Power Analysis Attacks on Falcon

The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon [GMRR22]

Mehdi Jelassi, Adrien Bouquet

2025

Overview

- Preliminaries
- Falcon signature scheme
- Power analysis on the preimage computation
- Hidden Parallelepiped attack on the trapdoor sampler
- Summary

Intro

- **Falcon**: **Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU.

Intro

- **Falcon**: **Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU.
- It is a **hash-and-sign** lattice-based signature scheme.

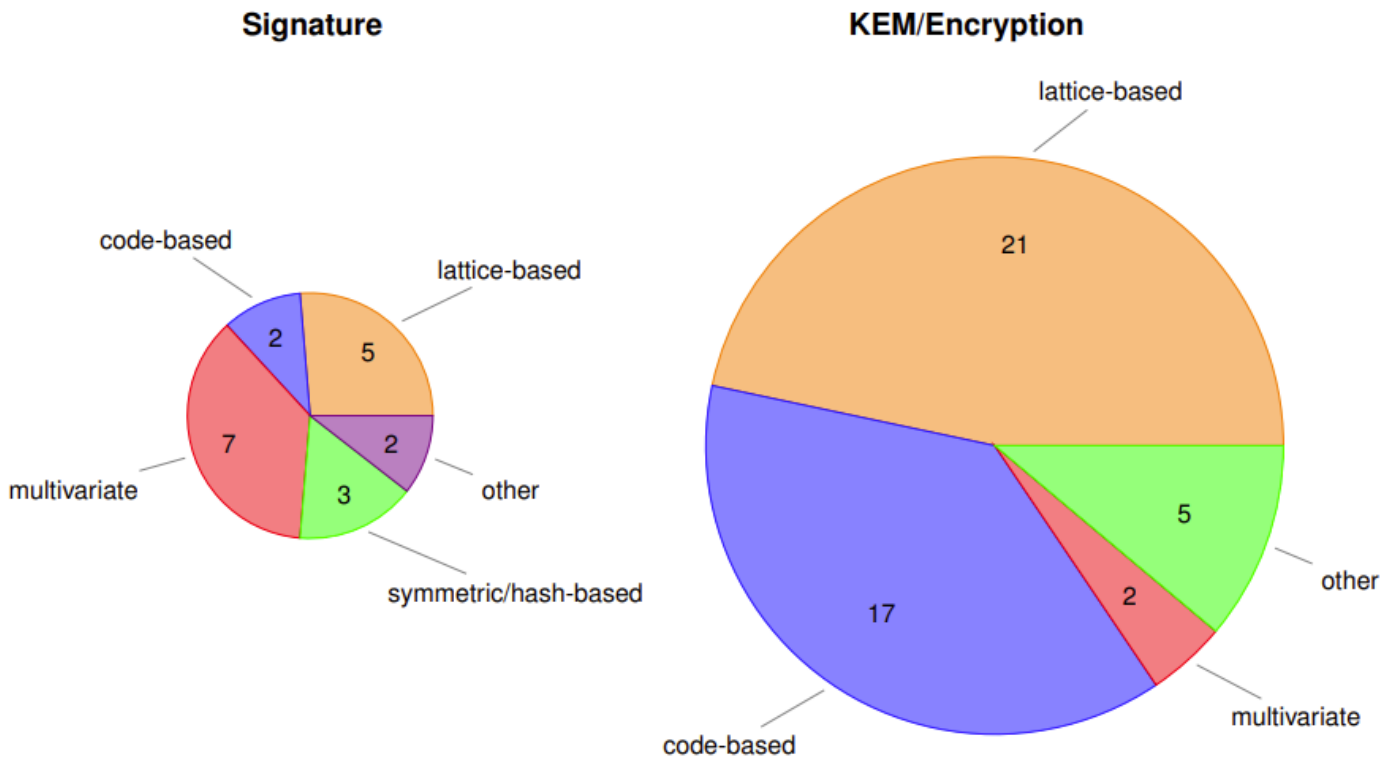
Intro

- **Falcon**: **Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU.
- It is a **hash-and-sign** lattice-based signature scheme.
- Falcon = GPV Framework + NTRU lattices + Fast Fourier sampling.

Intro

- **Falcon**: **Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU.
- It is a **hash-and-sign** lattice-based signature scheme.
- Falcon = GPV Framework + NTRU lattices + Fast Fourier sampling.
- Main advantage: **compactness**. Not known any post-quantum signature schemes getting $|pk| + |sig|$ = (bitsize of the public key + bitsize of a signature) to be as small as Falcon does.

NIST PQC Round #1 Submission



source: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/PQCrypto-April2018_Moody.pdf

Figure: Slide 808 Cryptography and Security course

NIST PQC 2022 Selected Algorithms

	Signature		KEM/Encryption	
Selected	CRYSTALS-DILITHIUM FALCON SPHINCS+	lattice lattice hash	CRYSTALS-KYBER	lattice

Next:

- KYBER becomes ML-KEM in FIPS 203 (Module-Lattice)
- DILITHIUM becomes ML-DSA in FIPS 204 (Module-Lattice)
- SPINCS+ becomes SLH-DSA in FIPS 205 (Stateless Hash)
- FALCON becomes FN-DSE (later)

Figure: Slide 809 Cryptography and Security course

Power Analysis

- **Threat model:** adversary has physical access to the device and captures EM and power measurements while the key-dependent computations are carried out.

"Power"
measure some physical quantity influenced by execution, e.g., power, EM, ...

- **Simple Power Analysis**
- **Differential Power Analysis**
- **Correlation Power Analysis**
- **Other leakage types**

Side channel attacks[1/2]

- Side-channel attack (SCA): a way to break a cryptosystem by exploiting physical information leaked during computations.

Side channel attacks[1/2]

- Side-channel attack (SCA): a way to break a cryptosystem by exploiting physical information leaked during computations.

- Simple power analysis (SPA): The power usage tells what kind of operation is performed (e.g. square and multiply algorithm in decryption in RSA)

Side channel attacks[1/2]

- Side-channel attack (SCA): a way to break a cryptosystem by exploiting physical information leaked during computations.
- Simple power analysis (SPA): The power usage tells what kind of operation is performed (e.g. square and multiply algorithm in decryption in RSA)
- **Differential Power Analysis (DPA):** a SCA that extracts secret information (e.g. signing key) by measuring a device's power consumption during computations.

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.
  - Attack small parts (subkeys) of the secret key:

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.
  - Attack small parts (subkeys) of the secret key:
    - for each guess the subkey and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model (like **Hamming weight of a string**:= number of symbols that are different from the zero string)

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.
  - Attack small parts (subkeys) of the secret key:
    - for each guess the subkey and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model (like **Hamming weight of a string**:= number of symbols that are different from the zero string)
    - Calculate the correlation between the modeled and actual power consumption. Do this for every data point in the traces.

Side channel attacks[2/2]

- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.
  - Attack small parts (subkeys) of the secret key:
    - for each guess the subkey and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model (like **Hamming weight of a string**:= number of symbols that are different from the zero string)
    - Calculate the correlation between the modeled and actual power consumption. Do this for every data point in the traces.
    - Decide which subkey guess correlates best to the measured traces.

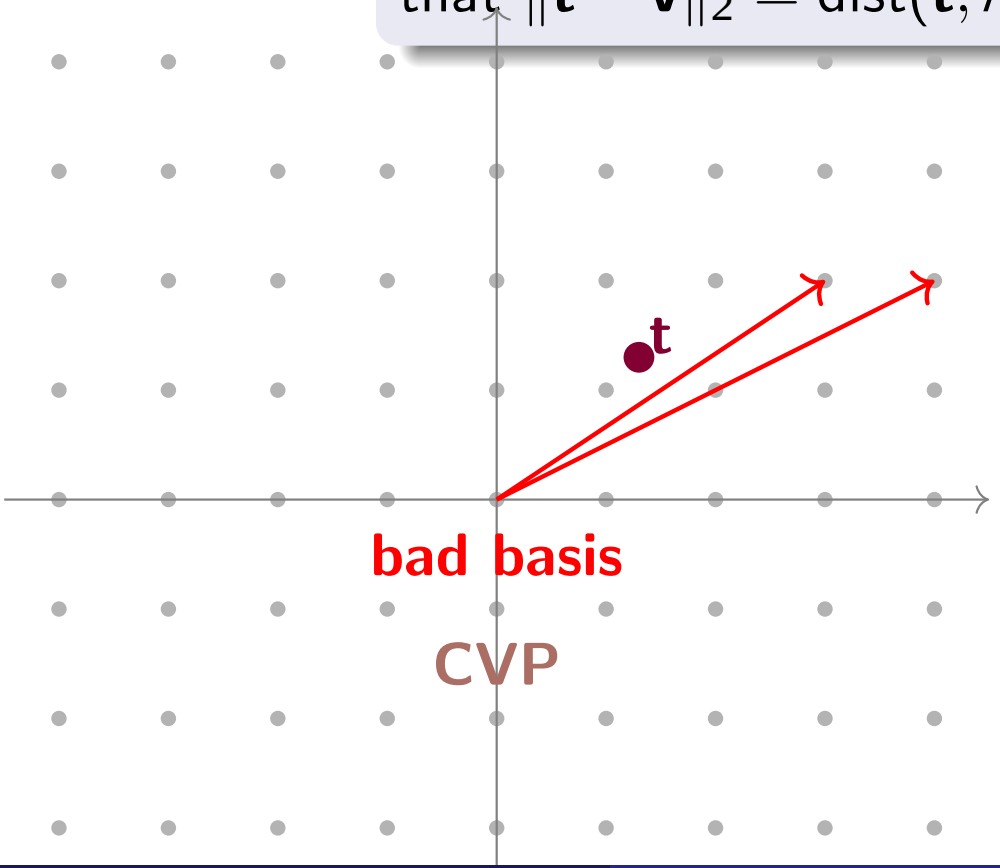
Side channel attacks[2/2]

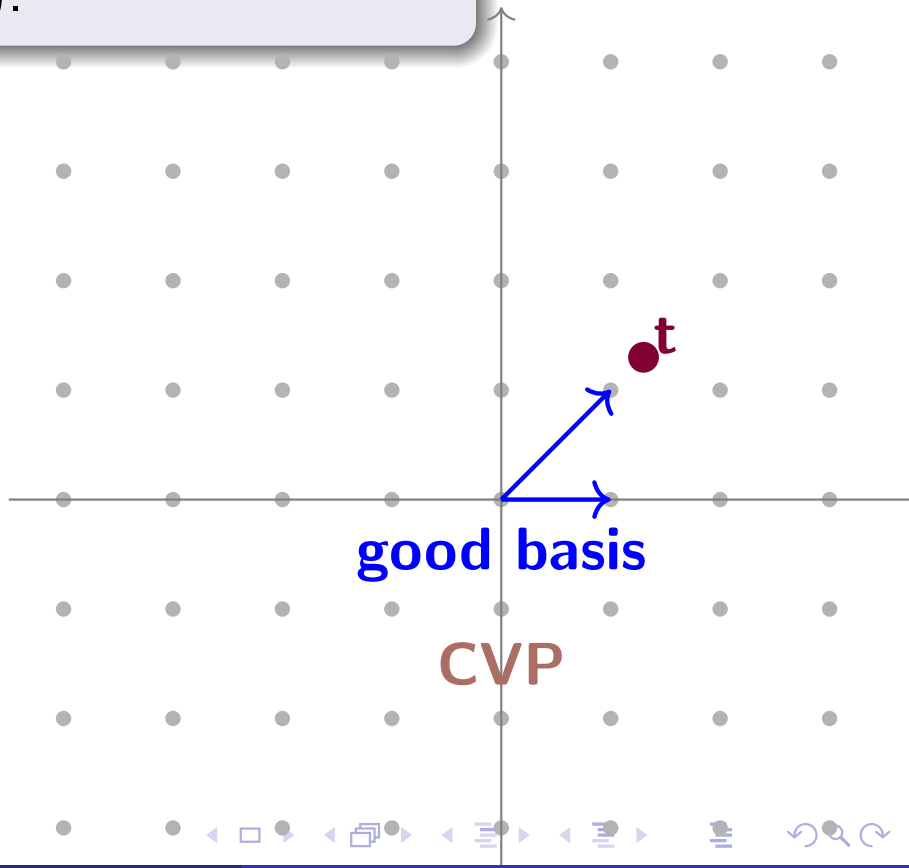
- **Correlation power analysis (CPA):** (e.g. of DPA)
  - Get the victim to sign several different plaintexts. Record a trace of the victim's power consumption during each of these signatures.
  - Attack small parts (subkeys) of the secret key:
    - for each guess the subkey and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model (like **Hamming weight of a string**:= number of symbols that are different from the zero string)
    - Calculate the correlation between the modeled and actual power consumption. Do this for every data point in the traces.
    - Decide which subkey guess correlates best to the measured traces.
  - Put together the best subkey guesses to obtain the full secret key.

Lattice-based cryptography [1/2]

Security from hard problem: Closest Vector Problem (CVP)

Given $\mathbf{B}$ a basis of a lattice $\Lambda \subset \mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$, find a $\mathbf{v} \in \Lambda$ such that $\|\mathbf{t} - \mathbf{v}\|_2 = \text{dist}(\mathbf{t}, \Lambda)$.





Lattice-based cryptography [2/2]

- CVP is easy to solve with a good basis(:=short and reasonably orthogonal) but hard with a bad basis.
- **Signature scheme:**
  - Convert the message to sign to a vector $c \in \mathcal{R}^n$
  - Use the good basis (secret key) to solve CVP
  - Anyone can verify the signature $\mathbf{v}$ with a bad basis (public key)

Remark: It is hard to derivate the good basis from the bad basis.

Hidden Parallelepiped Problem (HPP)

- Transcript Analysis: each document, signature pair (d, s) reveals some information on the sk, at least it reveals that $Sign(sk, d) = (d, s)$. $\rightsquigarrow$ sufficiently many transcripts may reveal information about the signing key or how to forge another document.

Hidden Parallelepiped Problem (HPP)

- Transcript Analysis: each document, signature pair (d, s) reveals some information on the sk, at least it reveals that $Sign(sk, d) = (d, s)$. $\rightsquigarrow$ sufficiently many transcripts may reveal information about the signing key or how to forge another document.
- [NR06] broke GGH signature scheme (and NTRUSign) by solving the Hidden Parallelepiped Problem (HPP). Need roughly n^2 signatures to break instances of GGH in a lattice of dimension n.

Hidden Parallelepiped Problem (HPP)

- Transcript Analysis: each document, signature pair (d, s) reveals some information on the sk, at least it reveals that $Sign(sk, d) = (d, s)$. $\rightsquigarrow$ sufficiently many transcripts may reveal information about the signing key or how to forge another document.
- [NR06] broke GGH signature scheme (and NTRUSign) by solving the Hidden Parallelepiped Problem (HPP). Need roughly n^2 signatures to break instances of GGH in a lattice of dimension n.

HPP

Recover $\mathbf{B}$ from independent samples drawn uniformly in $\mathcal{P}(\mathbf{B})$.

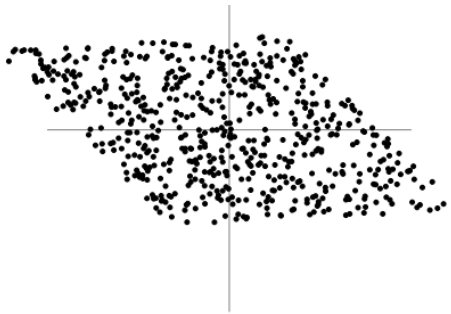


Fig. 1. The Hidden Parallelepiped Problem in dimension two.

Toward Falcon

- GGH is not secure anymore. Solution ? $\rightsquigarrow$ GPV framework

Toward Falcon

- GGH is not secure anymore. Solution ? $\rightsquigarrow$ GPV framework
- NTRU lattice:= lattice spanned by $B := \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ viewed in $\mathcal{R}^{2\times 2}$ with $\mathcal{R} := \frac{\mathbb{Z}_q[x]}{(x^n+1)}$. Note: $\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ and $B := \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ span the same lattice.
- **KeyGEN Falcon:** Draw $f, g \in \mathcal{R}$ with small coefficients, compute $F, G \in \mathcal{R}$ satisfying NTRU equation: $fG - gF = q \mod (x^n + 1)$. $h = gf^{-1} \mod q$ is the **public key** and f, g, F, G are the **secret keys**. Private Basis $B \in \mathcal{R}^{2n\times 2n}$, Public Basis $A \in \mathcal{R}^{2n}$ seen as $A := (1 \quad h^*) \in \mathcal{R}^2$ with $h^* := h(x^{-1})$.

Toward Falcon

- GGH is not secure anymore. Solution ? $\rightsquigarrow$ GPV framework
- NTRU lattice:= lattice spanned by $B := \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ viewed in $\mathcal{R}^{2\times 2}$ with $\mathcal{R} := \frac{\mathbb{Z}_q[x]}{(x^n+1)}$. Note: $\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ and $B := \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ span the same lattice.
- **KeyGEN Falcon:** Draw $f, g \in \mathcal{R}$ with small coefficients, compute $F, G \in \mathcal{R}$ satisfying NTRU equation: $fG - gF = q \mod (x^n + 1)$. $h = gf^{-1} \mod q$ is the **public key** and f, g, F, G are the **secret keys**. Private Basis $B \in \mathcal{R}^{2n\times 2n}$, Public Basis $A \in \mathcal{R}^{2n}$ seen as $A := (1 \quad h^*) \in \mathcal{R}^2$ with $h^* := h(x^{-1})$.
- NTRU problem: recover f, g from h.
- Key Recovery hard $\iff$ NTRU problem hard

Toward Falcon

- GGH is not secure anymore. Solution ? $\rightsquigarrow$ GPV framework
- NTRU lattice:= lattice spanned by $B := \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ viewed in $\mathcal{R}^{2\times 2}$ with $\mathcal{R} := \frac{\mathbb{Z}_q[x]}{(x^n+1)}$. Note: $\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ and $B := \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ span the same lattice.
- **KeyGEN Falcon:** Draw $f, g \in \mathcal{R}$ with small coefficients, compute $F, G \in \mathcal{R}$ satisfying NTRU equation: $fG - gF = q \mod (x^n + 1)$. $h = gf^{-1} \mod q$ is the **public key** and f, g, F, G are the **secret keys**. Private Basis $B \in \mathcal{R}^{2n\times 2n}$, Public Basis $A \in \mathcal{R}^{2n}$ seen as $A := (1 \quad h^*) \in \mathcal{R}^2$ with $h^* := h(x^{-1})$.
- NTRU problem: recover f, g from h.
- Key Recovery hard $\iff$ NTRU problem hard
- Falcon = GPV Framework + NTRU lattices + Fast Fourier sampling.

Falcon SIGN

Algorithm 1: FALCON.Sign(m, sk)

Input : A message m, a secret key $sk = (\hat{\mathbf{B}}, T)$
Output : A signature sig

1 $r \xleftarrow{\$} \{0,1\}^{320}$ uniformly
2 $c \leftarrow \texttt{HashToPoint}(r||m, q, n)$ $\quad c \in \mathcal{R}$
3 $\hat{\mathbf{t}} \leftarrow (\hat{c}, 0) \cdot \hat{\mathbf{B}}^{-1}$ • pre-image computation
4 **do**
5 $\quad \hat{\mathbf{v}} \leftarrow \texttt{ffSampling}(\hat{\mathbf{t}}, T)$ • trapdoor sampler
6 $\quad \hat{\mathbf{s}} \leftarrow (\hat{\mathbf{t}} - \hat{\mathbf{v}}) \cdot \hat{\mathbf{B}}$
7 **while** $\|\mathbf{s}\|^2 > \lfloor 2.42 \cdot n \cdot \sigma^2 \rfloor$ $\quad \sigma := \frac{1.17}{\pi\sqrt{2}} \cdot \sqrt{q \cdot \log\left(4n(1 + 2^{32} \cdot \sqrt{n/4})\right)}$;
8 **return** $sig := (r, \mathbf{s})$

- if $\|s\|^2 \leq \lfloor 2.42 \cdot n \cdot \sigma^2 \rfloor$, then the signature is accepted as valid. Otherwise, it is rejected.

Correlation Power Analysis on the preimage attack

- Original attack [KA21]: CPA on a polynomial multiplication in FFT between a public digest c and a private polynomial f: $\hat{c} \cdot \hat{f}$ since computing $\hat{t} \leftarrow (\hat{c}, 0) \cdot \hat{B}^{-1} \rightsquigarrow$ computing $\hat{c} \cdot \hat{f}$.

Correlation Power Analysis on the preimage attack

- Original attack [KA21]: CPA on a polynomial multiplication in FFT between a public digest c and a private polynomial f: $\hat{c} \cdot \hat{f}$ since computing $\hat{t} \leftarrow (\hat{c}, 0) \cdot \hat{B}^{-1} \rightsquigarrow$ computing $\hat{c} \cdot \hat{f}$.

- **2 main improvements:**

Correlation Power Analysis on the preimage attack

- Original attack [KA21]: CPA on a polynomial multiplication in FFT between a public digest c and a private polynomial f: $\hat{c} \cdot \hat{f}$ since computing $\hat{t} \leftarrow (\hat{c}, 0) \cdot \hat{B}^{-1} \rightsquigarrow$ computing $\hat{c} \cdot \hat{f}$.

- **2 main improvements:**
  - **Partial knowledge on $\hat{f} := FFT(f)$ leads to KR**: by experiments, only the 8 MSB (obtained by CPA) of the mantissa are required to derive the key.

Correlation Power Analysis on the preimage attack

- Original attack [KA21]: CPA on a polynomial multiplication in FFT between a public digest c and a private polynomial f: $\hat{c} \cdot \hat{f}$ since computing $\hat{t} \leftarrow (\hat{c}, 0) \cdot \hat{B}^{-1} \rightsquigarrow$ computing $\hat{c} \cdot \hat{f}$.

- **2 main improvements:**
  - **Partial knowledge on $\hat{f} := FFT(f)$ leads to KR**: by experiments, only the 8 MSB (obtained by CPA) of the mantissa are required to derive the key.
  - **halve the number of required traces** thanks to redundancy in multiplication of complex numbers.

Partial knowledge on $\hat{f} \implies$ KR

- **KR reduces** to recover f (actually $\hat{f}$ thanks to FFT^{-1}) from $\hat{c} \cdot \hat{f}$ because $h = g \cdot f^{-1} \mod q \rightsquigarrow$ recover g. As $fG - gF = q \mod (x^n + 1)$, $\rightsquigarrow$ recover F, G.

Partial knowledge on $\hat{f} \implies$ KR

- **KR reduces** to recover f (actually $\hat{f}$ thanks to FFT^{-1}) from $\hat{c} \cdot \hat{f}$ because $h = g \cdot f^{-1} \mod q \rightsquigarrow$ recover g. As $fG - gF = q \mod (x^n + 1)$, $\rightsquigarrow$ recover F, G.
- Float numbers in double precision (:=encoded over 64 bits):

$$(-1)^s \cdot 2^{e-1023} \cdot m$$

with [s=sign(1bit)| e= exponent(11bits)|m= mantissa(52bits)].

- sign, exponent and mantissa can be retrieved separately as they are computed separately in the multiplication.

Partial knowledge on $\hat{f} \implies$ KR

- **KR reduces** to recover f (actually $\hat{f}$ thanks to FFT^{-1}) from $\hat{c} \cdot \hat{f}$ because $h = g \cdot f^{-1} \mod q \rightsquigarrow$ recover g. As $fG - gF = q \mod (x^n + 1)$, $\rightsquigarrow$ recover F, G.
- Float numbers in double precision (:=encoded over 64 bits):

$$(-1)^s \cdot 2^{e-1023} \cdot m$$

with [s=sign(1bit)| e= exponent(11bits)|m= mantissa(52bits)].

- sign, exponent and mantissa can be retrieved separately as they are computed separately in the multiplication.
- [KA21] attack's retrieve the whole mantissa VS [GMRR22] improvement: enough to recover only the 8MSB of the mantissa.

Why is it a CPA ?

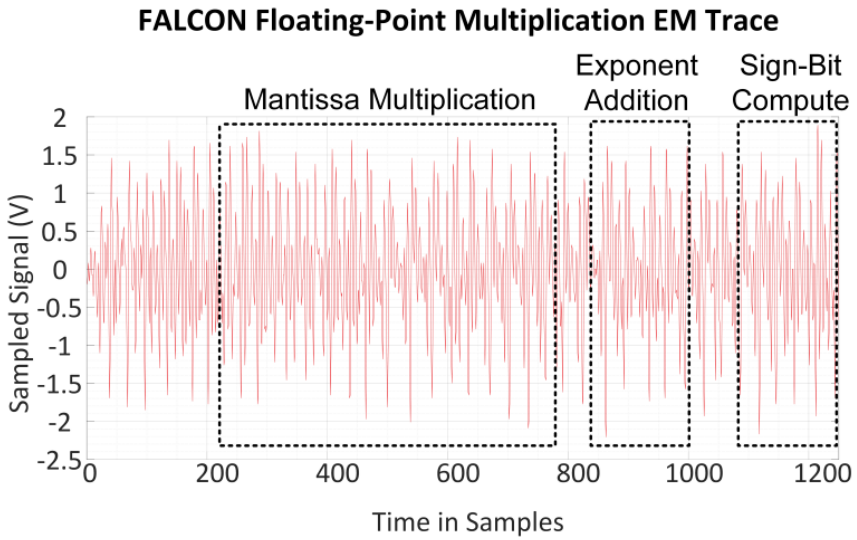


Figure: An example EM measurement trace from [KA21] showing the related mantissa, exponent, and sign computations.

- Generate many signatures pairs $(c_1, s_1), \ldots, (c_N, s_N)$.

Why is it a CPA ?

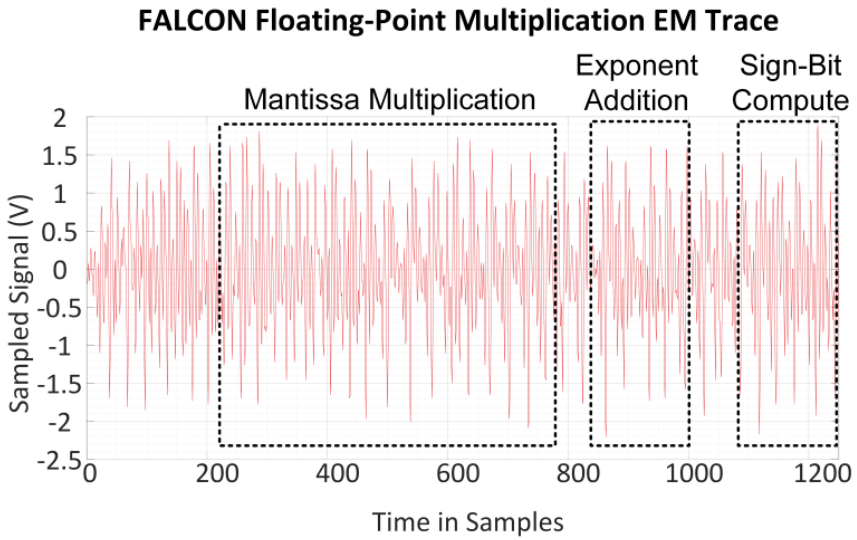


Figure: An example EM measurement trace from [KA21] showing the related mantissa, exponent, and sign computations.

- Generate many signatures pairs $(c_1, s_1), \ldots, (c_N, s_N)$.
- Target $\hat{c} \cdot \hat{f}$ computation. To recover $\mathrm{Re}(\hat{f}[i])$, recover sign, exponent and mantissa separately.

Why is it a CPA ?

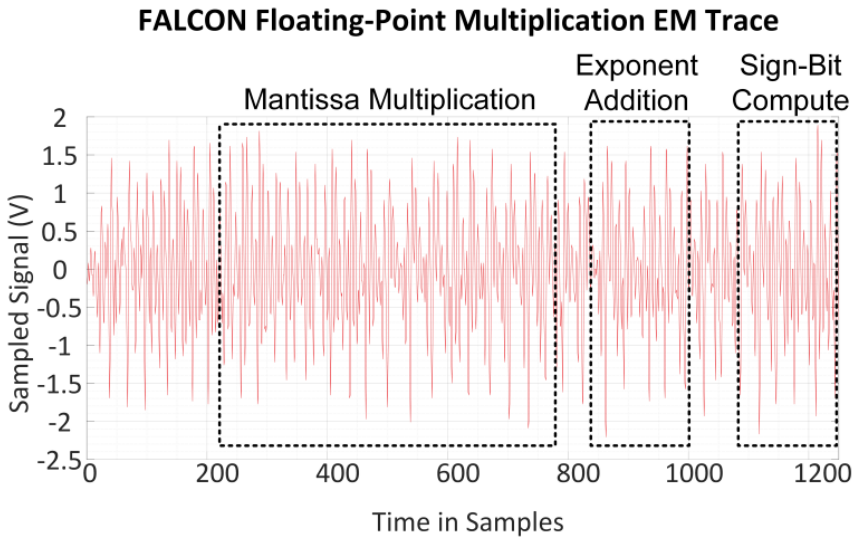


Figure: An example EM measurement trace from [KA21] showing the related mantissa, exponent, and sign computations.

- Generate many signatures pairs $(c_1, s_1), \ldots, (c_N, s_N)$.
- Target $\hat{c} \cdot \hat{f}$ computation. To recover $\mathrm{Re}(\hat{f}[i])$, recover sign, exponent and mantissa separately.
- To recover its mantissa, enough to recover only its 8 MSBs.

Why is it a CPA ?

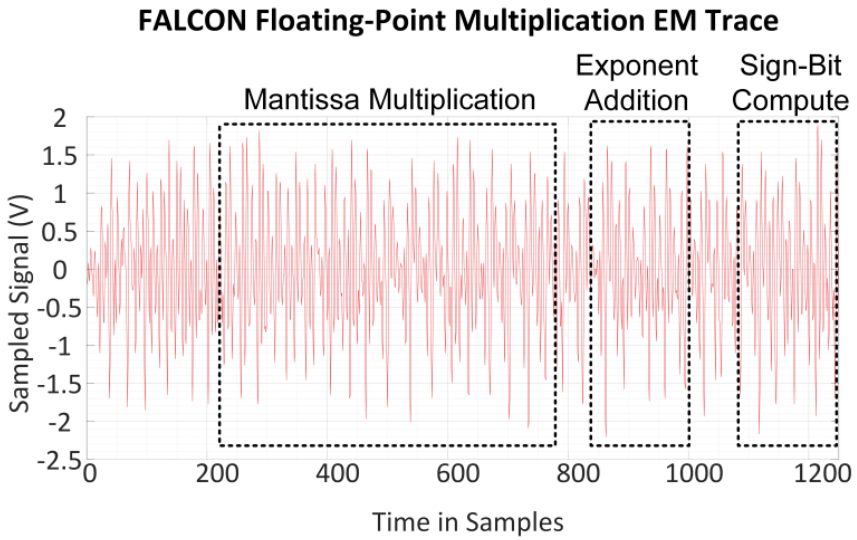


Figure: An example EM measurement trace from [KA21] showing the related mantissa, exponent, and sign computations.

- Generate many signatures pairs $(c_1, s_1), \ldots, (c_N, s_N)$.
- Target $\hat{c} \cdot \hat{f}$ computation. To recover $\mathrm{Re}(\hat{f}[i])$, recover sign, exponent and mantissa separately.
- To recover its mantissa, enough to recover only its 8 MSBs.
- guess a value for $\mathrm{Re}(\hat{f}[i])$ (since $\hat{c}$ is known) for which it best correlate its Hamming weight with the Hamming weight of our record.

Halve the number of required traces

- We consider the j-th digest c_j (in floating point). Let $\hat{c}_j[i] = \mathrm{Re}(\hat{c}_j[i]) + i\ \mathrm{Im}(\hat{c}_j[i])$, and $\hat{f}[i] = \mathrm{Re}(\hat{f}[i]) + i\ \mathrm{Im}(\hat{f}[i])$.

$$\begin{aligned}\hat{c}_j[i] \cdot \hat{f}[i] &= (\mathrm{Re}(\hat{c}_j[i]) + i\ \mathrm{Im}(\hat{c}_j[i])) \cdot (\mathrm{Re}(\hat{f}[i]) + i\ \mathrm{Im}(\hat{f}[i])) \\ &= \mathrm{Re}(\hat{c}_j[i])\,\mathrm{Re}(\hat{f}[i]) - \mathrm{Im}(\hat{c}_j[i])\,\mathrm{Im}(\hat{f}[i]) \\ &\quad + i\left(\mathrm{Re}(\hat{c}_j[i])\,\mathrm{Im}(\hat{f}[i]) + \mathrm{Im}(\hat{c}_j[i])\,\mathrm{Re}(\hat{f}[i])\right)\end{aligned}$$

Halve the number of required traces

- We consider the j-th digest c_j (in floating point). Let $\hat{c}_j[i] = \mathrm{Re}(\hat{c}_j[i]) + i\ \mathrm{Im}(\hat{c}_j[i])$, and $\hat{f}[i] = \mathrm{Re}(\hat{f}[i]) + i\ \mathrm{Im}(\hat{f}[i])$.

$$\begin{aligned}\hat{c}_j[i] \cdot \hat{f}[i] &= (\mathrm{Re}(\hat{c}_j[i]) + i\ \mathrm{Im}(\hat{c}_j[i])) \cdot (\mathrm{Re}(\hat{f}[i]) + i\ \mathrm{Im}(\hat{f}[i])) \\ &= \mathrm{Re}(\hat{c}_j[i])\,\mathrm{Re}(\hat{f}[i]) - \mathrm{Im}(\hat{c}_j[i])\,\mathrm{Im}(\hat{f}[i]) \\ &\quad + i\left(\mathrm{Re}(\hat{c}_j[i])\,\mathrm{Im}(\hat{f}[i]) + \mathrm{Im}(\hat{c}_j[i])\,\mathrm{Re}(\hat{f}[i])\right)\end{aligned}$$

$$\begin{cases}\mathrm{Re}(\hat{c}_j[i]) \cdot \mathrm{Re}(\hat{f}[i]), & \mathrm{Im}(\hat{c}_j[i]) \cdot \mathrm{Re}(\hat{f}[i]) \\ \mathrm{Im}(\hat{c}_j[i]) \cdot \mathrm{Im}(\hat{f}[i]), & \mathrm{Re}(\hat{c}_j[i]) \cdot \mathrm{Im}(\hat{f}[i])\end{cases} \Rightarrow \text{Recover } \mathrm{Re}(\hat{f}[i]), \mathrm{Im}(\hat{f}[i])$$

- better than [KA21] where all 4 four multiplications were used to recover $\mathrm{Re}(\hat{f}[i])$.

CPA on the Preimage computation

	Number of Traces	$\mathbb{P}$(**Recover one intermediate value**)
KA21	2000	0.5
GMRR22	2000	0.9

Comparison of probability of recovering one intermediate value between [KA21] and [GMRR22] attack with all multiplication patterns with 2000 traces with data (10×16 coefficients).

Unravelling the Hidden Parallelepiped Problem with side-channel information

- New side-channel attack on Falcon's implementation
- Target the Gaussian sampler
- Recover the private key

Attack's procedure

1. SPA on `BaseSampler` to obtain samples of z^+
2. `ffSampling` & Deformed HPP to recover private key
3. Direct Key recovery & Lattice Reduction
4. Proposed countermeasure

Discrete Gaussian distribution

- Gaussian function centered at $c \in \mathbb{R}^n$ of standard deviation $\sigma \in \mathbb{R}_{>0}$, for any $x \in \mathbb{R}^n$,

$$\rho_{\sigma,c}(x) := \exp\left(-\frac{\|x-c\|^2}{2\sigma^2}\right)$$

- Discrete Gaussian function for any $z \in \Lambda$, $D_{\Lambda,\sigma,c(z)} := \frac{\rho_{\sigma,c}(z)}{\sum_{x\in\Lambda}\rho_{\sigma,c}(x)}$

SPA on BaseSampler - Recall

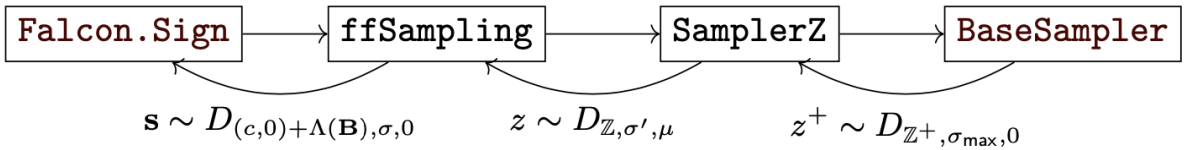


Figure: Signature's flowchart

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

1 $u \leftarrow$ `UniformBits`(72)
2 $z^+ \leftarrow 0$
3 **for** $i \leftarrow 0$ **to** 17 **do**
4 $\quad z^+ \leftarrow z^+ + [\![u < \mathrm{RCDT}[i]]\!]$
5 **end**
6 **return** z^+

- $[[u < RCDT[i]]] := 1_{\{u<RCDT[i]\}}$
- RCDT= table of 18 numbers of 72 bits. $RCDT[i] := 2^{72} - CDT[i]$

SPA on BaseSampler - Weakness

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

1 $u \leftarrow$ `UniformBits`(72)
2 $z^+ \leftarrow 0$
3 **for** $i \leftarrow 0$ **to** 17 **do**
4 $\quad z^+ \leftarrow z^+ + [\![u < \mathrm{RCDT}[i]]\!]$
5 **end**
6 **return** z^+

- Comparison $u < \mathrm{RCDT}[i]$ is a three 24-bits integer subtraction
- Result stored in 32-bits register
- What happens upon *underflows* on the last one?

- Register $= 1 \cdots 1 \,\|\, result$ (8 MSBs set to 1)
- Hamming weight increased by 8
- Retrieve the value of z^+ (especially $z^+ = 0$ or not)

Note: After `BaseSampler`, a sign is generated and $z \in \{0, 1\}$

SPA on BaseSampler - Trace

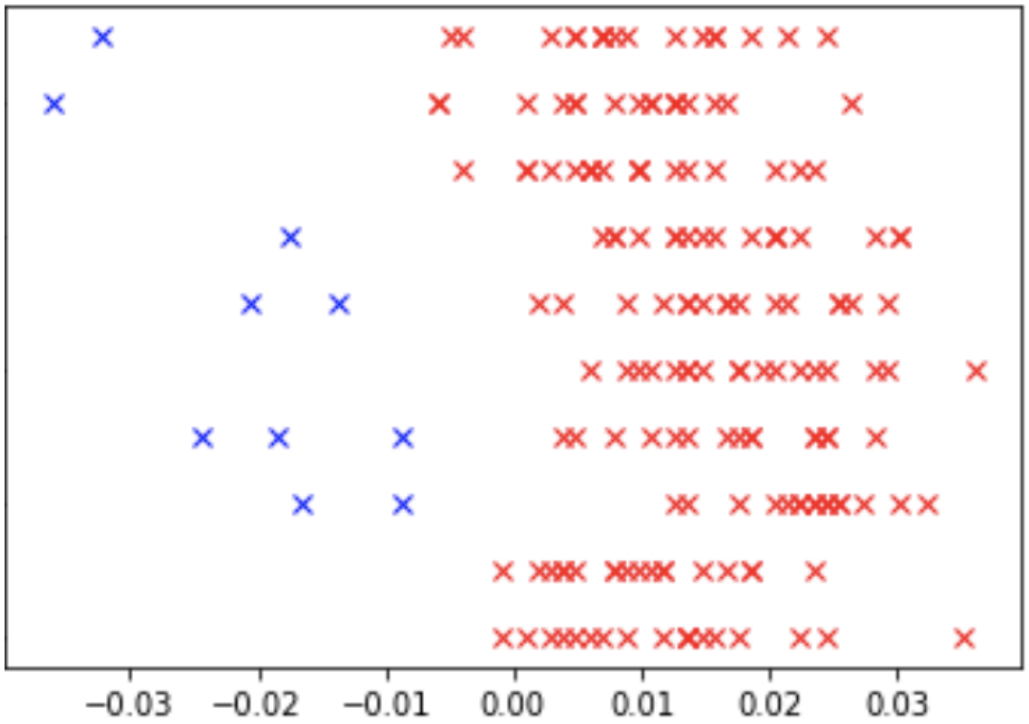


Figure: Power variation during the execution of the substraction. Each line is a different execution. Blue ticks stand for incrementation of z+, red ones for absence of it.

SPA on `BaseSampler` - Results

	Percentages of traces accurately classified
ELMO	100%
ChipWhisperer	94.2%
More efficient setup from [KH18]	100%

Table: Result of SPA attack on ELMO simulator, ChipWhisperer CPU and a more efficient setup from Kim and Hong in [KH18]

ffSampling

Input: target vector $\mathbf{t_n}$ and a private basis $\mathbf{B}$

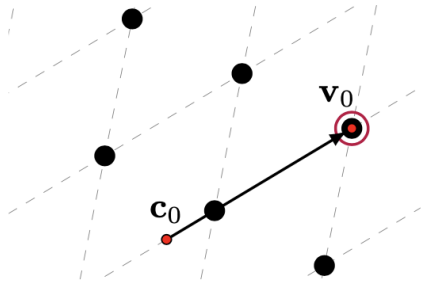
Output: $\mathbf{v} \sim D_{(c,0)+\Lambda(\mathbf{B}),\sigma,0}$

For $i = n-1, \ldots, 0$:

1. $x_i \leftarrow \langle \mathbf{t_i}, \mathbf{b_i} \rangle / \|\mathbf{b_i}\|^2$
2. Pick $z_i \sim D_{\mathbb{Z},\sigma',x_i-\lfloor x_i \rceil}$ with $\sigma' := \sigma / \|\mathbf{b_i}\|$, and let $k_i \leftarrow \lfloor x_i \rceil + z_i$
3. Let $\mathbf{t_i} \leftarrow \mathbf{t_{i+1}} - k_i \mathbf{b_i}$ and $\mathbf{v_i} \leftarrow \mathbf{v_{i+1}} + k_i \mathbf{b_i}$

return $\mathbf{v}_0$

- z_i is sampled by `SamplerZ` who calls `BaseSampler`
- Basic GPV: $\mathbf{v} = \lfloor \mathbf{cB}^{-1} \rceil \mathbf{B}$
- `ffSampling`: $\mathbf{v} = (\lfloor \mathbf{cB}^{-1} \rceil + z_i)\mathbf{B}$



Summary of attack on BaseSampler

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

1 $u \leftarrow \texttt{UniformBits}(72)$
2 $z^+ \leftarrow 0$
3 **for** $i \leftarrow 0$ **to** 17 **do**
4 $\quad z^+ \leftarrow z^+ + [\![u < \mathrm{RCDT}[i]]\!]$
5 **end**
6 **return** z^+

- $z^+ = 0 \rightsquigarrow z_i \in \{0, 1\} \rightsquigarrow y_i \in (-1, 1] \implies s \in \mathcal{P}(\tilde{B})$ in order to apply HPP solver.

Summary of attack on BaseSampler

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

1 $u \leftarrow$ `UniformBits`(72)
2 $z^+ \leftarrow 0$
3 **for** $i \leftarrow 0$ **to** 17 **do**
4 $\quad z^+ \leftarrow z^+ + [\![u < \mathrm{RCDT}[i]]\!]$
5 **end**
6 **return** z^+

- $z^+ = 0 \rightsquigarrow z_i \in \{0, 1\} \rightsquigarrow y_i \in (-1, 1] \implies s \in \mathcal{P}(\tilde{B})$ in order to apply HPP solver.
- $z^+ = 0 \rightsquigarrow z_i \in \{0, 1\}$ since in SamplerZ implementation, $z_i \leftarrow b + (2 \cdot b - 1)z^+$, for $b = 0$ or 1 with equal probability.

Summary of attack on BaseSampler

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

1 $u \leftarrow$ `UniformBits`(72)
2 $z^+ \leftarrow 0$
3 **for** $i \leftarrow 0$ **to** 17 **do**
4 $\quad z^+ \leftarrow z^+ + [\![u < \mathrm{RCDT}[i]]\!]$
5 **end**
6 **return** z^+

- $z^+ = 0 \rightsquigarrow z_i \in \{0,1\} \rightsquigarrow y_i \in (-1,1] \implies s \in \mathcal{P}(\tilde{B})$ in order to apply HPP solver.
- $z^+ = 0 \rightsquigarrow z_i \in \{0,1\}$ since in SamplerZ implementation, $z_i \leftarrow b + (2 \cdot b - 1)z^+$, for $b = 0$ or 1 with equal probability.
- $z_i \in \{0,1\} \rightsquigarrow y_i \in (-1,1]$ since $\mathbf{s} := (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$, $x_i := \langle \mathbf{t}, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$:

$$\mathbf{s} = \sum_{i \in [n]} y_i \cdot \tilde{\mathbf{b}}_i \quad \text{where } y_i = z_i - x_i + \lfloor x_i \rfloor.$$

Summary of attack on BaseSampler

Algorithm 2: `BaseSampler` ()

Output : An integer $z^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

```
u ← UniformBits(72)
z⁺ ← 0
for i ← 0 to 17 do
    z⁺ ← z⁺ + ⟦u < RCDT[i]⟧
end
return z⁺
```

- $z^+ = 0 \rightsquigarrow z_i \in \{0,1\} \rightsquigarrow y_i \in (-1,1] \implies s \in \mathcal{P}(\tilde{B})$ in order to apply HPP solver.
- $z^+ = 0 \rightsquigarrow z_i \in \{0,1\}$ since in SamplerZ implementation, $z_i \leftarrow b + (2 \cdot b - 1)z^+$, for $b = 0$ or 1 with equal probability.
- $z_i \in \{0,1\} \rightsquigarrow y_i \in (-1,1]$ since $\mathbf{s} := (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$, $x_i := \langle \mathbf{t}, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$:

$$\mathbf{s} = \sum_{i \in [n]} y_i \cdot \tilde{\mathbf{b}}_i \quad \text{where } y_i = z_i - x_i + \lfloor x_i \rfloor.$$

- but distrib(y_i) is not uniform over $(-1,1] \rightsquigarrow$ cannot directly apply HPP solver.

HPP

Let $\mathbf{B} := (\mathbf{b_0}, ..., \mathbf{b_{n-1}})$ a basis of n linearly independent vectors and let $\mathcal{P}(\mathbf{B}) = \{\sum_{i=0}^{n-1} x_i \mathbf{b_i},\ x_i \in [-1, 1]\}$, the parallelepiped spanned by $\mathbf{B}$. Given a sequence of poly(n) independent samples drawn uniformly at random in $\mathcal{P}(\mathbf{B})$, find a good approximation of $\mathbf{B}$.

Takeaways:

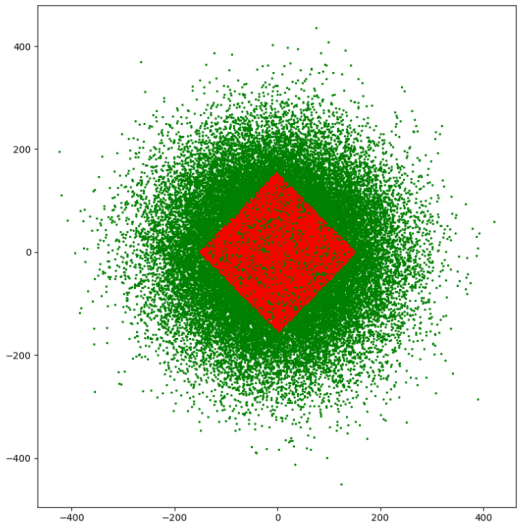
- A HPP solver proposed by Nguyen and Regev in [NR06] breaks NTRUSign and GGH signatures
- Falcon's signature are very similar to these signatures (GPV framework)
- Can we apply directly these solver to Falcon's signature ?

No, because of random perturbation added in their computation ☹

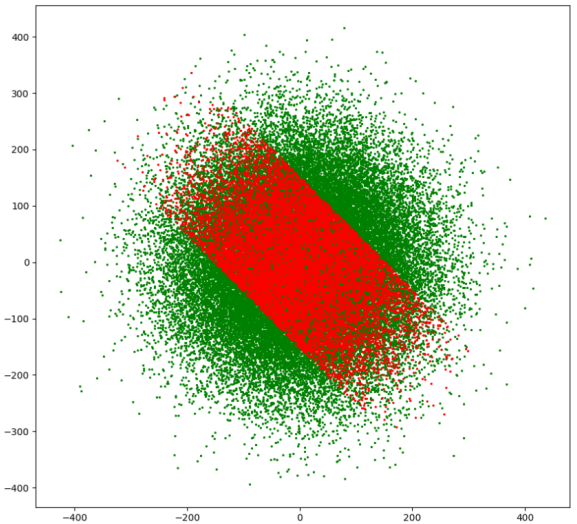
ffSampling & HPP - HPP Deformed variant

- Ducas and Nguyen in [DN12] extended the HPP attack to instance where a small perturbation is added $\delta \in [-1, 1]^n$
- The vector v becomes $(\lfloor \mathbf{cB}^{-1} \rceil + \delta)\mathbf{B}$
- if the perturbation is *partial* (i.e. some set of fixed coordinates indexes have $\delta = 0$), then the HPP solver can recover some $\pm\mathbf{b_i}$
- In our problem, the perturbation δ is z_i

$\Rightarrow$ We can apply the HPP solver on each index i by filtering signatures for which $z_i = 0$



(a) Signatures where $\mathbf{z} \in \{0, 1\}^2$ are in red



(b) Signatures where $z_0 \in \{0, 1\}$ are in red

ffSampling & HPP - Note

- $P[z_i \in \{0,1\}] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$

$\Rightarrow$ 41% to 56% signatures are kept for each index

- In practice, Falcon's signature are based on the Gram-Schmidt Orthogonalization (GSO) $\tilde{\mathbf{B}}$ rather than $\mathbf{B}$.
- HPP solver will return rows of $\tilde{\mathbf{B}}$
- Falcon use `ffLDL` algorithm for GSO which preserves the first three rows: $\tilde{\mathbf{b_0}}, ..., \tilde{\mathbf{b_3}} \approx \mathbf{b_0}, ..., \mathbf{b_3}$

$\Rightarrow$ $\tilde{\mathbf{b_0}} = \mathbf{b_0} = (g'_0, ..., g'_{n-1}, -f'_0, ..., -f'_{n-1}) = (g', -f')$, an approximation of the private key

Recovering the key - Two options

Direct Recovering (mere rounding):

- Lot of signature measurements: $+5$ mio for a high probability (> 0.99) to have an absolute error less than 0.5 on each coefficients
- Easy to compute

Lattice Reduction (Work/Measurement trade-off):

- Less signature measurements
- Solve lattice reduction problem using Leaky LWE / NTRU tool

$\Rightarrow$ More work and computation time

$\Rightarrow$ 1 mio measurements $\approx$ 1000 hours

$\Rightarrow$ With $+1.5$ mio, it becomes reasonable

Proposed countermeasure

Algorithm 3: Proposed countermeasure to mitigate our attack. This pseudocode corresponds to the last comparison during the computation of $[\![u < \mathrm{RCDT}[i]]\!]$ at line 4 of `BaseSampler`

Input : Two 24-bit variables $\overline{u}$ and $\overline{\mathrm{RCDT}[i]}$ stored in 32 bit registers.
a bit c carrying the result of the comparison of both least significant registers (corresponding to the 48 least significant bits of $u - \mathrm{RCDT}[i]$).
Output : 1 if $\overline{\mathrm{RCDT}[i]} + c > \overline{u}$ and 0 is $\overline{u} \leq \overline{\mathrm{RCDT}[i]} + c$

1 $b \leftarrow$ `0xffffff`
2 $b := b - \overline{u} + \overline{\mathrm{RCDT}[i]} + c$
3 **return** $b \gg 24$

- Replace *underflow* by *overflow*
- if $\overline{RCDT[i]} + c > \bar{u}$, it overflows and returns 1 (0 otherwise)

\+ Reduces the Hamming weight by a factor of 8

\+ Simple sufficient mitigation for this attack

– Weak security assurance (provable masked implementations would be better)

Recent work on Side Channel Attacks against Falcon

Thorough Power Analysis on Falcon Gaussian Samplers and Practical Countermeasure

Xiuhan Lin[1], Shiduo Zhang[2], Yang Yu[2,3,4(✉)], Weijia Wang[1,4], Qidi You[5,6], Ximing Xu[7], and Xiaoyun Wang[1,2,3,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
xhlin@mail.sdu.edu.cn
[2] Institute for Advanced Study, Tsinghua University, Beijing, China
{zsd,yu-yang,xiaoyunwang}@mail.tsinghua.edu.cn
[3] Zhongguancun Laboratory, Beijing, China
[4] State Key Laboratory of Cryptography and Digital Economy Security, China
wjwang@sdu.edu.cn
[5] State Key Laboratory of Space-Ground Integrated Information Technology, China
[6] Space star Technology Co., Ltd, China, youqd@spacestar.com.cn
[7] China Mobile Internet, xuximing@chinamobile.com

[Lin et al.]

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 2, pp. 276–303. DOI:10.46586/tches.v2024.i2.276-303

Masking Floating-Point Number Multiplication and Addition of Falcon

First- and Higher-order Implementations and Evaluations

Keng-Yu Chen[1] and Jiun-Peng Chen[1,2]

[1] National Taiwan University, Taipei, Taiwan, r11921066@ntu.edu.tw
[2] Academia Sinica, Taipei, Taiwan, jpchen@ieee.org

Keng-Yu Chen et al. [CC24]

Summary[1/3]

In this talk we have seen...

- **Side-channel security of Falcon**
  - floating-point operations = targets for SCA $\rightsquigarrow$ pre-image attack [KA21], improved by [GMRR22] (reduced number of traces and complexity)
  - GMRR22 provided the 1rst Power analysis using the leakage in BaseSampler and HPP solver.

Summary[1/3]

In this talk we have seen...

- **Side-channel security of Falcon**
  - floating-point operations = targets for SCA $\rightsquigarrow$ pre-image attack [KA21], improved by [GMRR22] (reduced number of traces and complexity)
  - GMRR22 provided the 1rst Power analysis using the leakage in BaseSampler and HPP solver.
- **Two attacks by [GMRR22]:**
  - Partial knowledge of $\hat{f}$ leads to KR.
  - power consumption leakage $\rightsquigarrow$ noting when z^+ is incremented $\rightsquigarrow$ noting when $z_i \in \{0, 1\} \rightsquigarrow$ collecting some $y_i \in (-1, 1] \implies s \in \mathcal{P}(\tilde{B}) \implies$ apply HPP or Deformed HPP solver $\implies$ KR.

Summary [2/3]

- 1rst Attack (on preimage computation): interesting for anyone looking to perform SCA on polynomial multiplication on floating points.
- 2nd Attack (on BaseSampler) worked because: $u < \text{RCDT}[i]$ is a 72 bits substractions $\rightarrow$ hard to perform on a device with only 32-bit registers. But Falcon allows to do three 24 bits substractions $\rightsquigarrow$ SCA works: noting when z^+ is incremented.

Summary[3/3]

-  These new attacks highlight the need for Side-channel protection for one of the 3 finalists of NIST's standardization campaign.
- **Recent works:** [CC24] provided the first masking floating-point multiplication and addition which protects Falcon's pre-image computation against the attack of [KA21]
- **Countermeasures:** effective and easy-to-implement countermeasures against leakages to protect Falcon's integer Gaussian sampler (Lin et al.)

Thank you!

Bibliography [1/3]


Morgane Guerreau, Ange Martinelli, Thomas Ricosset, and Mélissa Rossi. The hidden parallelepiped is back again: Power analysis attacks on Falcon. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(3):141–164, 2022. https://eprint.iacr.org/2022/057.pdf.


Emre Karabulut and Aydin Aysu. Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks. Cryptology ePrint Archive, Report 2021/772, 2021. https://ia.cr/2021/772.


Keng-Yu Chen and Jiun-Peng Chen. Masking Floating-Point Number Multiplication and Addition of Falcon: First- and Higher-Order Implementations and Evaluations. https://tches.iacr.org/index.php/TCHES/article/view/11428

Bibliography[2/3]


Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An Introduction to Mathematical Cryptography*. Volume 1. Springer, 2008. https://link.springer.com/book/10.1007/978-0-387-77993-5


Xiuhan Lin and Shiduo Zhang and Yang Yu and Weijia Wang and Qidi You and Ximing Xu and Xiaoyun Wang. Thorough Power Analysis on Falcon Gaussian Samplers and Practical Countermeasure. https://eprint.iacr.org/2025/351


Pierre-Alain Fouque, Paul Kirchner, Thomas Prest, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON: Fast-Fourier Lattice-Based Compact Signatures over NTRU. In *Proceedings of the NIST Post-Quantum Cryptography Standardization Process*, 2018. https://falcon-sign.info/falcon.pdf

Bibliography[3/3]


Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of LNCS, pages 271-288. Springer, Heidelberg, May / June 2006.


Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, 11th *SODA*, pages 937–941. ACM-SIAM, January 2000.


Suhri Kim and Seokhie Hong. Single trace analysis on constant time cdt sampler and its countermeasure. Applied Sciences, 8(10), 2018.


Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, ASIACRYPT 2012, volume 7658 of LNCS, pages 433–450. Springer, Heidelberg, December 2012.