Do Not Disturb a Sleeping Falcon

Floating-Point Error Sensitivity of the Falcon Sampler and Its Consequences

Eurocrypt 2025

Xiuhan Lin, Mehdi Tibouchi, Yang Yu, Shiduo Zhang

Overview

In this work, we mainly focus on security implications in Falcon's use and deployment caused by floating-point arithmetics (FPA) errors.

Overview

In this work, we mainly focus on security implications in Falcon's use and deployment caused by floating-point arithmetics (FPA) errors.

- Falcon Gaussian sampler is sensitive to FPA errors
- FPA carelessness + deterministic Falcon $\rightarrow$ exact signing key recovery

Overview

In this work, we mainly focus on security implications in Falcon's use and deployment caused by floating-point arithmetics (FPA) errors.

- Falcon Gaussian sampler is sensitive to FPA errors
- FPA carelessness + deterministic Falcon $\rightarrow$ exact signing key recovery

Practical attack cost in some derandomized settings

- different implementations + 10000 signing queries $\rightarrow$ full key recovery

Outline

- Background
- Floating-point errors sensitivity analysis
- Exploiting FPA discrepancies
- Sources of FPA discrepancies
- Countermeasures

Background

Falcon

In 2022, Falcon[1] was one of the three signatures selected by NIST for standardization.

[1] https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms

Falcon

In 2022, Falcon[1] was one of the three signatures selected by NIST for standardization.

Falcon's pros

- \+ most compact signature scheme in the 3rd round
- \+ fast signing (but slower than Dilithium) and verification

[1] https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms

Falcon

In 2022, Falcon[1] was one of the three signatures selected by NIST for standardization.

Falcon's pros

- + most compact signature scheme in the 3rd round
- + fast signing (but slower than Dilithium) and verification

Falcon is a lattice-based hash-and-sign signature scheme.

[1] https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms

Hash-and-sign construction

Early constructions (GGH & NTRUSign): $pk := \mathbf{G}, sk := \mathbf{B}$

Sign

1. Hash a message to random $\mathbf{t}$
2. Round $\mathbf{t}$ to $\mathbf{v} \in \mathcal{L}$ (using $\mathbf{B}$)

Verify

1. Check $\mathbf{v} \in \mathcal{L}$ (using $\mathbf{G}$)
2. Check $\mathbf{v} - \mathbf{t}$ is short

Hash-and-sign construction

Early constructions (GGH & NTRUSign): $pk := \mathbf{G}, sk := \mathbf{B}$

Sign

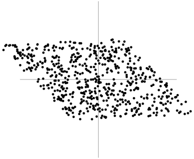
1. Hash a message to random $\mathbf{t}$
2. Round $\mathbf{t}$ to $\mathbf{v} \in \mathcal{L}$ (using $\mathbf{B}$)

Verify

1. Check $\mathbf{v} \in \mathcal{L}$ (using $\mathbf{G}$)
2. Check $\mathbf{v} - \mathbf{t}$ is short

Signing: uses deterministic algorithm to solve approx-CVP

- the distribution of signatures leaks information of $\mathbf{B}$, Insecure!
- broken by Nguyen and Regev [NR06][2]

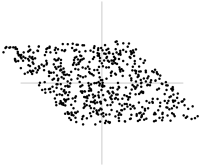


[2] [NR06]: Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. Nguyen and Regev.

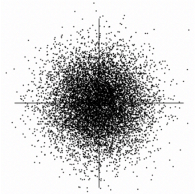
GPV framework

[GPV08][3] designed a provably secure hash-and-sign framework.

- deterministic Babai's algorithm $\Rightarrow$ trapdoor sampler
- signing $\Leftrightarrow$ lattice Gaussian sampling (prevent secret leakage)



$\Rightarrow$



Falcon is an efficient instantiation of the GPV hash-and-sign framework over NTRU lattices.

[3] [GPV08]: Trapdoors for Hard Lattices and New Cryptographic Constructions. Gentry, Peikert and Vaikuntanathan.

Falcon

Falcon = GPV + NTRU trapdoor + Fast Fourier Gaussian sampler (FFO)

- NTRU trapdoor $\Rightarrow$ compactness
- Fast Fourier Gaussian sampler $\Rightarrow$ fast signing

Falcon

Falcon = GPV + NTRU trapdoor + Fast Fourier Gaussian sampler (FFO)

- NTRU trapdoor $\Rightarrow$ compactness
- Fast Fourier Gaussian sampler $\Rightarrow$ fast signing

Falcon's cons

- overall complexity $\Rightarrow$ hard to implement it correctly
- key generation and signing heavily rely on floating-point arithmetics

Floating-point arithmetics in Falcon

For Falcon

- signing $\Rightarrow$ ring-efficient Klein-GPV sampler
- Klein-GPV sampler $\Rightarrow$ floating-point arithmetics (FPA)
- for FPA in Falcon, IEEE-754 double precision is sufficient

[4] https://doi.org/10.6028/NIST.IR.8413

Floating-point arithmetics in Falcon

For Falcon

- signing $\Rightarrow$ ring-efficient Klein-GPV sampler
- Klein-GPV sampler $\Rightarrow$ floating-point arithmetics (FPA)
- for FPA in Falcon, IEEE-754 double precision is sufficient

Status report on the third round of the NIST PQC standardization process[4]

NIST's concern

In particular, simplicity was an important factor in NIST's evaluation of FALCON, with the concern that the use of floating point arithmetic and more complex implementation could lead to errors that might affect security.

[4] https://doi.org/10.6028/NIST.IR.8413

Floating-point errors sensitivity analysis

Gaussian samplers in Falcon's signing procedure

$$\boxed{\begin{array}{c}\text{FFOSampler:}\\ \mathbf{s} \leftarrow D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}\end{array}} \longleftarrow \boxed{\begin{array}{c}\text{SamplerZ:}\\ y \leftarrow D_{\mathbb{Z},\sigma,c}\end{array}} \longleftarrow \boxed{\begin{array}{c}\text{BaseSampler:}\\ y_+ \leftarrow D^+_{\mathbb{Z}^+,\sigma_{\max},0}\end{array}}$$

Klein-GPV sampler

Input: NTRU basis $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$, center $\mathbf{c}$ and $\sigma \geq \|\mathbf{B}\|_{GS} \cdot \eta_\epsilon(\mathbb{Z})$

Output: a lattice point $\mathbf{u}$ follows a distribution close to $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$

```
u_n ← 0, c_n ← c
for i = n − 1, ···, 0 do
    c'_i = ⟨c_i, b̃_i⟩/⟨b̃_i, b̃_i⟩
    z_i ← D_{Z,σ_i,c'_i} where σ_i = σ/‖b̃_i‖
    c_{i−1} ← c_i − z_i b_i, u_{i−1} ← u_i + z_i b_i
return u_0
```

$\mathbf{u}_n \leftarrow \mathbf{0}, \mathbf{c}_n \leftarrow \mathbf{c}$

for $i = n-1, \cdots, 0$ **do**

  $c'_i = \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_i \rangle$

  $z_i \leftarrow D_{\mathbb{Z},\sigma_i,c'_i}$ where $\sigma_i = \sigma / \|\tilde{\mathbf{b}}_i\|$

  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \mathbf{b}_i, \mathbf{u}_{i-1} \leftarrow \mathbf{u}_i + z_i \mathbf{b}_i$

return $\mathbf{u}_0$

Gaussian samplers in Falcon's signing procedure

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{FFOSampler:}\\ \mathbf{s} \leftarrow D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}\end{array}} & \boxed{\begin{array}{c}\text{SamplerZ:}\\ y \leftarrow D_{\mathbb{Z},\sigma,c}\end{array}} \arrow[l] & \boxed{\begin{array}{c}\text{BaseSampler:}\\ y_+ \leftarrow D^+_{\mathbb{Z}^+,\sigma_{\max},0}\end{array}} \arrow[l]
\end{tikzcd}$$

SamplerZ (one-dimensional integer Gaussian sampler)

Input: A center c and $\sigma \in [\sigma_{\min}, \sigma_{\max}]$

Output: An integer z derived from a distribution close to $D_{\mathbb{Z},\sigma,c}$

$r \leftarrow c - \lfloor c \rfloor$

$y_+ \leftarrow$ BaseSampler()

$b \xleftarrow{\$} \{0, 1\}$

$y \leftarrow b + (2b - 1)y_+$

$x \leftarrow \frac{(y-r)^2}{2\sigma^2} - \frac{y_+^2}{2\sigma_{\max}^2}$

return $z \leftarrow y + \lfloor c \rfloor$ with probability $\frac{\sigma_{\min}}{\sigma} \cdot \exp(-x)$, otherwise restart.

Floating-point error sensitivity of SamplerZ

Our analysis focus the execution of SamplerZ, rather than the distribution.

SamplerZ (one-dimensional integer Gaussian sampler)

Input: A center c and $\sigma \in [\sigma_{\min}, \sigma_{\max}]$

Output: An integer z derived from a distribution close to $D_{\mathbb{Z},\sigma,c}$

1: $r \leftarrow c - \lfloor c \rceil$, $y_+ \leftarrow$ BaseSampler()

2: $b \xleftarrow{\$} \{0,1\}$, $y \leftarrow b + (2b-1)y_+$

3: $x \leftarrow \frac{(y-r)^2}{2\sigma^2} - \frac{y_+^2}{2\sigma_{\max}^2}$

4: **return** $z \leftarrow y + \lfloor c \rceil$ with probability $\frac{\sigma_{\min}}{\sigma} \cdot \exp(-x)$, otherwise restart.

Sensitivity of the centers

Let c and c' be two close floating-point numbers. For same σ and randomness,

1. if $\lfloor c \rceil = \lfloor c' \rceil$, then SamplerZ$(\sigma, c)$ and SamplerZ(σ, c') have the same execution with overwhelming probability;
2. if $\lfloor c \rceil \neq \lfloor c' \rceil$, then SamplerZ$(\sigma, c)$ and SamplerZ(σ, c') have an inconsistent execution.

Nearly-integer center in Gaussian sampling

For a nearly-integer center, SamplerZ is sensitive to floating-point errors!

[5] [GHN06]: Symplectic Lattice Reduction and NTRU. Gama, Howgrave-Graham and Nguyen.

Nearly-integer center in Gaussian sampling

For a nearly-integer center, SamplerZ is sensitive to floating-point errors!

For FFOSampler, we have: $c_i = \frac{\langle \mathbf{c}_i, \widetilde{\mathbf{b}}_{2n-1-i} \rangle}{\|\widetilde{\mathbf{b}}_{2n-1-i}\|^2}$ where $\mathbf{c}_i \in \mathbb{Z}^n$.

[5] [GHN06]: Symplectic Lattice Reduction and NTRU. Gama, Howgrave-Graham and Nguyen.

Nearly-integer center in Gaussian sampling

For a nearly-integer center, SamplerZ is sensitive to floating-point errors!

For FFOSampler, we have: $c_i = \frac{\langle \mathbf{c}_i, \widetilde{\mathbf{b}}_{2n-1-i} \rangle}{\|\widetilde{\mathbf{b}}_{2n-1-i}\|^2}$ where $\mathbf{c}_i \in \mathbb{Z}^n$.

Thus,

- $\Pr[c_0 \in \mathbb{Z}] = \Pr[c_1 \in \mathbb{Z}] = \frac{1}{q}$ (NTRU symplecticity [GHN06][5])
- $\Pr[c_{2n-2} \in \mathbb{Z}] = \Pr[c_{2n-1} \in \mathbb{Z}] = \frac{1}{\|(g,-f)\|^2} \approx \frac{1}{1.17^2 \cdot q}$
- $\Pr_{i \notin \{0,1,2n-2,2n-1\}}[c_i \in \mathbb{Z}] \approx 0$

[5] [GHN06]: Symplectic Lattice Reduction and NTRU. Gama, Howgrave-Graham and Nguyen.

Nearly-integer center in Gaussian sampling

For a nearly-integer center, SamplerZ is sensitive to floating-point errors!

For FFOSampler, we have: $c_i = \frac{\langle \mathbf{c}_i, \widetilde{\mathbf{b}}_{2n-1-i} \rangle}{\|\widetilde{\mathbf{b}}_{2n-1-i}\|^2}$ where $\mathbf{c}_i \in \mathbb{Z}^n$.

Thus,

- $\Pr[c_0 \in \mathbb{Z}] = \Pr[c_1 \in \mathbb{Z}] = \frac{1}{q}$ (NTRU symplecticity [GHN06][5])
- $\Pr[c_{2n-2} \in \mathbb{Z}] = \Pr[c_{2n-1} \in \mathbb{Z}] = \frac{1}{\|(g,-f)\|^2} \approx \frac{1}{1.17^2 \cdot q}$
- $\Pr_{i \notin \{0,1,2n-2,2n-1\}}[c_i \in \mathbb{Z}] \approx 0$

For $i \in \{0, 1, 2n-2, 2n-1\}$, $1/10000 < \Pr[c_i \in \mathbb{Z}] < 1/20000$.

[5][GHN06]: Symplectic Lattice Reduction and NTRU. Gama, Howgrave-Graham and Nguyen.

Exploiting FPA discrepancies

Key recovery from signature discrepancies

For a syndrome $\mathbf{u} = \text{Hash}(\text{msg})$, Falcon's signing inherently samples an integer vector $\mathbf{z} = (z_0, z_1) \in \mathcal{R}^2$ and outputs a short signature:

$$\mathbf{s} = \mathbf{u} - \mathbf{z} \cdot \mathbf{B}_{f,g} = \mathbf{u} - \mathbf{z} \cdot \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}.$$

Key recovery from signature discrepancies

For a syndrome $\mathbf{u} = \mathsf{Hash}(\mathsf{msg})$, Falcon's signing inherently samples an integer vector $\mathbf{z} = (z_0, z_1) \in \mathcal{R}^2$ and outputs a short signature:

$$\mathbf{s} = \mathbf{u} - \mathbf{z} \cdot \mathbf{B}_{f,g} = \mathbf{u} - \mathbf{z} \cdot \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}.$$

Due to FPA errors, the difference for same $\mathbf{u}$: $\Delta\mathbf{s} = \Delta\mathbf{z} \cdot \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$, i.e.

$$\Delta s_0 = s_0 - s_0' = (z_0 - z_0') \cdot g + (z_1 - z_1') \cdot G,$$
$$\Delta s_1 = s_1 - s_1' = (z_0 - z_0') \cdot (-f) + (z_1 - z_1') \cdot (-F).$$

Key recovery from signature discrepancies

For a syndrome $\mathbf{u} = \mathsf{Hash}(\mathsf{msg})$, Falcon's signing inherently samples an integer vector $\mathbf{z} = (z_0, z_1) \in \mathcal{R}^2$ and outputs a short signature:

$$\mathbf{s} = \mathbf{u} - \mathbf{z} \cdot \mathbf{B}_{f,g} = \mathbf{u} - \mathbf{z} \cdot \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}.$$

Due to FPA errors, the difference for same $\mathbf{u}$: $\Delta\mathbf{s} = \Delta\mathbf{z} \cdot \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$, i.e.

$$\Delta s_0 = s_0 - s_0' = (z_0 - z_0') \cdot g + (z_1 - z_1') \cdot G,$$
$$\Delta s_1 = s_1 - s_1' = (z_0 - z_0') \cdot (-f) + (z_1 - z_1') \cdot (-F).$$

Using simple exhaustive search, the key $(g, -f)$ can be exactly recovered when the FPA instability only occurs in the last two calls of SamplerZ.

Cryptanalytic impacts

Little impact on plain Falcon signature

- repeated randomness

[6] https://github.com/algorand/falcon

[7] [AAB+24]: Aggregating Falcon Signatures with LaBRADOR. Aardal, Aranha, Boudgoust, Kolby and Takahashi.

[8] [ZMS+24]: Quantum-safe HIBE: does it cost a LATTE? Zhao, McCarthy, Steinfeld, Sakzad and O'Neill.

Cryptanalytic impacts

Little impact on plain Falcon signature

- repeated randomness

Critical impact on deterministic Falcon (specified by Lazar and Peikert[6])

- SNARK-friendly signature aggregation [AAB+24][7]

[6] https://github.com/algorand/falcon

[7] [AAB+24]: Aggregating Falcon Signatures with LaBRADOR. Aardal, Aranha, Boudgoust, Kolby and Takahashi.

[8] [ZMS+24]: Quantum-safe HIBE: does it cost a LATTE? Zhao, McCarthy, Steinfeld, Sakzad and O'Neill.

Cryptanalytic impacts

Little impact on plain Falcon signature

- repeated randomness

Critical impact on deterministic Falcon (specified by Lazar and Peikert[6])

- SNARK-friendly signature aggregation [AAB+24][7]

Critical impact on Falcon-based IBE

- LATTE (H)IBE [ZMS+24][8] (considered for UK NCSC and ETSI standardization)

[6] https://github.com/algorand/falcon

[7] [AAB+24]: Aggregating Falcon Signatures with LaBRADOR. Aardal, Aranha, Boudgoust, Kolby and Takahashi.

[8] [ZMS+24]: Quantum-safe HIBE: does it cost a LATTE? Zhao, McCarthy, Steinfeld, Sakzad and O'Neill.

Sources of FPA discrepancies

Sources of FPA discrepancies

FPA does not obey associativity or distributivity. $\Rightarrow$ Weak determinism

Sources of FPA discrepancies

FPA does not obey associativity or distributivity. $\Rightarrow$ Weak determinism

We experimentally validate two possible sources for such FPA discrepancies in Falcon.

- two almost but not quite equivalent signing modes: dynamic mode and tree mode
- different floating-point instructions: FMA (Fused Multiply-Add)

Discrepancies between two signing modes

Different computation order in polynomial "split" operation:

In the dynamic mode (recursive layer $n = 4$):

$$t_1[0] = \frac{1}{2} \mathring{\times} \Big(\frac{1}{\sqrt{2}} \mathring{\times} \big(t[0] \mathring{-} t[1]\big) \mathring{-} \big(-\frac{1}{\sqrt{2}}\big) \mathring{\times} \big(t[2] \mathring{-} t[3]\big)\Big),$$

$$t_1[1] = \frac{1}{2} \mathring{\times} \Big(\big(-\frac{1}{\sqrt{2}}\big) \mathring{\times} \big(t[0] \mathring{-} t[1]\big) \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \big(t[2] \mathring{-} t[3]\big)\Big).$$

In the tree mode (recursive layer $n = 4$):

$$t_1[0] = \frac{1}{2\sqrt{2}} \mathring{\times} \Big(\big(t[0] \mathring{-} t[1]\big) \mathring{+} \big(t[2] \mathring{-} t[3]\big)\Big),$$

$$t_1[1] = \frac{1}{2\sqrt{2}} \mathring{\times} \Big(\big(t[2] \mathring{-} t[3]\big) \mathring{-} \big(t[0] \mathring{-} t[1]\big)\Big).$$

Discrepancies between two signing modes

Different computation order in polynomial "split" operation:

In the dynamic mode (recursive layer $n = 4$):

$$t_1[0] = \frac{1}{2} \mathring{\times} \left(\frac{1}{\sqrt{2}} \mathring{\times} \left(t[0] \mathring{-} t[1] \right) \mathring{-} \left(-\frac{1}{\sqrt{2}} \right) \mathring{\times} \left(t[2] \mathring{-} t[3] \right) \right),$$
$$t_1[1] = \frac{1}{2} \mathring{\times} \left(\left(-\frac{1}{\sqrt{2}} \right) \mathring{\times} \left(t[0] \mathring{-} t[1] \right) \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \left(t[2] \mathring{-} t[3] \right) \right).$$

In the tree mode (recursive layer $n = 4$):

$$t_1[0] = \frac{1}{2\sqrt{2}} \mathring{\times} \left(\left(t[0] \mathring{-} t[1] \right) \mathring{+} \left(t[2] \mathring{-} t[3] \right) \right),$$
$$t_1[1] = \frac{1}{2\sqrt{2}} \mathring{\times} \left(\left(t[2] \mathring{-} t[3] \right) \mathring{-} \left(t[0] \mathring{-} t[1] \right) \right).$$

FPA is not distributive, the computations of t_1 may evaluate different values in two signing modes, which might affect the centers of SamplerZ.

Discrepancies between two signing modes

Different computation order in polynomial "merge" operation:

In the dynamic mode (recursive layer $n = 4$):

$$t[0] = t_0[0] \mathring{+} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1]\right), \qquad t[2] = t_0[1] \mathring{+} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1]\right),$$

$$t[1] = t_0[0] \mathring{-} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1]\right), \qquad t[3] = t_0[1] \mathring{-} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1]\right).$$

In the tree mode (recursive layer $n = 4$):

$$t[0] = t_0[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{-} t_1[1]\right), \qquad t[2] = t_0[1] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{+} t_1[1]\right),$$

$$t[1] = t_0[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{-} t_1[1]\right), \qquad t[3] = t_0[1] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{+} t_1[1]\right).$$

Discrepancies between two signing modes

Different computation order in polynomial "merge" operation:

In the dynamic mode (recursive layer $n = 4$):

$$t[0] = t_0[0] \mathring{+} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1] \right), \qquad t[2] = t_0[1] \mathring{+} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1] \right),$$

$$t[1] = t_0[0] \mathring{-} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1] \right), \qquad t[3] = t_0[1] \mathring{-} \left(\frac{1}{\sqrt{2}} \mathring{\times} t_1[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} t_1[1] \right).$$

In the tree mode (recursive layer $n = 4$):

$$t[0] = t_0[0] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{-} t_1[1] \right), \qquad t[2] = t_0[1] \mathring{+} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{+} t_1[1] \right),$$

$$t[1] = t_0[0] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{-} t_1[1] \right), \qquad t[3] = t_0[1] \mathring{-} \frac{1}{\sqrt{2}} \mathring{\times} \left(t_1[0] \mathring{+} t_1[1] \right).$$

FPA is again not distributive, the computations of t may evaluate different values in two signing modes, which might affect the centers of SamplerZ.

Experimental results

We in total tested 10 instances with N signature queries:

- reference implementation: fpemu
- optimization ones: fpnative, avx2, avx2_fma

$N \times 10^{-3}$	10	20	30	40	50	60	70	80	90	100
fpemu_det_512	1	4	6	6	6	7	8	8	8	8
fpnative_det_512	2	5	7	7	8	8	10	10	10	10
avx2_det_512	1	6	8	8	8	8	8	9	9	9
avx2_fma_det_512	2	4	6	7	8	8	8	9	9	9
fpemu_det_1024	5	6	6	6	7	7	7	8	8	9
fpnative_det_1024	2	2	3	3	4	6	7	8	8	8
avx2_det_1024	3	4	5	5	6	6	7	7	7	7
avx2_fma_det_1024	1	3	4	7	8	9	9	9	10	10

Within 10,000 signature pairs, one can mount a full key recovery.

Discrepancies caused by FMA

FMA (Fused Multiply-Add) floating-point instructions

- disabled: round(round($a \cdot b$) $+ c$), round(round($a \cdot b$) $- c$)
- enabled: round($a \cdot b + c$), round($a \cdot b - c$)

Discrepancies caused by FMA

FMA (Fused Multiply-Add) floating-point instructions

- disabled: $\text{round}(\text{round}(a \cdot b) + c)$, $\text{round}(\text{round}(a \cdot b) - c)$
- enabled: $\text{round}(a \cdot b + c)$, $\text{round}(a \cdot b - c)$

FMA instructions are more accurate (just one rounding only) $\Rightarrow$ FPA discrepancies

Discrepancies caused by FMA

FMA (Fused Multiply-Add) floating-point instructions

- disabled: round(round($a \cdot b$) $+ c$), round(round($a \cdot b$) $- c$)
- enabled: round($a \cdot b + c$), round($a \cdot b - c$)

FMA instructions are more accurate (just one rounding only) $\Rightarrow$ FPA discrepancies

In both signing and key generation, FMA instructions are widely used.

Experimental results

We also tested 10 instances with N signature queries:

- dynamic mode: sign_dyn
- tree mode: sign_tree

$N \times 10^{-3}$	10	20	30	40	50	60	70	80	90	100
sign_dyn_512	2	4	5	7	7	8	9	9	10	10
sign_tree_512	4	6	6	8	8	8	8	9	9	9
sign_dyn_1024	2	4	6	8	9	9	9	9	9	9
sign_tree_1024	4	8	8	8	9	9	9	9	9	10

Exact secret key can also be recovered within 10,000 signature pairs.

Countermeasures

Countermeasures

We propose a NewSamplerZ with the stability of FPA errors.

- floor operation $\Rightarrow$ rounding to nearest integer, i.e. $\mathbb{Z} \Rightarrow 1/2 + \mathbb{Z}$
- restrict $\|(g, -f)\|^2$ to be an *odd* integer in key generation

Countermeasures

We propose a NewSamplerZ with the stability of FPA errors.

- floor operation $\Rightarrow$ rounding to nearest integer, i.e. $\mathbb{Z} \Rightarrow 1/2 + \mathbb{Z}$
- restrict $\|(g, -f)\|^2$ to be an *odd* integer in key generation

Other simpler tricks

- reordering computation order in tree mode / avoid reordered codes
- FMA disabled

Countermeasures

We propose a NewSamplerZ with the stability of FPA errors.

- floor operation $\Rightarrow$ rounding to nearest integer, i.e. $\mathbb{Z} \Rightarrow 1/2 + \mathbb{Z}$
- restrict $\|(g, -f)\|^2$ to be an *odd* integer in key generation

Other simpler tricks

- reordering computation order in tree mode / avoid reordered codes
- FMA disabled

To avoid FPA discrepancies, we suggest in the same settings:

- the same FPA implementation + the same signing mode

Conclusion

Conclusion

FPA carelessness + Deterministic Falcon = Attack[9]

[9] Artifacts: https://github.com/lxhcrypto/Det_Falcon_KATs

Thank you!