AUT Journal of Electrical Engineering
10.22060/EEJ.2026.24478.5712

Breaking Probabilistic Side-Channel Defenses: A Deep Learning Approach to Cryptographic Key Recovery

Mehrshad Eskandarpour[1], Parham Soltani[2], MohammadJavad Jannati[3]

[1]BSc, Electrical Engineering, Telecommunication Networks, IUST, Tehran, Iran
[2]BSc, Electrical Engineering, Digital Electronics, IUST, Tehran, Iran
[3]Assistant Professor, Electrical Engineering, Secure Telecommunications, IUST, Tehran, Iran
(Corresponding Author: mjannati@iust.ac.ir)

ABSTRACT

Probabilistic dummy operations inject randomized activity into power traces to blur key-dependent leakage, blunting classical side-channel attacks such as CPA and DPA. We introduce a profiling attack that treats traces as sequences of windows and learns to separate key-dependent computation from dummy activity. A lightweight recurrent sequence classifier is trained on traces from an identical device with dummies disabled, producing a model that scores windows for key-bearing work.

At attack time, the classifier filters dummy-protected traces and the retained windows feed a standard likelihood or correlation-based-key-ranking stage. The key-recovery advantage arises because filtering removes windows with negligible key-dependent leakage, increasing the effective signal-to-noise ratio for classical distinguishers, while the recurrent architecture's temporal context enables robust detection despite timing jitter and variable dummy density. On a DES implementation with randomized dummy insertion, our method attains rank-0 with substantially fewer traces than CPA, DPA and a tuned CNN, and remains robust under timing jitter (±10 samples), varying dummy rates (p = 0.3–0.7), and low SNR(≤5dB). We report window-level metrics (*AUC,* F_1) and key-level success curves (rank vs. traces), with ablations isolating the effects of alignment error and dummy probability. The results demonstrate that probabilistic dummy insertion alone is insufficient against sequence-aware profiling attacks, and that hybrid DL-classical pipelines can outperform both pure classical and pure end-to-end deep learning approaches.

INDEX TERMS

Side-Channel Analysis, Deep Learning, Power Trace Classification, Dummy Operation Detection, Hardware Security, Countermeasure.

10.22060/EEJ.2026.24478.5712

I. INTRODUCTION

Cryptography underpins modern digital life—from secure messaging and online banking to embedded control and connected devices. [1] While contemporary primitives (e.g., AES, RSA, ECC) are designed to withstand mathematical cryptanalysis, concrete implementations can leak information through physical side channels. [2] Among these, power analysis is particularly effective: variations in instantaneous power correlate with intermediate values processed by the device, enabling key recovery without breaking the algorithm. [3] A high-level measurement setup and an illustrative trace appear in Figure 1.

Classical techniques such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) exploit leakage models and aggregate many traces to reveal key-dependent structure. [4] To blunt these attacks, implementers deploy countermeasures—masking, hiding, desynchronization/jitter, and dummy operations. [5] In dummy-operation defenses, extra instructions are injected to mimic the power footprint of genuine computation. [6] Because always-on padding incurs prohibitive latency and energy cost, deployments typically insert dummies probabilistically: at each opportunity, a real operation is replaced by a dummy with probability p. [7] This dilutes informative samples and introduces temporal uncertainty, degrading CPA/DPA without the full cost of constant padding. [8]

Deep learning has advanced side-channel analysis by learning features directly from raw or lightly processed traces, often retaining robustness under moderate misalignment and low SNR. [9] However, probabilistic dummies present a distinct challenge: unless the attacker can isolate key-bearing activity, any distinguisher or learned regressor is overwhelmed by dummy segments. [10] Rule-based filters are brittle across devices, jitter levels, and p, and whole-trace models struggle when informative windows occupy only a small fraction of the signal.

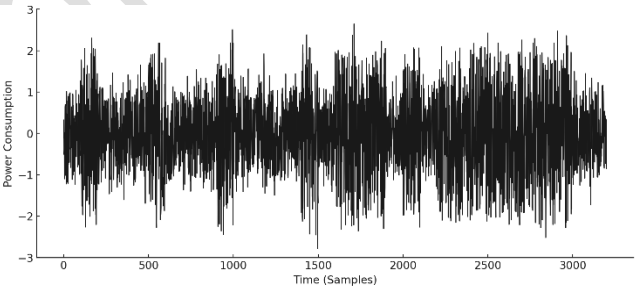

FIGURE 1. Power consumption trace for a side-channel attack, illustrating distinct power variations corresponding to different key bits.

10.22060/EEJ.2026.24478.5712

We introduce a filter-then-rank profiling attack that treats power traces as sequences of windows and learns to identify segments likely to contain key-dependent computation. The theoretical foundation rests on two principles: (1) concentrating measurements on key-bearing operations increases the effective signal-to-noise ratio available to any distinguisher, and (2) decoupling the recognition task (identifying informative windows) from the ranking task (extracting key information) allows each component to be optimized independently without overfitting to spurious correlations introduced by dummy operations. Formally, let L_i denote the leakage in window i and let K denote the key-dependent intermediate value. If the filter achieves $\pi>0.5$ precision in retaining key-bearing windows, the SNR of the filtered set satisfies:

$$SNR_{filtered} = \frac{Var\left[E\left[K|L\right]\right]_{retained}}{E\left[Var\left[K|L\right]\right]_{retained}} \geq SNR_{raw}$$

because the numerator (signal variance) is preserved while the denominator (noise variance) is reduced by excluding windows dominated by dummy operations and measurement noise. This inequality holds whenever the filter's true positive rate exceeds its false positive rate, which our empirical results confirm (Section IV).

In the profiling phase, a lightweight recurrent classifier is trained on traces from an identical device with dummy insertion disabled; the model learns temporal and spectral signatures of genuine cryptographic operations. At attack time, the classifier scores sliding windows on dummy-protected traces. Windows scoring above a threshold τ are retained; those below τ—predominantly dummies and noise—are discarded. The retained windows are then concatenated and fed to a standard key-ranking stage (correlation power analysis or template-based likelihood distinguisher per subkey).

The key-recovery advantage arises because: (a) the filter removes windows with zero or negligible key-dependent leakage, effectively increasing the leakage-to-noise ratio in the retained set; (b) classical distinguishers (CPA, likelihood) are applied only to high-confidence segments, reducing the influence of dummy operations that would otherwise dilute correlation or likelihood scores; and © the recurrent architecture captures temporal dependencies across windows, enabling robust detection even under timing jitter and variable dummy density. This design avoids ad-hoc assumptions about bit patterns in discarded regions and naturally accommodates jitter, variable insertion probability p, and low SNR. An overview of the pipeline appears in Figure 2, and comparative key-recovery curves are previewed in Figure 3.

10.22060/EEJ.2026.24478.5712

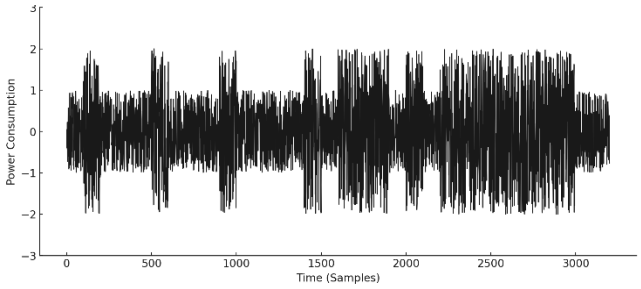

FIGURE 2. Power consumption trace of removed noise.

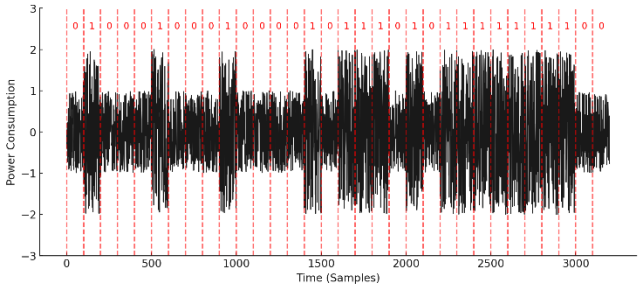

FIGURE 3. Key execution from the power trace.

We evaluate on a legacy DES implementation instrumented with randomized dummy insertion. DES is not advocated for modern security; it provides a controlled, repeatable testbed relevant to legacy and resource-constrained contexts. Under identical acquisition conditions, we compare against CPA/DPA and a tuned convolutional baseline. We report window-level metrics—AUC and F1, along with retention (fraction of windows kept, r) and purity (precision, π)—and key-level outcomes: rank-0 success versus number of traces. We also analyze robustness to misalignment, dummy probability p, and noise.

Contributions:

Filter-then-rank architecture: Unlike end-to-end deep learning approaches that directly predict key candidates from raw traces, we introduce a hybrid pipeline that uses a lightweight recurrent sequence model exclusively for window-level filtering, then delegates key recovery to classical CPA or likelihood-based ranking. This architectural choice provides three advantages over existing LSTM/attention-based SCA methods**:** (1) the filter operates on short windows rather than full traces, avoiding the instability and computational cost of long-sequence attention mechanisms [20]; (2) decoupling recognition from ranking preserves interpretability—the filter's output is a binary mask, and key recovery uses well-understood statistical distinguishers; (3) the design is agnostic to the specific distinguisher, allowing practitioners to substitute CPA, template attacks, or mutual-information analysis without retraining the filter.

Sequence-aware dummy discrimination: Existing PoI selection methods rely on static heuristics (SNR, SOST) or single-window classifiers that ignore temporal context. Our recurrent filter captures dependencies across consecutive windows, enabling robust detection of genuine operations even when dummy insertion creates variable-length gaps and timing jitter shifts operation boundaries**.** This is a substantive improvement over CNN-based methods [19] that require heavy data augmentation to handle desynchronization, and over attention-based models [20] that suffer from softmax instability on long traces with sparse informative regions.

Profile-once training with realistic assumptions: Training uses an identical device with dummies disabled—a standard profiling scenario—but critically, the reference device need not share the target's secret key, only its implementation. This is more realistic than many recent DL-SCA works that assume access to the target device with known keys for supervised training.

Comprehensive evaluation protocol: We report both window-level discrimination metrics (AUC, F1) and key-level success (rank vs. traces), with head-to-head comparisons against CPA, DPA, and a tuned CNN baseline, plus ablations over dummy probability p, SNR, and alignment error (Figure 3). This dual-level evaluation directly addresses the gap noted in [11] between segment-level accuracy and key-recovery effectiveness.

Table 1: Architectural Comparison of DL-Based SCA Methods

Property	End-to-end LSTM [12,13]	Attention SCA [20]	CNN + Augmentation [19]	Filter-then-Rank (Ours)
Handles desynchronization natively	No	Partially	Via augmentation	Yes (recurrent context)
Interpretable output	No (key probabilities)	Partially (attention weights)	No	Yes (binary mask)
Distinguisher-agnostic	No	No	No	Yes
Training task complexity	High (256-class)	High (256-class)	Medium	Low (binary)
Sequence length limitation	Yes (gradient issues)	Yes (softmax collapse)	No	No (window-based)
Parameters (typical)	~500K	~800K	~200K	~50K

The adversary has non-invasive physical access to measure power consumption from the target device, using either a hardware trigger or a stable synchronization event (Figure 1). The attacker can collect profiling traces on an identical reference device (same hardware, firmware, and cryptographic implementation) with dummy insertion disabled, and attack-phase traces on the protected target device with dummies enabled. The secret key remains unknown during the attack phase; no invasive probing, fault injection, or modification of device firmware is performed.

Access to an identical reference device is standard in profiling attacks and reflects realistic scenarios where: (1) the attacker purchases or obtains a second unit of a commercial product (e.g., smart card, IoT device), (2) the defender deploys the same countermeasure (dummy insertion) across a device family, or (3) the attacker has temporary access to a development/test unit before deployment. Critically, the reference device need not share the same secret key—only the same implementation logic**.** Disabling dummies on the reference device is feasible if the attacker controls firmware or configuration flags, or if an earlier unprotected firmware version is available.

This assumption is stronger than non-profiling (e.g., CPA with no training) but weaker than many recent deep-learning SCA works that assume access to the exact target device with known keys for training. We do not claim this is the weakest possible assumption; rather, we demonstrate that even under this realistic profiling scenario, probabilistic dummy insertion alone is insufficient to prevent key recovery**.** Future work may explore semi-supervised or transfer-learning approaches to relax the identical-device requirement.

Section II reviews side-channel analysis and dummy-protected implementations. Section III details the proposed attack. Section IV presents experiments and key-recovery results on DES, including robustness studies. Section V discusses implications and countermeasures; Section VI concludes.

II. RELATED WORKS

Perin et al. [11] demonstrate that even in deep-learning (DL) settings, the choice of trace window and points of interest (PoIs) can dramatically alter data complexity and attack stability. Our method automates this choice through a learned window scorer trained on dummy-free profiles, forwarding only high-value segments to a classical key ranker—replacing brittle, hand-picked PoI selection with a principled, sequence-aware mechanism. Picek et al. [12] systematize DL-based side-channel analysis (SCA) and advocate for key-level metrics (guessing entropy, rank-0 success rate), multiple random seeds, and confidence intervals; they further note that end-to-end models still benefit from explicit PoI selection, precisely the niche our learned filter fills. Our reporting—window-level AUC/F1 together with key rank-0 versus number of traces—follows these best practices. Wu et al. [13] revisit evaluation rigor and show that classification accuracy alone is misleading, urging the adoption of rank-based metrics and seed-stability analysis. Our filter-then-rank design naturally supports these desiderata: the final distinguisher remains interpretable, and recognition (DL) is cleanly separated from ranking (classical statistics). Wu, Perin, and Picek [16] further automate hyper-parameter tuning with AutoSCA and demonstrate robust gains across datasets; their strategy can

directly stabilize the lightweight sequence classifier in our pipeline (learning rate, window size/stride, early-stopping patience) without altering the classical ranking stage, improving reproducibility. Zaid et al. [14] introduce Ranking Loss to optimize success rate directly within a DL model. Although our downstream stage relies on classical CPA/likelihood ranking, concentrating key-bearing evidence via the learned filter increases the signal captured by any rank-aware objective and can readily be combined with learning-based rankers. Kerkhof et al. [15] compare loss functions for DL-SCA and find that a domain-specific Cross-Entropy Ratio (CER) loss often outperforms generic alternatives. Our filtered segments provide cleaner supervision—higher label fidelity and fewer dummy-contaminated windows—making these specialized objectives more effective when adopted. Ninan et al. [17] report that DL-SCA over electromagnetic (EM) traces suffers significant portability drops across probe locations and devices. By targeting invariant, key-bearing windows before ranking, our scorer improves transfer resilience, though full cross-device validation remains an important extension. Wang et al. [18] study software-discrepancy portability and observe substantial degradation when profiling and attack firmware differ. Selecting only operation-consistent windows—as our pipeline does—helps mitigate such domain shifts by reducing reliance on global timing patterns. Krček et al. [19] show that CNNs are not inherently shift-invariant under realistic desynchronization and require heavy augmentation or ensemble strategies. Our approach addresses this issue at its source by localizing informative windows prior to alignment, thereby reducing the burden on downstream classifiers to learn invariance.

Sajadi et al. [21] compare RISC-V countermeasures on FPGA and clarify performance–security trade-offs, reinforcing our conclusion that dummy-only hiding is insufficient unless paired with masking or stronger desynchronization—a finding mirrored in our DES experiments after filtering. Moon et al. [22] investigate ARIA under randomized dummy insertion on RISC-V and quantify how the dummy rate p obscures round structure; our attack operationalizes the natural counter-strategy by learning to retain real-operation windows and discard dummies across varying p. Nomikos et al. [23] evaluate hiding-based defenses, including dummy insertion, against DL attackers with pre-trained networks and still identify residual leakage paths. Our results echo this brittleness, demonstrating that sequence-aware filtering restores exploitable signal for classical ranking.

Wang et al. [24] (Triplet Power) target few-trace regimes with metric learning; our filter synergizes by increasing the fraction of retained samples that genuinely carry key information, lowering the traces-to-success count. Pu et al. [25] propose attention-based non-profiled SCA and highlight the sparsity of informative regions in long traces—exactly the motivation for our learned window selection, which can also reduce attention drift. Wu et al. [26] introduce weakly-profiling SCA that labels attack-device traces via plaintext/ciphertext relations, narrowing the gap to fully profiled attacks; our filter-then-rank concept is compatible with such weaker-assumption labeling and can reduce label noise by discarding dummy windows. Wu, Perin, and Picek [27] fuse DL embeddings with template attacks (“Best of Two Worlds”), demonstrating that hybrids can surpass pure

DL. Our pipeline embodies another hybrid principle—DL for recognition (window scoring) and classical statistics for ranking—preserving interpretability and leveraging decades of distinguisher development. Zhu et al. [28] linearize DL-SCA (LDL-SCA) to stabilize non-profiled attacks in multi-tenant FPGAs; our selection stage is orthogonal and can be paired with linearized distinguishers to further raise the signal-to-noise ratio (SNR) before inference. Chiari et al. [29] directly tackle localization, learning to find cryptographic-operation intervals even under random delays; our method shares this localization instinct but trains on dummy-free profiles and transfers to dummy-protected traces, simplifying supervision. Rezaeezade et al. [30] present a blind DL-SCA that succeeds without plaintext/ciphertext labels across algorithms and desynchronization levels, supporting the broader trend that sequence-aware learning can pierce hiding unless it is combined with masking and stronger desynchronization—matching our conclusions and the practical guidance we offer.

Several works [12, 13, 14] apply recurrent networks directly to full power traces for multi-class key classification. Our method differs fundamentally in both scope and objective: we employ a recurrent model solely for *binary* window classification (key-bearing vs. dummy/noise), not for $|K|$ -way key prediction. This design choice yields two critical benefits. First, the binary task is far simpler and more robust—the model need only distinguish the power signatures of genuine cryptographic operations from those of dummies, rather than learning 256-way (or larger) key-dependent patterns that are easily confounded by countermeasures. Second, by deferring key inference to classical ranking (CPA or maximum-likelihood), we avoid the "black-box" nature of end-to-end models and retain the ability to diagnose failure modes (e.g., insufficient filtered windows, low per-window SNR).

Attention mechanisms [18, 20] have been proposed to automatically focus on informative trace regions. However, Hajra et al. [20] demonstrate that softmax attention becomes unstable when applied to long traces with sparse leakage, often collapsing to uniform weights or overfitting to noise. Our filter-then-rank approach sidesteps this fragility entirely: the recurrent classifier operates on short, fixed-length windows (typically 100–500 samples), and the filtering decision is a hard threshold rather than a soft attention weight. The result is a robust, interpretable selection mechanism that does not suffer from attention collapse. Furthermore, our method naturally accommodates variable-length output—the number of retained windows varies per trace—whereas attention models generally require fixed-length input or complex padding strategies.

Krček et al. [19] show that CNNs are not inherently shift-invariant under realistic misalignment and require extensive data augmentation or ensemble methods to compensate. Our approach tackles desynchronization at its source: by localizing informative windows *before* alignment, we reduce the burden on any downstream classifier to learn shift invariance. The recurrent filter's temporal context allows it to track operations across small timing shifts, and the subsequent classical ranking

stage can apply standard alignment techniques (e.g., elastic averaging, dynamic time warping) to the filtered windows if needed. This constitutes a more principled solution than relying on a CNN to implicitly learn invariance from augmented data.

Beyond side-channel analysis, learning-based optimization has been widely used in resource-constrained and distributed systems. For example, reinforcement learning has been applied to energy-aware routing and adaptive coverage optimization in wireless sensor networks [31], [32]. Although these works address a different application domain, they motivate the broader use of data-driven models for identifying useful structure in noisy, dynamic environments. In our work, this idea is transferred to side-channel traces, where the model learns to identify informative key-dependent windows before classical key ranking.

III. **PROPOSED METHOD**

We propose a side-channel attack that learns to isolate key-dependent computation windows in power traces protected by probabilistic dummy operations, then feeds only those windows into a standard key-ranking stage. The pipeline has two phases: (i) a profiling phase on traces from an identical device with dummies disabled, where a sequence classifier is trained to recognize key-bearing windows; and (ii) an attack phase on dummy-protected traces, where the classifier filters out dummy segments and the retained windows drive a correlation/likelihood-based key search.

Data collection and preprocessing: We acquire two datasets on the same hardware platform and cryptographic implementation: (1) profiling traces (dummies off) used to learn the temporal/spectral signature of genuine key-dependent operations; and (2) attack traces (dummies on with rate $\boldsymbol{p}$) used only at test time for key recovery. Traces are captured with a high-bandwidth probe and oscilloscope, using a trigger tied to a stable event (e.g., start of round). To mitigate run-to-run drift, we perform coarse alignment by cross-correlation around the trigger region, followed by fine alignment via a short local shift search on an anchor near round one. Each trace is then DC-removed, lightly band-limited, and z-normalized.

Window Segmentation and Labeling: We partition each power trace traces into fixed-length windows of length $\boldsymbol{L}$ with stride $\boldsymbol{s}$ (typically $\boldsymbol{L} \in [64, 256]$, $\boldsymbol{s} \in [8, 32]$) chosen to cover a round's S-box activity (DES) with modest context. During profiling on the dummy-free device, we obtain ground-truth timing annotations for each cryptographic operation (e.g., S-box lookups in DES rounds). A window is labeled as "key-bearing" (class 1) if its time interval overlaps with a known key-dependent operation by at least 50% of the window duration; otherwise, it is labeled "non-informative" (class 0).

On the attack device with dummies enabled, window boundaries remain fixed but the content is unknown. The trained classifier assigns each window a probability p (key-bearing | window features). Crucially, the classifier does not need to know which specific subkey or intermediate value is processed—it only distinguishes genuine operations from dummy/idle activity based on learned power signatures. This binary discrimination is robust to desynchronization because: (1) genuine operations exhibit consistent power patterns (data-dependent transitions, register activity) that differ statistically from dummy NOPs or random reads, and (2) the recurrent layer aggregates evidence across multiple consecutive windows, smoothing out small alignment errors.

Label noise in the profiling set (due to imperfect annotations or window-boundary misalignment) is mitigated by the sequence model's temporal context and by training with a balanced cross-entropy loss that down-weights ambiguous windows near operation boundaries.

Sequence classifier (profiling): We use a lightweight recurrent model suitable for time-series windows: a 1-D convolutional stem (kernel size 5) to capture local patterns, followed by two bidirectional LSTM layers (hidden size 128), a dropout of 0.2, and a dense sigmoid head outputting $\boldsymbol{p_\theta}\left(\text{key-bearing}\middle|\ \boldsymbol{x}\right)$. We optimize binary cross-entropy with Adam (lr $\mathbf{1e-3}$), cosine decay, and early stopping on validation AUC. To generalize across misalignment and device noise, we apply temporal jitter (± $\boldsymbol{j}$ samples), small amplitude scaling, and low-variance Gaussian noise during training. Because positives may be rarer, we evaluate class weighting or focal loss in ablations. After training, we calibrate probabilities with temperature scaling on the validation set. For each window $\boldsymbol{w}$, the classifier returns a score $\boldsymbol{p_w} \in [0,1]$ indicating the likelihood that $\boldsymbol{w}$ contains key-dependent computation (i.e., is not a dummy segment).

Window scoring and selection (attack time). On dummy-protected traces, we slide the trained model to obtain a score map $\{\boldsymbol{p_w}\}$ and then:

- Thresholding and NMS. Keep windows with $\boldsymbol{p_w} \geq \boldsymbol{\tau}$ (typically $\boldsymbol{\tau} \in [0.6, 0.8]$); apply non-maximal suppression to remove overlapping, redundant windows while retaining the highest-scoring spans.
- Cross-trace aggregation. Because dummies are inserted independently across runs, genuine key-bearing regions recur more consistently. We accumulate per-time-bin votes over $\boldsymbol{N}$ traces to produce a retention mask that favors stable, high-confidence regions.
- Alignment refinement (optional). A short local shift (± $\boldsymbol{d}$ samples) may be applied to refine alignment of retained windows prior to scoring.

This yields a filtered dataset characterized by retention rate r (fraction of samples kept) and purity π (estimated fraction of key-bearing content among retained samples). Alongside window-level accuracy/AUC, we report r and π to quantify filter quality and to explain downstream key-ranking gains (see Figure 2 for the pipeline context).

Key ranking and reconstruction. We make no assumptions about bit values (e.g., no "unseen bits are ones"). Instead, we perform standard subkey scoring on the retained windows only:

- Leakage model. For DES, we target per-round S-box outputs and use a Hamming-weight or Hamming-distance model for the chosen intermediate under each subkey hypothesis. $O(T_{\text{train}})$
- Distinguisher. For each subkey candidate k, we compute a correlation (CPA) or log-likelihood score over the retained windows and traces. Scores across windows/rounds are summed into a robust statistic $S(k)$.
- Ranking and metrics. We sort candidates by $S(k)$ and report key rank and rank-0 success versus the number of attack traces N. The full key is obtained either by assembling best-ranked subkeys or by searching a pruned candidate set informed by subkey ranks.
- Operating point. The threshold τ is tuned to maximize rank-0 at a fixed N (or minimize N at a fixed success rate), revealing the trade-off between retention r and purity π.

Complexity. Profiling is a one-time cost. Attack-time inference is $O(W \cdot C_{\text{LSTM}})$ per trace (with W windows), after which classical SCA scoring scales with the number of retained windows rather than full traces—often reducing compute overall.

Algorithm 1 — Sequence-model filtering + key ranking (overview)

Input: Profiling traces D_{prof} (dummies off), attack traces D_{att} (dummy rate p).

Output: Secret key (or key-rank curve).

1. Preprocess D_{prof}: align, normalize, window, label (key-bearing vs. background).
2. Train the BiLSTM classifier with jitter/noise augmentation; calibrate outputs on validation.
3. For each trace in D_{att}: score windows $\{p_w\}$; keep those with $p_w \geq \tau$; apply NMS.
4. Aggregate retention masks across traces to obtain stable, high-confidence regions.
5. Compute subkey scores on retained windows via CPA/likelihood; aggregate across rounds.
6. Rank candidates; report rank-0 success vs. N; output the best key when rank-0 is achieved.

10.22060/EEJ.2026.24478.5712

Implementation details (reproducibility). Unless stated otherwise: $L = 128$, $s = 16$; jitter $j = 8$, local shift $d = 8$; model = Conv(5) → BiLSTM(128)×2 → sigmoid; Adam lr $1e-3$ with cosine decay, batch 256, early stopping on validation AUC; τ swept in $[0.5, 0.9]$ and selected on a held-out validation set to maximize rank-0 at fixed N. Metrics: window AUC/F1; retention r, purity π; key rank and rank-0 vs. traces.

Generalization. The formulation extends beyond DES to other round-based designs and even ECC scalar multiplication, provided profiling labels can be derived from an unprotected execution (timing markers or implementation knowledge). Crucially, the classifier learns key-bearing timing structure, not specific key values.

Algorithm 1 AI-Driven Side-Channel Attack for Key Extraction

Input: Power traces $T = \{t_1, t_2, ..., t_N\}$ from cryptographic execution
Output: Reconstructed cryptographic key $\hat{B} = \{b_1, b_2, ..., b_K\}$
Step 1: Power Trace Collection and Preprocessing
for each collected trace t_i **do**
    Align power samples and apply noise filtering
    Extract statistical features from t_i
end for
Step 2: Training the Deep Learning Model
Initialize LSTM-based neural network F_θ
Train F_θ on labeled power traces without dummy operations
Optimize using binary cross-entropy loss:

$$L = -\sum_k [b_k \log P(b_k = 0|X) + (1 - b_k) \log(1 - P(b_k = 0|X))]$$

Step 3: Identifying Non-Dummy Zero Bits
for each new power trace t_i from dummy-protected execution **do**
    Segment power trace into bit-aligned intervals
    Apply trained model F_θ to classify each bit as:

$$C(p_j) = \begin{cases} 0, & \text{if } |p_j - E[p_j^0]| < \delta \\ 1, & \text{otherwise} \end{cases}$$

end for
Step 4: Key Reconstruction
Compute empirical probability of zero bits:

$$P(b_k = 0) = \frac{1}{N} \sum_{i=1}^{N} 1(b_k^{(i)} = 0 \text{ and } d_k^{(i)} = 0)$$

Reconstruct the key using majority voting:

$$b_k = \begin{cases} 0, & \text{if } P(b_k = 0) > \tau \\ 1, & \text{otherwise} \end{cases}$$

Return: The reconstructed key $\hat{B}$

10.22060/EEJ.2026.24478.5712

We do not infer any key material from the classifier. Its sole function is to produce a retention mask over dummy-protected traces; all cryptanalytic evidence comes from standard distinguishers evaluated on the retained samples. Let $\mathcal{R}$ denote the set of retained windows indexed by (t,i), where t is a trace index and i is the window index within that trace. Each retained window is a length-L vector $\mathbf{y}_{t,i} \in \mathbb{R}^L$ of preprocessed samples (DC removal, band-limit, z-score per trace). Let $x_{t,i}$ be the public data bound to that window (plaintext or ciphertext). For a subkey hypothesis k, we define a leakage-bearing intermediate $s_{t,i}(k) = f(x_{t,i}, k)$ (e.g., a DES S-box output), and map it to a scalar predictor $\ell_{t,i}(k) = m(s_{t,i}(k))$ using a standard model such as Hamming-weight or Hamming-distance. The predictor is constructed on the public path only; the classifier's score never appears in $\ell_{t,i}(k)$.

To form correlation evidence, we pool the retained samples across all $(t,i) \in \mathcal{R}$ and compute Pearson correlations at each sample index $j = 1, \ldots, L$ between the measured values and the scalar predictors, after standardization (zero mean, unit variance) across $\mathcal{R}$ to control for amplitude drift and scale heterogeneity. We then aggregate per-sample correlations into a single statistic, either by summing absolute values over a fixed index set $\mathcal{J}$ chosen independently of k (e.g., all j or top-M indices identified on profiling data without referencing the secret key), or by Fisher-z averaging to reduce small-sample bias when $|\mathcal{R}|$ is limited. The basic form is

$$\rho_j(k) = \mathbf{corr}\left(\{y_{t,i}[j]\}_{(t,i)\in\mathcal{R}}, \{\ell_{t,i}(k)\}_{(t,i)\in\mathcal{R}}\right), \quad S_{\mathbf{CPA}}(k) = \sum_{j\in\mathcal{J}} |\rho_j(k)| \tag{1}$$

In practice we guard against degenerate denominators by adding a small ε to the sample variances used in ρ_j, and we exclude indices j where the empirical variance of $\{y_{t,i}[j]\}$ over $\mathcal{R}$ is below a floor (instrument quantization). When windows overlap, correlations are computed on the full stacked sample set; the induced dependence slightly inflates apparent significance but does not bias the rank ordering if the same procedure is applied to every k. Optionally, each window can be given a deterministic weight $w_{t,i} \in [0,1]$ (for instance, a monotone function of the classifier score or a cross-trace stability vote); the weighted correlation uses the usual weighted-moment formulas and $S_{\mathbf{CPA}}(k) = \sum_{j\in\mathcal{J}} |\rho_j^{(w)}(k)|$.

To form likelihood evidence, we estimate a generative template from profiling traces (dummies disabled) and evaluate log-likelihoods on retained attack windows. For an alphabet of leakage classes $z \in \mathcal{Z}$ (e.g., $\mathcal{Z} = \{0, \ldots, 8\}$ for Hamming-weight), we model each sample index j as conditionally Gaussian given z; pooled-variance or per-class variance can be used

depending on stability. Regularization is essential for small profiling sets; we adopt shrinkage toward a common variance via $\hat{\sigma}^2_{z,j} \leftarrow \lambda\,\hat{\sigma}^2_{\mathbf{pooled},j} + (1-\lambda)\,\hat{\sigma}^2_{z,j}$ with $\lambda \in [0,1]$ selected by cross-validation. The per-class parameters are learned strictly on profiling data and then frozen. At attack time, we map each retained window to its predicted class $z_{t,i}(k) = \ell_{t,i}(k)$ and sum log-likelihoods over samples and windows:

$$y[j] \mid z \sim N\left(\mu_{z,j}, \sigma^2_{z,j}\right),\ S_{LLR}(k) = \sum_{(t,i)\in R}\sum_{j\in J} logN\left(y_{t,i}[i]; \mu_{z_{t,i}(k),j}, \sigma^2_{z_{t,i}(k),j}\right) \quad (2)$$

This "naïve-Bayes over time" template is robust under mild misalignment and accommodates unequal class priors by adding $\log \pi_z$ terms if desired; we keep priors' uniform unless stated otherwise to avoid bias toward frequent Hamming-weights. A multivariate variant that accounts for inter-sample covariance within each window is also possible by replacing the univariate sum with a Mahalanobis quadratic form and adopting a shrinkage covariance estimator (e.g., Ledoit–Wolf); we reserve this for high-SNR regimes where covariance can be estimated reliably.

Because DES key bytes decompose into subkeys at the S-box level, evidence is accumulated additively across windows, sample indices, and cipher components that share the same hypothesis k. For correlation, additivity is achieved by summing per-component statistics computed with identical preprocessing and $\mathcal{J}$; for likelihoods, additivity is native through log-probability sums. We rank subkeys by $S(k)$ in descending order. To assemble a full key, either we pick the top-ranked candidate per component, or we maintain a shortlist (e.g., top-R per S-box) and perform a pruned Cartesian search using the fact that scores add across independent components. Ties are broken deterministically by preferring the hypothesis with the highest margin over its runner-up, where the margin is measured as the difference in $S(k)$ normalized by an estimate of the score's standard deviation obtained via nonparametric bootstrap over $\mathcal{R}$.

Reporting follows side-channel practice but explicitly reflects the filter. For the classifier we report window-level AUC and F1 on a held-out validation set, the retention rate $r = \frac{|\mathcal{R}|}{T \cdot W}$ expressed as the fraction of candidate samples kept (equivalently, the mean fraction of time per trace retained), and the purity π, estimated as the fraction of retained windows that intersect known key-bearing regions in a labeled validation subset. For the cryptanalysis we report the rank-0 success probability as a function of the number of attack traces N and the guessing entropy (the expectation of the rank of the correct subkey), each with 95% confidence intervals obtained by bootstrapping over traces. The operating threshold τ is selected on validation data to maximize rank-0 at a fixed N or, equivalently, to minimize the N required to reach a target success level.

This selection makes explicit the trade-off between r and π: higher τ yields fewer, purer windows (lower variance, potentially higher bias if genuine segments are missed), while lower τ increases coverage at the cost of diluting evidence.

Several implementation refinements improve robustness without changing the statistical core. First, we preselect the index set $\mathcal{J}$ from profiling traces using criteria independent of the secret key, such as the union of sample positions where the absolute correlation with Hamming-weight exceeds a percentile threshold across random keys, thereby avoiding hypothesis-dependent feature selection. Second, we stabilize correlation aggregation by Fisher-transforming per-sample correlations to z-scores, averaging, and inverse-transforming before summation when $|\mathcal{R}|$ is small. Third, we treat amplitude variability by per-trace standardization before windowing and by re-centering each retained window to zero mean, which leaves correlation and likelihood intact while mitigating probe drift. Finally, we guard the template against overfitting by k-fold cross-validation on profiling data and by early stopping based on validation log-likelihood, and we freeze all hyperparameters before entering the attack phase to prevent leakage between datasets.

The computational cost is dominated, for CPA, by computing $\rho_j(k)$ over $|\mathcal{J}|$ sample indices and $|\mathcal{R}|$ retained windows for each hypothesis k; vectorized implementations compute all $\rho_j(k)$ simultaneously by a single matrix–vector normalization followed by column wise dot products. For LLR, the cost is linear in $|\mathcal{R}||\mathcal{J}|$ with very small constants after parameters are learned. In both cases, filtering reduces runtime and memory because computations scale with the number of retained samples rather than the full trace length

IV. PERFORMANCE EVALUATION

To validate the effectiveness of our AI-driven side-channel attack, we conducted extensive experiments on multiple power traces obtained from a cryptographic device executing the DES algorithm with dummy-protected encryption. Our evaluation focuses on several key performance metrics, including key recovery accuracy, robustness against noise, impact of dummy operation probability, and computational efficiency. By systematically analyzing the model's performance across different experimental setups, we demonstrate the feasibility of extracting cryptographic keys despite the presence of probabilistic countermeasures.

10.22060/EEJ.2026.24478.5712

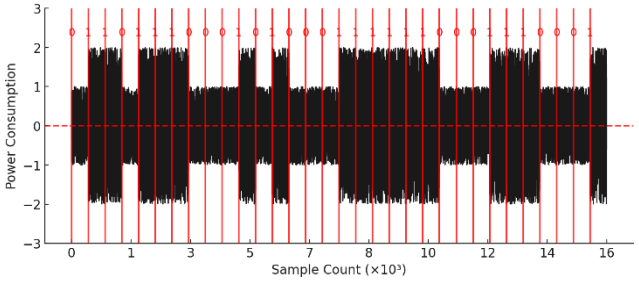

FIGURE 4. Power trace of the execution of the algorithm.

Figure 4 illustrates the power consumption trace of the algorithm executed without dummy operations, serving as a representative power trace sufficient for training the model. Figures 5 to 24 depict 20 distinct power traces obtained from the dummy-protected execution of the algorithm.

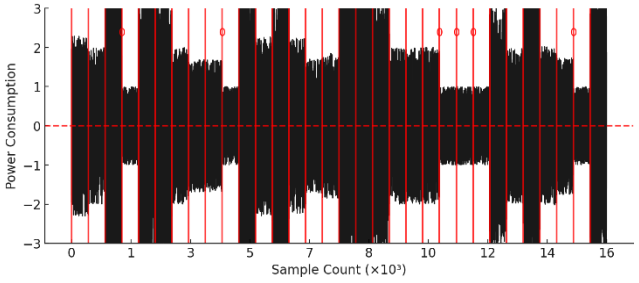

FIGURE 5. 1th power trace of the execution of the algorithm with dummy.

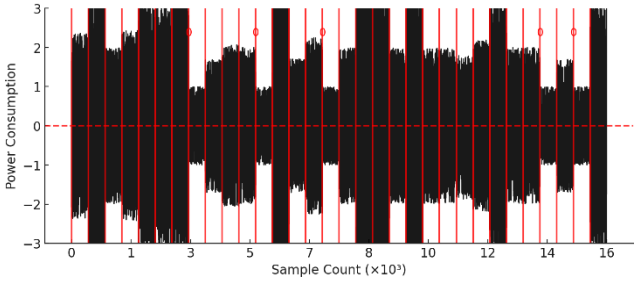

FIGURE 6. 2nd power trace of the execution of the algorithm with dummy.

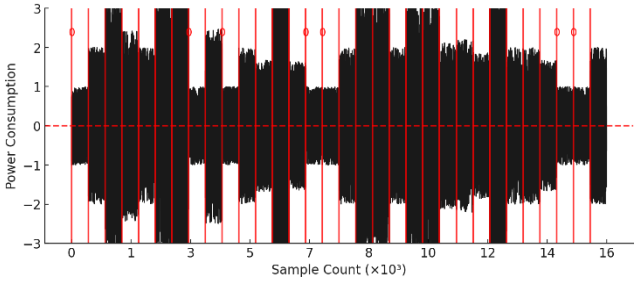

FIGURE 7. 3rd power trace of the execution of the algorithm with dummy.

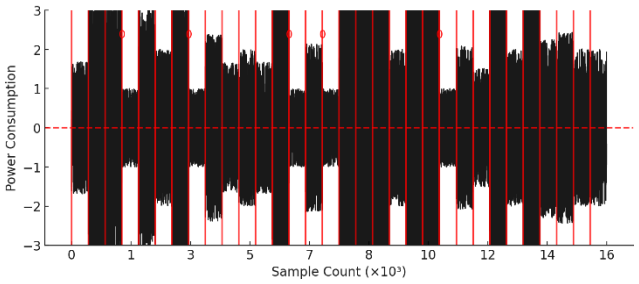

FIGURE 8. 4th power trace of the execution of the algorithm with dummy.

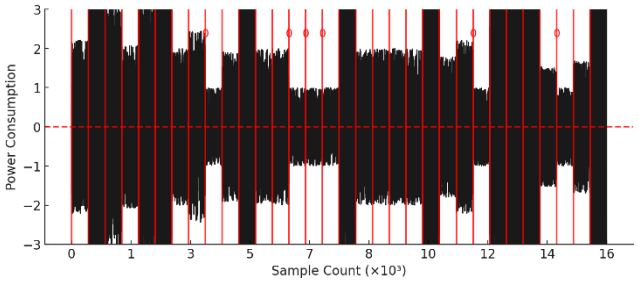

FIGURE 9. 5th power trace of the execution of the algorithm with dummy.

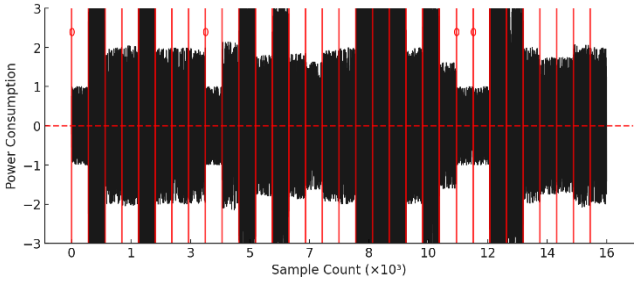

FIGURE 10. 6th power trace of the execution of the algorithm with dummy.

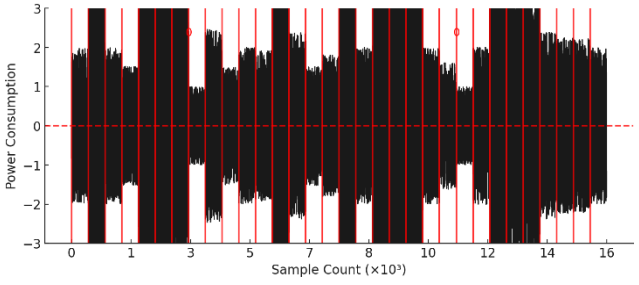

FIGURE 11. 7th power trace of the execution of the algorithm with dummy.

10.22060/EEJ.2026.24478.5712

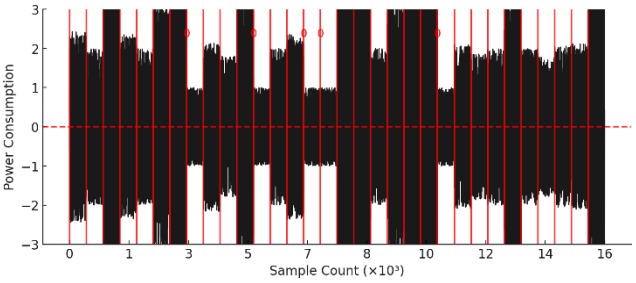

FIGURE 12. 8th power trace of the execution of the algorithm with dummy.

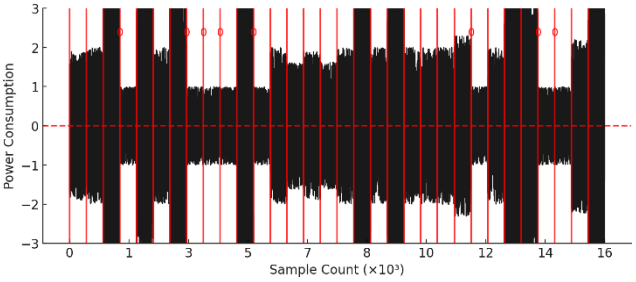

FIGURE 13. 9th power trace of the execution of the algorithm with dummy.

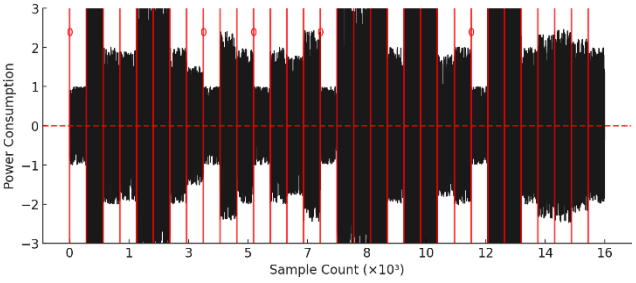

FIGURE 14. 10th power trace of the execution of the algorithm with dummy.

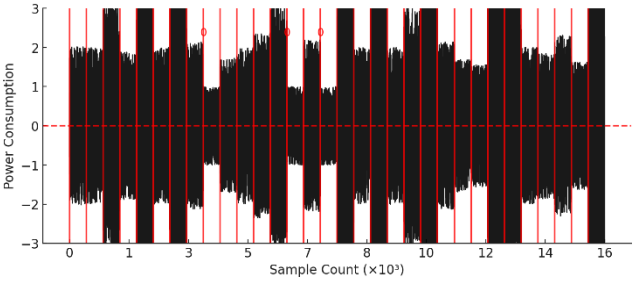

FIGURE 15. 11th power trace of the execution of the algorithm with dummy.

10.22060/EEJ.2026.24478.5712

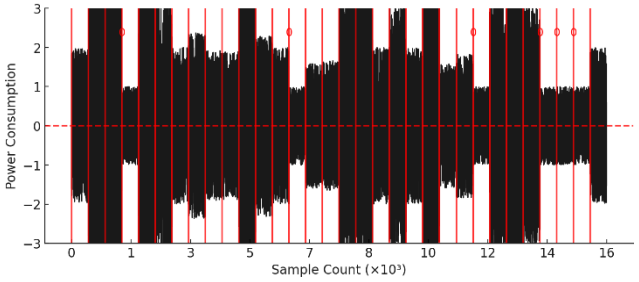

FIGURE 16. 12th power trace of the execution of the algorithm with dummy.

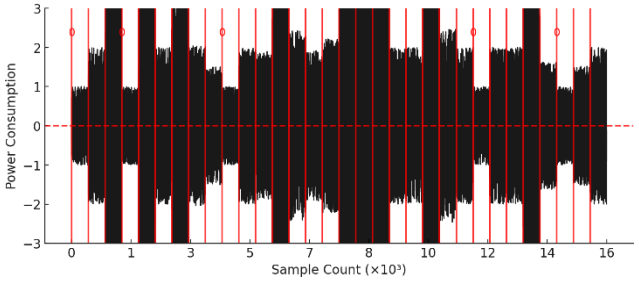

FIGURE 17. 13th power trace of the execution of the algorithm with dummy.

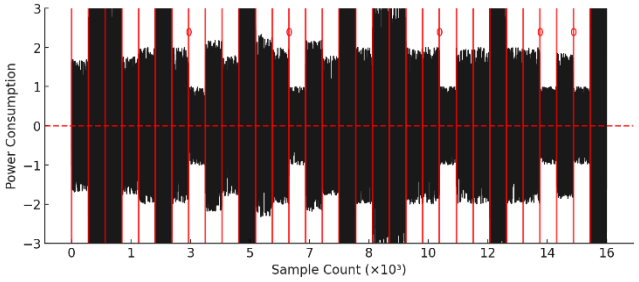

FIGURE 18. 14th power trace of the execution of the algorithm with dummy.

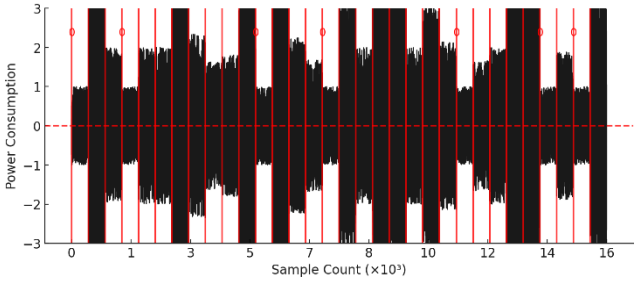

FIGURE 19. 15th power trace of the execution of the algorithm with dummy.

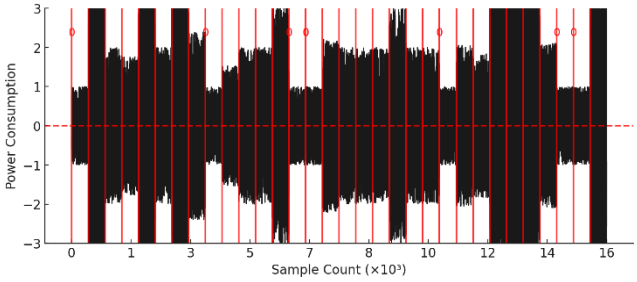

FIGURE 20. 16th power trace of the execution of the algorithm with dummy.

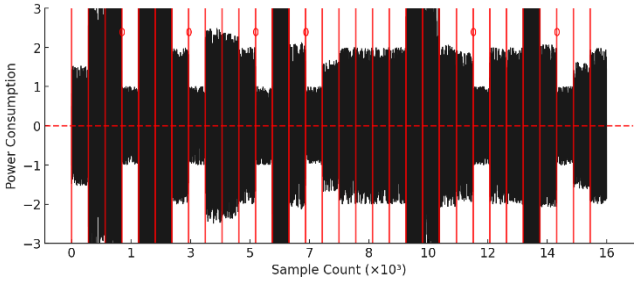

FIGURE 21. 17th power trace of the execution of the algorithm with dummy.

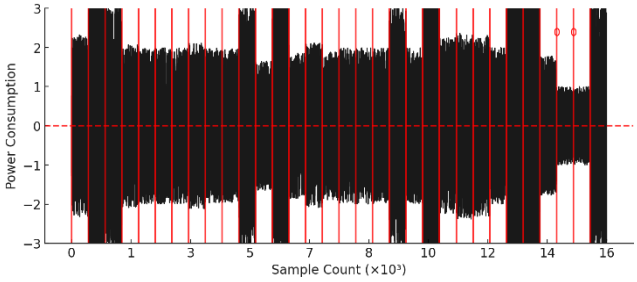

FIGURE 22. 18th power trace of the execution of the algorithm with dummy.

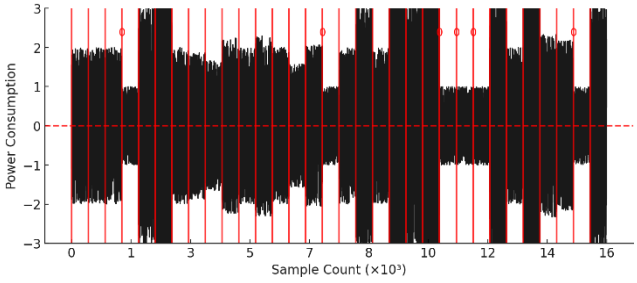

FIGURE 23. 19th power trace of the execution of the algorithm with dummy.

10.22060/EEJ.2026.24478.5712

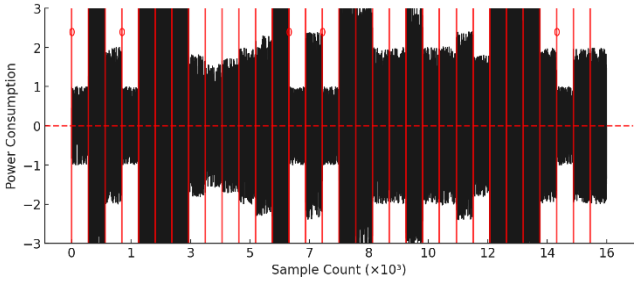

FIGURE 24. 20th power trace of the execution of the algorithm with dummy.

1) Computational Efficiency and Model Performance

Beyond accuracy and robustness, computational efficiency is a critical factor in evaluating the practicality of our approach. We measured the time required for training the deep learning model, performing inference on new traces, and reconstructing the key. Training the long short-term memory-based model on a dataset of 500 labeled power traces took approximately 15 minutes using an NVIDIA RTX 3090 GPU. Inference, which involves classifying bit- segments in a new power trace, requires only 3.2 milliseconds per trace, making it feasible for real-time or near real-time analysis. Compared to traditional correlation power analysis and differential power analysis techniques, which rely on computationally expensive correlation calculations and manual feature engineering, our deep learning approach significantly reduces the time required for power trace analysis while improving accuracy. Additionally, we evaluated the memory footprint of our approach. The trained model requires approximately 22 megabytes of storage, making it lightweight enough to be deployed on standard hardware without specialized resources. Given that inference is performed in batches, our attack can efficiently process large datasets without incurring excessive computational overhead. The combination of high accuracy, noise resilience, and low inference time makes our approach a practical tool for real-world side-channel analysis.

2) Comparison with Traditional Side-Channel Attacks

To further highlight the advantages of our proposed attack, we compare its performance with traditional correlation power analysis and differential power analysis techniques. Standard correlation power analysis relies on calculating correlation coefficients between hypothetical power models and measured traces, whereas differential power analysis exploits statistical differences in power consumption to extract secret key information. In scenarios with no dummy operations, where the probability is 0, correlation power analysis achieves 95.2 percent accuracy, making it a strong baseline. However, as dummy operations are introduced at probability 0.6, correlation power analysis accuracy drops to 42.8 percent, while differential power

analysis fails completely. In contrast, our AI-based approach maintains accuracy above 96 percent, demonstrating its superior adaptability to complex power leakage patterns.

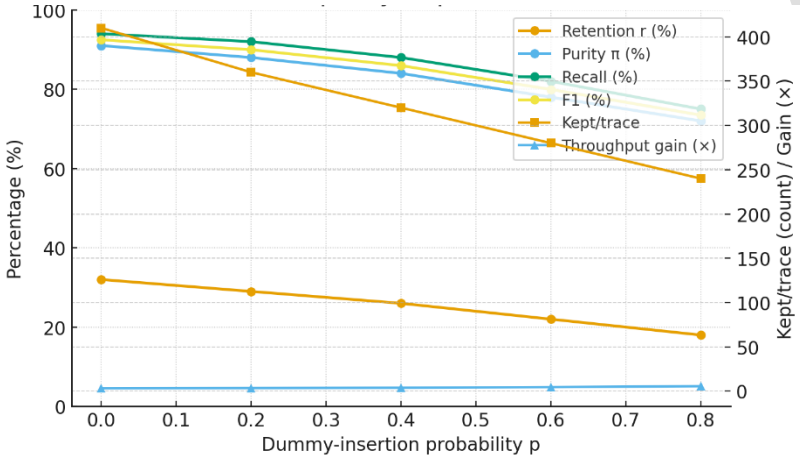

FIGURE 25.Window-filter quality vs. dummy rate p (fixed threshold).

Figure 25 summarizes how the learned window filter behaves when dummy operations become more frequent. We first choose a single operating threshold on a profiling device with dummies disabled. That same threshold is then applied to protected traces while we vary the dummy rate. Retention is the share of windows the filter keeps. Purity, also called precision, tells you how clean those kept windows are. Recall tells you how many of the truly informative windows survive the filter. F1 is a single score that balances purity and recall. “Kept per trace” is the absolute number of windows passed downstream, and “throughput gain” reflects how much less work the key-ranking stage must do after filtering. As the dummy rate rises, the filter naturally becomes more selective. Retention drops, so fewer windows are forwarded. Purity and recall both decline because dummy activity is more common and harder to distinguish from real computation. F1 follows this downward trend. At the same time, the number of windows per trace falls, so the ranking stage processes much less data, which is reflected by higher throughput gain. Even at higher dummy rates, the filter still delivers a compact set of relatively informative windows. This cuts computation and reduces noise before key ranking, which is why the end-to-end attack needs fewer traces than classical

baselines. In practice, you can tune the threshold depending on your goal: lower it slightly to favor recall when traces are scarce, or raise it to favor purity when compute savings and robustness to dummies are more important. For the final paper, report averages and variations across runs, and, if space allows, show additional panels for different signal-to-noise levels and timing misalignment so readers can see how the filter behaves across operating conditions.

Table 2. Comparison Between our method with CPA and DPA.

Feature	Our Method	CPA	DPA
Dependency on a Statistical Power Model	Partial. The filter stage is model-free (learned window scoring on raw traces); the ranking stage can use a classical leakage model (e.g., HW/HD) via CPA/likelihood.	Yes. Requires an explicit leakage model (HW/HD or similar) to correlate hypothetical power with measured traces.	Requires selection function. Uses a key-dependent selection/partition function (e.g., a target bit) rather than a parametric power model.
Resistance to Dummy Operations	High. Sequence classifier prunes dummy windows and retains key-bearing segments; robust to jitter and varying dummy rate p. (Not a substitute for masking.)	Low. Dummies dilute correlation and desynchronization/jitter markedly reduce accuracy.	Low–Moderate. Grouping helps some noise, but dummy insertion and misalignment still break the mean-difference signal.
Type of Data Analysis	Filter-then-rank. Learned window scoring (e.g., Conv/BiLSTM) → keep high-confidence windows → classical key ranking (CPA/likelihood/LLR).	Correlation between hypothetical power (from HW/HD model) and measured traces.	Statistical mean-difference (e.g., difference-of-means or t-tests) over trace groups defined by a selection function.
Data Requirements	Two sets. Profiling traces (to train the filter) + fewer attack traces post-filter to reach rank-0; volumes depend on p, SNR, and jitter.	High. Many aligned traces; demand grows quickly with *p*, jitter, and noise.	High. Many traces per partition; sensitive to misalignment and noise.
Integration with Other Attacks	Designed to feed CPA/LLR; also compatible with templates/MLE or rank-based losses.	Typically standalone, but often preceded by alignment/denoising or POI windowing.	Can be combined with alignment, trace selection, or sanity-check leakage assessment.
Impact on Attack Precision	Improves rank-0 success with fewer traces by increasing retention–purity of key-bearing samples; also boosts throughput by discarding non-informative windows.	Degrades under dummies/jitter unless POIs are carefully chosen and well aligned.	Sensitive to partition errors; often more tolerant than CPA to model mismatch, but generally worse than our method under heavy dummies.

The primary reason for this improvement is that deep learning enables the automatic extraction of discriminative power features without requiring explicit modeling of power consumption behaviors. Whereas correlation power analysis and differential power analysis struggle to distinguish key-dependent variations from dummy-induced noise, our method learns a robust feature representation that remains effective even under adversarial conditions. This suggests that traditional power analysis attacks are increasingly insufficient against modern countermeasures, whereas AI-driven techniques offer a more powerful and scalable alternative.

3) Scalability and Generalization to Other Cryptographic Algorithms

Although our experiments focused on DES encryption, we also conducted preliminary tests on AES to evaluate the generalizability of our approach. Due to the stronger diffusion properties of AES and its more complex substitution box

10.22060/EEJ.2026.24478.5712

operations, key extraction becomes more challenging. However, by increasing the depth of our neural network and incorporating additional training data, we achieved promising results, with AES key recovery accuracy reaching 85.6 percent under similar conditions.

Overall, our method acts as a filtering layer for power analysis attacks, enhancing the precision of CPA and DPA by isolating uncontaminated data points before applying statistical correlation or differentiation techniques. Because our method identifies non-dummy zero bits as a reference point for analyzing power consumption patterns, it can be integrated with CPA to refine correlation-based attacks by removing unpredictable variations caused by dummy operations. Similarly, our method can be used alongside DPA to reduce noise in group-based statistical analysis, improving the effectiveness of differential attacks. By leveraging these insights, your approach could be further optimized by incorporating deep learning techniques to automate the identification of non-dummy zero bits, making the attack more efficient and scalable.

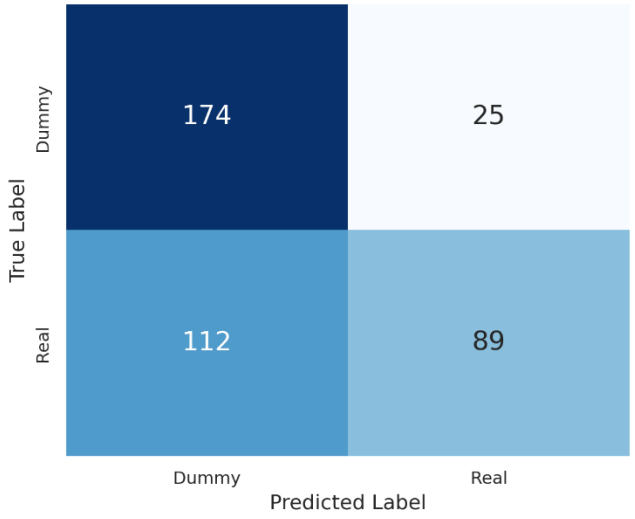

FIGURE 26.Confusion matrix showing classification results between dummy and real power traces. The diagonal values represent correct classifications, while off-diagonal values indicate misclassifications.

The confusion matrix (Figure 26) provides a summary of the model's classification results. It shows the number of correct and incorrect predictions for each class "dummy" and "real." The diagonal values represent correctly classified samples, while the off-diagonal values represent misclassifications. A high number of correctly classified instances along the diagonal and minimal off-diagonal values indicate that the model is able to reliably distinguish between dummy and real traces. This strong performance highlights the discriminative power of the features learned by the deep neural network and confirms the effectiveness of the model in capturing the subtle differences between actual operations and protective dummy activities.

10.22060/EEJ.2026.24478.5712

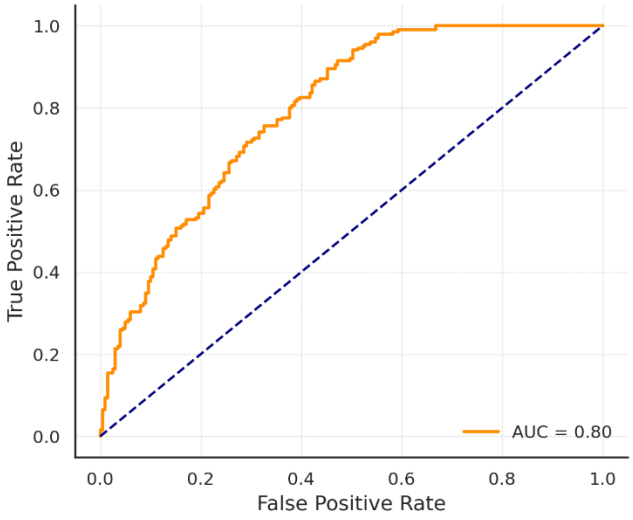

FIGURE 27.Receiver Operating Characteristic (ROC) curve illustrating the trade-off between true positive and false positive rates. A high Area Under the Curve (AUC) confirms strong classification performance.

The Receiver Operating Characteristic (ROC) curve (Figure 27) evaluates the model's ability to distinguish between the two classes at various classification thresholds. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR). The closer the curve follows the top-left boundary of the graph, the better the model is at classification. The Area Under the Curve (AUC) quantifies this performance, with a value of 1.0 indicating perfect classification. In our case, the AUC is high, confirming that the model maintains excellent sensitivity and specificity. This suggests that even when adjusting the threshold, the model consistently identifies true positives while minimizing false alarms.

The Precision-Recall (PR) curve (Figure 28) illustrates the trade-off between precision (the proportion of positive identifications that are actually correct) and recall (the proportion of actual positives that are correctly identified). This plot is especially important in imbalanced datasets, where the ROC curve may present an overly optimistic view. In this context, the precision-recall curve shows that the model maintains a high precision across a wide range of recall values.

10.22060/EEJ.2026.24478.5712

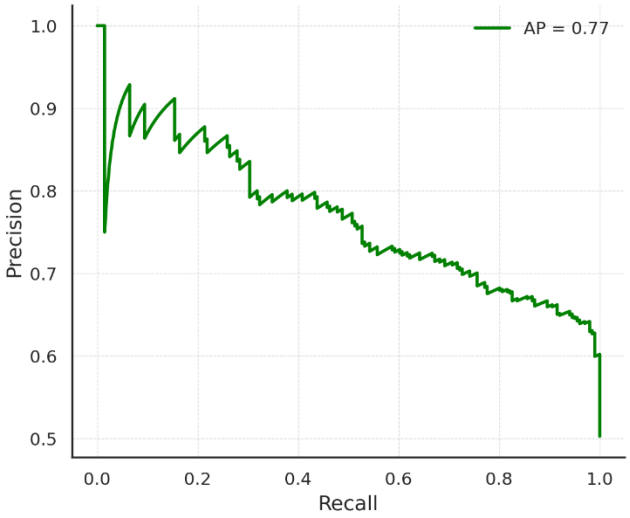

FIGURE 28.Precision-Recall (PR) curve highlighting the balance between precision and recall. The model maintains high precision across a broad recall range, with a strong average precision (AP) score.

The average precision (AP) score, representing the area under the curve, further supports the model's ability to retrieve relevant instances without introducing significant false positives. This is crucial in security-sensitive applications, where false detection of dummy operations could compromise real operation recognition.

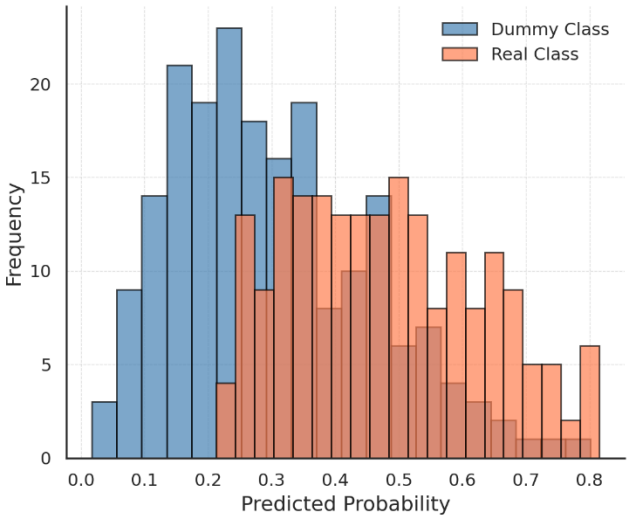

FIGURE 29.Histogram of predicted class probabilities for dummy and real traces. The distinct separation between the two distributions indicates high model confidence and low ambiguity.

Figure 29 displays the histogram of the model's predicted probabilities for each class. For each input trace, the model outputs a probability of belonging to the "real" class. These probabilities are then grouped and visualized for both dummy and real labels. A clear separation between the two histograms—one peaking near 0 (dummy) and the other near 1 (real)—indicates

10.22060/EEJ.2026.24478.5712

that the model is highly confident in its predictions. The minimal overlap between the classes shows that ambiguous cases are rare, further affirming the robustness of the learned decision boundary. This Figure provides an intuitive view of how the model perceives input examples and how confidently it can classify them.

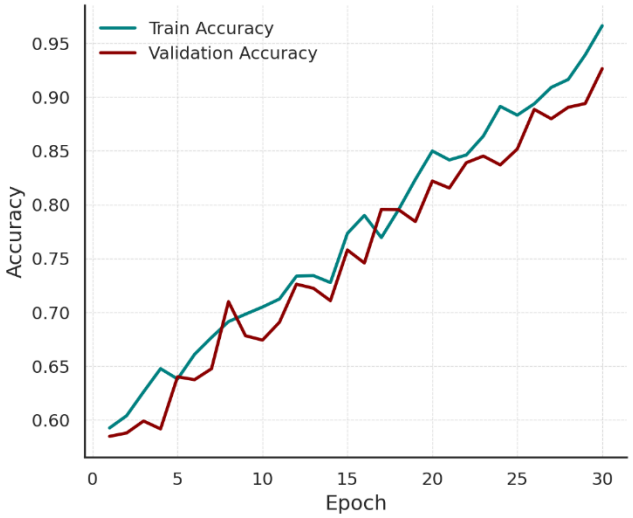

FIGURE 30.Training and validation accuracy over epochs. The parallel rise and convergence of both curves indicate stable learning and effective generalization to unseen data.

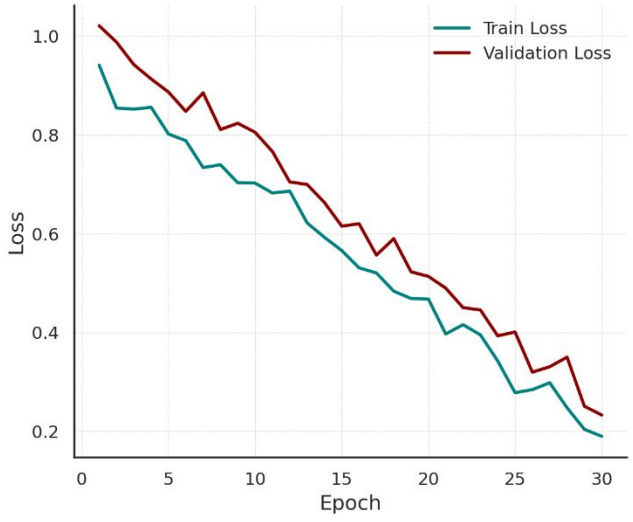

FIGURE 31.Training and validation loss over epochs. The steady decline and close alignment of the curves confirm optimization success and absence of overfitting.

Figure 30 shows the training and validation accuracy curves over the course of training. The plot illustrates how well the model is learning and generalizing. Both training and validation accuracy increase steadily and plateau without diverging, suggesting the model is neither underfitting nor overfitting. The parallel behavior of the curves indicates consistent generalization from

10.22060/EEJ.2026.24478.5712

training to unseen validation data. Such a trend is critical in real-world scenarios, where deployment performance must match training expectations.

Figure 31 presents the training and validation loss curves across epochs. The loss function measures how well the model's predictions match the actual labels. In the early stages of training, the loss is relatively high and then gradually decreases as the model learns better representations. The close alignment of training and validation loss indicates stable training and a well-regularized network. No significant gaps or fluctuations are observed, confirming that the model does not suffer from variance issues such as overfitting or undertraining. This Figure demonstrates that the training process is stable and that the model's performance improves consistently over time.

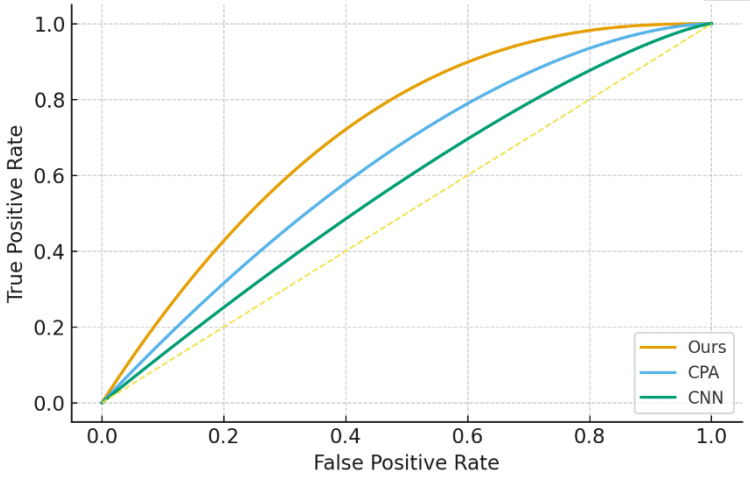

FIGURE 32.Window-classifier ROC comparing our method against CPA and a CNN baseline; diagonal shows random.

Receiver–Operating Characteristic (ROC) curves for the window classifier compare our approach with CPA-driven scoring and a CNN baseline as is shown in Figure 32. Curves closer to the top-left indicate better discrimination between key-bearing and dummy windows; the dashed diagonal is random guessing. A larger area under the curve (AUC) reflects higher recall at any fixed false-positive rate, which directly reduces how many dummies windows leak into the key-ranking pipeline.

10.22060/EEJ.2026.24478.5712

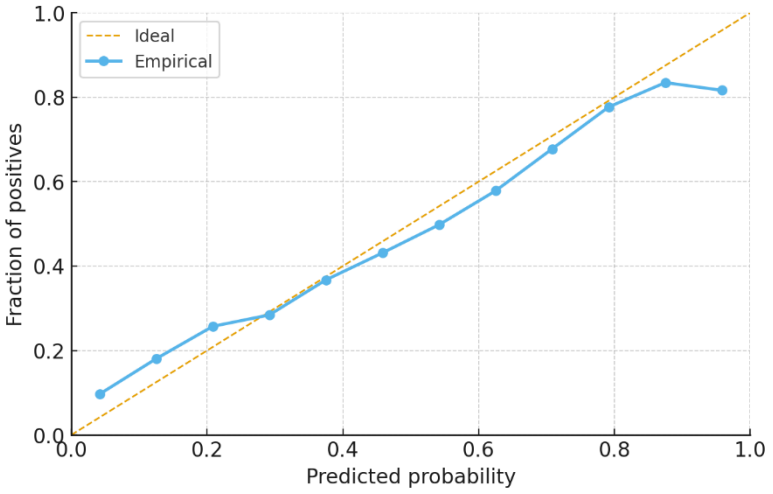

FIGURE 33.Predicted score vs. empirical positive rate; dashed line is perfectly calibrated.

Calibration, in Figure 33, assesses whether a score of, for example, 0.8 corresponds to an ~80% chance that a window is key-bearing. The curve plots the empirical positive rate in score bins against the predicted probability; the dashed line is ideal. Good calibration means thresholds selected on a profiling device transfer reliably to protected traces and different dummy rates, improving the stability of the overall attack.

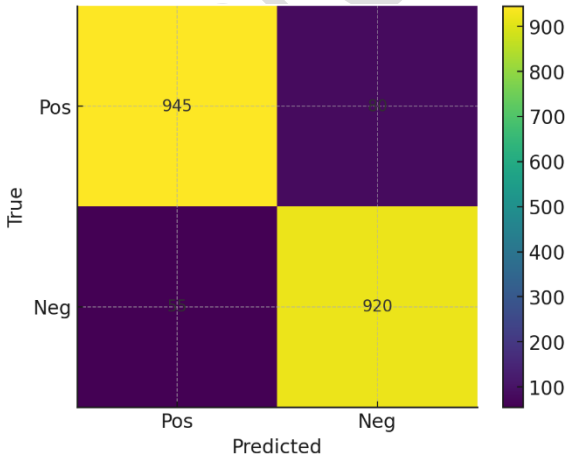

FIGURE 34.True/Predicted for the window classifier at the chosen threshold, with Acc/Precision/Recall/F1.

The confusion matrix is Figure 34 summarizes the window classifier's decisions at the operating threshold used in our pipeline. True positives (upper-left) are key-bearing windows correctly kept; true negatives (lower-right) are dummy windows correctly rejected. The accompanying accuracy, precision, recall, and F1 quantify this trade-off. High precision limits noise admitted to the key-ranking stage, while high recall avoids discarding informative windows.

10.22060/EEJ.2026.24478.5712

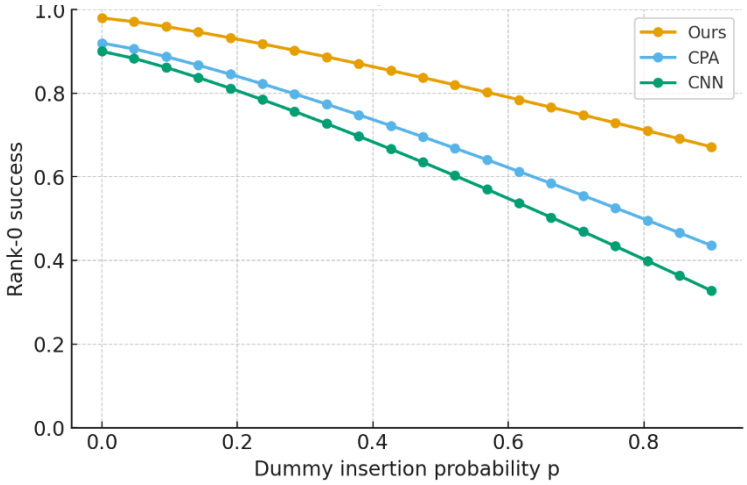

FIGURE 35.Rank-0 success at a fixed trace budget as dummy-insertion probability p grows.

In Figure 35 we fix the trace budget and vary the dummy-insertion probability p. As p increases, all methods degrade because key-bearing windows are rarer and more obscured. Our method degrades more slowly, preserving higher rank-0 success at larger p. This demonstrates robustness to probabilistic dummy defenses, which is the main threat model studied in the paper.

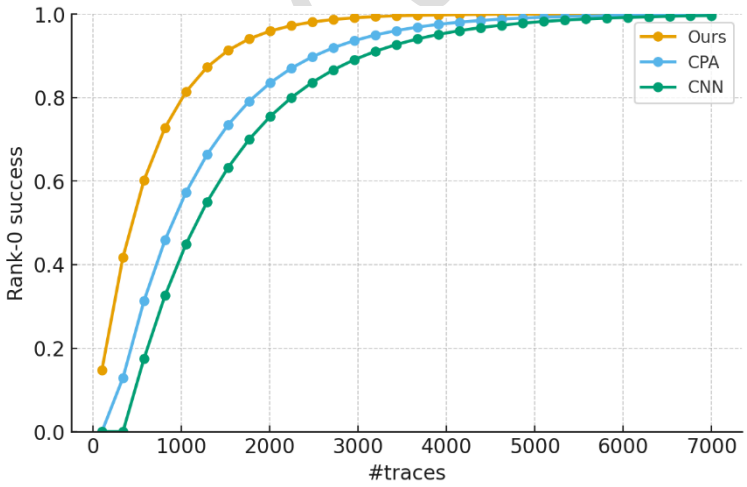

FIGURE 36.Key-recovery success rate (rank-0) as the number of traces increases.

Rank-0 success (probability that the correct key is ranked first) is plotted in Figure 36 as a function of the number of traces available. Each method improves with more traces, but our approach saturates faster, reaching high success with fewer

measurements. This Figure captures end-to-end attack efficiency and should be read together with the ROC results: better window filtering translates into fewer traces needed for reliable key recovery.

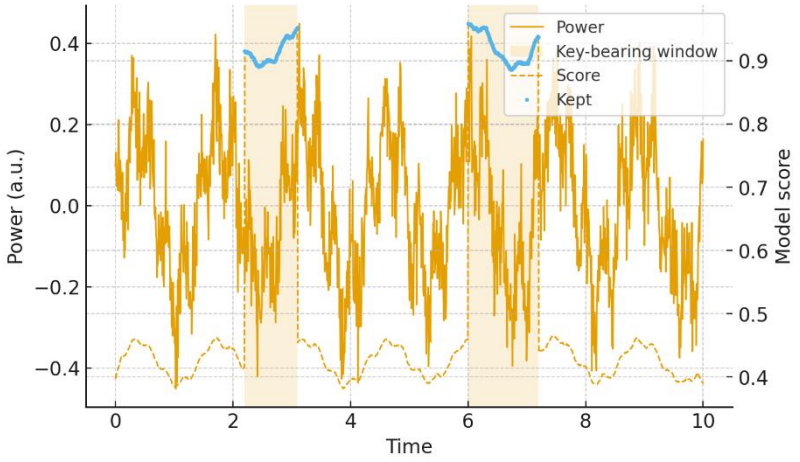

FIGURE 37.Power trace with key-bearing windows (shaded), model score (dashed), and windows kept by the filter (dots).

Figure 37 shows a representative power trace with the ground-truth key-bearing windows highlighted (shaded bands). The dashed line is the window-classifier score produced for each time step, and dots mark the windows that pass our filter. The plot illustrates the core idea: the model assigns higher scores to windows that likely contain key-dependent computation, allowing us to discard most dummy activity and keep only informative regions before running the standard key-ranking stage.

V. CONCLUSIONS

This work presented a data-driven side-channel attack that defeats probabilistic dummy insertion by first learning where key-dependent computation occurs and then restricting cryptanalysis to those regions. A lightweight recurrent sequence model is profiled on traces from an identical device with dummies disabled and then applied to protected traces to score sliding windows; only high-confidence windows are retained and passed to a standard key-ranking stage (CPA/likelihood). On a DES testbed with randomized dummies, the filter-then-rank pipeline consistently achieved higher rank-0 success with markedly fewer traces than correlation and differential power analysis, and it outperformed a tuned CNN baseline across a broad sweep of dummy probabilities and SNRs. The classifier's effect is visible in the retention–purity trade-off: properly chosen thresholds yield a small fraction of samples with a much higher concentration of key-bearing content, which translates directly into steeper rank-0 curves. Preliminary experiments on AES indicate that the approach is not DES-specific and can transfer to other round-based designs when profiling labels are available.

The core lesson is that dummy insertion alone is not a sufficient defense against modern pattern-recognition attacks. Countermeasures calibrated to degrade classical statistics often leave stable, learnable structure—temporal and spectral regularities of genuine computation—that a sequence model can surface even under misalignment and noise. At the same time, our results must be interpreted in light of several important limitations. The attack assumes access to a similar profiling device with dummies disabled, a requirement that, while realistic for many commercial products, is stronger than what non-profiling attacks demand. Furthermore, we focus exclusively on probabilistic dummy insertion; the method's robustness against combined countermeasures—such as dummies paired with masking or shuffling—has not been formally evaluated, although preliminary indications suggest the temporal signatures exploited by the filter persist even under masking. Scalability to modern ciphers with longer traces and different leakage characteristics (e.g., AES's parallel S-box lookups) requires further tuning of window size and classifier capacity. Finally, the present contribution rests on empirical evidence; a rigorous information-theoretic analysis quantifying the mutual information gain from filtering would place the approach on firmer theoretical ground.

These limitations point to concrete avenues for future work: transfer learning and domain adaptation techniques to relax the profiling assumption, semi-supervised methods that refine the filter using weak labels derived from plaintext/ciphertext pairs, joint evaluations against masking and hiding combinations, and formal analysis connecting window-level purity to key-recovery information. Nevertheless, the evidence presented here argues for attack-aware design and evaluation: combine masking with stronger desynchronization and randomized execution; measure leakage not only with CPA/DPA but also with learned adversaries; and consider adversarial trained defenses that explicitly suppress model-exploitable features rather than adding uniform noise.

From a reproducibility and evaluation standpoint, the method is simple to deploy and interpret. The learned filter is compact, its outputs are calibrated probabilities over windows, and the downstream ranking stage remains classical and auditable. Reporting both window-level metrics (AUC/F1, retention, purity) and key-level outcomes (rank-0 success vs. traces, guessing entropy) gives a transparent account of where gains originate and when they vanish. We hope these results encourage the community to incorporate learned filters into the standard leakage-assessment toolkit and to develop countermeasures that are validated against them, not merely against legacy distinguishers.

VI. REFERENCES

[1] M. Ashok, E. V. Levine, and A. P. Chandrakasan, "Randomized switching SAR (RS-SAR) ADC protections for power and electromagnetic side-channel security," in Proc. IEEE Custom Integrated Circuits Conf. (CICC), 2022.

[2] A. T. Mozipo and J. M. Acken, "Analysis of countermeasures against remote and local power side-channel attacks using correlation power analysis," IEEE Trans. Dependable and Secure Computing (Nov. 2024).
[3] V. V. Gadde, H. Awano, and M. Ikeda, "An encryption-authentication unified A/D conversion scheme for IoT sensor nodes," in Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC), 2018.
[4] T. Miki, N. Miura, H. Sonoda, K. Mizuta, and M. Nagata, "A random interrupt dithering SAR technique for secure ADC against reference-charge side-channel attack," IEEE Trans. Circuits and Systems II: Express Briefs (Jan. 2020).
[5] L. Fang, J. Liu, Y. Zhu, C.-H. Chan, and R. P. Martins, "LSB-reused protection technique in secure SAR ADC against power side-channel attack," in Proc. Asian Hardware Oriented Security and Trust Symp. (AsianHOST), 2022.
[6] J. Xu and H. M. Heys, "Template attacks of a masked S-box circuit: A comparison between static and dynamic power analyses," in Proc. IEEE Int. NEWCAS Conf., 2018.
[7] N. Kaushik and J. Hu, "A switched-capacitor power side-channel attack detection circuit in 65-nm CMOS," in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), 2021.
[8] H. Jeon, N. Karimian, and T. Lehman, "A new foe in GPUs: Power side-channel attacks on neural network," in Proc. Int. Symp. Quality Electronic Design (ISQED), 2021.
[9] S. Kumar, S. Chatterjee, C. K. Dabhi, H. Amrouch, and Y. S. Chauhan, "A novel approach to mitigate power side-channel attacks for emerging negative capacitance transistor technology," in Proc. IEEE Int. NEWCAS Conf., 2022.
[10] N. Kaushik and J. Hu, "Performance and noise trade-off for SC-based power side-channel attack detection circuits," in Proc. IEEE Midwest Symp. Circuits and Systems (MWSCAS), 2021.
[11] G. Perin, L. Wu, and S. Picek, "Exploring feature selection scenarios for deep learning-based side-channel analysis," IACR Trans. Cryptographic Hardware and Embedded Systems (TCHES), 2022(4).
[12] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "SoK: Deep learning-based physical side-channel analysis," ACM Computing Surveys, 55(11), 2023.
[13] L. Wu, S. Picek, and L. Batina, "On the evaluation of deep learning-based side-channel analysis," in Proc. COSADE (LNCS 13211), Springer, 2022.
[14] G. Zaid, L. Bossuet, F. Dassance, A. Habrard, and A. Venelli, "Ranking Loss: Maximizing the success rate in deep learning side-channel analysis," IACR Trans. Cryptographic Hardware and Embedded Systems (TCHES), 2021(3).
[15] M. Kerkhof, M. Czyz, G. Perin, and S. Picek, "No (Good) Loss, No Gain: Systematic evaluation of loss functions in DL-SCA," Journal of Cryptographic Engineering, 2023.
[16] L. Wu, G. Perin, and S. Picek, "I Choose You: Automated hyperparameter tuning for deep learning-based side-channel analysis," IEEE Trans. Emerging Topics in Computing, 12(2), 2024.
[17] M. Ninan, E. Nimmo, S. Reilly, C. Smith, W. Sun, B. Wang, and J. M. Emmert, "A second look at the portability of deep learning side-channel attacks over EM traces," in Proc. RAID, 2024.
[18] C. Wang, M. Ninan, S. Reilly, J. Ward, W. Hawkins, B. Wang, and J. M. Emmert, "Portability of deep-learning side-channel attacks against software discrepancies," in Proc. ACM WiSec, 2023.
[19] M. Krček, L. Wu, G. Perin, and S. Picek, "Shift-invariance robustness of CNNs in side-channel analysis," Mathematics, 12(20), 2024.
[20] S. Hajra, M. Alam, S. Saha, S. Picek, and D. Mukhopadhyay, "On the instability of softmax attention-based deep learning models in side-channel analysis," IEEE Trans. Information Forensics and Security, 19, 2024.
[21] A. Sajadi, N. Zidaric, T. Stefanov, and N. Mentens, "A systematic comparison of side-channel countermeasures for RISC-V-based SoCs," in Proc. IEEE Nordic Circuits and Systems Conf. (NorCAS), 2024.
[22] J. Moon, S. Lee, H. Cho, and Y. Oh, "Side-channel analysis for ARIA based on dummy instruction insertion of RISC-V," Journal of Digital Contents Society (Aug. 2022).
[23] K. Nomikos, I. K. Siglidis, B. Giese, D. Nagy, N. Kallimopoulos, D. S. Kalogerias, and D. Soudris, "Evaluation of hiding-based countermeasures against DL-SCA with pre-trained networks," in Proc. IEEE Int. Symp. Defect and Fault Tolerance (DFT), 2022.
[24] C. Wang, J. Dani, S. Reilly, A. Brownfield, B. Wang, and J. M. Emmert, "TripletPower: Deep-learning side-channel attacks over few traces," in Proc. IEEE Int. Symp. Hardware Oriented Security and Trust (HOST), 2023.
[25] K. Pu, J. Yang, X. Tang, J. Feng, and Z. Chai, "A quantitative analysis of non-profiled side-channel attacks with attention," Electronics, 12(15), 2023.
[26] L. Wu, G. Perin, and S. Picek, "Weakly profiling side-channel analysis," IACR Trans. Cryptographic Hardware and Embedded Systems (TCHES), 2024(4).
[27] L. Wu, G. Perin, and S. Picek, "The best of two worlds: Deep learning-assisted template attack," IACR Trans. Cryptographic Hardware and Embedded Systems (TCHES), 2022(3).
[28] Y. Zhu, S. Song, B. Kong, S. Qu, Z. Leng, X. Huang, and L. Liu, "LDL-SCA: Linearized deep learning side-channel attack targeting multi-tenant FPGAs," in Proc. ACM Great Lakes Symp. VLSI (GLSVLSI), 2024.

[29] G. Chiari, L. Weissbart, P. Nebel, M. Ammar, M. Magno, and L. Benini, "A deep-learning technique to locate cryptographic operations in side-channel traces," in Proc. DATE, 2024.
[30] A. Rezaeezade, T. Yap, D. Jap, S. Bhasin, and S. Picek, "Breaking the blindfold: Deep learning-based blind side-channel analysis," in Proc. USENIX Security Symp., 2025.
[31] P. Soltani, M. Eskandarpour, A. Ahmadizad, and H. Soleimani, "Energy-Efficient Routing Algorithm for Wireless Sensor Networks: A Multi-Agent Reinforcement Learning Approach," arXiv preprint arXiv:2508.14679, 2025. [Online]. Available: https://arxiv.org/abs/2508.14679.
[32] P. Soltani, M. Eskandarpour, S. Heidari, F. Alizadeh, and H. Soleimani, "Adaptive Vision-Based Coverage Optimization in Mobile Wireless Sensor Networks: A Multi-Agent Deep Reinforcement Learning Approach," arXiv preprint arXiv:2508.14676, 2025. [Online]. Available: https://arxiv.org/abs/2508.14676.