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

Emre Karabulut, Aydin Aysu

Department of Electrical and Computer Engineering
North Carolina State University
NC, USA
BLUF: FALCON uses different building blocks than typical lattice cryptosystem. Therefore, more investments are needed for their leakage detection and mitigation.

- Motivation
- Background
- The Proposed Side-Channel Attack
- Attack Environment and Equipment
- Evaluation Results
Motivation

- Finalists: 4 KEM and 3 digital signature schemes
- Six are cracked with side-channel attacks except for one
- FALCON!
The First Side-Channel Attack on FALCON

- We propose the first side-channel attack on NIST’s Round-3 post-quantum digital signature standard finalist FALCON

- We show a novel differential power analysis 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 submission package

- The attack succeeds with ~1k measurements (i.e., $2^{10}$ trials)
Requirements For A Differential Side-Channel Attack

An intermediate computation:
1) that combines a known value and a secret key and
2) the known value varies (i.e., not fixed)

<table>
<thead>
<tr>
<th>Input</th>
<th>Key Hypothesis</th>
<th>Power (µW)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Key=00</td>
<td>Key=01</td>
</tr>
<tr>
<td></td>
<td>state P_m</td>
<td>state P_m</td>
</tr>
<tr>
<td>I₁=01</td>
<td>01 1</td>
<td>00 0</td>
</tr>
<tr>
<td>I₂=0f</td>
<td>0f 4</td>
<td>0e 3</td>
</tr>
<tr>
<td>I₁₀₀₀₀=f₁</td>
<td>f1 5</td>
<td>f0 4</td>
</tr>
</tbody>
</table>

Key Hypothesis:
- Key=00
- Key=01
- Key=ff

State (state P_m):
- Power (µW):
  - Key=00:
    - Power: 1 µW
  - Key=01:
    - Power: 0 µW
  - Key=ff:
    - Power: 7 µW
Quick Introduction to FALCON Scheme

Algorithm 1 FALCON Key Generation Algorithm [5]
\begin{algorithm}
\textbf{Input:} A monic polynomial $\phi \in \mathbb{Z}[x]$, a modulus $q$
\textbf{Output:} A secret key $sk$ and a public key $h$
1: $f, g, F, G \leftarrow \text{NTRUGen}(\phi, q)$
2: $B \leftarrow \begin{bmatrix} g & -f \\ G & F \end{bmatrix}$
3: $\hat{B} \leftarrow \text{FFT}(B)$
4: $G \leftarrow \hat{B} \times \hat{B}^*$ \Comment{\times represents matrix multiplication}
5: $T \leftarrow \text{ffLDL}^*(G)$
6: \bf{for each} leaf of $T$ \bf{do}
7: \hspace{1em} leaf.value $\leftarrow \sigma / \sqrt{\text{leaf.value}}$
8: \bf{end for}
9: $sk \leftarrow (\hat{B}, T)$
10: $h \leftarrow gf^{-1} \mod(q)$
11: return $sk, h$
\end{algorithm}

Algorithm 2 FALCON Signature Generation Algorithm [5]
\begin{algorithm}
\textbf{Input:} a message $m$, a secret key $sk$, a bound $\beta^2$
\textbf{Output:} a signature $sig$ of $m$
1: $r \leftarrow \{0,1\}^{320}$ uniformly
2: $c \leftarrow \text{HashToPoint}(r||m)$
3: $t \leftarrow (\frac{1}{q} \text{FFT}(c) \odot \text{FFT}(F), \frac{1}{q} \text{FFT}(c) \odot \text{FFT}(f))$
4: \bf{do}
5: \hspace{1em} \bf{do}
6: \hspace{2em} $z \leftarrow \text{ffSampling}(t, T)$
7: \hspace{1em} $s \leftarrow (t - z) \begin{bmatrix} \text{FFT}(g) & -\text{FFT}(f) \\ \text{FFT}(G) & -\text{FFT}(F) \end{bmatrix}$
8: \bf{while} $s^2 > [\beta^2]$
9: \hspace{1em} $(s_1, s_2) \leftarrow \text{invFFT}(s)$
10: \hspace{1em} $s \leftarrow \text{Compress}(s_2, 8 \cdot \text{byteLen} - 328)$
11: \bf{while} $s = \perp$
12: \bf{return} $sig = (r, s)$
\end{algorithm}
FALCON Key Generation

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 $h$

1: $f, g, F, G \leftarrow \text{NTRUGen}(\phi, q)$
2: $B \leftarrow \begin{bmatrix} g & -f \\ G & F \end{bmatrix}$
3: $\hat{B} \leftarrow \text{FFT}(B)$
4: $G \leftarrow \hat{B} \times \hat{B}^*$ (× represents matrix multiplication)
5: $T \leftarrow \text{ffLDL}^*(G)$
6: for each leaf of $T$ do
7: $\text{leaf.value} \leftarrow \sigma/\sqrt{\text{leaf.value}}$
8: end for
9: $sk \leftarrow (\hat{B}, T)$
10: $h \leftarrow gf^{-1} \mod(q)$
11: return $sk, h$

- **NTRU equation:**
  \[ fG - gF = q \]
- **Public Key:**
  \[ h = gf^1 \]
- If we know either polynomial ‘$g$’ or ‘$f$’, we can recover the other secret polynomial
FALCON Signing

Algorithm 2 FALCON Signature Generation Algorithm [5]

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

1: $r \leftarrow \{0, 1\}^{320}$ uniformly
2: $c \leftarrow \text{HashToPoint } (r|m)$
3: $t \leftarrow \left( \frac{1}{q} \text{FFT}(c) \circ \text{FFT}(f), \frac{1}{q} \text{FFT}(c) \circ \text{FFT}(f) \right)$
4: do $\triangleright \circ$ represents FFT multiplication
5: do
6: $z \leftarrow \text{ffSampling } (t, T)$
7: $s \leftarrow (t-z) \begin{bmatrix} \text{FFT}(g) & -\text{FFT}(f) \\ \text{FFT}(G) & -\text{FFT}(F) \end{bmatrix}$
8: while $s^2 > [\beta^2]$
9: $(s_1, s_2) \leftarrow \text{invFFT}(s)$
10: $s \leftarrow \text{Compress}(s_2, 8 \cdot \text{sbyteLen} - 328)$
11: while $s = \perp$
12: return $sig = (r, s)$

- NTRU equation: $fG - gF = q$
- Public Key: $h = gf^1$
- If we know either polynomial ‘$g$’ or ‘$f$’, we can recover the other secret polynomial
- Attack target: Multiplication of known polynomial ‘$c$’ and secret polynomial ‘$f$’
FALCON FFT and Multiplication

Secret coefficients of $f$ can be recovered by targeting the FFT-domain multiplication.
Challenge of Attacking Multiplication

\[ \text{Same Hamming Weight} \quad \text{Different Hamming Weights} \]

Additions remove false positives: apply extend-and-prune!
Attack Environment and Equipment

- NIST’s reference software
- ARM-Cortex-M4 microcontroller clocked at 168 MHz
- EM Probe LS (low sensitivity) RISC-EMP430LS
- Scope sampling rate is 500 MS/s
Evaluation Results

1k measurements are sufficient to extract the sign, 100 traces are sufficient to extract exponent and mantissa.
Conclusion

- Unprotected FALCON implementations are vulnerable to side-channel attacks

- Side-channel leakage is different in FALCON

- Attacks only evolve over time*

- Invest in evaluating such side-channels and developing low-overhead defenses