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

Morgane Guerreau\textsuperscript{1}  Ange Martinelli\textsuperscript{2}  Thomas Ricosset\textsuperscript{1}  Mélissa Rossi\textsuperscript{2}

\textsuperscript{1}Thales,  \textsuperscript{2}ANSSI

September 20, 2022
Introduction

Two Power Analysis attacks on Falcon:

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

M. Guerreau, A. Martinelli, T. Ricosset, M. Rossi

The Hidden Parallelepiped is Back Again
Two Power Analysis attacks on Falcon:

- Efficient DPA attack on the preimage computation
- STA on the trapdoor sampler leading to HPP attack
**Closest Vector Problem:** Given a lattice $\Lambda$ and a vector $c$ in $\mathbb{R}^n$, find a vector $v$ in $\Lambda$ such that $\|v - c\|$ is minimal.
Closest Vector Problem: Given a lattice $\Lambda$ and a vector $c$ in $\mathbb{R}^n$, find a vector $v$ in $\Lambda$ such that $\|v - c\|$ is minimal.

CVP is easy to solve with a good basis, but hard with a bad basis.
Closest Vector Problem: Given a lattice $\Lambda$ and a vector $c$ in $\mathbb{R}^n$, find a vector $v$ in $\Lambda$ such that $\|v - c\|$ 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 $c$ in $\mathbb{R}^n$
Lattice-based cryptography

Closest Vector Problem: Given a lattice $\Lambda$ and a vector $c$ in $\mathbb{R}^n$, find a vector $v$ in $\Lambda$ such that $\|v - c\|$ 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 $c$ in $\mathbb{R}^n$
- Use the good basis (secret key) to solve CVP
Closest Vector Problem: Given a lattice \( \Lambda \) and a vector \( c \) in \( \mathbb{R}^n \), find a vector \( v \) in \( \Lambda \) such that \( \|v - c\| \) 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 \( c \) in \( \mathbb{R}^n \)
- Use the good basis (secret key) to solve CVP
- Anyone can verify the signature \( v \) with a bad basis (public key)
**Lattice-based cryptography**

**Closest Vector Problem:** Given a lattice $\Lambda$ and a vector $\mathbf{c}$ in $\mathbb{R}^n$, find a vector $\mathbf{v}$ in $\Lambda$ 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 $\mathbb{R}^n$
- 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**.
NTRUSign/GGH:

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

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

Signature: $s = \lfloor cB^{-1}\rfloor B$ with $B$ the private basis
NTRUSign/GGH:
- Message $m$ is hashed to a point $c := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $s = \lfloor cB^{-1} \rfloor B$ with $B$ the private basis
$\implies s - c$ belongs to the fundamental parallelepiped $\mathcal{P}(B)$. 
NTRUSign/GGH:

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

Signature: $s = \lfloor cB^{-1} \rfloor B$ with $B$ the private basis

$\implies s - c$ belongs to the fundamental parallelepiped $P(B)$. 
NTRUSign/GGH:

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

Signature: $s = \lfloor cB^{-1} \rfloor B$ with $B$ the private basis

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

Hidden Parallelepiped Problem: Recover $B$ from independent samples drawn uniformly in $\mathcal{P}(B)$ [NR06].
NTRUSign/GGH:
- Message $m$ is hashed to a point $c := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve CVP:

Signature: $s = [cB^{-1}]B$ with $B$ the private basis
$\implies s - c$ belongs to the fundamental parallelepiped $\mathcal{P}(B)$.

Hidden Parallelepiped Problem: Recover $B$ from independent samples drawn uniformly in $\mathcal{P}(B)$ [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 $c := H(m) \in \mathbb{Z}^n$
- Use round-off algorithm to solve $CVP$:

Signature: $s = \lfloor cB^{-1} \rfloor B$ with $B$ the private basis
$\implies s - c$ belongs to the fundamental parallelepiped $P(B)$.

Hidden Parallelepiped Problem: Recover $B$ from independent samples drawn uniformly in $P(B)$ [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: $s$ is not always the closest vector [GPV08].
Hidden Parallelepiped Problem

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

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

Hidden Parallelepiped Problem: Recover $B$ from independent samples drawn uniformly in $\mathcal{P}(B)$ [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: $s$ is not always the closest vector [GPV08].
Instantiation of GPV framework with NTRU lattices

Let $\mathbb{R} = \mathbb{Z}[[x]]/(x^n + 1)$. Private key: $f, g, F, G \in \mathbb{R}$ with $fG - gF = q \mod (x^n + 1)$. Private basis: $B = g - fG - F \Rightarrow b_0 = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1})$.

Sign $(m, B)$:
1. $r \leftarrow$ random salt
2. $c \leftarrow$ HashToPoint($r || m$)
3. $t \leftarrow c \cdot B^{-1}$ \quad preimage computation
4. $v \leftarrow ffSampling(t, B)$ \quad trapdoor sampler
5. $s \leftarrow (t - v) \cdot B$
6. return $(r, s)$. 

$\text{Falcon.Sign}$ $\text{ffSampling}$ $\text{SamplerZ}$ $\text{BaseSampler}$ $z + \sim D_{\mathbb{Z}} + \sigma_{\text{max}}, 0$

$z \sim D_{\mathbb{Z}} + \sigma', \mu$

$v \sim D(t, 0) + \Lambda(B), \sigma, 0$
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 \( fG - gF = q \mod x^n + 1 \)
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: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \implies \mathbf{b}_0 = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1})$
Instantiation of GPV framework with NTRU lattices

Let \( R := \mathbb{Z}[x]/(x^n + 1) \).

Private key: \( f, g, F, G \in R \) with \( fG - gF = q \mod x^n + 1 \)

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

**Sign(m, B):**

1. \( r \leftarrow \) random salt
2. \( c \leftarrow \text{HashToPoint}(r||m) \)
3. \( t \leftarrow c \cdot B^{-1} \)
4. \( v \leftarrow \text{ffSampling}(t, B) \)
5. \( s \leftarrow (t - v) \cdot B \)
6. return \((r, s)\)
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: \( \mathcal{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \implies b_0 = (g_0, \ldots, g_{n-1}, -f_0, \ldots, -f_{n-1}) \)

**Sign(m, B):**

1. \( r \leftarrow \text{random salt} \)
2. \( c \leftarrow \text{HashToPoint}(r||m) \)
3. \( t \leftarrow c \cdot \mathcal{B}^{-1} \)
4. \( v \leftarrow \text{ffSampling}(t, \mathcal{B}) \)
5. \( s \leftarrow (t - v) \cdot \mathcal{B} \)
6. return \((r, s)\)
Power Analysis on the preimage computation

**Sign(m, B):**

2. $c \leftarrow \text{HashToPoint}(r || m)$
3. $t \leftarrow c \cdot B^{-1}$
4. $v \leftarrow \text{ffSampling}(t, B)$
5. $s \leftarrow (t - v) \cdot B$
Differential Power Analysis on the preimage attack

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

Three improvements:

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

<table>
<thead>
<tr>
<th>Number of traces</th>
<th>Probability of success</th>
</tr>
</thead>
<tbody>
<tr>
<td>all patterns</td>
<td></td>
</tr>
<tr>
<td>noise reduction</td>
<td></td>
</tr>
<tr>
<td>SotA</td>
<td></td>
</tr>
</tbody>
</table>
Original attack: DPA on a polynomial multiplication in FFT between a public digest $c$ and a private polynomial $f$ [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 $c$ and a private polynomial $f$ [KA21].

**Three improvements:**

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

<table>
<thead>
<tr>
<th>Number of traces</th>
<th>Probability of success</th>
</tr>
</thead>
<tbody>
<tr>
<td>all patterns</td>
<td>noise reduction</td>
</tr>
</tbody>
</table>

1,000

0.5

0.7

0.9

0.2

0.6

0.4

0.8
Differential Power Analysis on the preimage attack

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

**Three improvements:**

- **Lowering the complexity** of exhaustive search: double precision is unnecessary to recover the key.
- **Halving the number of required traces** by combining 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 $c$ and a private polynomial $f$ [KA21].

Three improvements:

- Lowering the complexity of exhaustive search: double precision is unnecessary to recover the key.
- Halving the number of required traces by combining 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. $c \leftarrow \text{HashToPoint}(r || m)$
3. $t \leftarrow c \cdot B^{-1}$
4. $v \leftarrow \text{ffSampling}(t, B)$
5. $s \leftarrow (t - v) \cdot B$

- 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

\[ \mathbf{v} \sim D_{(t,0) + \Lambda(B), \sigma, 0} \]
\[ z \sim D_{\mathbb{Z}, \sigma', \mu} \]
\[ z^+ \sim D_{\mathbb{Z}^+, \sigma_{\text{max}}, 0} \]
1. Side-channel analysis on the BaseSampler

\[ \mathbf{v} \sim D_{(t,0)+\Lambda(B),\sigma,0} \]
\[ z \sim D_{\mathbb{Z},\sigma',\mu} \]
\[ z^+ \sim D_{\mathbb{Z}^+,\sigma_{\text{max}},0} \]

**BaseSampler()**:
1. \( u \leftarrow \text{UniformBits}(72) \)
2. \( z^+ \leftarrow 0 \)
3. for \( i = 0 \ldots 16 \) do
4. \( z^+ \leftarrow z^+ + [u < \text{RCDT}[i]] \)
5. end
6. return \( z^+ \)
1. Side-channel analysis on the BaseSampler

BaseSampler():

1. \( u \leftarrow \text{UniformBits}(72) \)
2. \( z^+ \leftarrow 0 \)
3. \textbf{for} \( i = 0 \ldots 16 \) \textbf{do}
4. \hspace{1em} \( z^+ \leftarrow z^+ + [u < \text{RCDT}[i]] \)
5. \textbf{end}
6. \textbf{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

Falcon.Sign $\rightarrow$ ffSampling $\rightarrow$ SamplerZ $\rightarrow$ BaseSampler

$\mathbf{v} \sim D(t,0)+\Lambda(B),\sigma,0$

$\mathbf{z} \sim D_{\mathbb{Z},\sigma',\mu}$

$\mathbf{z}^+ \sim D_{\mathbb{Z}^+,\sigma_{\max},0}$

**BaseSampler()**:

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

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

$\implies$ High difference in Hamming weight [KH18]
1. Side-channel analysis on the BaseSampler

BaseSampler()

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

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

\[ \Rightarrow \] High difference in Hamming weight [KH18]

We are able to retrieve the value of \( z^+ \) through STA.
“Shifted” Babai’s nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $\vec{t}$ before solving CVP.
“Shifted” Babai’s nearest-plane algorithm: A Gaussian translation vector $\vec{z}$ is applied to $t$ 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

\[ \mathbb{P}[z_i^+ = 0] \approx \text{erf}(\frac{\sqrt{2}}{2\sigma_i}) \in [0.4111, 0.5613] \text{ for all } i \in [0, n - 1] \]

(because of rejection sampling, not all \( z_i^+ \) are kept)
3. Application of HPP solver

\[ P[z_i^+ = 0] \approx \text{erf}\left(\frac{\sqrt{2}}{2\sigma_i}\right) \in [0.4111, 0.5613] \text{ for all } i \in [0, n - 1] \]

(because of rejection sampling, not all \(z_i^+\) are kept)

\[ \implies \text{we apply HPP solver from [DN12] on 40 to 55\% of the signatures} \]
3. Application of HPP solver

\[ \mathbb{P}[z_i^+ = 0] \approx \text{erf}\left(\frac{\sqrt{2}}{2\sigma_i}\right) \in [0.4111, 0.5613] \text{ for all } i \in [0, n - 1] \]

(because of rejection sampling, not all \( z_i^+ \) are kept)

\[ \implies \text{we apply HPP solver from [DN12] on 40 to 55\% of the signatures} \]

**Baibai’s nearest-plane algorithm:** uses \( \tilde{\mathbf{B}} := \text{GSO}(\mathbf{B}) \) instead of \( \mathbf{B} \) to solve CVP.
3. Application of HPP solver

\[ P[z_i^+ = 0] \approx \operatorname{erf}\left( \frac{\sqrt{2}}{2\sigma_i} \right) \in [0.4111, 0.5613] \text{ for all } i \in [0, n - 1] \]

(because of rejection sampling, not all \( z_i^+ \) are kept)

\[ \implies \text{ we apply HPP solver from [DN12] on 40 to 55\% of the signatures} \]

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

\[ \implies \text{ signatures belong to } \mathcal{P}(\tilde{B}), \text{ not } \mathcal{P}(B). \]
3. Application of HPP solver

\[ \mathbb{P}[z_i^+ = 0] \approx \text{erf}\left(\frac{\sqrt{2}}{2\sigma_i}\right) \in [0.4111, 0.5613] \text{ for all } i \in [0, n-1] \]

(because of rejection sampling, not all \(z_i^+\) are kept)

\[ \implies \text{we apply HPP solver from } [DN12] \text{ on } 40 \text{ to } 55\% \text{ of the signatures} \]

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

\[ \implies \text{signatures belong to } \mathcal{P}(\tilde{\mathbf{B}}), \text{ not } \mathcal{P}(\mathbf{B}). \]

\[ \implies \text{we can only retrieve rows of } \tilde{\mathbf{B}} \text{ with HPP solver.} \]
3. Application of HPP solver

\[ P[z_i^+ = 0] \approx \text{erf}\left(\frac{\sqrt{2}}{2\sigma_i}\right) \in [0.4111, 0.5613] \text{ for all } i \in [0, n - 1] \]

(because of rejection sampling, not all \(z_i^+\) are kept)

\[ \implies \text{we apply HPP solver from [DN12] on 40 to 55\% of the signatures} \]

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

\[ \implies \text{signatures belong to } \mathcal{P}(\tilde{\mathbf{B}}), \text{ not } \mathcal{P}(\mathbf{B}). \]

\[ \implies \text{we can only retrieve rows of } \tilde{\mathbf{B}} \text{ with HPP solver.} \]

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

\( \tilde{b}_0, \ldots, \tilde{b}_3 \approx b_0, \ldots, b_3 \) and \( \tilde{b}_n, \ldots, \tilde{b}_{n+3} \approx b_n, \ldots, b_{n+3} \)
4. Recovering the private key: Falcon-512

We combine several rows $b_i$ 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 $b_i$ 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 $b_i$ 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 $b_i$ 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]

---

![Graph showing the relationship between Estimated BKZ block size $\beta$ and Number of measured signatures.](image-url)

- More than 2500h CPU time
- Less than 10h CPU time
Conclusion

Sign(m, B):

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

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

\[ \text{Sign}(m, B): \]

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

- **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**

\[ \text{Sign}(m, B) : \]
\begin{align*}
    2. & \quad c \leftarrow \text{HashToPoint}(r \| m) \\
    3. & \quad t \leftarrow c \cdot B^{-1} \\
    4. & \quad v \leftarrow \text{ffSampling}(t, B) \\
    5. & \quad s \leftarrow (t - v) \cdot B
\end{align*}

- 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 II


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. $b \leftarrow 0xffffffff$
2. $b := b - \bar{u} + RCDT[i] + c$
3. return $b \gg 24$

State of the register before the last operation:

State of the register after the last operation (original implementation):

State of the register after the last operation (with countermeasure):

- Bit set to 0
- Bit set to 1
- Bit set to either 1 or 0