Hardware Root-of-Trust Support for Operational Technology Cybersecurity in Critical Infrastructures

Alan Ehret, Peter Moore, Milan Stojkov, and Michel A. Kinsy
Secure, Trusted, and Assured Microelectronics Center
Ira A. Fulton Schools of Engineering, Arizona State University

***Abstract*— Operational technology (OT) systems use hardware and software to monitor and control physical processes, devices, and infrastructure - often critical infrastructures. The convergence of information technology (IT) and OT has significantly heightened the cyber threats in OT systems. Although OT systems share many of the hardware and software components in IT systems, these components often operate under different expectations. In this work, several hardware root-of-trust architectures are surveyed and the attacks each one mitigates are compared. Attacks spanning the design, manufacturing, and deployment life cycle of safety-critical operational technology are considered. The survey examines architectures that provide a hardware root-of-trust as a peripheral component in a larger system, SoC architectures with an integrated hardware root-of-trust, and FPGA-based hardware root-of-trust systems. Each architecture is compared based on the attacks mitigated. The comparison demonstrates that protecting operational technology across its complete life cycle requires multiple solutions working in tandem.**

I. Introduction

Operational Technology (OT) systems consist of sensor inputs to understand an environment, actuation outputs to impact an environment, and monitoring outputs to report system and environment status to human operators. Often, an OT system's interaction with the physical environment makes its operation safety-critical. Nuclear Power Plants (NPPs) represent a good illustrative example. The potential public safety risks for these plants are great. Therefore, they do require high-security and high-assurance of their OT to prevent and mitigate cyber-attacks that aim to degrade safety and functionality [1]. System architects and operators face several challenges when maintaining the security of OT systems in NPPs. Complex supply chains create risks of counterfeit parts [2], [3]. Integration by third parties provides attackers an opportunity to tamper with otherwise trusted components and firmware [4], [5]. Maintenance often requires physical access, opening systems to risk of physical access-based attacks, including fault injections [6].

The safety-critical nature of OT systems makes them valuable targets for attack. In May 2021, the Colonial Pipeline operations were disrupted by a ransomware attack on the company's infrastructure [7]. In December 2015, a cyber-attack on the Ukrainian power grid caused power outages for approximately 225,000 people for several hours [8]. As safety-critical OT systems become more connected, attackers will have more opportunities to infiltrate and disrupt them. Future OT systems must be designed with security in mind to ensure their safe operation well into the future.

The use of hardware root-of-trust (RoT) solutions helps mitigate many of the OT security challenges by enabling a system to validate the integrity of hardware and software sub-components. Hardware RoT solutions leverage intrinsic hardware characteristics to create the first link in a chain of trust throughout the system. One characteristic is the static nature of hardware; a physical component is not as easily changed or updated as a software component. Another characteristic is the difficulty of directly observing the internal hardware state of integrated circuits due to both small size and time scales.

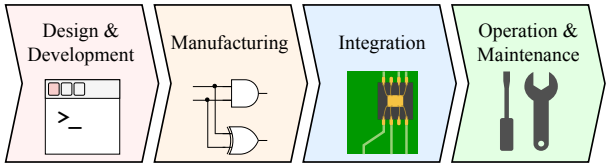

Fig. 1: Operational Technology life-cycle.

In all, OT systems are complex heterogeneous compute and control systems with legacy software and hardware, third party devices, ad-hoc or changing connected networks, etc. These systems face many cyber threats including ransomware, malware, insider threat, denial of service (DoS) attacks, among others. Therefore, addressing security concerns in OT systems presents many challenges covering software, firmware, hardware, and networking aspects. In this work, we focus our analysis on the key hardware-centric security aspects of OT systems, in particular, how cybersecurity concerns are addressed using hardware root-of-trust techniques. Three categories of hardware roots-of-trust, including hardware RoT peripheral devices, integrated System-on-Chip (SoC) architectures, and FPGA-based solutions are examined. Each solution is compared based on the attacks mitigated. The scope of attacks considered spans component manufacturing time, system integration, deployment, and operation. Based on the survey, we select a subset of attacks to use as a benchmark when evaluating future hardware RoT architectures for OT systems. Directions for future research are discussed.

II. Threat Model

Attacks against OT systems can occur at any stage of the device life-cycle. In this work, we consider four major OT system life-cycle stages, as presented in Figure 1:

1) **Design and Development:** The software and hardware of an OT system are described in preparation for manufacturing. Software is written, and schematics for the hardware are created.
2) **Manufacturing:** Individual components (integrated circuits, printed circuit boards, system enclosures, etc.) are fabricated.
3) **Integration:** Individual components are incorporated into sub-modules and complete systems.
4) **Operation and Maintenance:** Systems are deployed to the field, with regular inspection and repair.

One or more un-trusted parties in the supply chain may act in a malicious manner to attack a system. Previous attacks on infrastructure have been attributed to nation-states [9] and demonstrated risks to a victim's national security [10]. Therefore, this work assumes attackers are sophisticated, with plenty of resources at their disposal. Attacker capabilities are described in terms of attack potential, defined in [11]. Attacker capabilities are broken down into six categories: elapsed time, expertise, knowledge of the Target of Evaluation (TOE), access to TOE samples, equipment and tools, and window of opportunity.

Elapsed Time: Due to the long deployments of OT systems, attackers are assumed to have months or years to develop attacks.

Expertise: Supply-chain attackers are assumed to possess proficient-level understanding of attack paths, techniques, and tools.

Knowledge of TOE: Supply-chain attackers are assumed to have access to sensitive information related to their supply-chain task.

Access to TOE samples: Before integration time, attackers will not have access to a complete system, unless it is obtained as an end-user. After integration time, attackers will have access to nearly complete systems.

Equipment and Tools: Supply-chain attackers will have access to specialized equipment related to their position in the design, manufacturing and integration process.

Window of Opportunity: An attacker has the entire time frame required to complete their design, manufacturing, or integration task to execute an attack. Attackers are assumed to have a window of less than a day to execute attacks after deployment, i.e. during maintenance, due to access controls and high up-time requirements.

III. Hardware RoT Peripheral Devices

A. Smart Cards

Smart cards provide a hardware RoT in a small form factor that is suitable for use alongside a more complex compute element, such as a SoC or FPGA [12]. Smart cards provide a number of cryptographic functions that may be used in protocols such as measured boot in order to verify and attest to the state of the hardware and firmware of the host machine [13]. These functions include secure key generation and storage, encryption/decryption, and signature generation [14].

Isolating the cryptographic functions can protect them from hardware or software attacks against the host machine and allow the smart card to act as a RoT for the host. This allows smart cards to protect against hardware attacks such as malicious insertion or substitution and software attacks such as firmware and boot process manipulation. Additionally, smart cards' secure key storage prevents attackers from gaining access to these keys through RAM/storage manipulation. In addition to built-in features, smart cards can support user-defined applications to provide additional custom security features. Support for custom software allows system developers to tailor security features to specific system requirements and threats.

Systems may interface with smart cards through a card reader or by directly mounting discretely packaged versions of the Integrated Circuits (IC) inside a card on the system's printed circuit board. Smart Card ICs generally communicate with the host machine using serial bus communications, including Inter-integrated Circuit (I2C) or Serial Peripheral Interface (SPI) [12], [15], or with a wireless protocol such as Near-Field Communication (NFC) [16]. Two common smart card operating systems are Java Cards and MULTOS [14]. MULTOS has greater security restrictions for its applications, generally providing greater security. However, these extra restrictions increase the programming complexity for MULTOS applications [12].

B. OpenTitan

OpenTitan is an open source silicon RoT designed to provide a number of security features across a variety of use cases, including acting as a TPM [17]. OpenTitan supports customization and third-party validation. Provided security features include full boot attestation measurements, a key manager implementation that is only able to unlock stored secrets if its boot chain is signed correctly, and a key versioning scheme that supports updating.

OpenTitan's TPM features focus on supporting a secure boot process, which begins in ROM. As the ROM cannot be updated after manufacturing, it only performs basic setup and authentication of the next boot stage. The enhanced Physical Memory Protection (ePMP) registers ensure that only the code of the current stage of boot is executable. The ROM extension (ROM_EXT) region is an updatable region responsible for initializing the key manager and performing boot services such as attestation of device state. After setting up the key manager, attesting to the device state, and configuring the ePMP to allow for standard execution and block writes to the boot stages, the ROM_EXT jumps to the start of the device owner's code and secure boot ends [18]. This robust boot process, along with its other security features, protect OpenTitan from a number of attacks, including firmware manipulation, boot process manipulation, malicious insertion or substitution, and scan-chain state modification. Other TPM implementations follow a similar boot process that starts with ROM [19]. However, unlike some other hardware RoT designs such as Apple's T2 chip [20], OpenTitan is open source, making it usable and verifiable by third parties.

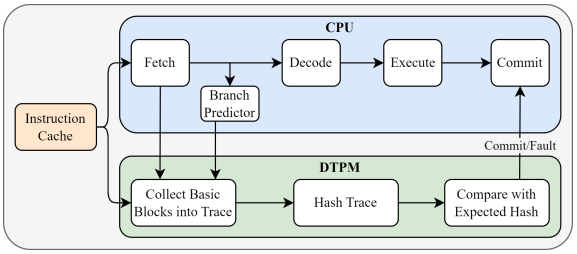

Fig. 2: The DTPM Architecture.

An implementation of the OpenTitan is the OpenTitan Earl Grey microcontroller, a low-power secure microcontroller. It has a single RV32IMCB core that achieves a CoreMark per MHz of 2.36 [18], similar to Intel's Pentium MMX (2.33) or Core 2 Duo T7200 (2.64) [21]. Power and area usage are not reported in OpenTitan's design specifications.

IV. Integrated Hardware RoT SoC Architectures

A. DTPM

The Dynamic Trusted Platform Module (DTPM), shown in Figure 2, is an on-chip TPM merged with the processor pipeline. The DTPM is similar to a TPM in that it is also used to verify executables. However, TPMs only verify these programs at load time, meaning an attacker can modify the program after it is loaded. The DTPM is designed to check programs at runtime to defend against attacks after program load time [22].

The DTPM verifies programs at runtime by calculating the hash of each trace in a program and comparing that to pre-computed hashes stored on disk. In order to balance pipeline stalls, number of hashes, and the size of the input to the SHA-1 hash function used, each trace is made up of at most four basic blocks. Each basic block is a section of code with one entry point and one exit point. To verify these traces, the DTPM must be built directly into the processor pipeline. Each time a new trace is loaded into the CPU, the DTPM begins hashing and verifying the trace. The trace must be verified before any instruction in the trace can commit.

While DTPM provides mitigation against runtime control-flow attacks, it carries significant performance overheads - including an average execution cycle increase of 250x. Thus, its designers implemented a number of different optimizations to reduce this impact. The first is to only check the 10% of the code that is most frequently executed, as processors spend 90% of their time executing this code. The second optimization is a 32 KB Hash Trace Cache (HTC). The HTC caches the hashes fetched from the disk, thus reducing the number of disk accesses needed. These optimizations reduce the performance impact from 250x to 1.35x. A third optimization is to

use part of the RAM in the operating system address space as another layer of caching. This reduces the performance impact from 1.35x to 1.18x. While the DTPM is the only surveyed hardware RoT that can defend against runtime control-flow attacks, the protection of only the 10% most commonly executed code means that programs running on a machine with an active DTPM are only protected from these attacks 90% of the time.

The DTPM has an area overhead of 2.5%. Its total dynamic power at maximum frequency was 0.0968 W, and its total standby leakage power was 0.0257 W. The DTPM does not require Instruction Set Architecture (ISA) or code modification [22].

B. RECORD

Reconfigurable Edge Computing for Optimum Resource Distribution (RECORD) is a System-on-Chip (SoC) architecture designed with a hardware RoT for low-power edge devices [23]. RECORD is made up of the RoT Unit, an RV32I RISC-V core, and domain-specific accelerators. The RISC-V core and the accelerators share a dedicated bus and communicate through data memory and are not logically connected to the RoT unit to isolate it from attack.

The RECORD Root-of-Trust Unit includes several features designed to mitigate attacks against unattended edge systems, where attackers may gain physical access. RECORD aims to mitigate firmware manipulation, boot process manipulation, malicious insertion and substitution, scan-chain state modification, off-chip RAM modification, clock fault injections, and timing side channels. RECORD does not defend against other side-channel attacks such as power and Electromagnetic (EM) side-channels.

The RoT unit is made up of a Built-In Self-Test (BIST) module, data bus access control modules, a programmable Finite State Machine (FSM), e-fused boot memory, programmable instruction memory, a programmable interrupt controller (PIC), and it executes RoT Unit code on the built-in RISC-V core. The programmable FSM receives inputs from the microcontroller and each accelerator. The FSM outputs set memory access policies enforced by the access control modules attached to each memory bus. The FSM configuration memory is e-fused at deployment time to prevent an attacker from altering it in the field. The BIST module checks the circuitry of the RoT unit for manufacturing defects and eliminates the need for a scan-chain that attackers could use to observe or alter internal processor state. RECORD's boot memory stores code used by the RoT unit for its software features. E-Fuses ensure the boot memory cannot be altered after deployment. The remaining programmable instruction memory code is verified using the hard-coded public key, allowing for updates from trusted parties only. These features allow RECORD to operate using little power while mitigating several powerful attacks based on physical access. The RoT unit requires 201 FPGA slices and 62 BRAM tiles, constituting a total of 13.7% slice overhead and 34.4% BRAM tile overhead.

C. Stratix 10 Security Device Manager

The Stratix 10 FPGA's Security Device Manager (SDM) [24] includes a microcontroller and other "hard" Intellectual Property (IP) blocks to protect an FPGA-based system from tampering. Features include cryptographic accelerators, secure key storage, temperature and voltage tamper sensors, redundant microcontrollers, scan-chain authorization, and a Physical Unclonable Function (PUF) [24]. Combined, the features of the SDM support tamper detection and prevention. The SDM features have been developed to mitigate bitstream/firmware tampering, bitstream cloning, and key theft attacks.

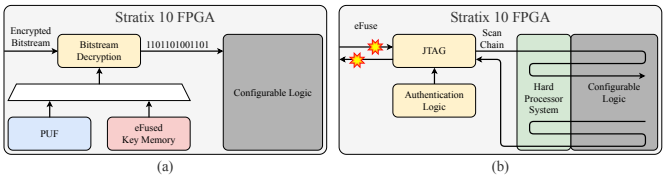

Fig. 3: Intel Stratix 10 FPGA (a) bit-stream encryption, and (b) scan-chain authentication.

Bitstream encryption significantly increases the difficulty of IP reverse engineering. An attacker must successfully decrypt a bitstream before they can begin to understand the logic circuit the bitstream describes. Understanding the logic circuit is an essential step in a bitstream tampering attack, as an attacker must understand the device circuit before malicious modifications can be made. Bitstream encryption based on e-Fused key storage and PUF challenge-response pairs are supported. Device-specific keys, either PUF based or stored in e-Fused secure memory, prevent duplication of FPGA bit-streams, i.e. an encrypted bitstream for one device cannot be used to configure another. To modify the bitsteams of multiple FPGAs with per-device keys, an attacker must obtain the unique key for each FPGA, further raising the difficulty of an attack. Secure debug authorization mitigates scan-chain attacks that might otherwise leak information about internal system state. To use a system's scan chain, a device user must authenticate against a secret key stored on the device. Additional inputs from the configurable logic fabric enable user-defined tamper detection triggers. Custom tamper triggers support anti-tamper enclosures around the device, increasing the difficulty for an attacker to access bitstream configurations or secret key memory. The device configuration interface can also be disabled with e-Fuses in response to a tamper event. The SDM supports several levels of tamper response, including alarm-only, configurable fabric reset, fabric reset and memory/key zeroization, and a device "self-kill", which sets off e-Fuses to prevent future FPGA bit-stream configurations. Figure 3(a) depicts the Stratix 10 bitstream encryption and Figure 3(b) depicts the Stratix 10 scan-chain authentication and e-Fuses.

V. FPGA-based Hardware RoT Architectures

A. Reconfiguration-Based Instruction Decoder Obfuscation

The authors of [25] leverage reconfigurable logic to implement an obfuscated CPU instruction decode unit. Reconfigurable logic allows for the creation of designs that cannot be directly targeted in manufacturing-time attacks. Therefore, the obfuscated instruction decode logic is used as a RoT in a system that aims to mitigate a narrow set of manufacturing-time hardware Trojans, namely code injection hardware Trojans.

Implementing the CPU's Instruction Decode Unit (IDU) in an FPGA means that the CPU's instruction decoding circuitry is not revealed to the manufacturer, significantly increasing the difficulty of a manufacturing-time attack. Instruction-decode hardware Trojans must target the reconfigurable logic, instead of directly targeting the decoding circuit, increasing Trojan complexity. Additionally, the regular patterns of reconfigurable logic could be leveraged to simplify hardware Trojan detection with destructive reverse engineering. Reconfigurable decode logic ensures instruction-decode Trojans cannot directly issue instructions in the processor pipeline. Figure 4 depicts a code injection Trojan and the reconfigurable IDU mitigation.

The designers found that this increases the minimum area of a code injection Trojan by 82.7%. Given that the FPGA IDU increases the area of the processor by 34.7%, a 1% increase in the area of the

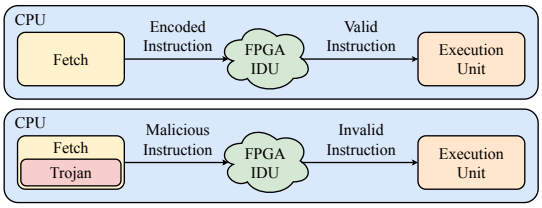

Fig. 4: A Reconfiguration-Based Instruction Decoder.

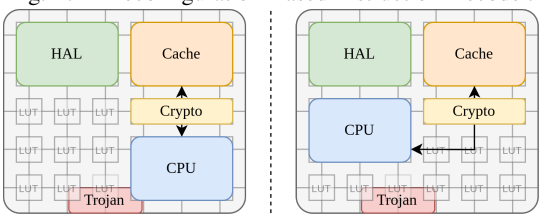

Fig. 5: MORPH defends against hardware Trojans by configuring hardware resources to different locations within the FPGA.

processor requires a 2.38% increase in the area of a code injection Trojan [25]. While this design only defends against one type of hardware Trojan, it can be effective in increasing the difficulty and size of potential hardware Trojans.

B. MORPH

Morph Onion-encryption Replication PRR HAL (MORPH) is an FPGA-based hardware RoT designed to protect against design- and fabrication-time hardware Trojans. It is implemented in an FPGA, in which a Partial Runtime Reconfiguration (PRR) based Hardware Abstraction Layer (HAL) acts as the trusted computing base and performs a number of security operations to defend against application code or data leaks. These security operations defend MORPH from many different kinds of attacks. MORPH's trusted computing base includes its BOOT HAL, HAL, and cryptographic key storage. If these structures are not compromised, MORPH is the only hardware RoT surveyed that can defend against design-time and fabrication-time hardware Trojans [26].

This defense is achieved through its morph operation, as even if an attacker knows the configuration of MORPH they will not know the location of the IP cores used. Movement of IPs is depicted in Figure 5. In this operation, the HAL rearranges hardware resources to different locations within the FPGA. This form of moving target defense varies the attack surface at runtime, increasing the difficulty of attack. The second security operation is onion-encryption, in which memory is encrypted in multiple layers along its path through the system. Memory is encrypted between the processor and the L1 cache, and between the L1 and L2 caches. This encryption helps protect against leakage of code and data - a compromise of one of the layers of encryption will not compromise data or code confidentiality. MORPH also includes three L2 cryptographic modules and a voting protocol between these three. This replication helps defend against design-time hardware Trojans, as any Trojan would need to successfully attack two of the three cryptographic modules. This replication is limited to the outermost cryptographic modules used in the onion-encryption. PRR is used by MORPH to dynamically relocate IP blocks on the FPGA fabric without interrupting the normal flow of execution, i.e. resets or reboots. Periodic reconfiguration of FPGA resources greatly increases the difficulty of a hardware Trojan attack, as the manufacturer cannot know the exact location of specific logic signals targeted by a hardware Trojan. The HAL in MORPH is more active than the HALs generally used in operating systems. It is responsible for bootstrapping the SoC, managing cryptographic keys, and randomly morphing itself and SoC cores. Code running on MORPH executes without knowledge of these operations.

MORPH's HAL processor uses 96 FPGA slices and 16 BRAMs, and the five AES modules required for MORPH use 370 FPGA slices. SoC reconfiguration takes about 10 ms, adding a 10% overhead to a baseline boot process. While its dynamic power usage was not quantified as this is application specific, its static power overhead was less than 1% [26].

VI. Enumerated Attack Descriptions

This section describes each attack mitigated by at least one surveyed hardware RoT architecture. For each attack, the requirements for successful mitigation are defined.

a) ***Design-Time Hardware Trojans****:* A malicious designer or IP vendor may leverage their knowledge of the target to insert a hardware Trojan into the system design. Trojans may be inserted at the Register Transfer Level (RTL) level, or as discrete components on a Printed Circuit Board (PCB). Third-party IP provides a powerful avenue of attack for hardware Trojans [27]. The complexity of modern IC design means that system architects frequently purchase third-party IP modules for use in their systems. To protect their IP, vendors often release modules as black boxes that are difficult for users to inspect. A well-designed hardware Trojan will not be observable during functional testing. To successfully mitigate design-time Trojans, a system must maintain continued and safe operation while under attack. Mitigation may include steps at design-time to establish trust in IP or reaction at runtime to reduce or limit the impact of a Trojan payload.

b) ***Fabrication-Time Hardware Trojans****:* A malicious manufacturer may exploit their access to specialized equipment to alter an otherwise secure design provided by a customer. Hardware Trojans may be inserted during intermediate steps in the manufacturing process. The specialized nature of the required equipment and tools makes it difficult for a customer to validate that a manufactured product matches their design. Examples include A2 [28] and MOLES [29]. The A2 hardware Trojan is inserted into a complete design between placed cells and routed wires. The A2 Trojan connects a sensitive control wire (e.g. processor privilege level) to a software controllable flag (e.g. floating point divide by zero). Fabrication-time hardware Trojans may also exist at different system abstraction levels ranging from the circuit level within an IC to whole components on a PCB. Ensuring a design from an un-trusted manufacturer is Trojan free is often not possible without destructive reverse engineering. Therefore, mitigation is limited to deployment and runtime for operational systems. Successful mitigation should detect or disrupt a Trojan payload, limiting the scope of damage it may cause.

c) ***Firmware Manipulation****:* System firmware is typically stored in non-volatile memory, commonly in external flash chips. Non-volatile firmware storage presents a simple target for attackers interested in gaining low-level control over a system. Attackers may corrupt sensitive portions of the firmware or entirely replace it with a malicious version, granting them control over the software execution environment of a system [30]. Re-writing firmware is possible without specialized tools and requires little expertise. Executing an attack that erases and re-writes firmware can take as little as a few minutes, enabling attacks at system integration time or during operation and maintenance. Successful mitigation of firmware manipulation should detect and prevent modified firmware from executing on a system.

d) ***Malicious Insertion****:* Insertions by an attacker can include hardware and software additions. Hardware Trojans are added to a device at design or fabrication time, while malicious insertion

refers to the addition of components after fabrication time. During system integration, an un-trusted link in the supply chain may insert additional components into a design. Alternatively, valid components may be replaced with counterfeits. Counterfeits may have functional differences or merely a reduced operating range across voltage or temperature. A system integrator's position in the supply chain provides them with the detailed knowledge required for a malicious insertion. Components may be replaced or inserted without additional equipment or tools beyond what is needed to integrate the system. Successful mitigation of malicious insertions should identify inserted components and limit their access to system resources.

e) ***Scan-Chain State Modification****:* An attacker with physical access to a device may access the scan-chain to observe or modify internal systems not accessible through system I/O. Scan-chain based attacks occur after device deployment on an operational system, or on a system removed from operation by the attacker. Successful scan-chain attack mitigation should prevent an attacker from changing or leaking internal system state through the scan-chain.

f) ***Off-Chip RAM Modification****:* Off-chip memory is vulnerable to tampering by an attacker with physical access to a system. Memory may be arbitrarily written by an attacker capable of probing the bus between the processor and memory. The tools required in a probing attack depend on the speed and complexity of the targeted memory. High-frequency memory buses will require more specialized equipment to probe, while low-speed buses may be probed with consumer-grade tools. These attacks may be mitigated by moving RAM on-chip or by encrypting data before storing it off-chip.

g) ***Timing Side-Channels****:* Attackers may observe timing side channels to leak information about a system. For instance, timing side channels in cryptographic operations may leak secret keys. Timing side-channels can be exploited with both physical and remote access to a system. Cache-based timing side channels have proven to be especially powerful tools for attackers to leak information [31] [32]. Side-channel attacks occur during system operation, limiting the window of opportunity for an attacker. However, supply-chain attackers can leverage their access to find or insert side-channels that support an attack during operation [29]. A system should ensure that its execution time is not dependent on sensitive or secret information (such as cryptographic keys) to mitigate timing side-channels. Systems may target a constant runtime or an obfuscated runtime with additional noise to prevent leakage of secret information.

h) ***Clock Fault Injection****:* With physical access to a system, attackers may tamper with input and output signals to the system and specific components. By altering a clock signal to operate beyond the device specification, an attacker can trigger incorrect behavior in digital circuits [33]. This incorrect behavior can cause processors to skip critical instructions or force digital logic to register an incorrect result. Successful mitigation of clock fault injection requires fault detection or prevention. Detection of faults allows for potential correction. Elevating the difficulty of successful fault injection increases the skill and effort required by an attacker, reducing the likelihood of a successful attack.

i) ***Runtime Control-Flow Attacks****:* Runtime control-flow attacks occur when an attacker can alter the execution path of a program in order to gain control of the victim's machine. Examples include return-to-libc [34], Return-Oriented Programming [35], and Control-Flow Bending [36]. These attacks are often enabled by memory corruption attacks such as buffer overflow and format string overflow [37]. Attackers can leverage successful runtime control-flow attacks to execute arbitrary code - [34] showed return-to-libc attacks to be Turing complete. Successful mitigation must either remove the software vulnerability that enabled the attack or prevent an attacker from impacting the system's state.

VII. RoT Architecture Comparison and Discussion

This section provides a comparison of the surveyed architectures and describes how each architecture mitigates an attack. Table I summarizes the comparison and attacks mitigated by each architecture.

a) ***Smart Cards****:* Smart cards support mitigation of firmware manipulation in a connected system by providing signature verification of firmware [13]. Malicious insertions can be mitigated with sealed keys based on the system configuration [13]. As a peripheral device, smart cards cannot mitigate main memory modification for their host device. However, direct memory modification attacks on the smart card itself are mitigated with on-chip memory, eliminating external memory buses for an attacker to probe. Similarly to memory modification, a smart card cannot mitigate timing side-channels or clock fault injections against its host system, but can mitigate direct attacks on the smart card. Cryptographic algorithms can be implemented to execute in constant time, decoupling execution time from secret data [14]. Use of an internal clock for critical logic can mitigate clock fault injection attacks.

b) ***OpenTitan****:* The OpenTitan platform provides a hardware RoT that mitigates firmware manipulation with firmware and bootloader authentication executed by code stored in ROM [17]. When acting as a Trusted Platform Module (TPM), OpenTitan can prevent malicious insertion on additional devices by authenticating the current system configuration before proceeding with a boot process. The OpenTitan Earl Grey microcontroller mitigates off-chip memory modification by eliminating off-chip memory [18]. Clock fault injections are mitigated with on-chip clocks and related control logic.

c) ***Dynamic Trusted Platform Module (DTPM)****:* The DTPM is built directly into a processor pipeline to measure and validate the processor's execution trace at runtime [22]. Runtime validation of the execution trace mitigates attacks between a TPM's Time-of-Check (TOC) and Time-of-Use (TOU) [22]. Instead of verifying firmware and applications at boot-time, the DTPM validates them at runtime, mitigating firmware manipulations that can occur after the TOC, early in the boot process. Control flow attacks are mitigated by verifying hashes of basic block traces, thus detecting execution of unexpected basic blocks [22]. The DTPM demonstrates the advantage of runtime-based defenses over one-time checks before application execution.

d) ***RECORD****:* The RECORD SoC architecture mitigates firmware manipulation during operation or maintenance with hardware-enforced signature verification on all programmable code and data [23]. When the device powers on, e-Fused boot memory executes a signature verification program before allowing execution of other software. A built-in self-test replaces an externally exposed scan chain for sensitive processor state, mitigating scan-chain attacks. Memory modification and probing attacks are made more difficult by including all memory on-chip. The RECORD SoC does not include a cache hierarchy, mitigating all cache-based timing side-channels. The single-cycle microcontroller executes each instruction with the same latency, simplifying the development of constant time software to mitigate other timing side-channels. Including an on-chip oscillator increases the difficulty of clock fault injection attacks, as the attacker cannot supply an arbitrary clock signal to the device.

e) ***Stratix 10 Security Device Manager****:* Little independent literature exists to evaluate the effectiveness of the Stratix 10 security features. However, public documentation is available to provide an understanding of the device's security features [24]. The Stratix 10 FPGA SDM features described in the public documentation focus

TABLE I: Hardware RoT Architectures and Mitigated Attacks

Threat/Attack	Smart Cards	Open-Titan	DTPM	RECORD SoC	Stratix 10 SDM	Reconfig. IDU	MORPH
Design-Time Hardware Trojans							✓
Fabrication-Time Hardware Trojans						✓	✓
Firmware Manipulation	✓	✓	✓	✓	✓		
Malicious Insertion	✓	✓	✓				
Scan-chain State Modification		✓		✓	✓		
Off-chip RAM Modification	✓	Some		✓			✓
Timing Side-Channels	✓			✓			
Clock Fault Injection	✓	Some		✓	✓		
Control-Flow Attacks			✓				

on anti-tamper and bit-stream authentication capabilities. Support for signed and encrypted bit-streams mitigates bit-stream (firmware) manipulation [24]. Built-in scan-chain authentication aims to prevent unauthorized users from accessing internal device state. Hard and user-configurable anti-tamper sensors can be used to detect power or clock fault injections, triggering a tamper response by the device. Tamper responses allow the FPGA to erase sensitive state to protect secret information and even completely disable device configuration, preventing future attacks.

f) ***Reconfiguration-Based Instruction Decoder Obfuscation****:* Re-configurable instruction decode logic increases the complexity of a manufacturing-time Trojan attack, as the final device configuration is shared with the device manufacturer. Trojans targeting the re-configurable instruction decode logic must either test for the current configuration before successfully executing an attack or provide dedicated decode logic outside of the re-configurable fabric [25]. Duplicating decode logic increases Trojan area by at least 82.7% [25], increasing their overhead and making them easier to detect. Reconfiguration is also leveraged to support instruction set randomization, preventing the identification of critical operations for attack or Trojan triggering [25]. Additionally, any Trojans inserted by the manufacturer may be defeated with updates to the configurable logic once they are discovered. However, ROM-based code injection Trojans are only one of many different Trojan attacks. A manufacturer capable of developing a ROM Trojan can likely target other areas of a processor beyond the instruction decode unit. Additional hardware modules could be replaced with re-configurable logic with a higher performance overhead, but such overheads have not been evaluated.

g) ***MORPH****:* The MORPH architecture includes several security features that aim to mitigate design-time and fabrication-time Trojans. A hardware abstraction layer controls IP core placement on an FPGA fabric to obscure the exact location of a module from the manufacturer [26]. Partial runtime reconfiguration supports dynamic placement of IP cores to further ensure a manufacturer cannot know the location at manufacturing-time, limiting manufacturing-time attacks to targeting the fixed FPGA fabric [26]. Encryption at each layer of the cache hierarchy prevents data leakage without compromising the encryption of every layer [26]. This encryption also prevents off-chip RAM modification from being able to control the chip's function. To prevent design-time Trojans from breaking encryption, multiple independently designed, functionally equivalent cryptographic modules are used in a voting scheme [26]. The MORPH architecture relies on a hard boot processor to configure the dynamic IP cores. Visual inspection or destructive reverse engineering may be used to validate the small boot core responsible for managing the FPGA reconfiguration [26]. Validating a small core is considered more tractable than validating an entire processor.

A. Benchmark Attack Selection

Generally, a single solution cannot mitigate every attack possible. As such, the following attacks are selected as a benchmark to support evaluation of future hardware RoT systems:

1) Firmware Manipulation*:* Altering the firmware of a system requires relatively little specialized hardware or software, compared to hardware-based attacks examined in this work. Widely available software tools lower the barriers required to execute such an attack. Often, the ability to write arbitrary firmware provides complete control of the software execution environment of a device. The relatively low effort required and risks to system integrity should make firmware manipulation a priority for evaluation and mitigation.

2) Hardware Trojans*:* Trojan attacks are high-effort, high-reward attacks. Inserting a Trojan requires a deep understanding of the system design and, for manufacturing-time Trojans, control over the component fabrication. However, the difficulty in detecting Trojans makes them one of the most challenging attacks to mitigate. Trojan attacks have been selected to represent a worst-case scenario supply-chain attack where an attacker has detailed knowledge of the design and manufacturing process.

3) Malicious Insertion*:* Instead of altering the construction of a component, malicious insertions replace whole components. Switching components in a system requires less effort than a Trojan insertion, but still provides opportunity to gain a high degree of control over a system. Additionally, malicious insertion may be motivated by cost-cutting pressure instead of a desire to directly attack a system. The reduced effort required, combined with the potential financial cost-cutting incentive, make malicious insertion an important threat to evaluate mitigations against.

Attacks including fault injections, physical probing, and many side-channel attacks require more resources, i.e., time and access, from an attacker than the other selected attacks. Such attacks have been omitted from the initial benchmark to emphasize the lower-cost supply-chain attacks. Analysis that aims to mitigate more capable attackers may extend the selected benchmarks with additional physical access-based attacks. Software-based vulnerabilities, including runtime control-flow attacks, are omitted because they are not unique to OT systems and may be mitigated with more general solutions.

VIII. Conclusion and Future Work

This survey has examined several hardware root-of-trust architectures that mitigate hardware and software attacks across the lifecycle of operational technology systems. The breadth of attacks faced by OT systems means that no single solution can mitigate attacks across design, manufacturing, integration and operation stages. The survey demonstrates that design-time attacks are especially difficult to mitigate, requiring focused solutions in custom architectures. Meanwhile, small, limited systems, such as smart cards, provide the strongest security protections at the expense of performance and flexibility. Results of the survey show that, together, reconfiguration and run-time monitoring can create a hardware root-of-trust capable of mitigating a variety of supply-chain attacks, including hardware Trojans and firmware manipulation. Reconfiguration adds resiliency to design, manufacturing, and integration time attacks, while run-time analysis detects malicious system behavior. Incorporating both techniques into systems offers a promising solution for the development of future operational technology systems resilient to supply-chain attacks.

REFERENCES

[1] D.-Y. Kim, "Cyber security issues imposed on nuclear power plants," *Annals of Nuclear Energy*, vol. 65, pp. 141–143, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306454913005781

[2] B. T. Horvath, "Not all parts are created equal: The impact of counterfeit parts in the air force supply chain," Air War College, Air University Maxwell AFB United States, Tech. Rep., 2017.

[3] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[4] 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, 2014.

[5] Z. Basnight, J. Butts, J. Lopez Jr, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[6] B. Selmke, F. Hauschild, and J. Obermaier, "Peak clock: Fault injection into pll-based systems via clock manipulation," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, 2019, pp. 85–94.

[7] J. R. Reeder and T. Hall, "Cybersecurity's pearl harbor moment," *The Cyber Defense Review*, vol. 6, no. 3, pp. 15–40, 2021.

[8] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, pp. 1–29, 2016.

[9] U.S. Government Accountability Office, "Solarwinds cyberattack demands significant federal and private-sector response (infographic)," Apr 2021. [Online]. Available: https://www.gao.gov/blog/solarwinds-cyberattack-demands-significant-federal-and-private-sector-response-infographic

[10] Cybersecurity Infrastructure Security Agency, "Crashoverride malware," *Cybersecurity and Infrastructure Security Agency (CISA), Tech. Rep. ICS Alert (TA17-163A)*, Jul 2021. [Online]. Available: https://www.cisa.gov/news-events/alerts/2017/06/12/crashoverride-malware

[11] Joint Interpretation Library, "Application of attack potential to hardware devices with security boxes, version 3.0," Tech. Rep., 2020.

[12] B. Karch and M. Rowland, "A review of technologies that can provide a 'root of trust' for operational technologies." 2022.

[13] K. E. Mayes and K. Markantonakis, *Smart cards, tokens, security and applications*. Springer, 2008, vol. 1.

[14] B. Gupta and M. Quamara, "A taxonomy of various attacks on smart card–based applications and countermeasures," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 7, pp. 1–1, 2021.

[15] "A700x family: Secure authentication microcontroller," 2013.

[16] N. A. Chattha, "Nfc—vulnerabilities and defense," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*. IEEE, 2014, pp. 35–38.

[17] "OpenTitan documentation," https://docs.opentitan.org/doc/use_cases/, accessed: 2023-3-28.

[18] LowRISC, "Opentitan earl grey chip datasheet," OpenTitan, Tech. Rep., 2019. [Online]. Available: https://docs.opentitan.org/hw/top_earlgrey/doc/

[19] J. Frazelle, "Securing the boot process: The hardware root of trust," *Queue*, vol. 17, no. 6, pp. 5–21, 2019.

[20] Apple, "Apple t2 security chip security overview," Apple, Tech. Rep., 2018. [Online]. Available: https://www.apple.com/mideast/mac/docs/Apple_T2_Security_Chip_Overview.pdf

[21] "Coremark scores," 2018.

[22] A. K. Kanuparthi, M. Zahran, and R. Karri, "Feasibility study of dynamic trusted platform module," in *2010 IEEE International Conference on Computer Design*. IEEE, 2010, pp. 350–355.

[23] A. Ehret, E. Del Rosario, K. Gettings, and M. A. Kinsy, "A hardware root-of-trust design for low-power soc edge devices," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2020.

[24] *Intel Stratix 10 Configuration User Guide*, Intel, 2022, version 22.1.

[25] B. Liu and B. Wang, "Embedded reconfigurable logic for asic design obfuscation against supply chain attacks," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014.

[26] G. Bloom, B. Narahari, R. Simha, A. Namazi, and R. Levy, "Fpga soc architecture and runtime to prevent hardware trojans from leaking secrets," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 48–51.

[27] "Report of the defense science board task force on high performance microchip supply," 2005.

[28] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 18–37.

[29] L. Lin, W. Burleson, and C. Paar, "Moles: Malicious off-chip leakage enabled by side-channels," in *2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*. IEEE, 2009, pp. 117–122.

[30] A. Cui, M. Costello, and S. Stolfo, "When firmware modifications attack: A case study of embedded exploitation," 2013.

[31] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," *CoRR*, vol. abs/1801.01203, 2018.

[32] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *CoRR*, vol. abs/1801.01207, 2018.

[33] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in *Smart Card Research and Advanced Application: 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Passau, Germany, April 14-16, 2010. Proceedings 9*. Springer, 2010, pp. 182–193.

[34] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 552–561.

[35] R. Roemer, E. Buchanan, H. Shacham, and S. Savage, "Return-oriented programming: Systems, languages, and applications," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 1, mar 2012. [Online]. Available: https://doi.org/10.1145/2133375.2133377

[36] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "Control-flow bending: On the effectiveness of control-flow integrity," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 161–176.

[37] K.-S. Lhee and S. J. Chapin, "Buffer overflow and format string overflow vulnerabilities," *Software: practice and experience*, vol. 33, no. 5, pp. 423–460, 2003.