Machine Learning-Based Anomaly Detection for Hardware Trojans using Ring Oscillator Networks

Giovanni Cornejo
Dept. of Electrical and Computer Eng
University of Florida
Gainsville, USA
cornejog@ufl.edu

Victor Alonso Hernandez
Dept. of Electrical and Computer Eng
University of Florida
Gainsville, USA
vichdz313@gmail.com

Stephen Singh
Dept. of Electrical and Computer Eng
University of Florida
Gainsville, USA
singhstephen@ufl.edu

***Abstract*—The increasing complexity and global distribution of integrated circuit supply chains have heightened the risk of malicious modifications, known as hardware Trojans (HTs). Detecting these hidden threats is challenging due to the unknown nature of the Trojan's size, type, location, trigger, or payload. Traditional detection methods rely on predefined patterns, limiting their effectiveness against the growing diversity and sophistication of HTs. As Trojans become more varied and sophisticated, these methods struggle to keep pace with the evolving landscape of potential threats. This study employs a machine learning approach to Trojan detection by analyzing frequency variations in a ring oscillator (RO) network implemented in an ASIC. Functioning as power supply monitors, these variations can indicate Trojan activity, as even small voltage dips caused by Trojan-induced switching can affect the RO frequencies. We focus on two distinct cases: (1) using known golden chip data for training and (2) training the classifier with completely unidentified samples, where the labels are unknown. The dataset included 33 chip samples, each containing frequency readings from eight ROs, with two rows of Trojan-free (golden) data and 23 rows of Trojan-inserted data. Cases were evaluated on various sample sizes using anomaly detection-based classifiers including Isolation Forest and SVM + KNN ensemble respectively. Performance was assessed using accuracy, true positive rate (TPR), and true negative rate (TNR). Results suggest that the Isolation Forest excels in minimizing false positives, achieving high accuracy and TNR, but struggles with detecting Trojan-inserted chips. In contrast, the SVM + KNN ensemble excels at detecting Trojans but with more false positives. These findings highlight the trade-offs between Trojan detection sensitivity and accuracy, emphasizing the importance of balancing detection performance in practical applications.**

***Index Terms*—hardware security, hardware Trojan detection, machine learning, anomaly detection, ring oscillator network**

I. Introduction

The rapid growth of integrated circuit (IC) complexity and the globalization of the semiconductor supply chain have introduced significant security risks, particularly the threat of malicious modifications to ICs known as hardware Trojans (HTs). These HTs are notoriously difficult to detect and can lead to severe consequences, such as data breaches, system failures, and compromised security, making it crucial to ensure the integrity of fabricated chips [1].

Traditional methods for detecting HTs have primarily relied on identifying known patterns to categorize and combat Trojans effectively [2]–[4]. While these approaches laid a foundational framework for detection, they face critical limitations. Huang *et al.* [5] highlighted that such methods struggle to cope with the diverse and evolving nature of HTs. The wide variation in Trojan size, type, location, trigger mechanisms, and payloads makes it nearly impossible to catalog every potential threat.

To overcome these challenges, machine learning (ML)-based detection techniques have emerged as a more dynamic and adaptive solution. By leveraging data-driven insights, ML models can learn to identify patterns indicative of HT presence and adapt to various application scenarios [5]. This paradigm shift offers a significant advancement, enhancing hardware security in the face of increasingly sophisticated threats.

Practical applications of ML-based HT detection span both pre- and post-fabrication stages, employing diverse models such as support vector machines (SVM), Naïve Bayes, and Random Forest [6]–[10]. These techniques have been effectively paired with side-channel power analysis to detect HTs by analyzing statistical features of power traces [6]. Additionally, ML-based methods have shown versatility across design stages, from analyzing netlist features like logic fan-ins and shortest paths to gate-level flip-flop inputs [8], to assessing register-transfer level (RTL) features such as gate types and node activity [9]. This flexibility highlights the potential of ML in addressing HT threats at multiple points in the hardware lifecycle.

Anomaly detection, in particular, has shown significant promise in addressing the challenges of HT detection [7], [11]. By identifying deviations from the expected behavior of a "golden" or Trojan-free (TF) chip, anomaly detection becomes especially valuable in scenarios where potential threats are undefined. In this work, we leverage side-channel analysis data obtained from monitoring the behavior of a ring oscillator (RO) network implemented within an application specific integrated circuit (ASIC) to detect HTs. These on-chip ROs allow the detection of frequency variations when a Trojan is activated. This assumption is based on the principle that switching activity from a Trojan or other malicious inclusions can influence the RO frequency within the chip [12]. By identifying these deviations, we aim to detect the presence of HTs.

Despite its promise, anomaly detection techniques face several significant challenges that must be considered:

1) **Process Variations**: Chips inherently exhibit manufacturing variations that can affect their electrical behavior. Fluctuations in properties such as capacitance and threshold voltage can mimic Trojan effects, complicating the identification of genuine threats [13]. These variations influence switching characteristics and introduce noise, potentially masking malicious modifications and reducing detection accuracy.
2) **Environmental Variations**: Environmental factors, such as temperature fluctuations across different regions of a chip, can cause natural frequency variations [12]. These discrepancies may resemble Trojan-induced anomalies, making it more difficult to differentiate between legitimate performance changes and malicious alterations.
3) **Data Limitations**: Machine learning classifiers require robust and representative training data to function effectively. However, obtaining sufficient labeled data that accurately reflects real-world Trojan behavior is a significant challenge. Limited or biased datasets hinder feature selection and reduce model efficacy. Furthermore, the absence of labeled data complicates detection, requiring classifiers to identify subtle deviations from expected behavior without explicit guidance. Techniques that operate without golden reference circuits have been proposed to address this issue [13], [14].

Overcoming these challenges demands a focus on high-quality training data, robust feature engineering, and the development of noise-resilient detection methods. Specifically, there are three possible scenarios designed to mimic real-world conditions: (1) training with both known golden and Trojan-inserted (TI) data, (2) using only limited golden data for training, and (3) training with completely unidentified samples. In this study, we focus on Cases 2 and 3. By investigating these scenarios, we aim to highlight the challenges and limitations of applying machine learning techniques to HT detection in compromised environments. The findings of this study contribute to the ongoing development of robust and reliable machine learning methods for hardware security, providing insights for future research and practical implementations.

The main contributions of this paper are:

- The analysis of an RO network as a novel mechanism for detecting HT-induced anomalies in integrated circuits.
- The implementation and comparison of two machine learning classifiers for HT detection.
- An exploration of how varying the training dataset size impacts the performance of these classifiers.

The remainder of this paper is structured as follows:

- **Section II**: Details the logic of the classifiers, the feature selection process, and the overall detection methodology.
- **Section III**: Presents the results and analysis of Trojan detection for each ML model, with a focus on how training dataset size influences performance.
- **Section IV**: Concludes the study by summarizing the findings and discussing potential directions for future work.

II. Implementation

A. Classifier Selection and Description

This study implements two distinct classification techniques, tailored to the specific data availability scenarios:

1) **Isolation Forest:** Selected for Case 2, Isolation Forest is an unsupervised anomaly detection technique that identifies outliers without requiring labeled data. It constructs decision trees to isolate individual samples, leveraging the principle that anomalies, due to their distinct feature values, are easier to isolate. Trained exclusively on TF samples, it establishes a baseline for detecting TI anomalies based solely on patterns observed in the golden data.
2) **SVM + KNN Ensemble:** Designed for Case 3, where no prior knowledge about the samples (TF or TI) is available, the SVM + KNN ensemble combines the strengths of both algorithms for anomaly detection. Support Vector Machines (SVM) are used to construct a hyperplane that maximizes the margin between classes. The SVM model is tuned via grid search to optimize the kernel type, regularization parameter (C), and kernel coefficient (*gamma*). K-Nearest Neighbors (KNN) serves as a secondary classifier, refining anomaly detection by evaluating proximity metrics. Post-classification refinement through clustering techniques is applied to minimize false positives.

These classifiers were selected to address the unique challenges of each case while ensuring adaptability and maximizing detection performance under the given constraints. The key differences between the classifiers are summarized in Table I.

TABLE I
Comparison of Classification Techniques

Criteria	Isolation Forest (Case 2)	SVM + KNN Ensemble (Case 3)
Training Data	TF Samples	Unidentified Samples
Approach	Unsupervised Anomaly Detection	Hybrid Supervised and Clustering
Strengths	Handles Golden-Only Data	Adapts to Unknown Samples
Weaknesses	Relies on TF Baseline	Higher Computational Cost

B. Algorithm Overview

Case 2: Isolation Forest

1) Extract TF samples from the dataset for training.
2) Train an Isolation Forest model using selected feature subsets (e.g., RO1-RO4, RO5-RO8, or all features).
3) Use the trained model to classify the remaining samples into as normal or anomalous.
4) Evaluate performance metrics (accuracy, precision, recall) across 20 randomized trials.

Case 3: SVM + KNN Ensemble

1) Perform dimensionality reduction using Principal Component Analysis (PCA) to project the data onto a two-dimensional space for better separation.

2) Train SVM using grid search to optimize kernel type, *C*, and *gamma*.
3) Apply KNN for anomaly refinement by detecting borderline cases that lie between class boundaries.
4) Refine false positives by applying K-Means clustering for post-classification adjustments.
5) Evaluate performance metrics (accuracy, precision, recall) across 20 randomized trials.

C. Model Architecture and Parameters

Isolation Forest:

- **Parameters:**
  - **Contamination (`contamination`):** Specifies the proportion of anomalies in the dataset. This parameter determines the decision boundary for classifying anomalies. We set this to `0.2`, assuming that 20% of the samples are TI.
  - **Number of Estimators (`n_estimators`):** Specifies the number of isolation trees used during training. We set this to 1500 to balance computational efficiency and model stability.
  - **Random State (`random_state`):** Ensures reproducibility of model training.
- **Feature Selection:** Experiments were conducted using three feature subsets to assess which feature groupings provided the best detection accuracy:
  1) **RO1-RO4:** Only the first four Ring Oscillator (RO) frequency measurements.
  2) **RO5-RO8:** Only the last four RO frequency measurements.
  3) **All Features:** All eight RO frequency measurements.

tSVM + KNN Ensemble:

- **SVM Parameters:**
  - **Regularization Parameter (`C`):** Balances error minimization and margin maximization. Larger `C` values prioritize classification accuracy, while smaller values focus on a wider margin. Tested values: `0.01, 0.1, 1, 10, 100`.
  - **Kernel:** Defines the type of hyperplane used to separate the classes. We experimented with:
    - ∗ `rbf` (Radial Basis Function): Effective for non-linear boundaries.
    - ∗ `linear`: Simple linear separation.
    - ∗ `poly`: Polynomial kernel for higher-order decision boundaries.
    - ∗ `sigmoid`: Useful for certain non-linear data patterns.
  - **Kernel Coefficient (`gamma`):** Controls the influence of individual points. Larger values lead to tighter decision boundaries, while smaller values produce smoother boundaries. Tested values: `scale, 0.01, 0.1, 1`.
  - **Random State (`random_state`):** Ensures reproducibility of model training.
- **KNN Parameters:**
  - **Number of Neighbors (`n_neighbors`):** Specifies how many nearest neighbors are considered for classification. Smaller values focus on local structure, while larger values broaden the scope. Tested values: `3, 5, 7, 10, 15, 20`..
  - **Distance Metric (`metric`):** Defines the method for measuring distance between samples. Metrics tested:
    - ∗ `euclidean`: Straight-line distance.
    - ∗ `manhattan`: Sum of absolute differences.
    - ∗ `chebyshev`: Maximum absolute difference across dimensions.
    - ∗ `cosine`: Angular distance between vectors.
  - **Weighting Scheme (`weights`):** Determines how neighbors contribute to classification:
    - ∗ `uniform`: Equal contribution from all neighbors.
    - ∗ `distance`: Neighbors closer to the sample have greater influence.
- **Post-Processing with K-Means Clustering:**
  - **Clustering Goal:** After combining SVM and KNN predictions, false positives are refined using K-Means clustering, grouping predictions and identifying a cluster that corresponds to normal samples.
  - **Silhouette Score:** Measures clustering quality. Higher scores indicate well-separated and compact clusters, validating the effectiveness of this post-processing step.

III. Experiments

This section outlines the specific implementation details for our project, including the methodology for identifying HTs for each case. The results for both cases are presented with detailed metrics and visual comparisons.

A. Simulation Parameters

To evaluate our classifiers, we conducted experiments using the following simulation parameters:

- **Data:** RO frequency measurements were extracted from TF and TI chips.
- **Sample Sizes:** Experiments were performed for 6, 12, and 24 training samples, as follows:
  - **Case 2:** Only known TF samples were available for training.
  - **Case 3:** Training samples were completely unidentified (mixture of TF and TI).
- **Metrics:** Classifier performance was evaluated using:
  - True Negative Rate (TNR)
  - True Positive Rate (TPR)
  - False Positive Rate (FPR)
  - False Negative Rate (FNR)
  - Accuracy
  - F1 Score

B. Case 2 Results

For Case 2, where only TF samples were used for training, the Isolation Forest classifier was employed. Table II summarizes the performance metrics for the **All Features** subset with 24 samples.

TABLE II
PERFORMANCE METRICS FOR CASE 2 (ALL FEATURES, 24 SAMPLES)

Metric	**Value**
True Negative Rate (TNR)	0.9241
True Positive Rate (TPR)	0.0417
False Positive Rate (FPR)	0.0759
False Negative Rate (FNR)	0.9583
Accuracy	0.9026
Precision	0.0179
Recall	0.0417
F1 Score	0.0260

Figure 1 presents the confusion matrix for Case 2, showing the distribution of true positives, true negatives, false positives, and false negatives. Additionally, Figure 3 demonstrates the effect of increasing the number of training samples. As shown, increasing the training sample size improves the accuracy of the Isolation Forest classifier. This highlights the critical role that correctly labeled training data plays in enhancing classifier performance.

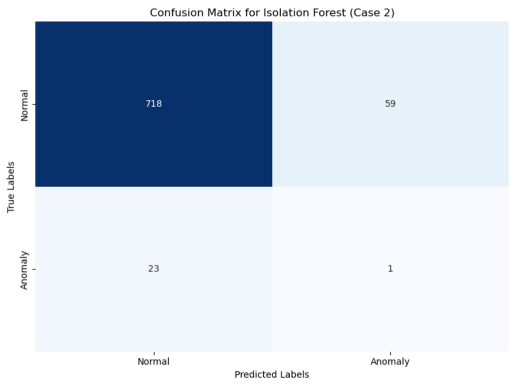

Fig. 1. Confusion Matrix for Case 2 (All Features, 24 Samples)

C. Case 3 Results

For Case 3, where training samples were completely unidentified, the SVM + KNN Ensemble classifier was utilized. Table III provides a summary of the performance metrics for the **All Features** subset with 24 samples.

Figure 2 presents the confusion matrix for Case 3. Furthermore, Figure 3 shows the impact of increasing the number of training samples. As with Case 2, increasing the sample size improves the classifier's accuracy.

D. Algorithm Result Comparison

The results of Case 2 and Case 3 reveal significant differences in classifier performance, as summarized in Table IV:

TABLE III
PERFORMANCE METRICS FOR CASE 3 (ALL FEATURES, 24 SAMPLES)

Metric	**Value**
True Negative Rate (TNR)	0.6441
True Positive Rate (TPR)	0.9769
False Positive Rate (FPR)	0.3559
False Negative Rate (FNR)	0.0231
Accuracy	0.8185
Precision	0.0179
Recall	0.0417
F1 Score	0.8495

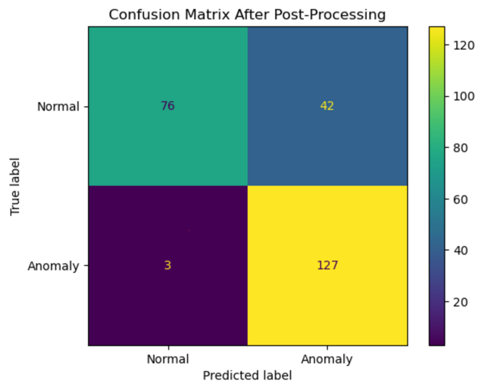

Fig. 2. Confusion Matrix for Case 3 (All Features, 24 Samples)

- **True Positive Rate (TPR):** Case 3 significantly outperforms Case 2, achieving a TPR of 97.69% compared to 4.17%. This indicates that the SVM + KNN Ensemble is highly effective at identifying TI chips.
- **False Positive Rate (FPR):** Case 3 exhibits a higher FPR (35.59%) than Case 2 (7.59%), suggesting that the presence of mixed training samples introduces some ambiguity.
- **Accuracy:** Case 2 achieves a higher accuracy (90.26%) compared to Case 3 (81.85%), likely due to fewer false positives in the Isolation Forest model.

Figure 3 visually compares the accuracy and F1 score across the two cases.

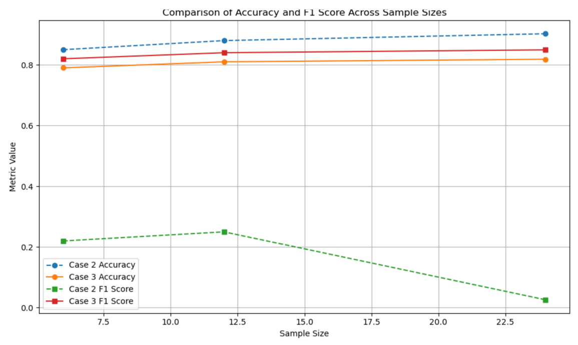

Fig. 3. Accuracy and F1 Score Comparison for Case 2 and Case 3

TABLE IV
COMPARISON OF PERFORMANCE METRICS BETWEEN CASE 2 AND CASE 3

Metric	**Case 2 (Isolation Forest)**	**Case 3 (SVM+KNN Ensemble)**
TNR	92.41%	64.41%
TPR	4.17%	97.69%
FPR	7.59%	35.59%
FNR	95.83%	2.31%
Accuracy	90.26%	81.85%
Precision	1.79%	75.15%
F1 Score	2.60%	84.95%

IV. CONCLUSION

Detecting hardware Trojans in fabricated chips remains a complex and challenging task due to the intricacies of modern integrated circuits and the subtle variations introduced by both manufacturing processes and runtime conditions. The objective of this project was to design machine learning-based classifiers to distinguish between TI and TF chips under different data availability scenarios. By leveraging variations in RO frequencies as a detection mechanism, two approaches were implemented and evaluated: the Isolation Forest for Case 2, where only TF samples were available for training, and an SVM + KNN Ensemble for Case 3, where the training samples were completely unidentified.

In Case 2, the Isolation Forest demonstrated strong performance in identifying TF chips, achieving a TNR of 92.41% and an overall accuracy of 90.26% using 24 samples. However, its TPR was only 4.17%, highlighting its limitations in detecting TI chips due to the absence of Trojan-related data during training. In contrast, the SVM + KNN Ensemble in Case 3 excelled in detecting TI chips, achieving a TPR of 97.69% and an F1 Score of 84.95%, indicating a well-balanced detection capability. However, the model's TNR (64.41%) and FPR (35.59%) suggested challenges in accurately differentiating TF chips compared to the Isolation Forest. These results emphasize the trade-offs between the two approaches: Case 2 prioritized minimizing false positives, while Case 3 was more sensitive to Trojan activity, albeit at the cost of increased false positives.

Overall, the findings reveal the strengths and weaknesses of each approach depending on the availability of labeled data and the specific detection objectives. While the Isolation Forest is well-suited for scenarios where high confidence in identifying normal chips is critical, the SVM + KNN Ensemble provides a more effective solution in contexts that require heightened sensitivity to Trojan activity. Future work could explore hybrid models or advanced feature engineering to balance these trade-offs more effectively, leveraging the strengths of both approaches. Additionally, investigating Case 1, where both golden and Trojan-infected data are available for training but the type of Trojan is unknown, could offer valuable insights into improving detection accuracy and robustness when more data becomes accessible. This project emphasizes the importance of tailoring machine learning techniques to the unique constraints of hardware Trojan detection, paving the way for more intelligent and reliable security mechanisms in chip design.

ACKNOWLEDGMENT

We would like to express our deepest gratitude to Dr. Farimah Farahmandi, our professor, for their invaluable guidance, support, and insights throughout the course of this project. Their expertise and encouragement have been instrumental in helping us achieve our objectives.

We also extend our sincere thanks to the University of Florida for providing the resources and learning environment that made this work possible. The knowledge and skills gained during this project will undoubtedly contribute to our future endeavors in the field of hardware security and machine learning.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 15–19.

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[4] S. Moein, T. A. Gulliver, F. Gebali, and A. Alkandari, "A new characterization of hardware trojans," *IEEE Access*, vol. 4, pp. 2721–2731, 2016.

[5] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A survey on machine learning against hardware trojan attacks: Recent advances and challenges," *IEEE Access*, vol. 8, pp. 10 796–10 826, 2020.

[6] R. Gayatri, Y. Gayatri, C. Mitra, S. Mekala, and M. Priyatharishini, "System level hardware trojan detection using side-channel power analysis and machine learning," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020, pp. 650–654.

[7] A. Kulkarni, Y. Pino, and T. Mohsenin, "Svm-based real-time hardware trojan detection for many-core platform," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016, pp. 362–367.

[8] J. Yang, Y. Zhang, Y. Hua, J. Yao, Z. Mao, and X. Chen, "Hardware trojans detection through rtl features extraction and machine learning," in *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2021, pp. 1–4.

[9] Y. Wang, P. Liu, X. Han, and Y. Jiang, "Hardware trojan detection method for inspecting integrated circuits based on machine learning," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 432–436.

[10] S. Kelly, M. Zhang Xuehui, Tehranipoor, and A. Ferraiuolo, "Detecting hardware trojans using on-chip sensors in an asic design," *Journal of Electronic Testing*, vol. 31, pp. 11–26, 2015.

[11] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori, "Hardware trojan detection using changepoint-based anomaly detection techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2706–2719, 2019.

[12] S. Kelly, X. Zhang, M. Tehranipoor, and A. Ferraiuolo, "Detecting hardware trojans using on-chip sensors in an ASIC design," *J. Electron. Test.*, vol. 31, no. 1, pp. 11–26, Feb. 2015.

[13] H. Salmani, "Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2017.

[14] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2939–2948, 2017.