Position Paper: On Using Trusted Execution Environment to Secure COTS Devices for Accessing Industrial Control Systems

Quanqi Ye¹, Heng Chuan Tan¹, Daisuke Mashima¹, Binbin Chen², and
Zbigniew Kalbarczyk³

¹ Advanced Digital Sciences Center
{Quanqi.ye, hc.tan, daisuke.m}@adsc-create.edu.sg

² Singapore University of Technology and Design
binbin_chen@sutd.edu.sg

³ University of Illinois at Urbana-Champaign
kalbarcz@illinois.edu

Abstract. Industrial Control Systems (ICS) are traditionally designed to operate in an “air-gapped” environment. With the advent of digital technologies, many ICS are adopting IT solutions to improve interoperability and operational efficiency. Thus, the air-gap assumption no longer holds in practice. Most ICS devices today are modernized with networking capabilities to facilitate system maintenance, upgrades, and troubleshooting. Since these devices are connected to the Internet, ICS networks face the same security threats as regular IT systems. In addition, ICS operators can connect commercial off-the-shelf (COTS) equipment to ICS networks to perform operational tasks. Those COTS devices are usually personal computers or even mobile devices, which can be infected with malware and become weapons against ICS. In this position paper, we examine the design challenges of establishing trust between COTS equipment and ICS. We also present some commonly used security solutions and discuss their deployment challenges due to issues caused by legacy systems. Finally, we introduce the Trusted Execution Environment (TEE), a technology commonly available on modern COTS devices, as a trust anchor for establishing secure communications with the ICS infrastructure. We discuss some research gaps related to the use of TEE and propose some recommendations to guide future research.

Keywords: Industrial Control Systems (ICS) · Cyber Physical System (CPS) · Trusted Execution Environment (TEE) · Commercial off-the-shelf (COTS) device · Trusted platforms.

1 Introduction

Industrial Control Systems (ICS) [47] are complex systems that operate critical infrastructures such as railway networks, electricity, water, gas, and heating systems. To achieve high availability and reliability of operations, ICS is equipped

with distributed sensors to collect data and programmable controllers to make decisions for automated processes. Any ICS failures in the process can potentially cause financial loss, damage to the environment, or even endanger people’s lives. These risks have made ICS and the critical infrastructures they manage prime targets for cyber attackers [22,52,11,15].

Several cases of malware-related attacks have been reported on critical infrastructures. Some notable examples are Stuxnet [31], Havex [21], BlackEnergy [51], Industroyer [1], and Triton [27]. Those attacks are created with a very high level of sophistication and are often tailored to target a specific ICS. Nevertheless, they all follow a standard attack pattern, i.e., the adversaries first penetrate the Information Technology (IT) network to plant the malware as an initial foothold [4]. The adversaries then pivot their way into the Operational Technology (OT) network to identify vulnerable OT devices. After gaining sufficient access privileges on the OT network, the adversaries finally launch an attack by issuing maliciously crafted commands (often via malware planted on the compromised devices [50]) to bring down the infrastructure with catastrophic consequences. Since OT networks are usually separated from the IT networks, a key step performed by the adversaries is to find a way to breach the “air-gap”. One way is to connect a malware-infected USB to a production machine, but that requires physical access to the operation plants. Another way is to compromise the Commercial Off-the-Shelf (COTS) devices.

ICS operators can physically connect COTS devices to the OT network (e.g., through USB or serial interfaces) or remotely via the Virtual Private Network. There are many use cases that require COTS devices for ICS operations. For example, an electricity utility company may equip its staff with hand-held devices to configure and troubleshoot smart meters during the installation and maintenance process. Since COTS devices are internet facing, the risks of being compromised are real and thus, cannot be ignored. If an adversary compromises the COTS device, the device can be used as a bridge to transfer malware to other devices in the ICS network.

To secure the interactions between the ICS network and the COTS devices, it is necessary to fortify and upgrade the existing systems of the COTS devices and the ICS network to prevent malware from self-replicating and infecting the system. However, upgrading the existing ICS infrastructure is not cost-effective due to legacy issues. Many ICS devices still use propriety hardware and software. It is also impractical to enforce policies to disable all remote and port access points to COTS devices due to the need for efficient and timely system maintenance, updates/upgrades, and emergency response situations. These limitations call for new methods to address COTS device/software security while minimizing infrastructure upgrades to the existing legacy systems.

In this position paper, we analyze the threats faced by an ICS when it interacts with COTS devices and discuss the various design constraints for securing such interactions. We also discuss the limitations of existing cybersecurity solutions to motivate our study. We then propose to use the Trusted Execution Environment (TEE) technology, which is already widely available on COTS de-

vices as a building block to secure the interactions. Towards this goal, we discuss challenges regarding the development, execution, and deployment of trusted applications on TEE, all of which play important roles in mitigating the security threats posed by COTS devices when brought into the ICS context.

2 Problem Statement

In this section, we give a brief overview of ICS using smart grid as an example of critical infrastructure. Then, we discuss the threat model and practical challenges of securing ICS.

2.1 ICS Overview

Modern ICS is built to monitor and control industrial processes. The typical architecture of an ICS (in the context of a power grid) is shown in Figure 1. The control center and the substations are connected via a dedicated network. The task of the control center is to monitor, coordinate and communicate system issues to mitigate downtime, whereas the substation is responsible for controlling the actual physical processes. Thus, the substation contains many supervisory and controller units such as remote terminal units (RTUs), programmable logic controllers (PLCs), intelligent electronic devices (IEDs), and human machine interfaces (HMIs). These devices communicate with each other through network devices such as routers, switches. The data collected is then forwarded by the router to the Supervisory Control and Data Acquisition (SCADA) system [8] in the control center, as shown in Figure 1.

COTS devices (e.g., commodity laptops, hand-held devices) are devices that interact directly or indirectly via remote access (e.g., VPN) with the substation equipment. They run different software and use different protocols to communicate. Operators use them to monitor, test, debug, and configure ICS devices, including carrying out firmware updates. Engineers can also connect COTS devices to set up some networks for troubleshooting and maintenance purposes. These COTS devices are connected as needed rather than permanently. Other COTS devices include storage devices, sensors, and IoT devices such as cameras and sensors. In this paper, our focus is on the security of COTS devices because of their ability to interact with various controller units to control the production process, an area that has not received much research attention compared to the control center.

2.2 Threat Model and Security Goal

Threat Model. Our threat model focuses on three main entities, the ICS operators, the COTS devices, and ICS infrastructure. The ICS operators are responsible for maintenance tasks such as updating/upgrading ICS devices, issuing control or configuration commands, including troubleshooting ICS production problems. The attacker’s objective is to gain control of the ICS network to

Fig. 1. A typical ICS and its interactions with COTS devices.

cause damages to the critical infrastructure. Specifically, we make the following security assumptions.

ICS Operators: We assume the ICS operators are trusted. They follow the best security practices to perform all tasks in the ICS network and take proper actions, e.g., stopping a procedure when anomalies are detected on their COTS devices.

COTS Devices: We assume that the operators and third-party contractors are allowed to use COTS devices, e.g., laptop computers, tablets, or mobile devices, to perform their tasks in the ICS environment. The same devices may also be used in the office environment to perform daily work outside of the ICS environment. We assume that the contractors can install software on their devices to facilitate maintenance works. As a result, these COTS devices are exposed to the Internet and thus to the malware in the wild. Thus, we assume that COTS devices are not trusted. Specifically, the operating system and applications running on the COTS devices can be compromised and exploited by attackers to gain access to the ICS network. COTS devices containing privileged software may be stolen or misplaced. If these stolen devices fall into the wrong hands, they may be misused for unlawful activities such as gaining unauthorized access to sensitive information, including physical theft of data to sell on the dark web.

ICS Infrastructure: We assume that the ICS infrastructure, including the control center, is secure and well-protected. Thus, the attackers have no physical access to the operational plants. The only way an attacker can attack the ICS is to

remotely compromise the COTS devices used by the ICS operators during system maintenance.

Security Goal. Our security goal is to prevent COTS devices from causing undesired behavior to ICS (e.g., issuing a maliciously crafted command or injecting fake sensor data), thereby damaging the critical infrastructure. This can be achieved by detecting/removing the malware from the COTS devices (i.e., checking the integrity of the COTS devices’ hardware and software) or preventing malware from interfering with the interactions between the COTS devices and the ICS.

2.3 Practical Challenges

This section summarizes the unique challenges and constraints of securing ICS under the threat model discussed above.

Legacy Devices: The vast majority of ICS devices are legacy devices that are built to operate for many years. They are designed to achieve high availability and reliability but not security. Most devices run on proprietary software that is no longer patchable. Furthermore, their operating systems are often resource-constrained [47]. Thus, there may not be enough computing resources to deploy state-of-the-art malware protection or attack detection solutions. Given that industrial processes hardly change, the ICS owner may be reluctant to upgrade the existing ICS devices and the network. It is also infeasible to stop and replace devices running in the ICS network to perform the upgrade due to the high cost of service outages.

Insecure communication protocols. Due to the resource constraint of ICS devices and the need to ensure high availability of service, many industrial protocols are not designed with security in mind. Protocols such as Modbus [34], DNP3 [10], IEC 60870 [56], including the recent IEC 61850 [24], lack encryption and authentication properties. Messages are usually transported in clear without proper authentication. These messages are therefore open to manipulation by attackers who manage to gain access into the network. Although security standards, such as IEC 62351 [25], have been proposed for securing IEC 61850 messages, it is rarely used due to its heavy reliance on PKI, which currently cannot be supported by legacy ICS devices.

Usability and Flexibility. The key requirements of ICS are high usability and flexibility. In the event of equipment stoppages due to faults or failures, operators must be able to rectify problems promptly and efficiently. Thus, operators are allowed to connect COTS devices to the ICS network. To minimize security risks, one could enforce: (1) the operators to disconnect their COTS devices from the Internet, or (2) implement rigid operating procedures that operators must follow when using the COTS devices for maintenance tasks. Unfortunately,

Table 1. Comparisons of Existing Solutions.

Solutions	Upgrade Needed	Low Cost	Scalable	Hard to bypass
Endpoint security	✓	✓	✓	✗
Encryption	✗	✓	✗	✗
BITW	✗	✗	✗	✓
IDS	✗	✗	✓	✗

these measures are hard to enforce in practice. In addition, these measures would decrease the usability and flexibility of ICS, defeating the intention of allowing the operators to use COTS devices to carry out timely maintenance tasks.

3 Review of Existing Security Solutions

Many security solutions have been proposed and developed in the literature [16,23,48] to secure ICS. This section discusses some of them and explains why they are not suitable for addressing our security goals. We summarize our findings and compare the different solutions in Table 1.

3.1 Standard Endpoint Security

Endpoint security refers to the use of authentication and access control techniques to protect ICS devices. These techniques prevent ICS devices from interacting with potentially insecure COTS devices and the software within them. By properly authenticating and authorizing users or software on the COTS device, only legitimate user/software can access and operate the ICS. However, many authentication and access control schemes have proven to be ineffective in preventing malware. For example, the secrets used by authentication can be stolen/guessed by an adversary if the COTS device is compromised [53,5]. Adversaries can also use social engineering techniques to trick the operators into disclosing private credentials to bypass access control measures [54]. In addition, the software might have exploitable bugs that allow adversaries to gain access to system hardware.

Another approach is to install an anti-virus software on the COTS device to search for malicious code patterns. However, the effectiveness of an anti-virus software depends on known signatures of malware. Therefore, anti-virus software must be updated frequently to remain relevant. Even then, anti-virus software cannot effectively defend against zero-day malware developed by an advanced attacker.

3.2 Cryptographic Primitives

Cryptography plays a key role in ensuring the confidentiality, integrity, and authenticity of the information in any communication process. These properties are

achieved using standard symmetric/asymmetric primitives [25], a combination of hashing operations with covert channels [7], or knowledge-based proofs [49]. Other works explore identity-based cryptography [12] as an alternative to simplify PKI-based key management system [43].

Regarding the use of cryptography, one major drawback is that additional security components (either in the form of software or bump-in-the-wire hardware) must be installed on or near the recipient ICS devices. This enhancement requires a significant upgrade to the existing infrastructure because most legacy ICS devices do not have sufficient computing resources to support secure cryptographic operations. Even when it is possible, protecting the communication will not help when the COTS device itself becomes malicious (e.g., by means of malware) or misused by remote attackers, as demonstrated in the Ukraine power plant attacks [11]. Cryptography, in particular encryption, is also not recommended for securing industrial protocols because most ICS operations are time-critical and delay intolerant. Encryption further reduces the data visibility required by intrusion detection systems to evaluate the packet contents for anomalies [14].

3.3 Bump-in-the-Wire Solutions

Bump in the wire (BITW) solutions aim to provide security “in front of” ICS devices. They are used when there is a lack of security enforcement in the communication protocol or when it is difficult to upgrade the end devices due to compatibility and stability issues.

Generally, BITW solution is deployed as a “proxy” (or “relay”) between two communication endpoints or devices. These BITW devices are configured to provide advanced security features (cryptographic protections [13] or enhanced validation of messages [33]) in a transparent, add-on manner. They do not require any upgrade to the existing ICS devices. BITW solutions have also been deployed to ensure the integrity and authenticity of firmware updates on legacy PLCs that do not have any built-in verification mechanisms for firmware images. One use case is [6], which uses a combination of digital signatures and machine-learning to secure the firmware update process. However, one major issue with deploying BITW solutions is cost because each ICS device must be retrofitted with a BITW device. The cost comes not only from the deployment of hardware and software but also from maintaining these devices.

3.4 Intrusion Detection System

An intrusion detection system (IDS) is a system used to detect unauthorized intrusions into a system or network [32,40,48]. It is an important means of defense against cyber attacks in ICS. They come in different flavors depending on security needs. For example, host-based IDS is deployed on a host computer or device to monitor system logs and configuration files for any suspicious changes in the file systems. Network-based IDS focuses on analyzing the network traffic

for malicious behaviors. Therefore, network-based IDS is usually deployed at strategic locations in the network to capture the most traffic.

IDS can also be categorized based on detection methods, namely (i) signature-based and (ii) anomaly-based. The signature-based IDS checks the current system status and activities against a database of known attack patterns or signatures. In contrast, an anomaly-based IDS compares the current system status with an established baseline of normal activities. For establishing the baseline, various approaches such as machine-learning models [44,39] and statistical analysis [55] have been developed. In comparing signature-based and anomaly-based, the latter is more common because it can detect zero-day attacks. However, the likelihood of false positives also increases.

While IDS solutions offer certain security guarantees, they are not without limitations. First, upgrading the ICS infrastructure to support host-based IDS on every ICS device is often impractical in terms of cost and performance overhead. Second, network-based IDSes cannot pinpoint which host or device is under attack because they only monitor the network traffic. More importantly, network-based IDSes cannot detect malicious messages that conform to the network protocol. Signature-based IDS, on the other hand, cannot detect zero-day attacks. Anomaly-based IDS is prone to data corruption due to limited samples to build the baseline models.

4 Trusted Execution Environment Overview

As described in Section 3, existing security solutions are only effective if substantial upgrades are allowed in the ICS environment. These upgrades typically require a complete revamp of the ICS infrastructure with careful planning and detailed policy-making.

With advances in trusted computing technology, most modern COTS devices are equipped with Trusted Execution Environment (TEE) technology by default. This advancement provides a promising direction for securing the interaction between COTS devices and ICS. In this section, we focus on the ARM TrustZone since most COTS devices we considered are handheld devices such as mobile phones and tablets. We explain the TEE concepts and discuss some of its security features.

4.1 TEE as Trust Anchor

As a trusted computing technology, TEE is widely supported by major chip manufacturers such as Intel on the desktop and ARM on the mobile platform. It is a technology that provides an isolated and secure execution environment for executing security-sensitive applications at the hardware level. TEE guarantees the confidentiality, integrity, and authenticity of program execution [18]. In general terms, the CPU creates two environments corresponding to two worlds: (i) secure world (TEE) and (ii) non-secure world (Rich Execution Environment, a.k.a., REE). Standard operating systems (OS), such as Windows, Linux, or

Fig. 2. A Typical TEE Structure.

Android, are implemented in the non-secure world. They are granted lower privileges compared to other secure OSes and trusted applications (TAs) in the secure world.

Examples of TEE technology are Intel Software Guard Extensions (SGX) [26] and ARM TrustZone [3]. Although they are implemented by different vendors, they provide similar security properties for the programs executed in the TEE. In more detail, TEE acts as a trusted anchor for secure communication with the server and as an authentication framework for personnel, software, and OSes running on the COTS device. Even if the rich OS running in the REE is compromised by a Trojan or malware, the secure OS running in the TEE is still safe due to the isolation between the REE and the TEE. This property provides a promising avenue for securing applications running on COTS devices that interact with ICS. A typical TEE structure (ARM TrustZone implementation) is shown in Figure 2. Specifically, TEE has the following features to mitigate threats from the COTS devices.

Secure Boot. Secure boot provides the integrity and authenticity properties of the program running in it. The boot process starts with a platform-dependent Root of Trust (RoT) that stores self-certified information. It is usually stored in the write-protected ROM (Read-only Memory), which is fused at the time of manufacturing and cannot be modified afterwards. The boot process is divided into multiple stages. Each stage loads, verifies and authenticates the image/firmware for the next stage. In other words, every program that is loaded and executed is verified and authenticated by the previous stage. These stages are chained together to form a Chain of Trust (CoT).

Secure OS and Trusted Application (TA). Many secure OSes have been developed in TEE to ease the hassle of interacting with the hardware interface. Most of them are proprietary, with a few exceptions that are open-sourced [37,38]. Similar to the Portable Operating System Interface (POSIX) standard for normal OS, GlobalPlatform has also published several standards [17,19] on secure OS to support TA development. To keep the size of the Trust Computing Base (TCB) to a minimum, the secure OSes are designed to occupy a small footprint. For this reason, a secure OS cannot provide as many functionalities as a standard OS. Similarly, TAs running in secure OSes need to keep their size as small as possible so that software bugs are unlikely to exist. They provide security support for the rich OS to complete different tasks that require high-level security. When the system boots, the rich OS and the secure OS will run in parallel in the REE and TEE worlds, respectively.

Worlds Switch. In ARM TrustZone, the secure monitor is responsible for switching between secure and non-secure worlds. This switching is achieved by executing a privileged ARM instruction called Secure Monitor Call (SMC).⁴ An exception is raised when SMC is executed. The exception can only be handled by the Secure Monitor running in the secure world. It can be configured to mask normal world exceptions during the handling of SMC so that the secure world is not vulnerable to attacks (e.g., denial of service — DoS) from the non-secure world. The secure monitor is a separated component that is not part of the secure OS. Its authenticity and integrity are verified and authenticated by the secure boot process. The world status, i.e., secure world and non-secure world, are distinguished by the Not-secure bit (NS-bit) of the secure configuration register (SCR) [2], which is only accessible in the secure world.

Access Control of Memory and Peripherals. The NS-bit is propagated to the memory, caches, and peripheral devices and can be configured to be accessible by the secure world or the non-secure world, or both. Different platforms have different implementations of secure OS behaviors when certain access violations are detected. For instance, when the non-secure world attempts to access secure memory, there is an external abort to the REE OS to stop the access and trigger a security violation. The same concept applies to peripheral devices.

Secure Storage. The storage functionality of the TEE should protect the stored data against confidentiality, authenticity, consistency, and atomicity. In addition to these properties, secure storage should also be bound to the host device. This means that the data stored in the secure storage can only be accessed and modified by the same device running the same TEE with the same software/firmware via the same authorized TA. If the attacker tries to disconnect the storage from one device and attach it to another, any attempt to read the data will fail.

In summary, TEE is used as a trust anchor in the COTS device. TEE offers reliable isolation to protect software components used for interacting with ICS

⁴ Intel SGX also has its counterpart instruction for entering into enclave [46].

Fig. 3. An example of using static analysis and binary rewriting to automatically partition Android applications for trusted execution. The non-security sensitive part will be executed in the normal world, while the security-sensitive commands and confidential data will be in the TEE.

and protect against the propagation of malware from the REE. More advanced protection technologies can be built on this basis to achieve the security goals outlined in Section 2.2. Some examples are secure deployment of applications, remove unwanted application/malware, verify application integrity, monitor applications' behaviors, and impose restrictions on their behaviors.

5 Research Problems Related to Using TEE

While TEE provides an array of useful features for securing the interactions between a COTS device and an ICS, there are still several gaps to overcome to achieve a practical end-to-end solution. In this section, we identify some of these gaps and propose some solutions to provide research directions for future work.

5.1 Auto-porting of Applications to TEE

An important research direction is to automatically port legacy applications to TEE, as shown in Figure 3.

To provide security and backward compatibility, it is desired to develop solutions that can automatically split an existing app so that some of the security-sensitive functionalities can be executed in TEE while the rest in the REE world.

However, not much research has been conducted in this direction, except for some early-stage efforts. For example, authors in [42] proposed a semi-automatic approach to partition an Android app to run on REE and TEE, respectively. Starting with confidential data (e.g., security key and password) that the engineer can annotate before the analysis begins, the proposed framework performs taint analysis [36] to track information flow paths and related components.

Then it extracts candidate functionalities that should be implemented in the TEE and wrapped them with the TEE APIs. The original app will also need to be re-factored to work with the extracted functionalities in the TEE. Only then can the partitioned app be deployed in the respective REE and TEE for execution.

This approach greatly alleviates the efforts needed to port legacy apps to TEE for execution. However, it has several limitations. Firstly, due to the intrinsic characteristic of the approach, the candidate functionality wrapped by the TEE APIs has variable code size, which in turn makes the size of the TA in the TEE unpredictable. As a result, it may bloat the size of TCB in TEE and thus increases the TEE's vulnerability. Secondly, to address confidential data in REE and from REE to TEE, the framework uses a special data structure. Due to this, the approach may not be able to generate valid candidate functionality code segments for TEE.

5.2 Deployment of Trusted Applications

While TEE provides a secure environment for hosting TAs developed by ICS companies, it is still challenging to deploy TAs securely on the COTS device due to the proprietary nature of the secure OS. The device vendor typically requires the developer to work with them (e.g., paying a premium to use dedicated development tools) in exchange for developing and deploying the TAs on the secure OS. Thus, this increases the cost and other management overheads for the ICS operating company.

Moreover, the secure OS is fragmented, which further exacerbates the integration efforts. To ease the deployment and cost burdens, a practical framework should be developed for ICS operating companies to deploy and control TAs on COTS devices. The framework should guarantee the following three features to enhance security and flexibility.

1. The framework should allow full control by the ICS operating company. In other words, even if the TA developer obtains approval from the secure OS vendor, the final decision should be made by the ICS operating company to determine which TA can be installed and executed on the operators' COTS devices. ICS operating company can enforce this requirement by installing a certificate on the operators' COTS devices and signing the approved TA. Then, it is the role of the TEE to verify and ensure that a TA running on the operators' COTS devices is properly authenticated against the ICS operating company's certificate. All the computation processes should be performed inside the TEE, such that even if the malware compromises the REE, the verification process will not be bypassed.

Fig. 4. Utilize TEE as a platform for attestation

2. The framework should enable the ICS operating company to update approved TAs or revoke outdated ones. This is achieved by storing a list of approved TAs on a trusted server in the company network. The pre-installed TA on the COTS then fetches the list and sends it to the TEE for verification.
3. To guarantee the authenticity of the approved TA list, the communication channel between the TEE and the company network must be secure and trusted. This can be achieved by using state-of-the-art communication techniques such as TLS to communicate with the trusted server in the company network. The TEE should be able to directly control the network adapter, the drivers and the memory used for storing and processing communication data to ensure that REE cannot interfere during the communication or verification process.

5.3 Secure Execution using TEE

Although TEE is assumed trusted in our threat model, all the peripherals that communicate with the TEE may be compromised. Some examples are the network adapters or serial port drivers, the logic that manages data storage and data transmission, and the human machine interface drivers. To this end, we propose attestation technique, as shown in Figure 4 to ensure their authenticity and integrity.

There are plenty of works in this direction [45,30,20,9], which can be generalized into three categories: hardware-based attestation, software-based attestation, and hybrid attestation. Hardware-based attestation uses dedicated hardware to attest the hardware and software running on the targeted platform.

Software-based attestation, on the other hand, uses software solutions for the attestation, e.g., by fingerprinting the current software states in the memory

and checking the checksum periodically [45]. Hybrid attestation achieves the same goal but combines the merits of hardware- and software-based attestation techniques [28].

Despite the existing research efforts, there are still some practical limitations when applying attestation in an ICS environment. Firstly, attestation relies on the existence of some database for authenticated images, which is difficult to scale when the number of components that require attestation increases. Secondly, many existing attestation solutions only attest the authenticity and integrity of the software at the moment of attestation. They do not provide guarantees for their run-time behaviors. For example, attestation cannot handle run-time exploits such as return-oriented programming attacks [41,29]. This can result in the time-of-check to time-of-use (TOCTOU) issues. Last but not least, since multiple components need to be attested on the COTS device, the execution of those attestation actions needs to be carefully orchestrated to avoid potential inconsistencies that could be exploited by attackers to bypass the checking.

Another related challenge is to prevent a malware or attacker who has footprints in the normal world from invoking security-sensitive functionality on a TA (e.g., encryption, authentication). Since TA can be uniquely identified by an ID in SMC, it is not a difficult task for attackers/malware for running the TA of interest. Besides, when an attacker physically owns the device, he could simply utilize the normal world app, just like a legitimate user, to invoke the TA. In order to counter such threats, it is desired that a TA implements an additional mechanism to detect anomalous access/usage patterns by means of machine learning technologies etc. On the other hand, as discussed in [35], TEE poses limitations in terms of memory and storage size, which makes it challenging to use computational intensive machine-learning/AI based approaches.

6 Conclusion

In this position paper, we analyzed the threat model faced by ICS when operators are allowed to connect their COTS devices to carry out maintenance tasks. We discussed different types of security solutions and compared their ineffectiveness and limitations. Emerging TEE technologies are considered promising to complement them, but a few issues must be addressed for practical integration. In this regard, we enumerated and discussed a few promising research directions. We believe this research will provide new insights into the use of TEE for overcoming security gaps in existing legacy ICS. This study will open up an opportunity to design cost-effective solutions to provide multiple lines of defense in an ICS context.

Acknowledgment

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

1. Anton, C., Lipovsky, R.: Industroyer: Biggest threat to industrial control systems since Stuxnet. <https://www.welivesecurity.com/2017/06/12/industroyer-biggest-threat-industrial-control-systems-since-stuxnet/> (Jun 2017)
2. Secure Configuration Register. <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0438d/BABGIHHJ.html>
3. Arm TrustZone Technology. <https://developer.arm.com/technologies/trustzone>
4. Assante, M.J., Lee, R.M.: The industrial control system cyber kill chain. SANS Institute InfoSec Reading Room **1** (2015)
5. Bai, G., Sun, J., Wu, J., Ye, Q., Li, L., Dong, J.S., Guo, S.: All your sessions are belong to us: Investigating authenticator leakage through backup channels on android. In: 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 60–69. IEEE (2015)
6. Benkraouda, H., Chakkantakath, M.A., Keliris, A., Maniatakos, M.: Snifu: Secure network interception for firmware updates in legacy plcs. In: 2020 IEEE 38th VLSI Test Symposium (VTS). pp. 1–6. IEEE (2020)
7. Bernieri, G., Cecconello, S., Conti, M., Lain, G.: TAMBUS: A novel authentication method through covert channels for securing industrial networks. *Computer Networks* **183**, 107583 (2020)
8. Boyer, S.A.: SCADA: supervisory control and data acquisition. International Society of Automation (2009)
9. Chen, B., Dong, X., Bai, G., Jauhar, S., Cheng, Y.: Secure and efficient software-based attestation for industrial control devices with arm processors. In: Proceedings of the 33rd Annual Computer Security Applications Conference. pp. 425–436 (2017)
10. Clarke, G., Reynders, D., Wright, E.: Practical modern SCADA protocols: DNP3, 60870.5 and related systems. Newnes (2004)
11. Donghui Park, Julia Summers, M.W.: Cyberattack on Critical Infrastructure: Russia and the Ukrainian Power Grid Attacks. <https://jsis.washington.edu/news/cyberattack-critical-infrastructure-russia-ukrainian-power-grid-attacks/> (Oct 2017)
12. Drias, Z., Serhrouchni, A., Vogel, O.: Identity-based cryptography (IBC) based key management system (KMS) for industrial control systems (ICS). In: 2017 1st Cyber Security in Networking Conference (CSNet). pp. 1–10. IEEE (2017)
13. Esiner, E., Mashima, D., Chen, B., Kalbarczyk, Z., Nicol, D.: F-Pro: A fast and flexible provenance-aware message authentication scheme for smart grid. In: 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). pp. 1–7. IEEE (2019)
14. Fauri, D., de Wijs, B., den Hartog, J., Costante, E., Zambon, E., Etalle, S.: Encryption in ICS networks: A blessing or a curse? In: 2017 IEEE International Conference on Smart Grid Communications (SmartGridComm). pp. 289–294. IEEE (2017)
15. Giles, M.: Triton is the world’s most murderous malware, and it’s spreading. <https://www.technologyreview.com/s/613054/cybersecurity-critical-infrastructure-triton-malware/> (Mar 2019)
16. Giraldo, J., Sarkar, E., Cardenas, A.A., Maniatakos, M., Kantarcioglu, M.: Security and privacy in cyber-physical systems: A survey of surveys. *IEEE Design & Test* **34**(4), 7–17 (2017)
17. GlobalPlatform: TEE Client API Specification. <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2C3A19C13215CE2ACB0A0069B09B5039?doi=10.1.1.183.2049&rep=rep1&type=pdf> (April 2010)

18. GlobalPlatform: Introduction to Trusted Execution Environments. <https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf> (May 2018)
19. GlobalPlatform: TEE Internal Core API Specification Version 1.1.2.50 (Target v1.2). https://globalplatform.org/wp-content/uploads/2018/06/GPD_TEE_Internal_Core_API.Specification.v1.1.2.50_PublicReview.pdf (June 2018)
20. Gu, L., Ding, X., Deng, R.H., Xie, B., Mei, H.: Remote attestation on program execution. In: Proceedings of the 3rd ACM workshop on Scalable trusted computing. pp. 11–20 (2008)
21. Havex. <https://www.cyber.nj.gov/threat-center/threat-profiles/industrial-control-system-variants/#h> (Aug 2017)
22. Holloway, M.: Stuxnet Worm Attack on Iranian Nuclear Facilities. <http://large.stanford.edu/courses/2015/ph241/holloway1/> (Jul 2015)
23. Hu, Y., Yang, A., Li, H., Sun, Y., Sun, L.: A survey of intrusion detection on industrial control systems. *International Journal of Distributed Sensor Networks* **14**(8), 1550147718794615 (2018)
24. IEC 61850 - Communication networks and systems for power utility automation - ALL PARTS (Feb 2020), <https://webstore.iec.ch/publication/6028>
25. IEC 62351:2020 - Power systems management and associated information exchange - Data and communications security - ALL PARTS (May 2020), <https://webstore.iec.ch/publication/6912>
26. Intel® Software Guard Extensions (Intel® SGX). <https://software.intel.com/sgx>
27. Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., Glycer, C.: Threat Research Attackers Deploy New ICS Attack Framework “TRITON” and Cause Operational Disruption to Critical Infrastructure. <https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html>
28. Koeberl, P., Schulz, S., Sadeghi, A.R., Varadharajan, V.: TrustLite: A security architecture for tiny embedded devices. In: Proceedings of the Ninth European Conference on Computer Systems. pp. 1–14 (2014)
29. Kornau, T., et al.: Return oriented programming for the ARM architecture. Ph.D. thesis, Master’s thesis, Ruhr-Universität Bochum (2010)
30. Kylänpää, M., Rantala, A.: Remote attestation for embedded systems. In: Security of Industrial Control Systems and Cyber Physical Systems, pp. 79–92. Springer (2015)
31. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* **9**(3), 49–51 (2011)
32. Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y.: Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications* **36**(1), 16–24 (2013)
33. Mashima, D., Gunathilaka, P., Chen, B.: Artificial command delaying for secure substation remote control: Design and implementation. *IEEE Transactions on Smart Grid* **10**(1), 471–482 (2017)
34. Modicon, I.: Modicon modbus protocol reference guide. North Andover, Massachusetts pp. 28–29 (1996)
35. Nakai, T., Suzuki, D., Fujino, T.: Towards Trained Model Confidentiality and Integrity Using Trusted Execution Environments. In: 2nd workshop on Artificial Intelligence in Hardware Security (2021)
36. Newsome, J., Song, D.X.: Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In: NDSS. vol. 5, pp. 3–4. Citeseer (2005)

37. Open Portable Trusted Execution Environment. <https://www.op-tee.org/>
38. Open-TEE Project. <https://open-tee.github.io/>
39. Ponomarev, S., Atkison, T.: Industrial control system network intrusion detection by telemetry analysis. *IEEE Transactions on Dependable and Secure Computing* **13**(2), 252–260 (2015)
40. Quincozes, S.E., Albuquerque, C., Passos, D., Mossé, D.: A survey on intrusion detection and prevention systems in digital substations. *Computer Networks* **184**, 107679 (2021)
41. Roemer, R., Buchanan, E., Shacham, H., Savage, S.: Return-oriented programming: Systems, languages, and applications. *ACM Transactions on Information and System Security (TISSEC)* **15**(1), 1–34 (2012)
42. Rubinov, K., Rosculet, L., Mitra, T., Roychoudhury, A.: Automated partitioning of android applications for trusted execution environments. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). pp. 923–934. IEEE (2016)
43. Saxena, A., Pal, O., Saquib, Z., Patel, D.: Customized PKI for SCADA system. *Int. J. of Advanced Networking and Applications* **1**(05), 282–289 (2010)
44. Schneider, P., Böttinger, K.: High-performance unsupervised anomaly detection for cyber-physical system networks. In: Proceedings of the 2018 workshop on cyber-physical systems security and privacy. pp. 1–12 (2018)
45. Seshadri, A., Luk, M., Shi, E., Perrig, A., van Doorn, L., Khosla, P.: Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)* pp. 1–16 (01 2005). <https://doi.org/10.1145/1095809.1095812>
46. x86 and amd64 instruction reference. <https://www.felixcloutier.com/x86/eenter> (May 2019)
47. Stouffer, K., Falco, J., Scarfone, K.: Guide to industrial control systems (ICS) security. NIST special publication **800**(82), 16–16 (2011)
48. Tan, H.C., Cheh, C., Chen, B., Mashima, D.: Tabulating Cybersecurity Solutions for Substations: Towards Pragmatic Design and Planning. In: 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia). pp. 1018–1023. IEEE (2019)
49. Tan, H.C., Lim, K., Keoh, S.L., Tang, Z., Leong, D., Sum, C.S.: Chameleon: A blind double trapdoor hash function for securing AMI data aggregation. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). pp. 225–230. IEEE (2018)
50. US-CERT: CrashOverride Malware. [Online]. Available: <https://www.us-cert.gov/ncas/alerts/TA17-163A> (2017)
51. Vann, P.: Casting a Light on BlackEnergy. <https://threatconnect.com/blog/casting-a-light-on-blackenergy/> (Sep 2017)
52. Walker, D.: 'Havex' malware strikes industrial sector via watering hole attacks. <https://www.scmagazine.com/home/security-news/havex-malware-strikes-industrial-sector-via-watering-hole-attacks/> (Jun 2014)
53. Ye, Q., Bai, G., Wang, K., Dong, J.S.: Formal Analysis of a Single Sign-on Protocol Implementation for Android. In: 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 90–99. IEEE (2015)
54. Ye, Q., Zhang, Y., Bai, G., Dong, N., Liang, Z., Dong, J.S., Wang, H.: LightSense: A Novel Side Channel for Zero-permission Mobile User Tracking. In: International Conference on Information Security. pp. 299–318. Springer (2019)
55. Yoo, H., Shon, T.: Novel approach for detecting network anomalies for substation automation based on iec 61850. *Multimedia Tools and Applications* **74**(1), 303–318 (2015)

56. Zeyou, L., Yunqing, C.: EXPERIENCE OF USING IEC 60870-5-103 AND IEC 60870-5-104 TRANSMISSION PROTOCOLS [J]. Automation of Electric Power Systems **4** (2003)