How do Small Transformer Models Learn Hard Math Tasks?

Eshika Saxena [1] **Kristin Lauter** [1]

Abstract

Recent works have demonstrated that transformers can be trained to recover sparse, binary cryptographic secrets in the Learning With Errors (LWE) problem, a foundational problem that underlies many post-quantum cryptographic schemes. However, as architectures have evolved to efficient encoder-only models, the mechanism by which these models recover the cryptographic secret has become more opaque. In this paper, we present the first layer-wise and embedding-level mechanistic interpretability analysis of encoder-only transformers trained on LWE samples. We reveal a surprising phenomenon: despite achieving near-zero exact prediction accuracy on the training objective, the models successfully recover the secret by bypassing the standard predictive pathways. We use dimensionality reduction, causal intervention, and linear probing and find that the secret is implicitly present in the positional embedding. Building on this mechanistic understanding, we introduce an architectural intervention that applies L_1 sparsity regularization directly to the positional embeddings. This modification forces the model to explicitly isolate the latent secret, transforming the computationally expensive post-hoc secret recovery process into a direct, human-interpretable parameter inspection. Our findings provide fundamental insights into how transformers allocate representational capacity when faced with high-noise, structured combinatorial problems.

1. Introduction

Recently, transformers have demonstrated remarkable capabilities in learning complex algorithmic and mathematical reasoning tasks, from modular arithmetic (Nanda et al., 2023; Saxena et al., 2025b) to symbolic integration (Barket et al., 2024). However, understanding how these models implement such algorithms internally remains a central challenge in mechanistic interpretability. While natural language provides a rich but noisy domain for interpretability research, structured mathematical problems offer a highly controlled, mathematically verifiable environment to reverse-engineer model representations (Gromov, 2023; He et al., 2024; Nanda et al., 2023).

In this paper, we investigate the internal learning dynamics of transformers on a fundamentally hard mathematical challenge: the Learning With Errors (LWE) problem. LWE is a foundational hard mathematical problem underpinning many post-quantum cryptographic (PQC) schemes (Chen et al., 2022). The security of these schemes relies on the difficulty of recovering a binary secret vector $\mathbf{s} \in \{0,1\}^N$ given a large collection of noisy linear samples $(\mathbf{a}, b)$, where $\mathbf{a} \in \mathbb{Z}_q^N$ is a publicly known random vector of length N, $b = \mathbf{a} \cdot \mathbf{s} + e \pmod q$ is a noisy dot product integer, e is a small Gaussian error term, and q is a large prime modulus. The attacker's goal is to recover $\mathbf{s}$ from many such pairs, knowing only $\mathbf{a}$ and b. Recent works, such as SALSA (Wenger et al., 2022), PICANTE (Li et al., 2023b), VERDE (Li et al., 2023a), and FRESCA (Stevens et al.), have demonstrated that transformers can be trained on these $(\mathbf{a}, b)$ pairs to predict b from $\mathbf{a}$. Once trained, the model is not used as a predictor directly; instead, it is queried as an oracle by a post-hoc *distinguisher* algorithm that probes the model's sensitivity to perturbations of individual entries of $\mathbf{a}$ to identify which positions correspond to active (nonzero) bits of $\mathbf{s}$ (Wenger et al., 2022; Li et al., 2023b;a). These attacks have successfully recovered binary secrets in certain parameter settings (N, q, and sparse $\mathbf{s}$). However, as these architectures have evolved from sequence-to-sequence models to efficient encoder-only architectures, the interpretability of how the secret is learned has diminished.

In our analysis, we found that a key mystery emerges when analyzing these state-of-the-art encoder-only models: they successfully recover the cryptographic secret vector $\mathbf{s}$ despite exhibiting near-zero exact prediction accuracy on the training objective (predicting b from $\mathbf{a}$). This suggests the model is not learning a generalized algorithmic pathway to compute the noisy dot product, but rather exploiting an alternative representational mechanism to solve the task.

This paper aims to open the black box of these models to

[1]FAIR at Meta. Correspondence to: Eshika Saxena <eshika@meta.com>, Kristin Lauter <klauter@meta.com>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

understand how transformers allocate representational capacity when standard predictive pathways fail. By applying mechanistic interpretability techniques, including embedding analysis, logit attribution, dimensionality reduction, causal intervention, and linear probing, we trace the flow of information through the network. We reveal a surprising phenomenon: the model bypasses the standard prediction mechanism and instead encodes the structure of the secret $\mathbf{s}$ directly within its positional embeddings.

Our findings provide fundamental insights into how transformers handle high-noise, structured combinatorial problems, demonstrating that models can develop "hidden" representational pathways to solve tasks. In summary, our key contributions are:

- **Mechanistic Analysis of LWE:** We provide the first layer-wise and embedding-level mechanistic interpretability analysis of encoder-only transformers trained on cryptanalysis tasks.
- **Discovery of Positional Memorization:** We reveal that under high-noise algorithmic tasks, transformers can memorize latent state variables directly within positional embeddings.
- **Sparse Positional Regularization:** We propose an architectural modification, applying L_1 sparsity regularization to positional embeddings, that forces the model to yield highly interpretable, easily extractable representations of the secret.

2. Related Work

Transformers on Algorithmic Tasks. While transformers were originally designed for natural language processing (Vaswani et al., 2017), recent work has demonstrated their surprising capacity to learn complex mathematical and algorithmic operations from examples alone. Charton (Charton, 2022) showed that transformers can learn linear algebra operations, including eigenvalue decomposition and matrix inversion, generalizing well beyond their training distributions. Similarly, transformers have been successfully applied to symbolic integration, differential equations, and modular arithmetic (Agarwal et al., 2021; Alfarano et al., 2024; Barket et al., 2024; Charton, 2024b;a; Kera et al., 2024; Lample & Charton, 2020; Nanda et al., 2023; Saxena et al., 2025b). Our work builds on this foundation by analyzing transformer behavior on Learning With Errors (LWE), a fundamentally hard combinatorial and linear algebraic problem over finite fields.

Mechanistic Interpretability and Grokking. Understanding how neural networks implement learned computations internally is the primary goal of mechanistic interpretability (MI) (Rai et al., 2024). A foundational framework for this analysis was introduced by Elhage et al. (Elhage et al., 2021), who conceptualized the transformer residual stream as a communication channel where independent attention heads read and write specific features. This "circuits" approach has been used to reverse-engineer specific model behaviors, such as the "induction heads" responsible for in-context learning (Olsson et al., 2022).

Of particular relevance to our work is the study of "grokking" on algorithmic tasks (Furuta et al., 2024; Nanda et al., 2023; Gromov, 2023). Nanda et al. (Nanda et al., 2023) fully reverse-engineered the mechanism by which small transformers learn modular addition, discovering that models develop discrete Fourier transforms and trigonometric identities within their weights. Furthermore, recent advances in MI have focused on feature extraction using sparse autoencoders to disentangle polysemantic representations into monosemantic, human-interpretable features (Cunningham et al., 2023). Inspired by these approaches, we apply MI techniques such as logit attribution and dimensionality reduction to cryptographic models, and we introduce a sparsity-based architectural intervention based on our findings.

Machine Learning Attacks on LWE. The application of transformers to LWE cryptanalysis began with the SALSA attack (Wenger et al., 2022), which demonstrated that a one-layer encoder-decoder transformer could learn the underlying structure of LWE problems. In these early sequence-to-sequence models, the secret recovery mechanism was relatively straightforward: researchers could read the secret bits directly by analyzing the cross-attention weights between the encoder and the decoder (Li et al., 2023b). Subsequent works, such as PICANTE (Li et al., 2023b) and VERDE (Li et al., 2023a), improved upon this by incorporating lattice reduction preprocessing (e.g. BKZ (Schnorr, 1987) and Flatter (Ryan & Heninger, 2023)) to reduce the variance of the data, allowing the models to recover secrets in much larger dimensions (up to $N = 1024$) (Stevens et al.; Wenger et al., 2025; Saxena et al., 2025a). These works also introduced distinguishers as a mechanism for secret recovery, where the trained model was queried with carefully designed inputs to determine the secret (Wenger et al., 2022; Li et al., 2023b;a).

More recent advancements, such as the FRESCA (Stevens et al.) architecture, Wenger et al. (Wenger et al., 2025), and Alfarano et al. (Alfarano et al., 2026), have shifted towards encoder-only models. These models are significantly faster and computationally less complex, as they do not require an auto-regressive decoder. Instead, the outputs are max-pooled across the sequence dimension and decoded by a linear layer. While encoder-only models are more efficient, they lack the cross-attention mechanism, making it unclear where exactly the secret is learned or encoded within the

model's weights. Our work resolves this ambiguity by analyzing the internal representations of these state-of-the-art encoder-only architectures.

3. Experimental Setup

Preprocessed Data. Following the methodology introduced by PICANTE (Li et al., 2023b) and improved in VERDE (Li et al., 2023a), we use LWE data preprocessed via lattice reduction algorithms (BKZ (Schnorr, 1987) and Flatter (Ryan & Heninger, 2023)) to reduce the variance of the samples. As noted by Nolte et al. (Nolte et al., 2024), applying lattice reduction produces a "cliff"-shaped asymmetry in the variance across the columns of the reduced matrix (we take the rows of this matrix as the input vectors for training). The early columns remain largely unreduced and retain high variance (the "cruel" region) with a steep dropoff as the remaining columns experience significantly reduced variance (the "cool" region). Consequently, the input vectors $\mathbf{a}$ that the model receives are not uniformly random; they exhibit this structured asymmetry. The exact boundary between these regions depends heavily on the LWE parameters, such as the lattice dimension (N) and modulus size (q). Prior work has demonstrated that transformers struggle to recover secrets if more than three non-zero entries secret fall in the cruel region (Wenger et al., 2025). To ensure our findings are robust across the critical threshold of learnability, we focus on the $N = 256, \log_2 q = 20$ setting. We utilize the preprocessed, open-sourced TAPAS datasets for LWE (Saxena et al., 2025a)[1]. We randomly generate 30 binary secrets with Hamming weights (number of non-zero entries) for which the models are able to consistently recover the secrets (10 secrets each for $h \in \{20, 21, 22\}$), rather than the maximum possible Hamming weight that can be recovered by the model.

Model Architecture and Training. We adopt the encoder-only architecture and angular embedding representation proposed by FRESCA (Stevens et al.) and improved by Alfarano et al. (Alfarano et al., 2026), shown in Figure 1. Following FRESCA (Stevens et al.), we map the integer entries in the input $\mathbf{a}$ to the unit circle using angular embeddings: $x \mapsto (\cos(2\pi x/q), \sin(2\pi x/q))$. The model processes the sequence of angular embeddings through standard transformer encoder layers (4 layers, 4 heads per layer, and hidden dimension of 256). We obtain the final model output by max-pooling the encoder's output sequence and projecting the resulting vector into $\mathbb{R}^2$. To calculate prediction accuracy, we convert this 2D point back into the integer space $\mathbb{Z}_q$ and compare it to the ground truth b.

We train one model per unique secret in our set of 30 randomly generated secrets with hamming weights $h \in \{20, 21, 22\}$ (10 per hamming weight). We train each model for 50 epochs using ∼1 million samples on a single NVIDIA H100 or H200 GPU, reserving 1,000 samples as a held-out test set for our analysis in Section 4. The training time for each model is approximately 2.5 hours.

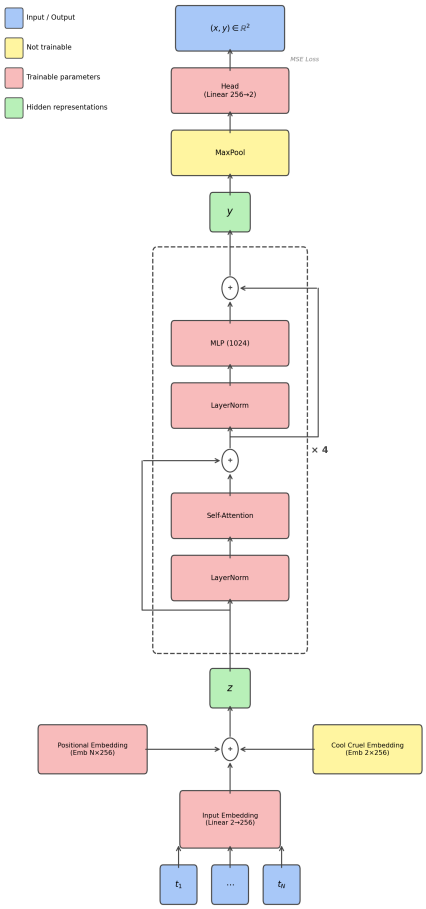

Figure 1. Encoder-only Architecture

We base our implementation on the open-sourced LWE-benchmarking repository[2] from Wenger et al. (Wenger et al., 2025), and we provide a comprehensive list of all model hyperparameters in Appendix A.

Distinguisher Algorithm. In ML attacks on LWE, the model is trained to predict b from $\mathbf{a}$, but the ultimate goal is to recover the secret vector $\mathbf{s}$ (Wenger et al., 2022). Secret recovery relies on a post-training inference technique known as a distinguisher (Li et al., 2023a; Stevens et al.).

The distinguisher exploits the trained model's sensitivity to

[1] https://huggingface.co/datasets/facebook/TAPAS (released under the CC-by-4.0 license)

[2] https://github.com/facebookresearch/LWE-benchmarking (released under the CC-by-NC license)

input perturbations. For each position $i \in 0, \ldots, N-1$, the distinguisher constructs two input vectors, $\mathbf{a}$ and $\mathbf{a}'$, that are identical except at index i. It then queries the model to obtain predictions $\hat{b}$ and $\hat{b}'$. The intuition is straightforward: if the true secret bit $\mathbf{s}_i = 0$, the model's prediction should be insensitive to changes at position i, yielding a small difference $|\hat{b} - \hat{b}'|$. Conversely, if $\mathbf{s}_i \neq 0$, perturbing a_i should significantly alter the predicted b, yielding a large difference.

By computing this sensitivity score for all N positions, the distinguisher ranks the indices and selects the top-h positions as the candidate secret ($\mathbf{s}^*$). The candidate is then verified statistically by computing the standard deviation of the residuals $b - \mathbf{a} \cdot \mathbf{s}^*$ over the initial LWE samples. If the standard deviation is approximately equal to the known error distribution σ_e, the secret is successfully recovered. A central goal of our interpretability analysis is to understand why the model develops this specific input-output sensitivity.

4. Methods and Results

To understand how encoder-only models learn LWE and where the secret resides, we apply a suite of mechanistic interpretability techniques. Our analysis progresses from behavioral observations to internal representation tracking across our 30 trained models. We provide representative figures from one model run with $h = 20$, and the tables quantify the trends in the figures across all 30 models. Finally, we propose and implement a targeted architectural intervention based on our findings.

4.1. Model Prediction Accuracy

Standard machine learning models are evaluated on their prediction accuracy. However, we observe a striking paradox in ML attacks on LWE: the model's exact prediction accuracy (where the predicted integer exactly matches the ground truth b) is 0%, yet the model can successfully recover the secret. As shown in Table 1, standard performance metrics like mean squared error (MSE) and mean absolute error (MAE) are comparable between models that recover the secret and those that do not. While exact accuracy is zero across all models, applying a relaxed metric with a 15% margin of error yields approximately 32% accuracy for models that fail to recover the secret, and 38% for those that succeed. We introduce this 15% margin to determine if the models' predictions approximate the true values. Given that random guessing under this relaxed metric would achieve 30% accuracy, both model types perform only marginally better than chance. Furthermore, our analysis found that the error distribution is roughly uniform rather than zero-centered. These findings provide no behavioral evidence that the models have learned the underlying modular arithmetic of the Learning with Errors (LWE) problem.

Table 1. **Model performance of models with successful secret recovery is only marginally better compared to models that do not recover** ($N = 256$, $\log_2 q = 20$, $h \in \{20, 21, 22\}$, 30 secrets); values report mean $\pm$ standard deviation. *Eval Loss* is mean squared error (MSE) computed in the angular space (↓ better). *MAE/q* is mean absolute error normalized by the modulus q (↓ better). *% Acc (15% margin)* is the fraction of predictions within 15% of the true value; random guessing achieves approximately 30% under this metric (↑ better). Bold entries indicate the better result in each column.

Status	Recovered	Not recovered
# Models	13	17
Eval Loss (↓)	$\mathbf{1.04 \pm 0.05}$	1.08 ± 0.04
MAE / q (↓)	$\mathbf{0.22 \pm 0.01}$	0.24 ± 0.01
% Acc (15% margin) (↑)	$\mathbf{38.2 \pm 3.0}\%$	$32.2 \pm 3.1\%$

Despite this apparent failure to predict b accurately, applying the perturbation-based distinguisher (see Section 3) successfully recovers the secret in many of these cases. This indicates that the model is not learning a generalized algorithmic solution to LWE, but rather memorizing the specific secret vector $\mathbf{s}$ in a way that makes its outputs highly sensitive to targeted input perturbations. The secret is encoded in the model's internal representations, but does not manifest itself as better predictions. In the subsequent sections, we open the black box to locate exactly where this memorization occurs.

4.2. Final Embedding Analysis

To determine if/how the model implicitly represents the secret, we first examine the final hidden representations before the prediction head. We extract the model's embeddings for a held-out test set of 1,000 samples. These embeddings represent the transformer's final representations of each position j for each sample k, prior to the max-pooling operation.

We compute the mean embedding per position across the 1,000 samples and apply Principal Component Analysis (PCA) for dimensionality reduction and easy visualization. As shown in the bottom panel of Figure 2, we see two outliers which correspond to the two positions that are both in the secret and in the "cruel" (unreduced) region of the preprocessed data.

To quantify the structure observed in the plot, we compute several metrics to assess what information is encoded in the embeddings. Specifically, we train two linear probes (logistic regression models) (Alain & Bengio, 2016) on the embeddings using 5-fold cross-validation: one to predict whether each index corresponds to a nonzero entry of the secret and one to classify each index into one of four labels—secret+cruel, secret+cool, non-secret+cruel, and non-secret+cool. We also compute the centroid separation ratio for both the two-cluster and four-cluster configurations,

defined as the ratio of inter-cluster Euclidean distance to intra-cluster Euclidean distance. In Table 2, we report the probe accuracy and separation ratios. The higher probe accuracy in recovered models demonstrates linearly separable structure in the embeddings, and the higher separation ratios imply more compact, well-separated clusters (especially in the 4 cluster configuration).

Table 2. **Final embedding clustering is stronger in models that recover the secret across all 30 models** ($N = 256$, $\log_2 q = 20$, $h \in \{20, 21, 22\}$); values report mean $\pm$ standard deviation. *Probe Acc.* is the 5-fold cross-validated accuracy of a linear probe trained to classify embeddings into 2 clusters (secret vs. non-secret indices) or 4 clusters (secret+cool, secret+cruel, non-secret+cool, non-secret+cruel); higher accuracy indicates more linearly separable structure. *Sep. Ratio* is the ratio of inter-cluster to intra-cluster distance; higher values indicate more compact, well-separated clusters. Bold entries indicate the better result in each column.

Status		Recovered	Not recovered
# Models		13	17
2 Clusters	**Probe Acc. %**	$\mathbf{93.0 \pm 2.6}\%$	$89.5 \pm 2.0\%$
	Sep. Ratio	$\mathbf{0.56 \pm 0.20}$	0.28 ± 0.09
4 Clusters	**Probe Acc. %**	$\mathbf{90.3 \pm 3.7}\%$	$84.9 \pm 2.4\%$
	Sep. Ratio	$\mathbf{2.34 \pm 0.43}$	1.24 ± 0.85

The generally improved separation between secret and non-secret indices in the recovered models confirms that the model internally distinguishes the secret. The secondary separation between the "cool" and "cruel" regions stems directly from the variance asymmetry introduced by the lattice reduction (Nolte et al., 2024). Because the cruel indices (the early columns of the reduced inputs) retain significantly higher variance, the model is forced to scale its representations differently for these positions to minimize the training loss. This structural feature of the preprocessed data essentially provides the model with a strong signal about the relative importance of different input dimensions.

Next, we explore how the model uses these embeddings to inform its predictions.

Logit Attribution. For each of the 256 hidden dimensions, we identify which position index holds the maximum value, tallying these "wins" across all dimensions and samples. However, since the model max-pools across positions before the prediction head, for each hidden dimension exactly one position "wins." Winning the max isn't equally important across all hidden dimensions as some dimensions matter more for the final prediction than others because the model's prediction head $\mathbf{W} \in \mathbb{R}^{2 \times d_{\text{hidden}}}$ assigns different importance to different dimensions. To account for this, we compute a weighted attribution score. Let $\mathbf{E} \in \mathbb{R}^{N \times d_{\text{hidden}}}$ be the embeddings before max-pooling, and let $p_d = \arg\max_i \mathbf{E}_{i,d}$ be the "winning" position index (i.e. the argmax) for hidden dimension d. We define the contribution of position p_d to the final logits as the magnitude of the activation scaled by the norm of the corresponding column in the prediction head:

$$\text{Attribution}(p_d) = ||\mathbf{W}_{:,d}|| \cdot |\mathbf{E}_{p_d,d}|$$

By summing these attribution scores across all dimensions $d \in 1, \ldots, d_{\text{hidden}}$ for each position, we obtain a more accurate measure of which input positions drive the final prediction. As shown in Table 3, the secret positions not only win the max-pooling operation more frequently, but they specifically dominate the hidden dimensions that matter most for the final output. Thus, the model is using the secret to predict b.

Table 3. **Logit attribution shows that models that recovered rely more heavily on secret positions when producing their predictions** ($N = 256$, $\log_2 q = 20$, $h \in \{20, 21, 22\}$, 30 secrets); values report mean $\pm$ standard deviation. *Attribution* measures the mean absolute logit contribution of positional embedding weights corresponding to secret indices ($\mathbf{s}_i = 1$) and non-secret indices ($\mathbf{s}_i = 0$); higher values indicate greater influence on the model's output. *Attribution Ratio* is the ratio of secret-index attribution to non-secret-index attribution; a higher ratio indicates that the model disproportionately relies on secret positions when producing its predictions. Bold entry indicates higher attribution ratio.

Status	Recovered	Not recovered
# Models	13	17
Attribution ($\mathbf{s}_i = 0$)	0.03 ± 0.01	0.07 ± 0.02
Attribution ($\mathbf{s}_i = 1$)	0.16 ± 0.05	0.12 ± 0.07
Attribution Ratio ($\mathbf{s}_i = 1/\mathbf{s}_i = 0$)	$\mathbf{5.35 \pm 2.39}$	2.32 ± 2.31

We also look at the results from the argmaxes of the embeddings without weighting by the prediction head norm and the norms of the embeddings for each index. In both cases, we see higher values on average for the secret indices, suggesting that the secret signal is in the embeddings themselves rather than the prediction head weights. We present these results in more detail in Appendix B.

4.3. Residual Stream Analysis

Having established that the final embeddings seem to have some understanding of the secret, we trace this representation backward through the network and analyze the residual stream (Elhage et al., 2021). We extract the mean per-position embeddings at the output of the input layer and after each of the four transformer blocks. Applying PCA to each stage (Figure 2), we observe a surprising result: the four distinct clusters (secret/non-secret, cool/cruel) emerge *immediately after the input layer block*, before any self-attention mechanisms are applied. The subsequent transformer layers actually obfuscate these clusters. This finding strongly suggests that the model's positional embeddings are the primary mechanism for storing secret information.

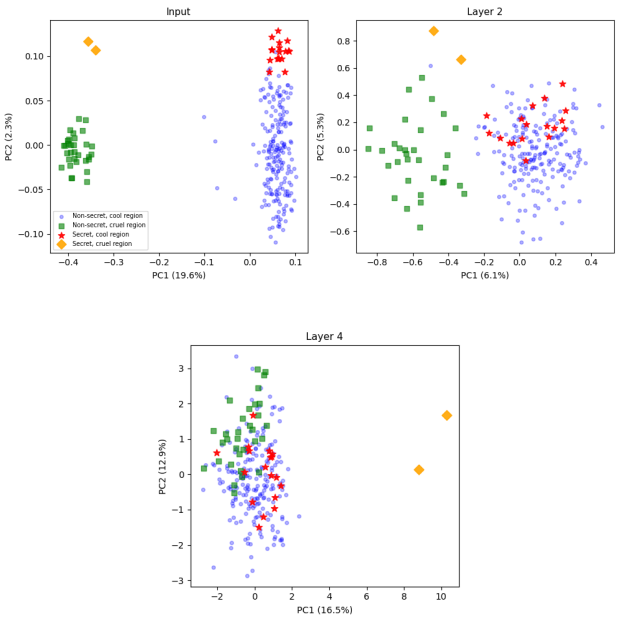

Figure 2. **Layer-wise PCA of mean per-position residual stream representations at input layer, second layer, and fourth (final) layer.** Secret and non-secret indices are clearly separated along PC1 (19.6% variance explained) at the input layer, prior to any self-attention computation. The full layer-by-layer progression is shown in Appendix C.

4.4. Positional Embedding Weight Analysis

Based on the layer-wise analysis, we isolate the learned positional embedding weight matrix (dimension $N = 256 \times d_{\text{hidden}} = 256$). We apply PCA directly to the rows of this weight matrix (Figure 3). We see a distinct cluster of secret indices to the right along PC1 (3.0% variance explained), providing geometric evidence that the model encodes secret-relevant structure specifically in the positional embeddings of the secret support.

To quantify this structure, we compute silhouette scores and centroid separation ratios for both the secret/non-secret clustering and the four-cluster configuration incorporating the cruel/cool distinction (see Table 4). The silhouette score measures how similar a point is to its own cluster relative to other clusters, ranging from -1 to 1, where higher values indicate more cohesive and well-separated clusters. Unlike the final embedding analysis, we do not train linear probes here, as the positional embedding weights are fixed (small sample size), and the sparsity of the secret leads to class imbalance. Although the metrics do not indicate strong clustering for any of the models, the recovered models consistently exhibit stronger cluster structure than the unsuccessful models across all metrics, reflected in higher silhouette scores and separation ratios.

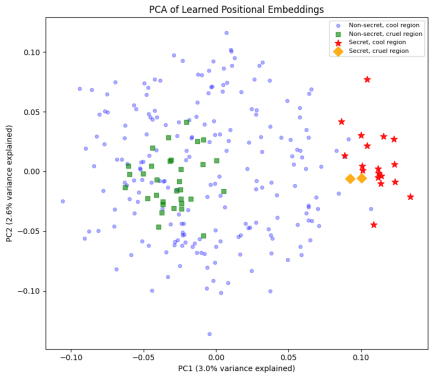

Figure 3. **PCA of the learned positional embedding vectors.** Secret indices cluster distinctly to the right along PC1 (3.0% variance explained), while non-secret indices occupy a largely overlapping central cloud.

Table 4. **Positional embedding weight clustering is consistently stronger in recovered models** ($N = 256$, $\log_2 q = 20$, $h \in \{20, 21, 22\}$, 30 secrets); values report mean $\pm$ standard deviation. Clustering is evaluated over 2 clusters (secret vs. non-secret indices) and 4 clusters (secret+cool, secret+cruel, non-secret+cool, non-secret+cruel). *Silhouette Score* measures how similar each weight is to others in its assigned cluster relative to the nearest other cluster, ranging from -1 to 1 where higher values indicate more cohesive and well-separated clusters. *Sep. Ratio* is the ratio of inter-cluster to intra-cluster distance, where higher values indicate greater separation. Bold entries indicate the better result in each column.

		Recovered	Not recovered
# Models		13	17
2 Clusters	**Silhouette**	$\mathbf{0.0264 \pm 0.0130}$	0.0076 ± 0.0108
	Sep. Ratio	$\mathbf{0.43 \pm 0.06}$	0.28 ± 0.09
4 Clusters	**Silhouette**	$\mathbf{0.0034 \pm 0.0125}$	-0.0015 ± 0.0087
	Sep. Ratio	$\mathbf{0.81 \pm 0.21}$	0.52 ± 0.11

Causal Intervention. To test whether the transformer's positional embeddings at secret positions are *necessary* for solving the LWE distinguishing task, we perform a causal intervention. Specifically, we zero out the learned positional embedding vectors at the h indices where the secret is 1 and re-run the perturbation-based distinguisher, comparing against a control where the same number (h) of non-secret positions are ablated. The results are in Table 5. When secret index embeddings are zeroed out, the distinguisher's sensitivity scores (total modular difference between the model's predictions on the original inputs and on inputs with position i perturbed) collapse and secret recovery fails as the number of correctly recovered bits drops sharply (top of Figure 4). In contrast, ablating an equal number of non-secret positions has minimal effect: the model still recovers the secret with nearly baseline performance (bottom of Figure 4). This confirms that the model has learned to encode secret-relevant information specifically in the positional embeddings of the secret support, rather than relying on a distributed or

redundant representation across all positions.

Table 5. **Causal intervention confirms that positional embeddings at secret positions are necessary for secret recovery** ($N = 256$, $\log_2 q = 20$, $h \in \{20, 21, 22\}$, 30 secrets); values report mean $\pm$ standard deviation. *Baseline* reports performance with no ablation. *Ablate Secret* reports performance after zeroing out the learned positional embedding vectors at the h secret indices ($\mathbf{s}_i = 1$). *Ablate Non-secret* reports performance after zeroing out h randomly selected non-secret positions as a control. The sharp drop under "Ablate Secret" compared to the near-baseline performance under "Ablate Non-secret" confirms that the model encodes secret-relevant information specifically at the secret positions rather than in a distributed representation.

Status	Recovered	Not recovered
# Models	13	17
Baseline	19.4 ± 2.1	6.5 ± 6.9
Ablate Secret	0.8 ± 2.8	0.0 ± 0.0
Ablate Non-secret	18.7 ± 3.6	6.31 ± 7.2

Figure 4. **Ablating secret positions causes large negative drops in sensitivity scores concentrated at the secret indices (top), whereas ablating an equal number of non-secret positions produces negligible changes across all positions (bottom).** The sensitivity score at position i is computed by the perturbation-based distinguisher as the total modular difference between the model's predictions on the original inputs and on inputs with position i perturbed; a higher score indicates that the model's output is more sensitive to that position. Each bar shows the change in sensitivity score (ablated $-$ baseline) at each position index; gray bands mark the true secret positions ($\mathbf{s}_i = 1$).

This causal intervention confirms our central mechanistic finding: when faced with the fundamentally hard mathematical task of LWE, an encoder-only transformer bypasses the intended predictive pathway. Instead, it leverages its capacity to overfit by encoding the non-zero indices of the secret vector directly into the magnitudes and directions of its positional embeddings. Because the secret information is explicitly isolated in these embeddings, we hypothesize that we can redesign the architecture to make this extraction process transparent by default.

4.5. Architectural Intervention: Factored, Sparse Positional Embedding

Given this understanding that the secret is present in the positional embeddings, we aim to make architectural changes that optimize for secret recovery by enabling us to directly recover the secret from the model. In standard transformer architectures, positional embeddings are primarily responsible for injecting sequence order information into the permutation-invariant self-attention mechanism (Vaswani et al., 2017). However, when these embeddings are fully learnable rather than fixed sinusoids, empirical studies have shown they can absorb task-specific structural biases beyond mere absolute position (Wang & Chen, 2020). In our LWE setting, the model exploits this capacity to encode the binary secret vector $\mathbf{s}$ directly into the spatial geometry of the embedding space. Inspired by the use of L_1 regularization in sparse autoencoders (SAEs) to force interpretable feature disentanglement (Cunningham et al., 2023), we redesign the positional embedding to match the prior structure of the LWE secrets we are targeting in this work (sparse, N-long, binary vectors). To do this, we decouple the magnitude of the embedding from its directional representation, a reparameterization technique related to recent low-rank adaptation methods (Liu et al., 2024). We decompose each positional embedding into a learnable scalar magnitude α_i and a unit-norm direction vector $\mathbf{v}_i$:

$$\text{pos_emb}[i] = \alpha_i \times \mathbf{v}_i, \quad i \in \{0, \ldots, N-1\}$$

To enforce the prior that the secret is sparse, we apply an L_1 sparsity penalty to the α scalars during training. Specifically, we sort the α values, sum all but the top-h values, and add this sum to the training loss multiplied by a hyperparameter $\beta = 0.01$. This encourages the model to drive the importance of non-secret positions to zero, cleanly isolating the secret indices in the top-h magnitudes of α. We can write the training loss as:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{MSE}} + \beta \sum_{i \in \mathcal{S}_h^c} |\alpha_i|, \beta = 0.01$$

where $\mathcal{L}_{\text{MSE}}$ is the standard MSE loss, $\mathcal{S}_h$ denotes the index set of the top-h values of $|\boldsymbol{\alpha}|$, and $\mathcal{S}_h^c = 0, \ldots, N-1 \setminus \mathcal{S}_h$ is its complement (i.e., all non-top-h positions).

We then train the model with this new positional embedding structure and sparsity loss regularization using the same setup and parameters as before. The top panel of Figure 5 shows that the learned α values do not place all secret indices in the top-h ranks. Instead, the top-h positions are dominated by a small cluster of non-secret "cruel" (unreduced) indices, which the model consistently assigns larger α values regardless of secret membership—a direct consequence of the high variance introduced by lattice reduction at those positions. The secret indices instead cluster between the top-h and top-$2h$ cutoffs, indicating that the model has learned to distinguish secret from non-secret positions, but that the cruel-position signal is stronger than the sparsity prior imposed by L_1 regularization. The concentration of

secret indices within the top-h to $2h$ window still enables recovery from direct parameter inspection. The bottom panel further shows that this structure emerges rapidly, within the first $\sim$10 epochs, with secret indices (red) decaying more slowly and converging to higher α values than most of the non-secret indices (blue).

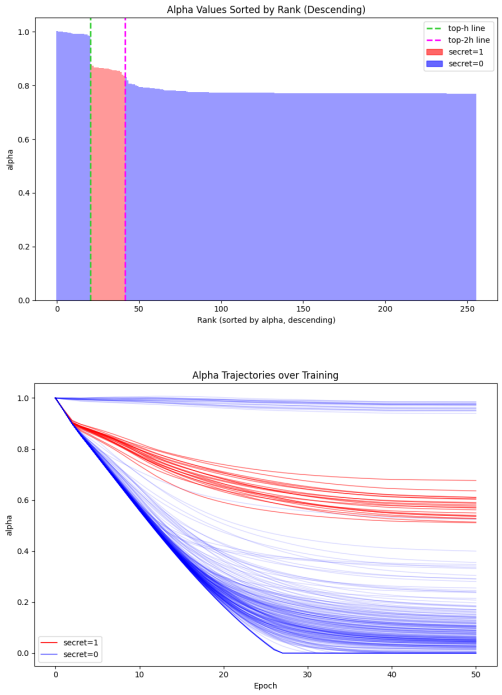

Figure 5. **Learned L_1 sparsity weights (α) for model trained with the factored, sparse positional embedding.** *Top*: α values sorted in descending order by rank, with bars colored by secret membership (red: $\mathbf{s}_i = 1$, blue: $\mathbf{s}_i = 0$). *Bottom*: Trajectories of individual α values over training epochs, colored by secret membership.

5. Conclusion

In this paper, we utilized the LWE problem not merely as a cryptographic target, but as a case study for understanding how transformers allocate representational capacity. By applying mechanistic interpretability techniques, we opened the black box of ML-based LWE attacks, revealing that encoder-only models bypass algorithmic reasoning to embed the secret directly in their positional embeddings. By leveraging this insight to design an interpretable architecture, we offer a new technique for extracting structured information from trained transformers. As LWE-based schemes underpin emerging post-quantum cryptography standards, understanding the mechanisms by which ML attacks succeed helps contribute to the long-term security assessment of these constructions.

Limitations and Future Work. While our work provides novel mechanistic insights into how encoder-only transformers can recover LWE secrets, some limitations and directions for future work remain. First, our empirical evaluation and analysis is restricted to settings where the SALSA-family attack already succeeds, using sparse binary secrets at Hamming weights near but below the learnability threshold—parameter regimes that do not yet reflect deployed cryptographic systems. Relatedly, the L_1 regularization on the positional embedding scalars is designed to be most effective when the secret is sparse and provides diminishing benefit as the secret becomes denser (at which point penalizing the non-secret positions would be more natural). Future work could explore whether alternative priors or regularization schemes can extend the learnability threshold to denser secrets. Second, our analysis relies on data preprocessed via lattice reduction, which introduces the variance asymmetry (the "cool"/"cruel" regions) that the model clearly exploits. This asymmetry also constrains recovery when many secret bits fall in the "cruel" region (Wenger et al., 2025). Future work could investigate whether a different form of structured memorization emerges on unreduced data or on other LWE settings, and whether such structures could inform improved architecture designs.

Impact Statement

This paper presents work whose goal is to advance the fields of machine learning interpretability and cryptanalysis. Our mechanistic analysis of ML-based attacks on the Learning with Errors (LWE) problem contributes to the long-term security assessment of post-quantum cryptographic schemes. Our work is restricted to parameter regimes that do not threaten deployed systems.

References

Agarwal, V., Aditya, S., and Goyal, N. Analyzing the nuances of transformers' polynomial simplification abilities, 2021. URL https://arxiv.org/abs/2104.14095.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Alfarano, A., Charton, F., and Hayat, A. Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. In *Proc. of NeurIPS*, 2024.

Alfarano, A., Saxena, E., Wenger, E., Charton, F., and Lauter, K. Improving ml attacks on lwe with data repetition and stepwise regression. *arXiv preprint arXiv:2604.03903*, 2026.

Barket, R., Shafiq, U., England, M., and Gerhard, J. Trans-

formers to predict the applicability of symbolic integration routines, 2024. URL https://arxiv.org/abs/2410.23948.

Charton, F. Linear algebra with transformers. *Transactions in Machine Learning Research*, 2022.

Charton, F. Can transformers learn the greatest common divisor? *arXiv:2308.15594*, 2024a.

Charton, F. Learning the greatest common divisor: explaining transformer predictions. In *ICLR*, 2024b.

Chen, L., Moody, D., Liu, Y.-K., 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.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Furuta, H., Minegishi, G., Iwasawa, Y., and Matsuo, Y. Interpreting grokked transformers in complex modular arithmetic. *arXiv preprint arXiv:2402.16726*, 2024.

Gromov, A. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.

He, T., Doshi, D., Das, A., and Gromov, A. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. *Advances in Neural Information Processing Systems*, 37:13244–13273, 2024.

Kera, H., Ishihara, Y., Kambe, Y., Vaccon, T., and Yokoyama, K. Learning to compute gröbner bases. *Advances in Neural Information Processing Systems*, 37: 33141–33187, 2024.

Lample, G. and Charton, F. Deep learning for symbolic mathematics. In *Proc. of ICLR*, 2020.

Li, C., Wenger, E., Allen-Zhu, Z., Charton, F., and Lauter, K. SALSA VERDE: a machine learning attack on Learning With Errors with sparse small secrets. In *Proc. of NeurIPS*, 2023a.

Li, C. Y., Sotáková, J., Wenger, E., Malhou, M., Garcelon, E., Charton, F., and Lauter, K. Salsa Picante: A Machine Learning Attack on LWE with Binary Secrets. In *Proc. of ACM CCS*, 2023b.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Nolte, N., Malhou, M., Wenger, E., Stevens, S., Li, C., Charton, F., and Lauter, K. The cool and the cruel: separating hard parts of LWE secrets. *Proc. of AFRICACRYPT*, 2024.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Rai, D., Zhou, Y., Feng, S., Saparov, A., and Yao, Z. A practical review of mechanistic interpretability for transformer-based language models. *CoRR*, abs/2407.02646, 2024. URL http://arxiv.org/abs/2407.02646.

Ryan, K. and Heninger, N. Fast practical lattice reduction through iterated compression. *Cryptology ePrint Archive*, 2023.

Saxena, E., Alfarano, A., Charton, F., Wenger, E., and Lauter, K. E. TAPAS: Datasets for learning the learning with errors problem. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025a. URL https://openreview.net/forum?id=91scW3DywW.

Saxena, E., Alfarano, A., Wenger, E., and Lauter, K. E. 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*, 2025b. URL https://openreview.net/forum?id=1e8hVvWi6Q.

Schnorr, C.-P. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 1987. URL https://www.sciencedirect.com/science/article/pii/0304397587900648.

Stevens, S., Wenger, E., Li, C. Y., Nolte, N., Saxena, E., Charton, F., and Lauter, K. Salsa fresca: Angular embeddings and pre-training for ml attacks on learning with errors. https://eprint.iacr.org/2024/150.

Vaswani, A., Shazeer, N., Parmar, N., et al. Attention is all you need. In *Proc. of NeurIPS*, 2017.

Wang, Y.-A. and Chen, Y.-N. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6840–6849, 2020.

Wenger, E., Chen, M., Charton, F., and Lauter, K. E. Salsa: Attacking lattice cryptography with transformers. *Proc. of NeurIPS*, 2022.

Wenger, E., Saxena, E., Malhou, M., Thieu, E., and Lauter, K. Benchmarking attacks on learning with errors. *Proc. of Oakland S& P*, 2025.

A. Hyperparameter Details

We list all hyperparameters used for the experiments in Table 6.

Table 6. Hyperparameters for experiments.

Parameter	**Value**
LWE Problem	
N	256
q	842779
σ	3.0
Secret type	Binary
Hamming weight	Varies (20–22)
Data	
Data budget	$\sim$1M
Batch size	256
Test size	1000
Reduced	True
Model Architecture	
Backbone	Transformer
Hidden dim	256
Encoder layers	4
Encoder heads	4
MLP expansion	$4\times$
Activation	GELU
Pooling	Max
Input embedding	Angular
Positional embedding	Standard / Factored
Attention dropout	0.0
MLP dropout	0.0
Optimization	
Optimizer	AdamW
Learning rate	5×10^{-6}
Min learning rate	5×10^{-7}
Weight decay	0.03
β_1, β_2	0.9, 0.98
ϵ	10^{-8}
Gradient clipping	5.0
Warmup steps	1000
Scheduler	Cosine annealing
Mixed precision (AMP)	True
Training	
Epochs	50
Steps per epoch	10000
Loss	MSE / MSE + sparsity regularization
Model seed	51

B. Argmax and Norm Results

We present a visual representation of the logit attribution results from Section 4.2 in Figure 6.

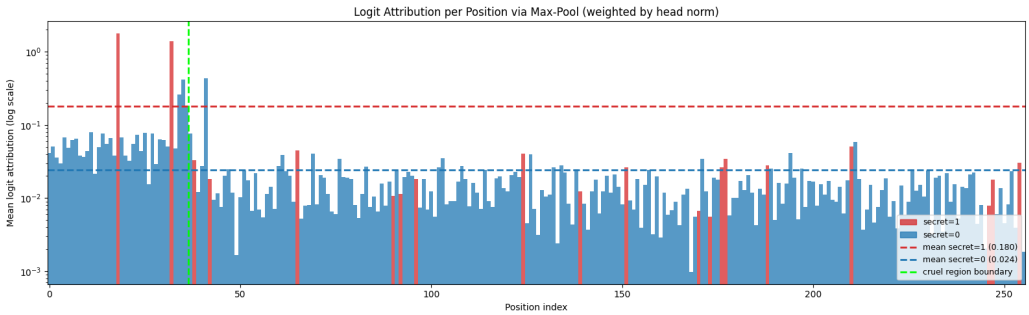

Figure 6. Logit attribution per position on a log scale, weighted by the prediction head norm. The magnitude of the final logits is strongly driven by the specific hidden dimensions corresponding to the secret indices ($s_i = 1$, shown in red), demonstrating that the model leverages the secret support to predict b.

We also analyze the argmax and norm of the embeddings:

(1) Argmax: For each of the 256 hidden dimensions, we identify which position index holds the maximum value, tallying these "wins" across all dimensions and samples. As shown in Figure 7, secret indices dominate the position of the maximum activations.

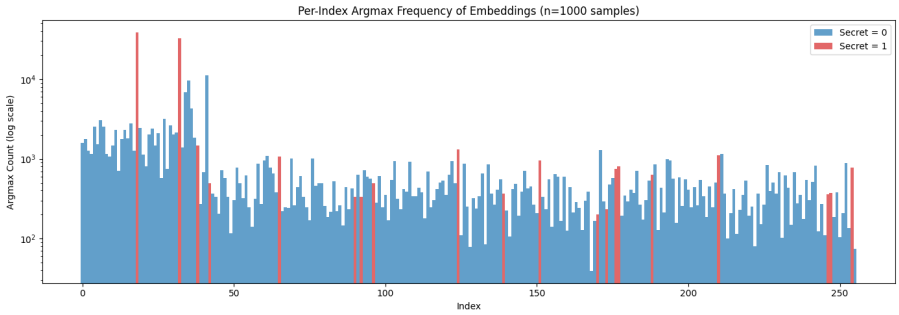

Figure 7. Argmax frequencies tallied across all dimensions and samples for cruel region (left) and cool region (right). Secret bit indices (red, $s_i = 1$) have higher argmax count than non-secret bit indices (blue, $s_i = 0$).

(2) Norm: For each sample, we take the norm of each of the $d_{\text{hidden}} = 256$ values that correspond to a given index of the input. However, we first apply the ReLU function $f(x) = \max(0, x)$ to the embedding to zero out any negative values. This is because many of the values are large negatives (indicating the unimportance of those features) and drive the norm down. Then, we take the mean of all the embedding norms across samples and plot in a bar chart as in Figure 8. We hypothesize that secret indices would have larger norms, again because they actually contribute to the output.

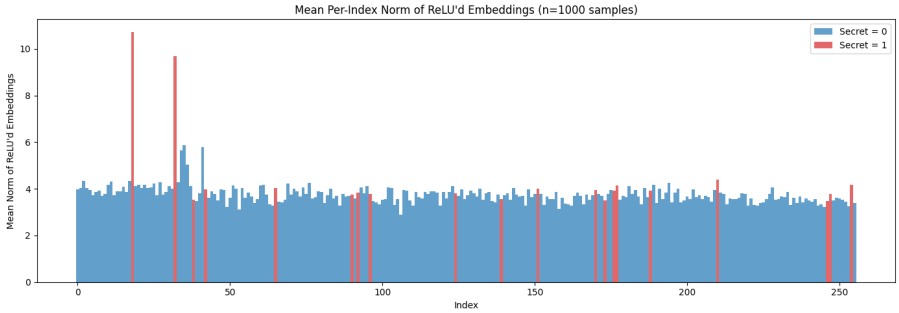

Figure 8. Mean per-index norm of ReLU'd embeddings is larger for secret bits (red, $s_i = 1$) than non-secret bits (blue, $s_i = 0$).

C. Residual Stream

We present the full PCA of embeddings after each layer in the model in Figure 9. The clusters start out more distinct and become more overlapping after each layer.

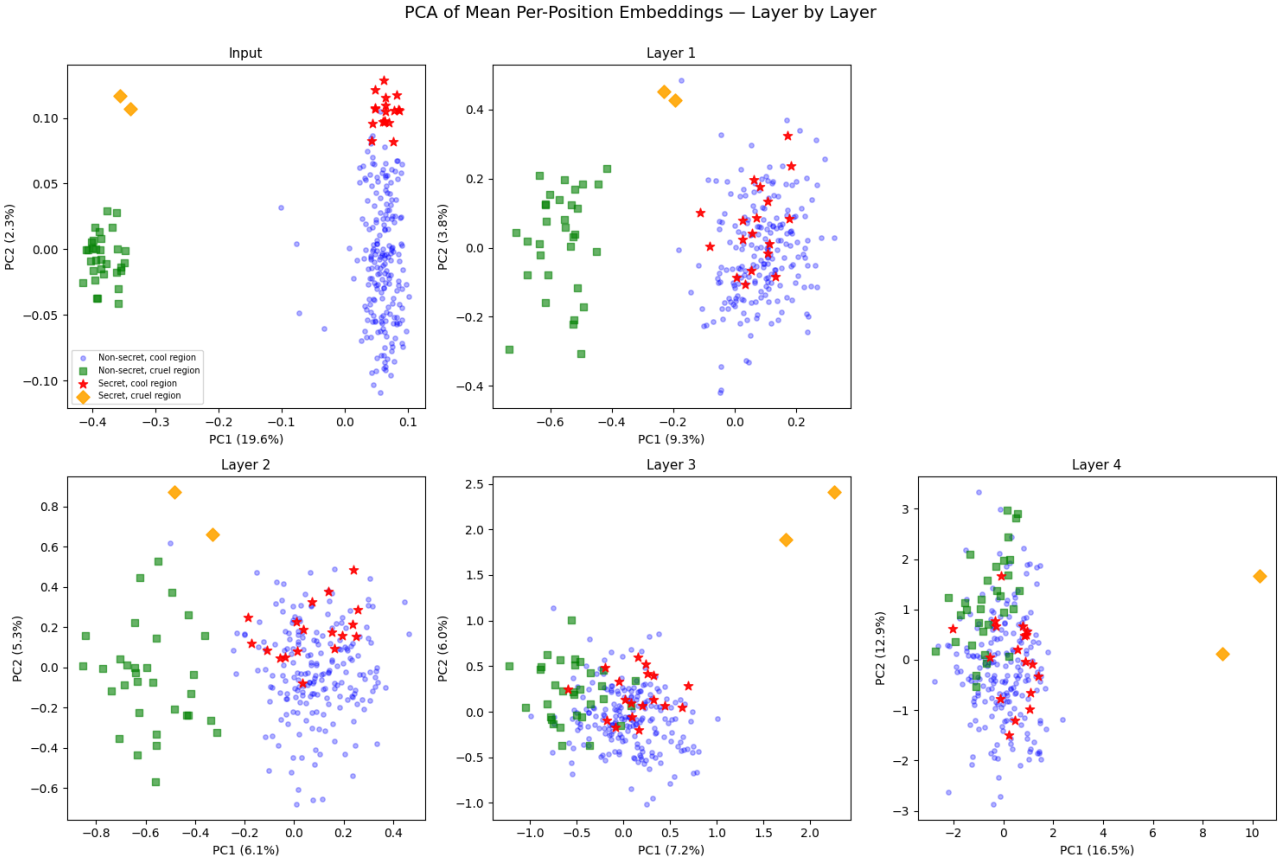

Figure 9. Layer-wise PCA of mean per-position embeddings after each layer block in the model. The clustering of secret indices emerges immediately after the input layer, prior to self-attention.