GALA: An Explainable GNN-based Approach for Enhancing Oracle-Less Logic Locking Attacks Using Functional and Behavioral Features

Yeganeh Aghamohammadi
University of California, Santa Barbara
Santa Barbara, CA, USA
yeganeh@ucsb.edu

Henry Jin
University of California, Santa Barbara
Santa Barbara, CA, USA
henry_jin@ucsb.edu

Amin Rezaei
California State University, Long Beach
Long Beach, CA, USA
amin.rezaei@csulb.edu

***Abstract*—With the rise of fabless manufacturing, the risks of piracy and overproduction in integrated circuits have become more pressing, making it crucial to analyze and prevent hardware-based attacks. Although existing machine learning oracle-less attacks on logic-locked circuits are able to report approximate keys, they often struggle to produce operationally effective keys because they focus mainly on the structural topology of the circuits. This paper addresses this limitation by incorporating both functional features, such as output corruptibility, and behavioral features, like power consumption and area overhead, into graph neural network-based circuit modeling attacks. With the help of both subgraph-level and graph-level attack strategies, we achieve notable improvements in rendering a meaningful key compared to existing oracle-less methods. In addition, our graph-level model is explainable, providing insights into the learning process and how the attack is executed. These findings are critical for chip design houses looking to identify and address security vulnerabilities, ultimately safeguarding hardware intellectual property.**

***Index Terms*—Logic Locking; Logic Encryption; Machine Learning; Graph Neural Networks; Explainability**

I. Introduction

As outsourcing manufacturing becomes the norm in the semiconductor industry, it raises critical concerns including hardware Intellectual Property (IP) theft and overproduction. Logic locking (also referred to as logic encryption or obfuscation) [1]–[15] helps mitigate these threats by incorporating key-controlled gates into circuit designs. Only the correct key—known exclusively to the designer—produces the proper output, while any incorrect key corrupts the output, thus protecting against unauthorized use.

One significant threat model is the Oracle-Guided (OG) attack [16]–[26], which relies on having both an activated Integrated Circuit (IC) and a logic-locked netlist—often leaked from untrusted foundries—to systematically eliminate incorrect keys using SAT solvers. But new Oracle-Less (OL) attacks are even more dangerous because they only need the locked netlist. This shows how vulnerable hardware IPs are in settings with limited resources and makes it even more important to have strong defenses.

Recent advances in Machine Learning (ML) have paved the way for more sophisticated OL attacks that use predictive models to infer circuit keys [27]–[33]. Graph Neural Networks (GNNs) are especially effective because circuits can be represented as graphs, where nodes correspond to gates and edges denote their connections. By learning relationships within these graph structures, GNNs excel at extracting patterns crucial for circuit unlocking [34]. While ML-based OL attacks often achieve high prediction accuracy—measured by minimizing the Hamming distance between the predicted and correct keys—they lack interpretability, making it difficult to understand why certain key predictions succeed or fail. This limitation complicates the development of effective countermeasures.

This paper introduces **GALA**, a functionality-oriented and behaviorally-guided OL attack that leverages GNNs to predict keys for logic-locked circuits. Unlike previous approaches, it factors in circuit functionality metrics—such as area and power consumption—to improve both explainability and efficacy. By addressing shortcomings in existing attacks, our methodology offers a fresh perspective on protecting hardware IP.

II. Preliminaries

This section reviews logic locking methods and ML-based OL attacks, identifies gaps in attacker strategies, and outlines our contributions to bridge these gaps.

A. Background

In XOR-based logic locking [1], key-bits are used in combination with random inverters and buffers. Key-bit-controlled XOR gates replace selected buffers and inverters, where an XOR gate hiding a buffer corresponds to a key-bit of "0," and an XOR gate hiding an inverter corresponds to a key-bit of "1." MUX-based logic locking [2] employs 2-to-1 MUXs to replace random signals. In this approach, MUX inputs consist of real signals and random dummy signals, while the selectors act as key-bits. The correct key must choose the real signal terminal over the dummy one. While traditional logic locking methods have been vulnerable to SAT-based OG attack [16], post-SAT locking schemes such as SAR-Lock [4] and Anti-SAT [5] have been introduced to significantly increase the number of input patterns required to eliminate incorrect keys

through SAT-based OG attacks. As a more advanced post-SAT method, Bilateral Logic Encryption (BLE) [6] applies obfuscation and integrated locking specifically to a sensitive component of the circuit which reduces performance overhead compared to locking the entire circuit.

Recent studies have demonstrated progress in ML-based OL attacks. UNTANGLE [31] formulates the key-extraction task as a link prediction problem and leverages a GNN to learn the composition of gates in the locked netlist. In addition, OMLA [32] classifies subgraphs to resolve key-bit values, extracting small subgraphs for each key-gate that represent these values. Finally, LIPSTICK [33] is a type of OL attack that utilizes GNNs to predict the correct keys for logic-locked circuits by iteratively refining its predictions through learning from the structure and behavior of the circuit.

One of the main drawbacks of ML models is their lack of interpretability. This limitation can be addressed by developing post-hoc explanation methods for predictions, which has given rise to the field of explainability [35]. To achieve accurate similarity estimations and discriminative feature representations, SGGNN [36] employs graph computation during both the training and testing phases of deep networks. Additionally, PGExplainer [37] utilizes a trained GNN model to provide coherent justifications for the predictions. Notably, PGExplainer can operate in an inductive scenario, inferring explanations for ambiguous nodes without requiring retraining of the explanation model.

B. Research Gaps

❶ The initial observation from state-of-the-art works is that ML-based OL attacks are fundamentally approximate, as they strive to identify a nearly accurate key by optimizing specific parameters, like reducing the Hamming distance between the actual key and the one predicted by the attack. So, the ultimate goal in these attacks is to report a key as close as possible to the correct key based on a defined parameter. This is justifiable because, unlike OG attacks which require access to an oracle to recover the exact key, OL attacks operate under a weaker threat model, relying solely on access to a locked netlist without requiring an activated IC. However, a key question remains: what advantage does an attacker gain from recovering an approximate key? We address this question in detail in Section III.

❷ Key Prediction Accuracy (KPA) which is initially used in [27], is a metric defined as the percentage of key bits accurately predicted for a given netlist, with each key bit being predicted independently. KPA has been widely used in research to evaluate the accuracy of recovered keys—and even adopted in cybersecurity competitions like the CSAW Logic Locking Conquest to rank participants—a high KPA does not necessarily imply that the locked circuit will function correctly or provide any practical advantage to the attacker. For instance, as it is illustrated in Fig. 1, if we add one extra key bit and XOR it with one of the outputs, there will be an approximate key with a very high KPA (i.e., only the newly introduced key bit is different from the correct key), but it has 100% output corruptibility if inserted into the locked circuit. This example highlights the fact that a useful key precision metric should go beyond the structure of the circuits. A recent study [33] has explored the effect of incorporating key Error Rate (ER) as a functionality feature into circuit labels to predict more accurate keys. However, similar to other OL attacks [27]–[32] it still falls short in considering the overall behavior of the circuit, including power consumption and area overhead, under the reported key.

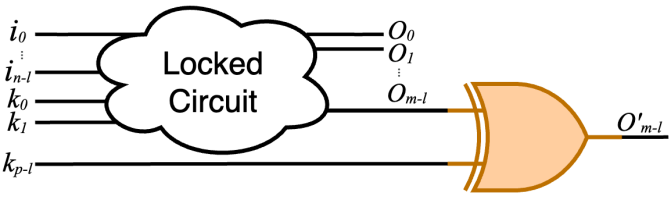


Fig. 1: Counterexample for using KPA as a meaningful reported key accuracy metric

❸ Additionally, in state-of-the-art works, a comprehensive security assessment of logic locking techniques is often neglected. Without insight into how a model arrives at its predictions—such as which features or circuit structures influence key recovery—designers and security analysts are left with a "black box" that offers little actionable feedback. This lack of transparency hinders the ability to validate the attack's effectiveness, understand its limitations, or improve the resilience of locking schemes. Moreover, it reduces trust in the results, especially in high-stakes environments where decisions about IP protection must be based on verifiable evidence. Thus, it is essential to consider explainable ML models [38] to provide more reliable and trustworthy predictions for IC design houses and decision makers.

C. Contributions

In this paper, we address the following two questions motivated by the identified research gaps in Section II-B.

First, **what other circuit functionality features, besides key ER, can we integrate into the GNN attack to improve key prediction accuracy while preserving the integrity of the feature learning process?** This question arises from the observation that relying solely on Hamming distance as a performance metric is inherently limited. While Hamming distance measures the bitwise difference between the predicted and actual keys, it does not capture the functional correctness of the recovered key or its impact on the circuit's behavior. We aim to assess the effectiveness of GNN attacks and which parameters or set of parameters are meaningful in this context. Thus, we will first explore why existing OL attacks struggle to report an approximate key with high precision, despite having high ML model accuracy, and to address this issue, we will introduce the incorporation of circuit behavioral features such as power consumption and area overhead into the learning process of both a subgraph-level attack [32] and a graph-level

attack [33], comparing the precision of the newly reported keys.

Second, **what features does the GNN model learn when only structural analysis is involved, and to what extent does each feature contribute to the model's inference?** Without explainability, it remains unclear which structural features are most influential in guiding the model's predictions. For instance, does the model focus more on nodes with high centrality, specific subgraph motifs, or particular gate combinations? And how do these features correlate with the likelihood of a node being part of the key logic? An explainable ML model would answer this question. Thus, we will employ an explainable GNN that utilizes techniques such as rule extraction and explainable reasoning [39]. This will help us understand data behavior and identify relationships between nodes, edges, and node features, as well as visualize the hidden representations learned by the GNN. By examining these relationships, we can uncover crucial patterns and topologies that are effective in determining key-bit values.

III. Prepare for GALA!

In this section, we propose GALA which is an explainable GNN-based Approach for enhancing oracle-less Logic locking Attacks using functional and behavioral features.

A. Problem Statement

Let's first formally define the logic locking problem. Considering n, m, and p be the sizes of the input, output, and key, respectively, we define the original circuit as $\mathbb{F} : \{0,1\}^n \rightarrow \{0,1\}^m$ and the locked circuit as $\mathbb{G} : \{0,1\}^p \times \{0,1\}^n \rightarrow \{0,1\}^m$ in which there is a p-bit correct key $K^* = (k_0^*, k_1^*, ..., k_{p-1}^*) : \{0,1\}^p$ such that $\mathbb{F}(X) = \mathbb{G}(X, K^*)$. The goal of the OL attack is to find an approximate key $K^a \approx K^*$, where the quality of approximation can be evaluated using well-defined quantitative metrics.

B. Claims

While KPA is defined in Section II-B, we expand on Key Precision Rate (KPR) to illustrate how it extends the concept of circuit "corruptibility" (widely studied in previous works) into a measurable metric for evaluating the functional fidelity of an approximate key. KPR measures the fraction of input patterns or operational conditions under which a logic-locked circuit, running on a given approximate key K^a, replicates the original circuit's functional behavior. In other words, KPA tracks correctness at the bit level, whereas KPR focuses on the actual usability of K^a for reproducing intended circuit outputs. To clarify our methodology for KPR, we adopt the following approach:

Input Pattern Sampling: For smaller circuits, we can exhaustively test all input vectors. For larger circuits, we sample a statistically significant set of input patterns.

Functional Comparison: We measure the fraction of correct outputs relative to the original circuit's outputs, averaged over all tested inputs. In experiments where we report just the "KPR increase" (e.g., relative gains over a baseline), we still compute KPR in this manner but focus on changes in functional fidelity over multiple runs.

Aggregate Statistics: We report average KPR for each circuit across multiple runs to smooth out anomalies. When partial results are given, they reflect incremental improvements or comparisons to highlight the efficacy of different attacks.

Our analysis leads to the following observations:

1) **High KPA Does Not Guarantee Functional Accuracy**: Although existing GNN-based attacks may achieve high KPA, they may fail to produce a functionally accurate key K^a. This discrepancy arises because correctly predicting a subset of key bits does not necessarily translate to an overall correct circuit response. Recent findings [33], [40] empirically show that high KPA does not guarantee functional equivalence (i.e., high KPR).
2) **High KPR Does Not Depend Solely on KPA**: An approximate key K^a may still exhibit a high KPR despite having several key bits predicted incorrectly. Functional behavior hinges on how sensitive the circuit is to certain key locations. Consequently, a high KPA can be sufficient but not necessary for high KPR—and vice versa. Even if KPR is high but remains below 100%, it may not yield a fully usable circuit for an attacker; partial functional correctness does not automatically translate into a reproducible or marketable IP clone.

Figures such as Fig. 1 illustrate cases where an approximate key with high KPA yields a lower-than-expected KPR. While formal proofs for these observations lie beyond this paper's scope, our empirical data (and prior examples in [33], [40]) confirm that a dual-metric approach encompassing both KPA and KPR is crucial for accurately assessing ML-based logic-locking attacks. These findings highlight that properly measured KPR can offer a more nuanced perspective on "usability" than KPA alone; however, it does not guarantee a circuit is fit for an attacker's use unless it is near or at 100%. By acknowledging the advantages and constraints of both metrics, we aim to shed light on the complexities of evaluating logic-locking attacks.

C. Attack

Consider an undirected graph $G = (V, E, X, A)$ where V represents the vertices, E shows the edges, X represents the feature vectors of the vertices, and A is the adjacency matrix of the graph. GNNs process this graph data, using the connections and node information to make predictions about individual nodes, connections, or the entire graph. GNNs achieve this by repeatedly refining the representation of each node based on its neighbors.

$$a_v{}^{(l)} = AGGREGATE^{(l)}(h_u{}^{(l-1)} : u \in \mathcal{N}(v)) \quad (1)$$

$$h_v{}^{(l)} = COMBINE^{(l)}(h_v{}^{(l-1)}, a_v{}^{(l)}) \quad (2)$$

$$h_G = READOUT(\{h_v{}^{(l)} | v \in G\}) \quad (3)$$

Equations 1 and 2 show the calculations behind the l-th layer of a GNN where $h_v^{(l)}$ represents the representation

of node v at the l-th layer and $N(v)$ represents the set of nodes adjacent to v. The choice of GNN method defines which aggregate and combine functions to use. In addition, for subgraph-level or graph-level classification, readout function in Equation 3 gives the entire graph representation through aggregation of node information in the final layer L.

Existing GNN-based attacks use the Graph Isomorphism Network (GIN) architecture as their GNN which has been shown to be as powerful as the Weisfeiler-Lehman graph isomorphism test in distinguishing graphs [39]. It updates node representations as shown in Equation 4 through a sum aggregator to ensure the aggregation is injective, where MLP represents a multi-layer perceptron. This introduces a non-linear transformation that allows the network to learn more complex representations of node features.

$$h_v^{(l)} = MLP^{(l)}(h_v^{(l-1)} + \sum_{u \in N(v)} h_u^{(l-1)})) \tag{4}$$

OMLA [32] represents the netlist as an undirected graph and extracts $|V_k|$ subgraphs using h-hop sampling, each centered on a key-gate $v \in V_k$. The sampler constructs a subgraph G_{sub} that includes v and all nodes within h-hops, assigning one-hot encoded feature vectors based on gate function, connectivity to primary/key inputs, and outputs. Distance encoding captures proximity to the central key-gate. To retain IN/OUT complexities despite using undirected graphs, OMLA assigns a signed value (-/+) to each node. An L-layer GNN then classifies the subgraphs, predicting key-bit values. Instead of relying solely on the final layer's embeddings, Equation 3 concatenates embeddings from all GIN layers $(0, 1, \ldots L)$ for richer graph representation.

LIPSTICK [33] unlocks the circuits by analyzing the entire graph structure alongside additional labels, such as the locking mechanism used, as well as incorporating functionality features such as ER, describing how the circuit performs under the incorrect key (i.e., focused on KPR) rather than how close it appears to the actual key (i.e., instead of focusing on KPA). ER is formally defined as the number of input patterns in which $\mathbb{F}(X) \neq \mathbb{G}(X, K)$, divided by all input patterns. LIPSTICK uses the same netlist-to-subgraph tool as OMLA to extract a graph representation from the Verilog files. It utilizes the GIN architecture along with Leaky ReLU activation to extract a graph-level embedding with the whole key as the target label, focusing on predicting a key with a low ER.

To build upon the foundations of OMLA and LIPSTICK, we introduce GALA, a GNN-based attack that integrates additional behavioral parameters—namely circuit-level area and power consumption (covering gate-level static and dynamic power)—into the feature maps used by these models. Although prior ML-based attacks often rely on structural cues alone, GALA leverages these extra circuit features to gain deeper insight into resource utilization patterns and thus identify potential vulnerabilities more effectively.

By incorporating the following additional features, GALA aims to enhance both KPA and KPR.

1) Power Consumption: We compute power consumption by combining gate-level dynamic power (derived from typical simulation workloads) and static leakage power. These values are aggregated for each gate or subcircuit.
2) Area Metrics: We derive area metrics from synthesized layouts, capturing how much physical space each gate or module occupies.
3) Graph Encoding: We normalize and encode both power and area metrics as node (and, where relevant, edge) attributes in the circuit's graph representation. This ensures that our GNN can exploit correlations between resource usage patterns and the functionality of locked gates.

In practical terms, designers often tweak power and area characteristics to hide or reveal locking configurations. By modeling these behavioral nuances, GALA can better pinpoint structurally significant nodes and edges that correlate with key bits, ultimately improving the likelihood of discovering an approximate key K^a that preserves intended circuit behavior (i.e., higher KPR).

In addition to higher KPA/KPR results, GALA offers a degree of interpretability by highlighting the key subcircuits or nodes most responsible for the predicted approximate key. To ensure the reliability of predictions and enhance interpretability, we employ the PGExplainer model [37] to analyze the decision-making process of GALA. By mapping predictions back to specific graph features or subgraphs, we identify how power and area parameters contribute to the model's understanding of security vulnerabilities. For example, circuits with high static power consumption or larger physical areas exhibit distinct patterns in the explainable graphs, aligning with known characteristics of resource-intensive designs. These insights underscore how GALA's extended feature set enables a deeper exploration of the complex interplay between design attributes and security flaws. Overall, GALA represents a significant advancement over OMLA and LIPSTICK by incorporating high-level design features into the GNN framework. These features not only improve the model's predictive performance but also provide actionable insights into the underlying vulnerabilities, addressing critical challenges in IC security.

IV. Experimental Results

We implemented OMLA [32] and LIPSTICK [33] on an Intel Core i5-1035G1 CPU, with a RAM size of 12.0 GB. We adopt ISCAS-85 [41] benchmarks that are used in the original

TABLE I: ISCAS-85 benchmarks [41] information

Bench.	Gates	Functionality
c1355	1503	32-bit single-error corrector
c1908	1289	16-bit single-error corrector and double-error detector
c2670	1262	12-bit arithmetic logic unit and controller
c3540	1403	8-bit arithmetic logic unit
c5315	1350	9-bit arithmetic logic unit
c6288	4703	16x16 multiplier
c7552	1241	32-bit adder and comparator

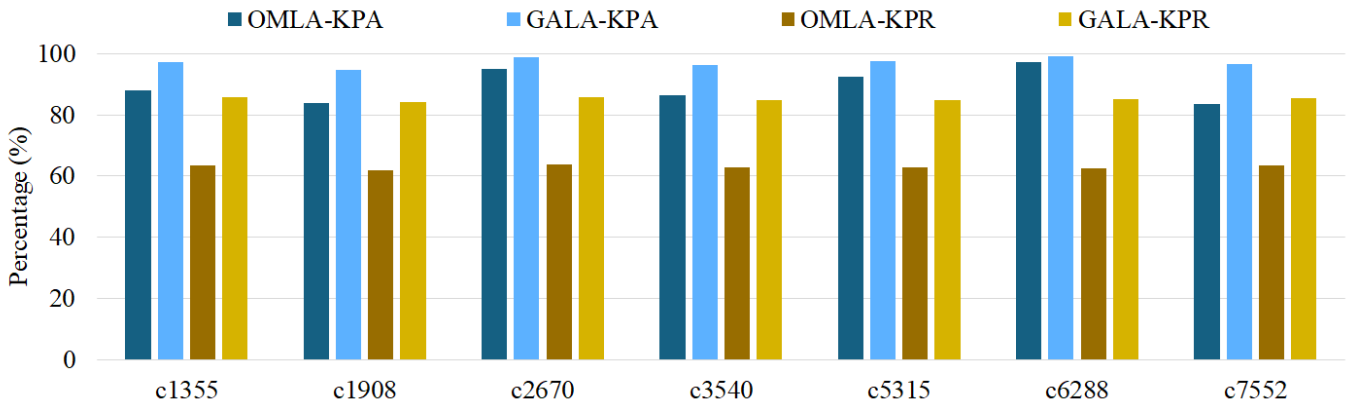


Fig. 2: GALA vs. OMLA attacking results

references as shown in Table I locked with five logic locking methods, including XOR-based locking [1], MUX-based locking [2], LUT-based locking [3], SAR-Lock [4], and BLE [6] all with a 64-bit key size. We also use two additional logic locking methods, Anti-SAT [5] and UNSAIL [8] for attack evaluation on unseen benchmarks during training. To make the training data for GALA, we then enhanced the locked benchmarks with extra power consumption labels taken from Synopsys Primetime PX that checks the switching activity to generate gate-level static, dynamic, and total power consumption. We also included the area overhead for each benchmark as a label.

A. GALA vs. OMLA Attack Analysis

Our testing results of OMLA and its GALA version are presented in Fig. 2 using XOR-based locking and single benchmark training. Using power and area information, the GALA version is able to achieve an average KPA of 97%, considerably higher than OMLA's original average KPA of 89% which solely considers the structure of the circuit. The added features allowed the model to capture more nuanced details of circuit behavior, contributing to higher accuracy and a more robust assessment of KPA. In addition, by just including these data, the KPR of OMLA enhances from on average 62% to above 85%. It means the GALA version of OMLA is able to report a key that not only has a high KPA (i.e., is closer to the correct key), but also has a higher KPR (i.e., lower output corruptibility) and thus can be profitable for the attacker.

B. GALA vs. LIPSTICK Attack Analysis

Fig. 3, in which X, M, L, S, and B refer to XOR-based, MUX-based, LUT-based, SAR-Lock, and BLE locking schemes, respectively, shows the KPA of the original LIPSTICK and its GALA version on datasets consisting of single and mixed locking on multiple benchmarks, following the same setup described in the LIPSTICK paper. We see that the inclusion of power and area data in GALA results in outperforming the LIPSTICK model in all evaluated cases, totaling to a KPA average of 91% compared to the original model's KPR average of 84% and a random key KPA average of 70%. Please note that here the average KPR is on a sophisticated training dataset and is not comparable with the KPA results in Section IV-A that include training only xor-based locking on single benchmarks individually.

We also evaluate the average KPR for 5, 10, and 50 random samples of the validation dataset, which consists of locking schemes seen during training in addition to the unseen locking mechanisms. In this regard, we find that the inclusion of new power and area features significantly improves model performance. Comparing LIPSTICK and its GALA versions, there is a KPR improvement of 14%, 17%, and 16% averaged over all single and mixed lock schemes for 5, 10, and 50 random samples, respectively.

C. Explainability Analysis

Fig. 4 presents the explainability analysis of the benchmark circuit "c3540" using the OMLA and GALA models. The underlying graph represents the circuit's structure, with nodes corresponding to logic gates and edges representing the interconnections. Overlaid on this graph are color-coded highlights that indicate the importance of specific components as determined by the models. Key nodes and edges are marked in red, orange, and yellow, where warmer tones signify a higher degree of influence on the model's predictions, while less relevant regions remain in grayscale. This visualization sheds light on the substructures most responsible for the model's decision-making, enabling targeted analysis and model refinement. A significant enhancement is observed when functionality and behavioral features are integrated into GALA. Using PGExplainer [37], GALA identifies meaningful substructures within the circuit, such as critical subcircuits that exhibit distinct and influential patterns of gate connectivity. For instance, subcircuits with high fan-out or fan-in connections were consistently highlighted, which aligns with areas of heightened attack vulnerability. In contrast, OMLA's structural-only approach demonstrates notable limitations. The explainability analysis for OMLA reveals isolated and fragmented subcircuits, which fail to offer actionable insights into the circuit's vulnerabilities. Without functional

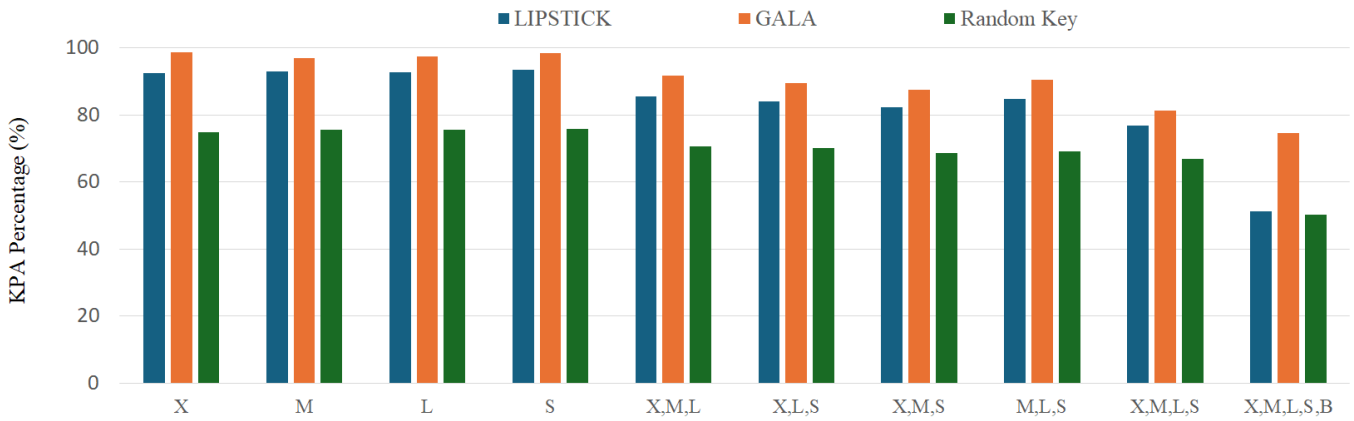
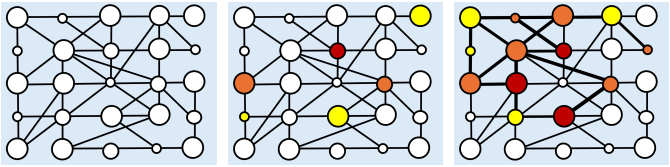


Fig. 3: GALA vs. LIPSTICK attacking results



(a) (b) (c)

Fig. 4: Explainability of the c3540 benchmark locked with XOR-based locking (a) Baseline (b) OMLA (c) GALA

or behavioral context, OMLA struggles to identify patterns indicative of successful attacks. This shortcoming underscores the importance of integrating richer circuit features into GNN-based attacks to enhance both their predictive power and interpretability. Lastly, while PGExplainer has been used in its standard form, the insights gained from its application are non-trivial. The integration of circuit functionality features into GALA results in explainability outputs that not only identify vulnerabilities but also propose pathways for circuit optimization. For example, areas highlighted as critical by GALA often correspond to gates with specific timing and power characteristics, which can be verified through traditional circuit analysis methods, thus providing an additional layer of validation for GALA's findings.

Please note that although we provide a single illustrative example in this paper—out of space constraints—our methodology applies similarly to other locked circuits: the trained GNN assigns higher attention weights to nodes or edges that have a strong correlation with correct key inference.

V. Discussion

We demonstrated that relying solely on the structural layout of a locked circuit is insufficient for a successful GNN-based attack. GALA sets itself apart from existing approaches by highlighting the importance of incorporating functional features such as key ER and behavioral metrics such as power consumption and area overhead to enhance both the accuracy and interpretability of these attacks.

To mitigate ML-based OL attacks like GALA, obfuscation techniques must evolve beyond simply hindering structural analysis. The focus should also shift toward concealing functional and behavioral characteristics. This could involve incorporating decoy components or introducing controlled noise within the circuit, making it significantly harder for attackers to extract meaningful information from the locked circuit. By masking structural, functional, and behavioral features, IC designers can create more resilient defenses against increasingly sophisticated ML-based OL attacks. In parallel, the development of more explainable models is essential. Explainability allows us to uncover the underlying patterns that the ML model leverages when guessing the key values. For designers, this provides invaluable insights into how specific elements of the chip architecture may contribute to weaknesses.

VI. Conclusion

This paper introduced GALA, a GNN-based attack for retrieving an approximate but meaningful key of existing logic locking methods by integrating functional and behavioral parameters into the GNN model. By incorporating power and area information into the GNN model, we enable a more comprehensive representation of the chip, leading to significantly improved performance in key prediction tasks. The success of this approach across multiple models demonstrates that functional and behavioral features are critical to the GNN model's ability to accurately interpret key values, indicating a generalizable trend. In addition, the integration of explainable GNNs into this analysis opens up new opportunities for hardware IP protection enhancement. By understanding the specific patterns and features that make logic-locked circuits susceptible to key recovery attacks, designers can better identify weak points in their designs. As GNNs become more prevalent in hardware security, explainability will be key in guiding future design and defense strategies.

Acknowledgment

This work is supported by the National Science Foundation under Award No. 2245247.

REFERENCES

[1] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1069-1074, 2008.

[2] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," In *IEEE Transactions on computers*, vol. 64, no. 2, pp. 410-424, 2013.

[3] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," In *IEEE design & Test of computers*, vol. 27, no. 1, pp. 66-75, 2010.

[4] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236-241, 2016.

[5] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199-207, 2019.

[6] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-SAT era by locking & obfuscation," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 13-18, 2020.

[7] R. Afsharmazayejani, H. Sayadi, and A. Rezaei, "Distributed logic encryption: Essential security requirements and low-overhead implementation," In *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 127-131, 2022.

[8] L. Alrahis, S. Patnaik, J. Knechtel, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "UNSAIL: Thwarting oracle-less machine learning attacks on logic locking," In *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2508-2523, 2021.

[9] M. R. Muttaki, S. Saha, H. M. Kamali, F. Rahman, M. Tehranipoor, and F. Farahmandi, "RTLock: IP protection using scan-aware logic locking at RTL," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1-6, 2023.

[10] J. Maynard and A. Rezaei, "DK lock: Dual key logic locking against oracle-guided attacks," In *International Symposium on Quality Electronic Design (ISQED)*, pp. 1-7, 2023.

[11] N. Limaye, A. Chowdhury, C. Pilato, M. Nabeel, O. Sinanoglu, S. Garg, and R. Karri, "Fortifying RTL locking against oracle-Less (untrusted foundry) and oracle-guided attacks," In *Design Automation Conference (DAC)*, pp. 91-96, 2021.

[12] K. Lopez and A. Rezaei, "Cute-Lock: Behavioral and structural multi-key logic locking using time base keys," In *Design, Automation & Test in Europe Conference (DATE)*, pp. 1-7, 2025.

[13] K. Lopez and A. Rezaei, "K-Gate Lock: Multi-key logic locking using input encoding against oracle-guided attacks," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 794–800, 2025.

[14] Y. Aghamohammadi and A. Rezaei, "CoLA: Convolutional neural network model for secure low overhead logic locking assignment," In *Great Lakes Symposium on VLSI 2023 (GLSVLSI)*, pp. 339–344, 2023.

[15] A. Rezaei and H. Zhou, "Sequential logic encryption against model checking attack," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1178-1181, 2021.

[16] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic locking algorithms," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137-143, 2015.

[17] H. Zhou, Y. Shen, and A. Rezaei, "Vulnerability and remedy of stripped function logic locking," In *Cryptology ePrint Archive*, paper 2019/139, 2019.

[18] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 95-100, 2017.

[19] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based attack on compound logic locking techniques," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1166-1171, 2021.

[20] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "SigAttack: New high-level SAT-based attack on logic encryptions," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 940-943, 2019.

[21] M. Zuzak, Y. Liu, I. McDaniel, and A. Srivastava, "A combined logical and physical attack on logic obfuscation," In *International Conference on Computer-Aided Design (ICCAD)*, Article 68, pp. 1-9, 2022.

[22] A. Rezaei, R. Afsharmazayejani, and J. Maynard, "Evaluating the security of eFPGA-based redaction algorithms," In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, article 154, pp. 1-7, 2022.

[23] P. -P. Chen, X. -M. Yang, Y. -T. Li, Y. -C. Chen, and C. -Y. Wang, "An approach to unlocking cyclic logic locking: LOOPLock 2.0," In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Article 155, 1–7, 2022.

[24] A. Saha, Akashdeep H. Banerjee, R. S. Chakraborty, and D. Mukhopadhyay, "ORACALL: An oracle-based attack on cellular automata guided logic locking," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* vol. 40, no. 12, pp. 2445-2454, 2021.

[25] Y. Hu, Y. Zhang, K. Yang, D. Chen, P. A. Beerel, and P. Nuzzo, "Fun-SAT: Functional corruptibility-guided SAT-Bbsed attack on sequential logic encryption," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 281-291, 2021.

[26] A. Rezaei, A. Hedayatipour, H. Sayadi, M. Aliasgari, and H. Zhou, "Global attack and remedy on IC-specific logic encryption," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 145-148, 2022.

[27] D. Sisejkovic, F. Merchant, L. M. Reimann, H. Srivastava, A. Hallawa, and R. Leupers, "Challenging the security of logic locking schemes in the era of deep learning: A neuroevolutionary approach," In *ACM Journal on Emerging Technologies in Computing Systems* vol. 17, no. 3, article 30, 2021.

[28] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 56-61, 2018.

[29] K. Shamsi and G. Zhao, "An oracle-less machine-learning attack against lookup-table-based logic locking," In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 133-137, 2022.

[30] L. Alrahis, S. Patnaik, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "GNNUnlock+: A systematic methodology for designing graph neural networks-based oracle-less unlocking schemes for provably secure logic locking," In *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1575-1592, 2021.

[31] L. Alrahis, S. Patnaik, M. A. Hanif, M. Shafique, and O. Sinanoglu, "UNTANGLE: Unlocking routing and logic obfuscation using graph neural networks-based link prediction," In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1-9, 2021.

[32] L. Alrahis, S. Patnaik, M. Shafique, and O. Sinanoglu, "OMLA: An oracle-less machine learning-based attack on logic locking," In *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1602-1606, 2021.

[33] Y. Aghamohammadi and A. Rezaei, "LIPSTICK: Corruptibility-Aware and Explainable Graph Neural Network-based Oracle-Less Attack on Logic Locking," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 606-611, 2024.

[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A comprehensive survey on graph neural networks," In *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, 2021.

[35] F. K. Dosilovic, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 0210-0215, 2018.

[36] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," In *European Conference on Computer Vision (ECCV)*, pp. 486-504, 2018.

[37] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," In *International Conference on Neural Information Processing Systems (NIPS)*, pp. 19620-19631, 2020.

[38] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782-5799, 2023.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," In *International Conference on Learning Representations (ICLR)*, 2019.

[40] A. Darjani, N. Kavand, S. Rai, and A. Kumar, "Discerning limitations of GNN-based attacks on logic locking," In *Design Automation Conference (DAC)*, pp. 1-6, 2023.

[41] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," In *IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 677-692, 1985.