FALCON Down: Breaking FALCON Post-Quantum Signature Scheme through Side-Channel Attacks

Emre Karabulut
Department of Electrical and Computer Engineering
North Carolina State University
NC, USA
ekarabu@ncsu.edu

Aydin Aysu
Department of Electrical and Computer Engineering
North Carolina State University
NC, USA
aaysu@ncsu.edu

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586131

***Abstract*—This paper proposes the first side-channel attack on FALCON—a NIST Round-3 finalist for the post-quantum digital signature standard. We demonstrate a known-plaintext attack that uses the electromagnetic measurements of the device to extract the secret signing keys, which then can be used to forge signatures on arbitrary messages. The proposed attack targets the unique floating-point multiplications within FALCON's Fast Fourier Transform through a novel extend-and-prune strategy that extracts the sign, mantissa, and exponent variables without false positives. The extracted floating-point values are then mapped back to the secret key's coefficients. Our attack, notably, does not require pre-characterizing the power profile of the target device or crafting special inputs. Instead, the statistical differences on obtained traces are sufficient to successfully execute our proposed differential electromagnetic analysis. The results on an ARM-Cortex-M4 running the FALCON NIST's reference software show that approximately 10k measurements are sufficient to extract the entire key.**

***Index Terms*—side-channel attacks, post-quantum cryptography, digital signatures**

I. Introduction

Advances in computing technology have a drastic effect on code breaking. Just like how Bombe, the first electromechanical computer, has solved the infamous Enigma cipher, the first practical quantum computer can crack today's encryption schemes. Indeed, it is well known that quantum algorithms offer exponential speedup on solving integer factorization [1] and (elliptic curve) discrete logarithm [2] problems that existing public-key systems rely on. Post-quantum cryptography, therefore, seeks alternative classical algorithms that can resist quantum cryptanalysis. The growing concern of the quantum threat has motivated the National Institute of Standards and Technology (NIST) to solicit and evaluate applications for a post-quantum cryptography standard, which is an ongoing process envisioned to be complete by 2023.

Although algorithms can be mathematically-sound against classical or quantum cryptanalysis, their implementations may leak secret information through side-channels [3]. These attacks find a correlation between secret values and implementation behaviors such as execution time, power consumption, and electromagnetic (EM) radiation. Among these attacks, physical side-channels (like the EM leakage) are of particular importance since they exist not because of bad design choices per se but due to the physics of data-dependent CMOS activity. These attacks can succeed with only a few tests obtained from the physical device and without needing any functional quantum computer. Side-channel attacks are important for NIST as well, as they are a criterion for determining the ultimate standard [4].

We propose the *first* side-channel attack on FALCON—a NIST Round-3 finalist for the post-quantum digital signature standard [5]. Although FALCON is one of the three finalists, no side-channel attacks have been published on it. Existing attacks on the other two finalists [6]–[8] or similar lattice cryptosystems [9]–[17] do not trivially extend to FALCON's unique computations. Specifically, FALCON applies a Fast Fourier Transform (FFT) over floating-point numbers rather than using the popular Number Theoretic Transform (NTT), whose side-channels have been thoroughly evaluated [18], [19]. We first show that a straightforward attack on the floating-point variables (and particularly the mantissa part) suffers from false positives due to multiplication. But we then propose a novel *extend-and-prune* strategy that resolves false positives by taking the intermediate additions into account.

We execute the proposed attack on the EM traces obtained from the FALCON reference software implementation running on the ARM-Cortex-M4 microcontroller. Notably, our attack does not require a pre-characterization of the target device under different keys, i.e., works without templates [20]. The results show that, on average, the targeted floating-point variables can be captured with over 99.99% probability with around 10k measurements. Upon extracting the targeted intermediate variables and reverting back to the prior steps of FALCON, we show that the adversary can recover the entire secret key and successfully sign arbitrary messages.

The contributions of this paper are as follows.

- We propose the first side-channel attack on NIST's Round-3 post-quantum digital signature standard finalist FALCON. We analyze the algorithm, reveal the vulnerable computations that can leak information and that leakage can cause forging signatures on arbitrary messages.
- We show that a straightforward attack on the targeted computations fails due to multiplication false positives and introduce a novel attack that can resolve false guesses through an extend-and-prune strategy.
- We apply the proposed attack on the reference software of FALCON taken from NIST's website and demonstrate that the proposed attack can extract the entire signing keys with a few thousand measurements when FALCON runs on an ARM-Cortex-M4 microcontroller.

The rest of the paper is organized as follows. Section II

Algorithm 1 FALCON Key Generation Algorithm [5]

Input: A monic polynomial $\phi \in \mathbb{Z}[x]$, a modulus q
Output: A secret key sk and a public key h

$f, g, F, G \leftarrow$ NTRUGen(ϕ, q)
$B \leftarrow \begin{bmatrix} g & -f \\ G & F \end{bmatrix}$
$\hat{B} \leftarrow FFT(B)$
$G \leftarrow \hat{B} \times \hat{B}^*$ ▷ $\times$ represents matrix multiplication
$T \leftarrow$ ffLDL$^*(G)$
for each leaf of T **do**
    $leaf.value \leftarrow \sigma / \sqrt{leaf.value}$
end for
$sk \leftarrow (\hat{B}, T)$
$h \leftarrow gf^{-1} mod(q)$
return sk, h

Algorithm 2 FALCON Signature Generation Algorithm [5]

Input: a message m, a secret key sk, a bound β^2
Output: a signature sig of m

$r \leftarrow \{0,1\}^{320}$ uniformly
$c \leftarrow$ HashToPoint $(r||m)$
$t \leftarrow (\frac{-1}{q} FFT(c) \odot FFT(F), \frac{1}{q} FFT(c) \odot FFT(f))$
do ▷ $\odot$ represents FFT multiplication
    do
        $z \leftarrow$ ffSampling (t, T)
        $\mathbf{s} \leftarrow (t - z) \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix}$
    while $s^2 > [\beta^2]$
    $(s_1, s_2) \leftarrow invFFT(s)$
    $s \leftarrow$ Compress$(s_2, 8 \cdot sbytelen - 328)$
while $s = \perp$
return $sig = (r, s)$

provides a background on the FALCON signature scheme and the underlying computations. Section III describes the proposed attack. Section IV shows the attack results. Section V discusses possible countermeasures along with other related aspects, and Section VI concludes the paper.

II. Background

This section provides background information about the FALCON digital signature algorithm and the threat model.

A. Threat Model

Our work follows the standard assumptions in EM side-channel attacks where the adversary has physical access to the device and captures EM measurements while the key-dependent computations are carried out [21]. Two notable advantages of our attack over some recent work on lattice cryptography side-channels is not requiring to craft special inputs [15], [22] or another source of vulnerability such as a timing side-channel [23] to extract the secret information.

B. The Falcon Post-Quantum Digital Signature Scheme

FALCON is a post-quantum, lattice-based, hash-and-sign signature scheme [5]. FALCON consists of key generation, signing, and signature verification procedures. Here we summarize the first two procedures since they are crucial to understand our attack. We refer the interested reader to the official FALCON specification document for details [5].

Algorithm 1 shows the key generation procedure that produces the secret key sk for creating and the public key h for verifying signatures. The inputs of the algorithm are the parameters ϕ and q: all operations occur over an n degree monic polynomial ϕ that is $x^n + 1$ for the binary case and $x^n - x^{n/2} + 1$ for the ternary case, while q is the modulus prime number. This algorithm first randomly samples the coefficients of the polynomials f and $g \in \mathbb{Z}[x]$ from a discrete Gaussian distribution and then computes F and $G \in \mathbb{Z}[x]$ that satisfy the NTRU equation $fG - gF = q \mod \phi$. The polynomials f, g, F, and G are called private elements. These polynomials are then combined, passed through the FFT, and converted into the full-rank Gram matrix G. To compute the binary tree T, FALCON applies an LDL decomposition on G. The key generation algorithm returns the public key h that satisfies the equation of $gf^{-1} = h$ and the secret key sk has two components $\hat{B}$ and T that are derived from four polynomials $f, g, F, G \in \mathbb{Z}[x]$. Thus, this key generation hardness is based on the quantum-resilient NTRU problem that relies on the difficulty of recovering f and g polynomials given the polynomial ring element h. The polynomial coefficients of these private polynomials f and g has a range of -127 to +127 integer values.

Algorithm 2 illustrates FALCON's signing procedure, which takes in a message m, the signing key sk, and a bound to check the validity of the signature, and returns a signature with two components: r and s. The algorithm first hashes the given message m with a uniformly random salt r to compute c. Then, the hashed value c goes to the FFT domain and gets multiplied private elements f and F to obtain short vectors s_1, s_2 where $s_1 + s_2 h = c$. Lastly, the vector s_2 is encoded (compressed) to generate the signature bitstring s.

C. FFT over Floating-Point Numbers

FALCON argues that it uses trapdoor samplers and thus needs to operate with floating-point arithmetic and with FFT rather than integer arithmetic and NTT [5]. FALCON approximates the floating-point numbers used in arithmetic operations similar to the IEEE 754 floating-point (double precision) standard. This approximation represents a floating-point number with 64 bits where the MSB is the sign bit, the following 11 bits are exponent and the rest 52 bits are mantissa. FALCON requires the floating-point arithmetic during signing and key generation steps we show in Algorihtms 1 and 2.

FALCON speeds up the multiplication of ring polynomials with FFT, which operates over the ring $\mathbb{Z}_q/\phi(x)$ where $\phi(x)$ is a monic reduction polynomial. FFT reduces the time-complexity by transforming polynomials in $\mathbb{Z}_q/\phi(x)$ to another domain where polynomial multiplication becomes a coefficient-wise (scalar) multiplication. FALCON's signing

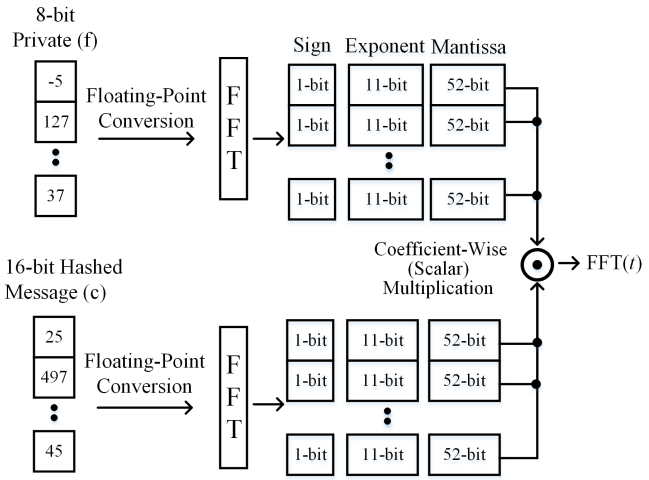

Fig. 1. The multiplication of the private element f and hashed message polynomial c. The attack targets the coefficient-wise (scalar) floating-point multiplications at the FFT domain.

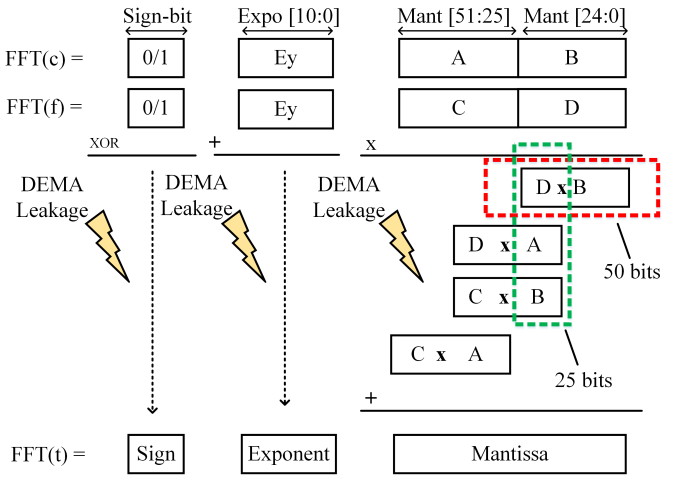

Fig. 2. The proposed attack on FALCON's floating-point multiplications. Targeting multiplications (shown in red dashed lines) creates false positives and targeting intermediate additions (shown in greed dashed lines) eliminates them.

procedure first converts the hashed message and the coefficients of private key elements (f, g, F, G) to the floating-point numbers, then applies the FFT domain transformation on them (see Algorithm 2, line 3). Therefore, the FFT algorithm converts the 8-bit integer coefficients of the private key elements to 64-bit floating-point coefficients. FFT algorithm applies floating-point addition, subtraction, and multiplication between the coefficients of the input polynomial. After passing to the FFT domain, FALCON signing algorithm performs a coefficient-wise (scalar) floating-point multiplication between the private key elements f and F and the hashed message output c.

III. The Proposed Side-Channel Attack

This section presents the proposed attack on FALCON digital signature algorithm and the related challenges. First, we describe the intermediate computation we target and why recovering that variable enables recovering secret keys and forging signatures. We then introduce the challenges of executing the side-channel attack on the targeted computation and how we addressed those challenges.

A. The Targeted Operation $FFT(c) \odot FFT(f)$ and Rationale

Our attack targets the floating-point multiplication within the computations of $FFT(c) \odot FFT(f)$ (see Algorithm 2, line 3). We argue that (i) a known-plaintext attack on these computations is possible and (ii) that capturing the coefficients of $FFT(f)$ with a side-channel attack enables the adversary to sign arbitrary messages.

FALCON's private elements consist of polynomials f, g, F, and G. These polynomials are used to compute the secret signing key components T and $\hat{B}$ (see Algorithm 2). As described in Section II, polynomials F and G form an NTRU equation with f and g; hence, if the adversary knows the polynomials f and g, it can compute F and G, derive the entire secret key, and successfully sign arbitrary messages. Since the public key h also is the product of gf^{-1}, the adversary needs to extract *either* the polynomial f or g to perform the successful attack. To that end, we target the operation $FFT(c) \odot FFT(f)$ (see Algorithm 2, step 3), where the FFT-based polynomial multiplication between the hashed message c and the private polynomial element of f occurs. Targeting this computation with a side-channel attack is feasible because c is known to the adversary and thus the secret f can be hypothesized and checked through the leakage.

Figure 1 shows the details of the targeted FFT-based multiplication between the hashed message c and the private element f. FALCON reference implementation first converts the integer coefficients of the polynomials c and f to floating-point. Then, FALCON applies an FFT transformation over these floating-point coefficients. After the FFT conversion, FALCON performs a coefficeint-wise (scalar) floating-point multiplication between the 64-bit coefficients of $FFT(c)$ and $FFT(f)$. Since the uniformly random salt r and the message m are public, $FFT(c)$ can be computed (and thus known) by the adversary. The proposed attack, therefore, focuses on the floating-point multiplications between the coefficients of the known $FFT(c)$ and secret $FFT(f)$. If the adversary successfully extracts the coefficients of the polynomial $FFT(f)$, it can recover the private element f because FALCON's FFT function is reversible and one-to-one.

We do not claim that side-channel vulnerabilities are exclusive to the floating-point multiplication within the FALCON signing algorithm. But we do claim that our proposed attack is sufficient to extract the signing keys. Other parts of the algorithm such as the key generation steps may also leak information.

B. Retrieval of $FFT(f)$ via Divide-and-Conquer

Figure 2 shows FALCON's floating-point multiplication steps on two coefficients. The multiplication operation takes in two 64-bit coefficients and generates a 64-bit output. The operation consists of three parts: mantissa multiplication, exponent addition and sign bit computation.

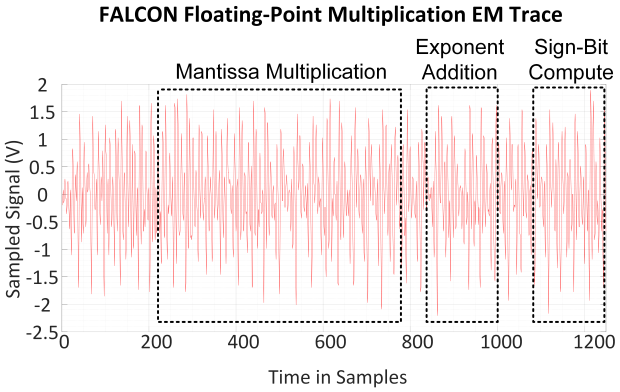

Fig. 3. An example EM measurement trace from our experiment showing the related mantissa, exponent, and sign computations.

The mantissa multiplication has four steps. First, it concatenates 52-bit mantissa of the input coefficients with a single bit '1' to make it the most-significant-bit (MSB). Second, it splits the mantissa bits as the higher-order 28 bits and the lower-order 25 bits. Third, it applies a straightforward schoolbook integer multiplication on the split mantissa bits of the two coefficients. Due to the splitting, four addition operations occur within the schoolbook multiplication, which would generate 106-bit products. Since mantissa size has to be ultimately 52 bits, the bottom bits are called the unused, sticky bits. The fourth step, therefore, is to round the multiplication product to 52 bits by removing those sticky bits.

FALCON applies the exponent addition on the 11-bit exponents of the input coefficients. Exponent addition also gets an extra carry bit from the mantissa multiplication result. The final operation is calculating the sign bit, which is an XOR operation between the MSB bits of the input coefficients.

We perform a divide-and-conquer strategy to attack FALCON's floating-point multiplication. Our attack method separately recovers mantissa, exponent, and sign bits, and it combines them to obtain the coefficients of $FFT(f)$. Without such a strategy, a straightforward differential attack would need to create massive tables having 2^{64} entries for each coefficient. Figure 3 shows a captured EM trace of the targeted floating-point multiplication between two coefficients. The red line is the EM signal and the black dash lines annotate during which time samples mantissa, exponent and sign operations are performed.

The key challenge in achieving this attack is eliminating the false positives that occur on the mantissa multiplication. Indeed, multiplication is known to generate false positives because the results are correlated, i.e., similar coefficients will produce a similar multiplication output. For example, the $FFT(f)$ coefficient '1' and '2' will generate the same values shifted by one binary digit. This is handled by our novel extend-and-prune technique.

C. Extend-and-Prune Technique to Remove False Positives

We address the key challenge of eliminating the multiplication false positive through a novel extend-and-prune technique. The crux of this technique is to use the top guesses obtained on attacking the multiplication (extend phase) and to evaluate their correctness by attacking the next operation (prune phase), which is the addition of intermediate products (see Figure 2).

Although the polynomial f coefficients are defined as integer numbers ranging between -127 and 127, these are converted to floating-point numbers and then passed to the FFT domain. Since the FFT algorithm mixes and diffuses all the input coefficients and since FFT performs floating-point arithmetic, the FFT-domain coefficients have the range of $[0, 2^{64}]$, even though the inputs have the range of $[-127, 127]$.

Our extend-and-prune is tuned for the implementation of FALCON floating-point arithmetic. For extracting the mantissa part, we first attack the lower-order 25 bits of the multiplication—figure 2 shows the two lower-order 25 bits as B and D where B is known and D is the secret value. Our differential attack on D×B multiplication follows the usual steps: creating hypothesis guesses on a secret lower-order 25 bits (D), computing the expected switching activity for each multiplication (using Hamming weight), and checking correlations between them and the corresponding EM measurements. We also repeat the same procedure on the D×A, where A is known and D is the secret value. This is the extend phase of our attack and is expected to generate false positives.

The second step in our attack is the prune phase, where we target the addition of D×B and D×A to prune false positive values and to recover the 25 bits of the secret mantissa. Unlike multiplication, the addition will *not* generate false positives: for example, the same coefficients of '1' vs. '2' generates results having different Hamming weights based on the other input of the addition. With sufficient many tests, the secret coefficients of '1' vs. '2' (and other cases that generate a false positive on the multiplication) can be differentiated from one another.

We follow the same aforementioned procedure to extract the D over addition operation. Higher-order bits of the mantissa multiplication follows the same steps of multiplications and additions. We, therefore, apply the same extend-and-prune technique on the secret coefficents' higher-order 27 bits. Note that we do not bypass the first multiplication and directly attack the addition operation because the D×B and D×A product bit locations do not align with each other, resulting in a decrease in the attack's success.

The proposed attack applies the same differential EM attack (DEMA) procedure to extract the exponent bits and sign bits. Figure 2 reflects that the floating-point implementation adds two exponents of the polynomials $FFT(f)$ and $FFT(c)$ and applies an XOR operation on the sign bits. Combined version of the separately recovered mantissa, exponent and sign bits represents one full coefficient.

IV. Evaluation Results

We used the public, reference software of FALCON Round-3 found in the submission package to the NIST standardization. We compiled the code with `gcc-arm-none-eabi-4_8-2014q1` compiler tool and with `-O0` flag, and then ported the generated executable on to an ARM-Cortex-M4. The processor is clocked at 168 MHz and the measure-

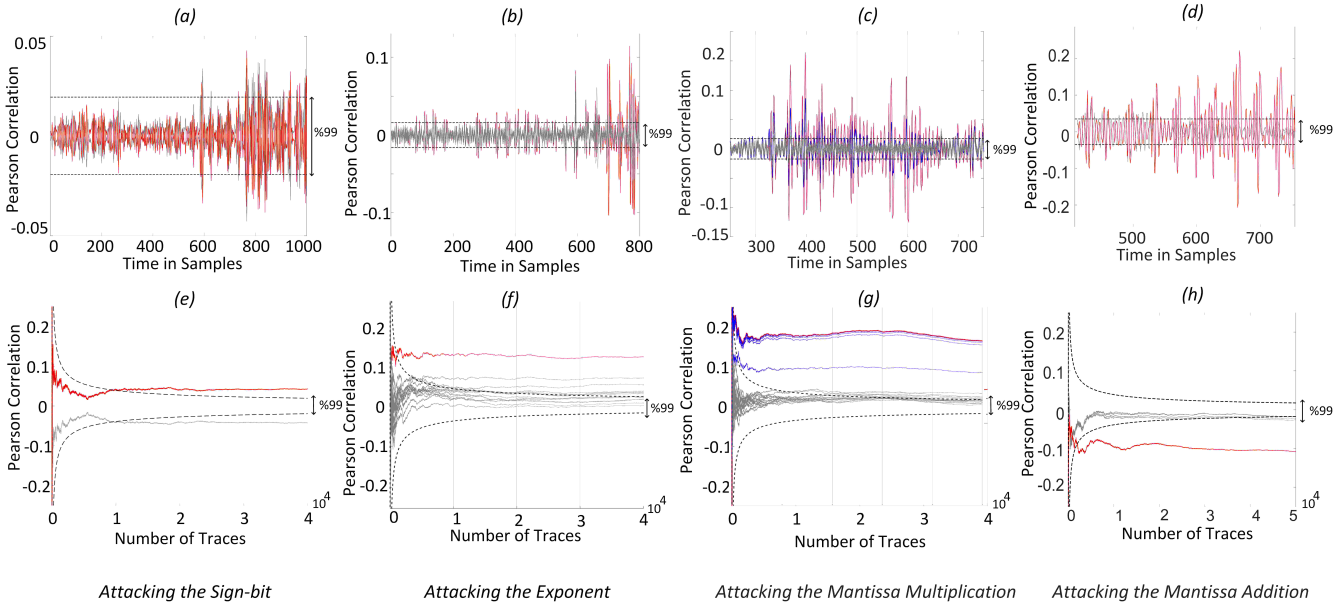

Fig. 4. The results of the proposed attack on a floating-point coefficient during FALCON signing. Correct guesses marked in bold red, significant false guesses in blue, and dashed lines mark the confidence interval of 99.99%. Attack returns the correct guesses for (a) sign, (b) exponent, (c), mantissa multiplication (c) and addition (d), validating our approach. For this coefficient, the correct values can be extracted with over 99.99% confidence in less than 10k measurements.

ments are obtained through a EM Probe LS (low sensitivity) RISC-EMP430LS, which is a near-field probe of measuring up to 1 GHz with 20 mV/1 T@1 MHz sensitivity. The EM probe measurements noise are reduced using a choke coil and sampled with a PicoScope 3206D Oscilloscope at 500 Ms/s.

For the proposed differential side-channel attack, we use the Pearson correlation coefficient based distinguisher on the hamming weight models [24]. This test aims to differentiate populations through their covariance, i.e., by checking if deviations from mean occur in a similar fashion. Correlation trace $r_{i,j}$ for a guess i is defined as

$$r_{i,j} = \frac{\sum_{d=1}^{D}\left[\left(h_{d,i} - \overline{h_i}\right)\left(t_{d,j} - \overline{t_j}\right)\right]}{\sqrt{\sum_{d=1}^{D}\left(h_{d,i} - \overline{h_i}\right)^2 \sum_{d=1}^{D}\left(t_{d,j} - \overline{t_j}\right)^2}} \quad (1)$$

where D is the number of traces each having T data points, $t_{d,j}$ is a EM trace with $0 < d \leq D$ and $0 < j \leq T$, $\overline{t_j}$ is the mean EM trace, $h_{d,i}$ is a leakage estimate in trace d for the guessed value i, and $\overline{h_i}$ is the mean of this estimate. The result $r_{i,j}$ returns a correlation trace with values between [-1,1] that estimates the linear relationship between the guesses r_i and the EM measurement for each guess i and time j. This trace depicts the significance and the timing information of the differential EM leak.

Figure 4 shows the results of our proposed attacks, and it validates that our attacks work in practice and can break FALCON. The attack executes on the coefficient 0xC06017BC8036b580, which means that the correct sign bit is 0x1, exponent bits are 0x406 and the mantissa bits are 0x017bc8036b580 (the higher-order bits are 0x00BDE40 and lower-order bits are 0X36b580). The correlation time plots in Figure 4 (a–d) respectively quantifies that the correlation traces ($r_{i,j}$)s for the correct and false-positive guesses for sign, exponent and mantissa cross the 99.99% confidence interval, while those that are within the margin are true negatives, i.e., we do not have a false negative case. The plot only shows up to top 21 guesses for the mantissa part for visual clarity—the attack evaluates up 2^{27} guesses for the higher-order mantissa bits and 2^{25} guesses for the lower-order bits.

Figures 4 (c) and (d) verify our hypothesis about the proposed attack. Indeed, when the attack executes on the multiplication of mantissa bits, the results in Figure 4 (c) illustrates that significant false positives do occur at the first step of the attack when it targets mantissa multiplications. The correlation result of the top-5 guesses, which include the correct guess and 4 false positives, are actually exactly the same (shown slightly different in the figure for visual clarity). However, the results in Figure (d) validates that with our extend-and-prune strategy, the false positives are all eliminated when the attack then trails the guesses in the earlier step by focusing on the intermediate additions.

Figures 4 (e–h) plot the correlation evolution taken at the leakiest time sample on Figures (a–d), respectively. These plots measure the number of traces needed to achieve a statistically significant (of 99.99%) correlation. The leakage of the correct guesses become statistically significant with as few as a thousand measurements when attacking the exponent and mantissa addition while other guesses die out.

The most challenging portion of the attack, in terms of the number of measurements needed, is extracting the sign bit, and it takes about a few hundred measurements to leak the correct value and about 9k measurements to make the leak statistically significant for this example. Overall, the measurement for all coefficients can be confidently acquired with less than 10k measurements. Note that the sign-bit leakage is symmetric for the positive vs. negative sign guesses. This indeed is expected and does not cause a problem for the attack because the correct guess consistently has a positive correlation at the maximum

leakage point (i.e., red trace in Figure 4 (e) appears the same way for both negative and positive signs).

We performed the proposed attack on FALCON-512 but the same attack is applicable to the other setting (i.e., FALCON-1024) because FALCON employs the same floating-point arithmetic implementations for both parameters. The difference between the two settings is the polynomial ring size. The polynomials in FALCON-512 have 512 coefficients, while FALCON-1024 works with polynomials having 1024 coefficients.

V. Discussions

A. Limitations of Our Attack

We proposed a generic attack that works without profiling the target device by (re)-configuring a secret key and that is still practical. We do not claim that our attack provides the lower bound on the number of tests needed to extract the key. It is possible to extend our attack by template [20] or machine-learning based [25], [26] profiling techniques and by using better measurement equipment.

B. Possible Countermeasures

The most popular techniques for side-channel mitigation is hiding and masking. While hiding aims making power-consumption constant, masking aims randomizing the intermediates values processed by the implementation. Although SABER, another NIST post-quantum signature finalist, has a masked implementation recently proposed [27], this does not yet exist for FALCON—such an implementation can be considered by the FALCON team.

C. NTT vs. FFT—A Side-Channel Perspective

Based on our analysis, one can argue that FFT has possibly a lower power/EM side-channel leakage compared to NTT. While our attack on FFT requires around 10k traces, NTT has shown to be vulnerable even with a single trace [19]. This is probably due to the amount of non-linearity introduced. While NTT applies a modular reduction prime p, this doesn't exist in FFT. Therefore, the attack is likely to distinguish and eliminate wrong guesses easier in NTT. Further research is needed to conduct a quantitative analysis.

D. Comparison with the Related Work

The difference between our approach and the attack on floating-point arithmetic in prior work [28] is two-fold. First, the earlier work focused on single-precision variables while we analyze double-precision. Second, the earlier work only focuses on a small part of the mantissa and cannot recover from false positives while our novel attack eliminates false positives and recovers the full mantissa—note the attack's precision in our scenario is more important since the adversary targets secret cryptographic keys rather than the weights of a neural network.

VI. Conclusions

This paper has demonstrated the first side-channel attack on the NIST post-quantum finalist FALCON. We have revealed the unique challenges of FALCON stemming from floating-point arithmetic and the signature construction, and we have addressed those challenges through novel attacks. The results have validated that our attacks do work in practice with a few thousand measurements and without a quantum computer. This paper, therefore, motivates the need for strong countermeasures against such attacks and including the related overheads in hardware/software performance figures.

VII. Acknowledgements

This research is supported in part by the by the National Science Foundation under Grant No. 1850373. NCSU is an academic partner of Riscure Inc., and we thank them for providing hardware/software support for side-channel analysis.

References

[1] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] J. Proos et al., "Shor's discrete logarithm quantum algorithm for elliptic curves," *Quantum Info. Comput.*, vol. 3, no. 4, pp. 317–344, Jul. 2003.

[3] P. Kocher et al., "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, 1999, pp. 388–397.

[4] D. Moody, "The 2nd round of the NIST PQC standardization process," https://csrc.nist.gov/CSRC/media/Presentations/the-2nd-round-of-the-nist-pqc-standardization-proc/images-media/moody-opening-remarks.pdf.

[5] P. A. Fouque et al., "Falcon: Fast-fourier lattice-based compact signatures over NTRU."

[6] A. Park et al., "Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations," *IACR TCHES*, pp. 500–523, 2018.

[7] Y. Liu et al., "On security of fiat-shamir signatures over lattice in the presence of randomness leakage." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 715, 2019.

[8] A. Fournaris et al., "Profiling dilithium digital signature traces for correlation differential side channel attacks," in *International Conference on Embedded Computer Systems*. Springer, 2020, pp. 281–294.

[9] A. C. Atici et al., "Power analysis on NTRU implementations for RFIDs: First results," pp. 128–139, 2008.

[10] M.K. Lee et al., "Countermeasures against power analysis attacks for the NTRU public key cryptosystem," *IEICE T FUND ELECTR*, vol. 93, no. 1, pp. 153–163, 2010.

[11] A. Wang et al., "Power analysis attacks and countermeasures on NTRU-based wireless body area networks," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 7, no. 5, pp. 1094–1107, 2013.

[12] A. Aysu et al., "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 81–88.

[13] T. Espitau et al., "Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing Against strongSwan and Electromagnetic Emanations in Micro-controllers," in *ACM CCS*, 2017, pp. 1857–1874.

[14] T. Oder et al., "Practical CCA2-Secure and Masked Ring-LWE Implementation," *IACR TCHES*, vol. 2018, no. 1, pp. 142–174, 2018.

[15] P. Ravi et al., "Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs," *IACR TCHES*, vol. 2020, no. 3, pp. 307–335, 2020.

[16] J. Bos et al., "Assessing the feasibility of single trace power analysis of Frodo," in *International Conference on Selected Areas in Cryptography*. Springer, 2018, pp. 216–234.

[17] T. Gellersen et al., "Differential power analysis of the picnic signature scheme," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 267, 2020.

[18] R. Primas et al., "Single-trace side-channel attacks on masked lattice-based encryption," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 513–533.

[19] P. Pessl et al., "More practical single-trace attacks on the number theoretic transform," in *LatinCrypt*. Springer, 2019, pp. 130–149.

[20] S. Chari et al., "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, 2003, pp. 13–28.

[21] K. Gandolfi et al., "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, 2001, pp. 251–261.

[22] A. Park et al., "Chosen ciphertext simple power analysis on software 8-bit implementation ring-LWE encryption," in *IEEE AsianHOST*, 2016, pp. 1–6.

[23] R. Primas et al., "Single-trace side-channel attacks on masked lattice-based encryption," in *Cryptographic Hardware and Embedded Systems – CHES 2017*. Cham: Springer International Publishing, 2017, pp. 513–533.

[24] E. Brier et al., "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, 2004, pp. 16–29.

[25] H. Maghrebi, "Deep learning based side channel attacks in practice," Cryptology ePrint Archive, Report 2019/578, 2019.

[26] J. Kim et al., "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR TCHES*, vol. 2019, no. 3, pp. 148–179, May 2019.

[27] M. Van Beirendonck et al., "A side-channel resistant implementation of SABER," Cryptology ePrint Archive, Report 2020/733, 2020.

[28] L. Batina et al., "CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel," in *USENIX Security Symposium*, 2019, pp. 515–532.