Protecting Cyber-Physical Systems via Vendor-Constrained Security Auditing with Reinforcement Learning

Nan Wang*, Kai Li*, Lijun Lu*, Zhiwei Zhao*, Zhiyuan Ma†
*School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China
†Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai, China

***Abstract*—Hardware Trojans may cause security issues in cyber-physical systems (CPSs), and recently proposed mutual auditing frameworks have helped build trustworthy CPSs with untrustworthy devices by requiring neighboring devices from different vendors. However, this may cause severe multi-vendor integration challenges, such as expensive, hard-to-maintain, and insufficient vendors to purchase devices. In this work, we improve the mutual auditing framework by maintaining the security of the CPSs with fewer vendors. First, the vendor-constrained security auditing framework is introduced to enhance the security of the CPS network with limited vendors, where side auditing detects the hardware Trojan collusion between neighboring nodes and infected node isolation stops the spread of active HTs. Second, a multi-agent cooperative reinforcement learning-based method is proposed to assign devices with proper vendors in the context of security auditing, and it provides solutions with a minimized number of offline nodes due to the HT infection. The experimental results show that our proposed method reduces the number of vendors needed by 40.95%, and only causes an increment of 0.39% infected nodes.**

***Index Terms*—Cyber-physical systems, hardware Trojan, security, vendor constraints, reinforcement learning**

I. Introduction

Cyber-physical systems (CPSs) are networks of devices embedded with sensing, computing, and communication capacities, and the proliferation of the deployment of third-party intellectual property circuits in CPSs has exposed these devices to a host of security vulnerabilities [1]. Many CPSs take over safety- and security-critical functions, such as autonomous vehicles/drones and smart cities, and these systems are vulnerable to hardware Trojan (HT) attacks due to the absence of safeguards for confidentiality, integrity, and availability [2]. HT attacks on CPS devices are intended to affect normal operations or take control of wireless transceivers [3], which may cause catastrophic consequences to the CPS network, such as device failure and denial-of-service attacks [1].

Many HT countermeasures have been developed for robust hardware design, testing, and verification, but most of them cannot guarantee the detection of all possible HTs in CPS devices [4]. Therefore, runtime monitoring is considered to be an important complementary method for detecting HTs [5], and many methods have been proposed to detect HT attacks on CPS devices at runtime [6]–[8]. However, traditional HT detection methods often require golden models, making them less adaptable to unknown HTs [9].

To improve the adaptability to unknown HTs, machine learning (ML)-based approaches, such as self-organizing maps [10], support vector machines [11], and local outlier factors [12], are adopted to enhance the security of CPS devices, with features extracted from real-time signals. The target is to improve the HT detection accuracy with limited ML computations, so that the approaches can be deployed in resource-constrained devices [13]. However, the detection of advanced HTs, such as A2, still remains as a serious issue [14].

It is difficult to guarantee the trustworthiness of CPS devices because sophisticatedly implanted HTs may still escape these detection schemes. Given this fact, researchers have also developed a mutual auditing framework [15], [16] to defend CPS networks against HT collusion at runtime, which enables designers to build trustworthy wireless communication channels without worrying about the security problems of devices. However, this framework requires all neighboring devices to come from different vendors, and might lead to severe multi-vendor integration challenges. These challenges include expensive, complex, hard-to-maintain, and insufficient vendors to purchase devices [17], making the traditional auditing framework potentially impractical in some scenarios.

In this work, we adopted the same CPS model presented in [15], [16], and modeled the number of available vendors as the vendor constraints. To address the multi-vendor integration challenges, a vendor-constrained security auditing framework is proposed to build trustworthy CPSs with untrustworthy devices, and both information leakage and HT collusion can be detected and protected. The contributions are summarized as follows.

1) To reduce the number of vendors needed, side auditing is introduced to the auditing framework, which helps to build trustworthy communications between nodes from the same vendor under the auditing of the third node coming from a different vendor.
2) The node with active HTs is an infected node, and once a node is detected as infected, the immediate node isolation is conducted before HT triggers are widely spread. Both the infected nodes and those with high infection probabilities are isolated from the network.
3) Vendor assignment of devices in the context of security auditing is solved using a multi-agent cooperative reinforcement learning-based method, and the number of offline nodes due to the HT infection is minimized.

978-3-9826741-0-0/DATE25/© 2025 EDAA

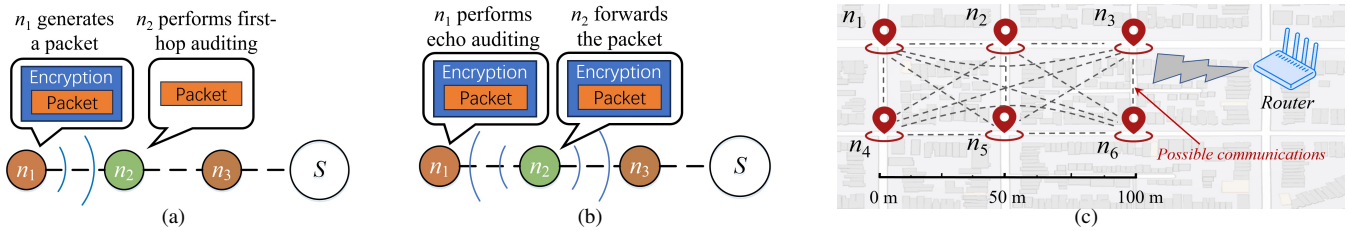


Fig. 1. Example of mutual auditing. (a) First-hop auditing: n_2 decrypts the packet generated by n_1 and checks the security guard bits. (b) Echo auditing: n_1 performs a bit-by-bit comparison of the original packet and the packet forwarded by n_2. (c) Example of CPS network in the context of mutual auditing.

4) The network simulations were conducted using NS-3 [18]. Compared to the mutual auditing scheme, our scheme requires fewer vendors with almost negligible impacts on the security and performance of the network.

II. Background

A. Threat Model in CPSs

The main integrity threats against CPS hardware are HTs, which are manipulations of hardware circuits with malicious intents [2]. The HTs in devices can be either self-triggered when the triggering conditions are satisfied or receive HT triggers sent by its infected neighbors from the same vendor. These HTs may escape the HT detection techniques, and a device with active HTs may utilize the wireless transmitter to collude with other devices in the same CPS, causing catastrophic consequences to the network [16]. This work focuses on the HT attacks inside the network and aims to defend against information leakage and node collusion among the nodes.

Information leakage: A node with active HTs may try to leak key information, such as collected data or secret keys, to the HT implanter secretly, and many nodes need to collude to establish secret communication channels to leak information. As a result, the HT implanter may successfully obtain confidential information without any physical access to the device, and little footprint may be left.

Node collusion or HT triggering: A node with active HTs may collude with other nodes, or trigger the hibernating HTs in other nodes. These two processes are similar, and an HT triggering example is given as follows. The nodes n_1, n_2, and n_3 are part of a smart traffic light system, where the devices in n_1 and n_2 are from the same vendor, while the device in n_3 is from a different vendor. The HTs in n_1 and n_2 become active when the counter reaches 800 or receives "800" as input. If the HT in n_1 is activated, it may embed trigger "800" in its message and in turn trigger the HT in n_2. However, the HT in n_3 has a different triggering condition, and it cannot be triggered by such a message. This triggering effect can propagate to the entire network in a short time.

For the packages transmitted in the CPS, the package payload is the "ideal" place to embed the key information or HT triggers, and HT can alter any bit or insert any information in the payload. In this work, the side-channel information is not utilized to embed the key information. It is based on the fact that a large number of gates are required to utilize the side-channel information to embed key information, and such attempts in circuits or PCB designs may be easier to be detected by the existing HT detection methods. Furthermore, the accuracy of such transmission in a complicated system cannot be guaranteed [19].

B. Countermeasures to HTs in CPSs

An HT is a malicious modification to the IC, which results in undesired behaviors of the applied electronic devices [6]. The detection techniques for HTs in devices can be classified into two categories: 1) logic testing-based and 2) side-channel signal-based. The logic testing-based techniques depend on rare conditions to activate HTs [7]. Whereas, side-channel signal-based techniques involve observing the effect of HTs on several physical parameters such as transient current, leakage, and path delay [8]. In both approaches, traditional methods compared the outputs of devices under test with the outputs of a golden model, making them less adaptable to unknown HTs. Recently, researchers have started to use ML-based methods to analyze the abnormalities of signals collected from the target devices to improve the HT detection accuracies [10]–[12]. However, it is still difficult to devise a single Trojan detection technique that is applicable to all the varieties of HTs [6], and therefore, the research of designing trustworthy CPS with untrustworthy devices becomes necessary.

Recently, a mutual auditing framework proposed in [15], [16] secured CPSs from HT collusion at runtime, which enables designers to build trustworthy wireless communication channels with untrustworthy devices. This auditing framework consists of *first-hop auditing* and *echo auditing*. First-hop auditing decrypts the received packet and verifies the security guard bits to check if the packet is maliciously tampered with (see the example in Fig. 1(a), where colors of nodes represent different vendors), and echo auditing verifies the correctness of packets forwarded by the next-hop node (see the example in Fig. 1(b)). *Auditor* and *auditee* devices are from different vendors to prevent HT collusion, and it is based on the fact that the probability of HTs implanted by different vendors having the same trigger is quite low.

III. Vendor-Constrained Security Auditing Framework

A. Motivation

Traditional mutual auditing framework requires that the neighboring devices must be purchased from different vendors to establish trustworthy communications, and a large number of vendors might be needed even in a small CPS network. An

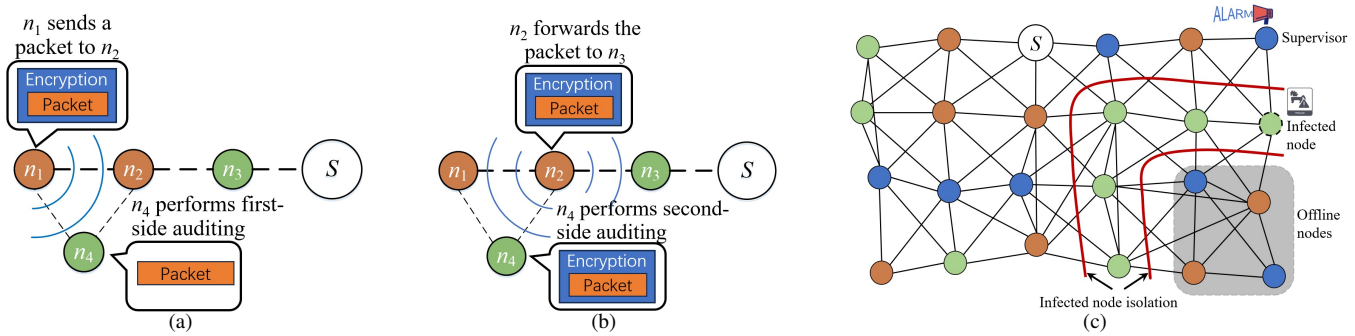


Fig. 2. Examples of side auditing and infected node isolation. (a) First-side auditing: n_4 decrypts the packet generated by n_1 and checks the security guard bits. (b) Second-side auditing: n_4 compares the packet forwarded by n_2 with the packet received from n_1. (c) Example of infected node isolation.

example of a smart traffic CPS network in Fig. 1(c) gives the reason. The maximum transmission range (*MTR*) is set to 200 meters, and two devices can establish a communication channel once the distance between them is within the *MTR*. There are 6 devices deployed in a 100×50 m^2 area, and communications can be established between any two devices. To secure this CPS network using mutual auditing, these 6 devices must be purchased from 6 different vendors, and this also leads to multi-vendor integration challenges although the scale of the network is small. These challenges include hard-to-maintain, excess budget, and complex device compatibility. In particular, in many CPS applications, such as military, devices are only from a relatively small number of vendors [20], and this makes the mutual auditing framework potentially impractical in such scenarios.

B. Vendor-Constrained Security Auditing

In this work, the number of vendors available to CPS designers is modeled as the vendor constraints. The vendor-constrained security auditing framework is proposed following the design of mutual auditing, which comprises the following three components: first-hop auditing, echo auditing, and side auditing.

Side auditing allows two neighboring nodes from the same vendor under the auditing of the third node (named as *supervisor*) coming from a different vendor, and this helps to reduce the number of vendors required. Let n_s be the supervisor of n_i and n_{i+1}, and each packet sent from n_i to n_{i+1} can also be received by n_s. n_s performs the *first-side auditing* by decrypting the packet and checking the security guard bits. After n_{i+1} forwards the packet, n_s also receives this packet and performs the *second-side auditing* by comparing this packet with the original packet received from n_i. Examples of side auditing are shown in Figs. 2(a) and 2(b).

C. Infected Node Isolation

Side auditing allows neighboring nodes from the same vendors, and a cluster is a set of neighboring nodes coming from the same vendor. If any node in a cluster is infected, this cluster is named as an infected cluster. Although the supervisor of side auditing can detect malicious attempts, it cannot stop the HTs in n_i triggering the hibernating HT in n_{i+1}, and this indicates that the HT triggering process can hardly be terminated within an infected cluster. Therefore, the infected cluster should be isolated from the remaining CPS network to stop the spread of HT affections.

Infected node isolation can minimize the damage caused by the infected nodes, and keep the remaining CPS network functioning normally. Fig. 2(c) presents an example of infected node isolation. When a supervisor detects an infected node, it immediately broadcasts an alarm to every node in the CPS. This alarm has the highest priority for transmition, while normal messages, including those containing HT triggers, have to wait in the sending queue. This can be realized by many existing techniques such as time-sensitive networking (TSN), where the flows with the highest priority obtain deterministic transmission times with ultralow latency [21]. Therefore, most nodes can receive the alarm before the malicious message arrives. After receiving the alarm, all the neighbors of the infected cluster will drop the messages sent by the nodes inside the infected cluster. However, the normal nodes surrounded by the infected cluster also become offline because all the paths to the server have been cut off (see the four nodes in the bottom right corner in Fig. 2(c)) .

D. Effectiveness Analysis

The flow of the vendor-constrained security auditing framework is described as follows, and the nodes are assigned to vendors using the method described in Section IV. During the runtime auditing stage, if a node n_i and its next-hop node n_{i+1} are from different vendors, n_{i+1} performs the first-hop auditing and then n_i performs the echo auditing. If n_i and n_{i+1} are from the same vendor, conducting first-hop and echo auditing between them can no longer be trusted, and instead, their supervisor conducts the side auditing on the packets received from n_i and n_{i+1}.

Effectiveness of vendor-constrained security auditing framework is explained as follows. If a transmitter is malicious and its packet is tampered with or inserted with HT triggers (either in plaintext or after encryption), these attempts are detected by first-hop auditing or first-side auditing. If a relay tampering with a packet or inserting HT triggers into a packet payload, these attempts are detected by echo auditing or second-side auditing. The first-side auditing and second-side auditing can be regarded

as substitutes for the first-hop auditing and echo auditing when the two neighboring nodes are from the same vendor; therefore, the workload of the vendor-constrained security auditing is similar to the traditional mutual auditing.

IV. Vendor Assignment with Reinforcement Learning

The vendor assignment in the context of vendor-constrained security auditing is solved using a multi-agent cooperative advantage actor-critic (MAC-A2C)-based method, which is described in this section.

A. Vendor Assignment Evaluation

The CPS network is modeled as an undirected graph $G = (V, E)$, where V is the node set consisting of all nodes in CPS and E is the edge set consisting of all possible communications between nodes. The process of vendor assignment is to determine the vendor of each node following the security auditing constraints, which are: 1) two neighboring nodes should be assigned to different vendors; 2) if two neighboring nodes are assigned to the same vendor, there must be a common neighbor coming from a different vendor and acts as the supervisor.

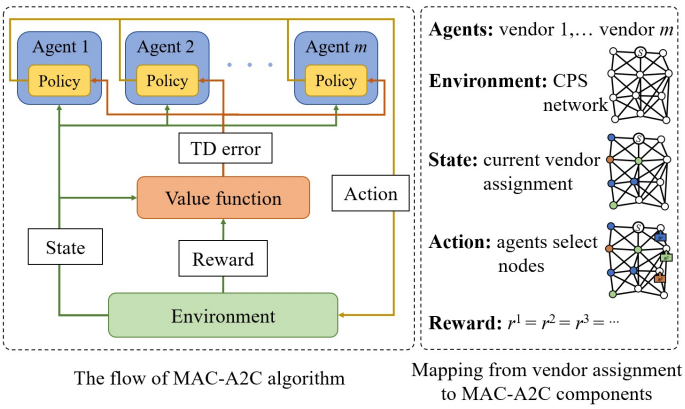
The goal of vendor assignment is to maximize the number of online nodes if any node is infected by HTs. The CPS designers might not have prior knowledge of implanted HTs, and therefore, we assumed that each node has the same probability of being infected with an HT. Let $OL(n_i)$ be the percentage of online nodes if n_i is infected, and the number of online nodes can be calculated using breadth-first search with the server to be the searching root after removing the infected cluster. The average percentage of online nodes after detecting an infected node is denoted as $OL(G)$, which can be calculated as follows.

$$OL(G) = \frac{1}{|V|} \sum_{n_i \in V} OL(n_i) \tag{1}$$

$OL(G)$ can be used to evaluate the results of vendor assignments, and the goal is to find a vendor assignment with the maximum $OL(G)$.

B. MAC-A2C-based Algorithm for Vendor Assignment

MAC-A2C is a class of reinforcement learning algorithms that combines both policy-based and value-based methods. The actors are responsible for selecting actions, and the critic evaluates the actors' actions. During training, the actors use an advantage function D_t to update its policy, where D_t is defined as the difference between the estimated value function and the actual reward obtained from taking an action in a given state. The critic is also updated using the temporal difference (TD) error between estimated value functions. At each time step, the agents observe the current state of the environment and select actions based on their current policies. The state is then transitioned into a new state and every agent obtains the same reward [22]. Based on the workflow of MAC-A2C, the learning paradigm is presented by mapping vendor assignment tasks onto five crucial components, which are shown in Fig. 3.



The flow of MAC-A2C algorithm

Mapping from vendor assignment to MAC-A2C components

Fig. 3. The workflow of MAC-A2C algorithm, and the mapping from vendor assignment to MAC-A2C components.

Agent: Agent usually refers to the object interacting with the environment. In our vendor assignment problem, each agent represents a vendor, and there are a total of m agents, where m is the vendor constraint. Each agent i holds a candidate node list NL_i, which consists of all the nodes that it can select. The node selected by an agent means that this node is assigned to the corresponding vendor, and the candidate node lists of all agents will be updated each time an agent selects a node.

Environment: The CPS network is mapped into the environment, which describes the nodes and their connections.

State: The current vendor assignment is mapped as a state, which can be further utilized in reward computation. Let S_t be the joint state at time t, where $S_t = [s_t^1, s_t^2, \cdots, s_t^m]$ and s_t^i is the state of agent i at time t.

Action: For the agent i at time t, its action a_t^i is either choose one node from its NL_t^i or choose nothing, and its action space is $\mathsf{a}_t^i = [nop, NL_t^i]$, where nop means no operation is conducted. The action space of all agents at time t is $\mathcal{A}_t = [\mathsf{a}_t^1, \mathsf{a}_t^2, \cdots, \mathsf{a}_t^m]$, and the joint action of all agents is $A_t = [a_t^1, a_t^2, \cdots, a_t^m]$.

Reward: In general, reward value is the most important feedback information from the environment that describes the effect of the latest action. There are three types of actions according to the action results. *Proceeding*: agents select some nodes and the vendor assignment process can continue; *halt*: all the agents do not choose any node in this step; *failure assignment*: there exists an unassigned node that does not belong to the candidate node lists of all agents. The halt action stops the vendor assignment in the current state, and failure assignment action leads to an infeasible solution. Therefore, the rewards of halt and failure assignment actions are set to 0. For the proceeding action, its reward is evaluated by the average percentage of online nodes if the selected nodes are infected. Finally, the reward for action A_t is given as follows.

$$r(A_t) = \begin{cases} \frac{1}{|V_{A_t}|} \sum\limits_{n_i \in V_{A_t}} OL(n_i), & A_t == \textit{proceeding action}; \\ 0, & \textit{otherwise}; \end{cases} \tag{2}$$

where V_{A_t} is the set of nodes selected by A_t.

At each time step t, the actor network outputs a probability distribution over the action space, and let $\pi_{\theta^i}(a_t^i | s_t^i)$ be the policy of agent i, where π_{θ^i} are the parameters of the policy. $V_\phi(s_t)$ is the value function, which is the expected total reward from the current state. $Q_\phi(S_t, \mathcal{A}_t)$ is the centralized critic that estimates

the value function of joint state and action space. The goal of training MAC-A2C model is to learn the policy parameters θ^i for each agent i, and Algorithm 1 describes the details.

Algorithm 1 Training of MAC-A2C Model

Input: Number of agents m; learning rates $\alpha_\theta, \alpha_\phi$; number of episodes K;
Output: Policy parameters θ^i for each agent i;

Initialize policy parameters θ^i and value function parameters ϕ;
for each episode $k = 1$ to K **do**
  Set initial state $S_t \leftarrow S_0$;
  for each time step t in an episode **do**
    for each agent i **do**
      Determine the action space $\mathcal{A}^i$ at state s_t^i;
      Select an action $a_t^i \sim \pi_{\theta_i}(a_t^i|s_t^i, \mathcal{A}^i)$;
    end for
    Execute the joint action $A_t = [a_t^1, \ldots, a_t^m]$;
    Calculate the reward $r(A_t)$ from the new state S_{t+1};
    Calculate the TD error $\delta_t = r(A_t) + \gamma Q_\phi(S_{t+1}, A_{t+1}) - Q_\phi(S_t, A_t)$, where γ is the discount factor;
    Update the value function parameters: $\phi \leftarrow \phi + \alpha_\phi \delta_t \nabla_\phi Q_\phi(S_t, A_t)$;
    for each agent i **do**
      Calculate its advantage function $D_t^i = Q_\phi(S_t, A_t) - V_\phi(S_t)$;
      Update $\theta^i \leftarrow \theta^i + \alpha_\theta \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t^i, \mathcal{A}^i) D_t^i$;
    end for
  end for
end for
Return the policy parameters θ_i for each agent i;

V. Experimental Results

A. Experimental Setup

The network simulations were conducted using NS-3 [18], and implemented with the C++ language. The MAC-A2C model was conducted using PyTorch (1.13.1) with CUDA toolkit (11.7). As suggested in [23], the learning rates were set as $\alpha_\theta = 0.001$ for the actors and $\alpha_\phi = 0.015$ for the critic. All the experiments were conducted on a host machine with Intel I9-7960X CPU, 128GB RAM, and RTX 3090 GPU. The square mesh networks are tested following the configurations given in [15], where every two neighboring nodes are first separated by 100 meters, and then the node positions are randomly moved. Each network contains a server which is placed at the center. Table I lists the configurations of the wireless CPS networks.

Our proposed scheme is evaluated against three other schemes: a baseline scheme (**BLS**) without security enhancement; a local outlier factor (LOF)-based scheme (**LOFS**) [12], which detects the anomalous behaviors of CPS devices using the LOF method based on time-series power waveforms; and a mutual auditing scheme (**MAS**) [15], which conducts only first-hop and echo auditing and requires every neighboring node from different vendors.

TABLE I
Experimental configurations

Parameters	Configurations
Channel model	WIFI_STANDARD_80211b
Medium access protocol	CSMA
Transport layer protocol	UDP
Routing algorithm	AODV
Maximum bandwidth	100 MB per second
Averaged packet size	300 B
Queue size	20
Average # of packets generated	100 per second

B. Vendor Assignment Results

The effectiveness of the MAC-A2C-based vendor assignment method is first presented by assigning vendors in a 7×7 mesh network. MTR is set to 240 meters and MAS needs 5 vendors for all nodes, but the vendor constraint is set to 3 in our method. Figs. 4(a) and 4(b) give the $OL(G)$ and value loss in the training process, respectively, and these results indicate that the model reaches its optimal at about episode 5000. The implemented MAC-A2C model is capable of devising vendor assignment in the context of vendor-constrained security auditing, and Fig. 4(c) gives a vendor assignment result.

The numbers of vendors required in multiple networks are also evaluated (see Fig. 4(d)), with two MTR configurations (120 and 240 meters). A larger MTR means that nodes with longer distances can be connected, and more communications can be established in the network. The results show that our scheme requires an average of 40.95% fewer vendors than MAS in all tested benchmarks. When the MTR is set to 120 meters and 240 meters, our scheme reduces the number of vendors by 33.33% and 48.57% compared to that of MAS, respectively.

C. Performance Analysis of the Mesh Networks

Then, the network evaluations after 5 minutes of simulation are presented in Table II, with MTR=240 meters. The vulnerability of the CPS is first evaluated, and we assume that each node in the network has an HT that is initially hibernating. The HT we focus on can be either self-triggered or triggered upon receiving an HT trigger sent by an infected node from the same vendor. We simulate the process of self-triggered HT by setting a self-triggering probability for each packet generated, and this probability is set to 0.01% [15]. A node with an active HT constantly embeds an HT trigger in every packet it generates, and all the HTs in the neighbors from the same vendor will be triggered. In the simulations, LOFS detects 87.8% of all anomalous behaviors caused by HTs, and we assume that the node with detected HTs will no longer send HT triggers.

The results in columns "# of vendors" and "ratio of infected nodes" show that our scheme requires fewer vendors than MAS, with an average increment of 0.39% infected nodes. The active HTs in MAS are all self-triggered, but in our proposed auditing scheme, additional HTs are triggered by receiving HT triggers sent from neighbors. The reason is that the supervisor cannot stop the transmission of triggers between nodes from the same vendor. On average, 20.76% of nodes are infected in LOFS, and it is because some active HTs successfully escape the detections

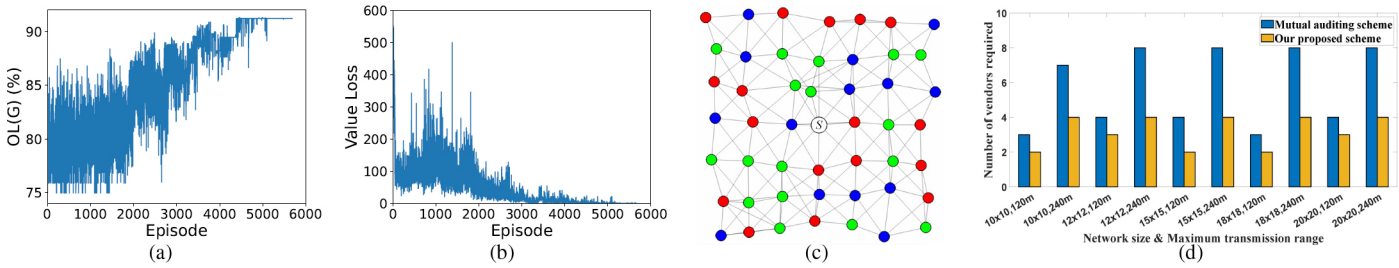


Fig. 4. Vendor assignment results of our proposed method. (a) $OL(G)$ in the training process. (b) Loss in the training process. (c) Vendor assignment results of a 7×7 mesh network, with MTR=240 meters. (d) Number of vendors required in different networks.

and trigger the HTs in the neighboring nodes. Furthermore, LOFS has to deploy a power monitoring device for each node to get the power waveforms.

The impacts on the latency (time interval between packet generation and its arrival at the server), packet arrival rate, and throughput are shown in columns "Lat, AR, TP", respectively. The delay of transmitting packages between neighbors is ignored in the experiments, and the latency can be used to evaluate the average computational cost of each method. Monitoring and analyzing the power waveforms in LOFS cause additional delays in each node, and both MAS and our proposed method need to perform message encryptions and verifications. Therefore, these methods increases the networks' latency and degrades the throughput compared to BLS. Meanwhile, our proposed scheme obtains slightly improved latency, packet arrival rate and throughput compared to those of MAS. The reason is that the first-hop auditing is no longer performed in our scheme if two neighboring nodes are from the same vendor, which saves the decryption and verification time.

Finally, the percentages of infected nodes are also compared at different times of simulation, with MTR=240 meters. Fig. 5(a) and Fig. 5(b) show the percentages of infected nodes in 10×10 and 20×20 mesh networks, respectively, within 20 minutes of simulation. Because BLS does not provide any protection to nodes, and if any node is infected, all the nodes from the same vendor will be infected within a short time. LOFS detects anomalous behaviors of nodes, but some infected nodes might still escape the detection. Our proposed scheme obtains almost equivalent results as MAS, indicating that the side auditing detects the malicious attempts and the infected node isolation stops the spread of active HTs. Furthermore, our proposed security auditing framework successfully reduces the number of vendors without bringing additional risks to the network if compared to MAS, and the number of infected nodes increases quite slowly, which gives the system managers enough time to handle such threats.

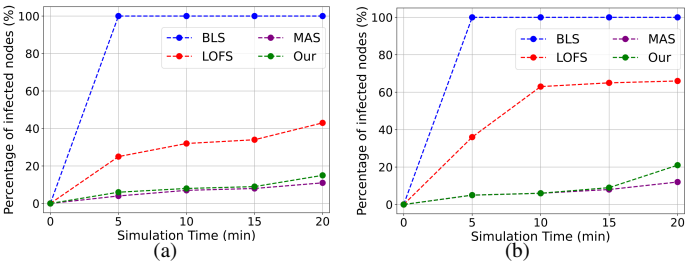


Fig. 5. Percentage of infected nodes at different simulation times. (a) Simulations in a 10×10 mesh network. (b) Simulations in a 20×20 mesh network.

TABLE II
EVALUATIONS OF SQUARE MESH NETWORKS

Network sizes	Schemes	# of vendors	ratio of infected nodes	Lat (ms)	AR (%)	TP (KB/s)
10×10	BLS	1	100%	22.23	99.86	28.48
	LOFS	1	25.32%	35.43	99.65	25.53
	MAS	7	4.13%	34.93	99.67	28.41
	Our	4	5.02%	33.97	99.59	28.39
12×12	BLS	1	100%	27.90	98.72	28.17
	LOFS	1	18.23%	42.87	99.45	27.54
	MAS	8	5.47%	40.31	98.59	28.09
	Our	4	5.86%	39.76	97.39	27.75
15×15	BLS	1	100%	33.51	99.57	28.39
	LOFS	1	19.47%	51.31	99.34	26.68
	MAS	8	2.67%	47.62	98.65	28.11
	Our	4	2.82%	44.19	98.75	28.14
18×18	BLS	1	100%	44.70	98.35	28.00
	LOFS	1	27.53%	63.07	98.71	26.02
	MAS	8	2.45%	56.74	94.21	26.83
	Our	4	2.63%	55.98	98.31	28.00
20×20	BLS	1	100%	57.47	98.07	27.96
	LOFS	1	13.27%	70.53	95.73	26.89
	MAS	8	2.13%	63.93	97.45	27.78
	Our	4	2.48%	63.65	98.07	27.70
Avg.	BLS	1	100%	37.16	98.91	28.20
	LOFS	1	20.76%	52.64	98.58	26.53
	MAS	7.8	3.37%	48.71	97.71	27.84
	Our	4	3.76%	47.51	98.42	28.00

VI. Conclusion

To mitigate the severe multi-vendor integration challenges caused by mutual auditing in CPSs, this work first introduces side auditing and infected node isolation to the auditing framework to protect CPSs with a limited number of vendors, and then proposes an MAC-A2C-based vendor assignment method for this vendor-constrained security auditing framework. Experimental results show that our scheme needs 40.95% fewer vendors than the mutual auditing scheme, with only 0.39% extra nodes infected, and almost no impact on the network's latency, arrival rate, and throughput.

VII. Acknowledgment

This work was supported by the National Key R&D Program of China under Grant 2022YFD2000400.

References

[1] A. Dhavlle, R. Hassan, M. Mittapalli, and S.M.P. Dinakarrao, "Design of hardware Trojans and its impact on CPS systems: A comprehensive survey," in *Proc. International Symposium on Circuits and Systems*, pp. 1-5, 2021.

[2] F. Regazzoni and I. Polian, "Securing the hardware of cyber-physical systems," in *Proc. Asia and South Pacific Design Design Automation Conference*, pp. 194-199, 2017.

[3] A.R. Díaz-Rizo, H. Aboushady, and H.-G. Stratigopoulos, "Leaking wireless ICs via hardware Trojan-infected synchronization," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp.3845-3859, 2023.

[4] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010-1038, 2021.

[5] B.J. Mohd, S. Abed, T. Hayajneh, and M.H. Alshayeji, "Run-time monitoring and validation using reverse function(RMVRF) for hardware Trojans detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2689-2704, 2021.

[6] K. Lingasubramanian, R. Kumar, N.B. Gunti, and T. Morris, "Study of hardware Trojans based security vulnerabilities in cyber physical systems," in *Proc. International Conference on Consumer Electronics*, pp. 1-6, 2018.

[7] K. Ma, C. Amarnath, A. Chatterjee, and J.A. Abraham, "Secure control loop execution of cyber-physical devices using predictive state space check," in *Proc. International Symposium on Quality Electronic Design*, pp. 1-8, 2023.

[8] S. Narasimhan, D. Du, R. Chakraborty, S. Paul, F.G. Wolff, C.A. Papachristou, K. Roy, and W. Bhunia, "Hardware Trojan detection by multiple parameter side-channel analysis," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2183-2195, 2013.

[9] P. Zhan, H. Shen, S. Li, and H. Li, "BGNN-HT: bidirectional graph neural network for hardware Trojan cells detection at gate level," in *Proc. International Symposium on Circuits and Systems*, pp. 1-5, 2023.

[10] S. Guo, J. Wang, Z. Chen, Y. Li, and Z. Lu, "Securing IoT space via hardware Trojan detection," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11115-11122, 2020.

[11] D. Utyamishev and I. Partin-Vaisband, "Real-time detection of power analysis attacks by machine learning of power supply variations on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 45-55, 2020.

[12] K. Hisafuru, K. Takasaki, and N. Togawa, "An anomalous behavior detection method for IoT devices based on power waveform shapes," in *Proc. International Symposium on On-Line Testing and Robust System Design*, vol. 10, no. 4, pp. 1837-1853, 2022.

[13] F. Khalid, S. R. Hasan, S. Zia, O. Hasan, F. Awwad, and M. Shafique, "MacLeR: machine learning-based runtime hardware Trojan detection in resource-constrained IoT edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3748-3761, 2020,

[14] D. Deng, Y. Wang, and Y. Guo, "Novel design strategy toward A2 Trojan detection based on built-in acceleration structure" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp.4496-4509, Dec. 2020.

[15] C. Liu, P. Cronin, and C. Yang, "Securing cyber-physical systems from hardware Trojan collusion," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 655-667, 2020.

[16] C. Liu, P. Cronin, and C. Yang, "A mutual auditing framework to protect IoT against hardware Trojans," in *Proc. Asia and South Pacific Design Design Automation Conference*, pp. 194-199, 2017.

[17] E.L. Rouzic, O. Renais, J. Meuric, *et al.*, "Operator view on optical transport network automation in a multi-vendor context," *Journal of Optical Communications and Networking*, vol. 14, no. 6, pp. 11-22, 2022.

[18] Network Simulator NS-3, https://www.nsnam.org/.

[19] K.S. Subramani, A. Antonopoulos, A.A. Abotabl, A. Nosratinia, and Y. Makris, "INFECT: INconspicuous FEC-based Trojan: A hardware attack on an 802.11a/g wireless network," in *Proc. International Symposium on Hardware Oriented Security and Trust*, pp. 90-94, 2017.

[20] J.P. Wong, O. Younossi, C.K. LaCoste, P.S. Anton, A.J. Vick, G. Weichenberg, and T.C. Whitmore, "Improving Defense Acquisition: Insights from Three Decades of RAND Research," RAND Corporation, 2022.

[21] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G," *Proceeding of the IEEE*, vol. 107, no. 6, pp. 944-961, 2019.

[22] T. Theodoropoulos, D. Kafetzis, J. Violos, A. Makris, and K. Tserpes, "Multi-agent deep reinforcement learning for weighted multi-path routing," in *Proc. The workshop on Flexible Resource and Application Management on the Edge*, pp. 7-11, 2023.

[23] D.P. Kingma and J. Ba, "Adam: A method for stochasic optimization," in *Proc. International Conference on Learning Representations*, pp. 1-15, 2015.