Detecting Hardware Trojans in PCBs Using Side Channel Loopbacks

Hammond Pearce, *Member, IEEE*, Virinchi Roy Surabhi, Prashanth Krishnamurthy, *Member, IEEE*, Joshua Trujillo, Ramesh Karri, *Fellow, IEEE*, and Farshad Khorrami, *Senior Member, IEEE*

***Abstract*— Malicious modifications to printed circuit boards (PCBs) are known as hardware Trojans. These may arise when malafide third parties alter PCBs premanufacturing or postmanufacturing and are a concern in safety-critical applications, such as industrial control systems. In this research, we examine how data-driven detection can be utilized to detect such Trojans at run-time. We develop a flexible and reconfigurable PCB test bed derived from the popular open-source programmable logic controller (PLC) platform "OpenPLC." We then develop a Trojan detection framework, which utilizes and analyzes multimodal side channels (e.g., timing, magnetic signals, power, and hardware performance counters). We consider defender-configurable input/output (I/O) loopback test, comparison with design-document baselines, and magnetometer-aided monitoring of system behavior under defender-chosen excitations. Our approach can extend to golden-free environments. Golden (known-good) versions of the PCBs are assumed not available, but design information, datasheets, and component-level data are available. We demonstrate the efficacy of our approach on a range of Trojans instantiated in the test bed.**

***Index Terms*— Anomaly detection, golden-free, machine learning (ML), printed circuit board (PCB), timing loopback, Trojan detection.**

I. Introduction

HARDWARE Trojans are malicious alterations to designs with the aim of introducing faults or undermining secrecy [1]–[3]. While they are commonly considered in the integrated circuit (IC) space, where alterations range from alterations to a single logic gate to the addition of whole IP blocks to System-on-Chips (SoCs) [3], such Trojans can also be introduced to printed circuit boards (PCBs) [4]–[6], where complicated and distributed supply and manufacturing networks provide opportunities for untrusted parties to alter or add hardware, firmware, and software implants. Given the recent high-profile alleged hack on SuperMicro [7], [8] and focus by the Defense Advanced Research Projects Agency (DARPA) [9] it is becoming increasingly important to consider such Trojans, especially when PCBs are destined for safety-critical systems, for instance, industrial cyber–physical systems (CPSs).

While other works examine detecting Trojans statically, e.g., via device inspection with X-rays [10], we, instead, consider hardware Trojan detection using operational tests. Multiple side channels are utilized for anomaly detection, including: 1) analog measurements, such as electromagnetic emissions, power, and temperature; 2) measurements from communication and analog/digital input/output (I/O) channels, including temporal patterns of activity; and 3) digital measurements, such as hardware performance counters (HPCs), CPU activity, and access patterns. To perform our analysis, we provide two novel contributions: first, a methodology for Trojan detection, which combines the aforementioned measurements in a side-channel loopback setting, and second, a novel test bed specifically designed for hardware Trojan research.

Our test bed is derived from the OpenPLC programmable logic controller (PLC) [11], and we term it the "OpenPLC NYU Trojan Edition (TE)." It supports static and dynamic insertion of Trojans via mechanisms built into the PCB, including rewiring jumpers, a secondary "Trojan" microcontroller, software Trojans colocated with the OpenPLC run-time, and surreptitious communication channels between Trojan components on the microcontroller. For evaluating detection strategies, it supports defensive monitoring based on measurements of digital and analog side channels, including power and magnetometer sensors. Our example detection methodology combines and analyzes the side channels using spatiotemporal feature extraction and machine learning (ML) methods. Given defender-controlled software-driven excitations and I/O connections in a side-channel loopback structure, a probabilistic analysis of side-channel patterns flags anomalies against expected behaviors.

To validate the test bed and approach, we studied a number of representative machine-in-the-middle (MITM) hardware Trojans. To reduce the reliance on golden/known-good integrated systems and components, we investigated whether we can apply techniques in a golden-free setting. Using design-time information, datasheets, and components without known-good integrated systems, we can disambiguate side-channel measurements in Trojan-free versus Trojan-containing PCBs.

Our novelty stems from our particular formulation for loopbacks in the multimodal setting and the experimental test

Manuscript received December 6, 2021; revised March 23, 2022; accepted April 11, 2022. Date of publication May 12, 2022; date of current version June 29, 2022. This work was supported in part by DoE Kansas City. Honeywell Federal Manufacturing & Technologies, LLC operates the Kansas City National Security Campus for the U.S. Department of Energy/National Nuclear Security Administration under Contract DE-NA0002839. *(Corresponding author: Virinchi Roy Surabhi.)*

Hammond Pearce, Virinchi Roy Surabhi, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami are with the Department of Electrical and Computer Engineering (ECE), NYU Tandon School of Engineering, Brooklyn, NY 11201 USA (e-mail: hammond.pearce@nyu.edu; virinchi.roy@nyu.edu; prashanth.krishnamurthy@nyu.edu; rkarri@nyu.edu; khorrami@nyu.edu).

Joshua Trujillo is with the Kansas City National Security Campus, Department of Energy, Kansas City, MO 64147 USA (e-mail: jtrujillo@kcnsc.doe.gov).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2022.3171174.

Digital Object Identifier 10.1109/TVLSI.2022.3171174

bed designed for research on Trojans and their detection. We view the detection process as closed loop, whereby a defender-controlled excitation causes changes, and outputs are validated across multiple/complementary side channels.

This article is organized as follows. Section II covers the related work. Section III details the OpenPLC "NYU TE" test bed. Section IV covers digital and analog side-channel measurements and their application for Trojan detection. Section V describes the different methods used to detect Trojans. Section VI discusses our corpus of representative Trojans, their effects, and their detection. Section VII presents the conclusion.

II. Related Work

Most Trojan detection techniques intentionally activate Trojans and detect them by analyzing side-channel data [3], [12], [13]. These statistics are compared to "golden models." Although many approaches have been proposed to detect Trojans in ICs, including golden-free settings [14]–[20], little research has been done to detect Trojans at the PCB level [21].

Some studies propose run-time monitoring of side channels to detect the anomalies in PCBs [6], [22]–[29]. Online power calculation and assessing the power usage of classes of genuine components on a PCB and the power used by the PCB are suggested in [6]. Acoustic side-channel data can be exploited to breach system security [30]. Electromagnetic emanations are used in [22] to identify anomalous code execution in PLCs. In [23], an electromagnetic side-channel-based spectrum model is developed to assess the detectability of various hardware Trojans. The electromagnetic fingerprints of hardware Trojan are shown to be effective in identifying an infected Trojan [24]. In our work, we use magnetic side channel to detect the effects of MITM Trojans that modify the temporal mappings from relay commands to relay switching behavior. HPCs are used to detect abnormal behavior in software [26]–[29], [31]. Temperature tracking using thermal sensors during run-time is another option [32], [33].

Fuzz testing is an approach that can be used to excite Trojans. Fuzz testing has been shown to be a very successful approach for exposing flaws in a wide range of software systems to date [34]. Emerging methods for fuzz testing conventional protocols, such as the industrial protocols MODBUS, PROFINET-DCP, and ZigBee, discover problems using protocol grammar [35] or data gathered from the communication device [36]. They test the state of the system under test using FSM models [37] or preset frameworks [38]. In this work, we use fuzzing degrees of freedom, such as loopback connections, bit rates, pseudorandom relay commands, and variations of time intervals between relay command changes.

Another line of defense for detecting Trojans is moving target defense (MTD). MTD is an effective defense method that dynamically modifies the features of configurations of a target system while keeping its core capabilities in order to diversify its defense mechanism and obfuscate the resultant attack surfaces. Instruction sets, address space layouts, IP addresses, port numbers, proxies, virtual machines, and operating systems are examples of system characteristics (and, hence, possible relevant aspects of attack surfaces) that may be dynamically altered to mislead attackers [39]. Existing MTD research has largely concentrated on low-level characteristics, such as instruction set randomization [40], [41] and address space layout randomization [42], [43]. Other MTDs, such as IP address randomization [44], [45], virtualization-based MTD [46], and software-defined networking-based MTD [47], [48], target network-level characteristics. In addition to varying the low-level protocol characteristics, additional defensive variety against attacks in, for example, IoT communication protocols can be achieved using MTD through communication protocol dialects [49]. In our work, we use MTD to detect hardware Trojans that spoof, for example, UART communication by dynamically modifying parameters, such as MODBUS bit rate.

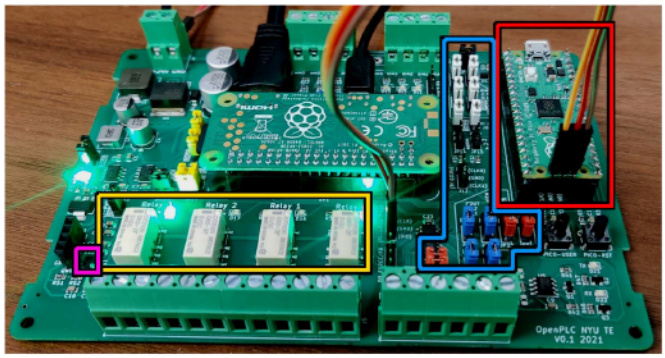

Fig. 1. OpenPLC "NYU TE." HW Trojans are emulated by Pico module (red outline, right-most box), with wiring changes supported by jumpers (blue outline, second-right). Output relays (yellow outline, second-left box, from left to right, relays have indices 3, 2, 1, an 0) are near the magnetometer/accelerometer (turquoise outline, leftmost box).

Timing side channels can detect Trojans that are in MITM configuration. A thorough investigation of point-to-point packet delay in an operating tier 4 network is conducted in [50]. As a method for identifying MITM attacks on mobile communications, a static analysis approach that uses exact timing is developed [51]. An approach that includes determining delays by using the timestamps of TCP packet headers is proposed in [52]. The effectiveness of timing analysis to detect a text-messaging MITM attack is illustrated in [53]. We use timing loopbacks to detect MITM hardware that snoops communication channels, such as UART or GPIO pins.

III. OpenPLC "NYU TE" Platform to Study Trojans and Trojan Detection

This section introduces the threat model and describes the Trojan PCB test bed. The test bed has: 1) a Raspberry Pi SBC running embedded Linux that runs application software and 2) the main board that provides platform I/O and can emulate Trojans that are novel, hidden, and trigger-able. Fig. 1 shows the general design of the test bed: the Raspberry Pi Pico (indicated by the red box) is the primary device used for emulating Trojan behaviors.

A. Threat Model

In this study, we consider an adversary who intends to plant a Trojan into a PLC in an MITM configuration. While the hardware supports run-time modification to sensor data (i.e., via advanced Trojans), we assume that the onboard side-channel measurement sensors are trustworthy, as they can be validated against external sensors if needed.

MITM Trojans can: 1) snoop on signals passing between peripherals and the main design and 2) edit relevant signals passing in and out of the design. The defender needs to catch the Trojans before the PCB is fielded. As this setting is predeployment, they control the environmental conditions (e.g., maintaining ambient conditions include electromagnetic interference (EMI) and calibrating the side-channel measurement sensors). The defender may use strategies for golden-free Trojan detection that include: 1) evaluating probabilistic equivalences against expected statistical distributions; 2) using design-based models to compare side-channel measurements; and 3) fuzzing hardware parameters, connections, and software test codes. The tests in this study are carried out separately without any external loads. Nevertheless, if any external load is connected or if the tests need to be carried out in the field, the side-channel sensors can be recalibrated.

While the test bed and Trojan detection methodology may also be used for pure software Trojans (i.e., in the Raspberry Pi), we exclude these from this analysis as out of scope.

B. General Function: Programmable Logic Controller

This PCB has applications in industrial control systems, i.e., as a PLC. Hence, it has features associated with PLCs, including a resilient 24-V power supply; four relay-isolated general-purpose outputs for controlling common industrial equipment (e.g., 24-V dc motors, lamps, and variable frequency drives); four optoisolated 24-V digital inputs for sensing in common industrial equipment (e.g., infrared beams, mechanical switches, and pneumatic sensors); four high-impedance 24-V-tolerant analog inputs for reading from variable-value industrial equipment (e.g., ultrasonic distance measuring devices); and a half-duplex RS-485 bus to communicate with other PLCs and I/O modules via MODBUS.

To implement the PLC, a Raspberry Pi Zero single-board computer (SBC) module [54] is used as an embedded application processor. This SBC can run Linux (Raspberry Pi OS—a derivative of Debian) giving us a lot of flexibility and connectivity, including support for HDMI display outputs and USB giving access to common peripherals, including keyboards and USB–Ethernet connectors. This enables simple debugging and support for graphical applications that use industrial human–machine interfaces (HMIs).

For PLC application software, the platform supports the use of programs written in the C and Python programming languages. Though not specifically used in this work, the PLC also supports the OpenPLC libraries and framework [11]. The PLC can use any of the IEC 61131-3 languages, including function blocks, ladder logic, and structured text.

C. Injecting and Configuring Trojans

The primary objective of our platform was to enable the insertion and configuration of a wide range of CPS-relevant embedded Trojans. In general, PCB-level Trojans are a superset of any malicious change to a design's hardware. For example, one can add or remove components, substitute low-quality components, and alter PCB traces—changing sources, destinations, and, in certain cases, track widths, although we consider track width changes out of scope.

Five main features enable Trojan insertion in our platform.

1) Pluggable Trojan Module: In this PCB, insertion of complex electronic Trojans is achieved via a socket for inserting extra hardware. This socket has a footprint compatible with Raspberry Pi Pico [55] (red outline in Fig. 1), enabling straightforward microcontroller-based Trojans while ensuring flexibility for custom Trojans based on other hardware. Furthermore, by making the module detachable, it can demonstrate how a Trojan is "added" and "removed" from the design.

Using Pico modules as the primary Trojan-based insertion platform can enable extremely complex and subtle Trojans: each module has an on-PCB RP2040 dual-core ARM-M0+ microcontroller that runs at up to 133 MHz with 264 kB of SRAM and 2 MB of on-PCB Flash memory. They also feature USB 1.1 compatible interfaces with device and host support. This platform supports flexible emulation to feature static (constant) and dynamic (triggered) malicious behavior.

The socket is designed to fit into the overall PCB such that it may both observe signals passively and interfere with the normal operation of the PLC's peripherals, including the inputs, outputs, and communication buses. To support this interference, wiring reconfigurability is provided via a number of different hardware "jumpers."

In addition to the socketable Trojan, the Raspberry Pi Zero SBC is attached to the main PLC PCB via a removable socket. This may enable other kinds of attacks, where the Raspberry Pi SBC is compromised, or an editing module is installed between the SBC and the PCB. An example Trojan here might be to install a Raspberry Pi Zero W with on-PCB Wi-Fi rather than the normal version without Wi-Fi.

2) Reconfigurable Wiring Jumpers: In addition to the Trojan's main socket, reconfigurable jumpers physically present on the PCB facilitate the ability to reroute certain input and output signals to enable or disable MITM Trojans. Among these are the optoisolated inputs, relay outputs, the I2C bus, and the RS-485 /UART bus. Passive components could be used to introduce unexpected changes to the electrical characteristics of these signals (for example, we could add a capacitor to an I2C bus to introduce faults). Reconfigurability is enabled by "jumper" wires. While, in general, jumper wires are not suitable for high-speed signals, in PLC applications (which have low-speed switching rates, e.g., MODBUS at 9600 baud), these wires are likely to not adversely affect system performance. When used in conjunction with the socketable Trojan, a complete Trojan experiment can tailor the PCB, choosing which signals are just readable and which are actually editable. This enables Trojans to change data values, insert delays, and reroute signals.

3) Software Trojan Support: Though not used in this work, as the Raspberry Pi Zero SBC at the core of the PLC is running a Debian Linux-based operating system, additional software Trojans that may interfere with correct operation may also be installed into the system. These could include keyloggers, cryptominers, or ransomware.

4) Back-Channel Communication: The most complex kinds of Trojans involve both hardware and software changes (or are capable of changing the behavior of the software from the hardware). To examine such "multidomain" Trojans, the

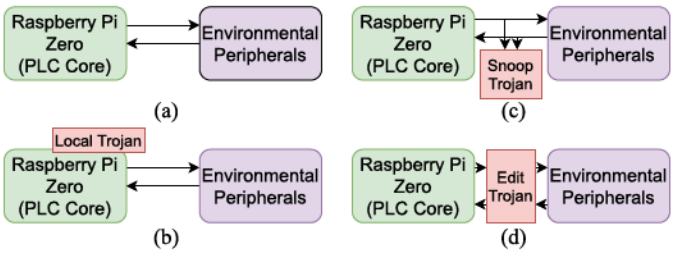

Fig. 2. Different categories of PCB-level Trojan. (a) No Trojan. (c) Snoop Trojan. (b) Local Trojan. (d) Edit Trojan.

PLC also includes features for back-channel I2C and UART communication by ensuring the connection between the I2C and UART pins of the Raspberry Pi SBC and the Pico sockets. As one example, if configured appropriately, the Pico can send shell commands to the Pi over the UART.

D. Example Trojans

Four Trojan insertion categories are depicted in Fig. 2.

1) *No Trojan:* It represents reference baseline performance.
2) *Local Trojan:* This configuration has software Trojans executing on Raspberry Pi Zero. These Trojans leak information or degrade performance. We do not discuss these Trojans in this article since we focus on hardware Trojans (i.e., Trojans enabled by introducing a hardware component, i.e., the Pico).
3) *Snoop Trojan*: This configuration has Trojans monitoring the signals passing between the peripherals and the Raspberry Pi Zero. These signals may be saved to memory and exfiltrated, or exported via extra connectivity, such as a USB peripheral on the Pico. Since they do not change the signals in any way, these are lightweight MITM Trojans.
4) *Edit Trojans:* They observe and edit all relevant signals passing in and out of the design. These Trojans allow for arbitrary MITM Trojan effects to be inserted into the signal lines. Trojans can degrade closed-loop system performance by inserting delays into the signal lines or modifying signal values. Trojans can leak information by sending signals to additional I/O lines or by modifying signals over I/O channels.

The test bed supports a variety of Trojan triggers, such as a manual input switch; a trigger when PLC inputs reach some value once (or repeatedly) or after a predefined sequence; triggered on observing specific control, addresses, or data bits on RS-485/UART or I2C buses; timed triggers that activate after some time; and any combination of these triggers.

E. Features for Defensive Monitoring

In order to monitor the performance of the PLC at runtime, the test bed supports the measurement of side-channel signals, as discussed in Section IV. These include on-processor digital side channels, analog measurements, such as power and magnetic signals, and timing measurements from I/O channels measured using defender-configured test software. For example, an output-editing Trojan may be detected via timing (it adds delays) or magnetic signature changes (the output pattern of relays is different than expected) or power (the relays may consume more/less power than expected), and these changes occur simultaneously.

To measure power consumption, Hall-effect current monitors are combined with an analog–digital converter. Three power measurement channels are captured: 1) current draw of the PCB; 2) current draw of Raspberry Pi; and 3) the current draw for the separate (Trojan) microcontroller socket. To measure local vibrations and magnetic fields, a six-axis accelerometer and a magnetometer-based inertial measurement unit (IMU) are placed near the relays on the PCB. We chose this since the functioning of the output relays affects these side channels and can detect changes in their outputs.

For more flexibility, we added connection points for sensor expansion. The first of these is a connector for QWIIC peripherals (e.g., external magnetometers). The second is power feeder jumpers, which can be replaced with wires to external power monitors (e.g., ammeters). Though not directly used in this study, we used these to calibrate internal sensor readings.

F. Open-Source Design

As it is a major contribution of this work, the PCB design developed in this project is made available online.[1]

IV. Multimodal Side-Channel Trojan Detection

A. Side Channels

The test bed supports a range of side-channel measurements.

HPCs: They measure processor activity over time. For example, HPCs [26]–[29], [31] can measure the number of occurrences of instructions, cycles, branches, L1/L2/L3 cache hits, and cache misses. The specific set of HPCs available is processor-dependent.

Power: Current (power) measurements [6] can be acquired via on-PCB or off-PCB sensors. Current measurements will vary based on the internal activities of the PLC. Time-varying activities may enable the detection of unexpected hardware/software components that increase/decrease the power draw and variations in temporal patterns of power consumption.

CPU Load and Frequency: CPU activity levels can be monitored using kernel-level CPU usage measurements. The Linux kernel separately measures user, system, and idle states. When a CPU governor dynamically modifies CPU frequency based on load, CPU frequency readings are a proxy for CPU usage.

Temperature: Temperature readings [32], [33] from one or multiple on-PCB sensors can be used to reflect the amount of activity of PCB components or subsystems.

Accelerometers: Three-axis accelerometers can be used to measure the acceleration of linear motion in the X-, Y-, or Z-axes. These sensors can provide indications of motion, shock, or vibration. In our PLC, which has partly mechanical components (relays), temporal variations of accelerometer measurements can provide indicators of hardware operation, e.g., relay clicking.

[1] https://github.com/kiwih/nyu-openplc-te

Magnetometers: Temporal variations of magnetic readings obtained using magnetometers can be used to sense hardware operation, e.g., relays on the OpenPLC PCB.

Software Timer: Timing measurements obtained via software on the main application processor on the PCB can be used to measure time intervals between events. Timers can be used to measure loopback times over different combinations of I/O channels. The presence of MITM Trojans and other hardware/software Trojans can cause variations in temporal behaviors that can be detectable using timing measurements.

Each side channel provides a different view of the system's behavior. Digital side channels, such as HPCs, provide information on CPU operation and can enable the detection of Trojan-induced variations in code execution. Power and temperature side channels provide aggregate views of the PCB behavior and can detect variations in system-level operation. Depending on the particular Trojan type/effects, it might be detectable using some or a combination of side channels. Fusing side channels can reduce both false negatives and false positives by detecting anomalies from the perspectives of different side channels and reducing sensitivity to benign random variations. The side channels have different sampling rates and granularities. HPC, accelerometer, magnetometer, and power are high bandwidth signals and benefit from high sampling rates. Lower sampling rates typically suffice for temperature, fan speed, CPU usage, and CPU frequency, which vary relatively slowly.

Side-Channel Data Collection: Given a PCB, multimodal side-channel Trojan detection entails collecting time-series measurements of side-channel signals and analyzing them to assign probabilistic likelihood scores of whether the measurements match expected system characteristics. The operating conditions of the system when collecting the side channels can be controlled by the defender. "Fuzzing" degrees of freedom to excite the system under different regimes include defender-controlled software excitations (e.g., running single or multiple instances of defender-created test codes), I/O connections and communication parameters, pseudorandom actuator commands, and hardware/software parameter variations. These defender-controlled excitations can increase the detectability of Trojans by measuring side channels over wider operating conditions, thereby improving detection accuracy and reducing false positives and false negatives. Controlling the fuzzing excitations/modes is a moving-target defense since the adversary does not know when and what fuzzing modes will be applied.

B. Side-Channel Measurement Framework

We use the OpenPLC "NYU TE" (see Section III) for our experimental studies of the Trojan detection methodology. All side channels discussed in this section can be measured on this PCB. HPCs are measured using Linux `perf` tool, CPU load and usage using the Linux `/proc/stat` file, and CPU frequency using `scaling_cur_freq` files in the Linux `/sys/devices` file system. The temperature of the core is read from `thermal_zone0/temp` files under `/sys/devices`. The PCB power draw is measured using an ACS712 Hall-effect current sensor and an ADS1115 ADC, which is a low power, 16-bit precision, and I2C compatible ADC. Accelerometer and magnetometer readings are read by an FXOS8700CQ sensor, which has a 14-bit accelerometer and a 16-bit magnetometer. All side channels are collected by a multithreaded framework on the Raspberry Pi Zero. While some sensors, such as HPCs and temperature, are read locally on the Pi Zero, current, accelerometer, and magnetometer readings are read into Raspberry Pi Zero via I2C. While different side channels have different sampling rates, the data collection framework ensures consistent timestamps using an integrated multithreaded architecture with a common time base. Accelerometer and magnetometer readings are collected at a sampling rate of 200 Hz. HPCs, CPU usage, CPU frequency, current, and temperature are collected at 100 Hz.

V. Software-Driven Multimodal Loopback

A. Overview of Approach

As shown in Fig. 3, multimodal loopback Trojan detection applies hardware/software excitations to an uncertain dynamic system (i.e., PCB on which the absence/presence of a Trojan is uncertain) and measuring responses from the system. The system responses are digital and analog side channels in Section IV. The responses can be used to adapt excitations, making them response-dependent and, hence, a "closed loop." The likelihood that the observed responses correspond to expected dynamic behavior based on the known design of the system is probabilistically evaluated. Deviation from the expected behavior is used to detect a Trojan. This provides the defender with multiple moving-target defenses (nature of excitations, responses to measure, and feedback structure from measured responses to dynamic adaptation of excitations), making it hard for the attacker to elude detection. This detection strategy elicits dynamic and closed-loop responses from the PCB system under defender-controlled operating conditions to probabilistically detect anomalies indicating Trojans. We consider two instantiations.

1) Statistical distributions of timing measurements under I/O loopback configurations to detect deviations from expected distributions modeled using PCB design information and, thereby, detect additional unexpected elements along I/O lines. Fuzzing degrees of freedom include which combinations of I/O lines to use for the loopback connections and parameters, such as bit rates.
2) Time series readings from the three-axis magnetometer to probabilistically estimate relay states and detect effects of MITM Trojans that modify the temporal mappings from relay commands to relay switching behavior. Fuzzing degrees include pseudorandom relay commands and variations of time intervals between relay command changes.

B. Golden-Free Anomaly Detection

The loop back-based Trojan detection can be applied in multiple contexts: comparing two PCBs (e.g., comparing a Trojan-inserted PCB with a known-good PCB): a comparison

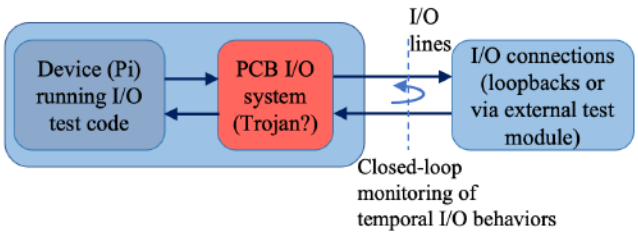

Fig. 3. "Closed-loop" loopback-based Trojan detection.

of a physical PCB with a design-based hypothetical, notional PCB, rather than a physical known-good PCB (i.e., golden-free anomaly detection in the absence of a gold-standard integrated PCB system) and detection of changes within a system over time (i.e., golden-free anomaly detection that relies on self-referencing). Strategies for golden-free Trojan detection are given as follows.

1) Evaluating probabilistic equivalences against expected statistical distributions for different operating conditions (e.g., evaluating probability distributions of expected timing measurements from a digital I/O loopback under different bit rates against expected distributions based on the original design and device datasheets).
2) Using design-based models (e.g., comparing magnetometer readings under relay command outputs from the processor with what is expected).
3) Using design-based models of feature-level side-channel relationships (e.g., correlations between HPC and power readings based on hardware/software designs).
4) Fuzzing hardware parameters, connections, and software test codes to increase the visibility of Trojan effects. By exercising the system over a range of operating conditions, including those not seen in normal operation, the excitations may violate assumptions made by the Trojan (e.g., "normal" bit rates) or push the system to an edge case to expose Trojan effects. Differential validation can model expected behavior variations under excitation changes.

Instead of predicting the exact values of side channels (which may be infeasible in a golden-free setting), the approach evaluates the semantic plausibility of side-channel data based on modeled abstractions/approximations (e.g., probability distributions, PCB-level behaviors, and semantic side-channel interrelations).

C. ML Models

To detect lightweight MITM Trojans using timing measurements under digital I/O loopbacks, the observed time series can be considered as samples from an underlying probability distribution. Empirical methods for comparing probability distributions, such as the Kullback–Leibler (KL) divergence, can be used. In the golden-free setting, the expected probability distribution can be modeled using design-based behavior under different bit rates. This can be refined using measurements from other PCBs and components (e.g., operating system overheads for read/write for a serial port).

To detect anomalies in observed relay command $\longrightarrow$ switching behavior using magnetometers, their readings are used to predict the physical relay states. Since there are four relays on the OpenPLC PCB, this corresponds to estimating a 4×1 relay state vector. Data for training the ML model to predict relay states from magnetometer measurements are obtained in a golden-free setting by toggling the physical relays and collecting magnetometer data (e.g., using an external I2C connection to access magnetometer readings). To mimic low-fidelity data that are generated using simulation-based methods, the empirically collected data are used to estimate one-component Gaussian mixture models that are then used to generate training data. Using synthetic data instead of the raw data for training the ML model introduces an information bottleneck, which mimics the reduced-fidelity data that could be obtained using off-PCB testing or simulation-based methods.

Since the relays close to the magnetometer generate a stronger magnetic response than those that are farther away (see Fig. 1), relay state prediction is difficult for the distant relays. To detect relay states from the fainter signals from distant relays, a progressive/residual approach is applied. Denoting the relays in the order of increasing distance from the magnetometer as $3, 2, 1,$ and 0, the magnetometer readings M (either at one sampling time or over a sliding time window) are used to predict relay states $\hat{r}_3$ and $\hat{r}_2$ for the two nearest relays, which generate strong enough magnetometer responses. Then, using $\hat{r}_3$ and $\hat{r}_2$ as auxiliary information for situational awareness, the data $\{M, \hat{r}_3, \hat{r}_2\}$ are used to predict $\hat{r}_1$, the state of the relay which is third in distance from the magnetometer. Thereafter, using $\hat{r}_1$ and auxiliary information, the data $\{M, \hat{r}_3, \hat{r}_2, \hat{r}_1\}$ are used to predict $\hat{r}_0$, the state of the relay which is farthest from the magnetometer. This progressive/residual approach enables the ML models to sequentially estimate fainter signals by first estimating states of relays closest to magnetometers and then using the estimated states to aid "downstream" ML classifiers of further-away relays. Random forest models [56] are used for the prediction tasks $M \longrightarrow \{\hat{r}_3, \hat{r}_2\}$, $\{M, \hat{r}_3, \hat{r}_2\} \longrightarrow \hat{r}_1$, and $\{M, \hat{r}_3, \hat{r}_2, \hat{r}_1\} \longrightarrow \hat{r}_0$. The generated time series of relay state predictions is then temporally filtered using separate median filters per relay to improve accuracy and robustness to intermittent ambient noise due to other electronic components on/off the PCB. By correlating generated time series of filtered relay state predictions and relay commands transmitted from the processor, anomalies in relay command $\longrightarrow$ switching behavior are flagged, as discussed in Section VI-F.

VI. Results: Trojans/Trojan Detection

Using the developed OpenPLC test bed (see Section III) and the side-channel-based Trojan detection methodologies (see Sections IV and V), experimental results of several sample types of Trojans and their detection accuracy are presented in this section. Specifically, we consider MITM Trojans implemented using the configurable jumpers and firmware on the Pico. The Pico is configured to run at its maximum speed (100 MHz). A Trojan framework was implemented on the Pico including firmware/software components, such as ring buffers, timers, and UART peripherals to facilitate the instantiation of Trojans.

While we consider detection of several types of Trojans, it is to be noted that these are within the unified approach (multi-modal loopback-based detection of MITM Trojans) discussed in Sections III–V in the context of two specific instantiations of the detection methodology (timing-based and magnetometer-based) in Sections VI-B–VI-F. Anomaly detection using timing measurements only relies upon "off-line" design information and expected behaviors of a clean system in terms of software counter-based measurements, variations under bit rate fuzzing, and so on. Anomaly detection using magnetic readings is based on flagging discrepancies relative to an ML model of expected patterns based on "off-line" magnetic calibrations (or low-fidelity simulated data, as discussed in Section V). Hence, both the timing-based and magnetometer-based instantiations represent golden-free applications of the methodology, as further highlighted in Sections VI-B–VI-F. The anomaly detection methodologies do not assume any prior knowledge of the Trojan structures or their effects.

A. GPIO and UART MITM Passthrough Trojans

In Sections VI-B–VI-D, we consider lightweight MITM passthrough Trojans and efficacy of timing side channels to detect these along the signal lines. We consider passthrough Trojans on the GPIO and UART I/O lines. While the PCB design allows for Pico-resident hardware Trojans that interfere with the I2C bus, we focus on Trojans that interfere with the GPIO (such as relay outputs and optoisolator inputs) and UART channels. To implement passthrough MITM Trojans, we set the GPIO and UART jumpers to the "editing" position. For GPIO signals (relays/optoisolators), we implement a polling-based architecture that copies GPIO signals from input to output. For the UART passthrough Trojan, we implement an interrupt-based architecture where the received data are retransmitted on the other channel.

B. Timing Channels Detect GPIO MITM Passthrough Trojans

Since the code on the Pico in the MITM passthrough Trojans is configured to retransmit the signal lines in as lightweight a way as possible, HPCs and magnetometer readings are not affected in observable ways. We detect them using a loopback, as in Fig. 3, where test code on the Pi is configured to implement a loopback. For GPIO, the test code executes a loop in which it toggles the digital output, waits for the digital input to reflect the toggle, and retoggles the output. The elapsed time is recorded after a preset number of toggles (set to 500). The expected time is computed based on the expected operation of the PCB. When the Trojan is present, the elapsed time under this loopback is longer due to the extra hardware and logic. The hardware change between the Trojan-free and Trojan variants for GPIO is shown in Fig. 4(a).

1) Pico-Based GPIO MITM: Fig. 5 (left) shows 5000 samples of the elapsed time for the 500 loopback cycles each for the Trojan-free and Trojan configurations. The time taken for 500 cycles is around 0.1 ms without a Trojan and around 0.4 ms with a Trojan. To evaluate to what extent the Trojan-free measurements can be predicted in a golden-free setting, we can perform "back of the envelope" calculations.

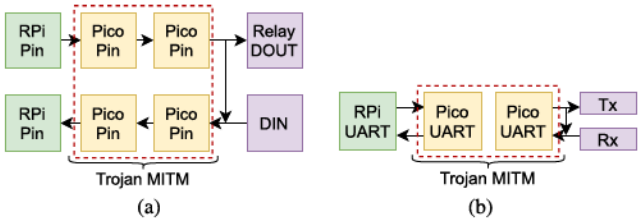

Fig. 4. Trojan MITM diagrams with loopbacks for (a) GPIO and (b) UART.

0.1 ms per sample corresponds to 200 ns per loopback cycle as the measurement sample corresponds to 500 loopback cycles. The peripheral clock on the Pi is 250 MHz or a 4-ns clock period. A test on a generic Raspberry Pi indicates that the system calls for GPIO in user-space require around 200–240 ns to emit and sense loopback. This corresponds closely with the measured Trojan-free time. However, measurements of the passthrough Trojan indicate that the Trojan adds ∼0.3 ms per sample or ∼600 ns per cycle. This extra time is due to the time taken by the Pico to copy the digital input to the digital output. The Pico is slower than the Pi, and this additional time is realistic for the required operation on the Pico.

To analyze the degree to which one can differentiate between the Trojan-free and Trojan configurations using the loopback test, we modified the test code on the Pi to record the number of iterations spent in the delay loop while waiting for the digital input to record the output toggle. This quantity, "n_waits," incremented over the 500 loopback cycles, was measured for the Trojan-free and Trojan configurations. It was observed that n_waits is close to zero without a Trojan as expected since the digital input is physically connected to the digital output in a loopback and reflects the output toggle instantaneously. With a Trojan, there are about five waits per loopback cycle. The additional wait time for the input to toggle is due to the time required by the Trojan to copy inputs to outputs along the output line from Pi and along the input line to Pi.

2) FPGA-Based GPIO Trojan: As the microcontroller-based Trojans are restricted to the clock rate of the device (100 MHz), we also implemented a more lightweight FPGA-based Trojan to study the detectability of even more stealthy MITM Trojans. For this purpose, we implemented a GPIO loopback scenario using a Basys 3 FPGA board (with Artix-7 XC7A35T-1CPG236C) as an FPGA-based MITM Trojan, which was a passthrough-style Trojan capable of disconnecting I/O signals. Fig. 6 shows 5000 samples of elapsed time for 500 loopback cycles in the Trojan-free and Trojan settings. The time required for 500 cycles is approximately 0.16 ms with the Trojan (i.e., ∼0.06 ms more with the Trojan for 500 cycles corresponding to ∼120 ns per cycle). The n_waits measurement incremented over 500 cycles as discussed above was noted to be ∼500 with the Trojan i.e., an average of one wait per loopback cycle compared to ∼0 wait per cycle without the Trojan. This is because, with a pure loopback, the next GPIO read following a write tends to reflect the updated bit flip. However, with an MITM Trojan that adds a small propagation delay, the next read misses the preceding write while the following read sees it (or a later read depending on the added delay). The added ∼120 ns matches the expected

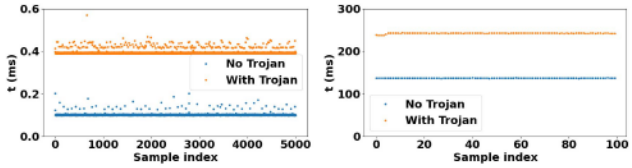

Fig. 5. Measurements of elapsed time in GPIO (left) and UART (right) loopback tests for Trojan-free and Pico-based Trojan configurations.

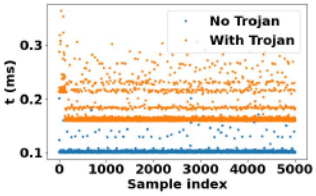

Fig. 6. Measurements of elapsed time in GPIO loopback test for Trojan-free and Trojan configurations with an FPGA as a more lightweight MITM Trojan instead of the Pico microcontroller.

time for one additional read since, as noted above, a pair of system calls for a read-write loopback takes ∼200–240 ns.

C. Timing Channels Detect UART MITM Passthrough Trojans

Fig. 4(b) demonstrates the UART loopback arrangement. As in the GPIO loopback test, the test code is implemented on the Raspberry Pi to execute a transmit–receive loop. The elapsed time is recorded after a preset number (set to 100) of write–read cycles. A time series of 100 measurements under the Trojan-free and MITM Trojan configurations are shown in Fig. 5 (right). We set the UART baud rate to 9600, which is typical for MODBUS communications [57]. To evaluate the extent to which the Trojan-free measurement can be predicted in a golden-free setting, the measured elapsed time in the Trojan-free configuration is ∼135 ms per 100 cycles. This is ∼1.35 ms per cycle and around 13 UART clock cycles. Sending a character with the 8N1 setting (8 bits, no parity bit, one stop bit) of a UART requires 8-bit data + 1 start bit + 1 stop bit = 10 bits. Since UART transmit and receive occur simultaneously, the expected time is 10/9600 = 1.04 ms. The added time with the passthrough Trojan is ∼107 ms per 100 cycles, i.e., ∼1.07 ms per cycle (i.e., around ten UART clock cycles). This extra time is what Pico needs to copy 10 bits from its UART inputs to its UART outputs.

D. Bit-Rate Fuzzing Detects UART MITM Passthrough Trojans

Another defense to detect passthrough UART Trojans is to dynamically vary the UART bit rate (baud). Since the application of the PCB would call for a specific bit rate (e.g., 9600 baud in MODBUS communications), the adversary would assume these bit rates when deploying the Trojan. Even if the adversary considers variable bit rates, they cannot adapt on-the-fly to elude detection. Dynamic bit rate fuzzing is a moving-target defense.

Varying bit rates as a "multirate loopback" test and the averages of elapsed times for 100 loopbacks for Trojan-free and Trojan configurations are shown in Fig. 7 (left). The mean

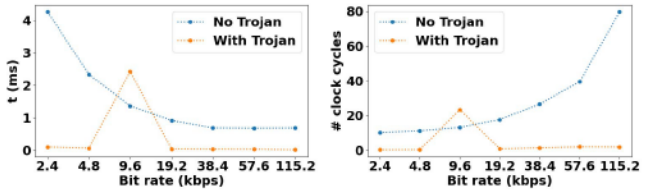

Fig. 7. Averages of elapsed time (left: in ms and right: in elapsed cycles) in multirate UART loopback test for Trojan-free and Trojan configurations.

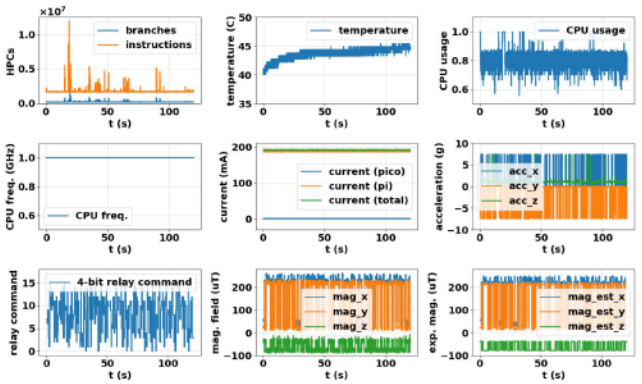

Fig. 8. Side-channel data collected with Pico passthrough configuration. In plot of current measurements, the total current is slightly higher than Pi current.

time measurements represented as numbers of clock cycles for one loopback cycle are shown in Fig. 7 (right). It is seen that increasing the bit rate decreases elapsed times due to faster communications in the Trojan-free configuration and saturates due to intrinsic processing overhead on Pi. Measurements with a Trojan are, however, anomalous. The elapsed times are spuriously small when the bit rates mismatch between Pi and Pico, and data corruption causes spurious reads.

It can be seen that, at low bit rates, the number of UART cycles corresponds to the required minimum number of cycles to transmit/receive. As the bit rate increases, the processing overhead on the Pi increases the number of cycles (since each cycle is shorter). These trends are expected in a Trojan-free configuration. In the Trojan configuration, a similar anomaly, as in Fig. 7 (left), is seen, where the mismatch of bit rates between the Pi and the Pico causes spurious small numbers of apparent clock cycles, indicating an unexpected element (i.e., MITM Trojan) on the signal lines.

Since the bit rate mismatch between Pi and Pico may corrupt the data, comparing the transmitted and received bytes can also enable the detection of MITM anomalies. For this purpose, fractions of matches between transmitted and received bytes were measured for Trojan-free and Trojan configurations. While there is a perfect match between transmitted and received bytes at the nominal bit rate of 9600 baud in both Trojan-free and Trojan configurations and at any bit rate in the Trojan-free configuration, the match rate degrades to close to zero in the Trojan configuration at bit rates other than 9600 baud. The discrepancy between Trojan-free and Trojan configurations, thus, detects the MITM Trojan.

UART-based fuzzing is simplified in our process as we do not consider specific external UART devices in the Trojan designs and experimental tests. This is because our process is

TABLE I
RELAY STATE PREDICTION VIA MAGNETIC CHANNEL

Relay	Accuracy	Precision	Recall	F1-Score
3	0.9996	1.0000	0.9992	0.9996
2	0.9996	0.9992	1.0000	0.9996
1	0.9956	0.9961	0.9950	0.9956
0	0.9467	0.9351	0.9600	0.9474

intended to occur at predeployment, prior to PLC interconnection with connected devices. The loopback detection functions by connecting the PCB's UART transmit port to the receive port. As such, any Trojan on the PCB that can edit the UART will interfere with the timing of this loopback, making them detectable irrespective of any external connections.

Though out of scope in this work, if external devices were required to be connected, these tests could still occur. In this scenario, defender-controlled challenge/response pairs to/from external devices would also feature alternative timing when an MITM Trojan was present.

E. Magnetic Channels Detect Relay Behavior

When applying the ML methodology for magnetometer-aided prediction of relay states in Section V, the accuracy for each of the four relays is summarized in Table I. As seen in Fig. 1, relays 3, 2, 1, and 0 are in the order of increasing distance from the magnetometer. Since the magnetometer-visible signals corresponding to the relays are progressively fainter, the accuracy of relay state estimation reduces as the distance from the magnetometer increases. The accuracy for the 4×1 relay state vector was $\sim$0.945.

Before proceeding to the quantitative results of Trojan detection in Section VI-F, it is worthwhile to consider a quick "back-of-the-envelope" calculation on whether accurate magnetometer-based relay state estimation translates into robust anomaly detection. With a probability p of correct detection of the relay state from magnetometer measurements, we expect a clean system to yield around $\mu_C = pN$ matches of commanded and magnetometer-based estimated relay states over N magnetometer measurement samples. For simplicity, we consider a binomial distribution with matches at each sampling time considered to be independent random variables. The standard deviation σ_C of the number of matches is $(Np(1-p))^{1/2}$. Consider a Trojan that modifies relay commands for one or more relays for at least some fraction η of the total time. Since we consider pseudorandom relay commands when testing for Trojans in Section VI-F, $(1/2^n)$ is the probability that the defender-commanded Trojan-overridden relay commands are identical (where $n = 4$ is the number of relays). Hence, with the Trojan, we expect around $\mu_T = \tilde{p}N$ matches, where $\tilde{p} = p(1-\eta(1-1/2^n))$. While, as noted above, $p \approx 0.945$ empirically, consider a less precise estimation $p = 0.8$ to evaluate Trojan detection feasibility. For $N = 1000$ and $\eta = 0.05$, we obtain $\mu_C = 800$, $\sigma_C = 12.65$, and $\mu_T = 762.5$. Hence, μ_T is three standard deviations away from the expected number of matches μ_C in a clean system making the observed distribution of matches over a time interval highly unlikely based on the hypothesis of a clean system. Over longer time intervals, the likelihood of detecting the anomaly is even higher (with $N = 10\,000$ and $\mu_T > 9$ standard deviations away from μ_C). Therefore, Trojans that manipulate the relay commands even a very small fraction of time can be detected.

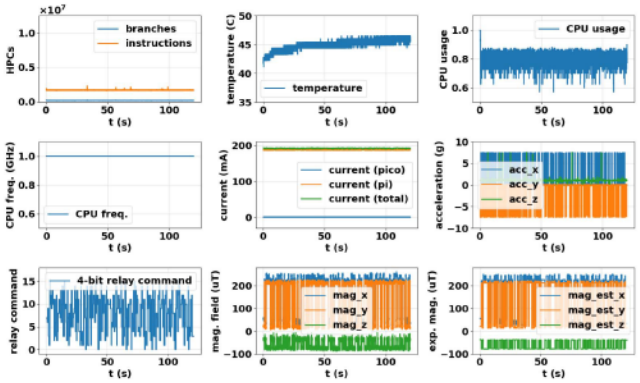

Fig. 9. Side-channel data collected with invert Trojan.

F. Magnetic Channels Detect Relay Command Edit Trojans

In this section, we consider Trojans that affect relay behavior. These relay-based Trojans are implemented as MITM behaviors on the Pico and, therefore, can also be detected using the timing loopback tests discussed above. However, we also study the detection of the Trojan effects using the magnetic side channel. This addresses the potential scenario in which the Pico is replaced by a much faster processor, as this may reduce the injected delays in signal retransmission such that they are not detectable by loopback times. Hence, we use the magnetic side channel to monitor the integrity of the temporal mappings between Pi-emitted relay commands and the relay switching behavior. To determine the efficacy of magnetic side channels to detect temporal anomalies, we consider the Pico passthrough configuration as the baseline. Sample data collected from the PCB when it is in Pico passthrough mode are shown in Fig. 9. Besides side channels discussed in Section IV, the figure shows relay commands that are sent by Pi and expected (average) magnetometer readings for the relay commands (using Gaussian model in Section V).

We employ the ML approach from Section V to evaluate anomaly likelihoods from sliding time windows of collected side-channel data. We apply the relay state prediction first to generate a time series of predicted relay states (as 4×1 vectors). To excite the relays, we transmit a sequence of pseudorandom commands from the Pi during the test. The application of pseudorandom relay commands also represents a moving-target defense since the attacker does not know the particular sequences of relay commands that are applied during the testing of a PCB. We compare the predicted relay states against the time series of relay commands. By combining binary indicators of per-relay matches/mismatches with different weights for different relays, we compute a time series of magnetometer anomaly likelihood scores. We use higher weights for relays closer to the magnetometer because the state of these relays can be determined more reliably by the magnetic readings. A mean filter temporally filtered the time series of anomaly likelihood scores to reduce false positives and false negatives. During temporal averaging, we ignore the time intervals around the relay switching in order to avoid spurious false positives during transients of physical relay

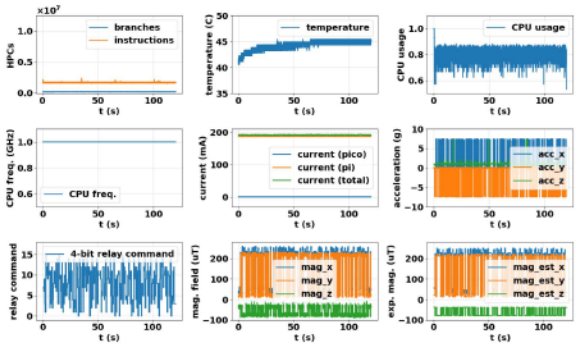

Fig. 10. Side-channel data collected with delay Trojan.

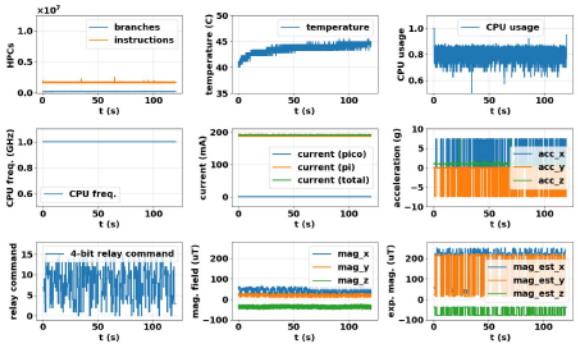

Fig. 11. Side-channel data collected with replay Trojan.

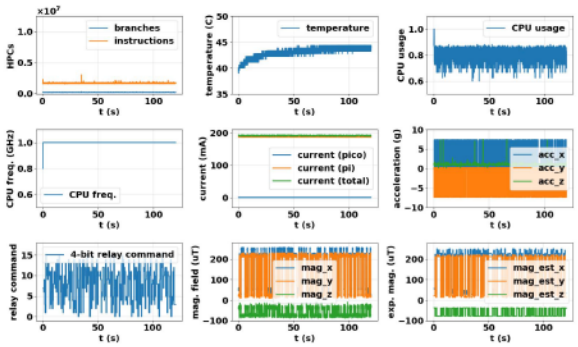

Fig. 12. Side-channel data collected with acoustic Info. Leakage Trojan.

switching. Since we consider Trojans that persist over time, the characteristic of whether or not a PCB is Trojan-injected does not change. Evaluation of the PCB requires considering the majority label over sliding time windows. As long as the Trojan-free versus Trojan determination is correct over 50% of the time, the evaluation will be correct.

By configuring the code on the Pico, we implemented MITM Trojans that modify the temporal mapping of relay commands to their switching behavior.

The side-channel data collected with the MITM Trojans are shown in Figs. 9–12.

1) **Invert Trojans** capture and emit all I/O channels. They invert the last two bits of relay outputs (relays 1 and 0). If relay 1 or 0 is commanded to open, they will close and vice versa. We choose these relays to test since they are the farthest from the magnetometer and are the hardest to measure making them challenging to detect.

TABLE II
TROJAN DETECTION ACCURACY USING SLIDING TIME WINDOWS OF SIDE-CHANNEL DATA

Trojan	Accuracy	Precision	Recall	F1-Score
Invert Trojan	0.877	0.911	0.836	0.872
Delay Trojan	0.959	0.925	1.000	0.961
Replay Trojan	0.959	0.925	1.000	0.961
Acoustic Info. Leakage Trojan	0.943	0.922	0.968	0.945

2) **Delay Trojan** injects a lag between the external I/O pins and the core Pi's I/O. For the GPIO (relay/optoisolator) signals, it samples every 10 ms and stores the results in a ring buffer with 100 elements. Elements are emitted when they are overwritten. All GPIO has an extra delay of 1 s. For UART signals (including RS-485), the Pico is configured to use interrupts to store characters in a 1024-element ring buffer. With each character, it stores an emit timestamp of 1 s in the future after capture. A second timer interrupt-based loop checks and emits the characters at appropriate times.
3) **Replay Trojan** extends the delay Trojan. The received data are stored in ring buffers. It is emitted when captured, so the delay is minimal. However, upon receiving a *trigger signal*, the Trojan enters a *replay mode*, where new signals are neither captured nor propagated. Instead, the previous 10 s of the ring buffer is emitted in a loop. In the tests, the trigger signal was applied before the side-channel data were collected.
4) **Acoustic information leakage Trojan** is a variant of the replay Trojan, which focuses on the UART/RS-485 bus. The captured data are stored in a 1024-element ring buffer. Upon data entering the buffer, an override method takes control of the relay 0 signal and "clicks" the relay in an audible pattern. A listener can capture the signal. This Trojan exfiltrates the UART data via audio. Since this Trojan leaks UART data as relay clicks, the magnetometer readings can detect anomalies relative to expected readings based on transmitted relay commands.

Table II summarizes the detection accuracy of Trojans over sliding time windows of the side-channel data. The false positive rate is ∼8.15%. This false positive rate quantifies the accuracy, precision, and F1-score in Table II. The recall is the % of time windows of the data from the Trojan configuration marked anomalous. When classifying a PCB as Trojan-injected or not, it is sufficient to consider the majority label over the sliding time windows of the binary Trojan likelihood estimates.

VII. CONCLUSION

This study demonstrates using side-channel signals for anomaly detection in embedded PCB systems using multimodal closed-loop monitoring of temporal properties under software-driven excitations and defender-controlled I/O. When measured against design-based baselines, timing measurements from I/O channels can detect lightweight MITM Trojans. Using inputs and fuzzing from hardware/software makes Trojan effects visible. It supports a moving-target defense where inputs and fuzzing are controlled by the defender. For our studies, we created a flexible and reconfigurable test bed, "OpenPLC NYU TE."

REFERENCES

[1] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, "Towards a comprehensive and systematic classification of hardware Trojans," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1871–1874.

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

[3] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Design Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, Dec. 2016.

[4] S. Paley, T. Hoque, and S. Bhunia, "Active protection against PCB physical tampering," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 356–361.

[5] M. McGuire, U. Ogras, and S. Ozev, "PCB hardware trojans: Attack modes and detection strategies," in *Proc. IEEE 37th VLSI Test Symp. (VTS)*, Apr. 2019, pp. 1–6.

[6] G. Piliposyan, S. Khursheed, and D. Rossi, "Hardware Trojan detection on a PCB through differential power monitoring," *IEEE Trans. Emerg. Topics Comput.*, early access, Nov. 3, 2020, doi: 10.1109/TETC.2020.3035521.

[7] 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, p. 42:1–42:25, 2020.

[8] J. Robertson and M. Riley. (Feb. 2021). *The Long Hack: How China Exploited a U.S. Tech Supplier*. Bloomberg. [Online]. Available: https://www.bloomberg.com/features/2021-supermicro/

[9] (Jul. 2019). *Safeguards Against Hidden Effects and Anomalous Trojans in Hardware (SHEATH)*. Defense Advanced Research Projects Agency (DARPA), Solicitation DARPA-PA-19-04-01. [Online]. Available: https://sam.gov/opp/25d32ec9a35117576b04d52896267319/view

[10] N. Asadizanjani, S. Shahbazmohamadi, M. Tehranipoor, and D. Forte, *Non-Destructive PCB Reverse Engineering Using X-Ray Micro Computed Tomography*. Portland, OR. USA: ASM Int., Nov. 2015, pp. 164–172.

[11] T. Alves. (Nov. 2021). *The OpenPLC Project*. [Online]. Available: https://www.openplcproject.com

[12] S. Bhunia *et al.*, "Protection against hardware trojan attacks: Towards a comprehensive solution," *IEEE Design Test*, vol. 30, no. 3, pp. 6–17, Jun. 2013.

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

[14] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2014, pp. 1–6.

[15] N. Shang, A. Wang, Y. Ding, K. Gai, L. Zhu, and G. Zhang, "A machine learning based golden-free detection method for command-activated hardware Trojan," *Inf. Sci.*, vol. 540, pp. 292–307, Nov. 2020.

[16] T. Hoque, S. Narasimhan, X. Wang, S. Mal-Sarkar, and S. Bhunia, "Golden-free hardware trojan detection with high sensitivity under process noise," *J. Electron. Test.*, vol. 33, no. 1, pp. 107–124, Feb. 2017.

[17] S. Faezi, R. Yasaei, A. Barua, and M. A. A. Faruque, "Brain-inspired golden chip free hardware trojan detection," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2697–2708, 2021.

[18] V. R. Surabhi *et al.*, "Hardware Trojan detection using controlled circuit aging," *IEEE Access*, vol. 8, pp. 77415–77434, 2020.

[19] V. R. Surabhi, P. Krishnamurthy, H. Amrouch, J. Henkel, R. Karri, and F. Khorrami, "Exposing hardware Trojans in embedded platforms via short-term aging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3519–3530, 2020.

[20] V. R. Surabhi, P. Krishnamurthy, H. Amrouch, J. Henkel, R. Karri, and F. Khorrami, "Trojan detection in embedded systems with FinFET technology," *IEEE Trans. Comput.*, early access, Jan. 27, 2022, doi: 10.1109/TC.2022.3146217.

[21] S. Ghosh, A. Basak, and S. Bhunia, "How secure are printed circuit boards against trojan attacks?" *IEEE Design Test*, vol. 32, no. 2, pp. 7–16, Apr. 2015.

[22] N. Boggs, J. C. Chau, and A. Cui, "Utilizing electromagnetic emanations for out-of-band detection of unknown attack code in a programmable logic controller," in *Proc. SPIE*, vol. 10630, Sep. 2018, Art. no. 106300D.

[23] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2939–2948, Oct. 2017.

[24] X. T. Ngo, Z. Najm, S. Bhasin, S. Guilley, and J.-L. Danger, "Method taking into account process dispersions to detect hardware Trojan horse by side-channel," in *PROOFS: Security Proofs for Embedded Systems*. Busan, South Korea: Springer, Sep. 2014.

[25] J. Park, V. R. Surabhi, P. Krishnamurthy, S. Garg, R. Karri, and F. Khorrami, "Anomaly detection in embedded systems using power and memory side channels," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2020, pp. 1–2.

[26] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori, "Run-time hardware Trojan detection using performance counters," in *IEEE Int. Test Conf. (ITC)*, Fort Worth, TX, Nov. 2017, pp. 1–10.

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

[28] 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, 2020.

[29] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Gothenburg, Sweden, Sep. 2014, pp. 109–129.

[30] S. R. Chhetri, A. Canedo, and M. A. A. Faruque, "Confidentiality breach through acoustic side-channel in cyber-physical additive manufacturing systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 2, no. 1, pp. 1–25, Jan. 2018.

[31] C. Malone, M. Zahran, and R. Karri, "Are hardware performance counters a cost effective way for integrity checking of programs," in *Proc. 6th ACM Workshop Scalable Trusted Comput.*, Oct. 2011, pp. 71–76.

[32] 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. 2022.

[33] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2013, pp. 532–539.

[34] P. Godefroid, M. Y. Levin, and D. Molnar, "Sage: Whitebox fuzzing for security testing: Sage has had a remarkable impact at Microsoft," *Queue*, vol. 10, no. 1, pp. 20–27, 2012.

[35] H. Yoo and T. Shon, "Grammar-based adaptive fuzzing: Evaluation on SCADA modbus protocol," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Nov. 2016, pp. 557–563.

[36] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A Modbus/TCP fuzzer for testing internetworked industrial systems," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom. (ETFA)*, Luxembourg, Luxembourg, Sep. 2015, pp. 1–6.

[37] B. Cui, S. Liang, S. Chen, B. Zhao, and X. Liang, "A novel fuzzing method for zigbee based on finite state machine," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 1, Jan. 2014, Art. no. 762891.

[38] D. Zhang, J. Wang, and H. Zhang, "Peach improvement on profinet-DCP for industrial control system vulnerability detection," in *Proc. Int. Conf. Electr., Comput. Eng. Electron.*, May 2015, pp. 1–6.

[39] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

[40] E. G. Barrantes, D. H. Ackley, S. Forrest, T. S. Palmer, D. Stefanovic, and D. D. Zovi, "Randomized instruction set emulation to disrupt binary code injection attacks," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, Oct. 2003, pp. 281–289.

[41] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, Oct. 2003, pp. 272–280.

[42] R. Hund, C. Willems, and T. Holz, "Practical timing side channel attacks against kernel space ASLR," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2013, pp. 191–205.

[43] J. Seibert, H. Okhravi, and E. Söderström, "Information leaks without memory disclosures: Remote side channel attacks on diversified code," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 54–65.

[44] E. Al-Shaer, *Toward Netw. Configuration Randomization for Moving Target Defense*. New York, NY, USA: Springer, 2011, pp. 153–159.

[45] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 127–132.
[46] H. Okhravi *et al.*, "Creating a cyber moving target for critical infrastructure applications," in *Proc. 5th Int. Conf. Crit. Infrastruct. Protection (ICCIP)*, Hanover, NH, USA, Mar. 2011, pp. 107–123.
[47] D. C. MacFarland and C. A. Shue, "The SDN shuffle: Creating a moving-target defense using host-based software-defined networking," in *Proc. 2nd ACM Workshop Moving Target Defense*, Oct. 2015, pp. 37–41.
[48] K. Wang, X. Chen, and Y. Zhu, "Random domain name and address mutation (RDAM) for thwarting reconnaissance attacks," *PLoS ONE*, vol. 12, no. 5, May 2017, Art. no. e0177111.
[49] Y. Mei, K. Gogineni, T. Lan, and G. Venkataramani, "MPD: Moving target defense through communication protocol dialects," 2021, *arXiv:2110.03798*.
[50] B.-Y. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot, "Analysis of point-to-point packet delay in an operational network," *Comput. Netw.*, vol. 51, no. 13, pp. 3812–3827, Sep. 2007.
[51] B. Aziz and G. Hamilton, "Detecting man-in-the-middle attacks by precise timing," in *IEEE Int. Conf. Emerg. Secur. Inf., Syst. Technol.*, Athens, Greece, Jun. 2009, pp. 81–86.
[52] V. A. Vallivaara, M. Sailio, and K. Halunen, "Detecting man-in-the-middle attacks on non-mobile systems," in *Proc. 4th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2014, pp. 131–134.
[53] A. T. Sherman, J. Seymour, A. Kore, and W. Newton, "Chaum's protocol for detecting man-in-the-middle: Explanation, demonstration, and timing studies for a text-messaging scenario," *Cryptologia*, vol. 41, no. 1, pp. 29–54, Jan. 2017.
[54] Raspberry Pi (Trading) Ltd. *Buy a Raspberry Pi Zero*. Accessed: May 5, 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero/
[55] (Nov. 2021). *Raspberry Pi Pico Datasheet*. Raspberry Pi (Trading) Ltd. [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf
[56] *Random Forest Classifier Implementation in Scikit-Learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.Rand%omForestClassifier.html
[57] *Modbus RTU Protocol Overview*. Accessed: May 5, 2022. [Online]. Available: https://www.rtautomation.com/technologies/modbus-rtu/

Hammond Pearce (Member, IEEE) received the B.E. degree (Hons.) in computer systems engineering and the Ph.D. degree in computer systems engineering from The University of Auckland, Auckland, New Zealand, in 2014 and 2020, respectively.

In 2019, he took part in the NASA International Internship Program and worked at the NASA Ames Research Center, Mountain View, CA, USA. He is currently a Research Assistant Professor with the Department of Electrical and Computer Engineering and the NYU Center for Cybersecurity, New York University (NYU), Brooklyn, NY, USA. His research focus is on industrial cybersecurity, including additive manufacturing and industrial informatics. His other research interests include IoT, cyber–physical systems (CPSs), compilers, and AI/machine learning (ML).

Virinchi Roy Surabhi received the B.Tech. degree in electrical engineering from IIT Kanpur, Kanpur, India, in 2018. He is currently working toward the Ph.D. degree at the NYU Tandon School of Engineering, Brooklyn, NY, USA.

His research interests include robotics, artificial intelligence, hardware security, cyber–physical systems (CPSs), and CPS security.

Prashanth Krishnamurthy (Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Madras, Chennai, India, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Polytechnic University [now New York University (NYU)], Brooklyn, NY, USA, in 2002 and 2006, respectively.

He is currently a Research Scientist and an Adjunct Faculty with the Department of Electrical and Computer Engineering, NYU, and a Senior Researcher with FarCo Technologies, New York, NY, USA. He has coauthored over 140 journal articles and conference papers, and a book. His research interests include autonomous vehicles and robotic systems, multiagent systems, data fusion, robust adaptive nonlinear control, machine learning, electromechanical systems modeling and control, cyber–physical systems and cybersecurity, large-scale systems, high-fidelity and hardware-in-the-loop simulation, and real-time embedded systems.

Joshua Trujillo received the Ph.D. degree in computer engineering from The University of New Mexico, Albuquerque, NM, USA, in 2015.

He is currently a Lead Engineer with Honeywell FM&T, Kansas City National Security Campus, Department of Energy, Kansas City, MO, USA. He holds five U.S. patents.

Ramesh Karri (Fellow, IEEE) received the B.E. degree in electrical and computer engineering (ECE) from Andhra University, Visakhapatnam, India, in 1985, and the Ph.D. degree in computer science from the University of California at San Diego (UC San Diego), La Jolla, CA, USA, in 1993.

He is currently a Professor of ECE with New York University (NYU), Brooklyn, NY, USA. He codirects the NYU Center for Cyber Security. He cofounded the Trust-Hub and founded the Embedded Systems Challenge, the annual red team blue team event. He has published over 300 articles in leading journals and conference proceedings. His research and education are in hardware cybersecurity, including trustworthy integrated circuits (ICs), processors, cyber–physical systems, security-aware computer-aided design, tests, verification, nanomeets' security, hardware security competitions, benchmarks and metrics, and additive manufacturing security.

Farshad Khorrami (Senior Member, IEEE) received the bachelor's degree in mathematics, the bachelor's degree in electrical engineering, the master's degree in mathematics, and the Ph.D. degree in electrical engineering from The Ohio State University (OSU), Columbus, OH, USA, in 1982, 1984, 1984, and 1988, respectively.

In 1988, he joined New York University (NYU), Brooklyn, NY, USA, where he is currently a Professor of Electrical and Computer Engineering (ECE). He has developed the Control/Robotics Research Laboratory, Brooklyn. He has published over 300 refereed journal articles and conference papers. His book *Modeling and Adaptive Nonlinear Control of Electric Motors* was published by Springer Verlag in 2003. He holds 14 U.S. patents. His research has been supported by the Army Research Office (ARO), NSF, the Office of Naval Research (ONR), DARPA, the Army Research Laboratory (ARL), the Air Force Research Laboratory (AFRL), NASA, and several corporations. His research interests include adaptive and nonlinear controls, robotics and unmanned vehicles, cyber security for cyber–physical systems, embedded systems security, machine learning, and large-scale systems.

Dr. Khorrami has served as a conference organizing committee member of several international conferences.