Reinforcement Learning for Hardware Security: Opportunities, Developments, and Challenges

Satwik Patnaik, Vasudev Gohil, Hao Guo, and Jeyavijayan (JV) Rajendran
Electrical & Computer Engineering, Texas A&M University, College Station, Texas, USA
{satwik.patnaik, gohil.vasudev, guohao2019, jv.rajendran}@tamu.edu

***Abstract*—Reinforcement learning (RL) is a machine learning paradigm where an autonomous agent learns to make an optimal sequence of decisions by interacting with the underlying environment. The promise demonstrated by RL-guided workflows in unraveling electronic design automation problems has encouraged hardware security researchers to utilize autonomous RL agents in solving domain-specific problems. From the perspective of hardware security, such autonomous agents are appealing as they can generate optimal actions in an unknown adversarial environment. On the other hand, the continued globalization of the integrated circuit supply chain has forced chip fabrication to off-shore, untrustworthy entities, leading to increased concerns about the security of the hardware. Furthermore, the unknown adversarial environment and increasing design complexity make it challenging for defenders to detect subtle modifications made by attackers (a.k.a. hardware Trojans). In this brief, we outline the development of RL agents in detecting hardware Trojans, one of the most challenging hardware security problems. Additionally, we outline potential opportunities and enlist the challenges of applying RL to solve hardware security problems.**

***Index Terms*—Reinforcement Learning, Hardware Security**

I. Introduction

A. Reinforcement Learning

Reinforcement learning (RL) enables a computing system (a.k.a. *agent*) to learn using a trial-and-error approach by exploring and exploiting the underlying environment. The agent interacts with the *environment* and gradually realizes how to take improved *actions* to maximize the total expected *reward* [1]. Over time, the agent learns to take optimal actions sequentially with limited or no prior knowledge regarding the environment. Formally, RL solves the underlying problem by modeling it as a Markov decision processes (MDP). An MDP is defined to be a 5-tuple $(\mathcal{X}, \mathcal{A}, P, R, \gamma)$: $\mathcal{X}$ is the set of states; $\mathcal{A}$ is the set of actions; $P(x_{t+1}|a_t, x_t)$ is the probability that action a_t in state x_t leads to state x_{t+1}; the reward function $r_{t+1} = R(x_t, a_t)$ gives reward r_{t+1} after taking action a_t in state x_t; the discount rate γ, $0 \leq \gamma \leq 1$, discounts future rewards to their present value.

B. Surge of RL in Solving EDA Problems

The pronounced capability shown by RL-based algorithms in reducing human effort and providing optimal solutions compared to heuristic and algorithm-driven computer-aided design (CAD) tools have led researchers in the electronic design automation (EDA) community to apply RL to domain-specific problems [2]. For instance, Google demonstrated their RL-based approach could generate optimal chip floorplans in under six hours compared to human-generated floorplans that take months [3]. Researchers have utilized RL for problems ranging from logic synthesis [4], optimization of parameters for placement [5], global routing [6], sizing of transistors [7], gate-sizing to achieve timing closure [8], etc.

C. Globalization of IC Supply Chain and Associated Threats

On the other hand, design companies employ the continual shrinking of technology nodes to develop faster and low-power systems, which necessitates access to advanced technology nodes. As the financial implications of commissioning, owning and maintaining a state-of-the-art technology node (e.g., 3nm) are exorbitant [9], design houses outsource the fabrication of integrated circuits (ICs) to third-party, off-shore, potentially untrustworthy foundries [10]. Outsourcing of fabrication leads to security concerns ranging from piracy of the design intellectual property (IP), unauthorized overproduction of ICs, to insertion of malicious circuits (a.k.a. hardware Trojans (HTs)) [10]. Security researchers widely recognize the insertion of HTs as a pernicious threat. This is because HTs inserted during fabrication cannot be removed, and the damage incurred by HTs has far-reaching consequences [11].

D. RL for Hardware Security: Opportunities

Cybersecurity researchers have used RL agents to develop promising approaches for some security problems, including intrusion detection [12], fuzzing [13], [14], and developing secure cyber-physical systems [15]. With the latest advancements in RL algorithms and from the perspective of hardware security, such autonomous RL agents are appealing as they can efficiently navigate high-dimensional search space and generate optimal actions in an unknown adversarial environment. However, using RL for hardware security problems is in its infancy, and researchers have primarily focused on employing RL for detection of HTs [16], [17], [18], [19] barring recent works on employing RL for insertion of HTs [20], [21].

II. Detecting Hardware Trojans using RL

Logic-testing and side-channel analysis are the two primary classes of techniques used to detect HTs.

Logic testing-based techniques apply test patterns and monitor the outputs to measure deviations from the expected, *i.e.,* golden output [17], [18]. These techniques suffer from three limitations: (i) generating test patterns that activate all possible combinational and sequential triggers are challenging, (ii) detecting HTs that are devoid of any triggering mechanism (e.g., always-on) or HTs without payloads, and (iii) requiring improvement in controllability and observability through design modifications, resulting in area and power overheads.

Side channel-based techniques monitor side-channel information (e.g., power, delay) instead of the output response. These techniques do not require the HT to be fully activated or propagate its impact to the primary outputs, rendering it a practical approach over logic testing-based HT detection techniques. These techniques suffer from two limitations: (i) multiple golden ICs are required to create the golden signature, and (ii) the impact of HTs on side-channels can be overshadowed by environmental noise (e.g., process variations). Next, we

summarize efforts by researchers in utilizing RL to detect HTs inserted by an untrustworthy foundry.

Test Generation using RL for Delay-based Side-Channel Analysis. Pan *et al.* [16] proposed an RL-based test generation method for delay-based HT detection. Unlike existing methods that rely on the delay difference of a few gates, this approach utilizes critical path analysis to generate test vectors that maximizes the side-channel sensitivity. The authors sub-divide the problem of generating effective test patterns to detect HTs into: (i) how to find a good initial test for triggering the HT, and (ii) how to efficiently generate proper succeeding tests to switch triggering signals.

TGRL. Pan *et al.* [17] attempt to solve the issues about scalability and detection accuracy by proposing a logic testing-based approach using a combination of testability analysis and RL. The authors train the RL model using a stochastic learning scheme that generates test patterns, continuously improving itself to cover as many suspicious nodes as possible. TGRL drastically reduces the test generation time (6.54× on average) and detects a vast majority of the Trojans in all benchmarks (96% on average), a significant improvement (14.5% on average) compared to state-of-the-art techniques.

DETERRENT. Gohil *et al.* [18] aim to find a minimal set of test patterns that can activate all combinations of rare nets.[1] This is based on the premise that a single test pattern can simultaneously activate multiple combinations of rare nets. The authors define *compatible* rare nets if there exists a test pattern that can activate all the rare nets (in a given set) simultaneously and develop an RL agent that generates maximal sets of compatible rare nets [18]. DETERRENT achieves two orders of magnitude reduction (169×) in the number of test patterns over TGRL [17] while improving accuracy.

AdaTest. Chen *et al.* [19] proposed AdaTest that leverages RL and integrates adaptive sampling to prioritize test samples that provide more information for HT detection. Such a process progressively generates test patterns with high 'reward' values. Although AdaTest demonstrates up to two orders of test generation speedup and two orders of test set size reduction compared to the prior works, they do not showcase any results with regards to other RL-based approaches [17], [18].

III. Challenges in Applying RL

To use RL effectively for hardware security problems, one needs to identify whether the underlying problem can be mapped to an MDP and whether there exists a notion of a sequential decision-making process in the actions.

Problem Complexity. The RL problem needs to be formulated so that the training process is efficient. The complexity of the training process is directly related to the complexity of the underlying game, which is measured either with the state-space complexity or game tree complexity.[2]

Evaluation Time. The reward computation time for RL should not be the bottleneck; in other words, the computation time for the reward should be quick and inexpensive. As opposed to EDA problems, where at times, the reward can be evaluated/computed by running processes within commercial tools, which can be computationally expensive, one needs to figure out methods to accurately characterize the underlying environment to generate rewards quickly. To address the challenge of long evaluation time, off-policy and offline RL methods can be applied, which do not require real-time interaction with the environment. Methods such as offline characterization and pre-computation of data can be performed based on domain-specific hardware security problems.

Generality. The current RL approaches for HT detection produce test patterns for individual benchmarks using separate agents. Whether a trained model can be transferred to unseen data remains a challenge. Designing model architectures that can work on unseen data using principles of transfer learning and/or meta-learning are promising directions.

[1]The problem of determining a minimal set of test patterns is a variant of the set-cover problem, which is NP-complete.

[2]State-space complexity is defined as the number of legal game positions obtainable from the initial position of the game, while the game tree complexity is defined as the number of leaf nodes in the solution search tree of the initial position of the game [22].

IV. Conclusion

In this brief, we summarized the efforts undertaken by hardware security researchers in utilizing RL to address one of the consequential hardware security problems, *i.e.,* the detection of HTs inserted by an untrustworthy foundry. Then, we outline some general challenges that need to be solved when applying RL to other hardware security problems. These challenges and research directions hopefully would inspire future research for employing RL in hardware security, both in the development of attacks and defenses.

Acknowledgments

The work was partially supported by the National Science Foundation (NSF CNS–1822848 and NSF DGE–2039610).

References

[1] R. S. Sutton *et al.*, *Reinforcement learning: An introduction*. MIT press, 2018.
[2] A. F. Budak *et al.*, "Reinforcement Learning for Electronic Design Automation: Case Studies and Perspectives," in *Proc. ASPDAC*, 2022, pp. 500–505.
[3] A. Mirhoseini *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.
[4] A. Hosny *et al.*, "DRiLLS: Deep Reinforcement Learning for Logic Synthesis," in *Proc. ASPDAC*, 2020, pp. 581–586.
[5] A. Agnesina *et al.*, "VLSI Placement Parameter Optimization using Deep Reinforcement Learning," in *Proc. ICCAD*, 2020, pp. 1–9.
[6] H. Liao *et al.*, "A Deep Reinforcement Learning Approach for Global Routing," *Journal of Mechanical Design*, vol. 142, no. 6, 2020.
[7] Z. Zhao *et al.*, "Deep Reinforcement Learning for Analog Circuit Sizing," in *Proc. ISCAS*, 2020, pp. 1–5.
[8] Y.-C. Lu *et al.*, "RL-Sizer: VLSI Gate Sizing for Timing Optimization using Deep Reinforcement Learning," in *Proc. DAC*, 2021, pp. 733–738.
[9] R. Zafar, "TSMC's Total 3nm Investment Will Equal At Least $ 23 Billion," https://wccftech.com/tsmc-3nm-investment-23-billion-project-end/, 2021, [Online; last accessed 2-May-2022].
[10] M. Rostami *et al.*, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proc. of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.
[11] K. Yang *et al.*, "A2: Analog Malicious Hardware," in *Proc. IEEE Symposium on Security and Privacy (S&P'16)*, 2016, pp. 18–37.
[12] G. Caminero *et al.*, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96–109, 2019.
[13] K. Böttinger *et al.*, "Deep Reinforcement Fuzzing," in *Proc. IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 116–122.
[14] D. Wang *et al.*, "SyzVegas: Beating Kernel Fuzzing Odds with Reinforcement Learning," in *Proc. 30th USENIX Security*, 2021, pp. 2741–2758.
[15] H.-D. Tran *et al.*, "Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control," *TECS*, vol. 18, no. 5, pp. 1–22, 2019.
[16] Z. Pan *et al.*, "Test Generation using Reinforcement Learning for Delay-based Side-Channel Analysis," in *Proc. ICCAD*, 2020, pp. 1–7.
[17] ——, "Automated Test Generation for Hardware Trojan Detection using Reinforcement Learning," in *Proc. ASPDAC*, 2021, pp. 408–413.
[18] V. Gohil *et al.*, "DETERRENT: Detecting Trojans using Reinforcement Learning," in *Proc. 59th ACM/IEEE DAC*, 2022, pp. 697–702.
[19] H. Chen *et al.*, "AdaTest: Reinforcement Learning and Adaptive Sampling for On-chip Hardware Trojan Detection," *arXiv:2204.06117*, 2022.
[20] V. Gohil *et al.*, "ATTRITION: Attacking Static Hardware Trojan Detection Techniques Using Reinforcement Learning," in *ACM SIGSAC CCS (to appear)*, 2022.
[21] A. Sarihi *et al.*, "Hardware Trojan Insertion Using Reinforcement Learning," in *Proc. of GLSVLSI*, 2022, pp. 139–142.
[22] H. J. Van Den Herik *et al.*, "Games solved: Now and in the future," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 277–311, 2002.