

Swing the Lure: How to Cheaply Mitigate Sign Leakage in Falcon

Xiuhan Lin¹, Mehdi Tibouchi², and Yang Yu^{1,3,4}

¹ Institute for Advanced Study, Tsinghua University, Beijing, China
lin-xiuhan@mail.tsinghua.edu.cn

yu-yang@mail.tsinghua.edu.cn

² NTT Social Informatics Laboratories, Tokyo, Japan
mehdi.tibouchi@ntt.com

³ Zhongguancun Laboratory, Beijing, China

⁴ State Key Laboratory of Cryptography and Digital Economy Security, China

Abstract. Falcon is one of the three post-quantum signature schemes selected by NIST for standardization so far. It is efficient and has particularly short signatures, but has the drawback of being difficult to implement correctly and securely, owing to its use of floating point arithmetic and secret-dependent lattice Gaussian sampling. In particular, obtaining implementations of Falcon that are protected against side-channel attacks seems to be a tremendous challenge that has no satisfactory solution so far. While it is theoretically possible to rely on generic countermeasures like masking, they are likely so costly as to be effectively inapplicable.

And yet, side-channels are an actual threat to Falcon: a number of them have been demonstrated against various parts of the signing algorithm, including floating point operations and Gaussian sampling. Multiple power analysis attacks in fact achieve full key recovery from very realistic numbers of traces. This is especially the case for a series of side-channel attacks that can be broadly categorized as based on “sign leakage” within a specific part of the one-dimensional Gaussian sampler.

In this paper, we take a closer look at this class of side-channel attacks, and obtain yet another one of a somewhat different flavor: while it also targets the sign of a variable in the one-dimensional Gaussian sampler, that variable is the input center, and in particular does not depend on the sampling randomness, making previously proposed mitigations modifying the base sampler ineffective.

Furthermore, we propose, analyze and experimentally validate a novel, inexpensive countermeasure to protect Falcon against this class of side channel attacks. The idea, taking inspiration from shuffling, is to twist the target vector by random roots of unity in every recursive call of the fast Fourier sampling algorithm: this averages away the sign leakage efficiently. While this mitigation technique is heuristic and does not offer the provable guarantees of masking, it significantly enhances the physical security of Falcon fairly cheaply. In particular, it provides a more robust and more widely applicable protection than the countermeasure of Lin

et al. (PKC 2025) at a much lower cost (incurring a 1.4-fold overhead in signing in dynamic mode and 2-fold in tree mode, compared to 3.5-fold and 6.8-fold for Lin et al. respectively). To the best of our knowledge, this is the first shuffling countermeasure proposed for Falcon.

Keywords: Falcon Signature Scheme · Lattice-Based Cryptography · Side-Channel Analysis · Side-Channel Countermeasures · Implementation Security

1 Introduction

Falcon [PFH⁺22], is one of the three post-quantum signatures selected by NIST for standardization in 2022; it will be standardized under the name FN-DSA. It is an efficient instantiation of the GPV hash-and-sign framework [GPV08] over NTRU lattices. It is the most compact scheme (both in terms of signature size alone and combined public key and signature size) among three NIST selected signature schemes, and has high efficiency for both signing and verification, which makes it a particularly appealing design. However, its signing procedure essentially performs lattice Gaussian sampling using a complex algorithm that relies on floating-point arithmetic. Moreover, it is crucial for security that the resulting Gaussian distribution is (statistically close to) the ideal one. This makes Falcon particularly challenging to implement correctly and securely, especially when accounting for the threat of side-channel attacks.

Generally speaking, side-channel attacks are regarded as one of the most serious threats against real-world deployed cryptosystems. With the post-quantum migration underway, the side-channel security of post-quantum schemes have been receiving increasing attention. Compared to other post-quantum standards, Falcon has a much more contrived algorithm, which complicates both side-channel analysis and protections. NIST has repeatedly emphasized the importance of investigating the implementation security of Falcon and encouraged in-depth analyses of it in the status reports of the PQC standardization process. Driven by this, a number of works have explored both attacks and defenses for Falcon implementations.

Side-channel attacks against Falcon. Fouque et al. [FKT⁺20] initiated the side-channel analysis of hash-and-sign lattice-based signature. They presented a practical timing attack against the DLP signature scheme [DLP14], which is the predecessor of Falcon, and a theoretical timing attack against the first round implementation of Falcon. The identified leakage was later sealed by the use of Howe et al.’s isochronous Gaussian sampler [HPRR20].

Since then, Falcon has been the target of various other side-channel attacks. Most of these attacks exploit one of two secret-dependent operations in the signing procedure: either the pre-image computation or the integer Gaussian sampling. Regarding the pre-image computation, Karabulut and Aysu [KA21] proposed the first differential electromagnetic attack, which targets the floating-

point multiplications within the pre-image computation, and was further improved by Guerreau et al. in [GMRR22].

More side-channel attacks focused on the integer Gaussian sampler. Guerreau et al. [GMRR22] introduced the first power analysis attack by exploiting the leakage, i.e., half Gaussian leakage, from the base sampler. Zhang et al. [ZLYW23] further improved their key recovery and identified new power leakage, i.e., sign leakage of the integer Gaussian samples and modeled the key recovery as *Learning Halfspace Problem*. Subsequently, Lin et al. [LZY⁺25] substantially reduced the number of traces for key recovery by using the symplecticity of NTRU lattices. More recently, Brinkmann et al. [BKM26] gave a new power analysis attack by using the sign leakage from the output of Falcon integer Gaussian samplers, i.e., random Gaussian value plus floored center, and more efficient halfspace learning algorithms.

Recent papers further demonstrate that the Falcon implementation remains vulnerable to diverse other leakage sources, including the in-place modular reduction within the number theoretic transform [QA26] and the bit-mask based constant-time conditional assignments that are present in floating-point conversion and floating-point addition [KLL⁺26]. Moreover, Falcon has also been the target of other implementation attacks, include fault analysis [MHS⁺19, BDS23, APT25] and cold-boot attacks [KTA26].

Side-channel protections for Falcon. While side-channel attacks on Falcon abound, countermeasures have yet to catch up. A major advance was the isochronous Gaussian sampler of Howe et al. [HPRR20], which addressed the timing vulnerability within the integer Gaussian sampler and made constant-time implementations feasible. However, only a handful of works have addressed countermeasures against other side-channel attacks.

Some provable countermeasures only focus on the specific operations. Chen [CC24] proposed masked gadgets for floating-point multiplication and addition. Berthet et al. [BPT⁺24] proposed a masked floor function. Then, Karabulut and Aysu [KA24] presented a masked hardware implementation of floating-point multiplication only. However, it was (not unexpectedly) show by Chen et al. [CCCY26] that masking floating-point multiplications alone is insufficient, and power analysis attacks remain, e.g., on the pre-image computation when masking floating-point multiplications but not the additions (which are costlier to mask).

For the Falcon Gaussian sampler, there is no provable countermeasures have been proposed so far. However, some of the attack papers have suggested heuristic mitigations against the leakage sources they uncovered: earlier ones [GMRR22, ZLYW23] proposed simple tweaks later shown to be defeated by classifiers trained on a small number of traces [LZY⁺25, §6.1]. Later on, Lin et al. [LZY⁺25] proposed a much more complex countermeasure that was shown to achieve significant reduction in classification accuracy for the target variables, but it still didn't eliminate the leakage entirely, and it was quite costly, amounting to a roughly 19-fold increase in the cost of integer Gaussian sampling (and 4- to 7-fold for the entire signing algorithm).

Despite some efforts, no masked implementation of the Falcon Gaussian sampler has been proposed so far. It is of course possible in principle to apply a naive, generic masking countermeasure to the entire algorithm, but this is expected to be tremendously costly due to the mixture of integer, floating point and bitwise operations, various rejection sampling steps, etc. Espitau et al. [EFG⁺22] proposed Mitaka as a simpler variant of Falcon and described a masking of it at any order. However, Prest [Pre23] uncovered a flaw in the masking security proof, and described an actual attack in the probing model. In addition, some masked integer Gaussian samplers have been designed [GR19, BBE⁺19, YMZ⁺26, AEHT26a, AEHT26b], but these proposals are only applicable to Gaussian samplers with fixed parameters. Unfortunately, the one-dimensional discrete Gaussians in Falcon’s signing algorithm are sampled with variable, secret-dependent centers and standard deviations. This is the main challenge that needs to be overcome to mask the integer Gaussian sampler efficiently, and it remains open so far.

1.1 Our Contributions

This paper further investigates the side-channel security of Falcon from the perspectives of both attack and defense.

On the cryptanalysis side, we identify new source of side-channel leakage in the centers of Falcon integer Gaussian samplers. In contrast with previous attacks [GMRR22, ZLYW23, LZY⁺25, BKM26], our attack is independent of the randomness of sampling, and in particular, it cannot be mitigated by the countermeasures suggested in [GMRR22, ZLYW23, LZY⁺25] (which is also the case, incidentally, for the attack of [BKM26]). We then mount a practical key recovery attack using this new leakage. By using more than 10,000 signatures, one can effectively recover the Falcon signing key with good probability.

On the protection side, we then propose a very simple countermeasure against not only the leakage we consider, but those of previous attacks as well. The main idea is to disrupt the correlations between leaks and the secret key by re-ordering the sampling basis, in a spirit similar to shuffling [RPD09, VMKS12]. Our initial countermeasure implements the basis re-ordering by multiplying by a single root of unity. However, we discover that a second order attack can still recover the hidden exponent of the monomial and then invalidate this initial countermeasure.

We further propose an efficient and effective countermeasure by adapting our initial design to the recursive algorithmic structure of Falcon lattice Gaussian sampler. This properly introduces more roots of unit to recursive calls at different levels, which defeats the hidden exponent recovery. In addition, our countermeasure reuses the precomputed Falcon tree, and has a relatively modest overhead: compared with the reference implementation, our countermeasure incurs an overhead of around $1.4\times$ on signing speed in dynamic mode, and $2\times$ in tree mode. We validate our countermeasure by leakage assessment on ChipWhisperer-Lite platform and find no exploitable leakage after 20,000 traces. To the best of our knowledge, this is the first shuffling protection against Falcon.

2 Preliminaries

Bold lowercase letters denote row vectors, i.e., $\mathbf{b} = (b_0, \dots, b_{n-1})$ and bold uppercase letters denote matrices, i.e., $\mathbf{B} = (\mathbf{b}_0, \dots, \mathbf{b}_{n-1})$. For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, their inner product is $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} a_i b_i$. The ℓ_2 -norm of $\mathbf{a}$ is $\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle}$ and the ℓ_1 -norm is $\|\mathbf{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$. The inverse and conjugate transpose of matrix $\mathbf{B}$ are denoted by $\mathbf{B}^{-1}$ and $\mathbf{B}^*$ respectively. For $x \in \mathbb{R}$, we write $\lfloor x \rfloor$ for the nearest integer to x and $\lfloor x \rfloor$ for the largest integer not exceeding x .

For a distribution D , $y \leftarrow D$ denotes sampling y from D , and $y \sim D$ denotes the random variable y distributed over D . For $y \sim D$, let $D(z) = \Pr[y = z]$ be the probability mass at z .

Lattices. A lattice $\mathcal{L}$ is the set of all integer combinations of linearly independent vectors $\mathbf{b}_0, \dots, \mathbf{b}_{n-1} \in \mathbb{R}^m$, i.e. $\mathcal{L} = \{\sum_{i=0}^{n-1} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$. We call n the rank and $\mathbf{B} = (\mathbf{b}_0, \dots, \mathbf{b}_{n-1})$ a basis of $\mathcal{L}$, and write $\mathcal{L} = \mathcal{L}(\mathbf{B})$. A lattice is full-rank when $n = m$.

Let $\mathbf{B} \in \mathbb{R}^{n \times m}$ be a matrix of full row rank. Its Gram-Schmidt orthogonalization (GSO) is $\mathbf{B} = \mathbf{L}\tilde{\mathbf{B}}$, where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is unit lower triangular and the row vector $\tilde{\mathbf{b}}_i$ of $\tilde{\mathbf{B}}$ are pairwise orthogonal so that $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^*$ is diagonal. The Gram matrix $\mathbf{G} = \mathbf{B}\mathbf{B}^*$ admits the decomposition $\mathbf{G} = \mathbf{L}(\tilde{\mathbf{B}}\tilde{\mathbf{B}}^*)\mathbf{L}^*$, which corresponds to the LDL* decomposition of $\mathbf{G}$ with $\mathbf{D} = \tilde{\mathbf{B}}\tilde{\mathbf{B}}^*$.

Gaussians. Let $\rho_{\sigma, \mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}\right)$ be the Gaussian function with standard deviation $\sigma > 0$ and center $\mathbf{c} \in \mathbb{R}^n$. We omit the center when $\mathbf{c} = \mathbf{0}$. For a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and $\mathbf{c} \in \text{span}(\mathcal{L})$, the discrete Gaussian $D_{\mathcal{L}, \sigma, \mathbf{c}}$ is defined by

$$D_{\mathcal{L}, \sigma, \mathbf{c}}(\mathbf{u}) = \frac{\rho_{\sigma, \mathbf{c}}(\mathbf{u})}{\sum_{\mathbf{v} \in \mathcal{L}} \rho_{\sigma, \mathbf{c}}(\mathbf{v})}$$

for any $\mathbf{u} \in \mathcal{L}$. It derives the Gaussian $D_{\mathcal{L} + \mathbf{c}, \sigma} = \mathbf{c} + D_{\mathcal{L}, \sigma, -\mathbf{c}}$ over a coset $\mathcal{L} + \mathbf{c}$. The special case $\mathcal{L} = \mathbb{Z}$ yields the integer Gaussian distribution $D_{\mathbb{Z}, \sigma, \mathbf{c}}$. The integer half Gaussian over $\mathbb{Z}^+$ is $D_{\mathbb{Z}, \sigma}^+(u) = \frac{\rho_{\sigma}(u)}{\sum_{v \in \mathbb{N}} \rho_{\sigma}(v)}$.

NTRU. Let n be a power-of-two and $\phi = x^n + 1$ be the $2n$ -th cyclotomic polynomial. Let $\mathcal{R} = \mathbb{Z}[x]/(\phi)$ and $\mathcal{K} = \mathbb{Q}[x]/(\phi)$. Give $f = \sum_{i=0}^{n-1} f_i x^i \in \mathcal{K}$, we denote the Hermitian adjoint of f by $f^* = f_0 - \sum_{i=1}^{n-1} f_i x^{n-i}$. In a typical NTRU scheme, the secret key is two small polynomials $f, g \in \mathcal{R}$ with f invertible modulo q , and the public key is $h = g/f \bmod q$. The NTRU lattice defined by h is $\mathcal{L}_{\text{NTRU}} = \{(u, v) \in \mathcal{R}^2 : u + vh = 0 \bmod q\}$. A natural basis of $\mathcal{L}_{\text{NTRU}}$ induced by (f, g) , also known as NTRU trapdoor, is $\mathbf{B}_{f, g} = \begin{pmatrix} g & -f \\ F & -F \end{pmatrix}$ where $(F, G) \in \mathcal{R}^2$ such that $fG - gF = q$.

Algorithm 1: Sign

Input: A message msg , secret basis $\mathbf{B}_{f,g}$ and acceptance bound $\lfloor \beta^2 \rfloor$
Output: A valid signature sig of msg

```

1  $r \xleftarrow{\$} \{0,1\}^{320}$ ,  $c \leftarrow H(r \parallel \text{msg})$ 
2  $\mathbf{t} \leftarrow (\text{FFT}(c), \text{FFT}(0)) \cdot \text{FFT}(\mathbf{B}_{f,g})^{-1}$   $\triangleright$  pre-image computation
3 do
4   do
5      $\mathbf{z} \leftarrow \text{ffSampling}_n(\mathbf{t}, \mathbf{T})$   $\triangleright$  trapdoor sampling
6      $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{z}) \cdot \text{FFT}(\mathbf{B}_{f,g})$   $\triangleright \mathbf{s} \sim D_{(c,0)+\mathcal{L}(\mathbf{B}), \sigma_{\text{sig}}, 0}$ 
7     while  $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$ 
8      $(s_0, s_1) \leftarrow \text{invFFT}(\mathbf{s})$   $\triangleright \mathbf{s}$  is in FFT
9   while  $s = \perp$ 
10 return  $\text{sig} = (r, s)$ 
```

Falcon signature scheme. Falcon [PFH⁺22] is an NTRU-based instantiation of GPV hash-and-sign framework [GPV08]. It uses an NTRU trapdoor $\mathbf{B}_{f,g}$ as the signing key and $h = g/f \bmod q$ as the verification key. Its signing procedure (Algorithm 1) boils down to sampling $\mathbf{s} = (s_0, s_1) \sim D_{(c,0)+\mathcal{L}(\mathbf{B}), \sigma_{\text{sig}}}$ where $c = H(r \parallel \text{msg})$ is the hash of a random salt r and the message msg . Signature verification checks that $s_0 + s_1 h = c \bmod q$ and $\|\mathbf{s}\| \leq \beta$ with $\beta = 1.1 \cdot \sigma_{\text{sig}} \sqrt{2n}$.

Falcon implements lattice Gaussian sampling with the fast Fourier sampler [DP16] (Algorithm 2). The sampler uses a precomputed Falcon tree $\mathbf{T}$ that encodes the GSO, equivalently the LDL^* , of $\mathbf{B}_{f,g}$. It decomposes lattice Gaussian sampling into a sequence of integer Gaussian samplings (Algorithm 3) with varying centers and standard deviations.

3 Side-Channel Attacks on the Falcon Sampler

Integer Gaussian sampling is the central building block of Falcon’s signing procedure, ensuring that signatures are provably statistically independent of the secret key. However, such samplers are a notorious target for side-channel attacks. While the Falcon sampler adopts an isochronous design [HPRR20], it is still subject to power leakage that enables practical key recovery [GMRR22, ZLYW23, LZY⁺25, BKM26]. In this section, we present a new side-channel attack on the Falcon sampler, which first targets at the Gaussian centers.

3.1 Known Types of Power Leakage

Prior to the presentation of our new side-channel attack, we briefly recall three types of leakage shown in existing works [GMRR22, ZLYW23, BKM26].

Half Gaussian leakage. The half Gaussian leakage reveals whether the half-Gaussian sample $y_+ = 0$ or not. It was first identified by Guerreau et al. [GMRR22]

Algorithm 2: ffSampling_n

Input: A target vector $\mathbf{t} = (t_0, t_1) \in \text{FFT}(\mathcal{K})^2$, a Falcon tree T
Output: $\mathbf{z} = (z_0, z_1) \in \text{FFT}(\mathcal{R})^2$

```

1 if  $n = 1$  then
2    $\sigma \leftarrow T.\text{value}$   $\triangleright \sigma \in [\sigma_{\min}, \sigma_{\max}]$ 
3    $z_0 \leftarrow \text{SamplerZ}(t_0, \sigma)$ 
4    $z_1 \leftarrow \text{SamplerZ}(t_1, \sigma)$   $\triangleright t_0$  and  $t_1$  are Gaussian centers
5   return  $\mathbf{z} = (z_0, z_1)$ 
6 end if
7  $(L, T_0, T_1) \leftarrow (T.\text{value}, T.\text{leftchild}, T.\text{rightchild})$ 
8  $t_1 \leftarrow \text{split\_fft}(t_1)$ 
9  $\mathbf{z}_1 \leftarrow \text{ffSampling}_{n/2}(t_1, T_1)$   $\triangleright$  First recursive call
10  $z_1 \leftarrow \text{merge\_fft}(\mathbf{z}_1)$ 
11  $t'_0 \leftarrow t_0 + (t_1 - z_1) \odot L$ 
12  $t_0 \leftarrow \text{split\_fft}(t'_0)$ 
13  $\mathbf{z}_0 \leftarrow \text{ffSampling}_{n/2}(t_0, T_0)$   $\triangleright$  Second recursive call
14  $z_0 \leftarrow \text{merge\_fft}(\mathbf{z}_0)$ 
15 return  $\mathbf{z} = (z_0, z_1)$ 

```

in Falcon’s reference implementation. Lin et al. [LZY⁺25] later conducted a thorough review on the source of this leakage.

Sign leakage of $y \sim -\lfloor \mu \rfloor + D_{\mathbb{Z}, \sigma, \mu}$. This leakage reveals the sign of the intermediate value y in `SamplerZ`. It was first identified and exploited in [ZLYW23], and its source was later closely examined in [LZY⁺25].

Sign leakage of $z \sim D_{\mathbb{Z}, \sigma, \mu}$. This leakage reveals the sign of the output z of `SamplerZ`. Brinkmann, Kraus and May [BKM26] recently proposed a key recovery attack using this leakage, but they did not provide experimental evidence for the source of this leakage. Our subsequent experimental results (Fig. 2) validate the existence of this leakage.

3.2 New Leakage of Gaussian Centers

We demonstrate a new power side-channel leakage in Falcon integer Gaussian sampler, which reveals the sign of the integer Gaussian centers. In contrast to previously identified leakage sources (Section 3.1), the new leakage does not involve the randomness in `SamplerZ`.

Source of the new leakage. The sign leakage of the center μ mainly stems from the operations in the Line 1 of Algorithm 3, and the corresponding code fragment is highlighted by in Listing 1. The decomposition of μ comprises the operations including the floor subroutine `fpr_floor`, explicit type cast (`int`), floating-point conversion `fpr_of` and the subtraction routine `fpr_sub`. In particular,

Algorithm 3: SamplerZ

Input: A center μ and standard deviation $\sigma \in [\sigma_{\min}, \sigma_{\max}]$

Output: An integer $z \in \mathbb{Z}$ such that $z \sim D_{\mathbb{Z}, \sigma, \mu}$

```

1  $r \leftarrow \mu - \lfloor \mu \rfloor$ 
2  $y_+ \leftarrow D_{\mathbb{Z}, \sigma_{\max}}^+$  ▷ via BaseSampler
3  $b \xleftarrow{\$} \{0, 1\}$ 
4  $y \leftarrow b + (2b - 1)y_+$ 
5  $x \leftarrow \frac{(y-r)^2}{2\sigma^2} - \frac{y_+^2}{2\sigma_{\max}^2}$ 
6 return  $z \leftarrow y + \lfloor \mu \rfloor$  with probability  $\frac{\sigma_{\min}}{\sigma} \cdot \exp(-x)$ , otherwise restart.
```

this kind of sign leakage is largely amplified in the routine `fpr_floor` and `fpr_of` due to the binary two's complement. A recurring pattern in these routines is the branchless two's complement conditional negation `x = (x ^ -(int64_t)t) + t`, where `t` $\in \{0, 1\}$ denotes the sign bit derived from the Gaussian center μ . The cast `-(int64_t)t` expands the 1-bit value into either `0x0000000000000000` or `0xFFFFFFFFFFFFFFFF`, which yields a maximum Hamming distance of 64 between the positive and negative cases. Thus, this very leaky operation can lead to large Hamming distance for the following intermediate values, which becomes observable by side-channel measurement.

```

1 int
2 Zf(sampler)(void *ctx, fpr mu, fpr isigma)
3 {
4     ...
5     int s;
6     fpr r, dss, ccs;
7     ...
8     s = (int)fpr_floor(mu);
9     r = fpr_sub(mu, fpr_of(s));
10    ...
11 }
```

Listing 1: The code fragment of SamplerZ.

Experimental validation. We practically evaluate the reference implementation of Falcon on the ChipWhisperer-Lite equipped with an STM32F415 UFO target board (ARM Cortex-M4), targeting at the specific operations shown in Listing 1. Power traces of the execution of **SamplerZ** are captured by PicoScope 3206D oscilloscope at a real-time sampling rate of 1GSa/s. To improve the Signal-

to-Noise Ratio (SNR), we use a Mini-Circuits 1.9 MHz low-pass filter to suppress high-frequency noise.

Fig. 1 exhibits the total SNR curve of **SamplerZ** against sign leakage of the input center μ , measured over 100,000 collected traces. The specified interval of SNR curve, marked by red dashed line, contains the most prominent peak, which corresponds to Line 8-9 of Listing 1. Furthermore, we perform Gaussian template attacks [CRR03] to validate the practical exploitability of this leakage. By using a handful of profiling traces, we achieve 100% classification accuracy over 10,000 traces.

3.3 Key Recovery from the New Leakage

We can use the sign leakage of integer Gaussian centers to mount a key recovery attack (Algorithm 4) on Falcon. The main idea is to reduce the key recovery to solving *Learning Halfspace Problem* in the noise-free case, which is similar to previous attacks [ZLYW23, BKM26] using different sources of leakage.

Definition 1 (Learning Halfspace Problem (LHP)). Give $\mathbf{b} \in \mathbb{R}^n$. A LHP oracle returns the sample pairs $(\mathbf{w}_i, l_i) \in \mathbb{R}^n \times \{-1, +1\}$, where $0 \leq i < m$, $\mathbf{w}_i$ is drawn from a fixed distribution and

$$l_i = \begin{cases} \text{sign}(\langle \mathbf{w}_i, \mathbf{b} \rangle), & \text{with probability } 1 - p, \\ -\text{sign}(\langle \mathbf{w}_i, \mathbf{b} \rangle), & \text{with probability } p, \end{cases}$$

for some noise flipping rate $p \in [0, 1)$. Give m independent LHP-sample pairs, the goal is to find the approximate direction of the secret vector $\mathbf{b}$.

The special case $p = 0$ is referred to as the noise-free LHP, whereas $p > 0$ corresponds to the noisy LHP. For noise-free LHP, we can equivalently recover $\mathbf{b}$ using the following sign constraints:

$$\begin{cases} \langle \mathbf{w}_i, \mathbf{b} \rangle \geq 0 & \text{if } l_i = +1, \\ \langle \mathbf{w}_i, \mathbf{b} \rangle < 0 & \text{if } l_i = -1. \end{cases}$$

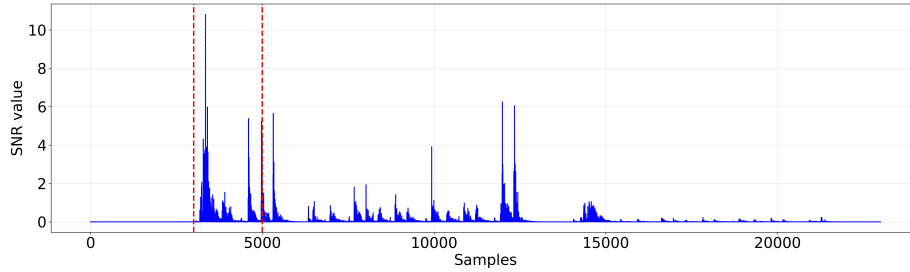


Fig. 1: The total SNR curve of **SamplerZ** for sign leakage of center μ . The red dashed line highlight the interval of Line 8-9 of Listing 1.

This linear system can be solved by linear programming (LP). In this work, we only consider the noise-free case due to the perfect classification accuracy for the new leakage as shown in Section 3.2. For noisy case, one can apply other LHP algorithms presented in [BKM26].

Constructing LHP samples. A key step of our key recovery is to construct LHP samples using the sign of Gaussian centers. While `ffSampling` (Algorithm 2) proceeds in a recursive way, it preserves the algorithmic structure of the Klein-GPV sampler [GPV08] — reducing lattice Gaussian sampling to a series of integer Gaussian samplings centered at the projection of the target on the GSO vectors. Therefore, the center μ_i has the following form:

$$\mu_i = \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\|^2}$$

where $\mathbf{c}_i$ is the updated target and $\tilde{\mathbf{b}}_i$ is the i -th GSO vector of the Falcon trapdoor basis.

Our key recovery only uses Gaussian centers in the first two calls of `SamplerZ`, denoted by $\mathbf{c}_{(0)}$ and $\mathbf{c}_{(1)}$, as the corresponding first two GSO vectors are orthogonal in the FFO basis, implying $\mathbf{c}_{(0)} = \mathbf{c}_{(1)} = \mathbf{c} = (c, 0)$ with $c = H(r\|\mathbf{msg}) \in \mathbb{R}_q$, while subsequent centers vary as per former integer Gaussian samples. Following the order of Falcon tree traversal in `ffSampling`, we have

$$\mu_{(0)} = \mu_{3n/2-1} = \frac{\langle \mathbf{c}, \tilde{\mathbf{b}}_{3n/2-1} \rangle}{\|\tilde{\mathbf{b}}_{3n/2-1}\|^2} \quad \text{and} \quad \mu_{(1)} = \mu_{2n-1} = \frac{\langle \mathbf{c}, \tilde{\mathbf{b}}_{2n-1} \rangle}{\|\tilde{\mathbf{b}}_{2n-1}\|^2}.$$

Using the symplecticity of the NTRU module as in [LZY⁺25] gives

$$\frac{\tilde{\mathbf{b}}_{3n/2-1}}{\|\tilde{\mathbf{b}}_{3n/2-1}\|} = -\frac{\tilde{\mathbf{b}}_0}{\|\tilde{\mathbf{b}}_0\|} \cdot \mathbf{P}(\mathbf{J}\mathbf{D})^{-1} \quad \text{and} \quad \frac{\tilde{\mathbf{b}}_{2n-1}}{\|\tilde{\mathbf{b}}_{2n-1}\|} = \frac{\tilde{\mathbf{b}}_0}{\|\tilde{\mathbf{b}}_0\|} \cdot (\mathbf{J}\mathbf{D})^{-1},$$

where $\mathbf{P} = \begin{pmatrix} & -\mathbf{I}_{n/2} \\ \mathbf{I}_{n/2} & \\ & -\mathbf{I}_{n/2} \\ & & \mathbf{I}_{n/2} \end{pmatrix}$, $\mathbf{J}$ is a $2n \times 2n$ reversed identity matrix,

$\mathbf{D} = \begin{pmatrix} -\mathbf{I}_n & \\ & \mathbf{I}_n \end{pmatrix}$ and $\tilde{\mathbf{b}}_0 = \mathbf{b}_0 = (g, -f)$. Therefore, $\mu_{(0)}$ and $\mu_{(1)}$ have the same sign with $\langle -\mathbf{c} \cdot \mathbf{P}(\mathbf{J}\mathbf{D})^{-1}, \mathbf{b}_0 \rangle$ and $\langle \mathbf{c} \cdot (\mathbf{J}\mathbf{D})^{-1}, \mathbf{b}_0 \rangle$ respectively. Furthermore, let $(c'_0, c'_1) = -\mathbf{c} \cdot \mathbf{P}(\mathbf{J}\mathbf{D})^{-1}$ and $(c''_0, c''_1) = \mathbf{c} \cdot (\mathbf{J}\mathbf{D})^{-1}$, then $c'_0 = c''_0 = 0$. Therefore, $\mu_{(0)}$ and $\mu_{(1)}$ have the same sign with $\langle \mathbf{c}'_1, -f \rangle$ and $\langle \mathbf{c}''_1, -f \rangle$, which gives our LHP samples.

Experimental results. We implemented our attack, in which the LP-based solver uses Gurobi Optimizer (version 13.0.2) [Gur26] under academic license. Our experiments run over 100 randomly generated key instances of Falcon-512 and Falcon-1024 respectively.

Algorithm 4: Our Key Recovery

Input: A set $\mathcal{S}$ of collected LHP samples
 $(\mathbf{c}, \text{sign}(\mu_{(0)}), \text{sign}(\mu_{(1)})) \in \mathbb{Z}_q^n \times \{-1, +1\} \times \{-1, +1\}$, modulus q ,
public key h

Output: Recovered secret key f

```

1 Initialize empty set  $\mathcal{S}'$ ,  $rows \leftarrow \{3n/2 - 1, 2n - 1\}$ 
2 for  $(\mathbf{c}, \text{sign}(\mu_{(0)}), \text{sign}(\mu_{(1)})) \in \mathcal{S}$  do
3   for  $row \in rows$  do
4     if  $row = 3n/2 - 1$  then
5        $(\mathbf{c}'_0^{(0)}, \mathbf{c}'_0^{(1)}) \leftarrow \mathbf{c} \cdot (-\mathbf{P} \cdot (\mathbf{J} \cdot \mathbf{D})^{-1})$ 
6        $\mathbf{w} \leftarrow \mathbf{c}'_0^{(1)}$ 
7       Append  $(\mathbf{w}, \text{sign}(\mu_{(0)}))$  to  $\mathcal{S}'$ 
8     end if
9     if  $row = 2n - 1$  then
10       $(\mathbf{c}'_1^{(0)}, \mathbf{c}'_1^{(1)}) \leftarrow \mathbf{c} \cdot (\mathbf{J} \cdot \mathbf{D})^{-1}$ 
11       $\mathbf{w} \leftarrow \mathbf{c}'_1^{(1)}$ 
12      Append  $(\mathbf{w}, \text{sign}(\mu_{(1)}))$  to  $\mathcal{S}'$ 
13    end if
14  end for
15 end for
16  $f' \leftarrow \text{Solver}(\mathcal{S}')$  ▷ Linear Programming
17  $f' = \frac{1.17\sqrt{q/2}}{\|f'\|} f'$ 
18  $f \leftarrow (\lfloor f'_0 \rfloor, \dots, \lfloor f'_{n-1} \rfloor)$ 
19 return  $f$ 

```

Our attack can fully recover the secret key within a few minutes by using tens of thousands of traces. From a smaller number of traces, one can also obtain an accurate approximation $\mathbf{b}''$ such that $\|\lfloor \mathbf{b}'' \rfloor - \mathbf{b}\|_1 \leq 3$ with high probability, which allows a full key recovery via a simple exhaustive search within half an hour. Table 1 presents the detailed experimental results.

4 Protecting Falcon against Leakage: An Initial Attempt

Side-channel protection for Falcon remains a long-standing challenge. Existing masking-based proposals only target very specific operations [KA24, CC24, KLL⁺26], while the protection of Falcon Gaussian sampler is still far from efficient and effective. In this section, we present a provisional attempt to design a practical countermeasure against known side-channel attacks (Section 3) on the Gaussian sampling. Our initial countermeasure seeks to re-order the vectors in the trapdoor basis during Gaussian sampling by multiplying some root of unity x^i , so as to hide the correlation between leaks and secret. It is conceptually similar to shuffling [RPD09, VMKS12] and incurs very modest overhead. However,

Table 1: The number of the successful key recovery over 100 randomly generated key instances. Here m denotes the number of used traces.

$m \times 10^{-3}$	10	11	12	13	14	15	16	17	18	19	20
Falcon-512	10	19	19	24	28	35	37	39	47	54	58
Falcon-1024	0	0	0	0	4	6	13	16	18	19	27

Algorithm 5: ShufSig

Input: A message msg , secret basis $\mathbf{B}_{f,g}$ and acceptance bound $\lfloor \beta^2 \rfloor$
Output: A valid signature sig of msg

```

1  $r \xleftarrow{\$} \{0, 1\}^{320}, c \leftarrow H(r \parallel \text{msg})$ 
2 do
3   do
4      $i \xleftarrow{\$} \{0, \dots, 2n-1\}, c' \leftarrow x^{-i} c$ 
5      $\mathbf{t}' \leftarrow \left( -\frac{\text{FFT}(c') \odot \text{FFT}(F)}{q}, \frac{\text{FFT}(c') \odot \text{FFT}(f)}{q} \right)$ 
6      $\mathbf{z}' \leftarrow \text{ffSampling}_n(\mathbf{t}', T)$ 
7      $\mathbf{a}' \leftarrow \mathbf{t}' - \mathbf{z}'$ 
8      $\mathbf{a} \leftarrow (a'_0 \odot \text{FFT}(x^i), a'_1 \odot \text{FFT}(x^i))$ 
9      $\mathbf{s} \leftarrow \mathbf{a} \cdot \text{FFT}(\mathbf{B}_{f,g})$   $\triangleright \mathbf{s} \sim D_{(c,0) + \mathcal{L}(\mathbf{B}), \sigma_{\text{sig}}, 0}$ 
10    while  $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$ 
11     $(s_0, s_1) \leftarrow \text{invFFT}(\mathbf{s})$ 
12  while  $s = \perp$ 
13 return  $\text{sig} = (r, s)$ 
```

we show that a simple shuffling is insufficient by providing a second order key recovery attack in the context of the power analysis of [BKM26].

4.1 Initial Countermeasure

All identified sources of leakage in the integer Gaussian sampler are eventually related to the GSO vector $\tilde{\mathbf{b}}_i$ with a certain index, and attackers can learn the secret through cumulative leaks at a fixed location. To defeat these attacks, a natural idea is to *shuffle* the sampling basis. However, a random shuffle does not preserve the algebraic structure of the basis required by `ffSampling`, thus loses the efficiency advantage of Falcon. We observe that multiplying the sampling basis by a root of unity x^i can maintain the required basis structure. Furthermore, this can be equivalently implemented by shuffling the target and then sampling with the original basis, so that the precomputed Falcon tree can be used as is. Based on this, we give an initial countermeasure using one global root of unity shuffling, illustrated in Algorithm 5.

Theorem 1 exhibits the correctness of our initial countermeasure.

Theorem 1. *Suppose that `ffSampling` is an ideal Gaussian sampler, then the output of Algorithm 5 has the same distribution as the original Falcon signatures generated by Algorithm 1.*

Proof. For simplicity, our proof is performed over $\mathbb{Q}[x]/(x^n + 1)$ instead of the FFT domain. First, we prove $\mathbf{s} \in (c, 0) + \mathcal{L}(\mathbf{B})$, i.e. $s_0 + s_1h = c \bmod q$. In Algorithm 5, `ffSampling` samples around a twisted target $\mathbf{t}'$ with the trapdoor $\mathbf{B}$, thus $\mathbf{s}' = \mathbf{a}'\mathbf{B} \in (c', 0) + \mathcal{L}(\mathbf{B})$, i.e. $s'_0 + s'_1h = c' \bmod q$. By multiplying x^i , it follows that $s_0 + s_1h = c \bmod q$, as $\mathbf{s} = x^i \cdot \mathbf{s}'$.

Then it suffices to show that the probability of $\mathbf{s}$ is proportional to $\rho_{\sigma_{\text{sig}}}(\mathbf{s})$. In fact, `ffSampling` ensures that

$$\Pr[\mathbf{s}] = \Pr[\mathbf{z}'] \propto \rho_{\sigma_{\text{sig}}}(\mathbf{s}') = \rho_{\sigma_{\text{sig}}}(x^{-i} \cdot \mathbf{s}) = \rho_{\sigma_{\text{sig}}}(\mathbf{s}).$$

Now the proof is completed. $\square$

4.2 Second Order Attack against the Initial Countermeasure

Algorithm 5 uses a single global root of unity shuffling to disrupt the map between leaks and secret vectors. Still, it could be feasible to attain alignment through second order analysis. We validate a second order key recovery attack against our initial countermeasure, which exploits the sign leakage of intermediate integer sample $z \sim D_{\mathbb{Z}, \sigma, \mu}$ in the setting of [BKM26].

The attack of [BKM26]. We briefly recall the attack of [BKM26] for completeness. In the signing procedure of Falcon, $\mathbf{z} = ((c, 0) - \mathbf{s}) \cdot \mathbf{B}^{-1}$. Let $\mathbf{b}'_i = (b'_0, b'_1)$ denote the i -th column of $\mathbf{B}^{-1}$ and z_i be the i -th coordinate of $\mathbf{z}$, then $z_i = \langle (c, 0) - \mathbf{s}, \mathbf{b}_i \rangle$. One can construct a noisy LHP sample,

$$\langle c - s_0, b'_0 \rangle = z_i + \langle s_1, b'_1 \rangle$$

where $\langle s_1, b'_1 \rangle$ acts as a small noise that can flip the sign of z_i with a low probability. In addition, the attack in [BKM26] uses all sign leaks of the second half of $\mathbf{z}$ to generate n LHP samples. Then one can fully recover the key by solving the noisy LHP instance using logistic regression (LR).

Next, in the context of `ShufSig`, we introduce the collection of valid LHP sample pairs, the fundamental principles of second order attacks and perform key recovery based on the attacks of [BKM26].

Constructing LHP samples for `ShufSig`. Our attack works with a set of signatures generated by `ShufSig`. We use j as a subscript to denote the variables in the j -th call of `ShufSig`. Let $\mathbf{B}_{i_j} = x^{i_j}\mathbf{B}$. Since

$$\mathbf{s}_j = x^{i_j}\mathbf{s}'_j = x^{i_j}(\mathbf{t}'_j - \mathbf{z}'_j)\mathbf{B} = \mathbf{c}_j - \mathbf{z}'_j x^{i_j}\mathbf{B} = (c_j, 0) - \mathbf{z}'_j\mathbf{B}_{i_j}$$

we have $\mathbf{s}_j = (c_j, 0) - \mathbf{z}'_j\mathbf{B}_{i_j}$. Let $\mathbf{w}_j = \mathbf{s}_j - (c_j, 0) = -\mathbf{z}'_j\mathbf{B}_{i_j}$ and $\mathbf{b} = (f^*, g^*)$, then

$$-z'_{j,1}[\mathbf{k}] = \frac{1}{q} \langle x^{-\mathbf{k}}\mathbf{w}_j, x^{i_j}\mathbf{b} \rangle.$$

From side channels, one can obtain the sign information $l_{j,k} = \text{sign}(-z'_{j,1}[k])$ that is

$$l_{j,k} = \text{sign}(\langle x^{-k} \mathbf{w}_j, x^{i_j} \mathbf{b} \rangle) = \text{sign}(\langle x^{-k}(s_{j,0} - c_j), x^{i_j} f^* \rangle + \epsilon)$$

with $\epsilon = \langle x^{-k} s_{j,1}, x^{i_j} g^* \rangle$. This gives n LHP samples for the same $x^{i_j} f^*$, i.e. $(x^{-k} \mathbf{w}_j, l_{j,k})$, for $k \in \{0, \dots, n-1\}$. Since the exponent i_j varies across each signing, accumulating these samples requires realignment.

Second order alignment. Give collected sample pair $(\mathbf{w}_j, l_j)$, we define the signed first moment as

$$\mathbf{d}_j = -\frac{1}{n} \sum_{k=0}^{n-1} l_{j,k} x^{-k} \mathbf{w}_j.$$

Let $\bar{\mathbf{v}}_{j,k} = x^{-k} \mathbf{w}_j$ and $\bar{\mathbf{u}}_j = x^{i_j} \mathbf{b}$, then $l_{j,k} = \text{sign}(\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle)$ and hence

$$\mathbf{d}_j = -\frac{1}{n} \sum_{k=0}^{n-1} \text{sign}(\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle) \bar{\mathbf{v}}_{j,k}.$$

Let $\bar{\mathbf{v}}_{j,k}^\perp$ be the projection of $\bar{\mathbf{v}}_{j,k}$ onto the orthogonal space of $\bar{\mathbf{u}}_j$ such that

$$\bar{\mathbf{v}}_{j,k} = \frac{\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle}{\|\bar{\mathbf{u}}_j\|^2} \bar{\mathbf{u}}_j + \bar{\mathbf{v}}_{j,k}^\perp, \quad \langle \bar{\mathbf{v}}_{j,k}^\perp, \bar{\mathbf{u}}_j \rangle = 0.$$

It follows that

$$l_{j,k} \cdot \bar{\mathbf{v}}_{j,k} = \frac{|\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle|}{\|\bar{\mathbf{u}}_j\|^2} \bar{\mathbf{u}}_j + \text{sign}(\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle) \bar{\mathbf{v}}_{j,k}^\perp.$$

Heuristically, the average of $\text{sign}(\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle) \bar{\mathbf{v}}_{j,k}^\perp$ over all k tends to be $\mathbf{0}$, while the component $\frac{|\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle|}{\|\bar{\mathbf{u}}_j\|^2} \bar{\mathbf{u}}_j$ will amplify in the accumulation. As a consequence, summing all $l_{j,k} \cdot \bar{\mathbf{v}}_{j,k}$ gives an approximate direction of $\bar{\mathbf{u}}_j$, namely

$$\frac{1}{n} \sum_{k=0}^{n-1} \text{sign}(\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle) \bar{\mathbf{v}}_{j,k} \approx \alpha_j \bar{\mathbf{u}}_j,$$

where $\alpha_j = \frac{1}{n\|\bar{\mathbf{u}}_j\|^2} \sum_{k=0}^{n-1} |\langle \bar{\mathbf{v}}_{j,k}, \bar{\mathbf{u}}_j \rangle| > 0$. Then we have

$$\mathbf{d}_j \approx -\alpha_j \bar{\mathbf{u}}_j = -\alpha_j x^{i_j} \mathbf{b}.$$

We then exploit second order correlations to estimate relative exponents for each signature. For a candidate exponent $\nu \in \{0, \dots, 2n-1\}$, we compute

$$\langle \mathbf{d}_j, x^{-\nu} \mathbf{d}_r \rangle \approx \alpha_j \alpha_r \langle x^{i_j} \mathbf{b}, x^{i_r - \nu} \mathbf{b} \rangle = \alpha_j \alpha_r \langle \mathbf{b}, x^{i_r - i_j - \nu} \mathbf{b} \rangle$$

and the maximum corresponds to $\nu = i_r - i_j \bmod 2n$. Once the relative exponent $i_r - i_j \bmod 2n$ is recovered, one can realign the LHP samples to the same exponent i_r . Applying the attack of [BKM26] can give a key recovery against our initial countermeasure.

Table 2: Key recovery against our initial countermeasure **ShufSig**. Here $\#\mathcal{LHP}$ is the number of LHP sample pairs, $\#\mathcal{SIG}$ is the number of used signatures and the success rate is the proportion of recovered secret key over 100 random key instances.

	$\#\mathcal{LHP}$	$\#\mathcal{SIG}$	Success Rate
Falcon-512	51k	100	49%
	102k	200	94%
	153k	300	94%
Falcon-1024	102k	100	74%
	204k	200	96%
	307k	300	100%

Experimental results. We performed key recovery experiments over 100 random key instance respectively for Falcon-512 and Falcon-1024. The LHP solver is implemented by the LR algorithm in scikit-learn package (version 1.9.1) with newton-cholesky solver and no regularization. Table 2 shows the concrete results.

5 Full-Fledged Countermeasure

In this section, we further strengthen our initial countermeasure presented in Section 4 with a protected fast Fourier sampler (Algorithm 6) that uses a fresh root of unity shuffling at each recursion. Additional root of unity shufflings effectively disrupt leakage alignment and thus average away the leakage, while maintaining modest overhead and simple implementation. We reiterate that our full-fledged countermeasure does not provide provable security guarantees like masking and the physical leakage of an individual call of Falcon integer sampler is observable. Nevertheless, linking the leaks and certain secret vector requires the hidden shuffle state, which effectively mitigates statistical analysis based attacks.

In the original **ffSampling** procedure, the recursive traversal of Falcon tree T proceeds in a fixed order. This opens up the avenue to canceling out the global root of unity shuffling as demonstrated in Section 4.2.

To overcome this issue, we propose to further re-order the recursive calls in **ffSampling** via fresh root of unity shufflings. This is equivalent to sampling around a shuffled center, followed by some post-processing with tweaked GSO coefficients. Therefore, even though the recursion retains its original order, the leakage in the calls of **SamplerZ** are associated with intermediate values dependent of all fresh exponents on the recursion path, which disables the alignment.

Moreover, we notice that $\mathbf{b}_{2i}$ and $\mathbf{b}_{2i+1}$ are orthogonal in the Falcon basis. This implies that the calls of **SamplerZ** at two adjacent leaf nodes of Falcon tree can be parallelizable. Hence, we randomly swap their order in the protected **ffSampling** for more robust protections.

A complete description of our protected fast Fourier sampler is provided in Algorithm 6. Applying it to replace the `ffSampling` sampler in our initial countermeasure (Algorithm 5), we obtain the full-fledged countermeasure.

By Theorem 1, the correctness of our full-fledged countermeasure follows from Theorem 2.

Theorem 2. *The output distribution of `NewffSamplingn` is identical to that of `ffSamplingn`.*

The proof of Theorem 2 requires Lemma 1 showing that `NewffSampling` manipulates with the correct LDL^* decomposition coefficients after the root of unity shufflings.

Lemma 1. *Let $\mathbf{G}$ be a Gram matrix over $\mathcal{R}^{2 \times 2}$ of the LDL^* decomposition $\mathbf{G} = \mathbf{LDL}^*$ with*

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ L & 1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} D_0 & 0 \\ 0 & D_1 \end{pmatrix}.$$

For integers j and k , define

$$\mathbf{T}_{j,k} = \begin{pmatrix} x^j & 0 \\ 0 & x^k \end{pmatrix}, \quad \mathbf{G}_{j,k} = \mathbf{T}_{j,k} \mathbf{G} \mathbf{T}_{j,k}^*.$$

Then the LDL^* decomposition of $\mathbf{G}_{j,k}$ is $\mathbf{G}_{j,k} = \mathbf{L}_{j,k} \mathbf{D} (\mathbf{L}_{j,k})^*$ where

$$\mathbf{L}_{j,k} = \begin{pmatrix} 1 & 0 \\ L_{j,k} & 1 \end{pmatrix}, \quad L_{j,k} = x^k L x^{-j}.$$

Proof. It holds that

$$\mathbf{T}_{j,k} \mathbf{D} \mathbf{T}_{j,k}^* = \begin{pmatrix} D_0 & 0 \\ 0 & D_1 \end{pmatrix} = \mathbf{D}$$

and

$$\mathbf{T}_{j,k} \mathbf{L} \mathbf{T}_{j,k}^* = \begin{pmatrix} x^j & 0 \\ 0 & x^k \end{pmatrix} \begin{pmatrix} 1 & 0 \\ L & 1 \end{pmatrix} \begin{pmatrix} x^{-j} & 0 \\ 0 & x^{-k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ x^k L x^{-j} & 1 \end{pmatrix} = \mathbf{L}_{j,k}.$$

It follows that

$$\mathbf{G}_{j,k} = \mathbf{T}_{j,k} \mathbf{L} \mathbf{D} \mathbf{L}^* \mathbf{T}_{j,k}^* = \mathbf{L}_{j,k} \mathbf{D} (\mathbf{L}_{j,k})^*.$$

Due to the uniqueness of the LDL^* decomposition, we complete the proof. $\square$

Proof of Theorem 2. We proceed by the induction on n . For $n = 1$, `NewffSampling` calls `SamplerZ` twice, then restores the order and signs of their outputs. Hence, the output distribution of `NewffSampling` is exactly the same with that of `ffSampling`. The claimed bound follows from the Rényi divergence analysis in [PFH⁺22].

Suppose that Theorem 2 holds for the case of dimension $n/2$. By Lemma 1, for $\mathbf{G}_{j,k} = \mathbf{T}_{j,k} \mathbf{G} \mathbf{T}_{j,k}^*$ with $\mathbf{T}_{j,k} = \begin{pmatrix} x^j & 0 \\ 0 & x^k \end{pmatrix}$, the root node of its Falcon tree is

$$L_{j,k} = x^k L x^{-j}$$

Algorithm 6: NewffSampling_n

Input: A target vector $\mathbf{t} = (t_0, t_1) \in \text{FFT}(\mathcal{K})^2$, a Falcon tree T

Output: $\mathbf{z} = (z_0, z_1) \in \text{FFT}(\mathcal{R})^2$

```

1   $j, k \xleftarrow{\$} \{0, \dots, 2n-1\}$ 
2  if  $n = 1$  then
3     $\hat{x}^j \leftarrow (-1)^j, \hat{x}^k \leftarrow (-1)^k$ 
4  else
5     $\hat{x}^j \leftarrow \text{FFT}(x^j), \hat{x}^k \leftarrow \text{FFT}(x^k)$ 
6  end if
7   $\tau_0 \leftarrow t_0 \odot \hat{x}^{-j}, \tau_1 \leftarrow t_1 \odot \hat{x}^{-k}$ 
8  if  $n = 1$  then
9     $\sigma \leftarrow T.\text{value}$   $\triangleright \sigma \in [\sigma_{\min}, \sigma_{\max}]$ 
10    $b' \xleftarrow{\$} \{0, 1\}$ 
11    $c_0 \leftarrow (1 - b') \cdot \tau_0 + b' \cdot \tau_1$ 
12    $c_1 \leftarrow b' \cdot \tau_0 + (1 - b') \cdot \tau_1$ 
13    $\xi_0 \leftarrow \text{SamplerZ}(c_0, \sigma)$ 
14    $\xi_1 \leftarrow \text{SamplerZ}(c_1, \sigma)$   $\triangleright c_0$  and  $c_1$  are Gaussian centers
15    $\zeta_0 \leftarrow (1 - b') \cdot \xi_0 + b' \cdot \xi_1$ 
16    $\zeta_1 \leftarrow b' \cdot \xi_0 + (1 - b') \cdot \xi_1$ 
17    $z_0 \leftarrow \zeta_0 \odot \hat{x}^j, z_1 \leftarrow \zeta_1 \odot \hat{x}^k$ 
18   return  $\mathbf{z} = (z_0, z_1)$ 
19 end if
20  $(L, T_0, T_1) \leftarrow (T.\text{value}, T.\text{leftchild}, T.\text{rightchild})$ 
21  $L_{j,k} \leftarrow \hat{x}^k \odot L \odot \hat{x}^{-j}$ 
22  $\mathbf{t}_1 \leftarrow \text{split\_fft}(\tau_1)$ 
23  $\mathbf{z}_1 \leftarrow \text{NewffSampling}_{n/2}(\mathbf{t}_1, T_1)$   $\triangleright$  First recursive call
24  $z_1 \leftarrow \text{merge\_fft}(\mathbf{z}_1)$ 
25  $\tau'_0 \leftarrow \tau_0 + (\tau_1 - z_1) \odot L_{j,k}$ 
26  $\mathbf{t}_0 \leftarrow \text{split\_fft}(\tau'_0)$ 
27  $\mathbf{z}_0 \leftarrow \text{NewffSampling}_{n/2}(\mathbf{t}_0, T_0)$   $\triangleright$  Second recursive call
28  $z_0 \leftarrow \text{merge\_fft}(\mathbf{z}_0)$ 
29  $z_0 \leftarrow z_0 \odot \hat{x}^j, z_1 \leftarrow z_1 \odot \hat{x}^k$ 
30 return  $\mathbf{z} = (z_0, z_1)$ 

```

while two child subtrees T_0 and T_1 are the same with those of the Falcon tree of $\mathbf{G}$. Let (z_0, z_1) be output of NewffSampling_n . The first recursive call of $\text{NewffSampling}_{n/2}$ uses τ_1 and T_1 . Let ζ_1 denote its output before undoing x^k . By the induction hypothesis, $\zeta_1 \sim D_{\mathcal{L}(\mathbf{B}_1), \sigma_{\text{sig}}, \text{split\_fft}(\tau_1)\mathbf{B}_1}$ where $\mathbf{B}_1$ corresponds to the subtree T_1 . Since multiplying x^k is an isometric automorphism over $\mathcal{L}(\mathbf{B}_1)$, then

$$z_1 = \zeta_1 \cdot x^k \sim D_{\mathcal{L}(\mathbf{B}_1), \sigma_{\text{sig}}, \text{split\_fft}(x^k \cdot \tau_1)\mathbf{B}_1} = D_{\mathcal{L}(\mathbf{B}_1), \sigma_{\text{sig}}, \text{split\_fft}(t_1)\mathbf{B}_1}.$$

The value subsequently used along with T_0 as the input of the second call of $\text{NewffSampling}_{n/2}$ is

$$\begin{aligned} \tau'_0 &= \tau_0 + (\tau_1 - \zeta_1)L_{j,k} \\ &= x^{-j}t_0 + (x^{-k}t_1 - \zeta_1)x^k Lx^{-j} \\ &= (t_0 + (t_1 - x^k\zeta_1)L)x^{-j} \\ &= (t_0 + (t_1 - z_1)L)x^{-j} \\ &= x^{-j}t'_0. \end{aligned}$$

Similarly, it follows from the induction hypothesis that

$$z_0 \sim D_{\mathcal{L}(\mathbf{B}_0), \sigma_{\text{sig}}, \text{split\_fft}(x^k \cdot \tau'_0)\mathbf{B}_0} = D_{\mathcal{L}(\mathbf{B}_0), \sigma_{\text{sig}}, \text{split\_fft}(t'_0)\mathbf{B}_0}.$$

This coincides with the original `ffSampling` sampling procedure. We complete the proof. $\square$

6 Experimental Evaluation

This section presents the experimental evaluation of our countermeasure with respect to side-channel security and performance.

6.1 Side-Channel Security Evaluation

We used a first order Test Vector Leakage Assessment (TVLA) to analyze the practical side-channel security of our countermeasure. All experiments for leakage assessment were conducted on Chipwhisperer-Lite platforms equipped with an STM32F415 UFO target board (ARM Cortex-M4). We evaluated four types of identified leakage, namely the three sources of leakage described in Section 3.1 and the leakage newly introduced in Section 3.2. Each experiment has two separate sets of traces S_0 and S_1 for each different source of leakage as shown in Table 3. We exploit these two sets to classify the collected traces and compute Welch's t -test. Note that the leakage used for classification are from the first call of `SamplerZ`, as all known attacks exploit the leakage from it. Furthermore, we use a fixed key and generate fresh random seeds to feed the `randombytes`, which produces new randomness for each signature.

Table 3: Two separate sets for different sources of leakage of the first call of `SamplerZ`.

Leakage	Classification
Half Gaussian leakage from y_+	$S_0 = \{y_+ \mid y_+ = 0\}, S_1 = \{y_+ \mid y_+ \neq 0\}$
Sign leakage of y	$S_0 = \{y \mid y \geq 0\}, S_1 = \{y \mid y < 0\}$
Sign leakage of z	$S_0 = \{z \mid z \geq 0\}, S_1 = \{z \mid z < 0\}$
Sign leakage of μ	$S_0 = \{\mu \mid \mu \geq 0\}, S_1 = \{\mu \mid \mu < 0\}$

Our countermeasure is proposed to protect the Falcon Gaussian sampler. The collected traces in our evaluation measure the whole process of signing except for the computation of the hashed message itself and its conversion to FFT representation at the very start of the signature algorithm, as those operations are on public values and hence do not need protection⁵. Since our collected traces contain an unusually large number of sample points, we use the threshold by using the approach of [DZD⁺17]. Specifically, the new threshold is computed by

$$\text{CDF}_{\mathcal{N}(0,1)}^{-1}(1 - \alpha_{\text{TH}}/2) \quad \text{where} \quad \alpha_{\text{TH}} = 1 - (1 - 0.01)^{1/d}$$

where $\text{CDF}_{\mathcal{N}(0,1)}^{-1}$ is the inverse of CDF of standard normal distribution and d is the number of sample points for collected traces. In our setting, we use a threshold of ± 6.47 with $d \approx 10^8$ for reference implementation and ± 6.55 with $d \approx 1.8 \times 10^8$ for our countermeasure implementation.

We report the detailed t -test results in Fig. 2 and observe that t -test curve for reference implementation has clear peaks and exceeds the threshold ± 6.50 . By contrast, the t -test results of our countermeasures have no values, which cross the threshold ± 6.56 .

6.2 Performance Evaluation

We also provide the performance comparison between the reference implementation and the implementation of our countermeasures as illustrated in Section 5. The performance benchmark is carried out on an Intel Core i5-1135G7 CPU, clocked at 2.4 GHz with hyper-threading turned off. We compile the standard C codes with Clang 10.0.0. Optimization flags `-O2` are switched on, consistent with the default configuration with `Makefile` provided in the reference implementation. We exploit the benchmarking tool in Falcon submission package [PFH⁺22] to measure the running clock time and obtain the cycle counts by accessing system counter.

⁵ Moreover, these operations tend to cause spurious t -test failures on the traces for μ and z , since the signs of those variables are correlated with the coefficients of the hashed message. This is not a vulnerability, since again the hashed message is public: those failures just reveal harmless correlations instead of exploited leakage of the secret key.

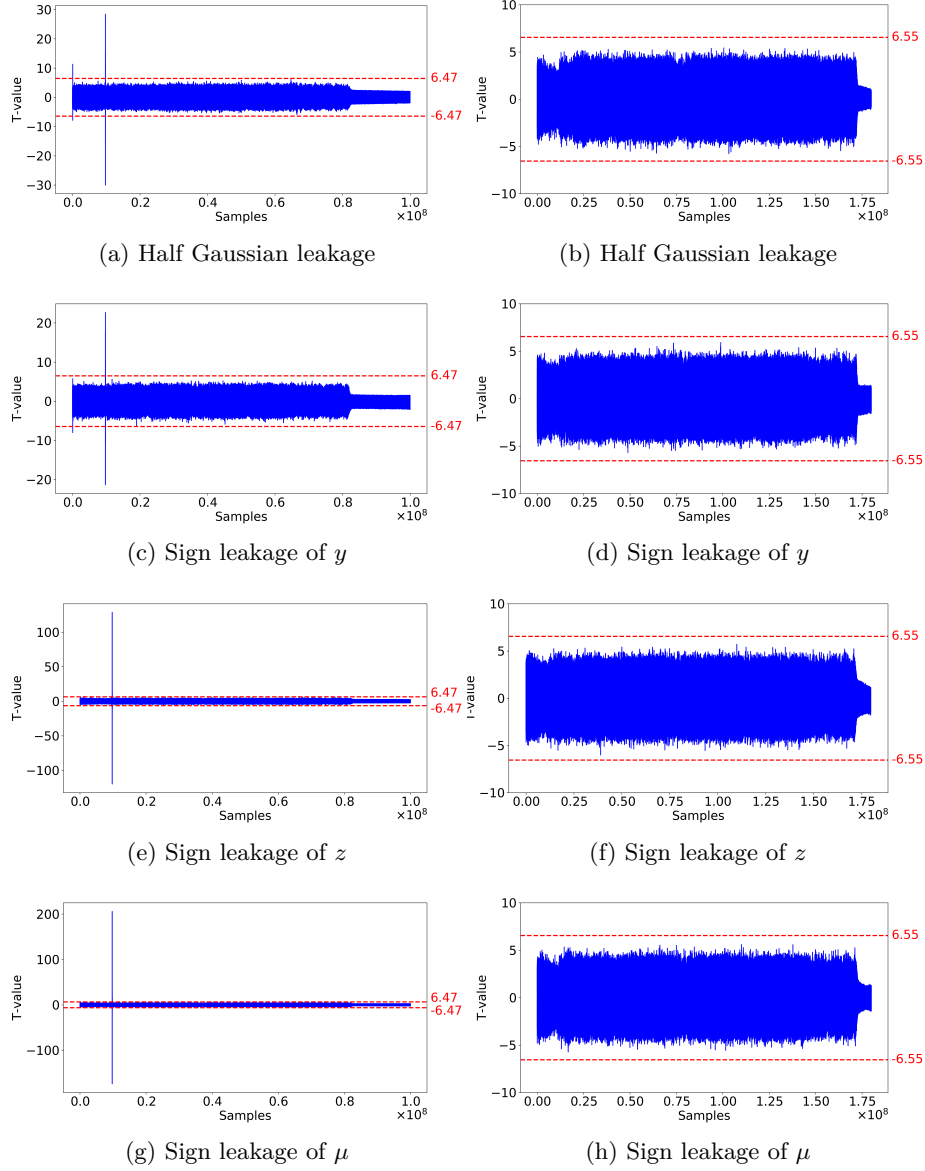


Fig. 2: Specific TVLA evaluation for the reference implementation and that of our countermeasures against different types of leakage. The left column is the assessment for the reference implementation, measured over 2,000 traces. The right column is the assessment for the countermeasure, measured over 20,000 traces.

As shown in Table 4, we report the performance benchmarks and give the corresponding comparison between the reference implementation and our countermeasures for two different signing modes and parameter sets. For Falcon-512 (resp. Falcon-1024), `sign_dyn` mode with our countermeasures yields around $1.4\times$ (resp. $1.5\times$) overhead measured by clock time and $1.5\times$ (resp. $1.5\times$) measured by cycle counts.

Table 4: Performance comparison with Falcon reference implementation, respectively for the signing dynamic (`sign_dyn`) and signing tree (`sign_tree`) modes.

Signing mode	Falcon-512		Falcon-1024	
	<code>sign_dyn</code>	<code>sign_tree</code>	<code>sign_dyn</code>	<code>sign_tree</code>
Reference implementation (ms)	4.7	2.1	10.2	4.5
Our countermeasures (ms)	6.8	4.3	14.9	9.5
Vs.	$1.4\times$	$2.0\times$	$1.5\times$	$2.1\times$
Reference implementation (Mcycles)	11.2	5.0	24.6	10.8
Our countermeasure (Mcycles)	16.4	10.5	36.0	22.9
Vs.	$1.5\times$	$2.1\times$	$1.5\times$	$2.1\times$

References

- AEHT26a. Calvin Abou Haidar, Thomas Espitau, Clément Hoffmann, and Mehdi Tibouchi. Maskaglia: A new, efficient approach to masked discrete Gaussian sampling. In Nadia Heninger and Mike Rosulek, editors, *CRYPTO 2026*, volume 16806 of *LNCS*, pages 164–195. Springer, 2026. [doi:10.1007/978-3-032-35415-0_6](#).
- AEHT26b. Calvin Abou Haidar, Thomas Espitau, Clément Hoffmann, and Mehdi Tibouchi. Slicing bits and cutting costs in cdt sampling: High-order masking of frodokem’s gaussian sampler, revisited. *IACR TCHES*, 2026(4):549–574, Sep. 2026. [doi:10.46586/tches.v2026.i4.549-574](#).
- APT25. Calvin Abou Haidar, Quentin Payet, and Mehdi Tibouchi. Crowhammer: Full key recovery attack on falcon with a single rowhammer bit flip. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part V*, volume 16004 of *LNCS*, pages 103–135. Springer, Cham, August 2025. [doi:10.1007/978-3-032-01901-1_4](#).
- BBE⁺19. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2147–2164. ACM Press, November 2019. [doi:10.1145/3319535.3363223](#).
- BDS23. Sven Bauer and Fabrizio De Santis. A differential fault attack against deterministic Falcon signatures. In Shivam Bhasin and Thomas Roche, editors, *CARDIS*, pages 43–61. Springer, 2023.
- BKM26. Marcus Brinkmann, Nicolai Kraus, and Alexander May. Halfspace learning for lattice signature key recovery from signs. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part III*, volume 16802 of *LNCS*, pages 359–390. Springer, 2026. [doi:10.1007/978-3-032-35377-1_12](#).
- BPT⁺24. Pierre-Augustin Berthet, Justine Paillet, Cédric Tavernier, Lilian Bossuet, and Brice Colombier. Masked computation of the floor function and its application to the FALCON signature. *CiC*, 1(4):9, 2024. [doi:10.62056/ay73z17s](#).
- CC24. Keng-Yu Chen and Jiun-Peng Chen. Masking floating-point number multiplication and addition of falcon first- and higher-order implementations and evaluations. *IACR TCHES*, 2024(2):276–303, 2024. [doi:10.46586/tches.v2024.i2.276-303](#).
- CCCY26. Keng-Yu Chen, Ming Qing Ching, Jiun-Peng Chen, and Bo-Yin Yang. When masking multiplication isn’t enough: Exploiting floating-point leakage in falcon’s pre-image computation. *IACR TCHES*, 2026(2):686–708, Apr. 2026. [doi:10.46586/tches.v2026.i2.686-708](#).
- CRR03. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, Berlin, Heidelberg, August 2003. [doi:10.1007/3-540-36400-5_3](#).
- DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*,

- pages 22–41. Springer, Berlin, Heidelberg, December 2014. [doi:10.1007/978-3-662-45608-8_2](https://doi.org/10.1007/978-3-662-45608-8_2).
- DP16. Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*, pages 191–198. ACM, 2016. [doi:10.1145/2930889.2930923](https://doi.org/10.1145/2930889.2930923).
- DZD⁺17. A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsu Fei. Towards sound and optimal leakage detection procedure. In Thomas Eisenbarth and Yannick Tégli, editors, *CARDIS 2017*, volume 10728 of *LNCS*, pages 105–122. Springer, 2017. [doi:10.1007/978-3-319-75208-2_7](https://doi.org/10.1007/978-3-319-75208-2_7).
- EFG⁺22. Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Cham, May / June 2022. [doi:10.1007/978-3-031-07082-2_9](https://doi.org/10.1007/978-3-031-07082-2_9).
- FKT⁺20. Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Key recovery from Gram-Schmidt norm leakage in hash-and-sign signatures over NTRU lattices. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 34–63. Springer, Cham, May 2020. [doi:10.1007/978-3-030-45727-3_2](https://doi.org/10.1007/978-3-030-45727-3_2).
- GMRR22. Morgane Guereau, Ange Martinelli, Thomas Ricosset, and Mélissa Rossi. The hidden parallelepiped is back again: Power analysis attacks on falcon. *IACR TCHES*, 2022(3):141–164, 2022. [doi:10.46586/tches.v2022.i3.141-164](https://doi.org/10.46586/tches.v2022.i3.141-164).
- GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. [doi:10.1145/1374376.1374407](https://doi.org/10.1145/1374376.1374407).
- GR19. François Gérard and Mélissa Rossi. An efficient and provable masked implementation of qtesla. In Sonia Belaïd and Tim Güneysu, editors, *CARDIS 2019*, volume 11833 of *LNCS*, pages 74–91. Springer, 2019. [doi:10.1007/978-3-030-42068-0_5](https://doi.org/10.1007/978-3-030-42068-0_5).
- Gur26. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: <https://www.gurobi.com>.
- HPRR20. James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 53–71. Springer, Cham, 2020. [doi:10.1007/978-3-030-44223-1_4](https://doi.org/10.1007/978-3-030-44223-1_4).
- KA21. Emre Karabulut and Aydin Aysu. FALCON down: Breaking FALCON post-quantum signature scheme through side-channel attacks. In *DAC 2021*, pages 691–696. IEEE, 2021. [doi:10.1109/DAC18074.2021.9586131](https://doi.org/10.1109/DAC18074.2021.9586131).
- KA24. Emre Karabulut and Aydin Aysu. Masking FALCON’s floating-point multiplication in hardware. *IACR TCHES*, 2024(4):483–508, 2024. [doi:10.46586/tches.v2024.i4.483-508](https://doi.org/10.46586/tches.v2024.i4.483-508).
- KLL⁺26. GyuSang Kim, JeongHwan Lee, Myeonghoon Lee, Seokhie Hong, and HeeSeok Kim. Secret key recovery of falcon using simple power analysis in conditional calculator. *IACR TCHES*, 2026(2):953–975, Apr. 2026. [doi:10.46586/tches.v2026.i2.953-975](https://doi.org/10.46586/tches.v2026.i2.953-975).

- KTA26. Hiroto Kaihara, Mehdi Tibouchi, and Masayuki Abe. Frozentru: Cold boot attacks on ntru-based hash-and-sign signatures. *IACR TCHES*, 2026(3):1007–1031, Jul. 2026. doi:10.46586/tches.v2026.i3.1007-1031.
- LZY⁺25. Xiuhan Lin, Shiduo Zhang, Yang Yu, Weijia Wang, Qidi You, Ximing Xu, and Xiaoyun Wang. Thorough power analysis on falcon gaussian samplers and practical countermeasure. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part I*, volume 15674 of *LNCS*, pages 229–258. Springer, Cham, May 2025. doi:10.1007/978-3-031-91820-9_8.
- MHS⁺19. Sarah McCarthy, James Howe, Neil Smyth, Séamus Brannigan, and Máire O’Neill. BEARZ attack FALCON: implementation attacks with countermeasures on the FALCON signature scheme. In Mohammad S. Obaidat and Pierangela Samarati, editors, *SECRYPT*, pages 61–71. SciTePress, 2019. doi:10.5220/0007834800610071.
- PFH⁺22. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>.
- Pre23. Thomas Prest. A key-recovery attack against mitaka in the t -probing model. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 205–220. Springer, Cham, May 2023. doi:10.1007/978-3-031-31368-4_8.
- QA26. Jinyi Qiu and Aydin Aysu. Ntt-peel: Bit shift side-channel in falcon’s number theoretic transform. *IACR TCHES*, 2026(2):709–735, Apr. 2026. doi:10.46586/tches.v2026.i2.709-735.
- RPD09. Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 171–188. Springer, Berlin, Heidelberg, September 2009. doi:10.1007/978-3-642-04138-9_13.
- VMKS12. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 740–757. Springer, Berlin, Heidelberg, December 2012. doi:10.1007/978-3-642-34961-4_44.
- YMZ⁺26. Mert Yassi, Soundes Marzougui, Raymond K. Zhao, Muhammed F. Esgin, Amin Sakzad, and Ron Steinfeld. Magnet: Masked gaussian now efficient and table-less. *IACR TCHES*, 2026(3):769–804, Jul. 2026. doi:10.46586/tches.v2026.i3.769-804.
- ZLYW23. Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang. Improved power analysis attacks on falcon. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 565–595. Springer, Cham, April 2023. doi:10.1007/978-3-031-30634-1_19.