A Differential Fault Attack against Deterministic FALCON Signatures

Sven Bauer and Fabrizio De Santis

Overview

- FALCON [PFH+22] is a post-quantum signature scheme based on the GPV framework [GPV07], a hash-then-sign construction.
- In standard FALCON, hashing the message is randomized. (We will see in a moment why).
- There are use-cases for a deterministic variant. Such a variant has been specified in [LPa17].
- Our attack is based on injecting faults in the trapdoor sampler. This produces different signatures for the same message hash.
- Such signatures lead to relatively short lattice vectors.
- Then, lattice reduction is used to find the private key.

SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^{\perp}(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^{\perp}(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^{\perp}(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).
2. Find random $v \in \mathcal{L}(B) = \mathcal{L}_q^{\perp}(A)$ s.t. v is `close' to $c$ (`pre-image sampling').

Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023
SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^\perp(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).
2. Find random $v \in \mathcal{L}(B) = \mathcal{L}_q^\perp(A)$ s.t. v is `close' to $c$ (`pre-image sampling').
3. Output $s := c - v$. Note that s is `short' and that $sA^T = cA^T - vA^T = H(M) - 0 = H(M)$.

Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^{\perp}(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).
2. Find random $v \in \mathcal{L}(B) = \mathcal{L}_q^{\perp}(A)$ s.t. v is \`close' to $c$ (\`pre-image sampling').
3. Output $s := c - v$. Note that s is \`short' and that $sA^T = cA^T - vA^T = H(M) - 0 = H(M)$.

Verifying a signature s for M

Check that

1. s is \`short' and
2. $sA^T = H(M)$.

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

The GPV signature framework

Defined by Gentry, Peikert and Vaikuntanathan in [GPV07].

Setup

Lattice $\mathcal{L}_q^{\perp}(A) = \mathcal{L}(B) \subset \mathbb{Z}^n$, B with 'short' rows. A will be the public key, B will be the private key. Hash function H maps messages to points in $\mathbb{Z}_q^n$.

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).
2. Find random $v \in \mathcal{L}(B) = \mathcal{L}_q^{\perp}(A)$ s.t. v is `close' to $c$ (`pre-image sampling').
3. Output $s := c - v$. Note that s is `short' and that $sA^T = cA^T - vA^T = H(M) - 0 = H(M)$.

Verifying a signature s for M

Check that

1. s is `short' and
2. $sA^T = H(M)$.

Security

Finding v close to c is hard without a short basis like B.

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

Sampling twice is problematic

Signing a message M

1. Find $c \in \mathbb{Z}_q^n$ s.t. $cA^T = H(M)$ (linear algebra).
2. Find random $v \in \mathcal{L}(B) = \mathcal{L}_q^{\perp}(A)$ s.t. v is `close' to $c$ (`pre-image sampling').
3. Output $s := c - v$. Note that s is `short' and that $sA^T = cA^T - vA^T = H(M) - 0 = H(M)$.

Suppose we sign the same c twice and sample v, v' in step 2 with $v \neq v'$. Then:

- We obtain two different signatures s, s'.
- An attacker can calculate $s - s'$. Because both s and s' are both short, so is $s - s'$.
- Also, $s - s' \in \mathcal{L}(B)$, because

$$s - s' = (c - v) - (c - v') = v' - v,$$

So an attacker can easily obtain a (relatively) short lattice vector. Repeating this leaks more and more information about the secret basis B. This has already been noted in the original paper about the GPV framework [GPV07].

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

Mitigating the risk of repeated sampling and the idea of our attack

Standard Falcon

Randomize the message hash, i.e. $cA^T = H(r, M)$ with some random r [PFH+22].

Deterministic Falcon

It is deterministic! The sampler returns the same v when the same message is signed again [LPa17].

Idea of the attack against deterministic Falcon

- Sign the same message repeatedly.
- Inject faults in the sampler.
- This gives the attacker different (valid) signatures.
- Differences between pairs of such signatures yield (relatively) short lattice vectors.
- Reduce these to find a short basis.

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

NTRU lattices [HPS98, PFH+22]

Setup

q prime, $n = 2^k$ and $\phi(X) = X^n + 1$.

Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

NTRU lattices [HPS98, PFH$^+$22]

Setup

q prime, $n = 2^k$ and $\phi(X) = X^n + 1$.

NTRU private key

$f, g, F, G \in \mathbb{Z}[X]/(\phi)$ with 'small' coefficients such that

$$fG - gF \equiv q \bmod \phi$$

and such that $f^{-1} \mod \phi, q$ exists.

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

NTRU lattices [HPS98, PFH^{+}22]

Setup

q prime, $n = 2^k$ and $\phi(X) = X^n + 1$.

NTRU private key

$f, g, F, G \in \mathbb{Z}[X]/(\phi)$ with 'small' coefficients such that

$$fG - gF \equiv q \bmod \phi$$

and such that $f^{-1} \mod \phi, q$ exists.

NTRU public key

$h = gf^{-1} \bmod \phi, q$, coefficients of h are typically 'large'.

SIEMENS

NTRU – from polynomials to integer lattices

Identify polynomials $a(X) = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]/(\phi)$ with their coefficient vector $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$.

Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

NTRU – from polynomials to integer lattices

Identify polynomials $a(X) = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]/(\phi)$ with their coefficient vector $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$.
Polynomial multiplication $a(X)b(X)$ is linear, so we can identify b with a matrix $B \in \mathbb{Z}^{n \times n}$:

$$B = \begin{pmatrix} b_0 & b_1 & \ldots & b_{n-1} \\ -b_{n-1} & b_0 & \ldots & b_{n-2} \\ -b_{n-2} & -b_{n-1} & \ldots & b_{n-3} \\ \vdots & \vdots & \ddots & \vdots \\ -b_1 & -b_2 & \ldots & b_0 \end{pmatrix}$$

(Homework: Check that $a \cdot B \leftrightarrow a(X)b(X)$.)

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

NTRU – from polynomials to integer lattices

Identify polynomials $a(X) = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]/(\phi)$ with their coefficient vector $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$.
Polynomial multiplication $a(X)b(X)$ is linear, so we can identify b with a matrix $B \in \mathbb{Z}^{n \times n}$:

$$B = \begin{pmatrix} b_0 & b_1 & \ldots & b_{n-1} \\ -b_{n-1} & b_0 & \ldots & b_{n-2} \\ -b_{n-2} & -b_{n-1} & \ldots & b_{n-3} \\ \vdots & \vdots & \ddots & \vdots \\ -b_1 & -b_2 & \ldots & b_0 \end{pmatrix}$$

(Homework: Check that $a \cdot B \leftrightarrow a(X)b(X)$.)

With this identification, the NTRU-lattice is:

$$\mathcal{L}\begin{pmatrix} g & -f \\ G & -F \end{pmatrix} = \mathcal{L}_q^{\perp}\begin{pmatrix} 1 & h^T \end{pmatrix}$$

(Homework: Check this equality.)

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

Signing with deterministic FALCON (simplified)

Require: A message M, a secret key $\mathtt{sk}$ consisting of $B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ and some pre-computed data structure T.
Ensure: A valid $\textsc{Falcon}$ signature s of M.

1: **procedure** $\textsc{Falcon-Sign}(\mathrm{M}, \mathtt{sk})$
2: $c \leftarrow \mathtt{HashToPoint}(\mathtt{salt}, M)$
3: $(t_0, t_1) \leftarrow (c, 0)\mathbf{B}^{-1}$
4: **do**
5: $(z_0, z_1) \leftarrow \mathtt{ffSampling}_n((t_0, t_1), T)$ ▷ Note: the sampling is deterministic!
6: $(s_1, s_2) \leftarrow ((t_0, t_1) - (z_0, z_1))\mathbf{B}$
7: **while** $\|(s_1, s_2)\|^2 > \lfloor \beta^2 \rfloor$
8: **return** (s_1, s_2)

Signing with deterministic FALCON (simplified)

Require: A message M, a secret key sk consisting of $B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ and some pre-computed data structure T.
Ensure: A valid FALCON signature s of M.

1: **procedure** FALCON-SIGN(M,sk)
2: $c \leftarrow \texttt{HashToPoint}(\texttt{salt}, M)$
3: $(t_0, t_1) \leftarrow (c, 0)\mathbf{B}^{-1}$
4: **do**
5: $(z_0, z_1) \leftarrow \boxed{\texttt{ffSampling}_n((t_0, t_1), T)}$ ▷ Note: the sampling is deterministic!
6: $(s_1, s_2) \leftarrow ((t_0, t_1) - (z_0, z_1))\mathbf{B}$
7: **while** $\|(s_1, s_2)\|^2 > \lfloor \beta^2 \rfloor$
8: **return** (s_1, s_2)

inject fault in this subroutine

SIEMENS

Signing with deterministic Falcon (simplified)

Require: A message M, a secret key sk consisting of $B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ and some pre-computed data structure T.
Ensure: A valid Falcon signature s of M.

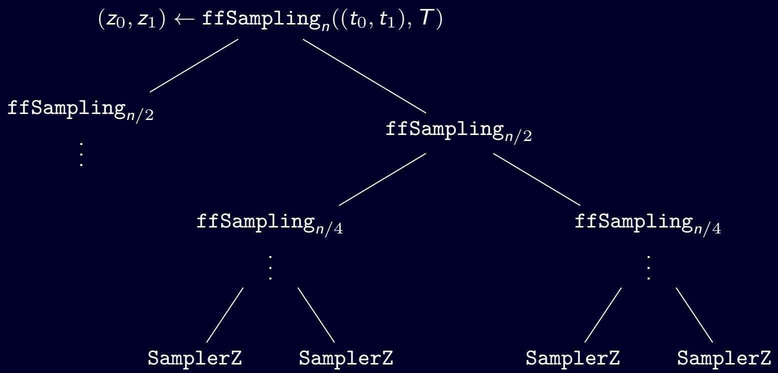
procedure Falcon-Sign(M,sk)
  $c \leftarrow \texttt{HashToPoint}(\texttt{salt}, M)$
  $(t_0, t_1) \leftarrow (c, 0)\mathbf{B}^{-1}$
  do
    $(z_0, z_1) \leftarrow \boxed{\texttt{ffSampling}_n((t_0, t_1), T)}$ ▷ Note: the sampling is deterministic!
    $(s_1, s_2) \leftarrow ((t_0, t_1) - (z_0, z_1))\mathbf{B}$
  while $\|(s_1, s_2)\|^2 > \lfloor \beta^2 \rfloor$
  return (s_1, s_2)

inject fault in this subroutine

Inject fault such that only z_0 is affected! (Details in the paper)

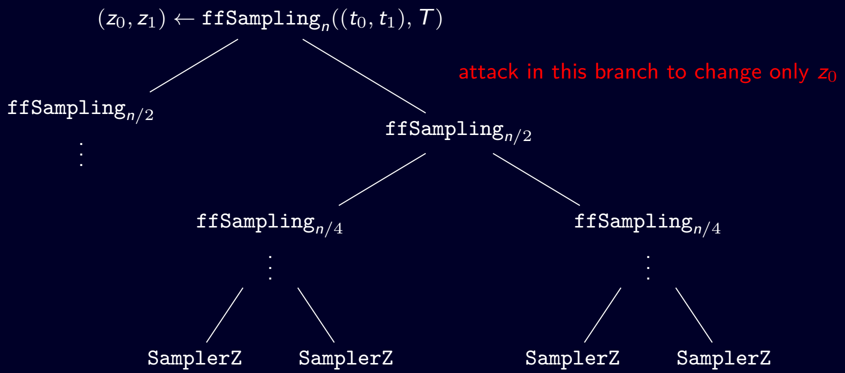
SIEMENS

Sampling in FALCON is defined via recursion (simplified)



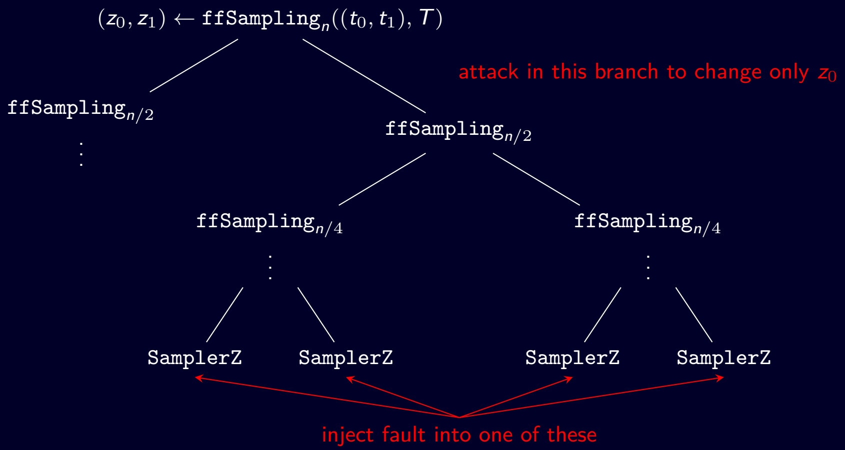
SIEMENS

Sampling in FALCON is defined via recursion (simplified)



 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

Sampling in FALCON is defined via recursion (simplified)



Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

Attack step one: Fault injection

Recall from previous slide that for a signature (s_1, s_2):

$$(s_1, s_2) = ((t_0, t_1) - (z_0, z_1))\begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$$

Hence

$$s_2 = (z_0 - t_0)f + (z_1 - t_1)F$$

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

Attack step one: Fault injection

Recall from previous slide that for a signature (s_1, s_2):

$$(s_1, s_2) = ((t_0, t_1) - (z_0, z_1))\begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$$

Hence

$$s_2 = (z_0 - t_0)f + (z_1 - t_1)F$$

Re-sign with a fault in z_0 to obtain (s_1', s_2'). Then

$$s_2 - s_2' = (z_0 - z_0')f$$

SIEMENS

Attack step one: Fault injection

Recall from previous slide that for a signature (s_1, s_2):

$$(s_1, s_2) = ((t_0, t_1) - (z_0, z_1))\begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$$

Hence

$$s_2 = (z_0 - t_0)f + (z_1 - t_1)F$$

Re-sign with a fault in z_0 to obtain (s_1', s_2'). Then

$$s_2 - s_2' = (z_0 - z_0')f$$

Doing this repeatedly, the attacker obtains a set:

$$\Delta = \{\delta_1 f, \delta_2 f, \ldots, \delta_m f\}$$

SIEMENS

Attack step two: Lattice reduction

The attacker now has a set $\Delta = \{\delta_1 f, \delta_2 f, \ldots, \delta_m f\}$.
Consider the lattice $\Lambda(\Delta)$ generated by Δ, and note

$$\Lambda(\Delta) \subset \Lambda(f)$$

Now:

- If Δ is large enough, then perhaps $f \in \Lambda(\Delta)$.
- If the rank of $\Lambda(\Delta)$ is small enough, the attacker may find f with lattice reduction.
- The other private key components can be calculated from f and the public key.

Making the lattice reduction feasible

The size of the attacker's lattice

- If the attacker's lattice $\Lambda(\Delta)$ is large, there is a better chance it contains the secret key component f.
- If it is smaller, lattice reduction is easier.
- Injecting a fault in one of the final five PRNG calls apparently works.

Two types of faults

Note: A fault can either affect the output of only one call to the PRNG or its internal state and hence also all future calls to the PRNG.

SIEMENS

Properties of the attack

Liberal fault model

Any fault that changes something in the PRNG works (instruction skip, data fault).

Combination with exhaustive search

Recall that the attacker works in $\Lambda(\Delta) \subset \Lambda(f)$.

So lattice reduction with Δ may not yield f but cf, with some polynomial c.

If $c(X) = X^k$, then cf is just a rotation of f. In this case, the attacker obtains an equivalent private key.

If c is sparse, the attacker can try to find c (and hence f) by exhaustive search.

SIEMENS

Code example on a Cortex-M4

Code from [LPa17], file `rng.c`, function `falcon_inner_prng_refill()`.

C code:

```
uint64_t cc;
uint32_t state[16];
...
state[14] ^= (uint32_t)cc;
```

Assembly code:

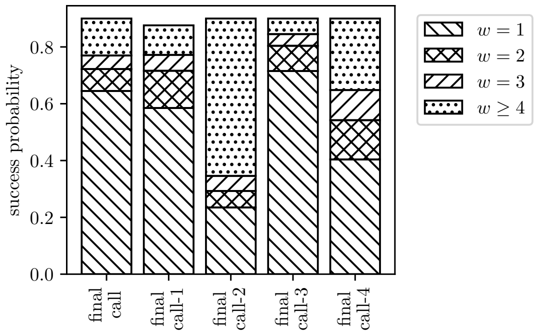
```
ldr     r2, [sp, #32]
ldr     r3, [sp, #136]
...
eors    r3, r2
```

Randomly changing r2 before eors is a suitable fault for the attack.
(Tested with gdb on a Cortex-M4 target. Successful key recovery with 100 faulty signatures. More details in the paper.)

SIEMENS

Simulation results

Attack with 50 faults against FALCON-512 simulated on a PC, the faults affected the PRNG seed, i.e. had a persistent effect. w is the number of non-zero coefficients of c, where cf is the polynomial recovered by lattice reduction.



More pictures in the paper.

SIEMENS

Countermeasures

- Note: Faulty signatures are valid. So, verification does not work as a countermeasure.
- Re-calculation of signatures or at least ffSampling is expensive and works only if the attacker cannot inject the same fault twice.
- Calculate a checksum over the PRNG output, then re-run the PRNG, re-calculate the checksum and compare.

Summary and outlook

Summary

- We have seen a fault attack against deterministic FALCON signature generation.
- The attack works under a very liberal fault model (any fault in the PRNG used for sampling will do).
- It has a high success rate.
- The attack can be combined with an exhaustive search step to reduce the number of faults required or increase the success rate.

Future work

- Transfer results to other signatures based on GPV framework (Mitaka, ModFalcon)
- Investigate suppression of entropy in standard FALCON to make the attack applicable in the non-deterministic setting.

 Unrestricted | © Siemens 2023 | Sven Bauer, Fabrizio De Santis | T CST SES-DE | 15 November 2023

SIEMENS

Contact

Sven Bauer, Fabrizio De Santis
Siemens AG Technology

Otto-Hahn-Ring 6
81739 München
Germany

E-mail svenbauer[at]siemens.com, fabrizio.desantis[at]siemens.com

SIEMENS

References

[GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, Cryptology ePrint Archive, Report 2007/432, 2007, `https://eprint.iacr.org/2007/432`.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, *NTRU: A ring-based public key cryptosystem*, Third Algorithmic Number Theory Symposium (ANTS), LNCS, vol. 1423, Springer, Heidelberg, June 1998, pp. 267–288.

[LPa17] David Lazar, Chris Peikert, and algoidan, *Deterministic falcon implementation*, `https://github.com/algorand/falcon`, Accessed 2022-11-17.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang, *FALCON*, Tech. report, National Institute of Standards and Technology, 2022, available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

SIEMENS