GALA

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

Yeganeh Aghamohammadi (UCSB) • Henry Jin (UCSB) • Amin Rezaei (CSULB)

31st Asia and South Pacific Design Automation Conference
Jan. 19-22, 2026, Hong Kong Disneyland

Presenter: Amin Rezaei

Contents

Introduction ✨

Logic Locking in 30 Seconds

- Goal: Protect chip IP from piracy/overproduction in fabless manufacturing

- Add key-controlled gates:
  - Correct key → Correct output
  - Wrong key → Corrupted output

- Attackers target locked netlists to recover the correct key

Threat Model: OG vs OL

- Oracle-Guided (OG): Attacker has activated chip + locked netlist (SAT-based attacks) → See our prior work on defending against these attacks: https://aminrezaei.com/en/publications

- Oracle-Less (OL): Attacker only has locked netlist → More practical with less resources

➢ Our focus: OL attacks using Graph Neural Network (GNNs)

Why GNNs?

- A circuit is a graph:
  - Gates → Nodes
  - Nets → Edges

- GNNs learn structural patterns useful for key inference

- But structure alone may be insufficient and even misleading → We inject behavioral + functional features

Metrics: KPA vs KPR

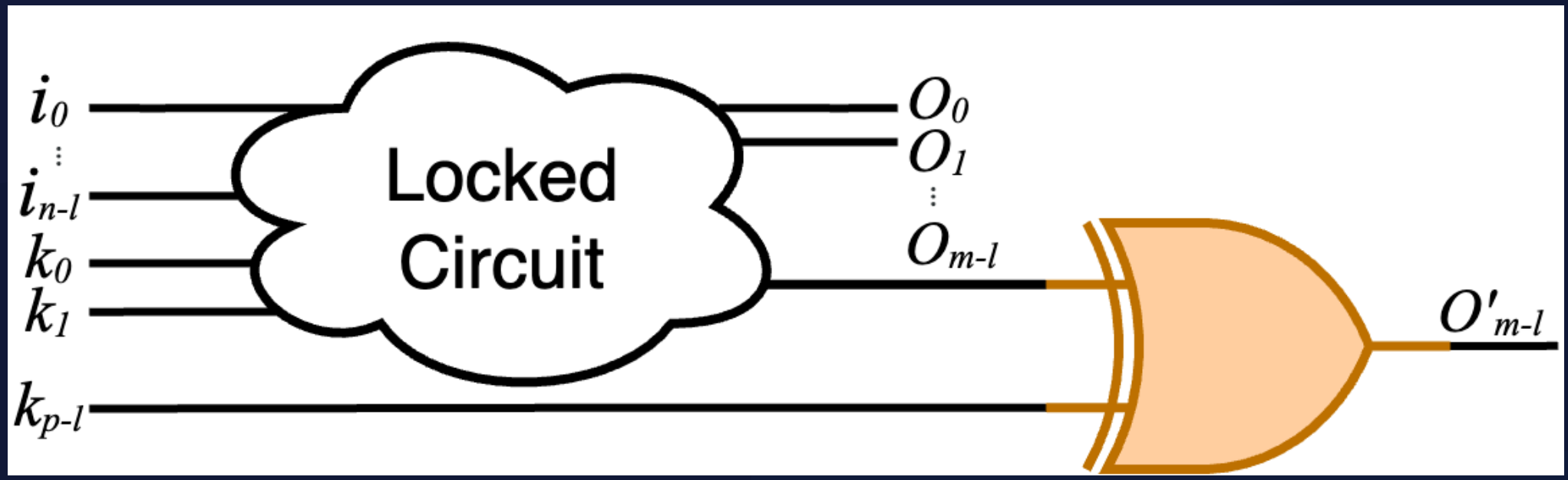
- KPA (Key Prediction Accuracy):
  - ➢ Per-bit correctness → Hamming-distance closeness
- KPR (Key Prediction Rate): Fraction of inputs where the reported key reproduces correct functionality → KPR directly measures how often your recovered key actually works
- Observation: High KPA can still yield unusable keys (low KPR)

How To Compute KPR in Practice

- Small Circuits: Exhaustive input enumeration
- Large Circuits: Random input sampling → Report averages (multiple trials) to reduce variance

Preliminaries ✨

Motivation: Close Key ≠ Good Key



A key can have high KPA but still cause high output corruptibility (i.e., low KPR)

Where GALA Fits (Related OL GNN Attacks)

- UNTANGLE[*]: Link-prediction style GNN attack
- OMLA[%]: Subgraph-level key-bit classification
- LIPSTICK[#]: Graph-level, corruptibility-aware + explainability
- GALA (This work): Inject behavioral (power/area) + emphasize usability (KPR) + explainable OL

[*] UNTANGLE - Alrahis et al., ICCAD 2021.
[%] OMLA - Alrahis et al., IEEE TCAS II, 2022.
[#] LIPSTICK - Aghamohammadi et al., ASP-DAC 2024.

Baseline: UNTANGLE[*]

- Uses GNN-based link prediction to recover obfuscated connectivity
- Represents an ML-based OL direction beyond pure structural heuristics
- GALA focuses on a gap: Behavior + functional usability

[*] UNTANGLE - Alrahis et al., ICCAD 2021.

Baseline: OMLA[%] (Subgraph-Level)

- Extract h-hop subgraph around each key gate
- Classify the corresponding key bit (0/1) from local structure
- Fast + localized, but can miss global/behavioral signals

[%] OMLA - Alrahis et al., IEEE TCAS II, 2022.

Baseline: LIPSTICK[#] (Graph-Level)

- Predicts the full key using a graph-level GNN
- Corruptibility-aware framing improves practical success beyond KPA alone
- Also supports interpretability → We extend explainability in GALA

[#] LIPSTICK - Aghamohammadi et al., ASP-DAC 2024.

Contributions ✨

What GALA Adds

- Functionality-Oriented Evaluation → Emphasize usability (KPR), not just bit closeness (KPA)
- Behavioral Features → Power consumption + area overhead as additional learning signal
- Two Attack Modes → Subgraph-level (OMLA-style) + graph-level (LIPSTICK-style)
- Explainable graph-level attack → Not just “a key”, but also “why this key”

GALA Feature Augmentation

- Structural Features: Topology (as in prior GNN OL attacks)
- Behavioral Features: Per-gate power (static/dynamic/total) + design area overhead
- Functional Emphasis: Evaluate and optimize for keys that preserve outputs (KPR)

Behavioral Signals: Where do We Get Them?

- Power: Gate-level power analysis (static + dynamic + total) via PrimeTime PX
- Area: Synthesized layout / overhead per benchmark
- Encode and normalize these as node attributes for the GNN

Two Modes: GALA-OMLA vs GALA-LIPSTICK

- GALA-OMLA: Add power/area to subgraph classifier
- GALA-LIPSTICK: Add power/area to graph-level predictor
- Same message-passing backbone class (GIN-style[&]) with richer node attributes

[&] GIN - Xu et al., ICLR 2019.

Explainability: PGExplainer[@]

- PGExplainer learns an edge mask to highlight the subgraph that drives predictions
- Makes analysis actionable: Locate vulnerable regions and failure modes

[@] PGExplainer - Luo et al., NeurIPS 2020.

Results ✨

Experimental Setup

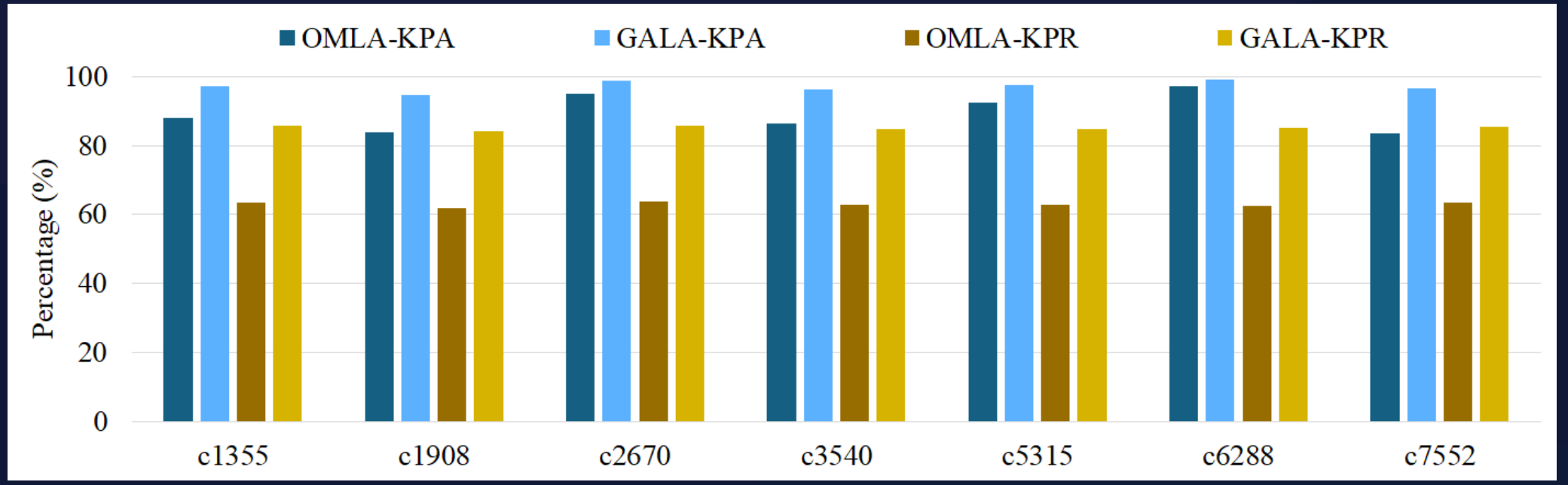
Bench.	Gates	Functionality
c1355	1503	32-bit single-error corrector
c1908	1289	16-bit single-error corrector and double-error detector
c2670	1262	12-bit arithmetic logic unit and controller
c3540	1403	8-bit arithmetic logic unit
c5315	1350	9-bit arithmetic logic unit
c6288	4703	16x16 multiplier
c7552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

ISCAS-85 benchmarks used in evaluation (key size = 64 bits)

Locking Schemes and Generalization

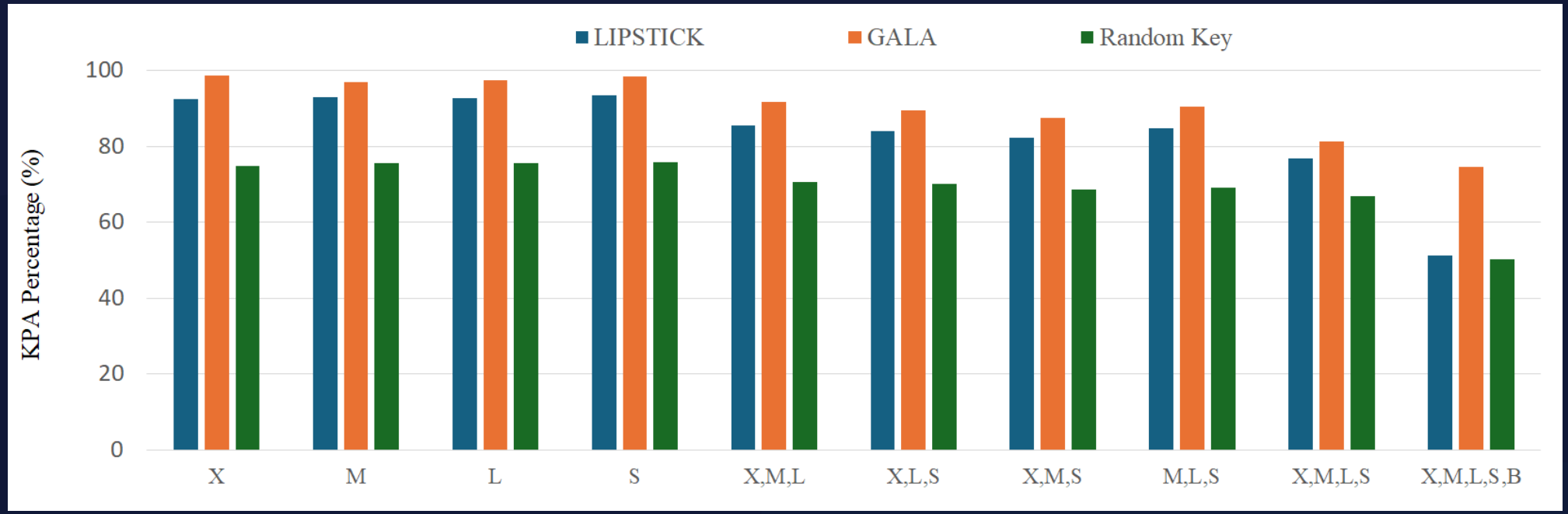
- Training/Eval Includes: XOR, MUX, LUT, SAR-Lock, BLE (64-bit keys)
- Generalization Test: Evaluate on unseen schemes (e.g., Anti-SAT, UNSAIL)
- Goal: Robustness across locking styles, not single-scheme overfitting

Results (Subgraph-Level): GALA vs OMLA



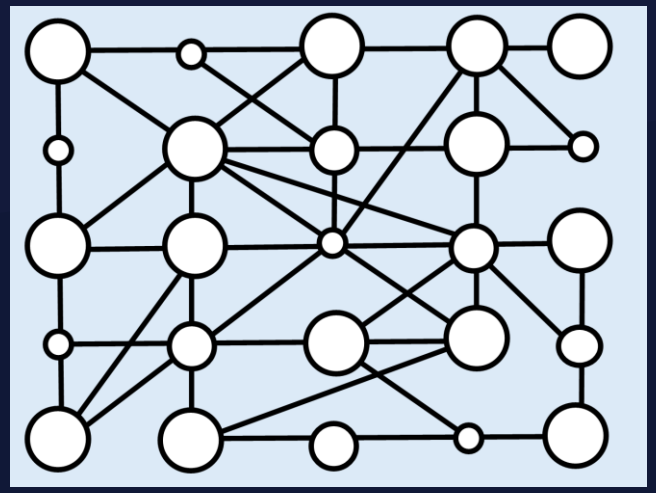
GALA vs. OMLA attack results

Results (Graph-Level): GALA vs LIPSTICK

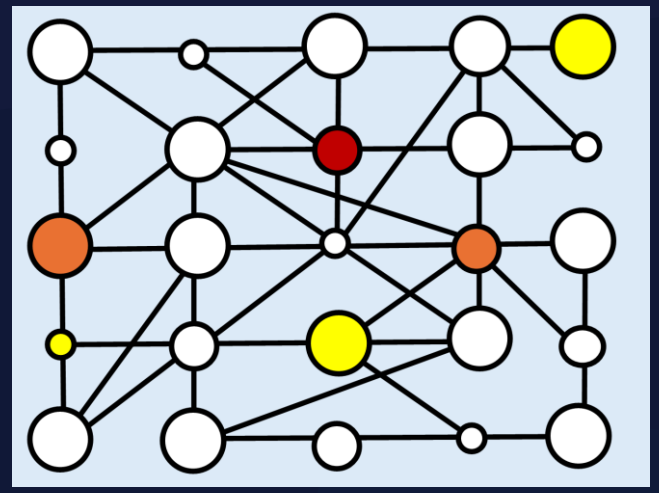


GALA vs. LIPSTICK attack results

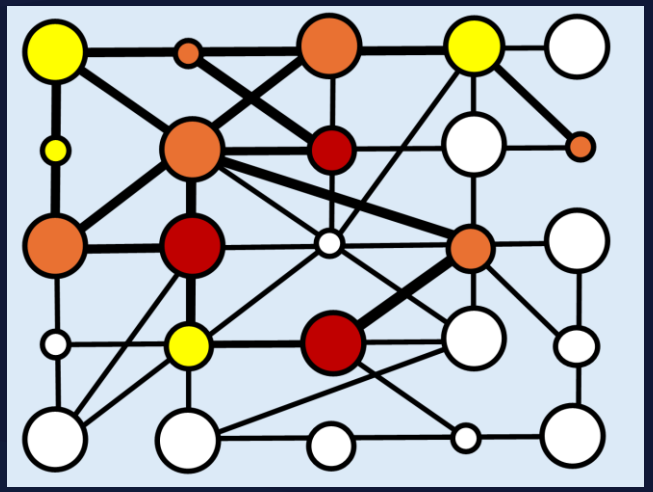
Explainability Result (Example)



(a) Baseline



(b) OMLA



(c) GALA

Explainability of the c3540 benchmark locked with XOR-based locking

Conclusion ✨

Take-Aways

- High KPA does not guarantee a usable key → Report KPR alongside KPA
- Behavioral features (power/area) measurably improve GNN-based OL attacks
- PGExplainer makes graph-level predictions interpretable and actionable
- Takeaway for defenders: Masking behavior may matter as much as masking structure

Acknowledgment & Repository

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

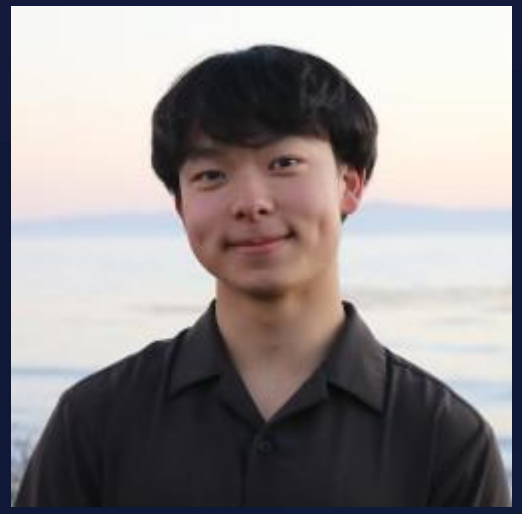
Award No. 2245247

- The source codes and created benchmarks are publicly available on our GitHub repository.

github.com/cars-lab-repo/gala

Thank You!

Yeganeh Aghamohammadi



Henry Jin

Amin Rezaei

github.com/cars-lab-repo/gala