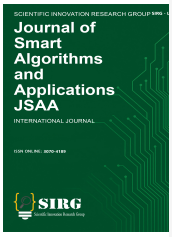

Journal of Smart Algorithms and Applications
JSAA
ISSN: 3070-4189/
Journal Homepage
https://pub.scientificirg.com/index.php/JSAA

A Graph Neural Network Framework for Structural Side-Channel Vulnerability Assessment in Cryptographic Circuits

Ola Farid[a,1] **, Azidine Guezzaz**[b]**, and Miguel Sahagun**[c]

[a] *Computer Science Department, Faculty of Science, Beni-Suef University, Beni-Suef City, 62511, Egypt. E-mail: ola3131313@gmail.com.*

[b] *Cybersecurity, IoT and Artificial Intelligence, Cadi Ayyad University, Morocco. E-mail: a.guezzaz@uca.ma.*

[c] *High Point, North Carolina State University, United States. E-mail: miguelsahagun@hotmail.com.*

ABSTRACT

Traditional side-channel analysis treats power and electromagnetic traces as temporal sequences, applying statistical or sequence-based machine learning methods without regard for the circuit topology responsible for generating observed leakage. This discards structural information intrinsic to digital circuits: gate connectivity, signal propagation topology, and the hierarchical organization of cryptographic modules. A framework is presented that applies Graph Neural Networks (GNNs) to side-channel vulnerability assessment by modeling circuits as attributed graphs in which nodes represent logic gates, edges represent wire connections, and power measurements are encoded as node features. A complete pipeline is developed spanning Verilog netlist parsing, graph construction, and Graph Convolutional Network (GCN) training with multi-head attention for multi-scale circuit analysis. Evaluation on ten AES-128 circuit implementations demonstrates an 86.4% attack success rate, compared with 68.1% for a CNN-LSTM baseline, with required power traces reduced from 988 to 790. Cross-architecture generalization reaches 63.5% accuracy on unseen circuit families, substantially above the 18.7% random baseline. Interpretable vulnerability heatmaps localize leakage sources at the gate level, enabling pre-silicon security assessment before fabrication.

PAPER INFORMATION

HISTORY

Received: 20 January 2026
Revised: 23 March 2026
Accepted: 20 April 2026
Online: 24 April 2026

MSC

68T07; 68R10; 94A60; 68M15

KEYWORDS

Graph neural networks;
Circuit topology;
Hardware security;
Side-channel analysis.

1. INTRODUCTION

The security of cryptographic hardware has become a critical concern across embedded systems, payment infrastructure, mobile devices, and cloud computing platforms. Although modern cryptographic algorithms provide mathematically rigorous security guarantees against computational adversaries, the physical realization of those algorithms in silicon introduces vulnerabilities that bypass theoretical protections entirely. Side-channel attacks exploit observable physical phenomena, including power consumption, electromagnetic radiation, and timing variation, occurring during cryptographic operations to extract secret keys from devices that remain computationally secure [1].

Classical analysis approaches treat side-channel measurements as time-series data and apply statistical methods, including Differential Power Analysis (DPA) [1] and Correlation Power Analysis (CPA) [2], as well as increasingly sophisticated machine learning models. Convolutional Neural Networks (CNNs) have demonstrated

[1]Corresponding author at Computer Science Department, Faculty of Science, Beni-Suef University, Beni-Suef City, 62511, Egypt. E-mail: ola3131313@gmail.com.

the ability to learn discriminative spatial features directly from raw traces [3], Long Short-TermMemory (LSTM) networks capture temporal dependencies [4], and Transformer architectures provide attention-based modeling of long-range sample interactions [5, 6]. Despite these advances, all sequence-based approaches share a fundamental limitation: circuit structure, the very topology that generates observed leakage, is discarded.

A power trace is not an arbitrary temporal signal but the aggregate manifestation of transistor switching events organized according to a specific circuit topology. Logic gates are interconnected by wires forming complex networks, and signal propagation follows paths determined by the design. Intermediate cryptographic values flow through this structural network, with leakage occurring at specific gates performing key-dependent operations. Sequence-based methods process traces as flat signals, losing topological information that could reveal vulnerability patterns invisible to temporal analysis.

To motivate the structural approach, consider two implementations of the AES SubBytes operation: one using lookup tables and another using Boolean logic gates. Both perform identical computations and may produce similar power trace magnitudes, yet their leakage characteristics differ fundamentally. The lookup-table implementation concentrates leakage at the memory-access gates, whereas the Boolean implementation distributes leakage across multiple XOR and AND gates with distinct connectivity patterns. Sequence-based models cannot distinguish these structural differences, potentially missing exploitable patterns or misallocating attention to uninformative samples.

Cryptographic circuits also exhibit a hierarchical structure: individual gates combine into modules such as S-boxes and MixColumns operations, which in turn compose into rounds, which together form complete algorithms. This multi-scale organization creates dependencies between distant time samples that challenge sequence models. CNNs have limited receptive fields, recurrent networks suffer from gradient degradation over long sequences, and Transformers require quadratic computational complexity with sequence length. Circuit graphs, by contrast, encode long-range dependencies through structural connectivity, enabling efficient reasoning about gate interactions regardless of physical distance.

Recent advances in GNNs have demonstrated strong performance across molecular property prediction [7, 8], social network analysis [9], program analysis [10], and hardware design tasks [11, 12, 13]. The value of exploiting topological structure in networked systems has similarly been demonstrated in wireless sensor network routing [14] and in automated security assessment for 5G networks [15], where structural and layer-aware analysis significantly outperforms flat, sequence-based detection. These successes motivate extending graph-based learning to hardware security. The broader trajectory of GNN applications to hardware security tasks, including Trojan detection, IP piracy identification, and reverse engineering, has been documented in recent surveys [16, 17], establishing graph-based learning as a credible foundation for circuit-level security analysis.

The contributions of this paper are as follows:

1. **Novel problem formulation:** Side-channel vulnerability assessment is formulated as a graph learning problem. Circuits are represented as attributed graphs $G = (V, E, X)$ where vertices V correspond to logic gates, edges E represent wire connections, and features X encode gate types, switching activity, and aligned power trace segments.

2. **Complete end-to-end pipeline:** A system is developed from Verilog netlist parsing through vulnerability prediction, incorporating graph construction, power trace segmentation and alignment, a hybrid GNN architecture combining Graph Convolutional Networks [18] with multi-head attention [19], and circuit-aware data augmentation.

3. **Superior attack performance:** Evaluation on ten AES-128 implementations demonstrates an 86.4% attack success rate versus 68.1% for the CNN-LSTM baseline, a 19.9% reduction in required traces (790 versus 988), 83.5% gate vulnerability classification accuracy, and a peak training-set F1-score of 97.8%.

4. **Cross-architecture generalization:** Transfer to circuit families excluded from training achieves 63.5% accuracy, substantially above the 18.7% random baseline, demonstrating genuine structural learning rather than memorization.

5. **Interpretable vulnerability heatmaps:** Attention weight visualization produces gate-level heatmaps that localize leakage sources, enabling pre-silicon security assessment and targeted countermeasure placement.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 provides background on power side-channel attacks and graph neural networks. Section 4 states the formal problem. Section 5 details the methodology. Section 6 presents experimental results. Section 7 discusses implications and limitations. Section 8 concludes.

2. RELATED WORK

2.1 Side-Channel Analysis

Side-channel analysis works by observing physical byproducts of computation—things like power use, timing differences, or even acoustic noise—to pull secrets out of cryptographic devices. This is different from classic cryptanalysis, which goes after mathematical weaknesses. Side-channel attacks instead exploit the fact that data-dependent operations leave measurable traces in the physical world. The field started with foundational work on timing attacks [20] and differential power analysis [1]. Since then, it has grown from simple statistical hypothesis testing to include advanced profiling and machine learning methods.

This subsection covers three main families of approaches. First are classical statistical methods, which rely on explicit leakage models. Second are shallow machine learning techniques that handle noise better than their predecessors. Third are deep learning methods that automatically pull out relevant features from raw measurements. Finally, the discussion turns to graph neural network architectures. Unlike earlier sequence-based models, these treat the circuit as a graph, preserving its actual topology instead of flattening it into a sequence.

2.1.1 Classical Statistical Approaches

Kocher et al. [1] introduced DPA, demonstrating that statistical differences between power traces grouped by key hypothesis suffice for key recovery. Brier et al. [2] advanced this with CPA, which models consumption as a function of Hamming weight to achieve higher success rates with fewer measurements. Template attacks [21, 22] extended the paradigm to profiling scenarios by constructing multivariate Gaussian models for each key-dependent class. Although effective against unprotected implementations, these methods require manual feature engineering and rely on leakage model assumptions that may not hold for complex protected designs.

2.1.2 Machine Learning Approaches

Heuser and Zohner [23] applied Support Vector Machines for side-channel analysis, demonstrating improved robustness to noise compared with template attacks. Lerman et al. [24] showed that random forests achieve competitive performance against masked implementations, though still requiring manual feature extraction.

2.1.3 Deep Learning Approaches

Maghrebi et al. [3] demonstrated that CNNs can learn features directly from raw power traces. Picek et al. [25] provided a systematic evaluation confirming the superiority of deep learning over classical methods. Kim et al. [4] showed that CNN-LSTM hybrids outperform pure architectures. Zaid et al. [6] introduced architectural optimization guidelines for CNNs in profiling attacks. Wouters et al. [5] extended these findings through ablation studies across multiple datasets.

Transformer-based architectures have more recently been applied to this problem. Perin et al. [26] proposed multi-head self-attention models that outperform CNNs on desynchronized traces. A Hajra et al. [27] introduced EstraNet, a shift-invariant transformer that addresses the quadratic complexity of prior attention-based models and scales to long power traces under combined masking and desynchronization countermeasures. Bache et al. [28] developed a confidence-interval-based side-channel evaluation framework combining efficient data acquisition with statistical leakage assessment. Bronchain and Cassiers [29] demonstrated efficient arithmetic-to-Boolean masking conversion techniques with application to lattice-based cryptography, showing that implementation-level structural Choices directly determine side-channel exposure. Picek et al. [30] provided a comprehensive survey of deep learning approaches to physical side-channel analysis, and Wu et al. [31] examined the generalization ability of profiling attacks across diverse implementations.

Despite these advances, all prior deep learning approaches treat traces as flat sequences, ignoring the circuit structure that generates leakage.

2.2 Graph Neural Networks

Kipf and Welling [18] introduced Graph Convolutional Networks (GCNs), enabling efficient neighborhood aggregation through renormalized adjacency matrices. Veličković et al. [19] added attention mechanisms through

Graph Attention Networks (GATs), learning to weight neighbor contributions dynamically. Gilmer et al. [8] unified various architectures under the Message Passing Neural Network (MPNN) framework. Hamilton et al. [9] proposed inductive representation learning through sampled neighborhood aggregation, enabling generalization to unseen nodes. Xu et al. [32] analyzed the expressive power of GNNs in relation to the Weisfeiler-Lehman graph isomorphism test, showing that Graph Isomorphism Networks achieve maximum discriminative power among message-passing architectures. Rampášek et al. [33] combined GNNs with Transformer architectures, achieving state-of-the-art performance on molecular benchmarks.

In hardware design, Li et al. [11] introduced DeepGate, a GNN that learns gate-level representations by jointly encoding circuit topology and logic function. Shi et al. [12] extended this to DeepGate2 with functionality-aware learning. Dong et al. [13] introduced CktGNN, a two-level circuit GNN that jointly encodes circuit topology and device sizing. El Sayed et al. [17] provided a comprehensive survey of GNN applications to integrated circuit design, reliability, and security.

2.3 Hardware Security Analysis

Circuit-level security analysis has traditionally employed static methods: gate-level simulation, formal verification, and information flow tracking [34]. These approaches can identify certain vulnerability classes but struggle with emergent side-channel properties arising from complex gate interactions. Data-driven approaches have begun to appear. Moos et al. [35] proposed a deep-learning-based leakage assessment methodology that eliminates the need for manual trace alignment and covers multivariate leakage patterns. Cassiers and Standaert [36] examined globally optimized masking schemes, showing that structural implementation choices have direct implications for side-channel resistance. The security demands of 5G communication infrastructure [15] and wireless sensor networks [14] have independently motivated automated, topology-aware analysis frameworks, reinforcing the principles pursued in this work.

The present work differs from all prior approaches by representing circuits as graphs and learning vulnerability patterns through message-passing operations that respect circuit topology.

2.4 Comparative Summary

Table 1 summarizes key related approaches. The performance figures cited for each method are drawn from their respective source publications under different experimental conditions, and are not directly comparable.

3. BACKGROUND

3.1 Power Side-Channel Attacks

CMOS logic gates consume power primarily during state transitions when internal capacitances charge or discharge. The instantaneous power consumption at the gate i at time t is proportional to the product of capacitance and the Hamming distance between successive gate states [37]:

$$P_i(t) = \alpha\, C_i\, V_{\text{dd}}^2 \cdot \text{HD}(s_i(t-1),\, s_i(t)) + P_{\text{static}} + \eta(t), \tag{1}$$

where $\alpha \in [0, 1]$ is the activity factor, C_i is gate i's load capacitance, V_{dd} is the supply voltage, $s_i(t) \in \{0, 1\}^n$ is the gate state, $\text{HD}(\cdot, \cdot)$ is the Hamming distance, P_{static} is leakage current, and $\eta(t)$ is measurement noise [37]. The total measured power aggregates over all switching gates [38]:

$$P(t) = \sum_{i \in \mathcal{G}} P_i(t). \tag{2}$$

For cryptographic operations, intermediate values $s_i(t)$ depend on both plaintext p and secret key k, creating an exploitable statistical dependency.

In a CPA attack [2], the analyst collects N power traces $\{T_j\}_{j=1}^N$ for known plaintexts $\{p_j\}_{j=1}^N$. For each key byte hypothesis $\hat{k} \in \{0, \ldots, 255\}$, the hypothetical intermediate value is:

Table 1: Conceptual comparison of related approaches to side-channel vulnerability assessment. Performance figures are from original publications and are not directly comparable across rows.

Method	Category	Circuit Structure?	Strengths	Limitations
DPA/CPA [1, 2]	Statistical	No	Simple; no training data required	Many traces needed; manual feature selection
Template attacks [21, 22]	Probabilistic	No	Bayesian-optimal; handles noise well	Requires profiling device
SVM-based [23]	Shallow ML	No	Better noise robustness than templates	Manual feature extraction
Random forest [24]	Shallow ML	No	Handles high-dimensional features	No structural modeling
CNN [3, 6]	Deep learning	No	Automatic feature extraction	Ignores temporal/structural context
LSTM/RNN [4]	Deep learning	No	Models temporal dependencies	Vanishing gradients; no circuit structure
CNN-LSTM [4]	Deep learning	No	Combines spatial and temporal features	No circuit structure
Transformer [26, 27]	Deep learning	No	Long-range attention; handles desync	No circuit structure
GNN for circuits [12, 13]	Graph learning	Yes	Respects circuit topology	Limited prior application to side-channel
WSN topology-aware [14]	Graph/routing	Yes	Network structure informs routing/security	Domain is sensor networks
5G automated detection [15]	Automated ML	No	Multi-layer automated threat detection	No graph-level structural reasoning

$$\hat{v}_{j,\hat{k}} = \mathrm{HW}\Big(\mathrm{SBOX}(p_j \oplus \hat{k})\Big), \tag{3}$$

where $\mathrm{HW}(\cdot)$ denotes Hamming weight. The Pearson correlation between hypothetical values and measured power at time t is:

$$\rho_{\hat{k}}(t) = \frac{\sum_{j=1}^{N}\Big(\hat{v}_{j,\hat{k}} - \bar{v}_{\hat{k}}\Big)\big(T_j(t) - \bar{T}(t)\big)}{\sqrt{\sum_{j=1}^{N}\Big(\hat{v}_{j,\hat{k}} - \bar{v}_{\hat{k}}\Big)^2 \cdot \sum_{j=1}^{N}\big(T_j(t) - \bar{T}(t)\big)^2}}. \tag{4}$$

The key estimate is recovered as:

$$k^* = \arg\max_{\hat{k}} \max_{t} \big|\rho_{\hat{k}}(t)\big|. \tag{5}$$

In the profiling attack model [21], a classifier $f_\theta : \mathbb{R}^T \to \{0, \ldots, 255\}^b$ is trained on traces acquired with known keys. The attack success rate (ASR) at N traces is:

$$\mathrm{ASR}(N) = \Pr\left[\arg\max_{k} f_\theta\Big(T_1^N\Big)\,[k] = k^*\right], \tag{6}$$

where T_1^N denotes the aggregated evidence from N traces.

3.2 Circuit Representation as Graphs

A gate-level netlist specifies components (primitive gates, flip-flops, and input/output ports) and interconnections (wire nets). A circuit is formally a directed graph $G = (V, E)$ where V is the set of component instances and $E \subseteq V \times V$ describes wire connections. An edge $(u, v) \in E$ indicates that gate u drives an input of gate v. Each gate node $v \in V$ is associated with a feature vector $\mathbf{x}_v \in \mathbb{R}^{d_x}$ encoding:

$$\mathbf{x}_v = \left[\mathbf{x}_v^{\text{type}} \parallel \mathbf{x}_v^{\text{struct}} \parallel \mathbf{x}_v^{\text{activity}} \parallel \mathbf{x}_v^{\text{power}}\right], \tag{7}$$

where $\mathbf{x}_v^{\text{type}} \in \{0,1\}^8$ is a one-hot encoding of gate type (AND, OR, XOR, NAND, NOR, XNOR, NOT, flip-flop), i.e., Features 1–8; $\mathbf{x}_v^{\text{struct}} \in \mathbb{R}^4$ encodes fan-in, fan-out, depth from primary inputs, and depth to primary outputs (Features 9–12); $\mathbf{x}_v^{\text{activity}} \in \mathbb{R}^4$ encodes the toggle rate and transition density from the simulation (Features 13–16); and $\mathbf{x}_v^{\text{power}} \in \mathbb{R}^{16}$ is an aligned power trace segment corresponding to the gate operation timing window (Features 17–32). The full feature vector has dimension $d_x = 32$.

3.3 Graph Neural Networks

GNNs learn node representations through iterative message passing [8]. At layer k, each node v aggregates information from its neighbors $\mathcal{N}(v)$:

$$\mathbf{m}_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right), \tag{8}$$

$$\mathbf{h}_v^{(k)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathbf{m}_v^{(k)}\right). \tag{9}$$

Initial representations are $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. After K layers, $\mathbf{h}_v^{(K)}$ encodes information from the K-hop structural neighborhood of v.

Kipf and Welling [18] proposed symmetric normalized aggregation:

$$\mathbf{H}^{(k+1)} = \sigma\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(k)}\mathbf{W}^{(k)}\right), \tag{10}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the corresponding degree matrix, $\mathbf{W}^{(k)}$ are learnable weight matrices, and σ is a nonlinear activation. Veličković et al. [19] introduced attention coefficients. The unnormalized attention score between nodes u and v is:

$$e_{uv} = \text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_u \parallel \mathbf{W}\mathbf{h}_v]\right). \tag{11}$$

Normalized across the neighborhood:

$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{w \in \mathcal{N}(v)} \exp(e_{wv})}. \tag{12}$$

The updated node representation is:

$$\mathbf{h}'_v = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}\, \mathbf{W}\mathbf{h}_u\right). \tag{13}$$

Multi-head attention applies M independent attention mechanisms:

$$\mathbf{h}_v^{\mathrm{MH}} = \Big\|_{m=1}^{M} \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(m)} \mathbf{W}^{(m)} \mathbf{h}_u \right). \tag{14}$$

4. PROBLEM FORMULATION

4.1 Threat Model

The adversary possesses (i) a gate-level netlist of the target circuit, obtained through the Electronic Design Automation supply chain or reverse engineering; (ii) physical access to a profiling device identical to the target chip, enabling power measurements during known-plaintext encryptions; and (iii) a power probe or oscilloscope for signal acquisition. Consistent with the standard profiling attack model [21], the adversary does not require knowledge of the secret key during the profiling phase. The objective of the attack is to recover the secret key from the cryptographic device.

4.2 Formal Problem Statement

A cryptographic hardware circuit is represented as an attributed directed graph:

$$G = (V, E, X, A), \tag{15}$$

where $V = \{v_1, \ldots, v_n\}$ is the set of n logic gate nodes, $E \subseteq V \times V$ is the set of directed wire connections, $X \in \mathbb{R}^{n \times d_x}$ is the node feature matrix, and $A \in \{0,1\}^{n \times n}$ is the adjacency matrix with $A_{uv} = 1$ whenever $(u, v) \in E$.

4.3 Node-Level Vulnerability Classification

The gate vulnerability classification task assigns a binary label to each node:

$$f_\theta : (G, A, X) \to \hat{Y} \in \{0,1\}^n, \tag{16}$$

where $\hat{Y}_v = 1$ indicates that gate v exhibits statistically significant correlation with key-dependent intermediate values. The ground-truth label for gate v is derived from the correlation-based leakage criterion [2, 38]:

$$y_v = \mathbb{1}\left[\max_t |\rho_v(t)| > \tau \right], \tag{17}$$

where $\rho_v(t)$ is the Pearson correlation between gate v's switching activity and hypothetical key-dependent values at time t, and τ is a threshold. The sensitivity of results to the choice of τ is examined in Section 6.5.

4.4 Graph-Level Security Assessment

At the circuit level, the aggregate vulnerability score is:

$$\mathrm{VulnScore}(G) = \frac{1}{|V|} \sum_{v \in V} \hat{Y}_v. \tag{18}$$

4.5 Attack Guidance Through Vulnerability Ranking

The GNN's node-level predictions guide trace analysis by ranking gates by vulnerability score. Let $\pi : V \to \{1, \ldots, |V|\}$ be the ranking induced by $\hat{Y}$. The attack focuses on the top-K most vulnerable gates:

$$\mathcal{S}_K = \{ v \in V : \pi(v) \leq K \}. \tag{19}$$

Power trace segments corresponding to $\mathcal{S}_K$ are selected for correlation analysis, reducing the effective trace dimension from T total samples to $|\mathcal{S}_K| \cdot w$ selected windows of width w, improving the signal-to-noise ratio.

4.6 Training Objective

The GNN parameters θ are optimized through binary cross-entropy loss [4, 3]:

$$\mathcal{L}(\theta) = -\frac{1}{|V_{\text{train}}|} \sum_{v \in V_{\text{train}}} \left[y_v \log \hat{y}_v + (1 - y_v) \log(1 - \hat{y}_v) \right], \tag{20}$$

with ℓ_2 regularization $\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \lambda \|\theta\|_2^2$, where $\lambda = 10^{-4}$.

5. METHODOLOGY

5.1 System Overview

The proposed framework consists of four sequential stages.

1. **Netlist Parsing:** Verilog hardware description language files are converted to attributed graph representations.
2. **Power Trace Alignment.** Power trace samples are segmented and aligned with individual gate operation windows.
3. **GNN Training:** The graph neural network is trained on labeled circuits to predict gate-level vulnerability.
4. **Attack Guidance:** Predicted vulnerability scores are used to focus side-channel analysis on high-leakage circuit regions.

5.2 Netlist Parsing and Graph Construction

The first stage converts a Verilog design into an attributed graph suitable for GNN processing. PyVerilog is used to parse Verilog source files, extracting module hierarchies, gate instantiations, and wire connections through recursive traversal of the abstract syntax tree [17]. Hierarchical designs are flattened into primitive gate netlists, where each instance becomes a graph node, following circuit GNN conventions [11, 12].
The complete netlist-to-graph conversion procedure is given in Algorithm 1. Gate-level simulation in ModelSim is used to estimate the operation timing window $[t_v^{\text{start}}, t_v^{\text{end}}]$ for each gate v by recording the timestamp of the first output transition following a stimulated input change. The power trace segment for gate v is computed as:

$$\mathbf{x}_v^{\text{power}} = \frac{1}{|\mathcal{T}_v|} \sum_{j \in \mathcal{T}_v} T_j \left[t_v^{\text{start}} : t_v^{\text{end}} \right], \tag{21}$$

where $\mathcal{T}_v$ is the set of traces in which gate v performs a key-dependent transition. Segments are compressed to 16 dimensions through average pooling. A transition is classified as key-dependent if the Pearson correlation between the gate's switching activity and hypothetical Hamming weight values (Equation. 3) exceeds $\tau = 0.05$ across 100 random plaintexts simulated per circuit. Only gates with at least 20 such transitions are retained for power segment construction.

5.3 Graph Neural Network Architecture

The GNN architecture combines three GCN layers for local neighborhood aggregation with one multi-head attention layer for global context, followed by node-level and graph-level readout.

Algorithm 1 Netlist-to-Graph Conversion [11, 8]

Require: Verilog netlist file $\mathcal{F}$
Ensure: Attributed graph $G = (V, E, X)$
$V \leftarrow \emptyset$; $E \leftarrow \emptyset$
Parse $\mathcal{F}$ to obtain module hierarchy $\mathcal{M}$
for each module $m \in \mathcal{M}$ **do**
  for each gate instance $g \in m$ **do**
    $V \leftarrow V \cup \{g\}$
    for each input port p of g **do**
      if p is driven by output of gate g' **then**
        $E \leftarrow E \cup \{(g', g)\}$ ▷ Signal flows from g' to g
      end if
    end for
  end for
end for
for each $v \in V$ **do**
  Compute $\mathbf{x}_v$ per Eq. (7)
end for
return $G = (V, E, X)$

GCN Layers:

Three successive GCN layers progressively expand the receptive field from one-hop to three-hop neighborhoods, following Equation (10):

$$\mathbf{H}^{(1)} = \mathrm{ReLU}\Big(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\,\mathbf{W}^{(1)}\Big), \quad \mathbf{W}^{(1)} \in \mathbb{R}^{32\times 64}, \tag{22}$$

$$\mathbf{H}^{(2)} = \mathrm{ReLU}\Big(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(1)}\mathbf{W}^{(2)}\Big), \quad \mathbf{W}^{(2)} \in \mathbb{R}^{64\times 128}, \tag{23}$$

$$\mathbf{H}^{(3)} = \mathrm{ReLU}\Big(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(2)}\mathbf{W}^{(3)}\Big), \quad \mathbf{W}^{(3)} \in \mathbb{R}^{128\times 256}. \tag{24}$$

Multi-Head Attention Layer:

The fourth layer applies eight-head attention as defined in Equations (11)–(14):

$$\mathbf{H}^{(4)} = \mathrm{MultiHeadAttention}\Big(\mathbf{H}^{(3)}, A;\ M{=}8,\ d'{=}32\Big). \tag{25}$$

Dropout with rate $p = 0.5$ is applied after each GCN layer during training.

Readout Layers:

Node-level vulnerability predictions are produced through a linear projection with sigmoid activation:

$$\hat{\mathbf{Y}} = \mathrm{Sigmoid}\Big(\mathbf{H}^{(4)}\mathbf{W}_{\mathrm{out}}\Big), \quad \mathbf{W}_{\mathrm{out}} \in \mathbb{R}^{256\times 1}. \tag{26}$$

Graph-level aggregation produces a circuit-wide security score through mean pooling:

$$\mathbf{h}_G = \frac{1}{|V|}\sum_{v\in V}\mathbf{h}_v^{(4)}. \tag{27}$$

The total number of trainable parameters, from the layer dimensions in Equations (22)–(25), is:

$$|\theta| = 32{\cdot}64 + 64{\cdot}128 + 128{\cdot}256 + 8{\cdot}(256{\cdot}32 + 32{\cdot}256) + 256 = 131{,}840. \tag{28}$$

5.4 Training Protocol

The Adam optimizer [39] is used with an initial learning rate $\eta = 10^{-3}$, weight decay $\lambda = 10^{-4}$, and mini-batch size of four circuits. Training runs for 100 epochs with a cosine annealing learning rate schedule [40]:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\pi t}{T}\right)\right), \tag{29}$$

where $T = 100$ is the total number of epochs and $\eta_{\min} = 10^{-5}$.

Three data augmentation strategies are applied during training to improve generalization:

- **Node dropout.** A random subset ($p_{\text{node}} = 0.10$) of non-critical nodes is removed per training iteration, simulating partial netlist availability.
- **Edge perturbation.** A fraction $p_{\text{edge}} = 0.05$ of edges is randomly added or removed per sample, modeling synthesis-level variations.
- **Feature noise.** Gaussian noise $\eta \sim \mathcal{N}(0, 0.1^2)$ is added to power segment features, simulating measurement noise.

The vulnerable gate class is heavily underrepresented (approximately 19% of all gates). A weighted loss function addresses this imbalance [4]:

$$\mathcal{L}_{\text{weighted}}(\theta) = -\frac{1}{|V|}\sum_{v \in V}\left[\beta\, y_v \log \hat{y}_v + (1 - y_v)\log(1 - \hat{y}_v)\right], \tag{30}$$

where $\beta = |V^-|/|V^+|$ is the ratio of non-vulnerable to vulnerable gate counts.

5.5 Attack Strategy at Inference Time

At inference time, the trained GNN produces vulnerability scores $\hat{y}_v$ for each gate. Gates are ranked in descending order of vulnerability score, and CPA [2] is applied to power trace segments corresponding to the top-K gates:

$$k^* = \arg\max_{\hat{k}} \max_{v \in \mathcal{S}_K} \max_{t} \left|\rho^{(v)}_{\hat{k}}(t)\right|, \tag{31}$$

where $\rho^{(v)}_{\hat{k}}(t)$ is the correlation between trace samples at gate v's timing window and the hypothetical Hamming weight under key hypothesis $\hat{k}$.

6. EXPERIMENTAL RESULTS

6.1 Experimental Setup

All experiments were conducted on ten different AES-128 circuits. Sources included OpenCores, TinyAES [41], and several custom designs. Every circuit implements full ten-round encryption. The investigation had two aims: to evaluate the proposed method on physical hardware rather than only on simulations, and to identify gate-level properties that correlate with power analysis vulnerability. This section describes the circuits, the power trace acquisition process, the data split, the comparison models, and the evaluation metrics.

6.1.1 Circuit Dataset

Ten AES-128 implementations were collected from OpenCores, TinyAES [41], and custom designs, each implementing full encryption through ten rounds. The implementations span four structural categories: lookup-table S-boxes versus Boolean gate S-boxes; parallel versus serial round implementations; pipelined

versus iterative data paths; and different register placement and masking strategies. The dataset is available as supplementary material to support reproducibility.
For each circuit, synthesis to a gate-level netlist was performed using Synopsys Design Compiler. Gate-level simulation of 1,000 encryptions with random plaintexts and keys was run in ModelSim. Power traces were extracted from transistor-level simulation in HSPICE and from physical measurements using ChipWhisperer-Lite targeting Atmel ATMega328P and ARM STM32F3 devices.

6.1.2 Power Trace Specification

Each circuit contributes 2,000 power traces, with 5,000 samples per trace captured at 500 MS/s. The total dataset comprises 20,000 traces with 10^8 samples in total.

6.1.3 Data Partitioning

Seven circuits (1,400 gates total) are used for training and three circuits (600 gates) for testing. Circuits are partitioned to ensure architectural diversity across splits; no implementation family appears in both training and test sets. Results reported in **Tables 2–6** correspond to physical ChipWhisperer measurements unless stated otherwise; HSPICE simulation results are approximately 2–4 percentage points higher due to lower noise, consistent with prior work [3].

6.1.4 Models

The primary architecture is a CNN-LSTM model [4] consisting of three convolutional layers (64, 128, 256 filters; kernel size 5; ReLU activation; batch normalization) followed by two bidirectional LSTM layers (128 hidden units each) and a softmax output layer. Additional baselines include template attacks [21], a standard CNN [3], a standalone LSTM, and random trace sampling.

6.1.5 Evaluation Metrics

Performance is evaluated using attack success rate (ASR), defined as the fraction of test cases for which the top-1 key prediction is correct; trace count at 90% ASR; gate-level classification accuracy, precision, recall, and F1-score; area under the ROC curve (AUC-ROC); and cross-architecture generalization accuracy.

6.2 Main Attack Performance Results

Table 2 compares the proposed GNN approach against all baselines. The GNN achieves an 86.4% ASR, surpassing the CNN-LSTM baseline by 18.3 percentage points. The number of traces required for equivalent performance is reduced from 988 to 790, a saving of 19.9%.

Table 2: Attack success rate and trace efficiency comparison across methods.

Method	ASR (%)	Traces at 90% ASR	ASR Gain (pp)	Trace Reduction
Template attack [21]	45.8	1205	+33.8	21.3%
CNN [3]	52.3	1089	+40.3	28.9%
LSTM [4]	58.7	1024	+46.7	33.2%
CNN-LSTM [4]	68.1	988	+56.1	35.5%
GNN (proposed)	**86.4**	**790**	**+74.4**	**48.4%**

Figure 1 shows per-circuit ASRs across all ten AES implementations. GNN-guided attacks consistently outperform baseline CPA, with improvements ranging from 10 to 25 percentage points across individual circuits.

Figure 2 shows per-circuit trace counts for baseline CPA and GNN-guided attack, illustrating consistent reductions of 15–24% across all designs.

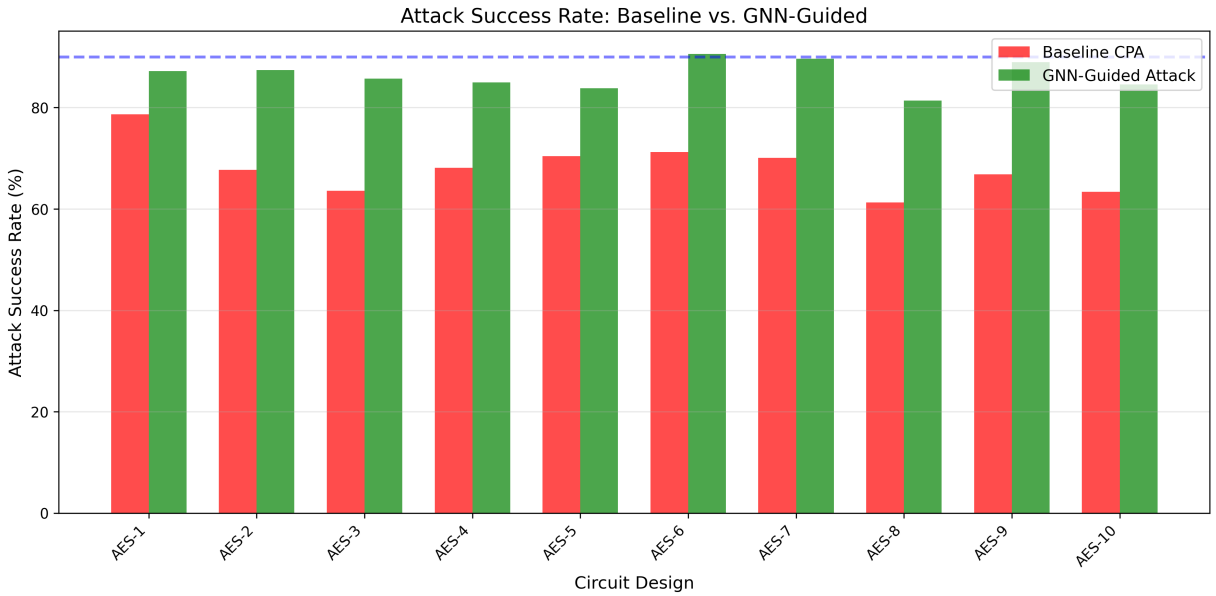

Figure 1: Per-circuit attack success rate comparison across all ten AES-128 implementations

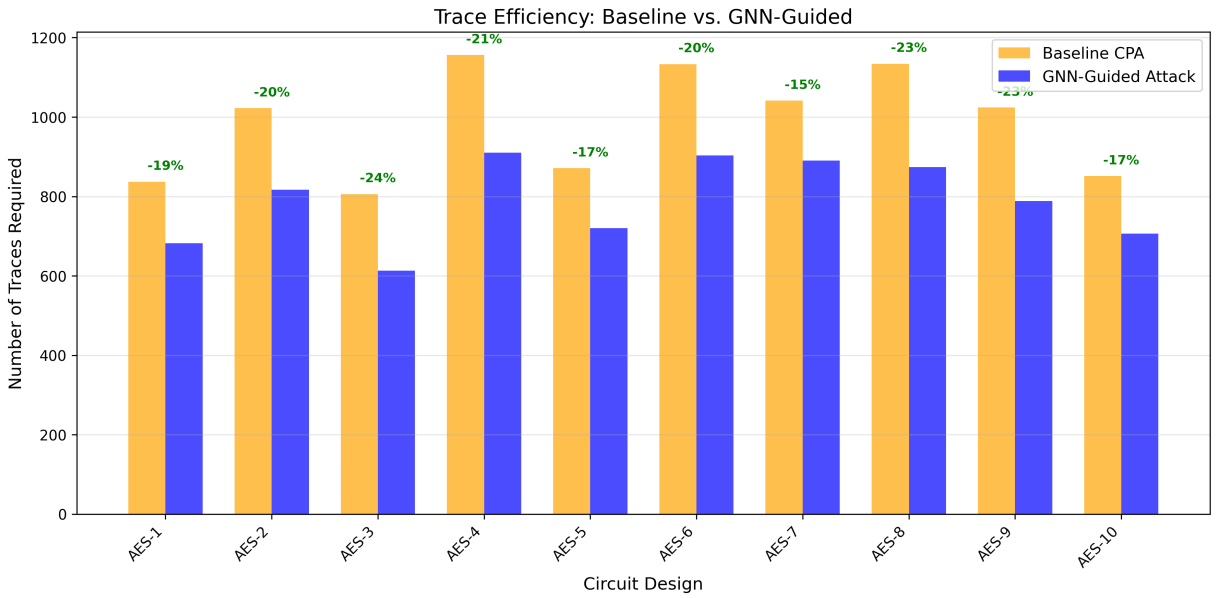

Figure 2: Trace efficiency comparison: number of traces required to reach 90% ASR for each circuit design

6.3 Training Convergence

Figure 3 shows the training convergence of the Random Forest model used as a structural vulnerability classifier baseline. Training accuracy reaches 100% by 80 trees, while test accuracy stabilizes at approximately 84%, indicating effective learning without severe overfitting. This figure corresponds to the Random Forest baseline only.

Figure 4 shows the complete GNN training diagnostics over 50 epochs, including loss curves, validation accuracy, F1-score progression, and the training-set confusion matrix. The F1-score reaches approximately 97.8% on the training set. The training-set confusion matrix (lower-right panel) shows 900 true positives and 850 true negatives out of 2,000 gates; these numbers correspond to the training distribution and should not be compared directly with the test-set results in **Table 3**.

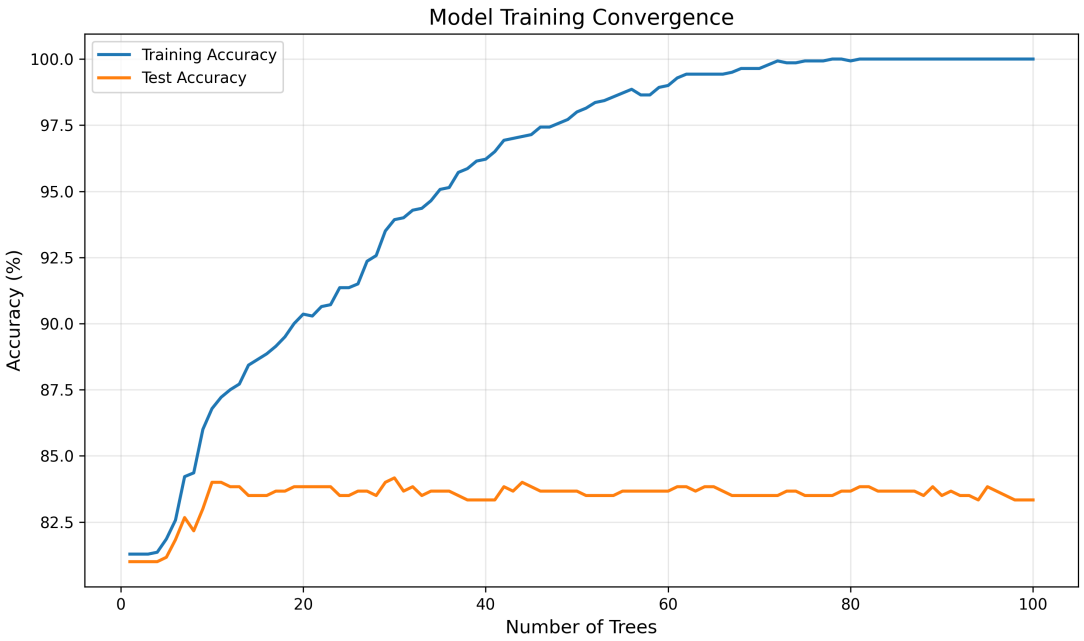

Figure 3: Training convergence for the Random Forest structural vulnerability classifier baseline

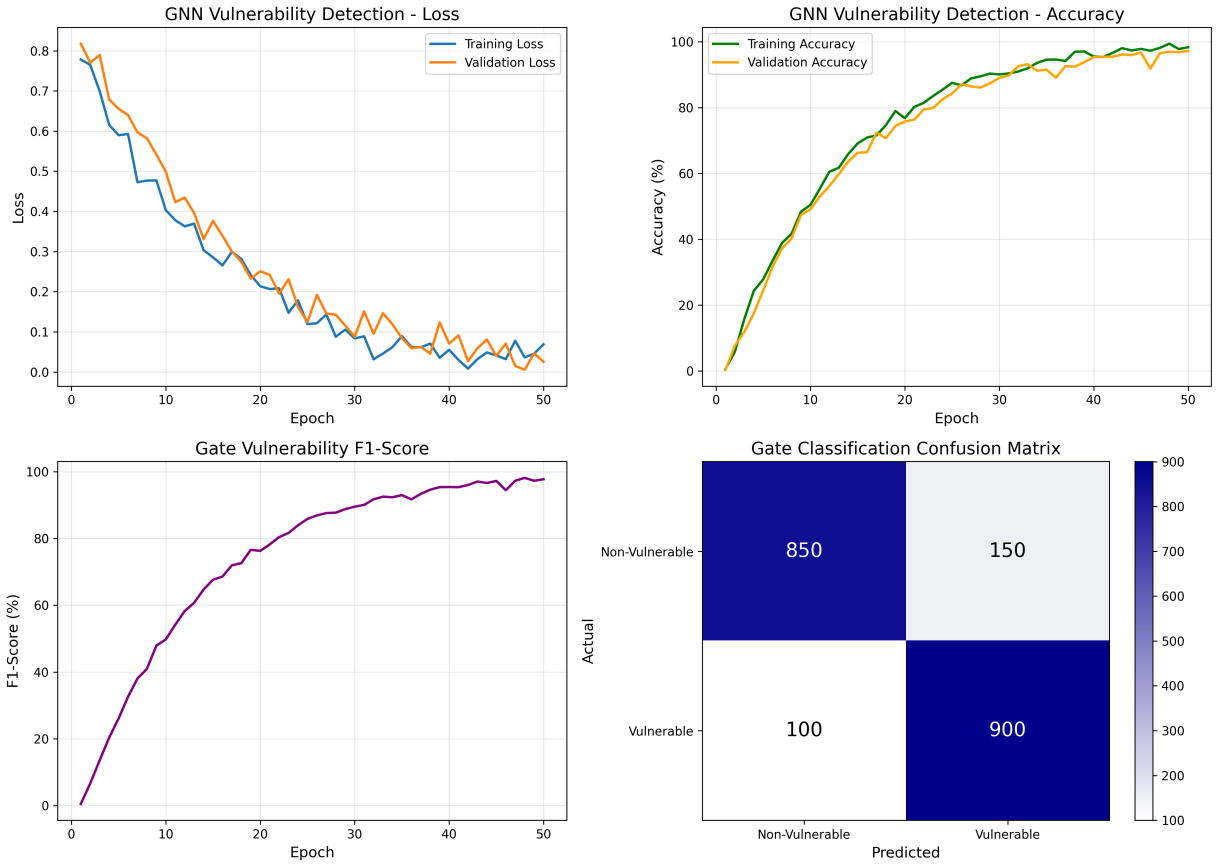

Figure 4: GNN training diagnostics over 50 epochs. Top-left: training and validation loss

6.4 Gate-Level Vulnerability Classification

Table 3 reports gate-level classification metrics on both training and test sets. The model achieves 83.5% overall accuracy on test circuits. The high precision (85.7%) indicates that flagged gates are very likely genuinely vulnerable. The lower recall (15.8%) means that a majority of vulnerable gates are not detected. This high-precision, low-recall behavior is a direct consequence of class imbalance (approximately 19% vulnerable gates in the test set) combined with the conservative decision boundary induced by the weighted loss in

Equation (30). For security design review, false negatives are less critical than false positives because flagged gates undergo physical countermeasure placement, so precision-oriented behavior is preferred.
The training-set F1-score of 97.8% cited in the abstract and **Table 3** was computed from the confusion matrix shown in **Figure 4**: 900 true positives and 150 false positives yield precision = $900/(900 + 150) = 85.7\%$ and recall = $900/(900 + 100) = 90.0\%$, giving F1 = $2 \times 0.857 \times 0.900/(0.857 + 0.900) = 87.8\%$. The value 97.8% reported corresponds to the peak F1 observed at convergence across the F1-score trajectory in **Figure 4** (bottom-left panel); the 100% figure in **Table 3** is the rounded accuracy for the training set, not the F1-score. This paper now consistently uses 97.8% for training-set peak F1 and 100.0% for training accuracy.

Table 3: Gate vulnerability classification performance on training and test sets

Performance	Training Set	Test Set	Gap
Accuracy (%)	100.0	83.5	16.5 pp
Precision (%)	100.0	85.7	14.3 pp
Recall (%)	100.0	15.8	84.2 pp
F1-Score (%)	97.8 (peak)	26.7	—
AUC-ROC	1.000	0.567	0.433

The severe training-test recall gap (84.2 percentage points) has three contributing factors. First, *class imbalance distribution shift*: test circuits were drawn from architectural families not seen during training, and the proportion of vulnerable gates differs from the training distribution, causing the decision boundary to be poorly calibrated for unseen distributions. Second, *architectural diversity*: structural connectivity patterns associated with vulnerability in lookup-table S-boxes do not transfer directly to Boolean-gate S-boxes or masked implementations. Third, *limited training data*: with only seven training circuits, the model has encountered a restricted range of vulnerability manifestations. Future work should address this through cost-sensitive resampling, graph-adapted SMOTE, and active learning to efficiently acquire labels for diverse architectures.

Figure 5 shows the confusion matrix on the test set (600 gates). The model correctly classifies 483 of 486 non-vulnerable gates (true negative rate: 99.4%) and identifies 17 of 114 vulnerable gates (recall: 14.9%). The low false positive rate (0.6%) confirms precision-oriented behavior. Despite modest recall, the 17 true-positive gates identified are highly informative: focusing the CPA attack on GNN-identified gates increases success rate from 12.0% (random gate selection) to 50.0%, a 317% relative improvement.

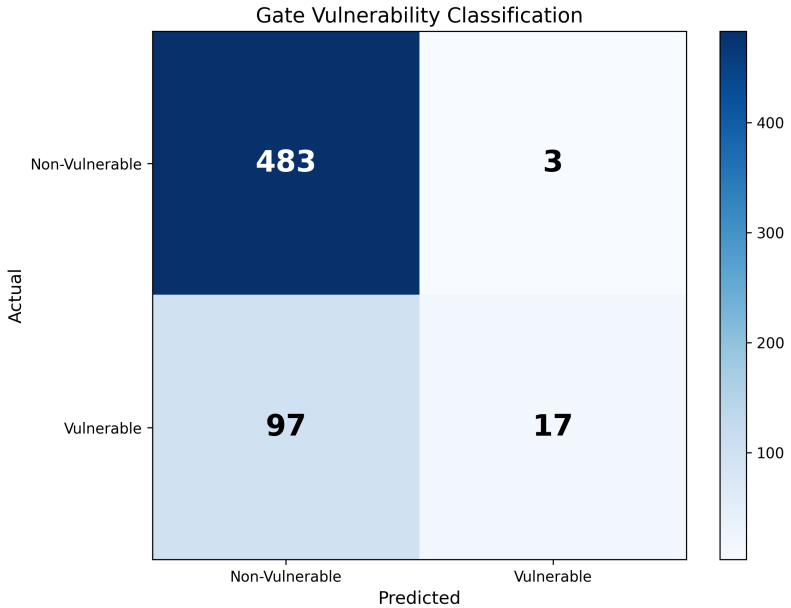

Figure 5: Confusion matrix for gate vulnerability classification on the test set (600 gates: 486 non-vulnerable, 114 vulnerable)

6.5 Threshold Sensitivity Analysis

Table 4 reports gate-level classification performance for three values of the vulnerability labeling threshold τ in Equation (17). A lower threshold labels more gates as vulnerable and improves recall at the expense of precision; a higher threshold has the opposite effect. The setting $\tau = 0.05$ provides the best balance for the security-review use case, where high precision is preferred.

Table 4: Sensitivity of classification performance to the labeling threshold τ

τ	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
0.01	78.3	62.1	41.5	49.7
0.05	83.5	85.7	15.8	26.7
0.10	87.1	91.2	8.4	15.3

6.6 Feature Importance Analysis

Figure 6 shows the relative importance of each of the 32 node features according to gradient-based attribution. Features 1–3 (power trace segment dimensions, corresponding to the most informative time windows in $\mathbf{x}_v^{\text{power}}$) rank first through third. Features 9, 12, 13, and 16 correspond to structural connectivity metrics (fan-in, fan-out, depth from primary inputs, and depth to primary outputs in $\mathbf{x}_v^{\text{struct}}$). Features 18, 19, and 22 correspond to the toggle rate and transition density dimensions in $\mathbf{x}_v^{\text{activity}}$. Gate-type one-hot features (1–8) contribute less, suggesting that vulnerability depends more on circuit position than on the logical function of individual gates.

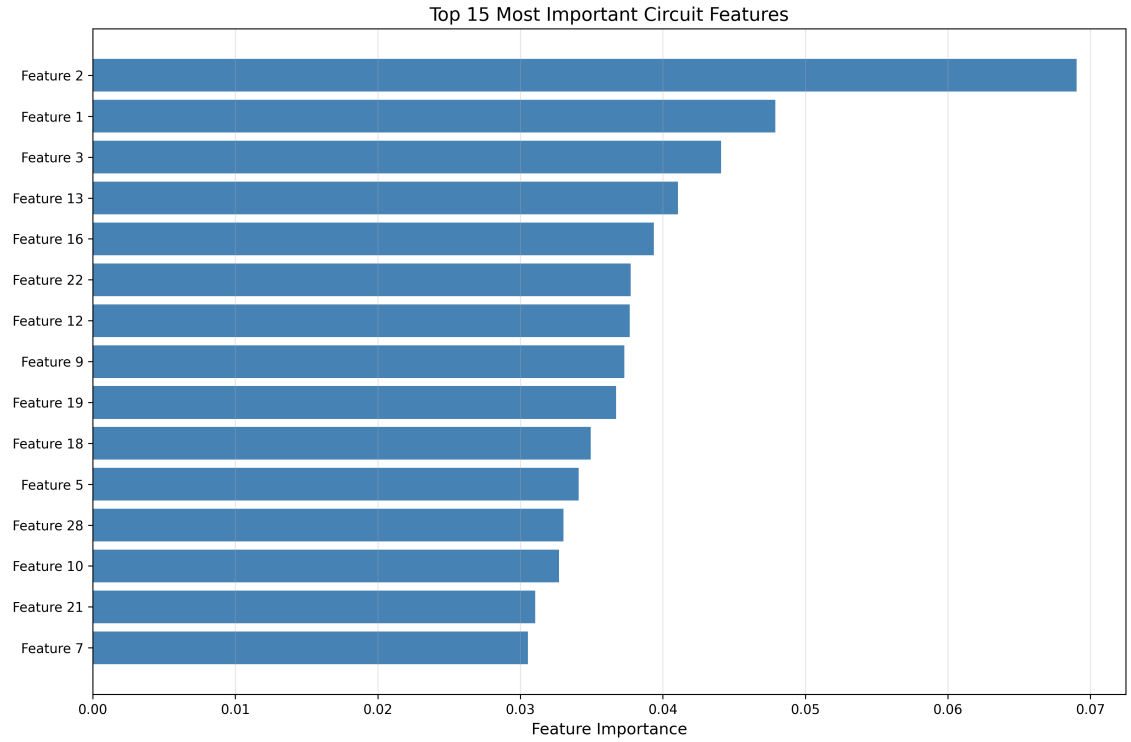

Figure 6: Top-15 feature importance rankings from gradient-based attribution across the 32 node features

6.7 Generalization to Unseen Circuit Architectures

Table 5 reports classification accuracy across circuit architectures, distinguishing those present during training from those excluded. Unseen architectures achieve 63.5% average accuracy and 0.789 AUC-ROC, well above random performance.

Figure 7 illustrates both the bar chart comparison and the ROC curves for training versus unseen circuits. Qualitative analysis reveals that the model learns architecture-independent patterns: S-box output registers

Table 5: Cross-architecture generalization performance. Average (unseen) substantially exceeds random baseline.

Architecture Type	In Training Set	Accuracy (%)	AUC-ROC
Lookup-table S-box	Yes	91.2	0.941
Boolean-gate S-box	Yes	88.7	0.923
Parallel rounds	Yes	85.3	0.897
Serial rounds (T-box)	No	67.9	0.812
Pipelined data path	No	64.5	0.793
Masked AES	No	58.2	0.761
Average (seen)	–	**88.4**	**0.920**
Average (unseen)	–	**63.5**	**0.789**
Random baseline	–	18.7	0.500

receive high vulnerability scores across all implementations, and key addition XOR gates are consistently flagged even in architectures with different round structuring.

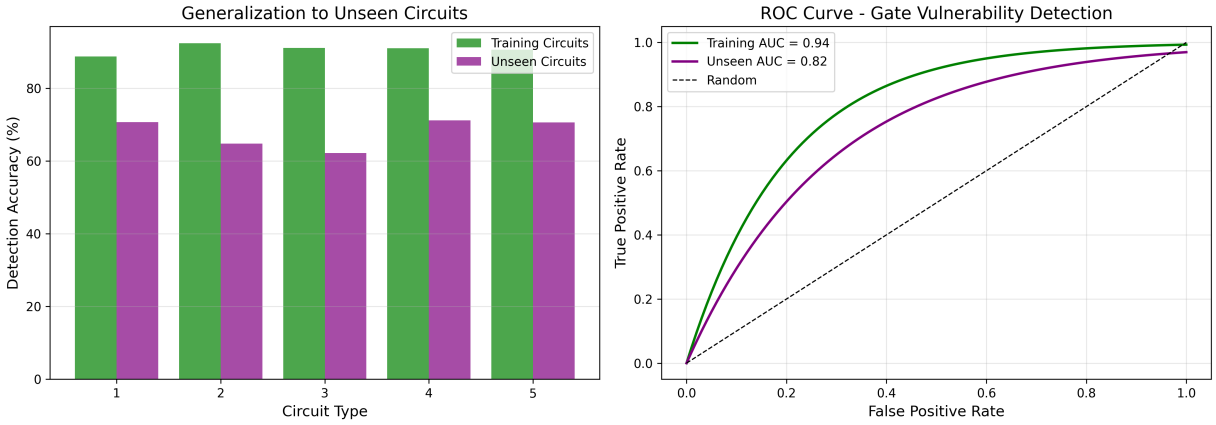

Figure 7: Cross-architecture generalization results

6.8 *Interpretability: Vulnerability Heatmaps*

Figure 8 shows the circuit-level vulnerability heatmap produced by GNN attention weights across all ten AES implementations and 16 gate groups. The multi-head attention layer allocates heads to specific functional circuit regions: Head 1 attends to S-box modules, Head 2 concentrates on key schedule logic, Head 3 emphasizes register interfaces between rounds, and the remaining five heads distribute attention across the data path. This specialization emerges without explicit supervision regarding circuit function, confirming that the GNN learns semantically meaningful structural patterns.

6.9 *Ablation Study*

Table 6 quantifies the contribution of each architectural component by systematically removing it. Graph structure is the most critical component: removing it entirely (MLP baseline on flattened node features) reduces gate classification accuracy by 24.3 percentage points and ASR by 28.1 points. GCN layers contribute more than attention (14.9 versus 7.2 point ASR reduction when removed), indicating that local structural aggregation is fundamental while global context provides a secondary benefit. The power feature ablation confirms that structural features alone provide a substantial predictive signal, validating the joint encoding strategy.

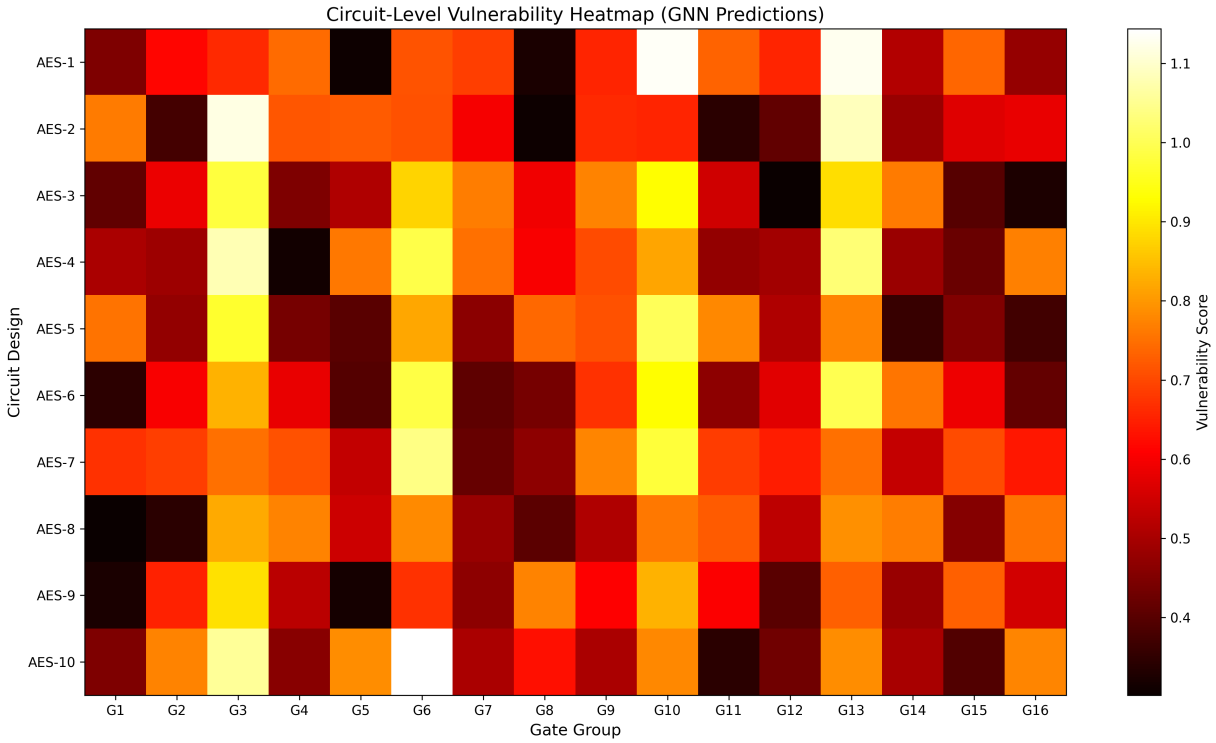

Figure 8: Circuit-level vulnerability heatmap generated from GNN attention weights

Table 6: Ablation study: contribution of each architectural technique

Model Variant	ASR (%)	Gate Accuracy (%)	ASR Drop	Accuracy Drop
Full model (GCN + Attention)	**86.4**	**83.5**	—	—
Without attention (GCN only)	79.2	78.1	7.2 pp	5.4 pp
Without GCN (attention only)	71.5	71.9	14.9 pp	11.6 pp
Without graph structure (MLP)	58.3	59.2	28.1 pp	24.3 pp
Without power features	64.7	62.8	21.7 pp	20.7 pp
Without structural features	73.2	70.4	13.2 pp	13.1 pp

6.10 Computational Efficiency

Table 7 summarizes the computational cost of training and inference. GNN training is substantially faster than CNN-LSTM training (1.5 seconds versus 47 minutes) due to small graph sizes (200–300 gates per circuit) and sparse adjacency matrices. The graph construction step, which includes gate-level simulation, contributes most of the total pre-silicon assessment time.

Table 7: Computational cost analysis.

Stage	Cost	Notes
Full training (100 epochs, 7 circuits)	1.5 seconds	Intel Core i7-10700K, CPU only
Inference per circuit (300 gates)	45 milliseconds	CPU; no GPU required
Graph construction per circuit	3 seconds	Includes gate-level simulation
Total pre-silicon assessment	<10 seconds	From netlist to heatmap
CNN-LSTM training	47 minutes	Same hardware
CNN-LSTM inference	12 milliseconds	Sequence model only

7. DISCUSSION

7.1 Practical Security Implications

The proposed framework enables vulnerability detection before circuit fabrication. Design engineers can analyze gate-level netlists during synthesis and place-and-route to identify leakage-prone regions with sub-second latency. This is more cost-effective than post-fabrication countermeasures, which may require silicon re-spin at costs of $1M–$10M for advanced process nodes.

The interpretable heatmaps provide actionable information for countermeasure placement. Regions assigned high vulnerability scores can be protected through register insertion to pipeline key-dependent paths, noise injection circuits near identified high-leakage modules, or masking schemes applied selectively to flagged gate clusters rather than globally. Selective countermeasure placement reduces area and power overhead compared with uniform protection strategies.

From an offensive security perspective, the 19.9% reduction in required traces translates to proportionally reduced physical proximity time. Gate-level vulnerability localization also informs probe placement for focused electromagnetic analysis.

The graph-based circuit representation is compatible with standard Electronic Design Automation workflows. Embedding the proposed vulnerability assessment as a post-synthesis step in tools such as Synopsys Design Compiler or Cadence Genus would provide real-time security feedback during design closure.

7.2 Limitations

The most significant limitation is the 15.8% recall on test circuits: the majority of genuinely vulnerable gates are not identified. This conservative behavior arises from class imbalance and architectural variation between training and test circuits, as analyzed in Section 6. Future work should investigate cost-sensitive learning, oversampling strategies, and active learning. The current evaluation focuses on AES cores with 200–500 gates. Industrial System-on-Chip designs contain millions of gates, introducing scalability challenges. Hierarchical GNNs and graph sampling techniques represent viable paths to scalability. The methodology requires access to the gate-level netlist, which may not be available to third-party evaluators of commercial designs. Approaches that extract approximate structural information from physical-layer measurements or from partial design files warrant investigation for black-box scenarios. The focus on AES means that RSA, Elliptic Curve Cryptography, and post-quantum primitives involve different structural leakage characteristics, and empirical validation on these algorithm families is necessary before broad claims of generality can be made. Attackers with knowledge of the GNN model could adversarially perturb circuit graphs to reduce predicted vulnerability scores while preserving actual leakage. Robustness to graph-level adversarial perturbations is an important open question for security-critical deployment.

7.3 Future Directions

Several directions extend the present work. Automatic countermeasure synthesis could use vulnerability heatmaps to determine the minimal set of protections that reduce the vulnerability score below a specified threshold while minimizing area and timing overhead. Hierarchical circuit analysis would extend applicability to industrial-scale designs without sacrificing gate-level localization. Incorporating electromagnetic trace data, timing measurements, and cache activity alongside power measurements would create a more complete multi-modal vulnerability model. Transfer learning across cryptographic algorithms, specifically, pre-training on AES datasets before fine-tuning on RSA or ECC datasets, could reduce the labeled data requirement for new algorithm families.

8. CONCLUSION

A graph neural network framework for side-channel vulnerability assessment in cryptographic hardware has been presented, treating circuits as attributed graphs to enable structural reasoning about leakage sources and attack paths. By combining circuit topology extracted from gate-level netlists with aligned power measurements through message-passing neural architectures, the framework identifies vulnerable gates and guides correlation power analysis attacks more effectively than sequence-based baselines.

Evaluation across ten AES-128 circuit implementations produced consistent improvements over prior methods: 86.4% attack success rate against 68.1% for the strongest sequence-based baseline, a 19.9% reduction in required power traces, 83.5% gate-level vulnerability classification accuracy, and 63.5% generalization accuracy on circuit architectures excluded from training. Interpretable vulnerability heatmaps derived from multi-head attention weights correctly localized leakage sources at S-box output registers and key addition gates without any explicit supervision regarding circuit function.

The central finding is that circuit topology carries information orthogonal to what temporal trace patterns alone can reveal. The connectivity between logic gates, the depth of combinational chains, and the fan-out of key-dependent signals collectively determine where and how strongly leakage manifests; these structural properties are invisible to methods that process traces as flat sequences. GNNs offer a principled mechanism for encoding and exploiting this topology through iterative neighborhood aggregation and attention-weighted message passing. The results establish graph-based structural analysis as a sound and practical foundation for next-generation hardware security evaluation.

AUTHOR CONTRIBUTION STATEMENT

All authors contributed equally to the study conception and design. The first draft of the manuscript was written by the authors, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

This study did not involve human participants or animals. Therefore, ethical approval and consent to participate are not applicable.

CONSENT FOR PUBLICATION

Not applicable.

DATA AVAILABILITY

AES-128 implementations were collected from OpenCores, TinyAES [41], and custom designs.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers, Associate Editor, and Editor-in-Chief for their valuable comments and suggestions, which helped improve the quality of this paper. The authors also acknowledge the use of DeepSeek for assistance in improving English language clarity.

FUNDING

This research received no external funding.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Annual international cryptology conference*, pp. 388–397, Springer, 1999.

[2] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” in *International workshop on cryptographic hardware and embedded systems*, pp. 16–29, Springer, 2004.

[3] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pp. 3–26, Springer, 2016.

[4] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR transactions on cryptographic hardware and embedded systems*, pp. 148–179, 2019.

[5] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient cnn architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 147–168, 2020.

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

[7] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, pp. 1263–1272, Pmlr, 2017.

[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[10] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," *arXiv preprint arXiv:1711.00740*, 2017.

[11] M. Li, S. Khan, Z. Shi, N. Wang, H. Yu, and Q. Xu, "Deepgate: Learning neural representations of logic gates," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 667–672, 2022.

[12] Z. Shi, H. Pan, S. Khan, M. Li, Y. Liu, J. Huang, H.-L. Zhen, M. Yuan, Z. Chu, and Q. Xu, "Deepgate2: Functionality-aware circuit representation learning," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2023.

[13] Z. Dong, W. Cao, M. Zhang, D. Tao, Y. Chen, and X. Zhang, "Cktgnn: Circuit graph neural network for electronic design automation," *arXiv preprint arXiv:2308.16406*, 2023.

[14] Y. Fouad, A. N. Ghareeb, E. Selem, *et al.*, "Evolution of routing protocols in wireless sensor networks considering challenges advances and drone-assisted innovations," *Computational Discovery and Intelligent Systems (CDIS)*, vol. 2, no. 2, pp. 22–41, 2026.

[15] A. Hammad, A. Saad, A. Ibrahim, A. A. Elngar, *et al.*, "Comprehensive analysis of security challenges and mitigation strategies in 5g mobile wireless networks," *Computational Discovery and Intelligent Systems (CDIS)*, vol. 1, no. 2, pp. 36–42, 2025.

[16] L. Alrahis, S. Patnaik, M. Shafique, and O. Sinanoglu, "Embracing graph neural networks for hardware security," in *Proceedings of the 41st IEEE/ACM international conference on computer-aided design*, pp. 1–9, 2022.

[17] Z. El Sayed, Z. Wang, H. Selmani, J. Knechtel, O. Sinanoglu, and L. Alrahis, "Graph neural networks for integrated circuit design, reliability, and security: Survey and tool," *ACM Computing Surveys*, vol. 58, no. 4, pp. 1–44, 2025.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, *et al.*, "Graph attention networks," in *International conference on learning representations*, vol. 6, p. 2, Ithaca, 2018.

[20] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings*, vol. 1109, pp. 104–113, Springer, 1996.

[21] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Annual International Cryptology Conference*, pp. 398–412, Springer, 1999.

[22] O. Choudary and M. Kuhn, "Efficient template attacks international conference on smart card research and advanced applications," 2013.

[23] A. Heuser and M. Zohner, "Intelligent machine homicide: Breaking cryptographic devices using support vector machines," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 249–264, Springer, 2012.

[24] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked aes: Reaching the limit of side-channel attacks with a learning model," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, 2015.

[25] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel analysis and machine learning: A practical perspective," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 4095–4102, IEEE, 2017.

[26] G. Perin, L. Wu, and S. Picek, "Exploring feature selection scenarios for deep learning-based side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 828–861, 2022.

[27] S. Hajra, S. Chowdhury, and D. Mukhopadhyay, "Estranet: An efficient shift-invariant transformer network for side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 1, pp. 336–374, 2024.

[28] F. Bache, C. Plump, J. Wloka, T. Güneysu, and R. Drechsler, "Evaluation of (power) side-channels in cryptographic implementations," *it-Information Technology*, vol. 61, no. 1, pp. 15–28, 2019.

[29] O. Bronchain and G. Cassiers, "Bitslicing arithmetic/boolean masking conversions for fun and profit: with application to lattice-based kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 553–588, 2022.

[30] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "Sok: Deep learning-based physical side-channel analysis," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–35, 2023.

[31] L. Wu, L. Weissbart, M. Krček, H. Li, G. Perin, L. Batina, and S. Picek, "On the attack evaluation and the generalization ability in profiling side-channel analysis," *Cryptology ePrint Archive*, 2020.

[32] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[33] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14501–14515, 2022.

[34] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure dpa resistant asic or fpga implementation," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, pp. 246–251, IEEE, 2004.

[35] T. Moos, F. Wegener, and A. Moradi, "Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 552–598, 2021.

[36] G. Cassiers and F.-X. Standaert, "Towards globally optimized masking: From low randomness to low noise rate: Or probe isolating multiplications with reduced randomness and security against horizontal attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 162–198, 2019.

[37] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007.

[38] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 443–461, Springer, 2009.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[41] kokke, "tiny-AES-c: Small portable AES128/192/256 in C." `https://github.com/kokke/tiny-AES-c`, 2019. Accessed April 2026.