PROMETHEUS

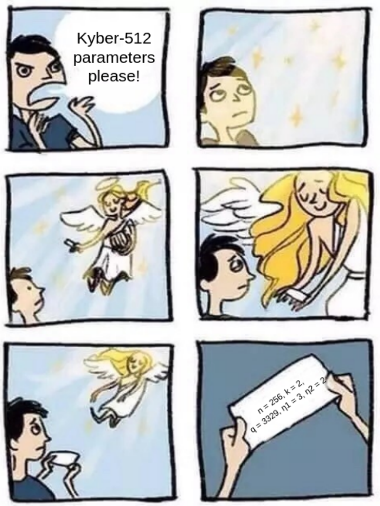
PRivacy preserving pOst-quantuM systEms from advanced crypTograpHic mEchanisms Using latticeS

Tools for Parameter Selection

Eamonn Postlethwaite, CWI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780701.

Parameters are ‘god given’



Parameters are ~~‘god given’~~ a human work

This presentation concerns the tools that have been {created, expanded, maintained} under the PROMETHEUS project – what they do, how they work, and how they have been used.

Specifically:

- the lattice-estimator [APS15] (`github:malb/lattice-estimator`),
- the leaky-LWE-estimator [DDGR20] (`github:lducas/leaky-LWE-Estimator`),
- NTRUFatigue-estimator [DvW21] (`github:WvanWoerden/NTRUFatigue`).

The high level *what*

Broadly speaking, any (lattice) estimator takes as input a subset of

$$\{parameters\} \times \{attacks\} \times \{attack\ models\},$$

and outputs estimates for the costs of the attacks. For example

$$\{\text{Kyber512}\} \times \{\text{primal uSVP}\} \times \{\text{cost: ADPS16, shape: GSA}\}.$$

The high level *what*

Different estimators aim for different levels of generality:

- lattice-estimator $\leftrightarrow$ high generality (many parameters, attacks, models),
- leaky-LWE-estimator $\leftrightarrow$ high specificity (some parameters, one attack in detail),[1]
- NTRUFatigue-estimator $\leftrightarrow$ high specificity (some parameters, one attack in detail).

1 Creates a flexible framework for introducing side channel information into lattice reduction, we will mostly discuss its model for the primal uSVP attack.

The high level *what*

Different estimators aim for different levels of generality:

- lattice-estimator $\leftrightarrow$ high generality (many parameters, attacks, models),
- leaky-LWE-estimator $\leftrightarrow$ high specificity (some parameters, one attack in detail),[1]
- NTRUFatigue-estimator $\leftrightarrow$ high specificity (some parameters, one attack in detail).

Ultimately, the aim is to integrate all impactful attacks into the lattice-estimator.

1 Creates a flexible framework for introducing side channel information into lattice reduction, we will mostly discuss its model for the primal uSVP attack.

The lattice-estimator

An implementation that came out of a paper [APS15] that systematised and improved the analysis of a large range of attacks against LWE.

Prometheus members are involved in its maintenance and improvement (e.g. recent MATZOV [MAT22] cost model).

The lattice-estimator

An implementation that came out of a paper [APS15] that systematised and improved the analysis of a large range of attacks against LWE.

Prometheus members are involved in its maintenance and improvement (e.g. recent MATZOV [MAT22] cost model).

The aim is to capture as many attacks and improvements as possible and automate their cost estimation against a wide range of popular parameters, e.g. dimension, modulus, secret and error distributions etc...

How to use: the lattice-estimator

Create parameter object.

```
sage: from estimator import *
....: params = LWE.Parameters(n=200, q=7981, Xs=ND.SparseTernary(384, 16), Xe=ND.CenteredBinomial(4))
```

Estimate suite of attacks given:

- a model for lattice reduction *cost*,
- and a model for lattice reduction *shape*.

How to use: the lattice-estimator

Then

```
sage: r = LWE.estimate(params, red_cost_model=RC.MATZOV, red_shape_model=Simulator.GSA)
```

gives

```
arora-gb             :: rop: ≈2^102.2, dreg: 9, mem: ≈2^101.2, t: 4, m: ≈2^51.6, tag: arora-gb, ↻: 2, ζ: 1, |S|: 1, prop: 1
bkw                  :: rop: ≈2^52.0, m: ≈2^41.6, mem: ≈2^42.5, b: 3, t1: 0, t2: 12, ℓ: 2, #cod: 155, #top: 1, #test: 44, tag: coded-bkw
usvp                 :: rop: ≈2^50.4, red: ≈2^50.4, δ: 1.010720, β: 72, d: 331, tag: usvp
bdd                  :: rop: ≈2^47.9, red: ≈2^47.1, svp: ≈2^46.5, β: 60, η: 87, d: 345, tag: bdd
bdd_hybrid           :: rop: ≈2^46.3, red: ≈2^46.2, svp: ≈2^41.7, β: 40, η: 22, ζ: 51, |S|: ≈2^12.3, d: 324, prob: 0.141, ↻: 31, tag: hybrid
bdd_mitm_hybrid      :: rop: ≈2^48.2, red: ≈2^47.7, svp: ≈2^46.4, β: 40, η: 2, ζ: 114, |S|: ≈2^47.1, d: 261, prob: 0.041, ↻: 111, tag: hybrid
dual                 :: rop: ≈2^51.9, mem: ≈2^18.8, m: 174, β: 73, d: 374, ↻: 1, tag: dual
dual_hybrid          :: rop: ≈2^46.6, mem: ≈2^35.5, m: 145, β: 40, d: 287, ↻: 21, ζ: 58, h1: 4, tag: dual_hybrid
```

How to use: the lattice-estimator

One can also specific a particular attack and its options

```
sage: LWE.primal_hybrid(params, mitm=False, babai=False, red_shape_model=Simulator.CN11)
```

to obtain

```
rop: ≈2^46.6, red: ≈2^45.6, svp: ≈2^45.6, β: 42, η: 22, ζ: 52, |S|: ≈2^17.5, d: 330, prob: 0.308, ↻: 13, tag: hybrid
```

Most importantly, read the docs!
https://lattice-estimator.readthedocs.io/en/latest/.

How to use: the lattice-estimator

One can also specific a particular attack and its options

```
sage: LWE.primal_hybrid(params, mitm=False, babai=False, red_shape_model=Simulator.CN11)
```

to obtain

```
rop: ≈2^46.6, red: ≈2^45.6, svp: ≈2^45.6, β: 42, η: 22, ζ: 52, |S|: ≈2^17.5, d: 330, prob: 0.308, ↻: 13, tag: hybrid
```

Most importantly, *write* the docs!
`https://lattice-estimator.readthedocs.io/en/latest/`.

How it works: the lattice-estimator

It has a highly modularised codebase

- top level Estimate class $\leftrightarrow$ lwe.py,
- initiating and transforming LWE parameters $\leftrightarrow$ lwe_parameters.py,
- each attack has its own file, with a class per variant $\leftrightarrow$ e.g. lwe_primal.py with class PrimalHybrid,
- reduction cost models $\leftrightarrow$ reduction.py,
- reduction shape models $\leftrightarrow$ simulator.py,
- error and secret (‘noise’) distributions $\leftrightarrow$ nd.py,
- several more, probability amplification, distinguishing, mitm...

Ultimately, the estimator uses a local minimum finder to minimise the cost of a given attack in a given model against given parameters.

Some useful notes...

It *does not* take structure into account; it maps straight from modules to integers.

Some useful notes...

It *does not* take structure into account; it maps straight from modules to integers.

It *does not* currently include the probabilistic uSVP techniques examined in [DDGR20, PV21] or the overstretched estimator of [DvW21].

Some useful notes...

It *does not* take structure into account; it maps straight from modules to integers.

It *does not* currently include the probabilistic uSVP techniques examined in [DDGR20, PV21] or the overstretched estimator of [DvW21].

It *does not* currently include any SIS estimation.

Some useful notes...

It *does not* take structure into account; it maps straight from modules to integers.

It *does not* currently include the probabilistic uSVP techniques examined in [DDGR20, PV21] or the overstretched estimator of [DvW21].

It *does not* currently include any SIS estimation.

It *does* contain many TODOs and (almost certainly) some bugs, please contribute or perform code review!

Some useful notes...

It *does not* take structure into account; it maps straight from modules to integers.

It *does not* currently include the probabilistic uSVP techniques examined in [DDGR20, PV21] or the overstretched estimator of [DvW21].

It *does not* currently include any SIS estimation.

It *does* contain many TODOs and (almost certainly) some bugs, please contribute or perform code review!

We *do* have a good understanding of which β succeed for different attacks, what is challenging is converting this into a more expressive cost.

The leaky-LWE-estimator

So called because its main function is the integration of leaks (i.e. side channel information) into the primal uSVP attack.

As a subtask improved our understanding of probabilistic aspects of this attack – in short consider the probability distributions of projections of the embedded vector, not their mean lengths.

The leaky-LWE-estimator

So called because its main function is the integration of leaks (i.e. side channel information) into the primal uSVP attack.

As a subtask improved our understanding of probabilistic aspects of this attack – in short consider the probability distributions of projections of the embedded vector, not their mean lengths.

There are excellent tutorials in [DDGR20, App. A] for the side channel aspects, I will focus on the probabilistic uSVP estimator.

How to use: the leaky-lwe-estimator

Set up parameters

```
sage: load("framework/instance_gen.sage")
sage: n, q, m = 640, 2**15, 640 + 16 # FRODO-640
sage: frodo_distribution = [9288, 8720, 7216, 5264, 3384, 1918, 958, 422, 164, 56, 17, 4, 1]
sage: D_s = get_distribution_from_table(frodo_distribution, 2 ** 16)
sage: D_e = D_s
```

Initialise the instance as inst

```
sage: _, _, inst = initialize_from_LWE_instance(DBDD_predict_diag, n, q, m, D_e, D_s)
```

How to use: the leaky-lwe-estimator

Integrate any short vector hints (we know 'q' vectors will be in our lattice)

```
sage: inst.integrate_q_vectors(q, report_every=20)
          Integrating q-vectors
 [...20]   integrate short vector hint             Unworthy hint, Rejected.
```

Perform estimation *without* probabilism or simulation (e.g. Simulator.GSA of lattice-estimator)

```
sage: inst.estimate_attack()
          Attack Estimation
 dim=1297          δ=1.003474         β=485.84

(485.8376072943104, 1.00347440808773)
```

How to use: the leaky-lwe-estimator

Perform estimation *with* probabilism and simulation

```
sage: inst.estimate_attack(probabilistic=True, ignore_lift_proba=True, lift_union_bound=True)
```

Accumulates probabilities over progressive BKZ tours, and gives

```
β= 495,   pr=1.4798e-01,        cum-pr=3.5544e-01        rem-pr=6.4456e-01
β= 496,   pr=2.0675e-01,        cum-pr=4.8870e-01        rem-pr=5.1130e-01
β= 497,   pr=2.7776e-01,        cum-pr=6.3072e-01        rem-pr=3.6928e-01
β= 498,   pr=3.5929e-01,        cum-pr=7.6340e-01        rem-pr=2.3660e-01
β= 499,   pr=4.4815e-01,        cum-pr=8.6943e-01        rem-pr=1.3057e-01
β= 500,   pr=5.4007e-01,        cum-pr=9.3995e-01        rem-pr=6.0054e-02
β= 501,   pr=6.3022e-01,        cum-pr=9.7779e-01        rem-pr=2.2207e-02
β= 502,   pr=7.1402e-01,        cum-pr=9.9365e-01        rem-pr=6.3506e-03
β= 503,   pr=7.8780e-01,        cum-pr=9.9865e-01        rem-pr=1.3476e-03
β= 504,   pr=8.4928e-01,        cum-pr=9.9980e-01        rem-pr=2.0311e-04
       Attack Estimation
 dim=1297                       β=496.36
```

The NTRUFatigue-estimator

A line of works [ABD16, CJL16, KF17] showed important security implications of setting the modulus q too large in structured lattice schemes. Initial results were asymptotic and experiments focussed on low power lattice reduction and *large* q.

The NTRUFatigue-estimator

A line of works [ABD16, CJL16, KF17] showed important security implications of setting the modulus q too large in structured lattice schemes. Initial results were asymptotic and experiments focussed on low power lattice reduction and *large* q.

This work makes huge strides towards concretising the size of q required for these attacks to function, and understanding precisely how lattice reduction functions on such instances.

The NTRUFatigue-estimator

A line of works [ABD16, CJL16, KF17] showed important security implications of setting the modulus q too large in structured lattice schemes. Initial results were asymptotic and experiments focussed on low power lattice reduction and *large* q.

This work makes huge strides towards concretising the size of q required for these attacks to function, and understanding precisely how lattice reduction functions on such instances.

As part of the software contribution the authors give a probabilistic estimator that determines an average blocksize β where the first of two potential cryptanalytic events (DSD and SKR) will occur.

How to use: the NTRUFatigue-estimator

Load the estimator and set up the instance

```
sage: load("estimator.sage")
sage: q, n, var, ntru, tours = 2**16, 512, 2./3., "matrix", 8
```

How to use: the NTRUFatigue-estimator

Load the estimator and set up the instance

```
sage: load("estimator.sage")
sage: q, n, var, ntru, tours = 2**16, 512, 2./3., "matrix", 8
```

Run the estimator

```
sage: beta, SKR, DSD, position = combined_attack_prob(q, n, sk_variance=var,
 ....:                                                ntru=ntru,
 ....:                                                fixed_tours=tours)
```

How to use: the NTRUFatigue-estimator

Inspect the results

```
sage: beta
194.5987821425533
sage: SKR
-2.049973797868008e-11
sage: DSD
0.9997352694931326
```

How to use: the NTRUFatigue-estimator

Inspect the results

```
sage: beta
194.59877821425533
sage: SKR
-2.0499737977868008e-11
sage: DSD
0.9997352694931326
```

Note that if we use a smaller $q = 12289$, we receive

```
sage: beta
285.85405500098994
sage: SKR
0.9998508564732661
sage: DSD
0.0
```

Use cases

A non exhaustive list of how these estimators have been used:

- NIST candidates: KYBER, SABER, FRODO, DILITHIUM,
- PROMETHEUS research: to estimate the security of signatures based on new Gaussian samplers over modules [BEP+21],
- non PROMETHEUS research: to estimate LSH based improvements to MitM attacks on ternary LWE [KM21], in FHE standardisation [ACC+18].

Conclusion

When are these tools for you? Whenever any subset of the following apply:

- you have designed a scheme based on LWE or NTRU,
- you want to understand how lattice attacks behave against concrete parameter choices for it,
- you want to know which attacks to consider (non automatically) in more detail,
- you want to understand the (lattice reduction based) implications of certain kinds of side channels,
- you want to rule out attacks against too large moduli.

Martin R. Albrecht, Shi Bai, and Léo Ducas.
A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes.
In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016.

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan.
Homomorphic encryption security standard.
Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

Martin R. Albrecht, Rachel Player, and Sam Scott.
On the concrete hardness of learning with errors.
Journal of Mathematical Cryptology, 9(3):169–203, 2015.

Pauline Bert, Gautier Eberhart, Lucas Prabel, Adeline Roux-Langlois, and Mohamed Sabt.

Implementation of lattice trapdoors on modules and applications.
In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 195–214. Springer, Heidelberg, 2021.

Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee.
An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero.
LMS Journal of Computation and Mathematics, 19(A):255–266, 2016.

Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi.
LWE with side information: Attacks and concrete security estimation.
In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2020.

Léo Ducas and Wessel P. J. van Woerden.
NTRU fatigue: How stretched is overstretched?
In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 3–32. Springer, Heidelberg, December 2021.

Paul Kirchner and Pierre-Alain Fouque.
Revisiting lattice attacks on overstretched NTRU parameters.
In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Heidelberg, April / May 2017.

Elena Kirshanova and Alexander May.
How to find ternary lwe keys using locality sensitive hashing.
In *Cryptography and Coding: 18th IMA International Conference, IMACC 2021, Virtual Event, December 14–15, 2021, Proceedings*, pages 247–264, Berlin, Heidelberg, 2021. Springer-Verlag.

MATZOV.
Report on the Security of LWE: Improved Dual Lattice Attack, April 2022.

Eamonn W. Postlethwaite and Fernando Virdia.
On the success probability of solving unique SVP via BKZ.
In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 68–98. Springer, Heidelberg, May 2021.