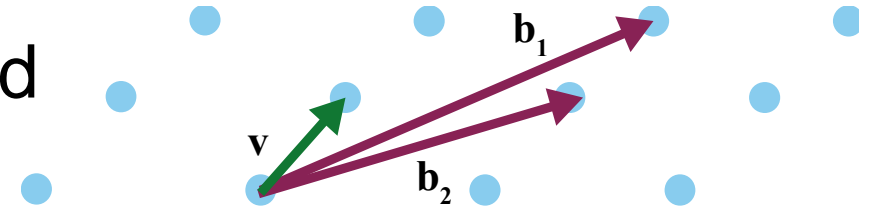


Benchmarking Attacks on Learning with Errors

Emily Wenger (with Eshika Saxena, Mohamed Malhou, Ellie Thieu, &
Kristin Lauter)
NIST PQC Seminar
August 6, 2024

Lattice cryptography: a leading post-quantum candidate

- Lattice cryptography schemes are believed to be quantum and classically secure.



- Lattice schemes rely on the learning with errors (LWE) hardness assumption.
- Two versions of LWE problem:
 - Search LWE: given $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m)$, recover \mathbf{s} .
 - Decision LWE: distinguish (\mathbf{A}, \mathbf{b}) from $(\mathbf{A}, \mathbf{x} \in \mathbb{Z}_q^m)$, where \mathbf{x} is uniform random mod q .

Variants of LWE

- Learning with Errors (LWE):
 - Sample is $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m)$, one line of this is $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$
- Ring Learning with Errors (RLWE):
 - Sample is $(a(x), b(x) = a(x)s(x) + e(x))$ with $a(x), b(x) \in R_q = \mathbb{Z}_q[X]/(X^n + 1)$
 - Define a coefficient embedding $\text{Emb}: R_q \rightarrow \mathbb{Z}_q^n$; $\text{Emb}(a(x)) = (a_0 \ a_1 \ \dots \ a_n)$.
 - Polynomial multiplication = multiplying by Skew-circulant matrix $\mathbf{A} = \text{Skew-Circ}(\text{Emb}(a(x)))$.
 - Used in Homomorphic Encryption
- Module Learning with Errors (MLWE):
 - Sample is (\mathbf{a}, b) with $\mathbf{a} = (a_0(x), a_1(x), \dots, a_k(x)) \in R_q^k$ and $b = \mathbf{a} \cdot \mathbf{s} + e \in R_q$.
 - Used in CRYSTALS-KYBER.

Numerous attacks proposed against LWE

- Lattice reduction \rightarrow LLL, BKZ, etc.
- uSVP \rightarrow recover short vectors in embedded lattice
- Dual and Dual Hybrid \rightarrow find short vectors in dual lattice, distinguish
- Machine Learning \rightarrow train ML models on reduced LWE data
- Cool and Cruel \rightarrow brute force secret guesses from reduced LWE data
- And many more

Concrete LWE attack performance not well-understood

Settings of published LWE attack experimental results (last 10 years)

- Besides ML and CC attacks (our work), highest evaluated $n \leq 256$.
- No systematic comparison of attack performance.
- Few works consider near-real world parameter settings or other realistic choices (RLWE, MLWE)

Attack type	Setting	Highest n	$\log q$
uSVP	LWE	200	$7 \leq \log_2 q \leq 21$
uSVP	LWE*	90	11, 12, 13
uSVP	LWE	100	7, 9
uSVP	LWE*	75	12, 13
uSVP/BDD	LWE	80	12
uSVP	LWE*	75	11, 13
MiTM	LWE	256	12
MiTM	RLWE	130	21, 22
ML	LWE	128	9
ML	LWE	350	32
ML	LWE	512	41
ML	LWE	1024	50
Cool & Cruel	LWE	1024	50

* LWE from Darmstadt Challenges

Existing efforts to benchmark concrete LWE attack performance are limited



Given the security-critical nature of LWE, we need concrete benchmarks of LWE attack performance at near-real-world settings.

Our proposal: concrete
benchmarks for LWE attack
performance

Proposed settings and attacks evaluated

Benchmark results

Lessons learned

Proposed settings and attacks evaluated

Benchmark results

Lessons learned

Proposed settings for LWE attack benchmarks

Useful benchmarks settings should use real-world parameters and have tunable hardness settings. We propose:

- Benchmarks based on standardized CRYSTALS-KYBER and HE parameters.
- Fix n and vary problem hardness via modulus size q and secret Hamming weight h .

Proposed benchmark settings

	Setting	n	MLWE k	$\log_2 q$	Secret distribution	Error distribution
Kyber settings	MLWE	256	2	12	Binomial, $\eta = 2$	Binomial, $\eta = 2$
	MLWE	256	2	28	Binomial, $\eta = 2$	Binomial, $\eta = 2$

Choosing initial attacks to benchmark

- Prefer attacks that:
 - Are open source
 - Run in less than 750GB of RAM
- This leaves 4 choices: uSVP, Dual Hybrid MiTM (Decision LWE), SALSA, Cool & Cruel
- Future work: expand set of attacks considered

uSVP attack

We solve uSVP using Kannan's embedding. Given LWE sample $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{(m \times n)}, \mathbb{Z}_q^m$, create Kannan's embedding via:

$$\begin{bmatrix} 0 & q\mathbf{I}_m & 0 \\ \mathbf{I}_n & \mathbf{A}^T & 0 \\ 0 & \mathbf{b} & 1 \end{bmatrix}$$

The space spanned by these rows contains a short vector $(\mathbf{s} \quad -\mathbf{e} \quad -1)$.

We run lattice reduction using BKZ2.0, incorporating improvements of [1].

Cool and Cruel (CC) attack: lattice reduction

Given LWE sample $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{(m \times n)}, \mathbb{Z}_q^m$, we first create the embedded lattice:

$$\Lambda = \begin{bmatrix} 0 & q \cdot \mathbf{I}_n \\ \omega \cdot \mathbf{I}_m & \mathbf{A} \end{bmatrix}$$

We reduce this matrix, finding unimodular transformation $[\mathbf{L} \ \mathbf{R}]$ which minimizes the norms of $[\mathbf{L} \ \mathbf{R}]\Lambda = [\omega\mathbf{R} \ \mathbf{RA} + q\mathbf{L}]$ and allows us to compute $(\mathbf{RA}, \mathbf{Rb}) = (\mathbf{A}', \mathbf{b}')$

We run lattice reduction using a combination of `BKZ2.0` [1] and `flatter` [2]. Running `flatter` first allows us to run `BKZ2.0` on larger n LWE problems.

[1] Chen and Nguyen, "BKZ2.0: Better Lattice Security Estimates." ASIACRYPT 2011.

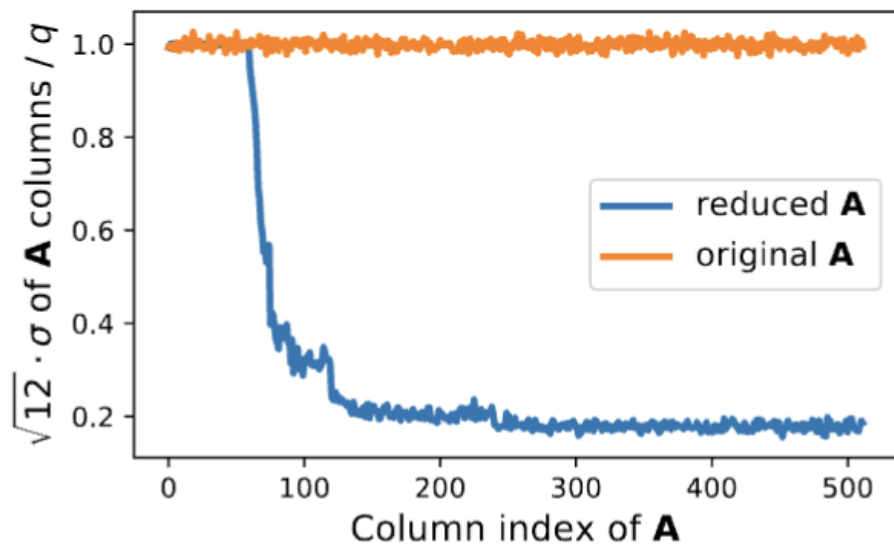
[2] Ryan and Hening, "Fast Practical Lattice Reduction through Iterated Compression." CRYPTO 2023.

Cool and Cruel (CC) attack: summary

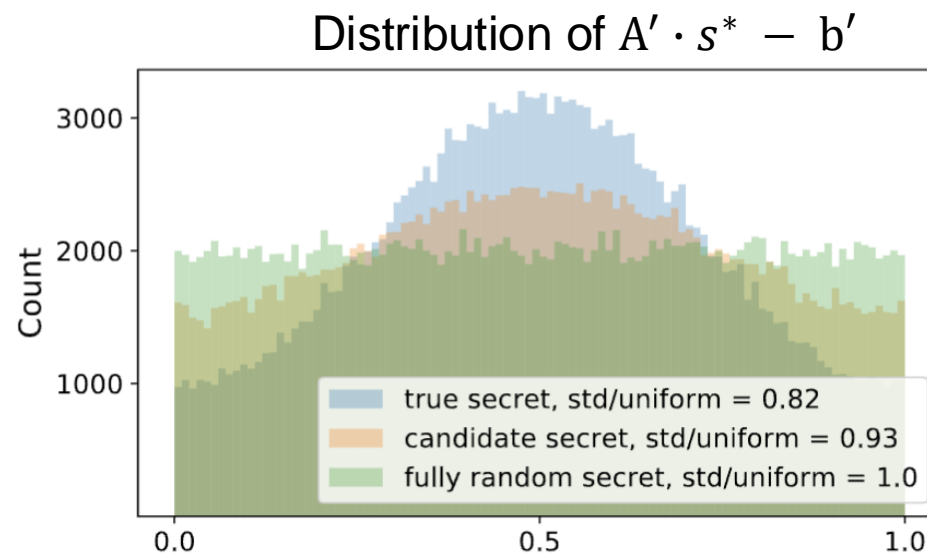
CC attack [1] leverages “cliff” shape in partially reduced LWE data.

Observe that $A' = (A'_u \ A'_r)$ and $b' = A'_u s_u + \cancel{A'_r s_r} + e'$.

Brute forces “cruel” (unreduced) bits, then easily recovers “cool” bits.



Reduced LWE data has “cruel” (unreduced) and “cool” (bits)

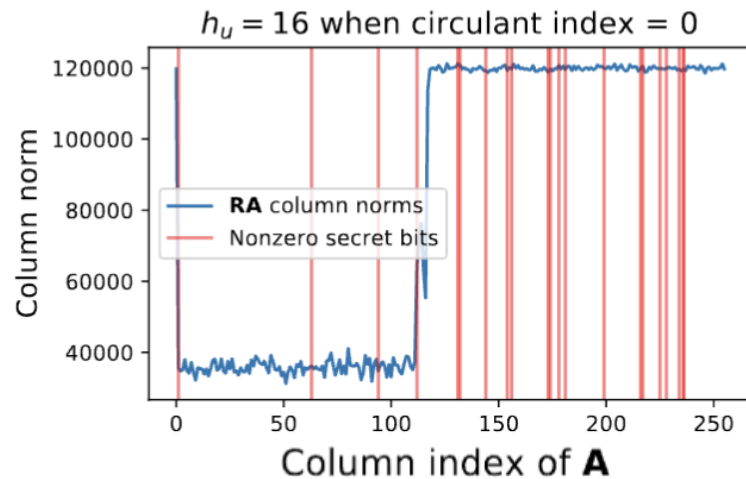


Correct cruel bit guess is statistically distinguishable from random, even with cool bits

Cool and Cruel attack: our contributions

For RLWE and MLWE settings, we can “shift” the cruel region by creating datasets from different rows of $\text{Skew-Circ}(\text{Emb}(a'(x)))$.

This allows us to find cruel region with fewer active secret bits at the cost of more parallel compute.



Example of cliff-shifting in RLWE setting, $n=256$

Cool and Cruel attack: our contributions

Greedy recovery algorithm of [1] does not work well for ternary/binomial secrets.

Since cool bits do not contribute much to $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, can treat cool bit recovery as linear regression:

- Let reduced matrix $\mathbf{A}' = (\mathbf{A}'_{\mathbf{u}} \ \mathbf{A}'_{\mathbf{r}})$. $\mathbf{A}'_{\mathbf{u}}$ is cool region and $\mathbf{A}'_{\mathbf{r}}$ is cruel region.
- Then $\mathbf{b}' = \mathbf{A}'\mathbf{s} \bmod q = (\mathbf{A}'_{\mathbf{u}} \ \mathbf{A}'_{\mathbf{r}}) \cdot (s_{\mathbf{u}} \ s_{\mathbf{r}})^{\mathbf{T}} = \mathbf{A}'_{\mathbf{u}}s_{\mathbf{u}} + \mathbf{A}'_{\mathbf{r}}s_{\mathbf{r}}$.
- If $s_{\mathbf{u}}$ is known, then we can compute $s_{\mathbf{r}}$ from the least-squares estimator

$$\hat{s}_{\mathbf{r}} = (\mathbf{A}'_{\mathbf{r}}{}^{\mathbf{T}} \ \mathbf{A}'_{\mathbf{r}})^{-1} \mathbf{A}'_{\mathbf{r}}{}^{\mathbf{T}} (\mathbf{b}' - \mathbf{A}'_{\mathbf{u}}s_{\mathbf{u}} \bmod q)$$

This approach allows us to recover ternary and binomial secrets.

ML attack: summary

 Preprocessing → subsample and reduce initial $4n$ LWE samples

 Transformer model → train model on reduced LWE samples

 Secret recovery  → extract secret prediction from model

 Secret verification → check if secret is correct

[1] Wenger et al, “SALSA: Attacking Lattice Cryptography with Transformers”. Proc. of NeurIPS, 2022.

[2] Li et al, “SALSA PICANTE: A Machine Learning Attack on LWE with Binary Secrets.” Proc. of CCS, 2023.

[3] Li et al, “SALSA VERDE: A Machine Learning Attack on LWE with Small, Sparse Secrets.” Proc. of NeurIPS, 2023.

[4] Stevens et al, “SALSA FRESCA: Angular Embeddings and Pre-Training for ML Attacks on Learning with Errors.” In submission, 2024.

ML attack: our contributions

Adapted preprocessing for RLWE/MLWE

Used cliff-shifting trick for RLWE and MLWE to improve secret recovery

Implemented “slope distinguisher” to recover general secrets:

- Idea: to find value of nonzero active secret bits, modify input elements to $\mathbf{a} + \delta \mathbf{e}_i$, where δ is small nonzero value and \mathbf{e}_i is i^{th} basis vector.
- If s_i is nonzero, then model f should give $f(\mathbf{a}) - f(\mathbf{a} + \delta \mathbf{e}_i) = \delta s_i$.

MitM attack: context

- Dual Hybrid Meet-in-the-Middle (MitM) attacks decision LWE.
- uSVP, C&C, and ML attack search LWE.
- Typically, dual attacks try to find short vectors in the *dual lattice*, then use these to distinguish in some way.
- One common approach: combine dual attack with meet-in-the-middle (MiTM) algorithm.
- We implement [Cheon et al 2019]: Dual Hybrid MitM

MiT_M attack: summary

[Cheon et al 2019] Dual Hybrid Mit_M in a nutshell:

1. Split LWE sample into two parts, $\mathbf{A} = (\mathbf{A}_1 \ \mathbf{A}_2)$.
2. Reduced the “scaled normal dual” embedding of \mathbf{A}_1 to get short vectors that “eliminate” \mathbf{s}_1 , the secret associated with \mathbf{A}_1 .
3. Use short vectors to create lookup table \mathbf{T} containing partial guesses of \mathbf{s}_2 .
4. Query with other partial \mathbf{s}_2 guesses until you find a collision.
5. Collision recovers \mathbf{s}_2 and solves decision-LWE for \mathbf{A} .
6. Additional work needed to recover \mathbf{s}_1 and solve search-LWE for \mathbf{A} .

MitM attack: our contributions

We innovate to help MitM run on larger n, q and more general secrets:

- The τ (# short vectors) and ζ (guessing dimension) given by Estimator do not work well in practice.
 - We increase ζ to better trade-off reduction and guessing time.
 - We find $\tau = 50$ is sufficient for all settings considered.
- We trade-off time and memory by only storing active secret bit guesses and exhausting bit values, allowing MitM to run on binomial secrets.
- Attackable Hamming weight constrained by the 750GB of RAM on our machines.

Proposed settings and attacks evaluated

Benchmark results

Lessons learned

Results on benchmark settings

Attack	Results	Kyber MLWE Setting $(n, k, \log_2 q)$			HE LWE Setting $(n, \log_2 q)$		
		$(256, 2, 12)$ binomial	$(256, 2, 28)$ binomial	$(256, 3, 35)$ binomial	$(1024, 26)$ ternary	$(1024, 29)$ ternary	$(1024, 50)$ ternary

Takeaways from benchmark experiments

- uSVP does not succeed.
- ML and CC attacks take roughly equivalent time.
- CC attack achieves highest h , but scales badly due to exhaustive search on cruel bits.
- Decision MitM attack has lowest compute requirements, but scales badly in search time and search space required.
- *In terms of recoverable secret Hamming weight h , CC is currently the best-performing attack on our benchmark settings*

Proposed settings and attacks evaluated

Benchmark results

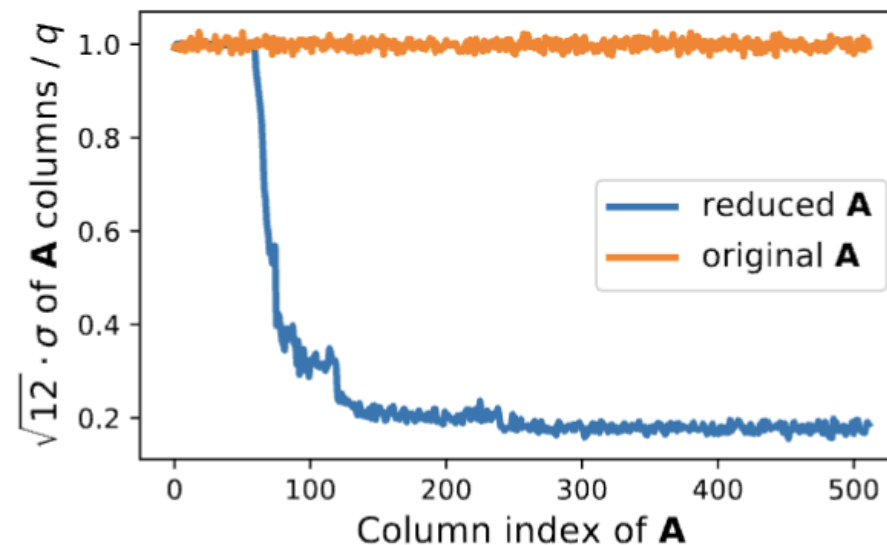
Lessons learned

“Cliff” in preprocessed LWE data

Simple visual inspection of preprocessed data led to observation of “cliff” shape.

Admits at least the Cool and Cruel attack, may admit more.

We are the first to demonstrate vulnerabilities in RLWE instances derived from 2-power cyclotomic rings.



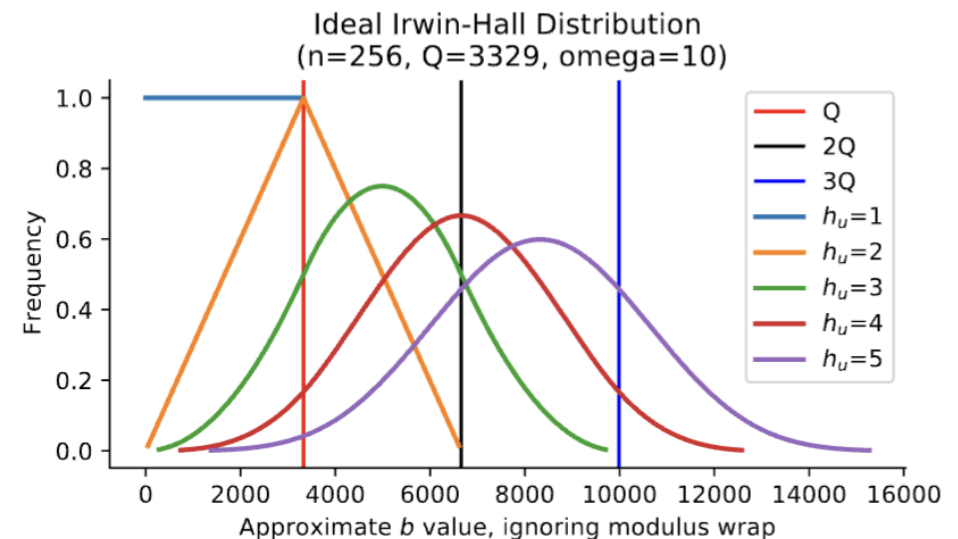
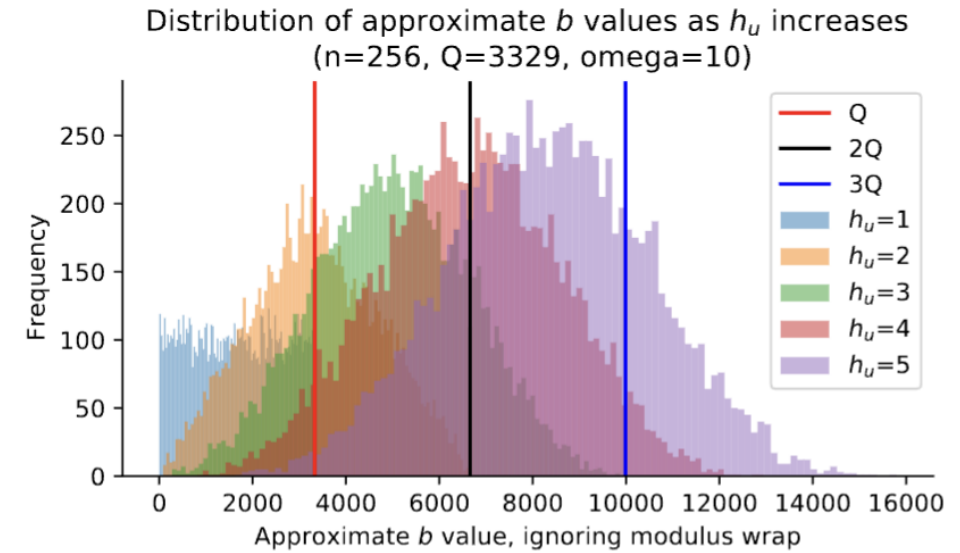
Reduced LWE data has “cruel” (unreduced) and “cool” (bits)

ML attacks recover up to 3 cruel bits

Applying CC insights to ML attack, we found ML models recover secrets with at most 3 cruel bits.

Sums of uniform random (cruel) elements follow Irwin-Hall distribution – see distribution.

This implies that ML models struggle with modular arithmetic – more work needed.

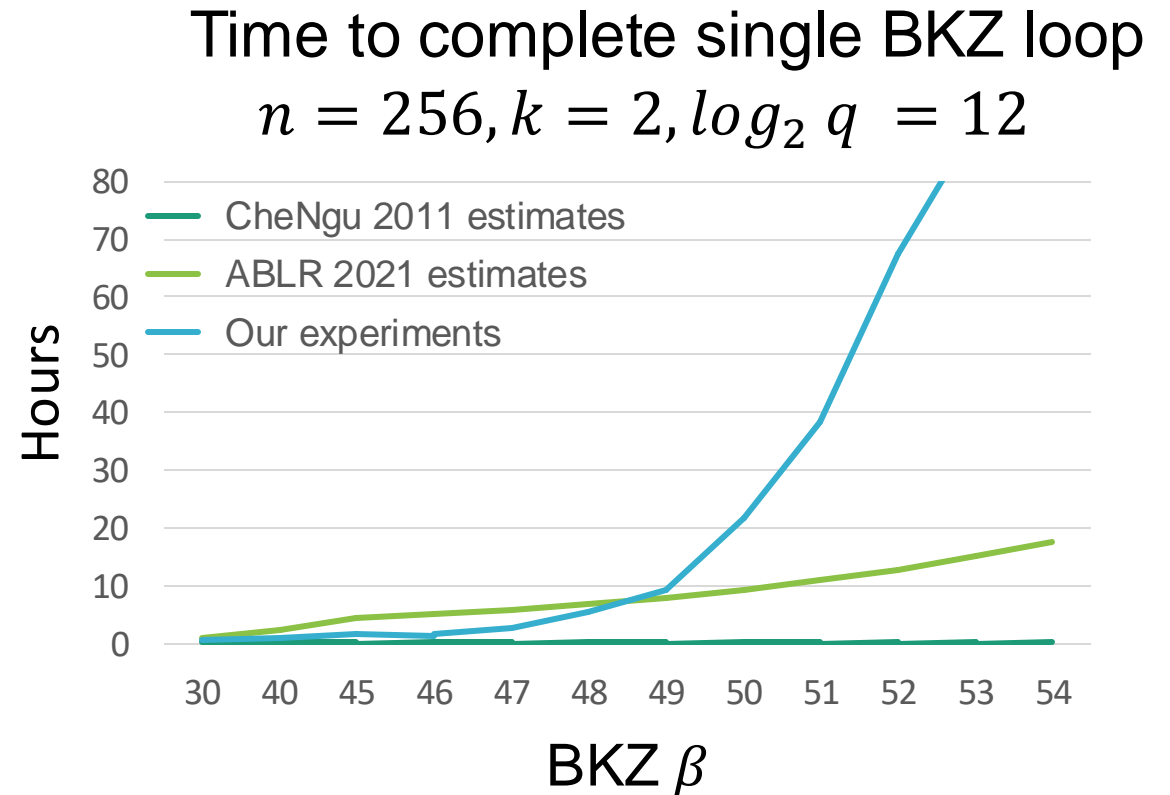


BKZ timing discrepancies

We also observed discrepancies in predicted vs. actual BKZ timings for $n = 256, k = 2, \log_2 q = 12$ (KYBER setting)

Exponential time trend for $\beta > 49$, but estimates do not predict this.

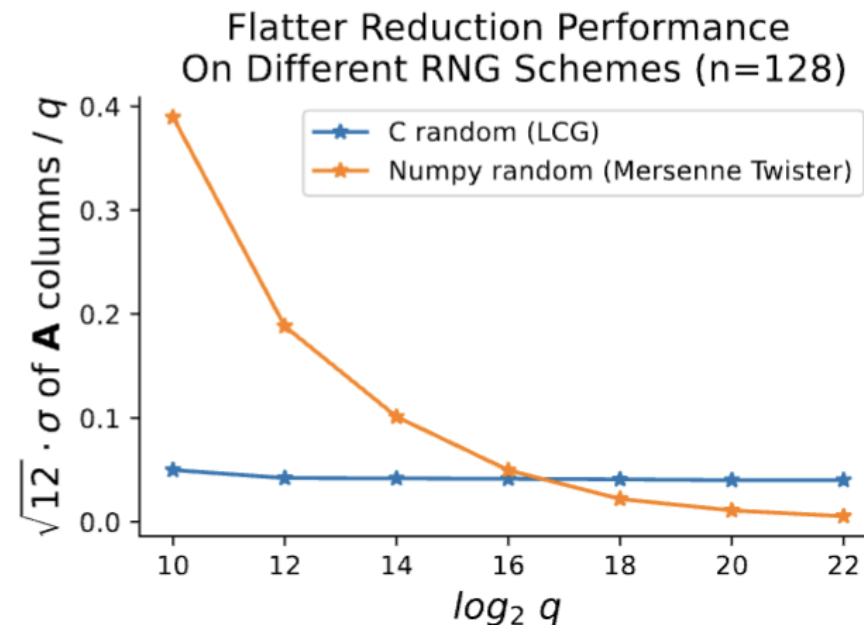
Additional study of this is needed!



Bad RNGs make LWE attacks very easy

LWE \mathbf{A} matrices constructed via LCGs are very vulnerable to lattice reduction and subsequent ML attacks.

It's (still) an easy mistake to make: the `C random()` library still uses an LCG under-the-hood on Mac/Linux. Coder beware!



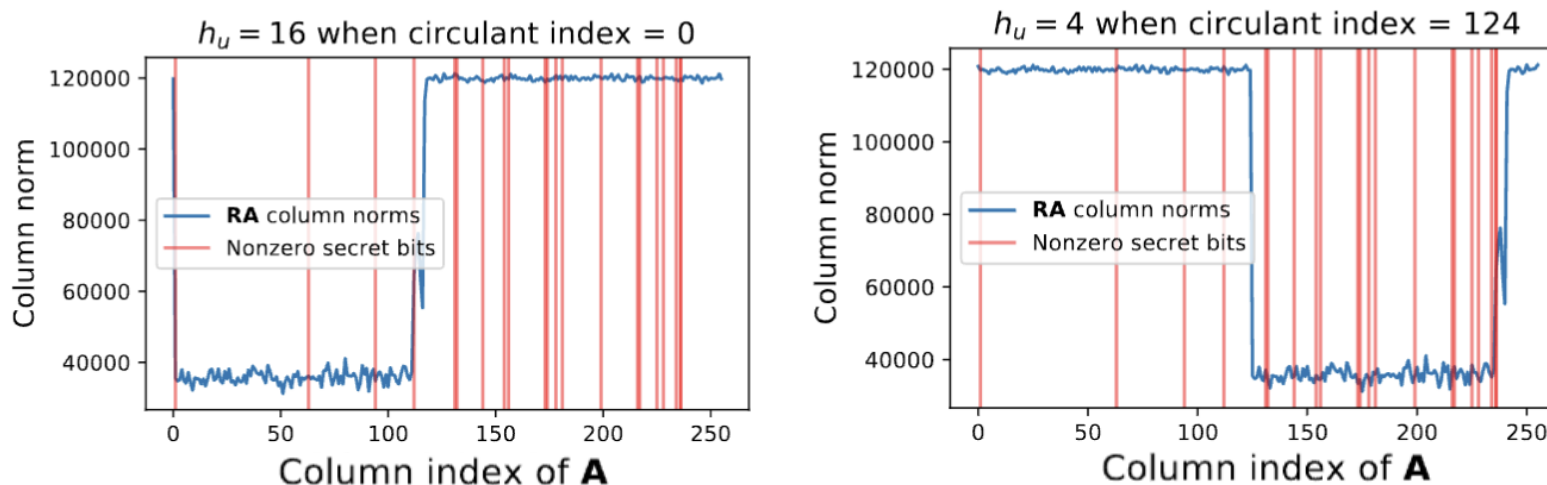
h	50	70	90
attack time (hrs)	3	3.75	3
recovery rate	5/5	5/5	5/5

ML attack performance on LCG-generated data
($n = 256$, $\log_2 q = 12$, varying h)

Bad RNGs could enhance CC attack

In RLWE/MLWE setting, we can enhance the CC attack by shifting around the cruel bits until we find a region with fewer active cruel bits.

Food for thought: A biased RNG that produces more 0s in a certain part of secret would make this attack *much* more powerful.



Example of cliff-shifting in RLWE setting, $n=256$

Join our benchmarking effort

- See: <https://github.com/facebookresearch/LWE-benchmarking>.
- Key contributions needed:
 - Optimizations of existing attacks
 - Implementation/evaluation of new attacks
- Additional future work:
 - Revisit theoretical estimates with concrete observations
 - Better understand “cliff” behavior in reduction

Thank you!

Questions?