# LMND: LANGUAGE-MODEL NEURAL DISTINGUISHERS VIA CONDITIONAL GENERATIVE MODELING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Block ciphers are a cornerstone of modern secure communication. Differential cryptanalysis assesses their security by distinguishing ciphertext pairs generated under a fixed input difference from random ones. Neural differential methods replace hand-crafted differential statistics with trained deep neural networks, learning directly from data the weak residual dependencies that are difficult to characterize explicitly, and thereby extending effective analysis to more rounds. Existing CNN-based neural distinguishers typically cast this task as binary classification: each ciphertext pair receives only one sample-level label, yielding coarse supervisory granularity, while the sigmoid output lacks an explicit probabilistic reference, making it difficult to endow the score with statistical meaning or to accumulate evidence across samples. To address these issues, we propose LMND, a neural distinguisher built on conditional generative modeling. Conditioned on one ciphertext, LMND autoregressively predicts the eight nibble tokens of the other, providing position-level supervision for every real pair. Exploiting the fact that fixed-key encryption is a permutation, it takes the uniform distribution as an analytic reference and converts the conditional likelihood into a distinguishing score that requires no negative training samples, no threshold fitting, and accumulates across pairs, with shared-weight bidirectional scoring enforcing invariance under ciphertext exchange. On 5$\sim$8-round Speck32/64, LMND attains the highest single-pair accuracy among all evaluated models; in a 9-round subkey-ranking experiment, it reduces the mean rank from $52.1$ to $23.7$ and raises the rank-0 success rate from $0.358$ to $0.416$ relative to Gohr's baseline.

## 1 INTRODUCTION

Block ciphers are among the most important foundational primitives of modern cryptography, providing essential protection for information confidentiality in the digital age. They are symmetric-key encryption algorithms whose design principles trace back to the confusion and diffusion described by Shannon (1949); their core operation is to reversibly transform a fixed-length plaintext block into a ciphertext block of the same length under the control of a secret key. Nowadays, block ciphers have been widely used in network communications, electronic payments, and data storage, providing efficient confidentiality protection for large-scale digital systems (Mouha, 2021; Dworkin, 2010).

Differential cryptanalysis is a classical method for evaluating the security of block ciphers. Its central idea is to study how a chosen plaintext difference propagates through encryption and to exploit statistical deviations of the resulting ciphertext-difference distribution from random behavior to reveal potential weaknesses in the cipher (Biham & Shamir, 1991). However, relying solely on output differences can miss statistical dependencies within ciphertext pairs, while attack design still requires expert knowledge (Gohr, 2019). Advances in deep learning offer a way to learn such statistical patterns directly from ciphertext pairs. Gohr (2019) formulated ciphertext-pair distinguishing as binary classification and used deep residual convolutional networks to learn statistical differences between ciphertext pairs generated with a fixed input difference and random pairs, constructing neural distinguishers for 5–8-round Speck32/64. The approach outperformed classical differential distinguishers in both distinguishing accuracy and 9-round subkey ranking. Compared with classical methods based on difference distributions, neural differential methods can exploit additional



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

features of ciphertext pairs to improve distinguishing and key recovery, thereby achieving better cryptanalytic performance with less reliance on manually constructed statistical models.

Despite these advances, neural differential methods still face two key challenges in learning and exploiting weak ciphertext-pair dependencies. First, supervision is coarse-grained. Weak residual statistical dependencies in real pairs may involve specific positions or their interactions, while binary training assigns each pair a single real/random label without directly identifying positions containing distinguishing clues. Second, training depends on random negative samples. Binary neural distinguishers such as Gohr's learn from both real and random ciphertext pairs (Gohr, 2019).

To address these challenges, we propose LMND, a conditional generative neural distinguisher. It predicts one ciphertext token by token given the other, providing finer-grained supervision, and learns from real pairs alone using the known uniform distribution as a scoring reference. These scores support single-pair distinguishing and subkey ranking. Specifically, we proceed in three steps. First, we develop an encoder–decoder Transformer that learns conditional dependencies between ciphertext nibble tokens using teacher forcing and token-level cross entropy (§3.1). Second, we derive a uniform-referenced likelihood score from the cipher's permutation property, with a fixed zero threshold for equal class priors. Shared-weight bidirectional scoring ensures ciphertext-exchange invariance, and score accumulation enables subkey ranking (§3.2–3.3). Third, we test single-pair distinguishing on 5–8-round SPECK32/64 and 9-round subkey ranking (§4.2–4.3). In addition, we ablate model designs and assess LMND–CNN prediction complementarity (§4.4–4.5).

## 2 Background and Related Work

This section reviews our research background: the SPECK32/64 block cipher, the differential cryptanalysis framework, neural distinguishers, and autoregressive density estimation underlying LMND.

**SPECK32/64.** SPECK is an addition–rotation–XOR (ARX) block-cipher family. The SPECK32/64 variant encrypts 32-bit blocks under a 64-bit master key, with 22 rounds in the full cipher (Beaulieu et al., 2013). Figure 5 in Appendix A illustrates one round, taking two 16-bit state words $(x_i, y_i)$ and a 16-bit round key $k_i$ generated from the master key. First, $x_i$ is rotated right by seven bits (ROR 7) and $y_i$ left by two bits (ROL 2). Next, the rotated $x_i$ is added to the original $y_i$ modulo $2^{16}$, retaining the lowest 16 bits of the sum. The sum is then XORed with $k_i$ to obtain the updated left word $x_{i+1}$. Finally, the rotated $y_i$ is XORed with $x_{i+1}$ to obtain $y_{i+1}$. The updated pair $(x_{i+1}, y_{i+1})$ forms the state for the next round. These operations define the round function $x_{i+1} = (\mathrm{ROR}_7(x_i) + y_i) \oplus k_i$ and $y_{i+1} = \mathrm{ROL}_2(y_i) \oplus x_{i+1}$. Starting from the two plaintext words, repeating this transformation for 22 rounds with the corresponding round keys produces the 32-bit ciphertext.

**Differential cryptanalysis.** Differential cryptanalysis is a classical method for attacking block ciphers (Biham & Shamir, 1991). It constructs plaintext pairs with a fixed nonzero bitwise-XOR difference, encrypts them under the same key, and measures ciphertext-difference frequencies. For two independent uniformly random blocks, their difference is also uniform: every difference occurs with the same probability. If certain output differences occur substantially more often under real encryption, the chosen input difference has left a statistical bias through these rounds, providing a basis for distinguishing real ciphertext pairs from random pairs. This bias can also support key recovery: one guesses a last-round subkey, partially decrypts the ciphertext pairs, and scores candidates by how well the resulting differences match the expected distribution. Classical methods use tools such as difference distribution tables to describe the probabilities linking input and output differences, together with differential trails to analyze how differences propagate across rounds. However, as the number of rounds increases, useful biases typically weaken while more propagation paths must be tracked and combined, making explicit modeling of these statistical patterns increasingly difficult.

**Neural distinguishers.** Neural distinguishers use neural networks to distinguish real ciphertext pairs from random pairs. The CNN-based method of Gohr (2019), illustrated in Figure 6(a) in Appendix B, first labels real pairs obtained by encrypting fixed-difference plaintexts under the same key as 1 and random pairs as 0. Each pair is then represented as four 16-bit words and processed by a residual CNN to extract dependencies among bits. Flattened features pass through fully connected layers to a sigmoid distinguishing score. The classifier uses mean squared error; after training, fixed weights and a 0.5 threshold separate real from random predictions. For subkey ranking, candidate keys partially decrypt the pairs, and the network outputs are converted to log-odds and summed

across pairs to rank candidates. Subsequent work improves neural distinguishers through input-difference selection, joint modeling of multiple ciphertext pairs, multi-scale feature extraction, and knowledge distillation (Hou et al., 2021; Chen et al., 2023; Hou et al., 2025; Zhang et al., 2024).

Recent work explores LLM-based neural distinguishers. Sakagami et al. (2026) fine-tune general-purpose pretrained LLMs on labeled ciphertext pairs using QLoRA. As shown in Figure 6(b), task instructions and ciphertext information, such as a ciphertext pair or its bitwise XOR, are serialized into a prompt, tokenized, and embedded. A Transformer backbone processes these representations, and a language-model output head generates a real-or-random label. Under this prompt-based binary classification setup, they report no accuracy improvement over ResNet on 5–8-round SPECK32/64.

**Autoregressive density estimation.** An autoregressive model predicts each token from the preceding tokens, so a sequence likelihood becomes a product of conditional probabilities (Bengio et al., 2003; Larochelle & Murray, 2011; Uria et al., 2016). Transformers provide a standard parameterization of these predictions (Vaswani et al., 2017). This formulation supports generating new sequences by sequential sampling and assigning likelihood-based scores to observed sequences.

## 3 METHOD

Figure 1 shows the main workflow of LMND. It contains three stages: model training, single-pair distinguishing, and subkey ranking. Figure 1(a) shows the model training stage: LMND uses only real ciphertext pairs, represents them as nibble sequences, and predicts one ciphertext position by position conditioned on the other to learn their statistical dependencies. Figure 1(b) shows the single-pair distinguishing stage: with the model parameters fixed, LMND compares the conditional likelihood with the uniform reference and combines scores from both conditional directions to decide whether an input pair is real or random. Figure 1(c) shows the subkey ranking stage: candidate subkeys are used to partially decrypt the observed ciphertext pairs, which are scored by the same trained model; summing these scores over the observed ciphertext pairs gives the candidate ranking.

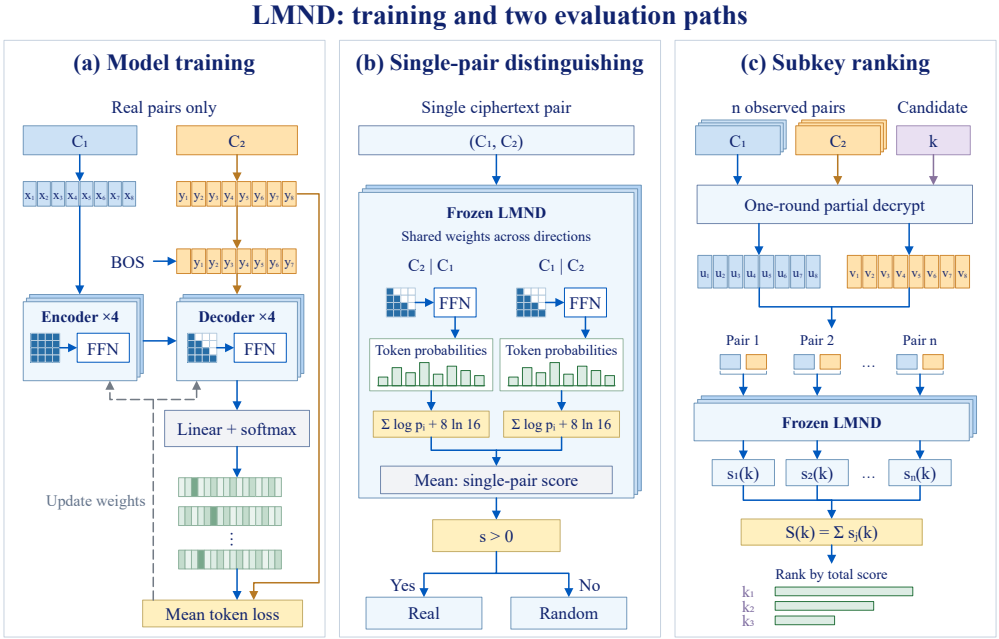


Figure 1: LMND overview: (a) model training; (b) single-pair distinguishing; (c) subkey ranking.

### 3.1 MODEL TRAINING

Model training learns the conditional distribution of one ciphertext given the other, as shown in Figure 1(a). Using only real pairs, an encoder–decoder predicts the target ciphertext token by token, providing a training target at each position. We first define the training data and representation, then introduce the conditional architecture that predicts token probabilities, and finally describe position-level training, which uses these probabilities to calculate the loss and update model parameters.

**Training data and representation.** We construct real ciphertext pairs and represent them as nibble sequences to provide the inputs and prediction targets for conditional training. Each sample is a pair $(C_1, C_2) \in \{0,1\}^{32} \times \{0,1\}^{32}$ drawn from $p_{\text{real}}$: a uniformly sampled plaintext and its fixed-difference counterpart are encrypted under the same key. The generator independently samples a uniform 64-bit master key for each pair, so the learned conditional distribution averages over these keys. The input difference $\Delta$ determines the training distribution rather than serving as a network input; it is fixed within a training stage and can change between curriculum stages. Each ciphertext is split into eight four-bit tokens, ordered left word then right word, with bit offsets $0, 4, 8, 12$ within each word. The source and target tokens are denoted by $x_i = C_{1,i}$ and $y_i = C_{2,i}$. The input vocabulary contains the 16 nibble values and BOS, while predictions cover only the 16 nibble values.

**Conditional architecture.** Conditional prediction requires modeling dependencies within and between ciphertexts. We use an encoder–decoder that predicts each target token from the complete source ciphertext and the preceding target tokens. In Figure 2, four bidirectional encoder layers represent the eight source tokens, and four causal decoder layers process shifted target tokens through masked self-attention and cross-attention to encoder memory. The model uses $d_{\text{model}} = 128$, eight attention heads, an FFN width of 512, and 1,857,664 parameters. It uses GELU, pre-norm, zero dropout, a shared $17 \times 128$ token embedding, and separate learned encoder and decoder positions, with no extra stack-final normalization. A bias-free Linear$(128, 16)$ head and softmax yield token probabilities; zero-initializing the head gives uniform initial predictions over the 16 nibble values.

**Position-level training.** The model requires supervision at each ciphertext position. We train it to predict the observed target tokens using token-level cross entropy. With teacher forcing, prediction at position $i$ receives $C_1$ and the observed prefix $C_{2,<i}$, with BOS marking sequence start. The training loss averages cross entropy across eight target positions and across the real ciphertext pairs:

$$\mathcal{L}(\theta) \;=\; -\,\mathbb{E}_{(C_1,C_2)\sim p_{\text{real}}}\,\frac{1}{8}\sum_{i=1}^{8}\log p_\theta(C_{2,i} \mid C_1, C_{2,<i})\,. \tag{1}$$

Each pair therefore contributes eight prediction losses rather than one binary loss. Training uses weight EMA with decay $0.9999$; Appendix C.1 lists the architecture and training configuration.

### 3.2 SINGLE-PAIR DISTINGUISHING

After training the conditional model, we use it to classify a ciphertext pair as real or random, as shown in Figure 1(b). We compare conditional likelihoods against a uniform reference in both ciphertext orders to obtain a decision. We establish the reference and construct directional scores, then average the scores from both directions and apply a zero threshold under equal class priors.

**Establishing the uniform reference.** A conditional likelihood alone cannot distinguish real from random pairs, so we use the known uniform distribution as a reference. Let $p_{\text{real}}$ denote the real-pair distribution and $p_{\text{unif}}$ that of two independent uniform blocks. We compare the hypotheses using the log likelihood ratio (Neyman & Pearson, 1933):

$$s^\star(C_1, C_2) \;=\; \log p_{\text{real}}(C_1, C_2) \;-\; \log p_{\text{unif}}(C_1, C_2), \tag{2}$$

Equation 2 compares joint distributions, whereas the preceding section learns a conditional distribution. The uniform marginal distribution of each ciphertext connects the two. For uniformly sampled plaintexts, encryption under a fixed key is a permutation of the block space, hence $p_{\text{real}}(C_1) = 2^{-32}$. Because training keys are sampled independently of the plaintexts, averaging over these keys preserves the uniform marginal. Under the random hypothesis, $p_{\text{unif}}(C_1, C_2) = 2^{-64}$. Factoring the joint distribution as $p_{\text{real}}(C_1, C_2) = p_{\text{real}}(C_1)p_{\text{real}}(C_2 \mid C_1)$ and substituting into Eq. 2 gives

$$s^\star(C_1, C_2) \;=\; \log p_{\text{real}}(C_2 \mid C_1) \;+\; 32 \ln 2, \tag{3}$$

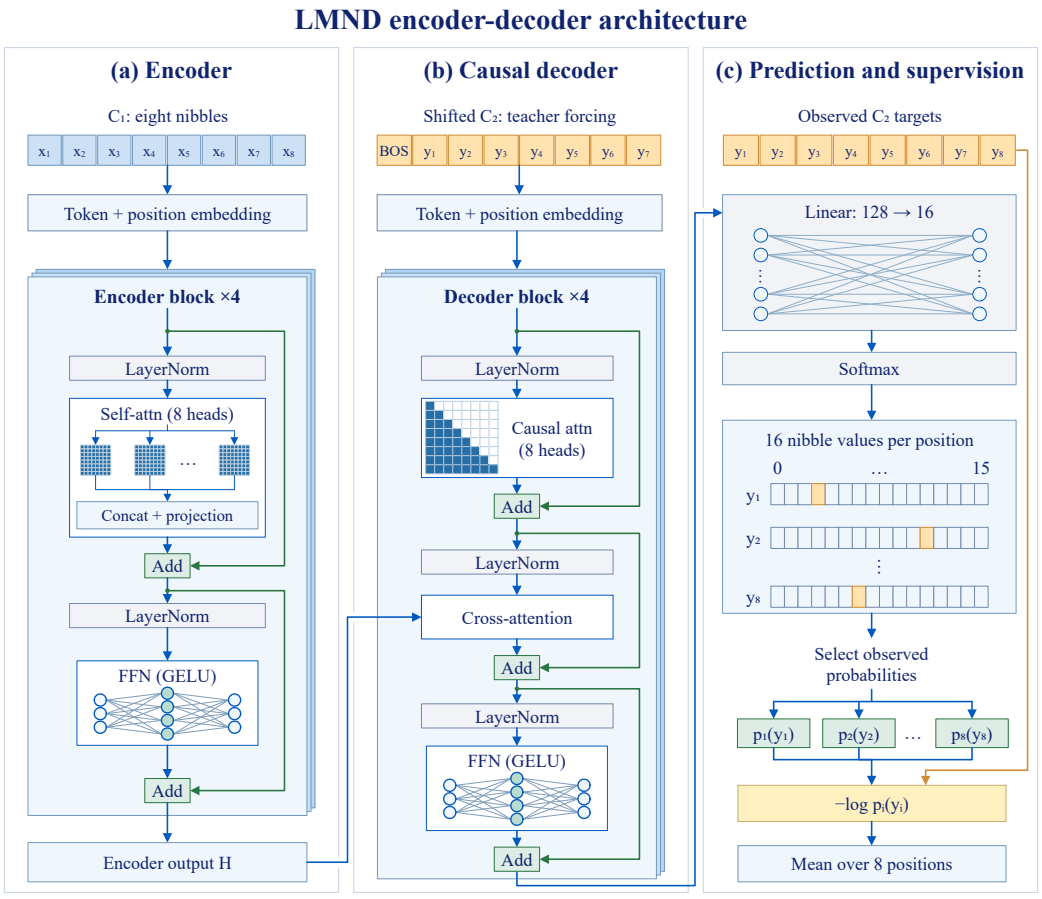


Figure 2: EncDecLM architecture: (a) encoder; (b) causal decoder; (c) prediction and supervision.

**Constructing the directional score.** We convert the token probabilities into a single-pair score by comparing the learned conditional likelihood with the uniform reference. This score provides a basis for distinguishing real ciphertext pairs from random ones. In Eq. 3, we replace the true conditional density with the learned approximation $p_\theta(C_2 \mid C_1)$. With the model weights fixed, we compute the conditional log likelihood by summing the log probabilities of the eight observed target tokens, each conditioned on $C_1$ and its observed target prefix. This evaluates the observed target ciphertext directly, without generating a new ciphertext by sampling, and yields the following directional score:

$$s(C_1, C_2) \;=\; \sum_{i=1}^{8} \log p_\theta(C_{2,i} \mid C_1, C_{2,<i}) \;+\; 8 \ln 16, \qquad \text{decide } \textit{real} \text{ iff } s > 0. \tag{4}$$

A positive score indicates that the observed target's conditional probability exceeds the uniform reference, giving a zero threshold under equal priors and equal misclassification costs without fitting it on held-out labels. When the learned conditional density equals the true density, this rule recovers the Bayes-optimal decision; otherwise, optimality and calibration depend on the accuracy of the estimated conditional density.

**Bidirectional averaging and decision.** The decision should not depend on ciphertext order. We average the two directional scores computed with shared frozen weights. Ciphertext exchange can change directional scores, but the real-pair distribution is exchange-symmetric: swapping the encryptions of $(P, P \oplus \Delta)$ yields those of $(P \oplus \Delta, P)$, with the same input difference. We obtain:

$$s_{\text{sym}}(C_1, C_2) \;=\; \tfrac{1}{2}\,[\,s(C_1, C_2) + s(C_2, C_1)\,], \tag{5}$$

The reverse direction encodes $C_2$ and evaluates $C_1$ autoregressively, requiring a second forward pass through the same model. The resulting decision classifies the ciphertext pair as real when $s_{\text{sym}} > 0$.

The zero threshold is determined by the uniform reference rather than by the average score of random samples. When the model itself is uniform, each token log probability is $-\ln 16$, so both directional and averaged scores are zero, as at zero initialization. For a general nonuniform model, writing $U$ for the uniform distribution over 32-bit target ciphertexts, the expected directional score on independent uniformly random ciphertext pairs satisfies $\mathbb{E}_{\text{unif}}[s] = -\mathbb{E}_{C_1}\mathrm{KL}(U \| p_\theta(\cdot \mid C_1)) \leq 0$.

### 3.3 SUBKEY RANKING

Subkey ranking accumulates scores over observed ciphertext pairs to rank candidate subkeys. This extends the preceding single-pair decision to the key-recovery setting. A thresholded label discards the magnitude of each pair's evidence, so we retain the continuous score and use candidate-dependent partial decryption to connect it to candidate-key hypotheses, as shown in Figure 1(c).

Let $D_k$ denote one-round partial decryption under candidate subkey $k$. For each candidate, we partially decrypt the $n$ observed ciphertext pairs, convert the resulting blocks to the training nibble representation, and apply the frozen directional scorer $s$ from §3.2. The candidate's total score is

$$S_{\text{total}}(k) = \sum_{i=1}^{n} s\big(D_k(C_1^{(i)}), D_k(C_2^{(i)})\big), \tag{6}$$

Candidates are ranked by decreasing total score, a composite ranking statistic. Model parameters and scoring remain fixed during search; only candidate-dependent inputs change. Section 4.3 specifies the attack's pair construction, preprocessing, candidate space, and evaluation protocol.

## 4 EXPERIMENTS

### 4.1 SETUP

We evaluate LMND on 5–8-round SPECK32/64, following established single-pair baselines (Gohr, 2019; Hou et al., 2025). Real pairs use difference $(\texttt{0x0040}, \texttt{0x0000})$; random pairs contain independent uniform blocks. Each epoch uses $10^7$ freshly generated real pairs, and validation contains $5 \times 10^5$ pairs per class. Testing uses five fixed, disjoint million-pair batches per round. The baselines are Gohr's depth-10 CNN and the single-pair ND1 model (Gohr, 2019; Hou et al., 2025).

Training and evaluation are conducted on a local server equipped with two AMD EPYC 7Y83 CPUs and four NVIDIA GeForce RTX 4090 GPUs. Table 6 in Appendix C.2 lists the hardware and software specifications. EncDecLM has 1,857,664 parameters and uses AdamW with a batch size of 5000. We maintain an exponential moving average (EMA) of the model weights with decay 0.9999.

### 4.2 SINGLE-PAIR ACCURACY

We first evaluate single-pair accuracy. It measures the fraction of correctly classified real and random ciphertext pairs, assessing the model's distinguishing ability from a single pair. Table 1 gives the main comparison. LMND achieves the highest mean accuracy among the three evaluated models at every tested round. Relative to ND1, its gains are $0.002$, $0.007$, $0.007$, and $0.003$ from 5 to 8 rounds. The accuracy of all three models decreases as the number of rounds increases, reflecting the weakening statistical signal in the ciphertext pairs. LMND declines from $0.933$ at 5 rounds to $0.517$ at 8 rounds but remains above both CNN baselines across the entire range of tested rounds.

LMND's bidirectional encoder integrates the complete source ciphertext, and its decoder combines the source representations with the target prefix through cross-attention to predict conditional probabilities at each target position. In LMND, each real pair provides eight prediction targets, versus one class label in Gohr's CNN baseline, providing finer-grained supervision for learning conditional dependencies across ciphertext positions. The single-pair experiment tests whether this approach improves distinguishing accuracy. Each round uses five disjoint million-pair test batches. At eight rounds, the accuracy gain over ND1 is $0.003$, larger than the standard deviations of $0.000479$ for LMND and $0.000297$ for ND1, supporting gains beyond the observed test-batch variation.

Stronger single-pair distinguishing can improve key recovery by helping to separate correct and incorrect subkey candidates. We apply the trained model to the key-recovery attack in Section 4.3.

Table 1: Single-pair accuracy (mean ± standard deviation).

| Rounds | Gohr CNN (Gohr, 2019) | ND1 (Hou et al., 2025) | LMND (ours) |
|---|---|---|---|
| 5r | $0.929 \pm 5.13\times10^{-4}$ | $0.931 \pm 2.28\times10^{-4}$ | $\mathbf{0.933 \pm 1.40\times10^{-4}}$ |
| 6r | $0.788 \pm 8.17\times10^{-4}$ | $0.789 \pm 3.27\times10^{-4}$ | $\mathbf{0.796 \pm 3.36\times10^{-4}}$ |
| 7r | $0.616 \pm 9.70\times10^{-4}$ | $0.615 \pm 7.40\times10^{-4}$ | $\mathbf{0.622 \pm 5.49\times10^{-4}}$ |
| 8r | $0.514 \pm 1.00\times10^{-3}$ | $0.514 \pm 2.97\times10^{-4}$ | $\mathbf{0.517 \pm 4.79\times10^{-4}}$ |

*Note.* The 8-round model uses staged initialization. We adapt the 7-round weights on 5-round pairs with difference $(\texttt{0x8000}, \texttt{0x840a})$, then use the adapted weights to initialize 8-round training.

### 4.3 Key-recovery attack

After model training, we next use the trained model to compare candidate subkeys through partial decryption. In the key-recovery process, we apply the inverse round function to both ciphertexts in an observed pair using each candidate last-round subkey, obtaining the intermediate values implied by that key guess. With its parameters fixed, LMND scores how closely the resulting pair matches the conditional relationships learned during training. For each candidate subkey, we sum continuous scores over all observed ciphertext pairs, rank candidates in descending order of their total scores, and select the candidate with the highest total score as the most likely value of the last-round subkey.

We evaluate this procedure on 9-round Speck32/64, using a frozen 7-round distinguisher to rank $2^{16}$ last-round subkey candidates from $n = 64$ ciphertext pairs (128 chosen plaintexts). We construct plaintext pairs with difference $\Delta_0 = (\texttt{0x0040}, \texttt{0x0000})$, apply one inverse round with a zero subkey to each plaintext, and then encrypt for nine rounds. After correct last-round decryption, the pair corresponds to seven intermediate rounds.

We compare D7-DDT (the classical statistic), N7 (Gohr's net7 log-odds; Gohr, 2019), and T7 (LMND). We run two experiments of 500 attacks each. Table 7 in Appendix D.1 reports mean rank with twice the standard error of the mean, rank quantiles, and success rate over 1000 attacks. Figure 3 visualizes mean rank and success rate. LMND uses zero-based midranks (average ranks for ties). Its SR counts attacks where the correct subkey has the unique highest score.

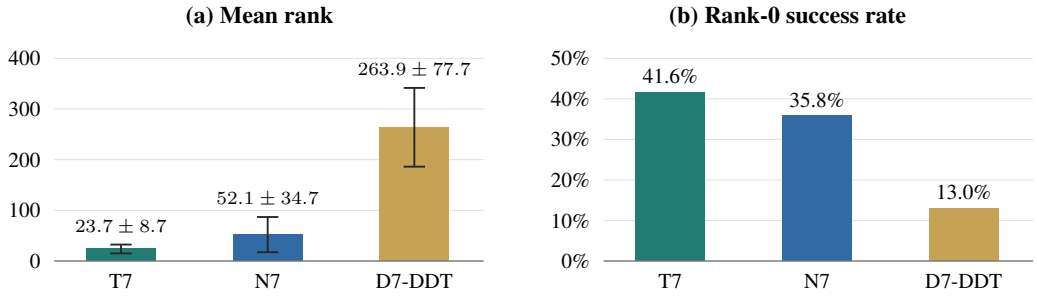


Figure 3: 9-round subkey ranking (error bars: ±2 standard errors of the mean).

As shown in Figure 3, LMND reduces the mean subkey rank from 52.1 to 23.7 and raises rank-0 success from 0.358 to 0.416 compared with Gohr's CNN. In contrast, the preceding 7-round single-pair evaluation shows only a modest accuracy increase, from 0.616 to 0.622. We analyze the reasons for this contrast in terms of score information, fine-grained supervision, and score accumulation.

**Score information.** LMND's continuous likelihood scores retain differences in evidence strength for subkey ranking that single-pair accuracy does not measure. Single-pair accuracy records whether scores are correctly thresholded; subkey ranking compares scores for intermediate pairs produced by candidate keys. Score differences can reorder candidates without changing real/random decisions.

**Fine-grained supervision.** Token-level supervision trains the model to estimate the conditional probabilities later used to score candidate subkeys. Each real pair provides eight token-prediction

targets rather than one class label in Gohr's CNN (Gohr, 2019), giving LMND denser supervision for learning weak dependencies. The conditional log probabilities optimized during training also form the likelihood terms of the attack score, linking the learned relationships directly to candidate-key comparison. More accurate estimates of these conditional probabilities can change the score gaps used to compare competing candidate subkeys and thus affect the order in which they are ranked.

**Score accumulation.** Score accumulation combines ciphertext-pair evidence to compare candidate subkeys. The total-score gap between the correct and a competing key sums their per-pair score differences (Eq. 6). When LMND's per-pair score differences consistently favor the correct key, summing 64 scores amplifies this advantage. Among $2^{16}$ candidates, overtaking closely scored wrong keys can substantially improve rank without a comparable increase in single-pair accuracy.

### 4.4 Ablations

We conduct two ablations on 5–7-round SPECK32/64 to examine how token order and conditioning layout affect distinguishing accuracy (Figure 4; detailed results in Table 8, Appendix D.2). First, we retain the encoder–decoder and replace LSB-first with MSB-first to test nibble prediction order. Second, we retain LSB-first ordering and replace the encoder–decoder with a causal Prefix model to assess the value of bidirectional source encoding before target prediction. We analyze these comparisons through target-prefix availability and source-position interactions, respectively.

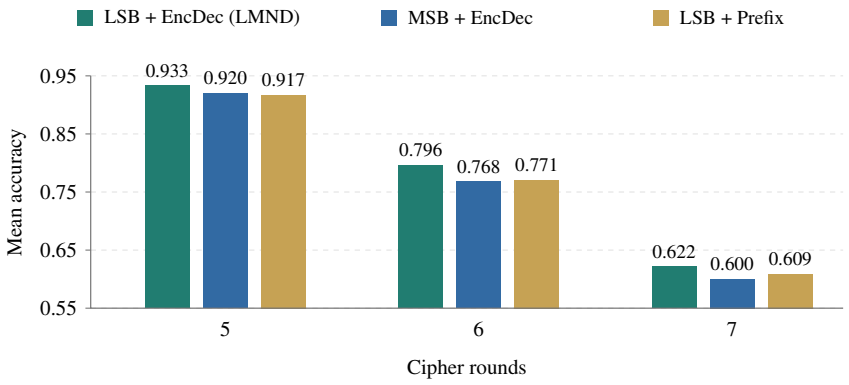


Figure 4: Mean accuracy of the three ablation configurations; the vertical axis starts at 0.55.

**Token order.** Token order determines which target values the model can use at each prediction step without changing the ciphertext bits. With the encoder–decoder architecture, LSB-first outperforms MSB-first at all three tested rounds; at seven rounds, accuracy increases from 0.600 to 0.622. LSB-first places the lower-order nibbles of each word in the observed prefix when predicting its higher-order nibbles, whereas MSB-first conditions lower-order predictions on higher-order values. The former follows the low-to-high direction of carry propagation in modular addition. Thus, ordering changes the model's conditional prediction tasks, not the amount of ciphertext information.

**Conditioning layout.** The conditioning layout determines how source-ciphertext information is combined before it is used to predict target tokens. With LSB-first token order, the encoder–decoder outperforms the causal Prefix variant at all three tested rounds, reaching 0.622 rather than 0.609 at seven rounds. The encoder integrates the complete source ciphertext through bidirectional attention, so each source representation includes information from all source positions. Cross-attention then lets the decoder use these contextualized representations together with the target prefix. In the causal Prefix variant, each source representation can aggregate information only from its own and preceding positions. The encoder–decoder therefore allows dependencies among all source positions to be modeled before the decoder uses the resulting source representations to predict the target tokens.

The above ablations both support LMND's use of LSB-first token order and an encoder–decoder architecture: replacing either design reduces distinguishing accuracy across all three tested rounds.

### 4.5 Output Complementarity Between Transformers and CNNs

Beyond distinguishing and key recovery, we also found from the test results that LMND and the CNN each correctly classify some ciphertext pairs misclassified by the other. We examine this complementarity between ND1 (Hou et al., 2025) and LMND on the same fixed benchmark using decision overlap, score fusion, and logit-bin accuracy (Tables 2 and 3, and Table 9 in Appendix D.3). We measure the fractions correctly classified by both models, only one, or neither, and the oracle union (at least one correct). Confidence-select chooses predictions by output confidence; z-avg averages standardized scores; score-level stacking uses logistic regression. Stacking is fitted to standardized ND1 sigmoid outputs and LMND scores on $2 \times 10^5$ ciphertext pairs disjoint from evaluation. Finally, we compare accuracy within quintiles of the absolute raw CNN logit.

The models make largely consistent decisions but retain complementary errors, while the tested fusion strategies yield small accuracy gains. At seven rounds, only CNN and only LMND are correct on 4.91% and 5.07% of samples, respectively, giving an oracle union of 0.6677 (Table 2). The best fusion accuracy on the same benchmark is 0.6193, only 0.0007 above LMND's 0.6186 and well below the oracle union (Table 3). At six to eight rounds, LMND's net accuracy gains are concentrated in the lowest absolute-CNN-logit bin, close to the CNN decision boundary, whereas the models perform similarly in high-magnitude bins (Table 9). LMND and the CNN can correct some of each other's errors, but the tested fusion methods provide only limited accuracy gains.

Table 2: Decision overlap and oracle union (test-set fractions).

| Rounds | CNN acc | T acc | both correct | only CNN | only T | both wrong | oracle union |
|---|---|---|---|---|---|---|---|
| 5r | 0.9309 | 0.9296 | 0.9195 | 0.0114 | 0.0101 | 0.0590 | 0.9410 |
| 6r | 0.7906 | 0.7934 | 0.7687 | 0.0219 | 0.0247 | 0.1847 | 0.8153 |
| 7r | 0.6170 | 0.6186 | 0.5679 | 0.0491 | 0.0507 | 0.3323 | 0.6677 |
| 8r | 0.5132 | 0.5143 | 0.4506 | 0.0625 | 0.0637 | 0.4231 | 0.5769 |

Table 3: Score-fusion accuracy. Fusion-fit: gain over Gohr's CNN in a separate evaluation.

| Rounds | confidence-select | z-avg | score-level stacking | fusion-fit gain |
|---|---|---|---|---|
| 5r | 0.9318 | 0.9234 | 0.9319 | +0.0010 |
| 6r | 0.7933 | 0.7890 | 0.7931 | +0.0021 |
| 7r | 0.6193 | 0.6193 | 0.6191 | +0.0025 |
| 8r | 0.5142 | 0.5142 | 0.5149 | +0.0008 |

## 5 Conclusion and Future Work

LMND models ciphertext-pair dependencies with an encoder–decoder Transformer trained on eight nibble predictions per real pair. Its conditional likelihood is scored against a uniform reference derived from the cipher's permutation property, giving a fixed decision threshold and an additive score for subkey ranking. LMND achieves the highest mean single-pair accuracy among the evaluated models at 5–8 rounds and improves mean subkey rank and rank-0 success over Gohr's net7 in the 9-round experiment. In addition, we analyze LMND–CNN complementarity through decision overlap, score fusion, and accuracy across CNN-logit bins. Each model correctly classifies some samples that the other misclassifies, while the tested fusion strategies yield only limited accuracy gains.

In future work, we plan to extend LMND to block ciphers with different round structures and block sizes. This will test the effectiveness of conditional generative modeling beyond Speck32/64 and guide ciphertext representation design for different cipher operations. We will also investigate higher-round distinguishing and key recovery, developing training strategies to capture weaker statistical dependencies after more diffusion rounds. We aim to narrow the round-count gap between neural cryptanalysis and full-round designs to assess block-cipher security margins in greater detail.

## REPRODUCIBILITY STATEMENT

No external dataset is required: ciphertext pairs are generated from the sampling procedures and input differences in Sections 4.1 and 4.3. Sections 3.1 and 3.2 specify the token representation, model, training objective, and scoring rule. Appendix C.1 records the optimizer, batch size, learning-rate schedules, exponential moving average, and warm-start settings. Section 4.1 describes validation and test construction and the baseline protocol; Section 4.3 describes attack preprocessing, ciphertext-pair counts, the candidate subkey space, attack repetition counts, and rank statistics.

## ETHICS STATEMENT

This work studies public, round-reduced versions of SPECK32/64 to evaluate learning-based cryptanalysis. It uses synthetic data and no human subjects or personal information. Cryptanalytic methods are dual use: they can improve security evaluation but may also assist attacks. Our results apply only to the reduced-round, chosen-plaintext setting studied here.

## AI USE STATEMENT

During manuscript revision, generative AI tools assisted with editing and translation, literature organization, figure design, and review of mathematical exposition and result interpretation. No experimental data were generated and no models were implemented, trained, or evaluated as part of this revision. The authors are responsible for verifying all AI-assisted material and the final content.

## REFERENCES

Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Paper 2013/404, 2013. URL https://eprint.iacr.org/2013/404.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL https://www.jmlr.org/papers/v3/bengio03a.html.

Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology – CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pp. 2–21. Springer, 1991. doi: 10.1007/3-540-38424-3_1.

Yi Chen, Yantian Shen, Hongbo Yu, and Sitong Yuan. A new neural distinguisher considering features derived from multiple ciphertext pairs. *The Computer Journal*, 66(6):1419–1433, 2023. doi: 10.1093/comjnl/bxac019.

Morris Dworkin. Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices. Technical Report SP 800-38E, National Institute of Standards and Technology, 2010.

Aron Gohr. Improving attacks on round-reduced Speck32/64 using deep learning. In *Advances in Cryptology – CRYPTO 2019*, volume 11693 of *Lecture Notes in Computer Science*, pp. 150–179. Springer, 2019. doi: 10.1007/978-3-030-26951-7_6.

Yufei Hou, Jie Liu, Shouxu Han, Zhongjun Ma, Xi Ye, and Xuan Nie. Improving deep learning-based neural distinguisher with multiple ciphertext pairs for Speck and Simon. *Scientific Reports*, 15:13696, 2025. doi: 10.1038/s41598-025-98251-1.

Zezhou Hou, Jiongjiong Ren, and Shaozhen Chen. Improve neural distinguishers of SIMON and SPECK. *Security and Communication Networks*, 2021:9288229, 2021. doi: 10.1155/2021/9288229.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 29–37, 2011. URL https://proceedings.mlr.press/v15/larochelle11a.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Nicky Mouha. Review of the Advanced Encryption Standard. Technical Report NISTIR 8319, National Institute of Standards and Technology, 2021.

Jerzy Neyman and Egon S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A*, 231:289–337, 1933. doi: 10.1098/rsta.1933.0009.

Tatsuya Sakagami, Masashi Hisai, and Naoto Yanai. Do LLMs make neural distinguishers wise? *arXiv preprint arXiv:2606.10692*, 2026. URL https://arxiv.org/abs/2606.10692.

Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28 (4):656–715, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016. URL https://www.jmlr.org/papers/v17/16-272.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Liu Zhang, Zilong Wang, and Baocang Wang. Improving differential-neural cryptanalysis. *IACR Communications in Cryptology*, 1(3), 2024. doi: 10.62056/AY11WA3Y6.

## A ONE ROUND OF SPECK32/64

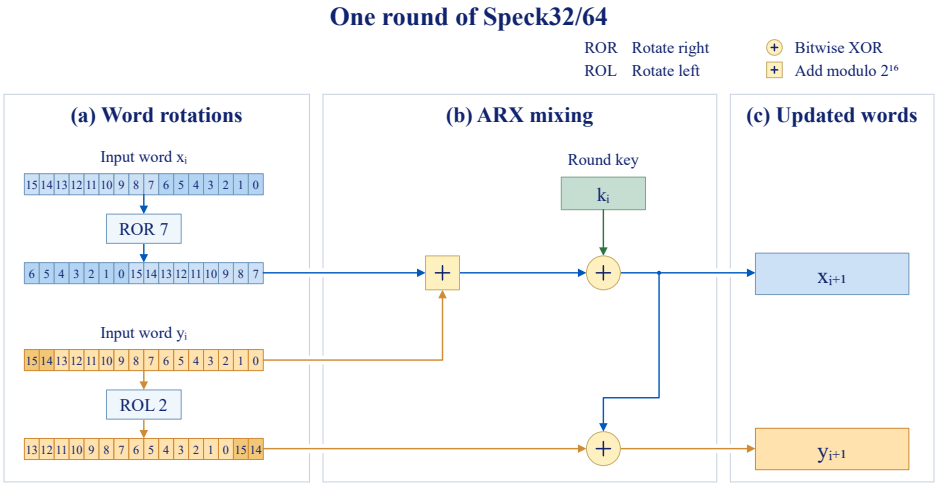


Figure 5: One encryption round of SPECK32/64.

## B NEURAL DISTINGUISHER ARCHITECTURES

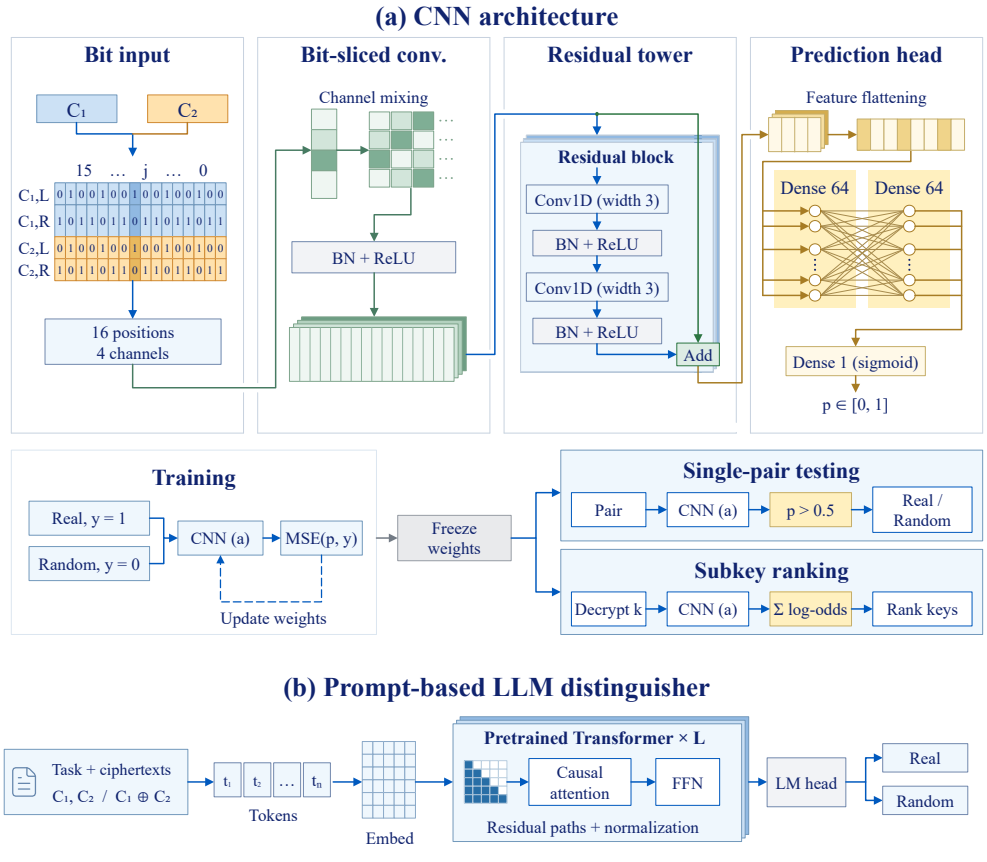


Figure 6: Neural distinguishers: (a) CNN architecture and workflow; (b) prompt-based LLM.

# C Implementation Details

## C.1 Hyperparameters and Training Configurations

Tables 4–5 summarize the model architecture and training configuration. The model shares input embeddings, uses a separate bias-free output head, and has no extra stack-final normalization. The 8-round curriculum adapts 7-round weights on 5-round pairs before 8-round training (note to Table 1).

Table 4: EncDecLM architecture.

| | |
|---|---|
| Type | encoder–decoder conditional generative model $p(C_2 \mid C_1)$ |
| Encoder | 4-layer bidirectional Transformer (8 tokens of $C_1$ as prompt) |
| Decoder | 4-layer causal Transformer (8 tokens of $C_2$, cross-attention to $C_1$) |
| $d_{\text{model}}$ / heads / FFN | 128 / 8 / 512 (GELU, pre-norm) |
| Dropout | 0.0 |
| Positional encoding | learnable (enc/dec, $8\times128$ each) |
| Input embedding | shared $17 \times 128$ table (16 nibble IDs and BOS) |
| Output head | bias-free Linear$(128, 16)$, zero-initialized; no BOS output |
| Tokenization | left then right 16-bit word; each split into 4 LSB-first nibbles |
| Loss | cross-entropy averaged over batch and the 8 positions of $C_2$ |

Table 5: LMND training configuration.

| | |
|---|---|
| Optimizer | AdamW (weight decay $= 0.01$) (Loshchilov & Hutter, 2019) |
| Learning rate | 500-step warmup, then linear decay: 5r $10^{-3}\to10^{-4}$; 6r $5\times10^{-4}\to5\times10^{-5}$; 7r $2\times10^{-4}\to2\times10^{-5}$; 8r final stage $2\times10^{-4}\to10^{-5}$ |
| Batch size | 5000 |
| Epochs | 5r $= 400$; 6r/7r/8r $= 200$ (warm-started round by round from 6r on) |
| Data scale | $10^7$ fresh pairs regenerated per epoch; validation $5\times10^5\times2$ |
| EMA | decay $= 0.9999$; final-step EMA saved separately from best-loss weights |

## C.2 Experimental Server Configuration

Table 6 lists the server's hardware and software. Training and evaluation each use one GPU.

Table 6: Experimental server configuration.

| Component | Configuration |
|---|---|
| CPU | $2\times$ AMD EPYC 7Y83 (64 cores each) |
| GPU | $4\times$ NVIDIA GeForce RTX 4090 (24 GB each) |
| Operating system | Ubuntu 22.04.5 LTS |
| Python | 3.11.15 |
| PyTorch | 2.6.0+cu124 |
| PyTorch CUDA | 12.4 |
| cuDNN | 9.1.0 |

# D Additional Experimental Results

## D.1 Subkey-Ranking Results

Table 7: 9-round subkey-ranking results (mean rank $\pm 2$ standard errors of the mean).

| Distinguisher | mean | P50 | P75 | P90 | P95 | P99 | Rank-0 SR |
|---|---|---|---|---|---|---|---|
| D7-DDT | $263.9 \pm 77.7$ | 9 | — | — | — | — | 0.13 |
| N7: Gohr net7 | $52.1 \pm 34.7$ | **1** | 8 | 36 | 108 | 893 | 0.358 |
| T7: LMND, ours | $\mathbf{23.7 \pm 8.7}$ | 2 | **6** | **24** | **58** | **531** | **0.416** |

## D.2 Detailed Ablation Results

Table 8: Ablation accuracy (mean $\pm$ standard deviation).

| Rounds | LMND: LSB+EncDec | MSB+EncDec | LSB+Prefix |
|---|---|---|---|
| 5r | $\mathbf{0.933 \pm 1.40{\times}10^{-4}}$ | $0.920 \pm 2.0{\times}10^{-4}$ | $0.917 \pm 2.0{\times}10^{-4}$ |
| 6r | $\mathbf{0.796 \pm 3.36{\times}10^{-4}}$ | $0.768 \pm 3.0{\times}10^{-4}$ | $0.771 \pm 4.0{\times}10^{-4}$ |
| 7r | $\mathbf{0.622 \pm 5.49{\times}10^{-4}}$ | $0.600 \pm 4.0{\times}10^{-4}$ | $0.609 \pm 7.0{\times}10^{-4}$ |

## D.3 Accuracy by CNN Logit Magnitude

Table 9: Accuracy by CNN logit magnitude ($\Delta_{\mathrm{acc}}$: CNN minus LMND).

**5r**, $n = 40000$ each

| $\lvert s_{\mathrm{CNN}}\rvert$ | CNN | T | $\Delta_{\mathrm{acc}}$ |
|---|---|---|---|
| (0.00, 2.20] | 0.7430 | 0.7373 | +0.0057 |
| (2.20, 3.48] | 0.9422 | 0.9419 | +0.0003 |
| (3.48, 4.70] | 0.9790 | 0.9788 | +0.0002 |
| (4.70, 6.21] | 0.9927 | 0.9925 | +0.0001 |
| (6.21, 13.32] | 0.9977 | 0.9977 | +0.0000 |

**6r**

| $\lvert s_{\mathrm{CNN}}\rvert$ | CNN | T | $\Delta_{\mathrm{acc}}$ |
|---|---|---|---|
| (0.00, 0.59] | 0.5733 | 0.5872 | −0.0140 |
| (0.59, 1.18] | 0.7040 | 0.7044 | −0.0004 |
| (1.18, 1.74] | 0.8097 | 0.8095 | +0.0003 |
| (1.74, 2.71] | 0.8886 | 0.8886 | +0.0000 |
| (2.71, 7.37] | 0.9775 | 0.9775 | −0.0000 |

**7r**

| $\lvert s_{\mathrm{CNN}}\rvert$ | CNN | T | $\Delta_{\mathrm{acc}}$ |
|---|---|---|---|
| (0.00, 0.18] | 0.5230 | 0.5353 | −0.0123 |
| (0.18, 0.35] | 0.5644 | 0.5647 | −0.0003 |
| (0.35, 0.53] | 0.6057 | 0.6026 | +0.0031 |
| (0.53, 0.76] | 0.6494 | 0.6478 | +0.0016 |
| (0.76, 3.80] | 0.7426 | 0.7426 | −0.0000 |

**8r**

| $\lvert s_{\mathrm{CNN}}\rvert$ | CNN | T | $\Delta_{\mathrm{acc}}$ |
|---|---|---|---|
| (0.00, 0.01] | 0.5018 | 0.5053 | −0.0034 |
| (0.01, 0.03] | 0.5052 | 0.5064 | −0.0012 |
| (0.03, 0.05] | 0.5092 | 0.5100 | −0.0008 |
| (0.05, 0.09] | 0.5169 | 0.5172 | −0.0003 |
| (0.09, 0.70] | 0.5328 | 0.5329 | −0.0000 |