# The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon [GMRR22]

Mehdi Jelassi* Adrien Bouquet†

***Abstract*—We investigate side-channel attacks on Falcon, a NIST Round-3 finalist for post-quantum digital signature standardization. Given its importance in post-quantum cryptography, analyzing Falcon's side-channel vulnerabilities is critical.**

**Fouque and al.[1] identified Falcon's subroutine discrete Gaussian sampling over integers as a potential vulnerability.**

**[GMRR22] adresses this concern in the second attack they propose but in their first attack, they build on the attack given by [KA21] exploiting electromagnetic leakage from floating-point multiplications in the Fast Fourier Transform to recover the secret signing key. [GMRR22] improves the attack by halving the number of traces and reducing complexity. In the second attack, [GMRR22] perform a simple power analysis during the signature execution to provide the exact value of the output of a subroutine called the base sampler. This intermediate value does not directly lead to the secret. Therefore [GMRR22] had to adapt the so-called hidden parallelepiped attack initially introduced by Nguyen and Regev in Eurocrypt 2006 to extract it.**

## I. Introduction

**Falcon** stands for **Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU. Its main advantage is **compactness**. Among post-quantum signature schemes, none are known to achieve both a public key and signature size as small as that of Falcon.

Falcon is built on the GPV framework, using NTRU lattices and Fast Fourier sampling, making it a highly efficient yet complex hash-and-sign scheme. This complexity comes through two features: the use of floating point arithmetics and the need for Gaussian sampling. The NIST emphasized the need for side-channel analysis on the finalists. [GMRR22][2] aimed at doing this analysis in the case of Falcon. [KA21][3] performed the first concrete power attack threatening Falcon implementations.

This attack targets a subroutine of the algorithm and focuses on the recovery of values encoded in floating points. [GMRR22] improved this attack and provided the first attack against Gaussian Sampler.

## II. Background

In this report, we will need the following concepts.

**Floating-point numbers** in double precision i.e. encoded over 64 bits:

$$(-1)^s \cdot 2^{e-1023} \cdot m$$

with $s$ the sign bit, $e$ the exponent encoded in 11 bits and $m$ the mantissa encoded in 52 bits.

Concerning **side-channel attacks**, we assume to be in the **Threat model** namely the adversary has physical access to the device and captures EM and power measurements while the key-dependent computations are carried out.

A Simple power analysis (SPA) is a side-channel attack where the power usage tells what kind of operation is performed. In algorithms, the metric for the cost is time and space complexity whereas in Power Analysis the metric is the number of traces. An important side-channel attack is **Correlation power analysis (CPA)** against signature scheme. It allows the adversary to find a secret key that is stored on a victim device. There are 4 steps to a CPA attack:

- Write down a model for the victim's power consumption. This model will look at one specific point in the signature algorithm. A model that we will use in this report is the **Hamming weight of a string** which is the number of symbols that are different from the zero string.
- Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures. Attack small parts (subkeys) of the secret key as follows. Consider every possible option for the subkey. For each guess and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model. Calculate the (Pearson) correlation coefficient between the modeled and actual power consumption. Do this for every data point in the traces. Decide which subkey guess correlates best to the measured traces.
- Put together the best subkey guesses to obtain the full secret key.

Finally, we need to recall Falcon signature scheme. First we define Key generation in Falcon. An NTRU lattice is the lattice spanned by $B := \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ viewed in $\mathcal{R}^{2\times 2}$ with $\mathcal{R} := \frac{\mathbb{Z}_q[x]}{(x^n+1)}$. But note that $\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ and $B := \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ span the same lattice.

- **KeyGEN Falcon:** Draw $f, g \in \mathcal{R}$ with small coefficients, compute $F, G \in \mathcal{R}$ satisfying NTRU equation: $fG - gF = q \mod (x^n + 1)$. $h = gf^{-1} \mod q$ is the **public key** and $f, g, F, G$ are the **secret keys**. Private Basis $B \in \mathcal{R}^{2n\times 2n}$, Public Basis $A \in \mathcal{R}^{2n}$ seen as $A := (1 \quad h^*) \in \mathcal{R}^2$ with $h^* := h(x^{-1})$.

We are in position now to define Falcon Signature algorithm.

**Algorithm 1** FALCON.SIGN$(m, sk)$

**Input:** A message $m$, a secret key $sk = (\hat{\mathbf{B}}, T)$
**Output:** A signature *sig*

$r \xleftarrow{\$} \{0,1\}^{320}$ uniformly
$c \leftarrow \texttt{HashToPoint}(r \,\|\, m, q, n)$ ▷ $c \in \mathcal{R}$
$\hat{\mathbf{t}} \leftarrow (\hat{c}, 0) \cdot \hat{\mathbf{B}}^{-1}$ ▷ • pre-image computation
**do**
  $\hat{\mathbf{v}} \leftarrow \texttt{ffSampling}(\hat{\mathbf{t}}, T)$ ▷ • trapdoor sampler
  $\hat{\mathbf{s}} \leftarrow (\hat{\mathbf{t}} - \hat{\mathbf{v}}) \cdot \hat{\mathbf{B}}$
**while** $\|\mathbf{s}\|^2 > \lfloor 2.42 \cdot n \cdot \sigma^2 \rfloor$
**return** *sig* $:= (r, \mathbf{s})$

where $\sigma := \frac{1.17}{\pi\sqrt{2}} \cdot \sqrt{q \cdot \log\left(4n\left(1 + 2^{32} \cdot \sqrt{n/4}\right)\right)}$.

The verification is defined as follows: if $\|s\|^2 \leq \lfloor 2.42 \cdot n \cdot \sigma^2 \rfloor$, then the signature is accepted as valid. Otherwise, it is rejected. In order to tackle Section 4, we recall the definition of a Discrete Gaussian distribution and the Hidden Parallelepiped Problem. **The Hidden Parallelepiped Problem (HPP):** Let $\mathbf{B} := (\mathbf{b_0}, ..., \mathbf{b_{n-1}})$ be $n$ linearly independent vectors and let $\mathcal{P}(\mathbf{B}) = \{\sum_{i=0}^{n-1} x_i \mathbf{b_i},\ x_i \in [-1, 1]\}$, the parallelepiped spanned by $\mathbf{B}$.
Given a sequence of poly(n) independent samples drawn uniformly at random in $\mathcal{P}(\mathbf{B})$, find a good approximation of $\mathbf{B}$. **Gaussian Distribution:** For $\sigma \in \mathbb{R}$ with $\sigma > 0$ and any $\mathbf{c} \in \mathbb{R}^n$, we call the Gaussian function centered at $\mathbf{c}$ of standard deviation $\sigma$ the function defined over $\mathbb{R}^n$ as $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) := \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right)$. We call discrete Gaussian distribution over $\Lambda$ of standard deviation $\sigma$ and center $\mathbf{c}$ the distribution defined for all $\mathbf{z} \in \Lambda$ by $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{z}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{z})}{\sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})}$.

## III. Improvement of pre-image attack

First we recall the attack given by [KA21], then we explain the two main improvements given by [GMRR22]. The attack works as follows. We measure the voltage power of victim's device with respect to time during the execution of Falcon Sign to $N$ messages. This yields $N$ transcripts $(c_1, s_1), \ldots, (c_N, s_N)$ where $c_i$'s are the digests and $N$ traces $P_1, \ldots, P_N$. We focus on the execution of Step 3 of Falcon Sign (Preimage-computation) cf Figure 1, namely $\hat{t} \leftarrow (\hat{c}, 0) \cdot \hat{B}^{-1}$. In $(\hat{c}, 0) \cdot \hat{B}^{-1} = (\hat{c} \cdot \hat{F}, \hat{c} \cdot \hat{f})$, we focus on $\hat{c} \cdot \hat{f}$. We want to recover the secret keys, namely $f, g, F, G$. But **Key recovery reduces** to recover $f$ (actually $\hat{f}$ thanks to InvFFT) from $\hat{c} \cdot \hat{f}$. Indeed, $h = g \cdot f^{-1} \mod q$ allows us to recover $g$ and given $f, g$ we recover $F, G$ via NTRU equation. Let's denote by $\hat{f}[i]$ the i-th coefficient of $\hat{f}$ and $\hat{c}_j[i]$ the i-th coefficient of $\hat{c}_j$. As coefficients of $f$ are real (since $f \in \mathcal{R} := \frac{\mathbb{Z}_q[X]}{(x^n+1)}$), by property of FFT one has $\hat{f}[i] = \overline{\hat{f}[n-1-i]}$, for $i = 0, \ldots, \frac{n}{2}$, where $\bar{\cdot}$ denotes the complex conjugate. Hence, it is enough to find only the first $\frac{n}{2}$ coefficients of $\hat{f}$.

In order to do so, we apply Correlation Power Analysis as follows. We compute for each trace the following Hamming weights where for a fixed time $t$, we denote by $P_1[t], \ldots, P_N[t]$ the measurements at time $t$ that yield intermediate values in the computation of $\hat{c}_i[j] \cdot \hat{f}[j]$ for some $j = 0, \ldots, \frac{n}{2}$ and any $i = 1, \ldots, N$ and denote by HW the Hamming weight. $P_j : \text{HW}(\text{Re}(\hat{c}_j[i]) \cdot \text{Re}(\hat{f}[i]))$ for $j = 1, \ldots, N$. Then, as $\hat{c}$ is known, we make a guess on $\text{Re}(\hat{f}[i])$ so that the $N$ correlations $\rho_1\left(HW\left(\text{Re}(\hat{c}_1[i]) \cdot \text{guess}\right), HW\left(\text{Re}(\hat{c}_1[i]), \text{Re}(\hat{f}[i])\right)\right), \ldots,$ $\rho_N\left(HW\left(\text{Re}(\hat{c}_N[i]) \cdot \text{guess}\right), HW\left(\text{Re}(\hat{c}_N[i]), \text{Re}(\hat{f}[i])\right)\right)$ are maximized.

If $N$ is large, i.e. we have many traces, then guess will most likely be $\text{Re}(\hat{f}[i])$ as desired. But if $N$ is small, there may be many possible values of guess which maximize the $N$ above correlations. But in that case, [GMRR22] had the idea to use the redundancy of complex multiplication as follows. Write $\hat{c}_j[i] = \text{Re}(\hat{c}_j[i]) + i\,\text{Im}(\hat{c}_j[i])$ and $\hat{f}[i] = \text{Re}(\hat{f}[i]) + i\,\text{Im}(\hat{f}[i])$ which are floating-points. We have the following complex multiplication

$$\begin{aligned}\hat{c}_j[i] \cdot \hat{f}[i] &= (\text{Re}(\hat{c}_j[i]) + i\,\text{Im}(\hat{c}_j[i])) \cdot (\text{Re}(\hat{f}[i]) + i\,\text{Im}(\hat{f}[i])) \\ &= \text{Re}(\hat{c}_j[i])\text{Re}(\hat{f}[i]) - \text{Im}(\hat{c}_j[i])\text{Im}(\hat{f}[i]) \\ &\quad + i\left(\text{Re}(\hat{c}_j[i])\text{Im}(\hat{f}[i]) + \text{Im}(\hat{c}_j[i])\text{Re}(\hat{f}[i])\right)\end{aligned}$$

From this, we extract: $\text{Re}(\hat{c}_j[i]) \cdot \text{Re}(\hat{f}[i])$, $\text{Im}(\hat{c}_j[i]) \cdot \text{Re}(\hat{f}[i])$, $\text{Im}(\hat{c}_j[i]) \cdot \text{Im}(\hat{f}[i])$, $\text{Re}(\hat{c}_j[i]) \cdot \text{Im}(\hat{f}[i])$. But the first two multiplications allow us to recover $\text{Re}(\hat{f}[i])$ and the second two to recover $\text{Im}(\hat{f}[i])$ instead of using four multiplications to recover the real part and four to recover the imaginary part. We explain below how precisely the first two lead to recovering $\text{Re}(\hat{f}[i])$. Instead of considering $P_1[t], \ldots, P_N[t]$ for a fixed time index $t$, we consider the concatenated traces $P_1[t] \cup P_1[t'], \ldots, P_N[t] \cup P_N[t']$ for $t' \neq t$, where we compute for $P_1'$: $\text{HW}((\text{Im}(\hat{c}_1[i])) \cdot \text{Im}(\hat{f}[i]))$,..., for $P_N'$: $\text{HW}(\text{Im}(\hat{c_N}[i]) \cdot \text{Im}(\hat{f}[i]))$. Then, it is unlikely that a guess different than $\text{Re}(\hat{f}[i])$ maximizes the following $2N$ correlations
$\rho_1\left(HW\left(\text{Re}(\hat{c}_1[i]) \cdot \text{guess}\right), HW\left(\text{Re}(\hat{c}_1[i]), \text{Re}(\hat{f}[i])\right)\right), \ldots,$
$\rho_N\left(HW\left(\text{Re}(\hat{c}_N[i]) \cdot \text{guess}\right), HW\left(\text{Re}(\hat{c}_N[i]), \text{Re}(\hat{f}[i])\right)\right)$
and $\rho_1\left(HW\left(\text{Im}(\hat{c}_1[i]) \cdot \text{guess}\right), HW\left(\text{Im}(\hat{c}_1[i]), \text{Re}(\hat{f}[i])\right)\right), \ldots,$
$\rho_N\left(HW\left(\text{Im}(\hat{c}_N[i]) \cdot \text{guess}\right), HW\left(\text{Im}(\hat{c}_N[i]), \text{Re}(\hat{f}[i])\right)\right)$

In a similar way, the adversary guesses $\text{Im}(\hat{f}[i])$. At the end, we recovered only the real and the imaginary part of one coefficient namely $\hat{f}[i]$ out of $\frac{n}{2}$ coefficients of $\hat{f}$. But if we consider FALCON-512 ($n = 512$), we need $\frac{n}{2} = 256$ for the real part and another $\frac{n}{2} = 256$ for the imaginary part so 512 coefficients to recover. We explained the attack given by [KA21] and how [GMRR22] were able to halve the number of traces. Now, we explain their second main improvement in this attack: partial knowledge of $\hat{f}$ leads to Key Recovery. [KA21] noted that during the multiplication $\hat{c}_j[i] \cdot \hat{f}[i]$ for $j \in \{0, \ldots, n\}$ and $i \in \{0, \ldots, \frac{n}{2}\}$ the sign, exponent and mantissa are computed separately (signs are xored, exponents added and mantissa multiplied) and so can be retrieved separately. But [KA21] needed the whole mantissa

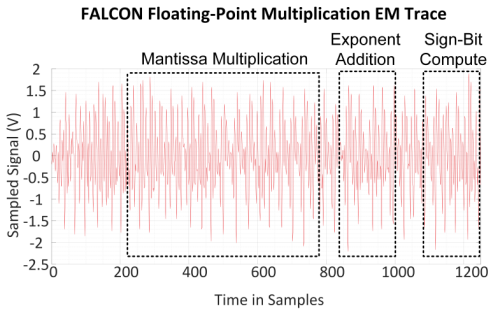


Figure 1: An example EM measurement trace from [KA21] showing the related mantissa, exponent, and sign computations.

in order to recover $\hat{f}$. [GMRR22] showed by experiments that recovering only the 8 MSB's of the mantissa is enough. Indeed, [GMRR22] experimented their attacks on both simulated traces and real acquisitions. Their simulated traces were generated using the ELMO simulator. The actual power traces were generated from a ChipWhisperer Lite with STM32F3 target. In the ELMO setup, 2000 traces were sufficient to recover the full key, whereas the ChipWhisperer setup required 5000 traces. Finally, [GMRR22] demonstrated through experiments that the probability of recovering a single intermediate value using all multiplication patterns—across 2000 traces with data consisting of $10 \times 16$ coefficients is approximately $0.9$. This significantly improves upon the result from [KA21], which achieves a success probability of only $0.5$.

## IV. Unravelling the HPP with Side-Channel Information

[GMRR22] attacked Falcon's Gaussian trapdoor sampler, more precisely the `BaseSampler` which led to a key recovery. The attack is structured in three parts. The first step consists of applying a simple power analysis on the `BaseSampler` to determine whether $z_i \in \{0, 1\}$ or not. In the second step, the paper leverages this knowledge by applying an HPP solver to filtered sets of signatures with $z_i \in \{0, 1\}$ to obtain an approximation of the private key. Finally, the attack recovers the key by performing a lattice reduction.

### A. SPA on the `BaseSampler`

The `BaseSampler` is a key component of FALCON's implementation. In fact, `SamplerZ` calls it to generate random integers following a half discrete Gaussian distribution centered at $0$ and of fixed standard deviation $\sigma_{max}$ (often $1.8205$ in practice). `BaseSampler` is constant-time and makes use of a reverse cumulative distribution table (RCDT) as presented in Algorithm 2. However, table-based samplers like `BaseSampler` are vulnerable to Simple Power Analysis. In our situation, the comparison $u <$ RCDT$[i]$ at line 4 is implemented with 3 consecutive 24-bit integer subtractions

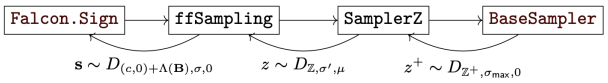


Figure 2: Signature's flowchart

and the result is stored in a 32-bit register. Upon underflows (i.e., when the subtraction's result is negative), the 8 most significant bits are set to 1, thus increasing the Hamming weight by 8 and the power consumption. Therefore, it's possible to distinguish on a trace whether the variable $z^+$ has been incremented or not. This side-channel attack has shown great results as $100\%$ of the signatures on the ELMO simulator and $94.2\%$ of the signatures on a ChipWhisperer were well correctly classified, thus making this attack feasible.

To perform the next step of the attack, only samples $z^+ = 0$ are relevant, and we assume that the samples are correctly computed after this step. Furthermore, a random sign will be generated after the `BaseSampler`, thus $z_i \in \{0, 1\}$. Hence, the attack consider these two values.

**Algorithm 2** `BaseSampler`

**Output:** An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{max},0}$

1: $u \leftarrow$ **`UniformBits`**$(72)$
2: $z^+ \leftarrow 0$
3: **for** $i \leftarrow 0$ **to** $17$ **do**
4: $\quad z^+ \leftarrow z^+ + [[u < RCDT[i]]]$
5: **end for**
6: **return** $z^+$

### B. Applying the HPP & Recovering the key

Once we know $z_i$, the next step consists in solving the Hidden Parallelepiped Problem and apply the HPP solver proposed in [4]. FALCON's signatures are generated using the `ffSampling` algorithm, which works analogously to Klein-GPV algorithm. The latter takes a target vector $\mathbf{t_n}$ and a binary tree T representing the GSO of the private basis as input and produces a vector $\mathbf{v} \sim D_{(c,0)+\Lambda(\mathbf{B}),\sigma,0}$.
The Klein-GPV [5] starts with $\mathbf{v_n} \leftarrow 0$ and for $i = n-1, ..., 0$:

1) $x_i \leftarrow \langle \mathbf{t_i}, \tilde{\mathbf{b_i}} \rangle / \|\tilde{\mathbf{b_i}}\|^2$
2) Pick $z_i \sim D_{\mathbb{Z},\sigma',x_i-\lfloor x_i \rfloor}$ with $\sigma' := \sigma / \|\tilde{\mathbf{b_i}}\|$, and let $k_i \leftarrow \lfloor x_i \rfloor + z_i$
3) Let $\mathbf{t_i} \leftarrow \mathbf{t_{i+1}} - k_i \mathbf{b_i}$ and $\mathbf{v_i} \leftarrow \mathbf{v_{i+1}} + k_i \mathbf{b_i}$

At the end, the algorithm returns $\mathbf{v_0}$. As indicated in line 2 of the algorithm, the variable $z_i$ is as a random perturbation, and hence does not lead to any information on the private key. Furthermore, we have:

- $z^+ = 0 \rightsquigarrow z_i \in \{0, 1\} \rightsquigarrow y_i \in (-1, 1] \implies s \in \mathcal{P}(\tilde{B})$ in order to apply HPP solver.
- $z^+ = 0 \rightsquigarrow z_i \in \{0, 1\}$ since in SamplerZ implementation ([1]), $z_i \leftarrow b + (2 \cdot b - 1)z^+$, for $b = 0$ or $1$ with equal probability.

- $z_i \in \{0,1\} \rightsquigarrow y_i \in (-1,1]$ since $\mathbf{s} := (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$, $x_i := \langle \mathbf{t}, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$:

$$\mathbf{s} = \sum_{i \in [n]} y_i \cdot \tilde{\mathbf{b}}_\mathbf{i} \quad \text{where } y_i = z_i - x_i + \lfloor x_i \rfloor.$$

- but the distribution of $y_i$ is not uniform over $(-1,1] \rightsquigarrow$ we cannot directly apply the HPP solver.

Nevertheless, the knowledge of $z_i \in \{0,1\}$, removes the probabilistic part of the sampling. By applying this constraint to all the coefficients and filtering all the generated signatures, it will disclose the parallelepiped, thus allowing us to apply an HPP solver. However, one can not assume $z_i$ to be zero for all $i = 0, \ldots, n$ because $n$ is large (i.e., $n \in \{512, 1024\}$). One can only assume $z_i = 0$ for some $i = 0, \ldots, n$. This will not fully disclose a parallelepiped but one direction is preserved. This is where the deformed variant of the HPP becomes relevant [6]. Applying the HPP solver on partial perturbation (i.e., indices where $z_i \in \{0,1\}$) returns a correct approximation for these indices. Therefore, we obtain $(g', -f')$ , which is an approximation of $\mathbf{b_0} = (g, -f)$.

Finally, the last steps consist in recovering the exact private key from the approximation $(g', -f')$ we obtained previously. As such, there are two options. First, we can perform a mere rounding but this requires a lot of signatures. In fact, with 5 millions signatures, we have a high probability $(> 0.99)$ to have an absolute error less than $0.5$ on each coefficient. On the other hand, we can use a Leaky LWE / NTRU estimator tool that will solve the problem with less signatures measurements. However, it requires more computation time and work as it is a trade-off. With 1 million signatures, it needs $\approx 1000$ hours to obtain the result, whereas with 1.5 million, the computation time starts to be reasonable ($\sim 24hours$) for such an attack.

### C. Countermeasure

Finally, the paper proposes one small countermeasure to their attack: The entry point of the attack was the vulnerability of the `BaseSampler`, especially the comparison $u <$ RCDT[$i$], when it underflows on the last subtraction. The author propose to replace that last subtraction by an addition as follows:

**Algorithm 3** Countermeasure on the last subtraction

**Input**: the 24 MSB of $\overline{RCDT[i]}$ and $\overline{u}$, and $c$ the results of the two previous comparison
**Output:** 1 if $\overline{u} < \overline{RCDT[i]}$, 0 otherwise.

1: $b \leftarrow$ `0xffffff`
2: $b \leftarrow b - \overline{u} + \overline{RCDT[i]} + c$
3: **return** $b \gg 24$

In fact, when $\overline{RCDT[i]} + c > \overline{u}$, $b$ will overflow, thus setting the 25th bits to 1 and the rest to 0, and this will return 0. This simple mitigation reduces the Hamming weight by a factor of 8, and makes it harder to detect the result of the comparison. However, this offers only a weak security assurance and a provable masking implementation would be better.

## V. Related Work and Improvements

[CC24][7] provided the first masking floating-point multiplication and addition which protects Falcon's pre-image computation against the attack of [KA21]. In response to the second attack presented by [GMRR22], Lin and al.[8] proposed effective and easy-to-implement countermeasures against leakages to protect Falcon's integer Gaussian sampler.

## VI. Conclusion

The attack on preimage computation is interesting for anyone looking to perform SCA on polynomial multiplication on floating points. [GMRR22][2] showed that a partial knowledge of FFT of the secret key leads to a key recovery. Also, the authors halved the number of required traces by using redundancy of complex multiplication. As sign, exponent and mantissa can be retrieved separately in the multiplication, it reduces adversary's exhaustive search on guessing $\mathrm{Re}(\hat{f}[i])$ from $2^{64}$ possibilities to "only" $2 + 2^8 + 2^{11}$ possibilities. On the other hand, the attack on `BaseSampler` worked since $u < RCDT[i]$ is a 72 bits substractions which makes it hard to perform on a device with only 32-bit registers. But Falcon allows to do three 24 bits substractions which leads to a side-channel attack which works by noting when $z^+$ is incremented. Keeping track on the calls by SamplerZ and ffSampling, this leads to a way to apply HPP solver of [NR06][4] and [ND12][6] in order to recover the key.

## References


[1] P.-A. Fouque, P. Kirchner, T. Prest, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over ntru," NIST Post-Quantum Cryptography Standardization Process, 2018, 2018. [Online]. Available: https://falcon-sign.info/falcon.pdf

[2] M. Guerreau, A. Martinelli, T. Ricosset, and M. Rossi, "The hidden parallelepiped is back again: Power analysis attacks on falcon," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 3, pp. 141–164, 2022. [Online]. Available: https://eprint.iacr.org/2022/057.pdf

[3] E. Karabulut and A. Aysu, "Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks," Cryptology ePrint Archive, Paper 2021/772, 2021. [Online]. Available: https://eprint.iacr.org/2021/772

[4] P. Q. Nguyen and O. Regev., "Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures." In Serge Vaudenay, editor, EUROCRYPT 2006, volume 4004 of LNCS, pages 271–288. Springer, Heidelberg, May / June 2006.

[5] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '00. USA: Society for Industrial and Applied Mathematics, 2000, p. 937–941.

[6] L. Ducas and P. Q. Nguyen, "Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures," in *Advances in Cryptology - ASIACRYPT 2012*, ser. Lecture Notes in Computer Science, X. Wang and K. Sako, Eds., vol. 7658. Springer, Dec. 2012, pp. 433–450. [Online]. Available: https://doi.org/10.1007/978-3-642-34961-4_27

[7] K.-Y. Chen and J.-P. Chen, "Masking floating-point number multiplication and addition of falcon: First- and higher-order implementations and evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/11428

[8] X. Lin, S. Zhang, Y. Yu, W. Wang, Q. You, X. Xu, and X. Wang, "Thorough power analysis on falcon gaussian samplers and practical countermeasure," Cryptology ePrint Archive, Xiuhan Lin and Shiduo Zhang and Yang Yu and Weijia Wang and Qidi You and Ximing Xu and Xiaoyun Wang, 2025. [Online]. Available: https://eprint.iacr.org/2025/351