Masked Computation of the Floor Function and Its Application to the FALCON Signature

Pierre-Augustin Berthet[1,3][0009−0005−5065−2730], Justine Paillet[2,3][0009−0009−6056−7766], and Cédric Tavernier[3][0009−0007−5224−492X]

[1] Télécom Paris, Palaiseau, France, berthet@telecom-paris.fr
[2] Université Jean-Monnet, Saint-Étienne, France, justine.paillet@univ-st-etienne.fr
[3] Hensoldt SAS FRANCE, Plaisir, France, <pierre-augustin.berthet,justine.paillet,cedric.tavernier>@hensoldt.net

Abstract. With the ongoing standardization of new Post Quantum Cryptography (PQC) primitives by the National Institute of Standards and Technology (NIST), it is important to investigate the robustness of new designs to Side Channel Analysis (SCA). Amongst those future standards is Falcon, a lattice-based signature which relies of rational numbers. It thus requires an implementation using floating point arithmetic, which is harder to design well and secure. While recent work proposed a solution to mask the addition and the multiplication, some roadblocks remains, most noticeably how to protect the floor function. In this work we propose several methods to protect the computation of the floor function. We provide mathematical proofs of our methods as well as formal security proof in the probing model using the Non-Interference concepts. We also discuss their application to the FALCON Signature.

Keywords: Floor Function · Floating-Point Arithmetic · Post-Quantum Cryptography · FALCON · Side-Channel Analysis · Masking

1 Introduction

With the rise of quantum computing, mathematical problems which were hard to solve with current technologies will be easier to breach. Amongst the concerned problem is the Discrete Logarithm Problem (DLP) which can be solved in polynomial times by the Shor quantum algorithm [23]. As much of the current asymmetric primitives rely on this problem and will be breach, new cryptographic primitves are studied. The National Institute of Standards and Technology (NIST) launched a post-quantum standardization process [6]. The finalists are CRYSTALS Kyber [4,18], CRYSTALS Dilithium [7,17], SPHINCS+ [2,19] and FALCON [21].

Another concern for the security of cryptographic primitives is their robustness to a Side-Channel opponent. Side-Channel Analysis (SCA) was first introduced by Paul Kocher [15] in the mid-1990. This new branch of cryptanalysis focuses on studying the impact of a cryptosystem on its surroundings. AS computations

take time and energy, an opponent able of accessing the variation of one or both could find correlations between its physical observations and the data manipulated, thus resulting in a leakage and a security breach. Thus, the study of weaknesses in implementations of new primitives and the ways to protect them is an active field of research.

While they have been many works focusing on CRYSTALS Dilithium and CRYSTALS Kyber, summed up by Ravi et al. [22], FALCON is noticeably harder to protect. Indeed, the algorithm relies on floating-point arithmetic, for which there is little litterature on how to protect it.

Related Work Previous works have identified two main weaknesses within the signing process of Falcon : the pre-image computation and the Gaussian sampler. The pre-image computation was proved vulnerable by Karabulut and Aysu [14] using an ElectroMagnetic (EM) attack. Their work was later improved by Guerreau et al. [11]. To counter those attacks, Chen and Chen [5] propose a masked implementation of the addition and multiplication of FALCON. However, they did not delved into the second weakness of Falcon, the Gaussian sampler.
The Gaussian sampler is vulnerable to timing attacks, as shown by previous work [10,8,16,20]. A isochronous design was proposed by Howe et al. [12]. However, a successful single power analysis (SPA) was proposed by Guerreau et al. [11] and further improved by Zhang et al. [24]. There is currently no masking countermeasure for FALCON's Gaussian Sampler. Existing work [9] tends to re-write the Gaussian Sampler to remove the use of floating arithmetic, thus avoiding the challenge of masking the floor function.

Our Contribution In this work we further expand the countermeasure from Chen and Chen [5] and apply it to the Gaussian Sampler. We propose a masking method based on the mantissa truncation to compute the floor function. We also propose a generic method to arithmetize the computation of the floor function. We discuss the application of both methods to masking FALCON.

Relying on the previous work of Chen and Chen [5], we also verify the higher-order security of our method in the probing model. Our formal proofs rely on the Non-Interference (NI) security model first introduced by Barthe et al. [1].

Finally, we provide some performances of our methods and compare them with the reference unmasked implementation and the previous work of Chen and Chen [5]. The implementation is tested on a personal computer with an Intel-Core i7-11850H CPU.

2 Notation and Background

2.1 Notation

– We denote by $\mathbb{R} \backsim \mathbb{Z}$ any real number which is not an integer. For $x \in \mathbb{R}$, we denote the floor function of x by $\lfloor x \rfloor$.

- We denote the floor function of x with n decimals of absolute precision by $\lfloor x \rfloor_n$. Absolute precision implies $\lfloor x \rfloor_n = \lfloor x \rfloor + 0, \{0\}^n \| [0;9]^*$. This also implies that $0 \leq \lfloor x \rfloor_n - \lfloor x \rfloor < 1 \times 10^{-n}$.
- We denote the floor function of x with n decimals of relative precision $\alpha \in [0,1;1)$ by $\lfloor x \rfloor_n^\alpha$. Relative precision implies $-\alpha \times 10^{-n} < \lfloor x \rfloor_n^\alpha - \lfloor x \rfloor < \alpha \times 10^{-n}$.
- Let $a, b \in \mathbb{R}^2$. Let $k \in \mathbb{N}$. We say that $a \equiv_k b$ if $\lfloor a \rfloor = \lfloor b \rfloor$ and they have the same number of 0 at least for the first k decimals: $\lfloor a \rfloor + 0, \{0\}^k \| [0;9]^*$. For instance, $4,05 \equiv_k 4,09$ at precisions $k = 0$ and $k = 1$ but not at precision $k = 2$.

2.2 FALCON Sign

FALCON [21] is a Lattice-Based signature using the GPV framework over the NTRU problem. In this paper we will focus on the Gaussian Sampler used in the signature algorithm. For more details on the key generation or the verification, please refer to the reference paper [21].

Signature The signature follows the Hash-Then-Sign strategy. The message m is salted with a random value r and then hashed into a challenge c. The remainder of the signature aims at building an instance of the SIS problem upon c and a public key h, *id est* finding $\boldsymbol{s} = (s_1, s_2)$ such as $s_1 + s_2 h = c$. To do so, the need to compute $\boldsymbol{s} = (\boldsymbol{t} - \boldsymbol{z})\mathbf{B}$, with $\boldsymbol{t}$ a pre-image vector and $\boldsymbol{z}$ provided by a Gaussian Sampler. Chen and Chen [5] focuses on masking the pre-image vector computation. In this work we intend to mask the Gaussian Sampler. The signature algorithm is detailled in Algorithm 1:

Algorithm 1: FALCON Sign [21]

Gaussian Sampler

2.3 Floor Function

The floor function is defined as follows:

Definition 1. *$\forall x \in \mathbb{R}$, the floor function of x, denoted by $\lfloor x \rfloor$, returns the greatest integer z such as $z \leq x$.*

There are several ways of computing the floor function. The first one relies on floating-point arithmetic. A floating-point is composed of a sign bit, exponent bits and a mantissa [13]. Computing the floor function on a floating-point is performed by truncating the mantissa according to the value of the exponent and of the sign. However, while this method is fast and efficient, it requires the use of the exponent and the sign. In this work we will present a method to perform this truncation in a secure manner.

Another way of computing the floor function is to use its associated Fourier series:

$$\forall x \in \mathbb{R} \backsim \mathbb{Z}, \lfloor x \rfloor = x - \frac{1}{2} + \frac{1}{\pi} \sum_{k=1}^{n} \frac{sin(2\pi kx)}{k}, \text{ with } n \to \infty \tag{1}$$

However, due to its discontinuities, Equation 1 suffers from the Gibbs phenomenon [3], as illustrated in Figure 2.3. It means this series cannot be used

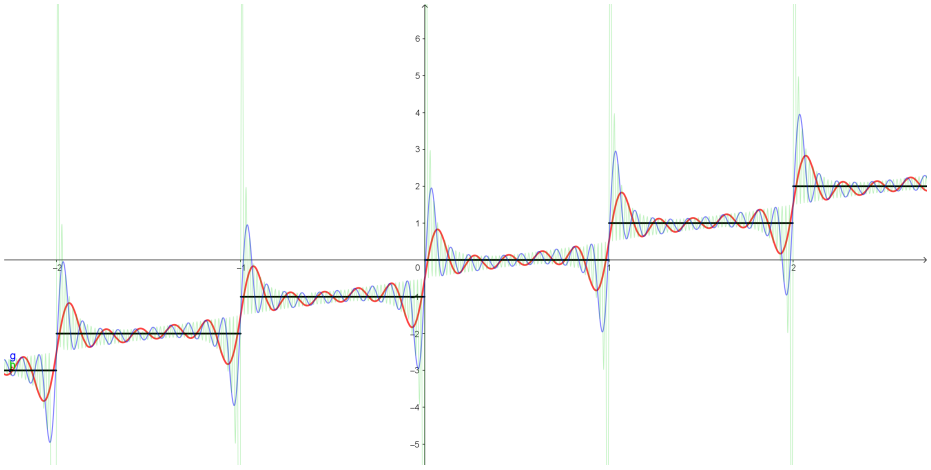

Fig. 1. Gibbs phenomenon for Equation 1 with n =5,10,50 and floor(x) in black

as it is to mask the floor function in reasonable time. Indeed, we extrapolate through empirical observations that we would require $n = 2.5 \times 10^{13}$ to have at least one decimal of absolute precision on the floor computation of 1×10^{-16}. Nonetheless, we will see in this work how we can adapt this Fourier series to compute the floor function in a secure way and in reasonable time with the required precision.

2.4 Masking

3 Masking of the Floor Function

3.1 Arithmetization method

The floor function can be arithmetized using Equation 1. However, as stated in Section 2.3, this method cannot be performed in reasonable time. We thus elaborate a strategy to optimize this arithmetization.

Absolute Precision The first idea relies on constructing the floor of $x \in \mathbb{R}$ rather than computing it. Indeed, if we can compute the floor of x with one decimal of precision, we can then multiply the result by 10 and then reapply our computation to the new value $y = 10 \times \lfloor x \rfloor_1$. Thus, we have the following proposition:

Proposition 1. *Let $x \in \mathbb{R}$. Let $y = 10 \times \lfloor x \rfloor_1$. Then $\lfloor y \rfloor_1 / 10 \equiv_2 \lfloor x \rfloor_2$.*

Proof. Let recall that $\lfloor x \rfloor_1 = \lfloor x \rfloor + 0,0 \| [0;9]^*$. Thus, $y = 10 \times \lfloor x \rfloor_1 = \lfloor x \rfloor \| 0 + 0,[0;9]^*$ as multiplying by 10 can be seen as a left-shift in base 10. By computing the floor of y with absolute precision 1 we have $y_1 = \lfloor y \rfloor_1 = \lfloor x \rfloor \| 0 + 0,0 \| [0;9]^*$. Finally, $y_1/10 = \lfloor x \rfloor + 0,00 \| [0;9]^* \equiv_2 \lfloor x \rfloor_2$ as dividing by 10 can be seen as a right-shift in base 10.

Thus, we can use Proposition 1 to construct the floor of x with the following Algorithm 2:

Algorithm 2: ConstructFloor(x,prec)

Data: $x \in \mathbb{R}$ and the precision $prec \in \mathbb{N}$
Result: $\lfloor x \rfloor_{prec}$

```
for i from 0 to prec-1 do
    y ← ⌊x⌋_1;
    x ← 10 × y;
return x × 10^{-prec}
```

Theorem 1. *Let $x \in \mathbb{R}$. Algorithm 2 returns $\lfloor x \rfloor_{prec}$ for precision prec.*

Proof. We use a recursive proof. The initial state is proven by Proposition 1. Let Theorem 1 be true for precision $prec - 1$. We have $X = \lfloor x \rfloor_{prec-1}$ the output of Algorithm 2. We reapply Algorithm 2 to $X \times 10^{prec-1}$ but with precision 1 and then divide by 10^{prec-1}. This is stricly equal to applying Algorithm 2 with precision *prec*. Algorithm 2 with precision 1 is proven by Proposition 1. Thus, we have $\lfloor X \rfloor_1 = \lfloor x \rfloor \| 0^{prec-1} + 0,0 \| [0;9]^*$. By dividing by 10^{prec-1}, we shift all the zeros stored in the integer part of X into the fractional part. Thus we have $\lfloor X \rfloor_1 / 10^{prec-1} = \lfloor x \rfloor + 0,0^{prec} \| [0;9]^* \equiv_{prec} \lfloor x \rfloor_{prec}$. We thus proved Theorem 1.

Relative Precision We discuss how to achieve absolute precision from relative precision. In relative precision, we have the possibility that the error from the result can be negative. This results in $\lfloor x \rfloor_n^\alpha = \lfloor x \rfloor - 0,\{0\}^n \| [0;9]^* = \lfloor x \rfloor - 1 + 0,\{9\}^n \| [0;9]^*$. Thus, we cannot construct the floor function using directly $\lfloor x \rfloor_n^\alpha$, as in Algorithm 2. We introduce a second idea: controlled bias.

We have the following Proposition 2:

Proposition 2. *Let $x \in \mathbb{R}$. Then $\lfloor x \rfloor_1 \equiv_1 \lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2}$.*

Proof. We have two cases to cover:

1. The error is negative. Thus $-\alpha \times 10^{-2} < \lfloor x \rfloor_2^\alpha - \lfloor x \rfloor \leq 0$ and $0 < \lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} - \lfloor x \rfloor \leq \alpha \times 10^{-2} < 1 \times 10^{-2}$. Thus, in this case, we have $\lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} \equiv_2 \lfloor x \rfloor_2$ and by construction of $\lfloor x \rfloor_n$, we have $\lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} \equiv_1 \lfloor x \rfloor_1$.
2. The error is positive. Thus $0 \leq \lfloor x \rfloor_2^\alpha - \lfloor x \rfloor < \alpha \times 10^{-2}$ and $\alpha \times 10^{-2} \leq \lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} - \lfloor x \rfloor \leq 2\alpha \times 10^{-2}$. This implies $0 < \lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} - \lfloor x \rfloor < 1 \times 10^{-1}$. Thus, in this case, we have $\lfloor x \rfloor_2^\alpha + \alpha \times 10^{-2} \equiv_1 \lfloor x \rfloor_1$.

We demonstrate that we can have an absolute precision of 1 from a relative precision of 2.

However, we can improve upon Proposition 2 by limiting the maximum value of α. We have the following proposition:

Proposition 3. *Let $x \in \mathbb{R}$ and $\alpha < 0.5$. Then $\lfloor x \rfloor_1 \equiv_1 \lfloor x \rfloor_1^\alpha + \alpha \times 10^{-1}$.*

Proof. We have already proven in Proposition 2 that for the negative bias, $\lfloor x \rfloor_1^\alpha + \alpha \times 10^{-1} - \lfloor x \rfloor \leq \alpha \times 10^{-1}$. As $\alpha < 0.5$, we have $\lfloor x \rfloor_1 \equiv_1 \lfloor x \rfloor_1^\alpha$. For the postive bias, according to Proposition 2 we have $\alpha \times 10^{-1} \leq \lfloor x \rfloor_1^\alpha + \alpha \times 10^{-1} - \lfloor x \rfloor \leq 2\alpha \times 10^{-1}$. As $\alpha < 0.5$, $2\alpha \times 10^{-1} < 1 \times 10^{-1}$. We have $\lfloor x \rfloor_1 \equiv_1 \lfloor x \rfloor_1^\alpha$.

Fourier Series We now need a way of computing the floor function with a relative precision of at least 2 in reasonable time. Our final idea revolves around aborting Equation 1 early for a small value of n. Then, we reapply Equation 1 on the new value, with a small twist:

Algorithm 3: ArFloorStep1(x,n,iter)

Data: $x \in \mathbb{R}$, the series limiter $n \in \mathbb{N}$ and the number of iterations $iter \in \mathbb{N}$
Result: $\lfloor x \rfloor_2^\alpha + \frac{1}{2}$

```
for i from 0 to iter-1 do
    x ← x + (1/π) Σ_{k=1}^{n} sin(2πkx)/k;
return x
```

Instead of computing the floor function, Algorithm 3 gives the floor function with a positive bias of $\frac{1}{2}$. Indeed, applying Equation 1 and aborting without this bias does not unsure correctness. For instance, for a value close to 0, using Equation 1 with a small n results in $-\frac{1}{2} + e$, with e small. Reapplying Equation 1 will give $-\frac{1}{2} + e - \frac{1}{2} + e_2 \approx -1$ as a result, which is not correct. Thus, it is important to remove $-\frac{1}{2}$ and only apply it after the successive early aborts and reapplications. As an example, Algorithm 3 return is proven by Theorem 2 for a specific set of parameters:

Theorem 2. *Let $x \in (0;1) \subset \mathbb{R} \backsim \mathbb{Z}$ with up to 19 decimals,* id est *$x = 0,[0;9]^{19}\|0^*$. We set $n = 3$ and $\alpha = 0.4$. Algorithm 3 returns $\lfloor x \rfloor_1^\alpha + \frac{1}{2}$ in 24 iterations.*

Proof. We denote $x + \frac{1}{\pi} \sum_{k=1}^{3} \frac{sin(2\pi kx)}{k}$ by $f(x)$. Let set $(U_n)_{n \in \mathbb{N}}$ such as $U_{n+1} = f(U_n)$. First, we will prove the convergence of $(U_n)_{n \in \mathbb{N}}$. Then, we use the Newton method and a visual proof to demonstrate Theorem 2.

PREUVE DE LA CONVERGENCE A FAIRE!!!!!!

For the second part of the proof, we use the Newton method on a graphical representation of $f(x)$(labeled Eq). The Newton method, in grey arrows in Figure , allows to graphically compute the next term U_{n+1} from U_n. We start at the point $(U_n, 0)$ (0.15 in this example) and find the point $(U_n, f(U_n))$ (arrow labeled u). We then find the point of $c(x) = x$ with ordinate $f(U_n)$ (arrow labeled v) and look at its abscissa (arrow labeled w), which gives us U_{n+1}. Thanks to this method, we can visually compute the number of iterations of Algorithm 3 required to have $0.46 \leq U_{n+1} \leq 0.54$(green area). We have the following Table 1:

Table 1. Subsets of $(0; 1)$ and number of iterations to reach $[0.46; 0.54]$

Number of iterations	Subsets
1	
At most 2	
At most 3	
4 or more	

Remark 1. The choice of setting $\alpha = 0.4$ in Theorem 2 is not the most optimal as taking $\alpha = 0.5$ is more advantageous as shown in Proposition 3. However, by taking $\alpha = 0.4$, we have that Algorithm 3 returns $\lfloor x \rfloor + err$ such as $0.46 \leq err \leq 0.54$. Thus, by retrieving to this result 0.45, we have $0.01 \leq err - 0.45 \leq 0.09$. By multiplying by 10, we have $0.1 \leq 10 * (err - 0.45) \leq 0.9$. As a result, when applying Algorithm 2 we avoid some extreme points and reduce the number of iterations required by Algorithm 3. This strategy is not specific to the current set of parameters and will always benefit our method.

3.2 Truncation method

4 Application to FALCON

4.1 Masked Floor Function For FALCON

4.2 Masking the Gaussian Sampler

5 Performances

6 Conclusion

Acknowlegdments

References

1. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 116–129. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978427, https://doi.org/10.1145/2976749.2978427
2. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The sphincs+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2129–2146. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363229, https://doi.org/10.1145/3319535.3363229
3. Bocher, M.: Introduction to the theory of fourier's series. Annals of Mathematics **7**(3), 81–152 (1906), http://www.jstor.org/stable/1967238
4. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: Crystals - kyber: A cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (April 2018). https://doi.org/10.1109/EuroSP.2018.00032
5. Chen, K.Y., Chen, J.P.: 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**(2), 276–303 (Mar 2024). https://doi.org/10.46586/tches.v2024.i2.276-303, https://tches.iacr.org/index.php/TCHES/article/view/11428
6. Chen, L., Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R.A., Smith-Tone, D.: Report on post-quantum cryptography, vol. 12. US Department of Commerce, National Institute of Standards and Technology ... (2016)
7. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1), 238–268 (Feb 2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://tches.iacr.org/index.php/TCHES/article/view/839
8. Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Side-channel attacks on bliss lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 1857–1874. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134028, https://doi.org/10.1145/3133956.3134028
9. Espitau, T., Fouque, P.A., Gérard, F., Rossi, M., Takahashi, A., Tibouchi, M., Wallet, A., Yu, Y.: Mitaka: A simpler, parallelizable, maskable variant of falcon. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 222–253. Springer International Publishing, Cham (2022)
10. Groot Bruinderink, L., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload – a cache attack on the bliss lattice-based signature scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016. pp. 323–345. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
11. Guerreau, M., Martinelli, A., Ricosset, T., Rossi, M.: The hidden parallelepiped is back again: Power analysis attacks on falcon. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(3), 141–164 (Jun 2022). https://doi.org/10.46586/tches.v2022.i3.141-164, https://tches.iacr.org/index.php/TCHES/article/view/9697

12. Howe, J., Prest, T., Ricosset, T., Rossi, M.: Isochronous gaussian sampling: From inception to implementation. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography. pp. 53–71. Springer International Publishing, Cham (2020)
13. Kahan, W.: Ieee standard 754 for binary floating-point arithmetic. Lecture Notes on the Status of IEEE **754**(94720-1776), 11 (1996)
14. Karabulut, E., Aysu, A.: Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks. In: 2021 58th ACM/IEEE Design Automation Conference (DAC). pp. 691–696 (Dec 2021). https://doi.org/10.1109/DAC18074.2021.9586131
15. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology — CRYPTO '96. pp. 104–113. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
16. McCarthy, S., Howe, J., Smyth, N., Brannigan, S., O'Neill, M.: Bearz attack falcon: Implementation attacks with countermeasures on the falcon signature scheme. Cryptology ePrint Archive, Paper 2019/478 (2019), https://eprint.iacr.org/2019/478, https://eprint.iacr.org/2019/478
17. NIST: Module-lattice-based digital signature standard. NIST FIPS (2024). https://doi.org/10.6028/NIST.FIPS.204.ipd
18. NIST: Module-lattice-based key-encapsulation mechanism standard. NIST FIPS (2024). https://doi.org/10.6028/NIST.FIPS.203.ipd
19. NIST: Stateless hash-based digital signature standard. NIST FIPS (2024). https://doi.org/10.6028/NIST.FIPS.205.ipd
20. Pessl, P., Bruinderink, L.G., Yarom, Y.: To bliss-b or not to be: Attacking strongswan's implementation of post-quantum signatures. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 1843–1855. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134023, https://doi.org/10.1145/3133956.3134023
21. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon. Post-Quantum Cryptography Project of NIST (2020)
22. Ravi, P., Chattopadhyay, A., D'Anvers, J.P., Baksi, A.: Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. ACM Trans. Embed. Comput. Syst. **23**(2) (mar 2024). https://doi.org/10.1145/3603170, https://doi.org/10.1145/3603170
23. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review **41**(2), 303–332 (1999). https://doi.org/10.1137/S0036144598347011, https://doi.org/10.1137/S0036144598347011
24. Zhang, S., Lin, X., Yu, Y., Wang, W.: Improved power analysis attacks on falcon. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 565–595. Springer Nature Switzerland, Cham (2023)