Recovering Rainbow's Secret Key with a First-Order Fault Attack

Thomas Aulbach[1], Tobias Kovats[2], Juliane Krämer[1], and Soundes Marzougui[3]

[1] Universität Regensburg, Regensburg, Germany
{thomas.aulbach, juliane.kraemer}@ur.de
[2] SBA Research, Vienna, Austria
tkovats@sba-research.org
[3] Technische Universität Berlin, Berlin, Germany
soundes.marzougui@tu-berlin.de

Abstract. Rainbow, a multivariate digital signature scheme and third round finalist in NIST's PQC standardization process, is a layered version of the unbalanced oil and vinegar (UOV) scheme. We introduce two fault attacks, each focusing on one of the secret linear transformations T and S used to hide the structure of the central map in Rainbow. The first fault attack reveals a part of T and we prove that this is enough to achieve a full key recovery with negligible computational effort for all parameter sets of Rainbow. The second one unveils S, which can be extended to a full key recovery by the Kipnis-Shamir attack. Our work exposes the secret transformations used in multivariate signature schemes as an important attack vector for physical attacks, which need further protection. Our attacks target the optimized Cortex-M4 implementation and require only first-order instruction skips and a moderate amount of faulted signatures.

Keywords: Rainbow · Fault injection attacks · Multivariate schemes · Post-quantum cryptography · Cortex M4 implementation

1 Introduction

Quantum computers pose a threat to public-key schemes based on the integer factorization or the discrete logarithm problem, like the widely deployed RSA and ECC. Since Shor published his famous algorithms [28] to solve these problems in polynomial time, their security depends on the technical feasibility of large scale quantum computers. Although there still is a certain amount of scepticism about the possibility of quantum computers being capable of factorizing integers and solving the discrete logarithm for cryptographically relevant instances in the upcoming decades [17, 19, 26], the National Institute of Standards and Technology (NIST) states that:

"[...] regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing" [1].

Therefore, a standardization process for post-quantum, i.e., quantum-resistant cryptographic algorithms was initiated. Currently five families of post-quantum cryptography are being studied, each relying on different mathematical assumptions. Concerning signatures, the remaining candidates in the currently ongoing third round of the standardization process mainly consist of lattice-based, hash-based, and multivariate schemes.

The evaluation criteria of NIST are not only the security and performance of the candidates, but also other properties such as resistance to side channel attacks and misuse resistance. Hence, NIST asked for efficient implementations that are protected against physical attacks, such as side channel and fault attacks. In these attacks, an attacker does not exploit mathematical weaknesses of a cryptographic scheme. Instead, in a side channel attack an attacker measures physical information during the computation of a cryptographic algorithm that she then analyzes to reveal secret data. In a fault attack, an attacker intentionally introduces faults into the computation such that it results in faulty outputs that she can analyze to learn secret data. In a first-order fault attack, an attacker induces a single fault during a computation, while in higher-order fault attacks, at least two faults are induced in the same computation. Since this is technically more complex, first-order faults are generally believed to be more realistic and, hence, more practically relevant.

In this work, we study first-order fault attacks on Rainbow, a multivariate signature scheme that was selected as finalist in the NIST standardization [14, 15]. We are aware of the significant improvements in mathematical cryptanalysis on the multivariate signature schemes GeMSS and Rainbow that have been published recently [3, 4, 30]. Especially the improved cryptanalytic approach presented by Beullens massively reduces the complexity of key recovery attacks against Rainbow, in particular against the parameter set for security level I [4]. However, NIST announced that there will be a fourth round where further post-quantum signature schemes can be submitted [4]. This became necessary since on the one hand, for security and practicality reasons diversity of (post-quantum) signatures is needed, while on the other hand the remaining signature candidates in the current third round are mostly based on structured lattices. Thus, for the fourth round NIST is especially interested in schemes that are not based on structured lattices and we expect that, despite the recent cryptanalytic results, other multivariate schemes will be submitted. For instance, the well-studied scheme UOV is likely to gain more attention soon, since it is explicitly not affected by Beullens latest approach. Hence, we believe that our results are relevant for the future development of multivariate signature schemes even if Rainbow turns out to be insecure or even broken. Our results reveal attack vectors for and weaknesses of a specific multivariate signature scheme, which have to be prevented in future developments or optimizations of other multivariate signature schemes, too.

[4] Announced by Dustin Moody at the third PQC Standardization Conference in June 2021 `https://www.nist.gov/video/third-pqc-standardization-conference-session-i-welcomecandidate-updates`

Related Work. The number of publications on the physical security of multivariate cryptography has increased in recent years, but is still manageable. Some effort was put into side channel analysis (SCA) of signature schemes. To name a few of them, Steinwandt et al. theoretically conducted differential power analysis (DPA) to reveal the secret seed and subsequently the affine bijections S and T used in FLASH and SFLASH already in 2001 [29]. Some years later, Okeya et al. were the first to experimentally verify a DPA attack against SFLASH [23]. More recently, in 2017, Yi and Li presented a DPA on enTTS [32], a signature scheme that contains some common features with Rainbow, such as the layer structure of the central map and the enclosing affine transformations. Finally, there are side channel attacks by Park et al. [24] and by Pokorny et al. [25] on Rainbow, both targeting the affine transformations via correlation power analysis (CPA).

The literature on fault attacks on multivariate signatures is less extensive. In 2011, Hashimoto et al. described some general ideas that might be applicable to multivariate schemes [18]. However, their ideas remain rather high-level and refer to several schemes at once. The authors in [32] mentioned a fault model that is supposed to facilitate the DPA on the central map F of the enTTS scheme, but they also did not provide a detailed description of their approach. They merely stated to "cause a fault, to change the unknown items during the signature generation". Krämer and Loiero transferred two ideas of [18] to UOV and Rainbow [21]. First, they analyzed how a faulted coefficient in the central map propagates through the signature and can be utilized to regain information about the secret transformation S. Second, they discussed the effect of fixing the vinegar variables across multiple signatures and show how information about the secret transformation T can be revealed by this. Shim and Koo developed the latter approach further to a full key recovery attack [27]. But the algebraic post-processing method they used still has a significant complexity of 2^{40}, rendering the attack impractical. The most practical attack yet, called *Quantumhammer* [22] by Mus, Islam, and Sunar, was performed on LUOV. The authors randomly induce bit flips in the linear transformation T and learn one bit of the secret key through each faulty signature. They also append an algebraic attack to the online fault injection phase, but they manage to limit the effort in the range of hours. In summary, there is no fault attack on Rainbow that presents a full key recovery in reasonable time.

Contribution. We introduce the first two efficiently executable fault attacks on Rainbow that lead to full key recovery. Both fault attacks only require first-order instruction skips and a moderate number of faulted signatures to be executed. We target the optimized Cortex-M4 implementation from [12].

1. We revisit the already existing theoretical approach of fixing the vinegar variables via a fault injection attack. This attack leads to partial leakage of one of the secret transformations used in Rainbow. The authors of [27] suggested to exploit this leakage by speeding up the key recovery attack using the notion of good keys [31]. Although they can reduce the number of variables and equations, the remaining system of quadratic equations is still

of significant complexity of around 2^{40} [27, Table. VII] for the Rainbow level I parameter set. We introduce a cryptanalytical method for circumventing this costly procedure and show how it is possible to recover the remaining bits of the secret key by just solving linear equations. Contrary to the previously suggested key recovery attack, this method can be applied to any possible parameter set of Rainbow and leads to full key recovery.

2. We present a new fault attack that targets the application of the linear transformation S. By collecting a small number of faulted signatures, we obtain enough input and output values of S to completely recover it. By knowledge of S, the complexity of Rainbow can be reduced to a small UOV instance with reduced parameter sizes [3]. To complete the key recovery, we apply the Kipnis-Shamir attack [5] for unbalanced oil and vinegar values [20], with remaining complexity $\mathcal{O}(q^{v_1-o_1})$. Considering Rainbows security level I parameter set, it holds $q^{v_1-o_1} = 2^{16}$. Compared to the first fault attack, this attack works with half the number of faulted signatures.

We see the algebraic post-processing that is used to further exploit the information gained by the fault attacks as a contribution of its own. It can be used as a plug-and-play method for all kinds of physical attacks. For instance, Section 3.1 proves that if an attacker obtains the block T_1 of the secret Rainbow parameters through any kind of leakage, she can achieve full key recovery without any further physical or computational effort.

Furthermore, we verify our attacks on an emulated ARM M4 architecture. On the one hand, this implies that we execute the compiled binary of the source code as a real signing device would and, therefore, can target the specific instruction of the assembly code that needs to be skipped. On the other hand, it verifies the feasibility of our attacks and proves the claims we made above for a given Rainbow key pair.

Finally, we suggest efficient countermeasures to prevent the mentioned attacks and make implementations of multivariate schemes more resilient against fault attacks.

1.1 Organization

In Section 2, we develop the background that is necessary for the presented attacks. This includes the Rainbow signature scheme, relevant simplifications applied by the authors of the corresponding NIST submission, and background information on fault attacks. In Section 3, we present the two fault attacks, together with a detailed description of the algebraic post-processing. We uncover the low-level instructions that need to be skipped in the practical fault attack in Section 4 and present our simulation. In Section 5, we suggest countermeasures to the previously described fault attacks and Section 6 concludes the work.

[5] In a very recent paper Esser et al. claim that there is another way to complete the key recovery, instead of using the Kipnis-Shamir attack [16]. If their findings hold true, this works with significantly lower complexity $\mathcal{O}(n^3)$, which is efficient even for higher parameter sets

2 Background

In this section, we recall useful background information for understanding the rest of the paper. This includes an overview on the Rainbow signature scheme and fault injection attacks.

Notation Let $\mathbb{F}_q$ be a finite field with q elements. By $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ we denote a vector and by $T \in M_n(\mathbb{F}_q)$ we denote a matrix with entries in $\mathbb{F}_q$. The multivariate quadratic maps $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ are given by m quadratic polynomials $p^{(i)}$ in n variables. To concatenate two strings x and y, $x||y$ is written. $\mathcal{H}(x)$ represents the application of a hash function $\mathcal{H}$ on a value x.

2.1 The Rainbow Signature Scheme

The Rainbow signature scheme [15] can be seen as a generalization of the unbalanced oil and vinegar (UOV) scheme [20]. It consists of several layers, where the oil and vinegar variables of the i-th layer are used as vinegar variables of the subsequent layer. Inserting them into the central polynomials of this layer, leads to easily solvable linear equations, since there are no quadratic oil terms in each layer just as it is the case for UOV. Initially, the authors suggested to use $u = 5$ layers, but it turned out to work best for $u = 2$ which is used by all currently suggested parameter sets. The case $u = 1$ constitutes the original UOV, the scheme whose security and efficiency Rainbow is supposed to improve.

More formally, let $\mathbb{F}_q$ be a finite field with q elements and $v_1 < \ldots < v_{u+1} = n$ be integers. Set $V_i = \{1, \ldots, v_i\}$ and $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ for $i \in \{1, \ldots, u\}$. Therefore, it holds $|V_i| = v_i$ and $|O_i| = o_i$ for $i \in \{1, \ldots, u\}$. The central map F of Rainbow consists of $m = n - v_1$ multivariate quadratic polynomials $f^{(v_1+1)}, \ldots, f^{(n)}$ of the form

$$f^{(k)}(x) = \sum_{i,j \in V_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in V_l, j \in O_l} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} x_i + \delta^{(k)}, \quad (1)$$

where $l \in \{1, \ldots, u\}$ is the only integer such that $k \in O_l$. In each layer l there remain no quadratic terms in the polynomials $f^{(k)}$ after inserting the values of the vinegar variables x_i for $i \in V_l$. This leads to an easily invertible central map $F : \mathbb{F}^n \to \mathbb{F}^m$ consisting of the m coordinate functions f. This special structure of $\mathcal{F}$ facilitates the signature generation and must be hidden in the public key. To this end, two invertible linear maps $S : \mathbb{F}^m \to \mathbb{F}^m$ and $T : \mathbb{F}^n \to \mathbb{F}^n$ are concatenated to the central map in order to generate the *public key*

$$\mathcal{P} = S \circ \mathcal{F} \circ T : \mathbb{F}^n \to \mathbb{F}^m. \quad (2)$$

Since the composed maps look like a random system of multivariate quadratic equations, it is hard to find a preimage under $\mathcal{P}$. By holding the *private key* maps $S, \mathcal{F}$, and T, this task becomes feasible. The signing and verifying procedure for the Rainbow signature scheme can be briefly summarized as follows. For the signing procedure we also present the pseudo code in Algorithm 1.

Signature Generation: To generate a signature for a message (or hash value) $w \in \mathbb{F}_m$, one performs the following three steps.

1. Compute $x = S^{-1}(w) \in \mathbb{F}^m$.
2. Compute a pre-image y of x under the central map $\mathcal{F}$.
3. Compute the signature $z \in \mathbb{F}^n$ by $z = T^{-1}(y)$.

Signature Verification: To check, if $z \in \mathbb{F}^n$ is a valid signature for a message $w \in \mathbb{F}^m$, one simply computes $w_0 = \mathcal{P}(z)$. If $w_0 = w$ holds, the signature is accepted, otherwise rejected.

Since both fault attacks presented in this work target the signing procedure, we shortly present it in Algorithm 1 to give the reader a first intuition of the code lines that render the signature scheme vulnerable.

The algorithm is located in Section 3.1 to keep it close to the presented fault attacks. We use a similar description as [14, Section 3.5], but align the notation tighter to the actual implementation and simplify the representation of the secret maps, as they are all chosen to be homogeneous in [14, Section 4], anyway. The first fault attack in Section 3.1 targets the sampling of random vinegar variables in Line 2. The second fault attack in Section 3.2 bypasses the application of S in Line 8. Since both lines are located in a loop, one has to consider under which conditions the fault might be annihilated by a repeated execution of the respective lines. We exit the while loop if the matrices given by $\hat{F}_1$ and $\hat{F}_2$ are invertible. If we assume the entries to be uniformly distributed in $\mathbb{F}_q$, the probability that a matrix $M_n(\mathbb{F}_q)$ is invertible is given by

$$\prod_{i=0}^{n}(\frac{q^n - q^i}{q^{n^2}}) = \prod_{i=1}^{n}(1 - \frac{1}{q^i}). \tag{3}$$

For the given parameters this evaluates to approximately 93%. We will carefully analyze what impact the injected fault might have on the conditions of the while loop in Section 4.2.

Remark 1. Very recently, Beullens presented an improved cryptanalytic approach that massively reduces the complexity of key recovery attacks [4], in particular against the Rainbow parameter set for security level I. His paper builds on an earlier analysis of him [3], where he introduced a new description of the Rainbow scheme, avoiding the presence of the central map and rather considering secret subspaces that satisfy certain equations under the public map $\mathcal{P}$ and its polar form $\mathcal{P}'$. He defines

$$\begin{aligned} O_1' \subset \mathbb{F}_q^n &:= \{x \in \mathbb{F}_q^n : x_i = 0 \text{ for } i \in \{1, \ldots, v_1\}\}, \\ O_2' \subset \mathbb{F}_q^n &:= \{x \in \mathbb{F}_q^n : x_i = 0 \text{ for } i \in \{1, \ldots, v_1 + o_1\}\}, \\ W' \subset \mathbb{F}_q^m &:= \{x \in \mathbb{F}_q^m : x_i = 0 \text{ for } i \in \{1, \ldots, o_1\}\}. \end{aligned}$$

The interesting point about these (public) subspaces is that all polynomials of the central map vanish on O_2' and the polynomials of the first layer vanish even on O'_1, i.e., it holds $\mathcal{F}(O'_2) = 0$ and $\mathcal{F}(O'_1) \subset W'$, respectively. The secret linear maps S and T now transform the given subspaces to the secret subspaces $O_1 = T^{-1}O'_1, O_2 = T^{-1}O'_2$, and $W = SW'$. The new technique in [4] finds a vector in the secret subspace O_2 with way less computational effort than needed in previous works. The attack is completed in similar style as in [3], where he uses the vector in O_2 to recover W efficiently by using the polar form of the public key and finally, applying the Kipnis-Shamir attack to compute O_1. The important take-away for our analysis in Section 3.2 is that recovering the secret transformation S is equivalent to detecting the secret subspace W using this notation. For more details we refer to [3, Section 5].

2.2 Conventions in the Specification

In the Rainbow specification [14], several simplifications are made. They are introduced to speed up the key generation process and reduce the key sizes, while it is argued that they do not weaken the security of Rainbow. First, the secret transformations S and T are chosen to be of the form

$$S = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \qquad \text{and} \qquad T = \begin{pmatrix} I & T_1 & T_2 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix}. \tag{4}$$

This is justified by the fact that, for every public map $\mathcal{P}$, there exists an equivalent secret key $(S, \mathcal{F}, T)$ with S and T as in Equation (4). Consequently, the inverse maps have the same structure and are given by

$$S^{-1} = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \qquad \text{and} \qquad T^{-1} = \begin{pmatrix} I & T_1 & T_4 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix}, \tag{5}$$

where $T_4 = T_1T_3 - T_2$. Furthermore, the Rainbow submitters restrict themselves to a homogeneous central map $\mathcal{F}$. As a result, the public map $\mathcal{P} = S \circ \mathcal{F} \circ T$ is homogeneous as well. Thus, the coefficients of every quadratic polynomial $f^{(i)}$ and $p^{(j)}$ for $i, j \in \{1, \dots, m\}$ can be collected in $n \times n$ matrices, by defining $F_i \in M_{n\times n}(\mathbb{F}_q)$ and $P_j \in M_{n\times n}(\mathbb{F}_{||})$ as the matrices that satisfy $f^{(i)}(x) = x^\top F_i x$ and $p^{(j)}(x) = x^\top P_j x$, respectively. Following this notation, Equation (2) can be turned into an equation of matrices of the form

$$F_i = \sum_{j=1}^{m} \tilde{s}_{ij}(\tilde{T}^\top P_j \tilde{T}). \tag{6}$$

Here, $\tilde{s}_{ij}$ denote the entries of S^{-1} and $\tilde{T} = T^{-1}$. This method of switching back and forth from public to secret matrices will play a major role in the analysis of our fault attacks. Interchanging the roles of F_i and P_j in Equation (6) represents the basic procedure of computing the public key from the private key. In the above form, the equation occurs less frequently in the literature, but is used, e.g., by Thomae in [31]. Due to the structure of the central polynomials in Equation (1), several parts of the matrices F_i are forced to be zero and this is obvious to any attacker. In more detail, the zero blocks of the matrices are given as

$$F_i = \begin{pmatrix} F_i^{(1)} & F_i^{(2)} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ for } i \in \{1, \dots, o_1\} \tag{7}$$

and

$$F_i = \begin{pmatrix} F_i^{(1)} & F_i^{(2)} & F_i^{(3)} \\ 0 & F_i^{(5)} & F_i^{(6)} \\ 0 & 0 & 0 \end{pmatrix} \text{ for } i \in \{o_1 + 1, \dots, o_2\}.$$

If an attacker somehow obtains secret information, she can use Equation (6) together with the structure of the occurring matrices given by Equation (5) and Equation (7), to further exploit this leakage. In Section 3.1 we will show, e.g., how an attacker that has T_1 at hand, is able to recover S_1 and subsequently T_3 and T_4, just by solving linear equations. Although this assumes a strong leakage in the first place, to the best of our knowledge this is the first work that demonstrates such a result.

2.3 Fault Attacks

Fault attacks against cryptographic schemes were first described 25 years ago [8]. Since then, a great variety of fault attacks has been developed. All fault attacks have in common that an attacker actively and intentionally disturbs the computation of a cryptographic algorithm so as to gain secret information from the faulty output. Both the kind of physical fault and the effect of a fault can be manifold. For instance, a fault can be injected via clock glitching, e.g., [9], or laser fault injection, e.g., [11], and it can be either transient or permanent. Most published fault attacks result in a zeroed or randomized value or an instruction skip, see, e.g., [6]. Instruction skips as also used in this work correspond to skipping, i.e., ignoring, selected lines of the program code. Since fault attacks were first described, most cryptographic schemes have been analyzed with respect to them and in recent years, fault attacks have been published for all five families of post-quantum cryptography, e.g., [6,9–11,21]. These works range from purely theoretical publications [6] to more practical attacks [22].

3 Full Key Recovery Attacks

In this section we present two different fault attack scenarios. Both rely on a first-order skipping fault model, i.e., the attacker is assumed to be capable of introducing a single instruction skip during the signing procedure. Compared to fault attacks that require randomizing or zeroing values in memory, the instruction skips belongs to the more practical ones and it has been shown that even higher-order skipping faults against real-world cryptographic implementations are possible [7]. The first attack already exists in the literature. It aims at fixing the vinegar variables across consecutive signature generations and leads to valid signatures. We significantly reduce the complexity of the post-processing that is necessary to achieve a full key recovery. Additionally, we introduce a completely new attack that benefits from the same efficient techniques and works with even less faulted signatures. They constitute the first fault attacks on the Rainbow scheme that lead to a complete revealing of the secret key which are executable on a desktop machine. In both cases, the messages that are chosen to generate the faulty signatures not need to fulfill special requirements, that are hard to control. The messages must be different from each other and should lead to linearly independent vectors at some point of the respective procedures, but the attacker can just discard a faulty signature if it does not meet this requirement and start over with a new message. We will emphasize this condition in the description of the two fault attacks.

In Section 3.1, we reinvestigate the case of fixed vinegar variables, which was presented in detail in [21]. The authors of [27] are the first to expand this to a complete key recovery attack. However, their approach is reliant on a costly post-processing step that involves solving a system of quadratic equations of moderate size. They investigate three different level I parameter sets of Rainbow, and even in the best case their complexities remain as high as 2^{38}. This is still considerably large to constitute another hurdle in the practicability of the fault attack. We introduce a method on how to bypass this step, leading to an easily executable full key recovery attack.

Furthermore, we present in Section 3.2 a new fault attack that gets along with significantly less faulted signatures. The faulted output reveals the secret transformation S and the attack can be completed to a full key recovery by a subsequent Kipnis-Shamir attack. Due to the knowledge of S, the parameters of the system to solve allow for an efficiently executable instance of the Kipnis-Shamir attack. After explaining the attacks, we also translate the procedure to the abstract secret subspace notation used by Beullens which we stated in Remark 1.

3.1 Attack 1: Full Key Recovery from Fixed Vinegar Variables

This fault attack aims to skip the random generation of vinegar variables during the signing process. Depending on the implementation, this results in either the vinegar variables from the previous signature being reused, or being equal to zero in case they are zeroed at the end of the signing process. Both cases have already been mentioned in [27] and are very similar. The main difference is that for the reuse model an additional, unfaulted reference signature is needed. In the following we focus on the optimized bitsliced implementation developed in [12]. Here, the vinegar variables are not zeroed at the end of the signing process and therefore skipping the generation of the new vinegar variables results in the same variables being used for successive signature computations.

Algorithm 1 Rainbow Sign

```
    Input message $d$, private key $(S, \mathcal{F}, T)$, length $l$ of the salt.
    Output signature $\sigma = (z, salt) \in \mathbb{F}_q^n \times \{0,1\}^l$ s.t. $\mathcal{P}(z) = \mathcal{H}(\mathcal{H}(d)||salt)$.
 1: repeat
 2:     $(z_1, \ldots, z_{v_1}) \leftarrow_R \mathbb{F}_q^{v_1}$
 3:     $\big(\hat{f}^{(v_1+1)}, \ldots, \hat{f}^{(n)}\big) \leftarrow \big(f^{(v_1+1)}(z_1, \ldots, z_{v_1}), \ldots, f^{(n)}(z_1, \ldots, z_{v_1})\big)$
 4: until IsInvertible($\hat{F}_1$) == True
 5: repeat
 6:     $salt \leftarrow_R \{0,1\}^l$
 7:     $y \leftarrow \mathcal{H}(\mathcal{H}(d)||salt)$
 8:     $y \leftarrow S^{-1}(y)$
 9:     $z_{v_1+1}, \ldots, z_{v_2} \leftarrow \hat{F}_1^{-1}(y_{v_1+1}, \ldots, y_{v_2})$
10:     $\big(\hat{f}^{(v_2+1)}, \ldots, \hat{f}^{(n)}\big) \leftarrow \big(f^{(v_2+1)}(z_{v_1+1}, \ldots, z_{v_2}), \ldots, f^{(n)}(z_{v_1+1}, \ldots, z_{v_2})\big)$
11: until IsInvertible($\hat{F}_2$) == True
12: $z_{v_2+1}, \ldots, z_n \leftarrow \hat{F}_2^{-1}(y_{v_2+1}, \ldots, y_n)$
13: $z \leftarrow T^{-1}(z)$
14: $\sigma = (z, salt)$
15: return $\sigma$
```

First, we show how the secret matrices $T_1 \in M_{o_1 \times o_2}(\mathbb{F}_q)$ and $T_2 \in M_{o_2 \times o_2}(\mathbb{F}_q)$ in Section 2.2 can be determined from the faulty signatures. Let z' be the error-free generated signature of an arbitrary message d'. According to the Rainbow specification, z' is defined by $z' = T^{-1} \circ \mathcal{F}^{-1} \circ S^{-1}(y)$, where $y = \mathcal{H}(\mathcal{H}(d')||salt) \in \mathbb{F}_q^m$. From an attacker's point of view, all intermediate values are unknown. What is known is that the first v_1 entries of $\mathcal{F}^{-1} \circ S^{-1}(y)$ consist of the generated vinegar values, whereas the remaining $m = o_1 + o_2$ entries are the corresponding solutions of the first and second layer of the central map under the chosen vinegar variables. Thus, we can write

$$z' = T^{-1} \begin{pmatrix} v \\ o_1' \\ o_2' \end{pmatrix},$$

with $v \in \mathbb{F}_q^{v_1}, o_1' \in \mathbb{F}_q^{o_1}$, and $o_2' \in \mathbb{F}_q^{o_2}$. By using the instruction skip indicated in Section 2.1 and elaborated in detail in Section 4.1, the attacker successively generates m signatures, all of which fall back to the same vinegar variables v as the reference signature z'. For $i \in \{1 \ldots m\}$, we denote these signatures by

$$z^{(i)} = T^{-1} \begin{pmatrix} v \\ o_1^{(i)} \\ o_2^{(i)} \end{pmatrix}.$$

The remaining entries $o_1^{(i)}$ and $o_2^{(i)}$ of the input of T^{-1} are under no control of the attacker and do not need to be considered in more detail. By subtracting the reference signature and multiplying with T, we receive

$$T(z^{(i)} - z') = \begin{pmatrix} v \\ o_1^{(i)} \\ o_2^{(i)} \end{pmatrix} - \begin{pmatrix} v \\ o_1' \\ o_2' \end{pmatrix} = \begin{pmatrix} 0 \\ \tilde{o}_1^{(i)} \\ \tilde{o}_2^{(i)} \end{pmatrix}, \tag{8}$$

for $i \in \{1 \dots m\}$. Let $Z \in M_{n\times m}$ be the matrix whose i-th column is defined by the vector $z^{(i)} - z'$. Then Equation (8) implies that the first v_1 rows of T map Z to $0_{v_1 \times m}$. If this linear system of equations can be solved uniquely, it reveals the first v_1 rows of T, more precisely the submatrices $T_1 \in M_{o_1\times o_2}(\mathbb{F}_q)$ and $T_2 \in M_{o_2\times o_2}(\mathbb{F}_q)$. Therefore, we need the columns of Z and thus the last m entries of $Tz^{(i)}$ to be linearly independent, since the first entries are identical to the entries of the reference signature. Following Equation 3 this happens with high probability and in case we draw a faulted signature that is linearly dependent of the previous, we can just disregard it and draw a new one. We note that Equation (8) does not provide any further information about the remaining rows of T.

Remark 2. The authors of [27] only utilize parts of the gained information for their algebraic key recovery attack. More precisely, they use certain entries of the submatrices to reduce the complexity of a key recovery attack introduced in [31] using the good key approach. However, this still requires solving a system of quadratic equations. In the following we show how this can be completely omitted by utilizing the whole submatrix T_1.

Recover the secret transformation S. We take a closer look at Equation (6). By also dividing the public matrices P_j and the secret transformation T into 3×3 block matrices we receive

$$\tilde{F}_i = \sum_{j=1}^{m} \tilde{s}_{ij} \left[\begin{pmatrix} I & 0 & 0 \\ T_1^\top & I & 0 \\ T_4^\top & T_3^\top & I \end{pmatrix} \begin{pmatrix} P_j^{(1)} & P_j^{(2)} & P_j^{(3)} \\ 0 & P_j^{(5)} & P_j^{(6)} \\ 0 & 0 & P_j^{(9)} \end{pmatrix} \begin{pmatrix} I & T_1 & T_4 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix} \right]. \tag{9}$$

The resulting matrices are not necessarily equal to the matrices F_i of the central map but the polynomials they represent are identical. Consequently, by denoting

$$\tilde{F}_i = \begin{pmatrix} \tilde{F}_i^{(1)} & \tilde{F}_i^{(2)} & \tilde{F}_i^{(3)} \\ \tilde{F}_i^{(5)} & \tilde{F}_i^{(5)} & \tilde{F}_i^{(6)} \\ \tilde{F}_i^{(7)} & \tilde{F}_i^{(8)} & \tilde{F}_i^{(9)} \end{pmatrix}, \tag{10}$$

it follows from Equation (7) that $\tilde{F}_i^{(5)}$ needs to be skew symmetric and $\tilde{F}_i^{(7)\top} + \tilde{F}_i^{(3)} = 0_{v_1\times o_2}$ and $\tilde{F}_i^{(8)\top} + \tilde{F}_i^{(6)} = 0_{0_1\times o_2}$ holds for the central maps of the first layer, i.e., for $i \in \{1, \dots, o_1\}$.

Now, we solely focus on the middle block $\tilde{F}_i^{(5)} \in M_{o_1 \times o_1}(\mathbb{F}_q)$ and observe that T_1 is the only part of the secret transformation T contributing to that block. Thus, neglecting the other submatrices turns Equation (9) into

$$\tilde{F}_i^{(5)} = \sum_{j=1}^{m} \tilde{s}_{ij} \left(T_1^\top P_j^{(1)} T_1 + T_1^\top P_j^{(2)} + P_j^{(4)} T_1 + P_j^{(5)} \right). \tag{11}$$

Note that the term inside the brackets is completely known to the attacker, since she has already recovered T_1. The remaining unknowns are now the entries of S^{-1}, in particular the $o_1 \cdot o_2$ entries of S_1. Since Equation (11) holds for all $i \in \{1, \dots, o_1\}$, the resulting linear system of equations is overdetermined and solving it provides exactly the entries of S_1.

Recover the remaining part of the secret transformation T. Having access to the complete transformation S, the attacker is able to exploit (9) even more. She now targets the $v_1 \times o_2$ block $F_i^{(3)}$ on the top right and the $o_2 \times o_1$ block $F_i^{(7)}$ on the bottom left. Similarly to (11), she derives

$$F_i^{(7)\top} + F_i^{(3)} = 0_{v_1 \times o_2} = \sum_{j=1}^{m} \tilde{s}_{ij} \left(P_j^{(1)\top} T_4 + P_j^{(1)} T_4 + P_j^{(2)} T_3 + P_j^{(3)} \right). \tag{12}$$

Now the attacker wants to solve for the unknowns in T_3 and T_4. By now, she has knowledge of all the entries $\tilde{s}_{ij}$, which turns (12) into a linear system of equations. Once more the number of equations exceeds the number of variables and its solution reveals the submatrices T_3 and T_4 and therefore the remaining part of the secret transformation T^{-1}. This finishes the key recovery attack. The algebraic post-processing of the fault attack can be summarized as follows.

Attack 1: Full key recovery from fixed vinegar variables. After successful execution of the fault attack, the attacker takes the reference signature z' and m faulted signatures $z^{(1)}, \dots, z^{(m)}$, obtained in the way described above and proceeds as follows.

1. Build the matrix $Z \in M_{n \times m}(\mathbb{F}_q)$ with columns $z^{(i)} - z'$ for $i \in \{1, \dots, m\}$.
2. Compute the echelon form of the matrix $T' \in M_{v_1 \times n}(\mathbb{F}_q)$ that fulfills $T'Z = 0$. It holds $T' = \left(I \; T_1 \; T_2 \right)$.
3. Insert T_1 into Equation (11). Solve the resulting system of linear equations to recover S.
4. Insert S into Equation (12). Solve the resulting system of linear equations to recover T_3 and T_4.
5. Use Equation (6) to obtain $\mathcal{F}$. The attacker recovered the full secret key $(S, \mathcal{F}, T)$.

Remark 3. This attack can also be translated to the more abstract language established in Remark 1. The difference of two signatures $z^{(i)} - z'$ that are generated with identical vinegar variables, can be seen as a vector in the secret subspace O_1. This becomes obvious when considering Equation (8), which shows that T maps this vector to a vector whose first v_1 entries are zero, i.e, an element in O'_1. Thus, the m linearly independent vectors of the matrix Z that are gained by our fault attack, together span the secret subspace O_1 from which the remaining secret subspaces can be deduced.

3.2 Attack 2: Secret Key Recovery by Skipping the Linear Transformation S

This fault attack aims to skip the application of the matrix S^{-1} during the generation of the signature. If the instruction skip is successful, the signing process evaluates to $\tilde{z} = T^{-1} \circ \mathcal{F}^{-1}(y)$. By inserting this faulted signature into the public map $\mathcal{P}$, an attacker receives $\mathcal{P}(\tilde{z}) = S \circ \mathcal{F} \circ T(\tilde{z}) = S(y) =: w \in \mathbb{F}_q^m$.

Since $y = \mathcal{H}(\mathcal{H}(d)||salt)$ is known to the attacker, this fault attack presents a method for deriving input-output pairs for the secret linear transformation S. Now, let $W \in M_{m \times o_2}(\mathbb{F}_q)$ be the matrix whose columns consist of vectors $w^{(i)} \in \mathbb{F}_q^m, i \in \{1 \dots o_2\}$, which are obtained in the manner described above, and $Y \in M_{m \times o_2}(\mathbb{F}_q)$ be the matrix whose columns consist of the corresponding starting vectors $y^{(i)}, i \in \{1 \dots o_2\}$. By dividing the matrices and vectors into blocks according to Equation (4), we receive

$$SY = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} Y_1 + S_1 Y_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix}. \tag{13}$$

Thus, the secret submatrix S_1 can be obtained via $S_1 = (W_1 - Y_1) * Y_2^{-1}$. Consequently, for the attack to be successful o_2 faulty signatures are needed and the starting vectors $y^{(i)}$ need to be chosen s.t. $Y_2 \in M_{o_2 \times o_2}(\mathbb{F}_q)$ is invertible.

Recover T by using S. Having access to the secret transformation S, an attacker can use the very same strategy to recover T_4 and T_3 as described in Section 3.1. By the time this step was applied during the post-processing of the first fault attack, the attacker had already learned T_1, which is not the case anymore. However, in order to exploit Equation (12) it is enough to know S, i.e., the attacker does not need any of the entries of T_1 to recover T_3 and T_4.

This procedure - although presented somewhat differently - was already proposed by Park et al. in [24, Section 4.2]. In their work, they used Correlation Power Analysis to obtain S and thus, faced the same challenge for the subsequent algebraic evaluation, i.e., the recovery of T under the knowledge of S. In order to obtain T_1, they suggest to use a similar approach, namely by focusing on the $o_1 \times o_2$ block $F_i^{(6)}$ and the $o_2 \times o_1$ block $F_i^{(8)}$ of (10). However, it is not possible to continue the attack like this, as we sketch in the following. Therefore, observe

$$F_i^{(8)\top} + F_i^{(6)} = 0_{o_1 \times o_2} = \sum_{j=1}^{m} \tilde{s}_{ij} \Big(T_1^\top (P_j^{(1)\top} + P_j^{(1)}) T_4 + T_1^\top (P_j^{(2)}) T_3 + \dots \\ \dots P_j^{(2)\top} T_4 + (P_j^{(5)\top} + P_j^{(5)}) T_3 + T_1^\top P_j^{(3)} + P_j^{(6)} \Big). \tag{14}$$

While it is true that only linear equations remain, after inserting the known values for T_3, T_4, and S, one can deduce from

$$T_1^\top \sum_{j=1}^{m} \tilde{s}_{ij} \Big((P_j^{(1)\top} + P_j^{(1)}) T_4 + (P_j^{(2)}) T_3 + P_j^{(3)} \Big) \overset{(12)}{=} T_1^\top 0,$$

that Equation (14) does not provide further information about the block T_1, since it is satisfied independent of its choice. The authors of [31] and [24] confirmed our findings in this regard.

Thus, one way to proceed and obtain T_1, is to fall back to the well-known Kipnis-Shamir attack on UOV [20]. Note that the knowledge of S is equivalent to the recovery of the secret subspace W, referring to the notation in Remark 1. Following [3, Section 5.3], this reduces the problem of finding O_1 to a small UOV instance with reduced parameter $n' = n - o_2$ and $m' = m - o_2$ and complexity $\mathcal{O}(q^{n'-2m'}) = \mathcal{O}(q^{n+o_2-2m}) = \mathcal{O}(q^{v_1-o_1})$. In case of Rainbow parameter set I, this leads to a very efficient method to finish the key recovery attack, since it holds $n' \approx 2m'$. For higher parameter sets this approach still remains infeasible, as we have $n' \gg 2m'$, rendering the Kipnis-Shamir attack inefficient.

Very recently, Esser et al. published a work on partial key exposure attacks [16], in which they cover Rainbow, among other schemes. They also treat the task of exposing T_1 after the remaining part of the secret matrices are known. Their approach builds up on a work by Billet and Gilbert [5] and has complexity $\mathcal{O}(n^3)$, which would be very efficient, even for larger parameter sets of Rainbow.

Attack 2: Secret key recovery by skipping the linear transformation S. After successful execution of the fault attack, the attacker takes the generated faulted signatures $z^{(1)}, \dots, z^{(o_2)}$ and the used starting values $y^{(1)}, \dots, y^{(o_2)}$ being of the form described above and performs the following steps:

1. Compute $w^{(i)} = \mathcal{P}(z^{(i)})$ for $i \in \{1, \dots, o_2\}$.
2. Build the matrices $W \in M_{m \times o_2}(\mathbb{F}_q)$ and $Y \in M_{m \times o_2}(\mathbb{F}_q)$ as described for Equation (13).
3. Recover S by computing $S_1 = (W_1 - Y_1) * Y_2^{-1}$.
4. Insert S into Equation (12). Solve the resulting system of linear equations to recover T_3 and T_4.
5. Obtain T by applying the Kipnis-Shamir attack to the reduced UOV instance.
6. Use Equation (6) to obtain $\mathcal{F}$. The attacker recovered the full secret key $(S, \mathcal{F}, T)$.

4 Code Analysis and Simulation

To implement the attacks described in Section 3, an in-depth analysis of the instruction code needs to be performed. The following section discusses how to uncover the low-level instructions that need to be skipped to achieve the desired behaviour of the fault attacks as specified in Section 3 based on the source code of the ARM Cortex M4 optimized round 3 submission by the authors of [12][6]. Furthermore, we present an elaborated simulation of our results.

4.1 Attack 1: Fixing the Vinegar Variables

Listing 1.1 shows the relevant code snippet for our first attack. The implementation proposed by [12] does not set the vinegar variables to zero after signature generation. Therefore skipping the function call to *prng_gen* in line 55 will leave them with the same values due to the temporary variable being reallocated to the same address at each function call. This, of course, assumes that the respective memory region is not overwritten between two function calls, which holds if the device acts solely as a signing oracle.

By analyzing the disassembly of the compiled binary, we find the relevant instruction given in Listing 1.2. By skipping the branch performed in line 0xdfb2, the desired behaviour is achieved and the vinegars remain constant for subsequent signatures.

4.2 Attack 2: Skipping the Linear Transformation S

To prohibit the application of the linear transformation S^{-1} we aim at skipping the function call to *gf256v_add* in line 178 of the source code shown in Listing 1.1. However, for this function being *inlined* - meaning the compiler inserts the function body instead of a branch - a single instruction skip does not suffice. Therefore the beforehand executed call to *gf16mat_prod_16_32* in line 173 is skipped, leaving the variable *temp_o* at its initial all-zero value and rendering the subsequent call to *gf256v_add* without effect. To achieve this effect, we target line 0xe070 of the assembly code shown in Listing. 1.2 with a first order fault.

Exiting the while loop In this paragraph, we discuss the probability of exiting the respective while loop on the first iteration, assuming that the fault injection was successful. Regarding the attack in Section 4.1, if the skip of the vinegar variables in Line 2 of Algorithm 1 is introduced successfully, the same vinegar variables are used again for consecutive signatures. Thus, the chosen vinegar variables already led to an invertible matrix $\hat{F}_1$ in the previous signature. Since $\hat{F}_1$ only depends on the vinegar variables $(y_1, \ldots, y_{v_1})$ and the polynomials of the first layer, the condition in Line 8 is always fulfilled. Regarding the attack in Section 4.2, the condition in Line 11 also depends on the solution of the first

[6] The source code can be found at `https://github.com/rainbowm4/rainbowm4.git`

```
 26  int rainbow_sign(...)
 27  {
...      ...
 51      while (!l1_succ) // until solution found
 52      {
...          ...
...          // skipped by Attack 1
 55          prng_gen(&prng_sign, vinegar, _V1_BYTE);
...          ...
 80      }
...      ...
...      // temp_o is initialized with zeros
155      uint8_t temp_o[_MAX_O_BYTE + 32] = {0};
...      ...
157      while (!succ) // until solution found
158      {
...          // skipped by Attack 2
173          gf16mat_prod_16_32(temp_o, sk->s1, _z + _O1_BYTE);
...          ...
...          // applying S
178          gf256v_add(y, temp_o, _O1_BYTE);
...          ...
228      }
...  ...
292  }
```

Listing 1.1. Relevant snippets of the rainbow_sign function in *rainbowm4/crypto_sign/rainbowI-classic/m4/rainbow.c*

```
...
0xdfb0 add      signature,sp,#0xe4
0xdfb2 bl       prng_gen
0xdfb6 add      _digest,sp,#0x6c
...
0xe06e add      signature,sp,#0x144
0xe070 bl       gf16mat_prod_16_32
0xe074 ldr      sk,[sp,#y[4]]
...
```

Listing 1.2. Relevant snippets of the assembly code corresponding to line 55 and line 173 of the rainbow_sign function in Listing 1.1.

layer in Line 9. Thus the probability of $\hat{F}_2$ to be invertible can be approximated by the probability of that a randomly generated matrix with entries in $\mathbb{F}_q$ is invertible which is given by Equation (3).

4.3 Simulation

To verify our assumptions and provide a first proof of concept, we implement a generic ARM M4 architecture simulation environment based on Unicorn [13], which itself is based on QEMU [2].[7] The validity of our results exceed the ones one would obtain by simple code modification - i.e., removing code lines one wishes to skip - as the compiled binary of the unmodified source code is executed within our simulation just as a real device would execute it. The 32-bit Reduced Instruction Set Computer (RISC) architecture as defined by ARM is emulated in its entirety.
The framework allows per-instruction execution of the compiled binary, cycle-accurate skipping faults and memory analysis at any given point during execution. This facilitates the validation of both attacks' feasibility through injection of the intended faults and subsequent analysis of the memory space mapped to the vinegar variables and y for the first and second attack, respectively. After verification of the skipping faults' effects on memory, signature collection is performed. Both attacks lead to successful recovery of the secret key, proving the feasibility of our attacks. In the following we give a brief overview of the core features of the simulation framework.

Key Generation For the generation of the public and secret key being computationally expensive and very time consuming within the simulation, we implement it on the host machine and subsequently map the keys to the simulated device memory.

Signing The signing algorithm runs entirely within the simulation. Upon executing the binary starting from the respective function's memory address, the secret key is mapped to the simulated device's memory. The address of the memory region holding the message to be signed, a buffer for the result and the key's address are written to the corresponding registers according to the calling convention. To implement the attacks, the simulation first stops at the address where we want to inject the fault. Then the instruction pointer is incremented as required by the length of the instruction to be skipped. Execution is subsequently resumed at the following instruction.

Verification For completeness, verification inside the simulation is also implemented. Of course, the adversary may implement verification on any device. It is merely used to verify successful fault injection and extract temporary variables that facilitate executing Attack 2.

[7] The codebase of our framework can be found at `https://anonymous.4open.science/r/double_rainbow_submission-E3CC`

4.4 Applicability to Other Implementations

The attack we introduced in Section3.1 is not directly applicable to the reference implementation of Rainbow that was submitted to the NIST Standardization Process [14]. This is due to the fact that the vinegar variables are zeroed at the end of the signing process there, so they can't be reused in a subsequent signing process by a first-order fault attack. See Section 5.1 for more details. The second attack, however, can be applied to the reference implementation, since the same steps as mentioned in Listing 1.1 are executed there.

5 Countermeasures

Countermeasures attempt to either verify the integrity of the executed algorithm before returning its result or ensuring that a system cannot leak secrets even when compromised. If the latter is the case, the returned value should either be a random number or an error constant. A traditional way to tackle this problem for the case of fault injections is to repeat the computation and compare the results. However, this approach is very expensive in terms of computation time and relies on the assumption that an attacker will fail to successfully inject faults in two subsequent runs of the algorithm. In this section we suggest countermeasures that can be adopted in order to avoid the attacks described in Section 3.

5.1 Countermeasures for Attack 1

For the first attack relying on keeping the vinegar variables constant, some countermeasures aiming for either zeroing or randomization can be employed.

Firstly, resetting the memory region mapped to the vinegars at the end of the function call to zero - as it is done in the original NIST submission in [15] - is the most straight-forward solution. Then, if the respective fault is injected, the system of equations is rendered non-solvable, leading to re-iteration of the loop until either a threshold number of iterations is reached and the function is exited or the fault injection fails and vinegar variables are sampled correctly. However, depending on the implementation, this can enable a different attack of higher complexity relying on partial zeroing of the vinegars, as described in [27].

Secondly, if the vinegars were to be saved in between subsequent function calls, they could be checked for equivalence before returning the signature. While this might seem a viable solution, care must be taken to ensure safe storage not to leak their values. Moreover, since simple checks are assumed to be easily skippable, the checking procedure has to be elegantly integrated in the signing procedure.

Thirdly, inlining the function call to *prng_gen* could prohibit the attack for some parameter sets. Depending on the implementation and the corresponding number of vinegars, the loop copying the random values to the vinegars can be exited earlier by injecting a fault, leaving them partially constant. While this prohibits our attack which requires all vinegars to stay constant, similar attacks with less stringent constraints might still be applicable. To further mitigate these, loop unrolling could provide remedy. However, this combined mitigation technique would introduce non-negligible overhead in code size which might render it inapplicable to constrained devices.

5.2 Countermeasures for Attack 2

For the second attack that aims to skip the application of the secret transformation S, an evident mitigation technique is to verify the signature before returning it. However, this leads to an overhead of around 25% [12], rendering this strategy very costly.

More practical, one could initialize the *temp_o* variable so that the skipping fault would result to an all-zero y after execution of *gf256v_add*. To achieve this, *temp_o* first $o_1/2$ bytes - i.e., the bytes that are affected by the subsequent addition - are initialized with y's first $o_1/2$ bytes (i.e., 16 for the parameter set of 32 oil variables in the bitsliced representation). The subsequent $\mathbb{F}_{256}$ addition - i.e., implemented as multiple consecutive binary XORs - then leads to an all-zero y, prohibiting leakage of the secret key through the collected signatures.

Furthermore, inlining the call to *gf16mat_prod_16_32* would prohibit a first-order skipping fault attack. However, due to this function being implemented in assembly language for optimization purposes, there is a discrepancy for the required build steps. Therefore this mitigation technique might not be trivial to implement.

6 Conclusion

This paper demonstrated how important it is to protect the secret transformations S and T in multivariate schemes against fault attacks. They are the only obstacles an attacker faces when trying to discover the structure of the central map $\mathcal{F}$. Due to their linearity, it is possible to recover them either partially (see Section 3.1) or in total (see Section 3.2), by collecting enough input and output vectors and analyzing their transformation. As the generated signature constitutes the output of T^{-1} and the hash value that is to be signed represents the input to S^{-1}, an attacker only needs to obtain an intermediate result, e.g., the input vector of T^{-1} or the output vector of S^{-1}, in order to gain secret information. If she is able to thoughtfully induce a fault that compromises one of these intermediate vectors, either by skipping a code line or forcing the algorithm to compute with the same values over and over again, the security of the scheme is no longer guaranteed.

For instance, it was already shown in [27] that UOV and LUOV are vulnerable to the attack that fixes the vinegar variables. Whereas the authors of [22] doubt that it is possible to fix a large portion of the vinegar variables by physical fault injection, we showed that this is indeed possible by a single instruction skip.

Specifically for Rainbow, we proved that it is not even necessary to recover the whole secret transformation T, by the means of a fault attack. The introduced algebraic attack restores the complete secret key of Rainbow on input of the submatrix T_1 by just solving linear equations. This is of course not limited to the evaluation of fault attacks, but also holds if T_1 is leaked through any other kind of side-channel analysis. In the light of the recent breakthrough in cryptanalysis [4], we acknowledge that the Rainbow parameter set for security level I is deprecated. However, the fault attacks we suggest, directly reveal either the secret subspaces O_1 (see Section 3.1) or W (see Section 3.2) and thus, work for any given parameter set, in particular for higher security levels and adapted parameters that are designed to meet new requirements.

Acknowledgement

This research work has been funded by the German Ministry of Education, Research and Technology in the context of the project Aquorypt (grant number 16KIS1022) and the project Full Lifecycle Post-Quantum PKI - FLOQI (grant number 16KIS1074). Furthermore, we want to thank Enrico Thomae and Namhun Koo for the helpful correspondence concerning the algebraic evaluation in Section 3.2.

References

1. Post-Quantum Cryptography. NIST Official Website, 2021. `https://csrc.nist.gov/projects/post-quantum-cryptography`.
2. Fabrice Bellard. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, page 41, USA, 2005. USENIX Association.
3. Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 348–373. Springer, 2021.
4. Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. Cryptology ePrint Archive, Report 2022/214, 2022. `https://ia.cr/2022/214`.
5. Olivier Billet and Henri Gilbert. Cryptanalysis of Rainbow. In *International Conference on Security and Cryptography for Networks*, pages 336–347. Springer, 2006.
6. Nina Bindel, Johannes Buchmann, and Juliane Krämer. Lattice-Based Signature Schemes and Their Sensitivity to Fault Attacks. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016*, pages 63–77. IEEE Computer Society, 2016.
7. Johannes Blömer, Ricardo Gomes Da Silva, Peter Günther, Juliane Krämer, and Jean-Pierre Seifert. A Practical Second-Order Fault Attack against a Real-World Pairing Implementation. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 123–136. IEEE, 2014.
8. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

9. Fabio Campos, Juliane Krämer, and Marcel Müller. Safe-Error Attacks on SIKE and CSIDH. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering - 11th International Conference, SPACE 2021, Kolkata, India, December 10-13, 2021, Proceedings*, volume 13162 of *Lecture Notes in Computer Science*, pages 104–125. Springer, 2021.
10. Laurent Castelnovi, Ange Martinelli, and Thomas Prest. Grafting Trees: A Fault Attack Against the SPHINCS Framework. In *PQCrypto*, volume 10786 of *Lecture Notes in Computer Science*, pages 165–184. Springer, 2018.
11. Pierre-Louis Cayrel, Brice Colombier, Vlad-Florin Drăgoi, Alexandre Menu, and Lilian Bossuet. Message-Recovery Laser Fault Injection Attack on the Classic McEliece Cryptosystem. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 438–467. Springer International Publishing, 2021.
12. Tung Chou, Matthias J. Kannwischer, and Bo-Yin Yang. Rainbow on Cortex-M4. Cryptology ePrint Archive, Report 2021/532, 2021. `https://ia.cr/2021/532`.
13. Hoang-Vu Dang and Anh-Quynh Nguyen. Unicorn: Next Generation CPU Emulator Framework. 01 2015.
14. Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.
15. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *International conference on applied cryptography and network security*, pages 164–175. Springer, 2005.
16. Andre Esser, Alexander May, Javier Verbel, and Weiqiang Wen. Partial Key Exposure Attacks on BIKE, Rainbow and NTRU. *Cryptology ePrint Archive*, 2022.
17. Roger A Grimes. *Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto*. John Wiley & Sons, 2019.
18. Yasufumi Hashimoto, Tsuyoshi Takagi, and Kouichi Sakurai. General Fault Attacks on Multivariate Public Key Cryptosystems. In *International Workshop on Post-Quantum Cryptography*, pages 1–18. Springer, 2011.
19. Gil Kalai. The Argument against Quantum Computers, the Quantum Laws of Nature, and Google's Supremacy Claims. *arXiv preprint arXiv:2008.05188*, 2020.
20. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 206–222. Springer, 1999.
21. Juliane Krämer and Mirjam Loiero. Fault Attacks on UOV and Rainbow. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 193–214. Springer, 2019.
22. Koksal Mus, Saad Islam, and Berk Sunar. QuantumHammer: a Practical Hybrid Attack on the LUOV Signature Scheme. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1071–1084, 2020.
23. Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume. On the importance of protecting Δ in SFLASH against Side Channel Attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(1):123–131, 2005.
24. Aesun Park, Kyung-Ah Shim, Namhun Koo, and Dong-Guk Han. Side-channel attacks on Post-Quantum Signature Schemes based on multivariate quadratic equations:-Rainbow and UOV. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 500–523, 2018.

25. David Pokornỳ, Petr Socha, and Martin Novotnỳ. Side-Channel Attack on Rainbow Post-Quantum Signature. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 565–568. IEEE, 2021.
26. Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.
27. Kyung-Ah Shim and Namhun Koo. Algebraic Fault Analysis of UOV and Rainbow with the leakage of random vinegar values. *IEEE Transactions on Information Forensics and Security*, 15:2429–2439, 2020.
28. Peter W Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM review*, 41(2):303–332, 1999.
29. Rainer Steinwandt, Willi Geiselmann, and Thomas Beth. A theoretical DPA-based cryptanalysis of the NESSIE candidates FLASH and SFLASH. In *International Conference on Information Security*, pages 280–293. Springer, 2001.
30. Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Improved Key Recovery of the HFEv-Signature Scheme. *Cryptology ePrint Archive*, 2020.
31. Enrico Thomae. A generalization of the Rainbow Band Separation attack and its applications to multivariate schemes. *Cryptology ePrint Archive*, 2012.
32. Haibo Yi and Weijian Li. On the importance of checking multivariate public key cryptography for Side-Channel Attacks: the case of enTTS scheme. *The Computer Journal*, 60(8):1197–1209, 2017.