Mitaka

A Simpler, Parallelizable, Maskable Variant of Falcon

Thomas Espitau, Pierre-Alain Fouque, Francois Gérard, Melissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, Yang Yu

Eurocrypt 2022

Lattice signatures

Two finalists among the three:

FALCON

"Hash-and-sign" in lattices [GPV'08]
+ **NTRU trapdoors** [DLP'14]

✓ compact, fast

✗ restricted parameter set, quite hard to implement and protect against side-channels

CRYSTALS-DILITHIUM

Fiat-Shamir "with abort" [Lyu12]
+ module lattices

✗ larger bandwith

✓ large range of parameter sets, easier to implement and protect against side-channels

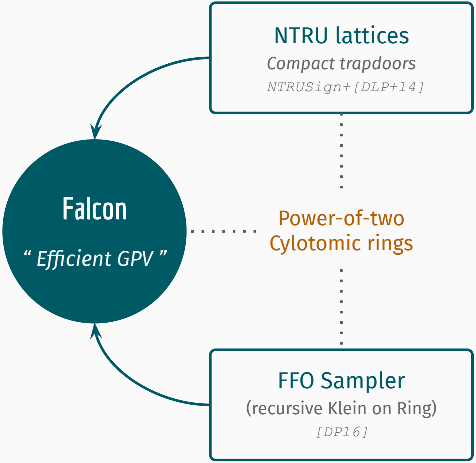
Two finalists among the three:

FALCON	CRYSTALS-DILITHIUM
"Hash-and-sign" in lattices [GPV'08] + **NTRU trapdoors** [DLP'14]	**Fiat-Shamir "with abort"** [Lyu12] + module lattices

Introducing: [Mitaka]

trying to reach best of both worlds

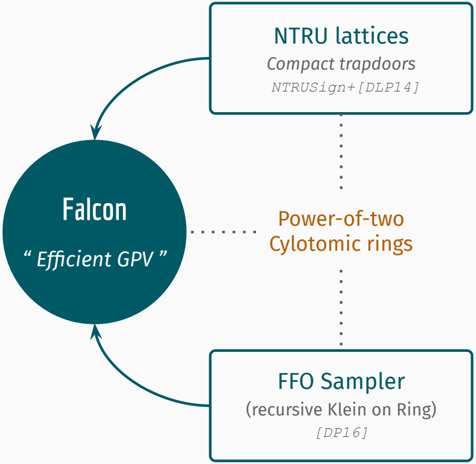
- ✓ compact, fast
- ✓ large range of parameters sets
- ✓ easier to implement and protect against side-channels
- ✓ implementable in fixed-point arithmetic

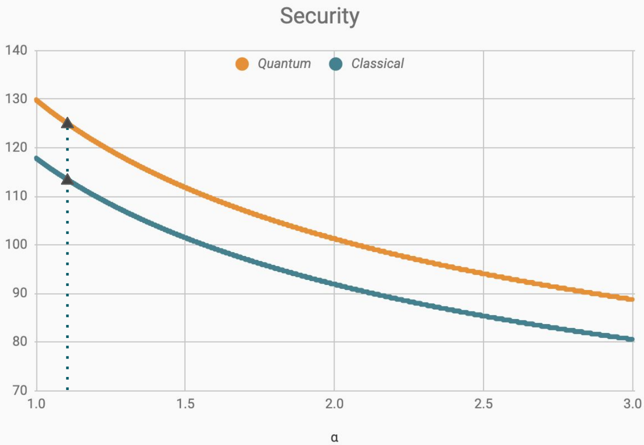


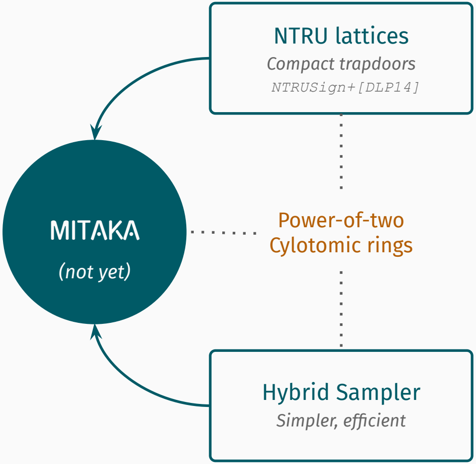
NTRU lattices: free rank 2 modules over cyclotomic ring

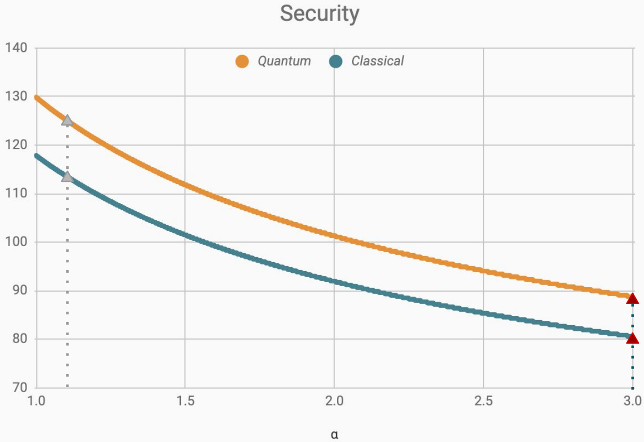
Quasi-linear **thanks to the ring** *but*

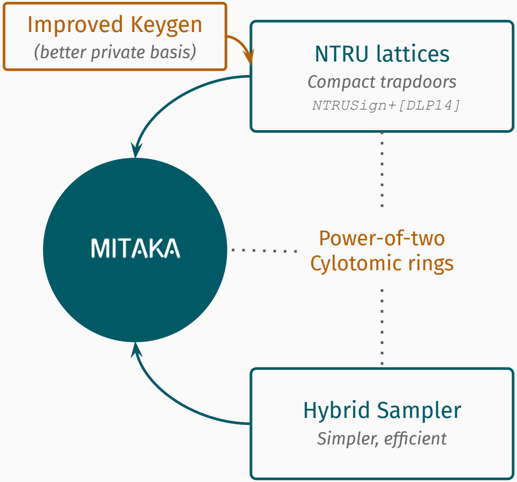
- Few parameter sets
- Complicated implementation
- Complicated masking

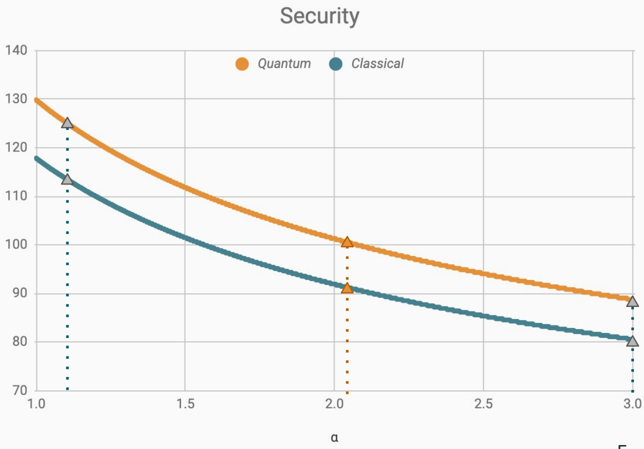

NTRU lattices
Compact trapdoors
NTRUSign+[DLP14]
Falcon
“ Efficient GPV ”
Power-of-two
Cylotomic rings
FFO Sampler
(recursive Klein on Ring)
[DP16]

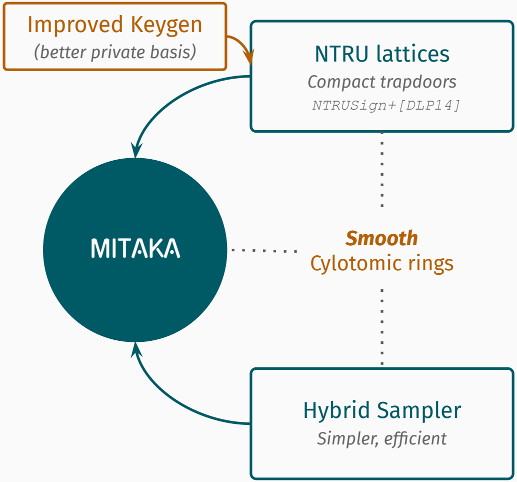

Security
Quantum
Classical
140
130
120
110
100
90
80
70
1.0
1.5
2.0
2.5
3.0
α

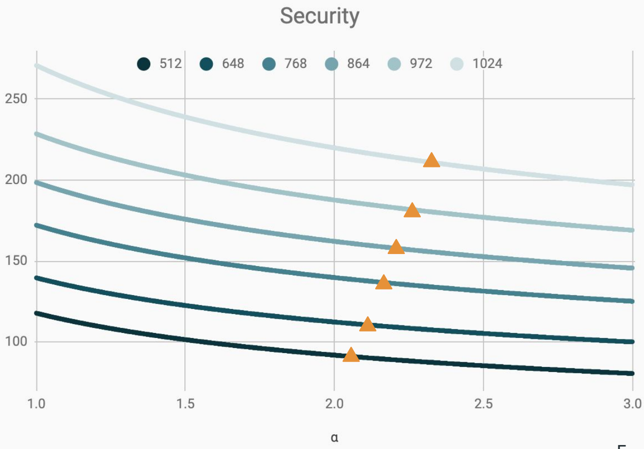

NTRU lattices
Compact trapdoors
NTRUSign+[DLP14]
MITAKA
(not yet)
Power-of-two
Cylotomic rings
Hybrid Sampler
Simpler, efficient

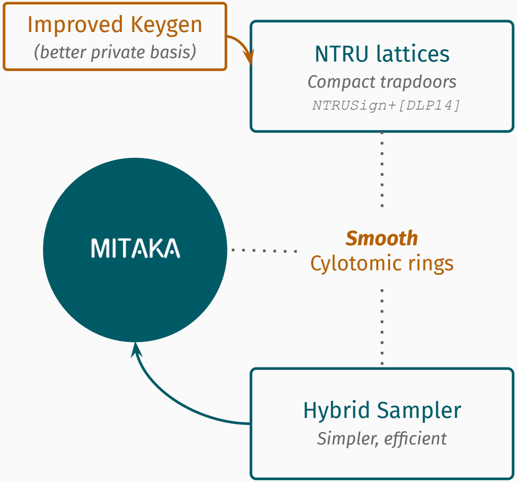

Security
Quantum
Classical
140
130
120
110
100
90
80
70
1.0
1.5
2.0
2.5
3.0
α

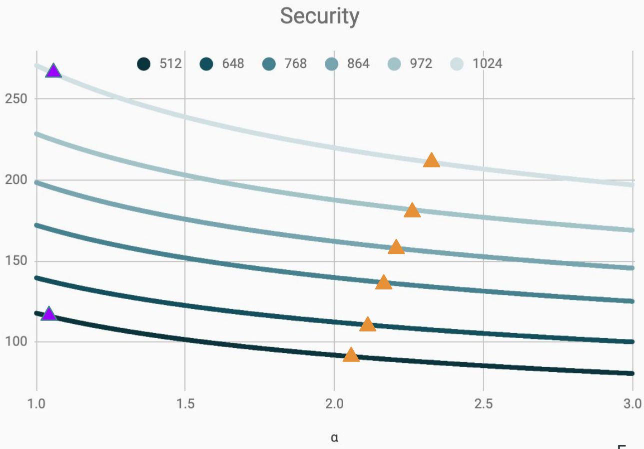

Improved Keygen
(better private basis)
NTRU lattices
Compact trapdoors
NTRUSign+[DLP14]
MITAKA
Power-of-two
Cylotomic rings
Hybrid Sampler
Simpler, efficient

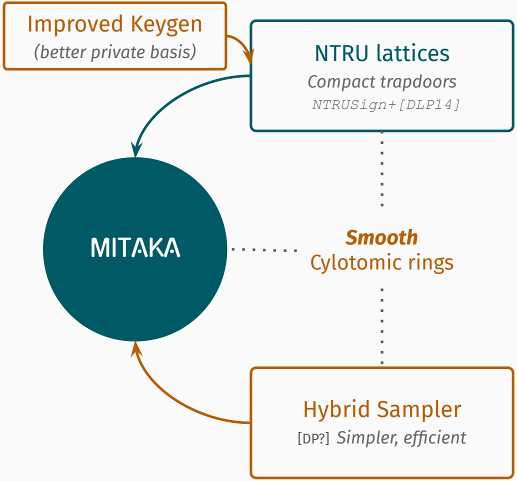

Security
Quantum
Classical
140
130
120
110
100
90
80
70
1.0
1.5
2.0
2.5
3.0
α


Improved Keygen
(better private basis)
NTRU lattices
Compact trapdoors
NTRUSign+[DLP14]
MITAKA
Smooth
Cylotomic rings
Hybrid Sampler
Simpler, efficient

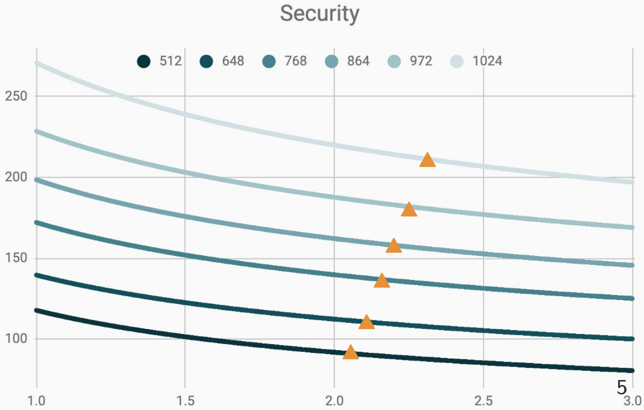

Security
512
648
768
864
972
1024
250
200
150
100
1.0
1.5
2.0
2.5
3.0
α

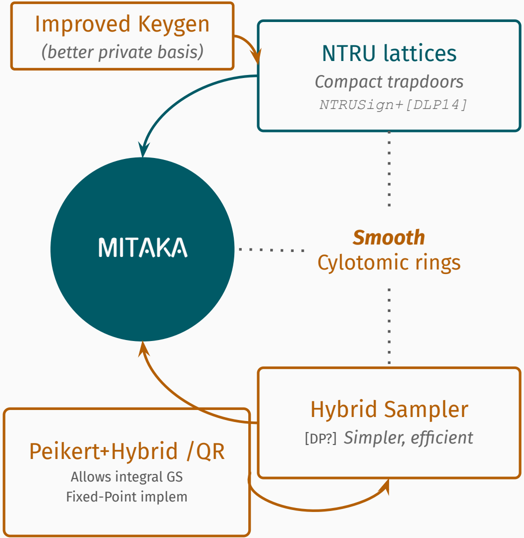

Improved Keygen
(better private basis)
NTRU lattices
Compact trapdoors
NTRUSign+[DLP14]
MITAKA
Smooth
Cylotomic rings
Hybrid Sampler
Simpler, efficient


Security
512
648
768
864
972
1024
250
200
150
100
1.0
1.5
2.0
2.5
3.0
α

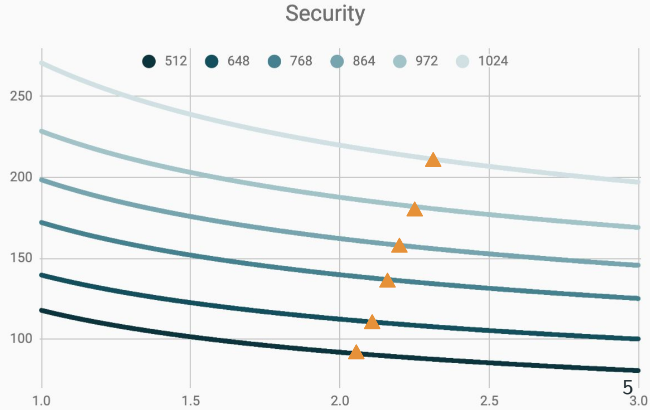


Simple | Efficient | Compact | Versatile | Maskable





Simple | Efficient | Compact | Versatile | Maskable



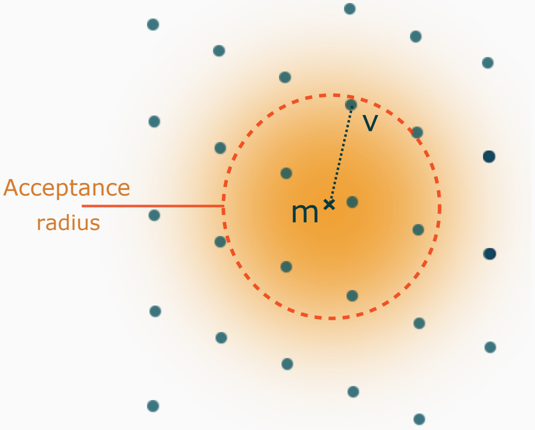
Hash-and-sign over lattices

Simplified $\text{Sign}_{\mathbf{sk},\sigma}(\text{msg})$:

1. $\mathbf{m} = \mathcal{H}(\text{msg})$
2. $\mathbf{v} \leftarrow \text{GaussianSampler}(\mathbf{sk}, \mathbf{m}, \sigma)$
3. Signature: $\mathbf{s} = \mathbf{m} - \mathbf{v}$.

Simplified $\text{Verif}_{\mathcal{L}=\mathbf{pk}}(\text{msg}, \mathbf{s})$:

1. If $\|\mathbf{s}\|$ too big, reject.
2. If $\mathbf{m} - \mathbf{s} \notin \mathcal{L}$, reject.
3. Accept.



Simplified $\mathrm{Sign}_{\mathbf{sk},\sigma}(\mathrm{msg})$:

1. $\mathbf{m} = \mathcal{H}(\mathrm{msg})$
2. $\mathbf{v} \leftarrow \mathrm{GaussianSampler}(\mathbf{sk}, \mathbf{m}, \sigma)$
3. Signature: $\mathbf{s} = \mathbf{m} - \mathbf{v}$.

Simplified $\mathrm{Verif}_{\mathcal{L}=\mathbf{pk}}(\mathrm{msg}, \mathbf{s})$:

1. If $\|\mathbf{s}\|$ too big, reject.
2. If $\mathbf{m} - \mathbf{s} \notin \mathcal{L}$, reject.
3. Accept.

Requirements

$\mathbf{CVP}_\gamma$ hard $\Rightarrow$ σ small $\Rightarrow$ **sk** has short vectors

Hard to compute **sk** just from **pk**	Easy to generate **pk** just from **sk**

sk is called *"a trapdoor"*

Generating trapdoors is an interesting challenge

[HPSS'00, AP'09, MP'12, DLP'14, CGM'19, GL'20, CPSWX'20...]

Sampling over (structured) lattices

Lattice Gaussian samplers = decoding + randomization

CVP solvers

Babai's Round-off:

$$\mathbf{u} = \mathbf{B}\lceil \mathbf{B}^{-1}\mathbf{t}\rfloor$$

Babai's Nearest Plane:

"adaptive" choice of hyperplanes

Gaussian samplers

Peikert sampler:

Randomize the whole integer rounding

Klein sampler:

Randomize each hyperplane choice

Ducas-Prest hybrid sampler: in between for modules over rings, where rounding is replaced by rounded-sampling over the ring.

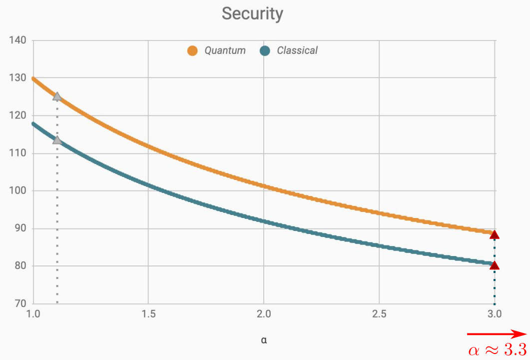
	Quality	Pros	Cons
Peikert	$s_1(\mathbf{B})$ (largest sing. value)	fast simple	worst quality (*lower security*)
Klein	$\max_i \lVert\widetilde{\mathbf{b}_i}\rVert$ (Gram-Schmidt)	best quality (*higher security*)	slower more involved
Hybrid	$s_1(\widetilde{\mathbf{B}})$	**Good tradeoffs when $\mathcal{R}$ has a *good basis***	

When $\mathcal{R} = \mathbb{Z}[x]/(x^d + 1)$, $d = 2^n$, and for NTRU q-ary lattices, qualities are $\alpha\sqrt{q}$

Asymptotic quality

Sampler	$\alpha\sqrt{q}$	Best achievable α
Peikert	$s_1(\mathbf{B})$	$O(d^{1/4}\sqrt{\log d})$
Hybrid	$s_1(\widetilde{\mathbf{B}})$	$O(d^{1/8}\log^{1/4} d)$
`Falcon`	$\max_i \lVert\widetilde{\mathbf{b}}_i\rVert$	$O(1)$

Concrete bitsecurity as a function of α, $d = 512$



Improving the Keygen

NTRU lattice $\mathcal{L}_{\mathsf{NTRU}}(a)$

$$f, g \in \mathcal{R} \to a := f^{-1} g \; [q].$$

$$\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} a \\ -1 \end{bmatrix} = 0 \; [q]$$

Trapdoor

Short basis **B** of $\mathcal{L}_{\mathsf{NTRU}}(a)$ with **good quality** wrt. a sampler.

$$\underbrace{\begin{bmatrix} f & g \\ ? & ? \end{bmatrix}}_{=B} \begin{bmatrix} a \\ -1 \end{bmatrix} = 0 \; [q]$$

Computing B

- Sample f, g Gaussians so that

$$\|(f, g)\| \approx \sqrt{q}$$

- Complete the basis: *unimodularity problem:* Euclid+geometry

Achieve good quality

Sample (f, g)'s until:

- `Falcon`: $\max(\|\widetilde{\mathbf{b}}_1\|, \|\widetilde{\mathbf{b}}_{d+1}\|) \approx 1.17\sqrt{q}$
- `Hybrid`: $s_1(\widetilde{\mathbf{B}})$ as close as possible to $\sqrt{q}$

Both metrics can be computed ***just with*** f, g

(naive) **KeyGen**:

1) **Do**
   $f, g \leftarrow D_{\mathbb{Z}^d, \sqrt{\frac{q}{2d}}}$
   Until f inv. mod q **And** $\|f, g\| \leqslant 1.17\sqrt{q}$;
2) *(F) quality check:* $\|\widetilde{\mathbf{b}}_{d+1}\| \leqslant 1.17\sqrt{q}$?
   else restart;
4) $\mathbf{b}_{d+1} \leftarrow \mathrm{NTRUSolve}(f, g, q)$;
   Compute all needed data;
   Output (pk, sk).

(naive) **KeyGen**:

1) **Do**
   $f, g \leftarrow D_{\mathbb{Z}^d, \sqrt{\frac{q}{2d}}}$
   Until f inv. mod q **And** $\|f, g\| \leqslant 1.17\sqrt{q}$;
2) *(F) quality check:* $\|\widetilde{\mathbf{b}}_{d+1}\| \leqslant 1.17\sqrt{q}$?
   else restart;
4) $\mathbf{b}_{d+1} \leftarrow \text{NTRUSolve}(f, g, q)$;
   Compute all needed data;
   Output (pk, sk).

(naive) **KeyGen**:

1) **Do**
   $f, g \leftarrow D_{\mathbb{Z}^d, \sqrt{\frac{q}{2d}}}$
   Until f inv. mod q **And** $\|f, g\| \leqslant 1.17\sqrt{q}$;

2) *(F) quality check:* $\|\widetilde{\mathbf{b}}_{d+1}\| \leqslant 1.17\sqrt{q}$?
   else restart;

2-bis) *(M) quality check:* $s_1(\widetilde{\mathbf{B}}) \leqslant 2.05\sqrt{q}$?
   else restart;

4) $\mathbf{b}_{d+1} \leftarrow \mathrm{NTRUSolve}(f, g, q)$;
   Compute all needed data;
   Output (pk, sk).

(naive) **KeyGen**:

1) **Do**
   $f, g \leftarrow D_{\mathbb{Z}^d, \sqrt{\frac{q}{2d}}}$
   Until f inv. mod q **And** $\|f, g\| \leqslant 1.17\sqrt{q}$;

2) *(F) quality check:* $\|\widetilde{\mathbf{b}}_{d+1}\| \leqslant 1.17\sqrt{q}$?
   else restart;

2-bis) *(M) quality check:* $s_1(\widetilde{\mathbf{B}}) \leqslant 2.05\sqrt{q}$?
   else restart;

4) $\mathbf{b}_{d+1} \leftarrow \mathsf{NTRUSolve}(f, g, q)$;
   Compute all needed data;
   Output (pk, sk).

- This already happens often in `Falcon`
- Need ***a lot*** of tries to reach 2.05
- **And randomness is expensive.**

(naive) **KeyGen**:

1) **Do**
   $f, g \leftarrow D_{\mathbb{Z}^d, \sqrt{\frac{q}{2d}}}$
   Until f inv. mod q **And** $\|f, g\| \leqslant 1.17\sqrt{q}$;

2) *(F) quality check:* $\|\widetilde{\mathbf{b}}_{d+1}\| \leqslant 1.17\sqrt{q}$?
   else restart;

2-bis) *(M) quality check:* $s_1(\widetilde{\mathbf{B}}) \leqslant 2.05\sqrt{q}$?
   else restart;

4) $\mathbf{b}_{d+1} \leftarrow \mathrm{NTRUSolve}(f, g, q)$;
   Compute all needed data;
   Output (pk, sk).

Our solution

+ Reuse randomness
+ Galois automorphisms

= *"Free"* blow-up of search-space

☺ better trapdoors in reasonable time

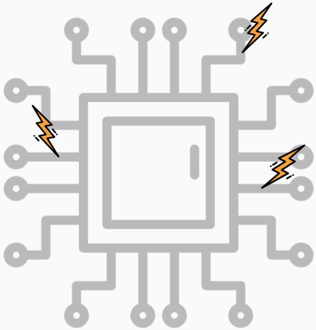
Masking Mitaka

t-probing attacker model [ISW03]

- Adversary obtains t intermediate values of the computation
- Successfully models practical **noisy side-channel leakage** [DDF14]

Provable security: t-probing security

- Any set of at most t intermediate variables is independent of the secret.



t-probing attacker model [ISW03]

- Adversary obtains t intermediate values of the computation
- Successfully models practical **noisy side-channel leakage** [DDF14]

Provable security: t-probing security

- Any set of at most t intermediate variables is independent of the secret.

Mitaka is maskable !

- Protect the whole scheme using *arithmetic masking*
- Standard multiplier + FFT → *pointwise multiplication*.
- Gaussian generation:
    - Generate arithmetic shares of gaussians [Offline]
    - Sum of them is a Gaussian ! [Online]

Implementation results

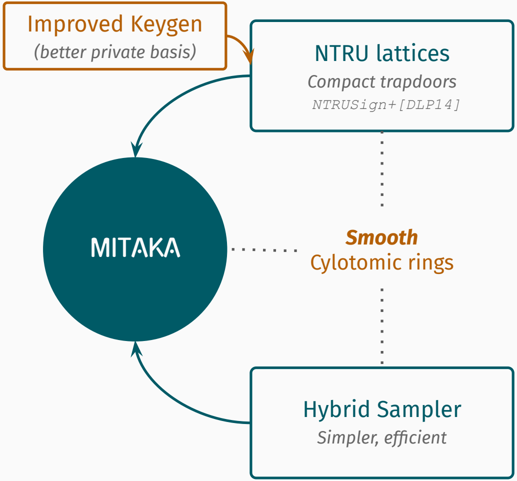
About performances (signatures / sec.)

	Falcon	Mitaka	Ratio
$d = 512$	2800	6300	**2.25**
$d = 1024$	1400	3100	**2.21**

experiments done with a ***non-masked & non constant-time*** *implementation*$^{(*)}$ *and reusing Falcon's C reference code (as submitted to NIST round 3)*

$(*)$: both schemes can be made constant-time with [BBEF+19], [ZSS'20], [HPRR'20]

Wrapping-up



Simple | Efficient | Compact | Versatile | Maskable