A Golden-Free Approach to Detect Trojans in COTS Multi-PCB Systems

Animesh Basak Chowdhury, *NYU Tandon School of Engineering, Brooklyn, NY, 11201, USA*

Anushree Mahapatra, *Innatera Systems, 2289EX, Rijswijk, The Netherlands*

Yang Liu, Prashanth Krishnamurthy, Farshad Khorrami, and Ramesh Karri, *NYU Tandon School of Engineering, Brooklyn, NY, 11201, USA*

Untrusted third parties in commercial-off-the-shelf (COTS) printed circuit board (PCB) supply chains may poison PCBs with hardware, firmware, and software implants. Hence, we focus on detection of malicious implants in PCBs. State-of-the-art hardware Trojan detection methods require a golden PCB system/model to detect malicious implants and do not scale to large-scale COTS PCB systems. We map a COTS PCB system to a graph and propose a golden-free methodology comprising a graph-based mathematical construction on "node" and "edge" equivalences, and clustering of identical nodes and paths and validation of hypothesized statistical properties on measured sidechannel data. We evaluate the methodology on a multi-PCB testbed with hierarchically networked PCB devices and several types of Trojans.

Modern supply chain for commercial-off-the-shelf (COTS) printed circuit board (PCB) systems is vulnerable (e.g., alleged Big Hack[1]) to malicious modifications (e.g., insertion of hardware, firmware, software, interconnect Trojans) causing harmful effects such as malfunctions and performance/security degradations. A system architect creates blueprint of complex COTS PCB systems under joint design manufacturing (JDM) model and outsources to third-party vendors for manufacturing the integrated system. The system architect does not have a "golden" integrated system.

INTRODUCTION

Malicious third-party manufacturers may inject Trojans that survive post-manufacture testing and are challenging to detect. We present a graph-approach using multiple sidechannels to detect Trojans in a system without requiring a golden system. The defender is unaware of the Trojan and its behavior. Our contributions include the following:

- A hypothesis that sidechannel invariances in identical/similar devices expose Trojans in multi-PCB systems. Validate hypothesis by statistical analysis of similarity across sidechannels by observing *node invariances*.
- A hypothesis that sidechannel invariances of communication characteristics expose anomalous connections among same communication interfaces and specifications, *path invariances*.
- Fusing sidechannels and communication channels plus fuzzing (test codes, hardware knobs).

Node and *path invariances* expose Trojans even without golden COTS systems. The primary novelty is the framework for formulation of node and path invariances based on design information in absence of golden data and anomaly detection based on invariances using clustering and statistical hypothesis testing.

Background

Trojan attacks and defenses on PCBs[1,2] have been reported including counterfeits, components with unadvertised radio links,[1] and modifications of inter-trace spacings.[2,3] Defenses against PCB Trojans compare measurements such as delays to a golden PCB.[4] Extant methods do not offer real-time detection and

0272-1732 © 2023 IEEE
Digital Object Identifier 10.1109/MM.2023.3300713
Date of publication 3 August 2023; date of current version 28 August 2023.

cannot detect Trojans inserted as a firmware/software patch during system operation into an integrated system of heterogeneous PCBs.

Sidechannel fingerprinting of ICs has been studied to obviate knowledge of golden IC design. Agrawal at al.[5] destructively test ICs from the same family to obtain sidechannel fingerprints and validate for consistency with untrusted ICs post-manufacturing. While golden-free, destructive testing is expensive. Further, Liu at al.[6] and Jin and Makris[7] present golden-free methods by fingerprinting various sidechannels of IC simulations. Our approach is domain-agnostic and, in contrast, uses sidechannel fingerprints of PCBs and communications between PCBs, as a low-cost golden-free technique scaling to multi-PCB systems. Sidechannels have been used to detect malware in COTS systems (e.g., thermal imagery,[8] CPU temperatures and fan acoustic emissions,[9] HPCs[10]) path delays,[7] bit power consistency.[11] We combine sidechannels in a golden-free graph framework to detect Trojans in multi-PCB systems.

Threat Model

We consider a JDM model where a commercial enterprise designs a distributed system (e.g., data center, distributed compute platform) and outsources manufacturing/integration to possibly untrusted parties and supply chain (adversary). Complex, distributed supply chains of multi-PCB systems introduce Trojan risks. The designer is trusted. Integrators (attackers) in supply chains can insert Trojans and ship back to the enterprise. Trojans can include machine-in-the-middle (MiTM) attacks through insertion of malicious components, substitution of components with counterfeits, and software/firmware Trojans on processors. The defender is assumed to have access to system design but does not have a golden integrated system. The defender can collect sidechannel measurements from the system to attempt to detect Trojans by validating against system design information. Sidechannel measurements are assumed to be trustworthy. The challenge addressed is to detect anomalies without a golden baseline.[12]

The proposed method is golden-free and does not require golden samples for detection of Trojans, including MiTM, counterfeits, hardware Trojans, and firmware/software Trojans. Trojans are detected by flagging violations of expected invariances in sidechannel signals from different components. An attack that modifies all components in a system with a Trojaned version will not result in invariance violations detectable within the system and may need external components/data to detect. Nevertheless, with distributed supply chains in large-scale COTS multi-PCB systems, it is unlikely for adversaries to corrupt all components in the system.

THE DEFENDER CAN COLLECT SIDECHANNEL MEASUREMENTS FROM THE SYSTEM TO ATTEMPT TO DETECT TROJANS BY VALIDATING AGAINST SYSTEM DESIGN INFORMATION.

PROBLEM OVERVIEW AND GRAPH MODELING

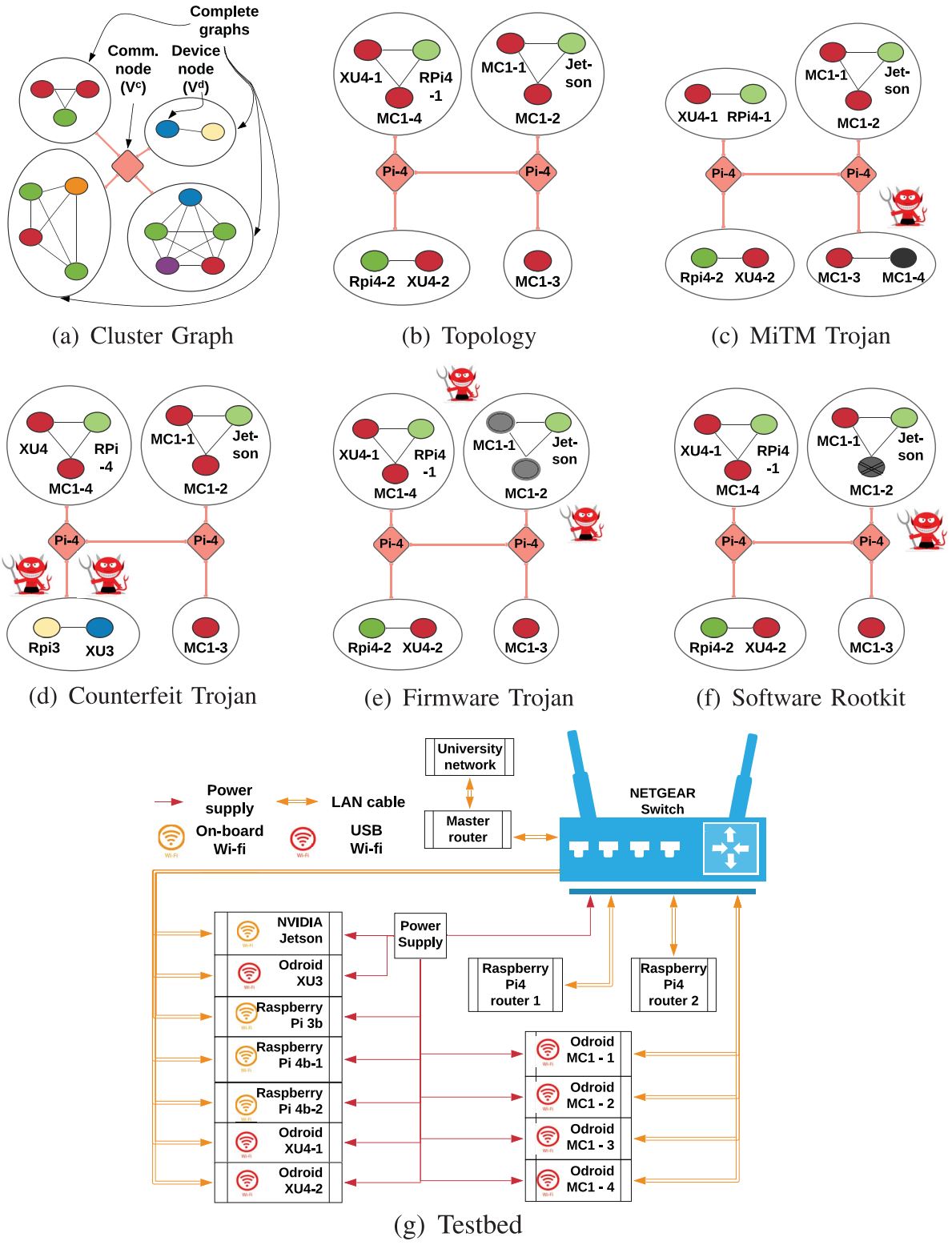
We developed a golden-free baseline model and anomaly detector framework for a multi-PCB COTS system. Problem statement: Given system design information (graph topology, offline information of PCB components) and online access to system, detect Trojans by identifying structural anomalies in graph nodes and paths that would be expected to behave similarly.

NEVERTHELESS, WITH DISTRIBUTED SUPPLY CHAINS IN LARGE-SCALE COTS MULTI-PCB SYSTEMS, IT IS UNLIKELY FOR ADVERSARIES TO CORRUPT ALL COMPONENTS IN THE SYSTEM.

Graph Model of a COTS PCB System

Consider COTS PCB system graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is the node set, and $\mathcal{E}$ the edge set. A node is a PCB and edge is connectivity between PCBs. Connectivity is defined by adjacency matrix $\mathbf{A} \in Z^{N \times N}$, where $[A]_{i,j} = 1$, if $(v_i, v_j) \in \mathcal{E}$. In our studies, the PCBs either perform specific tasks (dedicated PCBs) or are routers to communicate between dedicated PCBs. We classify nodes into 1) *device nodes* v^d performing computations and 2) *communication nodes* (routers) v^c. A PCB node v_i^d ($v_i \in \mathcal{V}$) associates with one communication node v_j^c ($v_j \in \mathcal{V}$).

We form *cluster graphs* of PCB nodes and connect each cluster graph to a communication node. All PCBs in a cluster graph use point-to-point communication without intermediate routers. Figure 1(a) shows a cluster graph of PCB nodes and a router communication node. The cluster graph formed around communication node v_i^c is $\Pi(v_i^c)$. Communication nodes interconnect cluster graphs. Figure 1(a) shows a network of PCB and



(a) Cluster Graph (b) Topology (c) MiTM Trojan

(d) Counterfeit Trojan (e) Firmware Trojan (f) Software Rootkit

(g) Testbed

FIGURE 1. Graph model of multi-PCB system. (a) Cluster graph around communication nodes. (b) Graph $\mathcal{G}$. (c)–(f) Trojans in hardware, firmware, software, interconnects [we consider: (c) MiTM snooping communications, (d) counterfeits, (e) firmware, and (f) library rootkit]. (g) Testbed.

TABLE 1. Sidechannels supported by PCBs.

Sidechannels		Single board PCB computers in testbed					
		Odroid			Raspberry		NVIDIA
		XU3	XU4	MC1	4b	3b	Jetson
Instns	Total	✓	✓	✓	✓	✓	✓
	Load	✗	✗	✗	✓	✓	✗
	Store	✗	✗	✗	✓	✓	✗
	Branch	✓	✓	✓	✓	✓	✓
Branch	taken	✗	✗	✗	✗	✗	✗
	mispredict	✓	✓	✓	✓	✓	✓
Cycles		✓	✓	✓	✓	✓	✓
L1 cache	d-cache miss	✓	✓	✓	✓	✓	✓
	i-cache miss	✓	✓	✓	✓	✓	✓
	load miss	✗	✗	✗	✓	✓	✗
	store miss	✗	✗	✗	✓	✓	✗
	d-cache access	✓	✓	✓	✓	✓	✓
	i-cache access	✓	✓	✓	✓	✓	✓
	d-cache read	✗	✗	✗	✓	✓	✗
	d-cache write	✗	✗	✗	✓	✓	✗
L2 cache	d-cache miss	✓	✓	✓	✓	✓	✓
	i-cache miss	✗	✗	✗	✗	✗	✓
	store miss	✓	✓	✓	✓	✓	✓
	d-cache access	✓	✓	✓	✓	✓	✓
Buffer misses	data trans	✓	✓	✓	✗	✗	✗
	instn trans	✓	✓	✓	✗	✗	✗
Power	CPU	✓	✗	✗	✗	✗	✓
	DRAM	✓	✗	✗	✗	✗	✗
	Total	✓	✓	✓	✓	✓	✓
Thermal	CPU	✓	✓	✓	✗	✗	✓
	Total	✓	✓	✓	✓	✓	✓
Link	setup	✓	✓	✓	✓	✓	✓
	round-trip	✓	✓	✓	✓	✓	✓
	throughput	✓	✓	✓	✓	✓	✓

communication nodes. $\mathcal{G}$ is an interconnection of cluster graphs formed around nodes.

The defender extracts the graph from the system design. Since the defender is trusted and has access to the system design, the extracted graph (which is based on system design and not hardware integration) is trusted.

Sidechannels

Communication sidechannels between devices are measured using a message passing interface (MPI) framework: connection setup time, latency, and throughput.

Hardware performance counters (HPCs) measure low-level microarchitecture events in processors (Table 1). Trojans (hardware, firmware, and software) impact HPC measurements. We use perf (https://perf.wiki.kernel.org/index.php) and PAPI (https://icl.utk.edu/papi) to monitor HPCs.

Power consumption can be monitored using onboard/external sensors. Trojans may impact power consumption. Thermal fluctuations across PCBs are sensitive to workload on PCBs and peripherals and can expose a Trojan when it is active for a substantial duration.

Similarity Metrics

We collect sidechannels on PCBs (nodes of $\mathcal{G}$) and connections between PCBs (edges of $\mathcal{G}$) and compare their statistical measures relative to expected equivalences based on design information. We use three similarity metrics to quantify equivalence properties on nodes and edges.

Bhattacharyya distance (ρ) measures divergence (limits 0 and 1 are distinguishable and identical, respectively) between two probability distributions $P(X)$ and $Q(X)$ over random variable X as follows:

$$\rho(P,Q) = \int_a^b \sqrt{P(X)Q(X)}\, dX$$

where $[a,b]$ is the range of values X can take.

% deviation of average (σ_{pd}) measures deviation of sample mean (μ) of a multivariate distribution P_{target} against a reference P_{ref} as follows:

$$\sigma_{\text{ref}}^{P_{\text{target}}} = \frac{\mu^{P_{\text{target}}} - \mu^{P_{\text{ref}}}}{\mu^{P_{\text{ref}}}} \times 100\%.$$

Dynamic time warping (DTW) measures the distance between two discrete time-series signals $X(n)$ and $Y(n)$ of equal/unequal length by finding an alignment. $c_p(X,Y)$ of alignment P is

$$c_p(X,Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{n_l})$$

where $c(x_{n_l}, y_{n_l})$ is cost of lth step of L-length alignment P. $\text{DTW}(X,Y)$ between X and Y is minimal total cost of optimal alignment $p*$ among all alignments as follows:

$$\text{DTW}(X,Y) = c_{p*}(X,Y) = \min\{c_p(X,Y) |\ p \text{ is an alignment}\}.$$

GOLDEN-FREE TROJAN DETECTION

While the defender has no access to the golden integrated PCB or corresponding sidechannel data, they have design information (e.g., PCB schematics, specifications of devices, and communication interfaces), which can yield expected system properties based on consistency characteristics, relative to sidechannels on and between PCBs. For a cluster graph of a multi-PCB system [Figure 1(a)], its topology yields homomorphic paths expected to have similar properties in communication sidechannels. The proposed Trojan detection approach uses both on-node (e.g., HPCs, power, and thermal) and inter-node sidechannels (e.g., communication-based measurements of setup time, latency, and throughput). The approach is based on the concept that in multi-PCB systems (effectively "systems of systems") in which multiple PCBs and/or inter-PCB links are identical in the intended design, expected equivalence properties can be modeled as expected node and path invariances even in absence of golden hardware. The specific invariance properties in a given multi-PCB system depend on which PCB nodes and/or inter-node links are expected to be identical based on design information. Any deviations from expected node and path invariances flag anomalies indicating modifications.

Since all sidechannels are noisy to different extents depending on the system/environment, the method addresses noise in the invariance evaluations: 1) Instead of relying on single samples or exact matches, we use statistical measures. 2) Defender-specified thresholds for flagging invariance violations are calibrated based on noise levels seen in different sidechannels.

We devise an algorithmic approach to validate or invalidate properties: Use offline information to color PCB nodes and collect sidechannel measurements → Bucket connection paths → Collect sidechannel measurements → Check satisfaction of properties.

To detect Trojans in a golden-free setting, the defender runs secret test codes with controlled fuzzing on tunable knobs (e.g., CPU frequency, CPU/IO/memory workloads and cache use) and collects sidechannels. For the attacker, these codes and fuzzing knobs are moving targets. We use invariance hypotheses to point

to anomalous components (hardware, firmware, software) or connections. Test codes to collect sidechannel measurements are run on processors on PCBs via a Secure Socket Shell command-line.

Node Coloring

After mapping the system to graph $\mathcal{G}$, the goal is to identify suspicious behavior. The designer knows intended system design and knows $\mathcal{G}$. We color similar nodes (similar system-on-chip, peripherals, networking interfaces) with the same color. In Figure 1(b), Odroid PCBs XU4 and MC1 are colored red, and Raspberry Pi4 and NVIDIA Jetson are green. The set of $v_d \in \mathcal{V}$ is partitioned into the same m color subsets. Routers (used for communication) are not colored.

Bucketing of Paths

We find identical paths in graph $\mathcal{G}$ by enumerating pairwise connections and applying Algorithm 1. Paths are identical for two pairwise connections if source and destination nodes have the same color and connections have the same path length. We collect communication sidechannels from pairwise connections in buckets and check their properties for golden-free detection of anomalies.

Algorithm 1: Bucketing Algorithm

Input: COTS graph G, colored nodes $V_C = \{V_{C1}, ...V_{C_m}\}$
Output: Connection buckets B, Connection set S
$B \leftarrow \phi$; $S \leftarrow \phi$;
for *edge* $< V_i^d, V_j^d > \in \mathcal{E}$ **do**
  $C_{\text{source}} \leftarrow C_k$, if $V_i^d \in V_{C_k}$ # Assign color to source node
  $C_{\text{dest}} \leftarrow C_l$, if $V_j^d \in V_{C_l}$ # Assign color to destination node
  $P_{i,j} \leftarrow Dijkstra(V_i^d, V_j^d)$
  path-length $\leftarrow \text{length}(P_{i,j})$
  $conID_{i,j} \leftarrow < C_{\text{source}}, C_{\text{dest}},$ path-length$>$
  if $\exists\ \text{con}ID_{i,j} \in S$ **then**
    $b_{\text{con}ID_{i,j}} \leftarrow b_{\text{con}ID_{i,j}} \bigcup P_{i,j}$, $b_{\text{con}ID_{i,j}} \in B$
  else
    $S \leftarrow S \bigcup \text{con}ID_{i,j}$; $b_{\text{con}ID_{i,j}} \leftarrow P_{i,j}$;
    $B \leftarrow B \bigcup b_{\text{con}ID_{i,j}}$
  end
end
return B, S

Invariance Properties

Following node coloring and path bucketing, we have sets of devices with identical characteristics and sets of buckets with identical paths. To detect anomalies, system properties from design information are inferred on sidechannels of the same color nodes and paths in the same bucket.

Property 1 (Path invariance): For each bucket b_k in connection buckets B, paths in b_k should exhibit similar communication sidechannels measurements (SC), i.e.,

$SC_{P_m} \equiv SC_{P_n} \forall P_m, P_n \in b_k, b_k \in B$.

Corollary: In Figure 1(b), based on communication specifications, we color XU4, MC1 red and Raspberry Pi-4 and NVIDIA Jetson green. We identify the bucket with red source node and green destination node that are one hop (RG_1). There are five connections: P_1: 'MC1-4 → Rpi4-1, P_2: XU4-1 → Rpi4-1, P_3: XU4-2 → Rpi4-2, P_4: MC1-1 → Jetson and P_5: MC1-2 → Jetson. Since $P_1, ..., P_5$ are in the same bucket, sampling distribution of connections sidechannels (S: setup time, L: latency, T: throughput) should be statistically invariant. We use ρ to measure similarity of sidechannel distribution of S, L, and T. Given a pair of connections, an instance where $\rho <$ threshold η is flagged anomalous. This path invariance is: $\rho(L_{P_m}, L_{P_n}) > \eta_L \wedge \rho(T_{P_m}, T_{P_n}) > \eta_T \wedge \rho(S_{P_m}, S_{P_n}) > \eta_S\ \forall P_m, P_n \in RG_1$, where η_L, η_T, and η_S are defender-specified threshold Bhattacharyya distances.

WE COLLECT COMMUNICATION SIDECHANNELS FROM PAIRWISE CONNECTIONS IN BUCKETS AND CHECK THEIR PROPERTIES FOR GOLDEN-FREE DETECTION OF ANOMALIES.

Property 2 (Node Invariance): Device nodes with the same color should exhibit similar sidechannel characteristics (SC):

$$SC_{V_i^d} \equiv SC_{V_j^d} \forall V_i^d, V_j^d \in V_{C_k}, V_{C_k} \in V_C.$$

Corollary: Odroid MC1 nodes have the same color. We use percentage deviation σ_{pd} to measure similarity of sidechannels. HPCs ($\text{HPC}_k, k \in [1, m]$) and thermal sidechannel (H) data are collected by running the same code on MC1 nodes. Node invariance holds when σ_{pd} of multivariate sidechannel data (thermal, HPC) for nodes of the same color are $<$ *threshold percentage* δ:

$$\forall V_i^{\text{MC1}}, V_j^{\text{MC1}}; i, j \in [1, 4]$$
$$p = [\text{HPC}_k, H] \text{ for } V_i^{\text{MC1}}$$
$$q = [\text{HPC}_k, H] \text{ for } V_j^{\text{MC1}}$$
$$\sigma_q^p < \delta.$$

Node invariances arise since color equivalences correspond to similarity of underlying nodes implying

expectation of similarity of sidechannel characteristics. Path invariances arise since paths in each bucket have similar inter-node properties (both end nodes and inter-node communication properties) implying expectation of similarity of inter-node communication sidechannel characteristics.

Dynamic fuzzing exposes anomalies/Trojans. Dynamically varying software, parametric, and hardware components when monitoring sidechannels amplifies differential Trojan behavior relative to Trojan-free as follows:

1) *Software fuzzing using crafted test codes*: Test codes are injected to collect HPC, power, and thermal sidechannel data when CPUs are executing controlled CPU/IO operations and to collect communication sidechannels while exchanging varying sizes of data packets among PCBs.
2) *Parameter fuzzing*: This includes changing protocol parameters such as bit rates and adjusting application-specific parameters to operate the system at/beyond its boundary conditions.
3) *Hardware fuzzing*: This includes changing device voltages and CPU clock scaling modes and overclocking.

Exponential combinations of dynamic fuzzing modes offer a moving target defense to increase likelihood of detecting Trojans, reduce false positives, and counteract evasion. The adversary is disadvantaged since they do not know the type and timing of fuzzing and measured sidechannels. Depending on type and location, controlled excitations expose Trojans either by localizing activity and measurements (e.g., using pairwise PCB communications) or violating attacker assumptions (e.g., varying bit rates from nominal).

EXPERIMENTAL RESULTS

Emulating COTS Multi-PCB System

We assembled a multi-PCB system [Figure 1(g), Table 2]. PCBs are interconnected via a network switch and two routers. The design has clusters of PCB devices connected to a router, emulating clusters of components on a PCB or multiple PCBs in a COTS system. The 48-port Netgear switch used as hub for routers supports multiple virtual local area networks (VLANs) (emulating clusters of components on single/multiple PCBs in a multi-PCB system). Router $R1$ allows remote access to testbed for configuration and testing. Routers $R2$ and $R3$ are connected to $R1$ and support dynamic setup of multiple clusters. We use Raspberry Pi4-b PCBs as routers. Every PCB is connected to one of the routers and is part of a cluster. PCBs in the same cluster communicate directly with each other. Communicating between PCBs in different clusters involves going through a router. This testbed is a heterogeneous collection of PCBs relative to CPU speed, CPU bit width, instruction set architectures, and interconnection capabilities.

The graph model is constructed by coloring nodes based on node similarity and then bucketing paths based on path equivalences (same colors of source and destination nodes, same path lengths). Odroid XU4 and MC1 nodes are marked with the same color since they have similar characteristics (similar processors and other components). Analogously, Raspberry Pi4 and NVIDIA Jetson PCBs have the same color. Node and path invariances follow from node coloring and path bucketing, which indicate expectations in terms of similarity of sidechannel characteristics due to similarity of nodes/paths. These expected invariances are independent of golden data. It is seen below that the expected invariances are indeed empirically satisfied in absence of Trojans and deviations from invariance properties illuminate existence of Trojans. We observe satisfaction/violation of path invariances using communication sidechannels and of node invariances using HPC, power, and thermal sidechannels.

We design test codes that vary by resources (IO/CPU) consumed and inter-PCB communications. Consecutive runs of an application are interspersed with sleep intervals.

Trojans in This Study [Figure 1(c)–(f)]

1) *MiTM*: Insert/modify a component to modify the input–output (IO) between PCBs. We move MC1-4 to another cluster, with a router Trojan sniffing inter-cluster communications.
2) *Counterfeits:* Replace good device with counterfeit; emulated by replacing XU4 with XU3, and Rpi4 with Rpi3.
3) *Firmware Trojan:* two variants—a) modify open-source MPI (https://www.mpich.org. . .) communication firmware in MC1-1 to log outbound packets and b) kernel module causing excessive data-cache flushes in MC1-2.
4) *Software rootkit* on MC1-2 to age devices.

One/multiple sidechannels can expose these Trojan types (e.g., MiTM by communication; counterfeit by power, HPC; firmware and software by power, HPC, thermal).

Trojan Detection Using Comm. Sidechannels

Snapshots of measurement time-series (Figure 2) are used to validate hypothesized properties. Graph nodes

TABLE 2. Description of SBCs and communication modules in testbed.

Category	Characteristics	Single-board computers in testbed							
		Odroid			Raspberry		Atomic	Asus	NVIDIA
		XU4	XU3	MC1	pi4b	pi3b	pi	Tinkerboard	Jetson Nano
CPU configuration	type	Arm Cortex A7+A15	Arm Cortex A7 + A15	Arm Cortex A7+A15	Arm Cortex A72	Arm Cortex A53	Intel atom x5-Z8350	Arm Cortex A17	Arm Cortex A57
	architecture	32-bit	32-bit	32-bit	64-bit	64-bit	32-bit	32-bit	64-bit
	micro-architecture	armv7f	armv7f	armv7f	armv8-A	armv8-A	x86	armv7-A	armv8-A
	# cores	8	8	8	4	4	4	4	4
	#processors	2	2	2	1	1	1	1	1
	frequency	1.5 GHz (A7), 2 GHz (A15)	1.5 GHz (A7), 2 GHz (A15)	1.5 GHz (A7), 2 GHz (A15)	1.5 GHz	1.2 GHz	1.92 GHz	1.8 GHz	1.43 GHz
	RAM	2GB	2GB	2GB	4GB	2GB	2GB	2GB	4GB
Cache	L1	64KB + 64KB	64KB + 64KB	64KB + 64KB	32KBx4 D + 48KBx4 I	32KBx4 D + 32KBx4 I	2MB smart cache	32KB D + 32KB I	32KBx4 D + 48KBx4 I
	L2	512KB + 2MB	512KB + 2MB	512KB + 2MB	1MB	2MB		1MB	2MB
GPU	type	Arm Mali T-628	Arm Mali T-628	Arm Mali T-628	-	-	Intel Integrated Graphics unit	Arm Mali T760 MP4	NVIDIA Maxwell
	frequency	600MHz	600MHz	600MHz	-	-	450MHz	650MHz	921MHz
Peripherals	# USB 2.0	4	4	1(host)	2	2	2	4	3
	# USB 3.0	1	1	0	2	2	1	-	1
	GPIO pins	20 + 10	20 + 10	0	40	40	20	40	40
Connectivity	ethernet	Gigabit ethernet (over USB3.0)	100 Mbps ethernet	Gigabit ethernet (over USB 3.0)	Gigabit ethernet (on-chip)	Gigabit ethernet (over USB 2.0)	Gigabit ethernet (on-chip)	Gigabit ethernet (on-chip)	Gigabit ethernet (on-chip)
	Wi-fi	USB dongle	USB dongle	no	onboard	onboard	no	onboard	onboard
	UART	2	1	0	6	1 + 1(mini)	1	4	3
	SPI	2	2	0	4	2	1	2	2
Sidechannels	Power	external	onboard	external	external	external	external	external	onboard
	Thermal	yes	yes	yes	yes	yes	yes	yes	yes
	HPC	yes	yes	yes	yes	yes	yes	yes	yes
Firmware	Linux version	4.14.180	4.14.180	4.14.141	5.4.51	5.4.51	4.19.126	4.17.134a	4.19.126

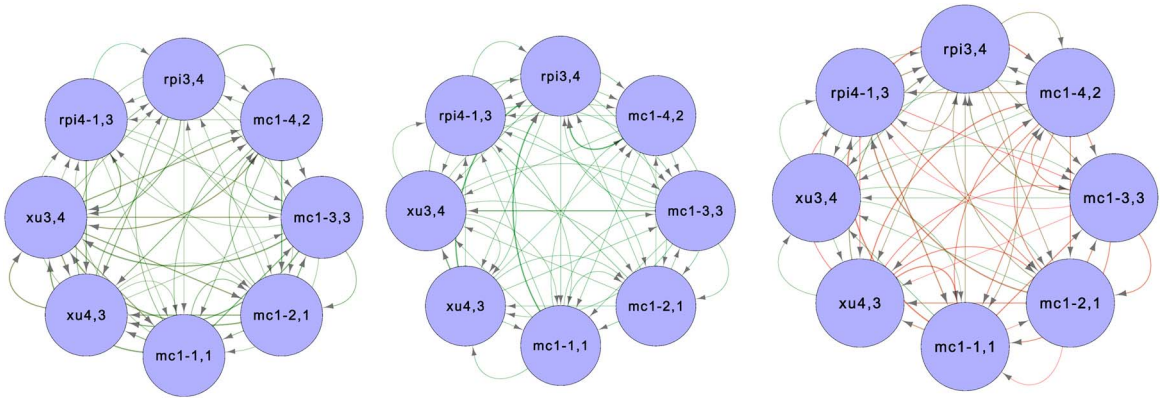
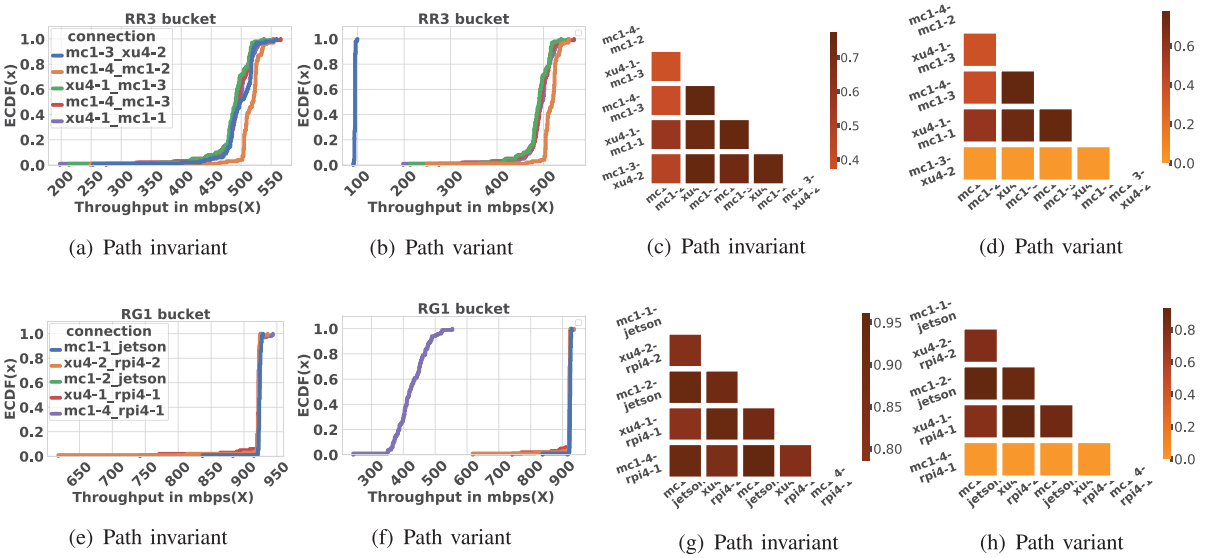


FIGURE 2. Snapshots over time window from communication sidechannels. Setup time (left), latency (middle), throughput (right). Feature means are edge colors (smaller → larger = green → red); std. deviations are widths.

in Figure 2 are labeled with nodes and clusters of our topology [Figure 1(b)]. Means and standard deviations (computed over time windows) of communication-based sidechannel measurements (setup time, latency, and throughput) are depicted using colors (means) and linewidths (standard deviations) of edges in the graph.

Edges exhibit similarities/dissimilarities depending on PCB and topology (e.g., higher latencies for connections between PCBs in different VLANs). From snapshots of on-PCB and inter-PCB sidechannels, predicate validation entails probabilistic verification of properties relative to expected behaviors or across different nodes/edges. Given expected interconnection topology, detecting MiTM Trojan (e.g., VLAN with adversary-injected MiTM Trojan, PCB moved into different VLAN) involves verifying equivalences/dissimilarities between measurements from different edges.

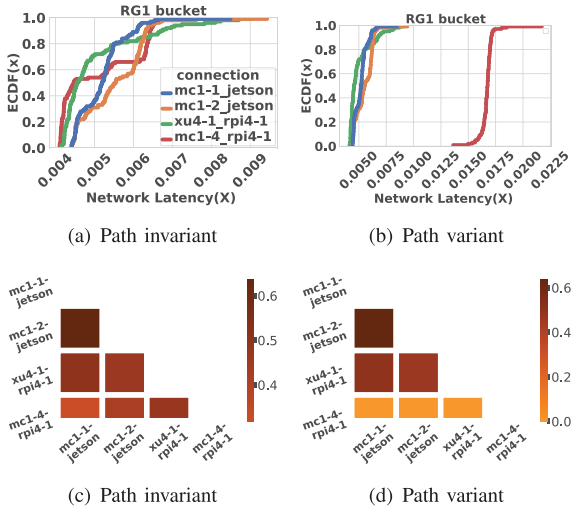
Considering sidechannel measurements as observations of random variables, likelihood of satisfaction of edge equivalence predicates is computed using empirical cumulative distribution functions (ECDFs) and divergences between ECDFs. Example ECDFs of communication sidechannels are shown in Figure 3. Divergences between calculated ECDFs are quantified using Bhattacharyya distances.



(a) Path invariant (b) Path variant (c) Path invariant (d) Path variant

(e) Path invariant (f) Path variant (g) Path invariant (h) Path variant

FIGURE 3. ECDFs and Bhattacharyya distance for RR3 (top two rows) and RG1 bucket (bottom two rows). (a), (c), (e), and (g) show path invariance among PCB-PCB connections. (b), (d), (f), and (h) show path variance indicating an anomaly.



(a) Path invariant (b) Path variant

(c) Path invariant (d) Path variant

FIGURE 4. ECDFs and Bhattacharyya distance on network latency for the RG1 bucket: (a) and (c) path invariance. (b) and (d) path variance indicating an anomaly.

We consider MiTM and counterfeit Trojans. Considering R and G nodes, likelihoods of satisfaction of path invariance for $R \leftrightarrow G$ and $R \leftrightarrow R$ connections are computed using ECDFs and distances between probability distributions of connections using ρ in Figure 3. We use a threshold of $\eta = 0.5$ to measure similarity using ρ. Pairwise connections in the RR bucket should satisfy path invariance as validated in Figure 3(a) for $R \leftrightarrow R$ connections with almost identical ECDFs and $\rho > 0.5$ [Figure 3(c)]. Higher ρ with identical ECDFs indicates two connections are path invariant. If a counterfeit (XU3) replaces XU4-2, it is detected as anomalous by throughput sidechannel ECDF of MC1-3 $\leftrightarrow$ XU4-2 connection ($R \leftrightarrow R$) by violating path invariance [Figure 3(b)]. ρ(MC1-3 $\rightarrow$ XU4-2) with other connections < 0.5, failing path invariance in Figure 3(d).

Figure 3(e) and (g) shows that pairwise connections in the RG bucket satisfy path invariance in the throughput sidechannel. We move MC1-4 (R) to a different cluster and implant an MiTM Trojan (router) to snoop on communications of MC1-4 with PCBs in other clusters. This Trojan is detected in Figure 3(f) since throughput ECDF of MC1-4 $\leftrightarrow$ RPi4-1 violates path invariance of $R \leftrightarrow G$ connections. Also, distance values MC1-4 $\leftrightarrow$ RPi4-1 in Figure 3(h) are < 0.5 validating this anomaly. When connection setup time is used as the communication-based feature instead of throughput, it was empirically noted that setup time does not exhibit as many deviations under MiTM modifications and is less illustrative than throughput measurements for Trojan detection. However, similar observations as with throughput can also be drawn from analysis of latency instead of throughput as seen in Figure 4 enabling Trojan detection.

Trojan Detection Using Power Sidechannels

We execute a CPU-intensive workload on each PCB for 80 s and collect power consumption measurements. We evaluate our hypothesis of *node invariance* using measurements for PCBs clustered with the same color. To maintain the same run-time execution environment, we set all boards to user space governor using dynamic voltage frequency scaling and run workloads at two operating frequencies: 1 GHz and 1.5 GHz. We use dynamic time warping (DTW) to measure the distance between power traces of devices in the same colored

TABLE 3. DTW in power sidechannels deviate when XU4-2 is replaced by XU3. $\mathrm{DTW}(XU4-2, D) > \delta_{\mathrm{DTW}}(=25)\ \forall D \in V^{d}_{\mathrm{red}}$ fails node invariance.

Setting	Device under Test	PCB devices							
		XU4-1		XU4-2		MC1-1		MC1-2	
		1G	1.5G	1G	1.5G	1G	1.5G	1G	1.5G
Node-invariant	XU4-1	-	-	12.64	16.33	12.99	22.43	16.38	22.32
	XU4-2	12.64	16.33	-	-	15.44	18.77	14.42	20.91
	MC1-1	12.99	22.43	15.44	18.77	-	-	11.62	19.50
	MC1-2	16.38	22.32	14.42	20.91	11.62	19.50	-	-
Node-variant	XU4-1	-	-	**95.21**	**151.95**	12.99	22.43	16.38	22.32
	XU4-2	**95.21**	**151.95**	-	-	**95.27**	**152.59**	**99.93**	**151.02**
	MC1-1	12.99	22.43	**95.27**	**152.59**	-	-	11.62	19.50
	MC1-2	16.38	22.32	**99.93**	**151.02**	11.62	19.50	-	-

TABLE 4. HPC sidechannels detect firmware cache-flush and MPI library Trojans using data cache access (DCA), instn cache access (ICA), data cache miss (DCM), total cache miss (TCM) HPCs.

Type	HPC	Freq. (GHz)	PCB					
			MC1-1		MC1-2		MC1-3	
			Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
Cache-flush	Total Cycles	1	6545.36	1536.25	**8788.87**	**2921.43**	7254.36	1836.25
		1.5	11237.78	3536.57	**17984.26**	**5852.15**	13498.56	4136.25
	L2 DCA	1	29.28	17.83	**181.80**	**121.38**	28.35	18.25
		1.5	32.80	23.00	**393.98**	**264.7**	31.22	23.66
	L1 DCM	1	82.24	33.66	**459.23**	**362.68**	98.87	57.09
		1.5	61.24	36.25	**3174.18**	**2023.85**	60.31	38.75
MPI	L2 DCA	1	**287.36**	**37.89**	211.24	32.25	214.25	29.68
		1.5	**331.72**	**47.89**	268.73	37.53	257.57	33.32
	L1 ICA	1	**795.25**	**99.65**	645.65	52.89	694.36	48.25
		1.5	**955.29**	**115.32**	821.81	77.16	827.15	81.39
	L2 TCM	1	**65.36**	**14.14**	85.25	15.55	81.27	18.14
		1.5	**70.72**	**16.17**	99.03	17.69	91.39	18.89

bucket. We consider counterfeit Trojan (XU4-2 replaced with XU3). We collect power traces for clean and counterfeit cases. The power traces are similar except for XU3. Table 3 shows pairwise DTW distances for devices in red bucket. To account for process variations and noise, we set threshold $\delta_{\mathrm{DTW}} = 25$. Node invariance is violated ($\mathrm{DTW}_{XU4-2 \to D} > 25,\ \forall D \in V^d_{\mathrm{red}}$) for the red bucket, detecting counterfeit (XU3).

Trojan Detection Using HPC Sidechannels

HPC traces were collected under software fuzzing by running CPU- and IO- intensive test codes on the same color nodes (in R colored bucket which has Odroids MC1 and XU4s) at two frequencies (1 GHz and 1.5 GHz). Two Trojans were injected: 1) cache flush Trojan in MC1-2 and 2) MPI software library rootkit in MC1-1. To address noise in HPC traces, we collected several samples (~75,000) and computed mean and standard deviation (Table 4). We set threshold $\delta = 20\%$ for checking node invariance property considering process variation across MC1 PCB chipsets. When Trojans are active, mean statistics of MC1-2 substantially increase (Table 4). It was noted that σ_{pd} is within δ for PCBs that satisfy node invariance property, except for MC1-2 and MC1-1. A defender can adjust threshold δ depending on test codes. The defender can use alternate CPU-intensive codes and IO-intensive codes, presenting a moving target for the attacker.

Trojan Detection Using Thermal Sidechannels

We run CPU-intensive codes on the PCBs operating at 1 and 1.5 GHz. Counterfeits may slow processing, resulting in more time to run a workload in a time window, thereby changing time intervals of different temperature levels. Thermal sidechannels are different between XU3 and XU4 PCBs. If the original design called for these PCBs to be identical, and if the adversary replaces one with a counterfeit (i.e., XU3), invalidation of similarity of thermal signals between the PCBs indicates an anomaly. Firmware/software Trojans can interfere with normal operation of PCBs, causing deviations from invariance properties. This can be detected by the thermal sidechannel. We use percent of deviation from mean to identify dissimilarities between thermal sidechannels of PCBs. Node invariance identifies MC1-1, MC1-2, and MC1-3 PCBs to have similar properties from specifications. Thermal measurements should be within a threshold percent deviation of mean. A rootkit on MC1-2 increases percent deviation exposing dissimilarity between PCB sidechannels (plots omitted for brevity). Percent deviation of temperature of MC1-2 relative to MC1-1 and MC1-3 increases from –8.5 to 47.4 and from –4.9 to 53.3, respectively. Deviations of MC1-2

are higher when PCBs operate at 1 GHz compared to 1.5 GHz. If the Trojan is resource-intensive, CPU temperature spikes are more conspicuous compared to average temperature rise at 1-GHz clock. Temperature variations at 1.5 GHz clock are at a higher level, hiding the Trojan.

CONCLUSION AND FUTURE DIRECTIONS

We developed a graph-theoretic approach to detect Trojans in multi-PCB systems. Our golden-free method is an unsupervised Trojan detection approach using multimodal sensing under controlled fuzzing. The approach considers on-node and inter-node sidechannels. Hence, even if a malicious modification/implant does not have measurable impact on one sidechannel, it could be detectable through other sidechannels. However, as with any Trojan detection methodology, it is understood that if a Trojan has no measurable impact on any of the considered sidechannels, it would not be possible to detect (e.g., a Trojan that passively monitors electromagnetic emissions without electrical/network connectivity to actual multi-PCB system, a counterfeit so similar to original component that there is no measurable difference in sidechannels). Nevertheless, for a realistic Trojan intended to maliciously affect either a node's operation or inter-node communication (e.g., MiTM), there would typically be a measurable impact on one/more sidechannels. Composite Trojans comprised of multiple malicious modifications are also detected using our approach since presence of multiple modifications results in flagging of deviations from multiple expected invariance properties. The on-node (software, firmware, counterfeit) and inter-node (MiTM) Trojans considered in experimental studies have been tested in multiple combinations and reliable detection of such composite Trojans has been experimentally noted. Future work based on proposed approach will explore more Trojan types and fuzzing-aided detection.

ACKNOWLEDGMENTS

This work was supported in part by DARPA MTO (FA8750-20-1-0502), DARPA I2O (HR00112390029), and NSF SaTC (2039615).

REFERENCES

1. D. Mehta et al., "The big hack explained: Detection and prevention of PCB supply chain implants," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 4, pp. 1–25, Aug. 2020, doi: 10.1145/3401980.
2. M. McGuire, U. Ogras, and S. Ozev, "PCB hardware Trojans: Attack modes and detection strategies," in *Proc. IEEE VTS Test Symp.*, 2019, pp. 1–6, doi: 10.1109/VTS.2019.8758643.
3. S. Ghosh, A. Basak, and S. Bhunia, "How secure are printed circuit boards against Trojan attacks?" *IEEE Des. Test*, vol. 32, no. 2, pp. 7–16, Apr. 2015, doi: 10.1109/MDAT.2014.2347918.
4. K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, May 2016, doi: 10.1145/2906147.
5. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 296–310, doi: 10.1109/SP.2007.36.
6. Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proc. ACM/EDAC/IEEE Des. Automat. Conf.*, 2014, pp. 1–6, doi: 10.1145/2593069.2593147.
7. Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardware-Oriented Secur. Trust*, 2008, pp. 51–57, doi: 10.1109/HST.2008.4559049.
8. N. Patel et al., "Towards a new thermal monitoring based framework for embedded cps device security," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 524–536, Jan./Feb. 2022, doi: 10.1109/TDSC.2020.2973959.
9. D. Paul-Pena, P. Krishnamurthy, R. Karri, and F. Khorrami, "Process-aware side channel monitoring for embedded control system security," in *Proc. IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, 2017, pp. 1–6, doi: 10.1109/VLSI-SoC.2017.8203468.
10. P. Krishnamurthy, R. Karri, and F. Khorrami, "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 666–680, Jun. 2019, doi: 10.1109/TIFS.2019.2923577.
11. Y. Zhang, H. Quan, X. Li, and K. Chen, "Golden-free processor hardware Trojan detection using bit power consistency analysis," *J. Electron. Testing, Theory Appl.*, vol. 34, no. 3, pp. 305–312, Jun. 2018, doi: 10.1007/s10836-018-5715-z.
12. "Microsystems exploration: Safeguards against hidden effects and anomalous Trojans in hardware (SHEATH)," Defense Advanced Research Projects Agency, 2019. [Online]. Available: https://sam.gov/opp/25d32ec9a35117576b04d52896267319/view

ANIMESH BASAK CHOWDHURY is a Ph.D. candidate at the New York University Center for Cybersecurity, Brooklyn, NY, 11201, USA. His research interests include machine learning for electric design automation and systems security. Chowdhury received his M.S. degree in computer science from ISI, Kolkata. Contact him at abc586@nyu.edu.

ANUSHREE MAHAPATRA is a research engineer at Innatera Systems, 2289EX, Rijswijk, The Netherlands. Her research interests include high-level synthesis, machine learning, and system-on-chip security. Mahapatra received her Ph.D. degree from Hong Kong Polytechnique. Contact her at am11019@nyu.edu.

YANG LIU is an M.S. student in the Electrical and Computer Engineering Department, New York University Tandon School of Engineering, Brooklyn, NY, 11201, USA. His research interests include machine learning, robotics, operating systems, network systems, and computer architecture. Liu received his B.E. degree in software engineering from Fudan University. Contact him at yl6978@nyu.edu.

PRASHANTH KRISHNAMURTHY is a research scientist and adjunct faculty in the Electrical and Computer Engineering Department, New York University (NYU) Tandon School of Engineering, Brooklyn, NY, 11201, USA. His research interests include cyberphysical systems, robotics, and controls. Krishnamurthy received his Ph.D. degree in electrical engineering from Polytechnic University (now NYU). He is a Member of IEEE. Contact him at prashanth.krishnamurthy@nyu.edu.

FARSHAD KHORRAMI is a professor of electrical and computer engineering at New York University, Brooklyn, NY, 11201, USA. His research interests include controls, robotics, cyber security for cyberphysical systems, embedded systems, and machine learning. Khorrami received his Ph.D. degree in electrical engineering from Ohio State University. He is a Senior Member of IEEE. Contact him at khorrami@nyu.edu.

RAMESH KARRI is a professor of electrical and computer engineering at New York University, Brooklyn, NY, 11201, USA. His research interests include hardware cybersecurity systems. Karri received his Ph.D. degree in computer science and engineering from the University of California at San Diego. He is a Fellow of IEEE. Contact him at rkarri@nyu.edu.