Improving ML attacks on LWE with data repetition and stepwise regression

Alberto Alfarano[*]
FAIR, Meta

Eshika Saxena
FAIR, Meta

Emily Wenger
Duke University

François Charton[‡†]
Axiom Math

Kristin Lauter[†]
FAIR, Meta

Abstract

The Learning with Errors (LWE) problem is a hard math problem used in lattice-based cryptography. In the simplest case of binary secrets, it is the subset sum problem with error. Effective ML attacks on LWE were demonstrated in the case of binary, ternary, and small secrets, succeeding on fairly sparse secrets. After lattice BKZ pre-processing, past ML attacks recovered secrets with up to three non-zero bits in the "cruel region [9]". We show that using larger training sets and repeated examples enables recovery of denser secrets. Empirically, we observe a power-law relationship between model-based attempts to recover the secrets, dataset size, and repeated examples. We introduce a stepwise regression technique to recover the "cool bits" of the secret, a substantial improvement over prior work.

Table 1: **Largest Hamming weights recovered (two settings).**

Settings	Ours	SALSA	C&C
n=256 $\log_2 q = 12$	**14**	8	12
n=256 $\log_2 q = 20$	**70**	33	-
n=512 $\log_2 q = 28$	**12**	-	**12**
n=512 $\log_2 q = 41$	**75**	63	60

Binary secret

Settings	Ours	SALSA	C&C
n=256 $\log_2 q = 12$	**12**	9	-
n=256 $\log_2 q = 20$	**55**	24	-
n=512 $\log_2 q = 28$	**10**	-	-
n=512 $\log_2 q = 41$	**75**	66	-

Ternary secret

1 Introduction and Related Work

Most current public-key cryptosystems, used to secure online interactions, are susceptible to quantum attacks based on Shor's algorithm [14]. To address this, post-quantum cryptography (PQC) schemes are being standardized [2], with ongoing research into their weaknesses and limitations. Many PQC systems rely on the Learning With Errors (LWE) [11] problem, which asks to recover a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ given pairs $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + \epsilon_i \mod q)$, with $\mathbf{a}_i$ randomly sampled from a uniform distribution over $\mathbb{Z}_q^n$, and ϵ_i sampled from a discrete Gaussian distribution with low standard deviation. LWE is hard when $\mathbf{s}$ is chosen uniformly, for large values of n, and not too large moduli q. Alternatives with *sparse* and binary, ternary or small secrets (i.e. non-zero coordinates are 1, 1 or -1, or small integers, respectively) are attractive for efficiency of homomorphic encryption. The potential weaknesses of these variants, for practical values of n and q, is under debate.

[*]Corresponding author: albealfa@meta.com

[†]Co-senior authors

[‡]Work done at FAIR, Meta

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

Recent works introduced SALSA, a transformer-based ML attack on LWE with small and sparse secrets [17, 8, 7, 15]. SALSA trains a model to predict b from $\mathbf{a}$. Secret coordinates can be recovered by comparing model predictions for close values of $\mathbf{a}$. Later works [8, 7, 15] used lattice reduction algorithms (BKZ [13, 3]) to preprocess LWE samples. BKZ reduction splits $\mathbf{a}$ into two components: the last coordinates of $\mathbf{a}$ (the "cool" region) have low variance, while the c first coordinates (the "cruel" region) keep the same variance [9]. BKZ reduction allows the recovery of secrets with higher Hamming weight h (number of non-zero secret bits) [8, 7, 15]. However, models struggle to learn the noisy dot product when there are more than three non-zero cruel bits [18]. For example, for $n = 512$, and $\log_2 q = 28$, the cruel region size is 224. Thus, secrets with $h \geq 10$ likely have at least 4 cruel bits and are not recoverable. Appendix A offers additional details on related work.

We present methods to recover secrets with more than 3 cruel bits. Overall, **our contributions** are:

- enabling secret recovery with up to 8 cruel bits using larger datasets and repeated examples [1];
- replacing the linear regression used to recover cool bits [18] by stepwise regression [5];
- exploring secret recovery as a function of data size, data repetition, secret Hamming weight and cruel bits, in four LWE settings, through BKZ-reduced and synthetic data samples (Appendix C);
- defining an empirical scaling power law relating secret recovery to data amount and data repetitions.

Overall, our attack recovers secrets with larger Hamming weights than previous works (Table 1), and demonstrates that the limitation of ML attacks to secrets with 3 cruel bits can be overcome.

2 Experimental settings

LWE settings. We consider four LWE parameter sets used in prior work [7, 9] (see Table 7 in Appendix B): $(n, \log_2 q) \in \{(256, 12), (256, 20), (512, 28), (512, 41)\}$ with standard deviation of the error equal to 3. Smaller q values make BKZ reduction harder and increase the size of the cruel region. Larger q values enable more reduction and higher recoverable Hamming weights. We select n for which running lattice BKZ reduction is reasonable, but brute-force attacks are not. In particular, for $n = 256$, recovering a $h = 14$ secret takes 10^{22} attempts, each requiring thousands of operations. This would take several months on the fastest current supercomputer (10^{18} FLOPS), at the frontier of current brute force capability. For $n = 512$, recovering a 75-bit secret requires 10^{91} attempts, which is far beyond any brute-force attack capability.

Data generation and reduction. We implement the AI-based attack described in Fresca [15, 18]. We sample $m \leq n$ vectors from $4n$ random Z_q^n vectors and stack them in a matrix

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

$\mathbf{\Lambda}$ is reduced using flatter [9] interleaved with BKZ2.0, yielding $\mathbf{R\Lambda}$ with lower variance. LWE pairs $(\mathbf{A}, \mathbf{b})$ become $(\mathbf{RA}, \mathbf{Rb})$ with the same secret, but a larger error ϵ. This process is repeated to generate the train and test sets. The first c columns, the cruel region, remain unreduced [9], while the last $n - c$ are reduced. Please see Appendix B and Table 7 for reduction parameters.

Synthetic data. We generated 400 million reduced pairs for $n = 256$ and $\log_2 q = 20$ and between 4 and 60 for other settings. BKZ required around 42M CPU hours to generate our datasets. We generated 400 million synthetic reduced pairs per setting in 1,000 CPU hours, matching the BKZ-reduced data coordinate variance. Appendix I, J shows similar recovery rates for models trained on reduced or synthetic data. Refer to Appendix C for an additional discussion on synthetic data.

Model. We train a 4-layer encoder-only transformer with embed dim $d = 256$ (4 heads) for larger q and $d = 512$ (8 heads) for smaller q. Each coordinate a_i is encoded by the angular embedding [15]. We add to the input a absolute position embedding and a "cool-cruel" embedding, indicating whether the current position is in the cruel or cool region (see Appendix D for additional details of this embedding). As in [15], the transformer output is max-pooled and mapped to (o_1, o_2).

Training. Loss is the mean-square error between (o_1, o_2) and correct answers $(\cos(\frac{2\pi b}{q}), \sin(\frac{2\pi b}{q}))$. To prevent output collapse to $(0, 0)$, prior work [12] added a penalty $\alpha r^2 + \beta \frac{1}{r^2}$. We set $\alpha = \beta = 0.1$ (see Appendix E for a deeper discussion). Training jobs use one V100 GPU (32 GB), for up to 2 billion examples and up to 5 days. We use the Adam optimizer [6] with batch size 256.

Secret recovery. After each epoch (2.5 million training examples), the distinguisher introduced in [7] is run on 1000 reduced LWE examples, and ranks the c cruel columns of the secret by their likeliness of being different from zero. We perform $15,000$ model based attempts, and in each of them cool bits are estimated via stepwise regression 4. Secret guesses are evaluated as in PICANTE [8]. If the secret is discovered training stops; otherwise another training epoch is run.

3 Recovering higher Hamming weights with large sets of repeated examples

Prior work found that ML attacks on LWE samples cannot recover secrets with $h > 3$ from non-reduced data, or more than 3 cruel bits on reduced data [9, 18]. In Appendix F we show that larger datasets and repeated data recovers secret with $h > 3$ from non-reduced data.

Here, we show that larger and repeated training sets allow recovery of more than 3 cruel secret bits from **reduced data**. For $n = 256$, $\log_2 q = 20$ (cruel region size 34, Table 2), and $n = 512$ $\log_2 q = 28$ (cruel region size 228, Table 3), we tested 4 secrets with $h = 4$ and $h = 5$ respectively and train 16 models each. Success is defined as at least one of the 16 models recover all cruel bits.

For $n = 256$ and $n = 512$, data repetition allows recovery from smaller datasets. For $n = 512$, no $h = 5$ secret was recovered from 400 million data without data repetition. Training on large, repeated examples recovers secret with more than 3 cruel bits, an improvement over prior work.

Table 2: $\boldsymbol{n = 256, \log_2 q = 20}$

Data budget	4|5 cruel bits 1x	2x	5x	10x	20x	100x
20M	0|0	0|0	0|0	0|0	0|0	0|0
50M	0|0	0|0	0|0	0|0	1|0	-
100M	0|0	0|0	1|0	1|0	1|0	-
200M	0|0	1|0	2|0	3|1	-	-
400M	3|1	4|1	4|1	-	-	-

Table 3: $\boldsymbol{n = 512, \log_2 q = 28}$

Data budget	4|5 cruel bits 1x	2x	5x	10x	20x	100x
20M	0|0	0|0	0|0	1|0	1|0	1|0
50M	0|0	1|0	1|0	1|0	1|0	-
100M	0|0	1|0	1|0	1|0	2|0	-
200M	0|0	2|0	2|0	2|1	-	-
400M	1|0	4|0	4|1	-	-	-

$x|y$ indicates x models recovered secrets with 4 cruel bits and y models recovered secrets with 5 cruel bits. "-" indicates experiments were not run.

These results also shed new light on the role of cool bits. In theory, secrets with $n = 256, \log_2 q = 20$, $c = 34$ should be easier to recover than those with $n = 512, \log_2 q = 28$, $c = 224$ (smaller n, better BKZ-reduction rate c). Yet, recovering secrets with 4 cruel bits requires only 20 million examples for $n = 512$, but 50 millions for $n = 256$. Thus, higher reduction rates and more cool bits seem to complicate cruel bits recovery.

4 Taming the cool bit noise: stepwise regression

Previous work assumes that once the cruel bits are known, cool bit recovery is easy, and proposes a linear regression recovery method [18] based on $(\mathbf{a}_{cool}, b_{cool} = b - \mathbf{a}_{cruel} \cdot \mathbf{s}_{cruel})$. We propose a new method for cool bit recovery, based on stepwise regression [5]. We fit linear regression to $(\mathbf{a}_{cool}, b_{cool})$. We iteratively set the cool bit with the smallest regression coefficient to zero. We repeat this process until the number of remaining bits matches the known value h_{cool}.

Stepwise regression recovers zeroes. Initially, because the secret is sparse, there are more zeroes than ones. After several steps, the remaining bits are mostly ones. At this stage, it is more efficient to flip all the remaining cool bits, and perform the regression on $b_{dual} = \mathbf{a}_{cool}(\mathbf{1} - \mathbf{s}_{cool}) = \mathbf{a}_{cool}^{\top}\mathbf{1} - b_{cool}$. We call this variant *dual stepwise regression*. We provide the motivation for stepwise in Appendix G.

Tables 4 and 5 compare linear, stepwise and dual stepwise regression in two settings with large BKZ reduction (see Appendix H for other settings). We test secrets with cruel Hamming weight h_{cruel} between 4 and 8, such that $h_{cruel}/h = c/n$. For each value of h, we run the three methods on 20 different secrets and report the number of secrets recovered (assuming cruel bits are known). Stepwise regression outperforms linear regression, and dual stepwise regression achieves the best results. In both settings, dual stepwise regression allows recovery of harder secrets with fewer samples.

Table 4: $n = 256, \log_2 q = 20$

Cool bits (Total h)	Linear 2M	Linear 20M	Stepwise 2M	Stepwise 20M	Dual 2M	Dual 20M
26 (30)	2	13	3	20	13	20
32 (37)	0	5	0	20	5	20
38 (44)	0	1	0	11	1	20
45 (52)	0	0	0	3	0	18
52 (60)	0	0	0	1	0	14

Table 5: $n = 512, \log_2 q = 41$

Cool bits (Total h)	Linear 1M	Linear 4M	Stepwise 1M	Stepwise 4M	Dual 1M	Dual 4M
40 (44)	20	20	20	20	20	20
50 (55)	20	20	20	20	20	20
60 (66)	13	18	17	20	20	20
70 (77)	6	17	15	19	20	20
80 (88)	0	12	10	18	19	20

Cool bits recovery out of 20 secrets. Higher is better.

5 Overall secret recovery and scaling laws

In Table 6, we report the highest recoverable Hamming weight h across four settings when we vary training set size, repetition and BKZ-reduced vs synthetic data. For both binary and ternary secrets, our method consistently recovers higher Hamming weights with at least 3 cruel bits. Optimal results require large datasets and extensive data repetition. Our improvements over prior work are most significant in settings with more reduction. With $n = 256$ $\log_2 q = 20$, prior work recovers $h = 33$ (binary) and $h = 24$ (ternary), while our method achieves $h = 70$ and $h = 55$. Appendices I and J detail the highest Hamming weight h for each distinct data quantity and repetition combination, showing similar results with BKZ-reduced data and synthetic data.

	Binary secret				Ternary secret			
n	256	256	512	512	256	256	512	512
$\log_2 q$	12	20	28	41	12	20	28	41
Best h (Best cruel)	**14** (5)	**70** (8)	**12** (5)	**75** (7)	**12** (5)	**55** (8)	10 (4)	**75** (7)
Previous best h	12	33	**12**	63	9	24	-	66
Data, repetition	4M, 15x	100M, 5x	20M, 15x	20M, 15x	200M, 5x	200M, 2x	50M, 15x	50M, 15x

Table 6: **Best recovery result for different settings on binary and ternary secrets.**

Next, we empirically study how the number of model-based attempts A needed to recover a secret depends on model parameters N, data amount D, and repetitions R for $n = 256$, $\log_2 q = 20$ and binary secrets of Hamming weight h. We define A as the number of attempts (from the distinguisher output) needed to recover the correct secret bits.

Model parameters law: To understand scaling between model parameters N and model-based attempts A, we vary the embedding dimension (256 to 1024) and layers (4 to 12) for three different secrets with Hamming weight $h = \{60, 65, 70\}$. We use 100M data and 1 repetition. As shown in Figure 1, the number of model based attempts is not improved by an increase in model parameters. We leave the exploration of larger models for future work.

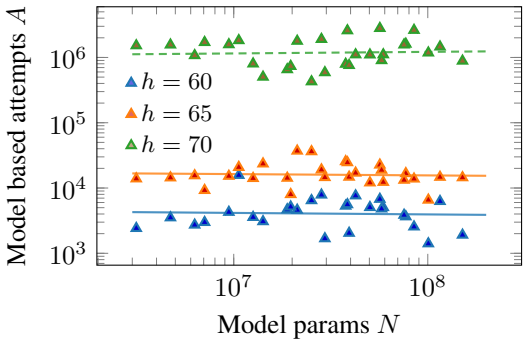

Figure 1: **Model parameters N vs model based attempts A for three different Hamming weight h.**

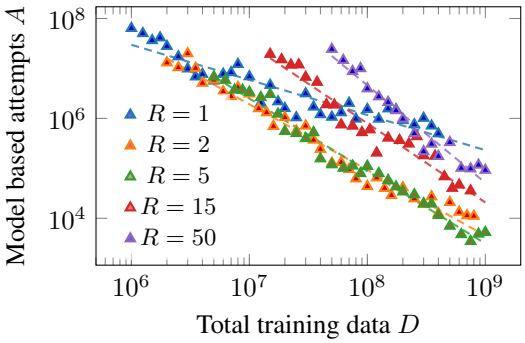

Figure 2: **Total training data D and repetition R vs model based attempts A for $h = 70$.**

Data-repetition law: We fix the model embedding dimension at 256 and 4 layers. We vary the distinct data amount from 1M to 400M and data repetitions R from 1 to 50. We define D as the total training data: distinct training data times R. Figure 2 show results from the best of 8 model initializations. We fit $\ln(A_R) = C_R - \alpha_R \ln(D)$ using least square errors across 5 distinct R regimes. Notably, our experiments reveal that α_1 is considerably lower than α_R, $R > 1$. Thus, data repetition reduces the need for distinct (costly) samples and reshapes the scaling law of A as a function of D. **Increasing data alone is important but insufficient**.

6 Discussion and conclusion

We introduced three techniques to improve ML attacks in LWE problem: larger training sets, repeated examples, and stepwise regression. These methods significantly increase both maximum recoverable Hamming weight and the proportion of secrets recovered for a given h. Our conclusions extend in two directions. First, larger training sets and repeated data are essential for recovering more cruel bits, confirming previous observations [1, 12]. Second, stepwise regression is beneficial, likely because the columns of matrix $\mathbf{A}$ are uncorrelated and stepwise regression enforces this inductive bias. Finally, by investigating scaling laws on LWE, we provide some insights for tackling harder LWE problems.

Our findings advance ML-based attacks on LWE and outperform non-ML strategies at comparable budgets and Hamming weights. On a broader level, this line of work is important for understanding potential limitations of PQC systems before being deployed at scale.

References

[1] François Charton and Julia Kempe. Emergent properties with repeated examples. *ArXiv preprint: 2410.07041*, 2024.

[2] Lily Chen, Dustin Moody, Yi-Kai Liu, et al. PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. *US Department of Commerce, NIST*, 2022. `https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4`.

[3] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In *Proc. of ASIACRYPT 2011*, 2011.

[4] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *Proc. of ICLR*, 2019.

[5] Michael Alin Efroymson. Multiple regression analysis. *Mathematical methods for digital computers*, pages 191–203, 1960.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

[7] Cathy Li, Emily Wenger, Zeyuan Allen-Zhu, Francois Charton, and Kristin Lauter. SALSA VERDE: a machine learning attack on Learning With Errors with sparse small secrets. In *Proc. of NeurIPS*, 2023.

[8] Cathy Yuanchen Li, Jana Sotáková, Emily Wenger, Mohamed Malhou, Evrard Garcelon, François Charton, and Kristin Lauter. SALSA Picante: A Machine Learning Attack on LWE with Binary Secrets. In *Proc. of ACM CCS*, 2023.

[9] Niklas Nolte, Mohamed Malhou, Emily Wenger, Samuel Stevens, Cathy Li, François Charton, and Kristin Lauter. The cool and the cruel: separating hard parts of LWE secrets. *Proc. of AFRICACRYPT*, 2024.

[10] Theodoros Palamas. Investigating the ability of neural networks to learn simple modular arithmetic. 2017.

[11] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proc. of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 2005. `https://dblp.org/rec/journals/corr/cs-DS-0304005.bib`.

[12] Eshika Saxena, Alberto Alfarano, Emily Wenger, and Kristin E. Lauter. Making hard problems easier with custom data distributions and loss regularization: A case study in modular arithmetic. In *Forty-second International Conference on Machine Learning*, 2025.

[13] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987.

[14] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994.

[15] Samuel Stevens, Emily Wenger, Cathy Yuanchen Li, Niklas Nolte, Eshika Saxena, Francois Charton, and Kristin Lauter. Salsa fresca: Angular embeddings and pre-training for ml attacks on learning with errors. `https://eprint.iacr.org/2024/150`.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

[17] Emily Wenger, Mingjie Chen, François Charton, and Kristin E Lauter. Salsa: Attacking lattice cryptography with transformers. *Proc. of NeurIPS*, 2022.

[18] Emily Wenger, Eshika Saxena, Mohamed Malhou, Ellie Thieu, and Kristin Lauter. Benchmarking attacks on learning with errors. *Proceedings of IEEE Symposium on Security & Privacy*, pages 279–297, 2025.

Appendix

A Related Work in Depth

SALSA [17] is the first ML attack on LWE. It uses a shallow sequence-to-sequence transformer [16], with shared layers [4], trained on 2 million unreduced LWE pairs, and recovers secrets from the trained model with a basic distinguisher. Limited to dimensions up to 128 and Hamming weight 3, it is proof of concept: all secrets recovered by SALSA could be found with exhaustive search. It also requires millions of eavesdropped LWE pairs with the same secret, an unrealistic assumption.

PICANTE [8] introduces pre-processing. It only requires $4n$ eavesdropped LWE pairs (a realistic assumption) which are sampled to form $n \times n$ matrices $\mathbf{A}$, and reduced by BKZ to produce matrices $\mathbf{RA}$ with a low standard deviation of their entries. Applying the same transformation to $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \epsilon$ yields reduced LWE pairs with the same secret, but increased noise. A transformer is then trained from 4 million reduced pairs, that can recover sparse binary secrets with $h = 31$ for $n = 256$ and $\log_2 q = 23$, and $h = 60$ for $n = 350$ and $\log_2 q = 32$.

VERDE [7] improves reduction techniques and distinguishers, recovering secrets with $h = 63$ for $n = 512$ and $\log_2 q = 41$. It also extends the recovery mechanism to ternary and small secrets, demonstrating that ternary secrets are not more secure than binary. Finally, it provides a theoretical explanation of the role of reduction: models struggle to learn modular additions that exceed a multiple of the modulus. FRESCA [15] introduces angular embeddings, an architecture for modular arithmetic, which halves the length of inputs, allowing secret recovery for $n = 1024$ and $\log_2 q = 50$.

The difficulty of learning modular addition, first observed by [10], was further studied by [12]. They show that noiseless modular addition of many integers can be learned by adding a regularizer to the loss introduced in FRESCA and adding sparser examples to the training data, in a manner of curriculum learning.

[9] offer a different perspective on BKZ reduction. They observe that the reduction of $\mathbf{a}$ is concentrated to the last coordinates (the cool region), while the c first (the cruel region) stay unreduced. Thanks to the reduction, the cool bits of the secret are easy to recover once the cruel bits known. Therefore, recovering secrets from reduced LWE pairs amounts to discovering their cruel bits, and ML attacks only recover secrets with up to three cruel bits. The paper also proposes a non-ML attack, where all possible cruel bits are enumerated, and the cool bits then guessed. It performs well for small dimensions and reduced data [18], but scales exponentially with the Hamming weight to recover, and requires large samples of LWE pairs for cool bit recovery. [18] provides a comparison between SALSA, Cool and Cruel and classic attacks on LWE for $n = 256$ and 1024.

Training from repeated examples was proposed by [1]. They show that multiplication modulo 67, a task that transformer cannot usually learn when trained from large datasets of single-use examples, can be learned with 100% accuracy when models are trained from smaller sets of repeated data. [12] demonstrate a similar effect for modular addition.

B BKZ-reduced dataset

We report some statistics in Table 7 of the BKZ-reduced datasets. c is the size of the cruel region (coordinates with variance larger than half the variance of the uniform law), σ_{cool} is the standard deviation of the coordinates of the cool region centered in $[-q/2, q/2]$, ρ is the mean of absolute values of off-diagonal pairwise correlation from the covariance matrix of the reduced samples and σ_ϵ is the standard deviation of $\mathbf{R}\epsilon$ centered in $[-q/2, q/2]$.

It is important to note that reduction creates a cool region of size $n - c$ but introduces an error term with standard deviation approximately equal to $q/\sqrt{12}$ (the standard deviation of one cruel bit).

C Synthetic dataset

Some of the experiments in the manuscript require very large training sets (up to 400 million reduced data), which requires a lot of computing resources. We reduce 400 million LWE samples for $n = 256$

Table 7: **Reduction statistics**. We report dataset size, CPU hours, c (size of the cruel region), σ_{cool} (standard deviation of the cool region), ρ (average non-diagonal pairwise absolute value correlation from the covariance matrix of $\mathbf{RA}$) and σ_ϵ (standard deviation of $\mathbf{R}\epsilon$).

n	$\log_2 q$	Dataset size (millions)	CPU hours (millions)	c	σ_{cool}	ρ	σ_e
256	12	4	0.1	143	$0.30q/\sqrt{12}$	0.18%	$0.88q/\sqrt{12}$
256	20	400	40.4	34	$0.23q/\sqrt{12}$	1.05%	$0.90q/\sqrt{12}$
512	28	60	1.0	224	$0.19q/\sqrt{12}$	0.12%	$0.70q/\sqrt{12}$
512	41	4	0.1	46	$0.15q/\sqrt{12}$	0.18%	$0.80q/\sqrt{12}$

$\log_2 q = 20$, but for the three other settings, we relied on synthetic data to complement smaller BKZ-reduced datasets (see Table 7).

To generate synthetic data, we observe that after reduction the coordinates are uncorrelated (i.e. the off-diagonal terms of the covariance matrix $(\mathbf{RA})^T(\mathbf{RA})$ are very small), i.e. the mean absolute value ρ is close to 0. This suggests a method for generating synthetic data. One million LWE samples are reduced using BKZ, and we use these reduced examples to measure c, σ_{cool} and σ_ϵ. The synthetic reduced $\mathbf{a}$ are then generated by sampling the first c coordinates from a uniform distribution, and the $n - c$ others from a centered distribution with variance σ_{cool}. We then compute $b = \mathbf{a} \cdot \mathbf{s} + \epsilon$, with ϵ a centered discrete Gaussian variable, with standard deviation σ_ϵ. Note that because all unreduced columns are assumed to have the variance of the uniform distribution, the synthetic data is a little less reduced than the BKZ-reduced data.

When BKZ-reduction is applied, the original matrix $\mathbf{A}$ and vector $\mathbf{b}$ of LWE samples are subjected to a linear transformation $\mathbf{R}$. The columns of $\mathbf{RA}$ are not correlated, and have a block structure: the c first column have high variance, and the $n - c$ last columns low variance. Multiplying $\mathbf{RA}$ by any block diagonal matrix of the form

$$\mathbf{O} = \begin{bmatrix} \Omega_1 & 0 \\ 0 & \Omega_2 \end{bmatrix} \tag{1}$$

with Ω_1 and Ω_2 two $c \times c$ and $(n - c) \times (n - c)$ matrices, near-orthogonal, (i.e. with condition numbers close to 1), should have little impact on the distribution of cool and cruel bits. This means we can generate, from BKZ-reduced samples $\mathbf{RA}$, a lot of synthetic samples, with the same secret, reduction factor, and noise, by sampling quasi-orthogonal matrices $\mathbf{O}$, and generating new data $\mathbf{ORA}$ and $\mathbf{ORb}$. We leave an analysis of this data augmentation method for future work.

D Cool and cruel embedding ablation

For $n = 256$ $\log_2 q = 20$ we train 16 different models with 400 million samples and 2 repetitions for 5 different secrets with 4 cruel bits. In Table 8 we compare the recovery rate (i.e. if the secret is fully recovered) with or without the cool and cruel embedding introduced in Section 2. We highlight that the new embedding allows for recoveries on all secrets.

Table 8: **Cool and cruel embedding comparison**.

h	CC embedding disabled	CC embedding enabled
28	**5/16**	**8/16**
30	0/16	**4/16**
33	**3/16**	**7/16**
34	0/16	**5/16**
36	0/16	**1/16**

E Bias in angular embeddings and ways to circumvent it

To predict $b = \mathbf{a} \cdot \mathbf{s} + \epsilon$ from $\mathbf{a}$, our transformer uses an angular embedding. The model outputs a point $P(x, y)$ on the real plane, the integers $i \in \{0, \ldots, q-1\}$ are encoded as the points $B_i(\cos(\frac{2\pi i}{q}), \sin(\frac{2\pi i}{q}))$ on the unit circle, and the model prediction is the point B_i closest to P. If P has polar coordinates $(r\cos(\theta), r\sin(\theta))$, with $r > 0$, the point B_{i_0} closest to P verifies $i_0 = \text{argmin}_i|\theta - \frac{2\pi i}{q}|$.

During training, the model learns to minimize a Mean Square Error (MSE) loss. If the angular embedding of B is (x_0, y_0), and if the model predicts $P(x, y)$, the loss is $l = (x - x_0)^2 + (y - y_0)^2$. Since the possible values of b are uniformly distributed on the unit circle, we can assume, without loss of generality, that $B = (1, 0)$. Therefore, the loss is

$$l = (1 - r\cos(\theta))^2 + r^2\sin^2(\theta) = 1 + r^2 - 2r\cos(\theta).$$

During the early stages of training, the model has not learned modular arithmetic and b is predicted at random. Suppose that all model predictions lie at a distance r of the origin, the average MSE loss is the integral of l over all possible angles, so

$$\mathcal{L}(r) = \frac{1}{2\pi r}\int_{-\pi}^{\pi}(1 + r^2 - 2r\cos(\theta))r\, d\theta = 1 + r^2.$$

Therefore, for a clueless model (a model that predicts b at random), the average loss is $1 + r^2$, and the optimizer can reduce the loss just by making r smaller. Model predictions will therefore tend to collapse towards the origin, at which point the loss becomes constant ($l(0) = 1$ no matter the prediction), and nothing can be learned anymore.

Note that collapse only happens when the model cannot predict b. If the model learns to predict b up to some error, i.e. that, assuming $B = (1, 0)$ the predicted value of θ lies in the interval $[-\varepsilon, \varepsilon]$ for some small ε, the average loss then becomes:

$$\mathcal{L}_\varepsilon(r) = 1 + r^2 - 2r\frac{\sin(\varepsilon)}{\varepsilon} = (r-1)^2 - 2r\left(\frac{\sin(\varepsilon)}{\varepsilon} - 1\right).$$

This shows that once the model starts learning, model predictions stop collapsing to the origin. In other words, model collapse only happens at the beginning of training. (Note, we assume here that model predictions are uniformly distributed over $[-\varepsilon, \varepsilon]$, this is a simplification. It can be shown that the same phenomenon appears if the distribution of predictions is unimodal, and centered on 0).

To prevent the model from collapsing in the initial phase of training, we add to the loss a penalty $\mathcal{P}(r) = \alpha r^2 + \frac{\beta}{r^2}$. The average loss (for a random prediction of b) then becomes

$$\mathcal{L}(r) = 1 + (1 + \alpha)r^2 + \frac{\beta}{r^2}.$$

It reaches a minimum for $\mathcal{L}'(r) = 2(1 + \alpha)r - 2\beta/r^3 = 0$ or $r^4 = \frac{\beta}{1+\alpha}$.

In Table 9 we experiment with two different settings and we report the hardest recovered secret by the two different settings. The first setting fixes $\beta = 1$ and $\alpha = 0$ to force predictions to remain close to the unit circle in the initial, "clueless", phase of learning. The second setting is inherited from Saxena et al. [12], where authors suggest $\alpha = \beta = 0.1$.

Overall the two settings are similar, with the second setting being more successful or at par on almost all data budgets.

Table 9: **Loss comparison**.

Data budget	Repetitions	$\alpha = 0, \beta = 1$	$\alpha = 0.1, \beta = 0.1$
$N = 256 \log_2 q = 20$			
50M	15x	60/8	65/8
100M	5x	65/8	65/8
200M	5x	65/8	65/8
400M	2x	65/8	**70/8**
$N = 512 \log_2 q = 28$			
20M	15x	12/4	**12/5**
50M	15x	12/4	12/4

F Recovering higher Hamming weight - non-reduced case

We consider models trained on **non-reduced data**, as in the original SALSA paper. This setting is the cleanest possible, as there can be no side effects due to BKZ reduction, cool bits, or noise amplification. We experiment with $n = 64$ and $\log_2 q = 20$, and secrets with $3 \leq h \leq 6$, in the presence and absence of noise. For each value of h, we train models on 3 different secrets, and report success if one is recovered at least.

As in SALSA, secrets with Hamming weight 3 are always recovered, even when the model is trained without repetition on one million sample only. Table 10 presents our findings for $h = 4$ to 6. Secrets with $h = 4$ are recovered without repetition for large data budgets, but repetition enables recovery even for small data budgets. For larger Hamming weights require large data budgets *and* repetition. These results, a major improvement over SALSA, demonstrate the potential benefit of large training samples of repeated examples.

Table 10: Secret recovery for different Data budgets (DB), and repetition levels, for secrets with Hamming weight $4, 5$ and 6. –: no secret recovered, ✓: recovery in the noiseless case only, ✓✓: recovery in all cases.

Data	Hamming weight 4					Hamming weight 5					Hamming weight 6				
budget	1x	2x	4x	10x	20x	1x	2x	4x	10x	20x	1x	2x	4x	10x	20x
1M	–	–	–	✓	✓	–	–	–	–	–	–	–	–	–	–
4M	✓	✓	✓✓	✓✓	✓✓	–	–	–	✓	✓	–	–	–	–	–
20M	✓	✓	✓✓	✓✓	✓✓	–	–	✓	✓	✓✓	–	–	–	✓	✓
50M	✓✓	✓✓	✓✓	✓✓	✓✓	–	–	✓	✓✓	✓✓	–	–	✓	✓	✓✓

G Stepwise regression motivation

ML-based attacks recover secrets by comparing model predictions for close values of $\mathbf{a}$. If $\mathbf{a}' = \mathbf{a} + \frac{q}{2}\mathbf{e}_i$, with $\mathbf{e}_i$ the i-th base vector, then the difference between the associated values of $b = \mathbf{a} \cdot \mathbf{s} + \epsilon$

$$b' - b = (\mathbf{a}' - \mathbf{a}) \cdot \mathbf{s} + \epsilon' - \epsilon = \frac{q}{2}\mathbf{s}_i + \epsilon' - \epsilon \mod q$$

has mean zero if $\mathbf{s}_i = 0$, and $q/2$ if $\mathbf{s}_i = 1$. If the transformer produces good predictions of b and b', i.e. $\mathcal{T}(\mathbf{a}) \approx b$ and $\mathcal{T}(\mathbf{a}') \approx b'$ (with $\mathcal{T}$ the transformer prediction), the difference $|\mathcal{T}(\mathbf{a}) - \mathcal{T}(\mathbf{a}')|$, averaged on a sample of vectors $\mathbf{a}$, should allow us to guess the corresponding secret bit.

Previous research (section 5 of [7]) suggests that transformers struggle to learn modular dot products when the sum $|\mathbf{a} \cdot \mathbf{s}|$ becomes larger than a multiple of q. With reduced data, this can happen in two cases: when the number of (unreduced) cruel bits in the secret exceeds a relatively small value, **but also** when the size of the reduced, cool, region becomes large. We believe this accounts for the observation, at the end of the Section 3, that the cruel bits for the "hard setting" $n = 512 \log_2 q = 28$, were easier to recover, than those of the easier setting $n = 256 \log_2 q = 20$, which enjoyed a higher reduction factor. Here, we investigate cool bit recovery, especially in the case when BKZ reduction is high, and the secret has a lot of non-zero cool bits.

Previous research assumes that once the cruel bits are known, cool bit recovery is easy, and proposes a linear regression recovery method [18]. Once the cruel bits are guessed, the quantity $b_{cool} =$

$b - \mathbf{a}_{cruel} \cdot \mathbf{s}_{cruel} = \mathbf{a}_{cool} \cdot \mathbf{s}_{cool} + \epsilon$ is computed ($\mathbf{s}_{cool/cruel}$ represent the restriction of $\mathbf{s}$ to the cool/cruel coordinates), and linear regression is used to predict $\mathbf{s}_{cool}$ from $\mathbf{a}_{cool}$ and b_{cool}.

The use of linear regression, here, is dubious for two reasons. First, we know that the coordinates of $\mathbf{a}$ are not correlated, and that the secret bits are independent. Linear regression, on the other hand, will compute and invert the test sampled covariance matrix $\mathbf{A}^T\mathbf{A}$, which will have non diagonal elements due to population error that will be amplified by matrix inversion. Second, linear regression ignores the fact that the dot product is computed modulo q. As a result, it underestimates the contributions of non-zero bits in the secret (which can "wrap" to zero when their sum exceeds q). Summarizing, we believe that cool bits should better be recovered one by one than all at once, and zero bits are easier to recover that non-zero bits.

We include the pseudocode to replicate the stepwise regression algorithm.

Algorithm 1: Linear Secret Backward Reduction

Input: Matrix $\mathbf{a}_{cool} \in \mathbb{Z}_q^{B \times cool}$, vector $b_{cool} \in \mathbb{Z}_q^{cool}$, int h_{cool}, flag `use_dual_algo`
Output: Secret guess g

$g \leftarrow$ vector of length $cool$ with all entries equal to 0
$active \leftarrow$ vector of length $cool$ with all entries equal to 1
$ones \leftarrow h_{cool}$
$zeros \leftarrow cool - h_{cool}$
while *(`use_dual_algo` $\wedge$ $|active| > 0$)* ***or*** *($\neg$`use_dual_algo` $\wedge$ $|active| > h_{cool}$)* **do**
  $use\_dual \leftarrow (ones > zeros) \wedge$ `use_dual_algo`
  `// One-step regression`
  if use_dual **then**
    $b \leftarrow (\mathbf{a}_{cool}^T \mathbf{1} - b) \bmod q$
  $X \leftarrow \mathbf{a}_{cool}[:, active]$
  $y \leftarrow b$
  $coef \leftarrow \arg\min_c \|y - Xc\|_2^2$
  $coef \leftarrow coef / \max(|coef|)$
  `// Remove weakest feature`
  $j^* \leftarrow \arg\min_j |coef_j|$
  $active[j^*] \leftarrow 0$
  if use_dual **then**
    $b \leftarrow (b - A[:, j^*]) \bmod q$
    $g[j^*] \leftarrow 1$
    $ones \leftarrow ones - 1$
  else
    $zeros \leftarrow zeros - 1$
if $\neg$*`use_dual_algo`* **then**
  $g[active] \leftarrow 1$
return g

H Linear regression

Similar to Section 4 we compare linear, stepwise and dual stepwise regression and we report the cool bits recovery for the two harder settings, where the BKZ-reduced data has a larger cruel region. As shown in Tables 11, 12, 13 and 14 dual stepwise regression shows the best performance.

Table 11: **Cool bits recovery $n = 256, \log_2 q = 12$** assuming cruel bits have been recovered

		Linear regression			Stepwise regression			Dual stepwise regression		
Cool bits	Total h	1M	2M	4M	1M	2M	4M	1M	2M	4M
8	18	20	20	20	20	20	20	20	20	20
18	41	8	20	20	20	20	20	20	20	20
23	52	0	3	7	0	9	20	13	20	20
28	63	0	0	0	0	0	2	0	3	19

Table 12: **Cool bits recovery $n = 256, \log_2 q = 20$** assuming cruel bits have been recovered

		Linear regression				Stepwise regression				Dual stepwise regression			
Cool bits	Total h	2M	4M	10M	20M	2M	4M	10M	20M	2M	4M	10M	20M
20	23	5	14	20	20	5	20	20	20	13	20	20	20
26	30	2	5	11	13	3	10	19	20	13	17	20	20
32	37	0	2	4	5	0	4	13	20	5	9	20	20
38	44	0	0	0	1	0	0	3	11	1	7	13	20
45	52	0	0	0	0	0	0	0	3	0	2	8	18
52	60	0	0	0	0	0	0	0	1	0	0	1	14

Table 13: **Cool bits recovery $n = 512, \log_2 q = 28$** assuming cruel bits have been recovered

		Linear regression			Stepwise regression			Dual stepwise regression		
Cool bits	Total h	1M	2M	4M	1M	2M	4M	1M	2M	4M
10	18	20	20	20	20	20	20	20	20	20
30	53	20	20	20	20	20	20	20	20	20
40	71	15	20	20	20	20	20	20	20	20
50	89	5	8	13	14	17	18	17	20	20

Table 14: **Cool bits recovery $n = 512, \log_2 q = 41$** assuming cruel bits have been recovered

		Linear regression			Stepwise regression			Dual stepwise regression		
Cool bits	Total h	1M	2M	4M	1M	2M	4M	1M	2M	4M
40	44	20	20	20	20	20	20	20	20	20
50	55	20	20	20	20	20	20	20	20	20
60	66	13	15	18	17	20	20	20	20	20
70	77	6	12	17	15	18	19	20	20	20
80	88	0	9	12	10	11	18	19	20	20

I Binary secrets

We report the hardest recovered binary secret, based on Hamming weight.

Highest Hamming weight and cruel bits recovered - binary secret.

Table 15: $n = 256, \log_2 q = 12$.

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	10/3	10/3	12/4	**14/5**	14/5
Synthetic data					
4M	10/3	10/3	10/3	**14/5**	14/5
20M	10/3	10/3	10/3	14/4	14/5
50M	10/3	10/3	12/3	14/5	-
100M	10/4	10/4	12/5	-	-
200M	12/5	12/5	12/5	-	-
400M	12/5	12/5	-	-	-

Best of 80 models.

Table 16: $n = 512, \log_2 q = 28$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	10/3	10/3	10/4	12/4	12/4
20M	10/4	10/4	10/5	**12/5**	12/5
50M	10/4	10/4	10/3	12/5	-
Synthetic data					
4M	10/3	10/3	10/3	10/3	12/4
20M	10/3	10/4	**12/5**	12/5	12/5
50M	10/3	10/4	12/4	12/4	-
100M	10/3	12/4	12/5	-	-
200M	12/4	12/4	12/4	-	-

Best of 80 models.

Table 17: $n = 256, \log_2 q = 20$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	55/6	55/6	55/7	60/8	65/8
20M	55/6	55/6	60/7	60/8	65/8
50M	60/8	65/8	65/8	65/8	-
100M	65/8	65/8	**70/8**	-	-
200M	65/8	70/8	70/8	-	-
400M	65/8	70/8	-	-	-
Synthetic data					
4M	55/6	60/7	60/7	60/7	60/8
20M	60/8	65/8	65/8	65/8	65/8
50M	65/7	65/8	**70/8**	70/8	-
100M	65/8	65/8	70/8	-	-
200M	65/8	65/8	70/8	-	-
400M	65/8	65/8	-	-	-

Best of 80 models.

Table 18: $n = 512, \log_2 q = 41$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	70/4	70/4	70/4	70/6	70/6
Synthetic data					
4M	70/4	70/4	70/4	70/5	70/6
20M	70/6	70/6	70/7	**75/7**	75/7
50M	70/6	70/6	70/6	75/7	-
100M	70/6	70/6	70/6	-	-
200M	70/6	70/6	-	-	-

Best of 80 models.

J Ternary secrets

We report the hardest recovered ternary secret, based on Hamming weight.

Highest Hamming weight and cruel bits recovered - ternary secret.

Table 19: $n = 256, \log_2 q = 12$.

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	10/3	10/3	10/3	10/3	10/4
Synthetic data					
4M	10/3	10/3	10/3	10/3	10/3
50M	10/4	10/4	10/4	10/4	-
200M	10/4	10/4	**12/5**	-	-
400M	10/4	10/4	-	-	-

Best of 80 models.

Table 20: $n = 512, \log_2 q = 28$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	8/3	8/3	8/4	8/4	8/4
50M	8/4	10/3	10/3	**10/4**	-
Synthetic data					
4M	8/3	8/3	8/3	8/3	8/4
50M	8/4	8/3	10/3	10/3	-
200M	10/3	10/3	**10/4**	-	-

Best of 80 models.

Table 21: $n = 256, \log_2 q = 20$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	40/5	40/5	45/5	45/5	45/5
50M	40/5	45/6	45/6	45/6	-
200M	50/6	**55/8**	55/8	-	-
400M	55/7	55/8	-	-	-
Synthetic data					
4M	40/5	40/5	40/5	45/5	45/5
50M	45/5	50/6	50/6	50/6	-
200M	45/5	50/6	50/6	-	-
400M	50/6	50/7	-	-	-

Best of 80 models.

Table 22: $n = 512, \log_2 q = 41$

	Repetition				
	1x	2x	5x	15x	50x
BKZ-reduced data					
4M	60/5	60/5	60/5	65/5	65/5
Synthetic data					
4M	60/5	60/5	60/5	65/5	65/5
50M	65/5	70/6	70/6	**75/7**	-
200M	70/6	70/6	-	-	-

Best of 80 models.