Computers & Security 163 (2026) 104816

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full Length Article

GLRA: Graph-based leakage risk assessment via minimal transmission cost path analysis

Xing Hu [a,b], Yang Zhang [a,*], Sheng Liu [a], Xiaowen Chen [a], Yaohua Wang [a], Shaoqing Li [a], Zhenyu Zhao [a], Keqin Li [c]

[a] College of Computer Science and Technology, National University of Defense Technology, China
[b] School of Physics and Electronic Science, Changsha University of Science and Technology, China
[c] State University of New York, USA

ARTICLE INFO

Keywords:
Hardware security
Information leakage
Transmission cost
Risk assessment
Path analysis

ABSTRACT

As integrated circuits are increasingly deployed in security-critical applications, assessing the risk of information leakage introduced during the design phase has become a key challenge. Logic-level structures may inadvertently enable sensitive data to propagate to externally observable points, posing serious security risks. Although anomaly-based techniques such as taint tracking and machine learning have been developed to detect or mitigate leakage threats, the absence of a unified and quantitative metric for evaluating leakage risk remains a major limitation. Without such a metric, existing methods can neither effectively identify real threats nor compare the effectiveness of protection strategies in a principled manner, leading to limited reliability and comparability in hardware security analysis.

To overcome these challenges, we propose GLRA, a graph-based methodology for leakage risk assessment via minimal transmission cost path analysis. Departing from the traditional "path existence" criterion used in anomaly label-based taint tracking, GLRA quantifies leakage risk by evaluating the difficulty of information propagation. A central premise of GLRA is that the transmission cost-defined as the effort required to propagate signals from sensitive sources to observable outputs-is inversely correlated with leakage likelihood: lower costs imply higher risks. Accordingly, we define controllability-based transmission cost metrics for basic logical units such as AND, OR, NOT, and DFF, which quantify the propagation effort imposed by each logic unit. By modeling the circuit as an edge-weighted graph where edges are annotated with the aforementioned transmission cost values, GLRA identifies the minimal path from sensitive sources to potential leakage points. In addition, to accurately quantify the risk of leakage, GLRA establishes a formulaic correlation between the transmission cost and the design's overall risk of information leakage. Experiments on cryptographic cores, debug infrastructure, and non-cryptographic logic demonstrate that GLRA accurately quantifies maximum-risk leakage paths, achieving a 18.75% improvement in detection precision over traditional anomaly-based approaches. GLRA correctly determines the presence or absence of leakage risks across all 16 evaluated benchmarks. Furthermore, it supports comparative analysis of leakage mitigation strategies across diverse hardware designs, providing quantitative insights into the effectiveness of protection mechanisms.

1. Introduction

In the field of Integrated Circuits (ICs), hardware design has traditionally focused on functionality, performance, and cost. However, hardware security has often been overlooked. As ICs are now widely used in critical sectors such as autonomous vehicles, implantable medical devices, and military communication networks, security risks have become a paramount concern and a non-negligible factor in the design of modern ICs.

Among various hardware security threats, information leakage has become a particularly critical concern. Unlike functional attacks that disrupt system operations, leakage-based threats aim to exfiltrate sensitive data while preserving the circuit's correct outputs under standard operating or test conditions, thus evading conventional detection mechanisms. Leakage risk refers to the likelihood that confidential information within a hardware design is propagated to unintended outputs or observation points, potentially leading to undesired exposure of sensitive data. This type of leakage can occur through various covert

* Corresponding author.
E-mail addresses: huxing@nudt.edu.cn (X. Hu), zhangyang@nudt.edu.cn (Y. Zhang), liusheng83@nudt.edu.cn (S. Liu), xiaowenc@kth.se (X. Chen), nudtyh@foxmail.com (Y. Wang), sqli@163.com (S. Li), zyzhao@nudt.edu.cn (Z. Zhao), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.cose.2025.104816
Received 12 June 2025; Received in revised form 5 August 2025; Accepted 26 December 2025
Available online 29 December 2025

channels, including hardware Trojans (HTs), malicious backdoors, or even poorly isolated data paths. As ICs are increasingly deployed in sensitive and high-assurance domains, failure to mitigate information leakage risks can lead to severe security breaches, jeopardizing not only individual devices but also the integrity of broader computing infrastructures.

To mitigate the risk of information leakage, both detection and defense methods have been developed. Existing detection techniques-including information flow tracking (Witharana et al., 2023; Guo et al., 2019), machine learning (Yasaei et al., 2021; Kande et al., 2023), and side-channel analysis (Atieh and Shahram, 2017; Chen et al., 2023)-have primarily adopted anomaly-based strategies, attempting to identify suspicious behavior by flagging deviations from expected circuit operation. However, these methods lack a unified, quantitative measure of leakage severity, making it difficult to interpret whether the observed anomalies truly indicate security-critical threats. For example, consider the AES-T1200 (available on Trust-Hub Salmani et al. (2013)), an HT that causes sensitive data exposure due to abnormal data flow. This Trojan activates when a 128-bit counter, starting at 0 and incrementing by 1 with each cycle, reaches its maximum value. This triggers a flaw in the encryption process, resulting in the unauthorized leakage of encryption keys. However, at a 1 GHz CPU frequency, a 128-bit counter would take approximately 1.080×10^{22} years to reach its maximum value, making it highly improbable within the typical operational lifespan of the hardware. As a result, this type of anomaly poses no real threat and is essentially risk-free. Therefore, anomaly detection alone is insufficient; it is equally important to accurately assess the actual risk of information leakage.

In terms of defense, a variety of security enhancement measures can be applied to counteract the security vulnerabilities of information leakage (Ren et al., 2018; Lee et al., 2022b). Using the Joint Test Action Group (JTAG) interface as an example, JTAG is a debug interface that provides access to the internal data of integrated circuits, introducing the risk of information leakage. To mitigate this risk, several approaches can be employed to strengthen its security. Techniques such as password protection and encryption enhance JTAG security by ensuring that only authorized individuals can access the data. However, the effectiveness of these defenses is often assessed qualitatively and lacks consistent quantitative evaluation. Their actual protection strength can vary significantly depending on the attack sophistication and application environment. Without a unified metric, it remains challenging to objectively compare different strategies or select appropriate protections for specific security requirements. This variability underscores the need for a quantitative evaluation to systematically assess leakage risks and defense effectiveness across diverse hardware contexts.

So accurately measuring the risk of information leakage is crucial for assessing the risks posed by hardware vulnerabilities, both in detection and defense methods. Various methods have been developed to measure and evaluate hardware vulnerability. Existing methods related to hardware security assessment, such as Security Verification (SV), COTD, TVF, and SecMiner (Farzana et al., 2019; Salmani, 2017; Kok et al., 2019; Zhang et al., 2020; Salmani, 2022; Cruz et al.; Hu et al., 2024; Ayalasomayajula et al., 2024a) attempt to evaluate vulnerabilities. These methods either focus on functional correctness through assertion-based checking, evaluate the controllability and observability of trigger conditions for HTs, or analyze signal propagation patterns at RTL level. However, none of these approaches are specifically designed to quantify risk of information leakage at gate level.

Thus, there is an urgent need for a systematic and quantifiable approach to evaluate gate-level information leakage risks. Our key insight is that the ease with which sensitive signals propagate from internal sources to observable outputs reflects the likelihood of unintended leakage. In other words, the more effortlessly a sensitive signal can traverse the circuit to reach an external node, the higher the potential risk of information disclosure. Based on this insight, we propose a novel quantification framework to assess leakage risks. The key contributions are as follows:

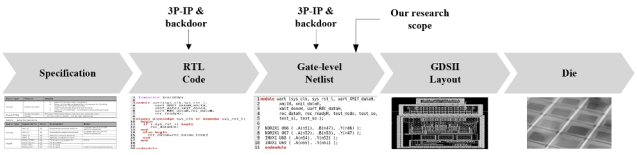

Fig. 1. IC design flow with potential security threats.

1 We propose a controllability-based transmission cost model for logic units such as AND, OR, and DFF to quantify the efforts required to transmit signals through them. These definitions provide metrics to assess the difficulty of signal propagation within these units.
2 Based on the transmission costs for logic units, we adopt an edge-weighted graph to quantify the risks for each leakage path. This method transforms hardware designs into graph structures, with edges assigned unit-specific transmission costs to quantify leakage paths.
3 We establish a formulaic correlation between transmission cost and information leakage risk at gate level. In contrast to conventional qualitative methods relying on anomaly detection, our method provides a quantitative framework for assessing the risk of information leakage.
4 We validate GLRA on AES, DES, RSA, and debug interfaces. Using AES as a baseline for risk normalization, GLRA identifies maximum-risk leakage paths via transmission cost analysis and improves detection accuracy by 18.75% over anomaly-based methods. It also enables comparative evaluation of protection mechanisms across diverse designs.

The structure of the paper is organized as follows: Section 2 reviews related work, and Section 3 discusses the motivation behind our research. Section 4 outlines the proposed methodology, focusing on the transmission cost metric and leakage risk assessment, respectively. Section 5 and Section 6 present the experimental results and discussion. Finally, Section 7 concludes the paper and provides suggestions for future work.

2. Related works

In this section, we provide a brief review of existing works on detection methods, defense methods, and evaluation methods related to information leakage risk and hardware vulnerabilities.

2.1. Detection methods for information leakage

2.1.1. Information flow tracking

Information flow tracking (IFT) involves monitoring the flow of data through a circuit to detect leakages or unauthorized transmissions. This method aims to identify deviations from expected data paths, which can indicate the presence of HTs, backdoor or other malicious modifications. Witharana (Witharana et al., 2023) and Ayalasomayajula (Ayalasomayajula et al., 2024b) use RTLIFT-based taint tracking to detect information leakage by tracing tainted data paths. Deutschbein (Deutschbein et al., 2021) employs IFT logic to generate simulation traces and analyze information flows. GLIFT (gate-level IFT) (Chen et al., 2021; Hu et al., 2016; Zhang et al., 2022; Meza et al., 2023; Zhao et al., 2024; Qin et al., 2019; Sun et al., 2024) is used to detect HTs that cause information leakage by assigning labels and monitoring their propagation at gete level. Guo (Guo et al., 2019) extends verilog type systems with a quantified information flow model to improve the expressiveness of security rules and evaluate hardware trustworthiness.

2.1.2. Machine learning methods

Machine learning (ML) techniques have emerged as a promising approach to enhance leakage HT or backdoor detection. By training models to distinguish between normal and anomalous behaviors, ML methods can indicate the potential presence of anomalous behaviors. Yasaei (Yasaei et al., 2021; Yasaei et al.) uses a graph-based approach, representing hardware design as data flow graphs generated from RTL (Register Transfer Level) and gate-level netlists, and employs graph neural networks (GNNs) to automatically extract features and identify the presence of HTs. Yen Yen et al. (2023) introduces an approach for HT detection by leveraging path-specific features at gate level. During the training phase, path classifiers are developed using the path features extracted from the training circuit, with support vector machines and random forest algorithms employed for model training. The trained path classifier is then utilized for detection, categorizing logical paths into HT paths and HT-free paths. Kande et al. (2023) investigates the use of large language models to automatically generate security-centric system verilog assertions from natural language prompts.

2.1.3. Side-channel analysis method

Side-channel analysis is a post-silicon technique widely used for detecting hardware Trojans that leak sensitive data during runtime. It collects physical measurements such as power consumption, timing delays, or electromagnetic emissions during chip execution and compares them against a golden reference from a known HT-free design. Atieh and Shahram (2017) proposes a method that measures path delays to identify circuit paths most affected by HTs. Fakir et al. (2018) enhances the power consumption ratio of HT circuits and minimizes variations in detection thresholds. This is achieved through fine-grained circuit partitioning and the use of optimized test pattern sets. Zhang et al. (2018) and He et al. (2018) both present frameworks that use simulated electromagnetic (EM) side-channel signatures from early-stage IC design data, employing neural networks for HT detection without requiring a golden chip reference. SCAR Srivastava et al. (2024) is a power side-channel analysis method designed for the silicon front-end, leveraging GNNs. SCAR accepts the RTL design as input, converting it into a control-data flow graph. It utilizes a GNN model to identify vulnerable modules through node-level classification tasks. Lakshmy et al. (2022) estimates fine-grained signal-level power side-channel leakage early in the design cycle by utilizing information flow analysis and the signal probability correlation factor, though its accuracy is limited by its assumption of acyclic input designs. Chen et al. (2023) proposes a deep learning framework combining ResNeXt and attention mechanisms to accurately detect and classify multiple HTs in AES circuits using side-channel data.

While side-channel methods are effective at runtime detection, they require physical access to fabricated chips and rely on trace-level differences, making them unsuitable for pre-silicon design-time analysis.

2.2. Defense methods for information leakage

To combat information leakage attacks in integrated circuits, various protection methods have been proposed, including antifuse technology, password-based authentication, and signature detection Ren et al. (2018). More recent approaches, such as the Secure JTAG method Lee et al. (2022a), employ a dynamic authentication mechanism that generates unique keys for each test data instance, effectively preventing unauthorized access. Another advanced method leverages physically unclonable functions (PUFs) (Chittoriya et al., 2022; Lee et al., 2022b) to protect against scan chain attacks by ensuring that access keys are uniquely generated and securely managed, thereby enhancing security.

2.3. Evaluation methods for hardware vulnerability

In recent years, several methods have been developed to vulnerability evaluation and security assurance in hardware designs. Farzana Farzana et al. (2019) proposes a property-driven approach (SV) for designing secure system-on-chip (SoC) architectures by developing a comprehensive set of reusable, architecture-agnostic security properties and metrics to guide design and facilitate quantitative security assessment. The controllability and observability for HT detection (COTD) technique (Salmani, 2017; Kok et al., 2019; Zhang et al., 2020; Salmani, 2022) measures the controllability and observability of signals at gate level to detect HTs that are difficult to trigger or observe. Cruz (Cruz et al.) proposes the HT vulnerability factor (TVF) as a novel metric to evaluate a hardware design's susceptibility to HT insertion. Using maximal clique analysis, TVF offers a detailed perspective on HT risks by examining the structural intricacies involved in embedding undetectable HTs, eliminating the need to define specific trigger sizes. Hu et al. (2024) proposes CA4TJ, a metric specifically designed for detecting always-on HTs. By quantifying the correlation between sensitive information and leakage ports in gate-level netlists, it identifies persistent always-on information leakage threats. SecMiner (Ayalasomayajula et al., 2024a) uses static analysis and data mining techniques to automatically generate and rank security assertions, effectively detecting potential information leakages in RTL hardware designs.

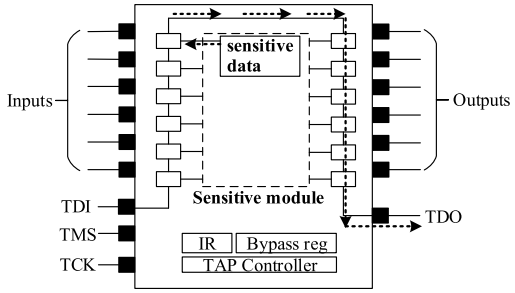

Fig. 2. JTAG design with potential leakage paths (backdoors).

3. Motivation

3.1. Information leakage type attack

As illustrated in Fig. 1, information leakage vulnerabilities can emerge at multiple stages throughout the integrated circuit (IC) design and manufacturing lifecycle. This work focuses specifically on the **design phase**, where logic-level structures are first instantiated and integrated. At this stage, several types of leakage threats may arise-even in the absence of explicit malicious behavior.

We categorize design-stage leakage risks into three main sources:

1) Hardware Trojans (HTs): Malicious entities may embed HTs within third-party intellectual property (3P-IP) cores during design time. These HTs can be either always-on or trigger-based. Always-on HTs constantly transmit sensitive data (e.g., cryptographic keys) through inconspicuous logic structures, while trigger-based HTs activate only under specific rare conditions and selectively leak critical information. These are intentional, covert attack vectors and are particularly challenging to detect.

2) Unintended Backdoors in Trusted Logic: Even logic components developed by trusted parties may inadvertently introduce severe leakage vulnerabilities. A typical example is the unprotected integration of debug infrastructure (e.g., JTAG) with sensitive logic such as cryptographic modules. As shown in Fig. 2, such integration may create unintended data paths that expose secret values (e.g., keys or internal states) to external interfaces like the Test Data Output (TDO) pin. These scenarios exemplify non-malicious yet dangerous design flaws-referred to as "inadvertent backdoors"-which arise when security constraints are not explicitly enforced during module interconnection.

3) Poor Design Practices in Regular Logic: Even general-purpose logic without any malicious intent can leak sensitive data if not carefully

designed. This may result from poor signal isolation, unintentional data reuse, or unsafe reuse of internal nets across sensitive and non-sensitive domains. For instance, if critical control or key signals are reused or routed near externally visible output ports, they may inadvertently propagate to observable points. Such issues typically stem from a lack of security awareness during logic synthesis and verification stages.

Distinct from Side-Channel Attacks: The attack model in this work differs fundamentally from runtime side-channel and fault injection attacks, which exploit physical effects (e.g., power consumption, timing, or electromagnetic emissions) during chip execution. In contrast, our work targets logic-level vulnerabilities embedded in gate-level netlists at design time. We analyze the static structure of the logic itself to identify potential information leakage paths-without relying on runtime behavior, simulation, or pattern matching.

3.2. Our motivation

In gate-level hardware designs, information transmission refers to the propagation of signals through interconnected units. For sensitive information to reach a potentially endpoint, a valid signal path must exist, connecting its source to the leakage point. However, different types of logic units impose varying levels of difficulty when transmitting a signal because they rely on specific input conditions. For example, a signal passing through an inverter is relatively straightforward, as the output depends solely on a single input. In contrast, signal propagation through an AND gate is conditional: the output reflects the target input only when all other inputs are set to logic 1. If any of the other inputs are 0, the output is forced to 0 regardless of the target input, thereby obstructing signal propagation. This conditional dependency increases the complexity of transmission through the gate.

Due to variations in transmission difficulty across different logic units, sensitive signals may reach potential leakage points via multiple paths, each associated with a distinct transmission cost. The ease of signal propagation along these paths directly correlates with the likelihood of information leakage: the lower the transmission cost, the higher the leakage risk, and vice versa. This correlation is supported by prior security research (Ayalasomayajula et al., 2024a), which observes that low-complexity information flows are generally more exploitable by attackers, while high-complexity paths with multiple branches or dependencies tend to increase the attack difficulty. Therefore, accurately assessing a design's overall leakage risk requires analyzing the transmission costs of all feasible paths and identifying the one that poses the greatest risk.

Based on this observation, we propose a transmission cost based metric to quantify the risk of sensitive information leakage. By calculating the transmission cost of sensitive data across different unit types, we can accurately assess the difficulty of signal propagation and identify vulnerable paths within the circuit. Furthermore, by modeling the relationship between transmission cost and leakage risk, we provide an evaluation of potential threats to information leakage.

4. Proposed method

The transmission cost metric is central to our methodology, enabling the precise evaluation of sensitive information leakage risk at gate level. By quantifying the difficulty of signal propagation through different logic units, it identifies critical paths prone to leakage, offering a detailed method for assessing circuit vulnerabilities. The following sections delve into the details of the proposed method.

Before introducing our method, it is necessary to distinguish our methodology from a relevant method called COTD (Salmani, 2017) (Controllability and Observability for HT Detection). COTD is designed to detect HTs by identifying signals that are difficult to trigger (with high controllability value) or observe (with high observability value), under the assumption that such signals are more likely associated with HTs. However, COTD is not well-suited for evaluating information leakage, which follows an opposite principle: sensitive signals that are easier to propagate and observe at the leakage endpoint pose a higher risk. In contrast to COTD, our method is specifically tailored to quantify and address the unique risks of information leakage by focusing on the ease of signal transmission and the vulnerability of critical paths within a circuit. This differentiation underscores the need for a dedicated approach that aligns with the unique nature of information leakage behavior.

4.1. Relationship between transmission cost and leakage risk

The correlation between transmission cost and information leakage risk in GLRA is grounded in Shannon's theory of communication channels. In this context, a signal propagation path from a sensitive source to an observable output can be analogized to a covert communication channel:

- Each valid signal path forms a potential leakage channel, enabling the propagation of sensitive information to external outputs.
- A lower transmission cost reflects fewer control constraints or logic conditions for signal propagation, which increases the effective observability of the source signal.
- As observability increases, the effective channel capacity of the path also increases, reducing the **conditional entropy** $H(X|Y)$, where X denotes a sensitive input and Y an observable output. A lower $H(X|Y)$ implies stronger correlation and greater potential for information leakage.

This information-theoretic perspective supports the central premise of GLRA: paths with lower transmission cost are more vulnerable to information leakage, and therefore play a dominant role in determining the overall risk level of a hardware design. This reasoning is intended as a heuristic justification grounded in information theory, and aligns with prior studies (Ayalasomayajula et al., 2024a) that highlight the relationship between flow complexity and exploitability. While not constituting a formal proof of security, this perspective reinforces the theoretical foundation of GLRA by emphasizing that lower transmission cost paths are more susceptible to information leakage.

Let $\mathcal{P} = (v_1 \to v_2 \to \cdots \to v_k)$ denote a directed path in the gate-level netlist, representing a sequence signal propagation. For each directed path $(v_i \to v_{i+1})$, we define a local transmission cost $TC(v_i \to v_{i+1})$ that reflects how easily a signal can be propagated from signal v_i to v_{i+1}, based on unit type. Then the total transmission cost of path $\mathcal{P}$ is:

$$TC(\mathcal{P}) = \sum_{i=1}^{k-1} TC(v_i \to v_{i+1})$$

This definition reflects the logical effort required to propagate sensitive information through logic units. We propose that lower TC corresponds to lower conditional entropy $H(X|Y)$, indicating greater leakage risk.

Postulate 1 (Risk Mapping): Given two paths $\mathcal{P}_1, \mathcal{P}_2$ with $TC(\mathcal{P}_1) < TC(\mathcal{P}_2)$, we postulate:

$$H(X|Y_1) < H(X|Y_2), \quad \text{thus } LR(Y_1) > LR(Y_2),$$

where Y_1 and Y_2 are the outputs of the respective paths. This postulate maps transmission cost to leakage risk through the inverse of conditional entropy.

Theorem 1. *If a hardware design has multiple sensitive signals, the overall risk of information leakage is primarily determined by the path with the minimum transmission cost.*

Proof. Let a hardware design include sensitive signals $S_1, S_2, \ldots, S_n$. For each signal S_i, suppose it has paths $P_{i1}, P_{i2}, \ldots, P_{im}$ to various outputs. Denote the minimum transmission cost for signal S_i as:

$$TC_{\min}(S_i) = \min_j TC(P_{ij}).$$

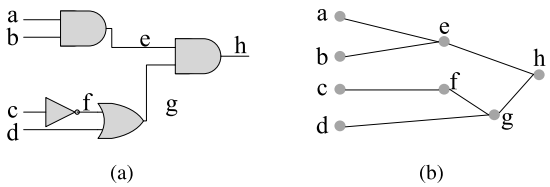

Fig. 3. Signal propagation through logic units.

Based on Postulate 1, the leakage risk of signal S_i satisfies:

$$LR(S_i) \propto \frac{1}{TC_{\min}(S_i)}.$$

The global leakage risk of the system is determined by the most vulnerable signal path (i.e., the one with the lowest TC):

$$LR_{\text{total}} = \max(LR(S_1), \ldots, LR(S_n)) \propto \frac{1}{\min\left(TC_{\min}(S_1), \ldots, TC_{\min}(S_n)\right)}.$$

Hence, the overall system leakage risk is governed by the smallest transmission cost among all minimum-cost paths from each sensitive signal. □

Corollary: By analyzing the minimum transmission cost from a sensitive signal A to all observable outputs, we can determine its leakage risk. If all such paths incur enough high transmission costs, A is deemed secure. Otherwise, the existence of a low-cost path indicates potential leakage.

This information-theoretic interpretation offers a principled rationale for using TC as a risk metric and supports its integration into hardware-level leakage analysis frameworks.

4.2. Transmission cost definition for logic units based on controllability

With the theoretical relationship between transmission cost and leakage risk, we define transmission costs for fundamental gate-level logic units to assess leakage risk. Fig. 3 provides an abstraction of the information flow, such as signal "a" passing through an AND gate to reach "e" and subsequently another AND gate to reach "h". Since different logic units exhibit unique propagation characteristics, we assign distinct transmission costs to each unit type. The differentiation in transmission cost across various unit types allows for a precise evaluation of information leakage risks, tailored to the specific leakage path in the gate-level netlist.

GLRA leverages the concepts of controllability and observability from the testing field to perform transmission cost analysis. Unlike the traditional design-for-testability domain, where observability and controllability are used to define the attributes of individual signals within a circuit, transmission cost measures the propagation characteristics of the path between two signals. To build on this foundation, we introduce two key definitions that are central to our method. These definitions provide two primary controllability metrics:

- **CC0 (Controllability of setting a signal to 0):** Quantifies the effort required to control the signal to logic 0.
- **CC1 (Controllability of setting a signal to 1):** Quantifies the effort required to control the signal to logic 1.

The transmission cost (TC) for both combinational and sequential logic units is defined based on the effort required to propagate a signal, accounting for the control states of other inputs. As illustrated in Fig. 4, the TC values are specified for representative logic units, including basic combinational gates (e.g., NOT, AND, OR) and sequential elements such as the D flip-flop (DFF). This formulation can be extended to more complex circuit structures as needed. The following sections will delve into the detailed calculation methods for these units, explaining how their unique signal propagation characteristics influence the transmission cost.

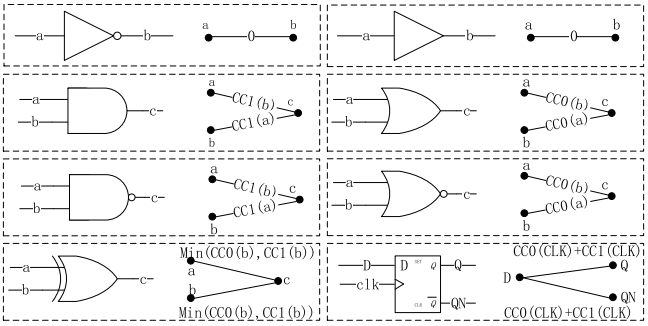

Fig. 4. Transmission cost of different logic units.

- **NOT:** $TC_{\text{NOT}} = 0$. The output of a NOT gate is directly and solely determined by its single input, requiring no additional control effort.
- **AND:** $TC_{\text{AND}} = \sum_{i=1}^{n-1} CC1(\text{other\_input}_i)$, where n represents the number of inputs. For a signal to propagate from the target input to the output, all other $n-1$ inputs must be set to logic 1. Any 0 input forces the output to 0, thereby blocking propagation. The cost reflects the effort to control these inputs to 1.
- **OR:** $TC_{\text{OR}} = \sum_{i=1}^{n-1} CC0(\text{other\_input}_i)$. Propagation through an OR gate requires all non-target inputs to be set to 0; otherwise, the output remains 1, independent of the target input. The cost aggregates the controllability to 0 for all other inputs.
- **XOR:** $TC_{\text{XOR}} = min(CC0(other\_input), CC1(other\_input))$. In a two-input XOR gate, the output depends on the target input only if the other input is fixed to 0 or 1. The transmission cost is the minimum effort required to control the other input to either logic level.
- **DFF:** $TC_{\text{DFF}} = CC0(\text{clk}) + CC1(\text{clk})$. Data propagates from input D to output Q only on a rising edge of the clock signal. Thus, the transmission cost accounts for the effort to control the clock to both 0 and 1. If a reset signal (rst) is present, it must be held at 0 to allow normal operation, and the cost is extended as: $TC_{\text{DFF}} = CC0(\text{clk}) + CC1(\text{clk}) + CC0(\text{rst})$.

Using these metrics, the transmission costs for different types of logic units are determined. To calculate $CC0$ and $CC1$ for each unit in TCs, we employ the SCOAP (Sandia Controllability/Observability Analysis Program) (Wu et al., 2006) method. The SCOAP method starts by calculating the controllability values for all signals, beginning at the primary inputs and moving towards the primary outputs. The controllability of each unit's output is computed after the controllability of all its input signals has been determined. For example, in the case of an AND gate, to set the output to 0, the method focuses on controlling at least one input to 0, selecting the minimum controllability effort among all the inputs. To set the output to 1, all inputs must be controlled to 1, and the controllability values of each input are summed up. Detailed rules for other unit types can be found in Wu et al. (2006). Specifically, the presence of feedback lines between standard cells may cause signals within loops to propagate repeatedly, leading to the controllability values of nodes being influenced by the outcomes of previous iterations. To address this challenge, SCOAP employs an iterative computation method, where nodes are continuously evaluated and updated until the controllability values converge to a stable state. This iterative process ensures the accuracy and reliability of the calculation results, even in the presence of feedback lines, effectively capturing the cyclic dependencies inherent in such circuits.

Once the controllability values for all logic units have been calculated, the transmission cost between logic units can be determined. The transmission costs calculated between different logic units will be used in the subsequent computation of transmission costs along the sensitive information's propagation paths.

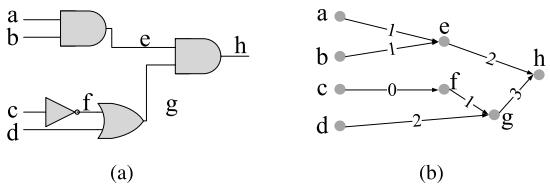

Fig. 5. A hardware design and its corresponding graph representation.

4.3. Edge-weighted graph for identifying shortest path

To identify the shortest transmission paths within a circuit, we calculate the transmission cost of signal paths from source to destination. To achieve this, we build an edge-weighted graph from the gate-level netlist by transforming hardware designs into graph structures. As shown in Fig. 5, the logic units within the netlist are represented as nodes, while the connecting signals are depicted as directed edges. Each edge represents the internal connection from a unit's input to its output, with a transmission cost assigned according to the logic-unit-specific metric introduced in Section 4.2 . By tracing the paths from the sensitive information to the leakage port, we can calculate the transmission cost for each path, taking into account the type of each unit along the path and the transmission cost metrics. The path with the lowest transmission cost represents the easiest path for information leakage.

Locating the path with the highest leakage risk is equivalent to identifying the shortest weighted path from the sensitive information to the leakage port. To compute the shortest path from signal A to signal B, we use Dijkstra's algorithm, which is well suited to find the shortest paths in weighted graphs (shown in Algorithm 1). The algorithm can be described in the following steps: it first initializes the transmission costs of all signals to infinity (∞), except for the source signal A, which is set to 0. A priority queue is initialized with A. While the queue is not empty, the signal u with the minimum transmission cost is extracted. For each neighboring signal v of u, the transmission cost from A to v via u is calculated. If this cost is smaller than the current known cost for v, the cost is updated, u is recorded as the predecessor of v, and v is added to the queue. This process continues until the destination node B is reached. The shortest path from A to B is then reconstructed by tracing back through the predecessors.

To handle reconvergent fanout and feedback structures, GLRA models the circuit as a directed graph with non-negative transmission costs on edges. Dijkstra's algorithm is used to identify minimal-cost paths from source nodes to observable outputs. In reconvergent structures, multiple branches are explored, and only the shortest path to each node is finalized upon its first extraction from the priority queue, ensuring accurate risk identification. The constructed propagation graph may contain cycles, especially due to feedback paths in sequential logic. However, since all edge weights are non-negative, any traversal through a loop only increases the cumulative transmission cost. Consequently, the shortest path to each node is finalized upon its first visit, and longer cyclic paths are naturally disregarded. This ensures that the presence of cycles does not affect the correctness or convergence of the Dijkstra-based analysis.

4.4. Formulaic leakage risk assessment based on minimal transmission cost path

Building upon the conclusions in Section 4.1, the leakage risk of a design is determined by the minimal transmission cost among all potential paths from sensitive sources to observable outputs. Accordingly, the evaluation of leakage risk focuses on this critical path. By leveraging the transmission cost definitions and graph-based path analysis introduced in Section 4.2 and Section 4.3, the cumulative cost along this path can be systematically quantified, providing a basis for evaluating the design's overall risk of information leakage.

Algorithm 1 Identifying the Shortest Path Within Gate-Level Netlist.

Input: Netlist G, Source signal A, Destination signal B
Output: Shortest path from A to B, Minimum transmission cost
Initialize $dist[u] \leftarrow \infty$ for all signals $u \in G$
$dist[A] \leftarrow 0$
Initialize priority queue Q
Insert $(A, 0)$ into Q
while Q is not empty **do**
  $(u, d) \leftarrow Q$.extract_min
  if $u = B$ **then**
    break ▷ Stop if destination B is reached
  end if
  for each neighbor v of u **do**
    $cost \leftarrow$ transmission_cost(u, v)
    if $dist[u] + cost < dist[v]$ **then**
      $dist[v] \leftarrow dist[u] + cost$
      $pred[v] \leftarrow u$
      Insert $(v, dist[v])$ into Q
    end if
  end for
end while
Initialize $path \leftarrow \emptyset$
$current \leftarrow B$
while $current \neq A$ **do**
  Insert $current$ at the beginning of $path$
  $current \leftarrow pred[current]$
end while
Insert A at the beginning of $path$
return $path, dist[B]$

However, to enable meaningful cross-design comparisons and facilitate risk interpretation, it is essential to standardize the quantified leakage risk. Therefore, the minimum TC among all paths is normalized to a scale between 0 and 1, where 0 indicates negligible risk and 1 corresponds to the highest potential vulnerability. This normalization ensures that the leakage risk metric remains consistent and interpretable across different hardware designs and protection mechanisms.

The standardization is based on a predefined threshold $TC_{\text{threshold}}$. Transmission costs greater than this threshold indicate negligible risk and are assigned a value of 0. Conversely, transmission costs below the threshold are scaled proportionally between 0 and 1, with lower costs corresponding to higher risks.

4.4.1. Threshold selection strategy based on baseline guidance

To support meaningful and interpretable leakage risk evaluation, our framework adopts a baseline-guided strategy for threshold selection. Specifically, we define the threshold $TC_{\text{threshold}}$ as the minimal transmission cost of a reference circuit (baseline) that reflects a target security level. This approach enables the evaluation framework to be grounded in concrete, security-relevant implementations, rather than relying on arbitrary or hard-coded constants.

To improve flexibility across heterogeneous IPs and varying security demands, we introduce a parameter α, which represents the desired security level. Larger values of α indicate stricter security expectations and therefore correspond to higher threshold values. The threshold is thus defined as:

$$TC_{\text{threshold}} = TC_{\text{Baseline}(\alpha)}$$

Here, $TC_{\text{Baseline}(\alpha)}$ denotes the minimal transmission cost of a reference design associated with security level α. For example:

- $\alpha = 3$: Baseline(3) represents a high-security design;
- $\alpha = 2$: Baseline(2) represents a medium-security design;
- $\alpha = 1$: Baseline(1) represents a low-security design.

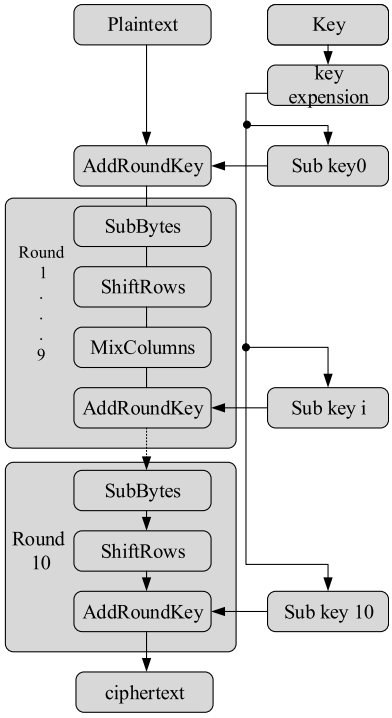

Fig. 6. Iterative process of key confusion and diffusion in AES.

These baselines serve as illustrative examples; in practice, they can be instantiated by representative secure designs appropriate for different application domains. This parameterized formulation allows GLRA to support context-aware leakage risk assessment, without enforcing a one-size-fits-all threshold policy. Designers can select an appropriate baseline based on the targeted security level of their system-choosing stricter reference circuits for critical modules, and more relaxed ones for non-sensitive components-thereby enabling tailored and interpretable risk evaluations aligned with practical needs.

In our implementation, we adopt $\alpha = 3$ as the default setting and select AES-128 encryption as the high-security baseline. AES is a widely adopted symmetric encryption algorithm that provides robust security for data. Its security stems from an iterative process of confusion and diffusion applied over multiple rounds, as illustrated in Fig. 6. Each round consists of operations such as SubBytes, ShiftRows, MixColumns, and AddRoundKey, which collectively create complex dependencies between plaintext, ciphertext, and the encryption key.

We use the transmission cost from the encryption key to the ciphertext after ten AES rounds as our baseline value for high-security designs. This cost reflects the structural propagation difficulty imposed by multiple layers of logic, and serves as a conservative reference for flagging potentially insecure designs during risk screening.

4.4.2. Standardization

Given $TC_{\min}$, the relationship between $TC_{\min}$ and LR (leakage risk) is calculated using the following formula:

$$LR(TC_{min}) = \begin{cases} 1 - \frac{TC_{min}}{TC_{\text{threshold}}} & \text{if } TC_{min} \le TC_{\text{threshold}} \\ 0 & \text{if } TC_{min} > TC_{\text{threshold}} \end{cases}$$

By mapping transmission costs to risk values between 0 and 1, designers can better understand and manage the security implications of their designs. when $TC_{min} = 0$, meaning that no transmission effort is required, LR reaches 1, indicating the highest possible risk. When TC_{min} is less than or equal to $TC_{\text{threshold}}$, the risk value is calculated as $1 - \frac{TC_{min}}{TC_{\text{threshold}}}$. As the transmission cost decreases, the risk value approaches 1, indicating higher leakage risk. If TC_{min} exceeds the threshold, the corresponding leakage risk value LR is set to 0. This does not imply the complete absence of risk, but rather indicates that the transmission cost is higher than the reference baseline and thus represents negligible risk in a relative sense. In other words, the LR value of 0 should be interpreted as "lower than the threshold-defined standard" rather than an absolute absence of leakage.

4.5. Complexity and scalability analysis of GLRA

The proposed GLRA framework operates through three sequential phases. First, a transmission cost assignment phase traverses all circuit units, assigning predefined transmission costs based on their types. Let n denote the number of logic units in the netlist. Since each gate is processed exactly once, the time complexity of this phase is $O(n)$.

In the second phase, for each sensitive source node, GLRA computes the shortest transmission cost paths to all observable outputs. The circuit is modeled as a directed graph comprising n nodes and m edges. Dijkstra's algorithm is employed to compute shortest paths, with a time complexity of $O(m + n \log n)$ per source node. Considering s sensitive sources, the overall complexity for this phase becomes $O(s(m + n \log n))$.

The final phase evaluates the leakage risk by selecting the minimum transmission cost from each sensitive source to the observable outputs. This evaluation requires scanning through the computed path costs, resulting in a complexity of $O(s \times o)$, where o denotes the number of output nodes.

In summary, the overall computational complexity of GLRA is dominated by the shortest path computation. Since the number of sensitive sources s is typically much smaller than the total number of logic units n in practical hardware designs, GLRA maintains favorable scalability and efficiency for pre-silicon hardware security verification.

5. Experiment and analysis

This section presents a comprehensive evaluation of GLRA across multiple types of information leakage risks introduced during the design phase. The experiments cover three representative scenarios: (1) leakage caused by stealthy HTs, which are maliciously inserted by untrusted designers or third-party vendors; (2) leakage through standard debug infrastructures such as JTAG, which are intentionally designed by trusted parties but can be exploited when left unprotected; and (3) unintentional leakage due to security-unaware design practices in regular logic (e.g., certain non-cryptographic designs), where insecure design practices may cause sensitive data to unintentionally propagate to observable outputs.

By structuring the experiments to include HT-infected and HT-free designs, designs with and without protected interfaces (e.g., JTAG with and without obfuscation), and non-cryptographic designs, we demonstrate that GLRA not only generalizes across design types but also distinguishes between secure and insecure usage of the same logic components. These capabilities highlight the novelty and practicality of GLRA as a design-time leakage assessment tool.

5.1. Experimental workflow

Fig. 7 outlines the overall steps of GLRA. For a given design D, the sensitive signal S_i is first identified. The selection of sensitive signals depends on the context and purpose of the hardware design. Typically, these signals include encryption keys, intermediate values in cryptographic circuits, or other critical data points that, if leaked, could compromise the security of the design. Next, the transmission costs between different logic units are calculated (see Section 4.2). Then, we construct the edge-weighted graph and calculate transmission costs for potential paths (P_{i1}, P_{i2}, …, P_{in}) from S_i (see Section 4.3). The transmission costs of these paths (TC_{i1}, TC_{i2}, …, TC_{in}) are then utilized to identify the most vulnerable paths in the hardware design, and ultimately quantify the overall risk of leakage in the hardware design.

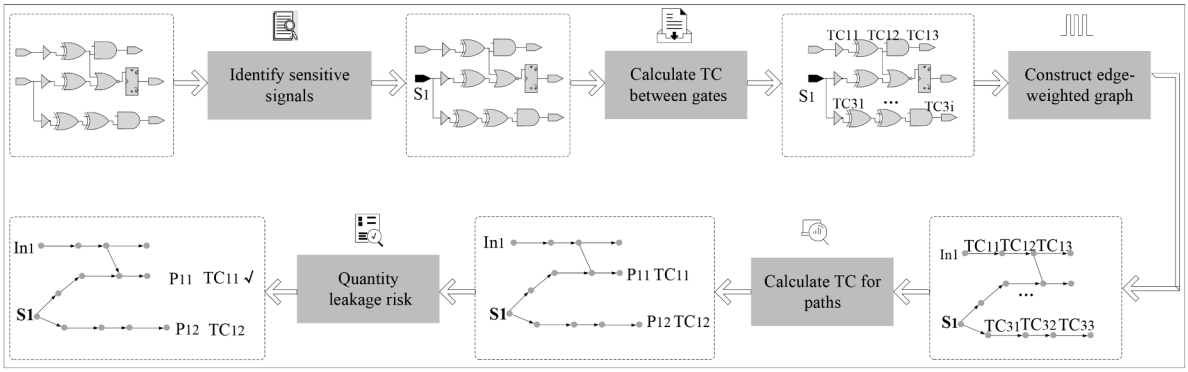

Fig. 7. Implementation process of GLRA.

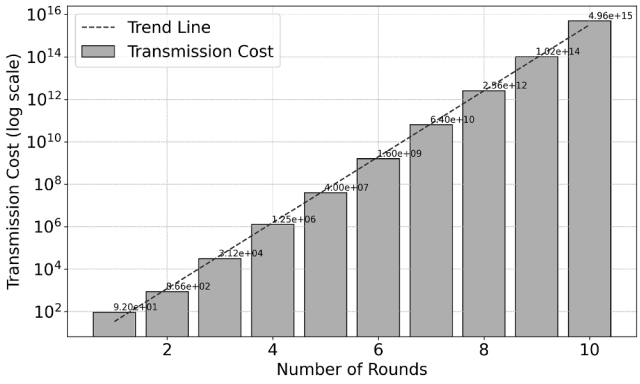

Fig. 8. Transmission cost vs. number of rounds in AES.

5.2. Information leakage risk assessment for HT-free AES with GLRA

The security of AES is primarily ensured through its iterative process of key confusion and diffusion over multiple rounds. Each round consists of several operations, including SubBytes, ShiftRows, MixColumns, and AddRoundKey, which together provide substantial confusion and diffusion. Analyzing the transmission cost from the AES key to the ciphertext in each round provides valuable insights into the hardware security of AES.

To perform our analysis, we synthesized the RTL code of the 128-bit LUT-based implementation of AES in Trust-Hub Salmani et al. (2013) to gate-level netlists using the logic synthesis tool. We then applied GLRA to analyze the synthesized designs and the results are summarized in Fig. 8. The transmission cost of the path is plotted on a logarithmic scale to effectively capture the exponential growth across rounds. Each bar represents the path transmission cost for a specific number of AES rounds, with the actual values labeled above the bars for clarity. The data indicates a significant increase in transmission cost as the number of rounds increases, with the cost increasing from 92 after the first round to 4.96×10^{15} after the 10th round.

A trend line, shown in Fig. 8, further emphasizes the exponential growth of the transmission cost. The trend line is generated using a polynomial fit on the logarithmic scale of the transmission cost data. The increase in transmission cost underscores the enhanced security provided by subsequent rounds of AES encryption, making it increasingly difficult for an attacker to steal the key from the output. As the number of encryption rounds increases, the leakage risk, as quantified by the formulaic assessment in Section 4.4, correspondingly decreases. This observation aligns with the conclusion that AES becomes increasingly secure with additional encryption rounds.

5.3. Leakage risk assessment on cryptographic benchmarks with GLRA

In this section, we evaluate the leakage risks of representative cryptographic hardware designs, including AES-128bit and RSA-32bit benchmarks from Trust-Hub (Salmani et al., 2013), as well as DES-64bit benchmarks. For AES, we analyze seven HT-inserted variants-AES-T100, T200, T300, T400, T700, T800, and T1200-covering both always-on and trigger-activated Trojans. The RSA benchmarks include RSA-free, RSA-T100, and RSA-T300. For DES, we manually inject HTs into the DES-free design to construct DES-T100 and DES-T200, demonstrating the applicability of our method to different cryptographic architectures.

Table 1 summarizes the experimental results for leakage risk detection, presenting the benchmark circuits, inserted Trojan types, ground truth labels, unit counts, minimum TC, LR, detection outcomes via GLRA, and corresponding run times in seconds. As shown in Table 1, for AES-T100, AES-T200, AES-T300, DES-T100, and DES-T200-representing always-on Trojans-GLRA identifies low transmission costs along key paths, yielding a high leakage risk score of 0.9999, which aligns with their continuous leakage behavior. For triggered HTs such as AES-T400, AES-T700, AES-T800, DES-T700, DES-T800, RSA-T100, and RSA-T300, although the transmission costs are moderately higher, they remain below the threshold, resulting in similarly high leakage risk scores. This confirms the method's ability to capture conditional leakage threats. Overall, GLRA successfully identifies high-risk paths in most designs, with the exception of AES-T1200, and also reports potential risks in DES-free and RSA-free samples. Notably, anomaly-based taint tracking methods classify AES-T1200 as exhibiting leakage risk, while DES-free and RSA-free are deemed risk-free. In contrast, GLRA provides a differentiated assessment of these cases. These results will be further investigated through detailed case-by-case analysis to validate the presence or absence of actual vulnerabilities.

5.3.1. Case study 1: AES-T1200

For AES-T1200, the minimum transmission cost from the encryption key to the output is measured as 5.0×10^{15}, which is approximately equal to the AES baseline used as the threshold. Based on our normalization strategy, this results in a leakage risk score of 0 under the AES-derived threshold, indicating negligible risk from the shortest-path perspective.

To further investigate this case, we refer to the dataset specification, which explicitly defines a known leakage path triggered by a 128-bit counter. The counter starts at zero and increments by one each cycle until it overflows, at which point the Trojan becomes active. The transmission cost associated with this specific triggered path is computed as 6.8×10^{38}, significantly higher than the AES-derived threshold.

To assess whether such a triggered path poses practical leakage risk, we analyze the time required for counters of different bit-widths to reach their overflow point, assuming a 1GHz system clock. As shown in Table 2, a 16-bit counter overflows in 2.08×10^{-12} years, while a 64-bit counter takes approximately 584 years. In contrast, a 128-bit counter

Table 1
Leakage risk detection results on encryption benchmarks.

Benchmarks	HT Type	Ground Truth	Unit No.	Min TC	LR	Leakage Detected		Run Time (s)
						GLRA	Existing Methods	
AES-free	HT free	No Leakage	206,537	5.0×10^{15}	0	× (TN)	× (TN)	53.451
AES-T100	Always-on	Leakage Exists	198,753	2	1	✓(TP)	✓(TP)	51.686
AES-T200	Always-on	Leakage Exists	198,793	3	1	✓(TP)	✓(TP)	50.238
AES-T300	Always-on	Leakage Exists	198,909	13	1	✓(TP)	✓(TP)	39.242
AES-T400	Triggered	Leakage Exists	199,158	1.5×10^{8}	1	✓(TP)	✓(TP)	37.593
AES-T700	Triggered	Leakage Exists	206,773	1.34×10^{2}	1	✓(TP)	✓(TP)	34.964
AES-T800	Triggered	Leakage Exists	206,839	1.309×10^{3}	1	✓(TP)	✓(TP)	34.808
AES-T1200	Triggered	No Leakage	207,235	5.0×10^{15}	0	× (TN)	✓(FP)	66.319
DES-free	HT free	Leakage Exists	15,959	6.8×10^{11}	0.9999	✓(TP)	× (FN)	2.504
DES-T100	Always-on	Leakage Exists	16,072	2	1	✓(TP)	✓(TP)	2.750
DES-T200	Always-on	Leakage Exists	16,112	3	1	✓(TP)	✓(TP)	2.765
DES-T700	Triggered	Leakage Exists	16,238	70	1	✓(TP)	✓(TP)	2.854
DES-T800	Triggered	Leakage Exists	16,256	670	1	✓(TP)	✓(TP)	2.770
RSA-free	HT free	Leakage Exists	4953	216	1	✓(TP)	× (FN)	3.560
RSA-T100	Triggered	Leakage Exists	5002	114	1	✓(TP)	✓(TP)	3.732
RSA-T300	Triggered	Leakage Exists	5284	125	1	✓(TP)	✓(TP)	2.915

Note: "Existing methods" denote RTLIFT, GLIFT, and SecMiner, which are representative taint-tracking-based anomaly detection techniques.

Table 2
Transmission cost and the time required for overflow for counters of different bit-widths.

Netlist	Unit No.	Sensitive Information	Leakage port	Min TC	LR	Trigger Time (year)
AES-T1200-counter16	206827	key[0]	Capacitance[0]	1.3E+5	1	2.08E-12
AES-T1200-counter32	206885	key[0]	Capacitance[0]	8.9E+9	1	1.36E-7
AES-T1200-counter48	206954	key[0]	Capacitance[0]	5.6E+14	0.8880	8.90E-3
AES-T1200-counter64	207001	key[0]	Capacitance[0]	3.6E+19	0	5.84E+2
AES-T1200-counter80	207066	key[0]	Capacitance[0]	2.4E+24	0	3.83E+7
AES-T1200-counter96	207140	key[0]	Capacitance[0]	1.5E+29	0	2.51E+12
AES-T1200-counter112	207184	key[0]	Capacitance[0]	1.0E+34	0	1.65E+17
AES-T1200-counter128	207235	key[0]	Capacitance[0]	6.8E+38	0	1.08E+22

requires 1.080×10^{22} years to reach its maximum value-far exceeding the operational lifespan of any realistic system.

Therefore, in the case of AES-T1200, although a specific leakage path is defined by the dataset, its associated transmission cost is significantly higher than the AES baseline. Further analysis reveals that the path is only activated upon the overflow of a 128-bit counter, which is practically infeasible within any realistic system lifetime.

5.3.2. Case study 2: RSA-free

For RSA-free, the minimum transmission cost from the encryption key to the output is measured as 216, which is significantly lower than the AES-derived threshold (5.0×10^{15}). This yields a normalized leakage risk of 1, indicating a high probability of key information leakage under the current evaluation framework. From a shortest-path perspective, the circuit exhibits insufficient protection against leakage.

To further understand this result, we refer to the design specification of RSA-free from Trust-Hub (Salmani et al., 2013). Although RSA is a widely adopted public-key encryption algorithm grounded in the computational hardness of integer factorization, its security relies heavily on the use of sufficiently large key sizes. The RSA-free benchmark adopts only a 32-bit modulus, which can be trivially factored using modern computing resources, rendering the private key easily recoverable from public parameters. This severely compromises the confidentiality of the system, even without any malicious hardware modification.

Furthermore, unlike symmetric encryption schemes such as AES that employ multiple rounds of non-linear confusion and diffusion, RSA operations are based on modular exponentiation, which lacks iterative obfuscation of the internal state. Consequently, the information flow from the key to the output in RSA is relatively direct, resulting in inherently low transmission cost paths. This structural simplicity contributes to the high leakage risk reported by the evaluation.

Therefore, although RSA-free does not embed an HT, its low transmission cost from the key to the output reflects a structural vulnerability stemming from both an insufficient key size and a lack of iterative obfuscation.

5.3.3. Case study 3: DES-free

For DES-free (64-bit), the minimum transmission cost from the encryption key to the output is 6.8×10^{11}, significantly lower than the AES baseline, resulting in a normalized leakage risk of 0.9999. This suggests that sensitive information can propagate to observable outputs with relatively low effort.

DES-free, though employing confusion and diffusion like AES, uses a shorter 56-bit key and fewer rounds, reducing its structural complexity. Consequently, DES-free exhibits a lower transmission cost and higher leakage risk under our evaluation framework, indicating weaker protection against key propagation.

Thus, DES-free shows moderate structural protection but falls short of AES-level security. The relatively low transmission cost reveals potential leakage paths, highlighting the framework's ability to reflect algorithmic differences in security strength.

5.3.4. Summary

We classify the detection results into four categories: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Specifically, TP and TN denote correct assessments of the presence or absence of leakage risk, while FP corresponds to incorrectly flagged leakage risk in secure designs, and FN indicates missed detections of actual leakage vulnerabilities. To evaluate detection performance, we adopt the accuracy metric defined as: Acc = (TN + TP) / (TN + TP + FN + FP).

As shown in Table 1, GLRA achieves perfect detection across all 16 evaluated benchmarks, resulting in an accuracy of 100%. In

contrast, traditional taint-tracking-based anomaly detection methods-such as RTLIFT, GLIFT, and SecMiner (see Table V for a detailed comparison)-misclassify AES-T1200 (FP), RSA-free (FN), and DES-free (FN), yielding an overall accuracy of 13 out of 16 (81.25%). These results indicate that GLRA improves detection accuracy by 18.75% over existing approaches, particularly in challenging cases where structural anomalies or subtle vulnerabilities are present.

Notably, GLRA flags potential leakage risks in HT-free designs such as RSA-free and DES-free, which are typically marked as safe by anomaly-based methods. This does not represent a misclassification but rather reveals the presence of structurally low-cost signal propagation paths. These paths may constitute latent vulnerabilities even in the absence of explicit Trojans. Such observations highlight GLRA's strength in proactively identifying risks and its utility in pre-silicon hardware security auditing and robustness evaluation.

5.4. Leakage risk assessment of unprotected and protected interfaces with GLRA

Unprotected interfaces like JTAG present substantial information leakage risks, particularly when integrated with modules handling sensitive information. Using GLRA, we calculate the transmission cost of the shortest leakage path from the sensitive information to TDO and the value is 1500, and the corresponding leakage risk is 1. The calculation result reveals that, without additional safeguards, the JTAG interface can inadvertently enable pathways for sensitive information leakage. It is necessary for designers to implement extra protective measures to prevent sensitive information leakage.

In response to the leakage risks associated with debug interfaces-using JTAG as a representative example-several protection techniques, such as eFuse, password authentication, encryption, and PUF-based methods, have been introduced. Although originally designed to enhance system security, we evaluate these techniques from the perspective of transmission cost analysis.

- **eFuse-based Protection:** eFuse-based protection irreversibly disables the JTAG interface, resulting in prohibitively high transmission costs and an evaluated LR of 0. While this eliminates leakage paths, it also prevents post-deployment debugging and maintenance.
- **Password-based Protection:** Password mechanisms aim to restrict access but leave the underlying JTAG structure largely intact. As a result, the transmission cost remains low, and the LR approaches 1. This indicates a high risk of information leakage, especially in cases where short passwords are used, since the associated increase in transmission cost is minimal, resulting in insufficient disruption of potential leakage paths.
- **Encryption-based Protection:** Applying encryption to JTAG communication substantially increases the transmission cost. When strong and consistently enforced encryption schemes (e.g., AES-based) are employed, the resulting leakage risk (LR) approaches 0. However, if encryption is applied only to part of the communication protocol, uses low-entropy keys, or lacks proper key management and enforcement mechanisms, residual leakage paths may persist.
- **PUF-based Protection:** Physically Unclonable Function (PUF)-based mechanisms enable dynamic key generation and enhance authentication, serving as a foundation for integrating encryption in debug access control. While the PUF itself does not directly raise the transmission cost of leakage paths, when combined with encryption schemes, it contributes to securing critical paths by enabling per-device key variability. In such cases, the overall leakage risk (LR) can be significantly reduced. However, the effectiveness of this protection relies on the robustness of the associated cryptographic implementation, and PUF designs may still face practical limitations such as instability under environmental variation or device aging.

As shown in Table 3, different JTAG protection techniques impose varying impacts on the transmission cost of sensitive signals, thereby influencing the evaluated leakage risk (LR). Techniques such as eFuse, encryption, and PUF-assisted key management significantly increase transmission cost, resulting in an LR of 0. In contrast, password-based protection retains low-cost leakage paths and yields a high LR of 1, particularly when weak or short passwords are used. These evaluations highlight GLRA's effectiveness in quantifying the impact of different protection mechanisms and revealing latent risks through transmission-cost path analysis.

Table 3
Evaluation of JTAG protection techniques.

Technique	LR	Impact
eFuse	0	Physically blocks leakage paths; disables debugging
Password	1	Retains low-cost paths; weak if password is short or static
Encryption	0	Blocks leakage with strong encryption requires secure key management
PUF + Crypto	0	Enables dynamic keying; leakage depends on encryption placement

5.5. Leakage risk assessment on non-cryptographic designs with GLRA

Although the evaluation in this work primarily focuses on cryptographic benchmarks (e.g., AES, RSA, DES) due to their well-defined leakage surfaces, GLRA is not limited to such domains. Since GLRA performs structural analysis directly on gate-level netlists using transmission costs, it is applicable to any logic design where the unintended propagation of sensitive signals poses a security concern.

To demonstrate this generalizability, we evaluate GLRA on several non-cryptographic designs, including two artificial intelligence accelerators and an embedded controller. As shown in Table 4, GLRA identifies potential leakage paths where sensitive data (e.g., model weights, program code) may reach external interfaces (e.g., output buffers or serial ports). For example, in `gc_nn_controller`, weight data from on-chip SRAM can propagate to unprotected output interfaces, indicating the need for encryption or masking. In `gc_nn_conv_core`, coefficient input ports can be observed directly through output channels, potentially allowing adversaries to reverse-engineer model behavior. For the `MC8051`, program data from ROM is observed reaching serial transmit ports, posing risks of instruction-level leakage.

These results confirm that GLRA effectively detects logic-level information leakage even in non-cryptographic systems. It supports the evaluation of emerging SoC control logic and machine learning accelerators, where unintended data exposure through design-time flaws may exist. This broad applicability demonstrates GLRA's potential as a unified framework for design-stage leakage risk assessment across diverse hardware systems.

5.6. Runtime analysis

To evaluate the practical efficiency of GLRA, we conducted runtime measurements across various benchmark circuits. These results, summarized in Table 1, confirm that GLRA achieves scalable performance in both small and large designs.

Specifically, for common cryptographic benchmarks such as AES, DES, and RSA, the full analysis completes within seconds to under a minute. To further assess scalability, we include large-scale non-cryptographic designs exceeding 1 million gates. For example, GLRA completes analysis on the gc_nn_controller, a 2.17M-gate AI accelerator, in 322.71 seconds and on gc_nn_conv_core (0.3M gates) in 59.88 seconds (as shown in Table 4). These results confirm that GLRA maintains efficient runtime even in industrial-scale SoC scenarios, making it suitable for pre-silicon security evaluations in real-world hardware development workflows.

Table 4
Leakage risk aasessment on non-cryptographic designs..

Design	Function	Gate Count	From	To	Min TC	LR	Time (s)
gc_nn_conv_core	AI Accelerator1	302,631	coef_fifo_in_data[0]	out_data[0]	622	1	59.88
gc_nn_controller	AI Accelerator2	2,174,903	sram_rdata_coreker[0]	coef_fifo_in_data[0]	160	1	322.71
MC8051	Embedded Controller	9152	rom_data_i[0]	all_txd_o[0]	8303	1	3.40

Table 5
Comparison with existing evaluation approaches.

Work	Targeted Level	Method	Leakage Detection	Leakage Quantification	Limitations
RTLIFT, GLIFT (Witharana et al. (2023)-Guo et al. (2019))	RTL, Gate	Security label tracking	✓	×	Misclassifies AES-T1200 (FP), RSA-free/DES-free (FN)
SV Farzana et al. (2019)	RTL / Gate	Property checking	∘	×	Misses indirect/subtle leakage paths; fails on subtle leakage benchmarks (e.g., AES-T1200, RSA-free)
COTD Salmani (2017), Kok et al. (2019), Zhang et al. (2020), Salmani (2022)	Gate	Trigger/observe difficulty analysis	∘	∘	High false positives on AES-like designs
TVF Cruz et al.	Gate	Maximal clique analysis	∘	∘	Limited accuracy for always-on leakage risks
SecMiner Ayalasomayajula et al. (2024a)	RTL	Static & statistical analysis	✓	✓	Assumes longer paths imply lower risk; ignores propagation difficulty; misclassifies AES-T1200 (FP), RSA-free/DES-free (FN)
GLRA (Ours)	Gate	Path transmission cost analysis	✓	✓	Sensitive to user-defined baseline selection

✓: *Supported;* ×: *Not supported;* ∘: *Partially supported.*

6. Discussion

6.1. Comparison with existing approaches

We summarize existing techniques and highlight their respective limitations shown in Table 5).

RTLIFT and GLIFT (Witharana et al. (2023)-Guo et al. (2019)) perform security label tracking at RTL and gate level. While effective in detecting information flow violations, they require shadow logic insertion and dynamic simulation with specific input patterns, which increases design complexity and risks incomplete coverage. Moreover, they rely on binary label propagation and cannot distinguish whether a detected path is practically exploitable. For instance, in AES-T1200, propagation exists but is infeasible due to extremely high complexity. In contrast, designs like RSA-free (32-bit) may be deemed secure due to correct logic but still harbor low-cost leakage paths that remain undetected.

Security verification (SV) Farzana et al. (2019) uses assertion-based checks to detect potential vulnerabilities early in the design process. While effective for identifying functional anomalies, SV is not tailored for security-specific violations and often misses indirect or subtle leakage paths Ayalasomayajula et al. (2024a). Moreover, it lacks quantitative evaluation capabilities and cannot assess the severity of leakage risks. For example, SV fails to detect information leakage in benchmarks such as AES-T1200 and RSA-free, where the leakage paths are subtle and not directly tied to functional violations. These limitations hinder its applicability in scenarios that require precise and comprehensive leakage risk assessment.

COTD and related techniques (Salmani, 2017; Kok et al., 2019; Zhang et al., 2020; Salmani, 2022) evaluate trigger difficulty by analyzing the controllability of rare events. These are effective for detecting stealthy or performance-degrading HTs. However, they are less effective for “always-on” leakage paths that do not require triggers. Moreover, in cryptographic designs like AES with high structural complexity, even benign paths may appear hard to control, leading to high false-positive rates.

TVF Cruz et al. provides structural insights by analyzing the test vector footprint and graph structures like maximal cliques to evaluate Trojan susceptibility. While informative for insertion risk, it is not tailored for detecting or quantifying information leakage, especially in always-on scenarios.

SecMiner Ayalasomayajula et al. (2024a) combines static RTL analysis with data mining to detect and rank information leakage paths. The core metric used by SecMiner assumes that the severity of information leakage decreases with the number of variables traversed along the path—i.e., the more intermediate variables a path includes, the lower its leakage risk. This assumption oversimplifies the propagation complexity by treating all variables equally, without distinguishing their functional roles or propagation difficulty. In contrast, our proposed GLRA approach defines a controllability-based transmission cost model that captures the logical constraints imposed by each gate-level component. This enables GLRA to provide a more accurate and physically meaningful assessment of leakage difficulty. Consequently, SecMiner may misclassify critical structural vulnerabilities that do not manifest through simulation behavior. As observed in our experiments, it fails to detect the AES-T1200 leakage path (FP) and overlooks subtle vulnerabilities in RSA-free and DES-free designs (FN), thereby limiting its applicability for gate-level risk quantification and security validation.

Our proposed method, GLRA, enables accurate detection and quantitative evaluation of information leakage at gate level. Through minimal transmission cost path analysis, GLRA systematically assesses the difficulty of signal propagation from sensitive sources to outputs, providing a precise framework for gate-level security validation.

6.2. Limitation and discussion on threshold adaptation

While GLRA offers a quantitative framework for gate-level leakage risk assessment and demonstrates superior accuracy compared to

anomaly-based approaches, it is important to clarify the scope of its applicability.

We propose a parameterized threshold adaptation strategy, where the threshold is modeled as a function of a user-specified security level α. For example, different security levels (e.g., high, medium, low) can be mapped to corresponding α values, which are then used to calculate threshold values through $TC_{\text{threshold}} = f(\alpha)$. This offers a tunable mechanism to tailor the evaluation to different risk tolerance levels or application domains.

However, there currently exists no universally accepted hardware security grading standard or benchmark IP core that can serve as a canonical baseline for leakage risk. As a result, it is challenging to select a reference design that is formally recognized as "secure enough" across heterogeneous SoC environments. The use of AES-128 in our framework stems from its well-established design and widespread recognition as a strong cryptographic primitive. Nonetheless, we emphasize that it is a practical surrogate rather than an absolute standard.

In future work, this thresholding mechanism can be further refined through statistical profiling of per-design transmission cost distributions or learned models derived from security-labeled circuit datasets. These enhancements would enable GLRA to provide adaptive, context-aware leakage assessments while preserving structural interpretability.

Importantly, GLRA is not intended to define an absolute or universally accepted notion of security level. Rather, it provides a structural and quantitative approach to evaluate the relative difficulty of sensitive signal propagation. What constitutes an "acceptable" leakage risk is inherently application-dependent. For instance, in heterogeneous SoC environments, non-cryptographic modules may tolerate greater levels of signal propagation without violating system-level security requirements. By offering tunable analysis capabilities, GLRA aims to support practical design-time security assessments under varying requirements, rather than enforcing a one-size-fits-all policy.

7. Conclusion and future work

In this study, we propose GLRA, a novel graph-based framework for quantifying gate-level information leakage risk. By modeling hardware circuits as edge-weighted graphs based on the transmission costs defined for various logic units, GLRA captures the propagation difficulty of signals from sensitive sources to observable outputs. Beyond the traditional "path existence" criterion used in anomaly label-based taint tracking, GLRA introduces a quantitative metric to identify the minimal-cost propagation path. Moreover, leveraging the formulaic relationship between transmission cost and leakage risk, GLRA enables quantifiable risk assessment across the entire design. Experimental results on AES, DES, and RSA benchmarks demonstrate that GLRA accurately identifies critical leakage paths. Specifically, for the challenging cases of AES-T1200, RSA-free, and DES-free, the computed minimal transmission costs are significantly higher or lower than the secure baseline threshold, allowing GLRA to correctly classify these cases, which are often misclassified by anomaly-driven approaches. Additionally, GLRA supports risk assessment of various debug interfaces protection mechanisms by quantifying the impact of different countermeasures on information leakage levels, demonstrating its potential for evaluating the effectiveness of security strategies. Importantly, we extend the evaluation to non-cryptographic scenarios, including SoC control logic and machine learning accelerators. These experiments demonstrate that GLRA generalizes well beyond cryptographic applications and effectively captures leakage risks arising from poorly isolated data channels. Overall, the results confirm that GLRA offers improved accuracy, interpretability, and applicability in gate-level hardware security analysis. Future work will focus on integrating GLRA into standard IC design flows to enhance pre-silicon security evaluation, as well as developing adaptive risk thresholding mechanisms to support context-aware and design-specific risk assessment that better aligns with the needs of hardware designers.

CRediT authorship contribution statement

Xing Hu: Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization; **Yang Zhang:** Writing – review & editing, Methodology, Formal analysis; **Sheng Liu:** Writing – review & editing; **Xiaowen Chen:** Writing – review & editing; **Yaohua Wang:** Writing – review & editing; **Shaoqing Li:** Writing – review & editing, Methodology; **Zhenyu Zhao:** Writing – review & editing; **Keqin Li:** Writing – review & editing.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Atieh, A., Shahram, E.B., 2017. A side-channel analysis for hardware trojan detection based on path delay measurement. J. Circ. Syst. Comput. 27 (9), 1850138.

Ayalasomayajula, A., Farzana, N., PalD., Farahmandi, F., 2024a. Prioritizing information flow violations: generation of ranked security assertions for hardware designs. In: 2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 128–138. https://doi.org/10.1109/HOST55342.2024.10545352

Ayalasomayajula, A., Farzana, N., Pal, D., Farahmandi, F., 2024b. Prioritizing information flow violations: generation of ranked security assertions for hardware designs. In: 2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 128–138. https://doi.org/10.1109/HOST55342.2024.10545352

Chen, C.L., Wang, S.X., Tan, J., Zhu, J.C., Hu, W., 2021. Hardware information flow security verification and vulnerability detection using yosys. Appl. Res. Comput. 38 (6), 1865–1869.

Chen, S., Wang, T., Huang, Z., Hou, X., 2023. Detection method of golden chip-free hardware trojan based on the combination of resnext structure and attention mechanism. Comput. Secur. 134, 103428. https://doi.org/10.1016/j.cose.2023.103428

Chittoriya, S., Shivdeep, K.K., Jha, D.M., Das, D.M., Sharma, R., 2022. A low-overhead puf based hardware security technique to prevent scan chain attacks for industry-standard dft architecture. In: 2022 IEEE 65Th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4. https://doi.org/10.1109/MWSCAS54063.2022.9859268

Cruz, J., Slpsk, P., Gaikwad, P., Bhunia, S., . Tvf: a metric for quantifying vulnerability against hardware trojan attacks. IEEE Trans. Very Large Scale Integr. VLSI Syst. 31, 7 969–979.

Deutschbein, C., Meza, A., Restuccia, F., Kastner, R., Sturton, C., 2021. Isadora: automated information flow property generation for hardware designs. In: Proceedings of the 5Th Workshop on Attacks and Solutions in Hardware Security, pp. 5–15.

Fakir, S., Michihiro, S., Michiko, T., . Variation-aware hardwareTrojan detection through power side-channel. 2018 of of IEEE In-ternational Test ConferencePiscataway, NJ. IEEE Press 1–10.

Farzana, N., Rahman, F., Tehranipoor, M., Farahmandi, F., 2019. Soc security verification using property checking. In: IEEE International Test Conference (ITC), pp. 1–10. https://doi.org/10.1109/ITC44170.2019.9000170

Guo, X., Dutta, R.G., He, J., Tehranipoor, M.M., Jin, Y., 2019. Qif-verilog: quantitative information-flow based hardware description languages for pre-silicon security assessment. In: 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 91–100. https://doi.org/10.1109/HST.2019.8740840

He, J., Liu, Y., Yuan, Y., et al., 2018. Golden Chip Free Trojan Detection Leveraging Electromagnetic Side Channel Fingerprinting. Vol. 16. IEICE Electronics Express. https://doi.org/10.1587/elex.16.20181065

Hu, W., Mao, B., Oberg, J., Kastner, R., 2016. Detecting hardware trojans with gate-level information-flow tracking. Comput. (Long Beach Calif) 49 (8), 44–52.

Hu, X., Zhang, Y., Li, Z., Li, S., 2024. Ca4tj: correlational analysis for always-on information-leakage hardware trojan detecting. In: 2024 IEEE International Test Conference in Asia (ITC-Asia), pp. 1–6. https://doi.org/10.1109/ITC-Asia62534.2024.10661328

Kande, R., Pearce, H.A., Tan, B., Dolan-Gavitt, B., Thakur, S., Karri, R., J, R.T., 2023. LLM-assisted generation of hardware assertions. Technical Report abs/2306.14027. A. University, U. of New South Wales, U. of Calgary, and N. Y. University.

Kok, C.H., Ooi, C.Y., Moghbel, M., et al., 2019. Classification of trojan nets based on scoap values using supervised learning. https://doi.org/10.1109/ISCAS.2019.8702462

Lakshmy, A.V., Rebeiro, C., Bhunia, S., 2022. Fortify: analytical pre-silicon side-channel characterization of digital designs. In: 2022 27Th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 660–665. https://doi.org/10.1109/ASP-DAC52403.2022.9712551

Lee, K., Lu, Z., Yeh, S., 2022a. A secure jtag wrapper for soc testing and debugging. IEEE Access 10, 37603–37612. https://doi.org/10.1109/ACCESS.2022.3164712

Lee, K.J., Liu, C.A., Wu, C.C., 2022b. A dynamic-key based secure scan architecture for manufacturing and in-field ic testing. IEEE Trans. Emerg. Top Comput. 10, 373–385. https://doi.org/10.1109/TETC.2020.3021820

Meza, A., Restuccia, F., Oberg, J., Rizzo, D., Kastner, R., 2023. Security verification of the opentitan hardware root of trust. IEEE Secur. Priv. 21, 27–36. https://doi.org/10.1109/MSEC.2023.3251954

Qin, M., Hu, W., Wang, X., Mu, D., Mao, B., 2019. Theorem proof based gate level information flow tracking for hardware security verification. Comput. Secur. 85, 225–239. https://doi.org/10.1016/j.cose.2019.05.005

Ren, X., Torres, F.P., Blanton, R.D., . Ic protection against jtag-based attacks. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2018, 1–1. https://doi.org/10.1109/TCAD.2018.2802866

Salmani, H., . The improved cotd technique for hardware trojan detection in gate-level netlist. GLSVLSI '22: Proceed. Great Lakes Sympos. VLSI 2022, 449–454.

Salmani, H., 2017. Cotd: reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. IEEE Trans. Inf. Forensics Secur. 12 (2), 338–350. https://doi.org/10.1109/TIFS.2016.2613842

Salmani, H., Tehranipoor, M., Karri, R., 2013. On design vulnerability analysis and trust benchmarks development. In: IEEE 31St International Conference on Computer Design (ICCD), pp. 471–474. https://doi.org/10.1109/ICCD.2013.6657085

Srivastava, A., Das, S., Choudhury, N., Psiakis, R., Silva, P.H., Pal, D., Basu, K., 2024. Scar: power side-channel analysis at rtl level. IEEI Trans. Very Large Scale Integr. (VLSI) Systems 32 (6), 1110–1123.

Sun, H., Yang, Z., Chen, X., Xu, H., Yuan, Z., 2024. Hardware information flow tracking based on lightweight path awareness. Comput. Secur. 147, 104072. https://doi.org/10.1016/j.cose.2024.104072

Witharana, H., Jayasena, A., Whigham, A., Mishra, P., 2023. Automated generation of security assertions for rtl models. J. Emerg. Technol. Comput. Syst 19 (1). https://doi.org/10.1145/3565801

Wu, C., Wang, L., Wen, X., 2006. VLSI Test Principles and Architectures: Design for Testability (Series in Systems on Silicon). Morgan Kaufmann, San Mateo, CA, USA.

Yasaei, R., Chen, L., Yu, S.Y., Faruque, M. A.A., . Hardware trojan detection using graph neural networks. https://doi.org/10.1109/TCAD.2022.3178355

Yasaei, R., Yu, S.Y., Faruque, M. A.A., 2021. Gnn4tj: graph neural networks for hardware trojan detection at register transfer level. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1504–1509. https://doi.org/10.23919/DATE51398.2021.9474174

Yen, C.H., Tsai, J.C., Wu, K.C., 2023. Using path features for hardware trojan detection based on machine learning techniques. In: 2023 24Th International Symposium on Quality Electronic Design (ISQED), pp. 1–8. https://doi.org/10.1109/ISQED57927.2023.10129300

Zhang, N., Lv, Z., Zhang, Y., et al., 2020. Novel design of hardware trojan: a generic approach for defeating testability based detection. In: 2020 IEEE 19Th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE. https://doi.org/10.1109/TrustCom50675.2020.00034

Zhang, S., Wang, S., Wang, J., Zhou, S., Yao, Z., 2022. Quantitative analysis of information leakage hardware trojans in ip cores. In: 2022 9Th International Conference on Dependable Systems and Their Applications (DSA). Wulumuqi, pp. 431–436. https://doi.org/10.1109/DSA56465.2022.00063

Zhang, Y., Quan, H., Li, X., et al., 2018. Golden-free processor hardware trojan detection using bit power consistency analysis. J. Electr. Test.: Theory and Appl. 34 (3), 305–312.

Zhao, Y., Qu, G., Zhang, Q., Li, Y., Li, Z., He, J., 2024. Static gate-level information flow for hardware information security with bounded model checking. In: 2024 IEEE 42Nd VLSI Test Symposium (VTS), pp. 1–7. https://doi.org/10.1109/VTS60656.2024.10538813