivariate quadratic equations over a small finite field F --- One instantiation: **Unbalanced oil and vinegar schemes**
- **Public Key**: Trapdoored homogenous multivariate map:

$$P(x) = (p_1(x), \dots, p_m(x)) : \mathbb{F}_q^n \to \mathbb{F}_q^m$$

where $p_1(x), \dots, p_{m(x)}$ are multivariate quadratic polynomials in n variables x=$(x_1, \dots, x_n)$

- **Trapdoor/Private Key**: Secret subspace $O \subseteq \mathbb{F}_q^n$ (oil space) of dimension m on which $P(x)$ evaluates to zero
- **Signing**: Hash message to target vector $t \in F_q^m$ and find preimage under map P -- Doing so without the trapdoor is hard.

Recap: Multivariate Signature Schemes -- Signing

- Signing a message: Find a preimage s to target vector t under map P
- **Observation**: This is easy to do given access to the oil space!
  - Define the polar form or differential of map P of a multi-variate quadratic polynomial as:

$$P'(x, y) = P(x + y) - P(x) - P(y)$$

  - To find a pre-image of a target vector t, having knowledge of the oil space O:
    1. first pick a random vinegar vector v and
    2. then solve $P(v + o) = t$ for a vector $o \in O$ by solving the linear equation system:

$$\underbrace{P'(v, o)}_{\text{Linear in } o} + \underbrace{P(o)}_{= 0} + \underbrace{P(v) = t}_{\text{fixed}}$$

Recap: MAYO

- **Drawback of UOV schemes**: UOV schemes have a small signature size, secure parameter sets lead to key pairs of enormous size (e.g., 50 KB)
- **Key Idea Of Mayo**: **Stretch the public key map P into a larger map**, such that it accepts k input vectors $x_i \in \mathbb{F}_q^n$

$$\mathcal{P}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1 \le i < j \le k} \mathbf{E}_{ij}(\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j)),$$

  where E_i are fixed, public, m-by-m matrices

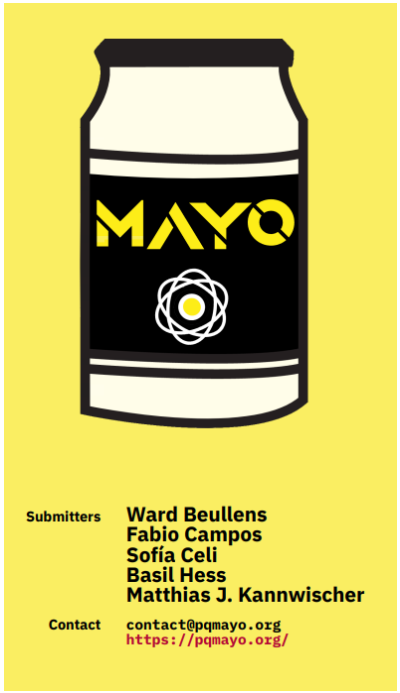
- **This allows for small parameters and public key and signature sizes**. For example, in Mayo NIST-Security Level 1 (=as hard as breaking AES-128):
  - **Params:** $\dim(O) = 8, m = 64, n = 66, q = 16, k = 9$**, Public Key**: 1168 Bytes, **Signature** 321 Bytes

Recap: MAYO Signature Algorithm

Algorithm 2 $\text{Sign}(M, \text{sk})$

1: $(\text{seed}, \mathbf{O}) \leftarrow \text{sk}$
2: $\text{salt} \leftarrow \{0,1\}^{2\lambda}$
3: $t \leftarrow \text{Hash}(M \| \text{salt})$
4: $\mathcal{P}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k) \leftarrow \sum_{i=1}^{k} E_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1 \le i < j \le k} E_{ij}\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j)$
5: $\mathbf{v}_i \leftarrow \mathbb{F}_q^{n-m} \times \{0\}^m$
6: **if** $\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k)$ does not have full rank **then**
7: return to step 5
8: **end if**
9: Solve $\mathbf{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t}$ for $\mathbf{o}_1, \ldots, \mathbf{o}_k \in \text{RowSpace}(\mathbf{O} \| \mathbf{I}_o)$.
10: **return** $\sigma = (\text{salt}, \{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i \in [k]})$

MAYO is a promising candidate, but research on resilience against physical attacks is missing



- Given its compactness and efficiency MAYO is a promising PQC Signature candidate
- More compact than ML-DSA (standardized Dilithium) and Falcon (selected for standardization)
- We can expect deployment on embedded systems, which need to be secure against physical attacks

Research Question: What is MAYO's attack surface against fault injection attacks?

This Paper: Loop-abort attacks against Mayo

We present two, first-order fault-injection attack scenarios against MAYO that allow full key recovery

Fault Injection attacks against Mayo

Faulting the sampling of one out of the k vinegar vectors reveals information about the corresponding oil vector --- which can allow for secret key recovery.

- Key Idea: Prior work has shown that **if an attacker knows one oil vector, recovering the entire oil space is merely a matter of solving linear equations** (with techniques borrowed from the reconciliation attack, see [Beu21b])
- Thus: If we can disturb the sampling of a vinegar vector, this reveals information about the corresponding oil vector, allowing us to recover the secret key!
  - Prior work has successfully attacked Rainbow by faulting the sampling of the vinegar vectors (and fault attacks on Rainbow [AKKM22]) – we combine this attack with the reconciliation attack to only use one fault)

This Paper: Loop-abort attacks against Mayo

We present two, first-order fault-injection attack scenarios against MAYO that allow full key recovery

We will establish the result in two steps:

1. **Given access to an oil vector, how can we recover the entire oil space?**
2. How can we can reveal an oil vector from a faulted signature?

Reconciliation attack for MAYO

Assume we recovered an oil vector $o_1 \in O$ and want to recover an additional oil vector $o_2 \in O$. We need to solve the system:

$$P(o_2) = 0$$
$$P'(o_1, o_2) = 0$$

- For a given o_1 the differential map $\mathrm{P}'(o_1, o_2) = 0$ imposes m linear equations in the entries of o_2 (each polynomial in $P'(o_1, o_2)$ provides a P'linear equation in n variables)
- Since $o_2 \in \mathbb{F}_q^n$ this system is under-constrained.
- But: $\dim(O) > n - m$, that is, we can fix $n - m$ entries of o_2 and solve linear system
- Full key recovery in $O(o \cdot m^3)$ (for all security levels, $o, m, n < 150$)

This Paper: Loop-abort attacks against Mayo

We present two, first-order fault-injection attack scenarios against MAYO that allow full key recovery

We will establish the result in two steps:

1. Given access to an oil vector, how can we recover the entire oil space?
2. **How can we can reveal an oil vector from a faulted signature?**

Recap: MAYO Signature Algorithm

Algorithm 2 $\text{Sign}(M, \text{sk})$

1: $(\text{seed}, \mathbf{O}) \leftarrow \text{sk}$
2: $\text{salt} \leftarrow \{0,1\}^{2\lambda}$
3: $t \leftarrow \text{Hash}(M \| \text{salt})$
4: $\mathcal{P}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k) \leftarrow \sum_{i=1}^{k} E_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1 \le i < j \le k} E_{ij}\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j)$
5: $\mathbf{v}_i \leftarrow \mathbb{F}_q^{n-m} \times \{0\}^m$
6: **if** $\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k)$ does not have full rank **then**
7: return to step 5
8: **end if**
9: Solve $\mathbf{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t}$ for $\mathbf{o}_1, \ldots, \mathbf{o}_k \in \text{RowSpace}(\mathbf{O} \| \mathbf{I}_o)$.
10: **return** $\sigma = (\text{salt}, \{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i \in [k]})$

```
for i=1 to k:
      v[i] = decode(shake256(randomness))
```

Vinegar vectors are sampled in nested loop in sequential order – we can fault to abort the loop, skipping the sampling of a vector!

Loop-abort attacks against Mayo (Scenario 1)

Scenario 1: Assume vinegar vectors are initialized to constant, known, value before being sampled.

Assume induced fault skips sampling of a vinegar vector: Resulting signature immediately reveals corresponding oil vector.

```
for i=1 to k: //Fault here
  v[i] = decode(shake256(randomness))
```

Assume we fault the sampling of vector v_k and all vectors are initialized to zero before: Then resulting signature s after faulting the sampling of v_k is :

$$s = (s_1, \ldots, s_k) = (v_1 + o_1 \ldots, v_k + o_k) = (v_1 + o_1, \ldots, \mathbf{0} + \boldsymbol{o_k})$$

Immediately revealing vector o_k!

Loop-abort attacks against Mayo (Scenario 2)

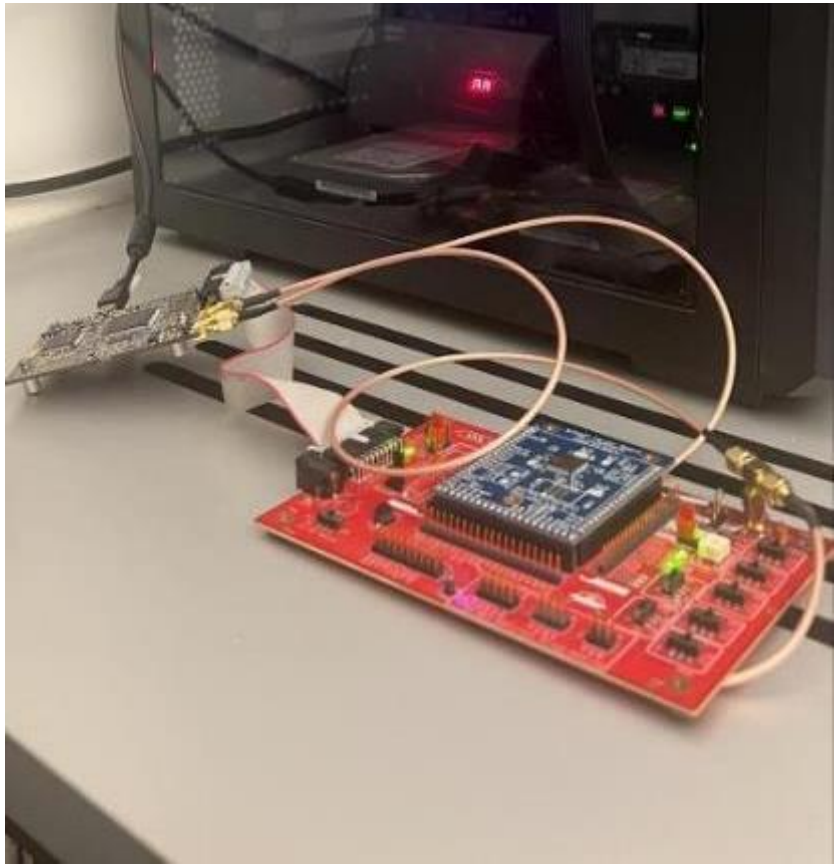
Scenario 2: Differential Fault Attack: Attack first lets the device compute a signature s correctly, then inject a loop-abort fault during the computation of a second signature s'.

If this fault leads to repetition of vinegar values, then can compute an oil vector from the two signatures

- Assume loop-abort fault results in at least one vinegar vectors v_i' being assigned the same value as in the first (correct) signature
- Then: can recover oil vector v_i given a pair of faulted and non-faulted signature (s_i', s_i) through $s_i - s_i' = v_i + o_i - v_i' - o_i' = o_i - o_i'$ which is again a vector the oil space!
- Does not require the other (non-faulted) vinegar values to be the same in both signatures – works against even the randomized version of MAYO!
- Example real-world scenario: Vinegar values are allocated in memory, but not initialized. Skipping the vinegar sampling could result in re-using the values that were in memory before.

Theoretical and Practical Evaluations succeeded

- **Theoretical Evaluation:** Simulated faults in the SAGE implementation of Mayo as submitted to the NIST competition: Key recovery in less than a minute for both scenarios!
- **Practical Evaluation:** Implemented a clock-glitching attack using the ChipWhisperer, executing Mayo's original C implementation on an STM32F4 target board.
    - Successfully skipped the vinegar sampling loop using a clock-glitch, resulting in an oil-vector revealing signature!



Conclusion

1	By faulting the sampling of the vinegar values, attackers can reveal an oil vector with only one fault.
2	One oil vector is sufficient to recover the entire oil space in minutes
3	This attack results in valid signatures and can attack the non-deterministic version of MAYO, and therefore need additional countermeasures to defend against