Breaking HuFu with 0 Leakage

A Side-Channel Analysis

Julien Devevey[1], **Morgane Guerreau**[2], Thomas Legavre[1,3,4], Ange Martinelli[1], Thomas Ricosset[3]

1. ANSSI
2. CryptoNext Security
3. Thales
4. LIP6, Sorbonne Université

Introduction

What is HuFu?

- Signature scheme based on unstructured lattices
- Based on the Hash-and-Sign paradigm [GPV08] (like Falcon)
- Round 1 candidate to NIST on-ramp post-quantum signature competition

Introduction

What is HuFu?

- Signature scheme based on unstructured lattices
- Based on the Hash-and-Sign paradigm [GPV08] (like Falcon)
- Round 1 candidate to NIST on-ramp post-quantum signature competition

Why attack it?

- Absence of structure counters attacks on Falcon
- Trapdoor sampling a la [MP12] is used in other contexts (IBEs...)

Results

- We target sensible multiplications and the base discrete Gaussian sampler with power analysis and recover many coefficients of the signing key.
- The attacks are completed using lattice reduction whose cost we estimate depending on the amount of recovered coefficients

1. The HuFu Signature Scheme

Hash-and-sign for Lattices and HuFu

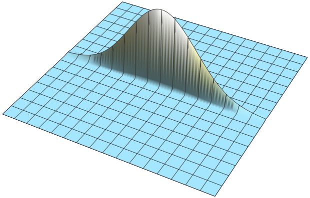
Generic framework for lattice-based signatures [GPV08] such as Falcon.
Instanciated as follows for HuFu:

- Verification key: a matrix $\mathbf{A} = (\mathbf{I}_m|\tilde{\mathbf{A}}|\mathbf{B})$ with $\mathbf{B} = p\mathbf{I}_m - \tilde{\mathbf{A}}\mathbf{S} - \mathbf{E} \bmod pq$,
- Signing key: $\mathbf{sk}^\top = q(\mathbf{I}_m|\mathbf{S}|\mathbf{E})$, a short basis of $\Lambda = \{\mathbf{A}\mathbf{x} = 0 \bmod pq, \mathbf{x} \in \mathbb{Z}^k\}$,
- Given a message μ, sign by giving a short preimage $\mathbf{x}$ of $\mathbf{u} = H(\mu)$ by $\mathbf{A}$,
- How is $\mathbf{x}$ sampled?

First Try

Take $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^k+\mathbf{v}/q,\bar{r}^2}$ and set

$$\mathbf{x} = \mathbf{sk} \cdot \mathbf{z}.$$

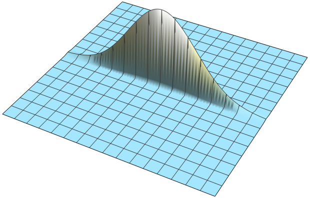


$$\mathbf{A} \cdot \mathbf{sk} \cdot \mathbf{z} = p\mathbf{v} \bmod pq$$

First Try

Take $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^k+\mathbf{v}/q,\bar{r}^2}$ and set

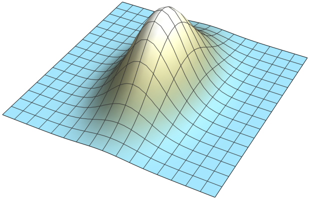
$$\mathbf{x} = \mathbf{sk} \cdot \mathbf{z}.$$



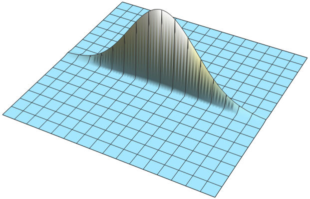
$$\mathbf{A} \cdot \mathbf{sk} \cdot \mathbf{z} = p\mathbf{v} \bmod pq$$

- Set $\mathbf{v} = \lfloor \mathbf{u}/p \rceil$: approximate preimage
- Add $\mathbf{u} \bmod p$ to get an exact preimage
- The distribution leaks sk!

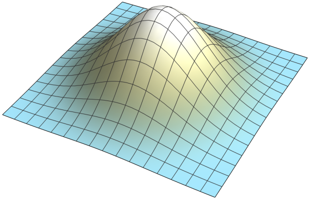
Adding a Perturbation



$+$



$=$

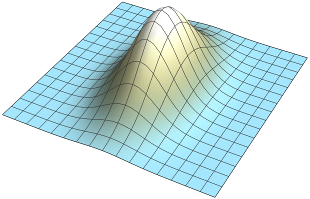


$\mathbf{p} \hookleftarrow D_{\mathbb{Z}^k,\Sigma_p}$
Sampled using Cholesky decomposition

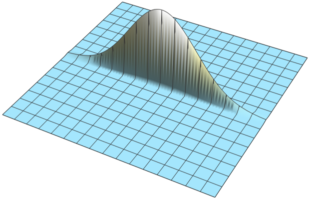
$\mathbf{sk} \cdot \mathbf{z}$
$\mathbf{z} \hookleftarrow D_{\mathbb{Z}^k+\mathbf{c},\bar{r}^2}$
$\mathbf{c} = \lfloor (\mathbf{u} - \mathbf{Ap})/p \rceil / q$

$\mathbf{x}$
Short approximate preimage of $\mathbf{u}$
Not leaky

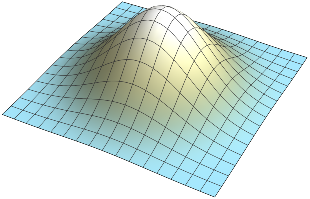
Adding a Perturbation



$+$



$=$



$\mathbf{p} \hookleftarrow D_{\mathbb{Z}^k,\Sigma_p}$
Sampled using Cholesky decomposition

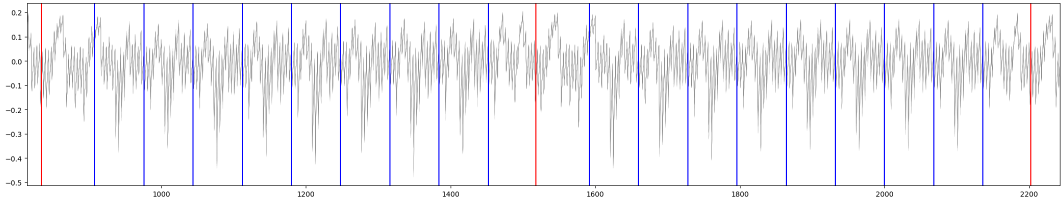
$\mathbf{sk} \cdot \mathbf{z}$
$\mathbf{z} \hookleftarrow D_{\mathbb{Z}^m+\mathbf{c},\bar{r}^2}$
$\mathbf{c} = \lfloor(\mathbf{u} - \mathbf{Ap})/p\rceil/q$

$\mathbf{x}$
Short approximate preimage of $\mathbf{u}$
Not leaky

2. Side-Channel Analysis

Experimental set-up

Acquisition device: **ChipWhisperer Lite** with a Cortex M4 target.
Targetted C code is taken from the NIST submission package.



Code & some power traces available on a GitHub repository (link in paper).

Feel free to reach out!

Overview of the leakage spots

Algorithm HuFu Sign

$\mathbf{p} \leftarrow \text{SampleP}(\text{sk})$
$(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) \leftarrow \mathbf{p}$
$\mathbf{v} \leftarrow \text{ComputeV}(\mathbf{A}, \mathbf{p}, \mu)$
$\mathbf{z} \leftarrow q \cdot \text{SampleZ}_d(\mathbf{v}/q)$
$\mathbf{x}_0 \leftarrow \mathbf{E}\mathbf{z} + \mathbf{p}_0$
$\mathbf{x}_1 \leftarrow \mathbf{S}\mathbf{z} + \mathbf{p}_1$
$\mathbf{x}_2 \leftarrow \mathbf{z} + \mathbf{p}_2$
if $\|(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)\| > B$ **then**
    goto 1
end if
return $\sigma = (\mathbf{x}_1, \mathbf{x}_2)$

Gaussian sampler
matrix-vector multiplication
matrix-vector multiplication

Leakage in matrix-vector multiplication

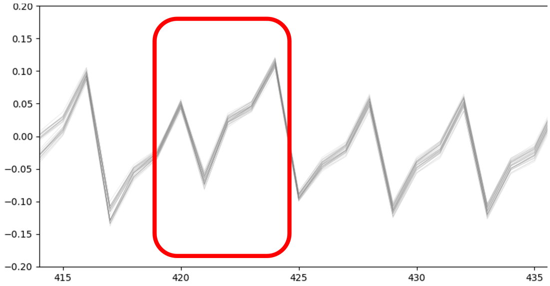
Targeted operations: $S_{i,j} \cdot z_i$ (resp. $E_{i,j} \cdot z_i$)

Coefficients of $\mathbf{S}$ (resp. $\mathbf{E}$) are ternary and follow a binomial distribution.
$\rightarrow$ only three possible outputs for $S_{i,j} \cdot z_i$:

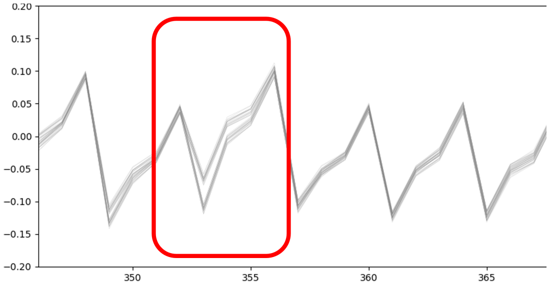
1. 0 (with probability 0.5)
2. z_i
3. $-z_i$

$\rightarrow$ we should see it in the power traces!

How to gain 0 leakage

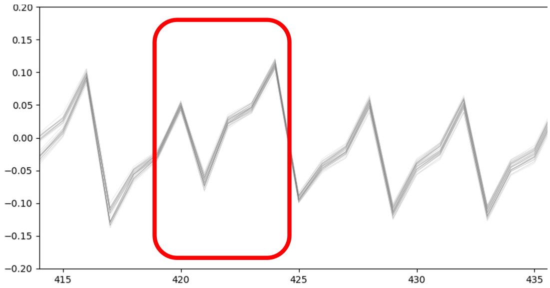


(a) $S_{i,j} = 0.$

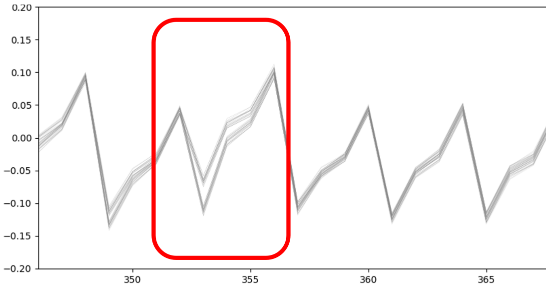


(b) $S_{i,j} \neq 0.$

How to gain 0 leakage



(a) $S_{i,j} = 0$.



(b) $S_{i,j} \neq 0$.

With 1,500 traces, we can recover 98% of the $S_{i,j}$ (resp. $E_{i,j}$) equal to zero.

A (simple) countermeasure

$\mathbf{x}_0$ is used only in the following (non-sensitive) check:

$$\|(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)\| > B$$

$\mathbf{x}_0 + \mathbf{e}$ can also be computed as follows:

$$\mathbf{x}_0 + \mathbf{e} = \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2$$

which totally removes the secret component $\mathbf{E}$.

A (simple) countermeasure

$\mathbf{x}_0$ is used only in the following (non-sensitive) check:

$$\|(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)\| > B$$

$\mathbf{x}_0 + \mathbf{e}$ can also be computed as follows:

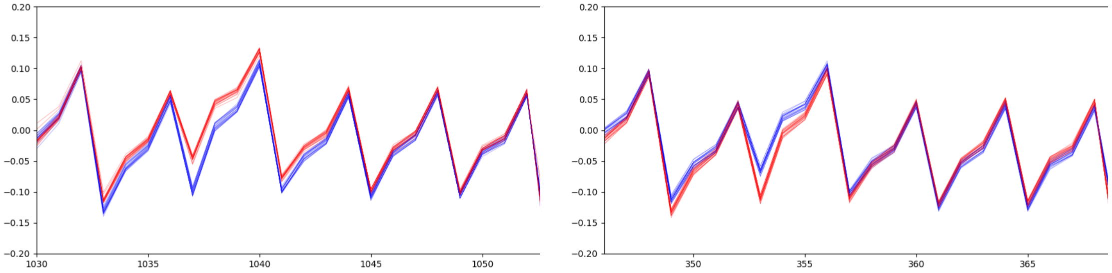
$$\mathbf{x}_0 + \mathbf{e} = \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2$$

which totally removes the secret component $\mathbf{E}$.

$\rightarrow$ let's improve our attack to gain additional information on $\mathbf{S}$!

How to gain more-than-0 leakage

What if we had (by any chance) the sign of z_i?



(a) $S_{i,j} = -1$. **(b)** $S_{i,j} = 1$.

Figure: Power traces in red (resp. blue) correspond to $z_i < 0$ (resp. $z_i > 0$).

Leakage in Gaussian sampler

SampleZ($\mathtt{center}$)**:**

1. $v \leftarrow \mathtt{Rnd}(72)$
2. $c \leftarrow (\mathtt{center} > 8) * (16 - 2 * \mathtt{center}) + \mathtt{center}$
3. $z^+ \leftarrow 0$
4. **for** $i = 0 \ldots 26$ **do**
5. $\quad z^+ \leftarrow z^+ + [\![v < \mathtt{RCDT}[c][i]]\!]$
6. **end**
7. $z \leftarrow [\![\mathtt{center} > 8]\!] * (27 - 2 * z^+) + z^+ - 13$
8. **return** z

input: $\mathtt{center} \in [0, 15]$
output: $z \in [-12, 12]$

Consequences on the attack

$z = 0 \implies z^+ \in [13, 14]$ (depending on `center` value)
This implies 13 or 14 incrementations in the for loop.

Previous attacks on other schemes were relying on the fact that

$$z = 0 \iff \text{no incrementation at all}$$

Now we have much more noise!

Consequences on the attack

$z = 0 \implies z^+ \in [13, 14]$ (depending on `center` value)
This implies 13 or 14 incrementations in the for loop.

Previous attacks on other schemes were relying on the fact that

$$z = 0 \iff \text{no incrementation at all}$$

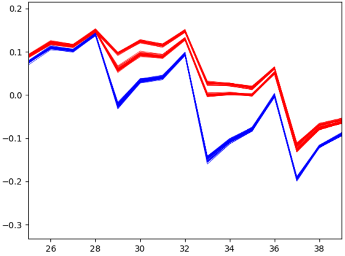
Now we have much more noise!

$\rightarrow$ we will not target the for loop

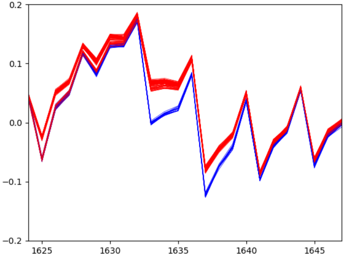
Targetted C code

```
int c = center;
c = (c > 8) * (16 - 2 * c) + c;                         // c computation
z = 0;
for (u = 0; u < TABLE_LEN; u += 3)
{
    uint32_t w0, w1, w2, cc;
    w0 = dist0[c][u + 2];
    w1 = dist0[c][u + 1];
    w2 = dist0[c][u + 0];
    cc = (v0 - w0) >> 31;
    cc = (v1 - w1 - cc) >> 31;
    cc = (v2 - w2 - cc) >> 31;
    z += (int)cc;
}
return (center > 8) * (27 - 2 * z) + z - 13;            // z computation
```

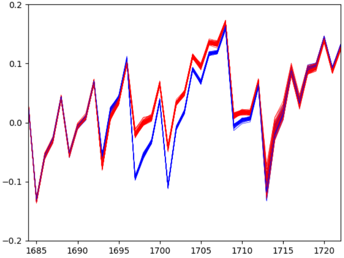
Sign recovery of z



(a) Computation of c.



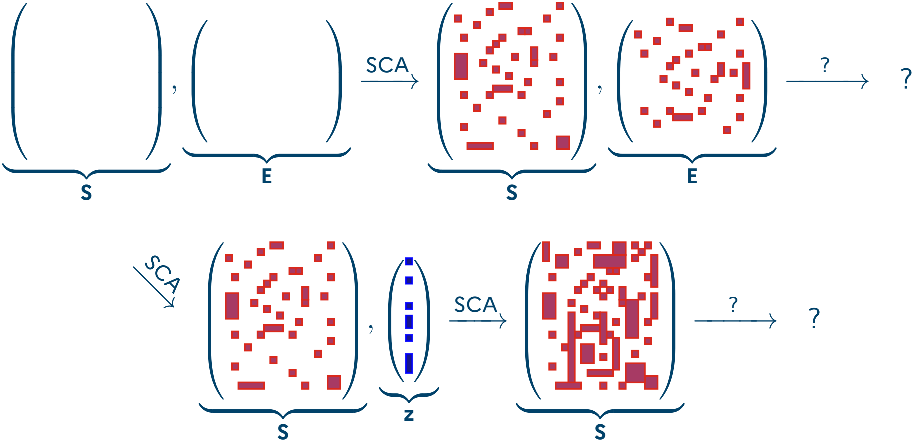
(b) First multiplication.



(c) Final subtraction.

With 1,500 traces, we can recover 75% of the $S_{i,j}$ given prior information on z_i.

Attacks Summary



3. Forgery

First attack when S and E are known

Given an LWE sample $\mathbf{As} + \mathbf{e}$ and some 0s of $\mathbf{s}$ and $\mathbf{e}$, how do we exploit them?

First attack when S and E are known

Given an LWE sample $\mathbf{As} + \mathbf{e}$ and some 0s of $\mathbf{s}$ and $\mathbf{e}$, how do we exploit them?

- Remove the i-th column of $\mathbf{A}$ if $s_i = 0$: dimension reduced by one.

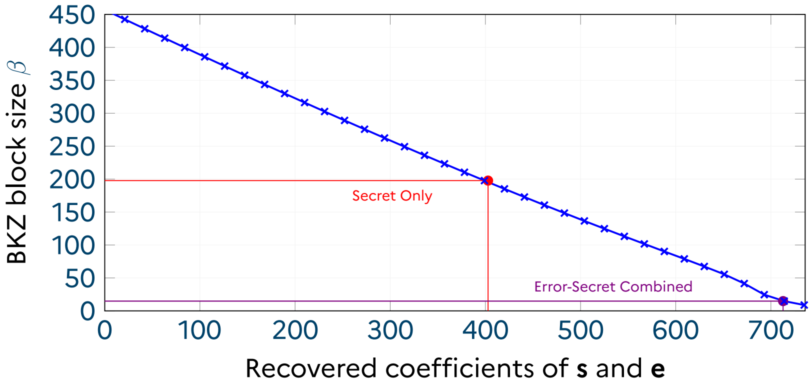
First attack when S and E are known

Given an LWE sample $\mathbf{A}\mathbf{s} + \mathbf{e}$ and some 0s of $\mathbf{s}$ and $\mathbf{e}$, how do we exploit them?

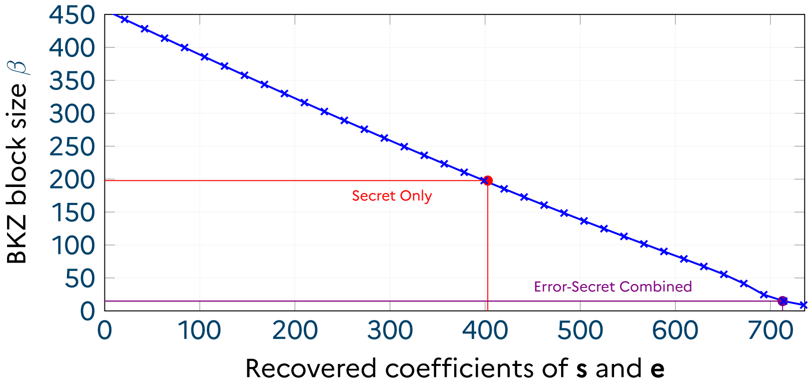
- Remove the i-th column of $\mathbf{A}$ if $s_i = 0$: dimension reduced by one.
- Write $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle$ if $e_i = 0$. Dimension reduced by one. Some rewriting involved to find a new LWE instance with one less dimension.

What is the cost of BKZ on the new LWE instance once every hint has been incorporated?

Remaining Cost of the Attack



Remaining Cost of the Attack



Conclusion: preventing the leakage on **E** is critical.

Forging for specific vectors

Assuming the first k columns $\mathbf{S}_k$ of $\mathbf{S}$ are known via the previous attack, what can we do with them?

Forging for specific vectors

Assuming the first k columns $\mathbf{S}_k$ of $\mathbf{S}$ are known via the previous attack, what can we do with them?

- If the target is $\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}$, then we set $\mathbf{p} = \mathbf{0}$, $\mathbf{v} = \lfloor \mathbf{u}/p \rceil$ and $\mathbf{z} = \mathbf{v}$. A signature would then be:

$$\begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{S} \\ \mathbf{I}_m \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} \mathbf{S}_k & \mathbf{0} \\ \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \mathbf{v}.$$

Forging for specific vectors

Assuming the first k columns $\mathbf{S}_k$ of $\mathbf{S}$ are known via the previous attack, what can we do with them?

- If the target is $\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}$, then we set $\mathbf{p} = \mathbf{0}$, $\mathbf{v} = \lfloor \mathbf{u}/p \rceil$ and $\mathbf{z} = \mathbf{v}$. A signature would then be:

$$\begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{S} \\ \mathbf{I}_m \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} \mathbf{S}_k & \mathbf{0} \\ \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \mathbf{v}.$$

This vector is short, but which message did we sign?

Finding specific vectors

- Choose any μ and compute $\mathbf{u} = H(\mu) = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix}$.
- Write $\mathbf{A} = \begin{pmatrix} \mathbf{A}_h \\ \mathbf{A}_l \end{pmatrix}$
- Find short $\mathbf{x}'$ such that $\mathbf{A}_l \mathbf{x}' = \mathbf{u}_2$ with lattice reduction
- Set $\mathbf{u}' = \mathbf{u} - \mathbf{A}\mathbf{x}' = \begin{pmatrix} \mathbf{u}'_1 \\ \mathbf{0} \end{pmatrix}$
- We are back to the previous case!

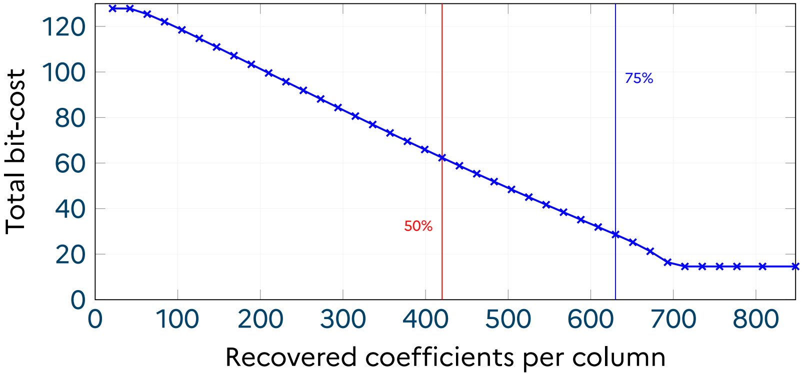
How much costs a forgery?

We start by gathering d coefficients per column.

- First step: complete k columns via lattice reduction: k times LWE with dimension reduced by d
- Second step: one more lattice reduction to find $\mathbf{x}'$: dimension reduced by k but bound B' on $\|\mathbf{x}'\|$ that worsens with k
- Third step: forgery for specific vectors (essentially free)

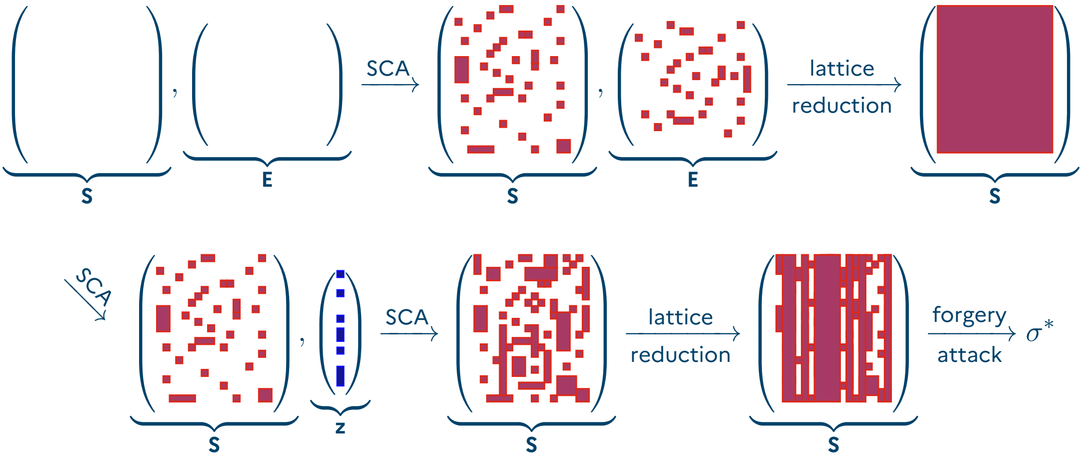
All that remains is to optimize over k.

Final Cost



Total bit-cost
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
Recovered coefficients per column
50%
75%

Attacks Summary



Conclusion

Our approach is flexible:

- Other schemes: our attacks targeted only SampleZ and the subsequent multiplication, which is a building block in [MP12] trapdoors.
- Improved protection: our lattice reduction analysis allows us to predict attacks with a reduced amount of recovered coefficients

Conclusion

Our approach is flexible:

- Other schemes: our attacks targeted only SampleZ and the subsequent multiplication, which is a building block in [MP12] trapdoors.
- Improved protection: our lattice reduction analysis allows us to predict attacks with a reduced amount of recovered coefficients

Thank you for your attention!