LEAKDIT: Diffusion Transformers for Trace-Augmented Side-Channel Analysis

Insup Lee, Daehyeon Bae, Seokhie Hong, and Sangjin Lee

***Abstract*—Deep learning has been extensively used in side-channel analysis (SCA), making trace data insufficiency and class imbalance a critical challenge. Although several studies have explored trace augmentation with generative models, two core limitations remain: (i) insufficient integration of SCA domain knowledge into the models and (ii) limited adoption of state-of-the-art diffusion transformers (DiT). This letter presents LEAKDIT, a domain-specific one-dimensional DiT that generates high-quality traces. LEAKDIT introduces a loss based on normalized inter-class variance (NICV) to produce realistic traces that preserve the leakage structure. Experimental results demonstrate that LEAKDIT improves SCA performance and reduces the number of required traces for key recovery.**

***Index Terms*—Side-channel analysis, data augmentation, deep learning, diffusion transformers.**

I. INTRODUCTION

WHEN cryptographic modules run on hardware, physical side channels (e.g., power consumption and electromagnetic signals) can leak sensitive information such as cryptographic keys and intermediate values, producing *traces*. Side-channel analysis (SCA) aims to recover secrets from these traces, making it a primary security concern for hardware and embedded systems [1]. Researchers have increasingly applied deep learning to SCA to automatically extract discriminative features from raw traces without explicit time alignment or manual selection of points of interest (POI). Because convolutional neural networks (CNNs) effectively learn local, translation-invariant patterns in time-series data, numerous studies have focused on CNN-based SCA [2]–[4]. However, the availability of sufficient and balanced high-quality training data remains critical to achieving reliable deep learning-based SCA performance. In data insufficiency scenarios, CNN-based SCA classifiers often overfit and suffer substantial performance degradation.

To address these issues, one promising solution is data augmentation. Since manual data collection is costly and introduces significant overhead, recent advances in deep generative models (e.g., generative adversarial networks (GANs) and diffusion models) offer an attractive alternative. Specifically, recent SCA studies have applied conditional GANs (cGANs) [5]–[7] and denoising diffusion probabilistic models (DDPMs) [8], [9]. However, previous studies [5]–[9] have mainly applied deep learning techniques directly to SCA, with limited consideration of SCA-specific domain knowledge such as key-dependent time points. Moreover, although diffusion models have demonstrated higher data generation quality than GANs [10], diffusion transformers (DiT) [11] combine diffusion with transformer architectures and represent the state-of-the-art in many conditional generation tasks; to the best of our knowledge, no prior work has examined DiT in SCA.

In this context, we present LEAKDIT, a DiT-based generator that emphasizes these SCA-specific aspects for augmentation of high-quality electromagnetic traces. LEAKDIT is derived from DiT [11], as DiT combines the strengths of diffusion models and transformers: (i) diffusion models enhance the capture of temporal dependencies and mitigate mode collapse compared to GANs, and (ii) transformers provide architectural stability. The core insight of LEAKDIT is to reflect domain knowledge of normalized inter-class variance (NICV) to control the quality of generated traces while effectively enhancing downstream deep learning-based SCA performance. Experimental results demonstrate that LEAKDIT successfully improves CNN-based SCA in terms of guessing entropy (GE) and also produces reasonable visualization results.

Received 27 October 2025; Revised 25 November 2025. This work was supported by the Agency for Defense Development by the Korean government (No. UG233015TD). *(Corresponding author: Daehyeon Bae.)*

Insup Lee, Daehyeon Bae, and Sangjin Lee are with the School of Cybersecurity, Korea University, Seoul 02841, Republic of Korea (e-mail: islee94@korea.ac.kr; dh_bae@korea.ac.kr; sangjin@korea.ac.kr).

Seokhie Hong was with the School of Cybersecurity, Korea University, Seoul 02841, Republic of Korea. He is now the CONTEC, Daejeon 34129, Republic of Korea (e-mail: shhong@contec.kr).

II. BACKGROUND AND MOTIVATION

This section describes the preliminaries for profiled SCA and the dataset preparation process.

A. Profiled Side-Channel Analysis

SCA is typically divided into two types: non-profiled and profiled SCA. In non-profiled SCA, the adversary directly applies techniques to traces from the target device without prior training on a clone (*profiling*) device. In contrast, profiled SCA assumes a strong attacker with access to a profiling device of the target, enabling the training of a leakage model transferable to the target device. Profiled SCA consists of two phases: profiling and attack. In the profiling phase, the adversary collects labeled traces on the profiling device under controlled inputs and known keys, then trains a template model. In the attack phase, the adversary applies the trained profile to traces from the target device, computes likelihoods for key candidates, and recovers the secret key.

B. ASCAD Dataset

We employ the ASCAD dataset [4], widely regarded as a benchmark for evaluating deep learning-based SCA. The dataset contains electromagnetic traces collected from an ATMega8515 (8-bit AVR microcontroller) executing AES with

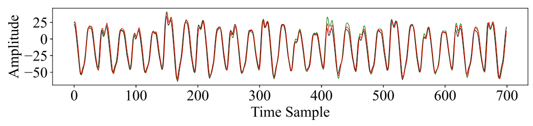

Fig. 1. Illustration of traces from the ASCAD$^{\text{sync}}$ dataset.

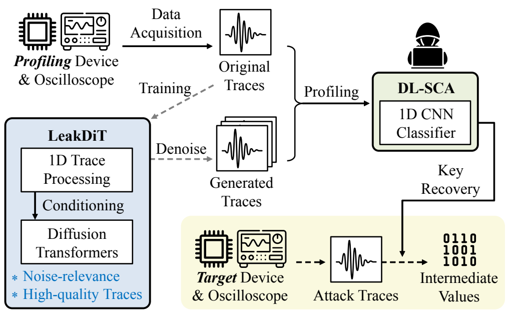

Fig. 2. System overview. Original electromagnetic traces are collected from a profiling device and oscilloscope. LEAKDIT trains on these original traces and then denoises them using diffusion transformers to produce high-quality synthetic samples. The combination of real and generated traces is then used to train a deep learning-based SCA classifier for effective key recovery.

first-order Boolean masking. We target the intermediate value corresponding to the third byte of the AES S-box output in the first round. The AES S-box is the nonlinear byte substitution in SubBytes. Each trace is labeled with the unmasked value $\text{sbox}(p[3] \oplus k[3])$, where p and k are the plaintext and the corresponding key byte, respectively. This leads to a 256-class classification problem over byte values 0-255. We use traces from the ASCAD fixed-key dataset under different jitter levels: a jitter-free ASCAD$^{\text{sync}}$ and a desynchronized ASCAD$^{\text{desync100}}$. As illustrated in Fig. 1, ASCAD traces are one-dimensional time series of 700 samples, where each sample corresponds to an electromagnetic measurement point. By default, we use the standard split of 50,000 profiling and 10,000 attack traces.

III. DESIGN OF LEAKDIT

As shown in Fig. 2, we propose LEAKDIT, a generative model with a domain-specific loss that produces electromagnetic traces to mitigate data scarcity and imbalance in deep learning-based SCA. After training LEAKDIT on the original profiling traces, we generate high-fidelity traces through a denoising process, which enhances overall SCA performance. We address class imbalance in the ASCAD dataset (Fig. 3), where the maximum count for label 0x67 is 244 and the minimum for 0xD5 is 154. In our experiments, we balance this distribution by oversampling minority classes to 244 (e.g., generating 90 traces for 0xD5). This section describes the architectural details and the training process of LEAKDIT.

A. LEAKDIT *Architecture*

LEAKDIT is a one-dimensional diffusion transformer (DiT) that models and generates electromagnetic traces. It partitions each input trace $\mathbf{x}_0 \in \mathbb{R}^{700}$ into non-overlapping patches and linearly projects them into an embedding space. We adopt fixed one-dimensional sinusoidal positional embeddings

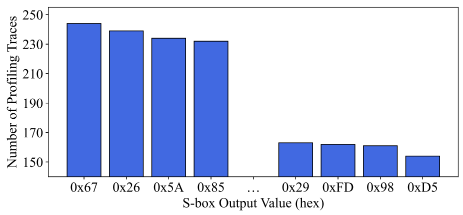

Fig. 3. Distribution of profiling traces according to S-box output value in the ASCAD dataset.

to preserve temporal information within the patch sequence. For conditional generation, we inject class-conditioning embeddings into the transformer backbone. The backbone of LEAKDIT follows the DiT architecture [11] and consists of stacked transformer blocks with adaptive layer normalization. Unlike latent diffusion models, LEAKDIT operates directly on raw traces without relying on any pretrained autoencoder.

B. *Electromagnetic Trace Augmentation*

The generative process follows the denoising diffusion probabilistic model (DDPM), which defines a forward process that gradually corrupts a clean trace $\mathbf{x}_0$ into a noisy trace $\mathbf{x}_t$ as follows:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{1}$$

where t is the diffusion timestep and $\bar{\alpha}_t$ is the cumulative noise schedule. The reverse process trains the network to iteratively denoise and reconstruct the original distribution as follows:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t), \Sigma_\theta(\mathbf{x}_t)), \tag{2}$$

where $\mu_\theta(\mathbf{x}_t)$ and $\Sigma_\theta(\mathbf{x}_t)$ are the mean and covariance estimated by the network, respectively. The trace augmentation process consists of two phases: training and sampling. In the training phase, given a training dataset $\mathcal{D}$ with traces $\mathbf{x}$ and class labels c (S-box output ranging from 0 to 255), we initialize the parameters θ of LEAKDIT. We then precompute the noise schedule $\{\bar{\alpha}_t\}_{t=1}^T$, which controls the incremental corruption applied to each trace. The network learns to predict the added noise ϵ, where $\epsilon \sim \mathcal{N}(0, I)$, by minimizing the mean squared error (MSE) as follows:

$$\mathcal{L}_{\text{MSE}} = \|\hat{\epsilon}_\theta - \epsilon\|^2, \tag{3}$$

where $\hat{\epsilon}_\theta$ is the predicted noise. To capture side-channel characteristics, we extend the diffusion objective by adding a domain-specific loss based on normalized inter-class variance (NICV), defined as follows:

$$\text{NICV}(\mathbf{x}, c) = \frac{\text{Var}[\mathbb{E}[\mathbf{x} \mid c]]}{\text{Var}[\mathbf{x}]}, \tag{4}$$

which quantifies the proportion of class-dependent variance relative to the total variance. In practice, we compute an empirical estimate of Eq. (4) over time indices using batches of traces. This estimate remains consistent with the original definition and ensures that the generated traces preserve the leakage structure observed in real measurements. Given real

TABLE I
TRAINING PARAMETERS OF THE PROPOSED LEAKDIT MODEL

Parameters	Value
Optimizer	AdamW
Learning rate	1e-4
Training epochs	160
Noise schedule	linear variance schedule
Classifier-free guidance scale	0.15
Diffusion timesteps	250

TABLE II
ARCHITECTURE OF THE ADOPTED CNN-BASED SCA CLASSIFIER

Layer Type	Structure
-	**Input** (1D electromagnetic traces)
Convolutional layer	Conv1D (64, kernel=11) + ReLU + AvgPool1d
Convolutional layer	Conv1D (128, kernel=11) + ReLU + AvgPool1d
Convolutional layer	Conv1D (256, kernel=11) + ReLU + AvgPool1d
Convolutional layer	Conv1D (512, kernel=11) + ReLU + AvgPool1d
Convolutional layer	Conv1D (512, kernel=11) + ReLU + AvgPool1d
-	Flatten
Fully-connected layer	Linear (4096) + ReLU
Fully-connected layer	Linear (4096) + ReLU
Fully-connected layer	Linear (256)
-	**Output** (predicted S-box output value)

traces $\mathbf{x}$ and reconstructed clean traces $\hat{\mathbf{x}}_0$ predicted by the reverse process, the NICV alignment loss is defined as follows:

$$\mathcal{L}_{\text{NICV}} = \left\| \text{norm}(\text{NICV}(\hat{\mathbf{x}}_0, c)) - \text{norm}(\text{NICV}(\mathbf{x}, c)) \right\|^2, \quad (5)$$

where norm$(\cdot)$ is ℓ_2 normalization over time indices. The final training objective combines both terms as follows:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{NICV}} \mathcal{L}_{\text{NICV}}, \quad (6)$$

where λ_{NICV} balances the contribution of the NICV regularizer. In our experiments, we set λ_{NICV} to 2.0 and 0.3 for ASCAD$^{\text{sync}}$ and ASCAD$^{\text{desync100}}$, respectively, achieving a stable balance between reconstruction fidelity and leakage preservation. In the sampling phase after training, we draw an initial Gaussian noise vector and iteratively denoise it with the learned reverse process to reconstruct clean traces. We adopt classifier-free guidance with scale s [11] to generate category-specific traces, as discussed in the next section.

IV. PERFORMANCE EVALUATION

For reproducibility, we use the ASCAD dataset [4] as described in Section II-B. We follow the diffusion hyperparameters of DiT [11] as summarized in Table I to implement LEAKDIT on ASCAD$^{\text{sync}}$, and set the number of epochs to 140 and the guidance scale to 0.2 for ASCAD$^{\text{desync100}}$. For comparison, we evaluate SCA models augmented with LEAKDIT against models trained on traces augmented by cGAN and DiT, a state-of-the-art variant of diffusion models. We assess performance using two SCA metrics: guessing entropy (GE) and Nt_{GE} [12], where GE is the rank of the correct key byte among 256 candidates and Nt_{GE} is the number of required traces to consistently achieve GE $= 0$. For the baseline classifier, as shown in Table II, we design a CNN with five convolutional and three fully connected layers. The classifier outputs logits over 256 S-box values, and we train it for 30 epochs, achieving the best GE performance as illustrated in Fig. 4. We adopt 100-fold cross-validation to ensure reliable evaluation. We implement the model in PyTorch and run it on an NVIDIA GeForce RTX 5090 GPU.

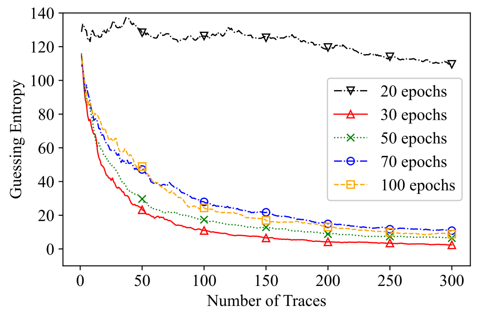

Fig. 4. Guessing entropy under different training epochs of SCA classifier.

TABLE III
NUMBER OF REQUIRED TRACES FOR KEY RECOVERY AFTER TRACE AUGMENTATION

	Original	cGAN	DiT	LEAKDIT
Nt_{GE}	1,390	1,470	1,520	**1,010**
Dataset Size (Profiling)	50,000	62,464	62,464	62,464

As shown in Table III and Fig. 5, we evaluate the impact of trace augmentation on SCA performance. Each generator produces 12,464 additional traces, for a total of 62,464 profiling traces. Overall, LEAKDIT performs better than the DiT baseline without NICV, indicating the benefit of the NICV term. Table III shows that LEAKDIT achieves the best results on ASCAD$^{\text{sync}}$, reducing Nt_{GE} from 1,390 to 1,010 (27.34% improvement). In contrast, we observe that Nt_{GE} increases after augmentation with cGAN and DiT, indicating that data augmentation itself does not guarantee performance improvement. For ASCAD$^{\text{desync100}}$, GE did not reach 0 within the available attack traces in our experiments, so we could not determine Nt_{GE} and instead focus on GE trends as shown in Fig. 5. Fig. 5(a) shows that LEAKDIT reduces GE faster under jitter-free conditions, compared with the baselines. Although Fig. 5(b) shows that cGAN performs slightly better than LEAKDIT under jittery conditions with fewer than 1,000 attack traces, cGAN appears to underrepresent intra-class jitter, which will be detailed in the next paragraph.

As shown in Fig. 6, we visualize ASCAD$^{\text{sync}}$ and ASCAD$^{\text{desync100}}$, with three traces for each byte class (0x00, 0xCC). For ASCAD$^{\text{sync}}$, Fig. 6(a) indicates extensive inter-class overlap; byte-dependent differences remain subtle and concentrate near the *rising* clock edges. Fig. 6(b) presents cGAN-generated traces that exhibit stronger amplitude distortions than the original, whereas Fig. 6(c) demonstrates that LEAKDIT follows the temporal patterns of the original data and preserves realistic inter-class overlap. For ASCAD$^{\text{desync100}}$, Fig. 6(d) shows strong desynchronization, which complicates modeling. Fig. 6(e) shows that cGAN captures inter-class differences but fails to reproduce intra-class jitter. Meanwhile, Fig. 6(f) presents synthetic traces from LEAKDIT that follow

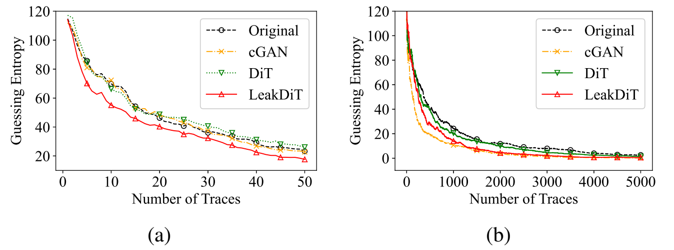

Fig. 5. Guessing entropy under trace augmentation using different methods on (a) ASCAD$^{\text{sync}}$ and (b) ASCAD$^{\text{desync100}}$ datasets.

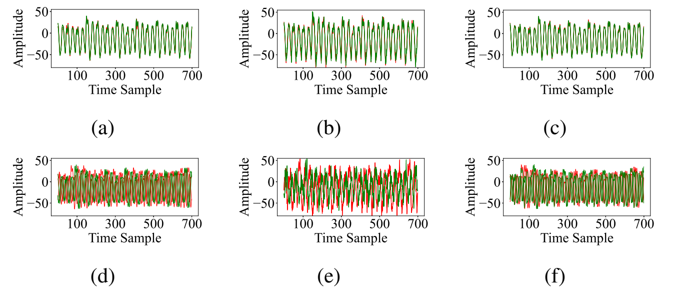

Fig. 6. Visualization of ASCAD$^{\text{sync}}$: (a) original, (b) cGAN, (c) LEAKDIT; and ASCAD$^{\text{desync100}}$: (d) original, (e) cGAN, (f) LEAKDIT.

the desynchronized timing patterns and reflect both inter-class and intra-class variability, matching the characteristics of the original set more closely.

To assess feasibility and practicality, we present training and generation times for 12,464 traces using cGAN, DiT, and LEAKDIT, as summarized in Table IV. cGAN trains and samples faster than diffusion-based models (DiT and LEAKDIT), but the higher fidelity observed in Fig. 6 justifies the additional computational cost. Because training and generation run offline, they do not introduce latency during the attack phase. Therefore, LEAKDIT remains a practical choice for producing high-quality traces that address data insufficiency in deep learning-based SCA.

Finally, we examine the effect of the classifier-free guidance scale s [11], which determines the conditioning level during generation. Since s controls the trade-off between trace quality and diversity, we adjust the scale to improve the SCA performance with LEAKDIT. Fig. 7 shows Nt_{GE} on ASCAD$^{\text{sync}}$ after profiling trace augmentation with LEAKDIT using its guidance scales ranging from 0.05 to 1.0. In our experiments, $s = 0.15$ achieves the lowest Nt_{GE} of 1,010, while $s = 0.1$ and $s = 0.2$ result in Nt_{GE} of 1,720 and 1,130, respectively. These results highlight the importance of selecting an appropriate hyperparameter to improve deep learning-based SCA performance.

V. CONCLUSION

This letter presented LEAKDIT, which addresses trace data insufficiency and class imbalance in deep learning-based SCA. To generate high-quality traces with a domain-specific generative model, we proposed an NICV alignment loss and integrated it into a one-dimensional DiT. Experimental results demonstrated that LEAKDIT reduces Nt_{GE} by 27.34% compared to the baseline, producing visually reasonable traces.

TABLE IV
FEASIBILITY ANALYSIS

Generator	Training Time/Epoch (s)	Training Epochs	Training Total Time (s)	Generation Total Time (s)
cGAN	1.53	200	306	0.032
DiT	156.99	140	21,979	2,937
LEAKDIT	159.98	160	25,597	2,947

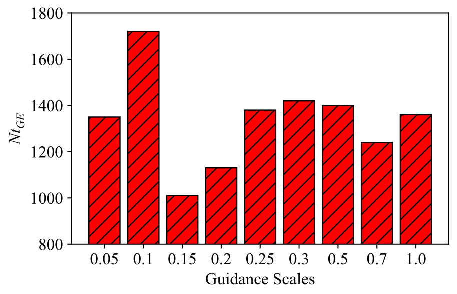

Fig. 7. Number of required traces for key recovery according to the guidance scales of LEAKDIT.

REFERENCES

[1] S. Maji, K. Lee, and A. P. Chandrakasan, "SparseLeakyNets: Classification prediction attack over sparsity-aware embedded neural networks using timing side-channel information," *IEEE Computer Architecture Letters*, vol. 23, no. 1, pp. 133–136, 2024.

[2] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing," in *Prof. of International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2017, pp. 45–68.

[3] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 148–179, 2019.

[4] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *Journal of Cryptographic Engineering*, vol. 10, no. 2, pp. 163–188, 2020.

[5] P. Wang, P. Chen, Z. Luo, G. Dong, M. Zheng, N. Yu, and H. Hu, "Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks," *arXiv preprint arXiv:2007.05285*, 2020.

[6] N. Mukhtar, L. Batina, S. Picek, and Y. Kong, "Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices," in *Proc. of Cryptographers' Track at the RSA Conference*, 2022, pp. 297–321.

[7] S. Karayalçın, M. Krček, L. Wu, S. Picek, and G. Perin, "It's a kind of magic: A novel conditional GAN framework for efficient profiling side-channel analysis," in *Prof. of Asiacrypt*, 2024, pp. 99–131.

[8] T. Yap and D. Jap, "Creating from noise: Trace generations using diffusion model for side-channel attack," in *Prof. of International Conference on Applied Cryptography and Network Security*, 2024, pp. 102–120.

[9] Z. Lu, Y. Longde, W. Fei, C. Aidong, Y. Ning, L. Xiang, Z. Jiancheng, Z. Yanlong, W. Shuo, and Z. Jing, "SCARefusion: Side channel analysis data restoration with diffusion model," *Microelectronics Journal*, vol. 156, p. 106546, 2025.

[10] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. of NeurIPS*, 2021, pp. 8780–8794.

[11] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proc. of ICCV*, 2023, pp. 4195–4205.

[12] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 1–36, 2020.