Single Trace Side-Channel Vulnerabilities Discovery Using Statistical Leakage Simulator

JINYI QIU

This paper presents a novel *single-trace* side-channel attack on FALCON—a lattice-based post-quantum digital signature protocol recently approved for standardization by NIST. We target the discrete Gaussian sampling operation within the FALCON key generation scheme and use a single power measurement trace to succeed. Notably, negating the 'shift right 63-bit' operation (for 64-bit values) leaks critical information about the '-1' vs. '0' assignments to intermediate coefficients. These leaks enable full recovery of the generated secret keys. The proposed attack is simulated on the ELMO simulator running both reference and optimized software implementation from FALCON's NIST Round 3 package. Statistical analysis with 20k tests reveals a full key-recovery success rate of 100% for FALCON-512. This work highlights the vulnerability of current software solutions to single-trace attacks and underscores the urgent need to develop single-trace resilient software for embedded systems in the presilicon phase.

1. INTRODUCTION

Widely adopted encryption schemes such as RSA [1] and elliptic curve cryptosystems [2] rely on problems such as the discrete logarithm [3] and integer factorization [4]. Unfortunately, quantum computers have been proven to solve these problems with exponential speedup [5], which motivates the need for alternatives.

To address this issue, the National Institute of Standards and Technology (NIST) initiated a standardization process for quantum-resilient (also known as post-quantum) cryptographic schemes that can withstand quantum cryptanalysis [6]. This standardization process selected three digital signature schemes: CRYSTALS-Dilithium [7], SPHINCS+ [8], and FALCON [9]. NIST selected FALCON due to its small signature sizes, which make it especially favorable for embedded systems applications.

While the chosen algorithms are claimed to be mathematically robust, their practical implementations may be vulnerable to side-channel attacks (SCA). These attacks exploit implementation characteristics such as execution time, power consumption, and electromagnetic radiation to extract secret values. An attacker can execute these attacks with only a few side-channel measurements from the physical device. The most extreme form of these attacks, known as *single-trace attacks* (a.k.a., simple power analysis), allows the adversary to extract secrets using measurements from just a single program execution. These attacks are perilous because single-trace measurements bypass common defenses like masking [10]. Moreover, they can also target subroutines such as key generation that generate a new secret each time it executes. FALCON is especially suitable for embedded system deployment, making it a prime target for side-channel attacks. Given the imminent real-world deployment of NIST's post-quantum algorithms, there is a critical need to expose these attacks and inform effective defenses.

Conducting physical side-channel measurements on PQC implementations is costly and time-consuming. Moreover, discovering these vulnerabilities post-deployment is often too late, as the security of critical systems may already be compromised. Therefore, it is crucial to identify potential leaks during the "pre-production" phase, while these algorithms are still in development. Furthermore, hardware variability necessitates a statistical approach to identify side-channel vulnerabilities independent of specific devices.

Off-the-shelf tools like the ELMO [11] statistical leakage simulator provide a cost-effective platform for simulating the power consumption of cryptographic implementations on low-power microprocessors. ELMO comprises two main components: an emulator and a set of leakage models. The emulator simulates the ARM M0 core, including its 3-stage pipeline, by executing Thumb assembly instructions. The leakage models then use the instruction flow produced by the emulator to predict power consumption, with minimal noise. Unlike other power simulators, ELMO does not depend on fixed assumptions about the processor's power model. Instead, it employs a statistical model-building process that evaluates the significance of 'promising' variables influencing power consumption. This nuanced approach allows ELMO to capture power consumption behaviors essential for accurately predicting and identifying potential side-channel leaks.

The primary objective of this project is to use a statistical leakage simulator to identify single-trace side-channel vulnerabilities in FALCON. By simulating power traces of FALCON implementations, this study aims to uncover and validate leakage points that could compromise their security. Unlike traditional side-channel analysis, which relies on physical measurements and device-specific characteristics, this project employs a statistical approach to reveal generic vulnerabilities in PQC algorithms.

We begin by setting up the ELMO simulator environment, including its configuration to emulate the execution on the ARM Cortex-M0 architecture. Then, we configure varying cryptographic inputs and keys to produce traces that reflect different operational scenarios. Once the simulated power traces are generated, we peform single-trace side-channel analysis using statistical techniques to identify leakage points. This approach enables a systematic exploration of potential vulnerabilities without the variability constraints of physical measurements. After identifying observable vulnerabilities, a thorough statistical analysis is performed to quantify the extent of leakage and the impact on FALCON's security scheme.

We report the finding of a *new* single-trace side-channel vulnerability that enables complete recovery of the session secret key using only side-channel information. This vulnerability resides in the discrete Gaussian sampling subroutine of FALCON's reference software implementation and is orthogonal to the vulnerability disclosed in the prior attack [12]. Compared to previous work, our attack eliminates the need for lattice reduction and avoids computationally intensive post-processing.

We first present the leaky operations used in the implementation and show how they leak important intermediate '0' and '-1' value assignments. Furthermore, we discuss the algorithm and demonstrate how these assignments can lead to full secret key recovery. We demonstrate how to locate these vulnerable operations in the power trace. We use statistical methods to extract these assignments with high accuracy and efficiency. Finally, we provide visualizations of our attack results and quantify our attack's success rate.

The contributions of this paper are as follows:

- We reveal a new side-channel vulnerability in FALCON's key generation process that could lead to full secret key recovery using only a single power measurement.
- We applied the attack on the ELMO simulator running reference and optimized software implementations from the FALCON NIST submission package. Our attacks extracted the full key recovery with success rate of 100%. The attack remains effective in both the reference and optimized implementations, as the same exploitable leakage exists in both versions.

2. BACKGROUND AND RELATED WORK

This section starts with an overview of FALCON and explains its secret polynomials. For brevity, we omit the details in FALCON's key generation process. Following this, we discuss the related work and ELMO's power model. Finally, we outline our threat model.

A. The Generation of FALCON's Secret Polynomials

FALCON stands for Fast-Fourier Lattice-Based Compact Signature Over NTRU. It is a digital signature scheme designed for the post-quantum era, meaning it would take a quantum computer a significant amount of time to break the mathematical trapdoors used in this algorithm. FALCON consists of three main parts: key generation, signature generation, and signature verification. The key generation process defines the NTRU-Lattice components f and g. The session's public key, secret key, and signature are derived from f and g without involving any randomness, as specified in Algorithm S1. This paper focuses on the discrete Gaussian sampling subroutine that generates f and g with the format shown below:

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots + f_{n-1} x^{n-1} \tag{S1}$$

$$g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots + g_{n-1} x^{n-1} \tag{S2}$$

The coefficients of f and g (namely, $f_0, f_1, ..., f_{n-1}$ and $g_0, g_1, ..., g_{n-1}$) are sampled individually using a discrete Gaussian distribution. The mean of this distribution is 0, while the standard deviation is determined by the degree n and the parameter q. Degree n is defined by the user, which is either 512 or 1024, and q is set to 12289.

The Gaussian sampling method generates values through iterative sampling performed during lines 3 and 4 of Algorithm S1. Within the sampling process, each iteration generates a random number to determine if the sample should be zero or non-zero. If the sample is non-zero, another random value is used to select a threshold-based index from a pre-computed table that aligns with the desired distribution. The sample's sign is then randomly set to positive or negative as the sample mean is 0. The signed sample is accumulated to a final sum, which is returned as the output after multiple iterations. While the mathematical details of this process are beyond the scope of the discussion, section 3 will demonstrate how the software implementation facilitates full key

Algorithm S1. NTRUGen(ϕ, q)

Require: A monic polynomial $\phi \in \mathbb{Z}[x]$ of degree n, a modulus q
Ensure: Polynomials f, g, F, G
1: $\sigma_{f,g} \leftarrow 1.17\sqrt{q/2n}$
2: **for** $i \leftarrow 0$ **to** $n-1$ **do**
3: $\quad f_i \leftarrow D_{\mathbb{Z},\sigma_{f,g},0}$
4: $\quad g_i \leftarrow D_{\mathbb{Z},\sigma_{f,g},0}$
5: $f \leftarrow \sum_i f_i x^i$ ▷ $f \in \mathbb{Z}[x]/(\phi)$
6: $g \leftarrow \sum_i g_i x^i$ ▷ $g \in \mathbb{Z}[x]/(\phi)$
7: $(F, G) \leftarrow NTRUSolve_{n,q}(f, g)$ ▷ Compute F, G such that $fG - gF = q \mod \phi$
8: **if** $(F, G) = \perp$ **then**
9: $\quad$ **restart**
return f, g, F, G

recovery.

B. Related Work

Prior works on the single-trace side-channel analysis of lattice cryptosystems have targeted several vulnerable components of the algorithm, such as the number theoretic transform (NTT) [13–15], discrete Gaussian sampling [16], polynomial multiplication [17–19], message encoding/decoding [20–23], and other elements such as ω-small sampling [24], cumulative distribution table (CDT) sampling [25, 26], Fujisaki-Okamoto Transform [27]. Although FALCON incorporates some of these components, it also includes distinct operations such as fast Fourier sampling and floating-point arithmetic. Earlier attacks cannot be trivially extended to the implementation of these units. Side-channel vulnerabilities in FALCON's software implementation remain largely unexplored. Previous studies have investigated multi-trace attacks on FALCON [28–30]; however, its susceptibility to single-trace attacks remains largely unknown, aside from a vulnerability [12] on the base sampler. This gap necessitates further investigation.

C. ELMO's Power Model and Effectiveness

ELMO, short for Emulator for power Leakage M0, is a power simulator that simulates power consumption by processing compiled binary code and generating power traces as if the target code were running on an ARM Cortex-M0. The simulator was first introduced in 2017 and has been adopted in several side-channel work [31–33] that verified its effectiveness including the most recently published side-channel attack on FALCON [12]. The simulation process follows these steps: first, the input is processed by Thumbulator [34], an open-source tool that extracts data flow and debug information. Next, this extracted information is fed into a pre-trained linear regression model. The format of the model is:

$$\mathbf{y} = \delta + \left[\mathbf{I_p}|\mathbf{I_s}|\mathbf{D}|\mathbf{DxI_p}|\mathbf{DxI_s}\right]\beta + \varepsilon$$

Within this formula, y represents the simulated power measurements, while ε is a constant noise term. Terms δ and β are pre-trained weight and bias terms. The terms I_p, I_s, D, DxI_p, and DxI_s are extracted from Thumbulator, with detailed information provided in Table S1. However, the key takeaway is that the power model places significant emphasis on the Hamming Weight of an instruction's operand and the Hamming Distance of its previous value. This means that if the Hamming Weight of a register value changes significantly, the power simulations will reflect this change. This characteristic is crucial, as the discovered leakage points leverage this property.

D. Threat Model

We adopt the well-established threat model for single-trace side-channel attacks [16, 24], assuming an adversary with physical access to the target device who can measure power consumption during cryptographic operations. The adversary is assumed to possess knowledge of the executing software, approximate the timing of specific computations, and intercept communication channels to capture exchanged public messages. During the characterization phase, the attacker can supply random inputs and analyze the software's power behavior with known values. However, at runtime, they are restricted to capturing a single power trace in their attempt to deduce the entire secret key. Other attacks such as fault injection, buffer overflow, and attacks targeting operating systems are considered out of scope.

Notation	Description
I_p	decoded previous instructions
I_s	decoded subsequent instructions
D	operand bits and bit transitions on hardware BUS Lines, concatenated
DxI_p	Hamming weights of two operands and their Hamming distances between current and previous instructions
DxI_s	Hamming weights of two operands and their Hamming distances between current and subsequent instructions

Table S1. Notation Definitions

3. UNDERLYING MECHANISM OF THE ATTACK

In this section, we first demonstrate how the target bit shift operation leaks intermediate secret variables. We also present proof-of-concept studies that validate this vulnerability. Subsequently, we explain how these leaked variables enable full recovery of FALCON's secret polynomial.

A. The Operations That Leak

This section describes how negating the 'shift 63-bit' operation can leak the value assignments of 0 and -1 and preliminary results are presented to substantiate this claim.

For a 64-bit variable, the 'right shift 63-bit' operation (expressed as '$(x \gg 63)$' in software) shifts the most significant bit (MSB) to the least significant bit (LSB) and clears all other bits. This operation produces only two possible outcomes: '0' or '1.' The negation of this result changes the value to either '0' or '-1,' represented in two's complement as all 0's or all 1's, respectively. When the processor writes the result, the '-1' case exhibits a Hamming Weight (HW) of 64, while the '0' case has a HW of 0. **Consequently, the '-1' case results in higher power consumption compared to the '0' case.**

Figure S1 illustrates the results of an experiment validating the preceding argument. The experiment involves performing a negation operation on a 64-bit value using the assembly code shown in Listing 1, aligning with the FALCON implementation. The `sbc.w` instruction is employed for two's complement negation (sign inversion) of the 64-bit value. The power consumption of

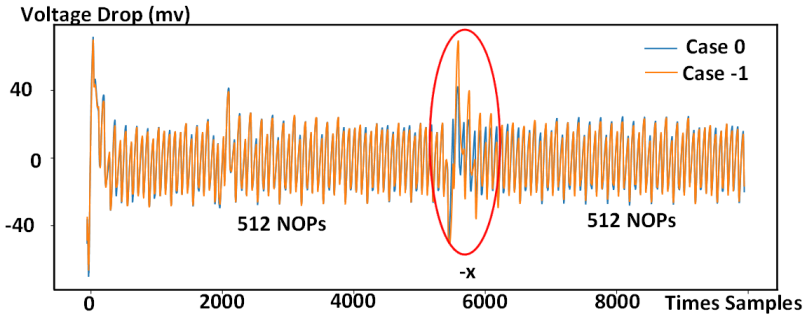

Fig. S1. Power traces of the leaky operation with varying inputs are presented. The trace in blue depicts the power consumption during the leaky operation when the secret is assigned the value '0', while the trace in orange shows the power consumption when the secret is assigned '-1.' A distinct difference in power consumption between the two cases is observable.

the operation $-(x)$ was measured, with x taking a value of either 1 or 0 (the result of $x \gg 63$). Assembly `NOP` instructions were inserted around the operation to isolate it. We plotted the two graphs on top of each other to show the difference, the blue graph shows the results when x is 0, indicating a peak voltage drop of only 40 mV. The orange graph illustrates the results when x is 1, revealing a peak voltage drop of 70 mV. Additionally, the power spike is more pronounced in the '-1' case compared to the '0' case, demonstrating the vulnerability. We then employed the same set of operations on ELMO to obtain the average power trace. We observed a clear difference between case '0' and case '-1', as shown in Figure S2.

B. Only a Few Variables Needed

This subsection explains how an adversary can recover FALCON's base component f and g using only a few intermediate variables. Recall, that the variables of f and g are generated using a discrete Gaussian sampling process. The reference C code implementation of this process from the NIST submission package is outlined in Listing 2. This subroutine is called n times to generate n variables for the polynomial that forms the base component of the NTRU lattice. The parameter n is the degree of the polynomial specified by the user. This code implementation is composed of a set of nested loops. The outer loop executes twice, while the inner loop executes 26 times for each outer loop execution.

Within Listing 2, line 20 contains the generated variable. To extract the generated secret coefficient, we trace the changes to this variable backward in this subroutine. Line 19 performs a pass-by-value operation followed by a pass-by-reference operation, but numerically it simplifies to `val = val + v`. Consequently, the adversary requires the value of `v` to deduce the returned result. In Line 18, `v` is XORed with `-neg` and added to `neg`, implying that the adversary must know both `v` and `neg` to determine `val`. We therefore identify the two attack points in this algorithm that will lead to full recovery on f and g. They are line 16 to learn the value of `v` and line 18 to learn the value of `neg` (both highlighted in Orange).

Line 16 is our first attack point because the value of `-(t & (f ^ 1))` can only take on '0' or '-1'. The reason is that `t` and `f` (not to be confused with the base lattice component f) can only take '0' or '1' due to the 'shift 63-bit' operation on line 8 and line 15, which clears everything other than the most significant bit (MSB). The negation of `t & (f ^ 1)` thus turns into a negation of '0' or '1'. The negation result, expressed in two's complement, will be all 0s (for 0) or all 1s (for -1). **This will cause a significant power consumption difference of the target device because the Hamming Weight (HW) of these two results differs by 64.** Additionally, in line 16, k is the sequence of the

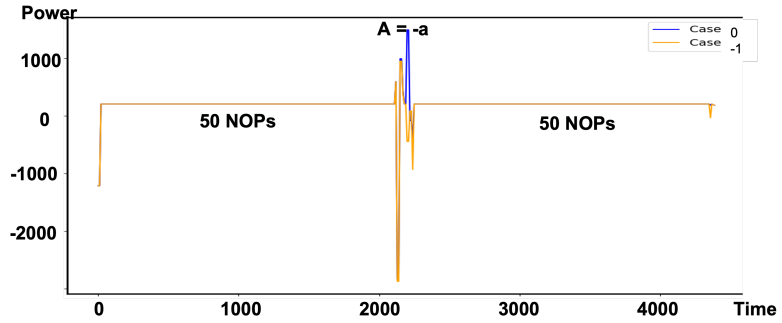

Fig. S2. simulated power traces of the leaky operation with varying inputs are presented. The trace in blue depicts the power consumption during the leaky operation when the secret is assigned the value '0', while the trace in orange shows the power consumption when the secret is assigned '-1.' A distinct difference in power consumption between the two cases is observable.

Listing 1. Assembly instructions corresponding to $-(x)$

```
negs    r2, r2;
sbc.w   r3, r3, r3, lsl #1;
strd    r2, r3, [r7, #24];
```

Listing 2. Gaussian sampling implementation from NIST submission package

```
1  mkgauss(RNG_CONTEXT *rng, unsigned logn){
2      ...
3      for (u = 0; u < g; u++) {
4          ...
5          r = get_rng_u64(rng);
6          neg = (uint32_t)(r >> 63);
7          r &= ~((uint64_t)1 << 63);
8          f = (uint32_t)((r - gauss_1024_12289[0]) >> 63);
9          v = 0;
10         r = get_rng_u64(rng);
11         r &= ~((uint64_t)1 << 63);
12         for (k = 1; k < (sizeof gauss_1024_12289)
13             / (sizeof gauss_1024_12289[0]); k++){
14             uint32_t t;
15             t = (uint32_t)((r - gauss_1024_12289[k]) >> 63) ^ 1;
16              v |= k & -(t & (f ^ 1));    // Highlighted
17             f |= t;}
18          v = (v ^ -neg) + neg;    // Highlighted
19         val += *(int32_t *)&v;}
20     return val;}
```

inner loop execution (a known number between 1 and 26). An adversary can infer the value of `v` by exploiting this vulnerability.

Line 18 is our second attack point because `-neg` can only take '0' or '-1'. This is due to the 'shift 63-bit' operation on line 6. The negation of `neg` on line 18 creates a similar vulnerability compared to our first attack point because of the HW difference. An adversary can infer the value of `neg` by attacking this vulnerability.

Since the attack points reside within a loop, and the current loop execution relies on the value of `v` generated from the previous loop iterations. Therefore, the attack success rate must be sufficiently high to ensure successful recovery. An incorrectly inferred intermediate value will result in outcomes that differ from the ground truth.

4. EXPERIMENTAL SETUP

We utilized the submission package from the NIST reference software implementation. The code was compiled using the `gcc-arm-none-eabi-4_8-2014q1` compiler with `-O0` optimization flag. We built ELMO's source code using `-O0` optimization flag. The power simulator was also set to `Cycle Accurate` mode to offer more correspondence with real-world power measurements. We also collected measurement traces from a real-world target device and compared their power

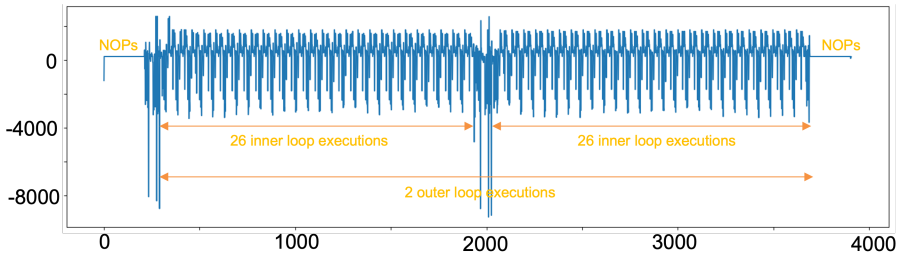

Fig. S3. Simulated full power trace of the subroutine. The two outer loop executions and the 26 inner loop executions are clearly observable.

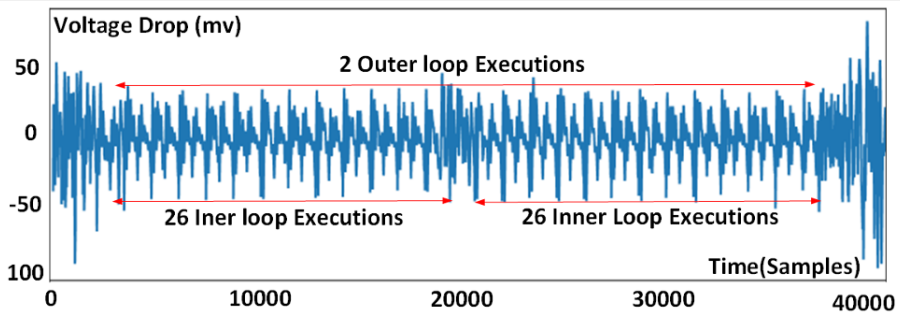

Fig. S4. Measured power trace of the subroutine running on the target device is shown. The two outer loop executions and the 26 inner loop executions are clearly observable.

trace shapes with the simulated results. The target device is an ARM Cortex-M4F CPU operating at 30MHz, which is a canonical setting to test side channels on embedded applications [24, 29]. Measurements were captured with a PicoScope 3206D oscilloscope, set to a sampling rate of 250MHz, using a Tektronix CT1 passive current probe with a bandwidth of 1–1000 MHz at 3 dB. No external amplification was applied to enhance the measurements. We omit a detailed analysis of the real-world measurements, as it falls outside the scope of this paper.

5. EXPLOITING THE FOUND VULNERABILITY

This section presents the proposed attack strategy for recovering the secret polynomials in FALCON. We will inspect the power trace and discuss how we identify the point of interest. As noted in the previous section, the attack requires a profiling stage in which the adversary has physical access to the hardware and software. During this stage, multiple measurements from different software inputs are used to build a profile that helps determine *when* the leakage occurs. However, once the profile is obtained, the adversary can infer the secret polynomial using a single power measurement.

A. Inspecting the Power Trace

Figure S3 shows the full power trace obtained when executing the discrete Gaussian sampling subroutine. The two outer loop executions and the 26 inner loop executions within each outer loop are distinguishable, reflecting the code's structure shown in Listing 2. We observe a recurring pattern for each inner loop execution approximately every 55 time samples, and for each outer loop execution, that is 1700 samples. Since the discrete Gaussian sampling subroutine is executed n times during the key generation process, the resulting power trace patterns will be easily distinguishable. The physically measured power traces demostrated very similar characteristics as the simulated one, as shown in Figure S4.

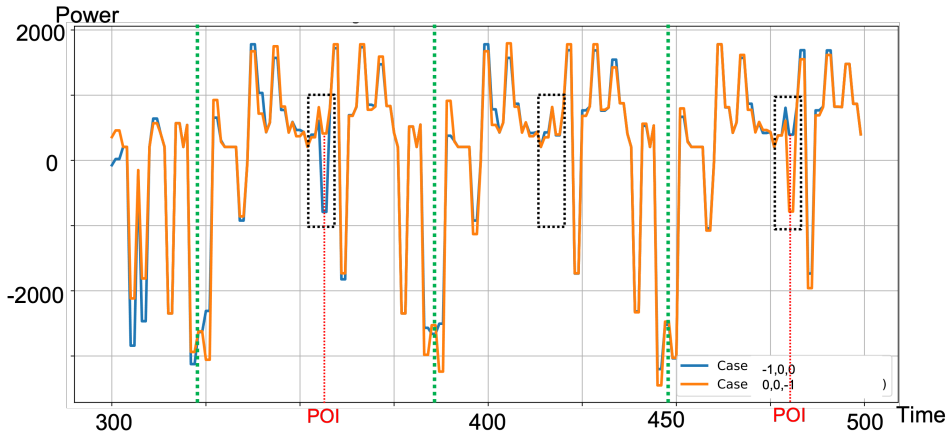

Fig. S5. The average power trace for the first vulnerability `-(t & (f ^ 1))`. The two cases from the first to the third outer loop is shown. Green vertical lines indicate inner loop executions, while black squares highlight regions of interest. The secret assignments, happening at the point of interest (POI) are identifiable through the average power traces.

B. Pinpointing the Point of Interest

We visually identify the secret intermediate coefficient assignments within the zoomed-in version of the average power trace as the area of interest and use the sum of square difference method to identify the point of interest(POI). For the vulnerability `-(t & (f ^ 1))`, we first assign values to variable `r` and adjust the values in the matrix `gauss_1024_12289` so that the value of `-(t & (f ^ 1))` in the first and third inner loop executions can be manually controlled. We then take 20k simulated measurements, setting `-(t & (f ^ 1))` to '-1' in the first loop and '0' in the third for the first 10k, and reversing these settings for the next 10k.

Figure S5 represents the full simulated power trace, zoomed in between time samples 300 and 500. We identify this segment as the first three inner loop iterations, based on the recurring lowest point in the power trace every 55 samples, which aligns with our observations during power trace analysis. We marked out the boundary of power trace for each of the inner loop using green lines for the user to distinguish them easier. To enhance visibility of power consumption differences between the two cases, we overlay their corresponding traces in a single graph. For both cases, most sections of the power trace remain identical. However, the highlighted region exhibits a distinct difference corresponding to the secret intermediate assignments. Specifically, during the first to third loop iterations, the orange trace represents the secret coefficient assignments '0', '0', '-1', while the blue trace corresponds to '-1', '0', '0'. This aligns with the power simulation results, where '-1' produces a higher-magnitude trace, and '0' results in a lower-magnitude trace.

To identify the point of maximum divergence between the two cases, we apply the sum of squared differences (SOSD) method [35]. The most significant deviation occurs in the third inner loop at time sample 480.

Similarly, for the second leaky operation, we controlled the value assignment of `-neg` and analyzed the corresponding power traces, as shown in Figure S6. We focused on the power trace segment between the end of the last inner loop and the start of the second outer loop. Applying the SOSD method, we identified the point of interest at time sample 1936.

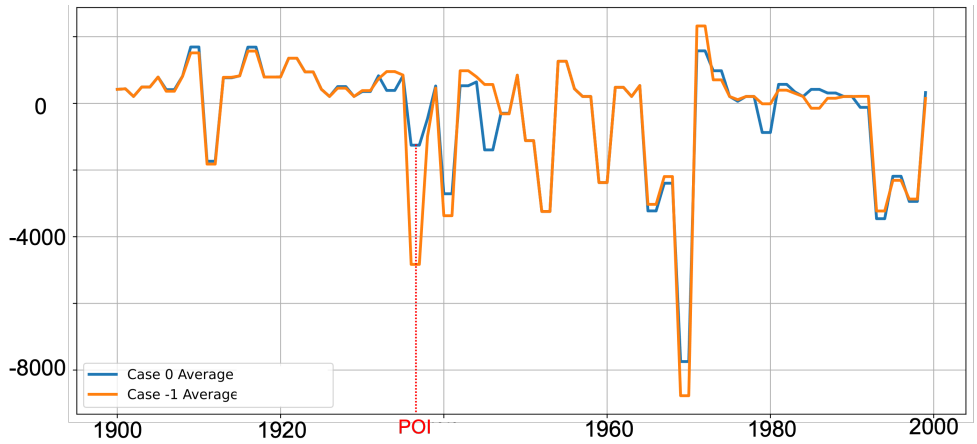

Fig. S6. The average power trace for the second vulnerability `-(neg)`. The two cases between the two outer loops is shown. SOSD is employed to identify the point of interest (POI). The secret assignments, happening at the POI are identifiable through the average power traces.

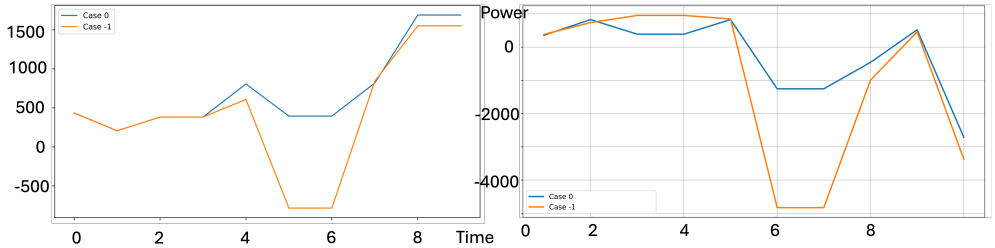

Fig. S7. The average trace around the leaky operations for the two attack points is depicted. The left figure corresponds to the first attack point, while the right figure represents the second attack point. At both attack points, there are only two possible cases: '0' and '-1'. It is observed that the power consumption for case '-1' is higher than for case '0'.

6. RESULTS AND ANALYSIS

In this section, analyze of the simulated side-channel information and present the results. We omit the results obtained from physical measurement for brevity. We begin by demonstrating that our attack successfully distinguishes between different intermediate value assignments through graphical visualizations. We then quantify our attack success rate. Finally, we analyze the impact of the proposed attack on the security of FALCON.

A. Attack Results

The left graph in Figure S7 shows the average power trace of the 10 samples around the point of interest (POI) when attacking `-(t & (f ^ 1))`. The right graph of Figure S7 illustrates the separation between cases '0' and '-1' as revealed by our attack for `-neg`, using average traces. The horizontal axis represents time samples, while the vertical axis reflects power consumption. Our observations reveal that the average power consumption for case '-1' is higher than for case '0'.

We then quantified the attacker's success rate by modeling the power distribution to derive a theoretical estimate. We selected the point of highest divergence between the two cases as the POI to build our attacker's model, though multiple POIs could be chosen to enhance the success rate on noisier platforms. For this POI, we calculated the average power μ_i and variance of power v_i.

Table S2 presents the average power μ_i and power variance v_i for cases 0 and -1 at the two attack

Statistics	`-(t & (f ^ 1))` case -1	`-(t & (f ^ 1))` case 0	`-neg` case -1	`-neg` case 0
Mean	-783.8	395	-4829.78	-1254.05
Variance	1.3×10^{-20}	1.3×10^{-22}	0	5.17×10^{-26}

Table S2. Result Statistics

points, `-(t & (f ^ 1))` and `-neg`. The variance v_i is nearly negligible, as simulation results are free from noise introduced by physical measurements. Consequently, an attack can perfectly distinguish between the two cases using the model built from the simulated power traces. The obtained model was applied to 20k collected measurements to derive classification labels. A comparison with the ground truth shows that in practice we didn't see any samples causing a false misidentification.

B. The Effect on FALCON's Security Schemes

Each discrete Gaussian sampling subroutine execution involves running the outer loop twice and running the inner loop 52 times. Our Attack on `-(t & (f ^ 1))` accomplished 100% accuracy and attack on `-neg` achieved 100% accuracy. Consequently, the overall success rate of our attack to extract one coefficient in FALCON is 100%.

Providing the success rate above, our overall success rate for inferring the two full secret polynomials, both f and g, for FALCON-512 is:

$$((100\%)^{512})^2 = 100\%$$

For FALCON-1024, Our overall success rate for inferring the two full secret polynomials is:

$$((100\%)^{1024})^2 = 100\%$$

We assert that this vulnerability represents a significant compromise of the FALCON cryptographic scheme.

7. DISCUSSIONS

In this section, we discuss defense methods and related issues. We also comment on the drawbacks of our attack.

A. Defense Methods

Defenses against single-trace side-channel vulnerabilities can be implemented at both the hardware and software levels. On the hardware side, constant power consumption hardware designs can mitigate information leakage [36, 37]. On the software side, techniques such as hiding can be implemented, where noise or random delays are introduced to reduce the correlation between power consumption and executed operations [38].

B. Applicability to Other Implementations

Our proposed attack also applies to other algorithms that negate a right-shifted 63-bit value (for 64-bit variables) on secret intermediate variables. The identified vulnerability exists in both FALCON's optimized and reference implementations, as the discrete Gaussian sampling subroutine is the same. Though we implemented our attack on the implementation of FALCON-512, such an attack also efficiently breaks FALCON-1024.

C. Calibration Factors of the Experiments

The generated power trace varies depending on the simulator's mode of operation. ELMO supports two modes: *Cycle-Accurate (CA)* and *Instruction-Accurate (IA)*. The *CA* mode generates a power measurement per clock cycle, while the *IA* mode produces a measurement per instruction execution. For this attack, we utilized the *CA* mode, as it provides simulation results more closely aligned with real-world measurements, particularly for instructions spanning multiple clock cycles. Switching to *IA* mode could alter the power trace shape, potentially revealing different points of interest.

For physical measurements, the platform's noise level decreases as the device's operating frequency is lowered. To reduce noise in our experiments, we set the development board to its minimum frequency of 30 MHz as in prior work [24]. Earlier works have conducted single-trace side-channel attacks at even lower frequencies, such as 8 MHz for attacks on the NTT [14]. Since our clock frequency is higher than in these studies, analyzing higher frequencies may require more sophisticated equipment for power measurement, additional probes for near-field electromagnetic leakage detection, or noise reduction through amplification and post-processing.

D. Drawbacks of Our Attack

Software configurations and peripheral settings could impact the power consumption of the target device. For example, changing the compiler options or flags could change the instruction sequence or add/remove some instructions. Our attack was conducted at the compiler optimization level `-O0`, where we achieved a high attack success rate. We used `-O0` because it preserves the intended structure and sequence of the originally developed code. An exhaustive evaluation of all compiler optimization settings and flags is left for future work.

Our attack was conducted on ELMO, which power model was built on a single device. For physical attacks to succeed on devices of different makes and models, it is essential to develop distinct power profiles that account for device-specific features such as pipelining and out-of-order execution. Even for devices of the same make and model, variations arising from manufacturing differences, device aging, and environmental conditions must still be considered. Overcoming these challenges may require advanced machine learning-based profiling techniques [39, 40] or building the template again on each new device under test. It is important to recognize that the challenge of cross-device single-trace side-channel attacks on post-quantum cryptosystems remains an open problem, as noted in several previous studies [14, 18, 21, 25, 41, 42]. It is also worth noting that simulated power traces are free from noises generated from physical measurements. For such attack to succeed with noise on specific devices, more advanced techniques may be required.

8. CONCLUSION

Although lattice cryptography is a versatile tool that offers quantum resilience at a reasonable cost, it includes unique operations that have not been thoroughly analyzed for side-channel vulnerabilities. This paper highlights a critical vulnerability in the software implementation of FALCON, specifically in the negation of the right-shift 63-bit operation. We have demonstrated that the discrete Gaussian sampling implementation in FALCON can expose intermediate value assignments, leading to full secret key recovery. Our results confirm the practicality of this attack on a power simulator. The vulnerability we uncovered is distinct from prior single-trace attacks and, by definition, from multi-trace attacks. As a result, the defenses proposed or implemented for other vulnerabilities must be re-evaluated, as they are likely ineffective against this specific form of leakage. We emphasize that this is an attack paper, and our primary goal is to inform the implementers of these algorithms about the vulnerabilities they introduce.

9. APPENDIX

This research, supported by the National Science Foundation (Grant No. 2241879), aligns with the concurrent paper published on HAWK 6 signature scheme [43] but differs in methodology and vulnerability scope.

The student focuses on side-channel attacks on cryptosystems using physical hardware. Therefore, this research project is original and does not overlap with any research assignments provided by the advisor, nor does it coincide with past or concurrent class projects. The student conducted the simulation and analyzed the results independently. While the use of the ELMO simulator is established, its application in discovering and validating single-trace side-channel vulnerabilities in post-quantum cryptographic protocols represents a novel approach. The study's novelty lies in the statistical evaluation of PQC implementations, contributing new insights into the security of these emerging cryptographic standards.

This document has been refined using generative AI tools to enhance the writing and language within each section. However, the core ideas, research, and results are solely the student's work.

REFERENCES

1. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM **21**, 120–126 (1978).
2. N. Koblitz, "Elliptic curve cryptosystems," Mathematics of computation **48**, 203–209 (1987).
3. J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," arXiv preprint quant-ph/0301141 (2003).
4. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review **41**, 303–332 (1999).
5. L. Chen, L. Chen, S. Jordan, *et al.*, *Report on post-quantum cryptography*, vol. 12 (US Department of Commerce, National Institute of Standards and Technology, 2016).
6. L. Chen, L. Chen, S. Jordan, *et al.*, *Report on post-quantum cryptography*, vol. 12 (US Department of Commerce, National Institute of Standards and Technology, 2016).
7. V. Lyubashevsky, L. Ducas, E. Kiltz, *et al.*, "Crystals-dilithium," Algorithm Specifications and Supporting Documentation (2020).
8. D. J. Bernstein, A. Hülsing, S. Kölbl, *et al.*, "The sphincs+ signature framework," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, (2019), pp. 2129–2146.
9. T. Prest, P.-A. Fouque, J. Hoffstein, *et al.*, "Falcon," Post-Quantum Cryptography Project of NIST (2020).
10. K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem," .
11. D. McCann, E. Oswald, and C. Whitnall, "Towards practical tools for side channel aware software engineering:'grey box'modelling for instruction leakages," in *26th USENIX security symposium (USENIX Security 17)*, (2017), pp. 199–216.
12. M. Guerreau, A. Martinelli, T. Ricosset, and M. Rossi, "The hidden parallelepiped is back again: Power analysis attacks on falcon," IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 141–164 (2022).
13. R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *International Conference on Cryptographic Hardware and Embedded Systems*, (Springer, 2017), pp. 513–533.
14. P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *International Conference on Cryptology and Information Security in Latin America*, (Springer, 2019), pp. 130–149.

15. I.-J. Kim, T.-H. Lee, J. Han, *et al.*, "Novel single-trace ml profiling attacks on nist 3 round candidate dilithium," (2020).
16. T. Espitau, P.-A. Fouque, B. Gérard, and M. Tibouchi, "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,* (2017), pp. 1857–1874.
17. A. Aysu, Y. Tobah, M. Tiwari, *et al.*, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST),* (2018), pp. 81–88.
18. J. W. Bos, S. Friedberger, M. Martinoli, *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.
19. Z. Qiao, Y. Liu, Y. Zhou, *et al.*, "Single trace is all it takes: Efficient side-channel attack on dilithium," Cryptology ePrint Archive (2024).
20. B.-Y. Sim, J. Kwon, J. Lee, *et al.*, "Single-trace attacks on the message encoding of lattice-based kems," Cryptology ePrint Archive, Report 2020/992 (2020). https://eprint.iacr.org/2020/992.
21. D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden, "Defeating newhope with a single trace," in *International Conference on Post-Quantum Cryptography,* (Springer, 2020), pp. 189–205.
22. Z. Huang, H. Wang, B. Cao, *et al.*, "A comprehensive side-channel leakage assessment of crystals-kyber in iiot," Internet of Things **27**, 101331 (2024).
23. T. Rabas, J. Buček, and R. Lórencz, "Single-trace side-channel attacks on ntru implementation," SN Computer Science **5**, 239 (2024).
24. E. Karabulut, E. Alkim, and A. Aysu, "Single-trace side-channel attacks on ω-small polynomial sampling: with applications to ntru, ntru prime, and crystals-dilithium," in *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST),* (IEEE, 2021), pp. 35–45.
25. S. Kim and S. Hong, "Single trace analysis on constant time cdt sampler and its countermeasure," Applied Sciences **8**, 1809 (2018).
26. K.-H. Choi, J. Han, and D.-G. Han, "Single trace analysis of visible vs. invisible leakage for comparison operation based cdt sampling," (2024).
27. S. Jendral, K. Ngo, R. Wang, and E. Dubrova, "Breaking sca-protected crystals-kyber with a single trace," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST),* (IEEE, 2024), pp. 70–73.
28. E. Karabulut and A. Aysu, "Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks," in *2021 58th ACM/IEEE Design Automation Conference (DAC),* (IEEE, 2021), pp. 691–696.
29. S. Zhang, X. Lin, Y. Yu, and W. Wang, "Improved power analysis attacks on falcon," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques,* (Springer, 2023), pp. 565–595.
30. S. McCarthy, J. Howe, N. Smyth, *et al.*, "Bearz attack falcon: implementation attacks with countermeasures on the falcon signature scheme," Cryptology ePrint Archive (2019).
31. D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "Lwe with side information: attacks and concrete security estimation," in *Annual International Cryptology Conference,* (Springer, 2020), pp. 329–358.
32. M. A. Shelton, N. Samwel, L. Batina, *et al.*, "Rosita: Towards automatic elimination of power-analysis leakage in ciphers," arXiv preprint arXiv:1912.05183 (2019).
33. Y. Wang, R. Paccagnella, E. T. He, *et al.*, "Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86," in *31st USENIX Security Symposium (USENIX Security 22),* (2022), pp. 679–697.

34. D. Welch, "Thumbulator - a thumb instruction set simulator," (2025). Accessed: 2025-02-19.
35. M. Abadi, A. Agarwal, P. Barham, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems," (2015).
36. K. Tiri and I. Verbauwhede, "Design method for constant power consumption of differential logic circuits," in *Design, Automation and Test in Europe,* (2005), pp. 628–633 Vol. 1.
37. X. Lou, T. Zhang, J. Jiang, and Y. Zhang, "A survey of microarchitectural side-channel vulnerabilities, attacks and defenses in cryptography," CoRR **abs/2103.14244** (2021).
38. M. Brisfors, M. Moraitis, and E. Dubrova, "Side-channel attack countermeasures based on clock randomization have a fundamental flaw," Cryptology ePrint Archive, Paper 2022/1416 (2022).
39. P. Kashyap, F. Aydin, S. Potluri, *et al.*, "2deep: Enhancing side-channel attacks on lattice-based key-exchange via 2d deep learning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems pp. 1–1 (2020).
40. J. Kim, S. Picek, A. Heuser, *et al.*, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 148–179 (2019).
41. C. Zhang, Z. Liu, Y. Chen, *et al.*, "A flexible and generic gaussian sampler with power side-channel countermeasures for quantum-secure internet of things," IEEE Internet of Things Journal (2020).
42. P. Ravi, M. P. Jhanwar, J. Howe, *et al.*, "Side-channel assisted existential forgery attack on dilithium-a nist pqc candidate." IACR Cryptol. ePrint Arch. **2018**, 821 (2018).
43. M. Guerreau and M. Rossi, "A not so discrete sampler: Power analysis attacks on hawk signature scheme," Cryptology ePrint Archive (2024).