DOI: 10.1587/elex.22.20250364
Received: June 19, 2025
Accepted: June 30, 2025
Publicized: July 9, 2025

IEICE Electronics Express, Vol.21, No.1, 1–6

LETTER

TrojanHound: Structure-Aware Subgraph Analysis for Hardware Trojan Detection in Gate-Level Designs

Xing Hu[1, 2], **Yang Zhang**[1, 2, a)], **Huan Guo**[1, 2], **Jiahe Shi**[1, 2], **Haowen Wang**[1, 2], **Zhenyu Zhao**[1, 2], **and Keqin Li**[3]

Abstract The integration of third-party IP cores into IC designs significantly increases the risk of stealthy hardware Trojans (HTs). Existing detection approaches, based on functional testing, logic verification, or machine learning, often focus on individual gates and overlook the structural patterns across Trojan circuits. Moreover, they lack diagnostic capabilities and typically require manual inspection. This work presents TrojanHound, a hierarchical detection framework that unifies gate-level analysis and circuit-level diagnosis. It leverages a graph neural network with adaptive neighborhood aggregation to identify suspicious nodes, merges related components into candidate subcircuits using subgraph fusion, and verifies HT presence through topology-aware analysis based on betweenness centrality. Experiments on 14 TrustHub benchmarks show that TrojanHound achieves a 95.64% true positive rate and a 97.44% F1-score, without false positives. By integrating structural learning with topological reasoning, this method enables accurate and automated HT detection in complex ICs.

Keywords: Hardware Trojans, gate-level netlist, graph neural networks, subgraph fusion, betweenness centrality

Classification: Integrated circuits

1. Introduction

The integration of third-party intellectual property (IP) cores into modern chip designs has become a cornerstone of efficient and cost-effective semiconductor development. Meanwhile, insufficiently secured electronic design automation (EDA) tools introduce stealthy attack vectors during critical design phases. However, this practice introduces significant security risks in the form of hardware Trojans (HTs), which can be embedded during the design phase to compromise system functionality or confidentiality.

Contemporary HT detection methodologies predominantly employ three technical paradigms: functional test pattern generation, logic-level signature analysis, and machine learning-based anomaly detection. However, our evaluation of TrustHub benchmarks reveals critical limitations in these approaches when confronting modern HT implementations. Foremost, single-gate analysis techniques—though computationally efficient—exhibit false positive rates due to their inability to distinguish malicious functionality from benign logic redundancy, particularly in power gating structures and error correction circuits. More critically, these methods fundamentally misapprehend HTs' structural nature as coordinated multi-gate circuits. Compounding these issues, current subgraph analysis frameworks lack systematic mapping between topological patterns and attack semantics. These cascading deficiencies underscore the imperative for a unified detection paradigm that synergistically combines granular gate-level feature extraction with macroscopic circuit-level topological reasoning, enabling both precise HT detection and diagnosis.

To address the limitations of existing methods, we propose TrojanHound, a betweenness-centric subgraph morphometry that offers significant innovations in HT detection and diagnosis. The key contributions of our work are as follows:

1) TrojanHound introduces a three-stage pipeline that seamlessly integrates gate-level analysis with circuit-level understanding. This approach bridges the granularity gap between individual gates and multi-gate HTs, enabling comprehensive detection and .

2) TrojanHound introduces Multi-Suspect Subgraph Fusion (MSSF), which clusters suspicious nodes into candidate HT components through dual functional-structural constraints. By leveraging HTs' intrinsic connectivity patterns, MSSF generates context-aware subgraphs that address detection blind spots in complex interconnect topologies inherent to conventional subgraph-based methods.

3) TrojanHound introduces a betweenness centrality metrics that aligns subgraph center with known HT structure characteristic. By integrating topological morphology analysis with node betweenness centrality metrics, this method precisely identifies triggers. Then graph-based visualization is employed to present the analyzed structures.

4) TrojanHound pioneers the integration of topological structural analysis with graph neural networks (GNNs), attaining a 97.4% F1-score metric while revealing signature "dumbbell-shaped" circuit configurations in HTs.

2. RELATED WORK

2.1 Functional Testing Methods

Functional testing detects HTs by analyzing circuit responses to specific input stimuli. The Unused Circuit Identification (UCI) method [1] identifies potentially infected regions by detecting logic never activated by any test pattern. Saha et al. [2] enhance test vector generation using a genetic algorithm to solve the Boolean satisfiability problem. Lyu et al. [3] improve rare trigger activation by generating test patterns based

[1] College of computer science and Technology, National University of Defense Technology, Changsha, China
[2] Key Laboratory of Advanced Microprocessor Chips and Systems, China
[3] State University of New York, USA
[a)] zhangyang@nudt.edu.cn

on repeated maximum clique sampling with SMT solvers. However, functional testing is limited by incomplete access to third-party IP cores and increasing design complexity. It also struggles to detect HTs that do not affect observable behavior, often requiring labor-intensive manual analysis.

2.2 Logical Analysis Methods

Testability metrics such as controllability and observability assess how easily circuit nodes can be set and observed, helping to flag low-testability nodes as potential HT targets. Probability-based methods extend this by analyzing signal switching activity under test vectors, exploiting HTs' low activation likelihood. The COTD framework [4] models testability thresholds for localization, while follow-up studies [5–9] improve sensitivity via refined thresholds and multi-dimensional correlation. Dynamic models based on switching probabilities [10, 11] further distinguish HT-infected from normal circuits. Hybrid methods by Huang et al. [12] and Su et al. [13] combine structural and behavioral features into high-dimensional representations, and Chen [14] introduces a multi-level framework incorporating formal verification. However, such logic analysis methods often yield high false positives due to similar low-activity patterns in benign circuits, requiring additional manual inspection.

2.3 Machine Learning Approaches

2.3.1 Traditional Machine Learning Approaches

Hasegawa et al. [15] pioneer systematic feature engineering, extracting 51 HT-related features from netlists. By applying feature importance analysis, they identified 11 optimal features and used a random forest classifier. Their subsequent work [16] introduce a multi-layer neural network architecture to end-to-end feature learning. Kurihara [17] validate the superiority of neural networks in complex pattern recognition tasks. Yamashita et al. [18] enhance adversarial robustness by selecting 24 HT-specific features from 76 candidates. Lu [19] propose an unsupervised clustering method based on information entropy and density-based clustering. Kok [20] and Sarihi [21] integrate testability metrics with structural characteristics, employing machine learning classifiers to discriminate between standard circuit gates and HT-infected components. Similarly, Karthikeyan [22] and Priyadharshini [23] leverage a fusion of controllability analysis and machine learning techniques to classify gates as either normal or anomalous.

2.3.2 Graph Neural Network (GNN) Techniques

GNNs excel in processing graph-structured netlist data. Muralidhar's GATE Net [24] employs supervised contrastive learning on signal path subgraphs (spanning gate-level nodes to inputs). TrojanSAINT [25] accelerates large-scale netlist analysis using GraphSAINT subgraph sampling. Hasegawa [26–28] develops a node-level detection framework leveraging GNNs and graph transformer architectures to analyze structural and behavioral anomalies in gate-level netlists, enabling precise localization of HT-compromised gates. This approach integrates graph transformer layers to capture long-range dependencies between circuit components, while multi-head attention mechanisms enhance anomaly pattern recognition across hierarchical netlist structures. The BGNN-HT model [29] introduces bidirectional structural context awareness with graph attention mechanisms. This approach effectively captures topological anomalies in HT-infected units.

2.3.3 Cross-Modal Fusion Methods

Emerging cross-modal techniques treat netlists as linguistic sequences: gates as "words" and interconnections as "syntax." GramsDet [30] and LMDet [31] utilize neural embeddings and language models, respectively.

While machine learning-based detection methods leverage learned patterns to identify HTs, they lack in-depth structural analysis of the malicious circuitry. In-depth structural/behavioral analysis of detected Trojan candidates streamlines collaborative analysis workflows.

3. Methodology

3.1 Attack Model

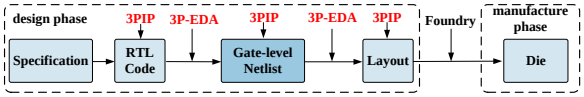

Fig. 1 Vulnerabilities in the semiconductor supply chain.

This study investigates vulnerabilities in the semiconductor supply chain (Fig. 1) that enable HT insertion through third-party intellectual property (3PIP) components, with particular emphasis on risks arising during integrated circuit (IC) design phases. The increasingly distributed and collaborative paradigm of modern chip development introduces critical security gaps that malicious actors can exploit to infiltrate design workflows. Specifically, unvetted 3PIP vendors may deliberately supply compromised cores containing concealed circuitry, while geographically dispersed design teams contracted for specialized tasks could leverage their privileged access to implant malicious logic. Beyond untrusted IP sources, vulnerabilities persist in technical workflows - insufficiently secured electronic design automation tools could surreptitiously insert Trojans during critical translation phases such as automated conversion of register-transfer-level designs to gate-level netlists. Unlike foundry-level attacks, these design-phase compromises enable more profound system penetration by allowing adversaries to strategically embed malicious payloads within mission-critical functional blocks prior to manufacturing. The analysis highlights how the semiconductor industry's growing dependence on external partnerships and automated toolchains, while economically imperative, inadvertently creates an expanded attack surface for sophisticated hardware exploits. This paper primarily concentrates on HT attack scenarios targeting gate-level netlist implementations.

3.2 Overall Framework

The proposed framework systematically addresses HT detection through a three-stage pipeline, as illustrated in Fig. 2. The framework initiates with suspect node screening leveraging GateGNN, a hybrid model combining graph neural networks (GNN) with gating mechanisms to prioritize nodes exhibiting abnormal signatures. This stage filters noise

from circuit graphs, generating candidate nodes for further analysis. Subsequently, the multi-suspect subgraph fusion (MSSF) module dynamically aggregates localized circuit regions centered on these candidates. The final subgraph analysis stage employs topology-aware detection techniques to identify HTs' characteristic dumbbell-shaped patterns.

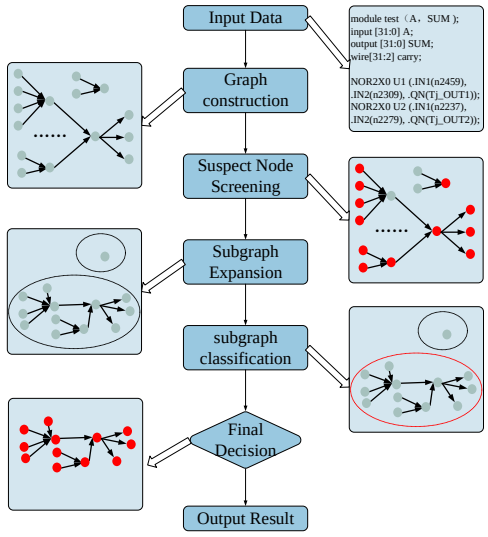

Fig. 2 Overall Framework.

3.3 Suspect Node Screening (GateGNN)

The GateGNN architecture comprises three specialized layers to achieve suspect node screening (shown in Fig. 3). 1) Embedding layer: Translates raw circuit graph nodes into feature vectors encoding functional properties (e.g., logic gate type). 2) GNN layers: Process node relationships via graph convolution mechanisms enhanced with hardware-aware constraints. Each layer iteratively refines node embeddings by adaptively aggregating neighborhood features. 3) Linear classification layer: Classify nodes as normal nodes or HT nodes. The basic architecture of the graph attention network is detailed in [32]. Through graph-based detection, this stage produces refined candidate nodes for further analysis.

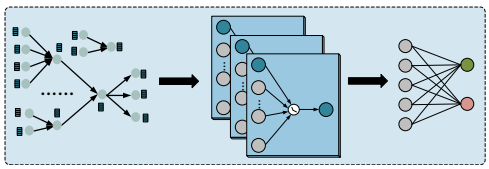

Fig. 3 The GateGNN architecture.

3.4 Multi-Suspect Subgraph Fusion (MSSF)

Following suspect node screening via GateGNN, the MSSF module dynamically constructs context-aware subgraphs by leveraging the intrinsic connectivity of Trojan structures, which is shown in Algorithm 1.

Algorithm 1 Multi-Suspect Subgraph Fusion with k-Hop Connectivity

Require:
$\mathcal{V}_s \subseteq V$: Set of suspect nodes
$G = (V, E)$: Circuit graph
k: Maximum expansion hops (default: 3, chosen based on Trojan structure and synthesis variations)
Ensure:
$\mathcal{G}_f$: Set of fused subgraphs
Initialize candidate queue $\mathcal{Q} \leftarrow \mathcal{V}_s$
Initialize fused subgraphs $\mathcal{G}_f \leftarrow \emptyset$
Initialize visited nodes $\mathcal{V}_v \leftarrow \emptyset$
while $\mathcal{Q} \neq \emptyset$ **do**
  $v_i \leftarrow \mathcal{Q}$.dequeue()
  if $v_i \notin \mathcal{V}_v$ **then**
    $\mathcal{V}_v \leftarrow \mathcal{V}_v \cup \{v_i\}$
    $\mathcal{N}_k(v_i) \leftarrow$ kHopNeighborhood(v_i, k)
    for all $v_j \in \mathcal{V}_s \setminus \{v_i\}$ **do**
      $\mathcal{B}_{ij} \leftarrow \mathcal{N}_k(v_i) \cap \mathcal{N}_k(v_j) \setminus \mathcal{V}_s$
      if $\mathcal{B}_{ij} \neq \emptyset$ **then**
        Construct subgraph $G_{ij} \leftarrow$ InducedSubgraph$(\{v_i, v_j\} \cup \mathcal{B}_{ij})$
        $\mathcal{G}_f \leftarrow \mathcal{G}_f \cup \{G_{ij}\}$
        for all $b \in \mathcal{B}_{ij}$ **do**
          if $b \notin \mathcal{Q} \wedge b \notin \mathcal{V}_v$ **then**
            $\mathcal{Q}$.enqueue(b)
          end if
        end for
      end if
    end for
  end if
end while
Merge overlapping subgraphs:
$\mathcal{G}_f \leftarrow$ MergeConnectedComponents$(\mathcal{G}_f)$

The MSSF algorithm dynamically connects suspect nodes through their k-hop neighborhoods to identify potential Trojan trigger paths. Starting with an initial set of suspect nodes, the algorithm iteratively: 1) discovers bridge nodes connecting pairs of suspects within k hops, 2) constructs candidate subgraphs containing these suspects and bridge nodes, and 3) expands the search by adding bridge nodes to the processing queue. A key innovation is the dual-phase connectivity analysis that first establishes local k-hop neighborhoods around individual suspects, then detects inter-suspect connections through shared bridge nodes. The final merging phase combines overlapping subgraphs using connected component analysis, effectively reconstructing complete Trojan topologies from fragmented initial detections. Thus, the MSSF aggregates suspicious nodes into context-aware subgraphs through dual functional-structural constraints.

The hyperparameter k determines the maximum neighborhood distance allowed for merging suspect nodes. In principle, Trojan nodes are typically tightly coupled and located within 1-hop proximity. However, due to structural transformations introduced by synthesis or obfuscation, such proximity may not always hold. Therefore, we conservatively set $k = 3$ to ensure robust subgraph reconstruction. This choice balances the risk of missing interrelated Trojan nodes (if k is too small) and incorporating excessive benign logic (if k is too large), and has been empirically validated across multiple benchmarks.

3.5 HT Diagnosis: Subgraph Analysis Based on Betweenness Centrality

In IC security, subgraph analysis establishes itself as a structural feature-based methodology for HT diagnosis. This approach enables trigger node localization and HT identification through mining anomalous topological patterns in netlists. The foundational principle stems from the observation that HTs – designed for covert activation and functional manipulation – inherently form distinctive subgraph structures. Quantitative pattern matching of these structures provides precise malicious component localization.

The HT triggering mechanism requires control signal propagation across multiple functional modules, generating unique betweenness centrality characteristics. To identify potential trigger nodes, we compute the betweenness centrality (CB) of each node within a suspicious subgraph and select the node with the highest CB value as the trigger candidate. This strategy reflects the assumption that the core trigger node serves as a central interconnection point in the HT's control path, typically exhibiting the most prominent structural influence in the subgraph.

Through decomposition of the global circuit graph into suspicious subgraphs using the aforementioned methods, we compute node-level betweenness centrality to quantify hub status:

$$CB(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

where σ_{st} denotes the total number of shortest paths between nodes s and t, with $\sigma_{st}(v)$ specifically counting paths traversing node v. The node with the maximum CB value in each subgraph is selected for further analysis.

HTs often adopt dumbbell-shaped topologies for covert activation and payload delivery. This structure typically includes three parts: a frontend trigger cluster with multi-input logic, a backend payload cluster linked to critical units, and a narrow trigger channel with minimal-node paths. Morphological analysis is then used to detect such distinctive connection patterns.

By combining topological morphology with node betweenness centrality, this method accurately identifies trigger nodes. The resulting structures are visualized graphically, enabling clear interpretation of concealed HT patterns while minimizing manual analysis effort through reduced global noise interference.

4. Experimental Analysis

4.1 Experimental Setup and Metrics

The dataset is obtained from TrustHub [33]. Within the GateGNN framework, we adopt a leave-one-out method, ensuring that the test data remains unseen during the training phase. The detection performance is quantified using standard statistical metrics derived from confusion matrix elements: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Key evaluation measures include: True Positive Rate ($TPR/Recall$) = $TP/(TP+FN)$, False Positive Rate (FPR)=$FP/(FP+TN)$, $Precision$=$TP/(TP+FP)$, $F1$=$2*Precision*Recall/(Precision+Recall)$.

4.2 Results Comparison

To address the precision limitations and interpretability constraints of existing learning-based HT detection methods ([26–28], collectively referred to as GateGNN), our framework incorporates graph fusion and subgraph-level analysis to enhance structural reasoning. Recognizing the lack of standardized datasets and the considerable variation in experimental settings across prior works—including differences in dataset construction, model depth, and hyperparameters—we adopt an internal ablation-style evaluation.

Specifically, we perform a controlled comparison between the full TrojanHound framework and its base GateGNN component, under identical training configurations, data splits, and optimization settings. This design isolates the incremental contributions of the subgraph fusion and topological diagnosis modules. Although external benchmarking against third-party methods is limited by inconsistent public benchmarks and implementation details, our evaluation offers a fair and reproducible way to quantify the added value of our structural extensions.

The comparative results on 14 TrustHub datasets are shown in Fig. 4, Fig. 5, and Fig. 6, demonstrating consistent improvements in true positive rate and F1-score, alongside a reduction in false positives. These improvements validate the effectiveness of incorporating structural topology into node-level detection for hardware Trojan identification.

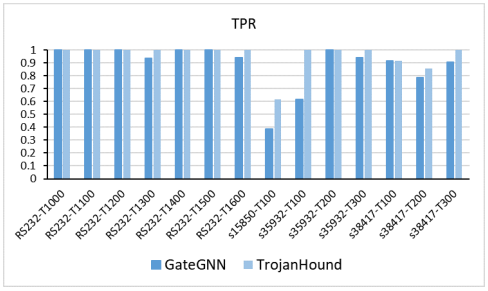

Fig. 4 Comparison of TPR between GateGNN and TrojanHound.

The comparative TPR evaluation demonstrates significant improvements in Trojan detection accuracy through the proposed TrojanHound framework. As shown in the benchmark results, TrojanHound achieves perfect recall (TPR=1.0) across 11 of 14 test cases, outperforming the baseline GateGNN approach that attains maximum recall in only 6 instances. Particularly noteworthy are the critical scenarios where TrojanHound resolves detection failures of the pure machine learning method: 1) In s15850-T100, recall improves by 60% (0.615 vs 0.385); 2) For s35932-T100, detection capability reaches full coverage (1.0 vs 0.615); 3) Complex Trojans like s38417-T300 show complete mitigation of previous 9.3% false negatives.

The comparative FPR evaluation reveals critical insights into the operational characteristics of TrojanHound versus the baseline GateGNN approach. Notably, both methodologies demonstrate perfect false positive rate (FPR) suppression (0.00%) across 10 of 13 test cases, including all RS232 and s15850 circuit variants. However, sub-

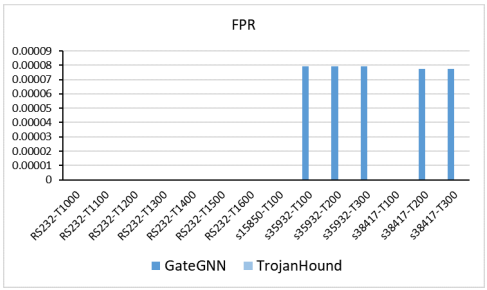

Fig. 5 Comparison of FPR between GateGNN and TrojanHound.

tle differences emerge in complex scenarios: TrojanHound achieves absolute FPR elimination (0.00%) for s35932-T300 and s38417 series, while GateGNN exhibits residual false alarms (0.0079% for s35932 variants, 0.0078% for s38417-T200/300). This 100% false positive mitigation in challenging cases confirms the enhanced specificity of our graph fusion mechanism. The persistent 0.00% FPR suggests robust discrimination between legitimate circuit patterns and sophisticated HTs, addressing a critical limitation in ML-based detection systems.

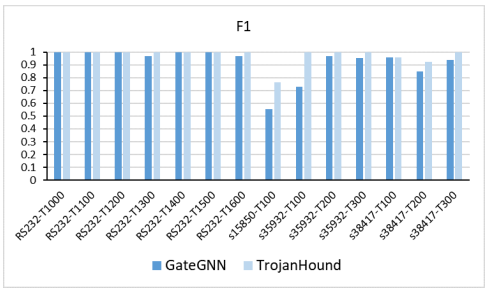

Fig. 6 Comparison of F1 between GateGNN and TrojanHound.

The comprehensive F1 evaluation demonstrates substantial performance enhancements achieved by the TrojanHound framework compared to the baseline GateGNN methodology. As evidenced by the benchmark results, TrojanHound attains perfect F1 (1.00) in 11 of 14 test cases, including all RS232 variants and critical s35932 configurations (T100-T300), while GateGNN achieves maximum scores in only 5 instances.

Table I Average TPR, FPR, and F1 Score Comparison: GateGNN vs. TrojanHound.

Method	TPR/Recall	FPR	F1
GateGNN [26–28]	88.7241%	0.0028%	91.9947%
TrojanHound	95.6371%	0.00%	97.4393%

The comparative evaluation (shown in Table I) demonstrates measurable improvements in HT detection through the proposed TrojanHound framework. With a statistically significant 7.8% increase in true positive rate (95.64% vs 88.72%) and complete suppression of false positives (0.00% FPR vs 0.0028%), the methodology achieves enhanced detection reliability while maintaining operational safety—critical requirements for industrial hardware verification. The 5.4% elevation in F1-score (97.44% vs 91.99%) reflects improved precision-recall balance compared to conventional graph neural network approaches. These quantitative improvements validate the effectiveness of integrating structural graph analysis with machine learning paradigms for hardware security applications, suggesting practical value for modern IC verification workflows.

4.3 Case Study

The s35932-T100 circuit analysis demonstrates the multi-stage detection capability of our framework. Initial screening with GateGNN identified 9 candidate nodes exhibiting anomalous behavior patterns. Subsequent multi-scale subgraph fusion (MSSF) refined these candidates into two structurally distinct subgraphs ($G1$ and $G2$).

Fig. 7 presents a heatmap visualization of betweenness centrality distribution within the Trojan-affected subgraph $G2$ of circuit s35932-T100. The thermal gradient (darkening with centrality increase) reveals a distinctive bipartite structure - a dumbbell-shaped topology connecting two high-centrality clusters through critical bridge nodes. This structural pattern aligns with established HT characteristics: 1) Dense interconnectivity between trigger and payload modules, 2) Strategic placement of high-centrality control nodes (e.g., $Tj\_Trigger$, and 3) Concentrated data pathways facilitating covert signal propagation. Contrastingly, subgraph $G1$ manifests as an isolated node lacking structural complexity essential for malicious functionality. Our diagnostic framework leverages these topological fingerprints to achieve higher classification accuracy, successfully distinguishing all Trojan-embedded nodes in $G2$ (true positives) from benign components in $G1$ (true negatives). The spatial correlation between centrality hotspots and known Trojan modules validates our graph-theoretic detection criteria for security-critical circuit analysis.

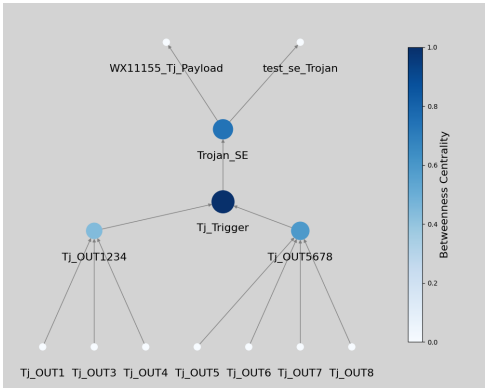

Fig. 7 Visualization of Trojan Gate Relationships

5. Conclusion

This paper introduces TrojanHound, a hierarchical and structure-aware framework for detecting HTs in gate-level netlists. By leveraging adaptive graph neural networks, subgraph fusion, and topological analysis, the approach captures both local and global circuit features for accurate and automated Trojan identification. Extensive evaluations on TrustHub benchmarks demonstrate superior detection per-

formance and zero false positives. The results validate the effectiveness of integrating graph learning with structural reasoning in advancing hardware security verification. Future work will focus on integrating the diagnostic process into the neural network framework in an adaptive manner, jointly optimizing key hyperparameters (e.g., k-hop expansion and betweenness centrality threshold) during training, to enable end-to-end Trojan detection and localization with improved automation and interpretability, so as to adapt to a wider variety of HT structures.

References

[1] Hicks M, *et al.*: "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," IEEE symposium on security and privacy. (2010) 2010 159 (DOI: 10.1109/SP.2010.18).

[2] Saha S, *et al.*: "Improved test pattern generation for HT detection using genetic algorithm and boolean satisfiability," Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop. Springer Berlin Heidelberg. (2015) 577 (DOI: 10.1007/978-3-662-48324-4_29).

[3] Lyu Y, *et al.*: "Scalable activation of rare triggers in HTs by repeated maximal clique sampling," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **40** (2020) 1287 (DOI: 10.1109/TCAD.2020.3019984).

[4] Salmani H: "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," IEEE Transactions on Information Forensics and Security. **12** (2016) (DOI: 10.1109/TIFS.2016.2613842).

[5] Kok C H, *et al.*: "Classification of Trojan nets based on SCOAP values using supervised learning," IEEE international symposium on circuits and systems (ISCAS). (2019) 1 (DOI: 10.1109/ISCAS.2019.8702462).

[6] Salmani H: "Gradual-N-Justification (GNJ) to reduce false-positive hardware Trojan detection in gate-level Netlist," IEEE Transactions on Very Large Scale Integration (VLSI) Systems. **30** (2022) 515 (DOI: 10.1109/TVLSI.2022.3143349).

[7] Salmani H: "The improved cotd technique for hardware trojan detection in gate-level netlist," Proceedings of the Great Lakes Symposium on VLSI 2022. (2022) 449.

[8] Tebyanian M, *et al.*: "SC-COTD: Hardware trojan detection based on sequential/combinational testability features using ensemble classifier," Journal of Electronic Testing. **37** (2021) 473 (DOI: 10.1007/s10836-021-05960-2).

[9] Lo P Y, *et al.*: "Semi-supervised trojan nets classification using anomaly detection based on SCOAP features," IEEE International Symposium on Circuits and Systems (ISCAS). (2022) 2423 (DOI: 10.1109/ISCAS48785.2022.9937236).

[10] Zou M, *et al.*: "Potential trigger detection for hardware trojans," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **37** (2017) 1384 (DOI: 10.1109/TCAD.2017.2753201).

[11] Mehta U, *et al.*: "Transition probability-based detection of hardware trojan in digital circuits," Proceedings of Sixth International Congress on Information and Communication Technology. (2021) 619 (DOI: 10.1007/978-981-16-2377-6_57).

[12] Huang K, *et al.*: "Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection," IEEE Transactions on Information Forensics and Security. **15** (2019) 3387 (DOI: 10.1109/TIFS.2019.2946044).

[13] Su Y, *et al.*: "A stealthy hardware trojan design and corresponding detection method," 2021 IEEE International Symposium on Circuits and Systems (ISCAS). (2021) 1 (DOI: 10.1109/ISCAS51556.2021.9401770).

[14] Chen X, *et al.*: "Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **37** (2017) 1370 (DOI: 10.1109/TCAD.2017.2748021).

[15] Hasegawa K, *et al.*: "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," IEEE International Symposium on Circuits and Systems (ISCAS). (2017) 1 (DOI: 10.1109/ISCAS.2017.8050827).

[16] Hasegawa K, *et al.*: "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS). (2017) 227 (DOI: 10.1109/ISCAS.2017.8050827).

[17] Kurihara T, *et al.*: "Evaluation on hardware-Trojan detection at gate-level IP cores utilizing machine learning methods," IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS). (2020) 1 (DOI: 10.1109/IOLTS50870.2020.9159740).

[18] Yamashita K, *et al.*: "Effective Hardware-Trojan Feature Extraction Against Adversarial Attacks at Gate-Level Netlists," IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS). (2022) 1 (DOI: 10.1109/IOLTS56730.2022.9897557).

[19] Lu R, *et al.*: "HTDet: A clustering method using information entropy for hardware Trojan detection," Tsinghua Science and Technology. **26** (2020) 48 (DOI: 10.26599/TST.2019.9010047).

[20] Kok C H, *et al.*: "Net classification based on testability and netlist structural features for hardware Trojan detection," IEEE 28th Asian Test Symposium (ATS). (2019) 105 (DOI: 10.26599/TST.2019.9010047).

[21] Sarihi A, *et al.*: " Multi-criteria hardware Trojan detection: A reinforcement learning approach," IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS). (2023) 1093 (DOI: 10.1109/MWSCAS57524.2023.10406091).

[22] K. S, *et al.*: "Hardware Trojan Detection using Unsupervised Machine Learning Algorithms in the Gate-level Netlist," IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). (2024) 1 (DOI: 10.1109/CONECCT62155.2024.10677111).

[23] Priyadharshini M, *et al.*: "A hardware trojan detection method for gate-level netlists employing the CAMELOT measure," 7th International Conference on Devices, Circuits and Systems (ICDCS). (2024) 183 (DOI: 10.1109/ICDCS59278.2024.10560972).

[24] Muralidhar N, *et al.*: "Contrastive graph convolutional networks for hardware Trojan detection in third party IP cores," IEEE International Symposium on Hardware Oriented Security and Trust (HOST). (2021) 181 (DOI: 10.1109/HOST49136.2021.9702276).

[25] Lashen H, *et al.*: "TrojanSAINT: Gate-level netlist sampling-based inductive learning for hardware Trojan detection," IEEE International Symposium on Circuits and Systems (ISCAS). (2023) 1 (DOI: 10.1109/ISCAS46773.2023.10181403).

[26] Hasegawa K, *et al.*: "Node-wise hardware trojan detection based on graph learning," IEEE Transactions on Computers. **74** (2023) 749 (DOI: 10.1109/TC.2023.3280134).

[27] Imangholi A, *et al.*: "FAST-GO: Fast, Accurate, and Scalable Hardware Trojan Detection using Graph Convolutional Networks," 25th International Symposium on Quality Electronic Design (ISQED). (2024) 1 (DOI: 10.1109/ISQED60706.2024.10528759).

[28] Chen M, *et al.*: "TrojanFormer: Resource-Efficient Hardware Trojan Detection Using Graph Transformer Network," 7th International Conference on Electronics Technology (ICET). (2024) 165 (DOI: 10.1109/ICET61945.2024.106727629).

[29] Zhan P, *et al.*: " Bgnn-ht: Bidirectional graph neural network for hardware trojan cells detection at gate level," IEEE International Symposium on Circuits and Systems (ISCAS). (2023) 1 (DOI: 10.1109/ISCAS46773.2023.10181569).

[30] Lu R, *et al.*: "GramsDet: Hardware Trojan detection based on recurrent neural network," IEEE 28th Asian Test Symposium (ATS). (2019) 111 (DOI: 10.1109/ATS47505.2019.00021).

[31] Shen H, *et al.*: "Lmdet: A "naturalness" statistical method for hardware trojan detection," IEEE Transactions on Very Large Scale Integration (VLSI) Systems. **26** (2017) 720 (DOI: 10.1109/TVLSI.2017.2781423).

[32] Veličković P, *et al.*: "Graph attention networks," arXiv preprint arXiv:1710.10903. (2017)

[33] B. Shakya, *et al.*, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits," Journal of Hardware and Systems Security. (2017) (DOI: 10.1007/s41635-017-0001-6).