Improvement of Side-Channel Attacks on Mitaka

Vladimir Sarde[1,2], Nicolas Debande[1]

[1] Cryptography & Security Group, IDEMIA Secure Transactions, Pessac, France
[2] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
vladimir.sarde@idemia.com, nicolas.debande@idemia.com

Abstract. Mitaka is a variant of Falcon, which is one of the three post-quantum signature schemes selected by the NIST for standardization. Introduced in 2021, Mitaka offers a simpler, parallelizable, and maskable version of Falcon. Our article focuses on its physical security, and our results are threefold. Firstly, we enhance a known profiled side-channel attack on an unprotected Mitaka implementation by a factor of 512. Secondly, we analyze the masked implementation of Mitaka described in the reference article, which incorporates a different sampler and a protective gadget. We expose the first side-channel flaw on this sampler. This flaw enables to break the masked implementation with a first-order side-channel attack. In this scenario, the key can be recovered using only three times more traces compared to the attack on the unprotected implementation. Finally, we discuss and recommend new countermeasures to mitigate these attacks.

Keywords: Post-quantum cryptography, Signature scheme, Mitaka, Side-channel analysis, Lattice

1 Introduction

The current asymmetric cryptographic schemes rely on the complexity of hard mathematical problems such as the computation of discrete logarithms or factorization. However, the development of quantum computers presents a security issue for these schemes as such computers have the potential to efficiently compute solutions to these mathematical problems. To address this risk, research on post-quantum cryptographic algorithms has intensified. The National Institute of Standards and Technology (NIST) is currently conducting a post-quantum cryptography standardization project to select and normalize some of these schemes. In the signature scheme category, three algorithms have already been selected and are in the drafting phase: Falcon [11], Dilithium [5] and SPHINCS$^+$ [3].

In this paper, our focus is on Mitaka [9], a variant of Falcon known for its simpler implementation. Both schemes follow the *hash and sign* paradigm and are based on the concept introduced in [13]. In this paradigm, the signature of a message is a point in a specific lattice, close to the hash of this message. For security purposes, these protocols use Gaussian samplers to randomize the selected lattice point. This aspect is the only one that differs between Falcon and Mitaka.

Falcon follows the *GPV framework* [12] optimized in [7], a fast yet complex sampler that can be challenging to implement accurately. On the other hand, Mitaka follows the framework developed by Ducas and Prest in [6], known as the *Hybrid Sampler*, which benefits from the same complexity while offering a simpler implementation, easy parallelization, and greater adaptability. The simplicity of Mitaka also allows several optimizations, making it a very compelling alternative for embedded systems. Firstly, Mitaka requires less computation than Falcon, drastically reduces the memory size of the orthogonal basis which is an important contributor to Falcon's memory consumption, and avoids the use of floating-point arithmetic [9]. These aspects are crucial for resource-constrained devices, such as smart cards, where implementing post-quantum standards presents a significant challenge. Secondly, Mitaka is easier to safeguard against side-channel attacks. The original article presents a masking scheme [9, Sect. 7], and in particular, a way to mask the sampler, which is recognized to be the most sensitive part of these protocols. This presents a significant advantage over Falcon, which is sensitive to various side-channel attacks, as those exposed in [16, 2, 15, 14, 21].

Among the previously mentioned side-channel attacks, the one introduced by Guerreau et al. in [14] on Falcon is particularly noteworthy. Recently, Zhang et al. improved this work in [21], significantly reducing the number of required traces from around 10^6 to 50,000. This enhancement makes the attack one of the most critical threats to Falcon within the current state of the art. Additionally, Zhang et al. have extended this approach to Mitaka, successfully retrieving the secret key with around $2 \cdot 10^6$ traces. The increased difficulty in attacking Mitaka arises from the hybrid sampler's operation, which significantly deteriorates the correlation between side-channel observations and the corresponding signatures. Consequently, extracting the key from Mitaka requires a significantly larger number of traces compared to the same attack on Falcon. Moreover, it is important to highlight that this attack targets an unmasked implementation of Mitaka. To the best of our knowledge, there have not been any practical attacks on the protected one. Nevertheless, it is worth mentioning that Prest in [20] has identified a theoretical flaw in the masking scheme related to the sampler when dealing with high-order masked implementations. Although this flaw remains theoretical, it compromises the security proof of the masking scheme.

Our Contributions. In this study, we significantly enhance the side-channel attack outlined in [21], targeting the unmasked implementation of Mitaka. Our improvement stems from the fact that the number of exploitable leakage points is not just one, as previously believed, but rather 512. This arises from the specific structure of Mitaka's hybrid sampler. Consequently, an attacker can construct their templates of equal quality and exploit the leakage with the same efficiency, all while reducing the number of generated signatures by a factor of 512. Furthermore, we delve into the analysis of the masking scheme proposed for Mitaka in the original article [9]. It was previously acknowledged in [20] that this scheme has a theoretical flaw in its sampler when handling high-order masked implementations. Here, we expose a new side-channel leakage in the one-dimensional sampler, recommended as a subroutine for this protected implementation. Leveraging this newfound leakage and the previous attack, we are able to exploit the theoretical vulnerability in a low-order masked context. Our study demonstrates that this masking scheme is susceptible to real-world first-order analysis, regardless of the order of protection. Subsequently, we showcase a successful first-order profiled attack against a first-order masked implementation. Finally, we propose several countermeasures to mitigate these attacks.

Roadmap. In Sect. 2, we lay out the notations used throughout the paper, recall the context, and previous fundamental results. Sect. 3 delves into the side-channel attack introduced in [14] and its enhancement presented in [21], targeting both Falcon and Mitaka. We describe our improvements for this attack in Sect. 4. Following this, Sect. 5 provides an analysis of the initially proposed masking scheme and of its corresponding sampler. The aforementioned attack is then adapted to this protected implementation. In Sect. 6, we present the experimental outcomes of these attacks, while Sect. 7 discusses potential countermeasures to safeguard Mitaka. Finally, we draw our conclusions in Sect. 8.

2 Preliminaries and Notations

2.1 Linear Algebra and Lattice

We use bold lowercase letters for vectors and bold uppercase letters for matrices. For a matrix $\mathbf{B}$, we note $\mathbf{b}_i$, the (i+1)-th column of $\mathbf{B}$. Vectors are considered as column vectors, we denote the transpose of a vector $\mathbf{v}$ by $\mathbf{v}^T$.

A lattice $\mathcal{L}$ of rank k is a discrete additive subgroup of $\mathbb{R}^l$, such that the vector subspace spanned by $\mathcal{L}$ is isomorphic to $\mathbb{R}^k$, with $k \leq l$. A lattice $\mathcal{L}$ spanned by a basis $\mathbf{B}$ is noted $\mathcal{L}(\mathbf{B})$. Several hard problems are known for lattices. The Mitaka scheme signs by solving a random instance of the *approximate closest vector problem*, named $ApproxCVP_\gamma$: given a basis $\mathbf{B}$ of a lattice $\mathcal{L} \subset \mathbb{R}^l$, and a point $y \in \mathbb{R}^l$, it is difficult to find $x \in \mathcal{L}$ such that $||x - y|| \leq \gamma$ for a fixed $\gamma \in \mathbb{R}$.

2.2 Template and Masking

The power consumption of a computing device is linked to the Hamming weight of the processed variables, a result of its physical structure. An attacker can exploit this property to perform template attacks, which involve two main phases. In the first phase, the attacker builds power consumption models for specific values of an intermediate variable using a device with a known secret key. In the second phase, the attacker matches a power trace from the target device with its templates to estimate the intermediate variable value.

There are several methods to protect an algorithm from these side-channel attacks. One of the most effective countermeasures is to mask the manipulated sensitive variables with random values. This breaks the correlation between the variable and the power consumption as the mask is unknown to the attacker. In this paper, we will focus on arithmetic masking. An arithmetic mask of degree

$t \in \mathbb{Z}$ on a variable a is a set of $t+1$ shares $\left(a^{(i)}\right)_{0\leq i\leq t}$ such that $a = \sum_{i=0}^{t} a^{(i)}$ and any subset of $l \leq t$ shares is independent of the secret a. For readability, we will note $\left(a^{(i)}\right)_{0\leq i\leq t} := [\![a]\!]$.

2.3 NTRU

Mitaka is defined over NTRU modules. Let $\mathcal{K}$ be the cyclotomic field defined by $\mathcal{K} = \mathbb{Q}(\zeta_n) \simeq \mathbb{Q}[x]/(x^n+1)$ with n a power of two and ζ_n a $2n$-th primitive root of 1. In this paper, we use $n = 512$, as our analysis focus on MITAKA-512. The ring of algebraic integers of $\mathcal{K}$ is $\mathcal{R} = \mathbb{Z}[\zeta_n] \simeq \mathbb{Z}[x]/(x^n+1)$. In this scheme, a secret key consists of two polynomials $f, g \in \mathcal{R}$ such that f is invertible modulo q, where $q \in \mathbb{N}$. In practice, f and g have small coefficients. From these two polynomials, one can calculate $F, G \in \mathcal{R}$ such that $fG - gF = q$. The public key is defined by $h = f^{-1}g \pmod q$ and the NTRU module associated to h is $\mathcal{L}_{\text{NTRU}} = \{(u,v) \in \mathcal{R}^2 :\ uh - v = 0 \pmod q\}$. A public basis of $\mathcal{L}_{\text{NTRU}}$ is $\{(1,h),(0,q)\}$, and a secret one is $\{(f,g),(F,G)\}$. We will note these bases by:

$$\mathbf{B}_p = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \text{ and } \mathbf{B}_s = \begin{pmatrix} f & F \\ g & G \end{pmatrix}.$$

The NTRU module $\mathcal{L}_{\text{NTRU}}$ is often seen as a lattice by associating a polynomial $h = \sum_{i=0}^{n-1} h_i x^i$ to a $n \times n$ matrix $M_h = [h, xh, ..., x^{n-1}h]$ where the j^{th} column corresponds to the coefficients of the polynomial $x^{j-1}h = \sum_{i=0}^{n-1} h_i x^i x^{j-1} \in \mathcal{R}$. Notice that multiplying these matrix is equivalent to multiplying polynomials in $\mathcal{R}$. In Mitaka, this equivalence leads to a $2n$-dimensional NTRU lattice, with a secret basis composed of small vectors.

2.4 Orthogonalization

Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and $\mathbf{B}$ a basis of $\mathbb{R}^n$. We define $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=0}^{n-1} u_i v_i$ as the inner product. To compute the orthogonal basis associated to $\mathbf{B}$ with respect to the inner product, we use the Gram–Schmidt process.

The NTRU module is a free $\mathcal{R}$-module of rank 2 in $\mathcal{K}^2$. Write $\mathbf{x}, \mathbf{y} \in \mathcal{K}^2$ by $\mathbf{x} = (x_0, x_1)$ and $\mathbf{y} = (y_0, y_1)$. There exists a natural $\mathcal{K}$-bilinear form over $\mathcal{K}^2$ defined by $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{K}} := {x_0}^* y_0 + {x_1}^* y_1 \in \mathcal{K}$, with * representing multiplication in the field $\mathcal{K}$. From this bilinear form, we can deduce a notion of orthogonality with respect to the module structure. To construct the orthogonal in this context, we just apply the Gram-Schmidt process. Thus, for $\mathbf{b}_0, \mathbf{b}_1 \in \mathcal{K}^2$ two linear independent vectors, we obtain: $\widetilde{\mathbf{b}}_0 := \mathbf{b}_0$, $\widetilde{\mathbf{b}}_1 := \mathbf{b}_1 - \langle \mathbf{b}_1, \widetilde{\mathbf{b}_0} \rangle_{\mathcal{K}} / \langle \widetilde{\mathbf{b}_0}, \widetilde{\mathbf{b}_0} \rangle_{\mathcal{K}} \cdot \widetilde{\mathbf{b}}_0$.

In this article, we use the following notations. We write $\mathbf{B} = \{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{2n-1}\}$ the basis of the lattice, $\mathbf{B}^{\mathcal{K}} = \{\mathbf{b}_0^{\mathcal{K}}, \mathbf{b}_1^{\mathcal{K}}\}$ the same basis seen as polynomials, $\widetilde{\mathbf{B}} = \{\widetilde{\mathbf{b}_0}, \widetilde{\mathbf{b}_1}, \ldots, \widetilde{\mathbf{b}_{2n-1}}\}$ the orthogonalized basis with respect to the lattice structure (i.e. with respect to $\langle .\ ,\ . \rangle$), $\mathbf{H}^{\mathcal{K}} = \{\mathbf{h}_0^{\mathcal{K}}, \mathbf{h}_1^{\mathcal{K}}\}$ the orthogonalized basis with respect to the free $\mathcal{R}$-module structure (i.e. with respect to $\langle .\ ,\ . \rangle_{\mathcal{K}}$), and $\mathbf{H} = (\mathbf{h}_i)_{i\in[0,2n)}$ the vectors of the basis $H^{\mathcal{K}}$ seen in the lattice structure. Notice that, by construction, $\mathbf{h}_i = \mathbf{b}_i$ if $i \leq n$ as $\mathbf{h}_0^{\mathcal{K}} = \mathbf{b}_0^{\mathcal{K}}$.

2.5 Gaussian Distributions

The Gaussian function with center $\mathbf{c} \in \mathbb{R}^m$ and standard deviation $\sigma \in \mathbb{R}$ is defined by the function $\rho_{\sigma,c}(\mathbf{x}) = \exp\big(-\frac{||\mathbf{x}-\mathbf{c}||^2}{2\sigma^2}\big)$. From this definition, we can construct a Gaussian distribution of density $D_{\sigma,c}(\mathbf{x}) = \frac{\rho_{\sigma,c}(\mathbf{x})}{(2\pi\sigma^2)^{m/2}}$. We can extend this definition to a full-rank lattice $\mathcal{L}$. Note $\rho_{\sigma,c}(\mathcal{L}) =: \sum_{x\in\mathcal{L}} \rho_{\sigma,c}(x)$. The discrete Gaussian distribution over $\mathcal{L}$, with center $\mathbf{c}$ and standard deviation $\sigma \in \mathbb{R}$, is defined by the probability function $D_{\mathcal{L},\sigma,c}(\mathbf{v}) = \frac{\rho_{\sigma,c}(\mathbf{v})}{\rho_{\sigma,c}(\mathcal{L})}$.

In this article, we are especially interested in the integer Gaussian $D_{\mathbb{Z},\sigma,c}$ with $c \in \mathbb{R}$. Finally, notice that the case $c \in [0,1)$ is enough to study as $D_{\mathbb{Z},\sigma,c} = l + D_{\mathbb{Z},\sigma,c-l}$ for all $l \in \mathbb{Z}$.

2.6 Mitaka and the Gaussian Samplers

In this part, we briefly outline the Mitaka scheme [9]. This protocol is based on the concept of *hash and sign* in a lattice. Let $\mathcal{L}$ be our NTRU lattice in $\mathbb{R}^{2n}$. Concretely, to sign a message $\mathbf{m}$, one hashes it with a salt r to a random point $\mathbf{c}$ in $\mathbb{R}^{2n}$ that is a priori not in $\mathcal{L}$. Then the signer solves the ApproxCVP, using the secret basis composed of small vectors of $\mathcal{L}$, to find $\mathbf{z} \in \mathcal{L}$ close to $\mathbf{c}$. Note that the public key is composed of long vectors of $\mathcal{L}$ from which it is hard to compute a solution. The signature is finally defined as $\mathbf{s} = \mathbf{c} - \mathbf{z}$. An observer can verify the signature by checking the resolution of ApproxCVP, i.e., by confirming that z belongs to $\mathcal{L}$ and that $||\mathbf{s}|| < \gamma$ for a public parameter γ.

Many variants of the *hash and sign* strategy have been proposed: [13], [12], [11], [9]. The main difference between all these schemes is the way they implement the resolution of ApproxCVP. This resolution, with the knowledge of a good basis, often relies on two algorithms: *Babai's Rounding* and *Babai's Nearest Plane*. Let $\mathbf{B}$ be our private basis of $\mathcal{L}$ and $\mathbf{c}$ be a point in $\mathbb{R}^{2n}$, these algorithms aim at finding a point $\mathbf{z} \in \mathcal{L}(\mathbf{B})$ as close as possible to $\mathbf{c}$. Both are explained in detail and illustrated in [19, Sect. 2.2]. The idea of Babai's Rounding is as follows: one calculates the coordinates of $\mathbf{c}$ in $\mathbf{B}$, seen as a basis of $\mathbb{R}^{2n}$, and then rounds every coordinate c_i to the closest integer. The point $\mathbf{z}$ resulting is in $\mathcal{L}$ as it is an integer combination of vectors of $\mathbf{B}$.

On the other hand, the idea of Babai's Nearest Plane is to construct $\mathbf{z}$ by successively projecting orthogonally $\mathbf{c}$. One starts by projecting $\mathbf{c}$ on $\widetilde{\mathbf{b}_{2n-1}}$, rounds the result to an integer z_{2n-1}, and subtracts $z_{2n-1}\mathbf{b}_{2n-1}$ to $\mathbf{c}$. The newly obtained $\mathbf{c}$ is in the hyperplane spanned by $\{\widetilde{\mathbf{b}_{2n-2}}, \widetilde{\mathbf{b}_{2n-3}}, \dots, \widetilde{\mathbf{b}_0}\}$ modulo the rounding. The process is then iterated with $2n-2,\ 2n-3, \dots,\ 0$. At the end, one can compute $\mathbf{z} = \sum_{i=0}^{2n-1} z_i\mathbf{b}_i \in \mathcal{L}$ close to $\mathbf{c}$.

Without any randomization, a signature based on these two previous algorithms follows a distribution that depends on the geometry of the secret basis. This is explained in detail and well illustrated in [19, Sect. 2.4, 2.5]. To counteract this flaw, a Gaussian noise is added during the rounding to ensure that the signature follows a public Gaussian distribution which does not depend on the secret basis. Concretely, instead of straightly rounding the coefficients in $\mathbb{Z}$

to calculate the z_i, an integer following a Gaussian distribution $D_{\mathbb{Z},\sigma}$ centered in the float value is sampled.

Therefore, the security of these schemes relies on the capacity to precisely and securely simulate the distribution $D_{\mathbb{Z},\sigma,x}$, with $x \in [0, 1)$. Note that when Babai's Rounding algorithm is used, a continuous Gaussian D_σ is first subtracted before performing the randomize rounding to prevent a bias on the covariance of the Gaussian over $\mathcal{L}$ [18]. These algorithms solving the ApproxCVP while including a randomization are called samplers.

The randomization of Babai's Nearest Plane was first used as a sampler, named *KGPV sampler*, in [12] and is written in Alg. 1. Note that Falcon uses an improvement of this sampler, named *Fast Fourier Nearest Plane*, described in [7]. However, the ideas of our study apply the same way in both cases.

The Mitaka's sampler is a mix between the ideas of Babai's Nearest Plane and Babai's Rounding, and is called the *Hybrid Sampler*. The structure of this sampler, introduced in [6], follows the framework of the KGPV sampler but at the ring level, i.e., in $\mathcal{R}$. The projections are computed in $\mathcal{K}$ with $\langle . \, , . \rangle_{\mathcal{K}}$. The randomization is accomplished at the ring level in two steps, as in the randomized Babai's Rounding algorithm. First, a continuous Gaussian is subtracted from the projection, a polynomial in $\mathcal{K}$, and then every coefficient of this polynomial is rounded by a discrete Gaussian distribution over $\mathbb{Z}$. Eventually, the obtained polynomial corresponds to a z_i in Alg. 1. As $\mathbf{c} \in \mathcal{K}^2$, only two projections, and thus iterations, are needed. Note that the pseudocode is available in [9, Alg. 3].

Algorithm 1: KGPV Sampler.

```
Input  : An orthogonal basis $\widetilde{\mathbf{B}}$ of $\mathbf{B}$, a center $\mathbf{c}$, and a standard deviation $\sigma$.
Output: A point $\mathbf{z} \in \mathcal{L}(\mathbf{B})$ following a distribution close to $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$.
1 $z \leftarrow 0$
2 for $i \leftarrow 2n - 1$ to $0$ :
3     $c_i \leftarrow \langle \mathbf{c}, \widetilde{\mathbf{b}}_i \rangle / ||\widetilde{\mathbf{b}}_i||^2$
4     $\sigma_i \leftarrow \sigma / ||\widetilde{\mathbf{b}}_i||$
5     $z_i \leftarrow$ SamplerZ$(\sigma_i, c_i - \lfloor c_i \rceil) + \lfloor c_i \rceil$
6     $\mathbf{c} \leftarrow \mathbf{c} - z_i \mathbf{b}_i$
7     $\mathbf{z} \leftarrow \mathbf{z} + z_i \mathbf{b}_i$
8 return $\mathbf{z}$
```

2.7 Generation of a Discrete Gaussian Distribution

Both Mitaka and Falcon need a subroutine to generate a discrete Gaussian distribution over $\mathbb{Z}$, i.e., $D_{\mathbb{Z},\sigma,c}$ with c in $[0, 1)$. Note that here, c corresponds to the decimal part of c_i the projection of $\mathbf{c}$ on $\widetilde{\mathbf{b}}_i$, as in Alg. 1. This subroutine is really challenging to implement for two reasons. Firstly, the sampler has to generate a Gaussian distribution for which the center and the standard deviation are dynamic, i.e., dependent on the point and the key. Secondly, the sampler is a key target for side-channel attacks. The most efficient and secure way is to use the *SamplerZ*, from [11, Sect. 3.9.3][3], described in Alg. 2. It is a three-step algorithm that takes as input the targeted center $c \in [0, 1)$ and a standard deviation σ. Firstly, a table based sampler, named *BaseSampler*, is called to generate z^+ following a discrete half-Gaussian distribution centered on 0 and with a constant standard deviation. Secondly, a random bit b is sampled to compute $z = b + (2b-1)z^+$. It can be seen as the sign of the random variable. The integer z thus follows a bimodal Gaussian distribution. Thirdly, a rejection sampling is applied to obtain the targeted distribution. Note that z, the output of the SamplerZ, is used to compute z_i in Alg. 1 a coordinate of $\mathbf{z}$ in the basis $\mathbf{B}$.

Algorithm 2: SamplerZ.

```
Input  : A center $c \in [0, 1)$, a standard deviation $\sigma \in [\sigma_{\min}, \sigma_{\max}]$.
Output: An integer $\mathbf{z}$ following a distribution close to $D_{\mathbb{Z},\sigma,c}$.
1 $z^+ \leftarrow \texttt{BaseSampler}()$
2 $b \overset{\$}{\leftarrow} \{0, 1\}$
3 $z \leftarrow b + (2b-1)z^+$
4 $x \leftarrow -\frac{(z-c)^2}{2\sigma^2} + \frac{(z^+)^2}{2\sigma_{\max}^2}$
5 $p \leftarrow \frac{\sigma_{\min}}{\sigma} \cdot \texttt{exp}(x)$
6 return $z$ with probability $p$, otherwise restart
```

3 Previous Work

In 2022, Guerreau et al. introduced in [14] a new template attack [4] on Falcon. This attack was then improved by Zhang et al. [21] the following year. As described before, Mitaka's sampler works partly the same way as Falcon's one, but at the ring level. Thus, the authors of [21] proposed applying the same attack to Mitaka, ignoring the continuous Gaussian noise subtracted in the randomized Babai's Rounding algorithm. In this section, we outline this attack without considering the continuous Gaussian noise for more generality and simplicity.

3.1 Attack Overview

Let us write the signature in the orthogonal basis as $\mathbf{s} = \mathbf{c} - \mathbf{z} = \sum_{i=0}^{2n-1} y_i \widetilde{\mathbf{b}_i}$. The authors suggested using side-channel observations to gather information on y_0, which we will denote as the offset on the vector $\widetilde{\mathbf{b}_0}$. This way, an attacker could be able to classify signatures into two sets according to the value of y_0. In each of these sets, the distribution of the signatures is biased in the direction of $\widetilde{\mathbf{b}_0}$. The

[3] Refer also to the Specifications v1.0 Sect. 4.4 for more details.

attacker can then approximate the direction of $\widetilde{\mathbf{b}_0}$ thanks to a statistical metric. Additionally, the norm of $\widetilde{\mathbf{b}_0}$ is also estimated. Since $\widetilde{\mathbf{b}_0} = \mathbf{b}_0$ by construction of the orthogonal basis, it is possible to recover the exact $\mathbf{b}_0$ through an exhaustive search. This leads to knowledge of the whole secret basis as $\mathbf{b}_0 = (f \quad g)$, from which F and G can be computed, thereby reconstructing the secret key.

Exploited Leakages. Firstly, it is worth recalling that having knowledge of z_i computed by the sampler is essentially the same as knowing the offset y_i, with the center $c \in [0, 1]$ of the Gaussian being the only differing factor. Indeed, at the $(2n-i)^{\text{th}}$ iteration, with $2n > i \geq 0$, the projection of $\mathbf{c} - \sum_{l=i+1}^{2n-1} z_l \mathbf{b}_l$ is calculated on the vector $\widetilde{\mathbf{b}_i}$, then the SamplerZ adds a Gaussian deviation to this projection to obtain the coefficient of $\mathbf{b}_i$. At this time, the coefficient of $\widetilde{\mathbf{b}_i}$ in the signature is exactly the offset coming from the Gaussian of center c produced by the SamplerZ. Then, the next iterations do not have any impact on this coefficient because we are handling $\mathbf{b}_j$ with $j < i$, yet $\mathbf{b}_j$ is orthogonal to $\widetilde{\mathbf{b}_i}$ when $j < i$.

The attack outlined in [14] and improved in [21] takes advantages of a side-channel flaw called half Gaussian leakage to identify signatures such that y_0 is in $(-1\ ,\ 1]$. Furthermore, in [21], the authors exposed a second side-channel flaw to obtain information on z_0: the sign leakage. In this study, we focus to this second leakage as it yields better results in the context of Mitaka. The targeted part in both the KGPV sampler and the Hybrid Sampler is the rounding with a discrete Gaussian distribution over $\mathbb{Z}$. The idea is to recover through template analyzes the bit b, which defines the sign of the offset produced by the SamplerZ. This bit is manipulated three times for different calculations in lines 2, 3 and 4 in Alg. 2. Depending on the b value, some variables flip sign, leading to important variations of Hamming weight and on power consumption.

Approximation of $\mathbf{b_0}$'s Direction. According to the sign drawn during the computation of z_0, and thus the sign of the offset y_0, the signatures are separated into two sets. Both sets can be put together by multiplying signatures with the minus sign by -1. From that classification, an attacker can compute an approximation, noted $\mathbf{b}_0^{raw}$, of the direction of $\mathbf{b}_0$. The authors of [21] proposed to use the first statistical moment, i.e., the expectation, to recover a multiple of $\mathbf{b}_0$. Indeed, the expectation for our selection of signatures is zero in every direction, as the distribution is Gaussian, except in the one of $\widetilde{\mathbf{b}_0}$. The more considered signatures, the more accurate the approximation is.

Approximation of $\mathbf{b_0}$'s Norm. By construction, each coefficient of $\mathbf{b}_0 = (f \quad g)$ is an integer sampled from $D_{\mathbb{Z},\sigma,0}$ with $\sigma_0 = 1.17\sqrt{q/2n}$. Thus, the norm can be approximated by $||\mathbf{b}_0|| \approx \sqrt{\sum_0^{2n-1} \sigma_0^2} = 1.17\sqrt{q}$. Then, one adjusts $\mathbf{b}_0^{raw}$ with this estimated norm, leading to a new vector named $\mathbf{b}_0^{adj}$. Eventually, all the coefficients of $\mathbf{b}_0^{adj}$ are rounded off to give the final approximation $\mathbf{b}_0^{int} = (f^{int} \quad g^{int})^T$.

Exhaustive Search. Eventually, an exhaustive search to recover $\mathbf{b}_0$ is performed independently on f^{int} and g^{int}. For every tried polynomial f^{int} (resp. g^{int}), g^{int} (resp. f^{int}) can be computed thanks to h. This allows for splitting the research in two and provides a verification criterion.

Let $\mathbf{e} = \mathbf{b}_0^{int} - \mathbf{b}_0$ be the error on $\mathbf{b}_0^{int}$. To make the exhaustive search practical, the strategy is to gather enough traces such that $||\mathbf{e}||_\infty \leq 1$ and $||\mathbf{e}||_1 \leq 7$. In practice, the limiting criterion is the second one. Hence, it is enough for an attacker to try every f^{int} and g^{int} with 3 errors of weight 1. This guarantees a total recovery of $\mathbf{b}_0$ in approximately 30 minutes. In Sect. 6, we provide an estimate of the number of traces required to achieve $||\mathbf{e}||_1 \leq 7$ in practical scenarios.

3.2 Practical Considerations

The authors of [21] obtained an accuracy of 100% on an ARM Cortex-M4 STM32F407IGT6 microprocessor for the sign classification, by using measured power consumption as side-channel information. In these conditions, they are able to recover the secret key with 170 000 traces on Falcon using this leakage. However, the continuous Gaussian noise disturbs the approximation of the $\mathbf{b}_0$'s direction by deteriorating the correlation between side-channel observations and the corresponding signatures. This increases significantly the number of required traces, leading to a key recovery with around $2 \cdot 10^6$ traces on Mitaka.

Note that the simultaneous exploitation of both half-Gaussian and sign leakage leads to secret key recovery with 45 000 traces for Falcon and $1.8 \cdot 10^6$ traces for Mitaka.

4 New Improvements of the Attack on Mitaka

In this section, we describe improvements of the previous attack on Mitaka to significantly decrease the number of needed traces.

4.1 Norm Estimation

As a first improvement, we propose to enhance the adjustment $\mathbf{b}_0^{adj}$ of $\mathbf{b}_0^{raw}$. In [21], Zhang et al. proposed adjusting $\mathbf{b}_0^{raw}$ by trying different norms in $\{1.17\sqrt{q}, 1.17\sqrt{q} - 1, \ldots, 1.17\sqrt{q} - 10\}$. We suggest taking the problem in an other way, by adjusting the norm of $\mathbf{b}_0^{adj}$ such that it minimizes $||\mathbf{b}_0^{adj} - \mathbf{b}_0^{int}||_1$ while keeping $||\mathbf{b}_0^{adj}|| \approx 1.17\sqrt{q}$. The idea is that if our estimate of the norm is accurate, we expect that the coefficients of $\mathbf{b}_0^{adj}$ will closely approximate the integer to which they will be rounded off. In our tests, we showed that this improves our estimation of the norm. Hence, it allows the attacker to find a better approximation of $\mathbf{b}_0^{int}$ and thus reduce the number of traces required to obtain an error norm $||\mathbf{e}||_1 \leq 7$. This also allows for avoiding the trials of all norm possibilities until a success is obtained.

Note that the exhaustive search from $\mathbf{b}_0^{int}$ to $\mathbf{b}_0$ can also be improved by prioritizing changes to coefficients further from an integer.

4.2 Multiplying the Points of Interest

The second contribution of this article is to notice that, in the context of Mitaka, the attack can be performed on 512 calls of the SamplerZ. Indeed, the previous attacks were only using the call that calculates the coefficient of $\mathbf{b}_0$. In Falcon, the projections and offsets are computed on the orthogonal basis $\widetilde{\mathbf{B}}$. Thus, the exploitation of the same leakage from another SamplerZ call would approximate another $\widetilde{\mathbf{b}}_i$, which is complicated to use to recover information on $\mathbf{B}$. Conversely, in Mitaka the sampling follows the basis $\mathbf{H}^{\mathcal{K}}$ on which, because of the Gram-Schmidt procedure, the first polynomial is $\mathbf{h}_0^{\mathcal{K}} = \mathbf{b}_0^{\mathcal{K}}$. Thus, the basis of the lattice associated to $\mathbf{H}^{\mathcal{K}}$, noted $\mathbf{H}$, has the same 512 first vectors as the secret basis $\mathbf{B}$. Therefore, by construction, any $\mathbf{h}_i = \mathbf{b}_i = (x^i f \;\; x^i g)^T$ with polynomial multiplications in $\mathcal{R} \simeq \mathbb{Z}[x]/(x^n+1)$. In other words, every approximation of $\mathbf{h}_i = \mathbf{b}_i$ is actually an approximation of $\mathbf{b}_0$ with re-ordered coefficients.

Thus, in the context of Mitaka, the 512 calls to BaseSampler leak exploitable information. Moreover, these 512 calls produce leakages that are independent of each other. Indeed, each call to BaseSampler uses fresh data and generates a new random number. The physical noise of different side-channel observations is also often assumed to be independent. To summarize, an attacker could try to approximate all $(\mathbf{b}_i)_{i \in [\![0,511]\!]}$, realign them to get 512 independent approximations of $\mathbf{b}_0$, and finally average them to obtain a noise-free $\mathbf{b}_0^{raw}$. The key recovery is then performed with the exact same method and we expect to obtain the key from 512 times fewer signatures. This means a complete key recovery with 4500 traces instead of 2.25 million.

Note that a gain of factor 512 can thus also be achieved on the number of traces needed to build the templates. Indeed, one can build a template set by gathering the $512 \times N$ segments from N signature computations.

Nevertheless, in practice, an additional difficulty appears compared to the previous attack: the attacker needs to identify the 512 correct points of interest in a power trace. Indeed, the SamplerZ is called 512 times, one per coordinate, and each call generates potentially several rejected samplings before one is accepted. In the previous state of the art, it was accepted that one could locate the right call of BaseSampler, corresponding to the accepted sampling z_0. In our context, each call to the SamplerZ is easily identified on a power trace. Then, to distinguish the accepted samplings from the rejected ones, we can analyze the reference implementation of Mitaka in C [8]. In the SamplerZ, if a sampling is rejected, we restart an iteration of the sampler procedure immediately. On the other hand, if the sampling is accepted, we get out of the call, increment a counter, recover the next coefficient in the structure, come back to the SamplerZ, compute a floor value and finally start the same loop as before. We stress that all these operations will make differences in the power trace, giving criteria to differentiate the status of the sampling.

4.3 Template Construction

In Sect. 3.2, we mentioned the impact of the continuous Gaussian noise to exploit the signatures in a key recovery. Additionally, this noise also impacts the creation of the templates. Indeed, during the building phase, the signatures should be labeled according to the sign of the discrete offset, drawn in the SamplerZ, in order to create a training set for the template analysis. However, from the knowledge of the key and the signature, the attacker can only compute the final offset y_i, which is the sum between the discrete offset and the continuous offset. Consequently, the continuous offset can add errors in the labeling, thus reducing the quality of the templates.

To mitigate the influence of this noise, an attacker could select, for the training, signatures such that $y_i > l$ (resp. $y_i < l$) with $l \in \mathbb{R}^+$ (resp. $l \in \mathbb{R}^-$) a wisely chosen threshold. The larger l is, the higher the probability that the sign drawn and the sign of y_i matches. On the other hand, the larger l is, the smaller the number of signatures selected for the construction of the templates. This trade-off is discussed in Sect. 6.3.

5 Attack Against Mitaka Masked Implementation

In the original article introducing Mitaka [9], the authors initially proposed and proved an efficient masking scheme for this algorithm. This reliable security against side-channel attacks is one of the main advantages of Mitaka over Falcon. The construction was proven in the t-SNI model introduced in [1]. Quickly, this proof was broken theoretically by Prest in [20]. In this section we will present this masking scheme, its weakness, and a practical attack on masked Mitaka.

5.1 The Masking Scheme

We are interested in the portion of the masking scheme related to the sampling of $D_{\mathbb{Z},r,c}$, usually performed by the SamplerZ. This is operated by the *Gauss Share-by-Share* algorithm, described in Alg. 3 which allows to generate a masked variable $[\![z]\!]$ following a Gaussian distribution $D_{\mathbb{Z},r,c}$. More precisely, Gauss Share-by-Share takes as input a masked center $[\![c]\!] =: (c^{(i)})_{i\in[\![0,t]\!]}$, and an unmasked standard deviation r. For the i-th share $c^{(i)}$, it generates the distribution $D_{1/B\cdot\mathbb{Z},r/\sqrt{t+1},c^{(i)}}$ such that the sum of all the shares gives $\sum D_{1/B\cdot\mathbb{Z},r/\sqrt{t+1},c^{(i)}} = D_{1/B\cdot\mathbb{Z},r,c}$ with $B := \lceil\sqrt{2(t+1)}\rceil$. To guarantee an output in $\mathbb{Z}$, and not in $1/B \cdot \mathbb{Z}$, an additional rejection sampling is used.

Quickly after the presentation of Mitaka, Prest [20] highlighted a flaw in the proof of the initially proposed masking scheme. He also described a theoretical attack that targets the masked implementation of the sampler in the t-probing model. His attack uses the fact that the differences between the shares $c^{(i)}$ of the input and the shares $z^{(i)}$ of the output are Gaussian and not uniform. That implies that if an attacker has information on the difference between some shares, she has information on the whole difference $c - z$ because the unknown part, resulting of the others shares, is 0 on average. This allows an attack of degree l on an implementation with a masking scheme of degree $t > l$, with $t \geq 4$.

Algorithm 3: Gauss Share-by-Share.

```
Input  : A center $[\![c]\!]$ arithmetically masked with degree $t$, an unmasked
         standard deviation $r$.
Output: A masked $[\![z]\!]$ such that $z$ follows a distribution close to $D_{\mathbb{Z},c,r}$.
1 $B \leftarrow \lceil \sqrt{2(t+1)} \rceil$
2 for $i \leftarrow 0$ to $t$ :
3     $z^{(i)} \leftarrow D_{1/B\cdot\mathbb{Z},c^{(i)},r/\sqrt{t+1}}$
4 if $z \notin \mathbb{Z}$                                // To check in a secure way
5     restart to step 2
6 return $[\![z]\!] = (z^{(0)}, \ldots, z^{(t)})$
```

5.2 Another Sampler

The generation of $D_{\mathbb{Z},r,c}$ is directly accomplished by the SamplerZ in the unmasked version. In the masked version, one has to generate $D_{1/B\cdot\mathbb{Z},r/\sqrt{t+1},c^{(i)}}$. To do so, the authors of Mitaka [9] chose a table-based approach, following the technique introduced by Micciancio and Walter [17]. This sampler allows for efficient sampling from Gaussian discrete distributions with varying centers and standard deviations. The part of the sampler dedicated to the generation of a discrete Gaussian with varying center $c \in [0,1]$ is referred to SamplerC and is detailed in Alg. 4. The main advantage of this algorithm is that it only requires base samplers with constant parameters. These last samplers are much easier to optimize and can be used to precompute samples in offline mode. Different approaches can be used to implement these subroutines of SamplerC, depending on the time and memory performance sought. In our study, we will rather consider the Knuth-Yao's implementation as a subroutine, even if that has only a low impact on our work.

The idea behind the SamplerC is rather simple. It takes as input the binary decomposition of the center $c = b_1 ... b_k \in 2^{-k}\mathbb{Z}$ with k digits of precision and randomizes-round it digit by digit with a discrete Gaussian. The first recursion computes g_k with the base sampler $D_{b_k+2\mathbb{Z},r_0}$. Hence, c and $2^{-k}g_k$ have the same last bit and therefore $c - 2^{-k}g_k \in 2^{-k+1}\mathbb{Z}$. Thus at the end of all recursions, the output g is such that $c - g$ is in $\mathbb{Z}$. By construction, g follows $D_{c+\mathbb{Z},r}$ and hence $c - g$ follows $D_{\mathbb{Z},r,c}$ as required. Notice that one can reduce the recursion depth by taking a center c in base $b > 2$. However, this requires storing b base samplers and their samples. Since memory is the first limiting criterion in a constrained device, we consider $b = 2$ in our analysis, which has little impact on our study.

5.3 Different Sampler, Same Leaks

In the masked version of Mitaka, the SamplerC is used as a subroutine of Gauss Share-By-Share, which splits the computation of the Gaussian deviation into $t + 1$ parts. However, in this subsection, we consider the hypothetical situation where the SamplerC is used directly by the Hybrid Sampler, as in the unmasked

Algorithm 4: SamplerC.

```
Input  : A center $c$ in $2^{-k}\mathbb{Z}$, a base sampler noted SamplerB for $D_{c'+2\mathbb{Z},r_0}$
         with fix $r_0$ and $c' = 0$ or 1.
Output: A real $g$ following a distribution close to $D_{c+\mathbb{Z},r}$.
1 if $k=0$
2     return 0
3 $g \leftarrow 2^{-k}\cdot$ SamplerB($2^k c$)
4 return $g+$ SamplerC($c-g$)                    // $c - g \in 2^{-k+1}\mathbb{Z}$
```

implementation. This allows us to better visualize the similarities between the leakages of the SamplerZ and the ones of the SamplerC. In this case, the coefficients of the polynomials $\mathbf{z}_i$ in $\mathcal{K}$, with $i \in \{0, 1\}$, used in the Hybrid Sampler are computed by taking $c-g$ with c the coefficients of the projection of $\mathbf{c}$ on $\mathbf{h}_i^{\mathcal{K}}$ and g the output of the SamplerC centered in c. Therefore, the sign of g directly provides the sign of the offset y, as defined in Sect. 3. Indeed, $g \in \{c, c+1, c+2, \ldots\}$ implies a positive offset, as $\mathbf{s} = \mathbf{c} - \mathbf{z}$, and conversely $g \in \{c-1, c-2, \ldots\}$ implies a negative offset. Hence, an attacker is able to recover the same information as previously by targeting the sign of g. On the other hand, notice that the SamplerC computes $g \sim D_{c+\mathbb{Z},r}$ by adding g_i whose size grows geometrically. Thus, the rough value of g is mainly determined by g_1 the result of the last call to the base sampler. Hence, an attacker can recover information on the sign of g observing only the sign of g_1. This relation is discussed in Sect. 5.4.

In the technique of [17], the base sampler computations are carried out offline. Thus, this part can be computed in batch and be stored in a secure way. However, the sampled value g_1 is still loaded, bitshifted and added. These several manipulations can be exploited by an attacker to construct her templates similarly to the previous case. Indeed, g_1 being small, negative and positive values have significantly different Hamming weights. Thus, as in the previous attack introduced in Sect. 3, one can build templates on the sign of g_1 that can be used to construct a bias set of signatures with respect to the sign. This leads to a similar attack as before.

Remark 1. The SamplerC computes $g \sim D_{c+\mathbb{Z},r}$ and the result following $D_{\mathbb{Z},r,c}$ is recovered by calculating $c - g$. This last subtraction could also lead to an important side-channel leakage, as sign variations occur depending on the result. This leakage could give perfect information on the sign. However, this leakage really depends on the implementation and on the analyzed device, so we do not consider it to retain a more general point of view.

5.4 Building the Templates

The attack adapted for SamplerC can be slightly modified to use the weakness of the masking scheme. In Gauss Share-By-Share, a distribution $D_{1/B\cdot\mathbb{Z},r/\sqrt{t+1},c^{(i)}}$ is computed for each share $c^{(i)}$ of the masked center $[\![c]\!]$. For an attacker willing to perform the same attack as before, the main difficulty comes from the creation of the templates. During the building phase, the attacker has, by hypothesis, access to the key and so can compute the offset y_0 applied in the direction of b_0. On the other hand, she can only observe shares of z_0 through side-channel observations. Let us assume that the attacker focuses on the sign of this offset. An ideal masking scheme should make the observation of one share independent of y_0. The attacker would thus be forced to combine the $t+1$ observations of all shares to retrieve an exploitable dependence and build her templates. This leads to an exponential growth in computing complexity and in the needed number of traces. Moreover, the combination of several observations is always technically challenging in practice.

Let us analyze the situation for an attacker trying to build a template to recover the sign of the value sampled for a given share in the specific case of $t = 1$, i.e., two shares in total. Let us assume, without loss of generality, that the attacker builds the template that aims at identifying a positive offset for the first share, noted $z^{(0)}$. As presented in 5.1, this masking scheme has a theoretical flaw in its conception. The discrete offset is the sum of all the shares, with the property that the sum of any subset of shares is 0 on average. Therefore each output of the SamplerC, called share by share, is directly correlated to the sign of the discrete offset y_i for a given coordinate. The attacker can thus use the sign of y_0 to isolate the signatures with positive first share in order to create her templates. Yet, this attack is hampered by different factors: the partiality of the leak, i.e., $\mathbb{P}(\text{sign}(g_1)|\text{sign}(g))$, the masking scheme, i.e., $\mathbb{P}(\text{sign}(z^{(0)}|\text{sign}(z))$, and the continuous Gaussian noise evoked in Sect. 2.6, i.e., $\mathbb{P}(\text{sign}(y)|\text{sign}(z))$. We show here, that these different bias do not prevent the construction. The partiality of the leak is estimated at 0.83 empirically by testing different combinations of centers and standard deviations that could occur in the context of Mitaka, ten thousand times each. The continuous Gaussian adds a level of noise whose intensity depends on the key. Thereafter, we will ignore this latter theoretically, as it is 0 on average, but consider it in every practical experimentation. Finally, the masking scheme decomposes the sampling into two shares, thus the couple of signs associated to a given couple of shares can take four values: $\{+,+\}_{0.25}$, $\{+,-\}_{0.25}$, $\{-,+\}_{0.25}$ and $\{-,-\}_{0.25}$, where the subscript 0.25 indicates the probability of each pair. However, if y_0, computed from the key by the attacker, is positive, then the distribution of the possible realizations becomes biased: $\{+,+\}_{0.50}$, $\{+,-\}_{0.25}$, $\{-,+\}_{0.25}$ and $\{-,-\}_{0}$. Indeed, the case $\{-,-\}$ leads always to a negative y, modulo the continuous Gaussian noise, as half of the mixed sign. From these observations, we can roughly assess an estimation of the rate of correct classification by $75\% \times 83\% + 25\% \times 17\% = 66.5\%$.

Furthermore, an attacker can also select signatures such that $y_0 > l$ with $l \in \mathbb{R}^+$, a wisely chosen threshold as explained in Sect. 4.3. This has a twofold impact. First, it reduces the proportion of small offset for a given share and thus reduces the partiality of the leak. Secondly, it enhances the bias in the masking scheme, as it reduces the proportion of shares with opposite signs in favor of shares with two same signs. In conclusion, an attacker is more likely to correctly label her side-channel observations than to mislabel them. Therefore, she has the ability to construct templates in a first-order manner and recover the key, despite the presence of masks. The experimental results are presented in Sect. 6.3.

Note that a similar analysis can be conducted for any degree of masking t, but the described bias decreases as t increases. Tab. 1 shows the probability estimation of correct labeling according to t. This rate gives only a rough idea, as is computed without considering the continuous offset.

Table 1. Probability of correct labeling g_1 knowing y depending on masking order t.

t	0	1	2	3	4	5	6
Probability	0.83	0.665	0.632	0.612	0.599	0.586	0.579

5.5 Exploiting the Templates

Once the templates are built, an attacker can try to recover the sign of every share computed by the SamplerC for a given execution of Gauss Share-by-Share during the matching phase. With this information, she can approximate the sign of the discrete offset generated and thus realize the same attack as in the case of the unmasked implementation. The success of this attack will depend, as before, on the partiality of the leak and the continuous Gaussian noise. The masking scheme no longer hinders the attack, as the templates are applied to each share. Nevertheless, a new parameter to consider for the attacker is the templates accuracy, noted a. It corresponds to $\mathbb{P}(\text{sign}(g_1)|L)$, with L the power trace and g_1 related to the specific targeted share.

Let us analyze here again the specific case of masked Mitaka with $t = 1$. The attacker aims at inferring the sign of the total offset z based on her template results regarding the sign of g_1 of each share. Concretely, if the observed sampling corresponds to $\{+, +\}$ (resp. $\{-, -\}$), the templates label the signature positively (resp. negatively). On the other hand, if the observed sampling has mixed signs, i.e. $\{-, +\}$ or $\{+, -\}$, the signature is discarded as no information is provided on the final sign. Recall that the leak gives only partial information, accurate at 83%, on the sign of one share. Let us now compute the theoretical success rate of the attack, depending on a. For each share independently, we have a probability a to detect the actual sign of g_1 and then a probability of 0.83 that this sign matches the sign of g. Thus, the probability of correctly predicting a $\{+, +\}$ (resp. a $\{-, -\}$) is estimated by $(a \times 0.83)^2$. However, in this context, we can show that the success rate of the classification of the sign of the total offset z using the strategy described above is actually equal to $0.83a + 0.17(1 - a)$. This arises from the symmetry of the situation and the fact that a mixed sign introduced by error has 1 chance over 2 of resulting in a final offset matching the predicted one. Experimental results corresponding to this masking scenario with degree $t = 1$ are available Sect. 6.2.

To conclude the description of this attack, we should also notice that the improvements discussed in Sect. 4.2 are relevant in a masked context too. Indeed, it is still possible to distinguish the rejected samplings from the accepted ones. However, it is worth noting that the masking scheme's initial computation of a sampling over $1/B \cdot \mathbb{Z}$ rather than $\mathbb{Z}$ introduces an additional level of rejection. Yet, if the masked sampling is rejected, the iteration restarts immediately. Conversely, if it is accepted, i.e., if it lies in $\mathbb{Z}$, the algorithm exits the Gauss Share-by-Share function, increments the counter, and performs $t + 1$ floating divisions to compute the next masked center. Thus, we expect to also distinguish rejections from acceptances at this level in a power trace. Therefore, this improvement enables us to reduce by a factor 512 the number of needed traces both to compute the templates during the building phase and to recover the key in the matching phase. This allows an attacker to be more selective in the traces used for construction, i.e., by using a larger l, leading to a very efficient attack.

6 Experimental Results

Our experiences on the unmasked implementation of Mitaka are based on the official implementation available in [8]. However, as far as we know, there is no public masked implementation of Mitaka. Thus, the corresponding experiments are based on our implementation of this scheme, following the description and pseudocode available in the original article [9, Sect. 7 and Annex I]. In the same manner, we use our own implementation for the SamplerC following the framework of [17]. In both masked and unmasked cases, we target the 512 points of interest as described in Sect. 4.2, which significantly reduces the required number of traces. Moreover, we apply the enhanced norm estimation from Sect. 4.1. Although having a minor impact on the trace count compared to the first improvement, it enables precise estimation of $\mathbf{b}_0$'s norm and allows one to skip the exhaustive search over multiple norm possibilities as seen in previous attacks.

6.1 On Naive Mitaka

In our context, signatures were generated using the official Mitaka implementation [8]. This process allowed us to export the exact intermediate variables. We then simulated the template analysis, i.e. the labeling phase by deliberately falsifying some intermediate variables to match the desired classification rate.

Note that in the naive setup, the classification rate is equal to the template accuracy a introduced in Sect. 5.5. This accuracy depends on the noise level of the environment and the quality of the measurements. Our results are depicted in Figure 1. It shows the weight of the error $\mathbf{e}$ as a function of the averaged number of traces used. Each curve is labeled with the corresponding classification success rate. These attacks exploit the sign leakage, except for the blue curve with pentagon markers which combines sign leakage and half Gaussian leakage, both with perfect classification, as defined in [21]. While the half Gaussian leakage is irrelevant in a masked context, it provides a useful comparison for unmasked scenarios. However, as the half Gaussian leakage does not enhance the attack's efficiency, we focused on the sign leakage to test various classification accuracies, i.e. different success rates for correct signature labeling. In [14, 21], the authors were able to construct such templates with a success rate of 1. The corresponding comparison curve in our graph is the green one with star markers. Notably, we achieved similar results to those presented in the original article but with 500 times fewer traces. Therefore, thanks to the proposed improvement, the required number of traces dropped significantly, from 2.2 million to only 4,500.

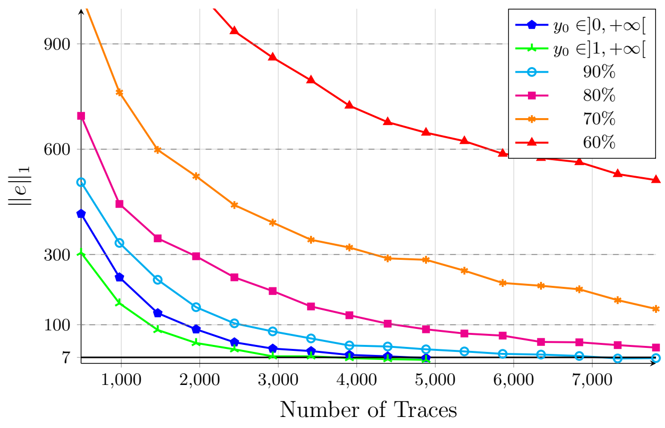

Fig. 1. Result of the improved attack on a naive implementation.

6.2 On Masked Mitaka

In the following, our experimental results correspond to the case of masked Mitaka with $t = 1$. We generated the signatures and the corresponding intermediate values from our own implementation, assuming that the sign of the targeted value is, as previously, recovered from side-channel with a given success rate. In our context, the targeted value is g_1, as defined in Sect. 5.3, independently in each share of the discrete Gaussian offset. .

As in Sect. 5.5, we will note a the accuracy of the templates. The experimental results of the attacks are described in Figure 2, which represents the weight of the error $\mathbf{e}$ as a function of the number of traces. The curves correspond to the results obtained for different levels of accuracy for the templates, varying from $a = 1$ in dark blue with pentagon markers to $a = 0.60$ in a noisier or in a more protected environment, in red with triangle markers. We add the curve corresponding to the perfect classification computed from the key in dotted green for comparison purposes. Our results take into account the continuous Gaussian noise, as we simulate the leakage after this noise was added. In the same manner, our implementation is subject to every actual bias and thus concretely describes an attacker's real configuration. Finally, note that the graph shows the number of generated traces, but only half of them are used, as mixed signs are discarded.

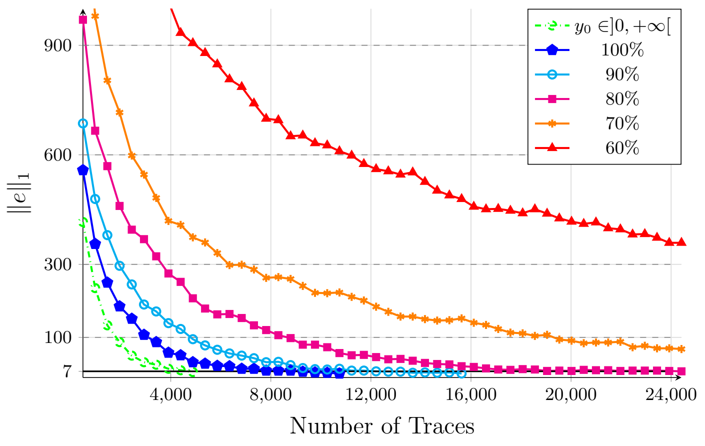

Fig. 2. Result of the improved attack on a masked implementation.

We notice that we obtain a graph that depicts curves that are very similar to the ones from Figure 1. Actually, the masking scheme can just be visualized as noise, as it increases the number of signatures mislabeled. This protection disturbs the attack but does not prevent it. Even with a lot of noise, implying a global classification of poor quality, the average of the signatures tends toward $\mathbf{b}_0$. With medium-quality templates, achieving an 80% success rate in labeling, an attacker can approach $\mathbf{b}_0$ with around 17 500 signatures generated. This is depicted by the purple curve with squares. However, the number of traces required for the attack may become prohibitive as template quality decreases.

6.3 Influence of the Threshold

In Sect. 5.4, we studied the template construction in the context of $t = 1$. We computed the theoretical success rate but did not consider the continuous Gaussian noise. Moreover, we suggested that the described bias can be enhanced by the choice of a threshold $l \in \mathbb{R}^+$. Therefore we present experimental results here, considering all factors. We obtain Tab. 2 for varying values of l, considering the leakage on the i^{th} coordinate. The second line corresponds to the proportion of signatures that are kept for the templates' construction, i.e., such that the global offset $|y_i| > l$. The third line (resp. fourth line) corresponds to the average proportion of positive (resp. negative) first share when $|y_i| > l$. Each result is expressed as a percentage and represents an average of ten thousand tests conducted with various centers and standard deviations. Note that in this case, these two parameters do not seem to have an impact on the result.

We notice a gap between the classification of the negative and positive cases. This is explained by a small asymmetry when the last iteration of the SamplerC takes as input the bit $c' = 0$ and when the base sampler, computing $D_{c'+2\mathbb{Z},r_0}$, outputs 0. In this situation, the offset produced by g_1 is null. The Hamming weight in this context corresponds to a positive offset. Therefore, this neutral

Table 2. Influence of l in labeling during the building phase.

Value of $l \in \mathbb{R}^+$	0	1	2	3	4	5
Proportion of selected traces	100	72.6	48.3	29.6	16.4	8.0
Proportion of correctly labeled positive offset	65.8	68.0	70.1	74.7	76.8	78.6
Proportion of correctly labeled negative offset	53.2	56.5	60.3	62.5	65.6	68.5

offset is classified as positive, making this class a little more represented. Indeed, as an attacker knows only the total offset during the building phase, the number of traces labeled positively is fixed. Thus, adding more representatives of this class increases the percentage of success. Note that an attacker could use this fact to construct a better-quality template for the positive offset. She could then use it to label as positive the 50% of traces with the highest similarity score and label the rest as negative. Alternatively, she could achieve a satisfactory classification of the positive cases and only retain traces labeled $\{+,+\}$ in the matching phase. This doubles the number of traces to generate, but it can be worthy if the quality of both templates is unbalanced.

These results and remarks ensure that a trade-off can be obtained to guarantee that the construction of the templates is carried out on a biased set.

7 Countermeasures

To protect Mitaka against these attacks, several countermeasures can be considered. In [21], the authors suggested reducing the signal-to-noise ratio (SNR) thanks to two implementation tips. The first one aims at limiting the variation of the Hamming weight. For example, one can encode the sign on $\{1, 2\}$ instead of $\{0, 1\}$. The second approach involves computing all possible scenarios during execution. For instance, one can calculate all computations for both positive and negative signs and then select results consistent with what was drawn. This avoids signal differences that depend on sensitive choices. Their results show that these countermeasures do not completely prevent the leakage, but by reducing the SNR, they increase the number of traces required for the attack.

However, these protections do not take into account a stronger attack, such as the one presented in Sect. 4.2. An interesting countermeasure could be to shuffle the loop that calls the SamplerZ. Indeed, the order does not impact the sampling because the same operation is computed independently on each of the 512 coefficients. We suggest implementing this shuffling similarly to a Random Start Index (RSI), using a unique random XOR with the list of indices. This makes the countermeasure almost free in terms of computation.

This countermeasure forces the attacker to attack "on average." During the construction of the templates, they can no longer assign the calculated sign for y_i to the corresponding segment in the trace. Their best approach is to focus on traces where the number of positive or negative signs is significantly greater than the expected value. As a result, the probability of each sample having this sign increases on average. This methodology is much less effective because, when combined with other biases, the correct classification rate for training the template becomes very low. Additionally, this prevents the use of a threshold l (see Sect. 5.4) that could have strengthened this bias. Finally, the Bienaymé-Chebyshev inequality shows that the required number of traces to find one that deviates sufficiently from the mean to be exploitable quickly becomes very large. Indeed, the sign distribution follows a uniform probability law of 1/2, so an attacker wishing to keep traces with 70% positive or negative signs will only retain about 1% of the generated traces. In this case, they will achieve a bias in each sampling of at most 56% (see Tab. 2).

An easy and inexpensive countermeasure that could complement the shuffling would be to implement the code around SamplerZ in constant time so that an accepted offset is no longer distinguishable from a rejected one in side-channel observations. In this scenario, an attacker would have to guess the accepted sampling from the rejected ones based only on the expected value of the rejection. As the difficulty of predicting the exact position of the accepted sampling depends on the targeted vector, a trade-off can be made between the number of targeted vectors (ranging from 1 to 512) and the accuracy in guessing the accepted sampling. This leads to an increase in the number of traces and noise. The exact acceptance ratio of the SamplerZ depends on the parameters, but we can roughly approximate it as 2, to give a concrete idea of the countermeasure. Thus, the noise is also multiplied by the same factor.

8 Conclusion

Falcon, currently drafted by NIST, could present challenges for embedded systems due to its complexity, especially regarding memory usage. In contrast, Mitaka offers a relevant alternative, sharing the same underlying paradigm as Falcon but with a simpler design. Recent advancements, especially the ANTRAG trapdoor generation discussed in [10], confirm Mitaka's potential. However, although initially thought to be more resilient to side-channel attacks, our study reveals vulnerabilities in its structure.

In this article, we significantly enhanced a previously known attack by tailoring it to Mitaka's specifications, resulting in a drastic reduction in the required number of traces. Moreover, we analysed the suggested masking scheme for Mitaka and exposed a novel side-channel weakness. We leveraged this vulnerability using a similar approach to the one used in the unprotected settings, targeting a first-order masked implementation of Mitaka and successfully recovering the key despite the protection. Finally, we suggest countermeasures to prevent this attack. Rather inexpensive, theses protections can be easily implemented.

Nevertheless, in a broader context, the advancements outlined in this article provide a framework for attacks targeting Mitaka's Gaussian sampler, as they expose a significant vulnerability in the functioning of the Hybrid Sampler. These insights could potentially be leveraged to exploit other side-channel leakages, posing a significant challenge to the practical security of Mitaka.

Acknowledgments. We thank the reviewers for their insightful comments.

References

1. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P., Grégoire, B., Strub, P., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: ACM CCS. pp. 116–129 (2016)
2. Bauer, S., Santis, F.D.: A differential fault attack against deterministic falcon signatures. IACR Cryptol. ePrint Arch. 422 (2023)
3. Bernstein, D., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The sphincs$^+$ signature framework. In: ACM CCS. pp. 2129–2146 (2019)
4. Chari, S., Rao, J., Rohatgi, P.: Template attacks. In: CHES. LNCS, vol. 2523, pp. 13–28. Springer (2002)
5. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. TCHES **2018**(1), 238–268 (2018)
6. Ducas, L., Prest, T.: A hybrid gaussian sampler for lattices over rings. IACR Cryptol. ePrint Arch. 660 (2015)
7. Ducas, L., Prest, T.: Fast fourier orthogonalization. In: ACM ISSAC. pp. 191–198 (2016)
8. Espitau, T.: Supporting code for Mitaka signature (2022), https://github.com/espitau/Mitaka-EC22/tree/main
9. Espitau, T., Fouque, P., Gérard, F., Rossi, M., Takahashi, A., Tibouchi, M., Wallet, A., Yu, Y.: Mitaka: A simpler, parallelizable, maskable variant of falcon. In: EUROCRYPT. LNCS, vol. 13277, pp. 222–253. Springer (2022)
10. Espitau, T., Nguyen, T.T.Q., Sun, C., Tibouchi, M., Wallet, A.: Antrag: annular ntru trapdoor generation. Cryptology ePrint Archive (2023)
11. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-fourier lattice-based compact signatures over NTRU, specification v1. 2. NIST Post-Quantum Cryptography Standardization Round 3 (2020)
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: SToC. pp. 197–206. ACM (2008)
13. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: CRYPTO. LNCS, vol. 1294, pp. 112–131. Springer (1997)
14. Guerreau, M., Martinelli, A., Ricosset, T., Rossi, M.: The hidden parallelepiped is back again: Power analysis attacks on falcon. TCHES **2022**(3), 141–164 (2022)
15. Karabulut, E., Aysu, A.: FALCON down: Breaking FALCON post-quantum signature scheme through side-channel attacks. In: ACM IEEE Design Automation Conference. pp. 691–696 (2021)
16. McCarthy, S., Howe, J., Smyth, N., Brannigan, S., O'Neill, M.: BEARZ attack FALCON: implementation attacks with countermeasures on the FALCON signature scheme. In: SECRYPT. pp. 61–71. SciTePress (2019)
17. Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: CRYPTO. LNCS, vol. 10402, pp. 455–485. Springer (2017)
18. Peikert, C.: An efficient and parallel gaussian sampler for lattices. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO. Lecture Notes in Computer Science, vol. 6223, pp. 80–97. Springer (2010)
19. Prest, T.: Gaussian Sampling in Lattice-Based Cryptography. Ph.D. thesis, École Normale Supérieure, Paris, France (2015), https://tel.archives-ouvertes.fr/tel-01245066

20. Prest, T.: A key-recovery attack against mitaka in the t-probing model. In: PKC. LNCS, vol. 13940, pp. 205–220. Springer (2023)
21. Zhang, S., Lin, X., Yu, Y., Wang, W.: Improved power analysis attacks on falcon. In: EUROCRYPT. LNCS, vol. 14007, pp. 565–595. Springer (2023)