Precision Tool for FHE Parameter Selection

FHE.org 2025

25 of March 2025

Beatrice Biasioli, Elena Kirshanova, Chiara Marcolla, Sergi Rovira
beatrice.biasioli@ibm.com
elena.kirshanova@tii.ae
chiara.marcolla@tii.ae
sergi.rovira@tii.ae

Contents

1 From theory . . .

- Relation between LWE Parameters
- Current Methods for Choosing FHE Parameters
- Our Contribution: A Formula-Based Approach

2 . . . to practice

- Reversed Formulas
- Generalization with Numerical Methods
- A Practical Solution: Our Tool for Secure Parameter Selection

From theory . . .

What are the parameters of an FHE scheme?

Almost all FHE schemes used today are based on the Learning with Errors (LWE) problem (or its algebraic variant):

Definition (*Search* LWE Problem (Regev))

Given $\mathbf{b} \in \mathbb{Z}_q^m$ and $A \in (\mathbb{Z}_q)^{m \times n}$, find an *unknown* vector $\mathbf{s} \in \mathbb{Z}_q^n$ such that

$$A\,\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q$$

where $\mathbf{e} \in \mathbb{Z}_q^m$ is *small* random error.

What are the parameters of an FHE scheme?

Almost all FHE schemes used today are based on the Learning with Errors (LWE) problem (or its algebraic variant):

Definition (*Search* LWE Problem (Regev))

Given $\mathbf{b} \in \mathbb{Z}_q^m$ and $A \in (\mathbb{Z}_q)^{m \times n}$, find an *unknown* vector $\mathbf{s} \in \mathbb{Z}_q^n$ such that

$$A\,\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q$$

where $\mathbf{e} \in \mathbb{Z}_q^m$ is *small* random error.

- $\mathbf{s}$ follows the distribution χ_s with standard deviation σ_s.
- $\mathbf{e}$ follows the distribution χ_e with standard deviation σ_e.
- Modulus q and the LWE dimension n.

What are the parameters of an FHE scheme?

Almost all FHE schemes used today are based on the Learning with Errors (LWE) problem (or its algebraic variant):

Definition (*Search* LWE Problem (Regev))

Given $\mathbf{b} \in \mathbb{Z}_q^m$ and $A \in (\mathbb{Z}_q)^{m \times n}$, find an *unknown* vector $\mathbf{s} \in \mathbb{Z}_q^n$ such that

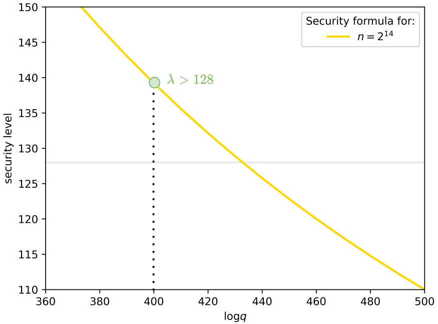
$$A\,\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q$$

where $\mathbf{e} \in \mathbb{Z}_q^m$ is *small* random error.

- $\mathbf{s}$ follows the distribution χ_s with standard deviation σ_s.
- $\mathbf{e}$ follows the distribution χ_e with standard deviation σ_e.
- Modulus q and the LWE dimension n.
- Security level λ.

What a complex life! (aka why choosing secure parameters is hard?)

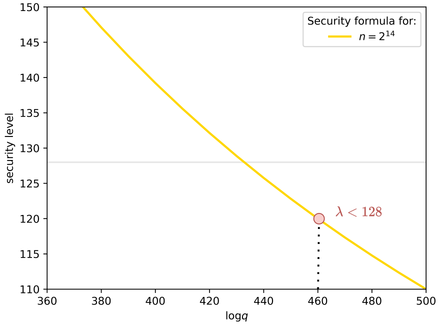
- To guarantee correctness , we need a large enough modulus q.



What a complex life! (aka why choosing secure parameters is hard?)

- To guarantee correctness , we need a large enough modulus q.
- Security problem: larger q decreases the security.

\# operations ↑ q ↑ λ ↓



What a complex life! (aka why choosing secure parameters is hard?)

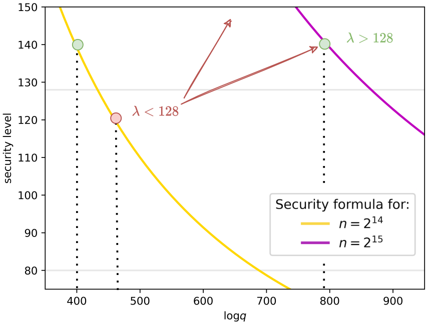
- To guarantee correctness , we need a large enough modulus q.
- Security problem: larger q decreases the security.

  # operations $\uparrow$ $\quad$ $q \uparrow$ $\quad$ $\lambda \downarrow$

- Efficiency problem: To increase the security level λ again, we need a larger dimension n.

  # operations $\uparrow$ $\quad$ $q \uparrow$ $\quad$ λ = $n \uparrow$

Performance



How are FHE Parameters Selected Today?

1. Lattice Estimator.

The Lattice Estimator (`https://github.com/malb/lattice-estimator`) [1] is a powerful software tool that provides the running time of (almost) any LWE attack.

```
sage: from estimator import *
....: sig=3.19
....:
....: Xs=ND.UniformMod(3)
....: Xe=ND.DiscreteGaussian(sig)
....: n = 2^15
....: q =2**881
....:
....: params = LWE.Parameters(n=n, q=q, Xs=Xs, Xe=Xe)
....:
....: LWE.primal_usvp(params)
....: LWE.primal_bdd(params)
....: LWE.dual(params)
....:
rop: ≈2^126.1, red: ≈2^126.1, δ: 1.004657, β: 318, d: 63223, tag: usvp
rop: ≈2^126.5, red: ≈2^125.9, svp: ≈2^124.9, β: 317, η: 371, d: 65241, tag: bdd
rop: ≈2^127.2, mem: ≈2^80.9, m: ≈2^15.0, β: 318, d: 65551, ʋ: 1, tag: dual
```

[1] Albrecht, Player, Scott - *On the concrete hardness of learning with errors* (2015)

How are FHE Parameters Selected Today?

Parameters for FHE scheme: $\lambda, n, q, \sigma_s, \sigma_e$

	Target parameter	Flexible	Easy to integrate with existing FHE libraries	Fast
Lattice Estimator [1]	λ	✓	✗	✗

[1] Albrecht, Player, Scott - *On the concrete hardness of learning with errors* (2015)

How are FHE Parameters Selected Today?

2. Precomputed tables. The Lattice Estimator has been adopted in [2] and [3] to provide specific tables for FHE parameters.

Homomorphic Encryption Standard

n	$\log_2(q)$ (Classical)		$\log_2(q)$ (Qu	
	Ternary	Gaussian	Ternary	
$\lambda = 128$				
1024	26	29	25	
2048	54	56	50	
4096	108	110	101	
8192	217	219	203	
16384	438	439	409	
32768	881	883	825	
65536	1776	1778	1663	
131072	3576	3578	3348	
$\lambda = 192$				
2048	37	39	34	36

distribution	n	security level	logq	uSVP	dec	dual
uniform	1024	128	29	131.2	145.9	161.0
		192	21	192.5	225.3	247.2
		256	16	265.8	332.6	356.7
	2048	128	56	129.8	137.9	148.2
		192	39	197.6	217.5	233.7
		256	31	258.6	294.3	314.5
	4096	128	111	128.2	132.0	139.5
		192	77	194.7	205.5	216.4
		256	60	260.4	280.4	295.1
	8192	128	220	128.5	130.1	136.3
		192	154	192.2	197.5	205.3
		256	120	256.5	267.3	277.5
	16384	128	440	128.1	129.0	133.9
		192	307	192.1	194.7	201.0
		256	239	256.6	261.6	269.3
	32768	128	880	128.8	129.1	133.6
		192	612	193.0	193.9	198.2
		256	478	256.4	258.8	265.1

Update version

[2] Albrecht et al. – *Homomorphic Encryption Security Standard*, `HomomorphicEncryption.org`, (2018)
[3] Bossuat, et al. - *Security Guidelines for Implementing Homomorphic Encryption* (2024)

How are FHE Parameters Selected Today?

Parameters for FHE scheme: $\lambda, n, q, \sigma_s, \sigma_e$

	Target parameters	Flexible	Easy to integrate with existing FHE libraries	Fast
Lattice Estimator [1]	λ	✓	✗	✗
Precomputed Tables [2,3]	λ, n, q, σ_e	✗	✓	✓

[1] Albrecht, Player, Scott - *On the concrete hardness of learning with errors* (2015)

[2] Albrecht, et al. - *Homomorphic Encryption Security Standard* (2018)

[3] Bossuat, et al. - *Security Guidelines for Implementing Homomorphic Encryption* (2024)

How are FHE Parameters Selected Today?

Parameters for FHE scheme: $\lambda, n, q, \sigma_s, \sigma_e$

	Target parameters	Flexible	Easy to integrate with existing FHE libraries	Fast
Lattice Estimator [1]	λ	✓	✗	✗
Precomputed Tables [2,3]	λ, n, q, σ_e	✗	✓	✓
Our work: Formula-based [4,5]	λ, n, q, σ_e	✓	✓	✓

[1] Albrecht, Player, Scott - *On the concrete hardness of learning with errors* (2015)
[2] Albrecht, et al. - *Homomorphic Encryption Security Standard* (2018)
[3] Bossuat, et al. - *Security Guidelines for Implementing Homomorphic Encryption* (2024)
[4] Kirshanova, Marcolla, Rovira - *Guidance for Efficient Selection of Secure Parameters for Fully Homomorphic Encryption* (2024)
[5] Biasioli, Kirshanova, Marcolla, Rovira - *A Tool for Fast and Secure LWE Parameter Selection: the FHE case* (2025)

Security of schemes based on (R)LWE problems

- The security of FHE schemes depends on the intractability of the LWE problem.
- Attacks on FHE schemes are based on finding efficient algorithms to solve lattice problems.

(Some) Hard problems on lattices

- ⋆ The Shortest Vector Problem (SVP) asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L}) = \min\{\|\mathbf{w}\| \,:\, \mathbf{w} \in \mathcal{L},\, \mathbf{w} \neq 0\}$.
- ⋆ The Unique Shortest Vector Problem (uSVP) asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$, which is guaranteed to be at least γ times smaller than $\lambda_2(\mathcal{L})$.
- ⋆ The Bounded Distance Decoding (BDD) problem asks to find $\mathbf{v} \in \mathcal{L}$ closest to the target $\mathbf{t}$ with the promise that $\|\mathbf{t} - \mathbf{v}\| \leq k$, where $k \ll \lambda_1(\mathcal{L})$.

Security of schemes based on (R)LWE problems

- The security of FHE schemes depends on the intractability of the LWE problem.
- Attacks on FHE schemes are based on finding efficient algorithms to solve lattice problems.

(Some) Hard problems on lattices

⋆ The Unique Shortest Vector Problem (uSVP) asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$, which is guaranteed to be at least γ times smaller than $\lambda_2(\mathcal{L})$.

⋆ The Bounded Distance Decoding (BDD) problem asks to find $\mathbf{v} \in \mathcal{L}$ closest to the target $\mathbf{t}$ with the promise that $\|\mathbf{t} - \mathbf{v}\| \leq k$, where $k \ll \lambda_1(\mathcal{L})$.

Attacks we have considered

✓ Primal BDD and uSVP attacks are currently the most efficient attacks (for FHE).

✓ Hybrid attacks often outperform others when the secret is sparse.

✗ Dual attacks No complete algorithm has been presented that outperforms primal attacks $\Longrightarrow$ Recently, the correctness of heuristic dual attacks was questioned.

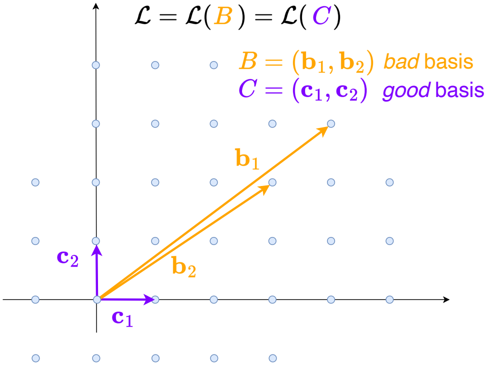
Security of schemes based on (R)LWE problems

- The security of FHE schemes depends on the intractability of the LWE problem.
- Attacks on FHE schemes are based on finding efficient algorithms to solve lattice problems.

The core part of this algorithm is based on a lattice reduction: starting from a *bad* lattice basis, find a *good* basis.

⇩

The BKZ-β (Block-Korkine-Zolotarev) algorithm works by calling multiple times an algorithm for SVP on sublattices of dimension β.



Formula Derivation

1. In any lattice-based attack, the main role is played by the lattice reduction parameter β $\implies$ determine the *optimal* $\beta = f(n, q, \sigma_e, \sigma_s)$.

2. The security parameter λ relates to β via the core-SVP model:

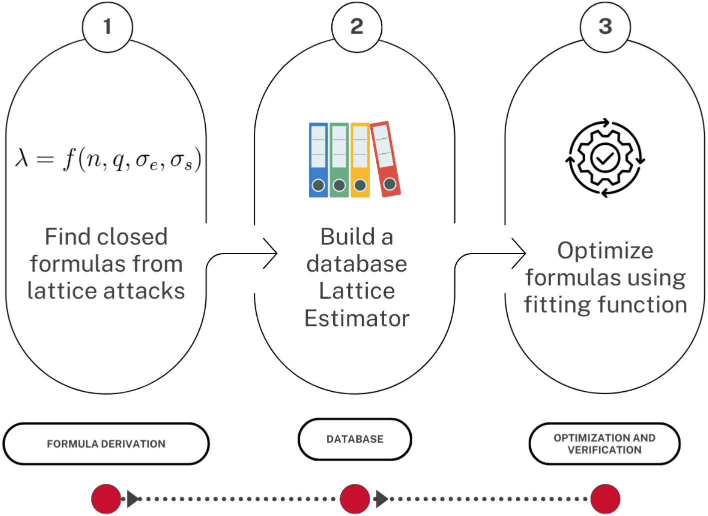
$$T_{\mathsf{BKZ}}(\beta, d) = 8d \cdot 2^{0.292\beta + 16.4} \implies \lambda = 0.292\beta + \log_2(8d) + 16.4,$$

   where d is the lattice dimension.

3. The previous expression can be reverted for any desired parameter, e.g.

$$n = g(\lambda, q, \sigma_e, \sigma_s).$$

Our Proposal – a Formula-based Approach



Optimization – uSVP

1. We find the *optimal* β for the uSVP attack (where $\zeta = \sigma_e/\sigma_s$):

$$\beta = \frac{2n\ln(q/\zeta)\ln\left(\frac{n\ln(n/\ln q)}{2\pi e\ln(q/\sigma_e)}\right)}{\ln^2\left(\frac{q\sqrt{n\ln(n/\ln(q/\sigma_e))/\ln q}}{2\pi e\sigma_e}\right)}$$

2. Given β, we obtain an expression for λ:

$$\lambda = 0.292\beta + \log_2\left(8\sqrt{\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4$$

Optimization – uSVP

1. We find the *optimal* β for the uSVP attack.
2. Given β, we obtain an expression for λ.
3. We built our database.

χ_s	Range of n	Range of $\log q$	σ_e	# points
$\mathcal{U}_3$	$[2^{10}, 2^{15}]$	$[10, 1600]$	3.19	5282
$\mathcal{U}_2$	$[2^{10}, 2^{11}]$	$[20, 64]$	3.19	42962

Optimization – uSVP

1. We find the *optimal* β for the uSVP attack.
2. Given β, we obtain an expression for λ.
3. We built our database.

χ_s	Range of n	Range of $\log q$	σ_e	# points
$\mathcal{U}_3$	$[2^{10}, 2^{15}]$	$[10, 1600]$	3.19	5282

For BGV/BFV/CKKS :

⋆ The secret $\mathbf{s}$ follows

Ternary distribution $\chi_s = \mathcal{U}_3$: Uniform over the ternary set $\{\pm 1, 0\}$, thus $\sigma_s = \sqrt{2/3}$.
Sparse distribution $\chi_s = \mathcal{HWT}(h)$ chooses a vector uniformly at random from $\{0, \pm 1\}^n$ with exactly h nonzero entries, where $h \leq n$ positive integer, thus $\sigma_s = \sqrt{h/n}$.

⋆ The error $\mathbf{e}$ follows the discrete Gaussian distribution $\chi_e = \mathcal{DG}(0, \sigma_e^2)$ with $\sigma_e = 3.19$.

⋆ The LWE dimension $n \in \{2^{10}, \ldots, 2^{16}\}$ and modulus $\log q \in \{20, \ldots, 2900\}$ depending on n and λ:
e.g. for $80 \leq \lambda \leq 192$ and $n = 2^{10} \implies 20 \leq \log q \leq 45$ or $n = 2^{16} \implies 1230 \leq \log q \leq 2900$.

Optimization – uSVP

1. We find the *optimal* β for the uSVP attack.
2. Given β, we obtain an expression for λ.
3. We built our database.

χ_s	Range of n	Range of $\log q$	σ_e	# points
$\mathcal{U}_3$	$[2^{10}, 2^{15}]$	$[10, 1600]$	3.19	5282
$\mathcal{U}_2$	$[2^{10}, 2^{11}]$	$[20, 64]$	3.19	42962

For DM/CGGI :

- ⋆ The secret $\mathbf{s}$ follows binary distribution $\chi_s = \mathcal{U}_2$: Uniform over the set $\{1, 0\}$, thus $\sigma_s = 1/2$.
- ⋆ The error $\mathbf{e}$ follows the discrete Gaussian distribution $\chi_e = \mathcal{DG}(0, \sigma_e^2)$, the centered at 0 with standard deviation σ_e that varies.
- ⋆ The LWE dimension $n \in \{2^{10}, 2^{11}\}$ and modulus $q \in \{2^{32}, 2^{64}\}$.

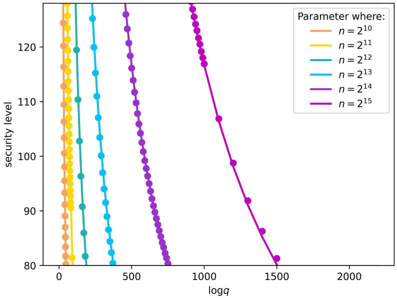
Optimization – uSVP

1. We find the *optimal* β for the uSVP attack.
2. Given β, we obtain an expression for λ.
3. We built our database.
4. We model our approximation formula with coupled optimization, finding `lambda`

$$\lambda = A\beta + B \ln\left(\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}\right) + C,$$

$A = 0.317747 \quad B = 2.071129 \quad C = 1.849214$ if $\chi_s = \mathcal{U}_2$

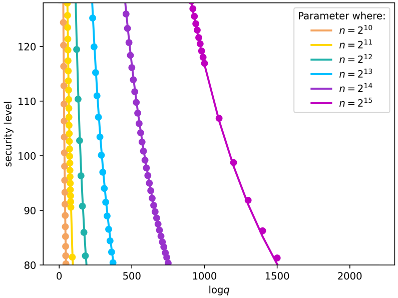
$A = 0.296208 \quad B = 0.800603 \quad C = 12.09086$ if $\chi_s = \mathcal{U}_3$.



Optimization – uSVP

1. We find the *optimal* β for the uSVP attack.
2. Given β, we obtain an expression for λ.
3. We built our database.
4. We model our approximation formula with coupled optimization, finding
5. We approximate our formula `lambda`:

$$\lambda \approx \tilde{A} \ln\left(\frac{\tilde{B} n}{\ln q}\right) \frac{n}{\ln q} + \tilde{C} \ln n + \tilde{D}$$



We model it with coupled optimization, finding

$$\tilde{A} = 0.445309 \quad \tilde{B} = 1.486982 \quad \tilde{C} = 0.950115 \quad \tilde{D} = 11.21416 \quad \text{if } \chi_s = \mathcal{U}_2$$
$$\tilde{A} = 0.833542 \quad \tilde{B} = 0.154947 \quad \tilde{C} = 1.469823 \quad \tilde{D} = 18.09877 \quad \text{if } \chi_s = \mathcal{U}_3.$$

Security formula for $\chi_s = \mathcal{U}_2$ and $\sigma_e = 3.19$

$n = 2^{10}$				$n = 2^{11}$			
$\log q$	Estimator	`lambda`	`lambda_s`	$\log q$	Estimator	`lambda`	`lambda_s`
20	175	178	172	37	193	193	188
24	144	145	142	46	152	152	149
25	139	139	136	50	139	139	136
26	133	133	130	53	130	130	128
27	128	128	125	54	128	128	126
28	123	123	120	57	121	121	119
30	114	114	112	62	110	111	109
33	103	103	101	67	100	102	101
37	92	92	90	74	91	93	91
42	81	81	80	84	80	82	80

. . . to practice

Reverting the formulas

✓ Formulas to compute the security parameter λ from the theoretical analysis of the attacks.

$$\begin{cases} \beta = f(n, q, \sigma_e, \sigma_s) \\ \lambda = g(n, q, \beta) = 0.292\beta + \log(8d) + 16.4 \end{cases}$$

Reverting the formulas

- ✓ Formulas to compute the security parameter λ from the theoretical analysis of the attacks.
- ✓ Formulas to compute the lattice dimension n .
- ✓ Formulas to compute the size of the modulus $\log q$.
- ✓ Formulas to compute the standard deviation of the error distribution σ_e .

$$\begin{cases} \beta = f(n, q, \sigma_e, \sigma_s) \\ \lambda = 0.292\beta + \log(8d) + 16.4 \end{cases}$$

Optimizing our Formulas: the Fine-Tuning Approach

We optimize the formulas, because we had to approximate some non-leading terms.

Culprit

$$\begin{cases} \beta = \dfrac{2n \ln(q/\zeta) \ln\left(\frac{n \ln(n/\ln q)}{2\pi e \ln(q/\sigma_e)}\right)}{\ln^2\left(\frac{q\sqrt{n \ln(n/\ln(q/\sigma_e))/\ln q}}{2\pi e \sigma_e}\right)} \\ \lambda = 0.292\beta + \log_2\left(8\sqrt{\dfrac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4 \end{cases}$$

Optimizing our Formulas: the Fine-Tuning Approach

We optimize the formulas, because we had to approximate some non-leading terms.

Fine-tuning phase that requires the construction of a database .

- ✓ For BGV, BFV and CKKS, we fix $\sigma_e = 3.19$, which is the standard choice.
- For CGGI-like schemes, we consider the standard combinations of n and $\log q$.

Optimizing our Formulas: Numerical Methods

Employ numerical methods for the resolution of the system of equations

$$\begin{cases} \beta = f(n, q, \sigma_e, \sigma_s) \\ \lambda = g(n, q, \beta) \end{cases}$$

Numerical methods are mathematical tools designed to solve numerical problems .

Pros of fine-tuned formulas :

- Fast
- Numerically-stable
- Easy to integrate in libraries
- Show how do parameters relate

Pros of numerical methods :

- Fast
- Do not require a database
- Allow parameters that have more complicated relations

A Python Wrapper of our Formulas - Find the Security Level λ

```
$ python3 estimate.py --param "lambda" --n "8192" --logq "200-203" --secret "ternary"
Secret dist. | LWE dim. | log q | Output
-------------+----------+-------+--------
ternary      | 8192     | 200   | 141
ternary      | 8192     | 201   | 140
ternary      | 8192     | 202   | 139
ternary      | 8192     | 203   | 139
```

- Can select a range for the parameter $\log q$
- The default error value is 3.19

A Python Wrapper - Comparison with the Lattice Estimator

```
$ python3 estimate.py --param "lambda" --n "8192" --logq "200-203" --secret "ternary" -v
Secret dist. | LWE dim. | log q | Output | lwe
-------------+----------+-------+--------+------
ternary      | 8192     | 200   | 141    | 140
ternary      | 8192     | 201   | 140    | 140
ternary      | 8192     | 202   | 139    | 139
ternary      | 8192     | 203   | 139    | 138
```

A Python Wrapper of our Formulas - Find the Security Level λ

```
$ python3 estimate.py --param "lambda" --n "32768" --logq "870"  --secret "ternary"

secret dist. | lwe dim. | log q | output
-------------+----------+-------+-------
ternary      | 32768    | 870   | 130
```

```
$ python3 estimate.py --param "lambda" --n "32768" --logq "870" --hw "128" --secret "sparse"
secret dist. | lwe dim. | log q | hw  | hybrid
-------------+----------+-------+-----+-------
sparse       | 32768    | 870   | 128 | 121
```

A Python Wrapper of our Formulas - Find the Security Level λ

```
$ python3 estimate.py --param "lambda" --n "1024" --logq "32" --secret "binary" --error "3.19"
Secret dist. | LWE dim. | log q | Output
-------------+----------+-------+--------
ternary      | 1024     | 32    | 106
```

```
$ python3 estimate.py --param "lambda" --n "1024" --logq "64" --secret "binary" --error "10000000"
Secret dist. | LWE dim. | log q | Output
-------------+----------+-------+--------
ternary      | 1024     | 64    | 81
```

A Python Wrapper of our Formulas - Find the Security Level λ

```
$ python3 estimate.py --param "lambda" --n "1024" --logq "32" --secret "binary" --table
Secret dist. | LWE dim. | log q | usvp | usvp_s | bdd | bdd_s | Output
-------------+----------+-------+------+--------+-----+-------+-------
binary       | 1024     | 32    | 106  | 105    | 102 | 105   | 105
```

A Python Wrapper of our Formulas - Find the LWE Dimension n

```
$ python3 estimate.py --param "n" --lambda "128" --logq "200;250;300;350" --secret "ternary"
Secret dist. | lambda | log q | Output | Pow
-------------+--------+-------+---------+------
ternary      | 128    | 200   | 7591    | 8192
ternary      | 128    | 250   | 9479    | 8192
ternary      | 128    | 300   | 11363   | 8192
ternary      | 128    | 350   | 13244   | 16384
```

- Can select different values of the the parameter $\log q$
- Additionally return the closest power of 2

A Python Wrapper - Find $\log q$ and σ_e

```
$ python3 estimate.py --param "logq" --lambda "128" --n "32768" --secret "ternary"
Secret dist. | lambda | LWE dim. | Output
-------------+---------+---------+--------
ternary      | 128     | 32768   | 769
```

```
$ python3 estimate.py --param "std_e" --lambda "128" --n "1024" --logq "32" --secret "binary"
Secret dist. | lambda  | LWE dim. | log q   | Output
-------------+---------+----------+--------+--------
binary       | 128     | 1024     | 32      | 6.985841945781924
```

Conclusions

Motivation:

- Significantly accelerate the parameter selection, maintaining flexibility

Achievements:

- Based on the theoretical analysis of the most efficient lattice attacks on the LWE problem
- Formulas that establish the relations among parameters and are easy to integrate in libraries
- Open-source tool implementing the results

Remark:

- Our analysis is valid for any LWE-based scheme, not restricted to FHE ePrint 2024/1895

GitHub repository

Thank you for your attention!

Cryptography Research Centre

Precision Tool for FHE Parameter Selection

FHE.org CONFERENCE 25.03.2025 SOFIA, BG

FHE.org 2025

25 of March 2025

Beatrice Biasioli, Elena Kirshanova, Chiara Marcolla, Sergi Rovira
beatrice.biasioli@ibm.com
elena.kirshanova@tii.ae
chiara.marcolla@tii.ae
sergi.rovira@tii.ae

Technology Innovation Institute