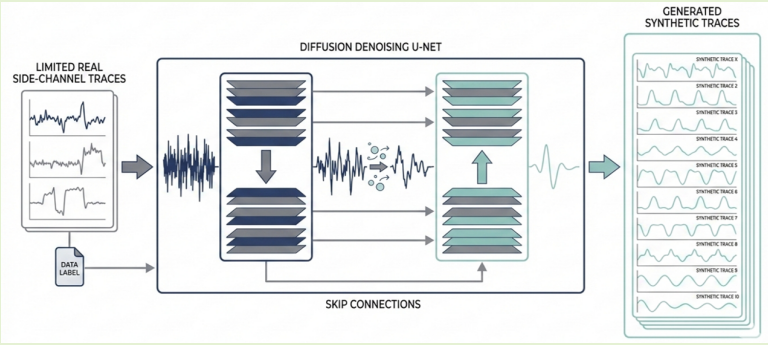
High-Fidelity Conditional Side-Channel Trace Synthesis using Diffusion Models

Zekai Zhang, Donglong Chen, *Member, IEEE*, Wangchen Dai, Jinfa Hong, Yu Hin Chan, Çetin Kaya Koç, *Life Fellow, IEEE*, Patrick S.Y. Hung and Ray C.C. Cheung, *Senior Member, IEEE*

***Abstract*— The evaluation of hardware security through physical side-channel measurements heavily relies on the volume and diversity of acquired sensor data, encompassing sufficient time-series instances for varying operational states. Synthesizing high-fidelity artificial measurement signals offers a compelling solution to augment limited empirical sensor data; however, manual crafting is infeasible, and existing automated generative approaches often fail to capture complex physical noise profiles. Recently, Denoising Diffusion Probabilistic Models (DDPMs) have emerged as superior alternatives to Generative Adversarial Networks (GANs) for modeling realistic, noisy data distributions. This paper explores the application of DDPMs for synthesizing side-channel sensor traces. We propose a novel conditional diffusion framework utilizing a specialized U-Net architecture, equipped with residual blocks and self-attention mechanisms, to learn and replicate complex temporal measurement dynamics. Trained on pre-processed measurement traces acquired via the industry-standard ChipWhisperer sensing platform, our model enables precise, conditional signal generation corresponding to specific hardware states. A critical advantage of our approach is its remarkable data efficiency: the proposed model requires merely 2,560 empirical measurement traces for training, highlighting the practical data efficiency of the proposed framework; however, such cross-study comparisons should be interpreted with caution when the experimental settings are not identical. Experimental results demonstrate the high physical fidelity of the synthesized signals. A classifier trained on empirical measurements achieves 97.7% label consistency accuracy when evaluated on our synthetic traces, closely rivaling the 99.6% SOTA accuracy and providing strong evidence that critical physical leakage characteristics are preserved under the evaluated setting. Furthermore, in hardware vulnerability assessments, analytical models augmented with our synthetic data achieve a Guessing Entropy of one, matching best-in-class performance. In simulated evaluation scenarios using these ChipWhisperer-based traces, models trained with our synthesized physical signals achieve 91.1% accuracy in same-device characterizations and 82.4% in challenging cross-device assessments. Ultimately, these generated traces serve as a promising signal augmentation tool, significantly enhancing profiling models and providing a more rigorous evaluation of hardware security under constrained sensor data conditions.**



***Index Terms*— Side-channel attacks, Diffusion model, Deep learning (DL), Conditional generation**

This work was supported in part by the Innovation and Technology Fund under Grant ITS/109/24; in part by the CityUHK Internal Grant under Grant 9440448; in part by the National Natural Science Foundation of China under Grant 62372417; in part by the Jiangsu Province 100 Foreign Experts Introduction Plan under Grant BX2022012; in part by the Scientific Research Innovation Capability Support Project for Young Faculty under Grant SRICSPYF-BS2025137; in part by the Guangdong Provincial Key Laboratory of IRADS under Grant 2022B1212010006; in part by the Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme under Grant 2025A0505000001; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515011274; in part by the Guangdong Province General Universities Key Field Project (New Generation Information Technology) under Grant 2023ZDZX1033; in part by the UIC Research Grant under Grant UICR04202401-21.

Z. Zhang, J. Hong, Y. Chan, P.S.Y. Hung and R.C.C. Cheung are with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR, China. E-mail: zekazhang2-c@my.cityu.edu.hk, jinfahong2-c@my.cityu.edu.hk, yhchan96@cityu.edu.hk, psyhung@cityu.edu.hk, r.cheung@cityu.edu.hk.

I. Introduction

SIDE-channel attacks (SCAs) exploit unintended physical emanations—such as power consumption and electromagnetic (EM) radiation—measured by external sensors to compromise the security of cryptographic devices [1]. The acquisition of these physical leakages relies heavily on precise

D. Chen is with Guangdong Provincial/Zhuhai Key Laboratory of IRADS, Beijing Normal-Hong Kong Baptist University, Zhuhai 519088, China. E-mail: donglongchen@bnbu.edu.cn.

W. Dai is with Sun Yat-sen University, Shenzhen, China. (e-mail: w.dai@my.cityu.edu.hk).

Çetin Kaya Koç is with Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China, and also with University of California, Santa Barbara, Santa Barbara, CA 93106 USA. (e-mail: cetinkoc@ucsb.edu).

Corresponding author: Donglong Chen.

sensing instrumentation (e.g., high-resolution oscilloscopes, near-field EM probes, and current sensors), which capture the temporal physical variations as time-series measurement signals, commonly referred to as traces. By applying signal processing and statistical analysis to correlate these sensed traces with internal data states, sensitive information can be extracted. Standard analytical methodologies, including Differential Power Analysis (DPA) [2] and Correlation Power Analysis (CPA) [3], alongside advanced profiling techniques [4], inherently depend on the fidelity and volume of the acquired sensor data.

A critical factor dictating the efficacy of these trace-based analyses, particularly in profiling scenarios, is the quantity and diversity of the acquired measurement data [5]. Constructing robust leakage models necessitates large datasets encompassing various operational states. However, practical sensor data acquisition is frequently impeded by physical and operational constraints. Target devices often incorporate physical countermeasures that degrade the signal-to-noise ratio (SNR), while restricted access interfaces limit the measurement duration and the volume of capturable signals. This inherent scarcity of physical measurement data severely restricts the generalization capability of analytical models, making it difficult to accurately evaluate the true vulnerability of the sensed hardware under real-world noise conditions.

To mitigate these measurement limitations, the generation of high-fidelity synthetic sensor data has emerged as a crucial technique for signal augmentation. Accurately replicating the complex physical characteristics, hardware-specific signatures, and environmental noise inherent in empirical sensor measurements through manual crafting is highly impractical. Consequently, data-driven approaches leveraging generative machine learning have gained traction to automate the synthesis of realistic physical signals. While Generative Adversarial Networks (GANs) [6] have been adapted to simulate SCA measurement traces [7]–[9], they frequently struggle with training instability and mode collapse, often failing to fully capture the intricate temporal dynamics and varying noise profiles characteristic of actual hardware sensor readings [10], [11].

Recently, Denoising Diffusion Probabilistic Models (DDPMs) [12], [13] have demonstrated superior performance in generative modeling, offering enhanced stability and the ability to synthesize highly realistic data distributions by modeling the progressive addition and removal of noise. Given their inherent mechanism of iteratively denoising signals, DDPMs present a highly suitable architecture for modeling the noisy, time-series nature of physical side-channel measurements.

Motivated by these advantages, this study investigates the application and optimization of DDPMs for the synthesis of side-channel sensor traces. Our research specifically focuses on conditional signal generation, enabling the precise synthesis of high-fidelity physical measurements that accurately correspond to specific operational states and cryptographic labels. The main contributions of this work are summarized as follows:

1) We propose a novel, data-efficient framework for high-fidelity side-channel trace synthesis, centered on a conditional Denoising Diffusion Probabilistic Model (DDPM). Requiring a remarkably small number of training traces, our specialized U-Net architecture—enhanced with residual blocks and self-attention—is uniquely tailored to capture complex temporal dependencies and subtle leakage features from just 2,560 raw traces **acquired via the industry-standard ChipWhisperer platform**.
2) We provide quantitative evidence that our model generates traces with exceptional fidelity and leakage preservation. This is demonstrated by a multi-faceted evaluation: our synthetic traces achieve 97.7% accuracy on label consistency, which are statistically hard to distinguish from **real hardware traces** (as shown by a near-random distinguishability accuracy of 0.55), and attain a low FID-like score of 0.08, indicating a close distributional match to the original data.
3) We demonstrate the practical utility and attack potential of the synthetic data in realistic downstream scenarios **evaluated on the ChipWhisperer architecture**. A model trained solely on our generated traces achieves high attack success rates, with 91.1% accuracy in a same-device setting and 82.4% in a challenging cross-device scenario **involving distinct target boards**. Crucially, the high quality of the generated leakage allows for a successful attack, reducing the guessing entropy to one. This validates that our approach captures fundamental, transferable leakage patterns, making it a powerful tool for robust data augmentation. The project code will be released upon acceptance.

II. Related Works

The challenge of limited data availability is a well-recognized hurdle in side-channel analysis (SCA), often restricting the efficacy of profiling attacks [5]. While early data augmentation techniques, such as adding noise [7] or over-sampling with SMOTE [14], provided some benefits, they often fail to capture the complex noise distributions and intricate leakage patterns of real-world measurements.

To address these limitations, research has shifted towards deep generative models for synthesizing entirely new, realistic traces. Generative Adversarial Networks (GANs) and their conditional variants (CGANs) have been a predominant approach. For instance, Gao et al. [15] employ a GAN with Transfer Learning to rapidly simulate EM leakage, reducing the data needed to evaluate new chip designs. Works by [16], [17], and [9] have demonstrated the ability of GAN-based models to generate labeled traces for augmenting datasets and enhancing attack performance. However, a common theme across these methods is their reliance on large datasets for training, often requiring tens of thousands of traces (e.g., 47,500 traces in [9]) to achieve high fidelity.

More recently, Denoising Diffusion Probabilistic Models (DDPMs) [12] have emerged as a powerful alternative, showing state-of-the-art performance in other domains [18] due to their stable training and high-quality sample generation. Their potential in SCA has been explored by recent studies.

TABLE I: Notation list.

Symbol	Description	Symbol	Description
$\mathbf{x}_0$	Original clean input feature vector	$\mathbf{x}_t$	Noisy feature vector at timestep t
$\mathbf{x}_T$	Final noisy vector sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$	c	Integer class label for conditional generation
c_{target}	Target label during conditional generation	$\boldsymbol{\epsilon}$	Gaussian noise added in forward process
$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, c, t)/\boldsymbol{\epsilon}_{\text{pred}}$	Predicted noise by neural network	$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$	Predicted mean of reverse distribution
$\boldsymbol{\theta}$	Parameters of the U-Net model	$\mathbf{z}$	Sampled noise for reverse step (if $t > 1$)
α_t	Defined as $1 - \beta_t$	$\bar{\alpha}_t$	Cumulative product $\prod_{i=1}^{t} \alpha_i$
β_t	Noise variance at timestep t	$\tilde{\beta}_t$	Alternative reverse variance term
σ_t	Standard deviation in reverse step, $\sqrt{\beta_t}$	σ_t^2	Variance of reverse process (can be fixed or learned)
L	Length of feature vector (e.g., 200)	F	Number of channels (typically $F = 1$)
$\boldsymbol{\mu}$	Mean vector used for normalization	$\boldsymbol{\sigma}$	Std. dev. vector for normalization
$\mathbf{x}_{0,\text{norm}}$	Normalized input using Z-score	G	Final denormalized generated feature vector
$\boldsymbol{\mu}_{\text{raw}}$	Mean of raw traces (before pre-trained model)	$\boldsymbol{\sigma}_{\text{raw}}$	Std. dev. of raw traces
$\boldsymbol{\mu}_{\text{feat}}$	Mean of feature vectors after pre-trained model	$\boldsymbol{\sigma}_{\text{feat}}$	Std. dev. of extracted features
f_{feat}	Pre-trained model feature extractor	D_{raw}	Dataset of raw power traces
D_{feat}	Dataset of model extracted feature vectors	$L_{\text{simple}}(\boldsymbol{\theta})$	Simplified MSE loss for training
$\mathcal{U}(1, T)$	Uniform distribution over timesteps	$\mathbf{I}$	Identity matrix used in covariance terms

Yap et al. [19] and Karayal et al. [20] successfully adapted DDPMs to generate synthetic side-channel traces, preserving leakage information and achieving high classification accuracy. A notable limitation, however, is that these pioneering DDPM-based methods remain highly data-intensive, demanding massive training sets of 71,168 traces [19] and 63,750 traces [20], respectively. This significant data requirement poses a practical barrier to their adoption.

In contrast to these data-hungry approaches, this work investigates a data-efficient DDPM for high-fidelity trace synthesis. We aim to demonstrate that it is possible to generate traces that preserve critical leakage information to a high degree, even when trained on a substantially smaller dataset. By developing a conditional model capable of generating traces for the full range of cryptographic labels from limited data, we offer a more practical and accessible tool for robust data augmentation in SCA.

III. Preliminary

To facilitate clarity and reproducibility, we summarize all key mathematical symbols used throughout the methodology and modeling sections in Table I. This symbol table consolidates notations related to the forward diffusion process, reverse denoising model, conditional embeddings, training pipeline, and generation algorithm. It serves as a concise reference to the variables, model components, and distributions involved in the design and implementation of the conditional DDPM framework. Each symbol is accompanied by a brief description and its contextual relevance, allowing readers to better follow the technical formulation and algorithmic steps.

IV. Denoising Diffusion Probabilistic Models (DDPMs)

DDPMs [12] represent a powerful class of deep generative models, drawing inspiration from concepts in non-equilibrium thermodynamics. They have demonstrated remarkable success in generating high-fidelity data, particularly in the image domain, often matching or exceeding the quality of Generative Adversarial Networks (GANs) [18]. The core idea involves systematically destroying structure in data through a fixed forward diffusion process and then learning a reverse denoising process to generate data.

The DDPM framework consists of two main processes: a fixed forward diffusion process and a learned reverse denoising process.

A. Forward Diffusion Process

The forward process, denoted by q, gradually adds Gaussian noise to an initial data sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ over T discrete timesteps. This process is defined as a Markov chain where the variance of the added noise at each step t is controlled by a predefined variance schedule $\{\beta_t\}_{t=1}^{T}$, with $0 < \beta_1 < \cdots < \beta_T < 1$. The transition probability is given by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{1}$$

where $\mathbf{I}$ is the identity matrix. A key property of this process is that we can sample $\mathbf{x}_t$ directly from $\mathbf{x}_0$ in closed form. Letting $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, we have:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{2}$$

As $t \to T$, if the schedule $\{\beta_t\}$ and the number of steps T are chosen appropriately, $\mathbf{x}_T$ becomes approximately distributed according to a standard isotropic Gaussian distribution, $\mathbf{x}_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$, effectively destroying the original data structure.

B. Reverse Denoising Process

The generative process involves reversing the diffusion. Starting from noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the model learns to gradually denoise it step-by-step to produce a sample $\mathbf{x}_0$. This corresponds to learning the reverse transition probabilities $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$. While the true reverse transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is tractable and Gaussian, calculating $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ requires marginalizing over all possible $\mathbf{x}_0$, which is intractable.

DDPMs approximate the reverse transition using a parameterized Gaussian distribution, typically sharing parameters across time steps:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \tag{3}$$

Here, $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ is the mean function learned by a neural network (often a U-Net architecture), parameterized by $\boldsymbol{\theta}$. The variance σ_t^2 is typically fixed to a time-dependent constant, often related to the forward process variances, such as $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Ho et al. [12] showed that instead of predicting the mean $\boldsymbol{\mu}_\theta$, the network can be trained to predict the noise component $\boldsymbol{\epsilon}$ that was added at step t. The mean can then be expressed as:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{4}$$

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is the output of the neural network parameterized by $\boldsymbol{\theta}$, aiming to predict the noise $\boldsymbol{\epsilon}$ used to generate $\mathbf{x}_t$ from $\mathbf{x}_0$ via Equation 2.

C. Learning Objective

The parameters $\boldsymbol{\theta}$ of the reverse process network are learned by optimizing a surrogate objective derived from the variational lower bound (VLB) on the data log-likelihood. In practice, DDPMs often employ a simplified objective function which has proven effective. This objective trains the network $\boldsymbol{\epsilon}_\theta$ to predict the noise that was added to an original data sample. Specifically, for a data sample $\mathbf{x}_0$ drawn from the true data distribution $q(\mathbf{x}_0)$, a timestep t sampled uniformly from $\{1, \ldots, T\}$, and Gaussian noise $\boldsymbol{\epsilon}$ sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the network is given the noisy input $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$. The optimization then aims to minimize the expected mean squared error between the true noise $\boldsymbol{\epsilon}$ and the noise predicted by the network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$:

$$L_{\text{simple}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(1,T),\mathbf{x}_0\sim q(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)||^2\right] \tag{5}$$

Here, the expectation $\mathbb{E}$ averages over the random choices of the timestep t, the initial data sample $\mathbf{x}_0$, and the noise sample $\boldsymbol{\epsilon}$. Essentially, the network learns to denoise $\mathbf{x}_t$ by predicting the noise component that corrupted $\mathbf{x}_0$.

V. Methodology

This work employs a conditional DDPM to generate synthetic time-series power traces conditioned on specific labels. The core idea is to learn a reverse process that can denoise a standard Gaussian noise input into a realistic power trace, guided by the conditional label information. The methodology encompasses data preprocessing, the diffusion model formulation, the network architecture, the training procedure, and the trace generation process.

A. Preprocessing

The primary dataset consists of physical power consumption traces acquired via the industry-standard ChipWhisperer platform. For computational processing, the data is structured such that each instance comprises an integer label (c) denoting its cryptographic class, followed by the corresponding time-series data points. Comprehensive details regarding the dataset properties are provided in the Experiments and Results section (see *Dataset*). To distill characteristics and potentially mitigate inherent measurement noise from these raw hardware traces, we utilize a pre-trained model as a feature extractor.

As shown in Algorithm 1, the model was independently pre-trained on a representative dataset (specified as 256 distinct traces, each paired with its ground-truth label c) using a standard supervised classification objective. The goal was to train the network to accurately predict the label c associated with an input raw trace, thereby encouraging the network's intermediate layers to learn discriminative features relevant to the underlying process generating the trace. For each normalized raw trace in our main dataset, we perform a forward pass through the pre-trained model and extract the activation vector from a designated intermediate layer, typically the penultimate layer (i.e., the input to the original classification layer).

These feature vectors, which represent a learned latent embedding of the original traces, form the actual input data for the subsequent diffusion modeling stage. Let the dimension of these feature vectors be L. To capture the most critical cryptographic leakage, the sequence length is carefully determined based on the Points of Interest (POIs) associated with the non-linear S-box computation. Consequently, all traces are standardized to a fixed length of $L = 200$. This specific dimension provides an optimal alignment window that fully encompasses the S-box leakage characteristics while effectively filtering out irrelevant background operations and measurement noise. For the standardization process, traces shorter than L are right-padded with zeros, while traces longer than L are truncated to this exact length L. For model input, each trace $\mathbf{x}_0$ is treated as a sequence of length L with a single channel, resulting in an input shape of $(L, 1)$. Prior to training, the trace dataset is normalized using Z-score normalization, applied feature-wise (i.e., per time step across all traces). The mean ($\boldsymbol{\mu}$) and standard deviation ($\boldsymbol{\sigma}$) are calculated across the training dataset for each of the L time points:

$$\mathbf{x}_{0,\text{norm}} = \frac{\mathbf{x}_0 - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \tag{6}$$

where the division is element-wise. A small epsilon is implicitly added to the standard deviation to prevent division by zero for time points with no variance. The calculated $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ vectors (each of shape $(1, L)$) are stored, as they are required later to denormalize the generated traces back to their original scale via the denormalize data function.

B. Conditional DDPM + U-Net Framework

The diffusion model operates in two phases: a fixed forward noising process and a learned reverse denoising process. This part has been mentioned before, so in the methodology section, we focus on the design of the model network architecture. The overview of the proposed Conditional DDPM + U-Net Framework shown in Fig. 1.

The noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is implemented using a U-Net [21] architecture adapted for 1D sequences, incorporating residual connections [22] and self-attention mechanisms [23]. The network takes the noisy trace $\mathbf{x}_t$, the corresponding labels c, and the timesteps t as input.

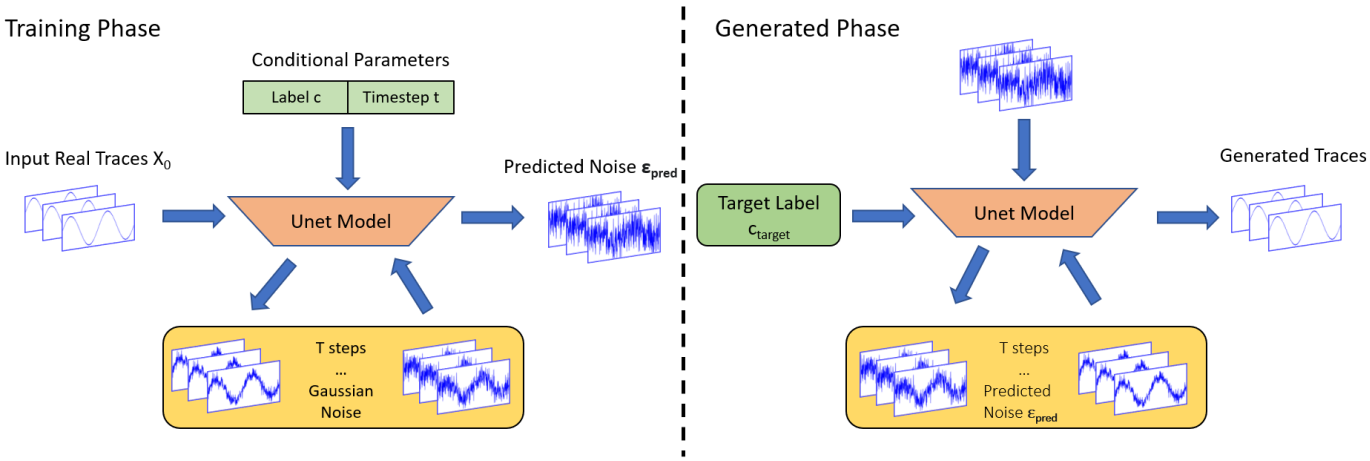


Fig. 1: The proposed Conditional DDPM + U-Net Framework.

Algorithm 1 Pre-trained feature extractor

Input: Dataset of raw traces $D_{raw} = \{T_i\}_{i=1}^{N}$, Pre-trained feature extractor f_{feat}, Raw trace normalization statistics $\boldsymbol{\mu}_{raw}, \boldsymbol{\sigma}_{raw}$.
Output: Dataset of feature vectors D_{feat}.

```
Initialize D_feat ← ∅
for i ← 1 to N do
    T_i ← D_raw[i]                          ▷ Load traces from dataset
    T_i,norm ← (T_i − μ_raw)/σ_raw          ▷ Element-wise normalization
    Tnew_i ← f_feat(T_i,norm)               ▷ Extract high-level feature vector
    D_feat ← {Tnew_i}_{i=1}^{N}             ▷ Add traces to dataset
end for
return D_feat
```

1) Conditional Embeddings: To effectively guide the denoising process towards generating feature vectors $\mathbf{x}_0$ corresponding to a specific target label c and appropriate for the current noise level defined by timestep t, the U-Net architecture incorporates two distinct conditional embeddings. We have shown this conditional input embedding in Fig. 2. These embeddings serve to inject label-specific and temporal information into the network's feature computations. Specifically, the discrete integer class label c is first transformed into a dense, continuous vector representation using an embedding layer followed by a dense projection layer, resulting in a 32 dimensional vector. Concurrently, the diffusion timestep t is converted into a continuous embedding, suitable for neural network processing, by employing sinusoidal positional encoding [23], analogous to those used in Transformer models. This time embedding, also processed by a final dense layer, yields a representation of dimension equal to 32. These label and time embeddings are then integrated into the main network, providing the crucial conditional context required by the noise predictor $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

2) U-Net Structure: The original U-Net [24] is a fully convolutional network characterized by a contracting path to capture context and a symmetric expanding path for precise

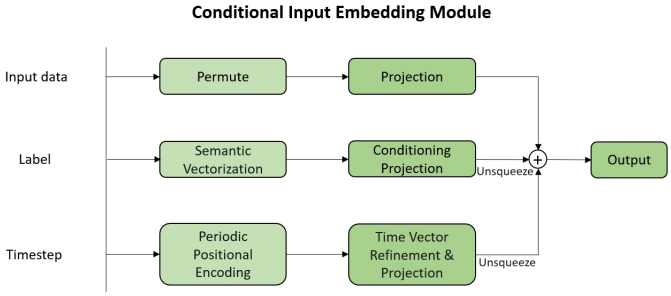


Fig. 2: The structure of conditional input embedding module.

localization. This architecture has proven highly effective in image-based tasks due to its ability to process inputs and outputs of similar spatial dimensions while leveraging skip connections to combine deep, semantic features with shallow, high-resolution features. The noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ employs a sophisticated U-Net architecture, specifically adapted for 1D sequential data like the extracted feature vectors $\mathbf{x}_t$. The Fig. 3 shown the architecture. It begins by projecting the input noisy feature vector $\mathbf{x}_t$ into a higher dimensional space with 32 channels using a 1D convolution with a kernel size of 1. Immediately following this projection, the computed label and time embeddings are integrated. These embeddings are reshaped to be broadcastable and added element-wise to the projected feature map. This early integration strategy is crucial as it allows the conditional information (label c and timestep t) to influence feature learning throughout the entire network depth, rather than being introduced only at later stages, enabling more effective conditioning of the predicted noise.

The core of the network follows the characteristic encoder-decoder structure of the U-Net. The downsampling path (encoder) progressively reduces the sequence length while increasing feature complexity and receptive field size. It consists of multiple stages, each typically comprising two `ResidualBlocks` followed by a maxpooling operation with a factor of 2. The `ResidualBlocks`, each containing two `Conv1D` layers (kernel size 5, ReLU activation) and a skip

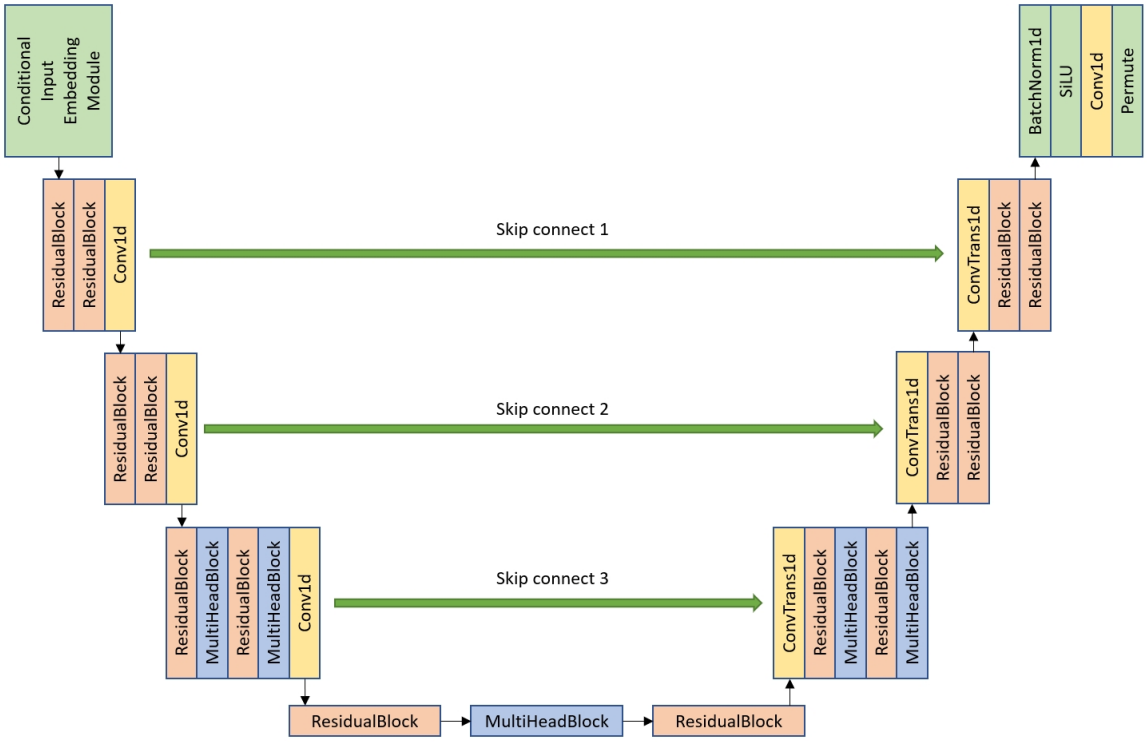


Fig. 3: The structure of U-Net

connection, are fundamental to enabling the training of a deep network by mitigating vanishing gradients and promoting feature reuse. A 1x1 convolution is strategically employed within the residual path when channel dimensions need to be adjusted, ensuring compatibility for the additive residual connection. As the network deepens, the number of channels is systematically increased (e.g., $32 \rightarrow 64 \rightarrow 128 \rightarrow 256$), allowing the network to learn increasingly complex feature representations at coarser resolutions. Furthermore, `MultiHeadAttentionBlocks` are incorporated at deeper levels within the downsampling path and the bottleneck. This inclusion is motivated by the need to explicitly model long-range temporal dependencies within the feature sequence, which might be missed by the inherently local nature of convolutions, thus providing a more global context. The bottleneck, or middle block, operates at the lowest sequence resolution and highest channel dimension, containing further `ResidualBlocks` and a `MultiHeadAttentionBlock` to process the most compressed representation of the input. The structure of ResidualBlock and MultiHeadAttentionBlock is shown in Fig. 4

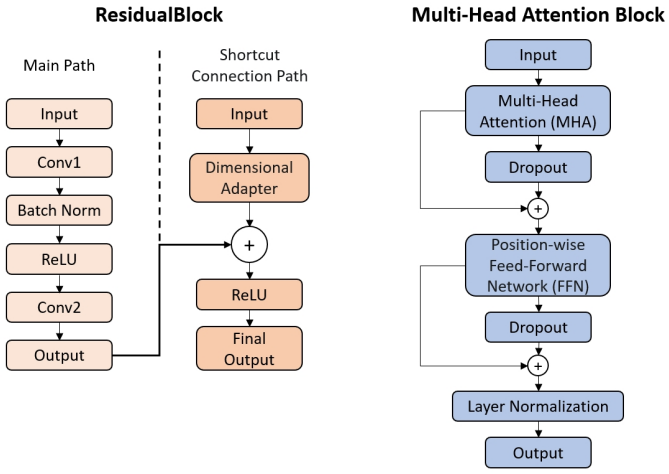


Fig. 4: The Structure of ResidualBlock and Multi-Head Attention Block

The upsampling path (decoder) symmetrically mirrors the encoder, aiming to gradually restore the original sequence length while refining the feature representations for the final noise prediction. Each stage involves an upsampling operation to increase the sequence length, followed by a crucial concatenation step. Here, the upsampled feature map is concatenated along the channel dimension with the corresponding feature map from the downsampling path via a skip connection. These skip connections are a hallmark of the U-Net design and are vital for preserving fine-grained, high-resolution details that might otherwise be lost during the downsampling process. They effectively bypass the information bottleneck, providing direct access to earlier feature maps. Before concatenation, the feature map from the skip connection passes through a dedicated 1x1 convolution to ensure its channel dimension aligns with the upsampled feature map. Following concatenation, the combined features are processed through `ResidualBlocks` and `MultiHeadAttentionBlocks`, mirroring the structure of the corresponding encoder stage, while the number of channels is progressively halved. This symmetric structure with skip connections allows the network to effectively combine high-level semantic information (from

the decoder) with low-level fine details (from the encoder).

Finally, a single `Conv1D` layer maps the refined high-dimensional feature map from the last upsampling stage back to the desired output shape, representing the predicted noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. This comprehensive U-Net design, augmented with residual connections and self-attention, provides a powerful and well-motivated architecture for the complex task of conditional noise prediction in the diffusion model.

C. Training Procedure

The core objective of the training phase is to optimize the parameters $\boldsymbol{\theta}$ of the U-Net model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ such that it accurately predicts the noise component $\boldsymbol{\epsilon}$ that was added during the forward diffusion process to corrupt an initial feature vector. Algorithm 2 demonstrated DDPM Training. This prediction must be conditioned on the resulting noisy feature vector $\mathbf{x}_t$, the corresponding label c, and the diffusion timestep t. We employ the Mean Squared Error (MSE) between the true noise $\boldsymbol{\epsilon}$ and the predicted noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, c, t)$ as the primary loss function. The MSE loss is a standard and effective choice for diffusion models, aligning well with the assumption of Gaussian noise in the underlying process and providing a clear optimization target. The formal objective is to minimize the expected squared error over the data distribution, timesteps, noise samples, and associated labels.

The model parameters $\boldsymbol{\theta}$ are updated iteratively using the Adam optimizer. Adam is selected due to its adaptive learning rate capabilities, which generally lead to faster convergence and robust performance across various tasks, making it a well-suited choice for training deep networks like our U-Net. To further enhance convergence stability and prevent overshooting minima, particularly in later training stages, a cosine decay schedule is applied to the learning rate. This schedule starts with an initial rate of 1×10^{-5} and smoothly anneals the rate towards zero over the total number of training steps, calculated across 3000 epochs.

D. Conditional Feature Vector Generation

Upon successful training, the model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ can be employed generatively to synthesize novel feature vectors conditioned on a desired target label c_{target}. Algorithm 3 demonstrated how to generate traces. This generation process effectively reverses the diffusion, starting from pure noise and progressively refining it based on the learned conditional distributions. The synthesis begins by sampling an initial state $\mathbf{x}_T$ from a standard Gaussian distribution, having the same dimensions as the feature vectors. This initialization reflects the endpoint of the forward noising process, representing maximum entropy from which structured data is to be formed. Subsequently, an iterative denoising procedure is executed, stepping backward through time from $t = T$ down to $t = 1$. In each step t, the core operation involves invoking the trained U-Net model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, c_{\text{target}}, t)$ to predict the noise that would have led to the current state $\mathbf{x}_t$. This prediction is crucially conditioned not only on the current noisy vector $\mathbf{x}_t$ and timestep t, but also on the target label c_{target}, which is embedded and supplied to the model at every step. This continuous conditioning ensures

Algorithm 2 Conditional DDPM Training

Input: Dataset $D = \{(\mathbf{x}_0, c)\}$, U-Net $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, Training settings (epochs E, steps T, schedule $\bar{\alpha}_t$, optimizer config).
Output: Optimized U-Net parameters $\boldsymbol{\theta}$.

Initialize $\boldsymbol{\theta}$. ▷ Initialize network parameters.
Initialize Optimizer and LR Scheduler.
for epoch $\leftarrow 1$ to E **do**
  Shuffle D. ▷ Shuffle data for stochasticity.
  for each batch $(\mathbf{x}_0, c)$ from D **do**
    $t \leftarrow$ `SampleTime`(T). ▷ Sample random timestep.
    $\boldsymbol{\epsilon} \leftarrow$ `SampleNoise`(shape of $\mathbf{x}_0$). ▷ Sample Gaussian noise.
    $\mathbf{x}_t \leftarrow$ `ApplyDiffusion`$(\mathbf{x}_0, \boldsymbol{\epsilon}, t, \bar{\alpha}_t)$. ▷ Create noisy sample.
    $\boldsymbol{\epsilon}_{\text{pred}} \leftarrow \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, c, t)$. ▷ Predict noise using U-Net.
    $L \leftarrow$ `ComputeLoss`$(\boldsymbol{\epsilon}, \boldsymbol{\epsilon}_{\text{pred}})$. ▷ Calculate MSE loss.
    $\boldsymbol{\theta} \leftarrow$ `OptimizeStep`$(\boldsymbol{\theta}, L)$. ▷ Update parameters via optimizer (e.g., Adam).
  end for
end for
return $\boldsymbol{\theta}$

that the denoising trajectory is steered towards generating a feature vector representative of the desired class c_{target}. The predicted noise is then used within the reverse sampling equation (detailed previously in the Reverse Process section) to estimate the less noisy vector $\mathbf{x}_{t-1}$. After T such iterative steps, the process yields the final generated feature vector $\mathbf{x}_0$ residing in the normalized feature space defined during preprocessing. To obtain the final synthetic feature vector G in its intended scale and distribution, this normalized vector $\mathbf{x}_0$ must be denormalized using the mean $\boldsymbol{\mu}_{feat}$ and standard deviation $\boldsymbol{\sigma}_{feat}$ that were computed from the training set feature vectors.

VI. Experiments and Results

In this section, we detail the experimental procedure for training and evaluating of our conditional diffusion model for trace synthesis. The process begins with loading trace segments and corresponding labels from the ChipWhisperer dataset. Initial data preprocessing, including padding traces to a uniform length of 200 points and feature-wise normalization (mean subtraction, division by standard deviation), was performed using Python libraries (NumPy, Pandas). The entirety of the model training, generation, and evaluation was conducted using Python 3, leveraging Pytorch. All computations were executed on a high-performance system equipped with an Intel® Core™ i9-14900K CPU and accelerated by an NVIDIA GeForce RTX 5090 GPU.

A. Dataset

To objectively assess our generative model and benchmark it against current state-of-the-art techniques, we deployed a specialized hardware setup for our experiments. Specifically,

Algorithm 3 Conditional DDPM Sampling

Input: Trained U-Net $\boldsymbol{\epsilon_\theta}$, Target label c_{target}, Total diffusion steps T, Diffusion schedule parameters $\{\alpha_t, \beta_t, \bar{\alpha}_t\}_{t=1}^{T}$, Feature dimensions (L, F), Normalization stats $\boldsymbol{\mu}_{feat}, \boldsymbol{\sigma}_{feat}$.

Output: Generated feature vector G.

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Start from pure noise with dimension (L, F).
for $t \leftarrow T$ down to 1 **do**
  if $t > 1$ **then**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample noise for reverse step.
  else
    $\mathbf{z} \leftarrow \mathbf{0}$
  end if
  $\boldsymbol{\epsilon}_{\text{pred}} \leftarrow \boldsymbol{\epsilon_\theta}(\mathbf{x}_t, c_{\text{target}}, t)$. ▷ Condition on $\mathbf{x}_t$, c_{target}, t.
  $\boldsymbol{\mu}_t \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\text{pred}} \right)$. ▷ Mean
  $\sigma_t \leftarrow \sqrt{\beta_t}$. ▷ Standard Deviation
  $\mathbf{x}_{t-1} \leftarrow \boldsymbol{\mu}_t + \sigma_t \mathbf{z}$. ▷ Apply reverse diffusion step.
end for
$G \leftarrow \mathbf{x}_0 * \boldsymbol{\sigma}_{feat} + \boldsymbol{\mu}_{feat}$. ▷ Denormalize using training stats.
return G

we acquired our initial dataset from a CW308T-XMEGA target board, a device tailored for side-channel evaluations. This setup features an ATXMEGA128D4 8-bit microcontroller, which includes 8 KB of SRAM and 128 KB of Flash memory. We interfaced this target with the widely recognized ChipWhisperer CW308 UFO baseboard to ensure the reliable extraction of power consumption traces.

Throughout our attack evaluations, we maintained strict isolation between datasets. The generative framework was optimized solely using the profiling traces. The synthesized traces generated from this profiling subset were then used exclusively as the training data for the downstream attack network. To validate the fidelity of the synthesized data, we applied this attack network only to genuine power traces from a distinct real-trace test set that was not used during the generative-model training stage. Furthermore, to simulate practical black-box conditions, every attack group was assigned a unique, undisclosed cryptographic key. Unless otherwise stated, the independently trained evaluator used for label-consistency assessment was also trained on a disjoint subset of real traces, separate from both the profiling subset and the final attack test set.

B. Model Architecture and Training

Our model employs a conditional 1D U-Net architecture that operates on preprocessed normalized trace segments. Key components include residual blocks with 1D convolutions, ReLU activations, and multi-head self-attention mechanisms within the deeper layers of the U-Net structure to capture long-range dependencies within the normalized trace segments. The model accepts a noisy normalized trace segment x_t, the conditioning label y, and the diffusion timestep t as input. Label and time embeddings, with dimensions specified in Table II, are generated and integrated into the network to guide the denoising process conditionally.

TABLE II: Key Hyperparameters for the Conditional Diffusion Model

Parameter	Value
Optimizer	Adam
β_{start}	1×10^{-4}
β_{end}	0.02
Activation Function	ReLU
Epochs	3000
Batch Size	256
Initial Learning Rate	1×10^{-5}
Learning Rate Schedule	Cosine Decay
Diffusion Steps (T)	12000
Beta Schedule Type	Cosine
Loss Function	MSE
Target Trace Length	200
U-Net Base Channels	256
Label Embedding Dim	256
Time Embedding Dim	256
Residual Block Kernel Size	5
Conv1d/ConvTranspose1d Kernel Size	4
Conv1d/ConvTranspose1d Stride	2
Conv1d/ConvTranspose1d Padding	1
Projection Kernel Size	1
Attention Heads	4

The model was trained to predict the noise ϵ added during a $T = 12000$ step forward diffusion process, which gradually transforms the initial data x_0 into noise x_T. This forward process is governed by a cosine variance schedule, defined by β_t ranging from $\beta_{start} = 1 \times 10^{-4}$ to $\beta_{end} = 0.02$. We utilized the Mean Squared Error (MSE) between the predicted noise $\epsilon_\theta(x_t, y, t)$ and the true noise ϵ as the loss function. Optimization was performed using the Adam optimizer with a Cosine Decay learning rate schedule, starting from an initial learning rate of 1×10^{-5} and decaying over the training duration. The model was trained for 3000 epochs with a batch size of 256.

Our U-Net architecture, all Residual Blocks employ 1D convolutions with a kernel size of 5. Spatial reduction and expansion are achieved using strided convolutions (Conv1D) and transposed convolutions (ConvTranspose1D), respectively, both utilizing a kernel size of 4, stride of 2, and padding of 1. All Self-Attention blocks are configured with 4 attention heads. Finally, the input and output projections utilize point-wise convolutions with a kernel size of 1.

C. Synthetic Data Generation

Synthetic trace segments were generated using the reverse diffusion process, guided by the trained model. Here, "trace segments" refer to the preprocessed and normalized 200-point

side-channel traces used as model inputs, rather than separately extracted latent feature vectors. Starting from pure Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ with dimensions matching the target trace length (200 points, 1 channel), the process iteratively denoises the sample for $t = T, T-1, \ldots, 1$. At each step t, the model $\epsilon_\theta(x_t, y, t)$ predicts the noise component conditioned on the current noisy trace segment x_t, the desired target label y, and the timestep t. This prediction is used to estimate the less noisy trace segment x_{t-1} according to the reverse diffusion formulation. We generated a comprehensive synthetic dataset by producing 100 samples for each of the 256 possible labels, resulting in a total of $256 \times 100 = 25600$ synthetic traces. These generated trace segments were subsequently denormalized using the saved mean and standard deviation from the original training traces. The generated dataset, including labels and traces. This choice was made to provide a class-balanced synthetic dataset with sufficient per-label diversity for downstream attack training, while keeping the generation cost manageable.

D. Evaluation Metrics

To assess the quality and fidelity of the generated synthetic traces, we employed several quantitative and qualitative evaluation methods:

Qualitative Visual Inspection: We visually compared randomly selected generated traces against real traces belonging to the same class. This provides an intuitive assessment of the model's ability to capture the characteristic shapes and patterns of the original data. Representative examples are shown in Fig. 5.

Distinguishability Assessment (Real vs. Fake): We evaluated how well a standard machine learning classifier can distinguish between real and generated traces. A dataset containing an equal number of real and generated samples (randomly sampled from the available data) was created, with binary labels indicating authenticity (1 for real, 0 for generated). A Random Forest Classifier from Scikit-learn was trained on 80% of this mixed dataset and tested on the remaining 20%. The classification accuracy serves as a measure of distinguishability. An accuracy close to 50% suggests that the generated samples are highly realistic and difficult to differentiate from real ones.

Feature-Space Similarity (FID-like Score): Inspired by the Fréchet Inception Distance (FID), we calculated a simplified distance metric in a feature space derived from the distinguishability classifier. We used the class probability outputs of the trained Random Forest classifier (from the previous step) as feature representations for both real and generated samples. Let P_{real} and P_{gen} be the probability outputs for the real and generated test sets, respectively. We computed the mean vectors (μ_{real}, μ_{gen}) and covariance matrices $(\Sigma_{real}, \Sigma_{gen})$ of these probability distributions. The FID-like score was calculated as:

$$\text{FID-like} = ||\mu_{real} - \mu_{gen}||_2^2 + ||\Sigma_{real} - \Sigma_{gen}||_F$$

where $||\cdot||_F$ denotes the Frobenius norm. A lower score indicates greater similarity between the distributions of real and generated samples in this feature space.

Label Consistency / Conditional Fidelity: This metric directly assesses the model's ability to generate traces that correspond accurately to the conditioning label y. We utilized a separate, pre-trained classification model which achieves high accuracy ($> 98\%$) in predicting the correct label from real trace segments. We fed the generated traces into this high-performance classifier and obtained its label predictions. The accuracy was then calculated by comparing these predicted labels against the original labels that were used to condition the diffusion model during generation. High accuracy on this task indicates that the generated traces strongly exhibit the features associated with the intended class, demonstrating successful conditional control.

Downstream Task Performance Evaluation: To assess the practical utility of the generated data, we evaluated its effectiveness as a substitute for real data in a realistic downstream application. We trained a model exclusively on our generated synthetic traces. In parallel, we trained an identical model exclusively on real traces. Both models were then tested on their ability to classify a separate, unseen set of real-world "attack traces." We evaluated two challenging scenarios:

- **Same-Device Attack:** The test attack traces originate from the exact same physical device and acquisition setup as the original training data. This measures the synthetic data's ability to replace real data in a standard, in-distribution setting.
- **Cross-Device Attack:** The test attack traces are acquired from a physically distinct target board featuring the identical microcontroller model (i.e., a different physical instance of the ATXMEGA128D4 chip). As extensively discussed by Das *et al.* in X-DeepSCA [25], inherent manufacturing process variations between physically identical chips introduce complex domain shifts. These shifts manifest as distinct noise spectra, baseline power offsets, and subtle temporal misalignments, making cross-device evaluation highly challenging. While [25] tackled this cross-device discrepancy by training on massive datasets (e.g., 200,000 traces)—making a direct quantitative comparison with our extreme low-data regime (2,560 traces) practically incomparable—we adopt their conceptual framework to evaluate generalization. This scenario tests whether the model trained on synthetic data can bridge this physical domain gap, indicating its potential to capture underlying, transferable leakage characteristics rather than merely overfitting to the source device's unique measurement noise.

The classification accuracy of the model trained on synthetic data was compared against the baseline performance of the model trained on real data.

E. Results

To validate the effectiveness of our proposed framework, we conducted a series of extensive experiments. The presentation of our findings is organized into two main parts. First, we showcase the main results; this establishes the overall performance of our approach. Following this, we conduct in-depth ablation studies to systematically investigate the model's

TABLE III: Quantitative Evaluation Results for Generated Traces.

Method	Value
Distinguishability Accuracy (RF)	0.55
FID-like Score (RF Probabilities)	0.08
Accuracy of Label Consistency	97.7%
Same-Device Attack Accuracy(Real)	98.6%
Cross-Device Attack Accuracy(Real)	92.8%
Same-Device Attack Accuracy(Generated)	91.1%
Cross-Device Attack Accuracy(Generated)	82.4%

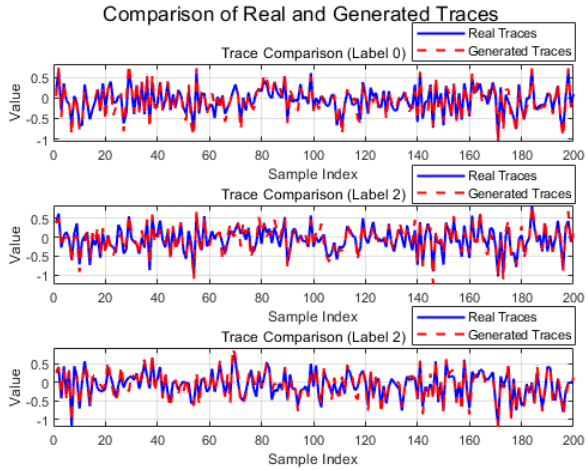


Fig. 5: Comparison of real and generated traces for selected labels (First 3 label). The generated traces exhibit similar morphological characteristics to the real data.

sensitivity to crucial hyperparameters. This analysis provides deeper insights into our design choices and their contributions to the final performance.

1) Main Results: The quantitative results of the proposed model are summarized in Table III. Visual inspection, as exemplified in Fig. 5, confirms that the generated traces qualitatively resemble the real traces, capturing typical signal shapes and variations.

The distinguishability accuracy achieved by the Random Forest classifier was 0.55, which is only slightly above the chance level of 0.50. This indicates that the generated traces are reasonably difficult to distinguish from real ones using this standard classifier. Correspondingly, the FID-like score was calculated as 0.08, suggesting a relatively small divergence between the real and generated distributions in the feature space defined by the classifier's probabilities.

This high level of utility is underpinned by the model's strong conditional control, which was assessed via the label consistency metric. To explicitly prevent any potential evaluation loop bias or overfitting to the generative feature space, we employed an independently trained evaluator network. Crucially, strict data isolation was enforced: this evaluation classifier was trained exclusively on a disjoint, held-out subset of real traces that were strictly separated from the data used to train the diffusion model. Under this rigorous and independent evaluation protocol, we achieved a stable and highly reproducible accuracy of 97.7% when classifying the generated traces according to their intended labels. This high accuracy robustly confirms that the diffusion model successfully learned the intrinsic mapping between labels and specific physical trace characteristics, rather than exploiting an evaluation loop artifact.

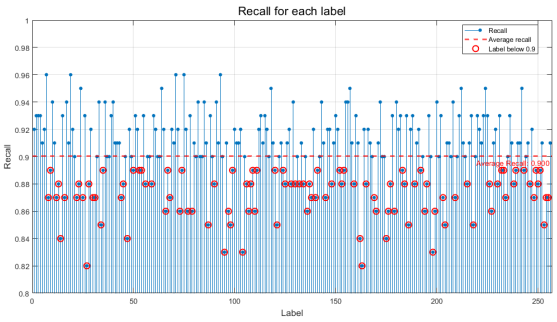


Fig. 6: Per-class recall for all 256 categories, visualized using a stem plot for clarity. The y-axis is scaled to the range [0.8, 1.0] to emphasize variations in performance among the classes. The horizontal dashed line represents the average recall, highlighting that the vast majority of classes perform well above this baseline.

To further analyze the per-class performance, we move beyond a single global accuracy number. While a confusion matrix offers a detailed breakdown of inter-class confusion, a 256×256 matrix is visually intractable. Therefore, to present a more interpretable overview, we visualize the per-class recall distribution in Fig. 6. This stem plot allows for an immediate and precise identification of underperforming classes, which appear as outliers below the predominant performance level. The vertical axis is deliberately scaled to the [0.8, 1.0] range to magnify these deviations and facilitate a more nuanced analysis of the model's strengths and weaknesses.

Crucially, the evaluation of the model's utility in practical scenarios demonstrated the effectiveness of the synthetic data. As shown in Table III, the classifier trained exclusively on generated traces achieved an attack accuracy of 91.1% in the same-device scenario, closely approaching the 98.6% accuracy of the model trained on real data. More impressively, in the challenging cross-device scenario, the model trained on generated data maintained this small performance gap, confirming that our generator captures generalizable and transferable features. This gap between the same-device and cross-device settings is expected, because traces collected from different physical boards inevitably introduce additional domain shift in noise characteristics, timing alignment, and leakage distribution. The higher same-device accuracy indicates that the generated traces align well with the source-device distribution, while the still-strong cross-device result suggests that the synthetic data also captures leakage-related features with a certain degree of transferability, although full cross-device generalization remains constrained by device-specific variation.

2) Ablation Studies: To validate our design choices and understand the model's sensitivity to key hyperparameters, we

conducted a series of ablation studies. We focused on two critical parameters for training the DDPM in the context of side-channel analysis: the training set size and the number of diffusion timesteps (T). Our goal was to find a configuration that offers the best trade-off between attack performance and data acquisition cost, as collecting a large number of traces is often impractical in real-world scenarios.

We first investigated the effects of training set size and our proposed U-Net architecture on the final attack accuracy. It is important to clarify the configuration hierarchy: while our main results utilize T=12,000 to achieve the absolute best generation fidelity, the diffusion timesteps for this specific ablation study were uniformly fixed at T=4,000. This design choice was driven by computational feasibility; training models on massive datasets (up to 25,600 traces) with T=12,000 would incur prohibitive computational costs. By evaluating both our model and the baseline fully convolutional U-Net [24] under the exact same T=4,000 protocol, we ensure a strictly fair comparison that sufficiently exposes the relative architectural advantages without excessive resource consumption.

The results, detailed in Table IV, reveal a striking improvement offered by our proposed architecture. Across all training set sizes, our model significantly outperforms the baseline fully convolutional U-Net. This superiority is especially evident in low-data scenarios: with just 6,400 traces, our model achieves a remarkable 81.8% accuracy, whereas the baseline model collapses at only 20.4%. As expected, the accuracy of both models improves with more data. Our model's performance steadily climbs, reaching a peak of 88.7% with 25,600 traces, compared to the baseline's 54.4%. However, relying on such massive datasets contradicts the practical constraints of real-world side-channel attacks. The exceptional performance of our model, even with a minimal number of traces, profoundly highlights its superior data efficiency, robustness, and enhanced feature-learning capabilities.

TABLE IV: Ablation study on the training set size and different U-Net structure. The number of diffusion timesteps was fixed at T=4,000.

Training Set Size	Fully Convolution U-Net Attack Accuracy (%) [24]	This Work U-Net Attack Accuracy (%)
6,400	20.4	81.8
12,800	44.7	87.2
25,600	54.4	88.7

Next, we explored the impact of the number of diffusion timesteps (T), which directly influences the granularity of the generation process. For this experiment, we fixed the training set size to a highly practical count of 2,560 traces and varied T from 8,000 to 14,000. The results, visualized in Fig. 7, reveal a clear trend. The attack accuracy steadily improves from 94.4% at T=8,000 to a peak of 97.7% at T=12,000. Interestingly, increasing the timesteps further to 14,000 resulted in a slight drop in accuracy to 96.8%. This suggests an inflection point where the model achieves optimal generation quality; beyond this point, additional timesteps may not yield further benefits and could introduce minor instability. Theoretically, there are two main reasons for this drop. First, too high a number of steps leads to cumulative error, where small errors in each of the thousands of generation steps accumulate and eventually degrade the quality of the final output. Second, it leads to diminishing returns as the model struggles to learn increasingly small and insignificant differences between adjacent time steps. Therefore, T=12,000 may represent an optimal balance point, providing enough granularity before the negative effects of error accumulation and inefficient learning start to dominate. This trend suggests that the diffusion process benefits from finer denoising granularity up to a certain point, because more steps allow the model to better refine subtle trace details. However, once the number of steps becomes too large, the marginal benefit of additional refinement diminishes, while error propagation across many reverse steps becomes more pronounced.

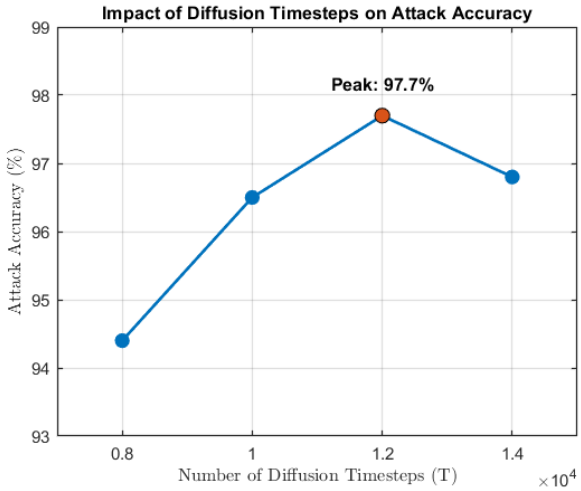


Fig. 7: Ablation study on the number of diffusion timesteps (T) with a fixed training set of 2,560 traces. The accuracy peaks at T=12,000, indicating an optimal point for the diffusion process.

Based on these findings, we selected the combination of 2,560 training traces and 12,000 diffusion timesteps as our final configuration. This choice balances exceptional attack performance with the practical constraints of real-world side-channel analysis, demonstrating our method's high efficiency in a low-data regime.

F. Comparison with State-of-the-Art Methods

To properly contextualize our results and ensure reproducibility, we first detail the evaluation protocol used to compute the Number of Traces to Guessing Entropy (NTGE).

Downstream Attack Model and Training: The profiling attack was conducted using a Convolutional Neural Network (CNN) architecture. To strictly evaluate the utility of our generative framework, this attack model was trained *exclusively* on our generated synthetic traces using a standard categorical cross-entropy loss function and the Adam optimizer.

Evaluation Protocol (NTGE Calculation): During the evaluation phase, we utilized a strictly disjoint, held-out set

of real-world attack traces. For each attack trace, the trained model outputs a probability distribution over the 256 possible intermediate leakage values (e.g., S-box outputs). These probability scores are then aggregated via maximum likelihood estimation to rank all 256 possible byte hypotheses for the secret key. The Guessing Entropy (GE) is calculated as the average rank of the correct key candidate over 100 independent attack experiments. Consequently, the NTGE is defined as the minimum number of attack traces required for the GE to converge to and stably remain at 1 (indicating the correct key is consistently ranked first).

Under this rigorous evaluation protocol, our experimental results demonstrate that the attack model, guided purely by the diffusion-generated data, achieves an exceptional NTGE of 1 when attacking the real-world trace subset. This indicates that our proposed model generates highly realistic traces and captures essential leakage-related features under the evaluated setting, enabling strong key-recovery performance from a single observation.

To highlight the significance of this achievement, we compare our performance with several state-of-the-art works that also leverage generative models for enhancing side-channel attacks. Table V provides a comparative summary of these methods, focusing on the generative model architecture, the size of the real training set required, and the final NTGE achieved.

As shown in Table V, our work demonstrates strong data efficiency and attack performance. While GAN-based methods in [16] and [17] require substantial training sets (50,000 and 4,500 traces, respectively) to reduce the NTGE to 108 and 57, our DDPM-based approach achieves an NTGE of 1 with a comparatively smaller training set of 2,560 synthetic traces. This comparison is intended as a contextual comparison of data efficiency and attack utility. The work by Karayalcin et al. [9] achieves an impressive NTGE of 1, but its framework relies on a more complex setup involving a reference device and a much larger dataset (47,500 traces) for training its feature-extracting CGAN.

Compared to another DDPM-based work by Yap et al. [19], which reported an NTGE of 5,730 on a challenging desynchronized dataset, our result of NTGE=1 on a synchronized dataset highlights the strong potential of our specific model architecture and training strategy under the evaluated setting. The ability to reach GE=0 with a single trace underscores the fidelity of our generated data and the effectiveness of the model in learning a near-perfect mapping from traces to labels. This level of performance, achieved without needing access to the original real training data for the final attack model, confirms that high-quality synthetic data generated by advanced diffusion models can serve as a powerful, and in some cases superior, substitute for real-world data in profiling attacks.

G. Discussion

The experimental results demonstrate that our proposed conditional U-Net diffusion model is capable of generating high-fidelity feature vectors conditioned on specific labels. The distinguishability accuracy between generated and real data, as evaluated by a Random Forest classifier, is close to 50%, and the model achieves a low FID-like score. This suggests that the distribution of the generated data closely mimics that of the real data, indicating a high degree of realism.

As shown in the performance comparison in Table V, our work demonstrates significant advantages in terms of efficiency and attack effectiveness. Although our classification accuracy of 97.7% is not the absolute highest among the compared methods (slightly lower than the 99.6% from [20]), our model excels in two critical aspects that underscore its superiority:

- **Top-tier Attack Performance**: In evaluating the practical utility of the generated traces, our method requires only a single trace to reduce the Guessing Entropy (GE) to 1. This is an exceptional result, matching the state-of-the-art performance achieved by [9].
- **Exceptional Data Efficiency**: Crucially, we achieved this outstanding performance using a training set of only 2,560 traces—the smallest among all compared works by a significant margin. This is substantially lower than other methods, which often require tens of thousands of traces. This clearly demonstrates our model's high learning efficiency and its immense potential in data-scarce scenarios.

Our most significant finding is that the model achieves a high accuracy of 97.7% on label consistency, while being trained on a remarkably small dataset. This result is particularly compelling because it was validated using an independent, high-performance classifier trained solely on real data. Achieving such high accuracy implies that the generated traces not only appear realistic but also precisely contain the class-specific features that the expert classifier relies on for discrimination. This provides strong evidence that the model has not merely learned to generate average-looking traces but has effectively captured the conditional dependencies defined by the labels.

The minimal performance drop observed when training an attack model with synthetic data instead of real data attests to the practical utility of our generated data. This demonstrates that the synthetic traces are not merely statistically similar, but also practically useful for this demanding application under the evaluated setting. The strong performance in the cross-device setting is particularly noteworthy. It suggests that our generative model has learned the fundamental, device-agnostic signal characteristics, enabling the trained classifier to generalize to unseen environments. This capability is crucial for creating robust models and significantly enhances the value of synthetic data, positioning it as a viable alternative to real-world data collection, especially in scenarios where data is scarce, private, or expensive to acquire.

Overall, the combination of visual similarity, low distinguishability, and particularly the high label consistency score validates our approach for generating synthetic, labeled feature vectors that are both realistic and accurately conditioned. This opens avenues for data augmentation, dataset balancing, and simulation studies where labeled time-series or feature vector data is required.

TABLE V: Performance comparisons with state-of-the-arts of Chipwhisperer Trace Generation.

Work	[16]	[17]	[9]	[19]	[20]	This work
Method	GAN	CGAN	CGAN	DDPM	DDPM	DDPM
Number of Training Set	50000	4500	47500	71168	63750	2560
Number of Traces to Guessing Entropy	108	57	1	5730	-	1
Classification Accuracy	-	91.25%	-	-	99.6%	97.7%

VII. Conclusions and Future Works

In this work, we proposed a conditional Denoising Diffusion Probabilistic Model (DDPM) utilizing a U-Net architecture to synthesize high-fidelity physical side-channel measurement signals. Our generative framework produces highly realistic temporal sensor data that accurately preserves state-specific physical leakage characteristics, as validated by a classification accuracy exceeding 97%.

The primary contribution of this study lies in the practical utility of this augmented sensor data for hardware security evaluation. A characterization model trained exclusively on our synthesized signals achieved performance nearly identical to one trained on empirical sensor data during rigorous vulnerability assessments, including challenging cross-device scenarios. This substantiates that our diffusion-based approach successfully captures fundamental, device-transferable physical leakage signatures.

Nevertheless, the current framework exhibits two primary limitations. First, its conditioning on a single operational variable is currently insufficient to model the complex, compounded physical leakages originating from simultaneous internal state interactions within the target hardware. Second, the iterative denoising process inherent to DDPMs is computationally intensive, posing performance bottlenecks for rapid or large-scale signal generation applications.

Future research will address these constraints by extending the conditioning mechanism to incorporate multivariate operational states, thereby capturing more intricate physical measurement dynamics. Additionally, we will explore advanced generation acceleration techniques, such as optimized sampling strategies, to enhance both the complexity of the synthesizable sensor data and the overall computational efficiency of the framework.

References

[1] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 251–261.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[3] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

[4] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–28.

[5] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology - EUROCRYPT 2009*, A. Joux, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 443–461.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf

[7] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, 08 2017, pp. 45–68.

[8] Y. Liu and A. Srivastava, "Ganred: Gan-based reverse engineering of dnns via cache side-channel," in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, ser. CCSW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 41–52. [Online]. Available: https://doi.org/10.1145/3411495.3421356

[9] 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 *Advances in Cryptology – ASIACRYPT 2024*, K.-M. Chung and Y. Sasaki, Eds. Singapore: Springer Nature Singapore, 2025, pp. 99–131.

[10] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 2234–2242.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 214–223. [Online]. Available: https://proceedings.mlr.press/v70/arjovsky17a.html

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[13] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *9th International Conference on Learning Representations, ICLR*, 2021.

[14] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 209–237, 11 2018.

[15] Y. Gao, H. Ma, Q. Zhang, X. Song, Y. Jin, J. He, and Y. Zhao, "Emsim+: Accelerating electromagnetic security evaluation with generative adversarial network and transfer learning," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9881–9893, 2024.

[16] H. Wang, T. Feng, and C. Liu, "Wavelet coefficients based generative adversarial networks for side-channel attack preprocessing," *Signal, Image and Video Processing*, vol. 19, 06 2025.

[17] W. Wan, W. Jun-Nian, H. Fan-Liang, and N. Feng, "Sca-cgan: A new side-channel attack method for imbalanced small samples," *Radioengineering*, vol. 32, no. 1, p. 125, 2023.

[18] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021.

[19] T. Yap and D. Jap, "Creating from noise: Trace generations using diffusion model for side-channel attack," in *Applied Cryptography and*

Network Security Workshops, M. Andreoni, Ed. Cham: Springer Nature Switzerland, 2024, pp. 102–120.

[20] S. Karayalcin, G. Perin, and S. Picek, "Diffuse some noise: Diffusion models for measurement noise removal in side-channel analysis," Cryptology ePrint Archive, Paper 2024/966, 2024. [Online]. Available: https://eprint.iacr.org/2024/966

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[25] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

Zekai Zhang received the First-Class Honors B.Eng. degree in Electrical and Electronic Engineering from University College London (UCL), in 2022. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, City University of Hong Kong. His research interests include side-channel analysis, hardware security, and deep learning.

Donglong Chen received the PhD degree from the Department of Electronic Engineering, City University of Hong Kong, in 2015. He was a visiting research scholar of COSIC, KU Leuven, Belgium, in 2013. After completing his PhD degree study, he spent four years with the industry including Huawei Technology Co., Ltd. and Tencent Technology Co., Ltd. He is currently an associate professor with the Faculty of Science and Technology, Beijing Normal-Hong Kong Baptist University, China. His research interests include cryptographic engineering, software/hardware co-design for algorithms, and privacy computing.

Wangchen Dai received the Ph.D. degree in electronic engineering from the City University of Hong Kong in 2018. After that, he had appointments at Hardware Security Lab, Huawei Technologies Company Ltd., and the Department of CSSE, Shenzhen University. He is currently working as a Senior Researcher with Zhejiang Lab, Hangzhou, China. His research interests include cryptographic hardware and embedded systems, fully homomorphic encryption, and reconfigurable computing.

Jinfa Hong received the B.S. degree in Electronic Engineering from Fuzhou University, Fuzhou, China, in 2022, and the M.S. degree in Electrical Engineering from the City University of Hong Kong, Hong Kong, in 2023. His research focuses on edge AI hardware design, with an emphasis on heterogeneous computing and operator optimization by algorithm-hardware co-design methodology.

Yu Hin Chan received his B.Eng. degree in Electronic and Communication Engineering from City University of Hong Kong in 2022, and his M.Sc. degree in Computing from Cardiff University, UK, in 2024. He is currently a Research Assistant at the Department of Electrical Engineering, City University of Hong Kong. His research interests include hardware-software co-design, secure communication protocols for RISC-V systems, and cryptography.

Çetin Kaya Koç (Fellow, IEEE) earned his B.S. in Electrical Engineering (summa cum laude) from İstanbul Technical University in 1980 and went on to complete his Ph.D. in Electrical and Computer Engineering at the University of California, Santa Barbara in 1988. His research spans a variety of areas, including cryptographic engineering, random number generation, finite field and ring arithmetic, homomorphic encryption, and machine learning. Dr. Koç holds positions in multiple organizations and leads a research group known as Koç Lab, where his work is supported by various funding institutions. The lab comprises postdoctoral researchers, Ph.D. and M.S. candidates, along with advanced undergraduate students, all collaborating on cutting-edge research in cryptography.

Patrick S. Y. Hung received his B.Sc. in Electrical Engineering from the University of Hong Kong and his M.Sc. and Ph.D. in Electrical Engineering from Stanford University. His research interests span computer architecture and computer arithmetic. He was awarded the CBI Overseas Scholarship in the United Kingdom and the Taishan Scholar Award in China. Dr. Hung previously served as a Consulting Assistant Professor at Stanford University and is currently an Adjunct Professor at the City University of Hong Kong.

Ray C. C. Cheung (Senior Member, IEEE) received the B.Eng. (Hons.) and M.Phil. degrees in computer engineering and computer science and engineering from The Chinese University of Hong Kong, Hong Kong, and the D.I.C. and Ph.D. degrees in computing from Imperial College London, London, U.K. He is a Professor with the Department of Electrical Engineering and the Department of Computer Science. He is also the Associate Provost (Digital Learning) and Director of the Institute of Future Learning at City University of Hong Kong. His current research interests include cryptographic processor designs and embedded system designs. Prof. Cheung served as the Technical Chair for FPT'02, the General Chair/Co-Chair for ARC'12, FPT'22 and ASAP'24. He is currently an Associate Editor of the Journal of Cryptographic Engineering (Springer) and the Frontiers in High Performance Computing (Frontiers), the Past Chair of the IEEE Hong Kong Section, IEEE Region 10 Student Activities Chair (North Asia), and the Chair of HK STEM Education Alliance. He has been named as Stanford's top 2% most highly cited scientist in 2024 and 2025.