IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2026, No. 2, pp. 686–708. DOI:10.46586/tches.v2026.i2.686-708

When Masking Multiplication Isn't Enough: Exploiting Floating-Point Leakage in Falcon's Pre-Image Computation

Keng-Yu Chen[1], Ming Qing Ching[2], Jiun-Peng Chen[3] and Bo-Yin Yang[3]

[1] EPFL, Lausanne, Switzerland keng-yu.chen@epfl.ch
[2] National Taiwan University, Taipei, Taiwan, b11901137@ntu.edu.tw
[3] Academia Sinica, Taipei, Taiwan, jiunpeng@ntu.edu.tw,byyang@iis.sinica.edu.tw

Abstract. In this paper, we present an improved correlation power analysis (CPA) attack on the pre-image computation of the digital signature scheme Falcon. Our attack exploits new side-channel leakage that multiplication masking schemes fail to protect. To enhance both the efficiency and accuracy of the attack, we develop new theoretical insights for recovering the secret floating-point numbers, which can also be leveraged to improve prior attacks. For mantissa recovery, we identify and correct a flaw in an earlier work and provide a more complete and practical analysis. For exponent recovery, we analyze the distribution of Falcon's secret key after the fast Fourier transform, reduce the number of required traces, and mitigate false positives. To validate our attack, we conducted two experiments targeting existing countermeasures on floating-point multiplication. In our environment, we successfully recovered the secret key using only around one thousand power traces. Our results demonstrate that protecting floating-point multiplication alone is insufficient to defend Falcon against side-channel attacks. A comprehensive masking including at least floating-point addition is necessary.

Keywords: Falcon · Side-Channel Analysis · Floating-Point Arithmetic · Masking

1 Introduction

The advent of quantum computing poses a significant challenge to the security of classical public-key cryptosystems. Shor's algorithm [Sho97], when executed on a sufficiently powerful quantum computer, can efficiently solve integer factorization and discrete logarithm problems, rendering schemes such as Diffie-Hellman key exchange [DH76], RSA [RSA78], ElGamal encryption [Elg85], and ECDSA [JMV01] insecure. To mitigate this threat, the National Institute of Standards and Technology (NIST) launched the post-quantum cryptography (PQC) standardization process in 2016 [oSTa]. After several rounds of evaluation, NIST has selected five algorithms for standardization [oSTb]. These include CRYSTALS-Kyber/ML-KEM [SAB+22] and HQC [AAB+22] for key-encapsulation mechanisms (KEM), as well as CRYSTALS-Dilithium/ML-DSA [LDK+22], Falcon/FN-DSA [PFH+22], and SPHINCS$^+$/SLH-DSA [HBD+22] for digital signatures.

The selected algorithms are proven secure against indistinguishability attacks (for KEMs) and forgery attacks (for digital signatures), given the hardness of underlying mathematical problems that are believed to be still hard even for quantum computers. Nevertheless, while these schemes offer theoretical security guarantees, their real-world implementations can suffer side-channel analysis attacks. Side-channel analysis exploits the physical behaviors, such as variations in execution time, power consumption, electro-

Received: 2025-10-15 Accepted: 2025-12-15 Published: 2026-04-23

magnetic emission, etc., from devices running cryptographic algorithms to extract secret information. Over the years, extensive research has been dedicated to this area.

Among the three selected post-quantum digital signature schemes, Falcon has the smallest signature + public-key size, which makes it a competitive candidate for systems aiming to minimize data transmission costs [oSTc]. However, analyzing its side-channel vulnerabilities is particularly challenging due to its floating-point arithmetic. Most cryptographic schemes, such as ECDSA and CRYSTALS-Dilithium, perform operations over the integer ring $\mathbb{Z}$ or modular rings $\mathbb{Z}_N$ for some integer N. In contrast, Falcon's signing process applies the fast Fourier transform to polynomials before further computations. While this improves efficiency, coefficients are now over complex field $\mathbb{C}$ and simulated by floating-point units in practical implementations. This makes the side-channel analysis and countermeasures for Falcon more complicated and in need of further research.

Related Work Previous works have identified two side-channel weaknesses of Falcon: discrete Gaussian sampling and pre-image computation. The discrete Gaussian sampling process generates a short signature vector following the discrete Gaussian distribution over lattices. Early works revealed that this process was susceptible to timing attacks [BHLY16, EFGT17, PBY17, MHS+19], and this problem was later fixed by an isochronous sampler [ZSS20, HPRR20]. Beyond timing attacks, the discrete Gaussian sampling also faces the threat of power analysis attacks. Guerreau et al. [GMRR22] first presented an attack that exploits the leakage from the base sampler to recover the full secret key. Building on their work, Zhang et al. [ZLYW23] revealed more leakage sources and developed a new algorithm to reduce the attack complexity. More recently, their work was further improved in [LZY+25], which identified new leakages and applied a more efficient algorithm to decrease the number of traces required to extract the secret key. They also proposed countermeasures to protect the discrete Gaussian sampling against these attacks. In addition, Qiu and Aysu [QA25] presented the first single-trace template attack on the discrete Gaussian sampling in the key generation process, demonstrating more threats against Falcon.

The second vulnerability, pre-image computation, involves multiplications of public hashed messages and secret keys after fast Fourier transforms. This computation consists of floating-point multiplications followed by floating-point additions for each coefficient. Karabulut and Aysu [KA22] first demonstrated an electromagnetic side-channel attack on this process. By analyzing the floating-point multiplication procedure in Falcon's reference implementation, they recovered the coefficients of the secret key in the FFT domain. Guerreau et al. [GMRR22] further improved their attack by reducing the number of required traces and narrowing the range of possible guesses. In particular, they showed that recovering a few most significant bits of the mantissa suffices for a full key recovery in the coefficient domain. The complexity of their attack is thus dominated by the recovery of the 11-bit exponent.

For countermeasures against side-channel attacks on the pre-image computation, Chen and Chen [CC24] proposed the first masking scheme for both floating-point multiplication and addition in Falcon. Their approach follows a standard technique of splitting sensitive values into random shares to prevent leakage of intermediate variables, and they provide a theoretical proof of security. The resulting overheads are, however, overwhelming compared to the unprotected implementation. The multiplication and addition cost 23× and 35× cycles respectively compared to unprotected ones for 2-shared masking, and 35× and 99× respectively for 3-shared masking. Besides, Karabulut and Aysu [KA24] introduced the first hardware masking scheme. While their solution experimentally offers more robust first-order security, it focuses specifically on the protection of floating-point multiplication, and floating-point addition was outside the defined scope of their work.

Given the high overheads, especially the floating-point addition, and the fact that recent

hardware scheme [KA24] focuses exclusively on masking multiplication, a natural question arises: Is it strictly necessary to protect floating-point addition to defend Falcon against side-channel attacks on its pre-image computation? The incentive to omit such protection is significant; for example, in the 2-share design in [CC24], the masked floating-point addition relies on complex gadgets like `SecAdd` and `SecFprNorm64`. Avoiding these operations would save approximately 69% of cycles and 72% of the required randomness. Therefore, we investigate whether this omission, though computationally attractive, creates a critical vulnerability that allows for secret key recovery.

Our Contribution In this paper, we answer this question in the affirmative. Specifically, we demonstrate a successful attack on a Falcon implementation that masks floating-point multiplications but omits masking for additions. Our primary contributions are as follows:

- We propose a novel correlation power analysis (CPA) [BCO04] attack on Falcon's pre-image computation. While our attack builds upon prior works [KA22, GMRR22], we identify and exploit new leakage sources that occur after floating-point multiplication. Specifically, our approach targets the floating-point multiplication product instead of the intermediate values used in [KA22, GMRR22], which requires a more careful analysis of the mantissa product and the exponent sum. As a result, it applies even to implementations that employ multiplication masking countermeasures, such as SecFprMul in [CC24] and the hardware design in [KA24].

- We provide a detailed analysis of the range of mantissa for attacks on the pre-image computation. We first point out a flaw in the mantissa estimation in [GMRR22], and we correct it by closely analyzing the distribution of Falcon's secret key after the fast Fourier transform. We clearly show that 11 bits of mantissa suffice to recover the full secret key in the coefficient domain with high probability.

- We derive a much narrower range of possible exponents compared to existing works by modeling the distribution of each coefficient of Falcon's secret key after the fast Fourier transform. The search space is reduced from 2048 elements in prior works [KA22, GMRR22] to no more than 20. Notably, we demonstrate that this reduction not only lowers the computation complexity but also reduces the number of required traces for successful key recovery. Additionally, we provide a theoretical analysis to claim that small exponents can be ignored with high-quality traces.

Our findings reveal a critical vulnerability: masking floating-point multiplication alone is insufficient to prevent side-channel attacks. To achieve robust security, a full masking scheme that covers both floating-point multiplication and addition is necessary.

Experiment To evaluate the effectiveness of our attack, we ran the masked floating-point multiplication in Falcon's signing algorithm and recorded power traces using a ChipWhisperer-Lite [Inc] with an STM32F405 target. The secret keys and messages were generated according to the specification of Falcon. In our experiment, we successfully recovered the full secret key with 1300 and 1200 traces on a standard desktop for two different countermeasure methods, respectively.

Outline In Section 2, we introduce the notation used throughout the paper and describe relevant background materials. In Section 3, we provide in detail our attack targeting the pre-image computation of Falcon. In Section 4, we present our experimental setup and performance evaluation.

2 Preliminaries

2.1 Notation

In this paper, let $n = 2^\kappa$ for an integer $\kappa \in \{9, 10\}$ and $q = 12289$ be a prime number. $\phi = X^n + 1$ is the cyclotomic polynomial. For a polynomial $f(X)$ modulo ϕ, we can write it as an n-by-n matrix, with the ith row representing the coefficients of $(X^i f \bmod \phi)$ for $0 \le i < n$. In this matrix representation, vector and matrix additions and multiplications map to polynomial additions and multiplications in the ring of polynomials modulo ϕ.

For a variable x, we denote its jth bit as $x^{(j)}$ and the ith bit to jth bit, where $j \ge i$, as $x^{[j:i]}$. Bit-wise XOR, AND, and OR operations are denoted by $\oplus$, $\wedge$, and $\vee$, respectively. The bit-wise negation of variable x is $\neg x$, and $\mathsf{HW}(x)$ is the Hamming weight of x. The $\ll$ and $\gg$ are unsigned left- and right-shifts of a variable.

Finally, for a proposition P, we define

$$[\![P]\!] = \begin{cases} 1, & \text{if } P \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Discrete Gaussian Distribution

Given two real numbers σ and c, define the function

$$\rho_{\sigma,c}(x) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right).$$

The *discrete Gaussian distribution* over integers is defined by its probability mass function

$$D_{\mathbb{Z},\sigma,c}(z) = \frac{\rho_{\sigma,c}(z)}{\sum_{x\in\mathbb{Z}} \rho_{\sigma,c}(x)}.$$

The numbers σ and c are the standard deviation and the center of the distribution, respectively.

We can extend the definition to a distribution over lattices. Let $\mathbf{B}$ be an m-by-m matrix for some integer m. The lattice $\mathcal{L}(\mathbf{B})$ formed by $\mathbf{B}$ is the set

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{zB} \mid \mathbf{z} \in \mathbb{Z}^m\}.$$

Let $\mathbf{c} \in \mathbb{R}^m$. Define analogously the function $\rho_{\sigma,\mathbf{c}}(\mathbf{v}) = \exp\left(-\frac{\|\mathbf{v}-\mathbf{c}\|^2}{2\sigma^2}\right)$. The discrete Gaussian distribution over a lattice $\mathcal{L}(\mathbf{B})$ is defined by the following probability mass function

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

In Falcon, each coefficient of its secret polynomials f and g is drawn from a discrete Gaussian distribution over integers. In addition, the signature is a vector drawn from a discrete Gaussian distribution over a lattice formed by its key.

2.3 Falcon Signature Scheme

In this section, we briefly describe the Falcon signature scheme. For more details, we refer the reader to the submission of NIST round-3 Falcon [PFH+22].

Falcon instantiates the GPV framework [GPV08] over NTRU lattices. The secret key consists of four polynomials $f, g, F, G \in \mathbb{Z}_q[x]/(\phi)$. The coefficients of f and g are sampled

from a discrete Gaussian distribution with standard deviation $\sigma_{\{f,g\}} = 1.17\sqrt{q/2n}$. Given f and g, the polynomials F and G are computed such that they satisfy the NTRU equation:

$$fG - gF = q \pmod{\phi}.$$

The public key is given by the polynomial $h = gf^{-1} \bmod (\phi, q)$. For convenience, we define the secret key matrix:

$$\mathbf{B} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right].$$

To sign a message, FALCON samples a lattice point from a discrete Gaussian distribution. It begins by drawing a random salt r and computing the hashed message $c = \mathsf{H}(\mathsf{r}\|\mathsf{m})$ for a hash function H. Next, it finds the *pre-image*

$$\mathbf{t} = \mathbf{c}\mathbf{B}^{-1} = (c, 0) \times \mathbf{B}^{-1}.$$

This is the main target of our attack. The *fast Fourier nearest plane* algorithm [DP16] is then applied to find a vector $\mathbf{z}$ such that $\mathbf{z}\mathbf{B}$ follows the discrete Gaussian distribution $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$. The signature vector is given by $\mathbf{s} = (\mathbf{t} - \mathbf{z})\mathbf{B}$. To reduce the signature size, the final output consists of the salt r and only the second coefficient s_2 of $\mathbf{s}$.

The complete algorithm Sign is detailed in Algorithm 1. The secret key sk includes the matrix $\mathbf{B}$ and a FALCON tree T, which is used in the fast Fourier nearest plane algorithm and is precomputed in the key generation.

Algorithm 1 Sign [PFH+22]

Input: A message m, a secret key sk and bound $\lfloor\beta^2\rfloor$
Output: Signature sig of m

1: $\mathsf{r} \xleftarrow{\$} \{0,1\}^{320}$
2: $c \leftarrow \mathsf{H}(\mathsf{r}\|\mathsf{m})$
3: $\mathbf{t} \leftarrow \left(-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), \frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f)\right)$ ▷ pre-image computation
4: **do**
5: **do**
6: $\mathbf{z} \leftarrow \mathsf{ffSampling}(\mathbf{t}, \mathsf{T})$
7: $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{z})\hat{\mathbf{B}}$
8: **while** $\|\mathbf{s}\|^2 > \lfloor\beta^2\rfloor$
9: $(s_1, s_2) \leftarrow \mathsf{invFFT}(\mathbf{s})$
10: $\mathsf{s} \leftarrow \mathsf{Compress}(s_2)$
11: **while** $\mathsf{s} = \perp$
12: $\mathsf{sig} \leftarrow (\mathsf{r}, \mathsf{s})$
13: **return** sig

2.4 Fast Fourier Transform

In Algorithm 1, the pre-image computation (Line 3) and the fast Fourier nearest-plane algorithm (Lines 6 and 7) apply the fast Fourier transform on polynomials. Let Ω_ϕ be the set of all complex roots $\{\exp\left(\frac{i\pi(2k+1)}{n}\right) \mid 0 \le k < n\}$ of ϕ, the fast Fourier transform of a polynomial $f(X)$ is

$$\mathsf{FFT}(f) = (f(\xi))_{\xi \in \Omega_\phi}.$$

If $f(X) = f[0] + f[1]X + \cdots + f[n-1]X^{n-1}$, we have

$$\hat{f} = \mathsf{FFT}(f) = (\hat{f}[0], \hat{f}[1], \cdots, \hat{f}[n-1]),$$
$$\hat{f}[k] = \sum_{m=0}^{n-1} f[m] \exp\left(\frac{i\pi \cdot (2k+1) \cdot m}{n}\right).$$

We say f is in the coefficient domain and $\hat{f}$ is in the FFT domain. Using the fast Fourier transform, polynomial additions and multiplications in the coefficient domain can be efficiently computed by performing the corresponding element-wise operations on each complex coefficient in the FFT domain. To transform the polynomial back to the coefficient domain, we apply the inverse fast Fourier transform:

$$f = \mathsf{invFFT}(\hat{f}) = f[0] + f[1]x + \cdots + f[n-1]x^{n-1},$$
$$f[k] = \frac{1}{n} \sum_{m=0}^{n-1} \hat{f}[m] \exp\left(\frac{-i\pi \cdot (2m+1) \cdot k}{n}\right).$$

Note that when f is real, which is the case in Falcon,

$$\begin{aligned}\hat{f}[n-1-k] &= \sum_{m=0}^{n-1} f[m] \exp\left(\frac{i\pi \cdot (2(n-1-k)+1) \cdot m}{n}\right) \\ &= \sum_{m=0}^{n-1} f[m] \exp\left(\frac{-i\pi m(2k+1)}{n}\right) \\ &= \overline{\hat{f}[k]},\end{aligned}$$

where $\overline{\hat{f}[k]}$ is the complex conjugate of $\hat{f}[k]$. That is, the second half of the coefficients of $\hat{f}$ can be recomputed by the first half. Consequently, to recover the full vector $\hat{f}$ and subsequently f, it suffices to recover only the first half of its coefficients.

2.5 Floating-Point Numbers and Operations

In the implementation of Falcon, a complex number $a + bi$ is represented by two floating-point numbers a and b for its real and imaginary parts. Due to hardware constraints, implementing Falcon on devices without a floating-point unit can be challenging [PFH+22]. To address this issue, Falcon provides an emulation of 53-bit precision floating-point operations in their reference implementation. It follows the IEEE-754 standard, where a 64-bit floating-point number is composed of a 1-bit sign, an 11-bit exponent, and a 53-bit mantissa, with a leading 1 omitted in storage. The exponent is biased by adding 1023 to simplify the comparison of two floating-point numbers. For convenience, we represent a 64-bit floating-point number x as a tuple (s, e, m), where $s \in \{0, 1\}$, $e \in [0, 2^{11})$, and $m \in [2^{52}, 2^{53})$ are integers. The numerical value of x is given by

$$x = (-1)^s \cdot 2^{e-1023} \cdot (m \cdot 2^{-52}).$$

The floating-point multiplication, denoted by $\mathsf{fpr\_mul}$ in Falcon's reference implementation, begins by XORing the sign bits, adding the exponents, and multiplying the mantissas of both operands. The exponent is also added by a constant offset -2100, which is for the exponent bias (1023×2), mantissa scaling (52×2), and right-shift adjustment (-50) to leave two extra bits for mantissa rounding. The mantissa product result is a 105- or 106-bit value, depending on whether a carry occurs. It is then rounded to 55 bits with the 55th bit always set. During rounding, the sticky bit is preserved. The above

computation assumes that both operands are non-zero. If either operand is zero, which can be determined by checking if the exponent is zero, the calculation is invalid and the mantissa is then set to zero. Finally, the sign bit, unbiased exponent, and 55-bit mantissa are packed into a floating-point number by the FPR subroutine, which applies exponent biasing and additional mantissa rounding. The complete algorithms fpr_mul and FPR, as described in [CC24], are given in Algorithms 2 and 3, respectively.

Algorithm 2 fpr_mul [CC24]

Input: Floating-point numbers $x = (sx, ex, mx)$ and $y = (sy, ey, my)$
Output: A floating-point number product of x and y

$s \leftarrow sx \oplus sy$
$e \leftarrow ex + ey - 2100$
$z \leftarrow mx \times my$
$b \leftarrow [\![z^{[50:1]} \neq 0]\!]$
$z \leftarrow z^{[106:51]} \vee b$
$z' \leftarrow (z \gg 1) \vee z^{(1)}$
$w \leftarrow z^{(106)}$
$z \leftarrow z \oplus (z \oplus z') \wedge (-w)$
$e \leftarrow e + w$
$bx \leftarrow [\![ex \neq 0]\!]$, $by \leftarrow [\![ey \neq 0]\!]$
$b \leftarrow bx \wedge by$
$z \leftarrow z \wedge (-b)$
return $\mathsf{FPR}(s, e, z)$

Algorithm 3 FPR [CC24]

Input: A sign bit s, an exponent e, and a 55-bit mantissa z
Output: A floating-point number x packed by s, e, and z

$e \leftarrow e + 1076$
$b \leftarrow [\![e < 0]\!]$
$z \leftarrow z \wedge (b - 1)$
$b \leftarrow [\![z \neq 0]\!]$
$e \leftarrow e \wedge (-b)$
$x \leftarrow ((s \ll 63) \vee (z \gg 2)) + e \ll 52$
$f \leftarrow \texttt{0XC8} \gg z^{[3:1]}$
$x \leftarrow x + f^{(1)}$
return x

3 The Proposed Attack

The pre-image computation consists of coefficient-wise complex number multiplications, where each operation involves two floating-point multiplications followed by a floating-point addition. Let c and f be the public hashed message and the secret key polynomial, respectively. For each coefficient k of $\hat{c}_1 + \hat{c}_2 i = \hat{c} = \mathsf{FFT}(c)$ and $\hat{f}_1 + \hat{f}_2 i = \hat{f} = \mathsf{FFT}(f)$, the pre-image computation produces the complex number

$$(\hat{c}_1[k] \times \hat{f}_1[k] - \hat{c}_2[k] \times \hat{f}_2[k]) + (\hat{c}_1[k] \times \hat{f}_2[k] + \hat{c}_2[k] \times \hat{f}_1[k])i,$$

where the operations $+, -$, and $\times$ are floating-point addition, subtraction, and multiplication, respectively. Our attack targets the four floating-point products $\hat{c}_1 \times \hat{f}_1, \hat{c}_2 \times \hat{f}_1, \hat{c}_1 \times \hat{f}_2$,

and $\hat{c}_2 \times \hat{f}_2$, all of which are computed using the fpr_mul routine described in Algorithm 2. By recovering all the coefficients of $\hat{f}_1$ and $\hat{f}_2$, we can run an inverse fast Fourier transform to reconstruct the secret polynomial f in the coefficient domain. With f and the public polynomial $h = gf^{-1}$, the remaining secret key components g, F, and G can be recomputed.

To recover $\hat{f}_1$ and $\hat{f}_2$, consider the sign bit, exponent, and mantissa in the function fpr_mul. The sign bit s is computed by the XOR of the sign bits of the two input floating-point operands. The exponent, after Line 6 in Algorithm 3, equals the sum of the exponents of the two floating-point operands, potentially incremented by a carry bit (Line 9 in Algorithm 2), unless it is overwritten to 0 (Line 5 in Algorithm 3). If the exponent is set to 0, one of the input floating-point numbers is 0, which happens with very low probability. The mantissa is not exactly the product of the input mantissas; however, their high-order bits coincide with high probability. The difference arises due to sticky bit handling (Lines 5 and 6 in Algorithm 2), rounding effects (Line 8 in Algorithm 3), and cases where the mantissa is forced to 0 when either operand is 0 (Lines 12 in Algorithm 2 and Line 3 in Algorithm 3). These effects have little impact on the high-order bits or occur only with very low probability. Finally, the sign bit, exponent, and mantissa are combined to form the resulting floating-point number x (Line 13 in Algorithm 2 and Line 9 in Algorithm 3).

Let (s_c, e_c, m_c) and (s_f, e_f, m_f) denote two input operands in fpr_mul, where (s_c, e_c, m_c) corresponds to a coefficient of either $\hat{c}_1$ or $\hat{c}_2$, and (s_f, e_f, m_f) corresponds to the matching coefficient of either $\hat{f}_1$ or $\hat{f}_2$. Let (s, e, m) represent their floating-point product. With high probability, we have

$$s = s_c \oplus s_f \tag{1}$$

$$e = e_c + e_f + w = e_c + e_f + [\![m_c \times m_f \geq 2^{105}]\!] \tag{2}$$

$$m^{[53:53-h]} = \begin{cases} (m_c \times m_f)^{[105:105-h]} & \text{if } w = 0 \\ (m_c \times m_f)^{[106:106-h]} & \text{if } w = 1 \end{cases} \tag{3}$$

Here, w indicates the presence of a carry in the mantissa multiplication, and h denotes the number of high-order bits under consideration.

3.1 Threat Model and Leakage Assessment

Our attack adopts a passive power analysis threat model, similar to [KA22, GMRR22]. The adversary measures power traces while the device executes the Sign algorithm of Falcon in Algorithm 1 with an unknown secret key. Crucially, we target implementations that employ multiplication masking countermeasures (e.g., [CC24, KA24]) but omit addition masking, due to the significant overheads of masking floating-point addition and the fact that existing attacks only target the intermediate values of floating-point multiplication.

Before detailing our attack, we first validate its feasibility by presenting a Test Vector Leakage Assessment (TVLA) [GGJR+11] to demonstrate that the targeted implementation exhibits exploitable leakage. We used a ChipWhisperer-Lite [OC14] setup with an STM32F405 board (featuring an ARM Cortex-M4 core) as the target platform to measure the power traces.

TVLA uses the fixed-versus-random Welch's t-test, which compares the power traces from two distinct sets: one recorded with fixed inputs and one with random inputs. For each time sample, the tester computes the t-statistic. In practice, the null hypothesis of indistinguishability (i.e., no difference between the fixed and random trace sets) is rejected when the t-statistic exceeds the threshold of $|t| > 4.5$. This corresponds to a p-value below 10^{-5} and indicates plausibly exploitable leakage.

Figure 1 illustrates the TVLA result for a sequence of two masked floating-point multiplications (SecFprMul from [CC24]) followed by one floating-point addition (fpr_add

from the FALCON reference implementation [PFH+22]). These operations collectively represent a segment of the complex number multiplication of a coefficient within FALCON. We recorded a total of 1000 traces: 500 with fixed inputs and 500 with random inputs. The ±4.5 significance threshold is depicted as a horizontal cyan dotted line in the figure.

As shown in the graph, the most significant leakage manifests toward the end of the masked multiplications, where the random shares are recombined. At this point, the values of sign s, exponent e, and mantissa m (in Equations (1), (2), and (3), respectively) become visible in the power trace. Our attack specifically targets these three values.

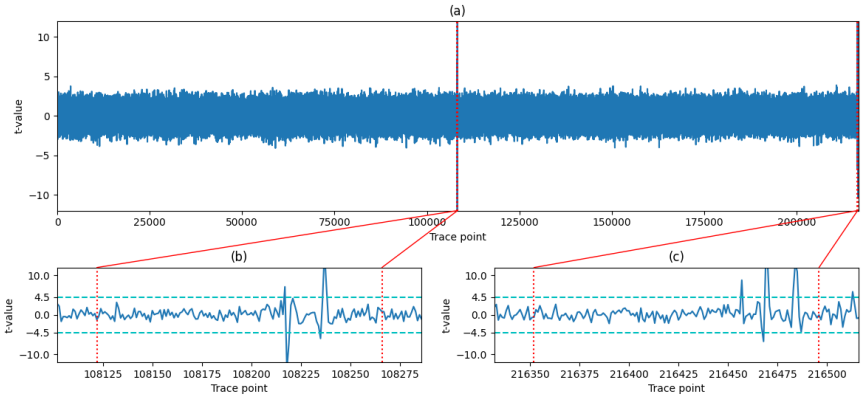

Figure 1: (a) TVLA result of two masked floating-point multiplication followed by one floating-point addition. (b)(c) depict the recombination of shares and FPR at the end of SecFprMul from [CC24].

3.2 Our Attack

We first provide an overview of our attack and detail the reasoning of each parameter in Sections 3.3 and 3.4.

For each coefficient $0 \leq k < \frac{n}{2}$, our attack proceeds as follows:

1. Capture N traces of FALCON's pre-image computation for the products $\hat{c}_1 \times \hat{f}_1$, $\hat{c}_2 \times \hat{f}_1$, $\hat{c}_1 \times \hat{f}_2$, and $\hat{c}_2 \times \hat{f}_2$ with N different polynomials $\hat{c}_1$ and $\hat{c}_2$.

   Let (s_c, e_c, m_c) be the k^{th} coefficient of either $\hat{c}_1$ or $\hat{c}_2$. We apply the technique proposed in [GMRR22, Section 3.3] to effectively double the number of traces to $2N$ by utilizing both $\hat{c}_1 \times \hat{f}_1$ and $\hat{c}_2 \times \hat{f}_1$ when recovering $\hat{f}_1$ and both $\hat{c}_1 \times \hat{f}_2$ and $\hat{c}_2 \times \hat{f}_2$ when recovering $\hat{f}_2$.

2. To recover the sign bit, determine whether $s_f = 0$ or $s_f = 1$ yields the highest correlation between the traces and $\mathsf{HW}(s_c \oplus s_f)$.

3. To recover the mantissa, find $m_f = m'_f \times 2^{52-h}$, where $m'_f \in [2^h, 2^{h+1})$, that yields the highest correlation between the traces and $\mathsf{HW}(m_c \times m_f)$.

   We detail the choice of h for mantissa recovery in Section 3.3.

4. To recover the exponents, use the mantissa obtained in the previous step to determine whether a carry occurs ($w = 0$ or 1). Find $e_f \in [e_\ell, e_u]$ that yields the highest correlation between the traces and $\mathsf{HW}(e_c + e_f + w)$, where

$$[e_\ell, e_u] = \begin{cases} [1016, 1031] & \text{for } n = 512, \text{ this covers over } 95\% \text{ of } f \\ [1013, 1031] & \text{for } n = 512, \text{ this covers over } 99.999\% \text{ of } f \end{cases}.$$

We justify the ranges $[1013, 1031]$ and $[1016, 1031]$ in Section 3.4.

After recovering all the coefficients in the real and imaginary parts, we compute the complex conjugate to obtain the coefficients in the second half. We then apply the inverse fast Fourier transform and round the real part of the resulting vector to the nearest integer to reconstruct the secret key f.

In a masked floating-point multiplication scheme, either SecFprMul proposed in [CC24] or the hardware design in [KA24], computations within fpr_mul are performed over secret shares. However, if floating-point addition remains unprotected, to correctly execute the pre-image computation, these shares must be recombined into sensitive intermediate values. Our attack targets the unmasked outputs of fpr_mul and thus remains effective even with this countermeasure.

3.3 Range of Mantissa

Based on Equation (3), the high-order bits of the correct mantissa m and the product guess $m_c \times m_f$ are likely to be equal with high probability. Therefore, we first select h as a parameter for our attack. We then enumerate the set

$$\{m'_f \times 2^{52-h} \mid m'_f \in [2^h, 2^{h+1})\},$$

to find the value m_f in it that results in the highest correlation in CPA. Our estimate of the mantissa is m_f, which shares the same $h+1$ most significant bits with the correct mantissa and loses $52-h$ low-order bits of information.

Enumerating all 2^h possible values of m'_f is inefficient when h is large. In our attack, h is selected based on the following considerations:

- Previous work [GMRR22] suggests that f can be recovered in the coefficient domain with correct sign bits, exponents, and $h \geq 5$. However, we show in Section 3.3.1 that their proof contains a flaw. By analyzing the distribution of exponents in Section 3.4, we show that $h \geq 10$ is sufficient for a successful attack in over 90% of cases and $h \geq 11$ for over 99.999% of cases.
- The accuracy of mantissa carry recovery impacts the recovery of exponents.
- A smaller h results in more noise from the lower-order bits of the mantissa, and requires more traces to recover it.
- A larger h increases the attack's execution time, as the time complexity is primarily determined by the recovery of the mantissa.

These factors are influenced by the quality and quantity of traces the attacker acquires, and they affect the time complexity and the success probability of the attack. In our experiments in Section 4, we set $h = 12$ to ensure a successful attack even in the presence of high noise in the power traces.

3.3.1 Lower Bound of h

In [GMRR22, Lemma 1], the authors claim that $h \geq 5$, or equivalently $h+1 \geq 6$ most significant bits, along with correct sign bits and exponents are sufficient to recover the secret key in the coefficient domain. This is because all coefficients of the secret key are bounded integers in the coefficient domain. One can recover them by rounding the numbers to the nearest integer, and it does not require high precision in the FFT domain. However, we demonstrate that there is a flaw in their proof. A small counterexample for $n = 4$ is

$$f = (-31, 18, -29, -30).$$

By computing $\hat{f} = \mathsf{FFT}(f)$ and retaining the sign bit, exponent, and 7 most significant bits of the mantissa of each coefficient, we obtain

$$\hat{f}' = (-2.9375 - 37i, -64 - 20.5i, -64 + 20.5i, 2.9375 + 37i).$$

Applying the inverse fast Fourier transform to $\hat{f}'$, we have

$$f' = \mathsf{invFFT}(\hat{f}') = (-30.531, 17.832, -28.750, -29.499).$$

Rounding f' to the nearest integer does not recover f. A counterexample for $n = 512$ is provided in Appendix A.

This inconsistent result arises because the relative error varies across coefficients. For each coefficient $0 \leq j < n$, let $\hat{f}'[j] = \hat{f}[j] + \hat{\epsilon}[j]\hat{f}[j]$, where $\hat{f}'$ represents an approximation of $\hat{f}$ and $\hat{\epsilon}$ is the relative error. If $\hat{\epsilon}[j]$ is not constant, after applying the inverse fast Fourier transform, $f'[j] \neq f[j] + \epsilon[j]f[j]$. Instead,

$$f'[j] = f[j] + (\epsilon \cdot f)[j]$$

where $\cdot$ denotes a polynomial multiplication. The term $(\epsilon \cdot f)[j]$ can exceed $\frac{1}{2}$ even if $|\epsilon[j]| < 2^{-6}$ and $|f[j]| < 2^5$ for each j, as suggested by [GMRR22]. This causes the rounding of $f'[j]$ to differ from the integer $f[j]$.

We correct their proof by Theorem 1. For this, we first show that if the coefficients of $\hat{f}$ in the FFT domain are bounded, the coefficients of f in the coefficient domain are also bounded.

Lemma 1. *Let $f(X) \in \mathbb{Z}[X]/(\phi)$ be a polynomial and $\hat{f} = \mathsf{FFT}(f)$. If the absolute value of each coefficient of $\hat{f}$ is bounded by a real number B, then the absolute value of each coefficient of f is also bounded by B.*

Proof. Recall the inverse fast Fourier transform of $\hat{f}$ is given by

$$f = \mathsf{invFFT}(\hat{f}) = f[0] + f[1]X + \cdots + f[n-1]X^{n-1},$$
$$f[k] = \frac{1}{n}\sum_{m=0}^{n-1} \hat{f}[m] \exp\left(\frac{-i\pi \cdot (2m+1) \cdot k}{n}\right).$$

For each coefficient $0 \leq k < n$, we have

$$\begin{aligned} |f[k]| &\leq \frac{1}{n}\sum_{m=0}^{n-1} \left|\hat{f}[m] \exp\left(\frac{-i\pi \cdot (2m+1) \cdot k}{n}\right)\right| \\ &\leq \frac{1}{n}\sum_{m=0}^{n-1} |\hat{f}[m]| \left|\exp\left(\frac{-i\pi \cdot (2m+1) \cdot k}{n}\right)\right| \\ &\leq \frac{1}{n} \cdot n \cdot B = B. \end{aligned}$$

□

Now, we show that if the coefficients of the secret key in the FFT domain are bounded, which we demonstrate in Section 3.4, then we only need the sign bits, exponents, and a few most significant bits of mantissa to recover the coefficients in the coefficient domain.

Theorem 1. *Let $f(X) \in \mathbb{Z}[X]/(\phi)$ be a polynomial and $\hat{f} = \mathsf{FFT}(f) = \hat{f}_1 + \hat{f}_2 i$, where $\hat{f}_1$ and $\hat{f}_2$ are its real and imaginary parts, respectively. Suppose that there exists an integer p such that $|\hat{f}_1[j]| < 2^p$ and $|\hat{f}_2[j]| < 2^p$ for all coefficients $0 \leq j < n$, then the knowledge for all the $\hat{f}_1[j]$ and $\hat{f}_2[j]$ of (1) their sign bits, (2) their exponents, and (3) their $p + 3$ most significant bits of their mantissas is enough to derive f exactly.*

Proof. For each coefficient $0 \le j < n$ and $b \in \{1, 2\}$, let $m_b[j]$ be the exact mantissa of $\hat{f}_b[j]$, and let $m'_b[j]$ be the approximation of $m_b[j]$, where the bits outside the $p+3$ most significant bits are set to 0. Let $\hat{f}'[j] = \hat{f}'_1[j] + \hat{f}'_2[j]i$ be the approximation of $\hat{f}[j]$, which shares the same sign bit and exponent as $\hat{f}[j]$, but its real and imaginary mantissas are $m'_1[j]$ and $m'_2[j]$, respectively, rather than $m_1[j]$ and $m_2[j]$.

Table 1: Variables used in our proof of Theorem 1

Variable	Description	Example
$\hat{f}_1[j]$	Real part of $\hat{f}[j] = \mathsf{FFT}(f)[j]$	$(s_1, e_1, (1\mathsf{X}_1\mathsf{X}_2\mathsf{X}_3 \cdots \mathsf{X}_{52})_2)$
$\hat{f}_2[j]$	Imaginary part of $\hat{f}[j] = \mathsf{FFT}(f)[j]$	$(s_2, e_2, (1\mathsf{Y}_1\mathsf{Y}_2\mathsf{Y}_3 \cdots \mathsf{Y}_{52})_2)$
$m_1[j]$	mantissa of $\hat{f}_1[j]$	$(1\mathsf{X}_1\mathsf{X}_2\mathsf{X}_3 \cdots \mathsf{X}_{52})_2$
$m_2[j]$	mantissa of $\hat{f}_2[j]$	$(1\mathsf{Y}_1\mathsf{Y}_2\mathsf{Y}_3 \cdots \mathsf{Y}_{52})_2$
$m'_1[j]$	approximation of $m_1[j]$	$(\overbrace{1\mathsf{X}_1\mathsf{X}_2\mathsf{X}_3 \ldots \mathsf{X}_{p+2}}^{p+3 \text{ bits}}000\cdots)_2$
$m'_2[h]$	approximation of $m_2[j]$	$(\overbrace{1\mathsf{Y}_1\mathsf{Y}_2\mathsf{Y}_3 \ldots \mathsf{Y}_{p+2}}^{p+3 \text{ bits}}000\cdots)_2$
$\hat{f}'_1[j]$	approximation of $\hat{f}_1[j]$	$(s_1, e_1, (1\mathsf{X}_1\mathsf{X}_2\mathsf{X}_3 \ldots \mathsf{X}_{p+2}000\cdots)_2)$
$\hat{f}'_2[j]$	approximation of $\hat{f}_2[j]$	$(s_2, e_2, (1\mathsf{Y}_1\mathsf{Y}_2\mathsf{Y}_3 \ldots \mathsf{Y}_{p+2}000\cdots)_2)$

Since the mantissas of $\hat{f}'_1[j]$ and $\hat{f}_1[j]$ share the same $p+3$ bits,

$$\frac{|\hat{f}_1[j] - \hat{f}'_1[j]|}{|\hat{f}_1[j]|} = \frac{|m_1[j] - m'_1[j]|}{|m_1[j]|} < 2^{-p-2} \implies |\hat{f}_1[j] - \hat{f}'_1[j]| < 2^{-p-2} \cdot |\hat{f}_1[j]| < 2^{-2}.$$

Similarly, $|\hat{f}_2[j] - \hat{f}'_2[j]| < 2^{-2}$, and therefore

$$|\hat{f}'[j] - \hat{f}[j]| < \sqrt{2^{-4} + 2^{-4}} = 2^{-\frac{3}{2}}.$$

Since the inverse fast Fourier transform is linear, by Lemma 1, $|f'[j] - f[j]| < 2^{-\frac{3}{2}}$ for all $0 \le j < n$. As the coefficients of f are integers, one can recover f by rounding f' to the nearest integer. □

In Section 3.4, we demonstrate that all the exponents of $\hat{f}_1$ and $\hat{f}_2$ in Falcon are less than 1031 with probability over 0.9 and less than 1032 with overwhelming probability. This implies that $|\hat{f}_1[j]|$ and $|\hat{f}_2[j]|$ are bounded by $2^{1031-1023} = 2^8$ and $2^{1032-1023} = 2^9$, respectively. By Theorem 1, the knowledge of 11 or 12 bits of the mantissas suffices to obtain f in the coefficient domain. Specifically, we may choose $h \ge 10$ for over 90% of f or $h \ge 11$ for over 99.999% of cases. It is important to note, as discussed at the beginning of this section, that the selection of h also depends on practical factors such as the quality and quantity of traces. Moreover, although our corrected proof yields a theoretically sound bound of h, we note that our result is a conservative lower bound. In practice, successful recovery may still be achieved with a lower h.

3.4 Range of Exponent

In [GMRR22], the authors demonstrate that the complexity of CPA is primarily limited by the recovery of the 11-bit exponent. In this section, we show that the range of candidate exponents can be further narrowed, reducing the number of exponent guesses from 2^{11} elements to no more than 20 elements. This reduction not only decreases the computational effort but also allows for a successful attack using fewer traces, as it mitigates the risk of false positives that exhibit high correlation.

3.4.1 Distribution of Coefficients of FFT(f)

We first show that we can model the distribution of the real and imaginary parts of each coefficient of $\mathsf{FFT}(f)$. Recall that the fast Fourier transform of a polynomial $f = f[0] + f[1]X + \cdots + f[n-1]X^{n-1}$ is

$$\hat{f} = (\hat{f}[0], \hat{f}[1], \cdots, \hat{f}[n-1]), \quad \hat{f}[k] = \sum_{m=0}^{n-1} f[m] \exp\left(\frac{i\pi \cdot (2k+1) \cdot m}{n}\right).$$

The real part of the coefficient $\hat{f}[k]$ is

$$\hat{f}_1[k] = \sum_{m=0}^{n-1} f[m] \cos\left(\frac{\pi \cdot (2k+1) \cdot m}{n}\right).$$

In Falcon, each $f[m]$ follows the discrete Gaussian distribution $D_{\mathbb{Z},\sigma_{\{f,g\}},0}$ with $\sigma_{\{f,g\}} = 1.17\sqrt{q/2n}$.

In the following, we use the continuous Gaussian distribution $\mathcal{N}(0, \sigma_{\{f,g\}})$, with mean 0 and standard deviation $\sigma_{\{f,g\}}$, to model each $f[m]$. Therefore, the random variable $G_{m,k} := f[m] \cdot \cos\left(\frac{\pi\cdot(2k+1)\cdot m}{n}\right)$ follows the Gaussian distribution $\mathcal{N}\left(0, \sigma_{\{f,g\}} \cdot \left|\cos\left(\frac{\pi\cdot(2k+1)\cdot m}{n}\right)\right|\right)$. Fix a coefficient $0 \le k < n$. Since $G_{m,k}$ for all m are identically and independently distributed, we have

$$\hat{f}_1[k] = \sum_{m=0}^{n-1} G_{m,k} \sim \mathcal{N}\left(0, \sigma_{\{f,g\}} \cdot \sqrt{\sum_{m=0}^{n-1} \cos^2\left(\frac{\pi \cdot (2k+1) \cdot m}{n}\right)}\right).$$

Moreover, $\sum_{m=0}^{n-1} \cos^2\left(\frac{\pi\cdot(2k+1)\cdot m}{n}\right) = \frac{n}{2}$. Therefore, each real-part coefficient of $\mathsf{FFT}(f)$ follows a Gaussian distribution:

$$\hat{f}_1[k] \sim \mathcal{N}(0, \sigma_{\{f,g\}} \cdot \sqrt{n/2}) = \mathcal{N}(0, 0.585 \cdot \sqrt{q}).$$

Similarly, the imaginary part follows the same distribution: $\hat{f}_2[k] \sim \mathcal{N}(0, 0.585 \cdot \sqrt{q})$.

Using the above result, we first examine the occurrence rate of various exponents of $\mathsf{FFT}(f)$. The results are given in Table 2.

Table 2: Occurrence probability (%) of exponents of Falcon's secret key f after the fast Fourier transform.

Exponent	<1013	1013	1014	1015	1016	1017	1018
Probability	0.001	0.001	0.002	0.004	0.009	0.019	0.038
Exponent	1019	1020	1021	1022	1023	1024	1025
Probability	0.076	0.153	0.307	0.615	1.230	2.457	4.899
Exponent	1026	1027	1028	1029	1030	1031	>1031
Probability	9.669	18.342	29.800	27.529	4.832	0.007	$3\cdot10^{-12}$

Despite a small fraction of exponents smaller than 1013 or larger than 1031, the probability of encountering an extreme exponent accumulates with n coefficients. To account for this, we apply the union bound to compute the occurrence probability of a

polynomial f where the coefficients of $\mathsf{FFT}(f)$ include at least one extreme exponent. That is,

$$\begin{aligned}
&\Pr\left[\exists\, 0 \le k < \frac{n}{2} : |\hat{f}_1[k]| < 2^{e-1023} \text{ or } |\hat{f}_2[k]| < 2^{e-1023}\right] \\
&\le \sum_{j=0}^{\frac{n}{2}} \Pr\left[|\hat{f}_1[j]| < 2^{e-1023} \text{ or } |\hat{f}_2[j]| < 2^{e-1023}\right] \\
&\le \sum_{j=0}^{\frac{n}{2}} \Pr\left[|\hat{f}_1[j]| < 2^{e-1023}\right] + \Pr\left[|\hat{f}_2[j]| < 2^{e-1023}\right] \\
&\le n \cdot \Pr[|\hat{f}_1[0]| < 2^{e-1023}].
\end{aligned}$$

Note that we only consider the first half of the coefficients since the second half are just their conjugates. This union upper bound of the probability that there is an extremely small or large exponent is given in Table 3.

Table 3: Upper bound of the occurrence probability (%) of Falcon's secret key f that contains at least one small (<1013 or <1016) or large (>1030 or >1031) exponent after the fast Fourier transform.

Exponent		<1013	<1016	>1030	>1031
Probability	$n = 512$	<0.616	<4.922	<4.043	$<1.5\cdot10^{-10}$
	$n = 1024$	<1.231	<9.843	<8.086	$<3 \cdot 10^{-10}$

For the lower bound, with probabilities of at most approximately 5% for $n = 512$ and 10% for $n = 1024$, at least one exponent is smaller than 1016. Therefore, we enumerate exponents starting from 1016 to cover around 95% and 90% of f for $n = 512$ and 1024, respectively. To handle around 99% of f, one can choose the lower bound 1013. However, we show in Section 3.4.2 that the exponents 1013, 1014, and 1015 can introduce false candidate exponents with high correlation, which reduces the accuracy of the attack and increases the number of traces required to recover the secret key. Therefore, by enumerating from 1016, we reduce the required number of traces at the cost of excluding a small fraction of f.

For the upper bound, we fix it at 1031 in all cases, as the probability of encountering an exponent exceeding this value is extremely small. Note that the upper bound also guides the selection of the number of high-order bits h in Section 3.3.1. While the conditions $h \ge 10$ and $h \ge 11$ are chosen based on whether any exponent exceeds 1030, we always enumerate the exponents up to 1031 when recovering the exponent, as adding a single candidate has little impact on the attack's runtime, but incrementing h by one bit roughly doubles the computation complexity.

3.4.2 Avoiding False Positives

In this section, we show that if the range of exponents includes 1013, 1014, and 1015, these values can yield high correlation when the true exponent is 1029, 1030, or 1031, respectively, and cause the attack to select the wrong candidates.

Table 4 presents the Hamming weights of the sums $e_f + e_c$, where $e_f \in [1013, 1015]$ represents the exponent of $\hat{f}$, and $e_c \in [1031, 1039]$ represents the exponent of $\hat{c}$. We also list the empirical occurrence probabilities of each value of e_c. For this, we generated 200,000 fresh samples of c whose coefficients are uniformly distributed over $[0, q)$, applied the fast Fourier transform, and counted the occurrences of each exponent value. The

probability is calculated by computing the average over all the coefficients and both real and imaginary parts.

Table 4: Occurrence probability (%) of exponents e_c of public hashed message c after the fast Fourier transform and Hamming weight of its sum with exponents e_f of Falcon's secret key f after the fast Fourier transform.

e_c		1031	1032	1033	1034	1035	1036	1037	1038	1039
Prob.		0.345	0.687	1.375	2.746	5.471	10.763	20.095	30.725	23.454
e_f	1013	9	9	10	1	2	2	3	2	3
	1029	4	4	5	2	3	3	4	3	4
e_f	1014	9	10	1	2	2	3	2	3	3
	1030	4	5	2	3	3	4	3	4	4
e_f	1015	10	1	2	2	3	2	3	3	4
	1031	5	2	3	3	4	3	4	4	5

The results show that the Hamming weights of $e_f + e_c$ for $e_f = 1013$, 1014, and 1015 often differ by 1 from those for $e_f = 1029$, 1030, 1031, respectively. However, for the values of e_c that can make the value $\mathsf{HW}(e_f + e_c) - \mathsf{HW}(e_f + 16 + e_c)$ different, their occurrence probabilities are small. For example, when $e_c > 1033$, $\mathsf{HW}(1013 + e_c)$ and $\mathsf{HW}(1029 + e_c)$ always differ by 1, and such values account for more than 95% of the cases. Since correlation is unaffected by adding a constant, a limited number of measurements with $e_c \leq 1033$ is insufficient to distinguish between $e_f = 1013$ and $e_f = 1029$.

In conclusion, we reduce the number of required traces by applying the lower bound 1016 to avoid false candidate exponents that result in similar correlation as the correct ones. This lower bound can cover all the exponents for most instances. If a sufficiently large number of traces is available, the lower bound may be extended to 1013 to ensure a successful recovery for 99% of instances.

3.4.3 Ignoring Small Exponents in Theory

In this section, we give a theoretical result to show that exponents less than 1019 can actually be disregarded without hindering the recovery of f, provided that the correlation of a correct candidate exponent and a false one is clearly distinguishable. In this case, the attacker can enumerate the exponents starting from 1019 and treat the exponent as 0 when no candidate with high correlation is found.

Let ℓ and h be two integers. The former defines the size of a small coefficient, whose absolute value is smaller than 2^ℓ, and h represents the number of high-order bits used in our attack. Let

$$\hat{f} = \hat{f}' + \epsilon = \hat{f}_0 + \hat{\xi}_1 + \hat{\xi}_2 i + \epsilon,$$

where

- $\hat{f}'$ is the approximation of $\hat{f}$ where the bits outside the $h+1$ most significant bits of the mantissas are set to 0.
- $\hat{\xi}_1$ (resp. $\hat{\xi}_2$) represents the small coefficients in the real part (resp. imaginary part) of $\hat{f}'$ whose absolute values are smaller than 2^ℓ.
- $\hat{f}_0$ is $\hat{f}'$ with the small coefficients replaced with 0.
- ϵ is the remaining error.

Let f', f_0, and ξ be the inverse fast Fourier transforms of $\hat{f}'$, $\hat{f}_0$, $\hat{\xi}_1 + \hat{\xi}_2 i$, respectively. By Section 3.3.1, for each coefficient k,

$$|f[k] - f'[k]| \leq 2^{9-h+\frac{1}{2}}.$$

Moreover, since the inverse fast Fourier transform is linear, $f' = f_0 + \xi$. Since each coefficient in $\hat{\xi}_1$ and $\hat{\xi}_2$ is smaller than 2^ℓ, each coefficient in $|\xi_1 + \xi_2 i|$ is smaller than $2^{\ell+\frac{1}{2}}$. By Lemma 1, for each coefficient k, $|f'[k] - f'_0[k]| = |\xi[k]| < 2^{\ell+\frac{1}{2}}$.

Now, if we select $h = 11$ and $\ell = -4$, we have

$$|f[k] - f'_0[k]| \leq |f[k] - f'[k]| + |f'[k] - f'_0[k]| < 2^{-\frac{3}{2}} + 2^{-\frac{7}{2}} < \frac{1}{2}.$$

By choosing the closest integer for each coefficient of f_0, we can recover f, whose coefficients are integers. In this case, we only need to consider candidates whose absolute values are at least 2^{-4}, or equivalently, exponents are at least $1023 - 4 = 1019$. If no exponent results in a correlation higher than a pre-defined threshold, we set the coefficient to 0. Alternatively, one can choose $h = 12$ and $\ell = -3$ or $h = 13$ and $\ell = -2$ to satisfy $2^{9-h+\frac{1}{2}} + 2^{\ell+\frac{1}{2}} < \frac{1}{2}$.

Depending on the choice of h, we can narrow the range of candidate exponents. However, it can be challenging to set the correlation threshold to determine whether an exponent should be ignored when the power traces contain notable noise. When the number of traces is sufficiently large and the correlation of a correct candidate is clearly distinguishable, one can still apply the result in this section.

4 Experiment and Evaluation

The experiments were carried out using ChipWhisperer-Lite [OC14] with an STM32F405 board as our attack target, which features an ARM Cortex-M4 core. We chose the masked floating-point multiplication implementation from [CC24] in our experiments. Each coefficient of f was drawn from the discrete Gaussian Distribution $\mathcal{D}_{\mathbb{Z},\sigma_{\{f,g\}},0}$ according to the scheme, and we employed the sampler given in the specification [PFH+22]. We designed two experiments that simulated different masking schemes, defined as follows:

(1) **Method 1:** Part of fpr_mul is masked. We followed the masked implementation of SecFprMul in [CC24, Algorithm 12] with the last subroutine SecFPR replaced with FPR. This implementation is intended to simulate the countermeasure in [KA24], where the packing and round phase is not masked.

(2) **Method 2:** The whole fpr_mul including FPR is masked. At the end of SecFprMul, we recombined the shares. This corresponds to the countermeasure in [CC24].

We clarify that our experimental setup does not strictly replicate the countermeasures in [CC24] or [KA24]. While [CC24] provides a full masking scheme for both multiplication and addition, and [KA24] is tailored for a hardware environment, we leverage these methods to instantiate a software-based implementation that masks only multiplications. These hybrid configurations allow us to simulate and evaluate the specific design trade-off discussed in Section 1, where addition masking is omitted for performance optimization.

For both methods, the experiment was divided into three phases: Trace Capturing, Analysis, and Recovery Phase.

4.1 Experiment Setup

Trace Capturing Phase The code was written in C and compiled using arm-none-eabi-gcc 10.3.1 under optimization for size -Os. In Method 1, the target board was programmed to

multiply two floating-point operands using 3-shared SecFprMul while the last subroutine was replaced with FPR. In Method 2, the target board was programmed with the 3-shared SecFprMul along with XORing the final result.

According to FALCON's specification, after hashing the message $c \leftarrow \mathsf{H}(\mathsf{r}\|\mathsf{m})$, each coefficient of c is in $\mathbb{Z}_q$. We generated each coefficient of c uniformly in the range $[0, q)$ and applied the fast Fourier transform to the FFT domain. c is generated as many times as N, the number of traces we captured. Next, both $\hat{c}_i$ and $\hat{f}_j$, where $i, j \in \{1, 2\}$ represent the real and imaginary parts, respectively, were sent to the target board to compute floating-point multiplications. As mentioned in the previous section, each coefficient induces 4 floating-point multiplications: $\hat{c}_1 \times \hat{f}_1, \hat{c}_2 \times \hat{f}_1, \hat{c}_1 \times \hat{f}_2$, and $\hat{c}_2 \times \hat{f}_2$.

Analysis Phase Both experiments used the same analysis technique detailed in Section 3. We set the number of high-order mantissa bits to $h = 12$ and considered two exponent ranges: $[e_\ell, e_u] = [1016, 1031]$ and $[1013, 1031]$. We adopted the method proposed in [GMRR22, Section 3.3] to combine the captured traces of $\hat{c}_1 \times \hat{f}_1$ and $\hat{c}_2 \times \hat{f}_1$ when attacking $\hat{f}_1$, and vice versa for $\hat{f}_2$. Hence, we were equivalently using $2N$ traces per coefficient. We only consider the first half coefficients of $\hat{f}$, as the remaining coefficients are their complex conjugates.

Recovery Phase Let $\hat{f}_{\mathsf{guess}}$ be the polynomial composed of all our guesses. For each coefficient of both real and imaginary parts, let the floating-point numbers of $\hat{f}$ and $\hat{f}_{\mathsf{guess}}$ be (s_f, e_f, m_f) and $(s_{\mathsf{guess}}, e_{\mathsf{guess}}, m_{\mathsf{guess}})$, respectively. We consider a successful attack on the real or imaginary part of a coefficient if all three components achieve the following criteria:

(1) $s_f = s_{\mathsf{guess}}$, the sign bit is recovered.

(2) $e_f = e_{\mathsf{guess}}$, the exponent part is recovered.

(3) $\dfrac{|m_f - m_{\mathsf{guess}}|}{|m_f|} < 2^{-11}$, the mantissa part is recovered. According to Section 3.3, this is the sufficient condition to recover f in the coefficient domain.

Next, we apply the inverse fast Fourier transform $\mathsf{invFFT}(\hat{f}_{\mathsf{guess}})$ to obtain our guess polynomial f_{guess}. We omitted the imaginary part after the inverse fast Fourier transform and rounded the real part to the nearest integer. The result is compared with f in the coefficient domain.

4.2 Evaluation Result

In Table 5, we show that the full secret key f in FALCON-512 can be successfully recovered using 1300 traces and 1200 traces for Method 1 and Method 2 respectively when the lower bound e_ℓ for the exponent is set to 1016. When we extended the range by lowering e_ℓ to 1013, which increases the attack's coverage, the number of traces required to recover the secret f escalated to 5500 and 5000 in Method 1 and Method 2, respectively, which is around a 4.2× increase.

To further analyze the performance of our attack with different numbers of traces, we define success rates of sign bit, exponent, and mantissa as the ratios of the number of correctly guessed coefficients to the total number of coefficients. Figure 2 illustrates the relation between the number of traces and the success rates for FALCON-512 in Method 1. With no more than 1000 traces, both the sign bit and mantissa achieve a 100% success rate, indicating that they were correctly recovered. However, the success rate of the exponent remains insufficient to recover the secret key f until the number of traces reaches 1300

Table 5: Number of traces required to successfully break FALCON-512 with different countermeasure methods and ranges of exponent.

Countermeasure	Method 1		Method 2	
Exponent Range	$[1016, 1031]$	$[1013, 1031]$	$[1016, 1031]$	$[1013, 1031]$
Trace Number	1300	5500	1200	5000

for $e_\ell = 1016$, where we recovered 1022 exponents among all 1024 real and imaginary coefficients, and all the exponents are fully recovered until 1700 traces. This shows that the overall success rate is mainly attributed to the exponent part of each coefficient. In addition, we only recovered 998 exponents with a success rate 97.47% for $e_\ell = 1013$ with 1700 traces. This is due to the presence of false-positive exponents being selected as the guesses. As the number of traces increases, these false positive exponents are ruled out, leading to recovering 1022 exponents and a successful key recovery with 5500 traces.

We also evaluate the results for FALCON-1024 in Figure 3, where success rate trends are similar to those of FALCON-512. With 4000 traces, our analysis yields a success rate of 98.1% (2008 coefficients) and 97.9% (2004 coefficients) for $e_\ell = 1016$ and $e_\ell = 1013$, respectively.

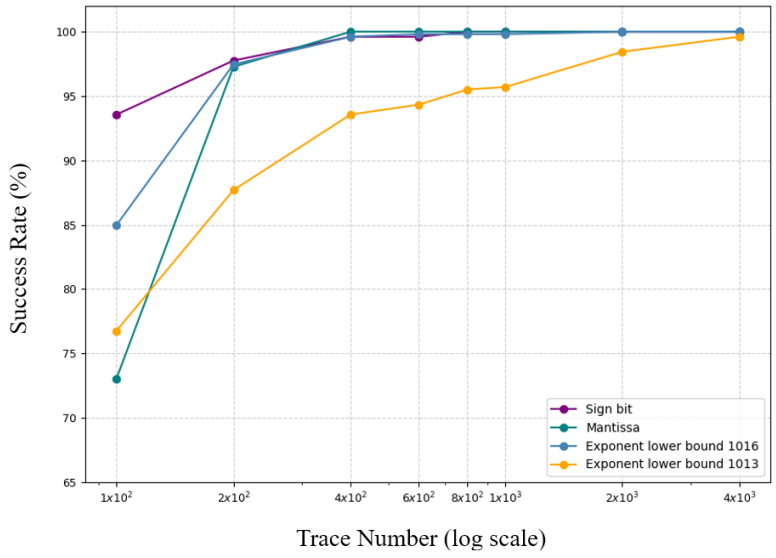

Figure 2: Success rate of sign bit, exponent, and mantissa in FALCON-512

5 Conclusion

In this work, we propose a novel correlation power analysis (CPA) attack on FALCON implementations that employ masking countermeasures for floating-point multiplication. We discover new sources of side-channel leakage and demonstrate how this leakage can be leveraged to recover the full secret key in implementations that apply only multiplication masking. Such implementations are considered because they are sufficient to resist previous attacks, complete masking is costly in existing works, and no hardware masking for floating-point addition has been proposed.

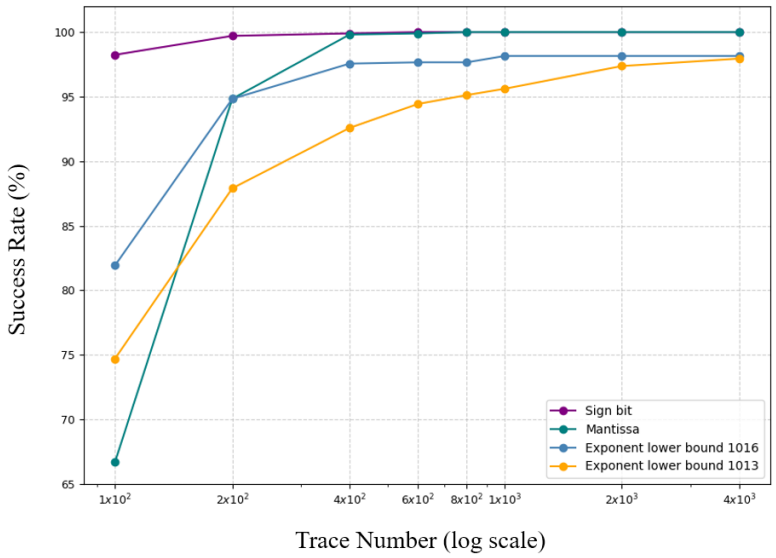

Figure 3: Success rate of sign bit, exponent, and mantissa in FALCON-1024

For side-channel attacks on the pre-image computation, we provide a detailed analysis of mantissa and exponent recovery in Section 3.3 and Section 3.4, respectively. We begin by identifying a flaw in the analysis of [GMRR22, Lemma 1], which claims that the knowledge of 6 bits of the mantissa is sufficient for reconstructing the secret key in the coefficient domain. We provide a counterexample and show in Theorem 1 that 11 and 12 bits are sufficient to recover over 90% and 99.999% of keys, respectively.

For exponent recovery, we model each coefficient of the secret key after the fast Fourier transform using a continuous Gaussian distribution. This approach allows us to reduce the number of candidate exponents from 2048, as considered in previous works, to no more than 20. We further show that narrowing the candidate range not only reduces the attack's runtime but also improves its accuracy. Specifically, exponents 1013, 1014, and 1015 can result in high correlation and be mistakenly interpreted as 1029, 1030, and 1031, respectively. Our experiments confirm that enumerating candidate exponents starting from 1016 significantly reduces the number of required traces for a successful attack by approximately a factor of 4.2 compared to starting from 1013.

While FALCON has been selected for standardization by NIST, its resistance to side-channel attacks has not been thoroughly investigated. Our results reveal the risks posed by improperly implemented designs. With an implementation that applies masking of floating-point addition, the leakage can be avoided, though at the cost of high overheads. Therefore, our work highlights the need for efficient and robust masking schemes for floating-point addition and other lightweight countermeasures to secure FALCON's pre-image computation against side-channel attacks.

Acknowledgements

We are supported by Academia Sinica Grand Challenge Project (AS-GCP-114-M01), and National Science and Technology Council grants 114-2221-E-001-014-MY3, 114-2634-F-001 -001 -MBK.

References

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Berlin, Heidelberg, August 2004.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 323–345. Springer, Berlin, Heidelberg, August 2016.

[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.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DP16] Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, pages 191–198, 2016.

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1857–1874. ACM Press, October / November 2017.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GMRR22] Morgane Guerreau, 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.

[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.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[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, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 53–71. Springer, Cham, 2020.

[Inc] NewAE Technology Inc. Chipwhisperer-pro (complete level 3 starter kit). https://store.newae.com/chipwhisperer-pro-complete-level-3-starter-kit/.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[KA22] Emre Karabulut and Aydin Aysu. Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks. In *Proceedings of the 58th Annual ACM/IEEE Design Automation Conference*, DAC '21, page 691–696. IEEE Press, 2022.

[KA24] Emre Karabulut and Aydin Aysu. Masking FALCON's floating-point multiplication in hardware. *IACR TCHES*, 2024(4):483–508, 2024.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[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. Cryptology ePrint Archive, Paper 2025/351, 2025.

[MHS+19] Sarah McCarthy, James Howe, Neil Smyth, Seamus Brannigan, and Máire O'Neill. BEARZ attack FALCON: Implementation attacks with countermeasures on the FALCON signature scheme. Cryptology ePrint Archive, Report 2019/478, 2019.

[OC14] Colin O'Flynn and Zhizhang David Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer, 2014.

[oSTa] National Institute of Standards and Technology. Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[oSTb] National Institute of Standards and Technology. Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms.

[oSTc] National Institute of Standards and Technology. Status report on the third round of the nist post-quantum cryptography standardization process. https://csrc.nist.gov/pubs/ir/8413/upd1/final.

[PBY17] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1843–1855. ACM Press, October / November 2017.

[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.

[QA25] Jinyi Qiu and Aydin Aysu. SHIFT SNARE: Uncovering secret keys in FALCON via single-trace analysis. Cryptology ePrint Archive, Paper 2025/146, 2025.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, oct 1997.

[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.

[ZSS20] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. Facct: Fast, compact, and constant-time discrete gaussian sampler over integers. *IEEE Trans. Comput.*, 69(1):126–137, January 2020.

A A Counterexample

We present a counterexample f to [GMRR22, Lemma 1] for $n = 512$. Preserving the sign bits, exponents, and 7 most significant bits of mantissas of $\mathsf{FFT}(f)$ results in a f' that cannot be rounded to f.

```
(-11,  28,  14,  24,  28,   4,  -8,  -8, -12, -26,  14,  29,
 -26,  26,  12,  10,  11, -12,  23,   8, -15,  30, -30,   7,
 -28, -10,  13, -12, -21, -30,  12,  -4, -10, -15, -19,  -4,
  13,  31, -22,  25,  21,  12, -30,   4,   8, -28,  18,  18,
 -30, -25, -24, -19, -29,   4,  28, -32,  21,  20,  28, -13,
  13, -11,  -4,  20, -17,  -1,  16,  24, -26,  19,  -6,  21,
 -17,  21,  16,  10, -28,   2,  21, -19, -16,  16,  15, -28,
 -26, -30,   2,  29,  -9, -28, -21,  28, -28,  -1, -20,   4,
  -4,   4,   1, -25, -21,  -7,  22,   2,  20, -28,  21, -31,
  18, -12, -27,  24, -13,  -1,  -8, -32,  -1, -31,  10,   2,
  24,  29, -29, -27,   5,  -5, -21, -16,  26, -25, -26,  31,
   8,  19,   2,  -2,  22, -17,  29,  31,  20,  31,   7, -12,
 -25,  28, -14,  25, -24,  26,  -8,  17,  14, -10,  -3, -26,
  30, -13, -21,  -5, -19, -24,   3, -22, -25, -13,  22,  29,
  24, -32,  19, -29,  -9, -27, -24,  23,  -6, -30,  25,   3,
  -4, -11,  13,   6,  16,   5,  11,   0,   9, -10, -15,  11,
  23,   4,  28, -14, -22, -26,  -4, -21,   2, -12,  27,   6,
  -2,   7, -16, -28, -28, -30,  23, -30,   5,  -9, -21,  17,
  21,  -1, -29,  14,   7,  27, -20,   0,  25, -18,  -4,  12,
 -19, -13, -20,   9,  -1,  24,  16,  21,  18, -17, -24, -19,
  12,   4,   3, -18, -32,  10,  -4,  21,  17, -16, -28, -23,
  16, -16,   2, -21, -21,   1, -26,  16, -11,  -4,  16,  16,
  18, -11,  24, -26,  28, -28,  15, -21,  20,  23,  19, -17,
 -24, -19,  20, -27,  23,  19,  -1,  19,  -5, -10,  -4, -17,
  10, -16,  18,  -1,  27,  -2, -23,  15,  -4, -13,   3, -24,
  12,  10, -14, -28,   9,   0,  -4, -12,   3,   8,  -8, -26,
  24,   1,  -7, -30, -15,   7,  28,  13, -19,  28,  28,   6,
  -6, -10,  20, -12,   4, -18,  25, -17, -31, -17,  -9, -28,
  -1, -22, -15,  20,  14,  13,  -8,  17,  -6, -31,  11,  12,
 -14,  28,   3,  24,  10,   1,  13, -10,  14,  19, -23,  -8,
 -21,  29, -18, -18,  26, -29, -30,  20,   9,  19,  -1,  18,
  14,  18,  12,  -4, -24,  15, -18, -32,  -2,  26, -31, -16,
  20,  22, -16,  -7,  29,  17, -17,  -5, -29,  25, -30, -22,
  24,   5, -19,  12,  -6, -26, -14,   6, -14,  -8,  -1, -22,
  27,   4,  31, -24,  28,  22,   8,  23, -15, -10, -16,  -3,
  27, -27,  21,  12,  26,  29, -10, -23,  -2,  -2,  12, -26,
   1,   8, -31,  -3,  22,   4,  31, -30,  -1, -23,  28, -10,
 -28,   0,  12, -28,  26,  -1,  -9,  14,  24, -14,  27, -18,
 -27,  -4, -30,   1, -24,  31, -16,  -4, -16, -20, -10, -21,
  23,  19,   1,   3,  17,  18,  -5, -23, -10,  -1,  20,   3,
  -7, -17,  24,  15,  17,  -5, -32,  17,  28, -22,  25,  -8,
  29, -20,   2,  22, -31, -17, -13,  11,  17, -12,  20,  23,
  15, -32,  24,  -6,  -4, -31,  30,   0)
```

Figure 4: A counterexample f of size $n = 512$.