The Hidden Parallelepiped is Back Again: Power Analysis Attacks on Falcon

Morgane Guerreau[1] Ange Martinelli[2] Thomas Ricosset[1]
Mélissa Rossi[2]

[1]Thales, [2]ANSSI

September 20, 2022

Introduction

Fast-Fourier Lattice-based
Compact Signatures over NTRU

Introduction

Two Power Analysis attacks on Falcon:

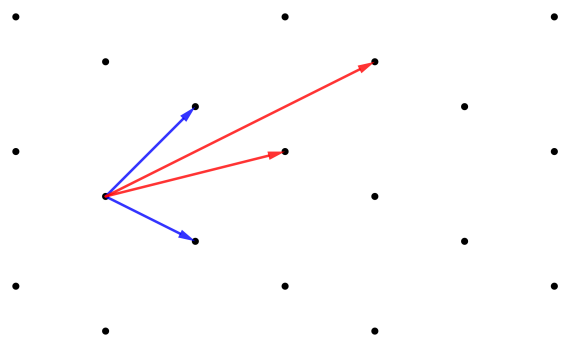
- Efficient DPA attack on the preimage computation
- STA on the trapdoor sampler leading to HPP attack

Lattice-based cryptography

Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.

Lattice-based cryptography

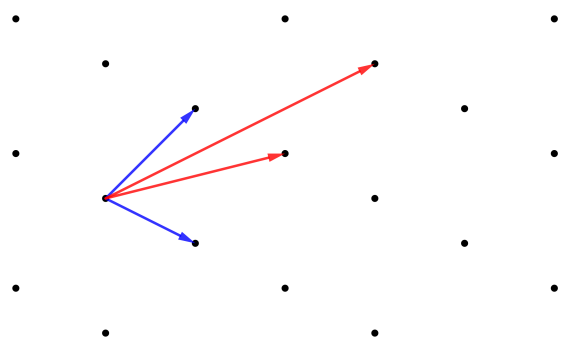
Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.



CVP is easy to solve with a good basis, but hard with a bad basis.

Lattice-based cryptography

Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.



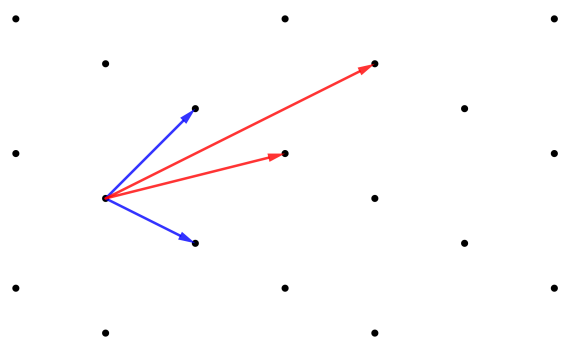
CVP is easy to solve with a good basis, but hard with a bad basis.

Basic signature scheme:

- Convert the message to sign to a vector $\mathbf{c}$ in $\mathcal{R}^n$

Lattice-based cryptography

Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.



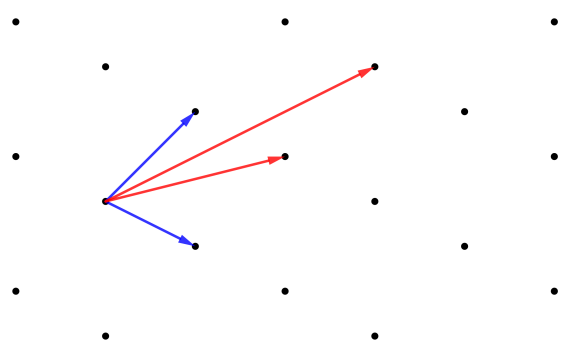
CVP is easy to solve with a good basis, but hard with a bad basis.

Basic signature scheme:

- Convert the message to sign to a vector $\mathbf{c}$ in $\mathcal{R}^n$
- Use the good basis (secret key) to solve CVP

Lattice-based cryptography

Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.



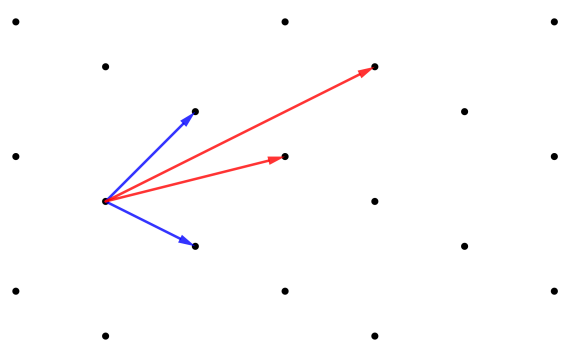
CVP is easy to solve with a good basis, but hard with a bad basis.

Basic signature scheme:

- Convert the message to sign to a vector $\mathbf{c}$ 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)

Lattice-based cryptography

Closest Vector Problem: Given a lattice Λ and a vector $\mathbf{c}$ in $\mathcal{R}^n$, find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v} - \mathbf{c}\|$ is minimal.



CVP is easy to solve with a good basis, but hard with a bad basis.

Basic signature scheme:

- Convert the message to sign to a vector $\mathbf{c}$ 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)

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

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis

$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.

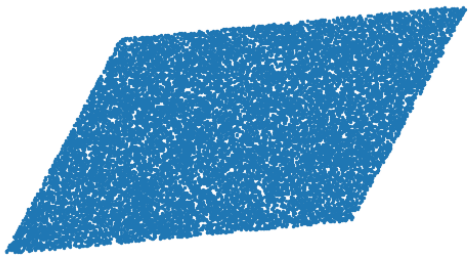
Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis

$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.

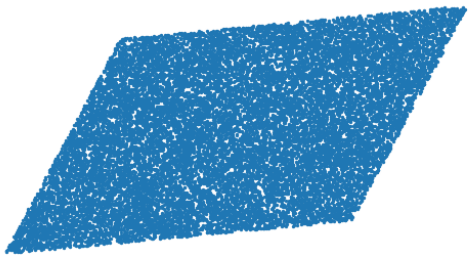


Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis
$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.



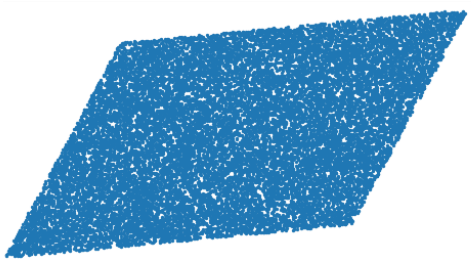
Hidden Parallelepiped Problem: Recover $\mathbf{B}$ from independent samples drawn uniformly in $\mathcal{P}(\mathbf{B})$ [NR06].

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis
$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.



Hidden Parallelepiped Problem: Recover $\mathbf{B}$ from independent samples drawn uniformly in $\mathcal{P}(\mathbf{B})$ [NR06].

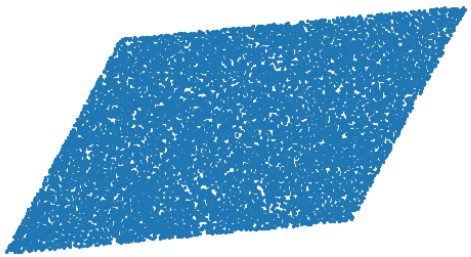
Deformed Parallelepiped Problem: Same as HPP when a *partial* perturbation is applied during round-off algorithm [DN12].

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis
$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.



Hidden Parallelepiped Problem: Recover $\mathbf{B}$ from independent samples drawn uniformly in $\mathcal{P}(\mathbf{B})$ [NR06].

Deformed Parallelepiped Problem: Same as HPP when a *partial* perturbation is applied during round-off algorithm [DN12].

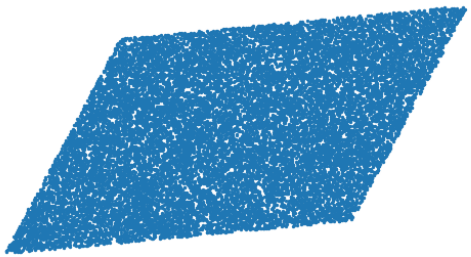
GPV Framework: Replace the round-off algorithm by a trapdoor sampler: $\mathbf{s}$ is not always the closest vector [GPV08].

Hidden Parallelepiped Problem

NTRUSign/GGH:

- Message m is hashed to a point $\mathbf{c} := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

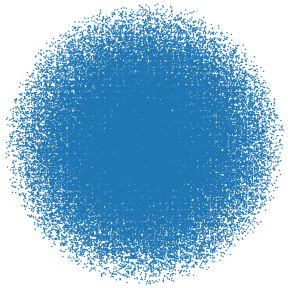
Signature: $\mathbf{s} = \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ with $\mathbf{B}$ the private basis
$\Longrightarrow$ $\mathbf{s} - \mathbf{c}$ belongs to the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.



Hidden Parallelepiped Problem: Recover $\mathbf{B}$ from independent samples drawn uniformly in $\mathcal{P}(\mathbf{B})$ [NR06].

Deformed Parallelepiped Problem: Same as HPP when a *partial* perturbation is applied during round-off algorithm [DN12].

GPV Framework: Replace the round-off algorithm by a trapdoor sampler: $\mathbf{s}$ is not always the closest vector [GPV08].



Falcon

- Instantiation of GPV framework with NTRU lattices

Falcon

- Instantiation of GPV framework with NTRU lattices
- Let $\mathcal{R} := \mathbb{Z}[x]/(x^n + 1)$.

Private key: $f, g, F, G \in \mathcal{R}$ with $fG - gF = q \mod x^n + 1$

Falcon

- Instantiation of GPV framework with NTRU lattices
- Let $\mathcal{R} := \mathbb{Z}[x]/(x^n + 1)$.

Private key: $f, g, F, G \in \mathcal{R}$ with $fG - gF = q \mod x^n + 1$

Private basis: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \Longrightarrow \mathbf{b_0} = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1})$

Falcon

- Instantiation of GPV framework with NTRU lattices
- Let $\mathcal{R} := \mathbb{Z}[x]/(x^n + 1)$.

Private key: $f, g, F, G \in \mathcal{R}$ with $fG - gF = q \mod x^n + 1$

Private basis: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \Longrightarrow \mathbf{b_0} = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1})$

Sign(m, B):

1. $r \leftarrow$ random salt
2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$
6. return $(r, \mathbf{s})$

- preimage computation
- trapdoor sampler

Falcon

- Instantiation of GPV framework with NTRU lattices
- Let $\mathcal{R} := \mathbb{Z}[x]/(x^n + 1)$.

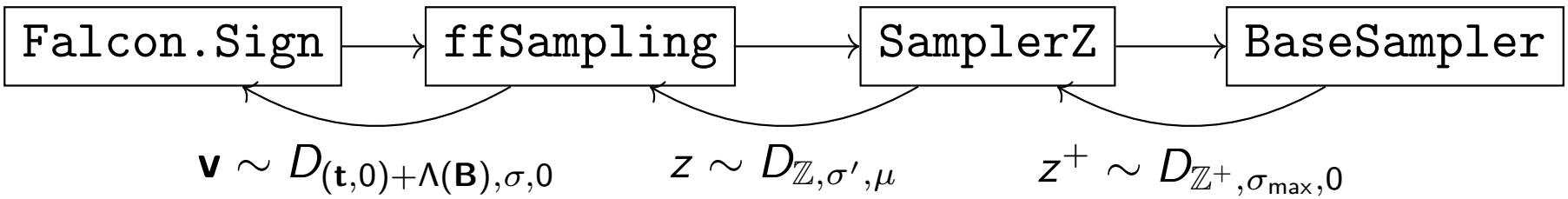
Private key: $f, g, F, G \in \mathcal{R}$ with $fG - gF = q \mod x^n + 1$

Private basis: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \Longrightarrow \mathbf{b_0} = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1})$

Sign(m, B):

1. $r \leftarrow$ random salt
2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$
6. return $(r, \mathbf{s})$

- preimage computation
- trapdoor sampler



Power Analysis on the preimage computation

Sign(m, B):

2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$

- preimage computation
- trapdoor sampler

Differential Power Analysis on the preimage attack

Original attack: DPA on a polynomial multiplication in FFT between a public digest $\mathbf{c}$ and a private polynomial $\mathbf{f}$ [KA21].

Differential Power Analysis on the preimage attack

Original attack: DPA on a polynomial multiplication in FFT between a public digest $\mathbf{c}$ and a private polynomial $\mathbf{f}$ [KA21].

Three improvements:

- Lowering the complexity of exhaustive search: double precision is unnecessary to recover the key.

Differential Power Analysis on the preimage attack

Original attack: DPA on a polynomial multiplication in FFT between a public digest $\mathbf{c}$ and a private polynomial $\mathbf{f}$ [KA21].

Three improvements:

- Lowering the complexity of exhaustive search:
double precision is unnecessary to recover the key.
- Halving the number of required traces by combinating patterns:
complex multiplications involve a lot of operations.

Differential Power Analysis on the preimage attack

Original attack: DPA on a polynomial multiplication in FFT between a public digest $\mathbf{c}$ and a private polynomial $\mathbf{f}$ [KA21].

Three improvements:

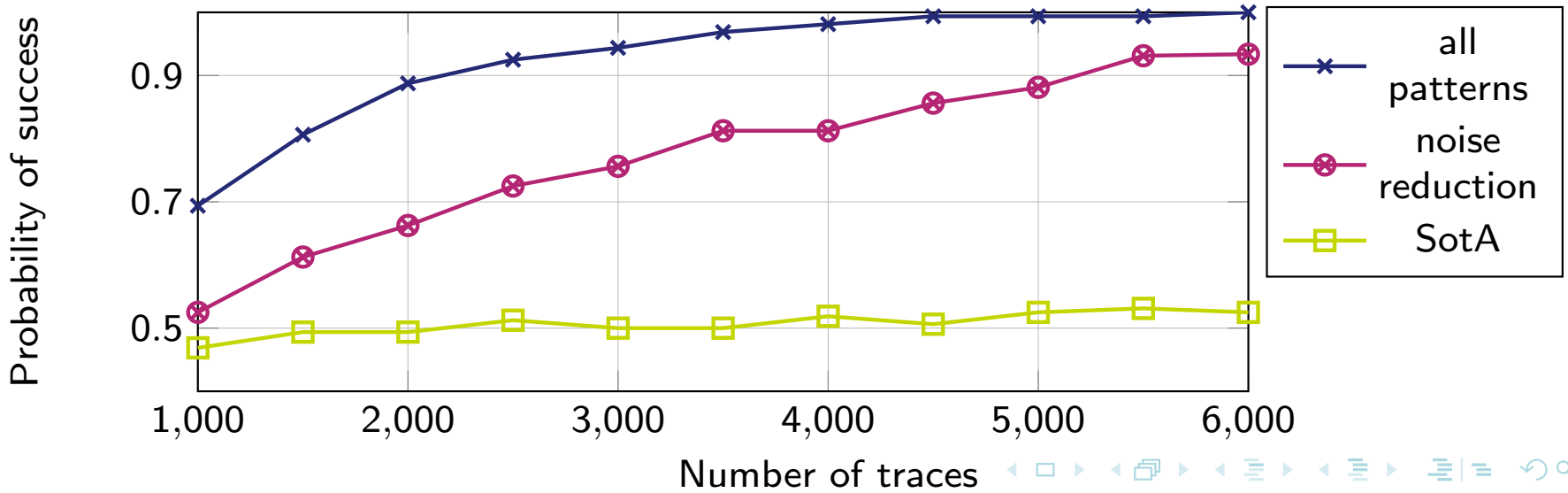
- Lowering the complexity of exhaustive search:
double precision is unnecessary to recover the key.
- Halving the number of required traces by combinating patterns:
complex multiplications involve a lot of operations.
- Mitigating the noise by grouping similar challenges:
we average power traces if challenges are the same (less precision).

Differential Power Analysis on the preimage attack

Original attack: DPA on a polynomial multiplication in FFT between a public digest $\mathbf{c}$ and a private polynomial $\mathbf{f}$ [KA21].

Three improvements:

- Lowering the complexity of exhaustive search:
double precision is unnecessary to recover the key.
- Halving the number of required traces by combinating patterns:
complex multiplications involve a lot of operations.
- Mitigating the noise by grouping similar challenges:
we average power traces if challenges are the same (less precision).



Hidden Parallelepiped attack on the trapdoor sampler

Sign(m, B):

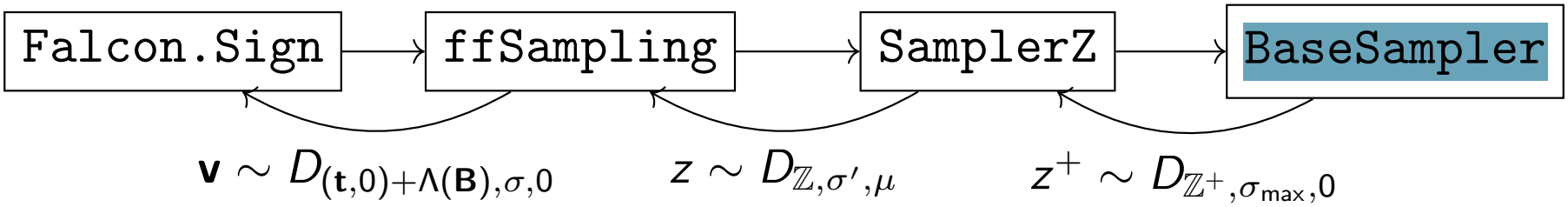
2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$

- preimage computation
- trapdoor sampler

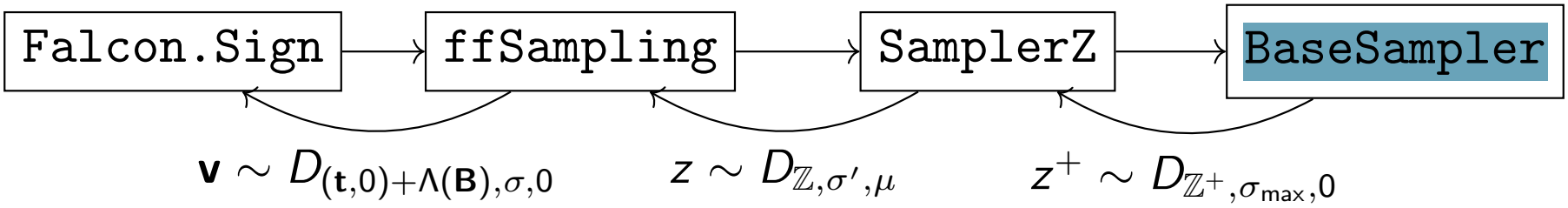
Hidden Parallelepiped attack on the trapdoor sampler

1. Side-channel analysis on the `BaseSampler` to recover samples
2. Utilisation of the samples to disclose a *deformed* parallelepiped
3. Application of HPP solver on filtered signatures
4. Private key recovering (possibly with lattice magic)

1. Side-channel analysis on the BaseSampler



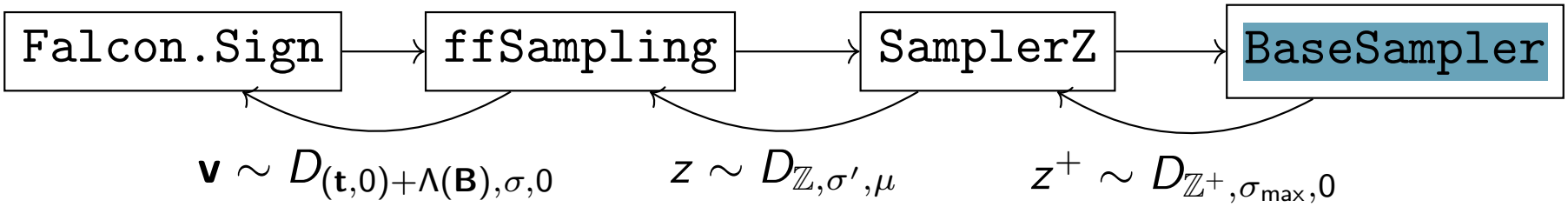
1. Side-channel analysis on the BaseSampler



BaseSampler():

1. $u \leftarrow \texttt{UniformBits}(72)$
2. $z^+ \leftarrow 0$
3. **for** $i = 0 \ldots 16$ **do**
4. $\quad z^+ \leftarrow z^+ + [\![u < \text{RCDT}[i]]\!]$
5. **end**
6. **return** z^+

1. Side-channel analysis on the BaseSampler

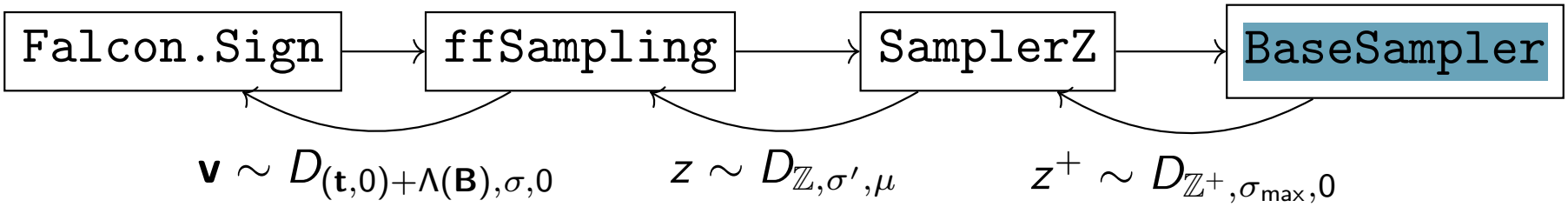


BaseSampler():

1. $u \leftarrow \texttt{UniformBits}(72)$
2. $z^+ \leftarrow 0$
3. **for** $i = 0 \ldots 16$ **do**
4. $\quad z^+ \leftarrow z^+ + [\![u < \text{RCDT}[i]]\!]$
5. **end**
6. **return** z^+

Comparison on line 4 is in fact three successive substractions of 24 bits values, exploiting register underflow.

1. Side-channel analysis on the BaseSampler



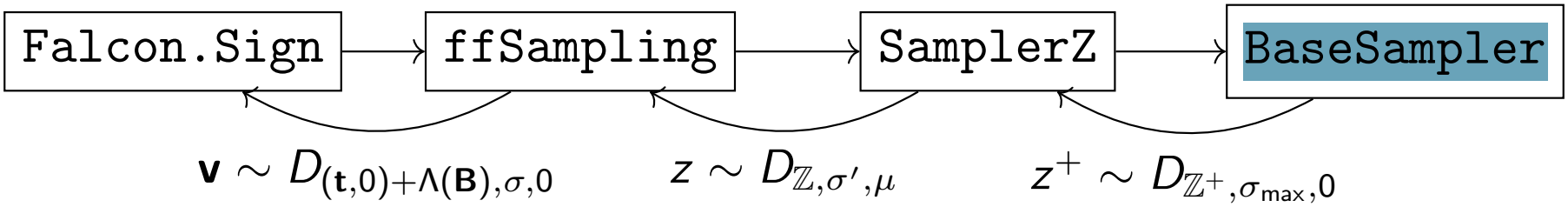
BaseSampler():

1. $u \leftarrow \texttt{UniformBits}(72)$
2. $z^+ \leftarrow 0$
3. **for** $i = 0 \ldots 16$ **do**
4. $\quad z^+ \leftarrow z^+ + [\![u < \text{RCDT}[i]]\!]$
5. **end**
6. **return** z^+

Comparison on line 4 is in fact three successive substractions of 24 bits values, exploiting register underflow.

$\Longrightarrow$ High difference in Hamming weight [KH18]

1. Side-channel analysis on the BaseSampler



BaseSampler():

1. $u \leftarrow \texttt{UniformBits}(72)$
2. $z^+ \leftarrow 0$
3. **for** $i = 0 \ldots 16$ **do**
4. $\quad z^+ \leftarrow z^+ + [\![u < \text{RCDT}[i]]\!]$
5. **end**
6. **return** z^+

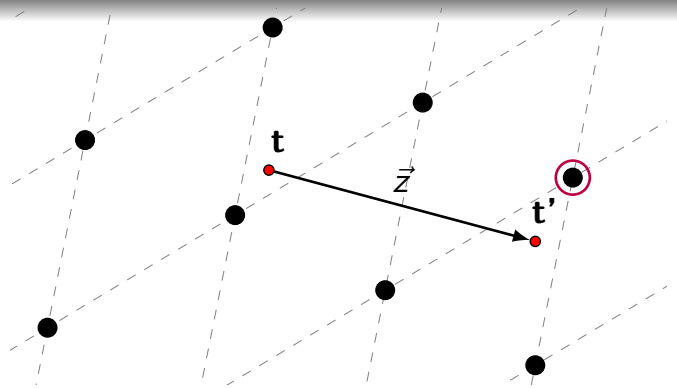
Comparison on line 4 is in fact three successive substractions of 24 bits values, exploiting register underflow.

$\Longrightarrow$ High difference in Hamming weight [KH18]

We are able to retrieve the value of z^+ through STA.

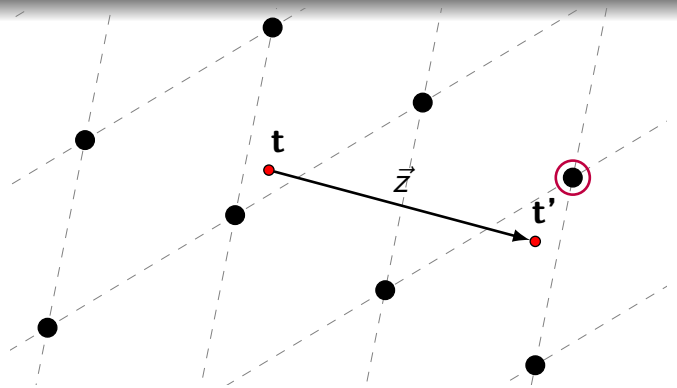
2. Disclosure of the HPP

"Shifted" Babai's nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $\mathbf{t}$ before solving CVP.



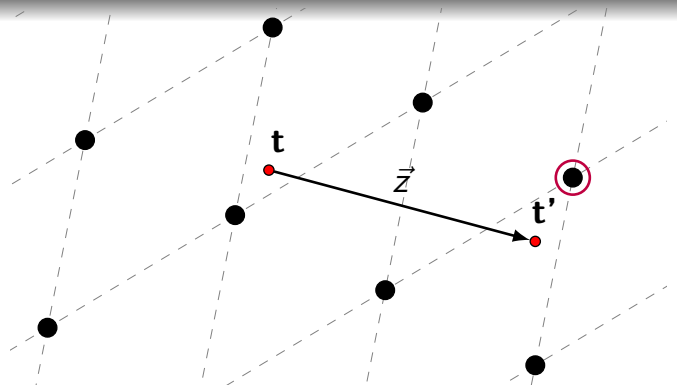
2. Disclosure of the HPP

"Shifted" Babai's nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $\mathbf{t}$ before solving CVP.

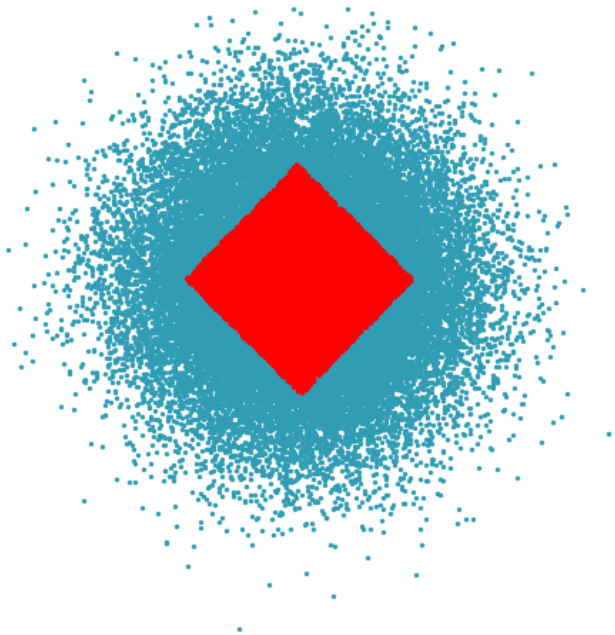


What happens when $\vec{z} := (z_0^+, \ldots, z_{n-1}^+) = (0, \ldots, 0)$?

"Shifted" Babai's nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $\mathbf{t}$ before solving CVP.

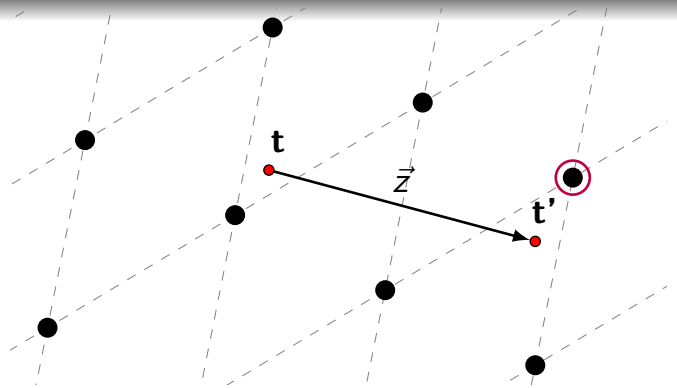


What happens when $\vec{z} := (z_0^+, \ldots, z_{n-1}^+) = (0, \ldots, 0)$?

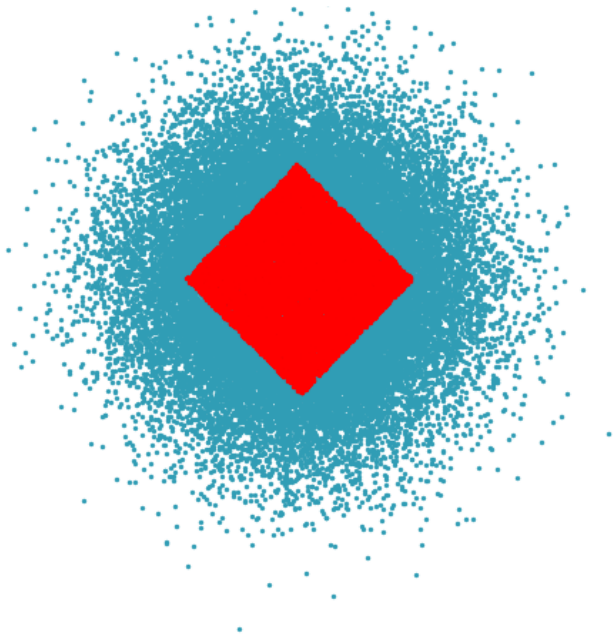


Filtering with all $z_i^+ = 0$

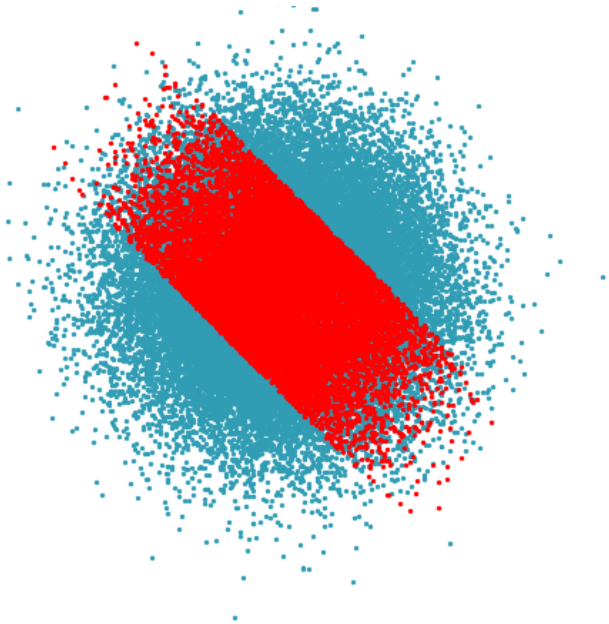
"Shifted" Babai's nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $\mathbf{t}$ before solving CVP.



What happens when $\vec{z} := (z_0^+, \ldots, z_{n-1}^+) = (0, \ldots, 0)$?



Filtering with all $z_i^+ = 0$



Filtering with only $z_0^+ = 0$

3. Application of HPP solver

$\mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

3. Application of HPP solver

$\mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

$\Longrightarrow$ we apply HPP solver from [DN12] on 40 to 55% of the signatures

3. Application of HPP solver

$\mathbb{P}[z_i^+ = 0] \approx \mathrm{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

$\Longrightarrow$ we apply HPP solver from [DN12] on 40 to 55% of the signatures

Baibai's nearest-plane algorithm: uses $\tilde{\mathbf{B}} := \mathrm{GSO}(\mathbf{B})$ instead of $\mathbf{B}$ to solve CVP.

3. Application of HPP solver

$\mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

$\Longrightarrow$ we apply HPP solver from [DN12] on 40 to 55% of the signatures

Baibai's nearest-plane algorithm: uses $\tilde{\mathbf{B}} := \text{GSO}(\mathbf{B})$ instead of $\mathbf{B}$ to solve CVP.

$\Longrightarrow$ signatures belong to $\mathcal{P}(\tilde{\mathbf{B}})$, not $\mathcal{P}(\mathbf{B})$.

3. Application of HPP solver

$\mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

$\Longrightarrow$ we apply HPP solver from [DN12] on 40 to 55% of the signatures

Baibai's nearest-plane algorithm: uses $\tilde{\mathbf{B}} := \text{GSO}(\mathbf{B})$ instead of $\mathbf{B}$ to solve CVP.

$\Longrightarrow$ signatures belong to $\mathcal{P}(\tilde{\mathbf{B}})$, not $\mathcal{P}(\mathbf{B})$.
$\Longrightarrow$ we can only retrieve rows of $\tilde{\mathbf{B}}$ with HPP solver.

$\mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613]$ for all $i \in [0, n-1]$
(because of rejection sampling, not all z_i^+ are kept)

$\Longrightarrow$ we apply HPP solver from [DN12] on 40 to 55% of the signatures

Baibai's nearest-plane algorithm: uses $\tilde{\mathbf{B}} := \text{GSO}(\mathbf{B})$ instead of $\mathbf{B}$ to solve CVP.

$\Longrightarrow$ signatures belong to $\mathcal{P}(\tilde{\mathbf{B}})$, not $\mathcal{P}(\mathbf{B})$.
$\Longrightarrow$ we can only retrieve rows of $\tilde{\mathbf{B}}$ with HPP solver.

Useful observation: Because of the algorithm used in Falcon to compute the GSO (`ffLDL` algorithm), we have the following:

$\tilde{\mathbf{b}}_\mathbf{0}, \ldots, \tilde{\mathbf{b}}_\mathbf{3} \approx \mathbf{b}_0, \ldots, \mathbf{b}_3$ and $\tilde{\mathbf{b}}_\mathbf{n}, \ldots, \tilde{\mathbf{b}}_\mathbf{n+3} \approx \mathbf{b}_n, \ldots, \mathbf{b}_{n+3}$

4. Recovering the private key: Falcon-512

- We combine several rows $\mathbf{b_i}$ to attenuate the noise on f, g.

Note: ulterior iterations of HPP solver are much less costly.

4. Recovering the private key: Falcon-512

- We combine several rows $\mathbf{b_i}$ to attenuate the noise on f, g.

Note: ulterior iterations of HPP solver are much less costly.

Then, two possible ways to recover the exact private key (f, g):

- Mere rounding when $\sigma_{(f',g')-(f,g)}$ is small enough

4. Recovering the private key: Falcon-512

- We combine several rows $\mathbf{b_i}$ to attenuate the noise on f, g.

Note: ulterior iterations of HPP solver are much less costly.

Then, two possible ways to recover the exact private key (f, g):

- Mere rounding when $\sigma_{(f',g')-(f,g)}$ is small enough
- Solve DBDD instance with Leaky LWE/NTRU tool [Dac+20]

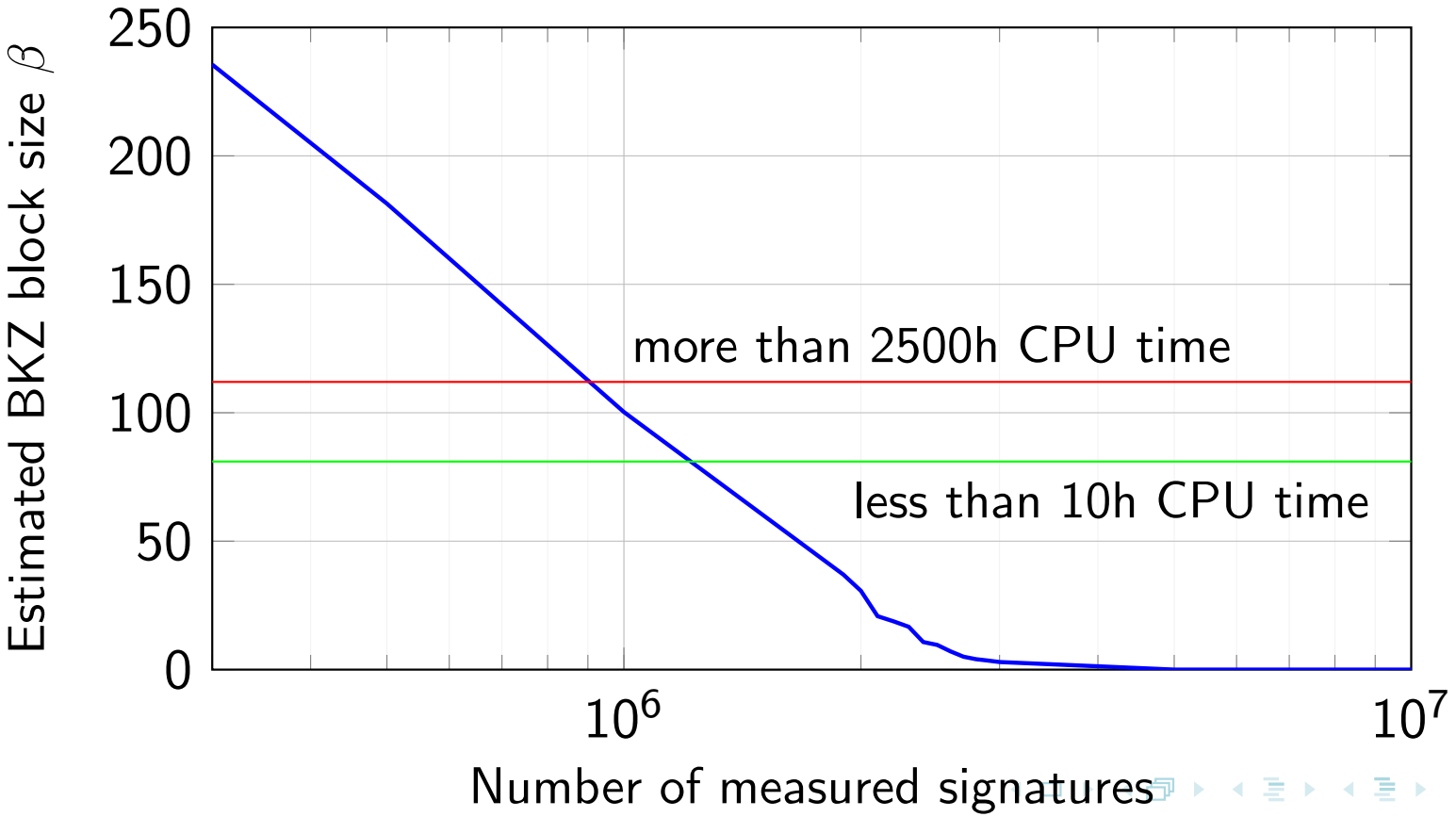
4. Recovering the private key: Falcon-512

- We combine several rows $\mathbf{b_i}$ to attenuate the noise on f, g.

Note: ulterior iterations of HPP solver are much less costly.

Then, two possible ways to recover the exact private key (f, g):

- Mere rounding when $\sigma_{(f',g')-(f,g)}$ is small enough
- Solve DBDD instance with Leaky LWE/NTRU tool [Dac+20]



Conclusion

Sign(m, B):

2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$

- preimage computation
- trapdoor sampler

Preimage computation: Improvement of State-of-the-Art attack.
Trapdoor sampler: Novel attack combining SCA and HPP.

Conclusion

Sign(m, B):

2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$

- preimage computation
- trapdoor sampler

Preimage computation: Improvement of State-of-the-Art attack.
Trapdoor sampler: Novel attack combining SCA and HPP.

Future works:

- Template attack on the `SamplerZ`
- Combination with [Fou+20] (replacing timing attack by STA)

Conclusion

Sign(m, B):

2. $\mathbf{c} \leftarrow \texttt{HashToPoint}(r||m)$
3. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
4. $\mathbf{v} \leftarrow \texttt{ffSampling}(\mathbf{t}, \mathbf{B})$
5. $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{v}) \cdot \mathbf{B}$

- preimage computation
- trapdoor sampler

Preimage computation: Improvement of State-of-the-Art attack.
Trapdoor sampler: Novel attack combining SCA and HPP.

Future works:

- Template attack on the `SamplerZ`
- Combination with [Fou+20] (replacing timing attack by STA)

Questions ?

References I

[Alb+19] Martin R. Albrecht et al. "The General Sieve Kernel and New Records in Lattice Reduction". In: *EUROCRYPT 2019, Part II*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. LNCS. Springer, Heidelberg, May 2019, pp. 717–746.

[Dac+20] Dana Dachman-Soled et al. "LWE with Side Information: Attacks and Concrete Security Estimation". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 329–358.

References II

[DN12] Léo Ducas and Phong Q. Nguyen. "Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 433–450.

[Fou+20] Pierre-Alain Fouque et al. "Key Recovery from Gram-Schmidt Norm Leakage in Hash-and-Sign Signatures over NTRU Lattices". In: *EUROCRYPT 2020, Part III*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12107. LNCS. Springer, Heidelberg, May 2020, pp. 34–63.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *ACM STOC* 40 (2008), pp. 197–206.

References III

[KA21] Emre Karabulut and Aydin Aysu. "FALCON Down: Breaking FALCON Post-Quantum Signature Scheme through Side-Channel Attacks". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 2021, pp. 691–696.

[KH18] Suhri Kim and Seokhie Hong. "Single Trace Analysis on Constant Time CDT Sampler and Its Countermeasure". In: *Applied Sciences* 8 (Oct. 2018), p. 1809.

[NR06] Phong Q. Nguyen and Oded Regev. "Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures". In: *EUROCRYPT 2006* 4004 (2006), pp. 271–288.

Partial countermeasure for BaseSampler

Main idea: invert the sign of the operands to replace the (hardware) underflow by a (logical) overflow.

Replace the last substraction by the following:

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